EPFL - BioRobotics Laboratory


# Experiment Software Manual

Semester Project: Online Optimization for the locomotion of RoomBots

Supervisors: Rico Möckel, Stéphane Bonardi, Soha Pouya, Massimo Vespignani
Professor: Auke Jan Ijspeert

June 2012

The Anh Nguyen(theanh.nguyen@epfl.ch)

## 1 Introduction

This is manual of the software which is used in experiments of the semester project: **"Online Optimization for the locomotion of RoomBots"**.The software is developed in C++ and use the OpenFramework toolkit [1]. 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.

## 2 Graphical User Interface

This section introduces Graphical User Interface of the software. Fig.1 presents the GUI of the main experiment software. As shown in Fig. 1, the GUI contains 6 main sections:

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:


Figure 1: The Graphical User Interface of the main software

- 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 $w < COUPSTRENGTH >$
  - 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. $< R >< X >< PHI >$
  - The range of parameters for PSO algorithm. When we set the range for a certain parameter, the parameter ID (as described in Table. 2 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 a experiment).

4. **WHITE section: Visual Monitor.**
   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 Experiment Procedure

This section introduces the procedure of a general experiment. In order to conduct experiments in a right order, please read this part carefully. The procedure of an experiment is presented as a sequence graph in Fig. 2

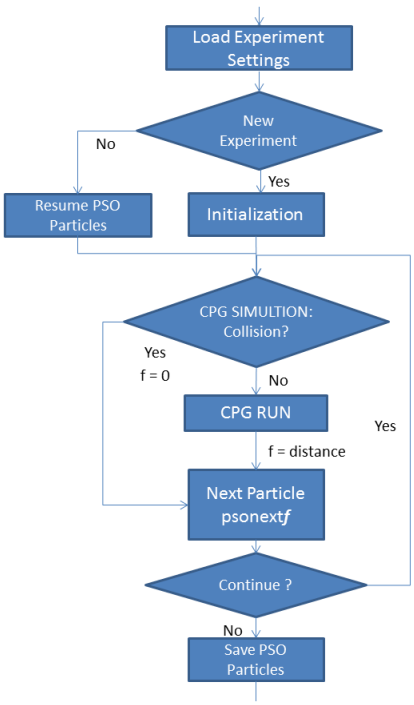


Figure 2: Experiment sequence

**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:

```
<CPG>
    <NUMOSC>6</NUMOSC>
    <MAXRUN>2</MAXRUN>
    <PARA>
        <OMEGA>1.0</OMEGA>
        <A>2.0</A>
        <COUPSTRENGTH>0.5</COUPSTRENGTH>

        <R>
            <I>2</I>
            <VAL>0.367</VAL>
        </R>

        <X>
            <I>2</I>
            <VAL>0</VAL>
        </X>

        <PHI>
            <I>1</I>
            <J>2</J>
            <VAL>0.151</VAL>
        </PHI>
    </PARA>
</CPG>

<PSO>
    <RANGE>
        <R>
            <I>2</I>
            <MIN>0.0</MIN>
            <MAX>3.14</MAX>
        </R>

        <X>
            <I>2</I>
            <MIN>-2.0</MIN>
            <MAX>2.0</MAX>
        </X>

        <PHI>
            <I>1</I>
            <J>2</J>
            <MIN>-3.14</MIN>
            <MAX>3.14</MAX>
        </PHI>
    </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 *¡PSO¿* tag). *Please note that Oscillator's ID in XML file starts from 0 to 5 instead of from 1 to 6.*

**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 *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 *cpgrun* command to Roombots) and press the *right button* of the mouse to go to next run or the **NEXT PARTICLE** (i.e. sending *psonext***f** 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 *"cpgrun"* command is included in the routine when the RUN button is pressed. The *"psonext"* is executed when the NEXT button is pressed with the fitness value calculate by the tracking system.

The PSO particles file is written in format.

1. N - The number of particles
2. Particle ID: 6 parameters values (or the particle's position) : fitnes value
3. Best Particle ID: 6 parameters values (or the best position of a particle) : fitnes value
4. The current iterations
5. The current particle
6. The fitness value of the last considered particle.

# 4 Experiment steps

1. Prepare Experiment Settings XML file (**ExpSettings.XML**). Define CPG parameters values and their range in XML file, follow the format mentioned in 3.
2. Execute the Experiment software.
3. Setup **Tracking System**
   (a) Select the reference Background Image by clicking button **TAKE BACKGROUND**
   (b) Click **RECORD** to record the Roombots' position to the file and the tracking image (which is shown in bottom-right image)
4. Obtain the PSO particles information.
   (a) Load the Experiment Settings by clicking **LOAD EXP SETTINGS**. The name of the setting file can be changed.
   (b) If a new experiment is carried out. Click **INIT** until there is no collision. In order to know whether there is a collision or not, please watch the Bluetooth data receive from Roombots.

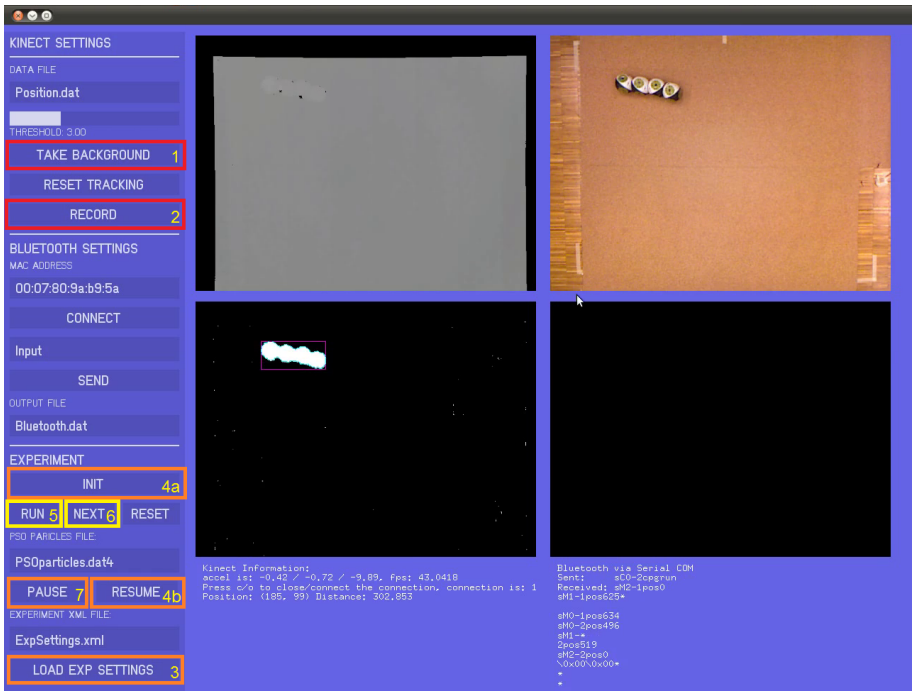


Figure 3: A sequence of main steps in the experiment software to start conducting an experiment

5. If an old experiment is resumed. Click **RESUME** to load the PSO particles information. The name of the PSO particles file can be changed.

6. After loading and initialization step, we can used both wireless mouse and wired mouse to control the experiments.

   (a) For wired mouse, click **RUN** to let Roombots run in 30s, click **NEXT** to repeat running until the number of running times reaches the number defined in the Experiment Settings XML file. At the last running times, the software will send command to Roombots to move to the next PSO particles.

   (b) For wireless mouse, in order to reduce the man-power, clicking **LEFT** mouse will perform same function with clicking **RUN** and clicking **RIGHT** mouse will perform same function with clicking **NEXT**. Please note that the position of the mouse should be not in the control section- the left hand bar to avoid the interference with other control commands.

7. Each PSO particle is saved every time the optimization uses it. The whole PSO particles are saved at the beginning of each iteration. Therefore, the probability of losing PSO particles information is minimized. Moreover, in order to resume conveniently in the next experiment, it is recommended that please save the PSO particles information by clicking **PAUSE** after 4 iterations.

The Experiment section generates commands and send them to Roombots automatically. Besides, users can input a command manually and send to Roombots in the Bluetooth Communication section. In case of losing connection, press **CONNECT** to reconnect to Roombots with specified MAC address.

# 5 Commands to control Roombots

In this section, the list of commands which are used to control and interact with the robot is provided in Table 1. Commands are sorted by their functions.

Table 1: Commands table

| Commands | Functions | Examples |
|---|---|---|
| | **Commands setting CPG parameters** | |
| "cpgcoupstrength" | Set coupling strength of CPG pair. | "$sC0-2cpgcoupstrength0-1p0.2$" sets coupling strength between CPG 0 and CPG 1 to 0.2 |
| "cpgcouppphase" | Set coupling phase of CPG pair. | "$sC0-2cpgcoupphase0-1p0.2$" sets coupling phase between CPG 0 and CPG 1 to 0.2 |
| "cpgmovement" | Set movement type of CPG. Movement is specified by parameter: 0 for oscillation, 1 for rotation, 2 for locked regime. | "$sC0-2cpgmovement5p0$" makes CPG number 5 oscillate. |
| "cpgsettimer" | Set timer for experiment (in milliseconds) | "$cpgsettimer30000$" imposes 30 seconds experiments. |
| "cpgoffset" | Set offset of CPG. | "$sC0-2cpgoffset5p0.2$" sets offset of CPG number 5 to 0.2 |
| "cpgreset" | Reset cpg, i.e. makes the robot come back to the initial position. | |
| "cpgomega" | Set frequency (phase velocity) of all CPGs | "$sC0-2cpgomega0.2$" sets frequency of all CPGs to 0.2 |
| "cpgnext" | Enable(1) / disable(0) step-by-step. | (To go to next step: "cpgenable1") |
| "cpgampl" | Set desired amplitude of CPG. | "$sC0-2cpgampl5p0.2$" sets amplitude of CPG number 5 to 0.2. |
| "cpga" | Set amplitude rise time of all CPGs. | "$sC0-2cpga0.2$" sets amplitude rise time of all CPGs to 0.2 |
| "cpgvalues" | Print CPG values | "$?C0-2cpgvalues0$" print the parameters values of CPG 0 |
| "cpgsetpart" | Load particle into CPG. | "$sC0-2cpgsetpart03p3.141:2.542:-5.23:9.321:-0.13:-4.32$" updates the CPG parameters with (3.141,2.542,-5.23,9.321,-0.13,-4.32). Note that the index of the particle should have 2 CHARACTERS (else it would not work). |
| | **Commands controlling Roombots with CPG** | |
| "cpgsimulation" | Makes a simulation to check if there is a collision with the current parameters. | |
| "cpgenable" | Enable/disable CPG control. "cpgenable1" to enable, "cpgenable0" to disable. | |
| "cpgrun" | Run cpg (without simulation) | |
| "cpgecho" | Enable/disable sending motor values to host | |

The table continues in the next page.

| | | |
|---|---|---|
| | **Commands setting search range for PSO** | |
| "cpgamplrange" | Set desired amplitude range of CPG for optimization. | "$sC0-2cpgamplrange5p0.2:0.6$" sets amplitude of CPG number 5 to be optimized between 0.2 and 0.6. The parameter can be fixed with a range of size 0 (ex: "0.3:0.3" fixes the parameter to 0.3). |
| "cpgoffsetrange" | Set offset range of CPG for optimization. | "$sC0-2cpgoffsetrange5p0.2$" sets offset of CPG number 5 to be optimized between 0.2 and 0.6. The parameter can be fixed with a range of size 0 (ex: "0.3:0.3" fixes the parameter to 0.3). |
| "cpgcoupphaserange" | Set coupling phase range of CPG pair for optimization. | "$sC0-2cpgcoupphaserange0-1p0.2:0.6$" sets coupling phase between CPG 0 and CPG 1 to be optimized between 0.2 and O.6. The parameter can be fixed with a range of size 0 (ex: "0.3:0.3" fixes the parameter to 0.3). |
| "psoparameterid" | Select CPG parameters ID wanted to used in PSO | "$sC0-2psoparameterid6p0:6:7:8:9:10:11:31$" set the ID of 6 parameters. |
| | **Commands dealing with PSO particles** | |
| "psoinit" | Initialize PSO, i.e. put the particles randomly in the optimization space. | |
| "psonext" | Go to next particle and set fitness of the current particle. | "$sC0-2psonext134$" assign fitness 134 to the current particle and begins evaluating the next particle. |
| "psobestpart" | Change the best particles. | "$sC0-2psosetbest03p4.000:5.000:2.678:-7.32:9.324:-3.24:45.32$" set best particle 3 to position (4.000:5.000:2.678:-7.342:9.324:-3.234) and fitness 45.32. Note that the index of the particle should have 2 CHARACTERS(else it would not work). |
| "psoparticle" | Change the particles positions. | "$sC0-2psosetpart03p4.000:5.000:2.678:-7.32:9.324:-3.34:45.32$" set particle 3 to position (4.000:5.000:2.678:-7.342:9.324:-3.234) and fitness 45.32. Note that the index of the particle should have 2 CHARACTERS(else it would not work). |
| "psocurrent" | Change current particle. | "$sC0-2psocurrent5$" set current particle to 5 ?C0-2psocurrent to know the current particle |
| "psoiteration" | Change or set the current iteration of PSO. | "$sC0-2psoiteration3$" set the current iteration to 3 |
| | **Other commands** | |
| "version" | Print firmware version | |
| "led" | Deal with LED | |
| "l" | Deal with LED | |

A prefix **s** is used for setting a value and a prefix **?** is used for requesting the current value. Each command ends with the end line character.

The convention of parameters ID is stated as in Table. 2 with N oscillators, here $N = 6$
By using command $psoparameterid(number\_of\_parameter)p(parameter)$ ::::: such as $psoparameterid6p0:6:7:8:9:10:11:31$. Using this function, users can select parameters flexibly without modifying the firmware code time by time.

Table 2: Parameters' IDs in our CPG model

| OSC | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| R | 0 | 1 | 2 | 3 | 4 | 5 |
| X | 6 | 7 | 8 | 9 | 10 | 11 |
| $\phi_{ij}$ i | 0 | 1 | 2 | 3 | 4 | 5 |
| j | | | | | | |
| 0 | 12 | 13 | 14 | 15 | 16 | 17 |
| 1 | 18 | ... | | | | ⋮ |
| ⋮ | | | | | | |
| 5 | ... | | | | | 47 |

# 6 Conclusion

The manual guides users how to use the software, XML file to conduct experiments in "Online optimization for the locomotion of Roombots". The software can be extended for other experimental purposes with other robots structures. For more information, please refer to the final report [2] or contact the author via email theanh.nguyen@epfl.ch

# References


[1] OpenFramework, "OpenFramework v.007." http://openframework.cc/.

[2] A. T. Nguyen, "Online optimization for the locomotion of roombots."