BioRobotics Laboratory

Online Optimization for the locomotion of RoomBots

Semester project

NGUYEN The Anh(th阮.mdnguyen@epfl.ch)
June 8th, 2012

Supervisors:
Rico Möckel, Stéphane Bonardi, Soha Pouya, Massimo Vespignani

Professor Auke Jan Ijspeert
Abstract

This semester project focuses on the problem of online locomotion learning in Roombots, a modular robot developed at the BioRobotic Laboratory of EPFL- École Polytechnique Fédérale de Lausanne. The online learning locomotion controller includes two components: a Central Pattern Generator (CPG) and a stochastic, gradient-free optimization algorithm, Particle Swarm Optimization (PSO). The CPG is implemented as a system of coupled nonlinear oscillators in Roombots, which has 3 degrees of freedom, each of which supports two type of movements: rotation and oscillation. Online learning involves optimization of the fitness value which is the speed of a certain locomotion or the travelling distance in a period of time. We used the PSO optimization method to find a set of CPG parameters of a gait that maximize the fitness value. CPG and PSO were implemented directly on Roombots and the optimization process was performed by themselves.

The main objective of the project is to fully make online locomotion learning with CPG and PSO optimization are implemented in a distributed fashion on physical Roombots in the real environment with fitness values calculated by tracking system. During the experiments, we used a simplest structure, called meta-module which is made of two modules. We used 10 PSO particles to find the best solution for 6 parameters of CPG model. The result shows that a good locomotion gait can be obtained in a few iterations and the results started to converge to the best one after a large number of iterations, which is larger than 10. An experiment of 10 PSO particles and 10 iterations usually last for over 4 hours. The explored good gaits are similar to the locomotion patterns of animals in nature such as worm, snake.

After obtaining good gaits, a number of experiments were conducted to evaluate the gaits’ robustness against changes in initial states and environments such as friction, slope and obstacles. The results reveal that initial state is the most affective factor to the gait’s performance.
Acknowledgement

I would like to thank Rico Möckel for his continuous guidance and support along this project. His advices during the project were greatly valuable for my work. I would like thanks also to Stéphane Bonardi, for his endless support on different aspects related to devices, software, Linux, and others. I would like to thank Massimo Vespignani for his mechanical support. Without his support, the Roombots could not has been fixed after many times of being broken during the experiments. Moreover, I also would like to thank Soha Pouya for her willing support and advice about optimization and simulation. Finally, I would like to thank Professor Auke Jan Ijspeert for giving me the opportunity to do this semester project and have a great time at the BioRobotic Laboratory.
## Contents

1 Introduction .......................................................... 3  
1.1 Motivation .................................................................. 3  
1.2 Objectives .................................................................. 3  
1.3 Contribution .............................................................. 4  
1.4 Report Structure ........................................................ 4  

2 Background ................................................................. 5  
2.1 Modular Robotics ....................................................... 5  
2.2 Roombots ................................................................. 5  
2.2.1 Specifications ........................................................ 5  
2.3 Controller Architecture ............................................... 7  
2.3.1 CPG Model .......................................................... 7  
2.4 Optimization methods .................................................. 8  
2.5 Collision check ........................................................... 8  

3 Implementation ............................................................ 9  
3.1 Firmware ................................................................... 9  
3.2 Experiment Software ................................................... 10  
3.2.1 Tracking System ..................................................... 12  
3.2.2 Centroid Detection ................................................ 15  
3.2.3 Fitness Value ........................................................ 15  
3.3 Experiments ............................................................... 15  
3.3.1 Experiment Environment ......................................... 15  
3.3.2 Experiment Setup .................................................. 15  
3.3.3 Experiment Procedure ............................................ 16  
3.3.4 Difference between simulation and real experiment configuration ................................. 18  
3.3.5 Choice of fixed parameter ........................................ 19  

4 Results & Discussion ....................................................... 20  
4.1 Results ..................................................................... 20  
4.1.1 Experiment 1 ........................................................ 20  
4.1.2 Experiment 2 ........................................................ 21  
4.1.3 Experiment 3 ........................................................ 23  
4.2 Gaits description ........................................................ 25  
4.3 Gaits Evaluation & Discussion ...................................... 26  
4.3.1 Different frictions evaluation ................................. 27  
4.3.2 Different initial states evaluation ............................ 27  
4.3.3 Obstacle evaluation ............................................... 28  
4.3.4 Slope evaluation .................................................... 30  

5 Conclusion and Future works ........................................... 32
1

Introduction

Similar to natural beings which are constituted from fundamental cells, robots can also be constructed from fundamental modules. Modular robotics systems offer the possibility to configure different robotic structures from multiple modules. The ability of self-configuration which allows modular robots to change their structure to adapt to new environments or recover from damage generates a number of gaits in different morphologies. However, designing an efficient locomotion for a certain morphology is a difficult and unsolved problem because of the high number of degrees of freedom and complex configurations. Given a robotic structure under a certain environment, it is not straightforward to design a best gait, in particular a robust gait. There are many aspects related to the term “best” gait. For example, the ability to move forward with low energy, low possibility to damage itself, the robustness on the initial state and surrounding environments. In the project, we simply did the optimization based on how fast the robot can move from a particular initial position and finally analyzed the performance of the gait discovered.

The project was carried out at the BioRobotic Laboratory of EPFL- Ecole Polytechnique Federale de Lausanne. In this project, we investigated the ability of locomotion online learning in Roombots. Roombots is a state-of-the-art modular robot designed in BioRobotic Lab. Unlike other modular robots, Roombots supports both oscillation and continuous rotations. By offering multiple types of movement, a certain structure constituted by many Roombots offers complex movements, which is not easy to design a best locomotion for this. In this project, a meta-module of Roombots will run learning algorithm in real time in order to obtain the best gait – the fastest one – with one initial orientation respect to the ground in the experiment conditions. The results of the online learning finally will be compared with the simulation results that have been achieved in the past and analyzed the robustness against various factors.

1.1 Motivation

The purpose of the project is to investigate Plug-and-play robots that can learn locomotion on their own and can adapt to different environment conditions with on-line learning.

1.2 Objectives

In this project, I implemented a software and improved the hardware code inside Roombots in order that the whole process of carrying out experiments of online learning with Roombots become easier and faster. The software includes three components:

1. Tracking System: Using Kinect as a sensor and Computer vision algorithm to process image data.
3. Experiment Section: Using buttons and mouse clicks to conduct experiments in a more convenient way.
The final goal is to reduce the time of the whole online learning process with Roombots experiments to find the best gait for a certain configuration of Roombots. The framework in the project includes three components: a distributed locomotion controller CPG, a stochastic optimization algorithm PSO, and a tracking system using Microsoft Kinect. CPG and PSO was implemented in Roombots, an excellent modular robot platform. Roombots allows us to build a complex structure and investigate a number of locomotion patterns with high degree of freedom and multiple types of movements. CPG are proved to be ideal building blocks for constructing locomotion controllers for a self-reconfigurable modular robot with on-line learning ability [1]. The concept of central pattern generators (CPGs) was found in the spinal cord of vertebrate animals. The CPGs include coupled phase oscillators to ensure synchronized behaviour. PSO, which was selected because of its robustness over local minima, optimizes the parameters in the CPG topology such as amplitude, offset and coupling phases of oscillators.

1.3 Contribution

The main contributions of the project are as follows:

- Implemented a software with user-friendly Graphical User Interface including the tracking system. The software reduces the workload in doing the experiments. As the limitation of the battery, we only can run the experiment in 3 or 4 iterations with a fully charged battery pack. With the support of the software, we are able to overcome battery drawbacks and reach to a further iterations in order to obtain the final optimization results. The software allows us to load /save perfectly the state of PSO optimization process in order to pause/ resume the experiment.

- Discovered good gaits using the experiment software and conducted a number of gait analysis. The work continued the previous work of Wilhelm Frédéric and Soha Pouya.

- Offered a convenient experiment platform for further researches and studies in locomotion online learning in Roombots.

1.4 Report Structure

The structure of the report is as follows: Firstly, previous work in modular robots and state-of-the-art work related to locomotion in modular robots is discussed. The second part is about software implementation. The third part demonstrates the experiments conducted, and the fourth part presents the obtained results and discussions for further research.
2

Background

In this part, previous projects related to Modular Robotics, Roombots specifications, CPG model and optimization algorithm will be discussed.

2.1 Modular Robotics

Modular robots, generally, are simple identical elements designed to build up more complex structures. In recent years, modular robotics has been a challenging and attractive research topic. There has been a great number of modular robotic platforms that were presented, for example, M-TRAN II[2], CONRO robot [3], Polybot [4], Morpho [5] and YAMOR [6]. Fig. ?? shows some examples of modular robots. They are chain-type robots which locomote in a pre-defined configuration and adapt to the environment. However, they have limitations either in their degree of freedom or in the supporting movements. Moreover, the modular robots have been industrialized with some mass products from Adaptronics (R), ATRON. The recent modular robot created by BioRob, Roombots, is an advanced one since it possesses 3-DOF and supports multiple types of movements.

There are some existing approaches to generate gait for a modular robotic system. The very first solution is time-driven which was proposed by Yim [7]. CPG method was proposed in [1] and [8]. In [9], CPG and PSO were used to investigate the performance of hybrid solution (a combination of rotation and oscillation) on locomotion online learning. However, the performance is evaluated on simulation only and it is a big gap from the reality. The recent semester project done by Fédéric Wilhelm in BioRob started to transfer the simulation work to real physical world with some initial results. However, the work had some limitations. As a continuation of the previous work, I built an experiment software and investigated further about the gait obtained through online learning. The PSO was used as the inspiration of the simulation work [9] and the hardware implementation work from Wilhelm [10].

2.2 Roombots

During this project, we worked on the Roombots robotic platform, developed here, at the Biorobotics Laboratory of EPFL. Roombots are reconfigurable robots designed to be adaptive furniture, like a table shown on Fig2.1. In this section, we will describe the main properties of the Roombots. For more details, you can refer to [11], [12].

2.2.1 Specifications

One Roombots module is made of two rounded cubes as in Fig. 2.2. Each module has three DoF (Fig. 2.2.(b)), and each exposed face can be connective to another module. Each of the three DoFs is controlled by a servo-mechanism. It uses a 10 mNm DC-motor (FH2232 from Faulhaber ®) for the actuation of the center DoF, and a 16 mNm DC-motor (FH2342) for the two outer DoFs [11]. Each module is controlled by a 16-bit
Digital Signal Controllerds PIC33FJ128MC802 from Microchip®, which sends commands to the motors and receive commands by Bluetooth, using Radio Frequency Communication protocol (RFCOMM). In this project, we decided to use a meta-module. For one meta-module, there are 4 types of configuration can be built as shown in Fig. 2.3. For the entire project, we only focus on the PER configuration, where the two connected cubes have perpendicular internal DoF. In the previous work, CPG and PSO was implemented on Roombots. However, there was lack of a Graphical User Interface software to communicate with Roombots.
Inputting commands manually consumed time and energy.

### 2.3 Controller Architecture

CPG network consists of a number of coupling oscillators. Control inputs for the CPG are amplitude, offset and coupling phase. Each oscillator is capable of generating either oscillation or rotation. Besides, the common parameters for all oscillators are frequency $\omega$ and convergence amplitude $a$.

#### 2.3.1 CPG Model

Thanks to the mechanical design of Roombots, each CPG controller can produce two types of basic movements for each DOF:

- Rotational movements that continuously rotate with increasing angle.
- Oscillatory movements that periodically oscillate.

The controller is built as a distributed system of coupled phase oscillators in order to keep the synchronization of all DOF. The equations for each oscillator (DOF) are shown in Equations (2.1) and (2.2):

\[
\dot{\phi}_i = 2\pi \cdot \omega_i + \sum_j w_{ij} \cdot r_j \cdot \sin(\phi_j - \phi_i - \psi_{ij}) \tag{2.1}
\]

\[
\dot{r}_i = a_i (R_i - r_i) + f_r(s) \tag{2.2}
\]

\[
\theta_i = r_i \cdot \sin(\phi_i) + X_i \quad \text{(Oscillation)}
\]

\[
\hat{\theta}_i = \phi_i \quad \text{(Rotation)}
\]

\[
\theta_i = X_i \quad \text{(Locked)}
\]

where $\theta_i$ is the servo input which can be derived with different functions corresponding to the desired servo movement. Variables $r_i$ and $\phi_i$ are state variables which represent amplitude and phase of the oscillation. The parameters $w_{ij}$ and $\psi_{ij}$ are respectively the coupling weight and phase bias of the coupling between oscillators i and j. $a_i$ is a positive constant which determines convergence rate to the desired value $R_i$. The parameters $R_i$, $X_i$, are amplitude and offset for each oscillator [13]. In this project, the CPG model for the meta-module contains 6 oscillators corresponding to the 6 DOFs as illustrated in Fig.2.4. In the scope of this project, we only deal with oscillations. Furthermore, we follow the convention of previous semester project to reduce the number of parameters for the optimization process, namely:

- Same frequency for all oscillators: $\omega_i = \omega$
- Anti-symmetrical coupling phase: $\phi_{ij} = -\phi_{ji}$
- Same coupling strength: $w_{ij} = w$
- Same amplitude convergence: $a_i = a$

![Figure 2.4: Structure of CPG model for a meta-module. The figure is taken from the semester report of Fédéric Wilhelm](image)
2.4 Optimization methods

This section introduces the optimization method implemented in the Roombots, which is Particle Swarm Optimization (PSO). We want to perform online optimization in which one evaluation of the fitness function corresponds to a physical experiment for a given gait in a period of time.

PSO is a stochastic, population based optimization method. In the PSO optimization method, a population of candidate solutions –particles- collaborate with each other and move in the search space with a velocity and a position. A position and velocity represent the particle’s parameter values and search direction respectively [14]. Each iteration, the velocity and position of a particle are updated by the Equations (2.3) and (2.4)

\[
\vec{v}_i(t + 1) = K \cdot [\vec{v}_i(t) + c_1 r_1 (\vec{p}_i - \vec{x}_i(t)) + c_2 r_2 (\vec{p}_g - \vec{x}_i(t))] \quad (2.3)
\]

\[
\vec{x}_i(t) = \vec{x}_i(t - 1) + \vec{v}_i(t) \quad (2.4)
\]

Where \(\vec{v}_i(t)\) is the velocity vector, \(\vec{x}_i(t)\) is the position vector at iteration \(t\). \(K\) is a constriction factor, \(c_1\) is the cognitive factor, and \(c_2\) is the social factor. \(r_1\) and \(r_2\) are two random number in the range of \([0, 1]\). \(p_i\) is the best known position of particle \(i\) and \(p_g\) is the global best known position vector. According to the [14], the \(c_1\) and \(c_2\) are selected such that \(c_1 = c_2 = 2\). Moreover, the adaptive inertia weight is applied as in [15].

The initial value of \(w\) is \(w_0 = 0.9\) and the final value is \(w_n = 0.4\). According to [9], PSO is a good choice for locomotion online learning in Roombots.

As a continuation of previous semester project done by Wilhelm, CPG and PSO were already implemented, the main work in dealing with firmware includes:

- Fixing bugs in Roombots firmware. Making commands and functions are consistent.
- Adding more features and communication commands to Roombots module.

2.5 Collision check

Since the experiment involved physical Roombots running on real environment, the collision check is critical to avoid Roombots damage itself. The collision check implemented concerns only internal collision, i.e. collision between Roombots cubes. The collision between Roombots and ground is minimized by using the plastic rubber foam floor. The collision has been implemented by Stéphan Bonardi.
3

Implementation

3.1 Firmware

Even CPG and PSO were already implemented, there are still critical bugs which consumed time and a number of tests to identify and fix. Moreover, I have added more commands and features to the CPG and PSO. The work can be summarized here:

1. Fixed bugs in Roombots Firmware.
   - There are some critical bugs in the given firmware code. After several experiments and debugging, the CPG can perfectly perform its function. Due to a mistake of type casting, Roombots could not perform CPG as in expectation.
   - The code was modified to make a synchronization between simulation and collision detection.
   - In the previous version, Roombots return to its initial state in a straight form after performing CPG. This behaviour leads a wrong distance measurement as the position can be change when the body of metamodule come back to initial state such as falling or rolling. The code was changed such that after running CPG for a pre-defined experiment time, the Roombots will not return immediately to the reset state but stop there for the distance measurement from software. In the next step, when PSO change to the next particle, the Roombots will be reset and start to run CPG again. By doing in this order, the travelled distance of Roombots can be measured exactly by the tracking system.
   - There are also some wrong commands which cause a difference between what users want and what Roombots perform. For instance the code of setting convergence rate and frequency.
   - The most important piece of code has been added into the firmware of the Roombots is the one allows users to set which CPG parameters will be used in the optimization.

The convention of parameters ID is stated as in Table. 3.1 with N oscillators, here $N = 6$

By using command $\text{psoparameterid}(\text{number_of \_parameter})p(\text{parameter})$ : such as $\text{psoparameterid6p0}$ :

<table>
<thead>
<tr>
<th>OSC</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>X</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Cp i=</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>j=</td>
<td>0</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>47</td>
</tr>
</tbody>
</table>

6 : 7 : 8 : 9 : 10 : 11 : 31. Using this function, users can select parameters flexibly without modifying
the firmware code time by time.
- Various functions has been included to allow users to get and set the values of the CPG parameters. Therefore, the model setting and checking can be performed for all parameters.

### 3.2 Experiment Software

This section will discuss about the Experiment software. The software is developed in C++ and use the OpenFramework toolkit [16]. The development environment using is CodeBlock IDE integrated with OpenFramework version 007 on Ubuntu 11. In order to build the software, there are some Openframework addons (ofxAddon) have been used as follows:

1. ofxKinect - to obtain image and depth data which are used for the tracking system.
2. ofxOpenCV - to process image data.
3. ofxSerial (modified to meet the purpose) - to communicate with Roombots via serial port.
4. ofxUI - to build the Graphic User Interface (GUI).
5. ofxXmlSettings - to work with XML file.

There are three main modules in the software:

1. Tracking System module.
2. Communication module.
3. Experiment module

Fig.3.1 presents the GUI of the main experiment software. As shown in Fig. 3.1, the GUI contains 6 main sections:

![Figure 3.1: The Graphical User Interface of the main software](image-url)
1. **RED section: Kinect Settings.**
   This section contains:
   
   - **DATA FILE text input:** to input the name of the file in which we want to store the positions of the Roombots.
   - **THRESHOLD value bar:** to set the threshold used for object detection method using subtract background algorithm. This value can be pre-set in the Program Settings XML file.
   - **TAKE BACKGROUND button:** to take the current frame as the referenced background.
   - **RESET TRACKING button:** to reset the position tracking image (the one at the bottom right in the WHITE section).

2. **YELLOW section: Bluetooth Settings.**
   This section contains:
   
   - **MAC ADDRESS text input:** to set the MAC address of the Roombot. This value can be pre-set in the Program Settings XML file.
   - **CONNECT button:** to connect to the Roombots.
   - **INPUT text input:** to input a command which will be sent to Roombots.
   - **SEND button:** to send a command to Roombots.
   - **OUTPUT FILE text input:** to define the file name of the file we want to save the Bluetooth data received from Roombot. This value can be pre-set in the Program Settings XML file.

3. **GREEN section: EXPERIMENT.**
   This section contains:
   
   - **INIT button:** to initialize the PSO process. After clicking the button, the "psoinit" command will be sent to Roombots.
   - **RUN button:** to execute the "cpgrun" command and capture the position of the Roombots at the same time the command is sent.
   - **NEXT button:** to execute the "cpgreset" or "psonext" based on the number of run times have been executed. If it reaches the defined run times, it will execute the "psonext". The purpose of running multiple times is to obtain better fitness values, as a consequence the optimization converges faster.
   - **RESET button:** to execute the "cpgreset" command which used to reset the state of Roombots to initial one.
   - **PSO particle file text input:** to set the name of the file we want to store the current configuration of the PSO and load the PSO particles from which. This file name is used for load and save the optimization state in case of running out of battery. The values of each parameters or the positions of particles in PSO will be saved. Here, the information of particles, best particles, number of iterations, current particle are saved into the file. This file name can be pre-set in the Program Settings XML file.
   - **PAUSE button:** to pause the experiment and save the PSO particles data into the PSO particles file.
   - **RESUME button:** to resume the experiment and load the PSO particles from the saved file.
   - **EXPERIMENT XML FILE text input:** to input the name of the file which we want to set up the parameters of the experiments. This value can be pre-set in the Program Settings XML file. The parameters of a experiments include:
     - Number of Oscillators \(<NUMOSC>\)
     - Number of Run times \(<MAXRUN>\)
     - Frequency value \(\omega<OMEGA>\)
     - Convergence amplitude \(a<A>\)
– Coupling weight \( w < \text{COUPSTRENGTH} > \)
– In order to set a certain gait to Roombots we can use the structure in XML to set up each parameter such as amplitude, offset, and coupling phase. \(< R > < X > < \text{PHI} >\)
– The range of parameters for PSO algorithm. When we set the range for a certain parameter, the parameter ID (as described in Table. 3.1 will be computed directly. Afterwards, the software will send the parameter IDs selection command to the Roombots. For more details, please refer the XML file in /bin/data folder of the main software.

- LOAD EXP SETTINGS button: to load the experiment settings (usually executed first in an experiment).

This section includes of four image sequences.

- the top-left: the depth image.
- the top-right: the color image.
- the bottom-left: the object detection result.
- the bottom-right: the position tracking results. In this section, the white dot is the current position of the Roombots, while the grey dots are the old positions.

5. ORANGE section: Program status and messages.

6. BLUE section: Bluetooth SEND/RECEIVED data.

Using the software and the XML settings feature, many steps of the experiments can be accomplished automatically, especially the fitness value measurement as well as the settings, load and save PSO particles steps. The main problem during creating the software is the inconsistency between the comments and the code which leads to wrong command creation.

3.2.1 Tracking System

To archive the goal of online learning in Roombots, a tracking system has been developed to compute fitness values of optimization process.

The tracking system is an intermediate step in order to achieve the final goal of having the Roombots can learn by itself with the location data obtained from accelerometer sensors, gyroscope sensors or other position detection systems. On the other hand, the tracking system would help to confirm the precision and evaluate the performance of the location detection sensor system in the development process. As opposed to other tracking methods which use the color image to track the whole body of a robot or IR image to track a LED on the body of a robot, the tracking system developed in this project uses the depth data from Microsoft Kinect \(^\text{\textregistered} [17]\) to track the whole body of the Roombots. Due to the rotation of the Roombot’s body, we cannot use a single LED to track the position. We started with the idea of using depth image obtained by Kinect because it is not affected by the change of surrounding environment such as ambient light. As an experiment often lasts for a long time, the change in lighting conditions is unavoidable.

Microsoft Kinect offers in addition to the RGB image, a depth image. The intensity of each pixel in the depth image corresponds to the inverse distance between the camera and the objects of the scene (up to a scale). The depth image from Kinect is usually noisy, so a simple threshold does not produce good results in most cases. First, given the depth data, we would like to know the range of the depth. The depth data is represented by a grey-scale image in which a pixel value ranges from 0 to 255. The higher the value, the closer to the camera. The maximum depth value is 145-6 and the minimum is 130-1, which we can observe from the Fig. 3.4. Notice that the depth value of the robot is approximately equal to the maximum depth value of the floor, which makes it difficult to separate the object using only the max-min threshold (or Near Far Threshold) on the captured depth image from the Kinect.
After investigating the depth image, several algorithms are tested for the tracking system.

**Roombot detection algorithms**

- The first and also the simplest one is the max-min threshold. However, the ambiguity between object and floor surface makes it hard to recognize the object.

- The second idea is the max-min threshold with the threshold of the area object. This is done by knowing in advance the estimated initial position of the object such as in the middle of the vision. Then the object will be detected in the local region which is a rectangular that could cover the whole object. Afterwards, max-min threshold will be applied in this local region in order to detect the position of the object. This is done by changing the area parameter in the function. However, the limitation of this solution is that the local region is not always well estimated to cover the whole body of the object. In other words, there will be a misclassification when there are some part of object are out of region.

- The third solution, also the current solution, is using the subtraction of the current depth image by the background image without the presence of Roombots. This solution does not depend on the max-min threshold to detect the robot. The maximum depth and minimum depth value of the floor will not affect the performance of the algorithm. However, the noise and the depth value variation affect the detection performance. As observed from the image ??, the result of a subtraction of pure two images is really noisy and could not be used to apply contour finding algorithm[18] to detect the object. In order to reduce the effect of noise, first **Gaussian Blur** is applied to current image and the background image and take the subtraction. The subtraction result becomes really good and able to do detection effectively. Fig. 3.3 shows the difference between with and without Gaussian Blur filter. Having the subtraction result, the difference image will be thresholded to get rid of the floor and emphasize only on the object. The threshold value used in the subtraction result is 3. Hence, in the subtracted image, a pixel having intensity value larger than 3 will have value 1 in the thresholded image and the one has intensity value lower than 3 will have value 0 in the thresholded image. For the flexibility of the software, the user can change this result to obtain good result in various surrounding environments. After setting threshold, we obtained clean image where only the Roombots is in white color while background is in black color, there are some tiny white parts because of peak noise but it does not affect the final result of the algorithm.

- The fourth solution is using K-mean to separate the object and the floor using 3-dimensions points [X Y Z]. The coordinates of each point is determined in the new coordinate system where the (0,0,0) is located in the middle of the image and has 0 depth value. Here Z is the depth value after multiplying a certain scale factor in order to emphasize on the depth when clustering. The method is inspired by the Computer Vision course. K-Means is a least-squares partitioning method that divides a collection of N-dimensional points into K groups. In our problem, N is 3 and K is 2. The algorithm can be summarized as follows:
1. Compute the Euclidean distance of each point from each cluster mean. Assign each point to the cluster it is nearest to.
2. Compute the mean of each cluster.
3. Iterate over the above two steps until the cluster centroids do not move any more or a fixed number of iterations is reached.

The algorithm has been implemented successfully inside the software. With the big object, the current implementation can segment the background and object very well. However, with objects having small depth and too far from Kinect like Roombots in our case, the algorithm does not work well. There should be an additional step to magnify the difference from object and floor in order to get good segmentation result. Fig. 3.4 shows the situation. The bottom-right subimage shows that the chair with high value of depth is segmented correctly, while the Roombot is not. On the other hand the bottom-left subimage shows that the background subtraction method working correctly, in which both the char and the Roombot are segmented. Because the subtract background method is working well in our case, the deal with K-mean is for future work. The advantage of K-mean method is that there is no background selection at the beginning of an experiment.

During the experiment, we noticed that the tracking results were still good and robust after one or two hours in most of the cases.
3.2.2 Centroid Detection

After determining the Roombot as a blob, we detected the centroid of the blob by the following equation (3.1)

\[
\bar{x} = \frac{\int \int f(x,y) \times x}{\int f(x,y)}; \bar{y} = \frac{\int \int f(x,y) \times y}{\int f(x,y)}
\]  

(3.1)

where \( f(x,y) \) is the pixel value at (x,y), which is either 0 or 255. \((\bar{x}, \bar{y})\) is the position of the centroid of the blob. The integration was done in the region of the rectangle cover the whole blob.

3.2.3 Fitness Value

In the tracking system, the fitness value is defined as the distance the Roombots move in a period of time (more precisely, the distance between two centroids of Roombots after 30s) in our experiments. The fitness value is the difference of two Roombots position after 30s in pixel unit. As the camera is placed such that the distortion is minimized, we found that there is about a Homography \( H \) (3x3 matrix) which converts a pixel in the image to a position in the floor ([19], refer to Estimation - 2D Projective Transformations section). Therefore, the distance in the real world is proportional to that in the image (in pixel unit). Approximately, 260 pixels in the Kinect image corresponds to 1 meters in the real world. Therefore, we have an approximated formula to convert the fitness value from the tracking system to the distance in real world, as in Eq. (3.2)

\[
d = \frac{f \times 120}{320}
\]  

(3.2)

where \( d \) is the distance in the real world (in cm) and \( f \) is the fitness value (in pixel).

3.3 Experiments

3.3.1 Experiment Environment

The area for the robot to move is about \( 2 \times 2 m^2 \) which is smaller than the vision area of the Kinect above. In order to reduce the risk of strong collision to the floor, a \( 2 \times 2 m^2 \) foam floor was placed above the wooden floor. In order to increase the frictions between Roombots and ground, we placed a cork mat on the foam floor. The fitness value is measured by the tracking system including the transient phase. Due to the fact that the cork is not flat, there are some bubbles on the surface and the center of mass of Roombot can be changed after battery replacement, there are variances in the fitness value. In order to obtain good fitness value over the non-homogeneity of environment and Roombot itself, each time the CPG will be executed two or three times and the fitness value will be determined as the average travelled distance of those two or three times. The other advantage of running multiple time is that in case of a good gait, the Roombot moves outside the vision range, we can locate Roombot in different initial positions. In the experiment, we also have three types of materials to test the robustness of the gait under the variations of the surface's friction. The three types of materials are Cork, textbf{Paper}, textbf{Plastic Rubber} as shown in Fig. 3.5 The experiments were carried out in the computer lab in BioRobotics Lab with a continuous change in the light conditions.

3.3.2 Experiment Setup

The experimental setup is shown in the Fig. 3.6
The robots are tested in a square area of 2mx2m. The distance travelled in one experiment time (30s) is measured by tracking the centroid of the blob of Roombots by a Kinect fixed to the ceiling. The distance travelled is fed to PSO algorithm as the fitness value. The online learning process is carried out as follows. For each particle, the CPG will set the parameter values from the particle’s position. Then, Roombots will run the CPG-based controller in 30s. At the end of each running time, the position of Roombots is detected by the tracking system. The running will be repeated several times to obtain the as precise as possible fitness value for further optimization. Afterwards, it goes for the next particles. When it reaches the last particles, the position and velocity of all particles will be updated and a new iteration starts with the first particles.

### 3.3.3 Experiment Procedure

The procedure of an experiment is presented as a sequence graph in Fig. 3.7.
First, we need to Load the Experiment settings by the XML file. With the Experiment setting, we can load a certain gait of CPG by loading the parameter values, or we can load a certain configuration CPG and PSO by loading the parameters values and ranges. An example of the XML settings is given below:

```xml
<CPG>
  <NUMOSC>6</NUMOSC>
  <MAXRUN>2</MAXRUN>
  <PARA>
    <OMEGA>2.0</OMEGA>
    <COUPSTRENGTH>0.5</COUPSTRENGTH>
    <R>
      <I>2</I>
      <MIN>0.0</MIN>
      <MAX>3.14</MAX>
    </R>
    <PHI>
      <I>2</I>
      <MIN>-3.14</MIN>
      <MAX>3.14</MAX>
    </PHI>
  </PARA>
</CPG>

<PSO>
  <RANGE>
    <R>
      <I>1</I>
      <MIN>0.0</MIN>
      <MAX>5.14</MAX>
    </R>
    <X>
      <I>2</I>
      <MIN>-2.0</MIN>
      <MAX>2.0</MAX>
    </X>
  </RANGE>
</PSO>
```
As we can see from the above example, the set up of an experiment is intuitive and convenient. In the XML file, we first define the number of oscillators, the number of Roombots running time per each PSO particle, the value for each CPG parameters such as amplitude \( R_i \) or the offset \( X_i \), and finally the range of each parameter which will be used in PSO process. By setting the range for the parameter which will be used in the PSO optimization, the parameter IDs will be sent directly to the Roombots. Therefore, we can easily change the optimization parameters without modifying the code inside the Roombots. If we just want to load a certain gait with specified parameters value, we can skip defining the range for PSO algorithm (i.e. skipping \(<\text{PSO}\_i\text{Tag}>\text{ tag}\).

Second, we divide the procedure into two types.

1. Start a new experiment.
   - We need to click on INIT to do **PSO initialization** until there is no collision detected.

2. Resume a paused experiment.
   - We need to **Resume PSO particles state** by loading PSO particles from a file (by clicking RESUME in the software) which contains the PSO particles. Normally, as in the configuration of the program, the particles are contained in \(\text{PSOParticles.dat}\) file.

Third, with the support of a wireless mouse, we can press the **left button** of the mouse to run **CPG RUN** (i.e. sending \(\text{cpgrun}\) command to Roombots) and press the **right button** of the mouse to go to next run or the **NEXT PARTICLE** (i.e. sending \(\text{psonext}\) command) when the number of running times reaches a **MAX_RUN_TIMES** values. The travelling distance in each running time is accumulated to the total distance, which in turn will be divided by the number of running times to obtain the average distance as the fitness value. This step is repeated until the battery drains or users want to pause the experiment. When the PSO algorithm moves to next particles, the current particle’s state is saved to a record file. Each time an iteration finishes, the states of all particles and best particles are saved into a record-all-particles file. If a user do not want to continue the experiment, in order to pause the experiment, all we need is to click **PAUSE** and the software will **Save Particles** state for future experiment resume.

Instead of inputting commands manually, they are generated automatically when users pressed a button and then sent to the Roombots. The “\(\text{cpgrun}\)” command is included in the routine when the RUN button is pressed. The “\(\text{psonext}\)” is executed when the NEXT button is pressed with the fitness value calculate by the tracking system.

### 3.3.4 Difference between simulation and real experiment configuration

After investigating the results from simulations, we realized that there is a difference in the real configuration and the simulated one. If one module has 3 DOFs, which are called s1,m1,and s2 (i.e. side 1, middle 1, and side 2, respectively). In real the s1 of module 1 is connect with s1 of module 2, while in simulation s2 of module 1 is connected to s1 of module 2. The Fig. 3.8 in which different type of motor was encode by different color represents the configuration difference.

![Figure 3.8: Difference between Simulation and Real configurations. Each oscillator corresponds to a motor (1 DOF) in a Roombot. The gray one represents for s1, green one does for m1, blue one does for s2.](image)
Therefore, there were mostly collisions when we tried to use the simulation results obtained to the
reality. However, the simulation results provided us useful information about the search space of PSO
algorithm.

### 3.3.5 Choice of fixed parameter

Due to the fact that the CPG model contains a large number of parameters, it is necessary to reduce
the number of parameter in order to make the PSO process feasible in term of times in real world. In additional
to fixing global parameters such as $\omega$, $a$, we need to impose some rules to reduce the number of parameters.

1. Fixed frequency, $\omega$. In an ideal condition, $\omega$ should be proportional to the speed of a gait. Here, a
small value of frequency is selected to reduce the damage and fluctuation caused by applying a strong
force to the ground.

2. Fixed convergence rate $a$ and coupling strength $w$ which has effect only in the transient phase.

3. Fixed some parameters’ values which has small impact to the performance of gaits.
   - We observed that the DOF 1 and DOF 6 has a little impact on a gait. Hence, we all parameters of
     them are set to 0.
   - We considered the two ends spheres as axial rotation invariant. Hence, $X_2 = 0$ and $X_5 = 0.$

In the experiments, we have the list of fixed parameters as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Fixed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega$</td>
<td>1</td>
</tr>
<tr>
<td>$a$</td>
<td>2</td>
</tr>
<tr>
<td>$w$</td>
<td>0.5</td>
</tr>
<tr>
<td>$X_2, X_5$</td>
<td>0</td>
</tr>
<tr>
<td>$X_1, X_6$</td>
<td>0</td>
</tr>
<tr>
<td>$R_1, R_6$</td>
<td>0</td>
</tr>
<tr>
<td>$\phi_{16}, \phi_{61}, \phi_{11}, \phi_{16}$</td>
<td>0</td>
</tr>
</tbody>
</table>

Hence, we have left 9 parameters open for PSO optimization, namely:

- $R_2, R_3, R_4, R_5$, amplitudes of the oscillators 2,3,4,5 respectively.
- $X_3, X_4$, offsets of the oscillator 3,4 respectively.
- $\phi_{23}, \phi_{34}, \phi_{45}$, coupling phases between respective oscillators.

During experiments, we tried to do the optimization with 6 parameters by fixing some parameters or mak-
ing equal to each other.
4

Results & Discussion

In this part, the results of main experiments will be presented and discussed.

4.1 Results

4.1.1 Experiment 1

Parameters selection

In this experiment, we investigated how Roombots learns a locomotion of tadpole (that is there is a head and a moving tail). Here, we selected the block which contains Motor 1 and Motor 2 as the head and the other will be a tail. In this experiment, the head is moving independently with the tail. The parameters selection and ranges setting were uploaded using the XML file and experiment software without changing the firmware of the Roombots. The parameters are selected as in the following table: The range of $[-2, 2]$ is chosen in order to minimize the collision risk.

![Table 4.1: Particle Swarm Optimization Search Space for Experiment 1](image)

Result

For the first experiment, we used 10 PSO particles and run 9 iterations. During the experiment, when there is a collision detected, the fitness value will be equal to 0 and continue with the next particle. For each particle, we let Roombots run two times to obtain an average fitness value. The fitness values along the iterations are presented in the Fig. 4.1 As we can observe from the Fig. 4.1, a good gait can be obtained in the first iteration. The fitness values of all particles fluctuates strongly but we can still observe a going upward trend in the figure. After 9 iteration, above half number of particles have a fitness value in range of 120 to 180. Moreover, the gaits that they experienced are roughly similar: the tail periodically moved from bottom left, go up, then bottom right and reversely to push the Roombots moving forward. Each time Roombots meta-module used its tail to move, it hit the floor with a strong force in order to move forward. Fig. 4.2 shows the variables of selected CPG parameters of the PSO particle 4 at which we obtained the
Figure 4.1: Fitness values of 10 PSO particles along 9 iterations. A high fitness value means a good result. The fitness values are coded by the color bar with red as the highest value and blue as the lowest value. At the first iteration, every particle has a fitness value of 0. There is a upward trend in fitness value along the iterations. After 9 iterations, many particles obtained good fitness values.

best gait and its fitness value. We can see that after changing dramatically in the beginning iterations, the values of them become more stable finally. Moreover, the fitness values kept an upward trend. Finally, the

Figure 4.2: The variables of selected CPG parameters of the PSO particle 4 and its fitness value. A fitness value of 260 corresponds to 1 m in real world.

best gait of the first experiments has a fitness value of 137 (or 51.4 cm) (i.e a speed of 1.7 cm/s). The gait is found by the particle 4. The found values of the parameters are summarized in Table 4.2:

### 4.1.2 Experiment 2

#### Parameters selection

The experiment 2 is similar to the experiment 1 except that now \( R_2 = R_3 = R_4 = R_5 \). We would like to investigate the effect of a symmetry in amplitude on a gait. The parameters are selected as in the following table:
Table 4.2: Best gait discovered in the Experiment 1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_2$</td>
<td>1.343</td>
</tr>
<tr>
<td>$R_3 = R_4$</td>
<td>0</td>
</tr>
<tr>
<td>$R_5$</td>
<td>$\pi$</td>
</tr>
<tr>
<td>$X_2$</td>
<td>0.070</td>
</tr>
<tr>
<td>$X_3$</td>
<td>-1.00</td>
</tr>
<tr>
<td>$\phi_{23}$</td>
<td>1.279</td>
</tr>
<tr>
<td>$\phi_{34}$</td>
<td>0.182</td>
</tr>
<tr>
<td>$\phi_{45}$</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.3: Particle Swarm Optimization Search Space for Experiment 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_2 = R_3 = R_4 = R_5$</td>
<td>$[0, \pi]$</td>
</tr>
<tr>
<td>$X_2$</td>
<td>$[-2, 2]$</td>
</tr>
<tr>
<td>$X_3$</td>
<td>$[-2, 2]$</td>
</tr>
<tr>
<td>$\phi_{23}$</td>
<td>$[-\pi, \pi]$</td>
</tr>
<tr>
<td>$\phi_{34}$</td>
<td>$[-\pi, \pi]$</td>
</tr>
</tbody>
</table>

Result

In this experiment, we used 10 PSO particles and run 10 iterations. The fitness values were computed similarly like the first experiment. The fitness values along the iterations are presented in the Fig. 4.3. Contrary to experiment 1, a good gait can only be obtained in the 6th iteration. The fitness values of all particles fluctuates strongly but we can still observe a going upward trend along iterations. After 10 iteration, 8 particles have a fitness value between 200 and 240, which can be considered to be converged as the gait are almost similar. The tail periodically moved to push the Roombots body forward and the head oscillate to change the direction. Fig. 4.4 shows the variables of selected CPG parameters of the PSO particle 10 and its fitness value along iterations. We can see that after changing dramatically in the first iterations, and then values become stable finally. Moreover, the fitness value kept going up even there are
some decreases during iterations. Finally, the best gait of the second experiment has a fitness value of 267.

![Figure 4.4: The variables of selected CPG parameters of the PSO particle 10 and its fitness value. A fitness value of 260 corresponds to 1 m in real world (or 100.1 cm) (i.e a speed of 3.34 cm/s). The gait is found by the particle 10. However, this gait is not robust it depends on the direction leaded by the head. A slight change in the direction causes a significant decrease of the fitness value. The discovered values of the parameters are summarized in Table 4.4:](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_2 = R_3 = R_4 = R_5$</td>
<td>2.121</td>
</tr>
<tr>
<td>$R_1 = R_6$</td>
<td>0</td>
</tr>
<tr>
<td>$X_2$</td>
<td>-1.54</td>
</tr>
<tr>
<td>$X_3$</td>
<td>-0.18</td>
</tr>
<tr>
<td>$\phi_{23}$</td>
<td>-2.13</td>
</tr>
<tr>
<td>$\phi_{34}$</td>
<td>1.543</td>
</tr>
<tr>
<td>$\phi_{45}$</td>
<td>0</td>
</tr>
</tbody>
</table>

### 4.1.3 Experiment 3

#### Parameters selection

This is the most important experiment. Unlike the previous ones, in this experiment, the locomotion of the Roombots is more symmetric. In this experiment, we set the amplitudes of four oscillators are equal, $R_2 = R_3 = R_4 = R_3$. As we neglected the effect of oscillator 1 and 6 in Fig. 2.4 (i.e. fixed these motors), the two ends of Roombots meta-module become two spheres, which can be considered as axial rotation invariant. Hence, we neglect the offset of oscillator 2 and 5 in Fig. 2.4, that means $X_2 = 0$ and $X_5 = 0$. Finally, the parameters are selected as in Table 4.5. The range of $[-2,2]$ is chosen in order to reduce the collision risk.

#### Result

In this experiment, we used 10 PSO particles and run 8 iterations. During the experiment, when there is a collision detected, a fitness value of current particle will be equal to 0 and PSO algorithm move to the next particle. For each particle, Roombots meta-module will run CPG-base controllers for 30s in three times. The final fitness value is equal to the average values of three running times.

The fitness values along the iterations are presented in the Fig. 4.5. As we obtained good gaits in this
Table 4.5: Particle Swarm Optimization Search Space for Experiment 3

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_2 = R_3 = R_4 = R_5 = R$</td>
<td>$[0, \pi]$</td>
</tr>
<tr>
<td>$X_3$</td>
<td>$[-2, 2]$</td>
</tr>
<tr>
<td>$X_4$</td>
<td>$[-2, 2]$</td>
</tr>
<tr>
<td>$\phi_{23}$</td>
<td>$[-\pi, \pi]$</td>
</tr>
<tr>
<td>$\phi_{34}$</td>
<td>$[-\pi, \pi]$</td>
</tr>
<tr>
<td>$\phi_{45}$</td>
<td>$[-\pi, \pi]$</td>
</tr>
</tbody>
</table>

Figure 4.5: Fitness value of 10 PSO particles along 8 iterations. The fitness values are generally increasing along the iterations. A high fitness value means a good result. The fitness values are coded by the color bar with red as the highest value and blue as the lowest value. At the first iteration, every particle has a fitness value of 0.

The experiment and the project time was running out, we could not continue more iterations. With the support of the software, continuing conducting more iterations in an experiment will be the future work. Different from Experiment 1, as in Fig. 4.5, a good gait can only be obtained in the 6th iteration. After 8 iterations, there are some particles produced a similar gaits which have fitness value larger than 320. Fig. 4.5 shows the variables of selected CPG parameters of the PSO particle 4 which produce the best gait and its fitness value along iterations. In this experiment, we took two results: one is a good gait from iteration 3 and one is the best gait in iteration 8.

Finally, the best gait of the third experiment has a fitness value of 395 (or 148.1 cm) (i.e a speed of 4.9 cm/s). The gait is found by the particle 4. Another good gait is taken from the third experiment in iterations 3. This gait has a fitness value of 324 (or 121.5 cm) (i.e a speed of 4.05 cm/s) following table (Table 4.6):

Table 4.6: Best gait discovered in the Experiment 3

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_2 = R_3 = R_4 = R_5$</td>
<td>2.065</td>
</tr>
<tr>
<td>$R_1 = R_6$</td>
<td>0</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.407</td>
</tr>
<tr>
<td>$X_4$</td>
<td>-0.03</td>
</tr>
<tr>
<td>$\phi_{23}$</td>
<td>2.418</td>
</tr>
<tr>
<td>$\phi_{34}$</td>
<td>3.103</td>
</tr>
<tr>
<td>$\phi_{45}$</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure 4.6: The variables of selected CPG parameters of the PSO particle 4 and its fitness value along 8 iterations. A fitness value of 260 corresponds to 1 m in real world.

Table 4.7: A good gait discovered in the Experiment 3

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_2 = R_3 = R_4 = R_5$</td>
<td>1.448</td>
</tr>
<tr>
<td>$R_1 = R_6$</td>
<td>0.0</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.708</td>
</tr>
<tr>
<td>$X_4$</td>
<td>0.044</td>
</tr>
<tr>
<td>$\phi_{23}$</td>
<td>1.375</td>
</tr>
<tr>
<td>$\phi_{34}$</td>
<td>3.102</td>
</tr>
<tr>
<td>$\phi_{45}$</td>
<td>0.057</td>
</tr>
</tbody>
</table>

4.2 Gaits description

We have obtained the best gain through experiment 3. Fig. 4.7 describe the trajectory of the Roombots during 30 seconds with the gait.

Figure 4.7: Best gait with a fitness value of 395 in average or 150cm (i.e a speed of 5cm/s).

The Roombots used two body parts as two arms to lift it’s center of mass and fall down to move forward. Moreover, it also used these two arms to crawling on the floor. The movement is approximately similar
to the combination of snake in a perpendicular direction with its body and lift-fall movement. As we can observe from the trajectory of each body part of the Roombots, it is obvious that it perform a twist or rotation the whole body during the movement. The locomotion pattern is similar to the movement of the snake on sand desert.

For the good gait, Fig. we also investigate the trajectory of it in different time intervals. The left image shows the periodicity of the gait. Roombots first move in the counter-clockwise direction with two arms move harmonically. Then Roombots makes a C-shape. After a small scroll in the body, Roombots move in the clockwise direction. This sequence of actions repeats periodically and makes Roombots move forward. In other words, the body periodically tightens and expands, while using the rotation of the two outer cubes to move forward. This locomotion pattern is also similar to the natural locomotion pattern in animals.

4.3 Gaits Evaluation & Discussion

Given a gait discovered by online learning, it is important to evaluate its robustness against changes in its initial states and surrounding environments such as the friction of the surface, the obstacles and so one. This part will evaluate the robustness of gaits taken from the experiment 3. First, when we did the experiment and post-evaluation on the gait obtained in experiment 1 and 2, we found that these gaits are sensitive to initial states and frictions. A slight change in the surface or initial position caused a big change in the fitness. Especially for the result from experiment 2, a small changes in the initial position can produce a big difference in the distance the Roombots travel. Sometimes, the final distance was reduced by half. Therefore, we did not continue the further robustness test about changing surface material and initial states on these gaits. However, a good gait obtained in just 3 iterations in experiment 3 is promingly robust.

In order to evaluate the robustness of a gait, we varied the floor surface (i.e. change the friction) and the initial state of Roombots (i.e. the side of Roombots). We tested with 3 types of material surface which have a high friction (Cork mat), medium friction (Scratched paper) and low friction (Plastic Rubber foam floor) (Fig. 3.5) and 4 initial states which corresponds to 4 orientations of the Roombots (Fig. 4.9). When we did the experiment we used only one orientation as the initial state, however, it could be interesting that the gait learn from a different initial state can adapt to others. To evaluate under a particular condition, we let the Roombots run 3 times, each of them lasts for 30 seconds.
4.3.1 Different frictions evaluation

We test with the same initial state (same orientation) and three materials (with videos available). Fig. 4.10 shows how robustness of the gaits to the materials. Here we tested with 2 gaits: one is a good gait and the other is the best gait obtained from experiment 3. In general, both gaits are robust against changes of friction as they still produced high fitness values with high distances travelled. However, the behaviour of two gaits are different. While with gait 1 the distance that Roombots travelled increases when changing material from frictional Cork to slippery plastic, that decreases with gait 2. Therefore, we conclude that different materials can alter the performance of the gait differently. The difference depends on what type of the movement is considered. In our case, the gait 2 mainly use the friction between Roombots body with the floor to move forward and lift their body and falling forward, while the gait 1 uses rolling of the body.

![Fitness Value (pixel) comparison](image)

Figure 4.10: Gait Robustness over the friction of a surface. A fitness value of 260 corresponds to 1 meter in real world

4.3.2 Different initial states evaluation

In this evaluation, we place the Roombot with different orientations and run the same gait for three times on two different type of materials(low and high friction materials). Fig. 4.11 show the result of the evaluation.

It is obvious that the performance of the gait with side 1 and 2 are greatly better than that of side 3 and 4. The reason behind this accounts for the non-symmetry in Roombots structure. Moreover, we can observe that the pair sides 1 and 2, 3 and 4 produce close results.
Figure 4.11: The robustness evaluation of gait 1 with the initial state. Blue and red color corresponds to cork and plastic foam material. Side 1 and 2 have more white part on the top, Side 3 and 4 have more black color part.

4.3.3 Obstacle evaluation

We placed an obstacle in the middle of the path to try to block the Roombots and observe how robust the gait is over the obstacle. The obstacle is placed in Fig. 4.12. Fig. 4.13 shows how Roombots with the best gait could overcome an obstacle. As in the best gait, Roombots lifts its body with higher amplitude than that in good gait, the best gait can pass a higher obstacle. However, with the same obstacle, different initial positions can affect the ability of overcoming an obstacle of a certain gait can. Fig. 4.14 shows an example where Roombots with a good gait failed to pass the obstacle. We can observe from Fig. 4.12 and Fig. 4.14 that the robustness against obstacle can affected by the initial position of the Roombots. We found that the ability of overcoming an obstacle of the best gait is better than that of the good gait.

However, when we increased the difficulty of the obstacle, Roombots with both gaits failed to overcome the obstacle. Fig. 4.15 shows an example.
Figure 4.13: Gait Robustness evaluation over obstacle: The sequence of images shows how Rooombots with the best gait could overcome an obstacle.

Figure 4.14: Gait Robustness evaluation over obstacle. The sequence of images shows how Rooombots with the good gait failed to overcome an obstacle.
Figure 4.15: Good gait vs. Big obstacle. Roombots cannot overcome the obstacle for two reasons: the obstacle is big and the plastic rubber surface is slippy.

4.3.4 Slope evaluation

We placed a slop of 10 degree. With the slippery material, it was difficult for the Roombot to reach the top of the slop. Moreover, it always suffered from the slip down. However, for the gait 1, the Roombots could reach closely to the top of the ramp. The result is slightly better when using paper with many scratches to increase the friction. Fig.4.17 and Fig.4.16 show that increasing the friction of the surface can improve

Figure 4.16: Gait Robustness evaluation over slope. The sequence of images shows how Roombots with the best gait and slippy plastic rubber surface could not reach the top of the slope.
Figure 4.17: Gait Robustness evaluation over slope. The sequence of images shows how Rooombots with the good gait and scratched paper surface could reach the top of the slope.

dramatically the performance of a gait over the slope.
Conclusion and Future works

The software with the tracking system is the user-friendly tool which makes the experiment on online optimization for locomotion of modular robots more convenient and time, energy-saving. We can conduct more experiments handily with more complex structures of Roombots in the future. Moreover, more features added to the new firmware of the Roombots would make the experiment comfortable.

The found gaits are interesting because most of them use two modules as two arms to bend and expand the body or lift and then fall or scroll the body to travel furthest or obtain a good fitness value. One result is a C-shaped gait which involves the periodic tightening and expanding the body with the combination of oscillation of two ending sphere to move forward. The second one which is also the best one found in this project is the combination of the C-shaped gait and lifting-falling body gait. In lifting-falling gait, the body make a L-shape to lift the central of mass of the body and use the oscillation of two outer spheres to fall down and move forward. Moreover, the gaits obtained from the experiment 3 are closely similar to the locomotion pattern of animals in nature.

The obtained gaits also were evaluated the robustness or performance against the changes in initial state and environments. The results showed that initial state and surface friction are two main factors that affects significantly to the performance of a particular gait.

In this project, the fitness value for PSO algorithm is only the distance between the final and initial value without concerning the direction of the movement. Using a fitness value including both speed and direction should be a challenging task for the step of the project.

In the future, based on the color image from Microsoft Kinect®, it is supposed that the current orientation of the Roombots can be detected from vision data. Knowing this information, a software can generate commands to send Roombots back to its initial position and orientation. Achieving this feature, the whole online optimization process will be run automatically. The Roombots can learn by itself without the support of human being.
Bibliography


