Otto-von-Guericke University Magdeburg

Department of Computer Science
Institute for Intelligent Cooperating Systems

Bachelor Thesis

Development and Evaluation of a Sphero Multi Swarm System

Author:
Michael Faber

June 15, 2023

Advisers:

Supervisor
Ph.D. Sanaz Mostaghim

Department of Computer Science
Otto-von-Guericke University
Universitätsplatz 2
39106 Magdeburg, Germany

Supervisor
Dr.-Ing. Christoph Steup

Department of Computer Science
Otto-von-Guericke University
Universitätsplatz 2
39106 Magdeburg, Germany

Faber, Michael:
Development and Evaluation of a Sphero Multi Swarm System
Bachelor Thesis, Otto-von-Guericke University
Magdeburg, 2023.

Contents

Abstract

In this thesis I set out to create a Sphero Multi Swarm System. To be better at recognizing the success of this I asked four leading questions. Is it even possible to create swarm behaviour with Sphero robots? Does it work in a tight corridor? Is it possible for two swarms to interact with one another? Does it still work in with tight corridors? In order to achieve this goal I first had to create a concept which would be able to do that.

For that I needed to be able to track multiple robots, make them move towards a goal and create swarm behaviour. For that I used several ROS implementations. A CNN based detection for the tracking, an EKF to remove wrong detections, the Navigation stack for the path finding and several swarm formulas for my swarm behaviour. This was created and tested in the swarmlab with the Sphero robots.

The evaluation confirmed that it was possible for the majority of my questions. However specific problems with the tracking and path finding became apparent which made it fail for the last question. The failed tracking was also due to failed implementation of the EKF. Additionally Bluetooth connectivity of the Sphero robots made it only possible for six different Sphero robots to work at the same time. This opens up the direction future works should take.

Acknowledgements

I want to thank my family and my friends for their support.

I also want to give a big thank you for the members of the swarmlab for their continued support of this work!

1 Introduction and Motivation

1.1 Motivation

The Sphero robot was the first kind of robot I actively programmed for. After being able to create movement it has been my goal to create some kind of Swarm behaviour. So it is my pleasure to see if I am able to create it.

Furthermore having interactions between two swarms is not yet prominent in the research field. So it has some interesting possibilities. It could be useful if you want to create sub swarms from a bigger swarm to do their own task without intervening with each other.

1.2 Aim of this thesis

It is my goal to create two different swarms consisting of Sphero robots. These two different swarms should be able to interact with one another. In order to test if I am able to do this, I will try to answer the following four questions:

1. Is it possible to create swarm behaviour with Sphero robots?
2. Can I create swarm behaviour with my Sphero robots in a tight corridor?
3. Is is possible to create two Sphero swarms which interact with one another?
4. Is it possible to create two Sphero swarms which interact with one another where both are contained in a separate tight corridor?

1.3 Structure of this thesis

In this thesis I will first talk about some related work in chapter 2. Here I'm trying to create an understanding of other works that will be needed for my own methods to work.

Following this will be the methods I used 3. Here I will explain everything that is necessary for me to create my own swarm behaviour.

Then I will test and evaluate if my methods actually worked 4.

After everything is done I arrive at my conclusion 5 whether I achieved what I set out to do or not. I will also specify some future work that can be done.

2
Related Work

2.1 Sphero and ROS

The Sphero is a small ball shaped robot with inbuilt odometer and an Inertial Measurement Unit a so called IMU. It has a differential drive and a low friction. In addition to that the Sphero possesses a bluetooth low energy interface. SPHERO (2022)

Thanks to that we can establish communication between the Sphero and another device be it a computer or a mobile phone.

There exist many different variants of Sphero robots. The one I am using is the so called SPRK+. For a better usage of the Sphero we use the Robot Operating System i.e. ROS. It is a node based system where each node can publish and subscribe to topics which contains specific information. JOSEPH (2017) Our entire system is running with different ROS nodes. What makes ROS great is also the possibility to either start a node as is or start it with a launch file where you can determine different variables and to start more than a single node at a time.

In Figure 2.2 I'm showing an example of what a possible ROS integration is looking like. Here I'm running a single Sphero robot with its navigation and tracking enabled. This shows how the different nodes interact with each other as not every node needs to communicate with every other node. It also becomes apparent that the Bluetooth communication is not working with ROS but is rather dealt with individually before the so called "Sphero node" brings the ROS integration.

Figure 2.1: Example of my Sphero SPRK+ robots

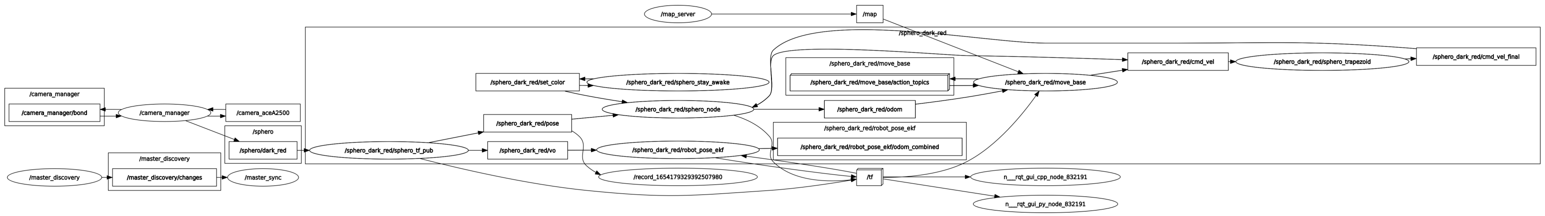

Figure 2.2: Example of running nodes with a Sphero robot

2.2 ROS Navigation Stack

To get an accurate Navigation which is intertwined with the ROS system I am using the Navigation stack also called navstack for short. For that to work it is only needed to provide odometry, sensor data if available and a goal pose in a provided costmap. If everything is correctly provided it gives you command velocities which will eventually reach the goal provided the given data is correct. Another prerequisite according to GUIMARÃES et al. (2016) is not only ROS but also a tf transformation tree. In addition it needs to be configured for the specific robot it is used for. This has to be done for it to work at a high level.

There are also some hardware requirements the robot needs to have according to LARRIBE (2020-09-14).

1. The robot needs to have either a differential drive or be holonomic wheeled and needs to be able to work with command velocities in the form of x velocity, y velocity, theta velocity.
2. It needs a mounted laser for map building and localisation
3. Works best for either square or circular shaped robots

However the need for a laser can be disregarded if you have some other kind of localisation as well as a known static map.

The actual path finding is calculated in the base local planner which is part of the navstack. Here we have a choice of two different algorithms, the Trajectory Rollout and the Dynamic Window Approach.

Both algorithms follow the same basic idea according to VARAS (2019-04-04):

1. Discretely sample in the robot's control space (dx,dy,dtheta)
2. For each sampled velocity, perform forward simulation from the robot's current state to predict what would happen if the sampled velocity were applied for some (short) period of time.
3. Evaluate (score) each trajectory resulting from the forward simulation, using a metric that incorporates characteristics such as: proximity to obstacles, proximity to the goal, proximity to the global path,

and speed. Discard illegal trajectories (those that collide with obstacles).

4. Pick the highest-scoring trajectory and send the associated velocity to the mobile base.

5. Rinse and repeat.

The only difference of DWA and Trajectory Rollout is how they do the very first step. The Trajectory Rollout samples the velocities over the entire simulation while the DWA only samples it from one simulation step. Both need the acceleration limit of the robot to be able to do that. This means that DWA is more efficient for robots with a higher acceleration limit.

For a robot to be able to move in an environment a map and a costmap is needed. The map consists of three different states. It can be free space, occupied space or unknown space. In comparison to that the costmap can have a value between 0 and 255. Thus if you depending on your threshold it will correlate to on of the three map states. In this map the color of each pixel describes whether or not the corresponding cell is occupied.

Now the color of the pixel can be interpreted in three different ways

1. Trinary: if the pixel has any other color than white the cell is occupied

2. Scale: this can give you a value between 0 and 100

3. Raw: will output a value between 0 and 255

2.3 Kalman Filter

In order to get an accurate measure of the Spheros position we need to filter the received position messages from the camera. To do that I decided to use the Extended Kalman Filter. An extended Kalman Filter is an extension of the normal Kalman Filter. The Kalman Filter gives a linear estimate about a mean and covariances about a current position. The extended Kalman Filter gives us a nonlinear estimation.

$$x_k = f(x_{k-1}, u_k) + w_k \tag{2.1}$$

$$z_k = h(x_k) + v_k \tag{2.2}$$

The formula 2.1 describes the state transition where w_k is process noise and u_k is a control vector. Folloing this the formula 2.2 describes the observation model where v_k is the observation noise.

The workings of an EKF can be roughly divided into two main cycles:

1. Predict
2. Update

If we go through both cycles of the EKF it does the following according to RIBEIRO (2004):

1. Consider the last filtered state estimate
2. Linearise the system dynamics
3. Apply the prediction step to the linearised system
4. Linearise the observation dynamics
5. Apply the update cycle to the linearised observation dynamics

At the end of these steps we attain our nonlinear estimate which can be used for localisation.

2.4 Transformations

One prerequisite of the navstack is the usage of the tf tree. TF is a ROS package that lets you track different coordinate frames over time. It also keeps track of their relationship in a tree like structure according to FOOTE (2013). This tree like structure also defines the hierarchy of the robot parts. Additionally this makes it possible to transform vectors between the different frames. In addition to that you can create queries to check your current pose in a map frame.

You can also do other things with this package like rotating, reflecting, scaling, shearing, projecting, orthogonalizing according to GOHLKE (2009).

Another useful function is the conversion between rotation matrices, euler angles and quaternions.

2.5 Camera Tracking

There exist many methods for an accurate localisation inside a specific area. In my case I will rely on a camera based tracking which is based on HOYER et al..

As can be seen in Figure 2.3 the detection process is divided into two parts.

The first part is basically a common object detection method which is why this method is so much better. After that it samples down the image and uses a CNN object detection to returns a type of robot and a bounding box back. HOYER et al. (2018). Following this it uses that data for the second stage. Here it takes the high resolution image and crops it down to the bounding box. Now for the second part the idea is that it uses a specifically trained neural network for the detected robot type. With this we get a position, an ID and an Orientation for our robot.

The CNNs can be trained with enough pictures meaning you can easily use any kind of robot. It is only necessary to create a new bounding box for a new robot. This makes it very easy to track new robots.

The training data acquisition can be divided into two steps. First there is the compositor. It is an algorithm which superimposes robot crops on background images with a random scale. HOYER et al. (2018) This creates a massive amount of data which can be used as it can contain multiple robots in different orientations. It also balances out all the important factors like the background type, robot type and id patterns. This will give us the necessary data that is then stored as a ground truth. Secondly there is the robot crops. These are essential for quality and the final data. For that to work these crops need to be without any visual background information.

2.6 Swarm Behaviour

A swarm is a conglomeration of different agents moving with a greater purpose and not just individually.

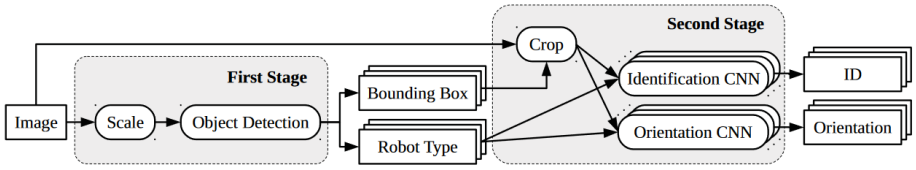

Figure 2.3: Architecture of the robot localization framework consisting of two stages. Adopted from HOYER et al. (2018)

According to Reynolds there exist three main criteria which every swarm follows: REYNOLDS (1987)

1. Cohesion: The need to move towards the center of local swarm members
2. Alignment: The need to steer towards the average orientation of your swarm
3. Separation: The need to avoid colliding with your fellow members

There are many real life examples of swarms. The most known ones are Ants, which are communicating via pheromones and are thus able to achieve many different things for the whole swarm. Otherwise Bees and Termites are known as swarm animals. Birds while flying in formation are well known swarm animals as well.

Through these real life examples it is possible to imitate their behaviour with robots.

To formulate an implementation of this behaviour there is this formula after MOSTAGHIM (2022).

$$\mathbf{f_{i,j}} = -(\mathbf{x_i} - \mathbf{x_j})(f_{attraction} - f_{repulsion}) \tag{2.3}$$

There are three main ways to formulate an attraction and repulsion function:

$$f_{ar_1} = f_{a_1} - f_{r_1} = \alpha - \frac{b}{||\mathbf{x_i} - \mathbf{x_j}||^2} \tag{2.4}$$

$$f_{ar_2} = f_{a_2} - f_{r_2} = \frac{\alpha}{||\mathbf{x_i} - \mathbf{x_j}||} - \frac{\beta}{||\mathbf{x_i} - \mathbf{x_j}||^2} \tag{2.5}$$

$$f_{ar_3} = f_{a_3} - f_{r_3} = \alpha - \beta \cdot e^{-\frac{||\mathbf{x_i} - \mathbf{x_j}||^2}{\gamma}} \tag{2.6}$$

α, β, γ are parameters which need to be chosen.

$\boldsymbol{x_i}, \boldsymbol{x_j}$ are members of the swarm.

These three attraction and repulsion functions while similar have different types of attraction and repulsion. In formula 1 2.4 it is a linear bounded attraction and a unbounded repulsion

Compared to the in formula 2 2.5 it is an almost constant attraction and an unbounded repulsion.

Lastly in formula 3 2.6 it is a linear bounded attraction and an unbounded repulsion.

Once $f_{repulsion} = f_{attraction}$ the swarm members have reached their optimal state and stop moving.

3 Methods

3.1 Overview

Now as I talked a bit about the related work for my plan it's now time to talk about my used methods.

For that let me give you an overview of my general ideas. On the ground level I have my arena where my Sphero robots will do their work. As they use Bluetooth LE to connect to a device I have Bluetooth dongles in my arena.

To start with I need to track my Sphero robots so I use a camera which is placed in the ceiling. Thanks to that I get an accurate view of everything. Thus I can use my trained CNNs to track my Sphero robots. As I get position data, an orientation and an id I can use an EKF to filter out bad data of my tracking. With this filtered data I can use my navstack for my path finding and create a swarm behaviour as they rely on position data to function.

My swarm behaviour is based on an irregular time step and will only send new goals once the last goal has been reached. Once my Sphero robots start their movement, my tracking continues and thus creating a cycle. This is visualized in my Figure 3.1.

3.2 Usage of my Camera Tracking

To correctly identify a Sphero robot I decided to use a camera based tracking as described in my related works. But before that is possible I need to create an rectified image from my raw camera image. For that I also use a standard ROS package image proc. In order to do that I created a

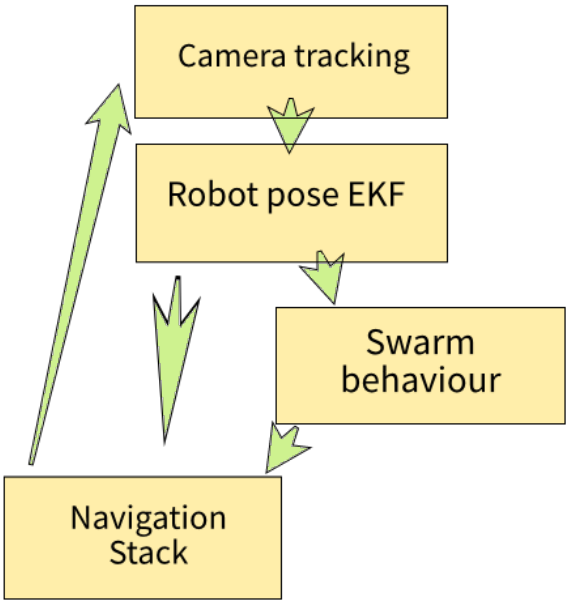

Figure 3.1: Interplay between my methods

new training set based on my Sphero robots. For that I trained my CNNs with 30000 pictures and later tested it with 3000. According to the test I achieved a 98 % accuracy to correctly identify the id and a 90 % accuracy to get a correct orientation.

Now with my correctly trained CNN it is possible to correctly track my Sphero robots. It does that with colors as IDs and gives me an Orientation. I trained the following ID: *dark blue,dark red, dark green, green, blue, red, yellow, blue green, magenta, purple, lime green, orange, light green and light blue.*

3.3 Path Finding

Now with a correct tracking I am using the navstack for my path finding. As I am getting good position data from the camera I need to create a visual odometry to further work with that.

I receive all the data I need so that is not a problem. Now I just need to publish transformations and the navstack can start its work.

For the path finding I decided to use the base local planner which is included in the navstack. As for the algorithm I decided to use the Trajectory Rollout which works as described in my Related Work under ROS Navigation stack.

3.4 Robot Pose EKF

To get an even better localisation I decided to use an EKF. For that I found a premade ROS Package Robot Pose EKF. Upon closer inspection of the code I found out that it was not possible to change the system noise. To rectify that I made that a variable which is changeable as a ROS parameter, meaning it is easily changeable in a launch file.

This package needs a particular set of parameters to work in a ROS environment. It needs to get an odometry of your robot, a visual odometry and IMU data. Theoretically it only needs either the odometry or the visual odometry to filter the data but in that case it is not as accurate.

Additionally it absolutely needs to get covariance matrices of your robot otherwise it will not work. This can pose a problem as you need to know the covariances of your robot otherwise you will never get a correct result. A covariance matrix is a 6x6 matrix where the diagonal corresponds to *linear x, linear y, linear z, angular x, angular y, angular z*

If you make your covariances to small the robot will simply never move from it's position no matter what. However if you do the opposite and those values that are too big the positions of your robot will simply converge even if the actual robot is not moving.

Lastly it also asks for your base frame and the output frame where it will publish its results.

3.5 Multi Swarm Behaviour

Now I get to my main problem of creating swarm behaviour. For that I use multiple Sphero robots with different colors to create a swarm.

There are a few methods to create swarm behaviour ,in my case I chose the known swarm formula 3 2.6 as described in my related work as my

attraction and repulsion function. Additionally the main formula needs to be used as well 2.3.

Now if we look at this formula 2.6 it is obvious that it describes the attraction and repulsion between two robot agents. α describes the linear attraction and the second part of the formula describes the unbounded repulsion between both agents. In order for this to work this needs to be calculated for every member of the swarm and added together.

Due to the nature of my setup I decided to use a different route based on the following paper GASPARRI et al. (2012) where they issued commands by a remote station and validated their findings.

Instead of every single robot calculating their attraction and repulsion with every single other robot in their swarm I decided to to it a little differently. I first calculate the center of my swarm based on every robot 3.1 and then calculate the attraction and repulsion based on the robot I am calculating it for and the center of the swarm. 3.2

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=0}^{n}\mathbf{x_i} \tag{3.1}$$

$$f_i = \alpha - \beta \cdot e^{-(||\mathbf{x_i}-\bar{\mathbf{x}}||^2)} \tag{3.2}$$

α, β are parameters which need to be chosen.

$\mathbf{x_i}$ is a member of the swarm.

$\bar{\mathbf{x}}$ is the center of the swarm.

Now the formula just needs to be added as an attraction and repulsion function to 2.3. But except for a swarm member $\mathbf{x_j}$ we have the center of the swarm $\bar{\mathbf{x}}$.

With this I am ready to have swarm behaviour for one swarm. Now if I have a second swarm I can also use this but there is still is no interaction between both different swarms.

As a possible interaction some kind of repulsion between both swarms comes to mind immediately.

So for this I decide to create a repulsion between a member of a swarm and the center of the second swarm. This is based on the unbounded repulsion mentioned in GASPARRI et al. (2012) and MOSTAGHIM (2022) .

$$f_i = \frac{\gamma}{(||\mathbf{x_i} - \bar{\mathbf{x_e}}||^2)} \tag{3.3}$$

γ is a parameter which needs to be chosen.

$\bar{\mathbf{x_e}}$ is the center of the enemy swarm.

To create an interaction I just need to once again calculate with this as the repulsion an 0 as the attraction in 2.3 . But except for a swarm member $\mathbf{x_j}$ we have the center of the enemy swarm $\bar{\mathbf{x_e}}$.

So if all these forces get added together I should have a successful swarm behaviour. The only thing left to do is to insert values for my parameters and to test it.

4

Experiments and Evaluation

4.1 Arena

To make experiments and gather useful data we have a arena on the ground with a 5m · 3m surface. In order to observe the arena we have installed a pinhole camera in the ceiling. This camera is connected with a computer so that it is possible to evaluate the video data. To get an accurate world frame we need to get an rectified image from the raw camera image. I choose to use the standard ROS package image proc to do just that.

Now I get an accurate image from the arena below which I can use for the detection. Additionally I needed to calculate the resolution to get accurate world coordinates.

$$resolution = \frac{realworldcoordinates}{pixelcoordinates} \tag{4.1}$$

As my Arena had a a 5m · 3m surface I calculated a resolution of 0,001696 .

For my path finding I implemented the navstack. I used a static map as my costmap. Though I changed it depending on which experiment I was doing. Other than that every single Sphero robot started its navstack otherwise it would not be possible to give out specific commands to them.

Now as I got closer to fulfill my prerequisites it was time to use an EKF.

For that I found two different implementations which both use ROS. The robot pose EKF and the robot localisation. Considering that implementing both of them would destroy the scope of my thesis I chose the robot pose EKF.

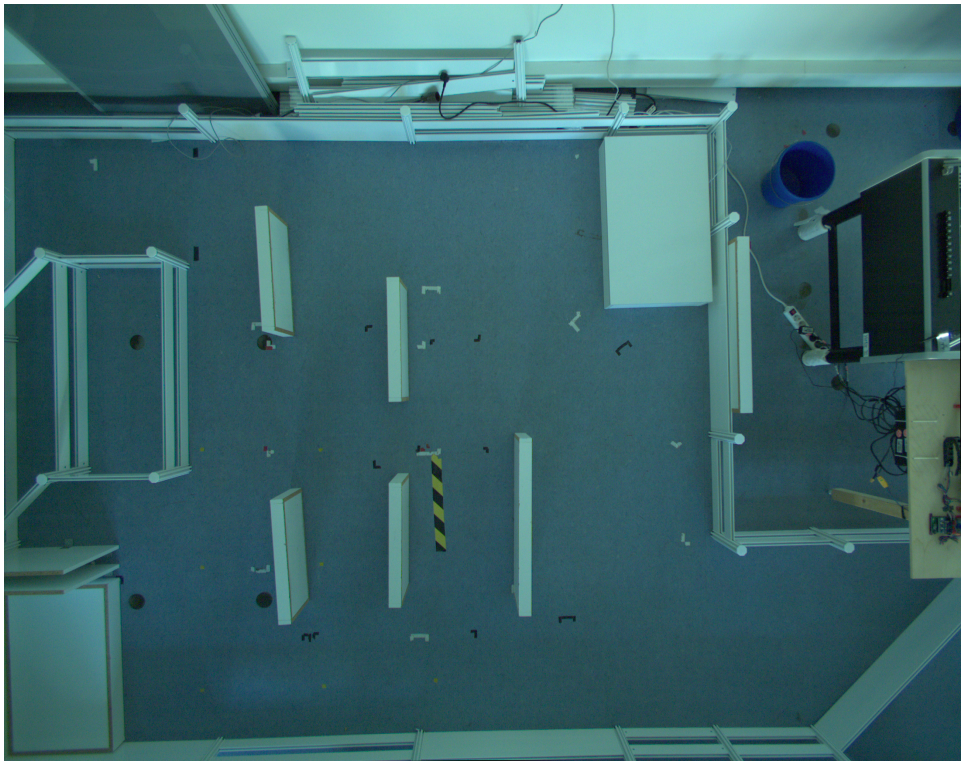

Figure 4.1: Arena pictured by the camera above

The robot pose EKF did not need any other non standard packages which made it easy to install. However this choice lead to another variety of problems. As mentioned I had to create a new variable to be able to change the system noise.

In order for the robot pose EKF to work I needed to give it an odometry, a visual odometry and some IMU data. This posed a problem for me. Even though the Sphero robot has an IMU integrated the current implementation of the Sphero driver is not able to correctly take it from the Sphero robot.

In addition to that the Bluetooth channels are under a heavy strain if you use more than one Sphero robot. Because of this I had to use four different Bluetooth dongles to get six different Sphero robots to reliably work without any major issues.

Even then I had to implemented an automatic reconnect should a Sphero robot lose their connection during their experiments.

So for that reason it only got the visual odometry and a virtual odometry based on the received command velocities from the navstack to work with. Theoretically this should be enough but after vigorous testing it became clear that it simply did not work.

The filtered poses I got from this implementation were simply wrong the moment my Sphero robots started to move. After three weeks of trying to get it to work I had to give stop in order to get everything else done.

Thus I could only use the direct tracking which tends to fail should the Sphero robots get too close to each other.

4.2 Creating Swarm Behaviour for one Swarm

Due to stability issues and problems with the correct tracking I decided to use three Sphero robots per swarm. I decided o use the following colors for my first swarm: magenta, dark blue and orange Now what is left is to use a costmap which describes my arena. 4.2 After fulfilling all the prerequisites I need, I can start with the main questions. Can I create swarm behaviour with my Sphero robots?To answer that question I created a simple experiment where I let the center of my swarm move between two Points in my map. Due to the created swarm behaviour my Sphero robots should strife

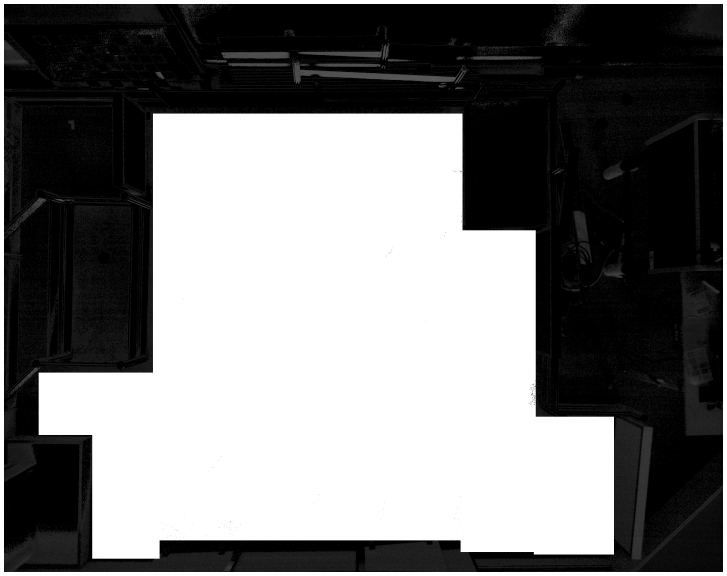

Figure 4.2: Costmap of my arena with no tight corridors

to stay around the center. Therefore I decided to use the following values for my parameters in formula 3.3. $\alpha = 1, \beta = 0.7$ To evaluate this I'm using three main criteria.

1. Cohesion Entropy of the swarm
2. Alignment Entropy of the swarm
3. Average distance to the center

To get an even better idea I also look at a still swarm and a swarm that was randomly placed without their behaviour enabled.

To get an average distance to the center I calculate the euclidean distance of every single member of my swarm to the center and divide by the number of my swarm members:

$$\bar{d}^t = \frac{1}{n}\sum_{i=0}^{n}(\left|\left|\mathbf{x_i}^t - \bar{\mathbf{x}}^t\right|\right|^2) \tag{4.2}$$

$\bar{d}^t$ describes the average distance of every member to the center of the swarm.

$\mathbf{x_i}^t$ is a member of my Sphero robot swarm over time t.

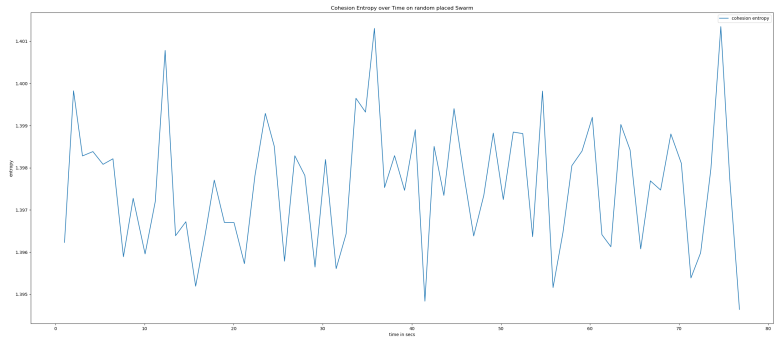

Figure 4.3: Cohesion one swarm randomly placed

$\bar{\mathbf{x}}^t$ is the center of my Sphero robot swarm over time t.

To calculate the entropy for a swarm I use this formula:

$$Entropy = \sum_{i=0}^{n} -\frac{1}{i} \cdot \log\left(\frac{1}{i}\right) \tag{4.3}$$

Now if we account for three swarm members we get a maximum and optimal entropy of 1.5849

So with the best Entropy in mind it is clear after looking at Figure 4.3 and 4.4 that a randomly placed swarm, does not get close to the optimum of 1.5849.

It is as expected quite a bit lower in the cohesion with an average of 1.398 and in alignment with 1.534. In addition to that the average distance is about 0.698 meters as we can see in Figure 4.5 .

We can already see some slight deviance in the results over time. This is a problem with the detection of the Sphero robots, which is a problem that follows us through all the collected data.

Even though the Sphero robots are not moving ,their position is slightly different every time. But when inspecting the average distance it is apparent that the difference is not decisive. These results can be used as an example of failed swarm behaviour.

In comparison to that if we look at the Figures 4.6 and 4.7 which is the Cohesion and Alignment entropy of a swarm that is not moving and it's swarm behaviour enabled we can see a clear difference.

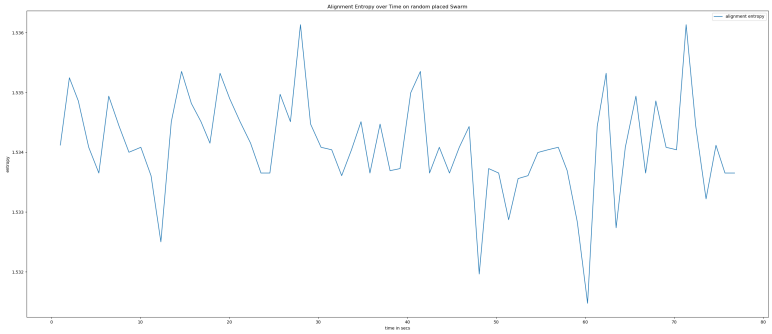

Figure 4.4: Alignment one swarm randomly placed

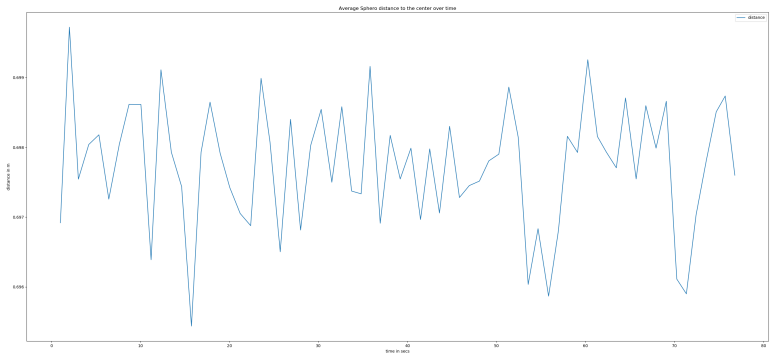

Figure 4.5: Average distance to the middle, one swarm randomly placed

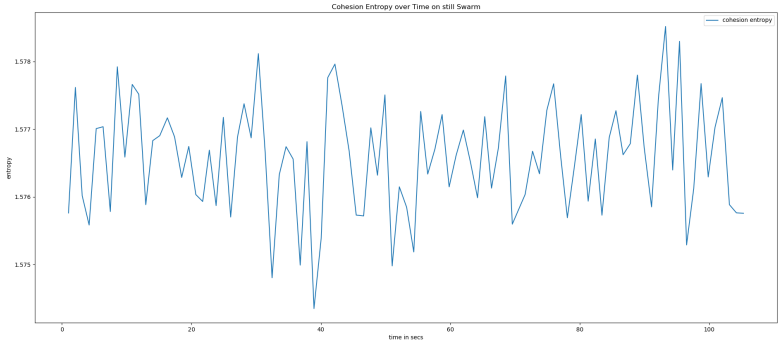

Figure 4.6: Cohesion one swarm, not moving

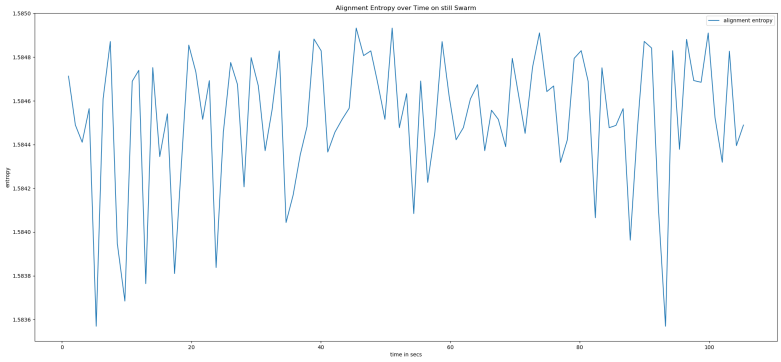

Figure 4.7: Alignment one swarm, not moving

Here there is an average cohesion entropy of about 1.5765 and an average alignment entropy of 1.5843. These results clearly show an optimal state of our swarm.

Each member of the swarm has reached an equilibrium at an average distance of 0.222 meters as per Figure 4.8 with my chosen parameters.

With both of these extreme cases in mind it is time to look at the actual data of my swarm.

Now we get to my actual experiment. Here I move the center of my swarm between two points in my map. As the swarm behaviour is enabled my Sphero robots should follow the center and try to regain their equilibrium. As to not overwhelm the bluetooth connection the Sphero robots only get a new goal to move to as long as it is:

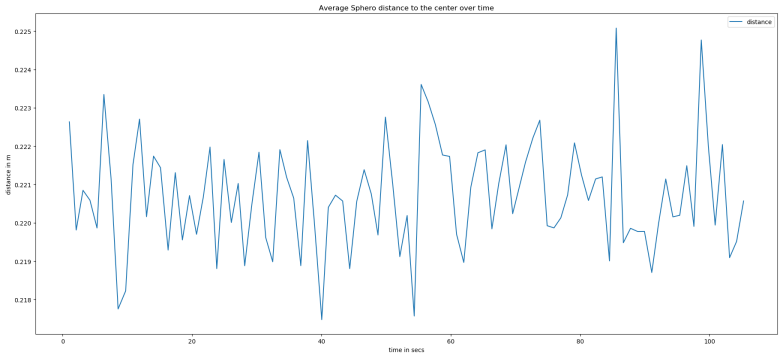

Figure 4.8: Average distance to the middle, one swarm, not moving

1. Inside the allowed area of the map
2. The new goal has a distance of at least 0.1 m
3. At least to seconds passed since the last goal

This allows the navstack to create a path for the Sphero robots to follow and get to the goal point. If I don't create these restrictions the Sphero robots will be completely overwhelmed with goal points. At this point the Sphero robots will simply stay still and not move no matter what.

So these restrictions are needed for anything to happen at all. Now as the center is moving between two predefined points I calculate my evaluation criteria.

As shown in Figure 4.9 it becomes clear that for a small moving swarm of three Sphero robots it stays relatively in the optimal range of cohesion entropy. In comparison to that the alignment entropy is quite unstable as shown in Figure 4.10 . It's ranging between 1.58 and 0.93.

This is easy to explain as every single member of the swarm is trying to individually follow the moving center and are thus breaking out of their equilibrium. Sadly the tracking is also slightly responsible for this as the detection of the orientation is not always accurate and tends to give us a false orientation every now and then.

The cohesion entropy has some low points with the lowest cohesion entropy at 1.05 at about the 58 seconds mark. If we look at the average distance in Figure 4.11, it becomes clear that it ranges between the optimum

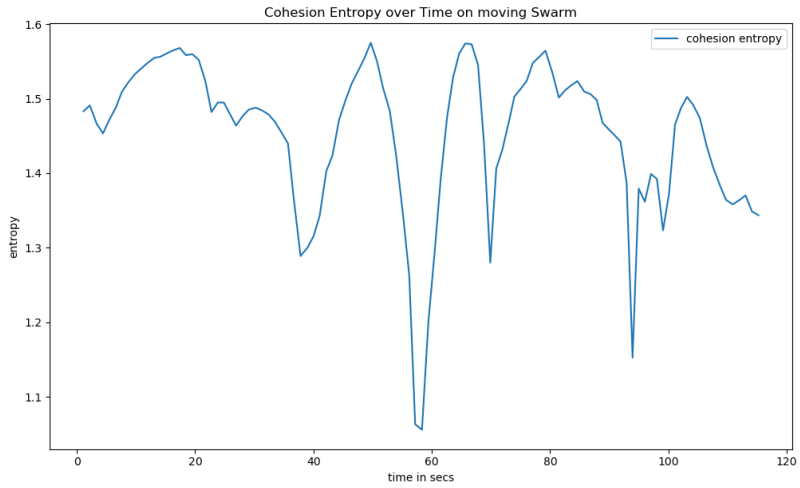

Figure 4.9: Cohesion one swarm no tight corridor

0.22 m and 0.97 m. These distances are created due to the amount of time it takes for a Sphero robot to get a decent route to drive to it's goal point. The center of the swarm is taking between 28 to 30 seconds to move between the two predefined points as shown in Figure 4.12 .

At the mentioned 58 second mark we have a low entropy but a low average distance. This means that not all members of the swarm are close to each other and are thus easier to pick out. This makes sense because they are moving individually to their goal points to get back in their optimal state. During this movement it tends to lower the entropy.

Now after looking at the data and the way the Sphero robots tried to achieve an equilibrium with the moving center I can say that for one swarm it is working. The entropy is staying in a good range with the optimal still swarm. There are of course some low entropy points which are even lower than the randomly placed swarm with 1.1 at the lowest but it never stayed in that low range and always returned to about 1.4 .

So I can answer my first question stated in 1 with yes!

4.3 Testing in a tight Corridor

After successfully answering the first question I arrive at the second.

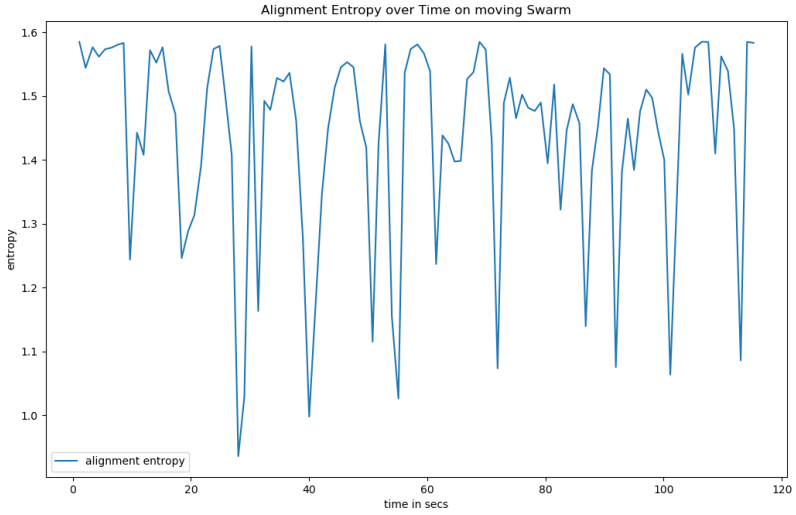

Figure 4.10: Alignment one swarm no tight corridor

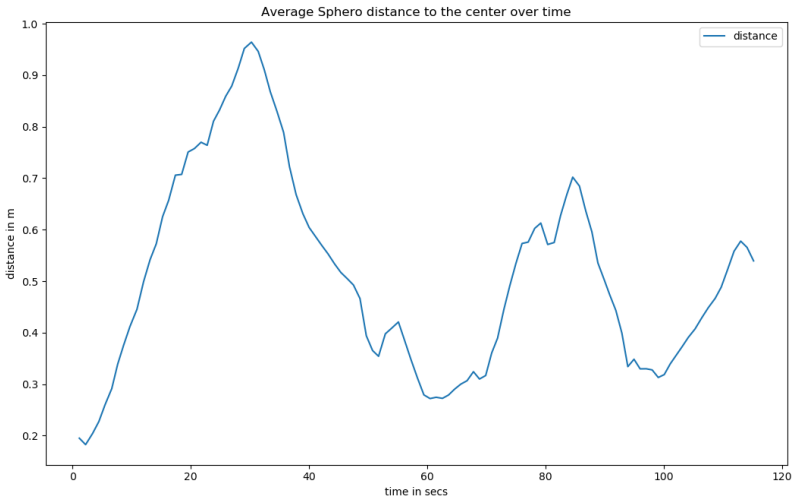

Figure 4.11: Average distance to the middle, one swarm no tight corridor

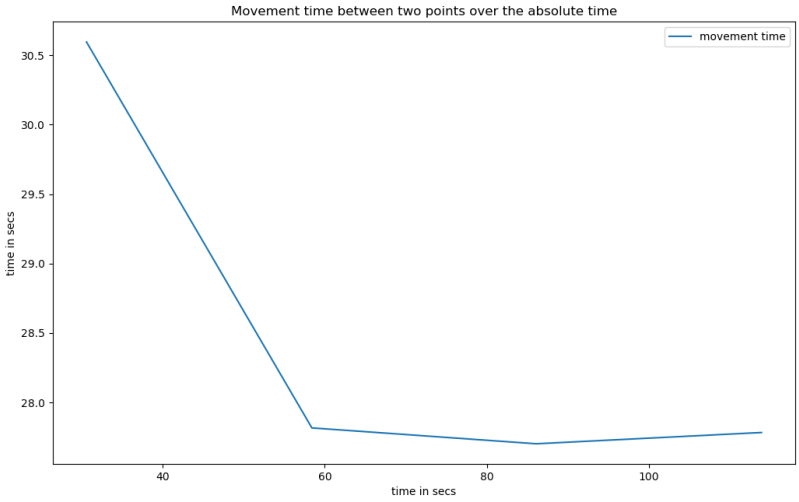

Figure 4.12: Average time for the center of the swarm to move between two fixed goal points with no tight corridor

Can I create swarm behaviour with my Sphero robots in a tight corridor?

To test that everything from my first question remains the same except for the costmap. In this experiment I still move my center between two points but now my map is considerably smaller as seen in 4.13. After letting everything run it becomes obvious that even though it is working there are some problems which were not as prevalent before.

Looking at the cohesion entropy in Figure 4.14 I can conclude that even though it is quite similar to Figure 4.9 the lowest cohesion entropy is at 0.87 . In addition to that the calculated alignment entropy Figure 4.15 does also have a lower point at 0.76 . It shows that the entropy is pretty similar to the experiment with no corridor just slightly worse.

Now if I look at the average distance as shown in Figure 4.15 it is showing a worse version of the average distance in the previous experiment. The highest average distance is at about 1.2 m which is 0.2 m higher than the same experiment without a small corridor. To get a better comparison the time it takes for the center of the swarm to move between the points is between 28 to 34 seconds Figure 4.17.

Figure 4.13: Costmap of my arena with tight corridors

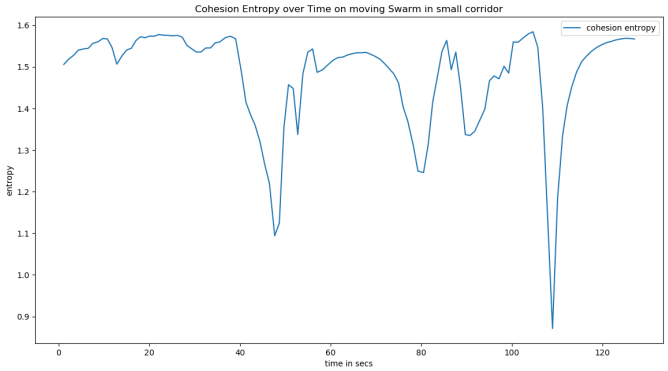

Figure 4.14: Cohesion one swarm in corridor

So why is everything just a little bit worse in a small space? These worse data points are relatively easy to explain. In a small space my Sphero robots are having a harder time to correctly move without hindering one another. In addition to that when they are getting to close to each other the tracking is starting to be unable to accurately track them. These inaccuracies leads to a longer time for my robots to calculate the best path to follow to its goal point. While this is happening the center of the swarm is continuing to move and thus we get a higher average distance.

Nevertheless the goal points my robots create are all in line with their swarm behaviour. And even though it is worse in a small corridor it still works somewhat. So I conclude this questions with a yes it works in a tight corridor even though it has problems.

4.4 Creating a second Swarm which interacts

Now I arrive at my third question: Is it possible to create two swarms which interact with one another?

In order to determine that I change my experiment a little bit. First I once again use the whole arena 4.2. Additionally I create a second swarm consisting of three Sphero robots and place them in the arena(insert arena picture?). I give them the following IDs: red, green and blue green. To get both swarms to interact I place them to the opposite of each other as well as parallel. Furthermore I give my parameter in formula (6) the following

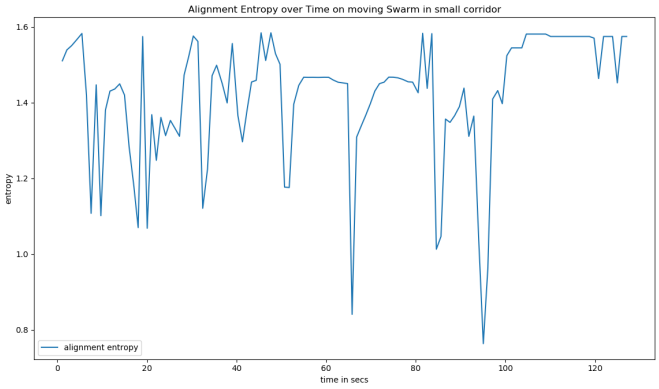

Figure 4.15: Alignment one swarm in corridor

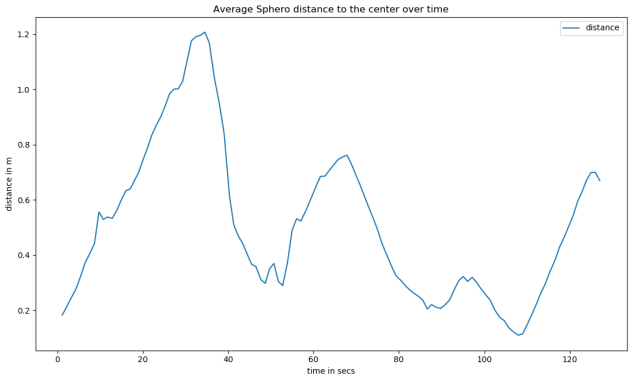

Figure 4.16: Average distance to the middle, one swarm in corridor

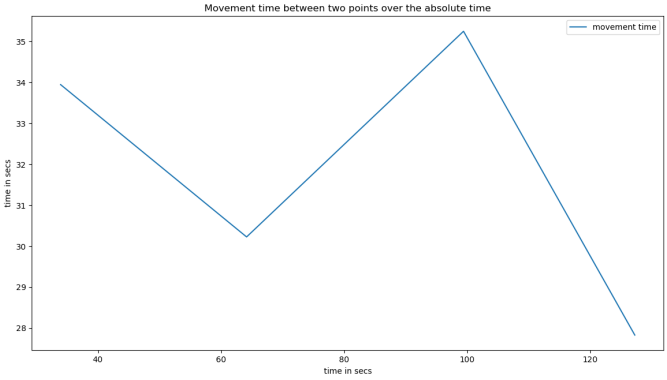

Figure 4.17: Average time for the center of the swarm to move between two fixed goal points with no tight corridor

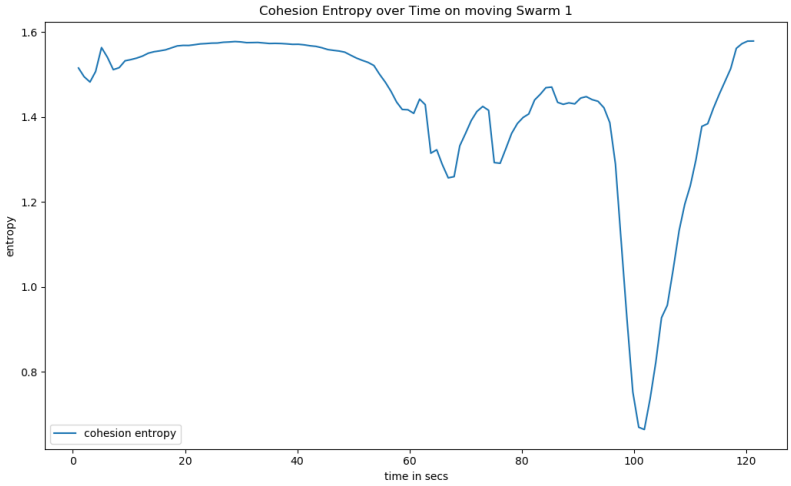

Figure 4.18: Cohesion two swarms, no corridor, swarm 1

value. $\gamma = 0.2$ In my first two experiments I let my swarm center move from one point to another. Now I do the same for the second swarm so that both swarms move by each other.

In addition to that I activate the unbounded repulsion for both swarms as described in my methods. With this I should have created two swarms that interact with each other. With all these preparations I start to calculate my evaluation criteria.

4.4.1 Swarm 1 Evaluation

Now as we look at the cohesion entropy in Figure 4.18 it stays quite high until later where it drops considerably to 0.73 . The same can be said about the alignment entropy in Figure 4.19 where it has the lowest drop to roughly 0.2 entropy. Additionally the average distance Figure 4.20 does not get over 1.2 which is slightly better than one swarm in a small corridor. As my earlier experiments have shown the time it takes for the center of the swarm to move between two points is between 28 to 32 seconds.

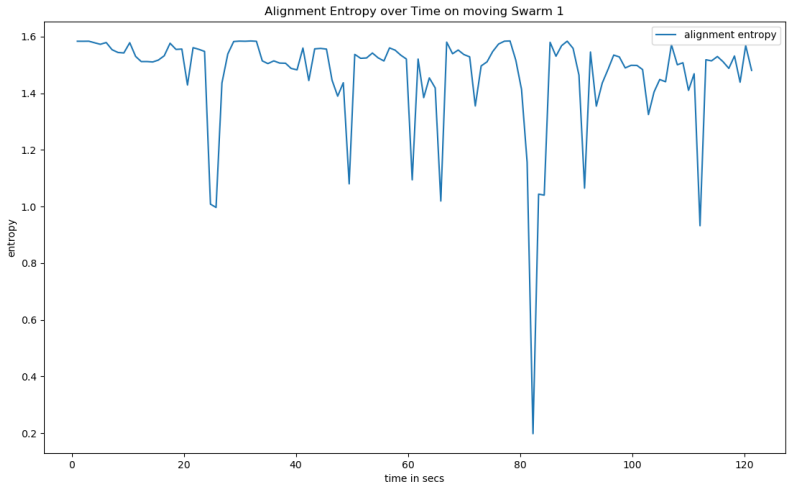

Figure 4.19: Alignment two swarms, no corridor, swarm 1

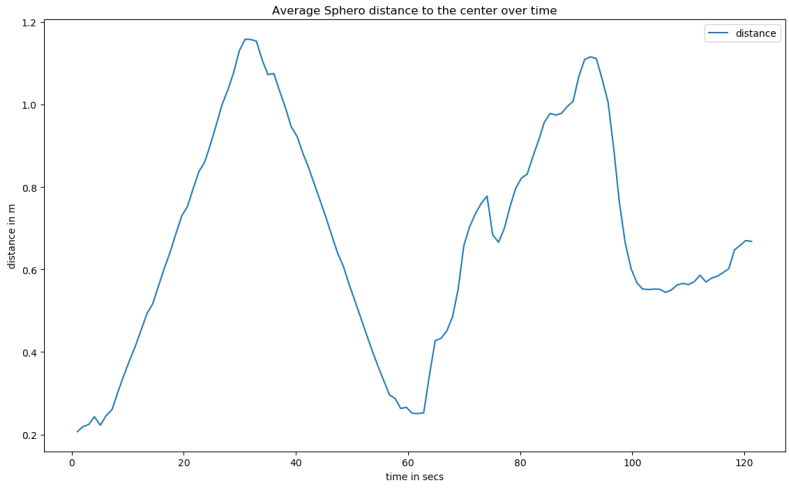

Figure 4.20: Average distance to the middle, two swarms no corridor, swarm 1

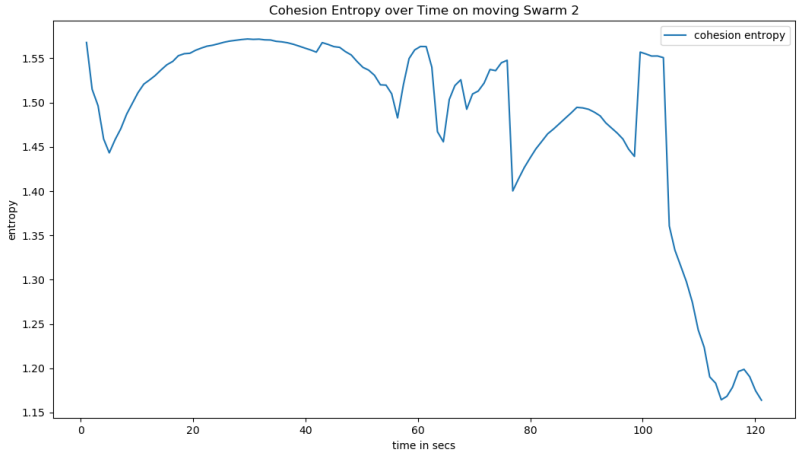

Figure 4.21: Cohesion two swarms, no corridor, swarm 2

4.4.2 Swarm 2 Evaluation

In comparison to that the cohesion entropy of my second swarm does not drop as low as my first swarm as it's lowest entropy is about 1.16 as seen in Figure 4.21 . However the alignment entropy Figure 4.22 shows a similar low point with 0.43 . Now we have a slightly higher average distance Figure 4.23 high point with 1.45 m which is the highest recorded data point yet.

4.4.3 Conclusion

With these data points I can say that it works only somewhat. Due to the enabled behaviour all Sphero robots are getting correct goal points however a few problems are starting to get apparent. With six Sphero robots in one arena the tracking is getting pushed to its limit. Each Sphero robot has its own unique color however the tracking starts to have problems to correctly identify the correct Sphero robot. This sometimes leads to a loss of location data which delays the movements to their correct goal points.

As the center of each swarm is moving between their goals the higher the average distance will be. This is why such a relatively high average distance of 1.4 m can be reached. Furthermore this also explains the strong low points in their entropy as not all Sphero robots manage to create a good path towards their goal in a similar amount of time. The actual local

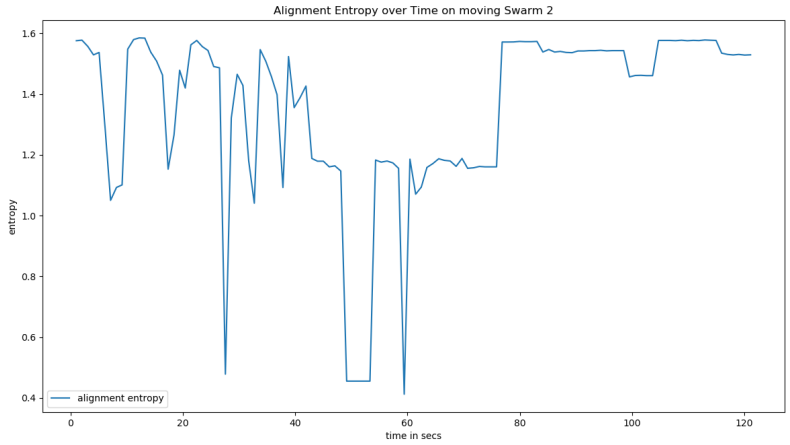

Figure 4.22: Alignment two swarms, no corridor, swarm 2

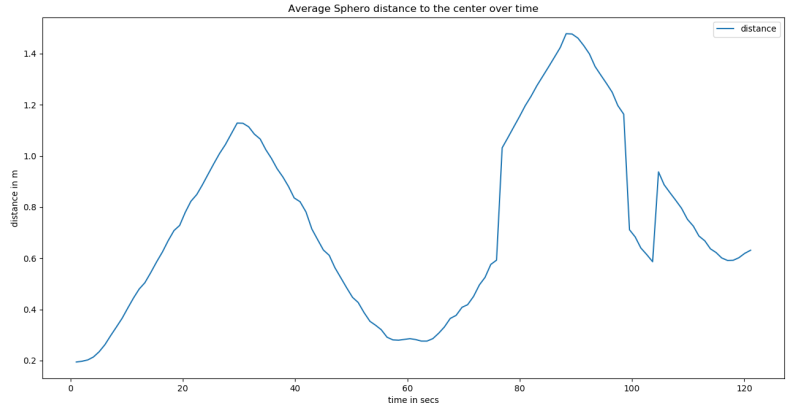

Figure 4.23: Average distance to the middle, two swarms ,no corridor, swarm 2

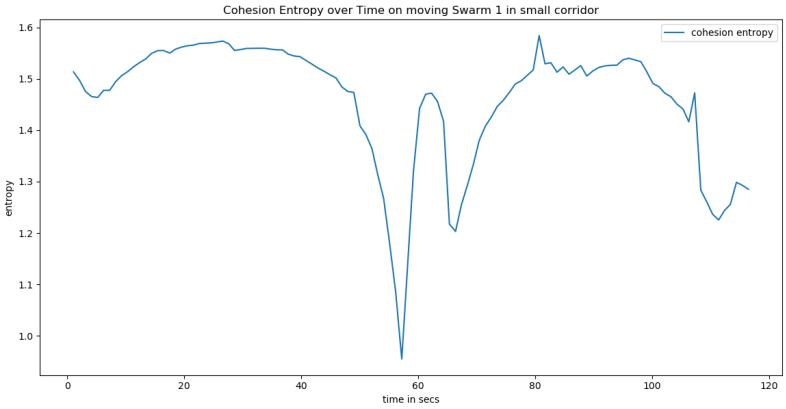

Figure 4.24: Cohesion two swarms, corridor, swarm 1

planner is also not designed with the Sphero robots in mind which makes the following of the global path not really possible.

Thus I come to the conclusion that yes it is working somewhat. However there are many problems which need to be addressed to make it work properly.

4.5 Testing in two tight Corridors

After the rough results of my last experiment I arrive at my last question: Is it possible to create two swarms which interact with one another where both are contained in a tight corridor?

To answer this I once again change my costmap 4.13 so that both swarms are contained in their small corridors. Everything else remains the same.

4.5.1 Swarm 1

As seen in Figure 4.24 the cohesion entropy stays relative close to 1.55 with only a few low drops to 0.9 at about 58 seconds.

Furthermore the alignment entropy 4.25 is ranging between the optimum 1.58 and the low point 0.8 .

The average distance as seen in Figure 4.26 is alternating between 0.2 m and 1.2 m which is roughly the expected distance.

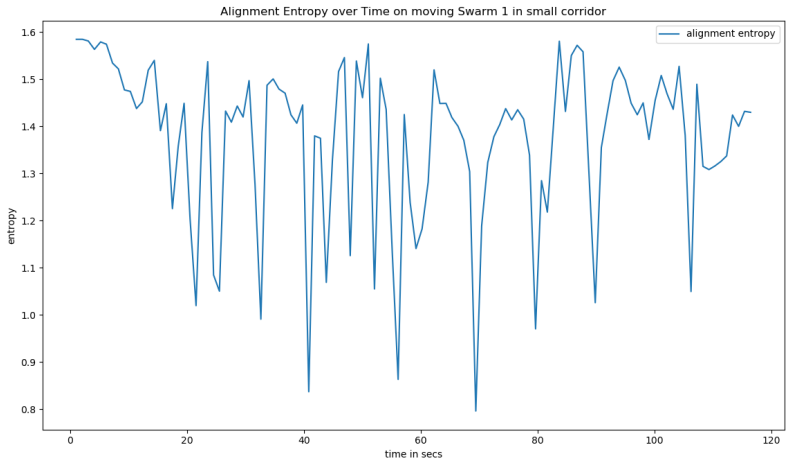

Figure 4.25: Alignment two swarms, corridor, swarm 1

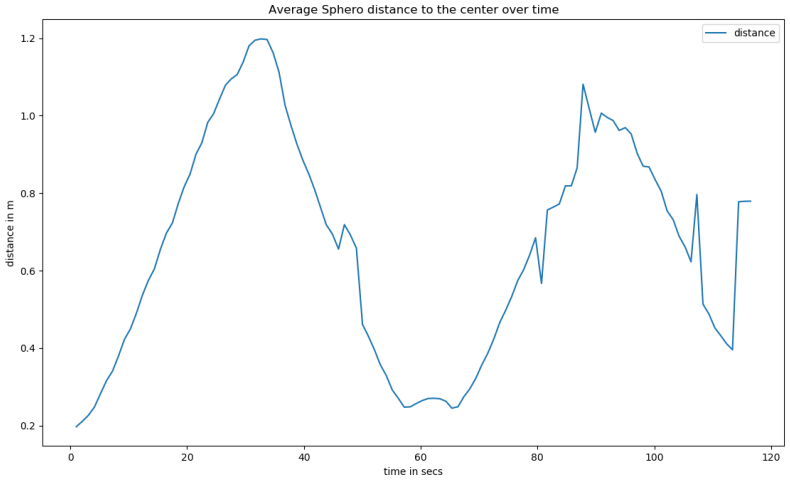

Figure 4.26: Average distance to the middle, two swarms no corridor, swarm 1

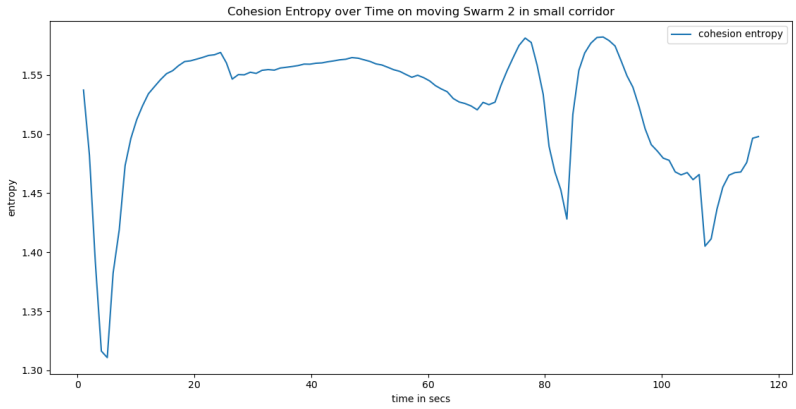

Figure 4.27: Cohesion two swarms, corridor, swarm 2

4.5.2 Swarm 2

Now for my second swarm we have a cohesion entropy that only drops to 1.32 at about 5 seconds in and other wise stays relatively near the optimum with only an occasional dip to 1.4 Figure 4.27.

In comparison to that the have the alignment entropy Figure 4.24 which as always has drops to the low point of 0.8 . However it is much less chaotic than every other dataset I have reviewed until now. Figure 4.28

Lastly If I look at my average distance I can see a relatively smooth line going from 0.2 to 1.2 and vice versa. Figure 4.29

4.5.3 Conclusion

If we only look at the entropy of both swarms it looks quite good however the relatively smooth line of more or less both average distances disputes that. As the Sphero robots start to move the entropy slightly dips. However as the different center of both swarms start to get closer and the unbounded repulsion starts to work, we got a problem as the Sphero robots start to stand still. This explains the high entropy despite the Sphero robots getting further away from their swarm center.

In addition to that the tracking continues to be a problem. Especially in a small corridor it sometimes cannot correctly localise the Sphero robots. This also leads to a movement stop.

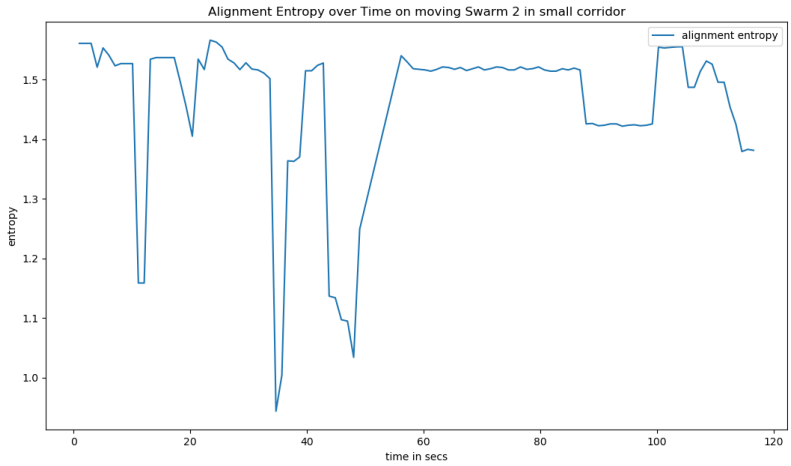

Figure 4.28: Alignment two swarms, corridor, swarm 2

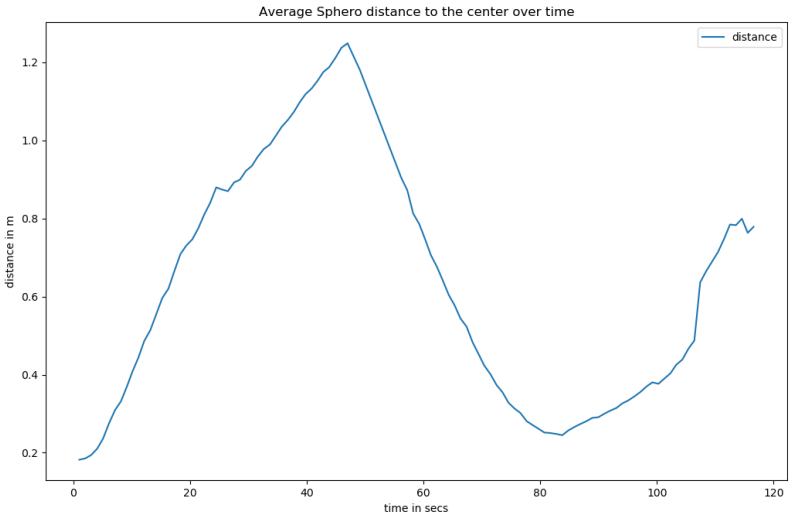

Figure 4.29: Average distance to the middle, two swarms , corridor, swarm 2

Thus I must answer that in my implementation it is not possible to have both swarms interacting with each other in a tight corridor even though the goal points of my Sphero robots are correct.

Lastly I have to mention that I did not do a stochastic analysis. I did not have enough time to do all the test for that as the experimentation with the Sphero robots is quite time consuming. There are two main factors playing into this which are the connectivity of the Sphero robots and their battery charge.

If a Sphero robot is moving it takes about 30 minutes before it has to recharge. The recharging process takes about 45-50 minutes before it has enough charge for proper experimentation. Additionally I only had nine working Sphero robots at any time.

Although it can be said that even though I did not do the all tests multiple times, each test was done in the same framework and gave a similar entropy which can be taken as a hint for a stochastic analysis.

5 Conclusions and Future Work

5.1 Conclusions

After all my experiments I can conclude that I managed to succeed for the most part. I managed to create a working swarm behaviour for one swarm of Sphero robots in our arena and somewhat in a small corridor. I only managed to achieve a somewhat satisfactory interaction between two swarms in an arena which ceased to work completely in my tight corridor experiment.

The main problems which lead to this result where mainly the localisation of my Sphero robots which ceased to work when they got to close to each other and the correct path finding in the local planner. Whilst my solution scales in theory it is a big problem with more then six Sphero robots at the same time due to tracking problems as well as an overwhelmed Bluetooth connection. Even though my finished tests do not include a stochastic analysis it is at least possible to see a hint of it due to the same framework giving similar results.

5.2 Future Work

For possible future work I have three specific things on my mind which would solve a lot of problems I experienced in my experiments. One possible avenue would be to create a local planner for the navstack which is specifically designed with the Sphero robot in mind. This would solve a lot of the path finding and movement problems. Another possible direction would be the the complete overhaul of the localisation.

Even though I created a training set and trained it accordingly it was not enough so it should be further refined to be better. Lastly an different implementation of an EKF filter would help. Be it the newer robot localisation node from ROS or a new implementation of an EKF filter it would help a lot with the localisation.

Further tests with an actual stochastic analysis should also be considered.

A

Abbreviations and Notations

Dataset and clustering acronyms

Acronym	Meaning
IMU	inertial measurement unit
ROS	Robot Operating System
navstack	Navigation Stack
tf	Transformations
EKF	Extended Kalman Filter
CNN	convolutional neural networks
DWA	Dynamic Window Approach

B

List of Figures

C

Bibliography

[FOOTE 2013] T. Foote. **tf: The transform library**. In: Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on, Open-Source Software workshop, 2013, pp. 1–6.

[GASPARRI et al. 2012] A. Gasparri, A. Priolo und G. Ulivi. **A swarm aggregation algorithm for multi-robot systems based on local interaction**. In: 2012 IEEE International Conference on Control Applications, 2012, pp. 1497–1502.

[GOHLKE] **Transformations**. `http://docs.ros.org/en/jade/api/tf/html/python/transformations.html`. Accessed on 23.05.2023, Version 20090418.

[GUIMARÃES et al. 2016] R. L. Guimarães, A. S. de Oliveira, J. A. Fabro, T. Becker und V. A. Brenner. ROS Navigation: Concepts and Tutorial, pp. 121–160. 2016. Springer International Publishing, Cham.

[HOYER et al. 2018] L. Hoyer, C. Steup und S. Mostaghim. **A Robot Localization Framework Using CNNs for Object Detection and Pose Estimation**. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), 2018, pp. 1388–1395.

[JOSEPH 2017] L. Joseph. **ROS Robotics Projects**. Packt Publishing, 2017.

[LARRIBE] **Wiki: Navigation**. `http://wiki.ros.org/navigation`. Accessed on 23.05.2023.

[MOSTAGHIM] **Swarm Intelligence Chapter 2 (part 1): Swarm Aggregation**. `https://www.is.ovgu.de/is_media/Teaching/Vorlesung+`

SwarmIntelligence/WS21_22/Chapter2_part1_Swarming-p-6830.pdf. Accessed: 2023–06-15.

[REYNOLDS 1987] C. W. Reynolds. **Flocks, Herds and Schools: A Distributed Behavioral Model**. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87, p. 25–34. 1987, Association for Computing Machinery, New York, NY, USA.

[RIBEIRO 2004] M. I. Ribeiro. **Kalman and Extended Kalman Filters: Concept, Derivation and Properties**. 2004.

[SPHERO] **Sphero Public SDK**. https://sdk.sphero.com/documentation. Accessed on 23.05.2023.

[VARAS] **Wiki: base local planner**. http://wiki.ros.org/base_local_planner?distro=noetic. Accessed on 23.05.2023.

Declaration of Academic Integrity

I hereby declare that I have written the present work myself and did not use any sources or tools other than the ones indicated.

Datum: .....

(Signature)