Multi-robot navigation in cluttered and dynamic environments

Clément Cosson, Paco Mermoud, Tifanny Pereira, Lavinia Schlyter
Distributed Intelligent Systems

***Abstract*—This project aims at implementing a navigation strategy for a multi-robot system formed by a group of simulated e-pucks moving throughout an environment with static and dynamic obstacles. Odometry and Kalman filtering for navigation were implemented; as well as the ability for the robots to flock and cruise around in specific formations.**

I. Introduction

Multi-robot systems (MRS) have become an important part of research in Robotics, as they enable a great number of operations, such as space exploration [1], land surveying [2] etc... MRS are faster, more efficient and result in less errors compared to single robot systems [3]. These often rely on navigation techniques and formations structures also known as motion coordination [2]. In this project, five localization techniques have been implemented: two of them are based on odometry, one on GNSS data and two on Kalman filtering of these two signals. Spacial formations and flocking are also investigated. We conclude on testing parameter optimization using particle swarm optimization (PSO), a concept for the optimization of non-linear functions developed by Kennedy and Eberhart [4].

Experiments

II. Localization

Navigation is known to be one of the most challenging tasks required of a mobile robot due to the proprioceptive sensor and effector noise, aliasing and other uncertainties [5].

In order to compute the position of the robots we use a frame, where the x axis points towards the robots initial gaze, the y axis is pointing north and z outside of the screen. This will be referred to as the *relevant* frame in the localization section.

The first localization technique is based on GNSS signals only (Webots ground truth) and updates the position estimates every second.

A. Odometry

Odometry is a technique of measuring the movement of a vehicle using proprioceptive sensory data to estimate change in pose over time. [6]. The odometry state estimate is the pose of the robot $\mu = [x \;\; y \;\; \theta]$.

*1) **Odometry with wheel encoders**:* Wheel encoders measure the motion of the wheel of the robots, by counting the number of "steps" of rotation of the wheels, and translate them into linear and angular velocity estimates. Then, the linear speed estimate is projected onto the relevant frame using the heading estimate θ obtained by integrating the angular velocity estimate. This technique will allow the robot to compute its relative pose with respect to its initial pose.

*2) **Odometry with accelerometer**:* The accelerometer being in the same frame as the robot and assuming that the robots do not slide, its position estimate is computed using solely its forward acceleration estimate. Furthermore, in this part, the heading estimate is computed by the wheel encoders, because estimating the robot's rotation using linear acceleration is highly inaccurate. The position estimate is obtained by integrating twice the projected forward acceleration onto the relevant frame.

Due to the systematic errors in the accelerometer, we first estimate the bias by adding a trajectory that does not move and compute the mean accelerations in order to mitigate this error.

B. Kalman Filter

In order to increase the accuracy of the pose estimates, odometry and GNSS signals are fused with a Kalman filter. Two kalman filters are implemented. For both algorithm, the position estimate is updated every second using GNSS signals, while the prediction step will be based on both odometry techniques described above. The filter incorporates all available measurements and combines them in such a manner that statistically minimizes the error.

1) Kalman filter with accelerometer: The state vector $\mu = [x \;\; y \;\; v_x \;\; v_y]$ includes the velocities. The actuator noise is a diagonal matrix with the two first components k_1 (for x,y) followed by k_2 (for v_x, v_y). The difference derives from the single and double integration (which may result in greater error). These parameters are tuned in section II-B3.

2) Kalman filter with wheel encoder: The state vector for the wheel encoders is the pose vector. In order to compute the general propagation law at each step, the actuator noise is given by:

$$R_t = \begin{bmatrix} k \mid \Delta s_r \mid & 0 \\ 0 & k \mid \Delta s_l \mid \end{bmatrix} \tag{1}$$

where k, R and b are values determined experimentally and are discussed in the following section.

3) Hyper-parameter tuning: In order to optimize the filter for the wheel encoders, we optimized the wheel axis (b), the wheel radius (R) and the constant factor (k) in the actuator noise. This is done empirically by grid searching for the parameters which minimize the different between true position and estimated one. The same approach was used for the two hyper-parameters k_1 and k_2.

III. Flocking

In this section, the problem of flocking in a group of distributed robots using only local sensing is studied.

In the flocking and formation part of this project, the positioning system for multirobot systems presented by Jim Pugh and al. [7] has been implemented. The major difference between their work and our implementation is that, for this project, each robot's receiver and emitter are of type "radio" in webots. Signals of type "radio" are transmitted without taking obstacles into account, hence no occlusions are possible.

For the flocking algorithm, two sets of rules are followed to enable swarm movement in distributed system: separation and cohesion as outlined by Reynolds [8]. The alignment rule is replaced by a migration urge to bias the swarm towards certain coordinates.

To avoid static obstacles, each robot is seen as a Braitenberg vehicle: the wheels are directly controlled by its 8 infrared proximity sensors. The obstacle avoidance behavior and Reynold's rules are combined according to the *motor schema*'s architecture proposed by Ronald C. Arkin [9]. In this framework, behaviors are simultaneously active and cooperatively combined. Each schema generates a vector representing the desired behavioral response and their gain value is dynamically set to indicate their relative importance.

IV. Formation

In this section, the problem of maintaining a formation in a group of distributed robots using only local sensing and minimal communication is studied.

In this project, there is always a single leader per flock and all the other robots are followers. The leader's objective is to move towards the goal while avoiding obstacles. The followers' goal is to maintain the formation while avoiding obstacles. In this framework, it is possible to assign the migration urge only to the leader, because no occlusions can occur.

Each type of robot (leader or follower) has its own controller. The architecture of both controllers is based on the subsumption architecture presented by Ronald C. Alkin in 1986 [10]. As shown in figure 1, the controller of the followers has two levels. The zeroth level's objective is to have robots establish and maintain a predetermined geometric shape and is achieved by the controller described in section IV-A. The first level's objective is to have robots avoiding avoiding hazards and is achieved by the Braitenberg controller described previously. Behaviors are selected competitively: an agent is either avoiding hazards or moving into formation, but it never does both simultaneously. When an agent is very close to an obstacle, the obstacle avoidance behavior suppresses the formation behavior and the agent initiates its avoidance policy. When this agent is no longer in the vicinity of the impediment, it resumes its formation policy. The controller of the leader has the exact same architecture as the one described above, but the followers' formation behavior is replaced by the leader's migration one.

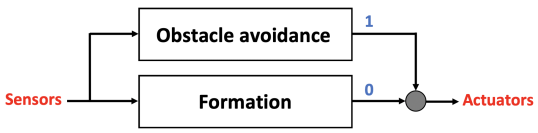


Fig. 1: Controller's architecture of the followers

In the following sections, the different formation and migration controllers will be described.

A. Formation controller

In this project, the type of robot formation is location based. Indeed, only the robot positions are considered for the formation to be correct, while the different directions they may be facing are ignored. The formation's heading is determined by the migration urge of the leader.

Furthermore, as proposed by Balch and al [11], a *neighbor-referenced* architecture has been implemented. Each robot maintains its position relative to one (or several) other predetermined robots based on range and bearing measurements. The leader only broadcasts its ID to the followers to help them maintaining the formation, but never participates itself in the consensus problem.

Two formation controllers have been implemented for comparison purposes. The first one uses graph theory and the second one is based on Maja Mataric's work [12].

1) Laplacian controller: In this section, the consensus problem with biases is solved for a group of non-holonomic robots, using the *Laplacian* based approach presented in [13].

Figure 2a shows the communication graph of the swarm. Each robot is represented by a circle containing its unique ID. The direction of the arrows underline the direction of the information's flow between the agents. The only directed edges are b,c,h and j, because the leader never process any ping messages. The edges a,d,e,f,g, and i are undirected, thus, their orientation can be chosen arbitrarily when defining the incidence matrix $I \in R^{N\text{x}E}$, where N is the number of agents and E is the number of edges. Then, the Laplacian matrix of the underlying graph representing the group of robots can be expressed as: $L = IWI^T \in R^{N\text{x}N}$, where W is chosen to be the identity matrix.

In this approach, each robot R_i has a set of neighbors N_i containing all robots R_j, such that R_i can measure its relative range e_{ij} and bearing α_{ij} to R_j. The control law for the follower's rotational control command ω_i and forward control command u_i is proportional and can be calculated as follows:

$$\left.\begin{array}{l} e_{x,i} = \frac{1}{|N_i|+1}\sum_{j=1}^{|N_i|}[-L_{ij}(e_{ij} - B_{ij})cos(\alpha_{ij})] \\ e_{y,i} = \frac{1}{|N_i|+1}\sum_{j=1}^{|N_i|}[-L_{ij}(e_{ij} - B_{ij})sin(\alpha_{ij})] \\ \Delta r_i = \sqrt{e_{x,i}^2 + e_{y,i}^2} \\ \Delta\theta_i = atan(e_{y,i}, e_{x,i}) \end{array}\right\} \longrightarrow \left\{\begin{array}{l} u_i = K_u \Delta r_i \\ \omega_i = K_\omega \Delta\theta_i \end{array}\right. \quad (2)$$

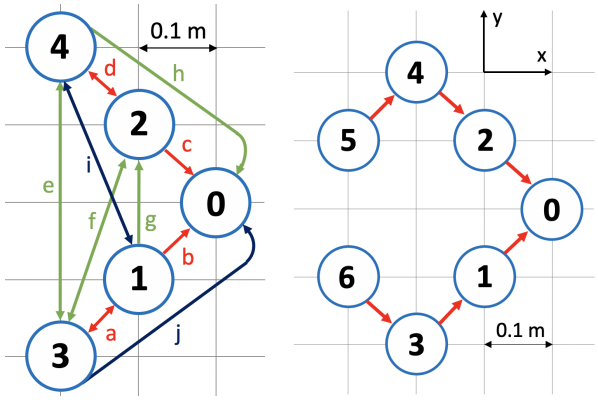
where B_{ij} is the desired range between R_i and R_j

The controller designed has two main functionalities. The first one allows to change the graph's number of edges E, for a fixed flock size of $N = 5$ robots. The number of edges can be either 4 (red edges on figure 2a), 8 (red and green edges), or 10 (red, green and blue edges). The second functionality allows to reduce the swarm size, while maintaining a fully connected graph. One of the three following possible combinations of (E, N) can be chosen:

- $(E = 3, N = 3)$ (robots 0,1 and 2 and edges b,c and g)
- $(E = 6, N = 4)$ (robots 0,1,2 and 3 and edges a,b,c,f,g,j)
- $(E = 10, N = 5)$ (all robots and all edges)

2) Mataric controller: The key idea presented by Mataric and al [12] is that every robot in the group positions itself relative to a single designated neighbor robot, its *friend*. Then, this *friend* references itself to another *friend*, and so on.

Figure 2b shows the communication graph of the swarm. Each robot is represented by a circle containing its unique ID and the arrows point towards each robot's friend. For this type of configuration, the flock size can be varied from 2 to 7.



(a) *Laplacian* based (b) *Mataric* based

Fig. 2: Communication graph for different controllers

At each time step, every agent i measures its distance r_i and angle θ_i to its friend and tries to keep it at a desired range $r_{target} = \sqrt{0.02}m$ and bearing $\theta_{target} \in \{\frac{\pi}{4}, \frac{7\pi}{4}\}$. Hence, each robot's objective is to minimize the range error $\Delta r_i = r_i - r_{target}$ and bearing error $\Delta\theta_i = \theta_i - \theta_{target}$.

For this controller, the follower's rotational control command ω_i is calculated as $\omega_i = K_\omega \Delta\theta_i$ and the follower's forward control command u_i can be computed according to three types of controllers:

- Proportional controller: (P)

$$u_{i,P} = K_u \Delta r_i cos(\Delta\theta_i) \tag{3}$$

- Proportional and integral controller: (PI)

$$u_{i,PI} = u_{i,P} + K_I \int_0^t \Delta r_i dt \tag{4}$$

- Nonlinear proportional and integral controller proposed by Falconi and al. [14]: (NPI)

$$u_{i,NPI} = u_{i,P} + K_I \int_0^t \Delta r_i f(\Delta\theta_i) dt \tag{5}$$

where $f(\Delta\theta_i) = \beta_I(\frac{-1}{1+e^{-s(||\Delta\theta_i||-\gamma)}}) + 1$

B. Migration controller

The leaders migrate towards a specific goal using their localization estimate, based on the localization technique described in section (II-B2), in a simple proportional controller.

In order to speed up the convergence, the migration weight is increased artificially as the leaders get closer to their goal.

In the crossing world, to make the crossing smoother between both flocks, one of the leaders' y-coordinate of the migration destination is slightly displaced when leaders are close and then changed back to its original value, when they move away.

V. PSO

In this section, the goal is to optimize a set of parameters for obstacle avoidance, the followers' formation controller and the flocking algorithm using Particle Swarm Optimization (PSO) to maximise the performance metrics.

The architecture of the PSO algorithm designed has several optional functionalities. Firstly, a noise resistant version of PSO can be run, allowing the algorithm to be more robust to potential lucky bad solutions, by re-evaluating the personal best particle. Secondly, the parameters' search space can be bounded. The third functionality allows to choose between spawning the robots in the arena in a fixed or random position. Finally, it is possible to initialize all the particles either to random values or to the hand-tuned parameters. The main idea behind this approach is to search an optimum near an already working solution, in order to possibly improve the convergence of the algorithm.

The parameters optimized with PSO and the fitness functions used for the three behaviors are described in the following sections.

1) Obstacle avoidance: In this setup, the 16 Braitenberg's weights are optimized in "pso_world_simplified_avoidance". The fitness used for this part is the one introduced by Floreano and Mondada [15]: $V(1-\sqrt{\delta V})(1-i)$, where V represents the mean speed of the wheels, δV the absolute difference between wheel speeds and i the activation value of the proximity sensor with the highest activity. The PSO uses a public individual heterogeneous optimization strategy.

2) Followers' formation controller: In this setup, the forward control command's weight K_u, the rotational control command's weight K_ω, the avoidance and formation thresholds are optimized in "pso_world_simplified_formation". The fitness used for this part is the formation performance metric provided for this project [16]. The PSO uses a public group homogeneous optimization strategy.

3) Flocking: In this setup, the cohesion weight and threshold, the separation weight and threshold and the migration weight of Reynold's algorithm are optimized in "pso_world_simplified_flocking". The fitness used for this part is the flocking performance metric provided for this project [16]. The PSO uses a public group homogeneous optimization strategy.

4) Flocking with obstacle avoidance: In this setup, PSO tries to optimize the obstacle avoidance parameters with the flocking parameters simultaneously by multiplying their respective fitness together in "pso_world_flocking". The PSO uses a public group homogeneous optimization strategy.

VI. Results

1) Localization techniques: After several tests we noted that using the GPS for updating the header is not satisfactory, this is due to it being updated every second and thus having two measures which differed by one second. The GPS heading computed a single one is not conceived as highly accurate compared to other methods [17].

The hyper-parameter tuning for several parameters in the Kalman wheel encoder method yielded the following results and we kept them throughout the project: $b = 0.05$, $R = 0.02$,

$k = k_1 = 0.05$ and $k_2 = 0.05$. The performance metric of all the localization techniques, that is the difference between the true position and the estimated ones, according to the metrics given are pictured in figure (3).

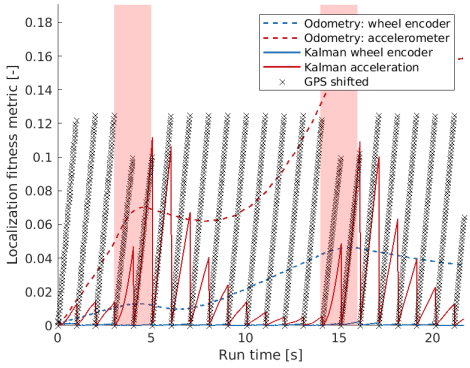
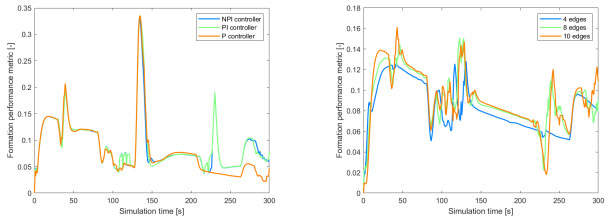


Fig. 3: Localization performance metric as a function of time

The best estimate is given by the wheel encoders combined with Kalman, we note the fitness being at almost zero, for this reason it was kept as the way of estimating the agents' pose in the further parts of this project. It is also interesting to note the oscillations caused by the GPS updates: the odometry estimates accumulate errors and every second the GPS correct them. The red bands in figure 3 corresponding to the start of a turn, highlight the increase in error, in both odometries but more significantly in the accelerometer which also worsens with time. Because the GPS is updated only at every second, its performance is perfect only periodically, otherwise it increases as the robot moves and when the robot is turning, the GPS estimates yield less error.

2) Formation: Figure 4a shows that PI and NPI controllers have similar performances, but they are more robust than a simple P controller. The real advantage of a NPI controller compared to a PI controller happens when the leader changes its trajectory, which has not been tested in this project.

Figure 4b highlights the fact that groups with many connections are more rigid. Indeed, the controller based on a graph of 10 edges always results in slightly better performance metrics, than the ones based on graphs with only 4 and 8 edges.



(a) *Mataric* controller for different controllers: P, PI and NPI (b) *Laplacian* controller for different number of edges

Fig. 4: Formation performance metrics with $MW = 0.03$, $N = 5$ on the "test_obstacles" world for different controllers

Intuitively, the higher the flock size, the higher the probability to encounter obstacles and hence, the worse the performance metric. As we can see, this intuition has been verified with the experimental results in table I.

TABLE I: Average of the formation metric over 45s for different flock sizes N (Mataric PI controller, $MW = 0.03$, "test_obstacles" world)

N	3	5	7
Average	0.3687	0.2508	0.1872

Finally, the experiments showed that *Mataric*'s controller led to looser formations compared to *Laplacian*'s controller. Thus, *Mataric*'s controller performed better on the "test_crossing" world, because it allowed the robots to be further apart and hence the two flocks could cross more easily. While *Laplacian*'s controller was performing better on the "test_obstacles" world, because since the formation is tighter, the followers always ended up avoiding the obstacles on the side that was closer to the leader.

3) Flocking, formation and PSO: The setup presented in section V-4 always led to poor results due to the difficulty of finding the right balance between the two types of fitness functions (i.e obstacle avoidance and flocking metrics). However, optimizing the three behaviors separately led to very good results when assembling them as we can see in figure 5. Indeed, the performance of the flocking and the formation with obstacle avoidance increased drastically with the weights obtained with PSO.

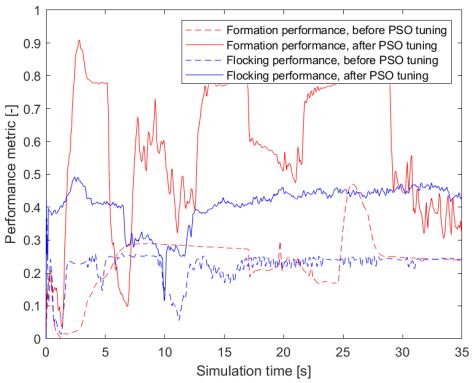


Fig. 5: Comparison between hand-tuned and PSO parameters for flocking and formation in "test_obstacles"

VII. Conclusion

The large diversity of options tested (Mataric/Laplacian, number of edges, P/PI/NPI, empirical weights or tuned with PSO...) allowed us to find an very good solution for each challenge. These optimal solutions are the following:

- World "localization": Kalman filter with wheel encoder and low-frequency GPS.
- World "crossing": Mataric PI controller, with non-PSO weights for both leader and followers.
- World "obstacles": Laplacian with maximum number of edges, with PSO weights for followers only.

These tailored methods produce satisfactory results. This can be seen by observing the robots behavior, and is confirmed by the metrics values.

REFERENCES

[1] Terry Huntsberger, P Pirjanian, A Trebi-Ollennu, H Das, H Aghazarian, A Ganino, M Garrett, S Joshi, and P Schenker. Tightly-coupled coordination of multi-robot systems for mars exploration. 2001.

[2] Tamio Arai, Enrico Pagello, Lynne E Parker, et al. Advances in multi-robot systems. *IEEE Transactions on robotics and automation*, 18(5):655–661, 2002.

[3] Ahmed Benzerrouk, Lounis Adouane, and Philippe Martinet. Stable navigation in formation for a multi-robot system based on a constrained virtual structure. *Robotics and Autonomous Systems*, 62(12):1806–1815, 2014.

[4] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[5] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots, Chap.5 p181*. MIT press, 2011.

[6] Martinoli. A. An introduction to localization methods for mobile robots in multi-robot systems. 2021.

[7] Jim Pugh, Xavier Raemy, Cédric Favre, R. Falconi, and A. Martinoli. A fast onboard relative positioning module for multirobot systems. *Mechatronics, IEEE/ASME Transactions on*, 14:151 – 162, 05 2009.

[8] Craig W Reynolds. Steering behaviors for autonomous characters. In *Game developers conference*, volume 1999, pages 763–782. Citeseer, 1999.

[9] Ronald C. Arkin. Motor schema - based mobile robot navigation. *The International Journal of Robotics Research*, 1989.

[10] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.

[11] Tucker Balch and Ronald Arkin. Behavior-based formation control for multi-robot teams. *Robotics and Automation, IEEE Transactions on*, 14:926 – 939, 01 1999.

[12] Jakob Fredslund and Maja Mataric. A general algorithm for robot formations using local sensing: And minimal communication. *IEEE Transactions on Robotics*, 18:837–846, 10 2002.

[13] R. Falconi, S. Gowal, J. Pugh, and A. Martinoli. Graph-based distributed control for non-holonomic vehicles engaged in a reconfiguration task using local positioning information. pages 1 – 6, 05 2009.

[14] Riccardo Falconi, Sven Gowal, and A. Martinoli. Graph based distributed control of non-holonomic vehicles endowed with local positioning information engaged in escorting missions. *Proceedings - IEEE International Conference on Robotics and Automation*, 05 2010.

[15] Francesco Mondada Dario Floreano. Evolution of homing navigation in a real mobile robot. *IEEE transactions on system, man, and cybernetics-part B*, 26(3), 1996.

[16] Martinoli. A. Moodle : Performance metrics. 2021.

[17] Ji Hyoung Ryu, Ganduulga Gankhuyag, and Kil To Chong. Navigation system heading and position accuracy improvement through gps and ins data fusion. *Journal of Sensors*, 2016, 2016.