Distributed Intelligent Systems, swarming algorithms*

Sarah Bonaly[1], Julien Calabro[2], Ismail Ben Salah[3] and Louis Munier[4]

***Abstract*— Since ages, social insects (e.g. ants) achieves complex tasks (e.g. nest construction, task allocation and decentralized control) despite having simple agents. Taking this as an inspiration, swarm robotics tries to achieve complex task, based the following principles : 1) Each robot is autonomous, 2) The behavior of the robot is based on local information (sensors and communication) 3) Having agents that are failing (shutdown, inconsistent information, isolation) should not compromise the complete group. The project goal was come with a strategy of multi-robot navigation in multiples environments and implement it. One way to ensure navigation and flocking of agents are the Reynolds rules. In this report, the selected strategy is described and its implementation on e-pucks. The test of the conceived strategy is done both on Webots and on real e-pucks. Performance of the system is monitored with multiples metrics**

I. Introduction

As part of the Distributed Intelligent Systems course, one of the main topics that has been discussed is collective flocking behavior. In order to have an applied understanding of these principles, it was asked to implement flocking behaviors on e-pucks (simulated in webots and also in the real world). To be able to assess how robust and functional was the selected strategy, two worlds were given : 1) *obstacle* having a flock of 5 e-puck and was composed with multiples static obstacles of different size and shapes. 2) *crossing* where two flocks of 5 e-pucks needed to be able to pass through each other,seeing the opponent flock as mobile obstacles. This report aims to present what was the selected strategy in the following terms : flocking global behavior, communication protocol between e-pucks, obstacle avoidance,the adaptation/modification needed inside the algorithm when executed in a real environment and the results obtained with. The conceived strategy is Reynolds-based with virtual force obstacle avoidance instead of Braitenberg obstacle avoidance.

II. Experiments

A. Communication strategies

The communication between the robots is one key aspect of the project. It allows the robots to get essential information about the surrounding environment to have a flocking behavior. Two main strategies are developed, one is used for the simulation with Webots and the other for the real experiment.

1) Communication process: The communication is based on infrared signals. A single ping can handle a number between 0 and 255 representing the message sended. From the intensity of the signal and the location of receiver sensor can extract important features as the distance and the position of the emitter in the receiver coordinate. The only information which is transmited through the message is the identity number of the robot emitter. In order to not saturate the communication canal, robots will send a message once every four cycles [1].

2) Communication in the simulation: For the simulation, the strategy is to allow the robots to communicate in a sequentially way. Each robots has a unique identity number between 0 and the total number of robots. Regarding this number, robots send messages or not. The robot number 1 send only if it received a message from the robot number 0, the 2 after the number 1, etc. In the simulation world, there is no robots out of range, they always receive massages wherever they are from the emitter even through obstacles or at high distances.

3) Communication in the real world: Because of stochastic uncertainties and lack of precision in different processes, the communication strategy had to be modified for the real experiments. As the range of the infra reds is limited, the idea is to use a time reference to determine whether or not a robot is out of range. For this purpose when a message is received, the corresponding time is stored in order to have access to the age of the message. This time is updated at each new received message. If the time of the last message is greater than an arbitrary timeout constant, the robot emitter is consider out of range and thus not belonging to the flock of the receiver anymore until a new message is received.

In order to allow the robots crossing in the second scenario, a unique identity number is given to each group of robots. As two groups should not merged and create a single one, when a message is received the group number is checked and this message is ignored if the group number of the emitter is the same as the one of the receiver. For both strategies robots localized themselves is the coordinate of the robot 0. This is done by considering the distances getting from the first received message as their position.

B. Flocking behavior

In both studied cases, the leader heading-based method was depreciated because of the presence of obstacles. If

*This work was not supported by any organization

[1]Sarah Bonaly, student with the departement of Robotics, EPFL, Switzerland, sarah.bonaly@epfl.ch

[2]Julien Calabro, student with the departement of Microtechnic, specialized in Robotics, EPFL, Switzerland, julien.calabro@epfl.ch

[3]Ismail Ben Salah, student with the departement of Computer Science, EPFL, Switzerland, mohammed-ismail.bensalah@epfl.ch

[4]Louis Munier, student with the departement of Microtechnic, specialized in Robotics, EPFL, Switzerland, louis.munier@epfl.ch

[1]A cycle here refers to one execution of the main loop

the leader cant communicate anymore with the rest of the flock which will loose its position referential and would be unable to swarm anymore. Thus, the Reynolds' Boids algorithm taken from [1] was implemented to perform the flocking behavior. This solution is mainly performing due to its accuracy and efficiency. Indeed, there is no need to follow a leader but in our situation a migratory urge is implemented to have a destination to reach. This allow to well simulate the flocking and tune it accordingly.

Reynolds' Boids is an efficient swarm algorithm with a computational complexity of $\mathcal{O}(n^2)$. It is mainly based on three basic rules, with different parameters (Weight and threshold) for each one of them, that should be tuned in order to obtain the best flocking behavior.

To implement it, the following rules are taken into account:

- Cohesion: match the speeds of flockmates.
- Separation: keep far enough from flockmates.
- Alignment : move towards the center of mass.

The unit-center-reference is used so each robot is referred to the center of mass of the flock using relative position. To do it, all the robots communicate their position between each of them as explained in the part II-A. In order to have a knowledge about the path traveled and the position where they have to go, odometry equations are implemented on each robot.

For this purpose encoders on the motors are used to get the number of steps counted during one cycle for each wheel. Thanks to these values the distances traveled by each wheel are computed, d_{left} and d_{right}, and then the variation of position and angle. The result is added to the previous position and angle values.

$$d\theta = \frac{d_{left} - d_{right}}{axe\_wheels} \qquad angle\ variation \qquad (1)$$

$$dx = Rsin(d\theta) \qquad x\ variation \qquad (2)$$

$$dy = R(1 - cos(d\theta)) \qquad z\ variation \qquad (3)$$

With R =curvature radius.

A custom weight is also applied on each of the Reynold's rules to obtain a relevant flocking with all the robots even if they encounter an obstacle and lose the visibility between each of them. Some metrics are implemented, to be able to assess the performance and get a feedback of which parameters combination leads to an enhancement of the flocking behavior. All these steps are made at first in simulation with Webots then adapt on the real e-pucks.

C. Obstacle avoidance strategies

In order to avoid obstacles, a dedicated algorithm is designed to do it and update the speed computed by Reynold's rules. A first approach was tested by applying Braitenberg to the speed computed.

1) Braitenberg obstacle avoidance: This algorithm was implemented because of its simplicity and intuitive method. It is possible to implement it with the Reynold's rules and it is very efficient due to its basic computing. Since there is eight sensors on the e-puck, a 16×1 matrix is coded to store the weights of each sensor. The principle is simple and this obstacle avoidance measure each sensor. When a measurement is bigger than a certain tuned threshold, it is add to the speed on the left or right wheel multiplied by its respective weight. It is compared to connect each sensor to each wheel with a factor to react to its environment.

This method was quickly aborted due to the number of factor to tune to have a good behavior of the flock. It is better used alone to avoid obstacle but is easily perturbed by the Reynold's rules. The fact is because of adding this weight on each wheel after the Reynold's computation it is subject to not well react with obstacles.

2) Virtual force: The second method explored is based on the paper of Ali E. Turgut on Self-organized flocking in mobile robot swarms [2]. In this paper they applied a virtual force on the robot due to the obstacles. This method is preferred because of our situation. By adding a second vector to the speed vector computed with the Reynold's rules it return a third vector which is the resulting speed given to the robot as shown on figure 1. In this generic example there is the pink "V" vector which is the speed given by the Reynold's rules, the blue "IRval" vector is the one given by sensing the obstacle and then the resulting speed is given by summing the speed with the inverse of the measurement vector. It is the brown one called V_{res}. It result of the implementation on the robot that is better than the Braitenberg method due to its ease to tune. It is also smarter than Braitenberg because of doing vector computation instead of summing weight factors to find. In the paper, this force is defined like the equation 4 where o_k is the detection level measured, o_{des} the level desired and C a scaling constant. This force is computed for each sensor and sum to have a resulting force on the robot. The angle given for each sensor is taken from the Webots e-pucks documentation and is show in table I where ps0 is the proximity sensor 0 and its angle is 1.27 rad. The zero angle is put on the sensor ps2 which is on the x axis.

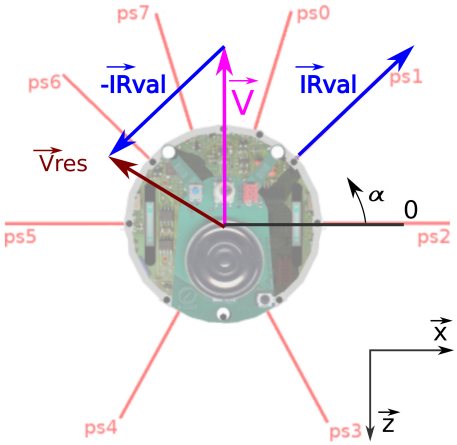

Fig. 1. Placement of proximity sensor and application of the virtual force avoidance

TABLE I

ANGULAR POSITION, IN RADIANS, OF ALL THE PROXIMITY SENSORS ON THE E-PUCK.

Sensor	ps0	ps1	ps2	ps3	ps4	ps5	ps6	ps7
Angle	1.27	0.77	0.0	5.21	4.21	3.15	2.37	1.87

$$f_k = \begin{cases} -\frac{(o_k - o_{des})^2}{C} & if o_k \geq o_{des} \\ \frac{(o_k - o_{des})^2}{C} & otherwise \end{cases} \quad (4)$$

The proximity sensor response against the distance given in the documentation of webots is shown in the figure 2. In our case the o_{des} variable is neglected and a simple threshold is implemented to know if there is an obstacle or not. Due to this and to the values returned by the infrared sensor it was decided to not implement the square of the detection value. This decision was taken to not saturate our obstacle avoidance controller. So the sum of the force applied to the robot is computed as follow where C is the number of the infra red sensors multiply by the maximum possible value of a sensor:

$$f_k = -\sum_{k}^{sensors} \frac{o_k}{C} \quad if o_k \geq threshold \quad (5)$$

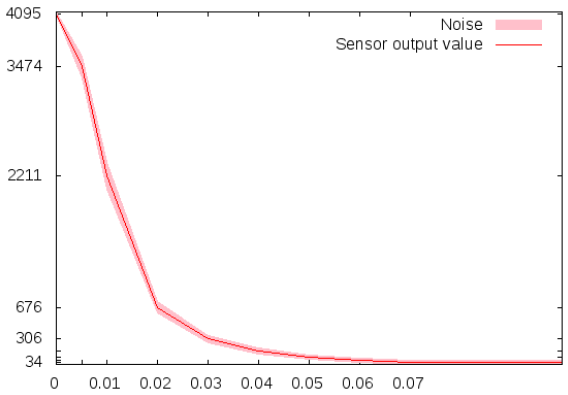

Fig. 2. Proximity sensor response against distance.

3) State machine: Since a force is applied to the robot it increase the value on the command send to the wheel in certain cases. To avoid a saturation of these values a state machine is implemented. With that state machine, the robot is able to differentiate where there is an obstacle to avoid or not and therefore adapt the weight of the Reynold's rules and the virtual force. Both coefficients are tuned by launching some simulation, print the value send to the wheels and avoid it to be bigger than the maximum speed value. The final vector speed translate in steps to send to the wheels is compute as follow:

4) Adding noise on sensor: If the sensors are good enough, there is no difference between two same measurement on two sensors. It results on a problem when the robot arrive right front an obstacle because the force could cancel the speed due to Reynold's rules. By having this reflexion, it was decided to add some noise on our measurements to avoid this situation. The noise is implemented like in algorithm 2 taken from the site [4] using the box muller method to have a random value on a normal distribution.

Algorithm 1: Add force to Reynold's speed vector

if $force_x$ *!= 0 or* $force_z$ *!= 0* **then**
  $x = K \cdot x - K\_F \cdot force_x$;
  $z = K \cdot z - K\_F \cdot force_z$;
else
  $x = K\_WITHOUT \cdot x$;
  $z = K\_WITHOUT \cdot z$;
end

Algorithm 2: Compute random numbers with box muller

Data: float m, float s // mean and sigma to compute random value
float x1, x2, w, y1;
static float y2;
static int use_last = 0;
if *use_last* **then**
  $x = K \cdot x - K\_F \cdot force_x$;
  $z = K \cdot z - K\_F \cdot force_z$;
else
  while $w \geq 1.0$ **do**
    $x1 = 2.0 \cdot (double)rand()/(double)RAND\_MAX - 1.0$;
    $x2 = 2.0 \cdot (double)rand()/(double)RAND\_MAX - 1.0$;
    $w = x1 \cdot x1 + x2 \cdot x2$;
  end
  $w = \sqrt{(-2.0 \cdot \log(w))/w}$;
  $y1 = x1 \cdot w$;
  $y2 = x2 \cdot w$;
  $use\_last = 1$;
end
return $m + y1 \cdot s$;

After computing this random value, it is multiplied by a constant $D = MAX\_SENS/2$ where MAX_SENS is the possible maximum value returned by a sensor then add to the measured value of the sensor.

D. Flocking metrics

To improve our complete algorithm a feedback on the simulation results is needed. This is why some metrics are implemented on the main parameters of the flock to have a good characterization of our flocking and improve it. All these metrics belongs 0 to 1 and are the following one:

1) Orientation: The orientation metric is the one who give the direction of each robot. This metric is maximized when all the robot have the same direction. In our case it is better when the robot goes to the migratory urge. It is compute as shown in the following equation:

$$o[t] = \frac{1}{N} \left| \sum_{k=1}^{N} e^{i\psi_k[t]} \right| \quad (6)$$

2) Cohesion: This metric compute the dispersion of the robot. In our situation it corresponds to the average distance

between each robot and their center of mass. It is closed to 1 when all the robot are closed to the center of mass and it is compute as follow:

$$c[t] = \left(1 + \frac{1}{N}\sum_{k=1}^{N} dist(x_k[t], \bar{x}_k[t])\right)^{-1} \quad (7)$$

3) Velocity: This one compute the average displacement velocity of the center of mass. It is maximal when all the robots goes with an high speed compared to their maximum one and with all the same.

$$v[t] = \frac{1}{v_{max}} max\left(proj_{\phi}(\bar{x}[t] - \bar{x}[t-1]), 0\right) \quad (8)$$

4) Performance: All these metrics reflect the overall performance only if all of them is taking into account in the same result. It is what is doing by computing the equation 9. This one give the performance of our flocking at an instant time t.

$$p[t] = o[t] \cdot c[t] \cdot v[t] \quad (9)$$

To have a feedback overall the simulation time, an average of all the performance metric is done.

$$\bar{p}[t] = \frac{1}{t}\sum_{k=1}^{t} p[t] \quad (10)$$

With all of them our system can be characterized and improved. The weight of all the parameters of the Reynold's rules and the weight of the obstacle avoidance compared to the Reynold's rules can be well tuned.

III. Results

A. Communication

For the real experimentation of the communication algorithm, robots act as they must be. Each robots receive well the messages emitted by the others with the right information. During experiments, a lot of messages are lost and thus are not transmitted. This has no influence on the performances of the communication as the amount of messages sent is big and subsequently the probability of a good transmission increase.

B. Odometry

The odometry algorithm has relatively good performances. The angular and translation errors are respectively 2 and 3 mm.

C. Simulation

The key and also the main difficulty of the simulation, is to tune the various parameters : thresholds and weights for the Reynolds'rules in order to obtain the best flocking behaviour. For each given scenario, two cases are considered depending on the distance to take an e-puck into a flock named 'MARGINAL_THRESHOLD' shown in the table II. Indeed, robots communicate with infrared sensors which have a maximum range of 22 cm. This value will be the real case and for the second one, which will be considered as the ideal case, a high range is taken, so that robots can always communicate. The first scenario will concern the one with one flock and obstacles and the second scenario the one with two flocks which must avoid each other. On the figure 3 the cohesion of the flock of the first scenario is between 0.84 and 0.92 which is a pretty good result. Indeed,it decreases when there is an obstacle because the communication is interrupted but observing the graph the flock reformed quickly. Similar observations can be done for the orientation on the figure 4. Indeed, the metrics is around 1 except when there is an obstacle that mean the flock goes in the correct direction defined by the migratory urge. On the figure 5 the graph shows the velocity of the flock which clearly decreases by getting closer to the migratory urge. Finally on the figure 6 the graph presents the flock performance in instant and the mean performance of the flock which also decrease by getting closer to the migratory urge because of the decline of the velocity. As for the first scenario the flock cohesion and orientation of the second scenario is pretty good in both cases although this time the cohesion decreases when the two flocks meet. However compared to the first scenario the velocity and so the performances of the flocks less decreases which means the velocity of the flock is more stable.

TABLE II
BEST PARAMETERS FOR THE BOTH SCENARIOS REAL CASE.

MARGINAL_THRESHOLD [m]	0.22
RULE1_THRESHOLD	0.05
RULE1_WEIGHT	0.9
RULE2_THRESHOLD	0.01
RULE2_WEIGHT	0.15/10
RULE3_WEIGHT	0.03/10
MIGRATION_WEIGHT	0.03/10
REYNOLDS_WEIGHT	80
OBSTACLE_WEIGHT	30 [second scenario] 40 [first scenario]

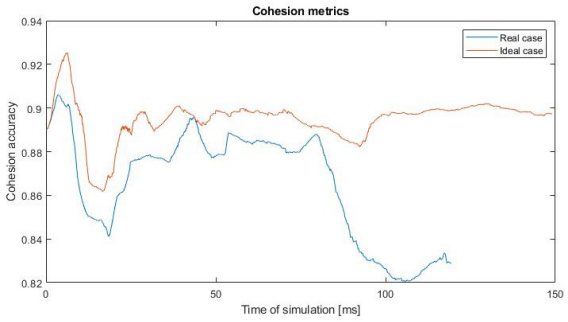

Fig. 3. Accuracy of the cohesion metric during the first simulation.

IV. Discussions

The best parameters obtained to perform as well as possible the desired flocking behavior were manually tuned that was not an optimal solution. Indeed, Reynolds'Boids algorithm gives an accurate and efficient flock behavior but

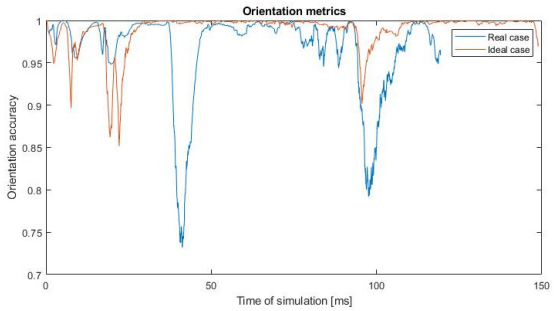

Fig. 4. Accuracy of the orientation metric during the first simulation.

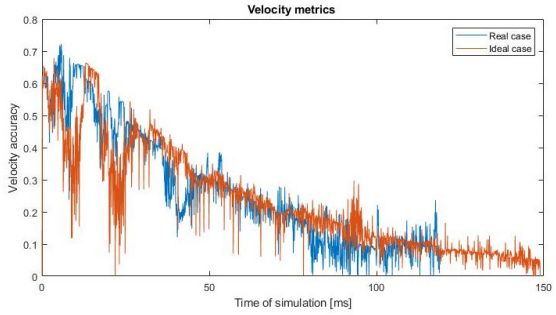

Fig. 5. Accuracy of the velocity metric during the first simulation.

it has lots of parameters to find. Particle swarm optimization algorithm (PSO) or a neural network would have been a better way to get the best parameters which thus give a better flocking behavior. Moreover, obtain quite good performances in the simulation will not necessary give the same behavior in the real experiments. Environmental parameters appears and should take into account because it will disturb the ideal performances obtained in the simulation.

V. Conclusions

To conclude, swarm algorithms permits to develop specific behaviors on real robots. Performances of those algorithm clearly depend on the number of parameters and on the way to get those that will give the best comportment. Simulation permits to improve and perform the developed strategy in order to find these parameters.Finally the time to tune parameters was an important factor in this project and differences between real world and simulation would have been take into account earlier.

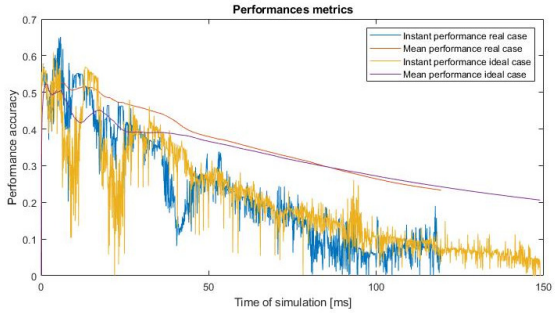

Fig. 6. Accuracy of the performance metric during the first simulation.

Acknowledgements

We would like to thank all the assistants for their advices during the overall project. The Pofr. Alcherio Martinoli is also thanked for this great experiment and to allow us to perform these algorithms on a real project to improve our knowledge. This project shows us all the difficulties to simulate and then adapt our code,which is something valuable.

References

[1] Alcherio Martinoli, Course of Distributed Intelligent Systems - W5: Collective Movements in Animal and Artificial Societies at EPFL, 2018 - 2019.

[2] Ali E. Turgut, Hande elikkanat, Fatih Gke and Erol Sahin, Self-organized flocking in mobile robot swarms © Springer Science + Business Media, LLC 2008.

[3] cyberbotics. (2018). Webots User Guide R2018b GCTronic' e-puck. [online] Available at: https://cyberbotics.com/doc/guide/epuck [Accessed 16 Dec. 2018].

[4] F. Carter Jr., D. (2018). Box Muller Code Examples. [online] Taygeta. Available at: https://www.taygeta.com/random/boxmuller.html [Accessed 16 Dec. 2018].