Artificial Intelligence Lab
Department of Informatics
University of Zurich
Andreasstrasse 15
8050 Zurich
Switzerland

ShanghAI Lectures

Fall Term 2012

Exercise 4: Swiss Robots

Distribution: 8 November 2012

Due Date: 19 November 2012

For questions, please contact Xiaoxiang Yu, yuxiaoxiang@ifi.uzh.ch

Total number of points: 25

1. INTRODUCTION

Within the field of robotics, the design and development of autonomous robots to perform tasks is of special interest. Although such robots have potential applications in a wide variety of domains, design and control techniques for such robots is far from being well established. In this regard, research by Maris and Boekhorst, 1996 has explored alternative design methodologies based on exploiting constraints created as a consequence of the morphology and interactions with the environment.

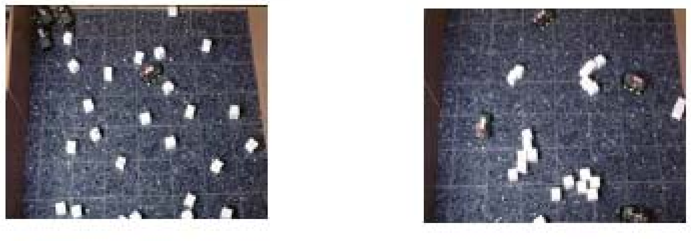


Figure 1: (a) Initial Configuration, (b) Final Configuration after a 20 min trial.

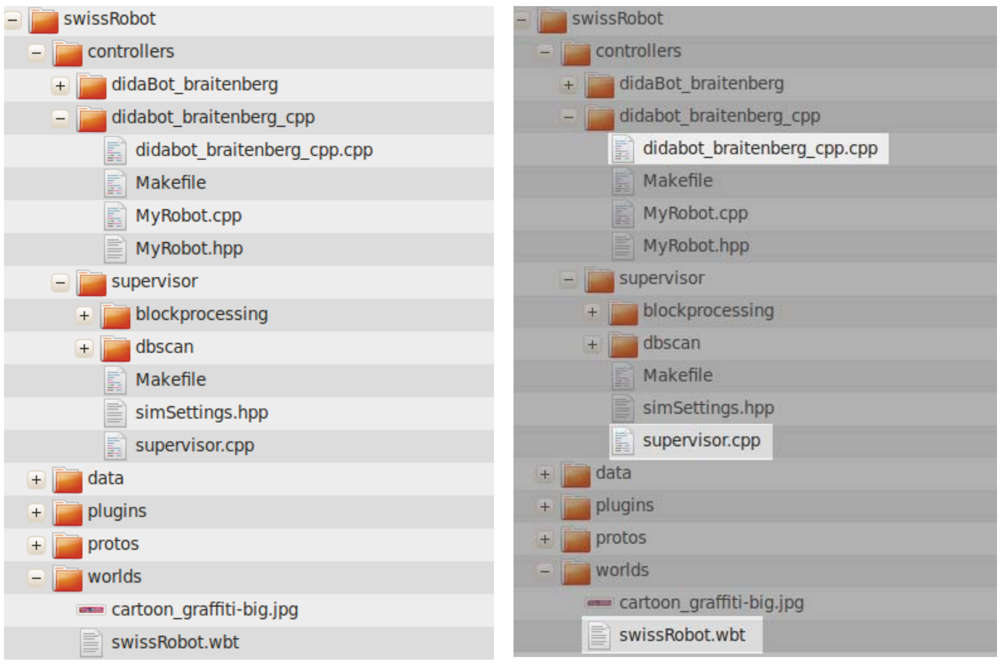
Maris and Boekhorst were inspired by examples from observations of social insect behaviour, where it was hypothesised that simple behavioural rules could lead to the emergence of complex, seemingly intelligent collective behaviour. They proposed that simple robots (simple in both morphology and control) could demonstrate the sophisticated collective behaviour of heap formation through a technique termed "strategy of errors". They utilised the robots known as Didabots to perform this experiment. This approach relies on robots reacting to obstacles, objects, and walls in the same way, i.e. by trying to avoid them. However, the objects could be pushed by the robot if they happen to be in the "blind spot" (frontal area of the robot not covered by distance sensors). This pushing behaviour over a period of time leads to the emergence of clusters (or a single cluster) as seen in Fig.1 . They report a formation time of under 20 minutes. This example was later termed as Swiss Robots (as presented in your course textbook [6]) and it is often cited to demonstrate various aspects of embodied agent design principles. In this exercise, you will get a chance to experience the effect of morphology and interaction with the environment, upon the behaviour of such robots.

2. INSTRUCTIONS ON LOADING AND EXECUTING A SIMULATION

You will be provided with a simulated version of the Swiss Robots in their arena along with the blocks that they have to heap ("clean"). The task environment is specified in the swissRobot.wbt world file. Please download the simulation from [3].

2.1 Loading the Environment

The files provided to you are organised as in the directory tree shown in Fig. 2. To load the world you can use the *swissRobot.wbt* world file.

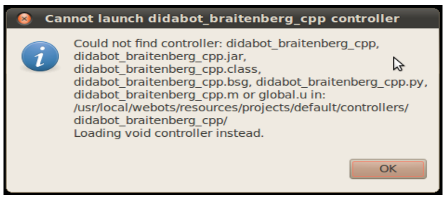


```
− swissRobot
  − controllers
    + didaBot_braitenberg
    − didabot_braitenberg_cpp
        didabot_braitenberg_cpp.cpp
        Makefile
        MyRobot.cpp
        MyRobot.hpp
    − supervisor
      + blockprocessing
      + dbscan
        Makefile
        simSettings.hpp
        supervisor.cpp
  + data
  + plugins
  + protos
  − worlds
      cartoon_graffiti-big.jpg
      swissRobot.wbt
```

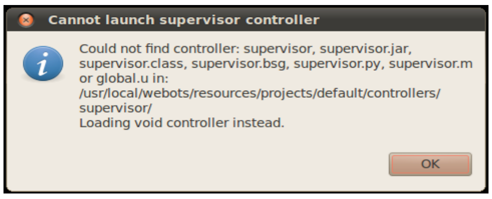
(a) (b)

Figure 2: Directory Structure you will be provided (a) Directory Tree (b) Files you need to directly access/modify are highlighted

To load the world, use *File → Open World* from the menu. Once loaded you will see the world as in Fig. 4 loaded along with the Scene tree with the various components. Immediately you will be faced with the pop up error messages as shown in Fig. 3. Ignore them for now, and click OK.


Cannot launch didabot_braitenberg_cpp controller

Could not find controller: didabot_braitenberg_cpp, didabot_braitenberg_cpp.jar, didabot_braitenberg_cpp.class, didabot_braitenberg_cpp.bsg, didabot_braitenberg_cpp.py, didabot_braitenberg_cpp.m or global.u in: /usr/local/webots/resources/projects/default/controllers/didabot_braitenberg_cpp/ Loading void controller instead.


Cannot launch supervisor controller

Could not find controller: supervisor, supervisor.jar, supervisor.class, supervisor.bsg, supervisor.py, supervisor.m or global.u in: /usr/local/webots/resources/projects/default/controllers/supervisor/ Loading void controller instead.

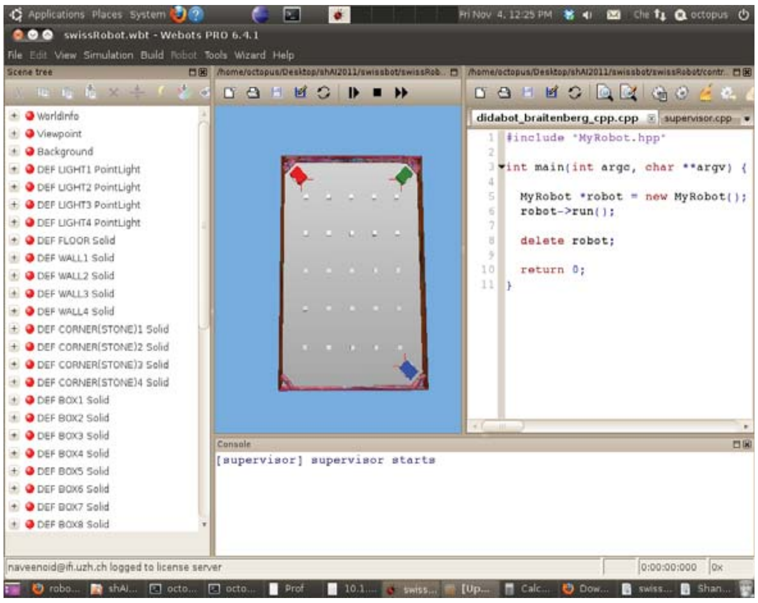
(a) (b)

Figure 3: Error Messages you will encounter : (a) Controller cannot be launched, (b) Supervisor cannot be launched

2.2 Components of the Environment

In Webots, the SwissRobot is modelled using a tree structure that reflects the hierarchy of how the parts are attached to each other. You find this hierarchy in the window on the left called "Scene Tree" as in Fig. 4. When you open the world, you will notice the following components (loaded in the scene tree) :

1. Arena : Grey coloured rectangular area which needs to be "*cleaned*". The Arena is depicted below in Fig. 4
2. Walls : Coloured walls that demarcate the area.
3. Boxes : 25 white boxes initially uniformly distributed spatially. They have to be heaped to accomplish the "cleaning".
4. Robot(s) : Up to 3 (red, green and blue) differential wheel robots[1] which utilise IR sensing.
5. Controller : The software component that provides the behavioural rules for the robot's behaviour. In this case, it is a Braitenberg style controller, which enables it to keep moving and to avoid obstacles. Braitenberg Controllers are explained in [6].
6. Supervisor : A component which allows specification of simulation parameters and allows post processing of world components. This component allows for accurate timing of an experiment and, upon completion, collects information about box positions.



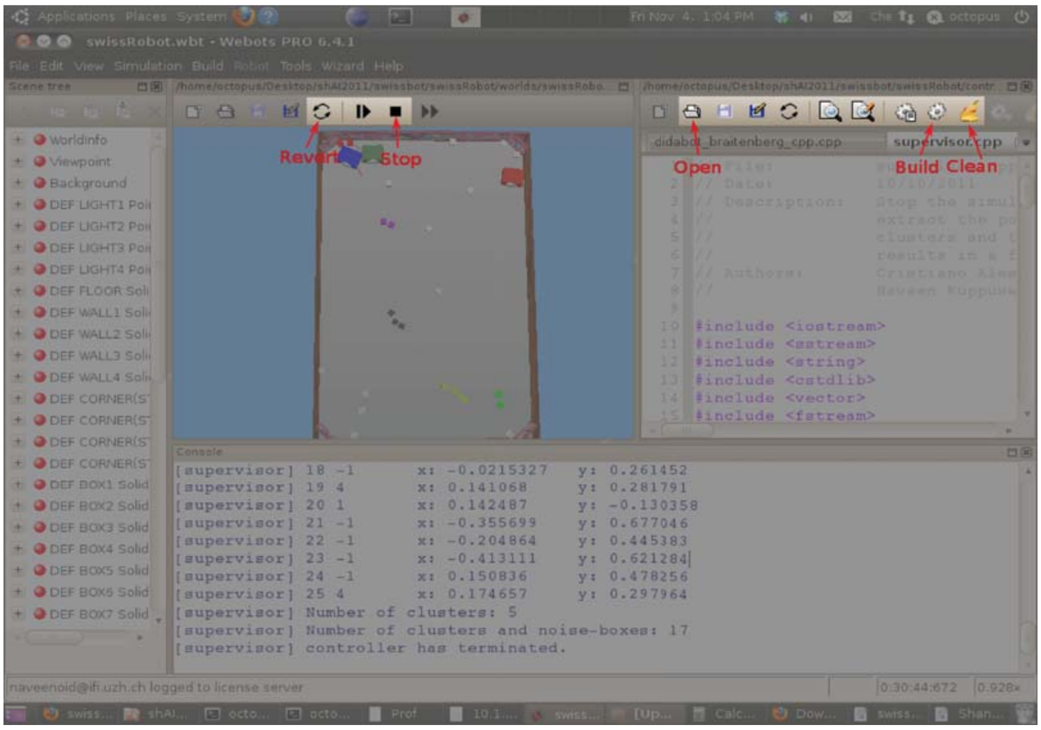
```
#include "MyRobot.hpp"

int main(int argc, char **argv) {

  MyRobot *robot = new MyRobot();
  robot->run();

  delete robot;

  return 0;
}
```

```
[supervisor] supervisor starts
```

Figure 4: Initial Condition of the SwissRobot Arena

1 These are robots with 2 wheels which can be controlled independently. Turning is accomplished by using different velocities at each wheel.

The UI elements that you need to pay attention to are shown in Fig. 5. The Controller and Supervisor editor window can be seen to the right of the screen. How and when to utilise these elements will be explained subsequently.



```
// File:
// Date:              10/10/2011
// Description:       Stop the simul
//                    extract the po
//                    clusters and t
//                    results in a f
// Authors:           Cristiano Alem
//                    Naveen Kuppusw

#include <iostream>
#include <sstream>
#include <string>
#include <cstdlib>
#include <vector>
#include <fstream>
```

```
[supervisor] 18 -1       x: -0.0215327   y: 0.261452
[supervisor] 19 4        x: 0.141068     y: 0.281791
[supervisor] 20 1        x: 0.142487     y: -0.130358
[supervisor] 21 -1       x: -0.355699    y: 0.677046
[supervisor] 22 -1       x: -0.204864    y: 0.445383
[supervisor] 23 -1       x: -0.413111    y: 0.621284
[supervisor] 24 -1       x: 0.150836     y: 0.478256
[supervisor] 25 4        x: 0.174657     y: 0.297964
[supervisor] Number of clusters: 5
[supervisor] Number of clusters and noise-boxes: 17
[supervisor] controller has terminated.
```

Figure 5: Important UI Elements needed for this excercise.

2.3 Building Controller and Supervisor

Controller and Supervisor have to be cleaned and built before any trial run. Load the controller using the "*File open"* button on the Controller editor (on the right hand side of the world) shown in Fig. 5. The controller is a C++ file, called *didabot_braitenberg_cpp.cpp* and is located in the folder *swissRobot/controllers/didabot_braitenerg_cpp/* as depicted in the directory tree in Fig. 2.

Click on "*Clean*" to ensure that no prior built versions exist, and click on the "*Build*" to start Building.

Repeat the same process for the supervisor as well; the supervisor is a C++ file, called *supervisor.cpp* and is located in the folder *swissRobot/controller/supervisor/* as in Fig. 2.

Once the controller and supervisor are built you are ready to perform a trial.

NOTE : WHENEVER ANY PARAMETERS ARE CHANGED IN THE WORLD FILE, PLEASE CLEAN AND REBUILD BOTH THE CONTROLLER AND THE SUPERVISOR, OR ELSE THE CHANGES WILL NOT BE REFLECTED IN THE BEHAVIOUR.

2.4 Running a Trial

To initiate a simulation run (or a trial) click "*Stop*" in the simulation menu in the middle of the screen, and then click on "*Revert*" to reset the world to the original saved version. Now clean and build the supervisor, and then initiate a trial by clicking on "*Run*". When the simulation is initiated, the robots move about the world, sometimes turning away from obstacles due to the braitenberg controller. Due to the blind spot in their front (caused by the wide spacing of the IR sensors) the robots tend to push some of the boxes. To visualise the distance sensor rays, you can use the following option from the webots menu: *View → Optional Rendering → Show Distance Sensor Rays*.

A single simulation trial has been preset to execute for 2 minutes. Simulations are deterministic for a given machine (platform) and a given webots version. This means that a certain simulation is repeatable when executed on the same computer but not necessarily across different computers.

NOTE : BEFORE MAKING ANY CHANGES ALWAYS STOP THE SIMULATION AND REVERT THE WORLD TO ITS ORIGINAL STATE. THIS IS IMPORTANT TO PREVENT CHANGES IN INITIAL CONDITION OF ROBOTS OR BOXES WITHIN THE ARENA

2.5 Clustering Algorithm

The performance of the robots is measured in their ability to form *"heaps"* of boxes, interpreted as clusters of boxes. At the end of a simulation (that has a fixed duration) a clustering algorithm (DBSCAN, refer to [4] for details) computes the number of clusters. Boxes that are not assigned to any cluster are denoted as "*noise*".

For this exercise, the heaping/cleaning performance is quantified by the sum of the number of noise boxes with the number of clusters themselves. Your task thus, will be to find parameters which can minimise the sum of clusters and noise boxes.

3. EXPERIMENTAL PROCEDURE

In this exercise you are required to perform a set of experiments to test:

1. Effect of sensor range
2. Effect of the size of boxes
3. Effect of number of Swiss Robots in the arena.

The outcome of each experiment is quantified by the Number of Clusters+Noise boxes (see Sec. 3.2.1).

3.1 How to Perform an Experiment

A single experimental run, referred to as a trial, can be performed by the procedure explained previously in Section 2.3 and 2.4. You may change parameters from the Scene Tree on the left. The parameters that you are required to modify for this exercise are described in Section 3.3. After changing the parameters, save the world from the File menu as : *File → Save World.* Initiate a trial by following the procedure explained earlier.

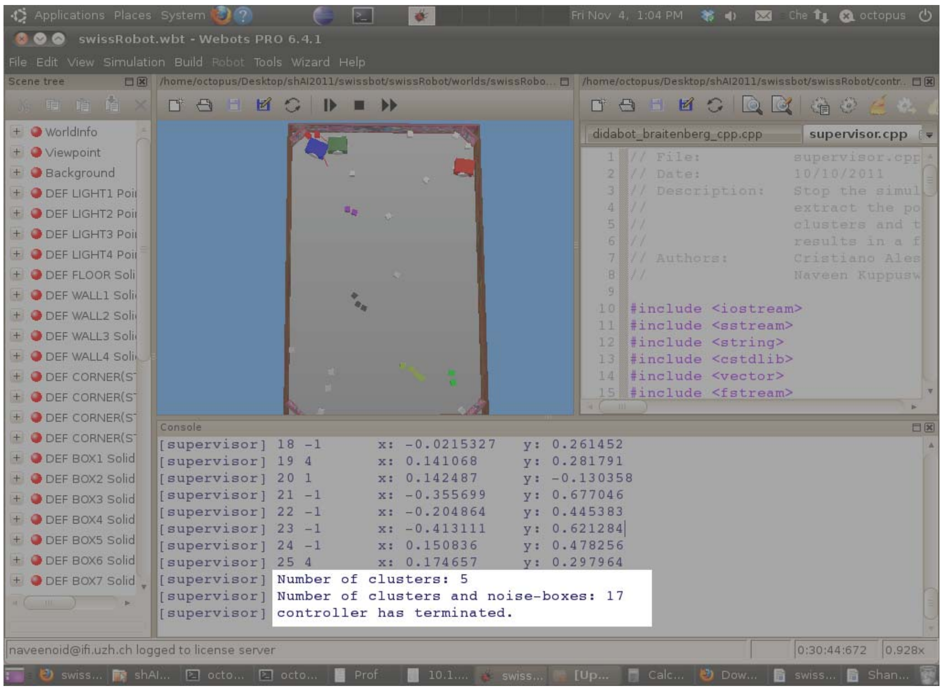
NOTE : AFTER PARAMETERS ARE CHANGED, REMEMBER TO CLEAN AND REBUILD THE SUPERVISOR AND THE CONTROLLER TO ENABLE THE CHANGES TO BE REFLECTED IN THE ROBOT BEHAVIOUR.

3.2 How to obtain the Results of a Trial

Heaping ("cleaning") is quantified in the form of number of clusters + noise boxes. This is computed and displayed on screen at the end of a trial. Additionally you may also obtain a screen shot at the end of a trial. Videos of complete trials may also be obtained.

3.2.1 How to find the outcome of a trial

The result of a trial is quantified as explained in Section 3.2. The trial outcome is computed upon completion of a trial, and is displayed on the Console Window. This is highlighted in the screen grab in Fig. 6.



```
// File:            supervisor.cpp
// Date:            10/10/2011
// Description:     Stop the simul
//                  extract the po
//                  clusters and t
//                  results in a f
// Authors:         Cristiano Ales
//                  Naveen Kuppusw

#include <iostream>
#include <sstream>
#include <string>
#include <cstdlib>
#include <vector>
#include <fstream>
```

```
[supervisor] 18 -1      x: -0.0215327   y: 0.261452
[supervisor] 19 4       x: 0.141068     y: 0.281791
[supervisor] 20 1       x: 0.142487     y: -0.130358
[supervisor] 21 -1      x: -0.355699    y: 0.677046
[supervisor] 22 -1      x: -0.204864    y: 0.445383
[supervisor] 23 -1      x: -0.413111    y: 0.621284
[supervisor] 24 -1      x: 0.150836     y: 0.478256
[supervisor] 25 4       x: 0.174657     y: 0.297964
[supervisor] Number of clusters: 5
[supervisor] Number of clusters and noise-boxes: 17
[supervisor] controller has terminated.
```

swissRobot.wbt - Webots PRO 6.4.1
File Edit View Simulation Build Robot Tools Wizard Help
Scene tree
WorldInfo
Viewpoint
Background
/home/octopus/Desktop/shAI2011/swissbot/swissRobot/contr..
didabot_braitenberg_cpp.cpp
supervisor.cpp
Console
naveenoid@ifi.uzh.ch logged to license server
0:30:44:672
0.928x

Figure 6: Result of a trial, number of clusters and number of noise boxes

3.2.2 How to make screen shots

If you wish to obtain a nice screenshot, first reorient the world view to obtain something similar to Fig. 6. This can be done either using a mouse, or from the following nodes in the scene tree:

```
Viewpoint → orientation

Viewpoint → position
```

you can take a screen shot using *File → Take Screenshot* from the menu.

3.2.3 How to make a video of your trial

You can take a video using *File → Make Movie* from the menu.

3.2.4 How to change simulation duration

The simulation duration is defined in line number 27 of the supervisor.cpp file. Remember that it is specified in seconds.

NOTE : AFTER YOU CHANGE THE SIMULATION DURATION, REMEMBER TO CLEAN AND REBUILD THE SUPERVISOR AND THE CONTROLLER TO ENABLE THE CHANGES TO BE REFLECTED IN THE ROBOT BEHAVIOUR.

3.3 Experiments to be Performed

The experiments you have to perform involve changing some of the parameters described in the Scene Tree and observing the performance obtained.

3.3.1. Effect of Sensor Range

The range of the 2 IR sensors in front of the robot can be modified by the nodes in the following path in the scene tree

```
ROBOT DifferentialWheels → children → LEFTSENSOR DistanceSensor→lookupTable
ROBOT DifferentialWheels → children → RIGHTSENSOR DistanceSensor→lookupTable
```

The actual format of the lookup table is described in [5]. You only need to adjust the elements in the second and third row of the first column, for changing range of the sensor. The scaling of 0 to 1024 need not be adjusted.

NOTE : ALWAYS MAKE SURE THAT YOU ADJUST ONLY THE ELEMENTS IN THE 2nd AND 3rd ROWS OF THE FIRST COLUMN. MAKE SURE THAT THE ELEMENT IN THE 3rd ROW EXCEEDS THE 2nd ROW ELEMENT BY A MARGIN OF 0.001

3.3.2. Effect of the box dimensions

The boxes to be clustered are defined as cubes. Size of all the cubes can be simultaneously modified in the node

```
BOX1 Solid → children → dirtBox Shape → geometry Box → size
```

NOTE : PLEASE CHANGE ALL OF THE DIMENSIONS -X,Y,Z- TO MAINTAIN THE CUBE FORM.

3.3.3. Effect of Number of Robots

The world file already provides you with three robots (1-Red, 2-Green, and 3-Blue). The initial locations of these robots as specified in the Scene Tree in `DEF ROBOT`$_i$ `→ Translation` (relative to the world) are as follows :

ROBOT	COLOUR	INITIAL POSITION (X,Y,Z)
1	Red	0.35,0,0.6
2	Green	-0.35,0,0.6
3	Blue	-0.35,0,-0.6

To change the number of robots that will actively participate in a trial, simply reposition any unnecessary robots outside the arena. A recommended setting is to change the Z value from 0.6 → 0.8 for Robot 2, and -0.6 → -0.8 for Robot 3.

NOTE : MAKE SURE THE ROBOTS YOU WISH TO REMOVE ARE PLACED SUFFICIENTLY FAR AWAY FROM THE ARENA, AND NOT OVERLAPPING WITH THE WALLS OF THE ARENA.

4. EXERCISES

1. (*8 Points*) In this task, you will examine the impact of the box size and distance sensor range on the performance of the robot in the cleaning task. Plot the performance (#of clusters+noise) as a function of the following parameters (in independent plots), in the ranges specified.

a) Size of the boxes (in the range 0.025-0.055m in steps of 0.005m – with sensor range fixed to 0.05m)

b) Range of the sensor (in the range 0.01-0.1m in steps of 0.01m – with size of boxes fixed to 0.025m)

Try to interpret your results. What conclusion on the "*best design*" for a cleaning task, can you draw from your results?

Deliverables : Plots, Parameters for the best design (size of boxes, range of sensors), and video of the best performance.

2. (*5 Points*) In this task, you will examine the impact of the number robots present in the arena, on the cleaning performance. Pick the best design parameters obtained from Task 1 and test the performance (#of clusters+noise) as a function of the number of robots present in the arena (3,2, and 1 robot). Try to interpret your results. What conclusions can you draw from your results.?

Deliverables : Plot, and a Video of each of the three cases.

3. (*12 Points)* Answer the following questions:

a) Can you conclude from the results of your experiments that the best possible design (parameter settings that you obtained), is general enough for any given initial condition of the world? Explain why.

b) Do you think your design in the virtual world would give similar performance in a real world scenario (like in the arena in Fig.1, from [1]). Explain why.

c) Would the dimensions of the arena have an impact on the cleaning performance of your best design? Explain why.

d) List at least 3 possible parameters that could impact the cleaning performance (parameters that are not taken into account in your experiments so far).

e) Compare your robot's "intelligent" cleaning behaviour with that of Simon's Ant on the Beach, from the viewpoint of the frame-of-reference issue.

f) What could be the role played by the agent-environment interaction in the behaviour that you observe?

g) "Seemingly complex and sophisticated behaviours can emerge from very simple rules" - Do you think your experiments support this statement? Explain.

h) Do you think this design strategy is applicable to real world floor cleaning robot problems, such as the design of the Roomba [7]. Justify.

4. (*2 Bonus Points*) How does the duration of a trial affect the performance? Choose the best design you have obtained in Task 1 and simulate the behaviour of all 3 robots for 10 minutes. Comment on the results.

5. (*No Points, for your curiosity*) How would you apply the concepts of self-organisation and emergence to explain the behaviour of the Swiss Robots?

6. (*No Points, for your curiosity*) How do you think the usage of artificial pheromones could be helpful to the behaviour of the robots (on this or other tasks)?

Acknowledgements

We thank Yvan Bourquin and Olivier Michel from Cyberbotics for their Webots support.

References

[1] Marinus Maris, and René te Boekhorst, "Exploiting Physical Constraints: Heap Formation through Behavioral Error in a Group of Robots", In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems* IROS-96.

[2] Webots, http://www.cyberbotics.com/

[3] ShanghAI LecturesWebsite http://shanghailectures.org/

[4] DBSCAN Algorithm reference : http://en.wikipedia.org/wiki/DBSCAN

[5] Webots Distance Sensor Documentation : http://www.cyberbotics.com/reference/section3.20.php

[6] Rolf Pfeifer, and Josh Bongard, "How the body shapes the way we think : a new view of intelligence", MIT Press, London, England, 2007. (p72-75 Frame of Reference and Swiss Robots, p80 – Braitenberg Vehicles)

[7] Roomba robot, http://en.wikipedia.org/wiki/Roomba