

TWO METHODOLOGIES TOWARD ARTIFICIAL TACTILE AFFORDANCE SYSTEM IN ROBOTICS

M. Ohka¹, N. Hoshikawa¹, J. Wada¹, and H. B. Yussof²

Graduate School of Information Science, Nagoya University,
Furo-cho, Chikusa-ku, 464-8601, Nagoya, Japan, ohka@is.nagoya-u.ac.jp

²Faculty of Mechanical Engineering, Univesiti Teknologi MARA
Shah Alam, Selangor Darul Ehsan, 40450, Malaysia

Abstract-If the theory of affordance is applied to a robot, performing the whole process of recognition and planning is not always required in its computer. Since the tactile sensing of a robot is important to perform any task, we focus on tactile sensing and introduce a new concept called the artificial tactile affordance system (ATAS). Its basic idea is the implementation of a recurrent mechanism in which information obtained from the object and the behavior performed by the robot's inducing the next behavior. We intend to implement ATAS based on the following two methodologies: (1) after each rule is transformed into an algorithm, a program module is coded based on the algorithm; ATAS is composed of several program modules, and a module is selected from the set of modules based on sensor information; (2) a set of rules is expressed as a table composed of sensor input columns and behavior output columns, and the table rows correspond to rules; since each rule is transformed to a string of 0 and 1, we treat a long string composed of rule strings as a gene to obtain an optimum gene that adapts to its environment using a genetic algorithm (GA). For methodology 1, we established an ATAS composed of 3 to 5 modules to accomplish such tasks as object grasping, pick and place, cap screwing, and assembling. Using methodology 1, a two-hand-arm robot equipped with an optical three-axis tactile sensor performed the above tasks. For methodology 2, we propose the Evolutionary Behavior Table System (EBTS) that uses a GA to acquire the autonomous cooperation behavior of multiple mobile robots. In validation experiments, three agents equipped with behavior tables conveyed an object to a specified goal with higher scores than the four-agent condition. Since the redundant agent does not interrupt the other agents, the agent acquires the collective behavior of not interrupting other agents based on its environment information. Methodology 1 is very effective for such fine control as handling tasks of humanoid robots, and methodology 2 is very useful to obtain general robotic behavior that is suitable for the environment.

Index terms: Affordance, Tactile sensor, Three-axis, Collective robot, Evolution, Behavior, Genetic algorithm

I. INTRODUCTION

In the theory of affordance first introduced by Gibson and advanced by Norman, animal behaviors are assumed to be induced by environmental information called affordance, a neologism coined by Gibson [1]. If the theory is applied to robots, performing the whole process of recognition and planning is not always required in a robot's computer because it can decide its behavior using the affordance embedded by designers.

Tactile sensation possesses a salient characteristic among the five senses because it does not occur without interaction between sensory organs and objects. Touching the object induces both deformation of it and the sensory organ. Since the tactile sensing of robots is important to perform any task [2], in this paper we focus on tactile sensing and introduce a new concept called the artificial tactile affordance system (ATAS).

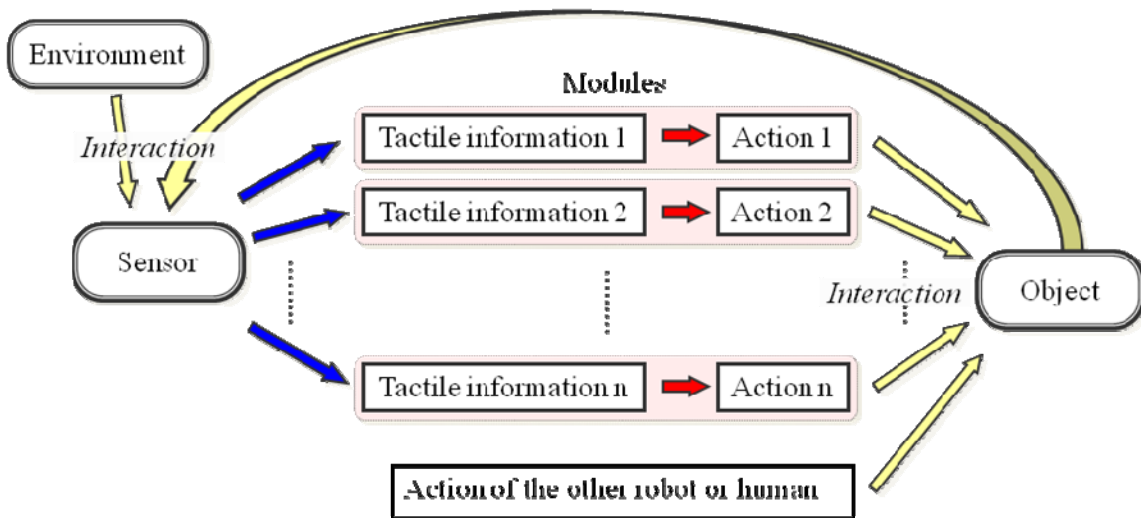


Figure 1. Artificial tactile affordance system (ATAS)

The basic idea of ATAS is the implementation of a recurrent mechanism, in which the information obtained from the object and the behavior performed by the robot itself induce the next behavior. To explain ATAS, we introduce the schematic block diagram shown in Fig. 1. When tactile information obtained from the environment is input into ATAS, a command for robot behavior is output, and the robot actuators are controlled based on that command. After that,

the environment is changed due to the deformation and the transformation caused by the robot behavior, and the result is sent as tactile information to the ATAS inlet by feedback loop.

In ATAS, a key point is producing a set of rules in which the sensor input patterns and behavior output patterns are if-clauses and then-clauses, respectively. This ATAS concept resembles an expert system in artificial intelligence [3] in which the matching degree between the fact selected by a fact database and the if-clause is evaluated; if the fact matches the if-clause, then the then-clause is displayed as a temporal conclusion, and simultaneously the then-clause is added to the fact database. The biggest difference between ATAS and an expert system is that in the latter, the fact database is possessed inside a computer but in ATAS the whole environment is treated as the fact database.

Since ATAS is categorized as a behavior-based control system, it more closely resembles subsumption architecture (SSA) [4][5]. Although in SSA each connection between sensors and actuators has priority, in ATAS no modules have priority and they are arranged in parallel form. In each module, we can include a relative complex procedure if we need it. Therefore ATAS is suited to such rather complex tasks as assembly and walking on uneven loads.

We implement ATAS based on the following two methodologies:

Methodology 1: after each rule is transformed into an algorithm, a program module is coded based on the algorithm, ATAS is composed of several program modules, and a module is selected from the set of modules based on sensor information to perform a specified behavior intended by a designer.

Methodology 2: a set of rules is expressed as a table composed of sensor input columns and behavior output columns, and a row of the table corresponds to a rule. Since each rule is transformed to a string of 0 and 1, we treat a long string composed of rule strings as a gene to obtain an optimum gene that adapts to the environment using a genetic algorithm (GA) [6].

If we adopt methodology 1 as the ATAS mechanism, an ATAS designer should produce program codes to implement program modules to realize such rather complicated tasks as robotic manipulation [7] and gating [8]. If we adopt methodology 2 as the ATAS mechanism, the robot can automatically obtain the relationship between sensor input and behavior output patterns after it has committed to the environment. However, a tremendous number of iterations are required to realize manipulation and gating because of the large number of degrees of freedom (DOF).

The final goal of this study is producing a hybrid system based on both methodologies 1 and 2 to solve the above two-sided problem. For such precise tasks as manipulation, the designer produces program modules based on methodology 1 and considering individual situations. In cases where qualitative behavior is regarded as important, such as collective robots, the designer adopts methodology 2 as a rule base in ATAS.

In this paper, we investigate methodologies 1 and 2 using examples. First, for methodology 1, we treated an assembly task for articulated-fingered hand robots and prepared such simple modules as enhancing grasping force against finger slippage and opening the hand against specified directional stimulus. For methodology 2, we developed a behavior rule for collective robots [9]-[14] to omit supervised learning. We borrowed inspiration from social insects that can adapt themselves to several environments. Based on our idea, we introduce the Evolutionary Behavior Table System (EBTS) in which feedback structure is achieved by evolutionary computation and a behavior simulator to adapt the system to the environment.

We evaluated both methodologies 1 and 2 using experiments and numerical simulation. On the basis of these results, we propose a hybrid system of methodologies 1 and 2 that combines their merits.

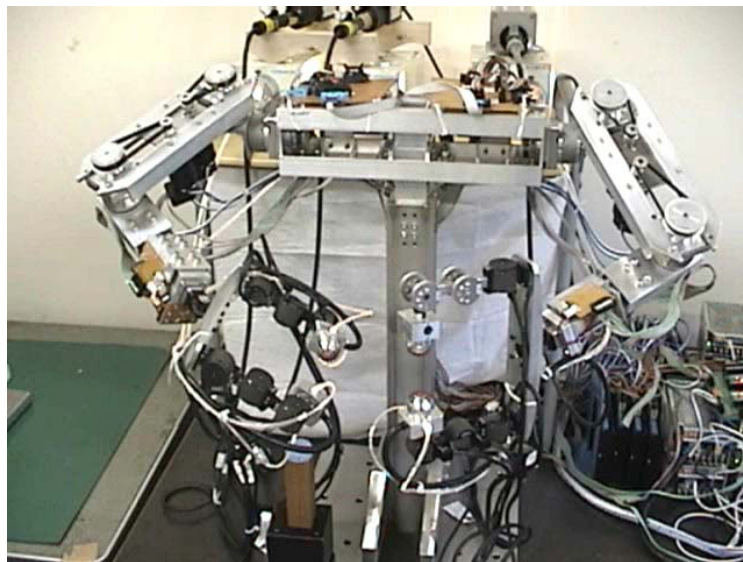


Figure 2. Two-hand-arm robot equipped with optical three-axis tactile sensors

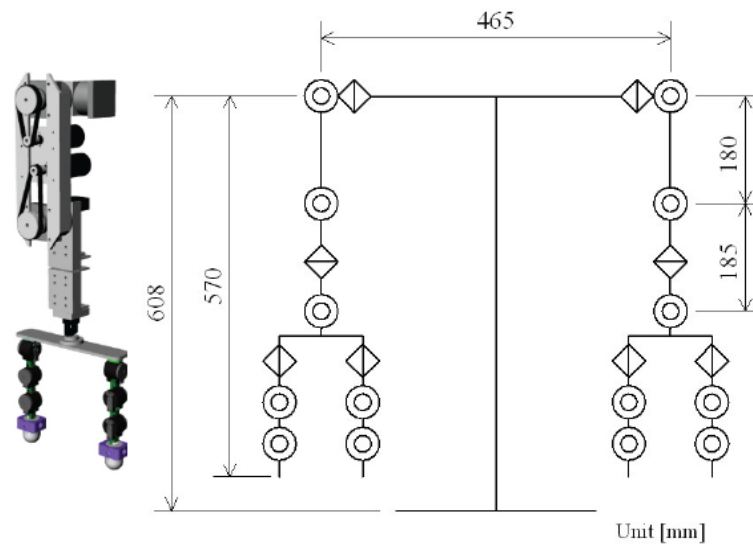


Figure 3. DOFs of two-hand-arm robot

II. METHODOLOGY 1: PROGRAM MODULES

a. Two-hand-arm Robot Equipped with Optical Three-axis Tactile Sensors

Since methodology 1 is applied to fine robotic motion, we adopted a humanoid robot as an example. Fig. 2 shows a two-hand-arm robot, which is an advance from a one-hand-arm robot presented in previous articles [15]. This two-arm-hand arm robot is not a complete humanoid robot but possesses the minimum functions for it. Thus we can discuss some basic feasibility studies using it.

On each fingertip, it has a novel, optical three-axis tactile sensor [16][17] that can obtain not only normal force distribution but also tangential force distribution. Although ordinary tactile sensors can only detect either normal or tangential force, since this tactile sensor can detect both, the robot can use the sensor information as an effective key to induce a specified behavior.

Figure 3 shows the structure of the present robot; the arm system's DOF is 5; each finger's DOF is 3. To compensate for the lack of arm DOF, this robot uses its finger's root joint as its wrist's DOF.

b. Behavior Induced by Interaction Between Environment and Robot

(1) Object Grasping

Figure 4 shows a typical program module type ATAS. Generally, in ATAS, n program modules are included. To accomplish object grasping tasks with ATAS, we used two basic behaviors: 1) grasping force enhanced by slippage to prevent the instability of grasping an object; 2) grasped object released based on the specified direction of the tangential force to prevent from crushing it.

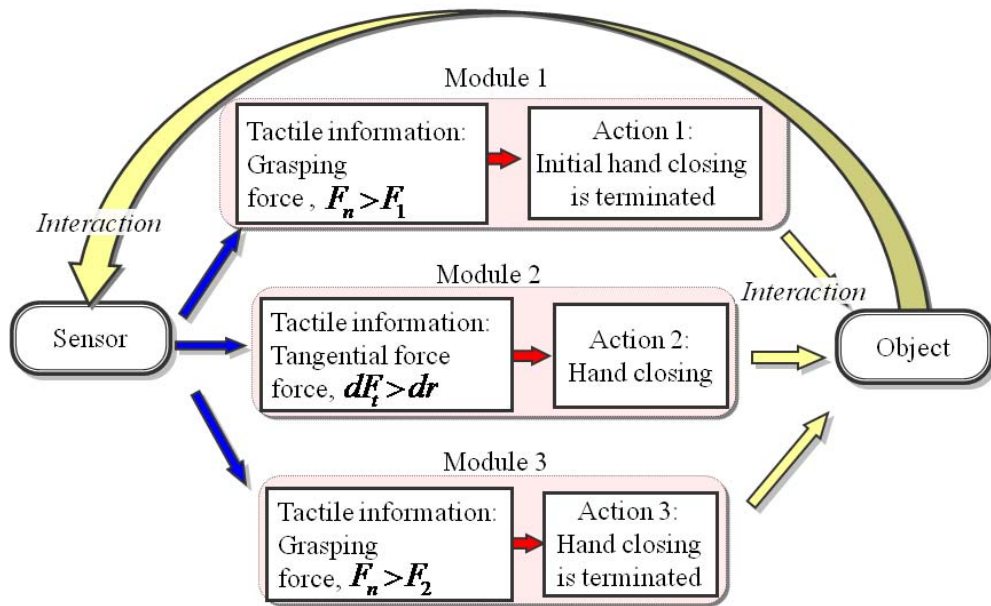


Figure 4. Object grasping task

We explain the simplest case so that readers can easily understand our methodology. If a robot grasps a slippery object, it must enhance the grasping force when slippage occurs. If the grasped object is very fragile, the robot should treat two contrary events: grasping and releasing.

To solve this contradiction, we produce three program modules. Module 1: if the grasping force reaches threshold of minimum grasping force F_1 when it first touches the object, the grasping force enhancement behavior is stopped (grasping force, $F_n > F_1$); Module 2: if partial slippage occurs, grasping force increasing behavior continues, and otherwise the behavior is stopped (slippage is estimated by whether tangential force increment dF_t exceeds threshold dr :

$dF_t > dr$); Module 3: if grasping force exceeds threshold of maximum grasping force F_2 , grasping force enhancement is immediately stopped to prevent crushing the object (grasping force, $F_n > F_2$).

Threshold of maximum grasping force F_2 should be changed based on the object's hardness. For example, since a hard object like a wood block is very tough, the threshold can be relative large. On the other hand, for a paper die, since the robot easily crushes it with only tiny grasping force, the threshold must be very low.

When the robot starts to grasp an unknown object, its fingertips approach it with at a very slow speed of 2 mm/sec. After the first touch during 100 msec, the robot measures the object's hardness. Based on the hardness, threshold F_2 is specified. Our previous studies evaluated and confirmed the above strategy. Module 2 worked just after 100 msec. Once grasping force reached F_1 , since a drastic change of grasping force is basically not needed, module 2 worked. The robot can grasp the object with suitable normal force that generated maximum static friction force because it adjusts the grasping force. In the following section, we will describe other cases of program modules to perform assembly tasks.

(2) Cap Screwing Task

A cap screwing task is performed based on modules 2 and 3 described in the previous section. The initial trajectory for the fingers is applied to the robot to start cap screwing; the termination condition of the cap screwing task must be determined. For these behaviors, we added two modules to the previous modules in Fig. 4:

Module 0 provides an initial cap screwing task in which the fingers follow the square trajectories in Fig. 5. This behavior is repeated until the behavior provided by Module 4 is defined by the following. Since the finger trajectory is intersected at the cap's contour, the finger tips slip on its surface. If slippage occurs, increment of the shearing force is observed. At this moment, module 1 is activated to enhance the grasping force.

Module 4 decides when to terminate the screwing task. Empirically, we know that much slippage occurs when the cap is tightened. If increment of shearing force dF_t exceeds 1.4 times of dF_{\max} , which is the maximum of dF_t during the first touch, the finger motion is terminated.

The fingertip trajectory is shown in Fig. 6. Modification of the initial trajectory is saturated after closing the cap. Although the initial finger trajectory is a rough rectangle decided to touch and

turn the cap, a segment of it is changed from a straight line to a curved line to fit the cap contour. The curved trajectory is not provided, but it is obtained through this task.

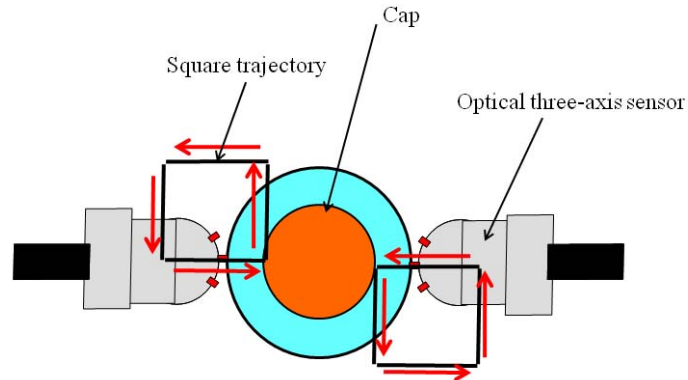


Figure 5. Initial square trajectory

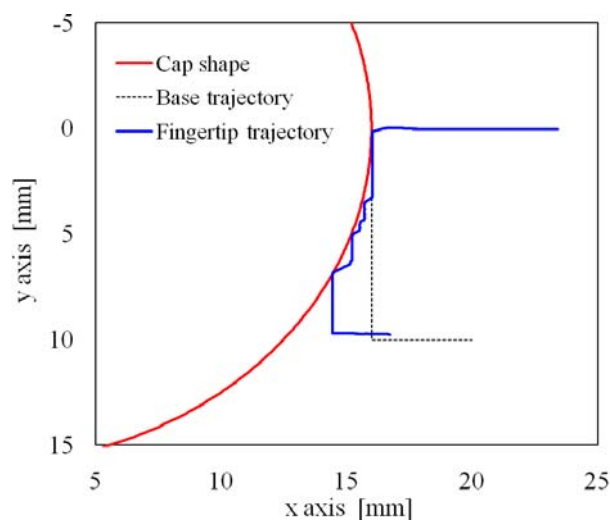


Figure 6. Modified trajectory based on interaction

(3) Picking and Placing Task

Since the picking task is accomplished by the object grasping explained by Fig. 4, releasing object behavior is added to accomplish the placing task. Due to this, modules 0 and 4 are replaced by new modules 0' and 4.

Module 0' conveys and sets an object down to a specified plane position after it picks the object up from a specified position. Laying down is terminated by the activation of module 4.

Module 4 releases the object when the tactile sensors catch the upward tangential force. If the robotic hand feels upward tangential force while moving down, the object's bottom has touched a table or a floor. In this module, the upward tangential force induces opening the hand.

Using this ATAS, the picking and placing task shown in Fig. 7 is accomplished.

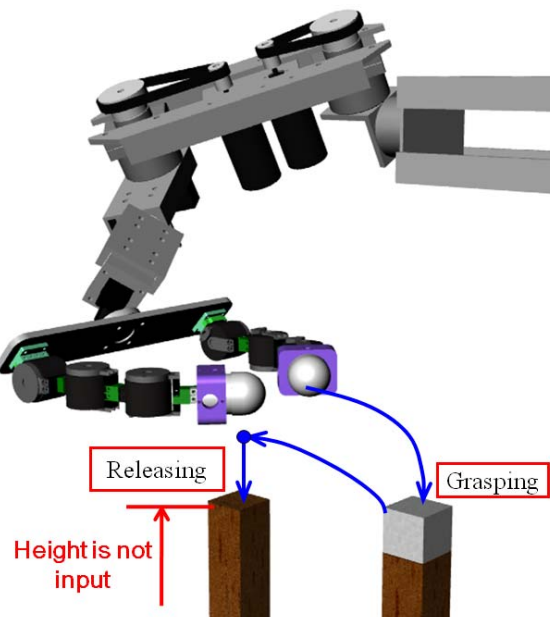


Figure 7. Picking and placing task

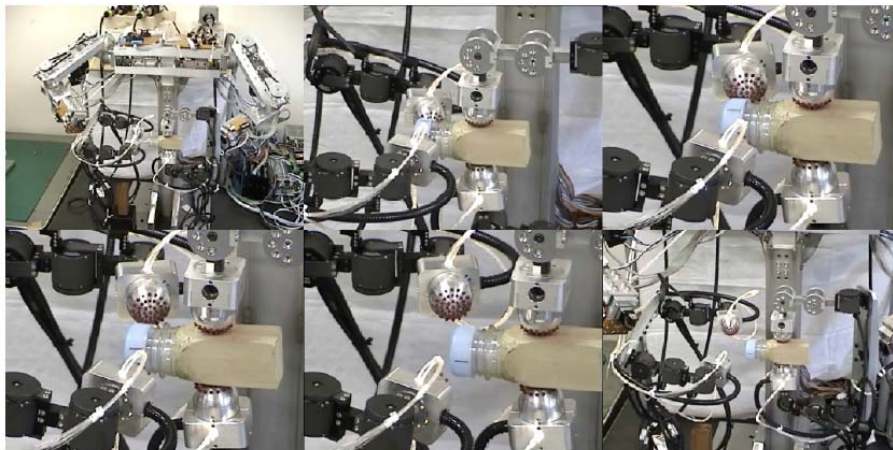


Figure 8. Assembly task

c. Assembly Task

If modules 0, 0', 1, 2, 3, and 4 are combined, the robot can perform simple assembly tasks. For this task shown in Fig. 8, the pick and initial finger motion along the square trajectory provided

by module 0 is started after the placing behavior induced by module 0' is completed. The robot screws the cap onto the bottle grasped by its left hand.

III. METHODOLOGY 2: EBTS

a. Behavior Acquisition Method

(1) Behavior table

We must define the relationship between environmental information and the behavior rules in a single-layered reflexion behavior system to control the agent's behavior. Stimulus and response are defined by sensor truth and motor state tables, respectively, which are shown in Table 1. These tables will be prepared for the mobile robot agent equipped with five sensors and two motors as introduced in the next section.

Table 1. Behavior tables

Num	Sensor value					Motor value			
	S1	S2	S3	S4	S5	L1	L2	R1	R2
1	0	0	0	0	0	1	0	0	1
2	0	0	0	0	1	1	1	1	0
3	0	0	0	1	0	0	0	0	0
⋮			⋮			⋮		⋮	
⋮			⋮			⋮		⋮	
⋮			⋮			⋮		⋮	
32	1	1	1	1	1	0	0	0	1

L1, R1	L2, R2	Motor state
0	0	Stop
0	1	Medium Speed
1	0	Medium Speed
1	1	High Speed

Truth table for sensor output and motor output

Motor state table

S1-S5 show the status of the five sensors mounted on an agent. L1-L2 and R1-R2 show the status of the left and right wheel motors, respectively. Because the sensor status is described by Boolean values (1 or 0), the number of total patterns of sensor status is $2^5 = 32$. An agent refers to the current values of L1, L2, R1, and R2 of Table 1 in every set period to decide its behavior.

(2) Behavior Acquisition Method

The behavior table is composed of the truth and motor status tables of the sensors. These tables can be expressed with a one-dimensional array like genes because each of the truth-values is

Boolean data. Therefore, the behavior table can be designed as a model of a simple genetic algorithm (SGA) comprised of Boolean data [6]. Since the behavior table is evolvable, we call it the Evolutionary Behavior Table System (EBTS). The design procedure of the behavior table using SGA is shown in the following.

First, we design a genotype array that has information about the behavior table. The length of the genotype array is shown with the next formula:

$$G = 2^{S_i} \times \sum_{j=1}^n M_j E_j, \tag{1}$$

where the number of sensors, motors, and the bit number of output gradation are S_i , M_j and E_j , respectively. Sensor patterns are calculated as 2^{S_i} : n is the number of actuator kinds.

The genotype model used for numerical experiments described in the subsequent chapter is shown in Fig. 9. The agent possesses five sensors and two-wheel motors of 2-bit gradation and has a length of $32 \times 2 \times 2 = 128$ bits as gene data information.

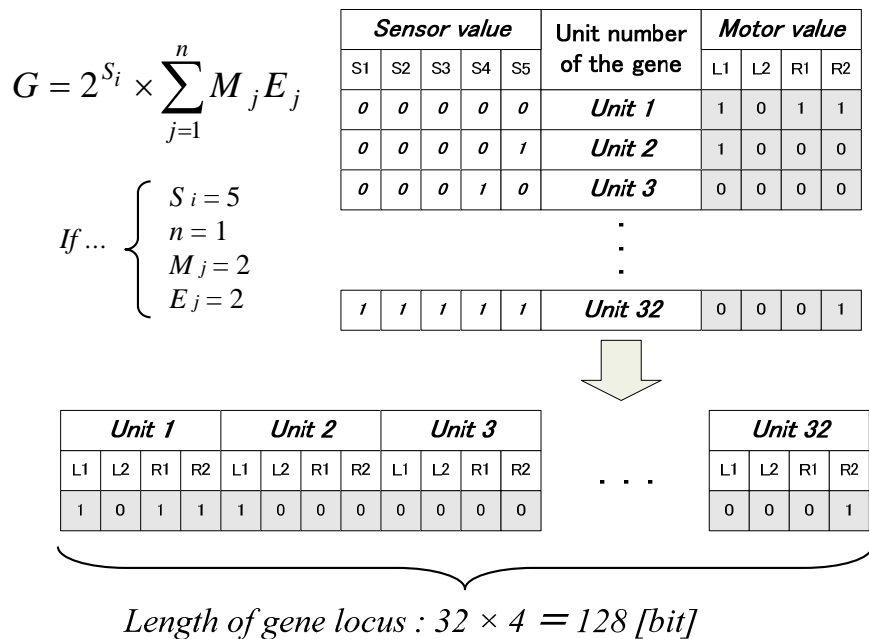


Figure 9. Genotype model

The agent's fitness value is calculated in a simulator, which is equipped with internal functions that can evaluate the efficiency and the task accuracy degrees of the agent. Then the simulator generates a behavior table from the genotype array of a one-dimensional vector composed of G

elements. The agent is evaluated on the basis of the task achievement degree in the simulator field during a specified period. The evaluation value obtained by this simulation is sent to the calculating system for genetic algorithms as fitness.

b. Simulation Procedure

(1) A. Simulator Design for Collective Robots

As a collective task, we adopted an object transportation problem because it can be easily compared to previous research. In the task, a circular mobile robot equipped with IR and photo sensors transports an object-emitting radiation light to a goal-emitting radiation light, as shown in Fig. 10. We presume that the circular mobile robot transports the object to the goal, as shown in Fig. 11.

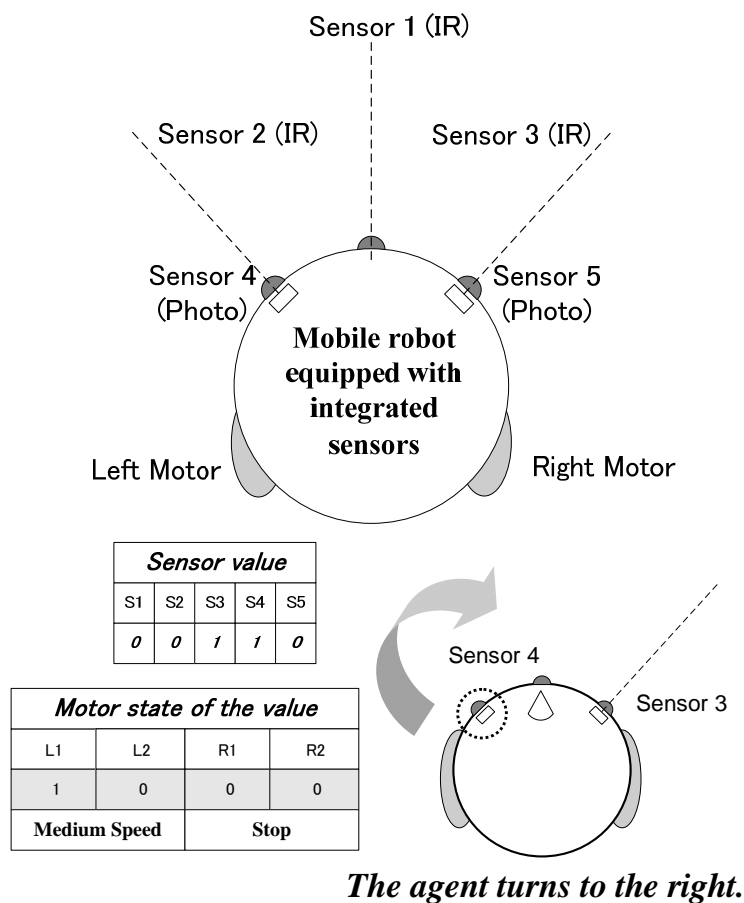


Figure 10. Model of circular mobile robot

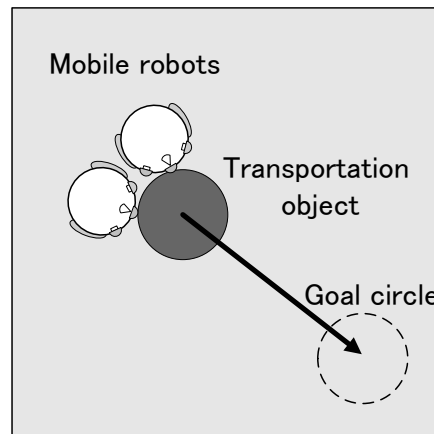


Figure 11. Object transportation problem

A simulator is composed of a map field and objects on it. The objects are categorized into two types: an autonomous mobility agent defined as a robot and a non-autonomous object defined as the transportation object. The agent can be equipped with multiple sensor inputs and behavior outputs as its module, which is formed to function as a suitable module based on the assumptions of numerical experiments.

In this object transportation problem, the agent is equipped with three IR sensors and two photo sensors that have a fixed sensing range. The IR sensors detect walls and other objects by Boolean values in the map field. The photo sensors detect the light intensity of the fixed range that is irradiated by the transportation object and the goal. An overall view of the evolutionary behavior table is shown in Fig. 12. The optimization procedure of the genetic algorithms is summarized as follows:

- 1) The population of the random gene data is produced as an initial value.
- 2) The evolutionary computation engine sends gene data to the simulator to evaluate the gene fitness.
- 3) Elite genes are selected based on the fitness.
- 4) A set of individuals is chosen based on roulette wheel selection.
- 5) A pair of individuals is selected and used for uniform crossover.
- 6) The newborn children from the pair mutate under a certain probability.
- 7) The children's gene data are sent to the simulator to evaluate their fitness.
- 8) The fitness of the elite group is compared with that of its children group. The group of the next generation population is selected from the high score group in descending order.

- 9) If it is not a final generation, it returns to the above procedure 3).
- 10) Evolutionary computation is finished.

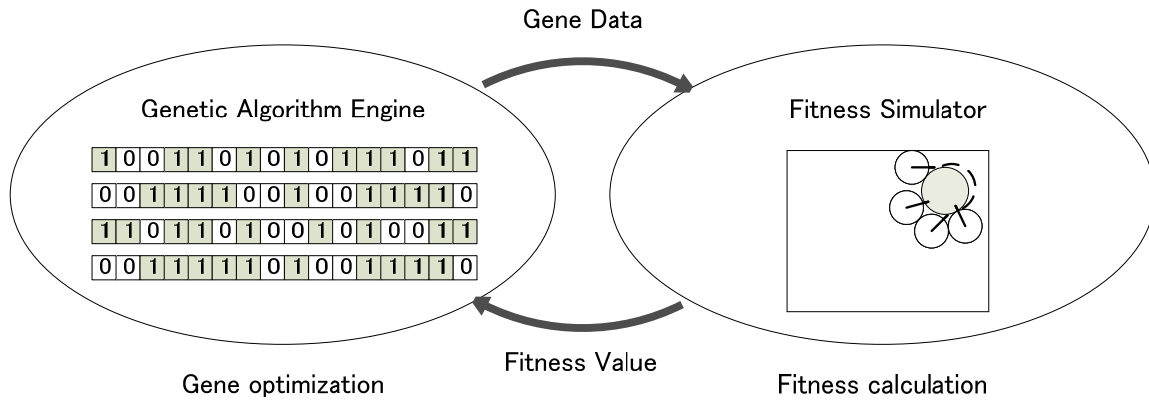


Figure 12. Evolutionary Behavior Table System (EBTS)

(2) Fitness Evaluation by Simulator

After the agents performed object transportation, the achievement degree of the task was evaluated as a fitness value of the gene. Evaluation functions for task achievement are composed of transportation accuracy value as E_1 , evaluated from the geometrical relationship between agents and the transportation object, and transportation efficiency value as E_2 , decided by reaching the goal. Evaluation function E_1 is defined as follows:

$$E_1 = \frac{K_1}{\|\mathbf{c}_A - \mathbf{c}_B\|} \quad (2)$$

where \mathbf{c}_A and \mathbf{c}_B are position vectors of the object and the goal, respectively. Coefficient K_1 is assumed to be 1,000.

On the other hand, evaluation function E_2 is defined by

$$E_2 = \frac{K_2}{t} \quad (3)$$

using spent time to transfer the object to the goal area. In this study, we assumed as $K_2 = 600$.

The fitness value of the genetic algorithm is calculated as the summation of E_1 and E_2 . In the fitness calculation, the agent often accidentally pushed the transportation object into the goal. To avoid such accidents, we divided the simulation time into task execution and task evaluation and

compared the positioning between the agent and the transportation object at the termination of the execution time with the termination of the evaluation time.

(3) Simulator Conditions in Numerical Experiments

The map field of this paper is assumed to be a Gaussian plane of 800×800 [pixel²] with an origin at coordinates (0, 0) in a series of numerical experiments. The transportation object and the agents are set at the origin and specified positions, respectively. In the left of Fig. 13, four agents are placed near each corner of the rectangular field, and the agents are numbered 1-4. To relax over-adaptation in the task procedure, eight goal positions are assumed, as shown in the right of Fig. 13.

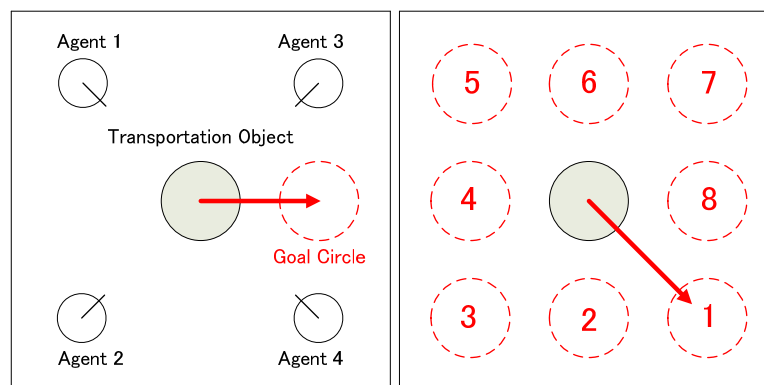


Figure 13. Field initialization and goal location

An individual tries object transportation eight times for each simulation. Evolution calculation, which continues for 2,000 generations, is repeated 10 times for each numerical experimental condition. The number of individuals, keeping the elite number, and mutation rate per 1-bit of gene are 10, 2, and 1%. The execution time of the simulation is divided into 180 frames of the task execution time and 90 frames of task evaluation time per goal. Since the simulation is performed for eight goals, 2,160 frames are calculated. One hundred points for one goal are provided to evaluate the largest score.

c. Numerical Experimental Result

(1) Number of Agents and Evolution

Figure 14 shows the relationship of agents 1-4 between maximum fitness and generation. The fitness of the multi-agent shows more than five times the difference compared to that of a single

agent. The single agent's efficiency value is lower than the others because it can't perform the collective task. Based on the assumptions of this numerical experiment, however, and since the single agent might transport the object to the goal, failure in the single agent's transportation task seems to be caused by failing to acquire the transportation task oriented to the general purpose.

In Fig. 15, the white circles numbered "1" show agents. Non-numbered white circles and open circles are the transportation objects and goals, respectively. As shown in Fig. 15, the low score in Fig. 14 is caused by an unlimited loop in the agent's circling behavior around a certain position because the only agent in the field can't escape from the loop once it is trapped in a deadlock. Figure 15 means that the agent is near the light source under the initial state, and the agent continues to circle counter-clockwise inside the light source range of the transportation object.

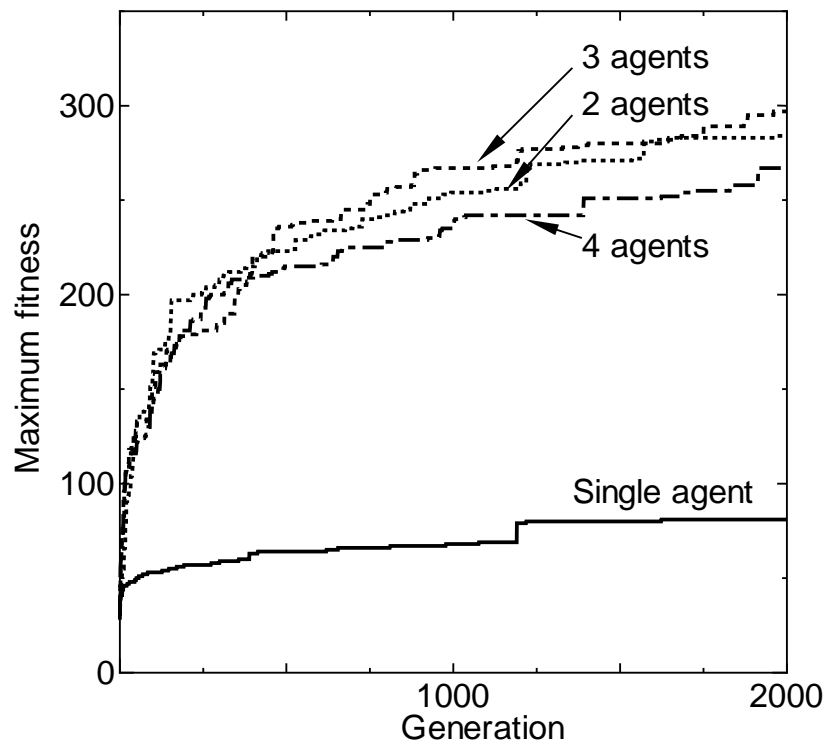


Figure 14. Relationship between maximum fitness and generation

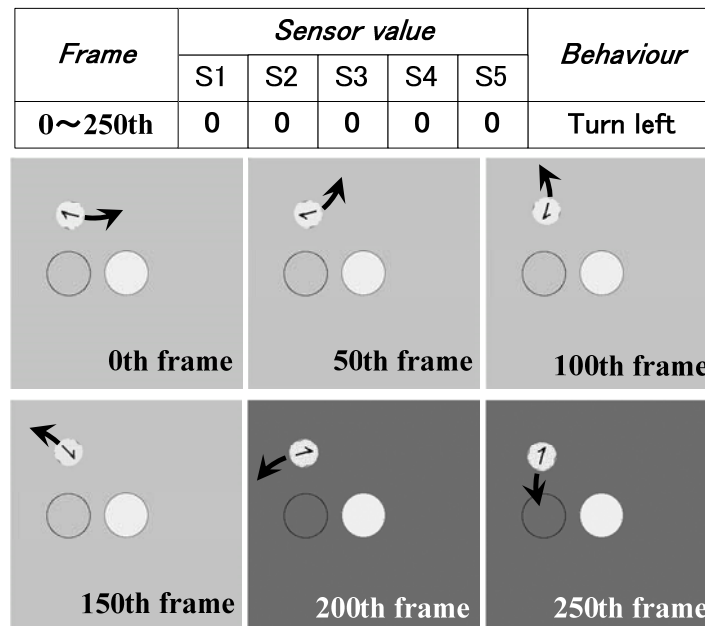


Figure 15. Single agent evolution result; agent is moving counter-clockwise around the same position

We found that an agent can escape the loop in a multi-agent environment that possesses more than two agents, as shown in Fig. 16, with help from the other agent (numbered as “2”) even if the agent falls into the loop, because some sensors of the agent may react to the other agent. One difference between the conditions of Figs. 15 and 16 reflects whether the other agent exists except for itself in the field. Therefore, acquiring the collective ability of the multi-agent by supporting the other agent and the multi-agent’s environment can be considered evolutionary pressure.

On the other hand, we confirmed that groups of three and two agents obtained higher scores than four agents, as shown in Fig. 14. This result seems contrary to the improvement of parallel efficiency by agent increases. We examined simulation movies of four agents to analyze the inclination, while these are eliminated in this paper. In the simulation movie results, we frequently observed that agents obstructed themselves from each other. In the designed field and in the condition of this numerical experiment, three agents are optimum to suitably carry out the object transportation task.

Frame		Sensor value					Behaviour
		S1	S2	S3	S4	S5	
0th	Agent 1	0	0	0	0	0	Go straight
	Agent 2	0	0	0	0	1	Turn right
100th	Agent 1	1	0	1	0	1	Turn right
	Agent 2	0	0	0	1	0	Turn right
200th	Agent 1	0	0	1	1	1	Turn right
	Agent 2	0	0	0	1	0	Turn right
219th	Agent 1	0	0	1	1	1	Turn right
	Agent 2	1	0	0	0	0	Turn left
225th	Agent 1	1	0	1	0	1	Turn right
	Agent 2	1	0	1	0	1	Turn right
250th	Agent 1	1	0	1	0	1	Turn right
	Agent 2	1	0	0	0	1	Turn left

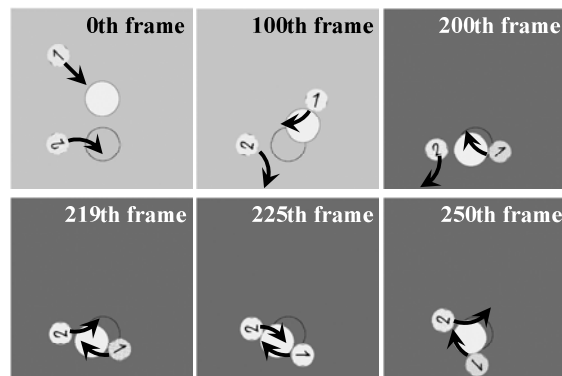


Figure 16. Two-agent evolution result; although second agent continues to turn clockwise before approaching object, it terminates its turn to cooperate with first agent conveying object

(2) Behavior Acquisition Accommodated to Environmental Condition

Figure 17 shows two examples (Case (1) two agents carry the object, while other two agents avoid the task; case (2) three agents carry the object, while an agent avoids the task) successfully completed the object transportation in the four-agent condition. In these examples, robots are denoted by circles numbered “1”, “2”, “3” and “4”.

As shown in Fig. 17, although the agents collectively transfer the object, the transportation is not always performed by all agents but from two to three agents. This is the reason that two and three agents obtain higher scores than four agents in Fig. 14. To successfully transfer with four agents, the redundant agent has to acquire the behavior for not interrupting the other agent’s task.

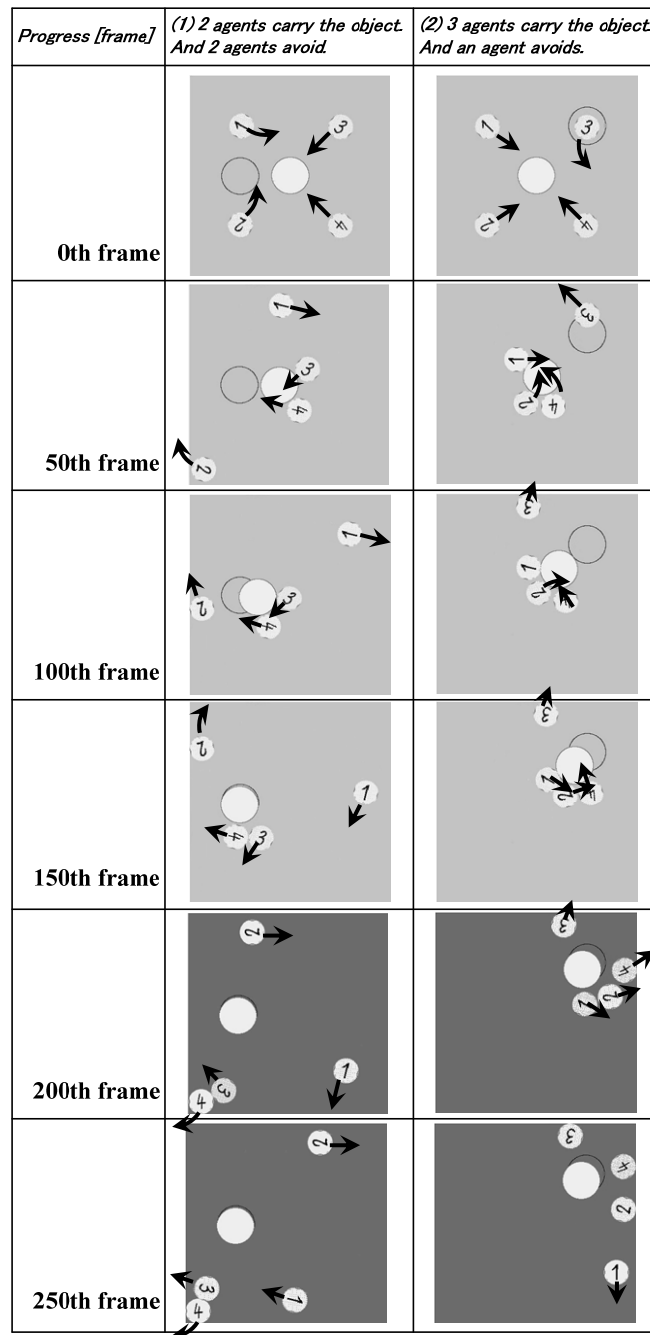


Figure 17. Four agents acquired a cooperative behavior as the collective task. They carry a transport object, or avoid other agents and the transport object not to disturb other agents' task

It is noted that the agents acquire collective behavior in the case (1) in Fig. 17. In this case, two agents convey the objects, while other two agents avoid the former two agents not to disturb the agents' task. Contrarily, in the case (2) Agent 3 is staying on the goal at first. When the photo-

sensors 4 and 5 of Agent 3 catch the goal light to immediately avoid passing the goal, it takes the action “going right-forward”. At the result of acquiring the behavior, although Agent 3 does not push the object, and it does not interrupt the other agents 1, 2 and 4 in escaping from the light, which is emitted from the goal. The above result shows that the agent acquires the collective behavior of not interrupting other agents from the environment information to enhance the efficiency of the collective task.

(3) Acquisition of Autonomy Cooperation Behavior of Multi-agents

Finally, we will discuss the efficiency of acquiring autonomy cooperation behavior of multi-agents in this EBTS using GA. Population in GA was ten and elitism selections were two in the numerical experiment. In this numerical experiment, the number of simulation trials was 20,000, as a result of optimizing the gene data until 2,000 generations. This final truth table obtained by the gene data doesn't always assure an optimum solution, but the calculation cost is reduced from 3.4×10^{38} to 2.0×10^4 because the combination numbers of input and output patterns are calculated as $2^{128} \cong 3.4 \times 10^{38}$ according to Eq. (1). If we used the top-down methodology, 3.4×10^{38} trials are needed to specify the optimum pattern. Therefore, we accomplish the automatic design of cooperative behavior of multi-agents for collective tasks.

IV. CONCLUSION

We implemented an artificial tactile affordance system (ATAS) based on the following two methodologies: (1) after each rule is transformed into an algorithm, a program module is coded based on the algorithm; ATASs are composed of several program modules, and a module is selected from the set of modules based on sensor information; (2) a set of rules is expressed as a table composed of sensor input columns and behavior output columns, and a row of the table corresponds to a rule; since each rule is transformed to a string of 0 and 1, we treat a long string composed of rule strings as a gene to obtain an optimum gene that adapts to the environment using a genetic algorithm (GA).

For methodology 1, we established ATASs composed of 2 to 4 modules to accomplish such tasks as object grasping, picking and placing, cap screwing, and assembling. Using methodology 1, a two-hand-arm robot equipped with an optical three-axis tactile sensor performed the above tasks. For methodology 2, we propose the Evolutionary Behavior Table System (EBTS) using a genetic

algorithm (GA) to acquire the autonomous cooperation behavior of multiple mobile robots. In validation experiments, three agents equipped with the behavior table conveyed an object to a specified goal with higher scores than the four-agent condition. Since the redundant agent does not interrupt the other agents, the agent acquires the collective behavior of not interrupting other agents based on the environment information.

Methodology 1 is very effective for such fine control as handling tasks of humanoid robots, and methodology 2 is very useful to obtain general robotic behavior that is suitable for its environment. Therefore we are considering a hybrid system including methodologies 1 and 2 to provide behaviors for collective humanoid robots. For the macro motion of a humanoid robot such as locomotion, the behavior is obtained as a circle by approximating it using methodology 2, but for such fine motion as manipulation, the behavior is obtained as kinematic data using methodology 1.

REFERENCES

- [1] J. J. Gibson, "The Ecological Approach to Visual Perception," Houghton Mifflin Company, 1979.
- [2] M. Ohka, "Robotic Tactile Sensors," Encyclopedia of Computer Science and Engineering, Wiley Encyclopedia of Computer Science and Engineering, 5-Volume Set, Editor: Benjamin, W. Wah, pp. 2454 – 2461 (Vol. 4), 2009.
- [3] P. H. Winston, "Artificial Intelligence (second edition)," Addison-Wesley, pp. 159-204, 1984.
- [4] Rodney A. Brooks, "A robust layered control system for a mobile robot," IEEE Journal of Robotics and Automation, RA-2(1), pp. 14-23, 1986.
- [5] Rodney A. Brooks, "Intelligence without representation," Technical Report MIT AI Lab, 1988.
- [6] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison Wesley, 1989.
- [7] M. Ohka, N. Morisawa, and H. B. Yussuf, Trajectory Generation of Robotic Fingers Based on Tri-axial Tactile Data for Cap Screwing Task, 2009 IEEE International Conference on Robotics and Automation, pp. 883-888, 2009.
- [8] H. Yussuf, M. Ohka, M. Yamano, and Y. Nasu, Collision Checking Strategy in Biped Humanoid Robot Navigation Toward Operation in Unseen Environment, Proc. of Tenth IASTED Inter. Conf. Control and Applications, pp. 48-54, 2008.
- [9] Valentino Braitenberg, "Adaptation in Natural and Artificial Systems," MIT Press, 1984.

- [10] Toshio Fukuda, Tsuyoshi Ueyama, and Fumihito Arai, "Control strategy for a network of cellular robots," IEEE International Conference on Robotics and Automation, pp. 1616-1621, 1991.
- [11] Ronald C. Arkin, "Cooperation without communication: Multiagent schema-based robot navigation," Journal of Robotic Systems, 9(3), pp. 351-364, 1992.
- [12] Maki K. Habib, Hajime Asama, Yoshiki Ishida, Akihiro Matsumoto, and Isao Endo, "Simulation environment for an autonomous and decentralized multi-agent robotic system," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1550-1557, 1992.
- [13] C. Ronald Kube and Hong Zhang, "Controlling Collective Tasks With An ALN," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 289-293, 1993.
- [14] Tsutomu Hoshino, Daisuke Mitsumoto, and Tohru Nagano, "Evolution of Robot Behavior and Its Robustness," Transactions of the Society of Instrument and Control Engineers (Transactions of SICE), Vol. 33, No 6, 533-540, 1997.
- [15] H. B. Yussuf, J. Wada, and M. Ohka, Object Handling Tasks Based on Active Tactile and Slippage Sensations in Multi-Fingered Humanoid Robot Arm, 2009 IEEE International Conference on Robotics and Automation, pp. 502-507, 2009
- [16] M. Ohka, H. Kobayashi, J. Takata, and Y. Mitsuya, An Experimental Optical Three-axis Tactile Sensor Featured with Hemispherical Surface, Journal of Advanced Mechanical Design, Systems, and Manufacturing, Vol. 2, No. 5, pp. 860-873, 2008.
- [17] M. Ohka, J. Takata, H. Kobayashi, H. Suzuki, N. Morisawa, and H. B. Yussuf, Object Exploration and Manipulation Using a Robotic Finger Equipped with an Optical Three-axis Tactile Sensor, Robotica, Vol. 27, pp. 763-770, 2009.