<table>
<thead>
<tr>
<th>項目</th>
<th>内容</th>
</tr>
</thead>
<tbody>
<tr>
<td>タイトル</td>
<td>コントロールオブオシロカルアグリゲーション:革新的制御手法を用いた自律システムにおける安定的動作の生成</td>
</tr>
<tr>
<td>著者</td>
<td>山内 翔</td>
</tr>
<tr>
<td>発行日</td>
<td>2014-09-25</td>
</tr>
<tr>
<td>DOI</td>
<td>10.14943/doctoral.k11520</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2115/57150">http://hdl.handle.net/2115/57150</a></td>
</tr>
<tr>
<td>ファイル情報</td>
<td>Sho_Yamauchi.pdf</td>
</tr>
</tbody>
</table>

Hokkaido University Collection of Scholarly and Academic Papers: HUSCAP
Control of Oscillator Aggregation for Generating Homeostatic Behavior in Autonomous Systems

Sho Yamauchi

Graduate School of
Information Science and Technology
Hokkaido University, Japan

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Information Science and Technology, Hokkaido University,
# Contents

1 Overview of intelligent robot controls  
1.1 Introduction .............................................. 12  
1.2 study of intelligent robot control .......................... 12  
1.3 Machine learning and artificial neural network  .......... 13  
1.4 Evolutionary robotics ..................................... 13  
1.5 Global and local optimization for complicated task ......... 13  
1.6 Human robot interaction .................................. 14

2 Concept of robot control with structured relationship of parameters  
2.1 Introduction .................................................. 15  
2.2 Synchronization in living organisms .......................... 15  
2.3 Concept of homeostasis in robotic system ................. 16  
2.4 Relationships between parameters .......................... 17  
2.5 Conclusion .................................................... 17

3 Structured relationship in flocking algorithm  
3.1 Introduction .................................................. 19  
3.2 Overview of flocking algorithms ............................ 19  
3.3 Flocking algorithm as consensus problem ................. 20  
3.4 Flocking algorithm .......................................... 21  
3.5 Extended flocking algorithm ............................... 24  
3.6 Expression of relationships in flocking algorithm .......... 26
6.3.4 \( f_{\text{prob}} \) .................................................. 69
6.4 Related study ....................................................... 69
6.5 Experimental result ............................................... 71
   6.5.1 Isochron ...................................................... 71
   6.5.2 Result ......................................................... 75
6.6 Conclusion ........................................................ 78

7 Oscillator aggregation in redundant robotic system 79
   7.1 Introduction .................................................... 79
   7.2 Redundant arm robot ............................................ 79
   7.3 Implementation for redundant robotic system .............. 80
      7.3.1 Convergence and searching mechanism to evaluate locomotion 80
      7.3.2 Allocation of servos and sensors ............................ 82
      7.3.3 Phase calculations using isochron .......................... 83
   7.4 Experimental result ............................................. 84
      7.4.1 (A) Default situation ......................................... 85
      7.4.2 (B) The case with obstacle .................................. 87
      7.4.3 (C) The case that a part of the wheel of robot is broken ... 91
      7.4.4 (D) The case that robot have to reach the object non-monotonically because of its physical constraint .......... 93
      7.4.5 Discussion .................................................. 95
   7.5 Conclusion ...................................................... 97

8 Summary 100
List of Figures

2.1 Concept of homeostasis in robotic systems by assuming the environment and the robot oscillators .................................................. 17

3.1 Lattice of agents in flocking algorithm .............................................. 22
3.2 Behavior of agents in flocking algorithm ............................................. 22
3.3 Relationships between agents in extended flocking algorithm .......... 26
3.4 Formations for agents in the experiment ............................................. 27
3.5 Experimental result of formation 1 ...................................................... 28
3.6 Experimental result of formation 2 ...................................................... 29
3.7 Experimental result of formation 3 ...................................................... 30

4.1 Flock shifting mechanism using $\Delta q$ ................................................. 34
4.2 Size of simulation robot ................................................................. 35
4.3 Hinge and sensor positions and initial state ..................................... 36
4.4 Agent allocation ................................................................................ 37
4.5 Graph of time to stand again when impacted from forward ............ 39
4.6 Simulation robot impacted from forward .......................................... 40
4.7 Graph of time to stand again when impacted from back ............... 41
4.8 Simulation robot impacted from back ............................................... 42
4.9 Graph of time to stand again when impacted from side ............... 43
4.10 Simulation robot impacted from side ............................................... 44
4.11 Humanoid robot used in this experiment ...................................... 45
4.12 Composition of humanoid robot .................................................... 46
4.13 Pressure sensors on bottom of the foot ........................................... 47
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.10</td>
<td>Graph of $\varepsilon(t)$ in various $n_R$ values</td>
<td>78</td>
</tr>
<tr>
<td>7.1</td>
<td>Screenshot of redundant arm robot</td>
<td>80</td>
</tr>
<tr>
<td>7.2</td>
<td>Roller of robot</td>
<td>81</td>
</tr>
<tr>
<td>7.3</td>
<td>Servo of robot</td>
<td>82</td>
</tr>
<tr>
<td>7.4</td>
<td>20-Dof robot and object</td>
<td>83</td>
</tr>
<tr>
<td>7.5</td>
<td>Cyclic arm motion and potential</td>
<td>84</td>
</tr>
<tr>
<td>7.6</td>
<td>Allocation</td>
<td>85</td>
</tr>
<tr>
<td>7.7</td>
<td>Normal states and their positions of Object and 20-Dof robot</td>
<td>86</td>
</tr>
<tr>
<td>7.8</td>
<td>Distance between object and arm</td>
<td>87</td>
</tr>
<tr>
<td>7.9</td>
<td>Acquired attitude of robot at each phase</td>
<td>88</td>
</tr>
<tr>
<td>7.10</td>
<td>Positions of object and the head of robot</td>
<td>89</td>
</tr>
<tr>
<td>7.11</td>
<td>Examples of oscillator state</td>
<td>90</td>
</tr>
<tr>
<td>7.12</td>
<td>$x_d$ transition</td>
<td>91</td>
</tr>
<tr>
<td>7.13</td>
<td>Return map examples</td>
<td>92</td>
</tr>
<tr>
<td>7.14</td>
<td>Initial state in obstacle placed field</td>
<td>93</td>
</tr>
<tr>
<td>7.15</td>
<td>Initial state of experiment D</td>
<td>94</td>
</tr>
<tr>
<td>7.16</td>
<td>The relationship between $\Delta f_p^{(k)} + \Delta f_s^{(k)} + f_{prob}^{(k)}$ and distance in situation (A)</td>
<td>96</td>
</tr>
<tr>
<td>7.17</td>
<td>The relationship between $\Delta f_p^{(k)} + \Delta f_s^{(k)} + f_{prob}^{(k)}$ and distance in situation (B)</td>
<td>96</td>
</tr>
<tr>
<td>7.18</td>
<td>The relationship between $\Delta f_p^{(k)} + \Delta f_s^{(k)} + f_{prob}^{(k)}$ and distance in situation (C)</td>
<td>97</td>
</tr>
<tr>
<td>7.19</td>
<td>One example of distance between object and arm in a experiment of (A)</td>
<td>98</td>
</tr>
<tr>
<td>7.20</td>
<td>Locomotion of the head of robot</td>
<td>99</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Experimental results of average error values ................. 46
Preface

The main feature that keeps states and structures stable can be seen in living organisms. This adjusting and adaptive feature is called homeostasis. This integrated adaptive feature is achieved by the cooperation of organs in living organisms. Living organisms in nature act dynamically due to this feature. The main purpose of this paper is to generate homeostatic behavior in an autonomous system such as robotic systems. Robot systems that use neural oscillators and central pattern generators (CPGs) have been developed to generate flexible and robust locomotion in robotic systems. CPGs is based on the specific neural systems that generate rhythm pattern observed in vertebrates and control periodic locomotions. It is possible to generate rhythmic locomotion that is robust against environmental change. They are constructed as artificial neural oscillators in an artificial system. These neural oscillators and its parameters are determined by human based on their experiences. However, homeostatic and adaptive behaviors for environmental changes have been observed in living organisms that have no neural circuits such as amoebas. Synchronization is considered to have occurred in these phenomena when each cell acts as an oscillator and an amoeba acts as an aggregation of oscillators. In this paper, the main focus is to generate homeostatic behavior seen in not only living organisms with no neural circuits but living organisms with neural circuits as the result of synchronization phenomena in artificial systems. For this reason, the concept that assume both the environment and the autonomous system as oscillator aggregation is proposed and the control method based on flocking algorithm in multi-agent systems is constructed as the first step. Also, the feature of synchronization phenomena is analyzed and extracted from extended flocking algorithm.
for constructing the control method to assume an autonomous systems as oscillator aggregation. Experimental results and its approaches are denoted as follows. Recent intelligent robots are introduced in chapter 1. The robots that need adaptive behavior in dynamic environment are the main focus of this chapter. In chapter 2, adaptive behaviors observed in living organisms is denoted. Based on such facts, the concept that assumes both the environment and the autonomous system as oscillator aggregations is proposed. The state of the autonomous system is expressed as the relationships between oscillators in this time. As the first step to construct the oscillator aggregation control method for autonomous systems, flocking algorithm in multi-agent system is focused as the representative of distributed algorithm that is robust and flexible in chapter 3. It is difficult to apply flocking algorithm to an single autonomous system because flocking algorithm is for multi-agent systems. Therefore, flocking algorithm is extended for applying to an autonomous system by expressing arbitrary lattice formation. In chapter 4, several experiment to confirm the effect of application of extended flocking algorithm to an autonomous system for generating adaptive behavior are shown. First, a simulation robot and a humanoid robot in real world are applied extended flocking algorithm. The task of each robot is to maintain their standing posture by dealing with external forces and disturbances. Next, the camera tracking system in the environment that its brightness changes is shown. This camera system is applied extended flocking algorithm to follow the brightness changes to keep tracking the target markers. Chapter 5 shows the result of analysis of the feature of extended flocking algorithm as synchronization phenomena. Each agent of extended flocking algorithm can be assumed as oscillators and their behaviors are possible to assume as synchronization phenomena. Based on these results in earlier chapters, the control method of oscillator aggregation for autonomous system is proposed in chapter 6. The autonomous system is assumed as oscillator aggregation and its state is expressed as phase relationships between oscillators. The mechanism to reconstruct and stabilize their phase relationships depending on the situation is introduced in this control method. The fundamental features are shown through simulation experiments. The experimental results of application of the control method in previous chapter are shown in chapter 7.
Redundant arm robot is prepared and is applied the control method.
Chapter 1

Overview of intelligent robot controls

1.1 Introduction

A plenty of robots are working all over the world. Some robots are controlled by human, others is required to work autonomously. The robots that should work autonomously need intelligent behavior and intelligent control method is needed to generate such behavior. Intelligent control method is studied by researchers in various contexts. These control methods are constructed based on their original backgrounds and they are introduced in this chapter.

1.2 study of intelligent robot control

There are lots of types of robots. They are used for nursing care, transportation, entertainment, auto driving, manufacturing and so on. However, autonomous and intelligent control is needed with expanding of the field that robots work. Some robots are desired to work alone even in the field humans is not able to work, others have to communicate with human flexibly. These are intelligent works and those robot need intelligent and adaptive behavior to deal with complicated environment. A lot of study exist to achieve such intelligent behavior.
1.3 Machine learning and artificial neural network

Machine learning is popular method to have robots behave intelligently. For highly adaptive behavior, artificial neural network is often used[1]. Artificial neural network is method for classification based on the neural network observed in living organisms. Some robots are made to classify signal from human body to move actuators[2][3]. Artificial neural network is also used to obtain suitable parameters for manipulator control[4]. Reinforcement learning is also popular for intelligent robot control. And there is the case that uses both property of reinforcement learning and neural network for reaching movement[5]. Several types of neural network, for example neural network having time delay elements[6] and recurrent neural network[7], exist and they are used to solve a lot of kind of problems such as inverse kinematics problem[8], underwater vehicle manipulator controlling[9]. On the other hand, distributed control approach for adaptive behavior is also proposed[10].

1.4 Evolutionary robotics

Evolutionary computing is used for intelligent robot control. Robotics using evolutionary computing is called evolutionary robotics[11][12]. Obtaining neural controller by genetic algorithm is one of the example of evolutionary robotics[13][14][15]. NEAT is an example to obtain structure and weights of neural network by evolutionary computing[16][17]. Also, several types of techniques using both genetic algorithm and neural network exist[18].

1.5 Global and local optimization for complicated task

Robots have to deal with complicated task when they are in real world and communicate with humans. Tasks human wants to have robot solve is not simple and have several objective functions. These objective functions are compete against each
other sometimes and robot should find the best answer with compromising something. These problems are called multiobjective optimization. There are approach to deal with such problems using genetic algorithm[19][20][21][22][23]. These techniques are used widely[24]. Particle swarm optimization is also used to approach such problems[25][26].

1.6 Human robot interaction

Other approach to use robot in real world exists. Human robot interaction focus on communication with human[27]. These robots appeared to be intelligent but the approach is different from the approach described above. In human robot interaction, main focus of these research is communication so necessary technique is such as capturing human motion[28], analysing human motion[29], perspective-taking[30], making motion[31] and robot designing[32]. Human robot interaction using crowdsourcing is also proposed[33] and long-term relationship between human and robot is considered[34]. Privacy in human robot interaction is also discussed[35]. Robot is also considered as interface for human and natural input method for human is studied[36]. Cognitive mechanism is one of the topic[37][38]. These robot is already used in the real world[39][40]. Paro is one of the example working in nursing care[41][42]. Also, there are robots working in entertainment[43][44]. Teleoperation is one of the topics[45]. The studies focus on collaborative work between human and robot exist[46][47][48][49].
Chapter 2

Concept of robot control with structured relationship of parameters

2.1 Introduction

The main feature that keeps states and structures stable can be seen in living organisms. This adjusting and adaptive feature is called homeostasis. This integrated adaptive feature is achieved by the cooperation of organs in living organisms. Living organisms in nature act dynamically due to this feature. Such feature seems to be needed for generating robust and flexible behavior in an autonomous system that is in dynamic environment. The concept of an autonomous system that has homeostatic feature is indicated in this chapter based on the behavior and mechanism of living organism.

2.2 Synchronization in living organisms

A lot of living organisms have neural circuits and behave intelligently. For example, C. elegans have 302 neurons and they are connected to the muscles[50][51]. C.
elegans processes information from various kind of internal and external stimuli and generates appropriate locomotions. However, intelligent and adaptive behaviors are observed in living organisms without any neural circuits. For example, predictive behaviors for periodic stimuli have been observed in living organisms that have no neural circuits such as amoebas. Synchronization is considered to have occurred\cite{52} in these phenomena when each cell acts as an oscillator and an amoeba acts as an aggregation of oscillators.

### 2.3 Concept of homeostasis in robotic system

The earth and living organisms are open non-equilibrium systems with flows of energy and materials\cite{53} that are closely related. For example, climate change affects living organisms\cite{54}. To deal with such effects like periodic changes on earth, many living organisms have mechanisms such as circadian rhythms\cite{55}. However, these adaptations in living organisms are not only for periodic phenomena. Amoebas, for instance, can expect further stimuli even after the periodic stimuli have stopped\cite{52}. We model these relationships with the environment and autonomous systems such as living organisms and robots by assuming that the environment and autonomous systems are both oscillators. These oscillators are also the aggregation of oscillators with some interactions. In addition, we assume that each oscillator has a condition and there is a term that causes the state to become unstable if the condition is not satisfied. When we consider the aggregation of oscillators that interacts, its state is multistable and it transit to other states due to the effect of the initial state and disturbances\cite{56}. We expect that these oscillators would transit to other available states because of such features when they are unstable until they become stable. After they reach a stable state, they synchronize and maintain their relationships. They construct the relationships where the autonomous system is more stable because the speed of change the state in an autonomous system appear to be faster locally. We consider such phenomena and the processes as homeostasis.
2.4 Relationships between parameters

While states of oscillators are stable, oscillators keep the relationships between other oscillators. These relationships are phase relationships. Thus, each state is expressed as phase relationships between oscillators. Now, we assume that each state, in other words the phase relationship, expresses the specific state of an autonomous system. Oscillators try to find the more stable state by changing their phase relationships and keep the phase relationship after they find the stable relationship. So, if we assume the phase relationships express the relationships between parameters of the autonomous system, the autonomous system try to find and keep the stable relationship of own parameters. We call the feature to try to find the better relationship ”Reconstruction” and the feature to keep the current relationship ”Stabilization”.

Figure 2.1: Concept of homeostasis in robotic systems by assuming the environment and the robot oscillators

2.5 Conclusion

The concept of an autonomous system that has homeostatic feature is indicated in this chapter. In this concept, the environment and an autonomous system are both considered as aggregations of oscillators. This concept is constructed based on the case of living organism such as amoebas and adaptive behavior such as prediction
and avoidance behavior is described as synchronization phenomena. Each state of an autonomous system is expressed as the phase relationships between oscillators and these oscillators are expected to have feature to find and keep the better relationship from the perspective of synchronization phenomena.
Chapter 3

Structured relationship in flocking algorithm

3.1 Introduction

Flocking algorithms are known as autonomous and distributed control methods for multi-agent systems. Complex patterns and flexible behaviors are achieved by these algorithms even though they are constructed from simple rules and each agent only follows the rules. These phenomena are known as swarm intelligent and most of them focus on the relationship between agents. Such behavior seems similar to the homeostatic behavior in living organisms and the feature that express the relationship between agent in algorithm is common in both flocking algorithm and the concept of this paper. As the preparation to use flocking algorithm as the first step to construct method for the concept, flocking algorithm is extended for a single autonomous system.

3.2 Overview of flocking algorithms

For constructing the algorithm to reconstruct and stabilize the state of the autonomous system, we focus on flocking algorithm. Flocking algorithm is the algorithm for multi-agent systems and have agent construct flock or swarm like birds
and fishes. Flocking algorithms are constructed as autonomous and distributed control system and each agent has limited sensor data. Agents in flocking algorithm generate complex patterns of movement. This is known as swarm intelligence and is flexible and robust[57][58]. Several algorithms employing swarm intelligence are proposed[59]. Boids[60] is famous example of flocking algorithm. Algorithm of boids has only 3 simple rules, ”Separation”, ”Alignment” and ”Cohesion”. Each rule is defined as follows.

**Separation:** steer to avoid crowding local flockmates

**Alignment:** steer towards the average heading of local flockmates

**Cohesion:** steer to move toward the average position of local flockmates

Each agent moves based on these simple rules and complicated flock is made from these agents. Robot systems using swarm intelligent exist. Some swarm robots exist and they are considered as multi agent system[61]. OSCAR is a single robot but using boids to control legs[62].

### 3.3 Flocking algorithm as consensus problem

Most of the flocking algorithm are explained as consensus problem. Consensus problem is used as the model case of autonomous vehicle or UAV to uniform their positions and attitudes and used for covering problem of sensor network. Problem setting and analytical solution are wide-ranging [63][64][65][66]. For example, consensus problem in networks with directed graphs and switching topology[67], consensus problem with the agents that have more complicated dynamics[68][69][70], consensus problem for gossip algorithm[71][72], consensus problem for event-based communication[73], consensus problem under the baud rate limitation[74] and consensus problem in real world [75][76][77][78][79]. Also, the study about security problem occur when these techniques are used in real world exists[80]. These consensus problem has close connection with graph theory[81][82][83]. The studies focus
on formation control also exist\cite{84}\cite{85}\cite{86}. Famous example of application of these study is multi UAV operation\cite{87}.

### 3.4 Flocking algorithm

We apply flocking algorithm to construct an algorithm for autonomous systems to generate homeostasis as a first step. Each agent of flocking algorithm work autonomously and distributed. Also, it works to maintain the ”flock” of agents and is able to adapt the environmental changes. This ”flock” means the relationships between agents. If we assume the agents as components of the autonomous system we want to generate homeostasis, it might be possible to express the relationships between components as the relationships between agents of ”flock”. However, flocking algorithms are for multi-agent systems and not for single autonomous systems. Thus, it is difficult to apply these flocking algorithm to a single autonomous system. We have to extend a flocking algorithm to be able to apply to a single autonomous system. However, without a theoretical framework, it is hard to extend the algorithm so we use a flocking algorithm presented by Olfati-Saber\cite{88}\cite{89}. In his work, a theoretical framework for design and analysis of distributed flocking algorithms is proposed as follows. Each agent tries to keep a fixed distance with other agents and moves keeping a lattice alignment (Fig. 3.1). Also, we assume the presence of a virtual agent that indicates the ideal movement and each agent follows the movement (Fig. 3.2). We consider a graph $G$ that consists of a set of vertices $V = \{1, 2, ..., n\}$ and a set of edges $E \subseteq V \times V$. Also, the adjacency matrix $A = [a_{ij}]$. Now, we assume that each node is an agent and $V$ is an agent set. Let $q_i, p_i, u_i \in \mathbb{R}^m$ denote the position, velocity, input of agent $i$ for all $i \in V$ respectively. Each agent has dynamics

$$\begin{cases}
    \dot{q}_i = p_i \\
    \dot{p}_i = u_i
\end{cases} \tag{3.1}$$
The set of neighbors of agent $i$ is denoted by

$$N_i = \{ j \in V : \| q_j - q_i \| < r \}$$

(3.2)

where $r > 0$ is the interaction range and $\| \cdot \| : \mathbb{R}^m \rightarrow \mathbb{R}$ is a Euclidean norm of $\mathbb{R}^m$. Agents are designed to maintain an identical distance with other agents denoted as follows, in other words, to construct uniformed lattice that satisfies equation (3.3).

$$\| q_j - q_i \| = d, \forall j \in N_i$$

(3.3)

We then have to determine the input $u_i$ to have agent follow virtual agent and try to satisfy equation (3.3) as much as possible. The input $u_i$ for agent $i$ is determined
by three terms as follows.

\[ u_i = f_i^g + f_i^d + f_i^γ \]  

(3.4)

where the term \( f_i^g \) is a gradient-based term of potential among agents, the term \( f_i^d \) is a velocity consensus/alignment term that acts as a damping force, and the term \( f_i^γ \) is a navigational feedback. We now determine each term of the right side of an equation (3.4). Here, \( \sigma \)-norm is defined as

\[ \|z\|_{\sigma} = \frac{1}{\epsilon} \sqrt{1 + \epsilon \|z\|^2} - 1 \]  

(3.5)

where \( \epsilon > 0 \) and this norm is differentiable everywhere. Also, gradient \( \sigma_\epsilon(z) = \nabla\|z\|_{\sigma} \) is defined as follows.

\[ \sigma_\epsilon(z) = \frac{z}{\sqrt{1 + \epsilon \|z\|^2}} = \frac{z}{1 + \epsilon \|z\|_{\sigma}} \]  

(3.6)

In addition, the bump function \( \rho_h(z) \) that is continuous at \([0, 1]\) is defined as follows.

\[ \rho_h(z) = \begin{cases} 
1 & z \in [0, h) \\
\frac{1}{2} [1 + \cos \pi \frac{(z-h)}{(1-h)}] & z \in [h, 1] \\
0 & \text{otherwise} \end{cases} \]  

(3.7)

where \( h \in (0, 1) \). With this bump function, each element of the adjacency matrix is determined as

\[ a_{ij} = \rho_h(\|q_j - q_i\|_{\sigma} / \|r\|_{\sigma}) \]  

(3.8)

where \( a_{ii}(q) = 0 \) for all \( i \) and \( q \). Now, we define the collective potential function \( V(q) \).

\[ V(q) = \frac{1}{2} \sum_i \sum_{j \neq i} \psi_\alpha(\|q_j - q_i\|_{\sigma}) \]  

(3.9)

where \( \psi_\alpha \) is determined by using action function \( \phi_\alpha(z) \) as follows.

\[ \phi(z) = \frac{1}{2} [(a + b) \sigma_1(z + c) + (a - b)] \]  

(3.10)
\[
\phi_\alpha(z) = \rho_h(z/\|r\|_\sigma)\phi(z - d_\alpha)
\]  
\[ (3.11) \]

\[
\psi_\alpha(z) = \int_0^z \phi_\alpha(s)ds
\]  
\[ (3.12) \]

where \(0 < a \leq b, c = |a - b|/\sqrt{4ab}, \sigma_1(z) = z/\sqrt{1 + z^2}.\) Then,

\[
f^g_i = -\nabla q_i V(q) = \sum_{j \in N_i} \phi_\alpha(||q_j - q_i||_\sigma)\sigma_1(q_j - q_i)
\]  
\[ (3.13) \]

\[
f^{d}_i = \sum_{j \in N_i} a_{ij}(q_j - p_j)
\]  
\[ (3.14) \]

\[
f^{\gamma}_i = -c_1(q_i - q_r) - c_2(p_i - p_r)
\]  
\[ (3.15) \]

where \(q_r, p_r\) are the position and velocity of the virtual agent respectively and \(c_1, c_2\) are positive constant. Here, \(f^g_i\) is an attractive/repulsive force as a function of distance between agents, \(f^{d}_i\) is a force that even out the speed of agent and \(f^{\gamma}_i\) is a force that have agent follow the virtual agent.

### 3.5 Extended flocking algorithm

Original flocking algorithm is to have agents configure uniform lattice satisfies equation (3.3). However, this algorithm cannot form any lattice that has an arbitrary distance among agents, so it is hard to apply this algorithm to other systems. Therefore, we extended the flocking algorithm to be able to form arbitrary lattice and provide several virtual agents. We consider a matrix \(D\). This matrix \(D\) has elements given by the distance \(d\) in equation (3.3) between each agent. The distance between agent \(i\) and \(j\) is denoted as \(d_{ij}\). Now we assume that there are \(n\) agents.

\[
D = \begin{bmatrix}
d_{11} & d_{12} & \cdots & d_{1n} \\
d_{21} & \ddots & \vdots \\
\vdots & \ddots & \ddots \\
d_{n1} & \cdots & \cdots & d_{nn}
\end{bmatrix}
\]  
\[ (3.16) \]
where $d_{ii} = 0, (i = 1, 2, ..., n)$, $d_{ij} = d_{ji}, (i, j = 1, 2, ..., n)$.

Also, we assume that each agent has its own virtual agent. Some virtual agents can be shared among a number of agents. Now we consider that $s$ virtual agents exist. The distance among virtual agents, in other words, the lattice of virtual agents, is denoted by matrix $R$ as

$$
R = \begin{bmatrix}
  r_{11} & r_{12} & \cdots & r_{1s} \\
  r_{21} & \ddots & \vdots & \\
  \vdots & \ddots & \ddots & \\
  r_{s1} & \cdots & \cdots & r_{ss}
\end{bmatrix}
$$

(3.17)

where $r_{ii} = 0, r_{ij} = r_{ji}, (i, j = 0, 1, ..., s)$. We define $r$ in equation (3.2) for each agent. $r_i, r$ for agent $i$ which has destination $k$, is given by

$$
J = \{1, 2, ..., s\}
$$

(3.18)

$$
J_{(-k)} = J - \{k\}
$$

(3.19)

$$
r_{i}^{\text{min}} = \min_{j \in \{j \in J_{(-k)} \mid r_{kj}\}} r_{kj}
$$

(3.20)

Therefore, (3.2),(3.8),(3.11) are changed as follows.

$$
N_i = \{j \in V : \|q_j - q_i\| < r_{i}^{\text{min}}\}
$$

(3.21)

$$
a_{ij} = \rho_h(\|q_j - q_i\|/\|r_i\|) \in [0, 1], j \neq i
$$

(3.22)

$$
\phi_\alpha(z) = \rho_h(z/\|r_i\|)\phi(z - \|d_{ij}\|)
$$

(3.23)

Also, if agent $i$ has a virtual agent $k$, (3.15) is given by

$$
f_i^\gamma = -c_1(q_i - q_{rk}) - c_2(p_i - p_{rk})
$$

(3.24)

where $q_{rk}$ and $p_{rk}$ are the position and velocity of virtual agent $k$ respectively. The relationships between agents in extended flocking algorithm are shown in Fig. 3.3.
3.6 Expression of relationships in flocking algorithm

The purpose of using flocking algorithm is to express the relationships between components of the autonomous system as "flock". In flocking algorithm proposed by Olfati-Saber, the relationships between agents are expressed as a fixed ideal distance $d$. This means the distances between agents are all the same and uniformed "flock" is made by the agents. However, relationships between components of the autonomous system are not the same and such inequable relationship has to be expressed as "flock". In extended flocking algorithm, each distance between agents is not all the same and each of them is expressed in ideal distance matrix $D$. Thus, extended flocking algorithm is able to express the relationships among components of the autonomous system as ideal distance matrix $D$. 
3.7 Demonstration of extended flocking algorithm

Arbitrary formation can be achieved by extended flocking algorithm. Now, we consider to form three formation in Fig. 3.4. Each blue circle indicates the virtual agent for one group. Results are shown in Fig. 3.5, Fig. 3.6 and Fig. 3.7.

3.8 Conclusion

Flocking algorithm is extended to express the arbitrary formation as the preparation to use flocking algorithm for constructing control method of a single autonomous system. Several formations are observed using this extended flocking algorithm and results are shown in this chapter.
Figure 3.5: Experimental result of formation 1
Figure 3.6: Experimental result of formation 2
Figure 3.7: Experimental result of formation 3
Chapter 4

Application of flocking algorithm for robot control

4.1 Introduction

The control method using flocking algorithm is introduced in this chapter as the first step for oscillator aggregation control in an autonomous system and several experiment to confirm the effect of the control method are shown in this chapter. At first, the control method is applied to humanoid robots for maintaining their postures. Next, the control method is applied to camera tracking system for adapting the brightness of surrounding environment. This control algorithm tries to maintain the specific relationships of parameters as agents.

4.2 Extended flocking algorithm and parameter tuning

Extended flocking algorithm is developed for generating homeostasis in an autonomous system by expressing the relationships between components of the system. We have to consider which kind of component to use in an autonomous system to apply extended flocking algorithm. Now we adopt control parameters of the au-
tonomous system as agents of extended flocking algorithm. In this case, the relationships between parameters are expressed in ideal distance matrix $D$ of extended flocking algorithm as the distance between parameters. Agents of extended flocking algorithm try to maintain their relationships. This is the behavior such as “Stabilization”. However, “Reconstruction” also should be implemented. So, we add one more function in extended flocking algorithm to generate ”Reconstruction” behavior and apply this algorithm to autonomous systems.

4.3 Implementation of probabilistic fluctuation term:

$P_{prob}$

In extended flocking algorithm, an input of agent $i$, $u_i$ is determined by three terms as follows.

$$u_i = f_i^g + f_i^d + f_i^\gamma$$

where the term $f_i^g$ is a gradient-based term of potential among agents, the term $f_i^d$ is a velocity consensus/alignment term that acts as a damping force, and the term $f_i^\gamma$ is a navigational feedback. Now the term $f_{prob}$ for generating probabilistic fluctuation is introduced.

$$u_i = f_i^g + f_i^d + f_i^\gamma + f_i^{prob}$$

$$f_i^{prob} = c_{prob} \theta v_{random}$$

where $c_{prob}$ is positive constant and $v_{random} \in (-0.5, 0.5)$ is a random number. $\theta$ is determined to express how the current state is comfortable. If the current state is uncomfortable, this value becomes bigger.

4.4 Allocation of parameters to agents

All the parameters in an autonomous system should be allocated to the agents of the flock. In an autonomous system, there are two types of parameters, control
parameters and sensors. Control parameters can be changes arbitrarily. Therefore, the value of control parameters are directly allocated to the agent of the flock. Agent’s position is assumed as one dimensional value and all normalized as $[0, 1]$. However, sensor input values are not able to control. Sensor input values are also allocated to the position of agents but they are uncontrollable. We call the agent allocated to the sensor “Sensor agent”. Because of this reason, the aim of this extended flocking algorithm is to reach the position of sensor agents to their virtual agents as close as possible by controlling other agents.

### 4.5 Other implementation

Each agent try to stay the neighborhood if the positions of the virtual agents are fixed. Thus, searching area of agents is limited. These movement is too small for "Reconstruction" and we implement another mechanism to move whole flock for searching wider area. In this mechanism, $\Delta q$ is determined based on summation of the difference of position between each sensor agent and its virtual agent. At time $t$ and $t+1$, the positions of center of gravity of flock is denoted as $q_G(t)$, $q_G(t+1)$ respectively. Also, at time $t$ and $t+1$, the summations of distances between sensor agents and their virtual agents are denoted as $\varepsilon(t)$, $\varepsilon(t+1)$ respectively. Now, the vector that moves flock $\Delta q$ is determined as follows.

$$
\Delta q = \left(1 - \frac{\varepsilon(t+1)}{\varepsilon(t)}\right) \{q_G(t+1) - q_G(t)\} \tag{4.4}
$$

Based on this vector, the positions of all the virtual agents at time $t+2$ is moved as follows.

$$
q_r(t+2) = q_r(t+1) + \Delta q \tag{4.5}
$$

where $q_r(t)$ is the position of the virtual agent at time $t$. The image of this mechanism is illustrated in Fig. 4.1. The purpose of this mechanism is to move all the positions of virtual agents by $\Delta q$ and this $\Delta q$ is determined to move whole flock to the position that is expected to reduce the distances between sensor agents and virtual agents.
4.6 Application of extended flocking algorithm

4.6.1 Application of extended flocking algorithm for humanoid robot control

Simulation

To confirm the effect to apply extended flocking algorithm to an autonomous system, the following problem is prepared. A simple robot is constructed on the simulation environment. The robot is pushed from forward, back, right and left. The task for the robot is to keep standing under this situation by controlling each joint of the robot. A simple robot used in this experiment is shown in Fig. 4.2. Its leg, foot and arm is simplified and each rigid body part of this robot is constraint by hinge. 5 hinges are prepared as servo motor and 4 pressure sensors are prepared on the bottom of the foot. Positions of the sensors are shown in Fig. 4.3. These 5 servos and 4 pressure sensors are allocated to the agent of extended flocking algorithm as
Figure 4.2: Size of simulation robot

shown in Fig. 4.4. The ideal distance matrix $D$ is determined as

$$D = \begin{pmatrix}
  h_1 & \cdots & h_5 & s_1 & \cdots & s_4 \\
  0 & \cdots & d_{15} & d_{16} & \cdots & d_{19} \\
  \vdots & \ddots & 0 & \vdots & & \vdots \\
  h_5 & & & & d_{51} & \cdots & d_{59} \\
  s_1 & d_{61} & \cdots & & d_{69} & & \\
  \vdots & & 0 & & \vdots & & \\
  s_4 & d_{91} & \cdots & d_{95} & d_{96} & \cdots & 0
\end{pmatrix} \quad (4.6)$$

Each position of virtual agents is determined as the initial state. The dynamics of this robot is not considered in this experiment. Each coefficient is as follows.
Figure 4.3: Hinge and sensor positions and initial state

- $h = 0.5$
- $a = 0.5$
- $b = 1.0$
- $c_1 = 0.6$
- $c_2 = 0.1$
- $c_{prob} = 0.001$

We compare this method with NEAT[16] as representative example of adaptive control for unknown dynamics to confirm the effect of this method. NEAT is used in
evolutionary robotics and an algorithm to get artificial neural system as controller evolutionarily. NEAT is superior because it adopts variable length genotype, effective crossover and differentiation of species. It is able to get the weights and structure of the neural controller suitable for the specific task simultaneously. Fitness \( s \) is determined as

\[
s = \sum_{i=0}^{y_t} y_i \tag{4.7}
\]

where \( y_t \) is a height of its head at time \( t \). The impact is added from random direction and with random power at each 1000 step. A recurrent neural network of 4-input and 5-output is prepared preliminary. At each generation, node addition probability \( p_n \), edge addition probability \( p_e \) and weight mutation probability \( p_w \) is determined as
- \( p_n = 0.05 \)
- \( p_e = 0.05 \)
- \( p_w = 0.1 \)

Each coefficient is determined as

- \( c_1 = 1.0 \)
- \( c_2 = 1.0 \)
- \( c_3 = 0.4 \)
- \( \delta_t = 3.0 \)

The population for each generation is 500 and both roulette selection and elite selection is adopted. The robot applied extended flocking algorithm and the robot with neural controller gained by NEAT is set standing in each trial. Also, impact is added from forward, back and side in each 1000 step. The time to stand again after lost their balance is recorded. 9000 steps is calculated in each trial and calculate the average of 10 trials. Fig. 4.5 shows the graph of the case that impact comes from forward. X -axis denotes ratio of size of the impact when the biggest impact is 1 that the robot does not take down without any control method. Y -axis the time step to stand again after the impact and 1 step is \( \frac{1}{60} \)(s). In Fig. 4.5, the robot applied extended flocking algorithm is able to keep standing even though it is impacted over twice as much as the impact that the robot with neural controller of NEAT is not able to keep standing. Screenshot is shown in Fig. 4.6. Fig. 4.7 shows the graph of the case that impact comes from back. In this case, the robot with extended flocking algorithm gets longer to stand sometimes though it is smaller difference compared with the case from forward. And in extended flocking algorithm case the results are scattered. This phenomenon results from the way of evaluation. In NEAT robot, the robot is able to make attitude arbitrarily while extended flocking algorithm robot tries to initial attitude. Thus, NEAT robot pulls its arm down while extended flocking algorithm robot raises its arm up and the center of gravity
Figure 4.5: Graph of time to stand again when impacted from forward

is kept higher. This is the reason why the result of extended flocking algorithm
robot worse. In spite of the difference, the results are almost the same. The result
of extended flocking algorithm robot would be improved if any mechanism to have
this robot other attitudes. Screenshot is shown in Fig. 4.8. Finally, the graph of
the case that impact comes from side is shown in Fig. 4.9. The result of extended
flocking algorithm robot is also better than that of NEAT robot.

**Humanoid Robot**

The effect of extended flocking algorithm is confirmed in previous experiment. How-
ever, it is simulation result and it is not clear whether it is still effective in real world.
To confirm the effect of application of extended flocking algorithm in real world, a
humanoid robot is prepared for experiment. In this experiment, humanoid robot
Kondo KHR-3HV shown in Fig. 4.11 is used. This robot has 10 sensors. 8 Sensors
are force sensing resistor FSR402 as pressure sensor. They are set on the bottom
of the foot as shown in Fig. 4.13. Other 2 sensors are acceleration sensors. Also
this robot consists of 17 servo motors. Components of this robot is shown in Fig.
4.12. Each servo motor and sensor is considered to be an agent that has scalar state value. Servo motor agents have angle value and sensor agents have their own specific value. Task of this robot is to repeat and maintain 2 postures. However, the robot is forced to grab flexible bar and affected by oscillation of this bar. 2 postures are shown in Fig. 4.15. Ideal values, in other words the positions of virtual agents, are set to the value that is recorded preliminary. \( \varepsilon(t) \) is determined as the difference between current values of pressure sensor and recorded values of pressure sensor. Although, this value is normalized. The results are shown in Fig. 4.16. This graph shows the result of time-variation of \( \varepsilon(t) \). Normal 1, 2 and 3 mean the result without extended flocking algorithm of 3 trial and Flock 1, 2 and 3 mean the result of the robot applied extended flocking algorithm of 3 trial. Without extended flocking algorithm, the results are worse (\( \varepsilon(t) \) is higher) and scattered. However, the results are almost the same and \( \varepsilon(t) \)'s are lower. From this results, applying extended flocking algorithm to an autonomous system is effective to generate adaptive behavior even in real world.
4.6.2 Application of extended flocking algorithm for camera tracking system

Experimental setting

We apply flocking algorithm for marker tracking camera system as one of the example[90]. This camera system tracks positions of line tracer with red, green and blue marker. Each marker is tracked by color extraction and its contours are defined from the result. From all the contours defined, the largest contour is selected and the center of the gravity of the contour is determined as the position of the line tracer. The distances between actual positions of line tracer and this tracked positions are observed in this system. These line tracer is made by Lego Mindstorms NXT in Fig. 4.17. The camera UCAM-DLY300TAWH(Fig. 4.18) is able to control brightness, contrast, saturation, sharpness, gamma and white balance. The range of values are as follows.

1. brightness       [-10, 10]
2. contrast        [0, 20]
Agents whose states are these values are prepared. The areas of tracked contour are used as sensor values. Therefore, there are 9 agents in this flock and the position $q_i$ of agent $i$ is allocated as follows.

1. brightness $q_1$
2. contrast $q_2$
3. saturation $q_3$
4. sharpness $q_4$
5. gamma $q_5$
6. white balance $q_6$
7. Detected area of red marker $q_7$
8. Detected area of green marker $q_8$
9. Detected area of blue marker $q_9$
The position $q_i$ of agent $i$ is

$$q_i = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$  \hspace{1cm} (4.8)$$

where $x$ is an actual value and $x_{\text{max}}$ and $x_{\text{min}}$ are max and min value of $x$ respectively. All the positions are normalized to $[0,1]$. The state in the bright room is called situation 1 and the state in the dark room is called situation 2 here. All the parameters in the state that tracks all the line tracers correctly in situation 1 are recorded preliminary and set as the positions of virtual agents. At first, we experiment in situation 1 with these virtual agents. Next, we experiment in situation 2 with the same parameters. Each coefficient is set as follows.

- $\varepsilon = 0.5$
- $h = 0.5$
- $a = 0.5$
- $b = 0.5$
- $c_1 = 0.5$
\[ c_2 = 0.5 \]
\[ c_{prob} = 0.01 \]

All the parameters recorded in situation 1 as the ideal values are as follows.

1. brightness \qquad 0
2. contrast \qquad 3
3. saturation \qquad 5
4. sharpness \qquad 5
5. gamma \qquad 150
6. white balance \qquad 4588

The ideal distances between agents are determined by the distances between their virtual agents. Ideal distance matrix \( D \) is shown in Fig. 4.19. In these experiment, we compare three situations, the system with no parameter tuning (Default), the system applied extended flocking algorithm without flock shift mechanism in eq. (4.5) (Algorithm 1) and the system applied extended flocking algorithm with flock shift mechanism (Algorithm 2). Default’s parameters are set as recorded values. In Algorithm 1 and 2, the positions of virtual agents are set as recorded values.
Result

The error value $e$ is determined as follows here.

\[ e_1 = q_R - q'_R \]  
(4.9)

where $q_R$ is the actual position of red marker and $q'_R$ is the tracked position of red marker.

\[ e_2 = q_G - q'_G \]  
(4.10)

where $q_G$ is the actual position of green marker and $q'_G$ is the tracked position of green marker.

\[ e_3 = q_B - q'_B \]  
(4.11)

where $q_B$ is the actual position of blue marker and $q'_B$ is the tracked position of blue marker.

\[ e = \sqrt{e_1^2 + e_2^2 + e_3^2} \]  
(4.12)

The error value difference between situation 1 and situation 2 with each algorithm
is shown in Fig. 4.22. The average of error value with each algorithm is also shown in Fig. 4.1. X-axis is a time scale and Y-axis is $e^{(2)}(t) - e^{(1)}(t)$ in Fig. 4.22 where $e^{(1)}(t)$ and $e^{(2)}(t)$ are the error values in situation 1 and 2 at time $t$ respectively. The error difference is the maximum value when situation changed from situation 1 to 2 by Default. However, the error difference gets smaller by Algorithm 1 and gets the smallest value by Algorithm 2. Also, the average values in situation 1 are approximately the same by Default, Algorithm 1 and Algorithm 2. On the other hand, the average value is the smallest by Algorithm 2 in situation 2. Therefore, Algorithm 2 adapts the change of the environment the most. From these results, it is clear that the system with fixed camera is able to adapt the change of the environment by applying extended flocking algorithm even though the ideal values, in other words positions of virtual agents, are not changed. Also, implementation of flock shifting mechanism by $\Delta q$ is also effective to improve the

![Diagram of humanoid robot]

Figure 4.12: Composition of humanoid robot

<table>
<thead>
<tr>
<th></th>
<th>situation 1</th>
<th>situation 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>52.315</td>
<td>167.374</td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>53.218</td>
<td>114.901</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>52.468</td>
<td>88.114</td>
</tr>
</tbody>
</table>

Table 4.1: Experimental results of average error values
ability to adapt changes of the environments. For example, the behaviors of three agents with Algorithm 2 in situation 1 and 2 are shown in Fig. 4.23 and Fig. 4.24 respectively. Also, the movement of virtual agents $\Delta q$ is shown in Fig. 4.25. It is observed that each agent not only maintains their distance from each other but tries to change their positions to adjust to the environmental changes. Agents keep moving as whole flock as well as move locally to balance with other agents. In addition, agents move only a little when the error value is small like situation. On the other hand, agents try to adjust by moving bigger when the error value is big like situation 2.

4.7 Conclusion

As the first step for oscillator aggregation control in an autonomous system, the control method using flocking algorithm for autonomous system is proposed in this chapter. Because the original flocking algorithm is difficult to apply to an autonomous system, extended flocking algorithm that expresses arbitrary lattice formation is used. The effect of this control method is confirmed through the experiments that
apply the extended flocking algorithm to the autonomous systems such as a humanoid robot in both simulation and real world and camera tracking system.
Figure 4.16: Graph of $\varepsilon(t)$

Figure 4.17: Line tracer used in this experiment
Figure 4.18: Camera used in this experiment

Figure 4.19: Ideal distance matrix $D$ in this experiment

\[
\begin{bmatrix}
q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 & q_9 \\
q_1 & 0 & 0.35 & 0 & 0 & 0 & 0.017 & 0.49 & 0.49 & 0.49 \\
q_2 & 0.35 & 0 & 0.35 & 0.35 & 0.35 & 0.33 & 0.14 & 0.14 & 0.14 \\
q_3 & 0 & 0.35 & 0 & 0 & 0 & 0.017 & 0.49 & 0.49 & 0.49 \\
q_4 & 0 & 0.35 & 0 & 0 & 0 & 0.017 & 0.49 & 0.49 & 0.49 \\
q_5 & 0 & 0.35 & 0 & 0 & 0 & 0.017 & 0.49 & 0.49 & 0.49 \\
q_6 & 0.017 & 0.33 & 0.017 & 0.017 & 0.017 & 0 & 0.47 & 0.47 & 0.47 \\
q_7 & 0.49 & 0.14 & 0.49 & 0.49 & 0.49 & 0.47 & 0 & 0 & 0 \\
q_8 & 0.49 & 0.14 & 0.49 & 0.49 & 0.49 & 0.47 & 0 & 0 & 0 \\
q_9 & 0.49 & 0.14 & 0.49 & 0.49 & 0.49 & 0.47 & 0 & 0 & 0
\end{bmatrix}
\]
Figure 4.20: Camera tracking experiment in situation 1
Figure 4.21: Camera tracking experiment in situation 2
Figure 4.22: Error value difference of tracking between situation 1 and situation 2

Figure 4.23: The behaviors of $q_1$, $q_2$, and $q_3$ in situation 1
Figure 4.24: The behaviors of $q_1$, $q_2$, and $q_3$ in situation 2

Figure 4.25: The values of $\Delta q$
Chapter 5

Synchronization phenomena observed in flocking algorithm

5.1 Introduction

In previous chapter, the method to control single system using flocking algorithm is proposed. It is also confirmed that this control method is effective for generating homeostatic behavior through several experiment. The main focus of this chapter is why the application of flocking algorithm is effective to generate homeostatic behavior. In the concept, the autonomous system should be considered as oscillators and synchronization phenomena happen. To confirm that the property of this application can be considered as oscillator aggregation, synchronization phenomena in extended flocking algorithm is observed in this chapter.

5.2 Analyze flocking algorithm as synchronization phenomena

In previous chapter, the application of extended flocking algorithm for generating homeostatic behavior through examples of autonomous systems. The application of extended flocking algorithm is effective for generating adaptive behavior. However,
it is difficult to understand why the application of extended flocking algorithm work in such way. Therefore, we try to explain how it works by assuming the process of adaptation of extended flocking algorithm as synchronization phenomena. If extended flocking algorithm can be considered as synchronization phenomena, while the flocking algorithm has been conventionally explained as a consensus problem, it can also be explained as a synchronization of oscillators.

5.3 Agents of flocking algorithm as oscillators

The conventional flocking algorithm is described as a consensus problem. However, in the extended flocking algorithm, the distances between the agents are defined arbitrarily. This makes it difficult to analyze the movement as a consensus problem. Therefore, the movements are described as a synchronization phenomenon instead. If the entire swarm is treated as a synchronization phenomenon of oscillators, then the swarm’s flexible feature may be applied to many different fields. Although the input of agents is defined by Eq. (3.4), if \( f_i^\gamma \) is described as the oscillation term, and \( f_i^g \) and \( f_i^d \) are both described as the interaction terms, then each agent can be described as an oscillator. In other words, the swarm formed by the extended flocking algorithm is generated by the synchronization between oscillators that interact through the interaction term. If the agents can be described as an oscillator, then we should be able to observe the mutual synchronization phenomenon between agents, and, when an external frequency is applied, forced synchronization phenomena should also be observed. In the following experiment, a swarm is generated using the extended flocking algorithm, in order to observe the above-mentioned synchronization phenomena.

5.4 Experimental setting

An experiment is performed to analyze the movement of agents in extended flocking algorithm[91]. Fifteen agents are placed randomly in a two-dimensional space, and are evenly divided into three groups. The groups are referred to as Flocks 1, 2, and 3,
respectively. The agents are divided into groups so that synchronization phenomena between the agents and the groups can be observed. The destinations for each of these groups are referred to as Destinations 1, 2, and 3. The agents of each group define the position of the virtual agent as their corresponding destination. The speed of the virtual agent is set to 0. The destinations are shown in Fig. 5.1. If

![Figure 5.1: Destinations](image)

the distances between agents for Flocks 1, 2, and 3 are $F_1$, $F_2$, and $F_3$, respectively,
then the following equation holds:

\[
  d_{ij} = \begin{cases} 
    50 & (i, j \in F_1) \\
    50 & (i, j \in F_2) \\
    50 & (i, j \in F_3) \\
    100 & (i \in F_2, j \in F_3) \\
    50\sqrt{5} & (i \in F_1, j \in F_2) \\
    50\sqrt{5} & (i \in F_1, j \in F_3) 
  \end{cases}
\]

The movements of each group were analyzed to check if interactive synchronization could be observed. The distances between maxima in behavior of agents are defined as one period. And the phase increases linearly from 0 to \(2\pi\) within that period. The return map[92] is used to check for interactive synchronization. Next, we check for forced synchronization when external frequency \(f_{ext}\) is applied to the right-hand side of input Eq. (3.4),

\[
f_{ext} = 0.5 \sin \omega t
\]

where period \(T_0\) is defined as

\[
T_0 = \frac{2\pi}{\omega} = 100
\]

The return map[92] is also used to check for forced synchronization.

### 5.5 Experimental results

The projection of the centers for each group is shown in Fig. 5.2. The x-position of each group against time \(t\) is shown in Fig. 5.3. The formation of agents at this time is shown in Fig. 5.5. The error between the defined swarm build and the actual swarm build is calculated as

\[
\varepsilon = \sum_i \sum_{j \in N_i} (\|q_j - q_i\| - d_{ij})^2
\]
In Fig. 5.6, error is plotted against time. Here, the swarm moves to eliminate error, and reaches a steady state. In the return map, when a certain agent’s phase, during the $i$th step, is 0, then another agent’s phase is described as $\Delta \theta_i$, and $\Delta \theta_i$ and $\Delta \theta_{i+1}$ are plotted as the x- and y-axes, respectively. Fig. 5.7 is the return map between two agents in Flock 2 and 3, where $i \leq 10$, $10 < i \leq 20$, $20 < i \leq 31$. Fig. 5.8 is the return map between two agents in Flock 2, where $i \leq 10$, $10 < i \leq 20$, $20 < i \leq 31$. In both figures, the phases reach $\Delta \theta_i = \Delta \theta_{i+1}$, which shows that the two agents are synchronized. Mutual synchronization between agents from other groups and between agents within its own group was confirmed. The return map obtained when external frequency was applied is shown in Fig. 5.9. In Fig. 5.9, the phases reach $\Delta \theta_i = \Delta \theta_{i+1}$ and forced synchronization was observed. These results indicate that the agents can be described as oscillators and synchronization phenomena can be generated.

Figure 5.2: Gravity center movements of each flock
5.6 Conclusion

By considering all the agent is oscillators, it is confirmed that extended flocking algorithm is possible to be assumed as synchronization phenomena. Synchronization happens between agents in the same flock, in the different flock and external force.
Figure 5.4: Gravity center y-way movements of each flock

Figure 5.5: Generated formation
Figure 5.6: Error sum of squares
Figure 5.7: Return map between two agents in different groups

(a) $i \leq 10$

(b) $10 < i \leq 20$

(c) $20 < i \leq 31$
Figure 5.8: Return map between two agents in the same groups
Figure 5.9: Return map between an agent and an external force
Chapter 6

Oscillator aggregation control for emergence of homeostasis in robotic systems

6.1 Introduction

Extended flocking algorithm is applied to an autonomous system and it is confirmed that that behavior can be considered as oscillator aggregations in earlier chapters. In the concept, an autonomous system should be considered as oscillators, thus the control method of oscillator aggregation for autonomous system control is proposed based on flocking algorithm in this chapter. This control method is for finding better phase relationships between oscillators and this feature is different from the control method of flocking algorithm that keeps the specific relationships between agents.

6.2 From flocking to synchronization

In previous sections, the application of extended flocking algorithm for generating homeostatic behavior in an autonomous system and the behavior can be describable as synchronization phenomena. The relationships between components of the au-
tonomous system are expressed as ideal distances. On the other hand, oscillators of synchronization phenomena keep phase relationships. We hypothesize that synchronization phenomena is the most important to generate homeostasis and construct minimum algorithm by extracting the element from extended flocking algorithm by oscillators.

6.3 Method of control with the interactions of oscillators

We assumed parameters of an autonomous system to be oscillators after the example of the method of control with a flocking algorithm[90]. We constructed the oscillators to correspond to all parameters. We considered the set of \( n \) oscillators. Let \( x^{(k)} \) denote the state of oscillator \( k \). We define the dynamics of oscillator \( k \) as

\[
\frac{d^2 x^{(k)}}{dt^2} = f_o^{(k)} + f_p^{(k)} + f_s^{(k)} + f_{prob}^{(k)}.
\]

The four terms on the right are determined as follows.

6.3.1 \( f_o \)

Each oscillator is designed based on a van der Pol oscillator. Term \( f_o \) is designed to express the behavior of a van der Pol oscillator. A van der Pol oscillator has dynamics of

\[
\frac{d^2 x}{dt^2} + \lambda(x^2 - 1)\frac{dx}{dt} + x = 0.
\]

We extend this oscillation from around the origin to around \( x_d \) and multiply its amplitude by \( \frac{1}{c} \).

\[
\frac{d^2 x}{dt^2} + \lambda\{c^2(x - x_d)^2 - 1\}\frac{dx}{dt} + (x - x_d) = 0.
\]
Therefore, we assume that oscillator $k$ oscillate around $x_d^{(k)}$ and then

$$f_o^{(k)} = -\lambda \{ c^2 (x^{(k)} - x_d^{(k)})^2 - 1 \} \frac{dx^{(k)}}{dt} - (x^{(k)} - x_d^{(k)}).$$  \hspace{1cm} (6.4)$$

The example of this extended van der Pol oscillator is Fig. 6.1.

![Extended van der Pol oscillator example](image)

Figure 6.1: Extended van der Pol oscillator example of eq. (6.3)

### 6.3.2 $f_p$

To keep the speed at which the state changes in each oscillator, we determine $f_p$ as

$$f_p^{(k)} = c_p \sum_i^n \left( \frac{dx^{(i)}}{dt} - \frac{dx^{(k)}}{dt} \right).$$ \hspace{1cm} (6.5)$$

where $c_p$ is a positive constant.
6.3.3 $f_s$

Interaction is needed to synchronize oscillators. We define simple interaction at this time as

$$f_s^{(k)} = c_s \sum_i^n (x_s^{(i)} - x_s^{(k)}), \quad (6.6)$$

where $c_s$ is a positive constant.

6.3.4 $f_{prob}$

We define the instability of whole aggregation $\varepsilon(t)$ at time $t$. This indicates how much the current state is unperturbed and $0 \leq \varepsilon(t) \leq 1$. Term $f_{prob}$ provides probabilistic fluctuations proportional to its instability $\varepsilon(t)$. $f_{prob}$ is determined as

$$f_{prob}^{(k)} = c_{prob}\varepsilon(t)r, \quad (6.7)$$

where $c_{prob}$ is a positive constant and $r$ is a random number in $(-0.5, 0.5)$. Because of this definition, the state transition probability of the aggregation of oscillators becomes larger in proportion to instability[93]. This definition is typical compared to the example in nature [52].

6.4 Related study

Robot systems that use neural oscillators and central pattern generators (CPGs) have been developed to enable such features to emerge in robotic systems. Some have been developed to generate rhythmic arm motion [94][95], while others have been developed to generate stable quadruped locomotion [96] and bipedal locomotion[97][98]. CPGs indicate the specific block of neurons generate rhythm pattern seen in vertebrates and control the rhythmic locomotions. Vertebrates is able to generate robust and rhythmic locomotion by this mechanism. This mechanism is prepared as neural oscillators when it is used in an artificial system. This is designed by human and all the parameters of this mechanism is determined empirically. To solve this
designing problems, there is a case that uses simulated annealing by assuming this is an optimization problem[99]. On the other hand, the purpose of the algorithm in this paper is to generate adaptive behaviors observed in living organisms that have no neural systems. The case of predictive avoidance behaviors of amoebas is one of the example[52]. The behaviors that maintain their state stable are observed in these living organisms even though they have no neural circuits. We assume these behaviors are to maintain their states determined by specific criteria $h(t)$ stable by dealing with environmental changes and call these behaviors homeostasis in this paper. In this time, $h(t) = \varepsilon(t)$ and we have the autonomous systems generate the behaviors to maintain this criteria small. It is achieved by adding perturbation in proportional to the criteria $h(t)$. This proportional perturbation induce reconstruction of phase relationships between oscillators. This means oscillators change their phase relationship stochastically[93]. However, oscillators try to maintain their relationships when $h(t)$ is small enough as shown in Fig. 6.2. This is because $h(t)$ is too small to have oscillators change their phase relationships. In other words, oscillators search the phase relationship to decrease $h(t)$ by themselves and try to fix the relationship at the point. Therefore, the algorithm proposed in this paper is different from CPGs. It is for generating adaptive behaviors of living organisms that have no neural systems and introducing the mechanism to add perturbation based on its own criteria. Oscillators are induced to reconstruct their phase relationships by this algorithm and it is possible to find better phase relationships to satisfy the criteria accordingly. As a result, CPGs are for generating specific robust locomotion but this algorithm is for generating behaviors to maintain the criteria $h(t)$ small. Thus, these two algorithms are different. The only case to add perturbation in CPGs is the case to add white noise to avoid stuck[100]. The mechanism to add perturbation in proportional to the specific criteria have not been proposed and it is proposed in this paper.
6.5 Experimental result

6.5.1 Isochron

The method to calculate the phase of non-linear oscillator outside of its limit cycle is needed. Isochron is used to calculate the phase outside of the limit cycle[101][102].

In this experiment, the isochron of the fixed range is calculated preliminary and these values are referred. Calculated isochron is shown in Fig. 6.3. This isochron is calculated by following steps.

1. Step 1:
   Calculate some steps until it becomes stabilized on the limit cycle from initial state. The image of this step is shown in Fig. 6.4.
2. Step 2:

Define current state as phase 0. From the state of phase 0, calculate $m$ steps and record the state over 1 cycle. After that, the state of phase 0 is determined.
as start point. Pick up the nearest state from the start point until all the recorded points are picked up. The image of this step is shown in Fig. 6.5.

![Figure 6.5: Calculate isochron: step 2](image)

3. Step 3:

Define the unit phase value $v_u$ as

\[
    v_u = \frac{2\pi}{n_s}
\]

(6.8)

where $n_s$ is a number of recorded points. Each recorded point is allocated index $i$ from the start point in number order. Each phase of recorded point $\theta_i$ is determined as follows.

\[
    \theta_i = v_u i
\]

(6.9)

Calculate fixed $k$ steps from each recorded point and also record the reached point after $k$ steps. These recorded points are called $k$-step points here. The image of this step is shown in Fig. 6.6.
4. Step 4: Divide fixed area into grid. Calculate $k$ steps from each grid and find the nearest $k$-step point. Each phase of grid $\theta_{a,b}$ is determined as the phase of corresponded recorded point of the nearest $k$-point where $a, b$ are the index of the grid. The image of this step is shown in Fig. 6.7.
6.5.2 Result

Based on our concept of this study, the environment and the autonomous system as oscillator aggregations are prepared. This autonomous system is consisted of 5 oscillators. These oscillators are the ones that described in this chapter. The dynamics of oscillators consisting this environment is determined as follows.

\[
\frac{d^2 x^{(k)}}{dt^2} = f_o^{(k)} + f_s^{(k)} + f_{\text{prob}}^{(k)}. \tag{6.10}
\]

\[
f_o^{(k)} = -\lambda \{ c^2 (x^{(k)} - x_d^{(k)})^2 - 1 \} \frac{dx^{(k)}}{dt} - (x^{(k)} - x_d^{(k)}). \tag{6.11}
\]

\[
f_s^{(k)} = \sum_i c_s^{(i)} (x_s^{(i)} - x_s^{(k)}). \tag{6.12}
\]

where \(c_s^{(i)}\) is a positive constant.

\[
f_{\text{prob}}^{(k)} = c_{\text{prob}}^{(i)} \varepsilon(t)r. \tag{6.13}
\]
where $c_{prob}^{(i)}$ is a positive constant and $r$ is a random number in $(-0.5, 0.5)$. Each $c_{s}^{(i)}$ and $c_{prob}^{(i)}$ is set randomly. The environmental oscillator aggregation and the autonomous system oscillator aggregation are independent except two nodes. Two oscillators are selected from each oscillator aggregations and they are connected one by one. Two oscillators in the autonomous system is affected by the oscillators in the environment according to the eq. 6.6. On the other hand, two oscillators are not affected. First, we confirm the effect of probabilistic perturbation. 15 oscillators are prepared for the environment and 5 oscillators are prepared for the autonomous system. Instability $\varepsilon(t)$ is defined as the difference of phase between the 4th and 5th oscillator in the autonomous system. $\varepsilon(t)$ is 0 when the 4 and 5th oscillators are the same phase. Thus, this autonomous system is expected to try to approximate two oscillator’s phases. We experimented 100 times and averaged and smoothed results are shown in Fig. 6.8. When $c_{prob} = 0$, the average value of $\varepsilon(t)$ gets the biggest

![Graph of $\varepsilon(t)$ in various $c_{prob}$ values](image)

Figure 6.8: Graph of $\varepsilon(t)$ in various $c_{prob}$ values

value. However, $\varepsilon(t)$ values decrease in proportional to the increase of $c_{prob}$ On the
other hand, the average $\varepsilon(t)$ value gets worse when $c_{\text{prob}} = 500$ From these results, it is clear that $f_{\text{prob}}$ term works to induce oscillators to reconstruct for searching better phase relationships. Although, $f_{\text{prob}}$ term does not work when the value is too big and the effect is too strong. Next, number of oscillators in the environment is changed. Results are shown in Fig. 6.9. These results are also averaged and smoothed. $n_E$ means the number of oscillators in the environment. These results are also the same even though $n_E$ changes. Thus, the complexity of the environment is not relative the ability of this algorithm. In other words, the autonomous system is able to adapt the environment even though the environment is complicated. Finally,

![Figure 6.9: Graph of $\varepsilon(t)$ in various $n_E$ values](image)

number of oscillators in the autonomous system is changed. Results are shown in Fig. 6.10. These results are also averaged and smoothed. $n_R$ means the number of oscillators in the autonomous system. When the number of oscillators $n_R$ is under 30, instability $\varepsilon(t)$ becomes small soon. And $\varepsilon(t)$ values become approximately the same finally. This result indicates that adaptation ability and its speed to adapt is
free from the complexity of the autonomous system. However, reconstruction does not work when $n_R$ is 100. This result means there is a capacity of oscillators to search the better state by reconstruction and stabilization.

### 6.6 Conclusion

The control method of oscillator aggregation for an autonomous system control is proposed in this chapter. This method is constructed to generate the adaptive behaviors of living organisms that have no neural circuit such as amoebas. The system applied this control method tries to keep its own criteria $\varepsilon t$ small by dealing with the change of the external environment. It is also confirmed that this control method has the features that is useful to generate homeostatic behavior in complex and dynamic environment based on reconstruction and stabilization in the concept.

![Graph of $\varepsilon(t)$ in various $n_R$ values](image)

Figure 6.10: Graph of $\varepsilon(t)$ in various $n_R$ values
Chapter 7

Oscillator aggregation in redundant robotic system

7.1 Introduction

The control of oscillator aggregation proposed in previous chapter is applied to the robot to confirm the effect of the method. Redundant arm robot is prepared for the experiment and the control method is applied to this robot. It is difficult to control this redundant robot analytically and confirm whether it is possible to control by this method and it is effective to generate homeostatic behavior.

7.2 Redundant arm robot

Redundant arm robot is prepared for the experiment. The screenshot is shown in Fig. 7.1. This robot is made of 20 servos (Fig. 7.3) and rollers are set under the servos (Fig. 7.2). The servo is Kondo KRS4031HV. The length of 1 unit of the robot is 10(cm) and its weight is 100.5(g). Maximum torque is 13.0(kg·cm).
7.3 Implementation for redundant robotic system

It is difficult to separate the state of oscillators into synchronized and asynchronous states under noisy environments[93]. It is also difficult to evaluate the states of robots only from the states of oscillators for this reason. Therefore, we defined the task of evaluating the effect of this mechanism in robots by converting it to an observable phenomenon. Let us consider a robot with 20 joints. Each joint is a servo and this robot has a link mechanism that moves in 2-Dimensional space. Each joint can move at angles between $-15^\circ$ to $15^\circ$. The main task is to move the head of the robot closer to the object (Fig. 7.4).

### 7.3.1 Convergence and searching mechanism to evaluate locomotion

We introduced two additional mechanisms to this robotic system. First, we introduced a mechanism to converge oscillation according to closeness to the object. Coefficient $c$ in Eq. (6.4) at time $t$ is represented by using instability $\varepsilon(t)$ at time $t$
Figure 7.2: Roller of robot

\[ c_a(t) = \frac{a_b}{(a_a/a_s) \varepsilon(t) + 1} \]

(7.1)

where \( a_b > a_s > 0 \). \( a_b \) and \( a_s \) are positive constant. These \( a_b \) and \( a_s \) are the coefficients that determine the maximum and minimum amplitude respectively. The amplitude becomes \( \frac{1}{a_b} \) times when \( \varepsilon(t) \) values is the minimum value 0 and becomes the minimum. When the value \( \varepsilon(t) \) is the maximum value 1, the amplitude becomes \( \frac{1}{a_s} \) times and becomes the maximum. We also introduced a mechanism to evaluate locomotion following the example of the method of hill climbing. The amount of change in \( x_{d}^{(k)} \) at time \( t \) is determined as

\[ \Delta x_{d}^{(k)}(t) = \left(1 - \frac{\varepsilon(t)}{\varepsilon(t - 1)}\right) \{x_{d}^{(k)}(t) - x_{d}^{(k)}(t - 1)\} \]

(7.2)

We expected that oscillators would maintain specific phase relationships by interaction at time \( t \) by introducing these mechanisms. This is expressed as the specific locomotion by a robot in the real world and its head moves on a specific limit cycle. Then, Eq. (7.2) is used to evaluate locomotion and it expresses which direction to
Figure 7.3: Servo of robot

go to get closer to the object with the specific relationships among oscillators at time $t$. The distance from the object is expressed as potential (Fig. 7.5) and the potential is radially and monotonically increasing so it is possible to reach the object with the method of hill climbing. However, if the oscillators do not maintain a specific relationship, such an effect is not expected by hill climbing and the number of possible angle combinations is $30^{20}$. Therefore, it is difficult to reach the object in an executable period of time.

7.3.2 Allocation of servos and sensors

We allocate the state of oscillator $k$ to the servo angle to apply the behaviors of oscillators to robot control as shown in Fig. 7.6. Let $\theta_{\text{min}}, \theta_{\text{max}}$ denote the minimum and maximum angles. Then, servo angle $\theta_k$ is represented by

$$\theta_k = x^{(k)}(\theta_{\text{max}} - \theta_{\text{min}}) + \theta_{\text{min}}$$  \hspace{1cm} (7.3)

There are 20 servos and we prepare the same number of oscillators. We also make the oscillators act as sensors. The state of oscillators acting as sensors is denoted
Figure 7.4: 20-Dof robot and object

as $x_s(t)$ at time $t$. The distance between the object and the head of the robot is denoted as $d(t)$ and we let $d_{\text{min}}, d_{\text{max}}$ denote the minimum and maximum distances. Then, $x_s$ is represented as

$$x_s(t) = \frac{d(t) - d_{\text{min}}}{(d_{\text{max}} - d_{\text{min}})} \quad (7.4)$$

The robot is controlled by these 21 oscillators and instability $\varepsilon(t) = x_s(t)$ in this case.

7.3.3 Phase calculations using isochron

Cases must be considered where oscillators are not on the limit cycle when calculating the phase of oscillators. An isochron is used for phase calculations\cite{101}\cite{102} to deal with this, where the isochron is preliminarily calculated for a fixed area and the actual phases of oscillators are obtained by using a reference. When the phases
of oscillators are obtained, \( x' \) is defined as

\[
x' = c_o(x - x_d)
\]  

(7.5)

and the phase of \( x' \) is obtained by using a reference.

### 7.4 Experimental result

To confirm the effect of this algorithm in a real robot system, we experimented in 4 cases.

- (A) Default situation
- (B) The case with obstacle
- (C) The case that a part of the wheel of robot is broken
- (D) The case that robot have to reach the object non-monotonically because of its physical constraint

Also, each coefficient is as follows.
7.4.1 (A) Default situation

The default situation is defined as Fig. 7.7. In Fig. 7.7, $x$ and $y$ indicate the axis of 2 dimensional coordinate system and this $x$ is nothing to do with the state $x$. Time variation of the distance between the object and the head of the robot is shown in Figure 7.6: Allocation

- $\lambda = 1.0$
- $a_b = 30$
- $a_s = 2.5$
- $c_p = 0.1$
- $f_s = 0.2$
- $f_{prob} = 0.1$
Fig. 7.8. This graph is the average value of 5 times. x-axis indicates time step. Unit time is $\Delta t = 0.02(s)$. Min in Fig. 7.8 is the minimum distance between the object and the head of the robot. Screenshots of this experiment is shown in Fig. 7.9. Also, the locomotion of the object and the head of the robot is shown in Fig. 7.10. In initial phase, the robot locomotes from the initial state and appears to back away from the object. After a while, it starts to reach the object and repeats the same oscillation near the object. Next, the examples of the state of oscillators are shown in Fig. 7.11.

A part of $x^{(k)}_d(t)$ is shown in Fig. 7.12. X axis of this graph also indicates the time(unit time: $\Delta t = 0.02(s)$). Oscillators generated the attitude that is shown in Fig. 7.9(c). It is observed that oscillators reconstruct their phase relationship frequently in early phase and maintain their oscillation small by repeating adjusting after their state became stable enough. Next, we confirm whether synchronization of oscillators induced. Return map is used to judge synchronization[92]. In the return map, when a certain agent’s phase, during the $i$th step, is 0, then another agent’s phase is described as $\Delta \phi_i$, and $\Delta \phi_i$ and $\Delta \phi_{i+1}$ are plotted as the x- and y-axes, respectively. 21 oscillators are expected to repeat the synchronous and asynchronous
Figure 7.8: Distance between object and arm

state complicatedly and it is difficult to separate these state but it is observed that the state that synchronization appeared to be occurred like inside the circle in Fig. 7.13. Also, these locked phase of synchronization is not always the same and not always in phase synchronization. There are several locked phase like $5.5 \sim 6.0\text{ (rad)}$ or $0.0 \sim 1.0\text{ (rad)}$ and it is observed that these oscillators constructs complicated phase relationships.

7.4.2 (B) The case with obstacle

Next, we experiment in the case with obstacle. The situation is the same as experiment (A) except an obstacle. The situation is shown in Fig. 7.14. This obstacle never be changed by the effect of the robot and is not recognized by the robot. Therefore, the body of the robot does not move as it is expected when the robot reaches the specific position. We assume that these effects of the obstacle as disturbance and external force of the robot and we confirm the behaviors of robot to
Figure 7.9: Acquired attitude of robot at each phase
Figure 7.10: Positions of object and the head of robot
Figure 7.11: Examples of oscillator state
stabilize their state by self parameter tuning when it is affected by disturbance and external force. In this time, $\varepsilon(t)$ is discontinuous because its body does not move as it is expected. The distance between the object and the head of the robot is shown in Fig. 7.8. This graph is also the average of 5 times. It is confirmed that it takes longer to get closer to the object because of the obstacle and it backs away sometimes. However, it reaches the object eventually even though there is an obstacle. It becomes harder for the robot to move because the broken wheels get stack with the floor in this case. $\varepsilon(t)$ changes continuously compared with the case (B) but different from case (A).

7.4.3 (C) The case that a part of the wheel of robot is broken

Next, we experiment in the case a part of wheel of robot is broken. Position of the robot and the object is all the same with the situation (A). In this case, dynamics of
Return map between oscillators 6 and 14

Return map between oscillators 13 and 4

Figure 7.13: Return map examples
the robot changes compared with the case (A). Thus, we confirm whether the robot is able to adapt and deal with such changes. The 4th and 8th wheel from the head of the robot is fixed not to rotate this time. The distance between the object and the head of the robot is shown in Fig. 7.8. This graph is also the average of 5 times. Because of the broken wheels, dynamics of the robot changes. It becomes harder for the robot to reflect the movement of servo to locomotion, the robot reached the object though.

7.4.4 (D) The case that robot have to reach the object non-monotonically because of its physical constraint

Oscillators keep specific phase relationship when they are synchronizing. However, there are several possible phase relationship at each time and the phase relationship changes stochastically. Therefore, the phase relationship is multi stable. The
probability to transit to the other phase relationship becomes bigger in proportional to the perturbation term $f_{prob}$. We experiment in the situation that initial attitude of the robot is changed like Fig. 7.15 to confirm the multi stability of phase relationship. The distance between the object and the head of the robot is shown in Fig. 7.8. This graph is also the average of 5 times. The distance does not decrease and it was observed that the robot could not decrease the distance even though it is able to decrease the distance if it moves to the direction $D_2$ in Fig. 7.15. This is because of the angle limitation of servos of the robot and physical constraint to collide with the root of its own body. The initial position becomes quasi-stable for this reason and the robot could not reach the object from the $D_1$ direction. It is expected that perturbation added in this setting is too weak to transit to the other phase relationship.

Figure 7.15: Initial state of experiment D
7.4.5 Discussion

Adaptation to the environment

(B) is able to be considered as the case of difference of external environment and (C) is able to be considered as the case of difference of internal environment in this experiment. From the result of (A) that the robot could reach the object, it is confirmed that the adaptation to static environment and a static object. Also, form the result of (B) and (C), the case of environmental change, the robot was possible to adapt by changing its own state when environment changes. The criteria $h(t)$ is the distance between the object in this time, thus the robot is able to maintain the criteria $h(t)$ small. From these result, it is confirmed that it is able to generate the adapting behavior to maintain the criteria $h(t)$ small, the definition of homeostasis in this paper, under the static environment by using the mechanism to have oscillators reconstruct and stabilize the phase relationship by adding perturbation in proportional to the criteria. On the other hand, the case that the robot is not able to deal with because of the behavior to find better phase relationship such as (D) is confirmed. However, this phenomenon is expected to be related to the implementation of this experiment and it can be solved as ordinal searching problem by adding bigger perturbation.

Effect of $\Delta f_p^{(k)} + \Delta f_s^{(k)} + f_{prob}^{(k)}$ and robot locomotion

We confirm the relationship between reconstruction/stabilization of oscillators and the position of robot. $\Delta f_p^{(k)} + \Delta f_s^{(k)} + f_{prob}^{(k)}$ and the distance between the object and the head of the robot is shown to confirm the relationships. The distance between the object and the head of the robot and $\Delta f_p^{(k)} + \Delta f_s^{(k)} + f_{prob}^{(k)}$ in (A), (B) and (C) are plotted in fig. 7.16, 7.17 and 7.18 respectively. These are results of five times in each situation. In every situation, $\Delta f_p^{(k)} + \Delta f_s^{(k)} + f_{prob}^{(k)}$ is decreased near 75 cm. These results indicate that oscillators start to make up the transition from reconstruction to stabilization of the phase relationships near 75 cm. Distance between the object and the head of the robot of one trial in (A) is shown in Fig. 7.19. Also, the locomotion of the robot near 75 cm is shown in Fig. 7.20. Firstly,
Figure 7.16: The relationship between $\Delta f_p^{(k)} + \Delta f_s^{(k)} + f_{prob}$ and distance in situation (A)

Figure 7.17: The relationship between $\Delta f_p^{(k)} + \Delta f_s^{(k)} + f_{prob}$ and distance in situation (B)
the locomotion is bigger, though after a while, the locomotion became smaller.

7.5 Conclusion

The control method of oscillator aggregation is applied to the redundant arm robot. Through several experiment, it is confirmed that this control method is possible to control redundant arm robot event though it is difficult to control analytically. Also, it is observed that the robot with this method can deal with internal and external changes of surrounding environment.
Figure 7.19: One example of distance between object and arm in an experiment of (A)
Figure 7.20: Locomotion of the head of robot
Chapter 8

Summary

The purpose of this study is to generate homeostatic behavior in an autonomous system. Recent intelligent robots were summarized in chapter 1. The robots that need adaptive behavior in dynamic environment were introduced in this chapter. In chapter 2, adaptive behaviors observed in living organisms was denoted. Based on such facts, the concept that assumes both the environment and the autonomous system as oscillator aggregations was proposed. The state of the autonomous system was expressed as the relationships between oscillators in this time. As the first step to construct the oscillator aggregation control method for autonomous systems, flocking algorithm in multi-agent system was adopted as the representative of distributed algorithm that is robust and flexible in chapter 3. It is difficult to apply flocking algorithm to an single autonomous system because flocking algorithm is for multi-agent systems. Therefore, flocking algorithm was extended for applying to an autonomous system by expressing arbitrary lattice formation. In chapter 4, several experiment to confirm the effect of application of extended flocking algorithm to an autonomous system for generating adaptive behavior were illustrated. The case of a simulation robot and a humanoid robot in real world were shown at first. The task of each robot was to maintain their standing posture by dealing with external forces and disturbances. Both robot could maintain standing posture appropriately. Also, the camera tracking system in the environment that its brightness changes was shown. This camera system was applied extended flocking algorithm to follow the
brightness changes to keep tracking the target markers. This system also could keep
tracking even though the brightness of surrounding environment changed. Chapter
5 illustrated the result of analysis of the feature of extended flocking algorithm as
synchronization phenomena. Each agent of extended flocking algorithm could be
assumed as oscillators and their behaviors were possible to assume as synchronization
phenomena. Based on these results in earlier chapters, the control method of
oscillator aggregation for autonomous system was proposed in chapter 6. The au-
tonomous system was assumed as oscillator aggregation and its state was expressed
as phase relationships between oscillators. The mechanism to reconstruct and sta-
ilize their phase relationships depending on the situation was introduced in this
control method. The fundamental features were shown through simulation exper-
iments. The experimental results of application of the control method in previous
chapter were shown in the last chapter. Redundant arm robot was prepared and
was applied the control method. This robot applied the control method was possible
to adapt the internal and external changes.
Acknowledgements

The author especially expresses to thank Professor Keiji Suzuki of Hokkaido University for instruction and guidance from the inception to the completion of the present research. Deep thanks are also due to Professor Masahito Kurihara, Professor Tetsuo Ono and Professor Masahito Yamamoto of Hokkaido University for beneficial comments to compile this study. He also wishes to thank Associate Professor Hidenori Kawamura of Hokkaido University for profitable discussions and suggestions. Many pieces of affectionate advice and discussion given by them were very useful to improve this research. He would like to thank all students of the office, Harmonious Systems Engineering, Complex Systems Engineering. The conversations between them often intenerated, and sometimes encouraged him. Finally, he appreciates all that his family, especially my parents, Akira Yamauchi and Kikumi Yamauchi, had done for supporting his life.
Bibliography


