Multi-robot navigation in cluttered and dynamic environment

Martin Chatton, Valentin Gabeff, Diana Petrescu

***Abstract*— In this paper, we study multi-robot self-organisation and navigation in a cluttered and dynamic environment. We propose a non-linear aggregation algorithm adapted from *Turgut et al.*[1] using proximal control for obstacle avoidance and infrared communication for both heading alignment and swarm cohesion. Results showed that the algorithm provides a consistent flocking navigation, given good communication range and sensors stability, and can still show visual consistency on real e-pucks [2] that have a very short communication range and noisy sensors.**

I. INTRODUCTION

Flocking of robots finds many applications ranging from housework to mobile multi-sensor networks. Most of the algorithms implemented so far make use of Reynolds' observations [3]. They describe a natural flock upon three rules: cohesion that maintain the group members close to each others, separation to avoid flock collapse and alignment to match robots' speed and direction. The advantage of these natural rules over a graph-based model is that it for example allows for flock flexibility using local communication only. Although some previous studies showed navigation strategies for single flock to navigate in a simple environment [1][4], fewer researches have been done in a dynamic environment with very short local range communication.

In this study, two tasks are addressed to characterise the swarm navigation strategy performance. First, a group of robots needs to self-organise and navigate through an arena filled with static obstacles. Second, two swarms of robots start at opposing ends of an arena and must cross each others while avoiding collisions when moving in opposite directions. Infrared (IR) sensors are used for both range and bearing (R&B) communication and obstacle detection. The performance of the group depends on its cohesion, the robots velocities alignment and the speed of the mass center towards a defined direction, called the migration urge, over time. We use miniature e-puck mobile robots described by *Mondala et. al*[2] to implement the flocking algorithm on physical robots. Swarm performance is assessed on the physically-based simulation platform Webots [5] and the algorithm is subsequently uploaded on six real e-pucks. Calibration of the IR sensors is performed before each physical experiment. Parameters are optimized using the personal best based noise-resistant variation of Particle Swarm Optimisation algorithm (NR-PSO) [6][7].

II. EXPERIMENTS

A. Navigation Strategy

The navigation strategy implemented in this study is inspired by the approach described in *Turgut et al.*[1] and is adapted to E-puck features and components. The approach described in the original paper is to compute the desired heading vector of a robot as a weighted sum of two different components: a *heading alignment* vector h, that is the averaged heading of the robots in the flock, and a *proximal* vector p that ensures cohesion, separation and obstacle avoidance at the same time. The heading component is measured using a compass and broadcasting the heading angle to the nearby flock-mates. The proximal component is computed using a lookup table, matching a sensor value to a resulting force. Depending on the nature of the detected object (robot vs obstacle), a target detection level is used as reference, e.g. 0 for an obstacle because one wants to be far enough not to detect the object, and 3 for a robot. This means that one wants the robot to be at a desired distance from others, and therefore will be attracted if too far and repulsed if too close. We had to adapt this algorithm such that it matches our aims and limitations.

For the heading component, we have access to a compass node in Webots simulation, but the E-puck does not have such a module. To solve this, we use odometry for the E-pucks, i.e. the current heading direction is incremented at each iteration based on the initial orientation and the wheel speeds. If the robots are aligned at the start of the simulation, they can use the same frame of reference and they can interpret all angles received. We therefore can broadcast the orientations to each robot in the local range.

The adaptation for the proximal component is slightly more complicated. Indeed, our robots cannot distinguish the nature of the detected object, and hence cannot compute attraction, separation and obstacle avoidance within the same formula. To solve this issue, we split the vector into two distinct ones. A proximal vector p that is responsible for separation and obstacle avoidance that we calculated using proximal sensors, and a cohesion vector c to attract robots within a given circle, the distance and position of the robots being computed with help of R&B communication. We also implement a migration behaviour, that was not a point addressed in the paper. We choose to include this directly into the heading alignment vector.

In our final algorithm, the desired *heading vector* $a(x, y)$ of a robot at a time step is the weighted sum of a *heading alignment* component $h(x, y)$, a *proximal* component $p(x, y)$,that takes into account the values read by the proximal sensors and a *cohesion* component $c(x, y)$ that maintain the robots in a flocking formation, where x, y are axis of the E-puck centered relative coordinated system. We designed the aggregation formula for each time step t and for each robot as:

$$a(t) = \frac{\alpha * h(t) + \Gamma(t) * p(t) + \Delta(t) * c(t)}{\|\alpha * h(t) + \Gamma(t) * p(t) + \Delta(t) * c(t)\|} \quad (1)$$

$\Gamma(t)$ is the non-linear coefficient for the proximal component :

$$\Gamma(t) = \beta^{\iota + s(t)} \quad (2)$$

$s(t)$ is the maximum sensor value read during the time step normalized with the maximum possible sensor value. This coefficient will grow exponentially as the robot gets closer to an obstacle or another robot. It encodes the separation and obstacle avoidance rules.

$\Delta(t)$ is the non-linear coefficient for the cohesion component :

$$\Delta(t) = \begin{cases} \delta^{\zeta + d(t)} & \text{if } d(t) > D \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where D defines the desired distance to the center of mass, 10 cm for the simulations and 7.5 cm for the real e-pucks experiment and $d(t)$ the distance to the center of mass computed as:

$$d(t) = \|c(t)\| \quad (4)$$

Similarly, when the robot gets outside a circle of radius D around the current center of mass of the flock, the coefficient grows exponentially with respect to the distance to the center of mass. However, cohesion direction $c(t)$ is not taken into account if the robot is within the desired distance area. This contributes to both separation and cohesion rules.

The heading direction vector for each robot is defined as

$$h(t) = \frac{M_b * \theta(t) + (1 - M_b) * l(t)}{\|M_b * \theta(t) + (1 - M_b) * l(t)\|} \quad (5)$$

$$l(t) = \frac{1}{N+1} \sum_{j=1}^{N} e^{i\theta_j} + e_1 \quad (6)$$

where M_b is a migratory bias parameter that defines how much the robot should head toward the migration direction with respect to other robot's heading θ_j. θ_m is the migration orientation and e_1 is the unit vector in the current heading direction, namely (1,0) in the e-puck centered coordinated system used in our implementation. This helps the robot not to be stuck if an obstacle is in the heading direction.

The proximal direction vector is computed as:

$$p(t) = \frac{\sum_{k=1}^{8} f_k(t) e^{i\phi_k}}{\left\| \sum_{k=1}^{8} f_k(t) e^{i\phi_k} \right\|} \quad (7)$$

$$f_k(t) = \frac{-O_k(t)^2}{(O_{max} - O_{no\_detect})^2} \quad (8)$$

where ϕ_k is the angle of proximal sensor k on the robot, O_k is a value between 0 and 7 corresponding to a linear transformation between the maximum sensor value and a minimum sensor value threshold. The {Max, Min} detection sensor values pairs are defined as {500, 100} and {200, 10} for the simulated and physically real environments respectively, with $O_{no\_detect} = 0$ in both cases. Direction $p(t)$ is then weighted by the position of each sensors and the value read.

The cohesion direction vector is given by:

$$c(t) = \frac{1}{N} \sum_{j=1}^{N} x'_j \quad (9)$$

where x'_j is the relative position of each robot computed from the R&B module. α, β, δ are aggregation coefficients, ι and ζ bias coefficients for non-linearity.

At each time steps, robots transmit their corresponding id and angle θ_j so that any robot can compute the relative position and get the relative heading of any other robot within the communication range. Otherwise, relative position vector is set to 0 and orientation to the migration orientation so that old values are not taken into account in the computations, adjusting the normalization factors accordingly. A parameter $time\_out$ defines the time after which those values are not considered anymore. Only messages from members of the current flock are considered, ensuring group independence. This allows to have different migration orientations for each flocks.

If the forward velocity component were to be computed from the projection of $a(x, y)$ on the heading direction of the robot, and the rotational speed from its phase $\theta_a(t)$, the wheels speed will most of the time reach their maximum value. This is because a is normalized. Hence a robot will never head straight at a lower speed. This is a main issue as the robot will not wait for his teammates if the center of mass is behind it and will turn instead, compromising the flock direction. One improvement made is the addition of a coefficient $dot_c h(t)$ to the forward speed computation which is the scalar product between the heading and cohesion vectors and is set to 0 if the robot is within the desired distance to center of mass. Turning in this case is only needed if an obstacle is in front of the considered robot.

Forward speed $u(t)$ and angular velocity $\omega(t)$ are given by:

$$u(t) = \begin{cases} 2\pi R a_x(t) \sqrt{(1 - |dot_{ch}(t)|)} & a_x(t) > 0 \\ & dot_{ch}(t) < 0 \\ 2\pi R a_x(t) & a_x(t) > 0 \\ & dot_{ch}(t) > 0 \\ 2\pi R \left|a_x(t)\right| \sqrt{(1 - |dot_{ch}(t)|)} & a_x(t) < 0 \\ & dot_{ch}(t) < 0 \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

$$\omega(t) = \frac{\kappa * \theta_a(t) * 8W_r}{l} \quad (11)$$

where W_r is the wheel radius and l the axle distance. κ defines then how much the robots needs to turn given the direction of a. One can observe that the robot will never

move in backward direction is the forward speed component is always positive.

The right and left wheel speeds given in number of ticks per seconds are then given by:

$$motor_{left} = \frac{(u(t) - \frac{\omega * l}{2}) * motor_{max}}{2\pi W_r} \tag{12}$$

$$motor_{right} = \frac{(u(t) + \frac{\omega * l}{2}) * motor_{max}}{2\pi W_r} \tag{13}$$

where $motor_{max}$ is set to 1000 motor steps per second for the non calibrated experiment and to 200 otherwise.

B. Flock navigation performance metrics

Instant and overall performance are computed from the orientation $o'(t)$ of the robots within a flock, their cohesion $c'(t)$ and velocity $v'(t)$ along the migration urge direction.

$$o'(t) = \frac{1}{N}\left|\sum_{j=1}^{N} e^{i\theta_j(t)}\right| \tag{14}$$

$$c'(t) = \left(1 + \frac{1}{N}\sum_{j=1}^{N} dist(x_j(t), \bar{x}(t))\right)^{-1} \tag{15}$$

$$v'(t) = \frac{1}{v_{max}} max(proj_{\theta_m}(\bar{x}(t) - \bar{x}(t-1)), 0) \tag{16}$$

wherev_{max} is set to $6.28cm/s$ for both calibrated and non calibrated experiments, and $proj_{\theta_m}(\bar{x}(t) - \bar{x}(t-1)$ the projection of the speed vector of the flock center of mass along migration direction.

Instant performance in is given by:

$$p'(t) = o'(t)c'(t)v'(t) \tag{17}$$

and overall performance as:

$$\bar{p}(t) = \frac{1}{t}\sum_{\tau=1}^{t} p'(\tau) \tag{18}$$

C. Environments

Four environments are considered in this study. Both scenarios are tested on a different open environment. Environment 1 corresponds to the navigation of a unique flock through static obstacles. Obstacles are brown, the flock is composed of 5 e-pucks with a maximum speed of 6.28 cm/s and a radio ranged of 1.5m. Environment 2 is similar to the first one except it does not contain any obstacles and two flocks are set in opposite sides of the arena with opposite migration urges. Environment 3 is the calibrated version of environment 1, with only 3 e-pucks, a radio range of 15cm, maximum speed of 2.574 cm/s and white obstacles. Environment 4 is the Particle Swarm optimization environment described in the following section, it contains wall so that falling e-pucks do not compromise fitness function. No calibrated version of environment 2 has been performed as we could not assess this scenario with only three real E-pucks.

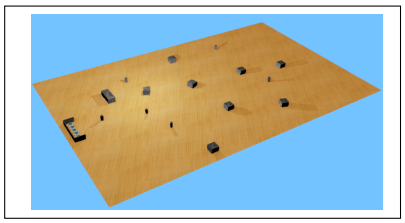

Fig. 1: Environment 1

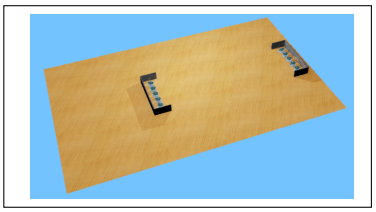

Fig. 2: Environment 2

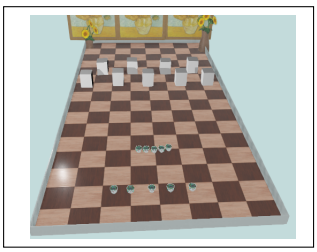

Fig. 3: Environment 4

D. Particle Swarm Optimization

In order to select the right parameters for our model, we use NR-PSO. Having a noise resistant version of the particle optimization algorithm is particularly suited for this case as we need our model to be robust to different environments and initial positions and orientations. At each iteration, the best particle found so far will then be evaluated again with different e-pucks initial positions. We optimize parameters $\alpha, \beta, \delta, \zeta, \iota, \kappa$ and M_b, yielding a 7-dimensional search space. As M_b must lie between 0 and 1, the update rule and initialization is adjusted accordingly. We use 7 particles in a public group homogeneous optimization strategy. Hence, each particle is initialized with a different set of weights, that is then transmitted to all robots from the flock and tested for 550 time steps and 700 for the calibrated version. The neighbourhood strategy used is index-based, where a particle is neighbour with the previous and following indices. Group fitness function is computed as:

$$F_p = 0.9\frac{1}{t}\sum_{\tau=1}^{t} f_p(t) + 0.1d_p \tag{19}$$

$$f_p = 0.3o'(t) + 0.4c'(t) + 0.3s'(t) \tag{20}$$

$$s'(t) = 1 - \frac{1}{ND}\sum_{j=1}^{N} s'_j(t) \tag{21}$$

$$s'_j(t) = \begin{cases} D - dist(x_j, \bar{x}) & \text{if } dist(x_j, \bar{x}) < D \\ 0 & \text{otherwise} \end{cases} \quad (22)$$

where d_p is the distance a flock travelled along the y axis. The PSO function slightly differs from the performance metric as separation is included in the fitness function and the traveled distance is used to include both obstacle avoidance and center of mass velocity performances. Robots can either start with completely random positions, or in a vertical or horizontal line. Each position is tested once and with 14 evaluations per PSO iterations, this ensure that particles are evaluated on a different environment every time. The environment is composed of a first line of 5 e-pucks representing the worst case when a flock needs to cross another one, followed by a section of static obstacles.

E. Transfer to physically real environment

Each sensor is first calibrated by computing the reference sensor value in an empty environment at the beginning of each experiment and subtracting it to all measured values during navigation. Our method does not update the look-up tables for sensors value with respect to distances as obstacle avoidance only plays a role when we get very close to an obstacle as given by equation (2), but the exact sensor value is not of importance as the value is normalized. We use 3 e-pucks for navigation on a bright environment with white obstacles. The maximum speed is reduced to 2.574 cm/s and each time step lasts for 100 ms, with a $time\_out$ parameter of 1 s. The maximal communication range has been measured to be approximately 15 cm by placing two robots at different lengths and orientations and taking into account the packet loss. Moreover, the IR communication library only allows to send one byte of information at a time. As our messages must include the id of the emitter, we have only a few remaining bits to communicate the orientation. We choose to limit the maximal number of robots to 8 (3 bits) to allow for better precision for the angles. We then discretize the angles into 32 possible values (11.25 degrees precision). This induces a maximum error of 5 degrees, which is a good enough approximation.

III. RESULTS

A. Odometry only based model

First we assessed model performance with 1.5m radio range and 5 simulated e-pucks. Results are shown on figure 4 for scenario 1 and 5 for flock crossing. We can observe that performance degrades every time an obstacle is encountered. Yet, instantaneous performance rises again right after obstacle avoidance. One would expect that performance degrades over time as we are dealing with odometry only. Yet this is not observed here, probably because simulation time was too short. As robots accumulate errors regarding their orientation, they are transmitted to other robots. Migration direction and heading are then compromised.

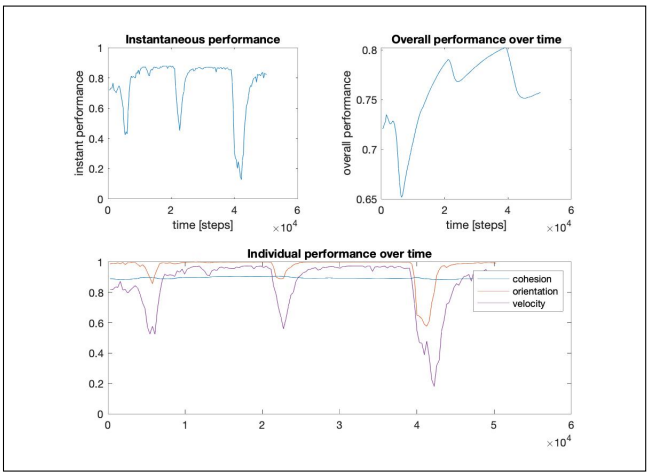

Fig. 4: Results of the Odometry-based controller on the obstacle world, handcrafted weights

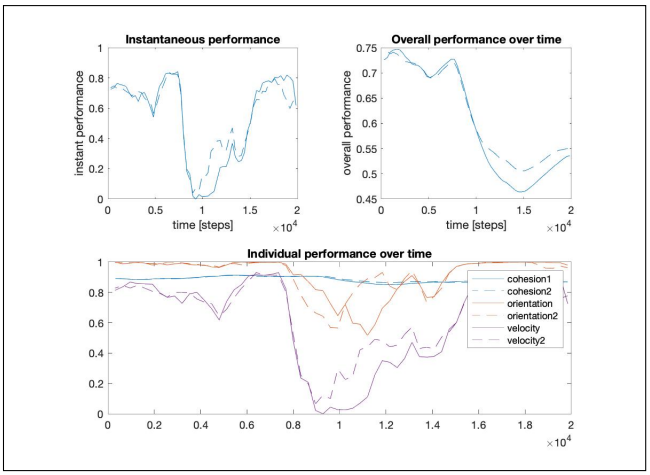

Fig. 5: Results of the Odometry-based controller on the crossing world, handcrafted weights

B. Addition of a compass node

To improve, our current model, we added a compass node to every robot. This has the advantage to start robots in any different initial orientations. Calibration of their orientation is then made with respect to the North at each time step. In this case, Particle Swarm optimization has been ran and resulting weights input to each controller. Results are displayed on figure 6 and 7. Although performance remains very high as long as no obstacle needs to be avoided, it is slightly worse than in the previous case. This can be due to randomness in simulations, but also because PSO has been performed in a world and is not fully suited for this one. This is further discussed in section III-D. This experiment can not be transferred to real hardware e-pucks for several reasons. Radio range for real e-pucks has been measured to be 15cm and is hence 10x smaller and robots do not possess a compass node.

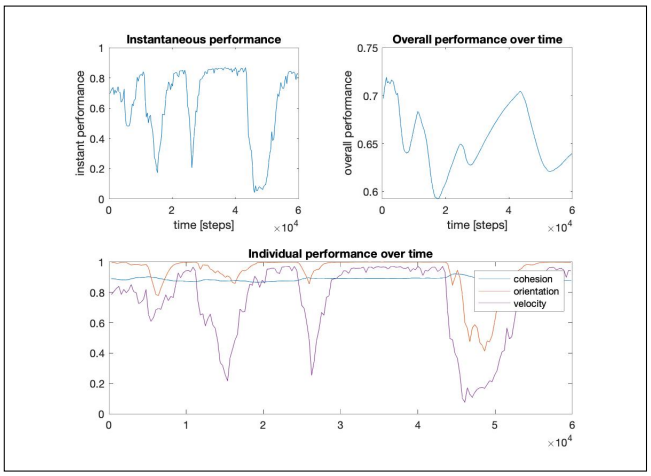

Fig. 6: Results of the compass-based controller on the obstacle world, PSO weights

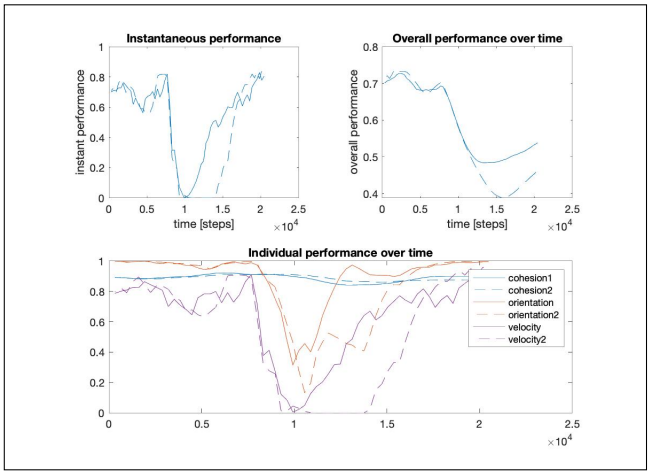

Fig. 7: Results of the compass-based controller on the crossing world, PSO weights

C. Calibrated experiments

The calibrated simulation aims to mimic the conditions of real experiments on E-pucks robots. For that purpose, we set the radio range to 15 cm, the maximum speed to 2.574 cm/s, the desired distance to the flock center of mass to 7.5 cm and the time step to 100 ms. Results are shown on 8. The weights are computed with PSO. We can observe that the score is on average five times lower than for previous experiments, due to the maximum speed that is set to one fifth of the previous max speed.

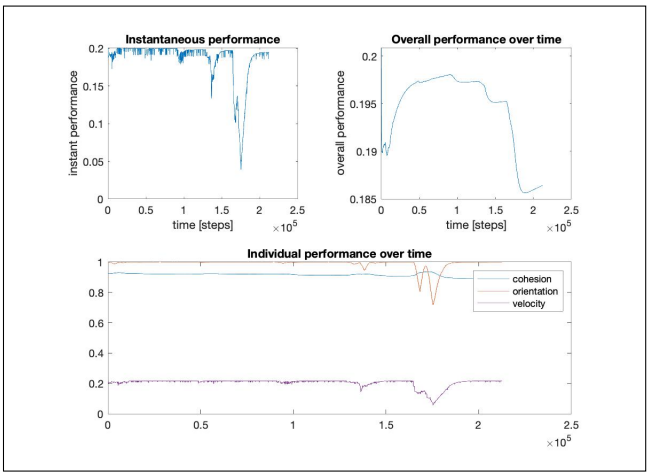

Fig. 8: Results of the calibrated simulation on the obstacle world

D. Optimization of model parameters

Figures 9 and 10 show results of NR-PSO after 25 iterations for normal and calibrated experiments respectively. In the calibrated experiment, the PSO environment 4 is adapted with the calibrated experiment parameters, considering only 3 robots and a first obstacle line of 2 robots only. We can see that during the first 15 iterations, parameters do not stabilize while they seem to reach an optimal value in the end. Yet, from a run to another, values are very different meaning that our search space contains many local minima. Moreover, when using PSO parameters, performance usually degrades fast. This is probably because the fitness function is not well suited for our application. Indeed it does not take into account sensors values and hence robots tend to have a strong heading component to travel as far as possible even though they might hit obstacles. Moreover, even though we limited initial value of M_b and κ to be smaller than one, inertia sometimes set those values to a higher value which has no physical meaning. Overall, this shows good convergence of the algorithm even though it did not improve our hand-crafted weight solutions.

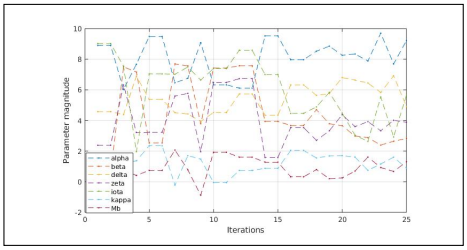

Fig. 9: Best parameters so far at each iteration

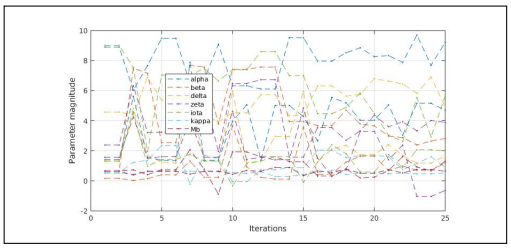

Fig. 10: Best parameters so far at each iterations for calibrated world

E. Visual assessment of physically real experiment

When transferring the PSO-learned weights to the E-puck experiments, the results were not as good as expected. The robots did not avoid obstacles correctly and the flocking behaviour was not apparent. This is probably explained by the low precision of the E-puck proximity sensors that affect obstacle avoidance, cohesion and heading alignment. By changing the weights, we were able to obtain a visually convincing flocking behaviour and correct obstacle avoidance.

IV. CONCLUSIONS

We have demonstrated a navigation strategy for flocks of e-pucks in cluttered and dynamic environments. The main advantage of this strategy is that it is easily scalable to other robots. The flocking parameters should be independent of the dimensions of the robot as those are explicitly used for speed control. Only dimension parameters and sensor angles will need to be updated. If needed, Particle swarm optimisation can be run for a new robots starting from the identified weights but for much less iterations. Moreover, the real-time sensors calibration make the method robust to different lighting conditions. Although the algorithm has a good simulation performance, many improvements can be made to improve its robustness. An amelioration could be to use a Finite State Machine (FSM) at the flock level. Depending on the situation, the flock will behave differently. Our current implementation tends towards this method as we have several conditions on how to update the forward velocity. One major limitation is the poor localisation of the second flock in the second scenario. Hence, flocks usually meet at the center instead of smoothly avoiding each others. A possible improvement would be to transmit an obstacle position as soon as it is detected to start flock avoidance sooner. As the current transmitter of e-pucks does not allow for sending complex messages, this was not implemented on the hardware devices.

Overall this study shows that to greatly improve the flock navigation in different kinds of environments, the strategy needs to be robust to many different situations. One must avoid to overfit on a specific scenario and room for explorations must kept. Noise resistant particle swarm optimization on various environment and more precise sensors should greatly improve this robustness.

REFERENCES

[1] Turgut AE, Çelikkanat H, Gökçe F, Şahin E. Self-organized flocking in mobile robot swarms. Swarm Intelligence. 2008;2(2):97-120.

[2] Mondada F, Bonani M, Raemy X, et al. The e-puck, a robot designed for education in engineering. Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions. 2009;1(1):59-65.

[3] Craig W. Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. In Proceedings of the 14th annual conference on Computer graphics and interactive techniques (SIGGRAPH '87), Maureen C. Stone (Ed.). ACM, New York, NY, USA, 25-34.

[4] Fredslund, Jakob & Mataric, Maja. (2002). A general algorithm for robot formations using local sensing: And minimal communication. IEEE Transactions on Robotics. 18. 837-846. 10.1109/TRA.2002.803458.

[5] Michel O., "Webots: Symbiosis Between Virtual and Real Mobile Robots". In Heuding J.-C., editor, Proc. of the First Int. Conf. on Virtual Worlds, Paris, France, July, 1998, Springer Verlag, pp. 254-263. See also http://www.cyberbotics.com/webots/.

[6] Pugh J, Martinoli A, Zhang Y. Particle swarm optimization for unsupervised robotic learning. 2005

[7] Di Mario EL, Navarro I, Martinoli A. A distributed noise-resistant particle swarm optimization algorithm for high-dimensional multi-robot learning. 2015.