IT3708 - Project 4: A Controller for Swarm Behaviour in Webots

Written by: Thomas Almenningen, Halvor G. Bjørnstad, Daniel Koziatek

Introduction

The main task for this project was to design a controller based on Brooks Architecture to explore cooperative transport of a box in a webots environment. The intended task is inspired from the swarm behavior of ants observed during food retrieval.

For this project we used Webots to simulate the e-puck and the environment . The e-pucks interact with the environment by using proximity- and IR-sensors, and left and right wheels are used as actuators.

All the code produced for this project was written in Python. The supplied behavior models were also translated into Python. Our code is not very pythonic as we translated the supplied c code very quick and almost 1:1 into equivalent verbose python. Our controller inherits from epuck_basic and has object references to stagnation, retrieval and search.

1. System Overview

1.1 Overall System Description

In our implemented system we have decomposed the problem into task-achieving actions which have direct access to actuators and sensors. Each layer is placed on top of each other where the higher levels have priority over the lower ones.The design is incremental meaning that the higher layers won't be triggered unless necessary.

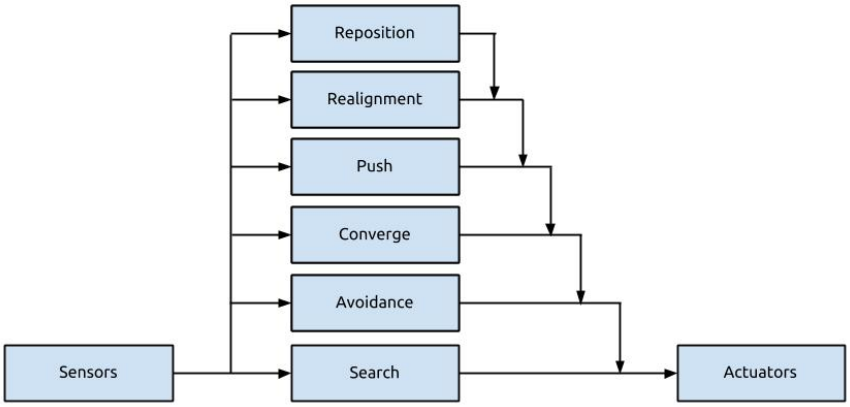

Figure 1: The behavior-based flow of our Brook's subsumption architecture

Starting from the bottom our Search and Avoidance behaviors ensure the survival of our system. When no IR-source is detected the system will explore the world in search of a food source. When the e-puck gets close to colliding the avoidance behavior will ensure that the robots stays alive by avoiding these. The distance threshold for detecting and dodging objects is set to *200*.

The retrieval module contains behavior that aims to find and ultimately push the box. The converge behavior is triggered when one or more of the IR-light sensors yield values lower than the IR threshold (3500). As an e-puck approaches to the food source the detected IR values will decrease(in the direction of the food source), at the point where any IR value is lower than the push threshold (100) the pushing behavior will be triggered. If the selected behavior is pushing and both of the e-puck's front sensors detect IR values lower than the *IR threshold* it will simply ram the box, maximizing both wheel speeds.

For the stagnation module we took some inspiration from the timed_review method described in chapter 6 of the master's thesis [1]. At the end of each stagnation test the feedback value will either be increased or decreased depending on the e-pucks success. This feedback value decides how long the e-puck will try to push the box between each review and is calculated by using the following formula. We felt that the original function output too high time values, so we modified it to fit our needs better.

$$timed\ review\ = (\frac{5}{(10-feedback)} * 100) + 20$$

If the selected behavior is push, the stagnation state is tested every timed_review iteration. The e-puck will try to realign twice before repositioning, resetting the realignment count.

1.2 Implemented Changes to the world

Initially in our project we struggled with IR-sensor problems, the detected IR values did not decrease as the e-pucks approached the food source, causing the retrieval module to malfunction. To circumvent this problem we modified the world, mostly by tweaking the food source. The food source's updated attributes are listed below:

```
DEF FOOD Solid
        Children
                Ambient Intensity: 0
                Attenuation: 1 2 3
                Color: 1 1 1
                Intensity: 0.35
                Location: 0 0 0
                on: True
                Radius: 75
                Cast Shadows: FALSE
```

2. Observed behavior

To be able to compare the later implemented improvements to our initial controller we decided to perform 10 timed runs to test its performance.

Run #	Time Spent	Notes
1	2:33	
2	2:32	
3	-	E-puck trapped in corner
4	3:23	
5	2:11	
6	6:11	
7	2:12	
8	2:48	
9	-	E-puck trapped in corner
10	3:32	
Avg.	3:10	

Table 1: Completion times for the push task with the initial system

Overall we are pretty satisfied with the current behavior of the collective system. The general logic is in place and working fairly well. We also managed to achieve the goal in 8 of 10 possible runs. The problem that occurred both times when we were unable to achieve the goal was that one or more e-pucks got trapped between the wall and the box, with the other e-pucks pushing from the other side.

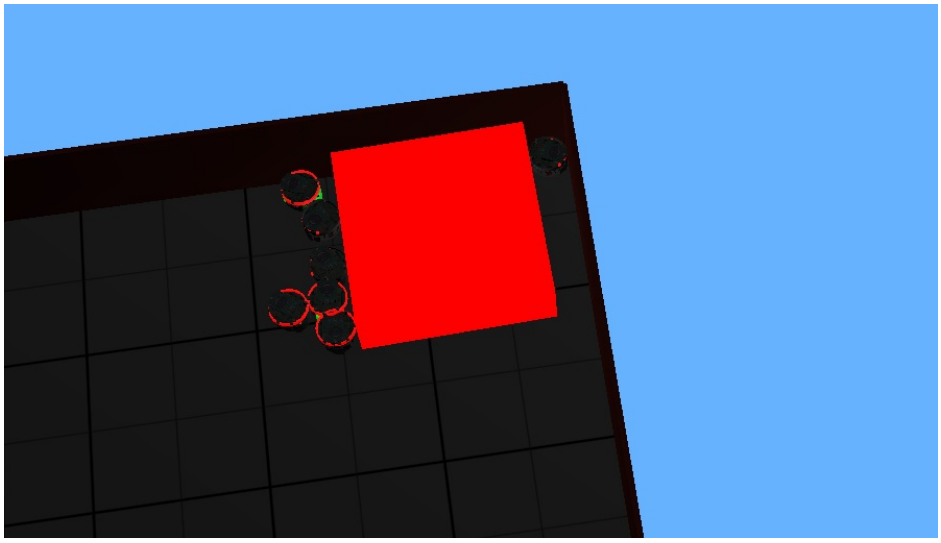

Figure 2: Trapped e-puck

Below we will look closer at the behavior of the e-puck by first reviewing the expected behavior, before taking a closer look at the unexpected behavior we observed during these runs.

2.2 Expected Behavior

All the e-pucks were currently converging to the box as they should. The pushing behavior was initiated at the right distance from the box. Stagnation was working more or less as intended as the e-pucks both tried to realign and found new positions to push from. We also noticed that e-pucks with no neighbors more often than not went into stagnation faster than e-pucks that had a neighbor pushing the box next to it.

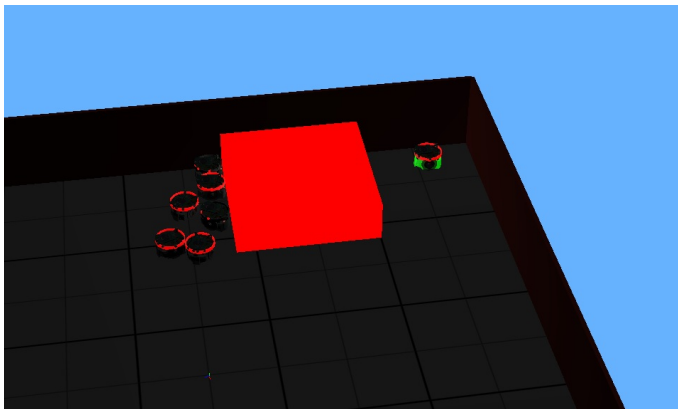

Figure 3: Find new spot behavior in action (Green e-puck)

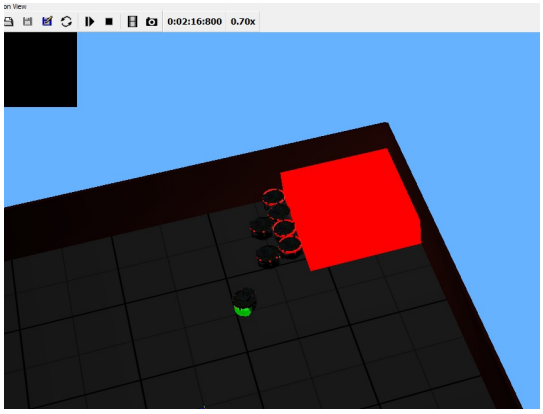

Figure 4: Success! [2:16]

2.3 Unexpected Behavior

During our ten initial test-runs we observed some unexpected behavior. The first thing we noticed was that e-pucks sometimes flipped over when multiple e-pucks were pushing a box side by side. When the e-pucks are grinding against each other they sometimes start to wobble, which sometimes causes the e-puck the flip over on the side.

The main problems we observed seemed to lay with neighbour detection. The proximity sensors seems to struggle more with detecting the distance to other e-pucks than for solid objects such as the walls and the box. E-pucks that were trapped between a wall and a neighboring e-puck believed they were next to two e-pucks, which caused them not to stagnate when they perhaps would have otherwise. It is also worth mentioning that the proximity sensors values in particular fluctuate violently, and that this is extra visible when there is some sideways movement when pushing the box. This problem sometimes caused e-pucks to go into stagnation even though they were surrounded by other e-pucks pushing the box from the same side. We tried setting the *NEIGHBOUR_LIMIT* as low as 100, but it did not resolve the problem completely.

We also experiences some problems related to the push evaluation function. This is mainly due to the highly fluctuating proximity sensor values which makes it hard to determine what value to assign *DISTANCE_DIFF_THRESHOLD* for it to work optimally. This problem led the e-pucks to sometimes think they weren't making any progress when they actually were pushing the box.

Other than this we did not spot any recurring general behavior pattern worth mentioning while running our initial tests. We fixed the glaring issues with our first controller before we were satisfied enough to perform the tests.

3. Identifying Areas for Improvement

In this part we will address some of the problems we observed while doing our initial testing. We will look closer into how these aspects of the controller could be improved and suggest possible solutions. Our main focus will be on the push evaluation function which from our initial runs seemed to have the most room for improvement.

An area of improvement we deemed worthwhile to delve deeper into was to monitor the acceleration of pushing e-pucks, if the acceleration in either the x or y direction had changed significantly over a period of pushing (even once) we believed it could signify that the e-puck was currently pushing with success. The main reason for our faith in the accelerometer was that its sensors are much more stable than the distance sensors. Using the accelerometer could prove to be a challenge, differentiating between being pushed and pushing was the largest challenge we foresaw.

Due to the relatively small size in the box there is not room for more than for 3-4 e-pucks to push next to each other. In part 2 we mentioned some problems with detecting neighbors. We suggest that we could proximity sensor 3 and 4 in addition to 2 and 5 to check for neighbors since this would also include e-pucks pushing the e-puck from behind.

Another thing we noticed was that the e-pucks sees walls as neighbors when when doing a push evaluation. A work around this issue could be to say that all sensor values coming from sensor 2 and 5 which are above *~2600* will be considered objects and not other e-pucks since neighboring e-pucks *normally* give lower proximity values than solid objects.

4. Implemented Improvements

Most of the improvements made to the new controller is a refinement of the constants used in the different behavior modules. We have also made some slight changes to our world, where we mainly tweaked the PointLight object.

To improve the push evaluation function we implemented a check which tries to separate neighbouring e-pucks from solid objects. This was done by adding additional check to the values of proximity sensors 2 and 5. We check if the proximity value is above a certain threshold (2600) to decide whether the detected object is a solid object or another e-puck. If the e-puck detects an object is will not count this as a neighbor, thus go into stagnation recovery.

We also tried to experiment with the accelerometer to see if we could improve the progress detection part of our push evaluation function. After a lot of trial and error (mostly error) we decided to give up. The main issues we encountered was setting a threshold to differentiate between significant and insignificant changes in the accelerometer's values and detecting that small changes many times could mean that the e-puck had stopped (slowly grinding to a halt). In retrospect we see the folly of our hopes, we first mistook the accelerometer for a speedometer. When it dawned on us that the accelerometer was no such thing we tried to salvage the situation, but ultimately failed.

We are now interested in comparing the performance of the new and improved controller compared to our initial version. As in part 1 we will do 10 test runs to see if the general performance of our system actually has improved.

Run #	Time Spent	Notes
1	0:42	
2	2:06	
3	1:43	

4	3:15	
5	2:22	
6	-	Multiple e-pucks on opposite sides of the box lead to no progress
7	1:57	
8	-	E-puck trapped between box and wall
9	2:36	
10	3:02	
Avg.	2:13	

Table 2: *Completion time for the push task with the improved system*

We are pleased to observe that our implemented improvements worked as we intended, a minute better average time is nothing to scoff at. Although we acknowledge that there is much randomness in how fast it takes the e-pucks to achieve their goal, we would like to emphasize that even if these 10*2 test runs have no statistical significance they confirm a trend we have seen while testing and developing our controllers.

Closing words

We would all like to thank both Keith Downing and Pauline Haddow for inspiring lectures and fun, but challenging projects . We would also like to thank Vinay Gautam and Ole Petter Olsen for precious help throughout the semester.

References

[1] - http://daim.idi.ntnu.no/masteroppgave?id=6180