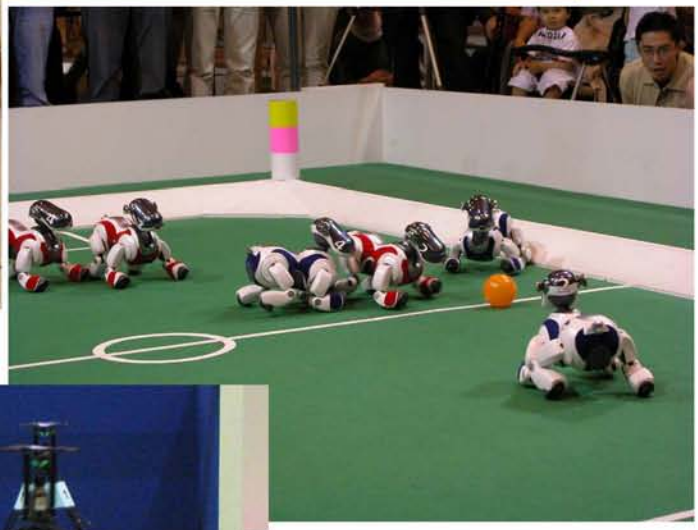
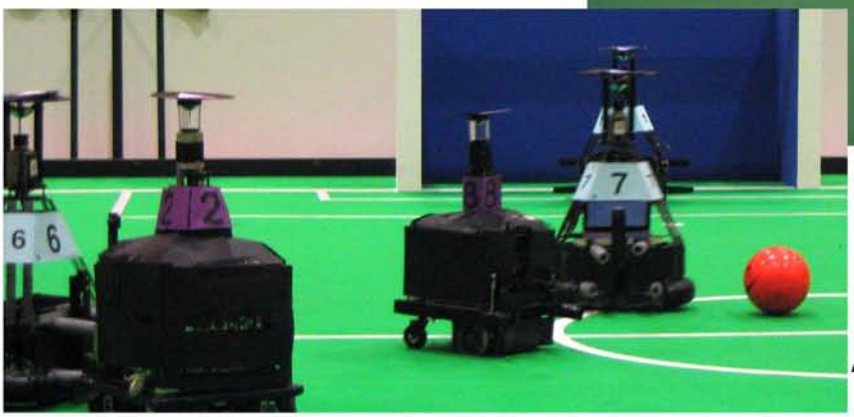
人工知能学会
第21回SIG-Challenge研究会





2005

RoboCup

2005年5月24日
東京全日空ホテル
(ロボカップ2005)

目次

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究資料
JSAI Technical Report
SIG-Challenge-0521-1 (5/24)

モジュール型学習機構における 例示の理解に基づいた自律的なタスク分解

Self Task Decomposition for Modular Learning System through Interpretation of Instruction by Coach

高橋 泰岳 [1,2], **西 智樹** [1] , **浅田 稔** [1,2]
Yasutake Takahashi ,Tomoki Nishi and Minoru Asada
[1] 阪大 ,[2] 阪大 FRC
[1]**Osaka University ,**[2]**HANDAI Frontier Research Center**

Abstract

One of the most formidable issues of RL application to real robot tasks is how to find a suitable state space, and this has been much more serious since recent robots tends to have more sensors and the environment including other robots becomes more complicated. In order to cope with the issue, this paper presents a method of self task decomposition for modular learning system based on self-interpretation of instructions given by a coach. Unlike the conventional approaches, the system decomposes a long-term task into short-term subtasks so that one learning module with limited computational resources can acquire a purposive behavior for one of these subtasks. Since instructions are given from a viewpoint of coach who has no idea how the system learns, they are interpreted by the learner to find the candidates for subgoals. Finally, top layer of the hierarchical RL system coordinates the lower learning modules to accomplish the whole task. The method is applied to a simple soccer situation in the context of RoboCup.

1 Introduction

Reinforcement learning (hereafter, RL) is an attractive method for robot behavior acquisition with little or no *a priori* knowledge and higher capability of reactive and adaptive behaviors [3, 2]. However, a simple and straightforward application of RL methods to real robot tasks is difficult because of the enormous time for exploration that scales exponentially with the size of the state/action space. The recent robots tend to have many kinds of sensors like normal and omni-vision systems, touch sensors, infrared range sensors, and so on. They can receive a variety of information from these sensors, especially vision sensors. This fact indicates that the difficulty of RL application to real robot tasks becomes more serious.

Fortunately, a long time-scale behavior might often be decomposed into a sequence of simple behaviors in general, and therefore, the search space can be divided into smaller ones. In the existing studies, however, task decomposition and behavior switching procedures are given by the designers (ex. [3, 9, 10]). Others adopt the heuristics or the assumption that are not realistic from the view point of real robot application (ex. [4, 5, 7, 8]). Doya et al. [6] have proposed MOdular Selection And Identification for Control (MOSAIC), which is a modular RL architecture for non-linear, non-stationary control tasks. This method decomposes a whole state space to a number of areas so that each expert module can produce good performance in the assigned area, however, all learning modules share the one state space that consists of a few variables.

When we develop a real robot that learns various behaviors in its life, it seems reasonable that a human instructs or shows some example behaviors to the robot to accelerate the learning before it starts to learn (ex. [12, 1]). This idea was applied to a monolithic learning module. To cope with more complicated tasks, this idea can be extended to a multi-module learning system. That is, the instruction will help a learner to find useful subtasks.

In this paper, we introduce an idea that the capability of a learning module defines the size of subtasks. We assume that each module can maintain a few number of state variables and this assumption is reasonable for real robot applications. Then, the system decomposes a long-term task into short-term subtasks with self-interpretation of coach instructions so that one learning module with limited computational resources can acquire a purposive behavior for one of these subtasks. We show experimental results with much more sensors such as normal and omni-vision systems and 8 directions infrared range sensors.

2 Basic Idea

There are a learner and a coach in a simple soccer situation (Fig. 1). The coach has *a priori* knowledge of a task to be played by the learner while s/he does not have any idea about the system of the learner. On the other

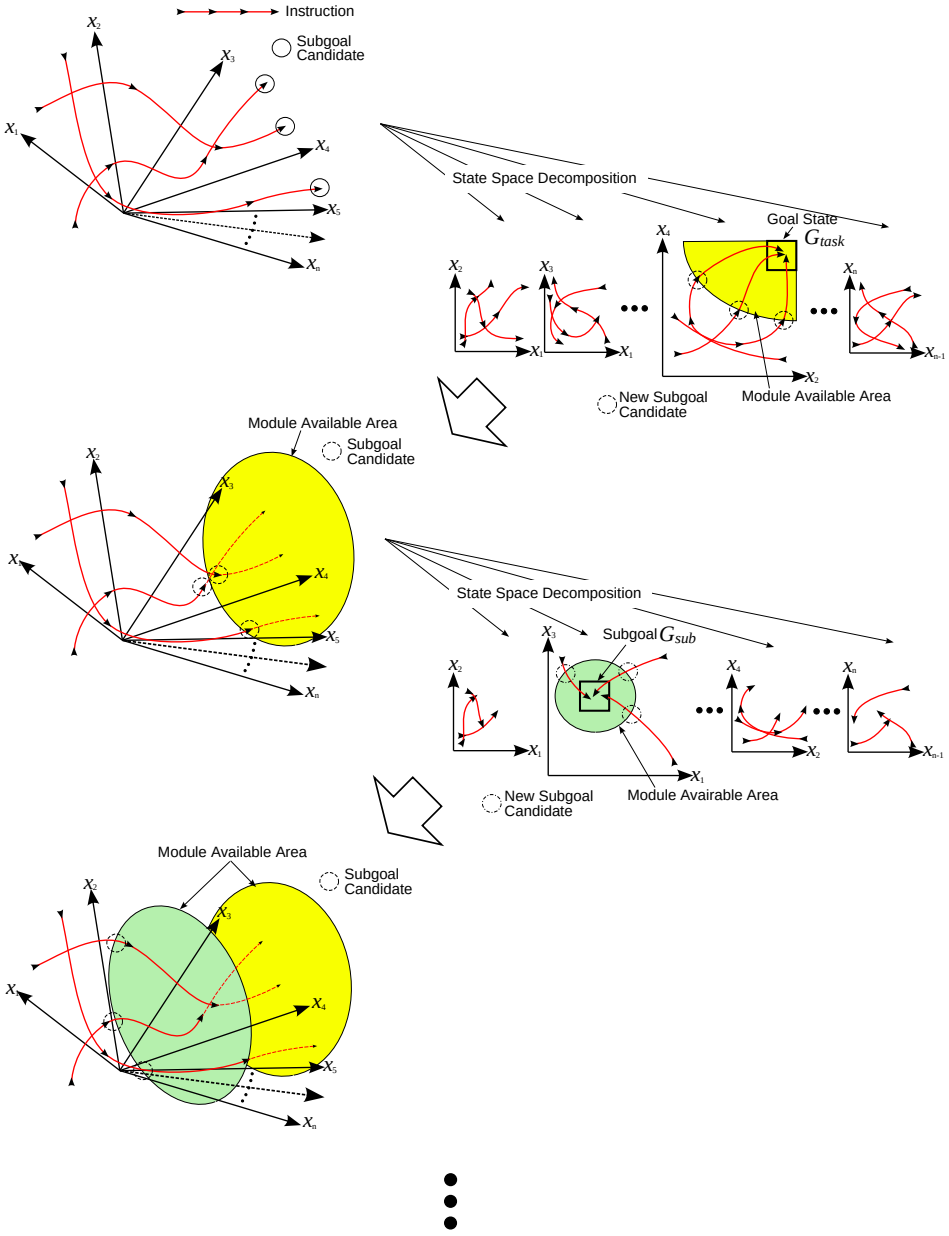


Figure 3: Rough sketch of the idea of task decomposition procedure

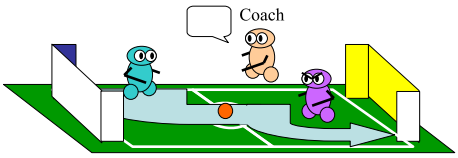


Figure 1: A coach gives instructions to a learner

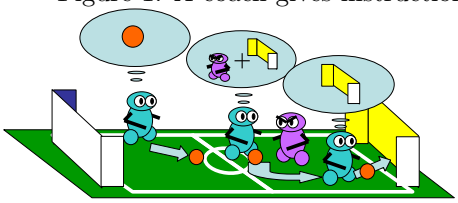


Figure 2: The learner follows the instructions and finds basic behaviors by itself

hand, the learner just follows the instructions without any knowledge of the task. After some instructions given by a coach, the learner decomposes the whole task into a sequence of subtasks, acquires a behavior for each subtask, and coordinates these behaviors to accomplish the task by itself . In Fig. 1, the coach instructs an shooting a ball into a yellow goal with obstacle avoidance. Fig. 2 shows an example that the system decomposes this task into three subtasks and assigns them to three modules that maintain state spaces consist of ball variables, opponent and goal ones, and goal ones, respectively.

Fig. 3 show a rough sketch of the idea of the task decomposition procedure. The top of the Fig. 3 shows a monolithic state space that consists of all state variables $(x_1, x_2, \cdots, x_n)$. The red lines indicate sequences of state value during the given instructions. As we assume beforehand, the system cannot have such a huge state space, then, decomposes the state space into subspaces that consist of a few state variables. The system regards that the ends of the instructions represent goal states of the given task. It checks all subspaces and selects one in which the most ends of the instruction reach a certain area (G_{task} in Fig. 3). The system regards this area as the subgoal state of a subtask which is a part of the given long-term task. The steps of the procedure are as follows:

1. find module unavailable areas in the instructions and regard them as unknown subtask.
2. assign a new learning module.
   (a) list up subgoal candidates for the unknown subtasks on the whole state space.
   (b) decompose the state space into subspaces that consist of a few state variables.
   (c) check all subspaces and select one in which the subgoal candidates reach a certain area best (G_{sub} in Fig. 3).
   (d) generate another leanring module with the selected subspace as a state space and the certain area as the goal state.
3. check the areas where the assigned modules are available.
4. exit if the generated modules cover all segments of instructed behaviors. Else goto 1.

We will explain the details of the implementations of above capabilities from 5 to 7.

3 Robot, Tasks, and assumption

Figure 4: A real robot

Figure 5: Captured camera images

Fig. 4 shows a mobile robot we have designed and built. The robot has a normal camera in front of body, an omni-directional camera on the top, and infra red distance sensors around the body. Fig. 5 show the images of both cameras. A simple color image processing is applied to detect the ball, the goal, and an opponent in the image in real-time (every 33ms). Table 1 shows 39 candidates for state variables. The mobile platform is an omni-directional vehicle (any translation and rotation on the plane). The tasks for this robot are chasing a ball, navigating on the field, shooting a ball into the goal, and so on. We assume that the given task has some absorbing goals, that is, the tasks are accomplished if the robot reaches to certain situations. We also assume

Table 1: The candidates of state variables

state variable	feature
Normal camera	
X_{pb}	Ball X
Y_{pb}	Ball Y
D_{pb}	Ball Distance
A_{pb}	Ball Area
X_{pm}g	Own Goal X
Y_{pm}g	Own Goal Y
D_{pmg}	Own Goal Distance
A_{pmg}	Own Goal Area
X_{pog}	Opponent Goal X
Y_{pog}	Opponent Goal Y
D_{pog}	Opponent Goal Distance
A_{pog}	Opponent Goal Area
Omni vision camera	
Xob	Ball X
Y_{ob}	Ball Y
D_{ob}	Ball Distance
A_{ob}	Ball Area
θ_{ob}	Ball Angle
X_{omg}	Own Goal X
Y_{omg}	Own Goal Y
D_{omg}	Own Goal Distance
A_{omg}	Own Goal Area
θ_{omg}	Own Goal Angle
X_{oog}	Opponent Goal X
Y_{oog}	Opponent Goal Y
D_{oog}	Opponent Goal Distance
A_{oog}	Opponent Goal Area
θ_{oog}	Opponent Goal Angle
θ_{bmg}	Relative Angle between ball and own goal
θ_{bog}	Relative Angle between ball and opponent goal
D_{bmg}	Relative distance between ball and own goal
D_{bog}	Relative distance between ball and opponent goal
IR sensor	
IR_n	IR north
IR_{ne}	IR north east
IR_e	IR east
IR_{se}	IR south east
IR_s	IR south
IR_{sn}	IR south west
IR_w	IR west
IR_{nw}	IR north west

that a long time-scale task can be decomposed into a sequence of simple sub-tasks and the state space of each simple sub-task consists of a few state variables.

4 Hierarchical Multi-Module Learning System

The architecture of multi-module learning system is the same one in [11]. We briefly review it here. The robot prepares learning modules of one kind, makes a layer with these modules, and constructs a hierarchy between the layers. The hierarchy of the learning module's layers can be regarded as a role of task decomposition. Each module has a forward model (predictor) which represents the state transition model, and a behavior learner (policy planner) which estimates the state-action value function based on the forward model in an RL manner (Fig. 6(b)). The state and the action are constructed using sensory information and motor commands, respectively at the bottom level.

The input to and output from the higher level are the goal state activation and the behavior command, respectively, as shown in Fig. 6. The goal state activation g is a normalized state value, and $g = 1$ when the situation is the goal state. When the module receives the behavior command b from the higher modules, it calculates the optimal policy for its own goal, and sends action commands to the lower module. The action command at the bottom level is translated to an actual motor command, then the robot takes an action in the environment.

An approximated action value function $Q(s, a)$ for a state action pair (s, a) and state value function $V(s)$ are given by

$$Q(s,a) = \sum_{s'} \hat{\mathcal{P}}^a_{ss'} \left[\hat{\mathcal{R}}^a_{ss'} + \gamma \max_{a'} Q(s', a') \right] , \quad (1)$$

$$V(s) = \max_a Q(s,a) , \quad (2)$$

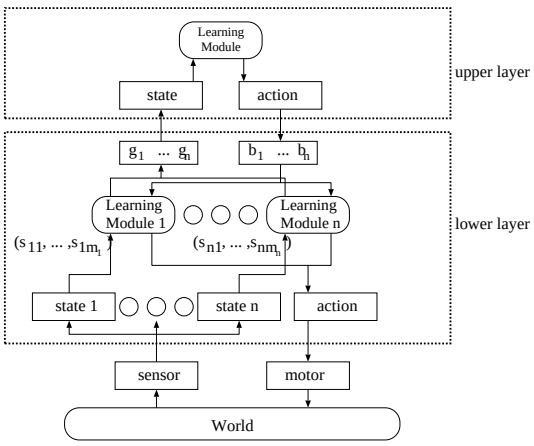
where $\hat{\mathcal{P}}^a_{ss'}$ and $\hat{\mathcal{R}}^a_{ss'}$ are the state-transition probabilities and expected rewards, respectively, and γ is a discount rate.

5 Availability Evaluation

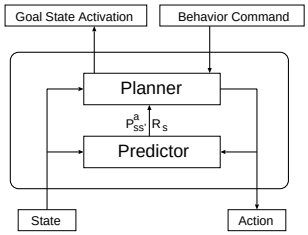
The learner needs to check the availability of learned behaviors that help to accomplish the task by itself because the coach neither knows what kind of behavior the learner has already acquired nor shows perfect example behaviors from the learner's viewpoint. The learner should suppose a module as valid if it accomplishes the subtask even if the greedy policy seems different from the example behavior. Now, we introduce AE in order to evaluate how suitable the module's policy is to the subtask:

$$AE(s, a_e) = \frac{Q(s, a_e) - min_{a'}\, Q(s, a')}{max_{a'}\, Q(s, a') - min_{a'}\, Q(s, a')} , \quad (3)$$

where a_e indicates the action taken in the instructed example behavior. AE becomes larger if a_e leads to the goal state of the module while it becomes smaller if a_e leaves from the goal state (see Fig. 7). Then, we prepare a threshold AE_{th}, and the learner evaluates the module as valid for a period if $AE > AE_{th}$. If there are modules whose AE exceeds the threshold AE_{th}, the learner selects the module which keeps $AE > AE_{th}$ for



(a) A whole system



(b) A module

Figure 6: A multi-layered learning system

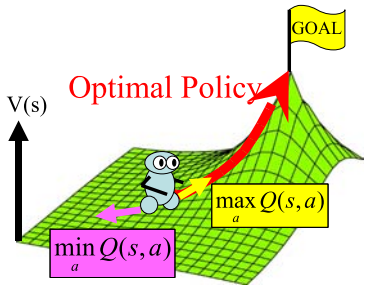


Figure 7: Sketch of state value function and action value

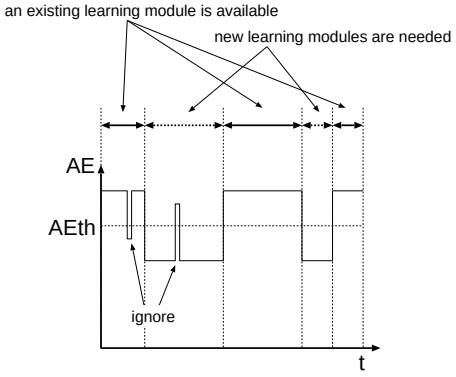


Figure 8: Availability identification during the given sample behavior

longest period among the modules (see Fig. 8). In Fig. 3, "Module Available Area" indecates the one in which $AE > AE_{th}$.

6 New Learning Module Assignment

If there is no module which has $AE > AE_{th}$ for a period, the learner creates a new module which will be assigned to the subtask (see procedure 2 in **??**). To assign a new module to such a subtask, the learner identifies the state space and the goal state. The system follow the two steps to select an appropriate state space and the goal state for the subtask:

- selection of one state variable that specifies the goal state, and
- construction of a state space including the selected state variable.

In order to find one state variable that specifies the goal state best, the system lines up the candidates for a goal region in term of state variables. On the other hand, in order to select another state variable, the system evaluates performance of Q value estimation.

Fig. 9 shows that the system regards the end points of the periods where no module is assigned as candidates for the goal state of the unknown task. In Fig. 3, "Subgoal Candidate" indicate the end point. The idea is that many instructed data reach to the same region if the instructed behaviors contains shared subtasks and the more sensitive state variable tends to be better for control. Each state variable is quantized into a number of blocks beforehand. The system takes histograms of the end points of the unassigned periods on the all state variables (Fig. 10). It regards the highest scored region on each state variable x_i as the goal region GR_i. Then, the system selects one state variable that maximizes the following evaluation measure C_i:

$$C_i = \sum_k l_{i,k} \quad ,$$

where i and $l_{i,k}$ are indices of the state variables x_i and the transition length on the state variables x_i of the k th segmented data of which end points are on the goal region GR_i, respectively. C_i describes that how many segmented data reach to the goal region candidate GR_i and how the state variable is sensitive during the segmented sequences.

The system constructs a state space including the selected state variable. It start from a state space with the one state variable. If the performance of the learned behavior for the subtask is bad (the success rate does not exceed a pre-determined threshold), it adds another state variables with which the accuracy of action value Q estimation improves. It is important to estimate the Q value correctly in order to acquire an appropriate behavior. We introduce an adequacy measure QE defined

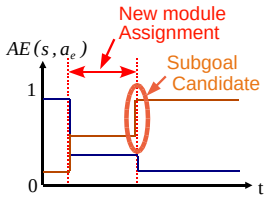


Figure 9: Subgoal candidate point

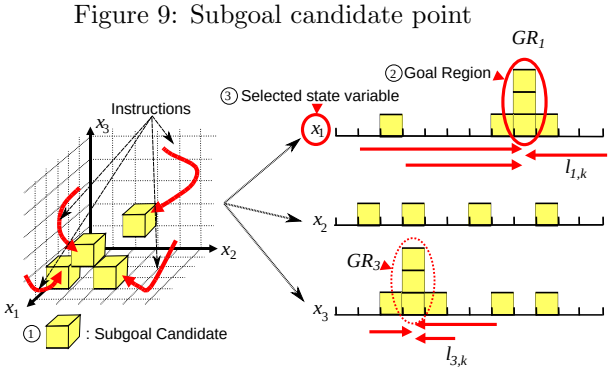


Figure 10: Specification of goal region and its state variable for a subgoal

as equations (4), (5), and (6) :

$$QE = \frac{1}{\sum_{\forall a \in A, \forall s \in S} e(s,a)} . \quad (4)$$

$$e(s,a) = \sqrt{P(1-P)} . \quad (5)$$

$$P = P(V(s') > V(s)|s,a) . \quad (6)$$

$P(V(s') > V(s)|s,a)$ indicates that the probability of the case in which the next state value $V(s')$ is greater than the current state value $V(s)$ when the robot takes action a at the state s and goes to the next state s'. The reason not to use just an error of state value estimation is that it is too sensitive to evaluate the state space with the state value estimation error because the state transition is often probabilistic. We expect that the rough criterion such as it distinguish the state value goes up or down is enough and robust to evaluate the state space.

Even if we got a module with a proper policy based on the state transition model and the reward model, there is no assurance that the module has acquired sufficient performance for the subtask. Before the system adds a new module to the available module database, it checks the success rate of the module. If the success rate is low, the system discards the module.

7 Learning behavior coordination

After the procedures mentioned above, necessary and sufficient modules are built at the lower layer. Then, the learning system needs to put a new learning module at the upper layer (top of the Fig.6(a)) which learns to coordinate the lower modules. The upper module has a state space constructed with the goal state activations of the lower modules. A set of actions consists of commands to the lower modules.

For example, suppose three modules at the lower level (say LM_1, LM_2, and LM_3), then the upper module has a

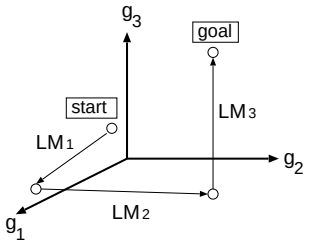


Figure 11: An example state transition on upper layer state space

state space based on their goal state activations (say g_1, g_2, and g_3). Fig. 11 shows an example state transition on the upper layer state space. At the initial situation, all lower modules activate low. The system sends a command to the module LM_1, then the goal state activation of LM_1, that is g_1, goes up. After LM_1 finishes its own task, the upper module sends a command to the module LM_2, and accomplishes the whole task by finally activating LM_3 at last.

8 Experiments

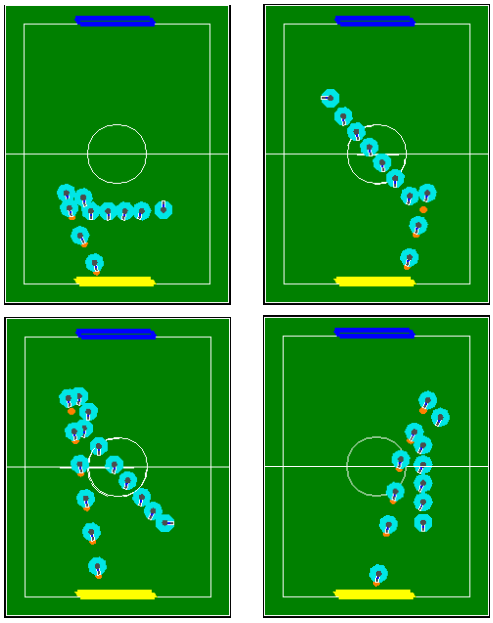


Figure 12: Examples of Instructed behaviors

We have instructed the robot from a simple behavior (ball chasing) to a complicated one (shooting a ball with obstacle avoidance) in [11], however, the there is a criticism that the step-by-step instruction implies task decomposition to the robot. Therefore we adopt only shooting behavior for the task decomposition and the coordination. Fig. 12 shows four examples of the behaviors instructed by the coach. The total number of instruction is 21 for this experiment.

According to the learning procedure, the system produced four modules for the instructed behaviors. The modules are $LM_1(A_{pb}, X_{pb})$, $LM_2(\theta_{og})$, $LM_3(Y_{ob}, X_{ob})$,

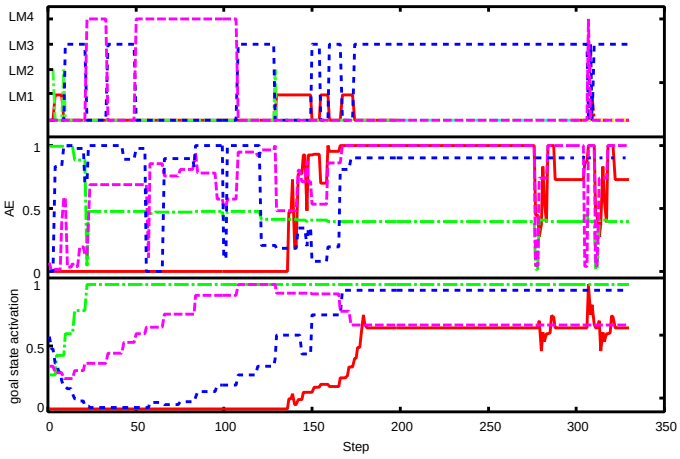


Figure 13: Sequences of the selected module, availability evaluations and goal state activations of modules through an instruction

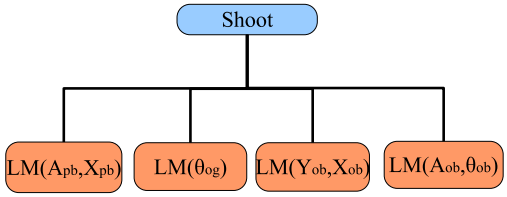


Figure 14: Acquired hierarchy for the shooting behavior

and $LM_4(A_{ob}, \theta_{ob})$. For example $LM_1(A_{pb}, X_{pb})$ indicates that the modules has a state space that consists of the area of ball on the normal camera image $(A_o b)$ and the x position of the ball on the normal camera image (X_{pb}). Fig. 13 shows sequences of the selected module, availability evaluations and goal state activations of modules through an instruction.

Fig. 15 shows the learned behaviors. The start positions of the behaviors are the same ones of the instructions for comparison. The trajectories of learned behaviors are different from the instructed behaviors. This fact indicates that the learner recognizes the subtasks based on its own modules, understands the objectives of the subtasks, and executes appropriate actions for them.

9 Conclusion

We proposed a hierarchical multi-module learning system based on self-interpretation of instructions given by a coach. We applied the proposed method to our robot and showed results for a simple soccer situation in the context of RoboCup.

References

[1] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Vision-based reinforcement learning for purposive behavior acquisition. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 146–153, 1995.

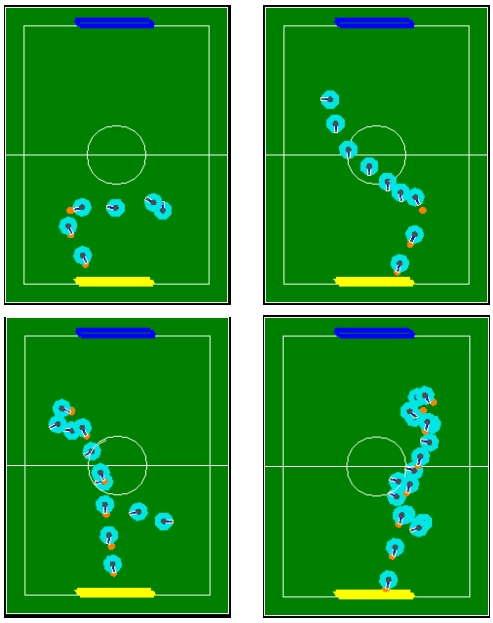


Figure 15: Acquired behaviors for shooting task

[2] M. Asada, S. Noda, S. Tawaratumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, Vol. 23, pp. 279–303, 1996.

[3] Jonalthan H. Connell and Sridhar Mahadevan. *ROBOT LEARNING*. Kluwer Academic Publishers, 1993.

[4] Bruce L. Digney. Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement environments. In Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer, Jordan Pollack, and Stewart W. Wilson, editors, *From animals to animats 4: Proceedings of The fourth conference on the Simulation of Adaptive Behavior: SAB 96*, pp. 363–372. The MIT Press, 1996.

[5] Bruce L. Digney. Learning hierarchical control structures for multiple tasks and changing environments. In Rolf Pfeifer, Bruce Blumberg, Jean-Arcady Meyer, and Stewart W. Wilson, editors, *From animals to animats 5: Proceedings of The fifth conference on the Simulation of Adaptive Behavior: SAB 98*, pp. 321–330. The MIT Press, 1998.

[6] Kenji Doya, Kazuyuki Samejima, Ken ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. Technical report, Kawato Dynamic Brain Project Technical Report, KDB-TR-08, Japan Science and Technology Corporatio, June 2000.

[7] Bernhard Hengst. Generating hierarchical structure in reinforcement learning from state variables. In Riichiro Mizoguchi and John K. Slaney, editors, *6th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000)*, Vol. 1886 of *Lecture Notes in Computer Science*. Springer, 2000.

[8] Bernhard Hengst. Discovering hierarchy in reinforcement learning with HEXQ. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML02)*, pp. 243–250, 2002.

[9] Peter Stone and Mamuela Veloso. Layered approach to learning client behaviors in the robocup soccer server. *Applied Artificial Intelligence*, Vol. 12, No. 2-3, 1998.

[10] Peter Stone and Manuela Veloso. Team-partitioned, opaque-transition reinforcement learning. In M. Asada and H. Kitano, editors, *RoboCup-98: Robo Soccer World Cup II*, pp. 261–272. Springer Verlag, Berlin, 1999.

[11] Yasutake Takahashi, Koichi Hikita, and Minoru Asada. Incremental purposive behavior acquisition based on self-interpretation of instructions by coach. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 686–693, Oct 2003.

[12] Steven D. Whitehead. Complexity and cooperation in q-learning. In *Proceedings Eighth International Workshop on Machine Learning (ML91)*, pp. 363–367, 1991.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究資料
JSAI Technical Report
SIG-Challenge-0521-2 (5/24)

モジュール型学習機構を用いたマルチエージェント環境における競合行動の同時学習

Simultaneous Learning to Acquire Competitive Behaviors in Multi-Agent System based on Modular Learning System

高橋泰岳 枝澤一寛 野間健太郎 浅田稔

Yasutake Takahashi, Kazuhiro Edazawa, Kentarou Noma and **Minoru Asada**
Dept. of Adaptive Machine Systems, Graduate School of Engineering, and
Handai Frontier Research Center, Osaka University
yasutake,eda,noma,asada@er.ams.eng.osaka-u.ac.jp

Abstract

The existing reinforcement learning approaches have been suffering from the policy alternation of others in multiagent dynamic environments. A typical example is a case of RoboCup competitions since other agent behaviors may cause sudden changes in state transition probabilities of which constancy is needed for the learning to converge. The keys for simultaneous learning to acquire competitive behaviors in such an environment are

- a modular learning system for adaptation to the policy alternation of others, and
- an introduction of macro actions for simultaneous learning to reduce the search space.

This paper presents a method of modular learning in a multiagent environment, by which the learning agents can simultaneously learn their behaviors and adapt themselves to the situations as consequences of the others' behaviors.

1 Introduction

There have been an increasing number of approaches to robot behavior acquisition based on reinforcement learning methods [1, 4]. The conventional approaches need an assumption that the state transition is caused by an action of a learning agent so that the learning agent can regard the state transition probabilities as constant during its learning. Therefore, it seems difficult to directly apply the reinforcement learning method to a multiagent system because a policy alteration of other agents may occur, which dynamically changes the state transition probabilities from the viewpoint of the learning agent. RoboCup provides such a typical situation, that is, a highly dynamic, hostile environment, in which agents must obtain purposive behaviors.

There are a number of studies on reinforcement learning systems in a multiagent environment. Kuhlmann and Stone [8] have applied a reinforcement learning system with function approximater to the keep-away problem in the situation of RoboCup simulation league. In their work, only the passer learns his policy to keep the ball away from the opponents. The other agents, receivers and opponents, follow fixed policies given by the designer beforehand.

Asada et al. [2] proposed a method that sets a global learning schedule in which only one agent is specified as a learner and the rest of agents have fixed policies. Therefore, the method cannot handle the alternation of the opponents policies. Ikenoue et al. [3] showed simultaneous cooperative behavior acquisition by fixing learners' policies for a certain period during the learning. These studies suggest that it is enable to acquire a reasonable behavior in a multi-agent environment if the learner can see the environment including the other agents almost fixed because the others keep their policies for a certain time. In the case of cooperative behavior acquisition, both agents do not have any reason to change their policies while they successfully acquire positive rewards with the result of the cooperative behavior for each other. The agents tend to update their policies gradually so that the state transition probabilities seem almost fixed from the view point of the other learning agents.

In case of competitive behavior acquisition in a multiagent environment, however, it is unlikely that the agent tends to select the action that causes positive rewards for the opponents and a negative reward for itself. The punished agent tends to change drastically its policy so that it can acquire a positive reward by which it gives a negative reward to the opponents. This policy alternation causes dynamic changes in the state transition probabilities from the viewpoint of the learning agent therefore it seems difficult to directly apply the reinforcement learning method to a multiagent system.

A modular learning approach would provide one solution to this problem. If we can assign multiple learning modules to different situations, respectively, in each of which the state transition probabilities can be regarded as constant, then the system could show a reasonable performance. Jacobs and Jordan [7] proposed the mix-

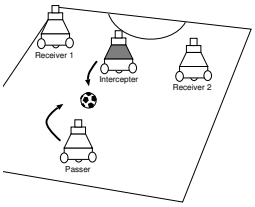


Figure 1: Task : 3 on 1

Figure 2: A real robot

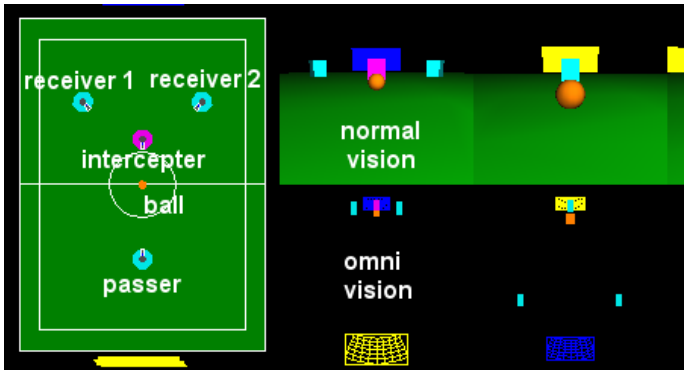


Figure 3: Viewer of simulator

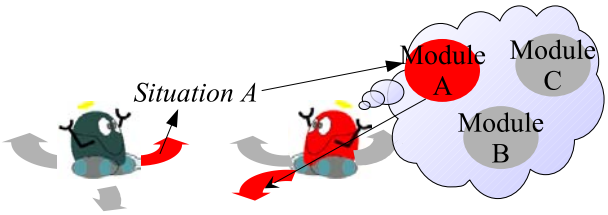


Figure 4: Adaptive behavior selection based on Multi-module learning system

ture of experts, in which a set of the expert modules learn and the gating system weights the output of the each expert module for the final system output. This idea is very general and has a wide range of applications (ex. [11, 9, 14, 13, 5]).

We adopt the basic idea of the mixture of experts into an architecture of behavior acquisition in the multi-agent environment. Takahashi et al. [12] have shown preliminary experimental results of behavior acquisition in the multi-agent environment, however, the learning modules were assigned by the human designer. In this paper, first, we show how it is difficult to directly introduce a multi-module learning system for even single agent learning in a multi-agent environment because of its complexity and introduce a simple learning scheduling which makes it relatively easy to assign modules automatically. Second, we introduce macro actions to realize simultaneous learning in multiagent environment by which the each agent does not need to fix its policy according to some learning schedule. Elfwing et al. [6] introduced macro actions to acquire a cooperative behavior with two real rodent robots. The exploration space with macro actions becomes much smaller than the one with primitive actions, then, the macro action increases the possibility to meet cooperative experiences and leads the two agents to find a reasonable solution in realistic learning time. We show the introduction of macro actions enable the agents to learn competitive behaviors simultaneously. We have applied the proposed multi-module learning system to soccer robots which participate in RoboCup competition and show experimental results on computer simulation and real robot implementation.

2 Tasks and assumption

Fig.1 shows a situation which the learning agents are supposed to encounter. The game is like a three on one; there are one opponent and other three players. The player nearest to a ball becomes to a passer and passes the ball to one of the teammates (receivers) while the opponent tries to intercept it.

Fig. 2 shows a mobile robot we have designed and built. Fig. 3 shows the viewer of our simulator for our robots and the environment. The robot has an omni-directional camera system. A simple color image processing is applied to detect the ball, the intercepter, and the receivers on the image in real-time (every 33ms). The left of Fig. 3 shows a situation in which the agent can encounter and the right images show the simulated ones of the normal and omni vision systems. The mobile platform is an omni-directional vehicle (any translation and rotation on the plane).

3 Multi-Module Learning System

3.1 Basic Idea

The basic idea is that the learning agent could assign one behavior learning module to one situation which reflects other agent's behavior and the learning module would acquire a purposive behavior under the situation if the agent can distinguish a number of situations in each of which the state transition probabilities are almost constant. We introduce a modular learning approach to realize this idea (Fig. 4). A module consists of a learning component that models the world and a action planner . The whole system follows these procedures:

- select a model which represents the best estimation among the modules,
- update the model, and

- calculate action values to accomplish a given task based on dynamic programming.

As an experimental task, we suppose ball passing with possibility of being intercepted by the opponent (Figs. 1 and 3). The problem for the passer (intercepter) here is to select one model which can most accurately describe the intercepter's (passer's) behavior from the viewpoint of the agent and to take an action based on the policy which is planned with the estimated model.

3.2 Architecture

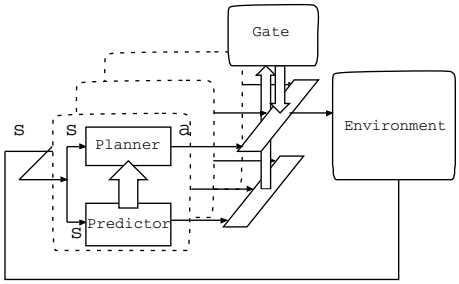


Figure 5: A multi-module learning system

Fig. 5 shows a basic architecture of the proposed system, that is, a multi-module reinforcement learning system. Each module has a forward model (predictor) which represents the state transition model, and a behavior learner (action planner) which estimates the state-action value function based on the forward model in a reinforcement learning manner. This idea of combination of a forward model and a reinforcement learning system is similar to the H-DYNA architecture [10] or MOSAIC [5]. The system selects one module which has the best estimation of a state transition sequence by activating a gate signal corresponding to a module while deactivating the gate signals of other modules, and the selected module sends action commands based on its policy.

3.2.1 Predictor

Each learning module has its own state transition model. This model estimates the state transition probability $\hat{\mathcal{P}}^a_{ss'}$ for the triplet of state s, action a, and next state s':

$$\hat{\mathcal{P}}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \qquad (1)$$

Each module has a reward model $\hat{\mathcal{R}}^a_{ss'}$, too:

$$\hat{\mathcal{R}}^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \qquad (2)$$

We simply store all experiences (sequences of state-action-next state and reward) to estimate these models.

3.2.2 Planner

Now we have the estimated state transition probabilities $\hat{\mathcal{P}}^a_{ss'}$ and the expected rewards $\hat{\mathcal{R}}^a_{ss'}$, then, an approximated state-action value function $Q(s, a)$ for a state action pair s and a is given by

$$Q(s,a) = \sum_{s'} \hat{\mathcal{P}}^a_{ss'} \left[\hat{\mathcal{R}}^a_{ss'} + \gamma \max_{a'} Q(s', a') \right] , \qquad (3)$$

where γ is a discount rate.

3.2.3 Module Selection for Action Selection

The reliability of the module becomes larger if the module does better state transition prediction during a certain period, else it becomes smaller. We assume that the module which does the best state transition prediction has the best policy against the current situation because the planner of the module is based on the model which describes the situation best. In the proposed architecture, the reliability is used for gating the action outputs from modules. We calculate an execution-time reliability ${}^{exec}g_i$ of the module i as follows:

$${}^{exec}g_i = \prod_{t=-T+1}^{0} e^{\lambda p_i^t}$$

where p_i is an occurrence probability of the state transition from the previous $(t-1)$ state to the current (t) one according to the model i, and λ is a scaling factor. The T indicate a period (step) to evaluate the reliability of the module and we set T as 5 in the following experiments. The agent continues to use the module for a certain period, for example 5 step or 1 second, after it changes the module in order to avoid oscillation of the policies.

3.2.4 Module Selection for Updating Models

We use an update-time reliability ${}^{update}g_i$ of the module for updating modules. The calculation of this reliability contains the future state transition probabilities:

$${}^{update}g_i = \prod_{t=t-T}^{t+T} e^{\lambda p_i^t} \qquad .$$

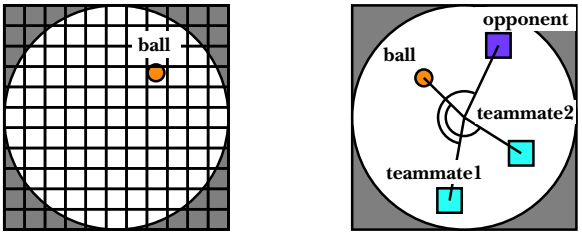
4 Behaviors Acquisition under Scheduling

As we mentioned in 1, first, we show how it is difficult to directly introduce the proposed multi-module learning system in the multi-agent system. We introduce a simple learning scheduling in order to make it relatively easy to assign modules automatically.

4.1 Configuration

The state space is constructed in terms of the centroid of the ball on the image, the angle between the ball and the intercepter, and the angles between the ball and the receivers (see Figs. 10 (a) and (b)). We quantized the ball position space 11 by 11 as shown in Fig. 10 (a) and the each angle into 8. As a result, the number of states becomes $11^2 \times 8 \times 8 \times 8 = 61952$. The action space is constructed in terms of desired three velocity values (x_d, y_d, w_d) to be sent to the motor controller (Fig. 7). Each value is quantized into three, then the number of action is $3^3 = 27$. The robot has a pinball like kick device, and it automatically kicks the ball whenever the ball comes to the region to be kicked. It tries to estimate the mapping from sensory information to appropriate motor commands by the proposed method.

The initial positions of the ball, the passer, the intercepter, and receivers are shown in Fig. 1. The opponent



(a) state variables (position) (b) state variables (angle)

Figure 6: State variables

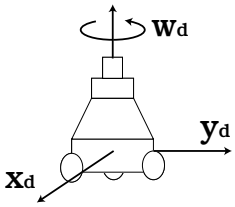


Figure 7: Action variables

has two kinds of behaviors; it defend the left side, or right side. The passer agent has to estimate which direction the intercepter will defend and go to the position in order to kick the ball to the direction the intercepter does not defend. From a viewpoint of the multi-module learning system, the passer will estimate which situation of the module is going on, select the most appropriate module to behave. The passer acquires a positive reward when it approach to the ball and kicks it to one of the receivers.

4.2 Learning Scheduling

We prepare a learning schedule composed of three stages to show its validity. The opponent fixes its defending policy as right side block at the first stage. After 250 trials, the opponent changes the policy to block the left side at the second stage and continues this for another 250 trials. Then, the opponent changes the defending policy randomly after one trial.

4.3 Simulation Result

We have applied the method to a learning agent and compared it with only one learning module. We have also compared the performances between the methods with and without the learning scheduling. Fig. 8 shows the success rates of those during the learning. The success indicates that the learning agent successfully kick the ball without interception by the opponent. The success rate indicates the number of successes in the last 50 trials. The "mono. module" in the figure indicates "monolithic module" system and it tries to acquire a behavior for both policies of the opponent. The multi-module system with scheduling shows a better performance than the one-module system. The monolithic module with scheduling means that we applied learning scheduling mentioned in **4.2** even though the system has only one learning module. The performance of this system is similar with multi-module system until the end of first stage (250 trials), however, it goes down at the second stage because the obtained policy is biased against the experiences at the fist stage and cannot follow the policy change of the opponent. Since the opponent takes one of the policies at random at the third stage, the learning agent obtains about 50% of success rate. "without scheduling" means that we do not applied learning scheduling and the opponent changes its policy at random from the start. Somehow the performance of the monolithic module system without learning scheduling is getting worse after the 200 trials. The multi-module system without learning schedule shows the worst performance in our experiments. This result indicates that it is very difficult to recognize the situation at the early stage of the learning because the modules has too few experiences to evaluate their fitness, then the system tends to select the module without any consistency. As a result, the system cannot acquires any valid policies at all.

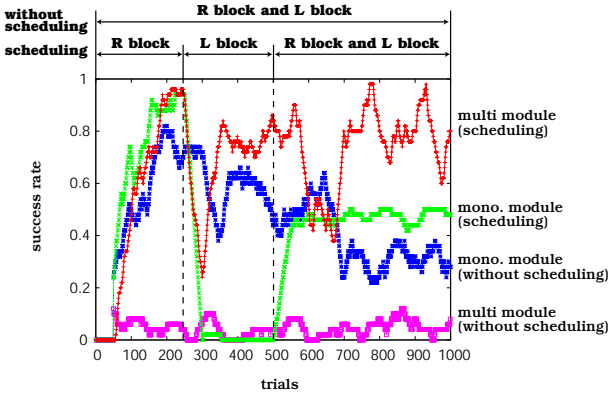


Figure 8: Success rate during the learning

5 Simultaneous Learning with Macro Actions

We introduce macro actions to realize simultaneous learning in multiagent environment by which the each agent does not need to fix its policy according to some learning schedule. In this experiment, the passer and the intercepter learn their behaviors simultaneously. The passer learns behaviors for different situations which are caused by the alternation of the intercepter's policies, that is, blocking left side or right one. The intercepter also learns behaviors for different situations which are caused by the alternation of the passer's policies, that is, passing a ball to left receiver or right one.

5.1 Macro Actions and State Spaces

Fig. 9 shows the macro actions of the passer and the intercepter. The macro actions for the intercepter are blocking the pass way to the left receiver and right one. On the other hand, the macroaction for the passer are turning left, turning right around the ball, and approaching to the ball to kick it. A ball gazing control is embedded to the both learners. The number of the actions

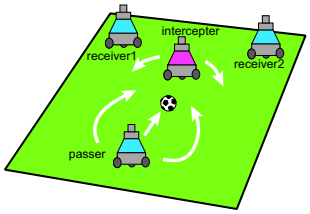
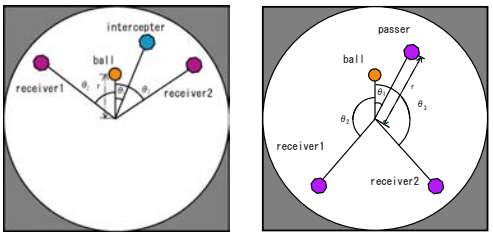


Figure 9: Macro actions



(a) passer (b) intercepter

Figure 10: State variables

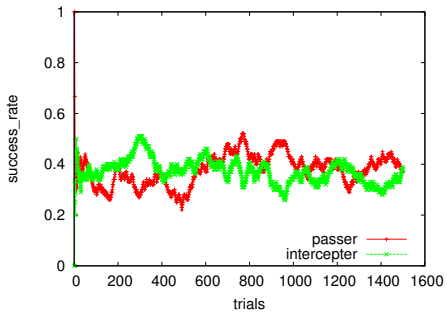


Figure 11: Sequence of success rates during simultaneous learning

reduced from 27 (see 4.1) primitives to 2 or 3 macro actions. The state space for the passer is constructed in terms of the y position of the ball on the normal image, and the angle between the ball and the centers of intercepter, the ones between the balls and the two receivers on the image of omni-directional vision. The number of the states reduced from 61952 (see 4.1) to 3773 because the set of macro actions enable us to select smaller number of state variables and coarser quantization. The state space for the intercepter is constructed in terms of the y position of the passer on the image of normal vision system, and the angle between the ball and the passer and the ones between the ball and the two receivers on the image of omni-directional vision. The number of the states is 2695.

5.2 Experimental Resutls

We have checked how the simultaneous learning of the passer and intercepter works on our computer simulation. Both agents start to learn their behaviors from scratch and have 1500 trials without any scheduling. Fig. 11 show the success rates during the simultaneous learning of the passer and the intercepter. This figure shows that the intercepter has higher success rate at the beginning of learning, the passer is getting to acquire the appropriate behaviors corresponding to the intercepter's behaviors, and the both agents have almost equal success rate at the end of learning stage. The sum of the both success rates is not 1 because the both player sometimes failed to pass or intercept simultaneously.

In order to check if the both learners acquire appropriate behaviors against the opponent's behaviors, we fixed one agent's policy and check that the other can select an appropriate behavior and its success rate. Table 1 shows this results. The both players have two modules and assign them to appropriate situations by themselves. LM and the digit number right after the LM indicate Learning Module and index number of the module, respectively. For example, the passer uses both LM0 and LM1 and the intercepter use only LM0, then the passer's success rate, intercepter's success rate, and draw rate are 59.0 %, 23.0%, and 18.0%, respectively. Apparently, the player switching multi-modules achieves higher success rate than the opponent using only one module. These results show that the multi-module learning system works well for both.

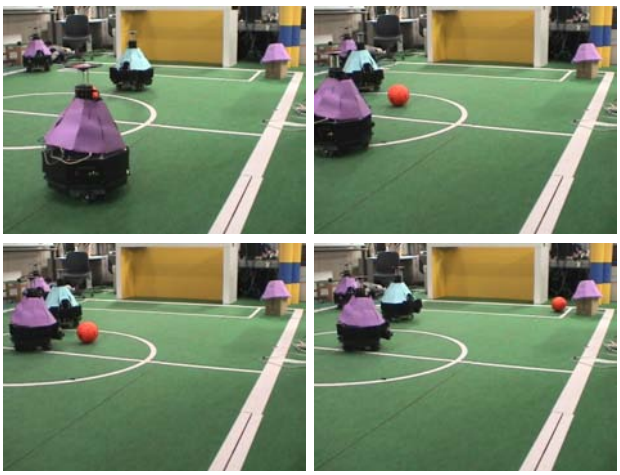


Figure 12: A sequence of a behavior of passing a ball to the right receiver while the intercepter blocks the left side

We have applied the same architecture to the real robots. Fig. 12 shows the one example behaviors by real robots. First, the intercepter tried to block the left side, then the passer approached the ball with intention to pass it to the right receiver. The intercepter found that it tried to block the wrong side and change to block the other side (right side), but, it is too late to intercept the ball and the passer successfully pass the ball to the right receiver.

6 Conclusion

In this paper, we proposed a method by which multiple modules are assigned to different situations which are caused by the alternation of the other agent policy and learn purposive behaviors for the specified situations as consequences of the other agent's behaviors.

We introduced macro actions to realize simultaneous learning of competitive behaviors in a multi-agent sys-

Table 1: Success rates for passer and receiver in different cases

passer	intercepter	passer's success rate[%]	intercepter's success rate [%]	draw rate[%]
LM0,LM1	LM0	59.0	23.0	18.0
LM0,LM1	LM1	52.7	34.3	13.0
LM0	LM0,LM1	25.6	55.0	19.4
LM1	LM0,LM1	26.0	59.3	14.7
LM0,LM1	LM,LM1	37.6	37.3	25.1

tem. We have shown results of a soccer situation and the importance of the learning scheduling in case of none-simultaneous learning without macro actions and the validity of the macro actions in case of simultaneous learning in the multi-agent system.

References

[1] M. Asada, S. Noda, S. Tawaratumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23:279–303, 1996.

[2] M. Asada, E. Uchibe, and K. Hosoda. Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development. *Artificial Intelligence*, 110:275–292, 1999.

[3] Shoichi Ikenoue Minoru Asada and Koh Hosoda. Cooperative behavior acquisition by asynchronous policy renewal that enables simultaneous learning in multiagent environment. In *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 2728–2734, 2002.

[4] Jonalthan H. Connell and Sridhar Mahadevan. *ROBOT LEARNING*. Kluwer Academic Publishers, 1993.

[5] Kenji Doya, Kazuyuki Samejima, Ken ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. Technical report, Kawato Dynamic Brain Project Technical Report, KDB-TR-08, Japan Science and Technology Corporatio, June 2000.

[6] Stefan Elfwing, Eiji Uchibe, Kenji Doya, and Henrik I. Christensen1. Multi-agent reinforcement learning: Using macro actions to learn a mating task. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages CD–ROM, Sep 2004.

[7] R. Jacobs, M. Jordan, Nowlan S, and G. Hinton. Adaptive mixture of local experts. *Neural Computation*, (3):79–87, 1991.

[8] Gregory Kuhlmann and Peter Stone. Progress in learning 3 vs. 2 keepaway. In Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors, *RoboCup-2003: Robot Soccer World Cup VII*. Springer Verlag, Berlin, 2004.

[9] Satinder P. Singh. The effeicient learnig of multiple task sequences. In *Neural Information Processing Systems 4*, pages 251–258, 1992.

[10] Satinder P. Singh. Reinforcement learning with a hierarchy of abstract models. In *National Conference on Artificial Intelligence*, pages 202–207, 1992.

[11] Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 1992.

[12] Yasutake Takahashi, Kazuhiro Edazawa, and Minoru Asada. Multi-module learning system for behavior acquisition in multi-agent environment. In *Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages CD–ROM 927–931, October 2002.

[13] J. Tani and S. Nolfi. Self-organization of modules and their hierarchy in robot learning problems: A dynamical systems approach. Technical report, Sony CSL Technical Report, SCSL-TR-97-008, 1997.

[14] Jun Tani and Stefano Nolfi. Self-organization of modules and their hierarchy in robot learning problems: A dynamical systems approach. Technical report, Technical Report: SCSL-TR-97-008, 1997.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究資料
JSAI Technical Report
SIG-Challenge-0521-3 (5/24)

自律移動ロボットにおける強化学習による軌道生成法の検討

Trajectory Generation for a Mobile Robot by Reinforcement Learning

清水 昌樹, 藤田 誠, 宮本 弘之
Masaki Shimizu, Makoto Fujita, Hiroyuki Miyamoto
九州工業大学大学院
Kyushu Institute of Technology, Kitakyushu
shimizu-masaki@edu.brain.kyutech.ac.jp

Abstract

強化学習はロボットに自律的に行動を生成させるための非常に有効な方法である。強化学習の中のQ-learningは比較的簡単なな方法であるが、タスクが複雑になるに従い学習時間が長くなる。さらに離散的に状態と行動空間を分割するため、ロボットはスムーズな行動を得ることができない。これら問題を解決するため、我々は二つの方法を提案する。我々の方法の有効性を確かめるため、我々はロボットサッカーのような動的環境におけるBall-To-Goalタスクの軌道生成にQ-learningを適用するシミュレーションを行った。

1 緒言

人間の仕事の一部をロボットに分担させたり、人間にとって作業が困難な環境や危険な環境におけるロボットの活躍が望まれている。その分野において近年、自律移動ロボットに関する研究が盛んである。ロボットが直面するタスクに対して人間が設計した行動計画は優れた性能を示す。しかし設計者の能力以上の複雑なタスクでは行動計画の設計者にかかる負担は大きくなる。

設計者の負担を軽減させるために強化学習を使いタスクの最適な行動計画を自律的に生成させることでロボットに行動を獲得させる。行動の指針を与える必要のある教師あり学習とは異なり、強化学習は環境の観測と可能な行動から自律的に学習が可能である。

もし観測可能な状態と行動が多すぎると学習は収束しない。さらにQ-learningでの状態や行動は不連続である。これらの問題のため、エージェントと呼ばれる学習者は実環境や連続空間に対して正確に反応することが難しい。最近、この問題を解くため多くの研究がなされている。例えばQ-learningに連続空間を使うためいくつかの研究が提案されている[3][4]。さらに階層型強化学習によって巨大な状態行動空間を解く研究も提案されている[5][6]。

本論分では上記の問題を解くため２つのシンプルな方法を提案する。最初の方法は本来未知な学習の初期値を設計者が設定することである。少ない知識を与えることによってエージェントは設計者の意図を理解し学習を始める。設計者が与えた初期値が最適な行動計画に近ければ、エージェントは全く間違った方向を探索しなくても良い。２つめの方法は、状態と行動を等間隔で分割しないことである。移動ロボットの場合、従来は設定した数に従って状態と行動を等間隔に分割していた。しかし様々なターゲットへ向かう行動を獲得するタスクでは、このような分割は便利ではない。従って我々はターゲット周辺方向の行動空間を細かく分割し、その他の行動空間を荒く分割した。

本論文では、これら２つの方法を使った１つの学習モジュールだけでBall-To-Goalタスクの行動軌道を２輪移動ロボットに獲得させることを目的とする。またロボットは単純なBall-To-Goalタスクだけでなく攻撃や守備のような状況に応じた高いレベルのタスクの複雑な行動軌道も獲得させる。提案した方法の有効性を調べるためシミュレーションにより実験を行った。

2 プルトイ法

Ball-To-Goalタスクはターゲットに向かうという意味からTo-BallタスクとTo-Goalタスクに分けることができる。そこでこの節では、まずロボットにBall-To-Goalタスクを学習させる前に人の手によって設計した理想的なTo-Ballタスクでの運動制御法を提案する。従来、我々が用いていた運動制御法では、前進や回転の速度を加えたものがモータに与えられる。この方法だとロボットは常に前進指令を受けていた。しかしこれではロボットの進行方向とボールの進行方向が大きく異なっていたとき、すばやくボールにアプローチできていなかった。これまでの運動制御法はロボットからボールまでの距離と角度か

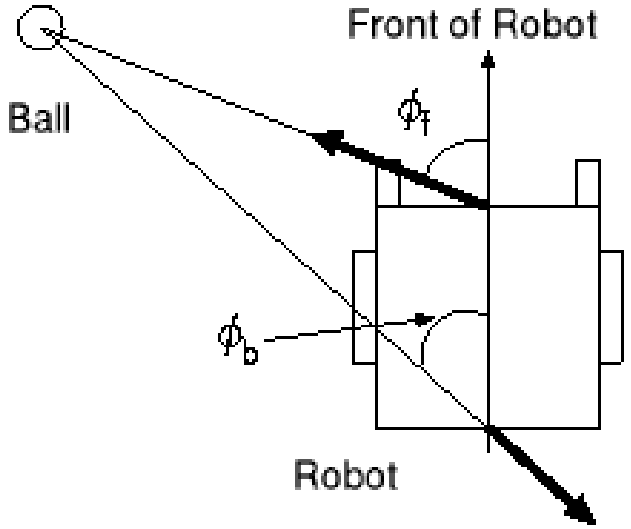


Figure 1: A new approach "Pulltoy"

ら以下のように計算される。

$$\begin{cases} V_r &= V_c + A \times L \\ V_l &= V_c - A \times L \end{cases} \quad (1)$$

ここで V_r は各ホイールの速度、A は角速度に関するパラメータ、L は左右のタイヤの間隔、V_c はロボットの基準速度である。式（1）による方法を便宜上、スラスター法と呼ぶ。

式（1）はスムーズに行動を生成するが、ボールがロボットの後方にあるときでさえ、前進に回転指令を加えボールにアプローチしていた。ロボットサッカーにおいては、ボールにどれだけ早くアプローチできるかが非常に重要になる。従って我々は人が行うように状況に応じて前進より回転を優先させる新しい運動制御法を提案する。

人はボールが自身の後ろにある場合、まずボールに向き、その後ボールへアプローチする。このようなアプローチがロボットには有効だと考える。しかし人間の行うアプローチを正確に解析し設計するのは難しい。従って新しいアプローチが必要になる。この論文ではボールにスムーズにアプローチする人の方法を簡略化した方法を提案する（図 1）。新しい方法は式の中にロボットの先端からボールへの角度を使う。この角度が大きくなると図中の矢印の方向に引っ張られすばやく回転する。我々はこの新しいアプローチをプルトイ法[1] と呼ぶ。

しかしロボットの先端から引っ張られるだけでは最適な行動を設計することはできないので、図 1 のようにロボットの後方とボールを結んだ角度も使用する。これは引張る力ではなく押す力になる。図 1 の Φ_b は式を最適化すると省略できる。従ってプルトイ法は式（2）になる。

[1] プルトイとは子供が小さな木馬や車の先端に結ばれた紐を引っ張って遊ぶおもちゃである。

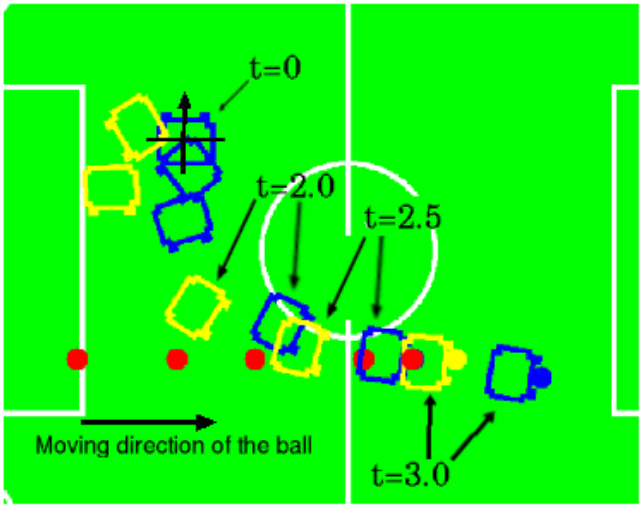


Figure 2: Comparison of Pulltoy and Thruster trajectory. Blue is Pulltoy. Yellow is Thruster. Ball is moving from left below to right.Pulltoy approach is acquiring the ball at t=2.5. Thruster is acquiring it at t=3.0.

$$\begin{cases} V_r &= V_{max} \\ V_l &= V_{max}(\cos(\phi_f) - \sin(\phi_f)) \end{cases} \quad (0 \leq \phi_f < \frac{\pi}{2})$$

$$\begin{cases} V_r &= -V_{max}(\cos(\phi_f) - \sin(\phi_f)) \\ V_l &= -V_{max} \end{cases} \quad (\frac{\pi}{2} \leq \phi_f \leq \pi) \quad (2)$$

（もし Φ_f がマイナスの場合、r と l が逆転する）ここで Φ_f はロボット先端からボールまでの角度、V_{max} はロボットの最大速度である。式（2）は非常に単純な方法である。さらに Φ_f はボールだけでなく他のターゲットに対しても使える。

我々は次にスラスター法とプルトイ法の軌道を比較した（図 2）。

図のようにロボットの背後にボールがある状況では特に効果的であることがわかった。さらにロボットからボールへの初期角度、距離を変え比較実験を行った（図 3）。縦軸はボールを得るまでの時間差でスラスター法からプルトイ法を引いている。横軸はロボットからボールまでの下図が初期角度、上図が初期距離である。これらの図からほとんどの条件においてスラスター法よりプルトイ法が効果的だといえる。もし初期角度が大きく、初期距離が小さい場合、その傾向は顕著になる。

3 強化学習

強化学習において学習エージェントは環境から状態を観測した後、ポリシーから行動を選択する。その後エージェントは変化した環境を観測することで報酬を得る。最終的にエージェントは受け取った報酬に従いポリシーを修正する。エージェントはこれを繰り返し学習する。我々は

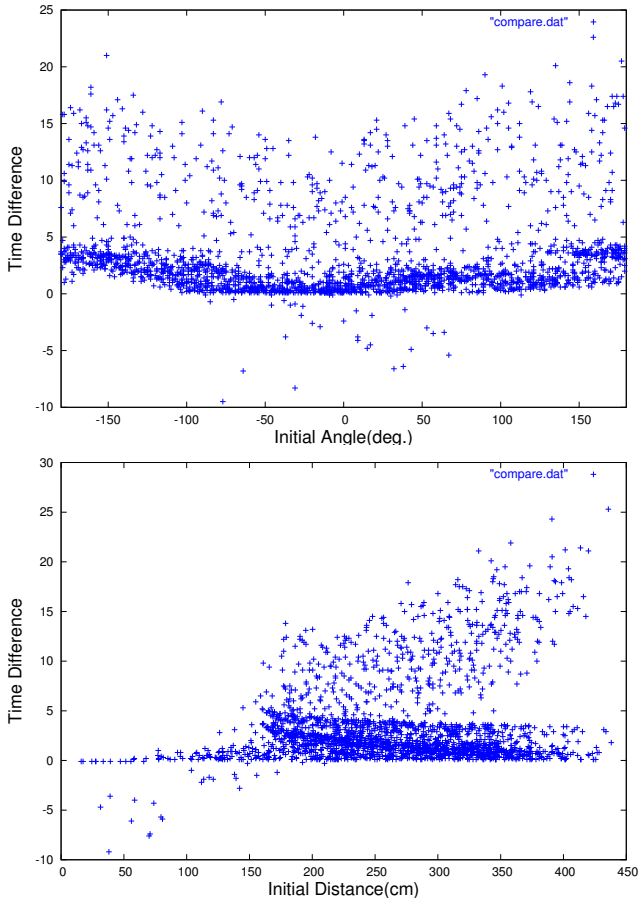


Figure 3: Graph of varying initial angle and distance. Upside in each graph represent that the Pulltoy approach could get the ball before the Thruster approach.

強化学習の１つである Watkins el al. によって提案された Q-learning を使う。Q-learning は行動価値関数 Q(s,a) と呼ばれる関数を持ち、ポリシーπにおける状態 s での行動 a を得るときの価値を計算する。この関数は式（３）で定義され更新される。

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (3)$$

ここで r は報酬、γ は学習率、s_{t+1} は $t+1$ での状態、α は一定のステップサイズパラメータである。

学習開始時において正しい行動価値関数が未知なためエージェントは探索行動をしなければならない。エージェントは手本を使わず無知な状態から探索を開始する。そのため報酬を学習初期の段階で獲得することは難しく、良い行動を獲得するには時間がかかる。我々はこの問題を解くため行動を選択するためのポリシーに最適な初期値を前もって設定する。人の手によって比較的設計が簡単なタスクでは初期値を設定できる。設定された初期値から探索を開始することによって効率的な学習が期待できる。この考え方は人にも言えることで、あるタスクを人に教える場合、まず手本を見せることが重要である。本論文

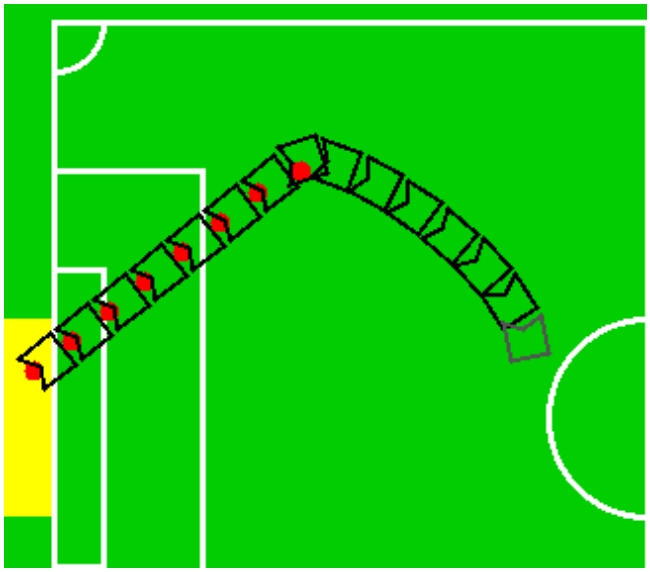


Figure 4: Trajectory of the robot for Ball-To-Goal task that generated by RL setting initial value

では前節で提案した To-Target タスクでのプルトイ法を Ball-To-Goal タスクでの強化学習法の初期値として使用した。

4 状況に応じた戦略の獲得

我々はエージェントに単なる Ball-To-Goal タスクだけでなく状況に応じたハイレベルなタスクでの軌道も獲得させる。ハイレベルなタスクではエージェントとボール、ゴールだけでなく複雑な環境の下でのタスクを考えた場合、通常の直接的なアプローチより、守備や攻撃などロボットの役割やターゲットの種類、敵ロボットの有無によってターゲットへのアプローチの仕方を変える方がより効果的であるといえる。例えば相手に攻められ守備をしなければならない場合、エージェントは自分のゴールとボールの間に入りながらアプローチする方が得点の危険性を考慮すると効果的である。

5 シュミレーション実験

Q-learning は一般的に観測可能な状態とエージェントの取りうる行動を空間的に分割する。本実験においても状態と行動の分割を使用した。状態にはロボットからボールおよびゴールへの角度と距離、ボールの速度を使用した。ボールへの角度の分割数は１２、距離は４、ゴールへの角度は１２、距離は４、ボール速度は５とした。移動ロボットの場合、一般的にエージェントの行動はロボットの周囲または取りうる行動をほぼ等間隔に分割する。しかしロボットサッカーでの Ball-To-Goal タスクの場合、行動空間は等間隔に分割すると効率が悪い。よって我々はターゲット周辺の行動空間を密に分割し、他の行動空間を荒く分割した。

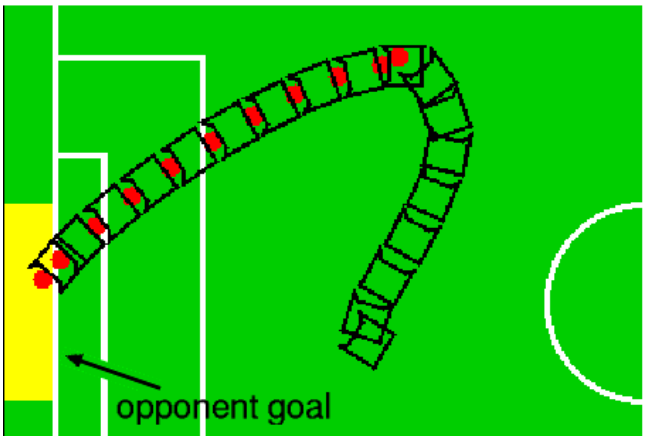


Figure 5: Generated trajectory for offense task

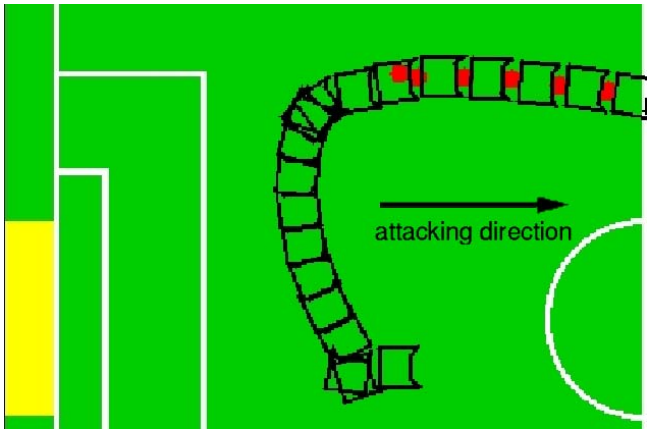


Figure 6: Generated trajectory for defense task

本論文ではシミュレータを使い Ball-To-Goal タスクにおいて軌道を生成させる以下の実験を行った。

5.1 良い初期軌道を与えたときの学習

ロボットの軌道は強化学習でのポリシーの初期値に 2 節でのプルトイ法を使い生成した。この結果を図 4 に示す。この時の報酬はボールをゴールへ入れた場合、1 を与えている。

プルトイ法やスラスター法などのように設計が簡単で一般的な方法では To-Ball や To-Goal としてターゲットを変えなければならない。しかし本論文で提案した方法を使った強化学習を使用することで初期位置からボールを獲得し、得点するまでを 1 つの軌道として生成することができる。また状態にボールの速度を含んでいるためボールの速度に対応した適切な行動を選択し軌道を生成できる。

しかし実験のような比較的簡単なタスクの場合、良い初期軌道を与えると学習後に生成される軌道は与えた軌道とほとんど変わらない。つまり今回与えた初期軌道であるプルトイ法は学習の手を借りずとも良好な手法であるといえる。

5.2 高度なタスクにおける軌道生成

図 5,6 は単にボールにアプローチするだけでなく役割を与えられた戦略下での高度なタスクでの軌道生成の結果を示す。報酬は役割に応じた方向を向きボールをキャッチした場合、1 を与えた。さらにその状態からボールをゴールへ入れた場合、さらに 1 を与えた。

図 5 で示した攻撃行動は両チームのロボット共ボールに触れていない場合に使用される。ロボットがボールを獲得したときすでにゴールの方向を向いているポリシーを獲得している。相手ロボットがボールを持ちゴールに向かっている場合、守備の戦略が最適である。この戦略ではゴールとボール間の軌道上に自身を移動させ、その後ボールへアプローチする。図 6 は守備戦略の場合、生成した軌道である。

攻撃や守備等の役割を与えられた場合、プルトイ法や他の運動制御法では満足な結果が得られない。報酬関数は複雑になるが人の手では設計が難しいタスクでは強化学習法が有効である。

6 結言

我々は実験結果から To-Ball タスクや To-Goal タスクのような To-Target タスクにおいてスラスター法よりプルトイ法が有効なことを示した。もしロボットが小さいサイズ、または強化学習や他の方法ではメモリが足りないような場合、プルトイ法は簡単なため非常に有効だといえる。ロボットは強化学習によって Ball-To-Goal タスクでの軌道を自律的に生成することができた。強化学習を実ロボットへの適用を考慮した場合、学習の効率化、高速化が求められる。この問題は初期値の設定により解決できた。しかし学習後の軌道は人の手による良い初期軌道とほとんど同様の軌道が得られた。このことから単純なタスクに限れば両手法とも有効であることがわかった。

強化学習法において状態空間が大きくなりすぎると学習が収束しなくなる。さらに状況に応じた戦略軌道を設計することは人の手においても難しい。良い初期軌道として用いたプルトイ法であっても戦略に応じた軌道を生成することはできない。これら問題にも我々の比較的単純な方法で対応できた。また攻撃や守備のような難しい戦略に従ってボールにアプローチする軌道も生成できた。今後は複雑になった報酬を簡素化する方法の検討および実ロボットへの適用を目指す。

参考文献

[1] R.S.Sutton, A.Barto, ”Reinforcement Learning:An Introduction”, MIT Press,1998.

[2] J.Peng, R.J.Williams, "Incremental Multi-Step Q-Learning", Machine Learning, Vol.22,pp283-290,1996.

[3] A.Sherstov, P.Stone, "On Continuous-Action Q-Learning via Tile Coding Function Approximation", In Under Review., June 2004.

[4] S.Hagen, B.Kröse, "Neural Q-learning", In Neural Computing & Applications, 12(2), pages 81-88, November 2003.

[5] A.Barto, S.Mahadevan, "Recent Advances in Hierarchical Reinforcement Learning", Discrete Event Dynamic Systems:Theory and Applications, 13,41-77,2003.

[6] Y.Takahashi, M.Asada, "Multi-Layered Learning Systems for Vision-Based Behavior Acquision of A Real Mobile Robot", Proceedings of SICE Annual Conference 2003 in Fukui, Vol.CD-ROM, pp.2937-2942, 2003.

スクリプト言語 Lua を用いたロボカップ四足ロボットリーグシミュレータ

A Simulator of Four-Legged Robot League in RoboCup using an Scripting Language Lua

小林隼人 [1]，井上淳 [1]，石野明 [2]，篠原歩 [3]

Hayato KOBAYASHI[1], Jun INOUE[1], Akira ISHINO[2], and Ayumi SHINOHARA[3]

[1] 九州大学大学院システム情報科学府，[2] 九州大学大学評価情報室，[3] 東北大学大学院情報科学研究科

[1]**Graduate School of Information Science and Electrical Engineering, Kyushu University**

[2]**Office for Information of University Evaluation, Kyushu University**

[3]**Graduate School of Information Sciences, Tohoku University**

{h-koba, j-inoue, ishino}@i.kyushu-u.ac.jp and ayumi@ecei.tohoku.ac.jp

Abstract

In RoboCup four-legged robot league, no modifications or additions to the robot hardware are allowed; hence, there is no difference between teams without software. For optimum results, to improve the software development process is critical. In this paper, we introduce a method of using the scripting language Lua in soccer robots, and we show the soccer simulator, in which Lua scripts are executed as in robots. Using the scripts in both the environments has made it easy to develop soccer robots.

1 はじめに

ロボカップ四足ロボットリーグは，四足のエンターテインメントロボット AIBO (ERS-210, ERS-7) [1]による 4 対 4 のロボットサッカーリーグである．四足ロボットリーグでは，ロボットは完全な自律型であり，ロボット同士での通信と審判からの指示を除いては，外部の人間による操作はもちろん，計算機による制御も許されていない．さらに，ロボットの改造も許されておらず，プログラムの優劣がサッカーの勝敗を大きく左右するリーグであるといえる．したがって，開発効率の向上は四足ロボットリーグにおける重要な課題のひとつであると言える．

四足ロボットリーグで用いている AIBO ERS-7 は 64bit RISC プロセッサーを CPU とし，64MB のメインメモリーと無線 LAN を持ち，OS を含むプログラムはメモリスティックにより供給される．したがって，AIBO を動かすプログラムの開発は，通常以下のようなサイクルで行われる．

1. PC 上で C++ プログラムを作成する．
2. PC 上のクロスコンパイラでコンパイルする．
3. 実行ファイルをメモリスティックにコピーする．
4. メモリスティックを AIBO に挿入する．
5. AIBO の電源を入れて AIBO を起動させる．
6. AIBO を動かして，プログラムの動作の確認をする．
7. AIBO の電源を切り，メモリスティックを抜く．
8. 手順 1 へと戻る．

手順 2 のコンパイルでは数分から数十分を必要とし，手順 5 の AIBO の起動には約一分程度を必要とする．さらに，手順 7 の AIBO の終了では，プログラムの不具合で強制終了してしまった場合，復帰に数分程度かかることがある．プログラムの作成とは直接関係のないこれらの時間は，プログラムの作成はもちろんのこと細かなパラメータの調整をするにあたって大きな弊害となっている．

上記の問題を解決するために，本稿では AIBO 上と同じスクリプトプログラムの実行を可能とするシミュレータを示す．これによって，上記の手順 2 から 7 すべてをシミュレータ上で行うことが可能となり，開発効率は格段に向上する．

四足ロボットリーグのシミュレータに関しては，これまでに ARAIBO [2] や，ASURA [5] や，GermanTeam[9] や，UChile1 [12] チームのシミュレータが報告されている．

特に GermanTeam は，戦略の記述に XML 形式で振る舞いを記述する XABSL [7]という独自の言語を用いたシュミレータを示している．しかし，XABSL は細やかな動作の記述を行うことができず，シミュレーションの範囲も戦術面の調整にとどまっている．

本稿では，スクリプト言語 Lua を使用したシミュレータを示す．汎用のスクリプト言語 Lua を用いることで，戦略の記述のみならず，細かな動作までもが記述可能である．また，この言語は組み込みが容易であるため，Jolly Pochie フレームワークに組み込むことで，シミュレータ上だけではなく，AIBO 上でも実行できる．

スクリプト言語の組み込みに関しては，MicroPerl の組

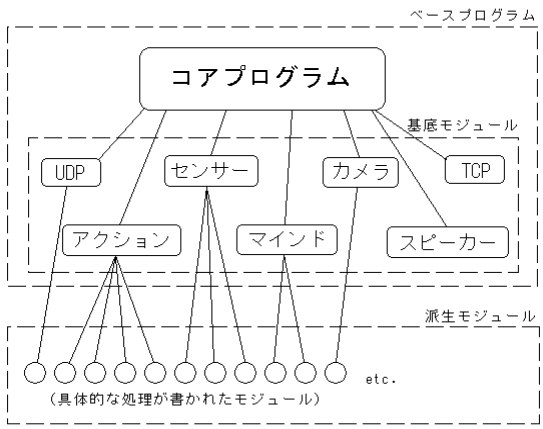


Figure 1: Jolly Pochie のフレームワークの概要.

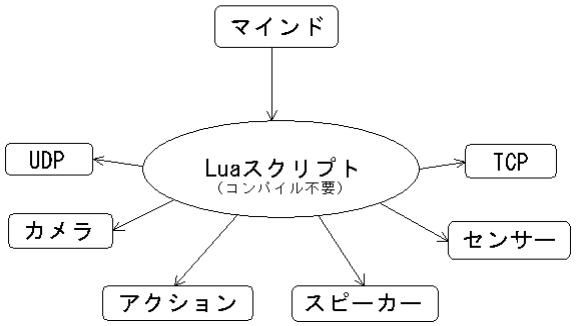


Figure 2: 組み込みの考え方.

み込み [6] が紹介されているが，MicroPerl は十分に大きな言語であり，わずかなメモリしか持たない AIBO には向いていないと考えられる．

本稿の構成は以下のとおりである．まず，第 2 節で Jolly Pochie フレームワークおよび Lua の組み込みについて述べる．次に，第 3 節でシミュレータの設計，実装について述べる．最後に，第 4 節で全体のまとめと OPEN-R TECHNO FORUM 2004 の結果を述べ，今後の課題について述べる．

2 Lua の組み込み

Lua は C 言語への組み込みを前提として開発されたスクリプト言語である．そのため，組み込みが容易であるのはもちろんのこと，実行エンジンが小さく，簡潔で明瞭な文法規則を持つといった特徴がある．Lua が組み込まれるプログラムはホストと呼ばれ，ホストは Lua の関数を呼び出したり，Lua の変数を読み書きしたり，C/C++ の関数を登録したりすることが出来る．通常それらの処理には，Lua が提供する C の API を用いて仮想スタックを操作することが必要だが，本稿ではそれを自動化する Luabind [8] ライブラリを用いてその処理を実現する．

2.1 Jolly Pochie フレームワーク

我々Jolly Pochie チームでは，独自のフレームワークを開発し，様々な機能を個々のモジュールとして作成し，それらを組み合わせてひとつの AIBO プログラムとすることを可能としている [4]．このフレームワークを採用することにより，各プログラマーはそれぞれの担当部分に集中して開発を進めることができる．図 1 に Jolly Pochie のフレームワークの概要を示す．

フレームワークの基本部分は OPEN-R [1] の OObject であり，OPEN-R 特有の型や関数を隠蔽し標準的な C++ のクラスを用いて各モジュールを記述することを可能としている．個々のモジュールが OPEN-R の関数を直接呼び出すことはない．

各モジュールは基底となるモジュールクラスを継承することによって作成され，フレームワークが受け取った OObject としてのイベントを，それを処理すべき各モジュールの呼び出しへと翻訳することによって実行される．

基底となるモジュールには，カメラの画像処理を担当するカメラモジュール，動作時の関節角度を計算するアクションモジュール，各センサーから送られてくる情報を処理するセンサーモジュール，サッカーにおける適切な戦略を作り出すマインドモジュールなどがある．

この中でも特に，マインドモジュールはすべてのモジュールを制御して AIBO を動作させるモジュールであるため，AIBO のプログラムの中核を担っている．したがって，Lua の組み込みは主にマインドモジュールに対して行う．

マインドモジュールは，40ms ごとにコアプログラムから呼ばれる mindNotify という特殊な関数を持っている．この関数の中でサッカーにおける戦略が記述される．例えば，カメラモジュール，センサーモジュールから情報を受け取り，アクションモジュールで AIBO を動かすといった処理を記述する．また，マインドモジュールには状態遷移モデルが導入されているため，mindNotify は状態に対応した関数を呼び出す事になっている．

2.2 実装

Lua を組み込む際の基本的な考え方を図 2 に示す．マインドモジュールが Lua スクリプトを呼び出し，Lua スクリプトが他のモジュールを呼び出すようにすることで，戦略の変更に関しては再コンパイルの必要をなくすことが出来る．これを実現するには，C++ の関数から Lua の関数を呼び出し，その Lua の関数内で C++ の関数を呼び出せるようにすればよい．

[1] SONY が提供する AIBO 用プログラム開発のための API.

Lua の関数の呼び出しは，Luabind ライブラリを用いることで容易に実現することが出来る．例えば，引数，戻り値のない Lua の関数 swingStart を呼び出すためには以下のように書く．JPLua::L は Lua の状態を保持する構造体のポインタである．

```
luabind::call_function<void>(JPLua::L,"swingStart");
```

一方，Lua の関数内で C++ の関数を呼び出すことも Luabind ライブラリを用いることで容易に実現することが出来る．例えば，基本動作を実行するモジュール BasicMotionJPM のメンバ関数 swingHead は，モジュールクラスとそのメンバ関数およびモジュールクラスのインスタンスを登録するだけである．具体的には以下のように書く．

```
module(JPLua::L) [
 class_<BasicMotionJPM>("BasicMotionJPM")
 .def("swingHead",&BasicMotionJPM::swingHead)
];
get_globals(JPLua::L)["basicMotion"] = this;
```

最後の行のインスタンスの登録は，Lua の変数 "basicMotion" に BasicMotionJPM の this ポインタを代入している．登録が済むと，Lua では以下のような記述で関数を呼び出すことが出来る．

```
basicMotion:swingHead(0, 0, 0)
```

先に述べたように Jolly Pochie では，AIBO プログラムは多数のモジュールを組み合わせることによって構成される．その際，これらの Lua への C++関数の登録はモジュールごとに行われるため，モジュールの構成の違いによって C++プログラムや Lua スクリプトを変更する必要はない．例えば BasicMotionJPM モジュールと互換性を持った新たなモジュール AdvancedBasicMotionJPM を作成したとしても，Lua への登録を AdvancedBasicMotionJPM の中で記述するだけで他は全く変更することはない．

2.3 考察

上述の通りマインドモジュールから Lua を通して，自由自在に他のモジュールを呼び出せるようになった．すなわち，今まで数分から数十分を要していたコンパイル時間が不要となり，開発効率が大幅に向上した．さらに，今まで使ってきた C++ の戦略プログラムも，単純に Lua の構文規則にしたがって変換（中括弧をなくして end を付ける等）することで使用することが出来る．

また，スクリプトは AIBO の再起動なしに再読み込みをすることが出来るため，FTP 経由でスクリプトを送信するなどすれば，AIBO の再起動に伴う数十秒から数分の無駄な時間もなくすことが出来る．したがって，一連の開発手順も以下のように簡単なものとなった．

1. PC 上で Lua プログラムを作成する．
2. Lua プログラムを FTP で AIBO に送信する．
3. AIBO に Lua プログラムの再読み込み命令を送る．
4. AIBO を動かしてプログラムの動作確認をする．
5. プログラムを修正して手順 2 へ戻る．

原理的には Lua スクリプトであらゆることを記述することが可能であるが，我々は Lua スクリプトでは，前進する，ボールを見るといった抽象度の高い振る舞いを記述することにした．歩行の関節角の計算や，画像処理のような計算時間がかかるが試行錯誤は行われない処理には C++で記述されたモジュールを用い，抽象度の高い振る舞いだけを Lua で記述することによって，プログラマはサッカーの戦略の作成に集中することが出来る．

同じくスクリプト言語 (MicroPerl) を組み込んだ UPENNALIZERS [3] のソースコードと比較すると，Lua の組み込みの方が遥かに簡単であることが分かる．また，スクリプトを比較しても，Perl は感覚的なコーディングが出来る半面，初心者には分かりづらいソースとなるが，Lua は Pascal に似た文法を持つため初心者にやさしい可読性のあるコードとなっている．

3 シミュレータ

前節では，サッカーの戦略を Lua スクリプトだけで作成できることを示した．しかし，作成した Lua スクリプトのテストには依然として実機によるテストが必要であり，このことが多大な試行錯誤を必要とする開発の妨げとなっている．そこで，本節では，その Lua スクリプトを一切書き換えることなく実行できるシミュレータを示す．シミュレータを用いることによって実機を使わずに開発を行うことができ，これにより開発効率は飛躍的に改善する．

まず，シミュレータを疑似環境，疑似能動的モジュール，疑似受動的モジュールの 3 つに分けて考える．図 3 に，現実の環境とシミュレータ環境の対応関係を示した．

3.1 疑似環境

疑似環境は，フィールドや AIBO やボールといった実世界の環境をシミュレーションする部分である．疑似環境では，それぞれの物体を三次元モデルで表現することによって，例えば鼻先に搭載された AIBO のカメラからボールが見えているかどうかを視覚的にも確認できるようにしている．これを VPython [11]という Python の三次元グラフィックスモジュールを用い実現した．

疑似能動的モジュールは，Lua スクリプトを呼び出す Jolly Pochie のフレームワークに属するモジュールをシミュレーションする部分である．具体的にはマインドモジュール，UDP 通信モジュール，アクションモジュールなどをシミュレーションするプログラムからなる．区別するため，これらのモジュールを能動的モジュールと呼び，

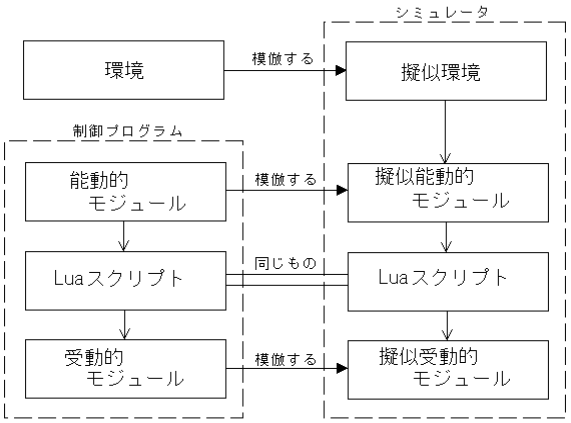


Figure 3: 現実の環境とシミュレータ環境の対応関係.

他のモジュールを受動的モジュールと呼ぶことにする．能動的モジュールが受動的モジュールと違うのは、一定時間ごとに，あるいはイベントごとに呼ばれる関数を持っている点である．

マインドモジュールは 40ms ごとに Lua スクリプトを呼び，アクションモジュールは 8ms ごとに関節の角度を指定する．また，UDP 通信モジュールはメッセージを受信した時に，Lua スクリプトのある関数を呼ぶことになっている．疑似能動的モジュールは、C++で実装される．

疑似受動的モジュールは，Lua スクリプトから呼び出されるような，受動的モジュールをシミュレーションする部分である．例えば，歩行やシュートといった AIBO に動作をさせるものや，画像認識の結果からボールの位置を計算するものがこれにあたる．本来なら受動的モジュールの関数は Luabind で登録されて Lua スクリプトから呼び出されるが，シミュレータでは画像処理や自己位置同定などの計算をする必要がないため，単に Lua 側でダミー関数を定義することにする．ただし，シミュレータにも使える OPEN-R に依存しないライブラリは，再利用のために登録することもできる．例えば，3つの首関節の角度とカメラの XYZ 座標の相互変換を行う HeadKinema ライブラリなどがこれにあたる．

3.2 疑似環境の実装

VPython を用いて，三次元空間上にサッカーフィールド，ボール，AIBO を表示して疑似環境を作る．サッカーフィールドとボールに関しては，VPython の基本的なオブジェクト（球，円柱，箱など）を組み合わせて簡単に表現出来る（図 4, 5）．

一方，AIBO は複雑で円みを帯びた形状をしているため，基本的なオブジェクトを組み合わせても正確に表現することは難しい．よって，AIBO のモデルは簡略化して胴部と頭部だけの簡単なものとした（図 6）．実際，耳や尻

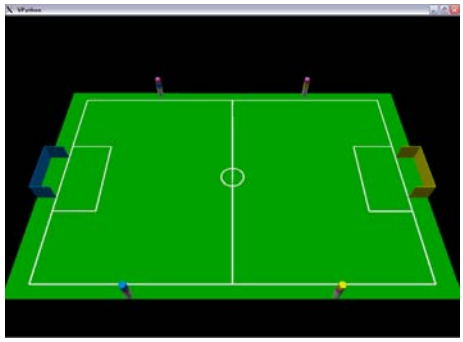


Figure 4: フィールドのモデル.

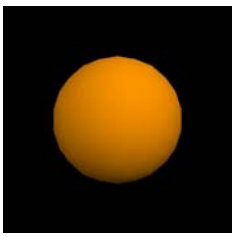


Figure 5: ボールのモデル.

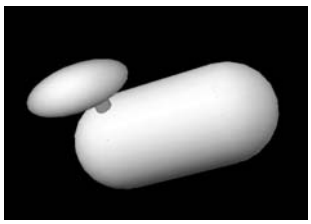


Figure 6: AIBO のモデル.

尾はシミュレーションする必要はなく，歩行に関しても天井カメラから計測した一歩の歩幅が設定ファイルとして保存してあるので，足の動き自体をシミュレーションする必要はない．頭部に関しては，視覚的に AIBO からボールが見えているかを判断するために，AIBO の仕様書を元に首関節の長さや接合部の位置まで正確に実装した．

これらのモデルの表示ルーチンは，マインドモジュールに同期させることにする．つまり，表示ルーチンは 40ms ごとに呼ばれ，同時にマインドモジュールも呼ばれることになる．

3.3 疑似能動的モジュールの実装

本稿で実装した能動的モジュールは，マインドモジュール，アクションモジュール，UDP 通信モジュールの 3 つである．これらのモジュールは C++ で実装され疑似環境から呼ばれる．Python から C++ を呼び出すためには，ラッパ関数を自動生成する SWIG(Simplified Wrapper and Interface Generator) [10] というツールを使った．

疑似マインドモジュールは，Lua の組み込みでの実装と同様に Lua スクリプトの関数を呼び出すようにすればよい．40ms ごとに mindNotify 関数を呼び出す処理は，疑似環境が提供する．

疑似アクションモジュールは，8ms ごとに呼び出すことはせず，処理を単純にするために疑似マインドモジュールに同期させて 40ms ごとに呼び出すことにする．首の動作に関しては，1 フレーム（8ms 区間）で動かせる制限角度が決まっているので，疑似アクションモジュールでは一度呼ばれるごとに 5 フレーム分の角度を送り込むこ

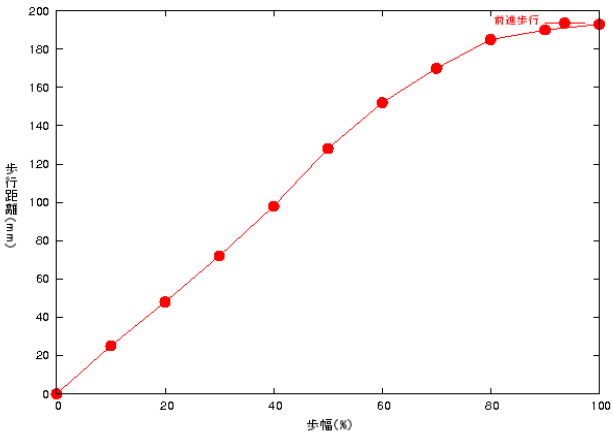


Figure 7: 前方歩行の歩幅と歩行距離のグラフ.

とにする．また，歩行動作に関しては 12 個の関節角度を計算する必要はなく，一歩の歩行距離が保存されたファイルさえあれば直接 AIBO を動かすことができる．

歩行距離は天井カメラから得た情報を元に計測する．天井カメラによる距離の計測方法については，青山ら [13] を参照されたい．歩幅に対する歩行距離は，床との摩擦や，モータの性質から線形的に変化することはない．したがって，まず歩幅を 10 段階に変化させて歩行距離の計測値を出し，この計測値から線形補完したグラフを元に歩行距離の近似値を出す．図 7 に，前方歩行の歩行距離を実際に線形補完したグラフを示す．

また，疑似アクションモジュールは 5 フレームごとに AIBO を移動させるため，1 歩にかかるフレーム数が必要となる．これは，既存の歩行パラメータから抽出することにする．シミュレータ上で一度に移動する距離は，1 歩の歩行距離を 1 歩のフレーム数で割り，それを 5 倍したものとなる．最後の端数フレームに関しては，1 フレームの歩行距離を端数倍したものを送る．

疑似 UDP 通信モジュールは単なる UDP サーバーであるが，同一ポートでは 2 つの UDP サーバーを立ち上げることができない．したがって UDP サーバーの実体は 1 つとして，受け取った情報を複数の疑似 AIBO プログラムに渡す仕組みを作った．したがって，実際の試合と同じようにゲームコントーラの情報を受け取り試合を始めることも可能である．

3.4 疑似受動的モジュールの実装

疑似受動的モジュールは，そのほとんどはダミー関数を作るだけで事足りる．なぜなら，各種センサー用のモジュールや音声出力モジュールなどのモジュールはシミュレータでは実装しないからである．よって，これらのモジュールの関数については C++ で新たに実装し登録するようなことはせず，単にスクリプトがエラーを出さないように

```
soundPlayer = {}

function soundPlayer:playSoundOnce()
end

function soundPlayer:playSoundStop()
end
```

Figure 8: ダミー関数の一部.

Figure 9: シミュレーターの実行の様子.

Lua でダミー関数を定義する（図 8）．一方，シミュレータ用に新たに実装しなければならないモジュールとして，ボールの位置を計算するモジュールや，自己位置を計算するモジュールなどが挙げられる．これらに関しては，シミュレータが常にすべての物体（AIBO，ボール，ゴールなど）のグローバル座標を保持しているため，複雑な画像処理をしなくても取得することが出来る．ただし，AIBO の鼻先のカメラからボールまでの相対距離や角度を出すためには，複雑な幾何計算が必要となるため，その計算をするために作られた C++ の自作ライブラリ HeadKinema を Lua に登録することにした．

3.5 考察

図 9 にシミュレータの実行の様子を示す．シミュレータ内の AIBO の動作は理想的な環境下でのものなので，実世界が理想的な環境に近ければ実世界の AIBO とほぼ同じ動作をする．

実機で動くスクリプトをシミュレータ上で動かせるようにしたことで，実機を使用せずにスクリプトの開発，動作確認が行えるようになった．したがって，シミュレータ上での作業手順は以下のような更に簡単なものとなる．ただし，ある程度シミュレータ上で動作確認が出来たら，最後に実機で動作確認や微調整をする必要がある．

1. PC 上で Lua スクリプトを作成する．
2. PC 上のシミュレータで動作の確認をする．
3. プログラムを修正して手順 2 へ戻る．

シミュレータの最大の利点は実機を使わずにプログラムの開発が出来ることである．これにより，実世界における複雑さに惑わされることなく開発作業に集中する事が出来る．特に複数の AIBO を使って動作確認を行いたい場合には有効である．また，今まではすべてのモジュールが揃わなければ，サッカーの戦略部分のプログラムを開発することが出来なかったが，シミュレータではダミー関数を作って簡単に動作確認まですることが出来る．これは AIBO の制御プログラムのように，大規模なプログラムを複数人で開発する際には大きな効果をもたらす．

シミュレータの欠点は，実世界を完全にシミュレートできないことである．例えば，障害物と衝突した場合や，ボールが完全に見えていない場合の処理をシミュレートするのは非常に難しい．実世界での誤差を計測してシミュレータに採り入れることも考えられるが，最終的には実機での動作確認をすることが必要となるだろう．

4 おわりに

本稿では，まずスクリプト言語 Lua をプログラムに組み込むことで，プログラム作成とは直接関係のないコンパイル時間を不要にした．さらに，AIBO 上と全く同じスクリプトプログラムの実行を可能とするシミュレータを示した．このことで，実機を使わずに動作確認が出来るようになり，プログラムの開発効率が大幅に向上した．

また，我々は本稿の技術を生かして，SONY が主催する OPEN-R TECHNO FORUM 2004 in Japan (2004/12/4) の競技種目に出場した．結果，シュート技術を競う PK 競技，歩行技術を競うレース競技において優勝することが出来た．

今後の課題としては，シュートにより移動するボールの方向や距離，視覚によるボールの位置情報などに誤差を加えることが挙げられる．そうすることで，シミュレータの環境が実世界に近付き，より実世界に適応するロバストな戦略の作成が出来るようになる．

参考文献

[1] AIBO SDE Homepage. `http://openr.aibo.com/openr/jpn/index.php4`.

[2] K. Asanuma, K. Umeda, R. Ueda, and T. Arai. Development of a Simulator of Environment and Measurement for Autonomous Mobile Robots Considering Camera Characteristics. In *Robot Soccer World Cup VII*, pp. 446–457, 2004.

[3] D. Cohen, Y. H. Ooi, P. Vernaza, and D. D. Lee. The University of Pennsylvania Robocup 2003 Legged Soccer Team. Technical report, UPENNALIZERS, 2003.

[4] J. Inoue, H. Aoyama, A. Ishino, and A. Shinohara. Jolly Pochie 2004 in the Four Legged Robot League. Technical report, Jolly Pochie, 2004.

[5] T. Ishimura, T. Kato, K. Oda, and T. Ohashi. An Open Robot Simulator Environment. In *RoboCup 2003*, Vol. 3020 of *LNAI*, pp. 621–627. Springer, 2004.

[6] T. Jenness and S. Cozens. Extending and Embedding Perl. *Linux Journal*, 2003:15, July 2003.

[7] F. Lötzsch, J. Bach, H.-D. Burkhard, and M. Jüngel. Designing Agent Behavior with the Extensible Agent Behavior Specification Language XABSL. In *7th International Workshop on RoboCup 2003*, LNAI. Springer, 2004.

[8] Luabind. `http://luabind.sourceforge.net/`.

[9] T. Röfer. German team robocup 2004 technical report. Technical report, German Team, 2004.

[10] Simplified Wrapper and Interface Generator. `http://www.swig.org/`.

[11] VPython. `http://vpython.org/`.

[12] J. C. Zagal and J. R. del Solar. UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League. In *8th International Workshop on RoboCup 2004*, LNAI. Springer, 2004.

[13] 青山, 井上, 坂井, 石野, 篠原. 遺伝的アルゴリズムによる四足歩行ロボットの高速歩行の実現．火の国情報シンポジウム論文集, March 2005.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究資料
JSAI Technical Report
SIG-Challenge-0521-5 (5/24)

統合シミュレーションのためのデータのバージョン管理の形式的考察

Consistency Management Framework for Database used in Integrated Simulations

野田五十樹
NODA, Itsuki
(独) 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
i.noda@aist.go.jp

Abstract

In this article, I show a mathematical formalization of *version control* that manage consistency among data that are used in integrated simulation system. When we apply integrated simulations to real-time evaluation of complex social phenomena like disaster and rescue, keeping consistency of data is an important issue for validate result, because multiple and delayed information are reported to the database continuously in such applications. The proposed formalization gives fundamental background of consistency and validity of database and simulation. I also investigate and discuss about cost of the management in several implementation style.

1 はじめに

災害など複雑な現象を計算機で扱う手法として、統合シミュレーションの構築が進められている[2, 1]。これら統合シミュレーションの特徴は、関連ある事象の変化を各々予測するシミュレータと結合し、相互に計算結果を反映しながら全体としての現象をシミュレーションする点である。

このようなシミュレーションを具体的な問題に提供する場合、誤差を含む膨大な量のデータを扱い、それらの間の依存関係を適切に管理する必要がある。また、シミュレーションを現場で利用、すなわち災害救助の例では災害が実際に発生した直後に被害想定や救助計画立案のためにシミュレーションを用いる場合、災害現場のセンサーデータの報告に遅延が生じることを考慮する必要がある。このような条件をふまえた上で、シミュレーションに対して整合性のある初期データを提供する枠組みが必要となる。本報告では、このようなデータの整合性を形式的に定義し、統合シミュレーションへの入出力を担うデータベースに求められる機能を整理することを目的とする。

2 要件

以下では、災害救助シミュレーションを例題として、統合シミュレーションの際のデータ管理に求められる要件を、いくつかのケースの形であげておく。

2.1 ケース 1:センシングにおけるノイズ

現場におけるシミュレーションの利用を考える場合、センサーから得られるデータがかならずしも真ではないことを前提としなければならない。特に同じ事象を観測する複数のセンサーがあった場合、それらが異なるデータを報告することは普通に起こり得る。例えば、先の中越地震では、気象庁の web サイトにおいて、同一地点の同一時刻の震度として 5 と 7 が繰り返し報告されていた。このような場合、震度 5 でシミュレーションすべきか、震度 7 を採用すべきか、あるいは間をとって 6 をとるのか、さらには 6 ± 1 といった幅や分布を伴ったデータをシミュレーションの入力として採用するのか、管理する方法が必要となる。

2.2 ケース 2:報告の遅延

広域に広がる対象を扱う場合、実際の対象の状態を計測するセンサーはデータベースから遠いところに存在する。よって、そのデータは必ずしもリアルタイムで入ってくるとは限らず、場合によってはシミュレーション開始後に入ってくる場合もある。例えば中越地震においても、通信や電力のトラブルのために山古志村のデータが 2 日間も遅延して報告された事は記憶に新しい。

一方、現場で利用されるシミュレーションではそのような遅延のある報告をいちいち待っていられない事が多い。よって、それら報告の遅延も厳密に管理し、シミュレーションに利用されたデータとの整合性をチェックできる機構が必要になる。

2.3 ケース 3:シミュレーションの依存関係

複数のシミュレーション結果が別のシミュレーションへの入力となる場合、元となるデータの整合性が重要となってくる。例えば、建物倒壊シミュレーションを震度 7、道路閉塞シミュレーションを震度 5 を前提として行い、それらを合わせて避難シミュレーションを行うことは、整合性があるとは言いにくい。[1] よって、シミュレーション結果をあわせる場合には、それらのシミュレーションの依存関係をたどって整合性をチェックできる枠組みが必要となる。

実際、災害救助シミュレーション[2, 1] では、10 以上のサブシミュレータが相互に依存して計算を行う。また、これらが扱うデータは膨大であるため、整合性チェックは軽量でスケーラブルな枠組みである必要がある。

2.4 要件のまとめ

上記の要件をまとめると、統合シミュレーションのためのデータベースに求められる機能は次のようになる。

- ある事象の観測あるいは推定として、複数あり得る値のどれをシミュレーションの初期値として用いたかを管理する手法。
- 複数のシミュレーション結果を統合する際に、そのデータの整合性をチェックできる機能。
- 遅延して報告された値に対し、報告以前に行われたシミュレーションとの整合性をチェックする機能。

3 形式的定義と定理

以上の要件や機能を形式的に議論するため、いくつかの概念と定理を整理する。

3.1 Database と Snapshot

ここでは**データベース**を、**データ** の集まりとみなす。また **データ** とはある時刻のある **事物** について観測あるいは推定された個々の値とする。データベースは動的であるとする。すなわち、データベースへはシミュレーションなどとは非同期にデータが書き加えられるものとする。

以下に形式的定義を述べる。

事物 ：**事物** θ は、個々の物体やその属性、現象などの同定子である。例えば、“建物 X”、“車 Y の速度”、“地点 Z の震度” などは事物である。事物同士には包含関係や上位下位関係があっても良い。

事象 ：**事象** e は、事物 θ と時刻 t の対である。よって、事象 e は $\langle\theta, t\rangle$ とも記述される。例えば、“時刻 t における人 X の負傷度” “時刻 t における車 Y の速度” “時刻 t における道 Z の閉塞度” は事物 である。

[1] このようなシミュレーションを行った場合、(震度 7 で) 建物倒壊で多くの被災者が出ているところに、(震度 5 で) あまり閉塞していない道路を使って救助隊が迅速に対応してしまうシミュレーションが怒ってしまうことが考えられる。

データ ：**データ** $d_{\theta,t,s}$ は、**センサー**あるいは**シミュレーション** s によって観測・推定された事象 $\langle\theta, t\rangle$ の値である。観測や推定にはノイズや前提条件の違いがあるので、同じ事象に対し異なるデータがあり得る。s はセンサーの同定子あるいはシミュレーションのセッションに対応する **バージョン** (次節参照) に対応する。

また、データがデータベースに報告された時刻を明記する場合は、$d_{\theta,t,s;\tau}$ ような記述を用いる。ただし、τ はデータベースにこのデータが報告された時刻である。

データベース ：データベース $\mathbf{DB} = \{d_{\theta,t,s;\tau}\}$ は (将来に渡って報告されるものを含む) 全データの集合である。

また、$\mathbf{DB}_{(0,\mathcal{T})} = \{d_{\theta,t,s;\tau} \in \mathbf{DB} | \tau < \mathcal{T}\}$ は、ある時刻 $\mathcal{T}$ におけるデータベースのスナップショット、すなわち、ある時刻 $\mathcal{T}$ までに報告されたデータの集合を表す。

3.2 バージョン

シミュレーション とは、データベース **DB** と以下のように入出力を行うものとして定義する。

1. ある事象の集合 $\boldsymbol{E}_{\text{ground}} = \{e_i | i\}$ に関するデータの集合 $\boldsymbol{D}_{\text{ground}}$ を、ある時刻 $\mathcal{T}_{\text{begin}}$ (一般にシミュレーション開始時刻) にデータベース **DB** から取得する。
2. 別の事象の集合 $\boldsymbol{E}_{\text{target}} = \{e_i | i\}$ についてのデータ $\boldsymbol{D}_{\text{target}}$ を推定する。
3. 推定されたデータ $\boldsymbol{D}_{\text{target}}$ を時刻 $\mathcal{T}_{\text{end}}$ (一般にシミュレーションの完了時刻) にデータベース **DB** に書き込む。

このシミュレーションの入出力に用いられるデータの集合 $(\boldsymbol{D}_{\text{ground}}, \boldsymbol{D}_{\text{target}})$ につけられた名前を **バージョン**と呼ぶ。よって、シミュレーションはあるバージョンのデータ集合をデータベースから受け取り、別のバージョンのデータ集合を作成してデータベースへ登録する処理とみなされる。

前節で述べているように、統合シミュレーションではいくつかのシミュレーション結果を合わせて別のシミュレーションの入力として用いることが生じる。この際に、それらのシミュレーション結果が合わせて用いてもよいものかどうかを区別する必要がある。ここではこの区別を行うために、バージョン間の**整合性**というものを定義する。

バージョン ：バージョン v は $\langle \boldsymbol{E}, \mathcal{T}, \boldsymbol{D}, \boldsymbol{G} \rangle$ という対で定義される。ここで、 $\boldsymbol{E} = \{e_i | i\}$ は事象の集合、$\boldsymbol{D}$ はそのバージョンに属するデータ集合、$\mathcal{T}$ はデータベース **DB** を操作 (入出力) した時刻、$\boldsymbol{G} = \{u_i | i\}$ は $\boldsymbol{D}$ が直接依存するバージョンの集合を表す。これらの対の要素は以下の条件を満たさなければならない。

$$\boldsymbol{D} \subset \mathbf{DB}_{(0,\mathcal{T})}/E = \{d_{\theta,t,s;\tau} | \langle\theta, t\rangle \in \boldsymbol{E}, \tau < \mathcal{T}\}$$

バージョンの生成 ： 新しいバージョンが生成されるには、**発生**、**抽出**、**算出**、**合同**の 4 種類の方法がある。

DB からセンサーにより直接観測されたデータのみを集めて新しいバージョン v をつくる場合、そのバージョンを **発生** するという。発生させられたバージョンは $v = \langle \boldsymbol{E}, \mathcal{T}, \boldsymbol{D}, \phi \rangle$ のような対で表される。ここで、$\boldsymbol{E}$ は関連する事象、$\boldsymbol{D}$ はセンサーから直接得られたデータからなる集合、$\mathcal{T}$ は $\boldsymbol{D}$ を **DB** から集めた時刻である。

既存のバージョン $u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle$ からいくつかのデータを取捨選択してバージョン v を作成する場合、そのバージョンを**抽出**するという。抽出された v は $\langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \{u\} \rangle$ という対で表される。ただし、$\mathcal{T}_v$ は抽出の時刻であり、また、$\boldsymbol{D}_v \subseteq \boldsymbol{D}_u$ および $\boldsymbol{E}_v = \{e | d_{\theta,t,s} \in \boldsymbol{D}_v, e = \langle \theta, t \rangle\}$ が満たされるものとする。

あるバージョン u を入力としてシミュレーションした結果をバージョン v として得る場合、そのバージョンを **算出** するといい、$u \triangleright v$ と記述する。算出されたバージョンは以下の対で表される。

$$u \triangleright v \Leftrightarrow v = \langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \{u\} \rangle$$

ここで $\boldsymbol{E}_v$、$\boldsymbol{D}_v$、$\mathcal{T}$ は各々シミュレーションの対象事象、結果、完了時刻である[2] 。

2 つのバージョン u、v の **合同** $(u \oplus v)$ とは、2 つのバージョンに含まれるデータを合わせたものを表すバージョンである。合同バージョンは仮想的なバージョンであり、事象集合やデータ集合は空集合として扱われる。すなわち、u と v の合同バージョンは以下の対で表される。

$$u \oplus v = \langle \phi, \mathcal{T}, \phi, \{u, v\} \rangle$$

ただし、$\mathcal{T}$ は合同の操作を行った時刻である。

バージョン v がバージョン u を**直接根拠**とする $(u \succ v$ と記述) とは、v が u から合同あるいは算出されている場合をさす。上で定義されているように、v とその直接根拠 u には $u \succ v \Leftrightarrow u \in \boldsymbol{G}_v$ という関係が成り立つ。ただし、$v = \langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \boldsymbol{G}_v \rangle$ である。

バージョン u がバージョン v の**根拠**である $(u \succ^* v)$ とは、u と v の間に再帰的に直接根拠の関係のパスがあることをさす。すなわち、

$$u \succ^* v \Leftrightarrow (u = v \text{ もしくは } u \succ u', u' \succ^* v)$$

である。また、バージョン v の根拠の集合を v の**軌跡** $(\mathbf{Tr}(v) = \{u | u \succ^* v\})$ と呼ぶ。

整合性 ：2 つのバージョン u, v が *1* **次的に無矛盾** $(u \sim v)$ であるとは、以下の条件を満たすときである。

$$\forall e \in \boldsymbol{E}_u \cap \boldsymbol{E}_v : \boldsymbol{D}_u/e = \boldsymbol{D}_v/e$$

[2] ここでは、シミュレーションの結果はすぐさまデータベースに記録されると仮定している。

ただし、$u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle$、$v = \langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \boldsymbol{G}_v \rangle$ である。

2 つのバージョン u, v が **無矛盾** $(u \sim v)$ であるとは、以下の条件を満たすときである。

$$\forall u' \succ^* u, \forall v' \succ^* v : u' \dot{\sim} v'$$

2 つのバージョンの合同 $u \oplus v$ をつくる場合、u と v は無矛盾である必要がある。

また、$v \sim v$ が成り立つ場合、バージョン v が **自己無矛盾** であるという。

3.3 世界線

データの定義で述べたように、ある事象に対して複数のデータがデータベースに登録されていることがあり得る。一方、事象の解析やシミュレーションでは、それらのデータを全て真として扱うとは限らず、目的に応じて取捨選択された一部のデータを真して解析などの入力とする。ここでは、一連のシミュレーションにおいて真として扱われたデータの集合を**世界線**と呼ぶ。

もちろんシミュレーションでは少しずつ異なる初期値を入力として解析を行うことがある得るので、世界線も複数、並行して存在し得る。このような場合、異なる初期値を用いていることから、それらの世界線は相互に矛盾していることになる。一方、一つの世界線の中では、それに含まれる全てのデータは相互に無矛盾である必要がある。ここで無矛盾とは「各々のデータが相互に矛盾しない条件あるいはデータに基づいて求められたものである」という状態をさす。以下ではこれらの概念を形式的に定義する。

世界線 ： **世界線** $\boldsymbol{w}$ はデータベース **DB** の任意の部分集合である。すなわち $\boldsymbol{w} \subseteq \mathbf{DB}$ である。世界線 $\boldsymbol{w}$ が同じ事象 e に対し複数のデータを含んでいる場合はモンテカルロ的に解釈され、その複数のデータは e の値の分布に準じていると見なす。一方、世界線がある事象についてデータを含んでいないときは、その事象については考慮しない ('don't-care') ものとして扱う。

データベースから世界線を選びだす時に制約は存在しない。よって、世界線の全集合はデータベースの巾集合 $_2\mathbf{DB}$ である。

世界線はデータベースの時間変化とは独立である。すなわち、データベースに新たなデータが登録されても、世界線は変化しないものとする。

世界線の射影 ：世界線 $\boldsymbol{w}$ のうち、ある時刻 t に関する事象のデータのみを集めたものを時刻 t における世界線 $\boldsymbol{w}$ の**射影**、あるいは**タイムスライス**と呼び、$\boldsymbol{w}/t$ と表す。すなわち、$\boldsymbol{w}/t = \{d_{\theta,t',s;\tau} \in \boldsymbol{w} | t' = t\}$ である。

また世界線 $\boldsymbol{w}$ のうち、ある事物 θ に関するデータのみを集めたものを事物 θ における世界線 $\boldsymbol{w}$ の**射影** と呼び、

$\boldsymbol{w}/\theta$ で表す。すなわち、$\boldsymbol{w}/\theta = \{d_{\theta',t,s;\tau} \in \boldsymbol{w} | \theta' = \theta\}$ である。

同様に世界線 $\boldsymbol{w}$ のうち、ある事象 e に関するデータのみを集めたものを事象 e における世界線 $\boldsymbol{w}$ の**射影** と呼び、$\boldsymbol{w}/e$ で表す。すなわち、$\boldsymbol{w}/e = \{d_{e',s;\tau} \in \boldsymbol{w} | e' = e\}$ である。

世界線の無矛盾性 ：以下の条件を満たした場合、またそのときに限り世界線 $\boldsymbol{w}$ がバージョン v を**支持**すると呼び、$\boldsymbol{w} \vdash v$ と記述する。

$$\forall u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u \rangle \not\succ^* v, \forall e \in \boldsymbol{E}_u : \boldsymbol{w}/e = \boldsymbol{D}_u/e$$

また、以下の条件を満たした場合、またそのときに限り、世界線 $\boldsymbol{w}$ が **無矛盾**であると呼ぶ。

$$\forall d_{\theta,t,s;\tau} \in \boldsymbol{w} : s \text{ is a sensor id or } \boldsymbol{w} \vdash s$$

言い換えれば、世界線を構成するデータのうちシミュレーションにより求められたものは、それが属するバージョンの根拠が全て世界線に含まれている場合に限り、世界線は無矛盾となる。

一般にシミュレーションは世界線を伸ばしていく作業と言える。ここで重要なのは、世界線が無矛盾性を維持するようにシミュレーションを行わなくてはならないことである。もし世界線が無矛盾に保てないとすると、そのシミュレーションは異なる相互に相容れない根拠に基づいて計算を行っていることになり、全体としては意味のない計算を行うことになる。

3.4 無矛盾性に関する定理

ここでは、世界線およびバージョンの無矛盾性の維持に関していくつかの定理を与える。

補題 1. バージョン v が自己無矛盾であり、かつ、世界線 $\boldsymbol{w}$ が v の軌跡に属する全てのデータから構成されている場合、v の任意の根拠 u は世界線 $\boldsymbol{w}$ に支持される。すなわち、以下の式が成り立つ。

$$\begin{aligned} \forall v \quad : \quad & v \textit{ is self-consistent}, \\ & \boldsymbol{w} = \{d | u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle \not\succ^* v, d \in \boldsymbol{D}_u\} \\ \rightarrow \quad & \forall u \not\succ^* v : \boldsymbol{w} \vdash u \end{aligned}$$

証明 *(補題 1).* バージョン v の根拠の 1 つ u が世界線 $\boldsymbol{w}$ に支持されていないとする。すなわち、

$$\exists u = \{\boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u\} \not\succ^* v, \exists e \in \boldsymbol{E} : \boldsymbol{w}/e \neq \boldsymbol{D}_u/e.$$

とする。定義より、$\boldsymbol{w}$ は $\boldsymbol{D}_u$ の上位集合である。よって、$\exists d \in \boldsymbol{w}/e, d \notin \boldsymbol{D}_u/e$ が満たされる限り $\boldsymbol{w}/e \neq \boldsymbol{D}_u/e$ が成り立つ。これは、

$$\exists u' \{\boldsymbol{E}_{u'}, \mathcal{T}_{u'}, \boldsymbol{D}_{u'}, \boldsymbol{G}_{u'}\} \in \mathbf{Tr}(v) \quad : \quad u' \neq u, \quad d \in \boldsymbol{D}_{u'}$$

である。しかしこれはバージョン v の自己無矛盾性の定義に反する。

よって、根拠 u は $\boldsymbol{w}$ に支持される。 □

この補題より、次の定理が導き出される。

定理 1. バージョン v が自己無矛盾である場合、それを支持する無矛盾な世界線 $\boldsymbol{w}$ が存在する。

$$\forall v : v \textit{ is self-consistent} \rightarrow \exists \boldsymbol{w} : \boldsymbol{w} \textit{ is consistent}, \boldsymbol{w} \vdash v$$

証明 *(定理 1).* 次のような世界線 $\boldsymbol{w}$ を考える。

$$\boldsymbol{w} = \{d | u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle \not\succ^* v, d \in \boldsymbol{D}_u\}$$

補題 1 より世界線 $\boldsymbol{w}$ はバージョン v の任意の根拠を支持する。一方、任意のシミュレーションされたデータ $d_{\theta,t,s;\tau} \in \boldsymbol{w}$ に対して、[3] そのデータが属するバージョン s は v の軌跡に含まれる。よって、s は世界線 $\boldsymbol{w}$ に支持される。

よって、無矛盾であり、かつ、v を支持する世界線 w は存在する。 □

この定理により、シミュレーションが意味あるものとする、すなわち、シミュレーションの結果が無矛盾な世界線の一部となるためには、そのシミュレーションが出力するデータのバージョンの自己無矛盾性にのみ配慮すれば良いことになる。

この定理を元に、以下ではバージョンの生成に関する定理を導き出す。

定理 2. 発生させられたバージョンは自己無矛盾であり、それを支持する無矛盾な世界線が存在する。

証明 *(定理 2).* 定義により、発生させられたバージョンに含まれるデータは全てセンサーから直接きたものであり、他のバージョンとの依存関係はない。よって、自己無矛盾である。

よって、定理 1 より、発生バージョンを支持する無矛盾な世界線が存在する。 □

定理 3. 自己無矛盾なバージョン u からバージョン v を抽出する場合、以下の条件を満たす場合に限り、v は自己無矛盾であり、それを支持する無矛盾な世界線が存在する。

$$\begin{aligned} & \forall v : u \supset v, \\ & \quad v \textit{ is self-consistent} \leftrightarrow \forall e \in \boldsymbol{E}_v : \boldsymbol{D}_u/e = \boldsymbol{D}_v/e \end{aligned}$$

ただし、u、v は各々$\langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle$ および $\langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \boldsymbol{G}_v \rangle$ である。

[3] センサーから得られたデータは根拠を持たないので、無矛盾性には抵触しないため、考慮しない。

証明 *(定理 3).* もし上記の条件が満たされている場合、u and v は無矛盾である。また、u は自己無矛盾であることから、v もまた自己無矛盾である。

逆に、条件 $\forall e \in \boldsymbol{E}_v : \boldsymbol{D}_u/e = \boldsymbol{D}_v/e$ が満たされない場合、v に含まれるある事象 e が存在して、かつ、$\boldsymbol{D}_u/e \neq \boldsymbol{D}_v/e$ を満たすものが存在する。よって、u と v は矛盾することになり、v は自己無矛盾でなくなる

よって、v の自己無矛盾性と上記の条件は等価であり、定理 1 より、抽出バージョンを支持する無矛盾な世界線が存在する。 □

定理 4. バージョン u が自己無矛盾である場合、それから算出されたバージョン v は自己無矛盾であり、v を支持する無矛盾な世界線が存在する。すなわち、

$$\forall u : \textit{self-consistent} \rightarrow$$
$$\forall v : u \rhd v \quad \rightarrow \quad v \textit{ is self-consistent},$$
$$\exists \boldsymbol{w} : \boldsymbol{w} \textit{ is consistent}, \boldsymbol{w} \vdash v$$

ただし、ここではシミュレーションは入力されたものと同じ事象に関するデータを出力しないものとする。

証明 *(定理 4).* 定義により、$\mathbf{Tr}(v)$ is $\{v\} + \mathbf{Tr}(u)$ が成り立つ。また、$\boldsymbol{E}_u \cap \boldsymbol{E}_v = \phi$ であることから、u の任意の根拠は v と無矛盾である。よって、バージョン v は自己無矛盾である。

よって、定理 1 から、抽出バージョンを支持する無矛盾な世界線が存在する。 □

定理 5. *2* つのバージョン v、u が自己無矛盾でありかつ相互に無矛盾であるとする。この場合、この *2* つのバージョンの合同バージョンは自己無矛盾であり、それを支持する無矛盾な世界線が存在する。すなわち、

$$\forall u, v : \textit{self-consistent}, u \sim v \rightarrow$$
$$u \oplus v \textit{ is self-consistent},$$
$$\exists \boldsymbol{w} : \boldsymbol{w} \textit{ is consistent}, \boldsymbol{w} \vdash u \oplus v$$

証明 *(定理 5).* 定義より、

$$\mathbf{Tr}(u \oplus v) = \{u \oplus v\} + \mathbf{Tr}(u) + \mathbf{Tr}(v)$$

である。x と y を $\mathbf{Tr}(u \oplus v)$ に含まれる 2 つのバージョンとする。もし x と y が共に $\mathbf{Tr}(u)$ に含まれている場合、u が自己無矛盾であるため、x と y は無矛盾である。x と y が共に $\mathbf{Tr}(v)$ に含まれている場合も同様である。

$x \in \mathbf{Tr}(u)$ 、$y \in \mathbf{Tr}(v)$ かつ、$x \notin \mathbf{Tr}(v)$ 、$y \notin \mathbf{Tr}(u)$ である場合、u と v は相互に無矛盾であるので、x と y は相互に無矛盾である。

x と y のいずれかが $u \oplus v$ と一致する場合、$u \oplus v$ が事象を含まないことから、x と y は 1 次的似無矛盾である。

よって、合同バージョン $u \oplus v$ は自己無矛盾であり、定理 1 よりそれを支持する無矛盾な世界線が存在する。 □

4 バージョン管理のコスト

前節で議論した無矛盾性を保ちつつ統合シミュレーションを行うためには、データベースに以下のような機能を追加する必要がある。

- バージョンに関する情報 $\langle \boldsymbol{E}, \mathcal{T}, \boldsymbol{D}, \boldsymbol{G} \rangle$ を記録する機能。特に、$\boldsymbol{D}$ は多数に上ることを前提に扱われることが必要。
- 既存のバージョンの無矛盾性を保ちつつ、新しいデータを登録する機能。特に、シミュレーション開始後にそのシミュレーションに関係するデータが挿入された場合、関係するバージョンの無矛盾性を管理する方法が必要。
- 2 つのバージョンの無矛盾性をチェックする機能。前節の定理にあるように、バージョンの合同を行う場合、その無矛盾性を高速にチェックする機能が必要。

以下ではこれらの機能を実装する方法として、$\boldsymbol{D}$ の記録方法に着目し、**正記法** と **負記法** の 2 つの方法とその計算コストについて考察する。

4.1 正記法

あるバージョンに関する $\boldsymbol{D}$ の最も単純な記録方法は、それに含まれるデータを列挙する方法である。これを **正記法** と呼ぶ。

4.1.1 必要記憶容量

当たり前ではあるが、正記法ではデータの数に比例した記憶容量 ($O(|\boldsymbol{D}|)$) が必要となる。これは、災害救助のような大規模なシミュレーションではかなりのコストになる。

4.1.2 データの登録コスト

新しいデータを **DB** にいれる場合、その新しいデータはこれまで生成されたバージョンには全く含まれないため、正記法では特に余計なコストは生じない。

4.1.3 無矛盾性チェックのコスト

前節の定理で述べているように、シミュレーションにおいて成長する世界線を無矛盾に保つためには、バージョンの合同を行う際の無矛盾性の確保が一番問題となる。

2 つのバージョン u と v の無矛盾性をチェックするためには以下の操作が必要である。

$$\forall u' \in \mathbf{Tr}(u), \forall v' \in \mathbf{Tr}(v), \forall e \in (\boldsymbol{E}_{u'} \cap \boldsymbol{E}_{v'}) :$$
$$\text{check } \boldsymbol{D}_{u'}/e = \boldsymbol{D}_{v'}/e$$

ただし、$u' = \langle \boldsymbol{E}_{u'}, \mathcal{T}_{u'}, \boldsymbol{D}_{u'}, \boldsymbol{G}_{u'} \rangle$ および $v' = \langle \boldsymbol{E}_{v'}, \mathcal{T}_{v'}, \boldsymbol{D}_{v'}, \boldsymbol{G}_{v'} \rangle$ である。この操作は $O(\sum |\boldsymbol{D}_{u'}| + \sum |\boldsymbol{D}_{v'}|)$ の計算量を必要とする。

4.2 負記法

$\boldsymbol{D}$ のもう一つの記録方法として、そのバージョンが関係している事象 $\boldsymbol{E}$ のなかで使われなかったデータを記録する方法が考えられる。これを **負記法** と呼ぶ。

一般に、使われなかったデータは以下の場合に生じる。

- 1 つの事象に対し異なる値が報告されており、その一方を真、他方と偽として扱ってシミュレーションを行う場合。
- あるシミュレーションが時刻 $\mathcal{T}$ に開始され、その後、そのシミュレーションに関係する事象のデータが報告された場合。

負記法では以下のデータ集合を記録することになる。

$$\bar{\boldsymbol{D}}_v = \sum_{e \in \boldsymbol{E}_v} (\mathbf{DB}_{(0,\mathcal{T}_v)}/e - \boldsymbol{D}_v)$$

ここで注意しておかなければならないのは、上の式で $\mathbf{DB}$ の変わりに $\mathbf{DB}_{(0,\mathcal{T}_v)}$ を用いていることである。つまり $\bar{\boldsymbol{D}}_v$ は時刻 $\mathcal{T}_v$ において使われなかったデータのみを記録する。これ以降に入ってきたデータについては、そのデータのタイムスタンプ $\mathcal{T}$ とこのバージョンのタイムスタンプ $\mathcal{T}_v$ を比較することで $\boldsymbol{D}_v$ に含まれるかどうかが判定できる。

4.2.1 必要記録容量

負記法による記録容量は $O(\left|\bar{\boldsymbol{D}}\right|)$ である。一般に関連する事象で使われないデータは、使われるデータより少ないため、正記法よりは効率が良いと考えられる。

4.2.2 データの登録コスト

新しく登録されるデータは既存のバージョンでは使われなかったデータであるので、負記法では本来ならばいちいち $\bar{\boldsymbol{D}}_v$ に追加する必要がある。しかし上で述べているように、データが記録された時刻とバージョンのタイムスタンプの比較を行うことで $\boldsymbol{D}_v$ に含まれているかどうかを容易に判定できるため、実際には余計なコストは生じない。

4.2.3 無矛盾性チェックのコスト

2 つのバージョン u と v の無矛盾性をチェックするためには以下の操作が必要である。

$$\forall u' \in \mathbf{Tr}(u), \forall v' \in \mathbf{Tr}(v), \forall e \in (\boldsymbol{E}_{u'} \cap \boldsymbol{E}_{v'}) :$$
$$\begin{cases} \text{check } \bar{\boldsymbol{D}}_{v'}/e - \bar{\boldsymbol{D}}_{u'}/e = \mathbf{DB}_{(\mathcal{T}'_u,\mathcal{T}'_v)}/e & \text{if } \mathcal{T}'_u < \mathcal{T}'_v \\ \text{check } \bar{\boldsymbol{D}}_{u'}/e - \bar{\boldsymbol{D}}_{v'}/e = \mathbf{DB}_{(\mathcal{T}'_v,\mathcal{T}'_u)}/e & \text{if } \mathcal{T}'_v < \mathcal{T}'_u \end{cases}$$

ただし、$u' = \langle \boldsymbol{E}_{u'}, \mathcal{T}_{u'}, \boldsymbol{D}_{u'}, \boldsymbol{G}_{u'} \rangle$ および $v' = \langle \boldsymbol{E}_{v'}, \mathcal{T}_{v'}, \boldsymbol{D}_{v'}, \boldsymbol{G}_{v'} \rangle$ であり、$\mathbf{DB}_{(\mathcal{T}_x,\mathcal{T}_y)}$ は時刻 $\mathcal{T}_x$ から 時刻 $\mathcal{T}_y$ の間に記録されたデータの集合である。この操作の計算コストは、$O(\sum \left|\mathbf{DB}_{(\mathcal{T}_{v'},\mathcal{T}_{u'})}/\boldsymbol{E}\right| + \sum \left|\bar{\boldsymbol{D}}_{u'}\right| + \sum \left|\bar{\boldsymbol{D}}_{v'}\right|)$ である。

5 補足的な議論

正記法と負記法のどちらが効率がよいかは一般に決めることはできない。しかし大規模シミュレーションでは多くの場合、データの数は非常に多くなるため、前節の考察では負記法の方が記憶容量の点で有利と思われる。

また、4 節では射影の操作 (例えば $\mathbf{DB}_{(\mathcal{T}'_u,\mathcal{T}'_v)}/e$) を無視している。これは、一般的な RDB を用いた場合、これらの操作は $\log N$ 程度の高速な手法が用意されていることを前提としているためである。

また、事象集合 $\boldsymbol{E}$ の管理コストも上記の議論では無視している。これは、シミュレーションに関係する事象というのは、一般に機械的に指定されることを前提としている。例えば災害シミュレーションでは、入力としては指定された領域の特定の種類の地物 (「建物」や「道路」、「震度分布」など) という形で与えられる。これらをリストアップすることも可能だが、汎用的なデータベースを用いていれば、必要に応じて検索により求めていてもそれほどコストは生じないと考えられる。

この他、以下のような問題が形式化の上で残っている。

- バージョンの抽出に関する計算コスト。
- 統計的処理の合同や、モンテカルロシミュレーションの結果の解析に対するバージョンの考え方。

参考文献

[1] Satoshi Tadokoro. Japan government special project: development of advanced robots and information systems for disaster response (daidaitoku) — for earthquake disaster mitigation in urban areas —. In *Proc. of IEEE Workshop on Safety Security and Rescue Robotics (SSRR03)*, 2003.

[2] Satoshi Tadokoro. An overview of japan national special project daidaitoku (ddt) for earthquake disaster mitigation in urban areas. In *Proc. of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA2003) Workshop on Advanced Robots and Information Systems for Disaster Response*, 2003.

照明変動に頑健な ID 認識とロボットの姿勢検出

Robust Under Varying Light Condition and Accurate Detection of Object Orientation and ID

永橋知行，清水彰一，藤吉弘亘

Tomoyuki NAGAHASHI , Shoichi SHIMIZU , Hironobu FUJIYOSHI

中部大学 工学部情報工学科

Dept. of Computer Science, Chubu University, Japan

kida@vision.cs.chubu.ac.jp, shiyou@vision.cs.chubu.ac.jp, hf@cs.chubu.ac.jp

http://www.vision.cs.chubu.ac.jp/

Abstract

This paper describes a novel approach to detecting orientation and identity of robots without color segmentation. The continuous DP matching calculates the similarity between the reference pattern and the input pattern by matching the intensity changes of the robot markers. After the continuous DP matching, a similarity value is used for object identification, and correspondences of the optimal route obtained by back tracing are used for estimating the robot's orientation. This method archives orientation estimation of less than 1 degree and robustness with respect to varying light condition.

1 はじめに

より正確にロボットをビジュアルフィードバックで制御するには，ロバストなビジョンアルゴリズムが必要である．特に，ロボカップ小型リーグでは，照明変化に対してロボットの姿勢 (向き) や ID を正確に認識する必要がある．一般的なグローバルビジョンシステムでは，画像の色変換後，色識別，ID 認識，ロボットの姿勢 (向き) の流れとなる．ロボットの向きを得るには，最小 2 乗法[Murakami, 2003]やモーメント法[Ball, 2004]による手法が提案されている．このような姿勢推定の結果は，前処理である色識別の結果に依存する．色識別として，CMVision というライブラリが提供されており[Bruce, 2000]，多くのチームで使用されている．また，照明変動に対して頑健な色識別のための色校正法[Egorova, 2004]が提案されている．しかし，一般に色校正には多くの時間が必要である．また，校正時の照明環境下以外では，その色識別精度は低下するという問題がある．

本稿では，このような問題を解決する手法として，サブマーカの色識別を必要としない ID 認識とロボットの向きを連続 DP マッチングを利用した計算手法について提案する．本手法は，ロボット上部のマーカの色変化パターンを連続 DP によりパターンマッチングし，その最小累積距離を ID 認識に，バックトレースによる最適パスの対応からロボットの向きを同時に計算する．

2 ロボットのマーカと従来法

2.1 従来法

ロボカップ小型リーグでは，直径 50 mm の青色または黄色の円をメインマーカとしてロボット上部につける．また，サブマーカとしてロボットの上部にオレンジ色 (ボールに使用)，黄色，青色以外の 3 色のパッチを追加して，ID 認識や向きの検出に使用可能である．現在多くのチームで使用されているパッチの配置例を Figure 1 に示す．

Figure 1: 従来の ID プレート

(a)white bar は，白いバーと背景の黒とのエッジ点から最小 2 乗法やモーメント法[Ball, 2004]により角度を求める．また，ID はバー以外のサブマーカの配置から認識する．このような手法では，ロボットの方向と ID 認識の 2 つのプロセスを必要とする．また，ロボットの台数が増えた場合，サブマーカに色を用いる必要がある．

(b)butterfly はバタフライ型と呼ばれ，マーカの対称性からロボットの回転角度を求め，高精度に ID 識別が可能であることが報告されている[Bruce, 2003].

(c)pie slice-based は，ロボットの中心から円形に走査して，2 色以上の色配置を検出し，ID と角度を認識する

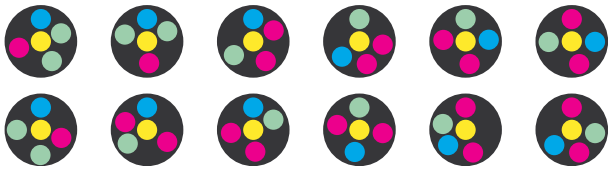


Figure 2: ID パターン例 (提案手法)

手法である[Hibino, 2002]．しかし，pie slice-based は円上のデータ数が 40 個であるため，向きの解像度は 8 度と低い．

上記に示したマーカによる従来法は，色識別の結果を利用しているため，影領域等での ID 誤認識や向きの推定精度の低下を招くという問題がある．

2.2 提案手法のマーカ

本稿では，色識別を行わずに円周上の輝度パターンを直接マッチングする手法を提案する．本手法では，Figure 2 に示すように，適当なサブマーカをメインマーカの周りに自由に設置すれば良い．そのため，設置する上での手間を大幅に削減できる．さらに，サブマーカの色配置から ID を識別することから，多くの ID を容易に作成することが可能である．また，認識時には，ID 毎のルールを決定する必要はない．

3 DP マッチングによる物体識別

カメラ画像から得られたロボット上部の円周上の輝度パターンと予め登録したパターンとの DP マッチングを行う．DP マッチング後，その最小累積距離を ID 認識に，バックトレース後の対応からロボットの向き推定に利用する．その処理の流れを Figure 3 に示す．

1. 色変換 (RGB to YUV)
2. メインマーカの抽出
3. 円周上の輝度パターンの作成 (1 次元化)
4. 連続 DP マッチングによる ID 認識
5. バックトレースによる向き推定

3.1 メインマーカの中心位置検出

メインマーカを中心とする円周上の色成分を基に ID の認識，及びロボットの向きの算出を行うため，メインマーカの中心位置を予め正確に求める必要がある．以下に，円周上の 3 点を用いた円中心位置の高精度検出法について述べる．

メインマーカの円周上の 3 点が与えられると，その 3 点から円を表すパラメータを推測することが可能である．中心座標を (x_c, y_c) とし，メインマーカ円周上の 3 点を (x_i, y_i), (x_j, y_j), (x_k, y_k) とすると，式 (1) の関係が成り立つ．

$$\begin{aligned}(x_c - x_i)^2 + (y_c - y_i)^2 &= (x_c - x_j)^2 + (y_c - y_j)^2 \\ &= (x_c - x_k)^2 + (y_c - y_k)^2\end{aligned} \quad (1)$$

式 (1) は連立一次方程式となるため，ガウスの消去法を用いて円の中心位置 (x_c, y_c) を解くことができる．以下に 3 点推測による円の検出手順を示す．

Step1 入力画像よりメインマーカの輪郭点を抽出する．

Step2 輪郭点からランダムに 3 点選び，式 (1) を用いて円中心を求める．

Step3 求めた円中心座標に重みをつけて投票する．

Step4 Step2~3 の処理を n 回繰り返す．

最終的に投票数が最大の画素を中心座標 (x_c, y_c) とする．

3.2 円周パターンの 1 次元化

3.1 で求めた中心から半径 r(本稿では 10 pixel) の円周上の YUV 成分を Figure 4 に示すように，1 度間隔で 360 点取得する．角度 θ における円周上の座標 (x, y) は次式となる．

$$x = r\cos\theta + x_c, \qquad y = r\sin\theta + y_c \quad (2)$$

実際には，半径 10 pixel の円周上の点は 76 点であり，360 点の分解能がない．そこで，座標 (x, y) の座標点における輝度値 $I(x, y)$ を，Figure 5(a) に示すように 4 近傍の輝度値からバイリニア補間により求める．

$$\begin{aligned}I(x,y) &= (1-n)((1-m)I(0,0) + mI(1,0)) \\ &+ n((1-m)I(0,1) + mI(1,1))\end{aligned} \quad (3)$$

バイリニア補間による Y 成分の抽出結果を Figure 5(b) に示す．バイリニア補間を行うことで，ロボット向き検出時におけるサブピクセル推定が可能となる．最後に，補間した $I(x, y)$ の Y 成分を，0~255 の値に正規化する．Figure 4(b) に 1 次元化した円周上のデータ (YUV) を示し，各角度 θ に対する輝度値関数を次式のように定義する．

$$I(\theta_j) = I(r\cos\theta_j, r\sin\theta_j) \qquad j = 0, \cdots, 359 \quad (4)$$

3.3 連続 DP マッチングによる ID 認識

各ロボット上部のパターンから，予めメインマーカの中心からある半径 r を持つ円周上の輝度パターンを登録しておく．この作業は，画面上からロボットの正面 (0 度) となる位置をユーザがマウスクリックにより与える．このように，本提案手法は従来法で用いた ID 認識のためのルール設定を必要とせず，短い時間で簡易に ID の設定が可能である．

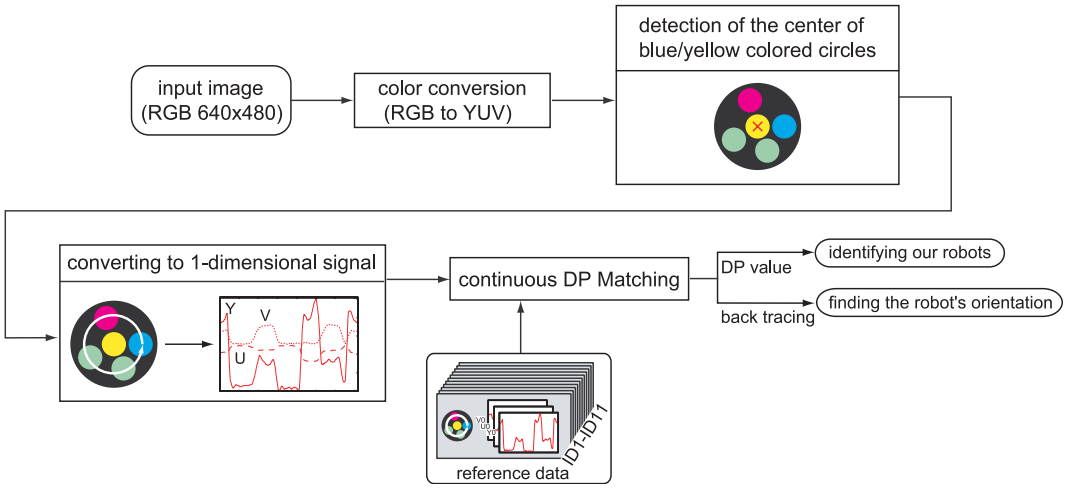
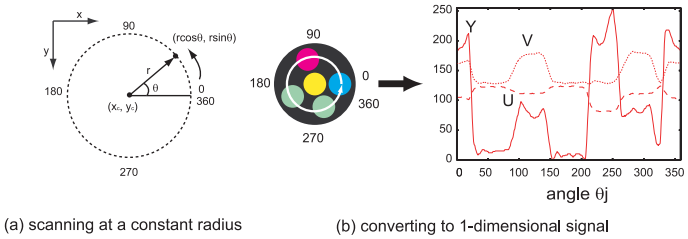
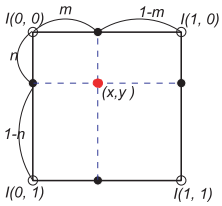


Figure 3: 連続 DP マッチングによる ID 認識と向きの推定

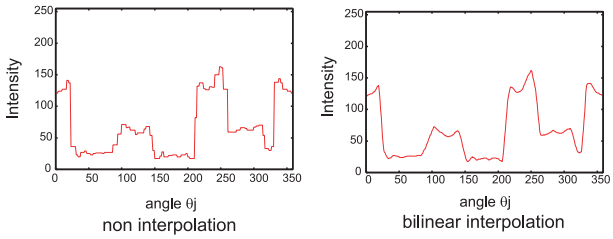


(a) scanning at a constant radius　　(b) converting to 1-dimensional signal

Figure 4: 1 次元データの作成法



(a) bilinear interpolation



(b) example of interpolation

Figure 5: バイリニア補間

3.3.1　連続 DP マッチング

DP マッチングは，異なる二つの時系列信号の類似度と最適な対応を求める手法であり，音声認識等の，さまざまな分野で利用されている．DP マッチングは非線形の伸縮により，長さの異なるパターンの類似度を計算することができる[Sakoe, 1978].

提案手法では，データの始点を固定しない連続 DP マッチングを行い，更に，入力パターンを 2 周分用意することで始点の対応位置に関する問題を解決する．以下に，用いる連続 DP マッチングの適用法について示す.

DP マッチングにおける始端領域においては，式 (5) に示すような初期値を設定する．

$$\begin{cases} g(i,0) & = 0 \quad (i = 0,1,\ldots,I) \\ g(0,j) & = \infty \quad (j = 1,2\ldots,J) \end{cases} \tag{5}$$

連続 DP マッチングに用いる DP パスを Figure 6(a) に示す．各格子点 (i,j) における累積距離 $g(i,j)$ を次式より求める．

$$g(i,j) = \min \left\{ \begin{array}{ll} g(i-1,j-2) + 2 \cdot ld(i,j-1) & : (a) \\ g(i-1,j-1) + ld(i,j) & : (b) \\ g(i-2,j-1) + 2 \cdot ld(i-1,j) & : (c) \end{array} \right\} + ld(i,j) \tag{6}$$

式 (6) は，Figure 6(a) に示す対照的な 3 つのパスから最も特徴が似ているパスを選択する．このとき，選択されたパスに対応したラベル $(a),(b),(c)$ をバックトレースの際に用いるために記憶しておく．

ローカルディスタンス $ld(i,j)$ は式 (7) より求める．

$$ld(i,j) = (I_t(\theta_i) - I_{t-1}(\theta_j))^2 \tag{7}$$

格子点 (i,j) までに選択された経路の長さを式 (8) より求める．

$$c(i,j) = \left\{ \begin{array}{l|l} c(i-1,j-2) + 3 & if(a) \\ c(i-1,j-1) + 2 & if(b) \\ c(i-2,j-1) + 3 & if(c) \end{array} \right. \tag{8}$$

以上の操作を各格子点にて行い，式 (9) より累積距離をそれまで辿った経路の長さで正規化する．

$$G(i) = \frac{g(i,J)}{c(i,J)} \tag{9}$$

3.3.2　ID 認識

マウスクリックにより各ロボットに対して，ロボットの正面を 0 度とした 360 度までの 1 次元化したパターンを

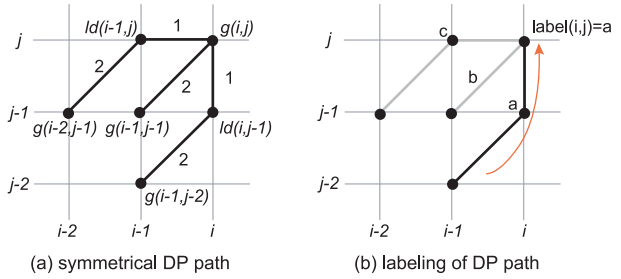


Figure 6: 対称型 DP パス

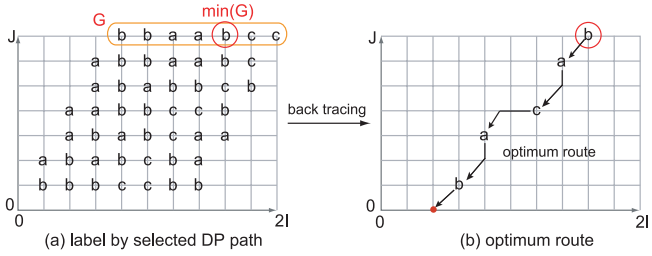


Figure 7: バックトレーシング

登録パターンとする．カメラ画像からメインマーカを検出し，入力パターンと全登録パターンとの DP マッチングを行う．そして，累積距離が最小となる登録パターンの ID を入力パターンの ID と決定する．

3.4 バックトレースによる向きの検出

ロボットの向き検出するには，3.3 で述べた DP マッチングの際にラベル付けされた DP パスを用いて，バックトレースすることで入力パターンと登録パターンの局所的な対応を求める．その処理を以下に示す．

1. 選択したパスの記憶
   各格子点 (i, j) において，最小累積距離を求める際に，式 (6) により選択されたパスをラベル (a), (b), (c) として記憶する．
2. 開始点の決定
   正規化累積距離を $J/2 \leq i \leq 2I$ の区間で求め，最小となる i' フレームを以下の式より求める．

$$i' = argmin_{(J/2 \leq i \leq 2I)} G(i, J) \tag{10}$$

3. バックトレース
   i' フレームを開始点とし，バックトレースを行う．Figure 7 に示すように，各格子点においてラベル付けされた (a), (b), (c) のどれかのパスを参照してバックトレースすることで最適ルートを求める．(a) のパスは脱落を意味し，(c) のパスは挿入を意味する．(b) のパスは，入力パターンの i フレームと登録パターンの j フレームが 1 対 1 に対応していることを意味する．そこで，(b) のパスが選択された場合のみ，登録パターンのフレームナンバー θ_j を入力フレーム θ_i から引いた値 θ をロボットの向きを表す角度とする．この処理を，バックトレースが $j = 0$ となる始点まで行う．

$$\theta = \theta_i - \theta_j \tag{11}$$

4. バックトレースの終了
   (b) のパスが選択された場合に求めた角度 θ の平均を求め，これをロボットの最終的な向きとする．

以上より，バックトレースにより得られる最適パス上の対応からロボットの向きを推定することが可能となる．Figure 8 に，同一 ID を持つロボットのバックトレース例を示す．DP マッチングにより得られた最適パスが示す対応が正確であることがわかる．

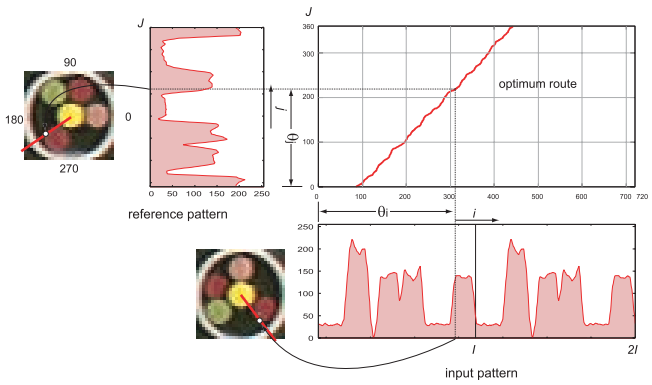


Figure 8: バックトレース例

4 実験

本手法の有効性を評価するために，ロボットの向き推定と照度変化に対する ID 認識を，シミュレーションと実画像を用いた実験を行う．

4.1 ロボット姿勢推定実験結果

4.1.1 シミュレーション実験

0~359 度の間を 1 度間隔で回転した ID パターンを擬似的に生成し，これを入力画像として回転角度である向きを求める．ID パターンには 11 種類用意し，計 3960 パターンに対して向きの推定実験を行った．推定した角度の平均誤差を Table 1 に示す．Table 1 の “最小 2 乗法” は，白いバーのエッジを最小 2 乗法により求めた結果であり，“2 次モーメント” も同様に白いバー領域を 2 次モーメント法により求めた結果である．また，実画像を用いた場合，メインマーカ領域を正確に求められるとは限らないため，提案手法では中心位置のずれに対した実験を行った．Table 1 より，本手法の精度は 1.0 度未満であることがわかる．また中心ずれとは，中心位置からの誤差を表し，中

Table 1: 方向算出結果の平均誤差 [度]

中心ずれ	提案手法		従来法	
	SSD	DP	最小 2 乗法	2 次モーメント
0	0.30	0.76	0.85	1.08
1	1.71	1.10	-	-
2	4.20	1.75	-	-

Table 2: 方向算出結果の平均誤差 [度]

	提案手法	最小 2 乗法	2 次モーメント
白いバー	0.85	1.17	0.96
ID パターン	0.95	-	-

心位置から 8 近傍にずれた場合を “中心ずれ 1” ，さらに 16 近傍の外側の画素を “中心ずれ 2” とする．中心位置がずれた場合，SSD の結果に比べ DP の方が，より精度良く求めることが可能である．これは，中心位置がずれることで生じる長さの伸縮を, DP マッチングが吸収して対応を求めることができるからである．

4.1.2　実画像を用いた実験

実際のカメラを高さ 4,000 mm の位置に設置し，マーカの方向の算出実験を行う．マーカを 1 度づつ回転させたときの真値との平均誤差を Table 2 に示す．実画像を用いた実験においても，シミュレーション実験と同様に，提案手法は従来法と同等以上の精度を得ることができた．また，Figure 1(b) のパターンである白いバーに対して，提案手法で方向を推定した結果，平均誤差 0.85 度となった．このように，提案手法は，バタフライ型や白いバーを用いた ID マーカの向きを正確に求めることが可能である．これは，敵チームの正面向きからパスの方向を推定することに利用できるため，インターセプト等の戦略への応用が期待できる．

4.2　ID の認識実験結果

4.2.1　シミュレーション実験

照明変動に対する頑健性を評価するために，擬似的に照度を変化させた画像を作成しシミュレーション実験を行う．CG における光源の照度を 0.1 から 1.1 まで 0.1 刻みに変化させることによって，擬似的に照明変動を作り出す．評価実験には，1 度刻みで 360 度回転したものに照明を変化させた計 43560 パターンを使用する．Figure 9 に認識結果を示す．本手法は中心ずれが 8 近傍以内であれば，照明変動に対して安定して ID 認識が可能であることがわかる．しかし，中心位置が 2 ピクセルずれた際，ID 認識率は低下している．これは，縦横それぞれに 2 ピクセル (14 mm) ずれたとすると，メインマーカの端に近い位置となり，色の変化パターンを含むように 1 次元データを生成できないからである．

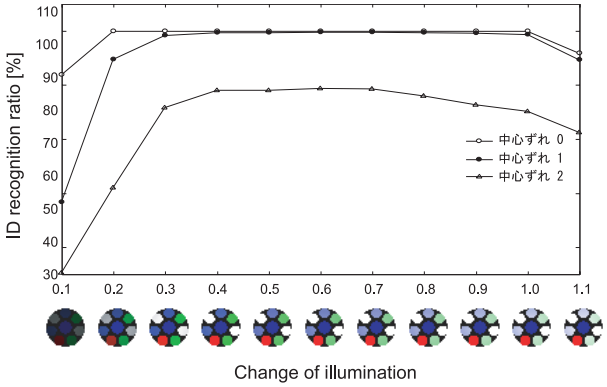


Figure 9: ID 認識実験結果

4.2.2　実画像を用いた実験

実画像における照明変動に対する評価実験を行う．照明を 100 lux から 2,900 lux まで 100 刻みで変化させた際のロボット上部の画像を Figure 10 に示す．また，11 種類の ID を 100 lux から 3,000 lux まで変化させた 330 パターンの画像に対して ID 認識実験を行った結果を Figure 11 に示す．従来法では，600 lux から 1,000 lux の画像に対して認識率を 100%になるように色識別の閾値処理を手動で設定した．一方，提案手法は 1,000 lux で撮影した 1 パターンを登録パターンとして実験に用いた．Figure 11 より提案手法は，色識別を行う従来法と比べて照明変動に頑健であることがわかる．これは，提案手法が色識別を行わず，ロボットの中心から一定の半径で走査して得られた輝度パターンの変化をマッチングに利用するからである．

5　考察

提案手法のメリットを以下に挙げる．

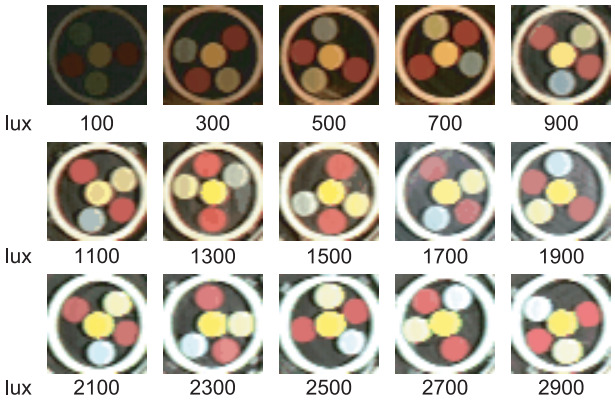


Figure 10: 100 lux から 2900 lux までの実画像

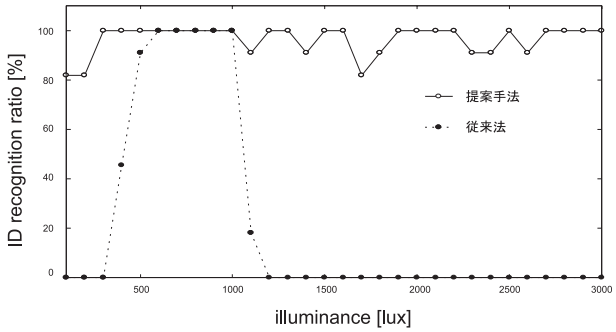


Figure 11: 照度変化に対する認識率

- **ID の初期登録の簡便性**
  ID の初期登録は，ID の中央 (メインマーカの中心) とロボットの向いている方向をマウスクリックにより指定するだけで，各 ID の 1 次元データに変換し，これを登録パターンとして記憶する．このように各 ID 毎にサブマーカの配置からルールを作成する必要がない．

- **多くの ID を作成することが可能**
  従来法では，サブマーカより向きを求める手法があり，マーカの配置を予め考慮する必要がある．また，白いバーを用いた場合は，サブマーカを配置するスペースが狭くなるため，ID の種類を多く用意することができない．しかし，提案手法では，サブマーカの配置や大きさ等の配慮が必要なく，無作為に取り付けるだけで，ID を作成することができる．そのため，容易に複数の ID パターンを作成することが可能である．

- **照明変動にロバストな ID 認識**
  提案手法では，YUV の Y を正規化した 1 次元データから DP マッチングを用いて ID を認識する．そのため，従来法では必要とするサブマーカの色識別と，その閾値を決定するための色校正を必要としない．

- **対戦相手のロボットの向きを取得可能**
  バタフライ型の ID は，実験で使用したパターンと同種であると考えられるため，初期登録さえしておけば，敵ロボットの ID とロボットの角度を求めることが可能である．また，白いバーを用いたマーカについても，角度を求めることができることを実験で示した．これにより，対戦相手のロボットの方向も知ることができるため，パスのインターセプト等，戦略の幅が広くなる．

以上に挙げたメリットに対して，提案手法のデメリットを以下に述べる．

本手法では，円周パターンの 1 次元化を行っている．そのため，円の中心が正確に求まっていなければ，正確に 1 次元化を行うことが困難になる．また，ID 認識及び向きを正確に行うには，中心位置の誤差を約 2 画素以内 (14 mm) に抑える必要がある．ID 認識を行う際，1 つのパターンに対し全ての登録パターンとマッチングをとらなければならない．よって，ロボットの数が増えるとその計算量が多くなる．

6 まとめ

本稿では，登録パターンと入力パターンの 1 次元データを DP マッチングにより，ID 認識とロボット方向を同時に求める手法について提案した．提案手法のロボット方向を約 1 度未満の誤差であり，従来法と同精度であることを示した．また，ID 認識では，実画像における実験より，300 lux から 3,000 lux の照度変化に対し高い ID 認識率を得ることができ，照明変動に対して頑健であることを確認した．また，提案手法はルールベースを必要としないので，敵ロボットの ID 並びに向きを知ることができる．そのため，敵ロボット間のパスのインターセプト等の戦術幅を広げることができると考えられる．

参考文献

[Murakami, 2003] K.Murakami, S.Hibino, Y.Kodama, T.Iida, K.Kato, S.Kondo, and T.Naruse: “Cooperative Soccer Play by Real Small-Size Robot”, RoboCup 2003: Robot Soccer World Cup VII, Springer, pp. 410-421, 2003.

[Ball, 2004] Ball D., Wyeth G., Nuske S: “A Global Vision System for a Robot Soccer Team”, Proceedings of the 2004 Australasian Conference on Robotics and Automation (ACRA), 2004.

[Bruce, 2000] J. Bruce, T. Balch, M. Veloso: “Fast andIn-expensive Color Image Segmentation for Interactive Robots”, In Proceedings of IROS-2000, Japan, 2000.

[Egorova, 2004] A.Egorova, M.Simon, F.Wiesel, A.Gloye, and R.Rojas: “Plug & Play: Fast Automatic Geometry and Color Calibration for Cameras Tracking Robots”, Proceedings of ROBOCUP2004 SYMPOSIUM, 2004.

[Bruce, 2003] J.Bruce, M.Veloso: “Fast and Accurate Vision-Based Pattern Detection and Identification”, Proceedings of ICRA-03, the 2003 IEEE International Conference on Robotics and Automation (ICRA’03), May, 2003.

[Hibino, 2002] S.Hibino, Y.Kodama, Y.Nagasaka, T.Takahashi, K.Murakami and T.Naruse: “Fast Image Processing and Flexible Path Generation System for RoboCup Small Size League”, RoboCup 2002: Robot Soccer World Cup VI, Springer, pp. 53-64, 2002.

[Sakoe, 1978] H. Sakoe, S. Chiba: “Dynamic Programming Algorithm Optimization for Spoken Word Recognition”, IEEE Trans. Acoust., Speech, and Signal Process., Vol.ASSP-26, pp. 43-49, 1978.

小型 (F180) リーグのためのイメージモザイクによるグローバルビジョンシステム

Mosaic-Based Global Vision System for Small Size Robot League

林裕司, 遠山聖司, 藤吉弘亘

Yuji Hayashi, Seiji Tohyama, Hironobu Fujiyoshi

中部大学 工学部　情報工学科

Dept. of Computer Science, Chubu University, Japan

yuji@vision.cs.chubu.ac.jp, sei@vision.cs.chubu.ac.jp, hf@cs.chubu.ac.jp

http://www.vision.cs.chubu.ac.jp/

Abstract

In the RoboCup F180 Small Size League, a global vision system using multiple cameras has been used to capture the whole field view. In the overlapping area of two camera's views, a process to merge information from both cameras should be needed. To avoid this complex process and rule-based approach, we propose a mosaic-based global vision system which produces high resolution images from multiple cameras. Three mosaic images, which take into account the height of each object such as our robots, opponent robots, and the ball on the field are generated by pseudo corresponding points. Our system archives a position accuracy of better than 14.2 mm(mean: 4 mm) over a 5,500 × 4,000 mm field.

1 はじめに

ロボカップ小型リーグでは，2004 年のルール変更に伴いフィールドサイズが 5,500 × 4,000mm に広くなった．これにより，従来用いられてきた 1 台の広角カメラによるグローバルビジョンでは認識精度の低下を招くため，複数台のカメラを用いる手法が主流になりつつある[Ball, 2004a]-[Ball, 2004b]．2 台のカメラを用いた場合，各カメラ画像においてロボットの位置推定や ID 認識等のビジョン処理の後，2 枚の画像が重なる領域において認識結果の統合処理が必要である[Ball, 2004a].

このような複雑な統合処理を回避する一手法として，2 枚の画像から 1 枚のモザイク画像を生成し，ビジョン処理を施すことが考えられる．しかし，RoboCup ではボールやロボット等の高さが異なるオブジェクトが複数存在するため，フィールド面上の対応点を用いて生成したモザイク画像では，高さを持つオブジェクト (ロボット等) が画像の重なり領域で二重に見え，認識が困難となる問題が生じる．そこで，我々は認識対象オブジェクトのフィールド面からの高さを考慮したイメージモザイクによるグローバルビジョンシステムを提案する．本システムでは，自由な高さでのモザイク画像を生成することが可能であるため，高精度なロボットの位置推定が可能である．

2 2 台のカメラによる重なり領域での ID 認識の問題

2 台のカメラを用いて，フィールド全体を高解像度 (5[mm/pixel]) で撮影するには，Figure 1 のように画像の一部を重ね合わせて，左半分をカメラ 1，右半分をカメラ 2 がカバーするように配置する必要がある．画像の一部を重ねることにより，認識結果の統合処理が必要になる．統合処理には以下の手法が考えられる．

A. 最新フレームのカメラ結果を優先 2 台のカメラ間が非同期の場合，常に最新フレームの認識結果を用いる．

B. 境界による判定統合 2 台のカメラの重なり領域内において，境界を予め決定する．各カメラ画像において境界から内側のみ認識対象とする．

C. トラッキング結果による統合 重なり領域においては，前フレームでトラッキングが成功したカメラの処理結果を使用.

D. 重み付き統合 各カメラ画像の認識結果を重み付けして共通の座標に統合して使用.

例えば，Figure 1 においてカメラ 1 の画像上のロボット 2 は，ID 用マーカーの一部が切れているため A，C，D ではロボット 1 との識別が不可能である．また，B では，カメラを取り付ける位置が変わる度に，ロボット上部の ID 用マーカが切れない場所に境界を再設定する必要がある．

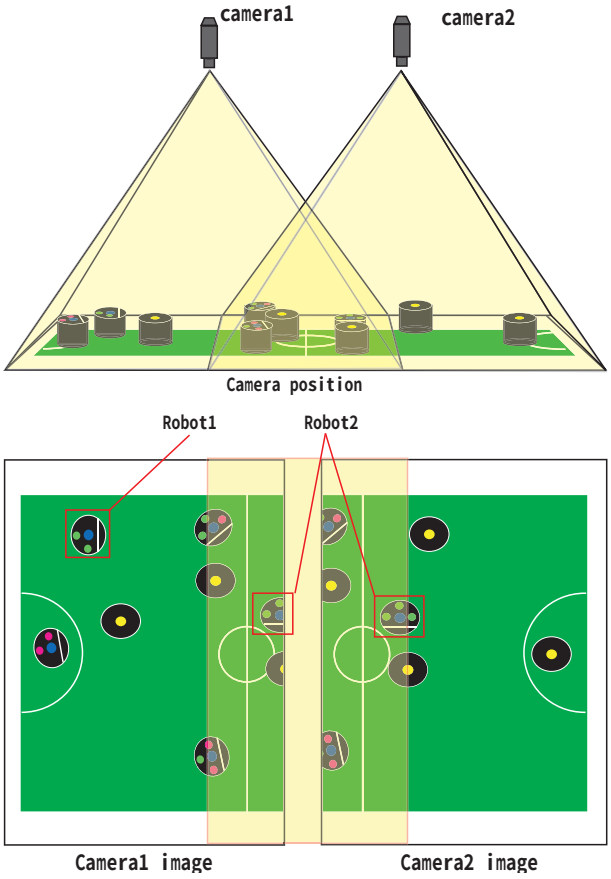


Figure 1: カメラを 2 台使用時の設置例

これらの手法では，認識後に統合処理を行うため，画像端での誤認などが起こる．そこで，認識処理前の画像 2 枚をイメージモザイク処理により 1 枚の高解像度画像を生成する．これにより認識前に統合処理を行い誤認を防ぐ．

3 An Image Mosaic Based Global Vision System

2 台のカメラから取得した各画像を平面射影変換により，1 枚のモザイク画像を生成する．処理画像を 1 枚にすることで，画像端での ID の誤認や各カメラからの認識結果の統合処理を省くことができる．しかし，平面射影変換では対応を求めた基準平面上に存在しない物体は，モザイク画像上において，二重に見えるなどの問題が生じる．そこで，本システムでは，ボール，味方ロボット，敵ロボットの高さに対応した平面における 3 枚のモザイク画像を生成する．その後，各モザイク画像に色識別，ID 認識等の処理を施す．本システムでは，カメラキャリブレーション[Tsai, 1987]により求めた外部・内部パラメータから擬似的に対応点を生成し，対象とする高さにおけるモザイク画像を生成するため，ロボット等の高さを考慮した位置推定が可能となる．

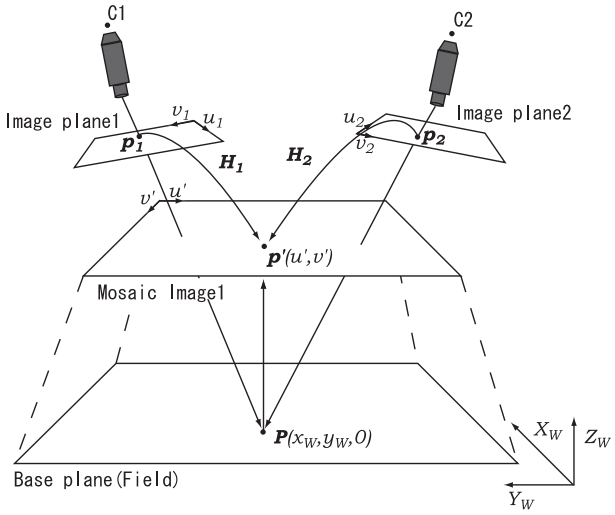


Figure 2: フィールド面での射影変換行列の算出

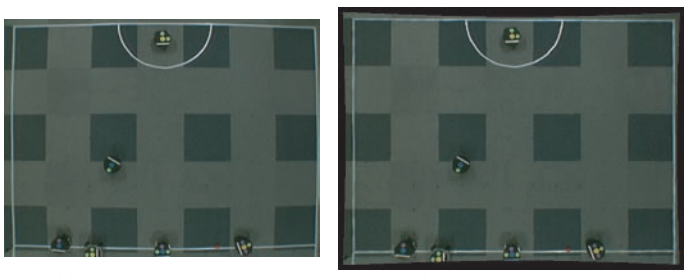
3.1 フィールド面のモザイク画像の生成

Figure 2 にフィールド面と各カメラ画像間の関係を示す．カメラ 1 における座標 $p_1 = [u_1, v_1]^T$ と世界座標 $P = [x_W, y_W, 0]^T$ に対応したモザイク画像座標 $P' = [u_m, v_m]^T$ から平面射影行列 H_1 を求める．平面射影行列は，Homography と呼ばれ，3 次元空間の各平面毎に異なる値を持つ．また，2 枚の画像間における 4 点以上の対応点から求まり，3 × 3 の 9 要素，自由度は 8 の行列となる．同様にカメラ 2 の画像座標 p_2 からモザイク画像への平面射影行列 H_2 を求める．これらの関係は次式で表現される．

$$p' = H_1 p_1 = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \quad (1)$$

$$P' = H_2 p_2 \quad (2)$$

$$P = \alpha P' \quad (3)$$



(a)Before lens correction　(b)After lens correction

Figure 3: レンズ歪み補正

3.1.1 レンズ歪み補正

カメラから取得した画像には，レンズ特有の歪みが含まれる．歪みを含んだ画像座標における対応点から平面射影行列 H を求めると誤差が生じ精度が低下する．そこで，カメラキャリブレーションから求めたレンズ歪み係数 k_1, k_2 を用いて補正を行う．歪みを含んだ画像座標 (u, v) から歪みの含まない座標点 (X_u, Y_u) への変換を式 (5) に示す．(u_c, v_c) は画像座標の中心，r は画像中心からの距離，S はせん断係数である．

$$X_d = u_c - u \quad , \quad Y_d = v_c - v \tag{4}$$

$$\begin{aligned} X_u &= S X_d (1 + k_1 r^2 + k_2 r^4) \\ Y_u &= Y_d (1 + k_1 r^2 + k_2 r^4) \end{aligned} \tag{5}$$

レンズ歪み補正後の対応点から，平面射影行列を求めることで，より正確な平面射影変換，即ち位置推定が可能となる．Figure 3 に，レンズ歪み補正後の画像を示す．歪み補正することにより，画像端における湾曲していたラインが直線に補正されていることがわかる．

3.1.2 モザイク画像座標と世界座標の対応

平面射影変換で用いる座標系として，どちらか一方の画像座標系に他の画像座標系を変換する考え方 (相対座標系) と，どちらの画像を新しい座標系，例えば世界座標に座標変換する考え方 (絶対座標系) の 2 つがある．本手法では，モザイク後の画像座標から，世界座標を直接求めることができる後者を用いる．このとき，分解能を 5mm/pixel となるように世界座標系を設定すると，フィールドサイズは 5,500 × 4,000mm であるため，モザイク画像の解像度は 1,100 × 800 画素となる．しかし，実際のカメラ画像の解像度は 5.5mm/pixel であるため，モザイク処理の際に，値が存在しない画素が生じる．このような画素は最近隣内挿により補間する．そのため，ビジョン処理により求めたモザイク画像上の座標から世界座標を式 (6) により求めることが可能となる．

$$\begin{aligned} &\text{世界座標 } P[\text{mm}] = \\ &5[\text{mm/pixel}] \times \text{モザイク画像の座標 } P'[\text{pixel}] \end{aligned} \tag{6}$$

3.1.3 フィールド面のモザイキング

フィールド面のモザイク画像の生成過程を以下に示す．

Step1　カメラ 1(C1) の画像上で位置が確認できるフィールド面の特徴点 P を選定し，その世界座標 $P(x_W, y_W, 0)$ とカメラ 1 の画像座標 $p_1(u, v)$ を計測する．

Step2　式 (6) より世界座標 P をモザイク画像座標 P' に変換する．

Step3　モザイク画像座標 P' とカメラ 1 の画像座標 p_1 の 4 点以上の対応点群から平面射影行列 H_1 を求める．カメラ 2(C2) も同様の方法で平面射影行列 H_2 を求める．

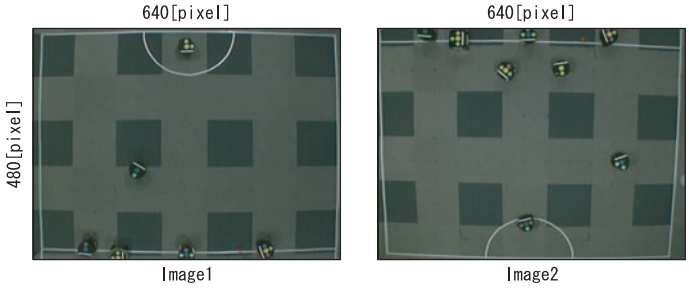


Figure 4: カメラ画像

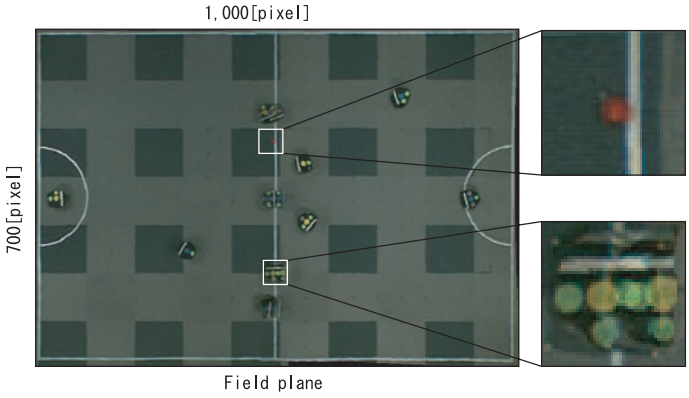


Figure 5: フィールド面でのモザイク画像

Step4　求めた H_1 と H_2 によりモザイク画像を生成する．その際に，モザイク画像上の重なり領域ではブレンディング処理を施す．

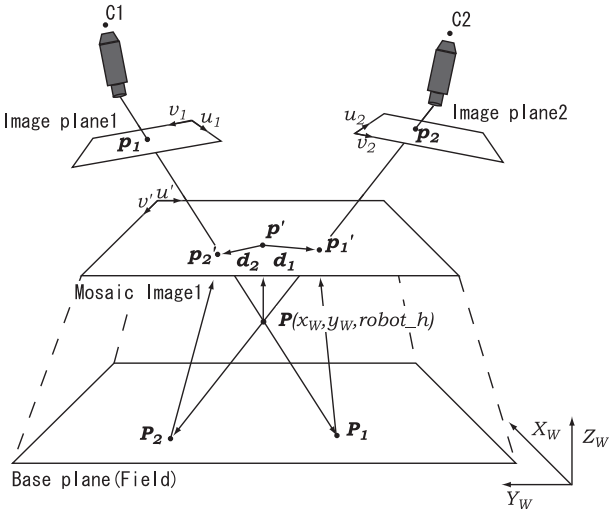


Figure 6: 二重に見える原因

Figure 4 に各カメラからの画像を，Figure 5 にフィールド面で生成したモザイク画像を示す．フィールド面上のボールは，継ぎ目がないモザイク画像が生成できていることがわかる．しかし，フィールド面から高さを持つロ

ボット上部の ID マーカが二重に見えるという問題が発生している．これは，Figure 6 に示すように，フィールド面から高さを持つ世界座標 $P(x_W, y_W, Robot_h)$ は，カメラ $C1$ では世界座標 P_1 へ射影され，誤差 d_1 が生じる．同様にカメラ $C2$ では P_2 に射影され，誤差 d_2 が生じるためである．

3.2 高さを考慮した仮想平面上のモザイク画像の生成

フィールド面上の特徴点から求めた平面射影行列は，フィールド面と高さの異なるロボット上面での対応を取ることはできない．高さを考慮したモザイク画像を生成するためには，求めたい高さを持つ平面上の特徴点を求め，平面射影行列を計算する．しかし，特定の高さを持つ平面上に存在する特徴点を実際に設置し測定するのは非常に困難である．そこで，Figure 7 に示すようにフィールド面の特徴点から求めたい高さの平面上 (仮想平面) の画像座標を擬似的に生成し，仮想平面上での平面射影行列を求める．

Step1　カメラ 1(C1) の画像上で位置が確認できる特徴点 $P(x_W, y_W, 0)$ を選定する．

Step2　フィールド面での世界座標 P を Z_w 軸方向に生成したい仮想面の高さ (Robot の高さ)$Robot_h$ を持つ世界座標 $Q(x_W, y_W, Robot_h)$ を求める．

Step3　世界座標 Q から外部・内部パラメータを用いて以下に逆投影を行う式を示す．逆投影により画像座標 $q_1(u_1, v_1)$ を求める．R は回転行列，T は平行移動ベクトル，f は焦点距離，s はせん断係数である．

$$\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = \frac{1}{Robot_h'} \begin{pmatrix} fk_u & fs & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} (R \mid -RT) \begin{pmatrix} x_W \\ y_W \\ Robot_h \\ 1 \end{pmatrix} \quad (7)$$

Step4　世界座標 Q から仮想平面 (高さ $Robot_h$) のモザイク画像座標 Q' に変換し，Q' と q_1 の関係から平面射影行列 H_1' を求める．カメラ 2(C2) も同様の方法で画像座標 H_2' を求める．

Step5　求めた H_1' と H_2' より仮想平面でのモザイク画像を生成する．その際に，モザイク画像上の重なり領域ではブレンディングを施す．

Figure 8 に，ロボットの高さを 150mm とした際のモザイク画像を示す．フィールド上のボールはモザイク画像上にずれが生じるが，高さを持つロボット上部のマーカは継ぎ目がなく生成されていることがわかる．イメージモザイクによるグローバルビジョンでは，認識対象オブジェクトの高さにおけるモザイク画像を生成するため ID の誤認識や統合処理を必要としない．また，モザイク画像には，従来のビジュアルアルゴリズムが適用できるというメリットがある．

4 実験

高さを考慮したモザイク画像を用いたロボットの位置推定実験を行う．

4.1 実験概要

本実験では，2 台のカメラを高さ約 3,000mm，それぞれフィールド面上の 4,900 × 3,400mm の領域 (重なり領域 300mm) が視野に入るように設置する．カメラキャリブレーションには 500mm の正方形が描かれたキャリブレーションシートを床に敷き，正方形の頂点をランドマーク点 (カメラ 1 は 48 点，カメラ 2 は 40 点) として用いた．ロボットの位置推定実験として，高さを考慮したモザイク画像とフィールド面のモザイク画像の各 61 点の画像座標から世界座標を求めた．

4.2 実験結果

Table 1 にフィールド面でのモザイク画像と仮想面 (高さ 150mm) でのモザイク画像から計算した世界座標と実測値の誤差を示す．仮想面でモザイクすることにより位置推定精度が向上することがわかる．仮想面でのモザイク画像を用いた位置推定では，平均誤差が約 4mm であり，実際のカメラ画像の解像度 (5.5mm/pixel) より小さいことから，高精度な位置推定を実現しているといえる．Figure 9 に各位置における誤差分布を示す．フィールド面のモザイク画像では Figure 9(b) に示すように，高さをもつオブジェクトが光軸中心から離れる程その誤差は増大していることがわかる．一方，仮想平面でのモザイクした場合は，全体的に推定誤差が小さいことがわかる．

4.3 処理時間

本グローバルビジョンシステムのハードウェア構成を Table 2 に示す．モザイク画像を生成する際，毎フレームごとに H を用いて座標変換すると計算コストが高くなる．そこ

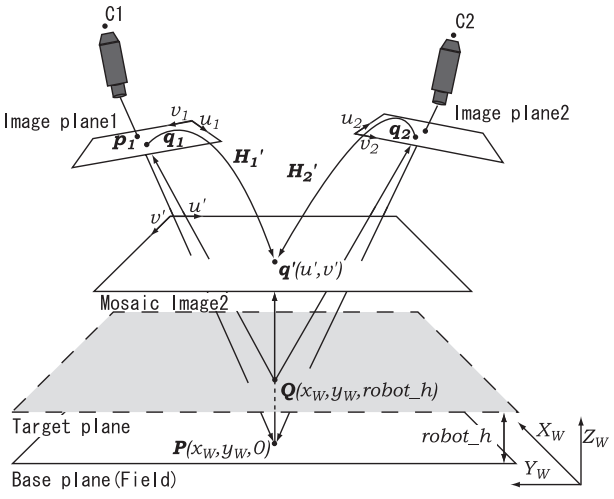


Figure 7: 仮想平面の算出

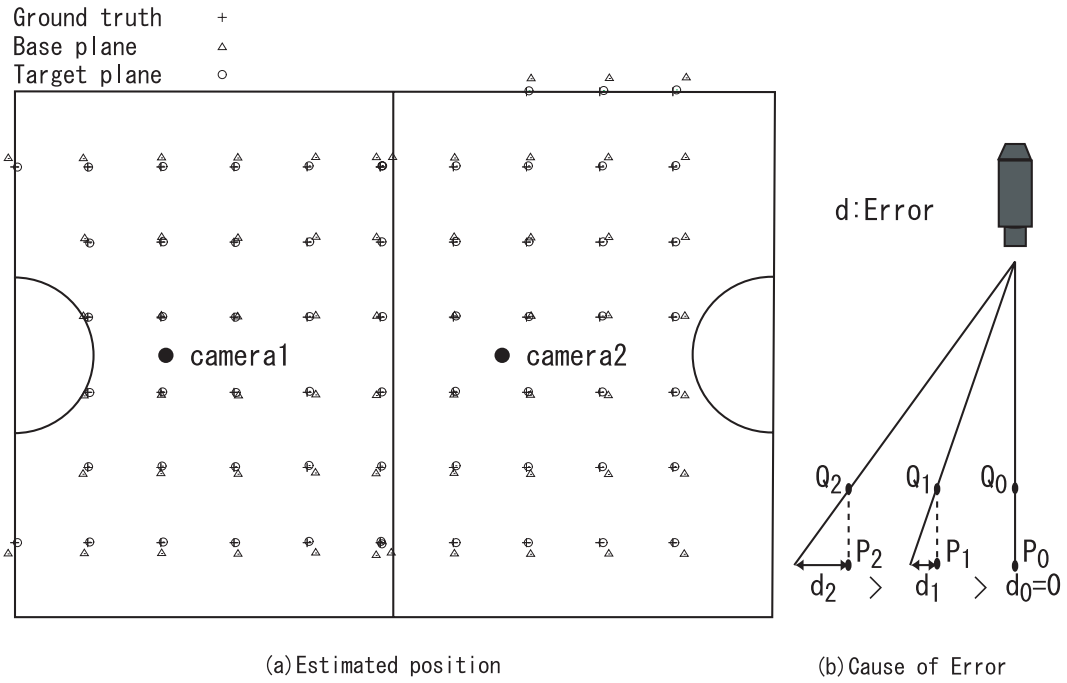


Figure 9: 位置推定誤差

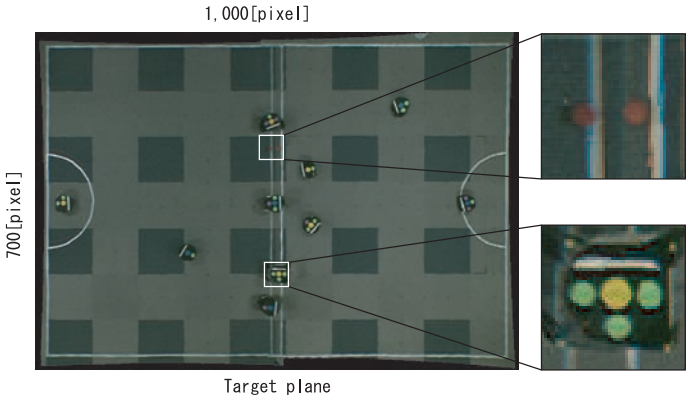


Figure 8: 仮想平面でのモザイク画像

Table 1: 位置推定誤差 [mm]

高さ	平均誤差		標準偏差		最大誤差	
	x 方向	y 方向	x 方向	y 方向	x 方向	y 方向
フィールド面	43.5	34.3	24.9	20.7	95.7	77.1
仮想面	3.7	4.3	2.8	2.8	12.0	14.2

で，モザイク画像座標は，予めカメラ画像と対応を表す LookUpTable(LUT) を作成する．作成した LUT による 3 枚のモザイク画像の生成に要する時間は (フィールド面用モザイク，味方 Robot 用モザイク，敵 Robot 用のモザイク) 約 39ms，色識別に約 38ms，ID 認識に約 4ms，合計 81ms(約 12fps) ある．12fps のフレームレートでは高速なビジュアルフィードバック処理を実現できないため，過去フレームで求めたオブジェクトの座標周囲 30pixel(約 150mm) のみ処理を施すことにより，モザイク画像生成に約 5.5ms，色識別に約 5.5ms，ID 認識に 1.1ms，合計 12.1ms(82fps) の高速化が実現できている．

Table 2: 実験環境

CPU	XEON DUAL PROCESSOR 3.2GHz
Memory	1GB
カメラ	DXC9000(SONY)
ズームレンズ	VCL-0716(SONY)
広角レンズ	WCV-65(FUJINON)
キャプチャーカード	CT-3301RGB(サイバーテック社)

5 まとめ

本稿では，イメージモザイキンクによるグローバルビジョンシステムを提案した．提案したシステムは，2 台のカメラ画像を仮想平面上でイメージモザイキングすることにより，画像の重なり領域での ID 誤認を軽減し，モザイク画像に対して従来のビジョンアルゴリズムを適用できるというメリットがある．また，ロボットの高さを考慮したモザイク画像を生成するため，高精度な位置推定を評価することを実験により示した．今後の課題として，さらなる処理速度の向上を目指す予定である．

参考文献

[Benosman, 2002] Ryad Benosman, Jerome Douret, and Jean Devars: A Simple and Accurate Camera Calibration for the F180 RoboCup League, RoboCup 2001, INAI 2377, pp.275-280, 2002

[Ball, 2004a] David Ball, Gordon Wyeth and Stephen Nuske: A Global Vision System for a Robot Soccer Team, Proceedings of the 2004 Australasian Conference on Robotics and Automation (ACRA), Canberra, Australia, 2004.

[Egorova, 2004]
Anna Egorova, Alexander Gloye, Cuneyt Goktekin, Achim Liers, Marian Luft,Raul Rojas, Mark Simon, Oliver Tenchio, and Fabian Wiesel: FU-Fighters Small Size 2004, RoboCup 2004 Symposium, Small Size League Team Description, 2004. *Artificial and Human Intelligence*,

[Kiat, 2004] Ng Beng Kiat: LuckyStar 2004, RoboCup 2004 Symposium, Small Size League Team Description.

[Ball, 2004b] David Ball, Gordon Wyeth: UQ RoboRoos 2004:Getting Smarter, RoboCup 2004 Symposium, Small Size League Team Description, 2004.

[Tsai, 1987] R. Y. Tsai: A verstile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the Shelf TV Cameras and Lenses, In IEEE Journal of Robotics and Automation, Vol.RA-3, Num.4, pp. 323-344, 1987.

[Zhang, 2000] Z. Zhang: A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330-1334, 2000.

[Szeliski, 1994] R. Szeliski: Image mosaicing for tele-reality applications Rroc.IEEE Workshop on Applications of Computer Vision, pp.44-53, 1994.

[Hibino, 2002] Shinya Hibino, Yukiharu Kodama, Yasunori Nagasaka, Tomoichi Takahashi, Kazuhito Murakami, and Tadashi Naruse: Fast Image Processing and Flexible Path Generation System for RoboCup Small Size League, pp.53-64, 2002.

[Bruce, 2000] James Bruce, Tucker Balch and Manuela Veloso.: Fast and Inexpensive Color Image Segmentation for Interactive Robots, In Proceedings of IROS-2000, Japan, October 2000.

行動による通信 B-MODEM

The B-MODEM

西野順二、戸田英治

Junji NISHINO, Eiji TODA

電気通信大学 システム工学科

Dept. of systems engineering,

The University of Electro-Communications

nishino@se.uec.ac.jp

Abstract

This paper presents a novel communication method B-MODEM using actions for carrier instead of wireless communication medium. Behavior modulation is a kind of selection from action sequences leading given goal situations. The proposed method shows effectiveness with an experimental result on scenario communications.

1 はじめに

マルチエージェントシステムの協調作業において、相互の通信は意志疎通を行い行動の同期を実現する重要な要素である。しかし、多対多の通信環境での電波無線通信や音声通信は混信や外乱による不到達など通信の失敗の発生が頻繁であり、協調行動の実現に悪影響を与えていることが少なくない。特に RoboCup サッカー実機部門のような多数のロボットが開かれた環境で動作するためには電波の管理が必須であり、技術的にも、また行政上でも大きな課題となっている。

本稿では、無線、音声に頼らない多対多の通信手段として、エージェントの行動とその視覚的観測による手法、B-MODEM を提案する。物理的な意味でエージェントの行動とはすなわち位置、配置の変更であり、これらは他のエージェントから光学的に観測することができる。行動の中に協調のためのメッセージを混入することができれば、電波や音声に頼らない通信が可能となる。このような行動による通信は、ジェスチャーやボディランゲージとして実用にある。しかし、いわゆるジェスチャーは、本来目的とする行動をいったん止めたうえで通信のためだけの動作を行うものであり、時間効率の面では時間的な制約があるなかでの通信としては不満である。本稿で提案する B-MODEM は、目的とする行動を微小に変調することでメッセージを送ることを定式化するものである。より効率の良い行動による通信のありかたを考え、目的行動と通信の混合方式の例をいくつか示す。

以下では、B-MODEM の実用上の背景、数理的な原理、実装例とその方法を説明し、RoboCup シミュレーションサッカー[Pet99]での実験を通じて有効性を示す。

2 B-MODEM の位置付け

エージェントの行動は位置の物理変化であり、実際のロボットなどでは遠隔から光学的に観測できる。この光学的 (視覚的) な情報を用いてメッセージ通信を行うことが可能である。B-MODEM(Behavior Modulation and DE-Modulation) は、本来そのエージェントに必要な目的行動、サッカーの例ではドリブルによるボール運びなど、を行いつつその上にメッセージを載せるものである。例えば自動車運転では左折の前にそれを示すため左路側に幅寄せすることが推奨されており、これも目的行動をメッセージによって変調した例である。

エージェントの行動によってメッセージ通信を行うことについて、従来からジェスチャーやボディランゲージなどを対象に、ノンバーバルコミュニケーションとして研究がなされている[黒川 94, 岡 98, 山中 03, 西田 03]。しかし、ジェスチャーや手話は、通信のためだけに動作を行うことに関するものであり、目的の行動を行いながら同時にメッセージを送ることについては扱われていない。さらに、狭い意味でのボディランゲージは人間が無意識に表出するメッセージを扱うものであり、目的の行動と同時に表出されるものの積極的な通信手段ではない。

いっぽう、意図的に、目的行動と同時にメッセージを行動で送り受信するという、より一般的で実用的な枠組はこれまであまり顧みられてこなかった。実際には B-MODEM が提案する通信モデルには、ジェスチャーや手話も含まれている。本稿ではとくに目的行動とメッセージを意図的に効率良く混合することを扱う。

3　行動による通信の原理

3.1　通信路としての行動

一連の動作により実現される目的を持った動きをエージェントの行動 A と呼ぶ。動作は物理的な運動すなわちアクチュエータの稼働により引き起こされる行動の基本要素である。とくにエージェントに対して与えられたタスクを遂行するための一連の動作列を、目的行動と呼ぶことにする。

タスクの遂行は、現在の状態 S から目標状態 G へ遷移することであり、動作列 $A = (a_1 a_2 \cdots a_n)$ で表すことができる。一般に S から G までの状態遷移ルートとそのための動作列には複数の代替案 A_j があり、このため目的行動は動作列の集合 $B = \{A_1, A_2, \cdots, A_m\}$ となる。

行動を変調して通信メッセージを載せることは、ある目的に適合した目的行動集合 B の中から、一つの行動 A_i を選び出すことである。このため現状態からゴールへ到達する動作列の代替案が一つに制限される場合は、行動に変調を加えることができないことになる。ただし実用上は目的行動が一つに制限されることはまれである。通信メッセージの復調では、観測した動作列の推測値 $(a'_1 a'_2 \cdots a'_n)$ と B との照合処理を行うことでメッセージとしての行動 A_j を抽出する。このとき行動 A_i の番号 i が入力アルファベットであり、抽出された行動 A_j の番号 j が出力アルファベットである。

一般に動作の遂行や観測にはゆらぎが伴う。入力 i が与えられて出力 j を観測する確率を p_{ij} とおくと、行動による通信を行うときその通信路行列は

$$T = [p_{ij}] \tag{1}$$

で与えられる。

3.2　変調と復調

メッセージ発信側のエージェントを S、 受信側を R とし、S は、時刻 T においてメッセージ M と目的行動 A を持つとする。R は、時刻 T 以降 S の行動 A' を観測する。

目的行動 A とは、現在の S の状態を目的の状態 G に変更する動作列である。A を表す動作列では多くの場合に動作の組合せは複数存在し、そのすべてを毎度網羅することは不可能である。そこで、発信側がメッセージを行動に混合し通信を行うとき、標準となる行動列 A を随時変更しメッセージを含んだ行動列を生成するものとする。

変調すなわち動作列の書き換えはつぎのように行われる。メッセージを $M = (m_1, m_2, \cdots, m_l)$ という一連の動作列で表現すると、目的行動とメッセージの混合は、

$$A + M = (am_1, am_2, \cdots, am_l, \cdots, a_n) \tag{2}$$

のように表される。

ここで、am_i にはいくつかの実現方法がある。任意の a_i と m_i とが加法性を持つならば、$am_i = a_i + m_i$ と与えて、

$$A + M = (a_1 + m_1, a_2 + m_2, \cdots a_l + m_l, \cdots a_n) \tag{3}$$

のように表すことができる。このとき全体のステップ数は n であり、標準となる目的行動と同等の効率を実現できる。

また動作相互に干渉が無く、時刻にも依存せず、順序のみで行動が定まるならタイムシェアリングを行って、

$$A + M = (a_1, m_1, a_2, m_2, \cdots a_l, m_l, \cdots a_n) \tag{4}$$

と表せる。ただし、この場合は総ステップ数が l+n ステップに拡大しているため、効率は必ず下がってしまう。目的行動に時間的な変移の耐性があるとすると、

$$A + M = (m_1, m_2, \cdots m_l, a_1, a_2, \cdots a_n) = (M, A) \tag{5}$$

と並び変えることができる。これはメッセージ行動と目的行動を逐次的に行う意味となり、いわゆるジェスチャーによるメッセージ通信そのものである。

形式的には時間で混合することが可能だが、このとき生成した動作列が適切に遂行されゴール状態へ到達できる保証はなく、目的行動となっていない場合がある。このため、それぞれの目的行動とメッセージの相互依存関係の度合により、混合方法を選択する必要がある。動作の相互依存関係については、１) パラメータ加算による状態遷移への非線形効果：増幅による行動失敗、２) 並行実行動作の相互干渉：同時に実行することによる歪み、３) 時間ずれによる無効化：コンビネーション効果の減衰、などがある。

4　サッカーにおける実装例

RoboCup サッカーシミュレーションは、11 対 11 の人工エージェントによる試合を行う、マルチエージェント協調の標準問題である。

動作として、Dash、Turn、Kick、TurnNeck を行うことができる。エージェントの状態には、(x, y, dx, dy, r) 位置、速度、方角がある。ボールの状態は (x, y, dx, dy) 位置と速度がある。エージェントとボールの状態は動的であり、現在の速度によって次ステップの位置が更新され、また速度は摩擦による一定の減衰を受ける。動作 Dash(n) によりエージェントの速度は加速度パラメータ n に応じて増加する。動作 Turn(n) によりエージェントの方角はトルクパラメータ n に応じて変化する。エージェント近傍に存在するボールの速度は、動作 Kick(n,d) により加速度パラメータベクトル n (d は相対方向) に応じて変化する。動作 TurnNeck(n) によりエージェントの視線方向はトルクパラメータ n に応じて変化する。

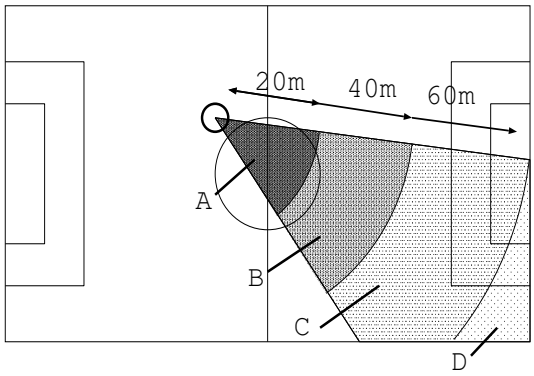


Figure 1: Range of agent visual cognition area. All in A; Team, Distance and Angle in B other stochastic; Distance and Angle in C other stochastic; Distance and Angle only in D

Table 1: 動作の加算可能性

D+D	○パラメータ加算可能
D+T	×同時実行不可能
D+K	×
D+N	◎並行実行可能
T+T	○
T+K	×
T+N	◎
K+K	○
K+N	◎
N+N	○

観測には図 1 に示した距離による歪みがあるが、対象物すなわち発信者の位置と方向は読み取ることができる。ただし、その値には確率的なノイズが重積している。

4.1 サッカー行動の変調

サッカーシステムで実施可能な 4 種類の動作、Dash, Turn, Kick, TurnNeck をそれぞれ、D, T, K, N と呼ぶことにする。ここでのメッセージはこれらの 5 つの行動の列として、たとえば (D D T K D) のように表すことができる。これらの動作の順序組合せによる特性は次のとおりである。

連接には次の特性がある。二つの動作の間に他の動作 (O) を追加したときの状態遷移と比べたときの効果である。

(D,D) 非線形 (増強効果)、(D,T) 非線形 (打消効果)、(D,K) ボールとの位置変化に応じて効力が変化する非線形効果がある (増強・打消両方)、(D,N) 干渉なし、(T,*) 干渉なし、(K,*) 干渉なし、(N,*) 干渉なし。

以上をまとめると、TurnNeck はつねに他の動作と加法的にも、時間連結でも混合可能である。加算的混合では、Dash, Turn, Kick は相互干渉が強く、連接においても干渉がある。とくに Kick は実行不可能なときがある。

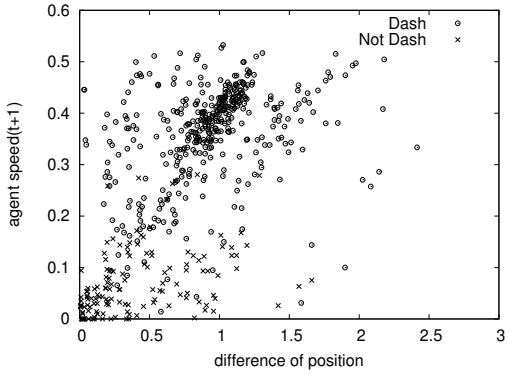


Figure 2: Dash

4.2 復調と解釈

ダッシュを行ったあとの、速度と位置の変移を図 2 に示す。意図的にダッシュ行動をせずともそれまでの慣性で運動が続くため、速度と位置の変移だけの観測ではダッシュをしたかどうかを、完全に弁別することはできないことが分かる。

いっぽうターン後の角度変化の頻度から、角度変化によってターンの有無を容易に判断できることが分かる。

4.3 通信の同期

行動はつねに行うものであり一方通信はそのうちの一部である。このため通信の開始と終了を知らせる通信の同期は重要である。RoboCup サッカーシミュレーションでは視線の方向から読み取ったアイコンタクトによって同期をとることができる。これは TurnNeck 動作により、通信をしたいエージェントの方角に視線を向けることで実現する。受信側は、自身を直視するエージェントがあれば、しばらく停止してアイコンタクトを成立させて同期を取る。

5 実験

RoboCup サッカーシミュレーションシステムをもちいて実験を行い B-MODEM 通信の効果を検証する。メッセージの変調方法として、予めメッセージを送るジェスチャー法と、混合方式を実装し、比較する。効率の計測は伝達所用時間にもとづいて行う。比較のため、音声通信 (Say プロトコル) による伝達を用いた場合も実施する。またメッセージ行動の系列を 2 種類用意してこれらの比較も行う。

5.1 場面設定

実験に用いた環境として図 3,4,5 に示した 3 つのシナリオを用意した。同一のスタート状態から一つを選びコミュニケーションにより行動の同期をはかる。

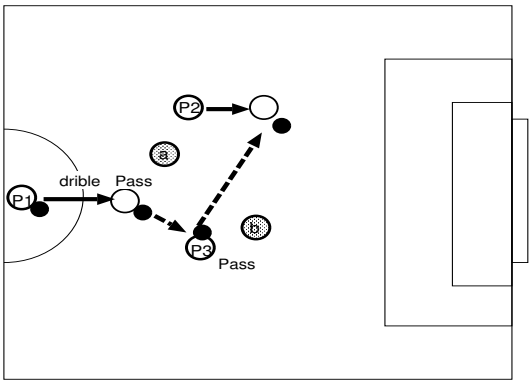


Figure 3: Scenario 1

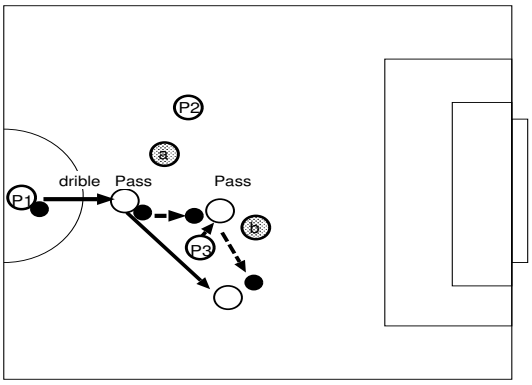


Figure 5: Scenario 3

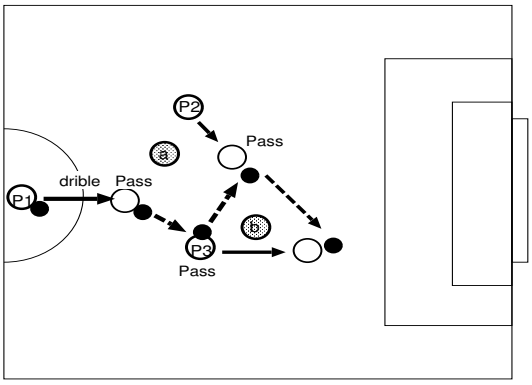


Figure 4: Scenario 2

Table 2: 目的行動を混合した伝達の結果

変調方式	シナリオ	受信成功率 (%) P2	受信成功率 (%) P3	伝達成功率 (%)
1	1	58	32	22
	2	52	81	50
	3	68	22	20
2	1	100	100	100
	2	100	100	100
	3	100	100	100

5.2 実験結果

各条件ごとに 150 回の試行を行った結果を、表 2 と図 6 にまとめる。

図 6 から、ジェスチャー方式すなわちメッセージ伝送をあらかじめ終了してから目的行動を行う場合には平均 21.8 ステップを必要することが分かる。これは目的行動のみの 16.0 と比較して 1.36 倍の時間消費である。

行動系列冒頭のアイコンタクトは、メッセージ伝送の主客を伝えるとともに、メッセージの開始の同期を伝える働きがある。目的行動とメッセージに共通部分 (D) があるため、この同期を取ることで解釈がより容易になっている。

いっぽうメッセージ混合方式の場合には、メッセージの一部をアイコンタクトおよび移動と重ねあわせ、平均ステップ数は 16.1　17.5 になりジェスチャー方式と比較して大幅な短縮ができている。

6 B-MODEM の効果

本稿で提案した手法は、物理的な運動と視覚認識を応用した通信を実現しており、以下のような利点がある。

1. 混信に対して強い耐性を持っている。
2. それ自体ノイズ源となる無線機器をロボットに搭載する必要が無い。
3. 視覚観測による通信では視線方向の指向性が高く、メッセージとしての行動を観測した時点で、能動的に通信相手が明らかになる。

一方で、以下のような問題点が明らかになった。

1. 多対多の環境では、無線では発信者、受信者の確認のコストが高くなる
2. 視覚により観測される物理情報とメッセージとして発行した動作とのギャップがあり、推定が１００％とはならない
3. 視覚センサの負担が大きい
4. 指向性が高いため、視野外のエージェントからのメッセージを受け取れない
5. 行動系列の認識が難しい。

7 まとめと今後の課題

従来、音声や電波無線によって担われていた通信路を、行動による発信と視覚的な受信に置き換え、混信の無いコミュニケーションを実現する B-MODEM を提案した。

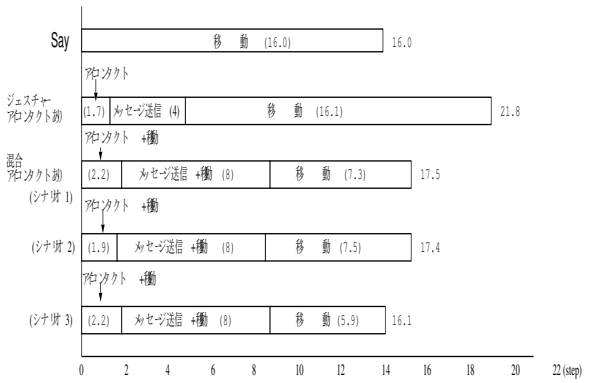


Figure 6: Mean time for execute a scenario

RoboCup サッカーシミュレーションをもちいた実験を行い、ターンという認識性の高い行動を、通信の主体とすることで、通信確度を高くすることができることを明らかにし、マルチエージェント環境における提案手法の有効性を示した。

行動を通信路とみなすとき、行動が慣性の法則など物理法則にしたがっているため、実際には状態を持つ通信路として扱う必要がある。このときの、通信路としての相互情報量と通信容量の定式化や、より使い勝手の良い変調方式を検討することが、今後の課題である。

参考文献

[Pet99] Peter, S., V. Manuela, and P. Riley: *RoboCup-98: Robot Soccer World Cup II*, Vol. 1604 of *Lecture Notes in Artificial Intelligence*, chapter Champion Teams: The CMUnited-98 Champion Simulator Team, pp. 61–76, Springer, 1999.

[岡 98] 岡, 武田, 西田: 人の集団とロボットとのノンバーバルコミュニケーションの実現法の提案, 第 16 回日本ロボット学会学術講演会予稿集, pp. 397–398, 1998.

[黒川 94] 黒川: ヒューマンコミュニケーション工学シリーズ ノンバーバルインターフェース, オーム社, 1994.

[山中 03] 山中, 岡本, 中西, 石田：非言語的合図に基づいて仮想都市を案内する対話エージェント, 情報処理学研究報告 グループウェアとネットワークサービス, Vol. 47, No. 16, pp. 89–94, 2003.

[西田 03] 西田：人間とロボットの意思疎通, 情報処理, Vol. 44, No. 12, 2003.

〒 162 東京都新宿区津久戸町 4-7 OS ビル 402 号室 03-5261-3401 Fax: 03-5261-3402

(本研究会についてのお問い合わせは下記にお願いします.)

AI チャレンジ研究会
主 査
奥乃 博
京都大学大学院 情報学研究科
知能情報学専攻 音声メディア分野
〒 606-8501 京都市左京区吉田本町
Tel: 075-753-5376 Fax: 075-753-5977
okuno@nue.org

担 当 幹 事
浅田 稔
大阪大学大学院 工学研究科
知能・機能創成工学専攻 創発ロボット工学講座
〒 565-0871 大阪府吹田市山田丘 2-1
Tel: 06-6879-7349 Fax: 06-6879-7348
asada@ams.eng.osaka-u.ac.jp

担 当 幹 事
光永 法明
株式会社国際電気通信基礎技術研究所
知能ロボティクス研究所
〒 619-0288 京都府「けいはんな学研都市」
光台 2-2-2
Tel: 0774-95-1401 Fax: 0774-95-1408
mitunaga@atr.jp

Executive Committee
Chair
Hiroshi G. Okuno
Dept. of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University
Yoshida-honmachi Sakyo-ku,
Kyoto, 606-8501, JAPAN

Secretary in Charge
Minoru Asada
Dept. of Adaptive Machine Systems
Graduate School of Engineering
Osaka University
2-1 Yamada-oka, Suita,
Osaka 565-0871, JAPAN

Secretary in Charge
Noriaki Mitsunaga
Dept. of Intelligent Robotics and Communication Laboratories
Advanced Telecommunications Research Institute International
2-2-2 Hikaridai, Keihanna Science City, Kyoto 619-0288, JAPAN

SIG-AI-Challenges web page; `http://winnie.kuis.kyoto-u.ac.jp/SIG-Challenge/`