Multi-robot Navigation in Cluttered and Dynamic Environments

Qiu Huajian[1], Vuk Pajovic[2], Darko Lukic[3]

Abstract— **The control of multi-robot navigation can be a challenge in real world. In this paper, we implement and optimise our strategy in two cluttered and dynamic environment. Finally, the algorithm is transferred into real e-puck robots and its performance justify the effectiveness of our design choice.**

I. INTRODUCTION

Mobile robots can replace static sensors to perform dense measurements in the environment, like localising odour source in the air or detecting bacteria in a lake. Those tasks can be performed with a single mobile robot or we can leverage by utilising multiple mobile robots to speed up the task execution [1], [2]. These environments are usually clattered and reliable communication is not feasible, and keeping a robot in a group is a challenge. Therefore, in the following text, we propose a solution based on Reynolds rules [3] and potential field [4].

A brief overview on theoretical background will be given in the section Experiments II where necessary concepts will be explained, such as Reynolds rules, potential field and PSO. Moreover, in the section our implementation will be carefully described such as communication between robots, flock avoidance strategy and limitation of real-robots.

After, in section Results III performance measurements for different scenarios will be given. Convergence of PSO will be explained as well as comparison of different algorithms. Finally, in Conclusions III a discussion on the results will be given.

II. EXPERIMENTS

In order to test control algorithms the following scenarios are used. One or more groups of robots are located in imperfect environment, interacting with each other to achieve the target while maintain the collective behaviour. The goal of the project is to implement multi-robot navigation in cluttered and dynamic environments. Two scenarios are presented, in each, our robots have to avoid obstacles within the arena while retaining the collective aggregation.

A. Reynolds Rules

According to Reynolds rule, the coordination inside flocking behaviours is based on three simple rule: cohesion, separation and alignment. There is no central control and each robots behave autonomously. It introduces robustness and wide adaptability to multi robots control in imperfect environment. While in other control algorithms, like leader-follower, the flocking performance highly depends on specific individuals. This will introduce high variance in the metrics evaluation and make optimisation of parameters harder. Therefore, to achieve precise localisation and robustness, we choose Reynolds rules to implement our controlling strategy.

B. Migration Urge

To make the robots move in the desired direction, we plan to implement the migratory urge. Since we don't have access to the global positioning system we have to rely on odometry. But, the odometry is prone to drift and we expect a slight deviation from the goal position. The migration urge is represented as a normalised vector inside the local coordinate system of each robot. The vector starts from the average position (centre) of group and ends at a same point inside the map. Since it is a normalised vector with length 1, the speed contribution magnitude of this term will not change in the progress. The apparent movements of robots will be the mixture of different actions stimulated by environment.

C. Potential Field

In order to avoid obstacles, certain obstacle avoidance algorithm had to be used. The two main candidates for this were using Braintenberg with the structure like neural network, and potential field algorithm for obstacle avoidance. Firstly, using Braitenberg where are many coefficients to be optimized. After all, with in mind to perform PSO, Braintenberg would need more coefficients, while with the implementation of the potential field, this should be reduced. Moreover, with the comparison to Braitenberg, potential field algorithm performed more naturally with flocking used and also since it has less coefficients, it was faster and more intuitive to calibrate it. Even though potential field works with obstacles which are circular, it is assumed that it is the case here even though in the Webots simulation, all of the obstacles were rounded. It didn't affect much the results, as it was seen later. The first parameter used was OBS RANGE, which is the range within the obstacles are effecting robots. If they are inside that range from the obstacle, then potential field is affecting robots, and thus the potential field vector should be calculated. Direction of the potential field vector is calculated to be normal to the shortest vector connecting robot and the obstacle. Given that each sensor has position, by interpolating between the strengths of each sensor, the final vector could be estimated. Because there were not exactly the positions of each sensor in the datasheets of epuck, this meant that that vectors are assumed as in the

[1]School of Basic Sciences, Mathematics, École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland huajian.qiu@epfl.ch

[2]School of Engineering, Robotics, École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland vuk.pajovic@epfl.ch

[3]School of Engineering, Robotics, École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland darko.lukic@epfl.ch

figure 1. All of the vectors are normilized. The final vector is estimated with the following equation:

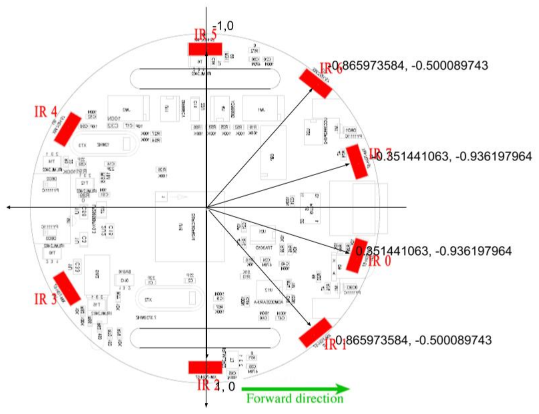

Fig. 1. Description of the sensor vectors

$$nx = \frac{1}{sum_{sens}} \sum_{1}^{NUM_{sens}} s_i * v_{xi} \tag{1}$$

$$ny = \frac{1}{sum_{sens}} \sum_{1}^{NUM_{sens}} s_i * v_{yi} \tag{2}$$

Where, s_i is sensor intensity and x_i and y_i are coordinates of each sensor. After estimation of the vector connecting obstacle and a robot, potential field vector has direction which is normal to this one. Magnitude of potential field vector is calculated as a value of sensor with maximal value subtracted by OBS RANGE and multiplied by the second parameter, which is OBS GAIN, or the gain of potential vector. Furthermore, with the sensor used, values which are provided are from the range of 50 until 4000. And when robot is away values are around 70, but when it is pretty close values are going high exponentially. Because of that, in order to have more fluent way of reacting to the obstacle, the logarithm is added, and final construction of vector looks like:

$$V = obsgain(log10(s_{max} - obsrange))\vec{V} \tag{3}$$

With the vector of potential field, simple P controller is implemented to make robot to follow the reference, which is in this case potential field vector.

D. *Flocking with potential field algorithm*

Given that the objective was to make robots going in a more natural way, mixing of two main algorithms had to be done in the same way. This prevented using state machine, because with empirical experience of it, sometimes it doesn't act natural. Because of that, in order to calculate the the final speed of robot, firstly the supposed speeds from two algorithms are both measured. After, the following principle is used for each wheel when there robot is inside the range of the obstacle.

$$v_i = \frac{log_{10}(s_{max} - OBS)}{log_{10}(s_{max})} p_i + (1 - (\frac{log_{10}(s_i - OBS)}{log_{10}(s_{max})))} f_i, \tag{4}$$

Where i stands for left or right wheel, p is output of potential field and f is output of flocking. If robot is not in the range of the obstacle, only f speeds are used on the wheels.

E. *Communication*

E-Puck utilises IR sensors to communicate with other robots. Since it uses a light there are multiple limitations:

- it requires line of sight,
- modulation method allows only very limited throughput,
- maximum distance is limited on around 30cm and
- the communication is generally unreliable.

The mentioned limitations of IR communication on E-Puck reflects the real-world communication issues between multiple robots. Therefore, we designed our algorithms to be robust with the unreliable communication.

Two controllers are designed, one which utilises short messages (1 byte per message) and other which uses long messages (6 bytes per message). Both controllers periodically broadcast messages and other E-Pucks receive them. Based on the message the robot can determine range, bearing and ID of the other robot which allows the robots to know each other relative position. This is minimal amount of information for Reynolds flocking.

1) Extended Messages: An alternative controller which uses extended messages is created. In addition to ID, the second controller also broadcasts absolute position of the robot (see Table I). It allows a group of robots to determine its absolute position more precisely, e.g. if a robot experience a huge drift it can use absolute position of the neighbouring robots to come back on the track. This is especially useful in the presented scenario where robots purely rely on odometry for localisation.

TABLE I
FORMAT OF A LONG MESSAGE

Name	Group ID	Robot ID	X Position	Y Position
Type (bytes)	Integer (1)	Integer (1)	Float (2)	Float (2)

This idea can be further extended to additionally improve the localisation by sending a confidence of determined position. The purpose of this controller is to be used as comparison, to show if the flocking algorithm can leverage of additional communication throughput.

F. *PSO Optimisation*

PSO (Particle Swarm Optimisation) is optimisation technique that creates a set of solutions as swarm of particles moving in virtual search space. The technique can be used to improve behaviour of a single or multiple robots by evaluating performances for different robot configurations [5]. In our problem we identified two potential places to utilise this technique, for improving obstacle avoidance and flocking. The obstacle avoidance algorithm is based on potential field and has only two parameters that have clear physical representation, therefore, it will not be covered in this project. In contrast, flocking uses 6 parameters and they are not easy to calibrate manually.

PSO can be utilised to optimise performance of a group or individual, solution can be public or private and team diversity can be homogeneous or heterogeneous. Since our results are measured by performance of group and robots should not be specialised, an optimal solution is to use group-public-homogeneous co-optimisation strategy.

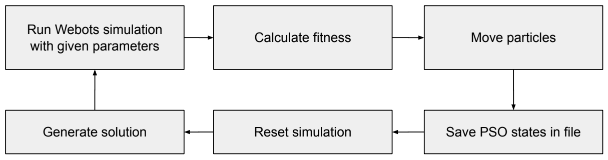

Fig. 2. Flocking optimisation using PSO in Webots

Webots has support of numerical optimisations [1] and it is used to evaluate solutions proposed by PSO (see Fig. 2). The performance evaluation starts by position robots in a world (see Fig. 3), PSO proposes parameters that are sent to the robots, robots use the parameters to move around, the flocking performance is measured and particles are moved accordingly. Then, PSO can propose another set of parameters and the cycle repeats.

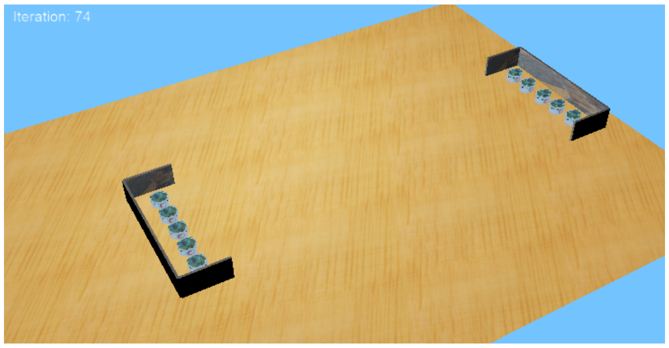

Fig. 3. Example of a world used performance evaluation in Webots

The purpose of PSO in our project is to find a set parameters for flocking that satisfies the following:

$$max\left(\frac{1}{N}\sum_{k=1}^{N} o[t]c[t]v[t]\right) \tag{5}$$

where $o[t]$ is orientation metric (see Eq. 7), $c[t]$ cohesion metric (see Eq. 8) and $v[t]$ velocity metric (see Eq. 9).

Generally, the robots operate in noisy environment (different conditions like obstacles, presents of other robots, unreliable communication etc) and for this purpose noise-resistant implementations of PSO (eg. evaluating parameters multiple times) can determine parameters that are better suited for real-world usage. Even though it is useful we didn't apply it because it would require a lot of time to be evaluated.

G. *"The devil is in the detail"*

During the implementation of all algorithms, there were always limitations when it comes to the real world. There is a problem in communication within the robots, which is not really robust and precise. Range and bearing from the IR sensors are not working behind the obstacles and also they depend on other robots orientations. Because of that, firstly when robots are turning really fast, processing of the messages are turned off. By doing it robot won't catch bad information and since it is turning fast only when it is avoiding the obstacle, it doesn't affect the algorithm. On the other hand, because communication is not enough robust, sometimes it happens that when robot is in front and it loses connection, it continues going forward without knowing for other robots. In this case, if it is in the front, it will never be in the proximity of others so it will lose them. To prevent that, there is mechanism, which is calculating when robot is in front of the flock and losing connection, while turning it off for 5 seconds, making possible for others to catch them and retain the communication.

1) Crossing Other Flocks: In dynamic environments it is very hard for robots to bypass obstacles because obstacles are not static. One such scenario is a flock robots that has to bypass another flock of robots. In this scenario we can exploit the fact they all can communicate to each other and optimise flocking. Therefore, two solutions are proposed. The first solution is based on a flock avoiding another flock which can defined as macroscopic dispersion (see Fig. 4) - individual robots are not directly aware they need to avoid other flock. The second solution is simple, whenever a robot notice a robot from a group of higher priority in its neighbourhood the robot stops. This allows robots to not disturb other flock during obstacle avoidance.

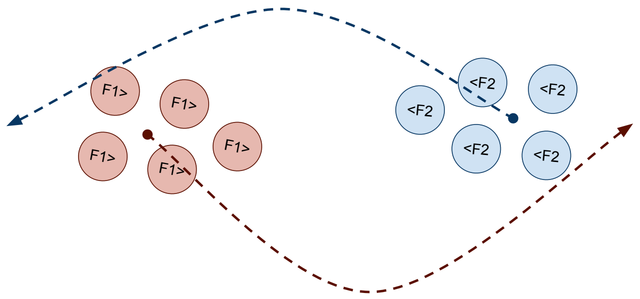

Fig. 4. Obstacle avoidance using dispersion on macroscopic level

III. RESULTS

In this section results obtained in Webots simulation, with real e-puck robots, and gap between real-world and the

[1]Webots documentation: Using numerical optimization methods (https://cyberbotics.com/doc/guide/using-numerical-optimization-methods)

simulation will be presented. General metric for comparison is given by the following expression:

$$\bar{p}[t] = \frac{1}{t}\sum_{k=1}^{N} o[t]c[t]v[t] \tag{6}$$

where $o[t]$, $c[t]$ and $v[t]$ are given by the Eq. 7, 8 and 9 respectively.

$$o[t] = \frac{1}{N}\left|\sum_{k=1}^{N} e^{i\psi_k[t]}\right| \tag{7}$$

$$c[t] = \left(1 + \frac{1}{N}\sum_{k=1}^{N} dist(x_k[t], \bar{x}_k[t])\right)^{-1} \tag{8}$$

$$v[t] = \frac{1}{v_{max}} max(proj_{\phi}(\bar{x}[t] - \bar{x}[t-1]), 0) \tag{9}$$

A. *Simulation*

In the table (see Tab. II) above typical performance measurements are given.

TABLE II
PERFORMANCE MEASUREMENT IN DIFFERENT SCENARIOS EVALUATED IN 900 ITERATIONS

Scenario	**Typical Fitness** $[10^{-4}]$
Short messages Potential field `obstacles.wbt`	4.53
Short messages Potential field `obstacles_tough.wbt`	2.14
Long messages Braitenberg `crossing.wbt`	2.84
Short messages Potential field `crossing.wbt`	3.8

B. *PSO*

As described, PSO is used to improve flocking performance and in this section PSO results will be elaborated for multiple scenarios.

1) Flocking among obstacles - `obstacles.wbt`*:* First, PSO is applied on robots that operate in world `obstacles.wbt` (see Fig. 7) and the fitness values are presented in the following plot (see Fig. 5) with configuration defined in Tab. III.

First, the flocking parameters are calibrated manually and slightly deviated parameters are used to create initial swarm. Even though, the parameters are initially calibrated PSO managed to improve further to obtain even better metrics. After 25×20 (data on the plot is aggregated, see Alg. 1) evaluations PSO hit the plateau and it is not able to provide any better result. For the further improvement flocking algorithm needs to be changed or PSO should be run with different configuration.

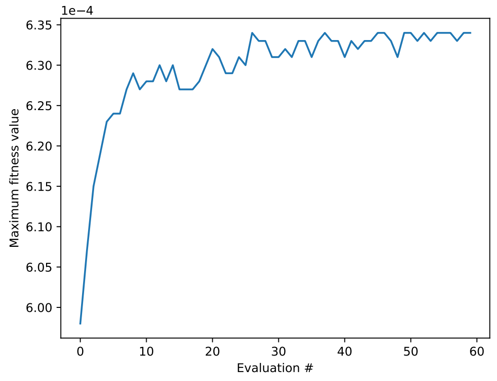

Fig. 5. PSO fitness values in `obstacles.wbt` aggregated using size of bin 20 (see Alg. 1)

TABLE III
PARAMETERS USED TO CONFIGURE PSO

Name	**Value**
# Parameters	6
Swarm Size	6
Neighbourhood Size	1
Neighbourhood Type	Fixed Radius
Max Velocity	2
Personal Best Attraction	2
Neighbourhood Best Attraction	2
Simulation Steps	700

2) Flocking among other robots - `crossing.wbt`*:* Even though a configuration for the following PSO optimisation is the same as for `obstacles.wbt` (see Tab. III) it is interesting how PSO handles an environment which changes over time. In this environment we have two group of robots moving towards each other (see Fig. 3) and the objective is to bypass each other while still maximising the metrics.

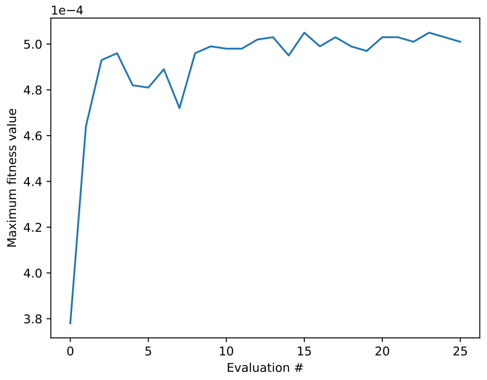

Fig. 6. PSO fitness values in `crossing.wbt` aggregated using size of bin 20 (see Alg. 1)

However, even though PSO managed to improve performance (see Fig. 6) the fitness values are more volatile. The reason for the volatile results is dynamic environment and inconsistent fitness measurements between multiple evaluations which PSO cannot handle very well. Also, we can observe significantly lower performance compared to the previous scenario due to lower velocity - robots are more densely distributed than obstacles in `obstacles.wbt`.

C. Real-world

Real-world behaviour is similar until certain point. First, communication is different and estimation of position is not well done. Sometimes, robot is not able to see other flock members, which makes flocking difficult. With loss of communication, usually robots are losing formation, and in most cases it ends with one robot leaving group. Since the objectives of the course were not to estimate performance in the real-world, here quantitative analysis is not done. In the second scenario, the proximity sensors have higher noise when detecting other e-pucks, which makes more difficult for them to avoid each other.

IV. CONCLUSIONS

The project implements a set of basic rules that allow robots to perform a flocking behaviour. It is realised using Reynolds rules of flocking and in addition obstacle avoidance algorithm is implemented. Two possible obstacle avoidance algorithms are evaluated and with the potential field algorithm, PSO optimization is performed. In the end, proposed controller is implemented on the real epuck where few drawbacks are observed. Most common problem is lack of communication and mistakes in calculating relative positions between each other, as well as not being able to communicate when there is an obstacle in between.

APPENDIX

Algorithm 1: Algorithm used to aggregate PSO performance evaluations (fs)

```
$aggregated\_fs \leftarrow []$
**while** $i < length(fs)$ **do**
    $fs\_subset = fs[i : i + bin\_size]$
    $aggregated\_fs.append(aggregated\_fs)$
```

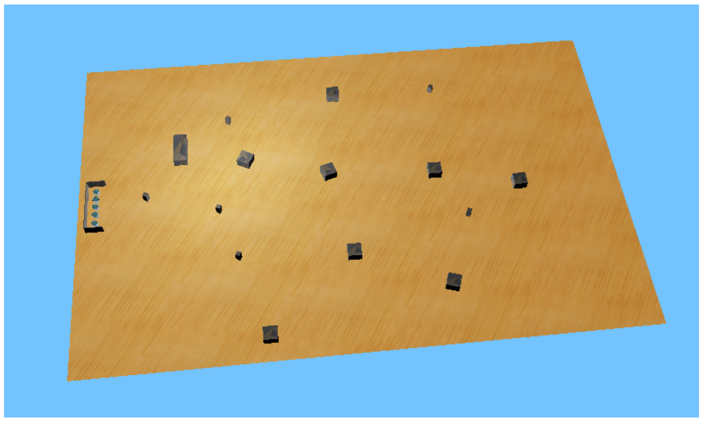

Fig. 7. `obstacles.wbt` in Webots

REFERENCES

[1] J. M. Soares, A. P. Aguiar, A. M. Pascoal, and A. Martinoli, "A distributed formation-based odor source localization algorithm - design, implementation, and wind tunnel evaluation," in 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 2015, pp. 1830–1836.

[2] A. Quraishi, A. Bahr, F. Schill, and A. Martinoli, "Autonomous Feature Tracing and Adaptive Sampling in Real-World Underwater Environments," in 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 5699–5704.

[3] C. Reynolds, Flocks, Herds, and Schools: A Distributed Behavioral Model

[4] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in 1991 IEEE International Conference on Robotics and Automation Proceedings, 1991, pp. 1398–1404 vol.2.

[5] J. Pugh, A. Martinoli, and Yizhen Zhang, "Particle swarm optimization for unsupervised robotic learning," in Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005., Pasadena, CA, USA, 2005, pp. 92–99.