ENG-466 Distributed Intelligent Systems – Course Project

Nicolas Marbot, Christopher Stocker, Xavier Suermondt and Eliott Zemour

***Abstract*— In this report we explore some fundamental concepts of swarm robotics. The aim of the project is to implement a navigation strategy for a multi-robot system composed of simulated e-pucks on a series of different static and dynamic environments.The goal is to keep formation and flocking when presented with two clear challenges (i) obstacle avoidance of static objects, (ii) collision avoidance of other moving e-pucks. This goal is divided into the three following tasks: (A) Localization techniques for individual robots, (B) Spatial Coordination Solutions, and (C) Parameter Optimization. The quality of our solutions are evaluated on a set of metrics defined by the project outline.**

I. Introduction

Throughout the content of the course "*Distributed Intelligent Systems*", one got in touch with essential concepts of robotics and programming. At EPFL, engineers created the "e-puck", a small mobile robot whose goal is to understand distributed intelligent systems in a physical setting. This consists in having the decision making for a group of robots decentralized. This is noticeable in nature, as an example of the course was the ant species. The following document aims to better grasp this concept applied on e-pucks. Given the context, the experiments were conducted exclusively on *Webots*, a simulation software using the **C** language. Group of robots will have to maneuver in a closed environment filled with obstacles. The first part of this project will be to implement localization techniques. Then, the goal of the next section is to develop a collective movement based on communication between robots. Finally, the last part is dedicated to the optimization of the latter in order to enhance performance. The final result is to obtain group of robots that will be able to avoid obstacles and other group of robots while staying in formation.

II. Experiments

Two *Webots* worlds were provided for testing of flocking behaviour. The first world *test_obstacles.wbt* tests the obstacle avoidance behavior for different types of static obstacles. The flock is composed of five agents. Each agent shares the same migration urge such that it forces the flock to pass through the obstacles. The second world *test_crossing.wbt* tests the collision avoidance behavior of our flock in the presence of another flock. In contrast to the first world, this world contains only dynamic objects. The performance of our controller to both of these test worlds are defined by metric functions defined below.

A. Localization Techniques for Individual Robots

The e-pucks' localization was determined a priori using the robot's kinematics and odometry based on the wheel encoders.

1) Odometry estimation: The proprioceptive sensory data retrieved from the wheel encoders are influenced by noise. Accelerometer-based odometry introduced cumulative errors of uncertainties in measurements

For this reason our position odometry can accumulates errors over time. The accuracy metric for our localization was calculated by finding the error between the true position x^i_{true} and the estimated position $x^i_{estimated}$ using equation1.

$$e_{loc} = \sum_{i=1}^{N} \left\| \left(x^i_{\text{true}} - x^i_{\text{estimated}} \right) \right\| \tag{1}$$

The true position of the robot x^i_{true} was obtained by the supervisor.

2) GPS estimation: Global Positioning System (GPS) or more generally speaking Global Navigation Satellite Systems (GNSS) provide an absolute level of localization. They are however limited by their resolution and availability. GPS only showed an improvement but was subject to white noise. The GPS coordinates are updated at every **1s** interval. This navigation system uses dead reckoning in between time steps using the previous velocity as measured by the GPS and the time step to determine an approximation of the position.

3) Kalman: In order to simulate reality noise is added to both accelerometer and GPS readings. The accuracy of the position measurement can be improved using a sensor fusion algorithm. In this case the data from a noisy GPS and imperfect odometry can be combined using a Kalman filter to provide an improved estimation of the localization. This can be done under the assumption that the errors in measurement follow a probabilistic Gaussian distribution. The algorithm used to do this is shown below.

Algorithm 1: Kalman Filter

Variables

I: Identity Matrix — X : Position Vector
A : Prediction step — B_{enc} : Control Matrix
Cov : Covariance Matrix — U : State Variable
R : Proc. Noise Matrix — Q : Meas. Noise Matrix
C : Measurement Model

Function: kalman_compute_enc

$X_{new} = AX + B_{acc}U$
$Cov_{new} = ACovA^T + Rdt$
$K_t = Cov_{new}C^T + inv(CCov_{new}C^T + Q)$
$X_{new} = X_{new} + K(Z - CX_{new})$
$Cov_{new} = (I - KC)ACov_{new}$
$X = X_{new}$
$Cov = Cov_{new}$

A higher R (noise covariance) reduces the Kalman Gain K which filters more noise, however slows down the speed of

the filter. A more robust filter would take into consideration the changes in Q and R with respect to time since in this case Q and R are assumed to be constant.

B. Spatial Coordination Solutions

1) Flocking: This part aims to create a flocking behaviour of the epucks. To perform this collective movement model, one shall implement the Reynolds rules. There are three of them : **Separation**, **Alignment** and **Cohesion**. The robots are able to communicate via the Webots functions related to their emitter receiver. Indeed, it will allow them to make dynamic adjustments to avoid collisions, match their velocity and orientation with nearby flockmates and finally stay relatively close to them. The flock is given a migratory urge towards a specific location, that has to be decided in the following line of code :

```
float migr[2] = {x_coordinate,y_coordinate};
```

The communication is done by sending ping messages between robots (function send_ping()) which are then treated with the function process_received_ping_messages(). In the latter, the range and bearing between the robots are calculated in the following way:

```
message_direction =
    wb_receiver_get_emitter_direction(receiver);
message_rssi = wb_receiver_get_signal_strength(
    receiver);
double y = message_direction[2];
double x = message_direction[0];
theta = -atan2(y,x);
theta = theta + my_position[2]; // find the
    relative theta;
range = sqrt((1/message_rssi));
```

The distance between the pair is a function of the signal intensity *message_rssi* and the bearing can be determined with the measured directions of the leader *message_direction*. At each time step we evaluate the flocking metric defined below.

$$M_{fl}[t] = o[t] * dfl[t] * v[t] \tag{2}$$

The first parameter $o[t]$ calculates the orientation between robots.

$$o[t] = 1 - \frac{1}{N_{\text{pairs}}} \sum_{j=1}^{N_{\text{pairs}}} abs\left(H_{\text{diff},j}[t]\right)/\pi \tag{3}$$

Where $N_{\text{pairs}} = \frac{N(N-1)}{2}$ is the number of inter-robot pairs for N robots and $(H_{\text{diff},j}[t])$ measures the difference of heading between the inter-robot pairs.

The second parameter measures the distance between robots.

$$d_{fl}[t] = \left(1 + \frac{1}{N}\sum_{k=1}^{N} ||x_k[t] - \bar{x}[t]||\right)^{-1} \times \left(\frac{1}{N_{\text{pairs}}} \sum_{j=1}^{N_{\text{pairs}}} \min\left(\frac{\Delta xj}{D_{fl}}, \frac{1}{(1 - Dfl + \Delta xj)^2}\right)\right) \tag{4}$$

With x_k the position of robot k and $\bar{x}$ the center of flock, Δxj inter-robot distance of pair j and D_{fl} the targeted flocking distance.

The last parameter measures the velocity of the team towards the goal direction.

$$v[t] = \frac{\|\bar{x}[t] - \bar{x}[t-1]\|}{D_{\max}} \tag{5}$$

D_{max} is the maximal distance possible per timestep given the robots maximal speed v_{max}.

2) Formation: The algorithm in this part is the leader-follower one. The leader will behave to follow a specific trajectory. It will send a ping to the followers via the emitter device, and the followers will calculate their respective range and bearing with respect to the leader. Also, the ping message has for content the orientation of the robot. Indeed, it will be needed to process the information for the followers. Their goal range and bearing are the ones when the simulation is initialized. At initialization, they are aligned with the leader, two robots on its right and two others on its left. This configuration would be difficult to obtain in real life, as e-pucks can alter the signal quality by their proximity. Indeed, some of them will be on the way of the signals of others. This has to be taken into account if one wants to physically replicate this simulation.
The leader is given a migratory urge towards a specific location in the same way as in the flocking part. In addition to this urge, the Braitenberg algorithm will be used to allow the leader to avoid obstacles. In order to determine the performance of this part, we calculate the metric presented in the following expressions. $M_{fo}[t]$ is the metric at each time step, where $d_{fo}[t]$ is given in (7) which represents the position accuracy and $v[t]$ is given in (5) and represents the velocity accuracy. N is the number of robots, x_k the position of the robot with g_k its target position. $v[t]$ is the same as in the previous part about flocking.

$$M_{fo}[t] = d_{fo}[t] * v[t] \tag{6}$$

$$d_{fo}[t] = \left(1 + \frac{1}{N}\sum_{k=1}^{N} \|x_k - g_k\|\right)^{-1} \tag{7}$$

The process to treat the ping sent by the leader is an adapted version of send_ping() and process_received_ping_messages() functions from the flocking algorithm.

C. Optimization

The optimization was done using the meta-heuristic technique of particle swarm optimization. This process required several minutes of iterations. The best performance in *calcfitness* **0.376073**, and a best fit cluster of **0.011508**.

III. Results

A. Localization Techniques for Individual Robots

1) Odometry estimation: The accuracy of the estimated position varied in function of which parameter we accounted into our prediction. The crude model utilizes only odometry

to estimate its position. Odometry is sensitive to errors that arrive from the integration of velocity measurements to retrieve position estimates. The error is particularly relevant. The results show how odometric uncertainty grows along a straight or curved line, especially for the accelerometer odometry as it can be seen in Figure 1.The results are way more satisfying in the case of the wheel encoder in Figure 2.

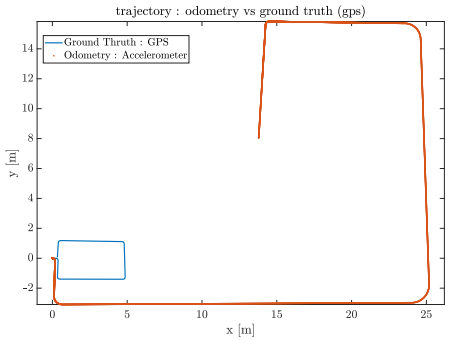

Fig. 1. Comparison between the accelerometer odometry trajectory and the GPS measurement.

2) GPS estimation: The GNSS enabled in the project operates at a frequency of 1 Hz. Between GPS recordings the uncertainty of the position increases and so does the error on this estimated value. Furthermore each GNSS are not free from noise and cannot directly provide the orientation, only position.

3) Kalman: The sensor fusion from odometry and GPS results in a more precise approximation of the robots position. The Kalman filter was compared to the ground truth obtained by the supervisor. This combination allows a significantly improved accuracy, as it can be seen in Figure 2.

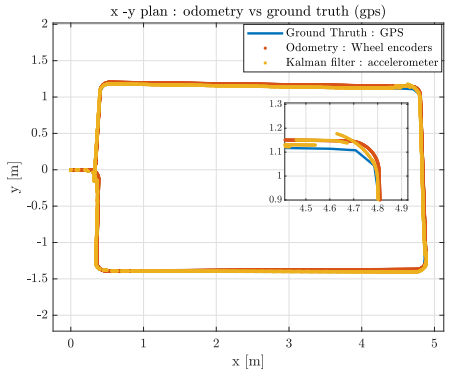

Fig. 2. Comparison between the accelerometer/Kalman and the wheel encoder odometries trajectories and the GPS measurement.

In Figure 3, we can observe the logarithmic evolution of the performance metrics for i) the wheel encoder ,ii) the accelerometer and iii) the Kalman/accelerometer odometries. The temporal means of these methods are respectively 0.0265, 28.753 and 0.0260. These results match the observation from Figures 1 and 2, where the accelerometer was very inaccurate due to error accumulation and the two other methods were giving similar and accurate results.

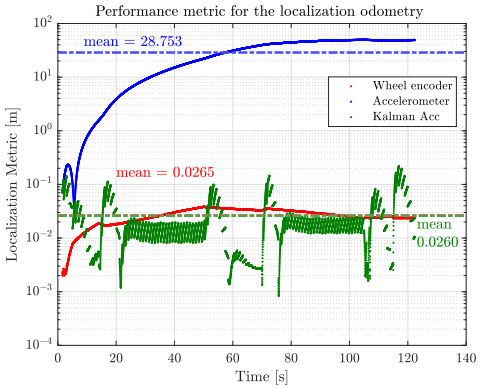

Fig. 3. Comparison of performance metrics for the three studied methods.

B. Spatial Coordination Solutions

1) Flocking: The Reynolds rules implemented ensure that the e-pucks are waiting for each other, keeping a similar heading and progressing fast through the environment towards the migratory urge. However, since the performance of the flocking algorithm is very sensitive to the associated weights and thresholds, trade-offs have to be found. We observed that prioritizing the migration towards a goal often leads to good velocity performance but poor cohesion. Concerning the performance measurement, the metric has been plotted as function of time in Figure 4. Its mean value is 0.0612, which is relatively low but the shape of the plot corresponds to the simulation. Indeed, we observe a few perturbation in the beginning(first small obstacles), then a big drop of performance due to the large obstacle that follows and so on. The total time of simulation went up to 5 minutes.

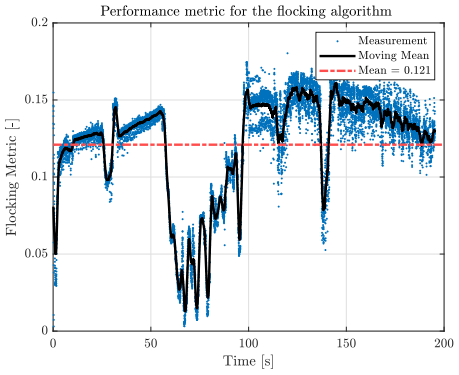

Fig. 4. Time evolution of the performance metric for the flocking algorithm in test_obstacles.wbt.

Secondly, we have tried our flocking algorithm on the crossing world. In Figure 5, we can see the evolution of the metric for both sides. Surprisingly, the left flock had issues to respect the Reynolds rules, and this can be seen on the metric, where the right side is more accurate than the left side.

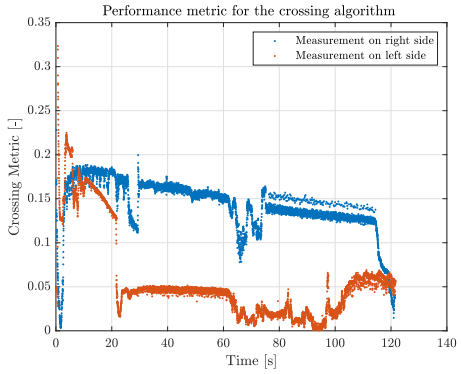

Fig. 5. Evolution of the metric for both sides in the crossing situation.

2) Formation: This test highlights the difficulty to make obstacle avoidance and formation control work together. Indeed, the simulations show that the swarm is conserved but not in the ideal alignment that could be expected. The formation takes approximately 1 minute to get out of the test_obstacles environment and some of the robots are blocked by obstacles. Even though the robots tend towards a mutual direction, the leader position, they are still scattered. As such, we can say that the algorithm can be optimised, as the observed behavior is approximate. Indeed, the errors certainly come from angles definitions. The overall metric has been calculated via the supervisor, with a value of 0.325 on average for the total simulation(50 seconds).

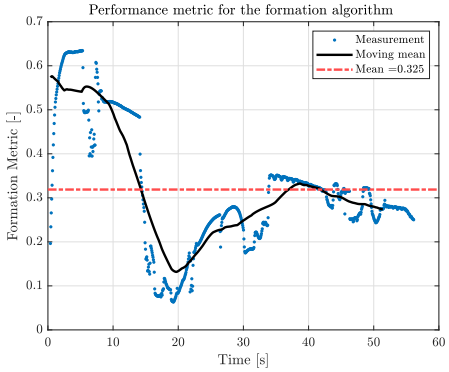

Fig. 6. Time evolution of the performance metric for the formation algorithm.

One of the drawbacks of this method is the fact that the leader and follower have a wheel speed maximum limit. Because of this, if followers have to avoid an obstacle while the leader keeps going forward at maximum speed, the followers will not be able to catch up the leader. Indeed, the leader is unaware of the position of its followers and will not slow down for them, in contrary of the flocking algorithm.

C. Optimization

1) Particle Swarm Optimization: The parameters of our controllers were optimized with Particle Swarm Optimization (PSO) algorithm. PSO is a promising meta-heuristic technique that can be used to optimize large number of parameters. For this reason we decided to use PSO to fine tune our flocking parameters and not our Braitenberg parameters for obstacle avoidance as we felt they were sufficiently robust. The flocking behavior uses seven parameters adapted from Reynold's weights. The custom world *new_test_obstacles* was designed to test the flocking parameters in the presence of obstacles. The simulation is run over 2'000 time steps to allow for the robots to cross the arena before restarting. In order to accelerate the learning process we increased the time step to 64 milliseconds. Furthermore, to reduce noise of lucky trials the the fitness of each particle is calculated as the averaged over multiple runs. We defined our fitness function our performance metric defined for flocking as described in equation 2.

This objective of the project was particularly challenging since the validation of the PSO structure needed many iterations. The optimization was also sensitive to chosen objective function and the respective parameters we wished to optimize.

IV. Conclusion

As a conclusion, this project allowed to better grasp the mechanisms behind collective intelligence robotics. The first part presents the advantages and drawbacks of the different localization possibilities and how they can be used together to optimize this process.
Secondly, the core of this project was studied in the following par : the collective movement algorithms. The first one is using the Reynolds rules and the second one was the leader-follower mechanism. The first one showed interesting results, in terms of obstacle avoidance and migratory urge. However, the crossing situation highlighted some issues of cohesion. Moreover, the process of findings the right weights for each rule can be laborious. Nonetheless, it still showed the most efficient results among the two methods. Concerning the leader-follower one, it presents to advantage to be easily scaled up, as the formation is decided depending on the initial position. Thus, it does not require further coding when we want to add a follower. However, the formation is far from being perfect as it needs precise data communication that is difficult to achieve in real life. The method presented here showed some problems with the angle management.
Finally, the goal of the last part is to find an Optimization process in order to determine the best coefficients possible.