Report Project - Intelligent Distributed Systems

Jacopo Endrizzi [240307], Thade Pfluger [236862]

2023 - 2024

***Abstract* - Intelligent distributed systems are multi-agent systems in which the actors share their sensed information among each other to improve knowledge. Inspired by a social experiment, this project work aimed to verify whether or not the application of consensus algorithms actually improves the performance of robots in finding objects. We step-wise reduced the a priori knowledge of the system and simulated collaborative strategies to complete the search missions. We investigated various cooperation and information-sharing behaviors and proposed a suitable hardware implementation consisting of mobile robots and the communication infrastructure. The application of consensus algorithms proved effective to reduce the time and distance covered for each robot to reach its target.**

Key words: distributed robotics, consensus algorithm, flocking behavior.

I. Introduction

I. Problem description

There is a social experiment in which a group of children are each asked to find a balloon with their name written on it. It takes place in a closed room and the kids and balloons are placed on opposite sides of it. At a signal, the children run around in search of their own balloon. As you can imagine, it gets quite chaotic. What is more, the total time each child spends looking for his or her balloon is considerable. The counter-intuitive but more effective solution to this confusion seems to be for each child to run to the nearest balloon, read the name on it and deliver the balloon to the correct owner. In this way, in a very short time, all the children can play with their own balloon. This experiment should teach them that working together makes it easier and quicker to complete tasks.

The main interest of this project is the collaborative part. Following some abstractions and assumptions, we try to recreate the kid-to-balloon environment with robots, targets and sensors. Kids, as human beings, are full of natural sensors that work together to reach goals. For robots, however, we have to decide which and how many sensors are needed depending on the objectives. In general, exploiting more sensors gives us a better performance, but we can drastically reduce their quantity, and therefore the cost, by acting intelligently. One of the "smart moves" is the implementation of consensus algorithms, a way of achieving agreement on single values across multiple distributed processes. These techniques introduce intelligent collaboration between robots, allowing them to move together in an optimal way. Special communication protocols are required and care must be taken to avoid message accumulation. The development idea we want to follow is to gradually reduce the amount of information available from the room's sensors and to focus instead on robots talking to each other and taking joint decisions.

II. Problem formulation

We start by creating the necessary elements: room, kids and balloons. A basic rectangle represents the room whose dimensions depend on the number of elements (kids and balloons) present to avoid problems of overcrowding. For visualization purposes only, we prefer the width to be greater than the height. Kids are created as a *struct* with some basic fields: x and y positions, x and y velocities, radius r and identification number ID. We choose to represent the kids as circles. Several other fields are added later as needed by the functions. A similar procedure is followed for the balloons. In this case, the basic fields are: x and y positions, edge l and identification number ID. We have *edge* instead of *radius*, because balloons are represented as squares.

Now that we have the basic elements (sensors are created and deployed later), a specific function deals with their distribution. We can choose between a random allocation or following specific patterns. In particular, the random type is necessary to test the flexibility and adaptability of the implemented models, while we will use the patterns to compare the model's performance. Taking into account the different sizes of kids and balloons, the function avoids overlaps between the elements and with the walls. For easier visualization, it also shows each child and balloon with its identity number and specific color. Figure 1 shows an example of a random distribution. In A you can find the patterns.

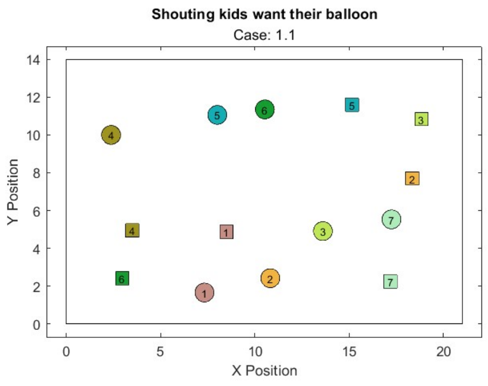


Figure 1: Random distribution of kids and balloons in the room

Once the robots and targets are in the room, we can add sensors for optimal guidance. We choose to place the sensors on the walls at equal angular intervals. This simplifies the distance calculations and guarantees full coverage of the room. To find the optimal number of deployed sensors, we look for a solution that should give us the best performance with the least number of sensors. To find it, we create a grid of reference positions and compare the estimates made by different numbers of sensors. We implement a consensus algorithm with Metropolis-Hastings weighting to get only one common value for each position estimate. This procedure will be useful later when we want to locate the balloons and the children's movements. The final number of sensors is chosen to minimize the maximum error of the estimates on both the x-axis and the y-axis. In the case in which one set of sensors minimizes the maximum error on the x-axis and another set of sensors minimizes the maximum error in the y-direction, we compare the performance of both sets on the two axes and take the overall minimum. This analysis avoids falling into the case where simply increasing the number of sensors gives better performance.

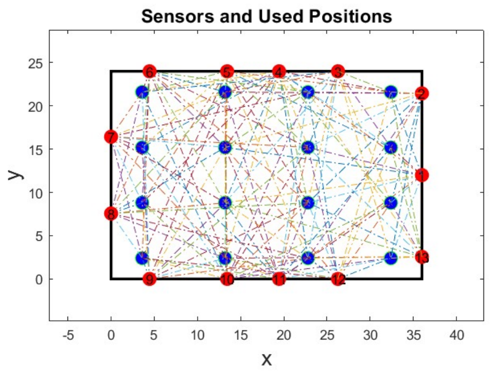


Figure 2: Optimal sensors number with used reference positions

To allow comparison between the different sets, we assume that all sensors have the same uncertainties, a behavior that does not exist in real situations. To compensate for this false assumption, we will later add an external random noise to the kids' position estimates. Figure 2 shows the distribution of the optimal number of sensors after all estimates have been made. A closer look shows that not all sensors (red dots on the wall) are connected to every single reference position (blue dots in a grid). This is because we assume that the sensors have a limited range.

II. Adopted Models

The aforementioned position estimation is implemented in a separate function that can be called at each iteration. In figure 3 we show the outcome of this function. The actual position is plotted as a blue circle and the estimated one as a red cross. When dealing with the real scenario, we don't have access to the actual positions, but only to the estimates returned as the consensus of all sensors in range. Therefore, this function is very important.

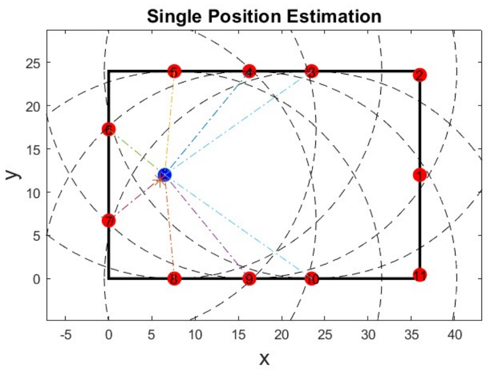


Figure 3: Example of Position Estimation

I. System Models

Without going into too much details, in this section we present the models we used to simulate a possible real-world implementation.

1. Social Force Model

A Social Force Model (SFM) describes the behavior of moving pedestrians in crowded spaces. Their interactions with the environment and other people are modelled according to a set of forces. These are the main forces to consider:

- **Attraction to goal**: forces that move the pedestrians towards their goal;
- **Repulsion from obstacles**: forces that prevent pedestrians from colliding with obstacles like the room's walls;
- **Social forces**: both attractive forces for tendency of walking in groups and repulsive forces to avoid collision with other pedestrians

Each pedestrian has a desired direction of movement and, according to the net effect of the listed forces, their velocity is updated over time. They can accelerate, decelerate or change direction. In this way, the model simulates the collective behavior of pedestrians.
Our implementation is based on the work of [1] although it only applies to the estimated position we obtain as sensor readings with some modifications. The complex force equations used in their model result in some unwanted and unrealistic rebounds as well as overlapping behavior which is difficult to handle by tuning the parameters.

2. Distributed Least-Squares Algorithm

The Distributed Least Squares Algorithm allows us to solve the least squares problem in a distributed way. This means that we can avoid a central node receiving all the information, while all nodes compute a locally unbiased weighted least-squares estimate. It can be demonstrated that, at steady state, all the estimates converge to the global weighted least-squares solution.
We briefly illustrate the algorithm. Each node starts with a local estimate of the desired parameter. Then the algorithm iterates through these steps:

- **Local computation**: each node considers local composite information matrix and composite information state, initialises its composite elements and provides its contribution.
- **Communication**: nodes exchange their local computations with neighbours.
- **Aggregation**: each node aggregates received information to update its local estimate.
- **Convergence Check**: convergence is tested on a predefined criterion.

When the convergence criterion is met, the algorithm stops and each node has the same estimate of the solution.
Since we obtain an average consensus problem, we introduce a Metropolis-Hastings weighting scheme for the protocol design as presented in [4].

II. Robot, sensors and actuators

For our robotic hardware, we decide to use omnidirectional robots with Mecanum or Swedish wheels. These robots are able to move freely in the $2D$ plane due to the absence of holonomic constraints. This is the best fit for the rapid changes in direction of motion that the SFM may produce. Furthermore, the robots are assumed to be robust, so that contact with other robots, although to be avoided, doesn't lead to failure or breakage of system components. This should represent the expected chaos that will inevitably occur when a bunch of kids run across the room.
The most important sensors that each robot must be equipped with include a 360° obstacle detection system to spot other robots in the vicinity, and a camera or RFID (Radio Frequency Identification) recognition system to determine to whom the observed target belongs. Figure 4 shows our robot kit of choice, it provides plenty of space to mount sensors, antennas, etc. Its round shape can easily be covered with a shell to protect it from collisions. Last but not least, its shape and colorful design match our simulation perfectly.
Apart from this, we treat the robots as a black box and do not specify their actuators in detail. We assume that the next destination or direction of movement is translated internally to the robots' actuators.

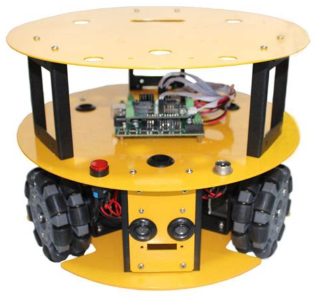


Figure 4: Omnidirectional robot [2]

The sensors placed in the room, also called anchors, reference nodes, or base stations, need to communicate wirelessly with our mobile robots, which we can also call tags, targets, or mobile nodes. We decided to use wireless LAN base stations according to the newly released IEEE 802.11az standard [11]. For the targets, we use compatible antennas mounted on the mobile robots. The specifications are described in more detail in the next section. Yet, one assumption needs to be emphasized up front. That is, the range of such a wireless system would easily be sufficient to cover the entire area of the room, but since our hypothesis is based on sensors with limited range, we will proceed with this assumption in mind.

III. Communication System

The goal of our project is to demonstrate the advantage of collaboration over solo work by each robot. This suggests that the communication system plays a crucial role for the successful demonstration of our initial hypothesis. In this section, we first give an overview of what information is exchanged between the actors at what time, and later we look at possible hardware implementations that allow for the desired behavior.

Figure 5 outlines the information flow between the robots (round shape) and the sensors (house shape). The scheme shows a general overview that actually differs slightly from case to case. The specific description of each case can be found in section 3.

After successfully connecting to the network, the system enters the idle state. From there, the robots are assigned their first target and start moving. The first direction depends on the evaluated case and can be either directly towards the correct final goal, towards the closest balloon, or a rendezvous to form a swarm. Also in the following iterations the sent positions are always determined according to the selected case and can be e.g. random or determined by the occupancy map. The waiting time represents the simulation time for the SFM and can be adjusted in the parameter settings. In practice, it also accounts for the time in which the robots start to drift due to the lack of external feedback on their positions. A new localization is then performed using the exteroceptive sensors.

Appropriate sensors on the robots sense their environment for other robots and exhibit an obstacle detection and avoidance behavior that does not require communication with the sensor nodes of the room, since the implementation of which is part of the robot internal SFM. Similarly, the robots have sensors to detect a target next to them, as well as the functionality to determine the number written on the target. If the reached target belongs to the robot, it stops its movement. If it belongs to another robot, the target ID is shared with the nodes in range, which in turn form their consensus about this robot's position and send it back as the new destination for the respective other robot. This process is repeated until all robots have reached their correct destination.

The message exchange between the sensor nodes and the mobile robots must to be achieved by wireless communication. Yet, each node can only reach some robots to estimate their position. This is because we assume that the range of the wireless system is not sufficient to cover the entire room.

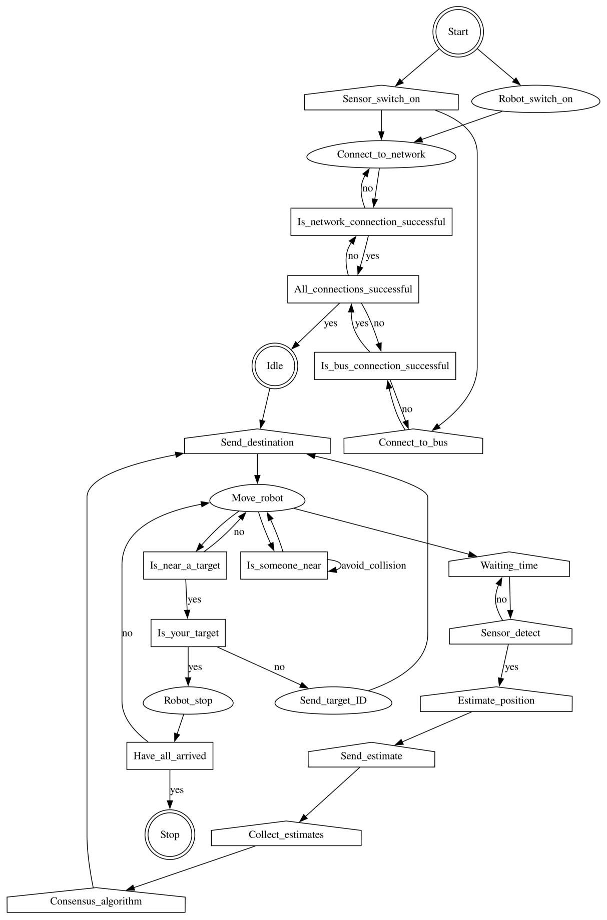


Figure 5: Communication control-flow diagram

Consequently, each of the nodes is also not in range for wireless communication with every other node but the network is still jointly connected. That means no node is isolated from all the others and building a consensus is possible. However, we intend to connect the evenly distributed sensors via a wired backbone around the walls of the room to reduce problems related to wireless communication like reflections and message congestion.

The lecture slides noted that combining wired cabling with a wireless network is unusual and presents some difficulties [3]. Not knowing exactly how such systems might interact and what a hardware implementation of a theoretical information flow diagram might look like, we still reason that it might make sense for our case and chose to combine a WLAN network with Real-Time Ethernet (RTE) as a backbone.

The first difficulty presented deals with transmission support. We learned that the low throughput can be resolved by using a high performant WLAN, a criteria our chosen standard IEEE 802.11az fulfills. Also infrastructure-wise, the many small subnetworks connected over the wired backbone should limit the packet rate on each of them. Regarding the second issue with determinism, i.e. transmission delays and network congestion, we studied a solution with Time-Division-Multiple-Access ($TDMA$). Our chosen standard, IEEE 802.11az, instead uses Orthogonal Frequency Division Multiple Access ($OFDMA$) and Multi-User Multiple-Input Multiple-Output ($MU-MIMO$). It is argued that it has improvements with respect to the Fine Timing Measurement (FTM) scheme used in the basic IEEE 802.11 standard [9]. The FTM technique improved positioning accuracy down to about 1–2 meters in 2016, while IEEE 802.11az can go to sub 1 meter, and even accurately locate in the range of less than 0.1 meter [11].

The third and final problem addressed in the lecture slides was robustness. Since our simulation is not intended to represent an industrial environment, issues of electromagnetic interference and latency are expected to have little impact. However, mounting multiple antennas on our robot platform to increase the reliability of message delivery, as well as using a 5 GHz WLAN band, are some easy to implement solutions. Our chosen standard even uses a 6 GHz band [11].

Because of our multiple anchors and mobile tags the chosen WiFi topology is a Multipoint-to-Multipoint mesh network. When a robot transitions between the ranges of WiFi base stations, it performs the so called hand-off, which means it changes its point of attachment. Connection to such a Basic Service Area (BSA), the spatial area covered by one of the WLAN access points (AP), can be initiated by a special protocol called scanning, consisting of a probe request and probe response to a new set of nodes or APs in range [3].

For the wired backbone, that connects those BSA on the MAC sub-layer [3], we chose Profinet with TDMA as the RTE option. The TCP/IP standard should be used for device setup and network startup. For the remaining data transfer, RT_CLASS 1, which is real-time, event-based data communication, should be sufficient. The more powerful classes 2 and 3 are overkill for our use case.

III. Presentation of cases

In this section we present the different cases that we want to analyze and discuss. We gradually reduce the amount of prior knowledge about the room and focus our attention on the importance of robot cooperation. We want to demonstrate that when the robots act individually, they need much more time and distance to reach their goals, compared to a team of communicating robots. The different cases we will present are the following:

- **Full knowledge**: Robot ID's, positions and corresponding targets are known. Thanks to the sensors, kids are guided towards their correct balloon.
- **Restricted knowledge**: Robot ID's, positions, and target positions are known. Thanks to the sensors, kids are guided towards the nearest balloon.
- **Exploration**: Only robot ID's and positions are known. Kids wander around looking for their balloon.
- **Mapped exploration**: Only robot ID's and positions are known. Kids avoid returning to already explored spaces.

Before proceeding, we discuss some issues common to all cases. We inherit some constraints directly from the choice of model for robot movement. The

SFM always requires knowledge of the initial positions and final destinations of the agents. In particular, the final destinations are the basis for calculating the attractive forces that push the robots towards their destination. If we don't have any information about the final destination as in the case of an unknown environment, we have to find a way to "fake" it. Our workaround is presented later. Another problem is that the model computes positions, velocities and forces all together for one step. This becomes a problem when we work with estimated values instead of actual values. We will try to illustrate the current workflow of the unmodified SFM with estimated positions:

1. We start from the actual positions of the robots. Their estimates are communicated as initial conditions to the SFM;
2. the SFM computes forces and returns final positions as the next actual positions.
3. These new "actual" positions, however, are derived from estimates, are then estimated again, and fed back into the model as initial conditions for the next step.

This error propagates significantly. Moreover, in step 2., we clearly see another problem: the calculated forces for the estimated positions are applied to the actuators of the physical robots, which by nature are at their actual position. These are the forces that move the robots! This can cause many problems, as these forces can cause them to collide with each other, miss their target, or hit the walls. The SFM clearly is in need of an improvement to become valid for our use case. We introduce two workarounds to these problems: we reduce the optimization step time and slightly change the way we perform the second estimation (i.e. point 3.). Obviously, we cannot choose the optimization step too small, otherwise the system will not be able to find a solution in a considerable amount of time. Before estimating the final position of a step again, we shift the computed trajectory, beginning from the last estimate, into the last actual known position. Only now the newly obtained final point of the computed path, which represents our new actual position, is estimated again.

To ensure that the estimation with consensus algorithm is better than a single sample, we build a grid of reference positions for testing. In the case of consensus, all sensors that detect the considered position (i.e. it is in their range) share their values to arrive at an agreed estimate. In the single sample case, the sensor closest to the actual position is the one that performs the estimate. We collect all the errors and analyze them. In figure 6 we can clearly see the difference. With the consensus estimation, all values stayed in a much thinner Gaussian bell, while with a single estimation, the values are much more spread out. We report an obtained case:

$$\begin{aligned} \mu_{x_{cons}} &= 0.0001m & \mu_{x_{single}} &= 0.0002m \\ \mu_{y_{cons}} &= -0.0003m & \mu_{y_{single}} &= 0.0003m \\ \sigma_{x_{cons}} &= 0.013m & \sigma_{x_{single}} &= 0.037m \\ \sigma_{y_{cons}} &= 0.026m & \sigma_{y_{single}} &= 0.075m \end{aligned}$$

We are not surprised that the mean errors are quite similar, since all the errors are generated by a random function that picks values from a uniform distribution. It is the standard deviation that undoubtedly marks the difference.

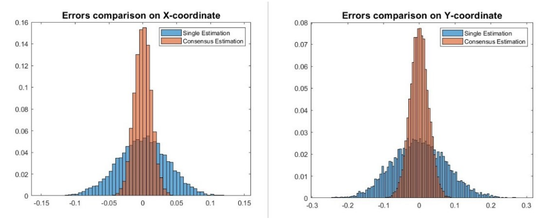


Figure 6: Errors comparison in position estimation

IV. Case description and results

I. Full Knowledge

As a first case, we consider having complete information about both robots and targets. This case should serve as a reference for all subsequent implementations. In particular, we want to test the correct functioning of the SFM and the estimation by the sensors. We divide this setup in two sub-cases. First, we consider the absolute positions of the robots. This means that the SFM receives the real positions of the robot as input. In figure 7 we show a plot of the result. The deviation from straight paths is due to the repulsive forces in the SFM that avoid collisions between robots. If a single robot would be simulated, we would observe a perfectly straight path.

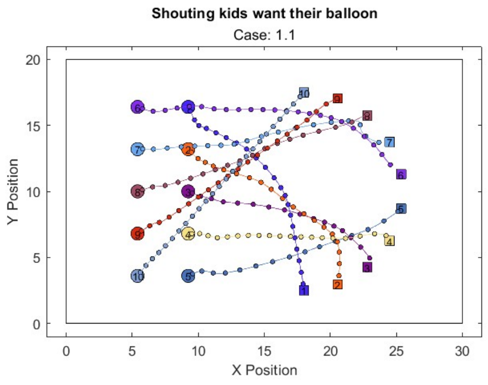


Figure 7: Case 1.1

In the second sub-case, we move closer to a real environment where the actual positions can only be estimated. Hence, we make use of the corresponding function. In figure 8 we show the final plot with all robots that have reached their targets. We can observe that the trajectories are no longer leading straight to the goal as in figure 7. Estimated positions are shown as empty circles of the same color as the actual positions.

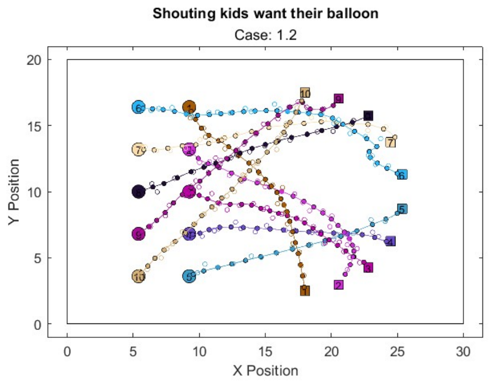


Figure 8: Case 1.2

Everything seems to be working fine. We can proceed by removing some of the available information. In this first case the only collective behavior is the mutual avoidance between kids and obstacles. We will now improve the cooperation by making the robots talk to each other to achieve their goals more efficiently.

II. Restricted Knowledge

This second analysis considers full knowledge of the robots (identities, positions, velocities), but only the position of the targets. This means that the sensors cannot directly guide the children to their balloons. Therefore the sensors indicate the way to the closest balloon for each kid. As before, we will divide this case into two sub-cases. Our main interest is to show the importance of collaboration. In the first sub-case, once a kid reaches a balloon, it verifies to whom it belongs (i. e. it checks the balloon's ID). If it's its own balloon, the kid stops; if it belongs to someone else, it selfishly proceeds to the next closest balloon. In figure 9 we see the result of the simulation, which already shows much more movement with respect to case 1.

As a second sub-case, we introduce communication between robots. When a kid now "discovers" the identity of a balloon that is not its own, it sends a message containing its current position and the ID

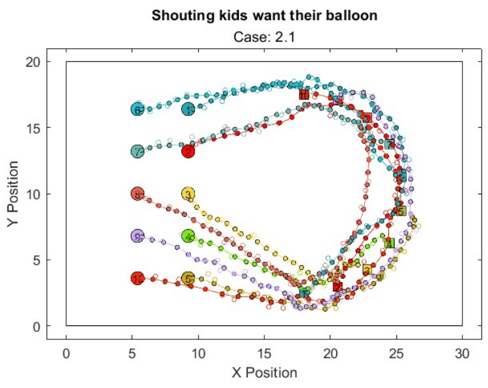


Figure 9: Case 2.1

of that balloon. The information is shared via the sensors over the wired backbone, so that the sensors in reach of the corresponding kid can tell it about its correct final destination. Another piece of information shared with all other robots is to avoid checking this balloon's ID again. We assume a failproof ID recognition. In figure 10, we can observe a run with this implementation. Already, the scene becomes more clear.

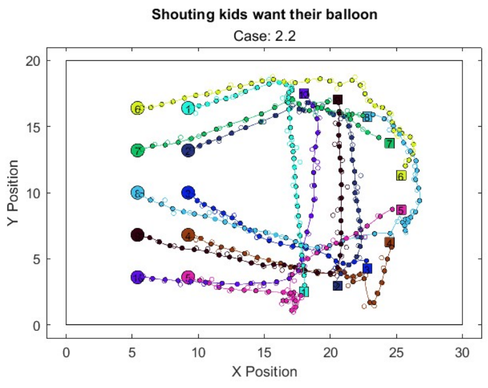


Figure 10: Case 2.2

We immediately understand how much a simple collaboration protocol like this can save in terms of exploration path and time. Robot communication avoids repetitions in the discovery of balloon IDs. The robot movements appear much more organized and "intelligent".

The next step will consider quite a different approach, as we will remove all the information about the targets. This will completely change the way we interpret the "destinations", a fundamental parameter of the SFM. Finally, we are also ready to introduce collaborative behaviors of the robots.

III. Exploration

As a third case, we focus on strategies to actually find the balloons. In fact, we now take away any kind of a priori knowledge about the targets. We know neither their positions nor their identities. The kids have to look for their targets without any sensor guidance. The sensors still remain relevant since they estimate the kids' positions, which in turn are important as reference target points once a balloon has been found. The concept of the first sub-case works like this: Kids make their way through the environment with a randomized movement (explanation below). When one kid finds a balloon, it checks whether it belongs to it (i. e. it checks the correspondence between ID numbers). If the answer is yes, the kid stops; if it belongs to another kid, it shares its estimated position along with the ID number of the discovered balloon, and continues its random movement. As before, the respective kid is informed about the discovery and guided to its destination. However, it is not the actual position of the balloon that is communicated, but rather the estimate of the position of the kid who found it. In cases where the estimate is far from the actual position of the balloon, this can lead to problems with not finding the balloon because it may be outside the detection range of the robot's sensors. But thanks to the random nature of movement, the kid continues to circle around the estimate destination until it eventually detects the balloon. This type of exploration is not ideal, and starting a spiral from the estimate position to scan the surrounding area could be a possible way to achieve faster and more reliable detection. For now we will stick with the random approach without the spiral.

As mentioned before, the SFM requires a destination as input to properly compute the attraction forces towards that destination. For the random movement we therefore created another function. Around its previous heading we define a set of possible positions of which the robot randomly chooses one as its next target.

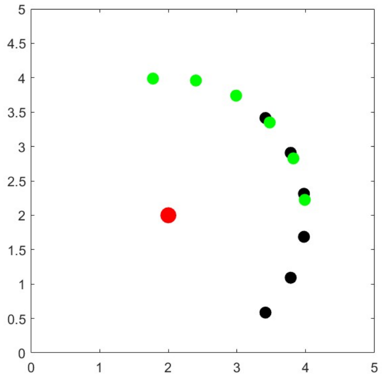


Figure 11: Possible positions

The left side of figure 11 shows the robot (red dot) and equally spaced positions on a 90° arc (black dots) around the current direction of movement. To prevent kids from leaving the room, we first check which of the possible new positions are located inside the room. Then, one of these is randomly chosen as its new target. If all are located outside, the the orientation is rotated until some do (green dots).

This preference for a pedestrian to move forward can also be seen in figure 12, which shows a simulation. As soon as a kid is told the location of its balloon, it decisively runs towards it, as can be seen by the dark colored, fairly straight path.

It is clear that with this fully exploratory approach, the time and distance traveled can increase significantly. This is understandable: the robots have no idea where to go. That's why we think that a little help can be valuable. But before we present case 3.2, we will introduce some more fancy strategies for moving the robots together in a swarm. When dealing with multiple agents, consensus algorithms are quite flexible, but they are also extremely powerful for some specific problems. Literature is quite rich on the argument [7] [8] [6]. Two of the most recurrent situations can be suitable to our particular case: flocking behavior and rendezvous strategy.

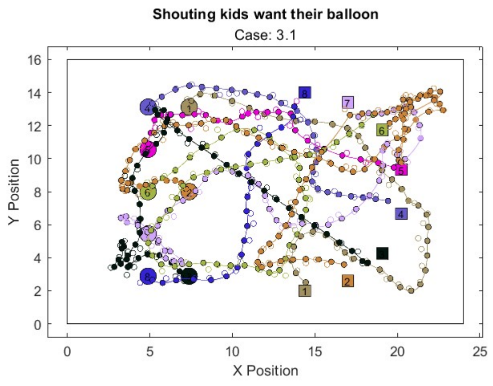


Figure 12: Case 3.1

1. Flocking

By "flocking" we mean multiple agents moving together in a common direction. The whole group reacts coordinately to a dynamic environment while preserving the integrity of all agents. The most commonly implemented algorithm is the Boids model introduced by Craig Reynolds in [10].

Since it is also quite simple, we briefly show the mathematics behind it. For each robot i, we define its position and velocity at time t as:

$$\begin{aligned} p_i(t) &= [x_i(t), y_i(t)]^T \\ v_i(t) &= [v_{x_i}(t), v_{y_i}(t)]^T \end{aligned} \tag{1}$$

Three forces are considered to govern each robot:

- **Separation** (or collision avoidance): avoid crowding neighbors;

$$F_i^{sep} = \sum_{j \in \mathcal{N}_i} \frac{p_i - p_j}{||p_i - p_j||^2} \tag{2}$$

- **Alignment** (or velocity matching): steer towards the average heading of neighbors;

$$F_i^{align} = \frac{1}{||\mathcal{N}_i||} \sum_{j \in \mathcal{N}_i} v_j - v_i \tag{3}$$

- **Cohesion** (or flock centering): move towards the average position of neighbors.

$$F_i^{coh} = \frac{1}{||\mathcal{N}_i||} \sum_{j \in \mathcal{N}_i} (p_j - p_i) \tag{4}$$

The total force F_i acting on robot i is a weighted sum of these forces:

$$F_i = w_s F_i^{sep} + w_a F_i^{align} + w_c F_i^{coh} \tag{5}$$

where w_s, w_a and w_c are weights to control the influence of each parameter.
Velocity and position of each robot are updated according to:

$$\begin{aligned} v_i(t+1) &= v_i(t) + \Delta t \cdot F_i \\ p_i(t+1) &= p_i(t) + \Delta t \cdot v_i(t+1) \end{aligned} \tag{6}$$

This basic model is not sufficient for our problem. We would need to add another force, called F_i^{avoid}, to prevent running over obstacles (i.e. balloons). The formulation is the same as for the separation force, but instead of considering other robots, it considers the targets positions. So the total force becomes:

$$F_i = w_s F_i^{sep} + w_a F_i^{align} + w_c F_i^{coh} + w_o F_i^{avoid} \tag{7}$$

We perform two types of simulations to compare the flocking with the individual behavior. The former can be observed in figure 13, while the latter can easily be achieved by muting the contributions of cohesion and alignment forces by setting their weights to 0. In particular, we use the following weights:

$$\begin{aligned} w_{s_{flock}} &= 0.5 & w_{s_{single}} &= 0 \\ w_{a_{flock}} &= 0.1 & w_{a_{single}} &= 0 \\ w_{c_{flock}} &= 1.5 & w_{c_{single}} &= 5.0 \\ w_{o_{flock}} &= 1.5 & w_{o_{single}} &= 1.5 \end{aligned}$$

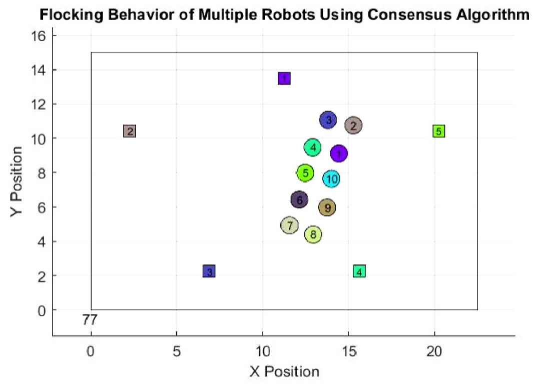


Figure 13: Flocking behavior

We consider 10 robots and 5 targets. It turns out that the individual robots reach their targets faster. Flocking slows the system down a lot. We observe that each robot in the swarm needs about 4 times longer and 2.5 times the distance compared to the individually traveling robots to complete their mission. This might be caused by the lower velocities in the flock. Most dominantly however is the fact that the swarm keeps a very stable formation, and robots tend to shield others from reaching their targets with their repulsive forces. In figure 13 this can be seen for robot number 5 (light green), assuming that the flock moves to the upper right where the fifth balloon is located. Robots 1 and 10 would prevent robot 5 from reaching its target. This represents a significant limitation of this setup. However, also the SFM has its own limitation; the cohesiveness of the swarm. Even though similar force terms are present, they are implemented in a more intricate way with exponential terms of which the parameters are difficult to tune. This results in unwanted behavior where sometimes no stable distance between the robots is maintained and they are overlapping in the simulation (crashing in reality). A dedicated tuning of parameters should be introduced to optimize the movements of small groups as presented in [5]. Such an improvement to the SFM could lead to better overall performance and is to be considered for a possible future stage of development.

2. Rendezvous

In the rendezvous problem, multiple agents start from different positions in a bounded environment and must meet at a common point. Each robot shares its information about a target and the group objective is to reach a general agreement on that particular target. In simple terms, multiple robots will all move together to a given position. The model is straightforward and we show it in the following.
We assume to have n robots in two-dimensional space (the model can be generalized). Let the position of robot i be expressed by the vector $x_i(t) \in \mathbb{R}^2$. In discrete-time, the position of robot i is updated as:

$$x_i(t+1) = x_i(t) + u_i(t) \tag{8}$$

where $u_i(t)$ is the control input at time t for robot i. For control inputs, we exploit a consensus algorithm to move the robots towards the average position of their neighbours. So, for robot i:

$$u_i(t) = k \sum_{j \in \mathcal{N}_i} (x_j(t) - x_i(t)) \tag{9}$$

where $\mathcal{N}_i$ is the set of neighbours of robot i and k a positive weighting factor. Finally, a convergence criterion must be set to stop the robots when they are close enough to their target. For our purpose, we have to slightly modify this model. We introduce a repulsion force, inversely proportional to the distance between the robots, to avoid collisions. This repulsive force exerted on robot i by robot j is as follows:

$$f_{rep}(x_i, x_j) = \frac{k}{||x_i - x_j||^2} \cdot \frac{x_i - x_j}{||x_i - x_j||} \tag{10}$$

where k is a positive weighting factor that determines the strength of the repulsion.
Then, instead of making the robots converge to their average, we want them to meet in a specific point that we decide. To do so, we incorporate an attraction term into the control law that guides towards a target position x^*:

$$u_i(t) = -k(x_i(t) - x^*) \tag{11}$$

where k is a positive weighting factor determining the strength of the attraction.
The corrected control input for robot i becomes:

$$\begin{aligned} u_i(t) =& k_1 \sum_{j \in \mathcal{N}_i} (x_j(t) - x_i(t)) - k_2(x_i(t) - x^*) + \\ &+ \sum_{j \neq i} f_{rep}(x_j(t), x_i(t)) \end{aligned} \tag{12}$$

with k_1 and k_2 being positive weighting factors. In figure 14 we show the result of this model.

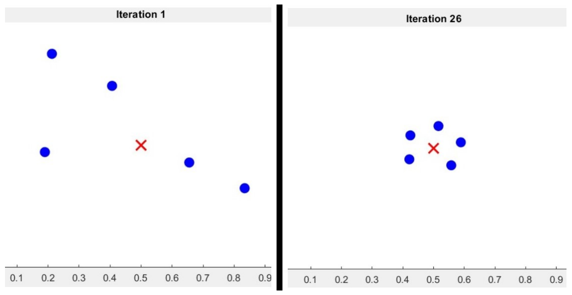


Figure 14: Rendezvous strategy

Having illustrated those two types of movement, we are ready to present the second sub-case of random exploration. We start in one of the predefined starting distributions. Similar to the rendezvous algorithm, the SFM inherently makes robots converge at their own center, if given as the next position to all agents, to form a crowd. As mentioned before, the SFM's behavior is less stable than the algorithm presented and could benefit from an integration. They then flock around looking for their balloons. Behaving in this way, when a target is discovered, the corresponding kid is already there to take it (i. e. it stops). In figure 15 we show one fast as well as one very slow outcome of this simulation.

As before, this completely exploratory approach turns out to not be very efficient. Robots may pass zones they've already visited more than once. Forming a crowd and covering a larger area of nearby positions may avoid close misses, but they still move without any sort of strategy. In the long run the kids will cover the whole space, but it may take several hundreds or thousands of steps. That is why we should introduce a way to keep track of their movements and the areas they have already visited.

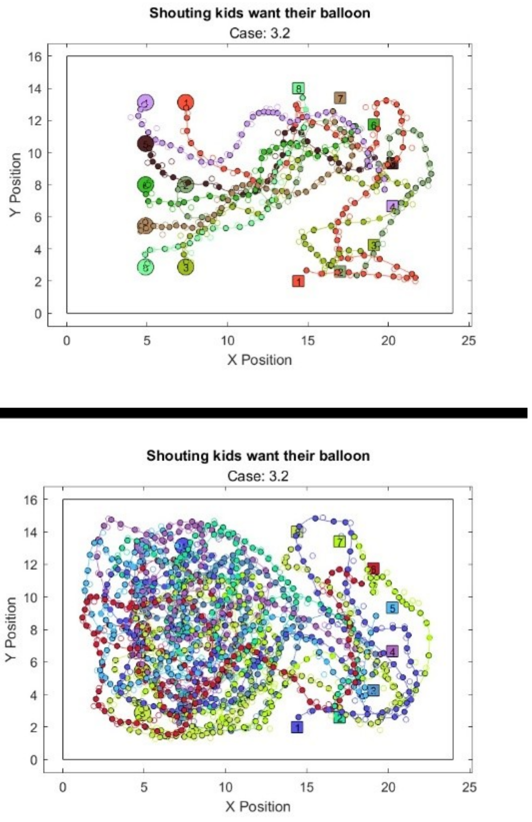


Figure 15: Case 3.2

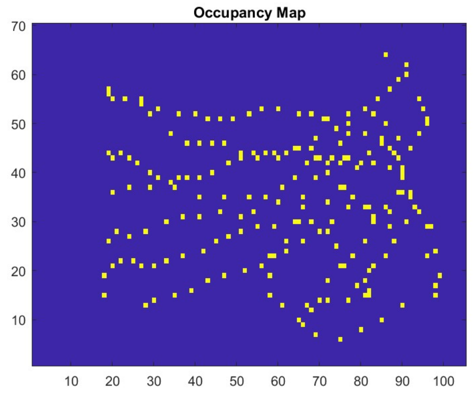
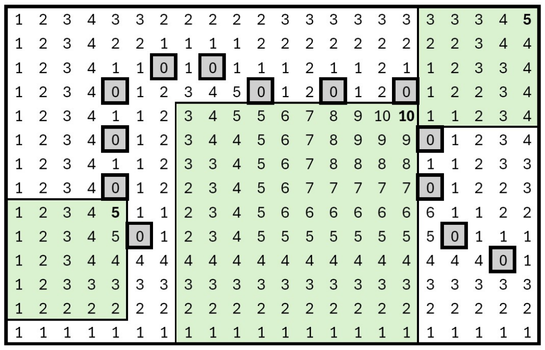


Figure 16: Occupancy map example

IV. Mapped Exploration

In a further stage of development, we want the group of robots to head towards the still unexplored areas in the environment. In practice, such a procedure could be achieved with e. g. SLAM (simultaneous location and mapping) sensors on the robots. For our simulation, however, we choose a simplified approach to building such an occupancy map 16.

We start with a zero matrix with dimensions five times larger than the size of the room for better resolution. On each iteration, the current positions of the robots are mapped onto the occupancy map and marked with a one. In a subsequent step, the largest unexplored regions in the occupancy matrix are determined and mapped onto another matrix with identical dimensions, also initially zero. We iterate only over the unexplored positions (zeros in the occupancy matrix) and assign to each cell the minimum value of the three adjacent cells (on its left, bottom-left, and bottom) increased by one. In this way, each cell contains the size of the empty square to its lower left, and the visited positions remain zero accordingly. The principle is illustrated in figure 17. Then, the center points of these resulting squares are calculated, and with a weighting scheme between the areas' sizes and their distance to the robots, the new destination is determined.



1	2	3	4	3	3	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	**5**
1	2	3	4	2	2	1	1	1	1	2	2	2	2	2	2	2	2	2	3	4	4
1	2	3	4	1	1	0	1	0	1	1	1	2	1	1	2	1	1	2	3	3	4
1	2	3	4	0	1	2	3	4	5	0	1	2	0	1	2	0	1	2	2	3	4
1	2	3	4	1	1	2	3	4	5	5	6	7	8	9	10	**10**	1	1	2	3	4
1	2	3	4	0	1	2	3	4	4	5	6	7	8	9	9	9	0	1	2	3	4
1	2	3	4	1	1	2	3	3	4	5	6	7	8	8	8	8	1	1	2	3	3
1	2	3	4	0	1	2	2	3	4	5	6	7	7	7	7	7	0	1	2	2	3
1	2	3	4	**5**	1	1	2	3	4	5	6	6	6	6	6	6	6	1	1	2	2
1	2	3	4	5	0	1	2	3	4	5	5	5	5	5	5	5	5	0	1	1	1
1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	0	1
1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 17: Working principle occupancy map

We combine all the strategies mentioned above, such as rendezvous and flocking, to guide all the robots together to the less frequented areas. Initially, we make the robots converge to their own center of mass for them to form a flock. Then, they swarm to the less visited areas according to the occupancy map explained earlier. Once at least one robot has arrived near the center of the square, the algorithm determines a new destination. When a balloon is discovered, the corresponding kid can reach it quickly because it's part of the flock. In figure 18, we show a simulation with this implementation.

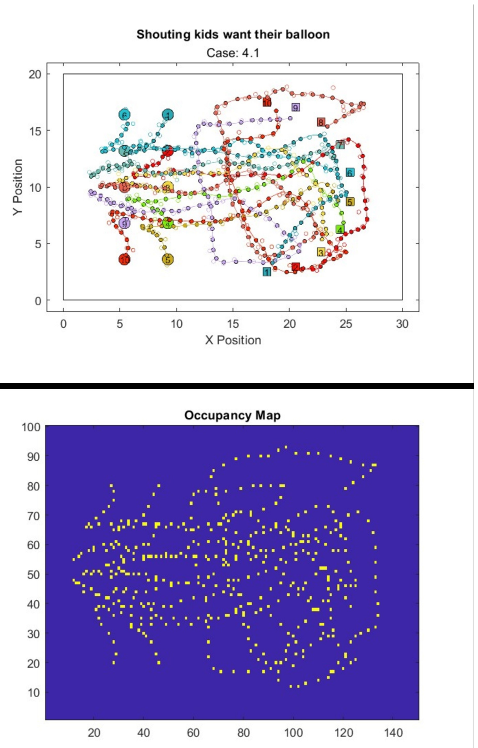


Figure 18: Rendezvous strategy with completion map

The kids manage to find all the balloons without wasting time returning to already visited positions. We see the convergence of their intentions as their trajectories overlap. Due to the initial distribution with all kids on one side, the algorithm makes them explore that zone before moving to the larger unexplored area. For the randomly distributed starting positions, as displayed in figure 19, the exploration is very fast and well distributed.

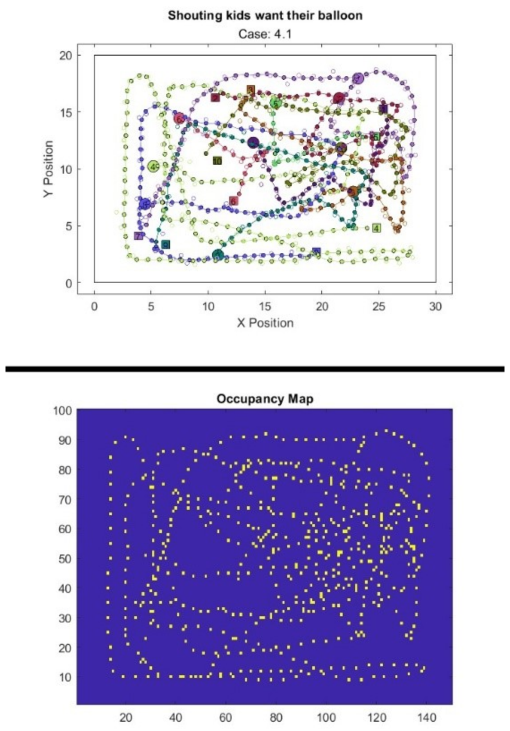


Figure 19: Rendezvous strategy with completion map

Sometimes children move away from a zone without fully exploring it because another area presents much more unexplored space. This is due to the fact that we know the room in advance, an assumption that doesn't necessarily hold true in unknown, real world environments. Further development of this case could be the fully autonomous exploration and mapping of the room by the robots themselves.

V. Statistical results

To prove the effectiveness of this project, we launch one hundred simulations of each case and collect the data to analyze the statistics. We use the same number of kids (eight) and the same initial distribution (grid-arc formation) to obtain comparable results. In particular, we are interested in the total distance travelled by every kid and the time needed to reach the objective. From the SFM we know the initial and final positions and velocities of every step and can thus linearly estimate the distance travelled. To obtain the time of a step, we simply divide the distance by the average velocity. The total time is computed as the sum of times of all iterations. We briefly remind about the cases' characteristics to help with following the subsequent analysis. Case 1 considers full knowledge of both kids and balloons features. In sub-case 1.1 actual positions are used, while in sub-case 1.2 the estimated ones. Case 2 leads every kid to the nearest balloon. In sub-case 2.1 there is no collaboration; instead, in sub-case 2.2, once a kid discovers a balloon that does not belong to him, the right kid is guided towards it. Case 3 is reserved to random exploration. In sub-case 3.1 we have egoistic explorers, in sub-case 3.2 kids try to move as a group to inspect larger areas together. Case 4.1 implements the occupancy map to keep track of already visited zones.

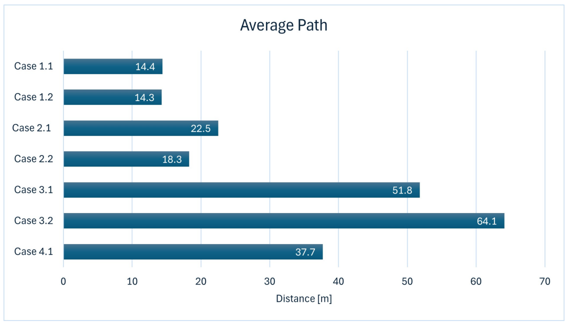


Figure 20: Average travelled path for each case

In figure 20, we report the average paths travelled by kids for each case. We immediately notice that sub-case 1.1 and sub-case 1.2 show pretty similar results, almost equal. This does not surprise us since in both situations the final destinations are provided. The little difference is related to the desired velocities randomly assigned at the beginning of every simulation. Assuming the same velocities for both sub-cases we expect that with the estimated positions the travelled paths are slightly longer. This is due to the curvature related to the uncertainties in contrast to straight lines when knowing the actual positions.

Comparing sub-cases 2.1 and 2.2, we can see a relevant improvement when kids communicate the identities of discovered balloons. In fact, case 2.2 avoids useless recognition steps that can occur in case 2.1. Once a balloon is identified, the corresponding owner is immediately called there and does not waste time and breath to verify other balloons. It is to be mentioned that these results are obtained with a grid-arc distribution of kids and balloons. Especially at the beginning, multiple kids converge on the same closest balloon. Considerably better results are expected when kids and balloons are spread randomly across the room since it is less probable that more kids are attracted by the same unknown balloon. The more kids initially discover a unique balloon, the faster correct matches can be assigned.

Cases 3.1 and 3.2 show, instead, some anomalies. Here we expected to see relevant improvements when the swarm behaviour is introduced. But the average path of kids of case 3.2 to complete the simulation exceeds the one of the first sub-case. One possible explanation is the sub-optimal flocking behavior obtained with our implementation of the SFM. However, another aspect must be taken into account: the kids arriving last. When only one or two kids remain without their balloons, the flocking effect is lost and the last kid may take long to arrive at destination. To investigate this, we sort the data from fastest to slowest to arrive for all of the hundred simulations. In figure 21 are the results.

It clearly appears that in sub-case 3.1 the last kid influences the average time a lot while the other kids generally arrive quickly. Instead, for the flocking behavior of sub-case 3.2 performances are affected even more significantly by the slowest kid. Computing the standard deviation for all cases, we observe that only

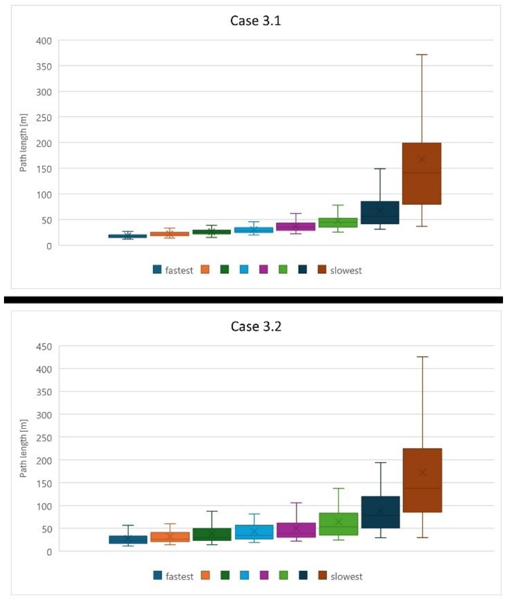


Figure 21: Path length from fastest to slowest kids in Case 3 (outliers not included)

for sub-cases 3.1 and 3.2 it exceeds the mean. It seems that the very high values of the last kids weigh too much on the computation. This induces us to not consider the average as truly representative of the behavior of this case. The median obtained is 32.1 m for the sub-case 3.1 and 40.9 m in sub-case 3.2. Ranges are respectively 823.2 m and 663 m. In C we depict the histograms of both sub-cases.

Case 4.1 confirms our intuition that the introduction of an occupancy map will considerably lower the average path obtained with the random exploration. The result targets the gap between full knowledge cases and wandering ones. Even in this case, sometimes the problem with the last kids occurs. With just one or two kids, the process of fulfilling the occupancy map may slow down 22. The standard deviation is 20.2 m, so a considerable but not huge spread. The median in this case lays at 31.3 m, while the range results at 168.4 m. The related plots showing the total travel times are available in B.

As a last observation of all cases we can look at figure 23. A few runs are shown in different colours

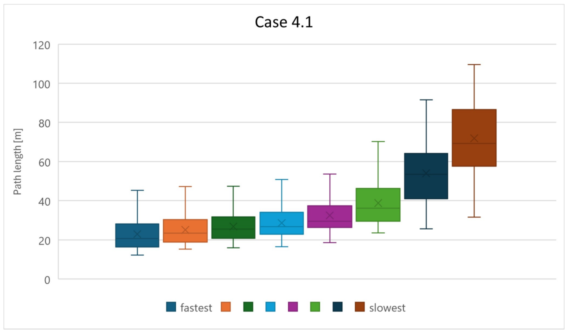


Figure 22: Path length from fastest to slowest kids in case 4.1

but the data is not sorted anymore from first to last. A high peak on the left thus means that in this specific run kid number one covered a long distance. As we proceed from left to right over the total of our seven cases the differences between the simulations increase. For the first cases we have almost always the same behaviors. In case 3 we have the highest peaks, while with the occupancy map we return to a more balanced situation.

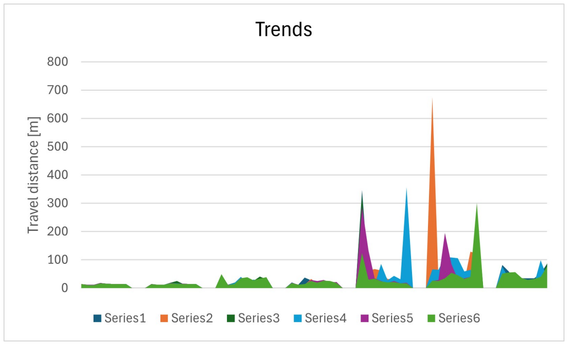


Figure 23: Multiple runs of different cases

VI. Conclusions

In this report we presented a logical approach to show the effectiveness of consensus algorithms for a collaborative task inspired by a social experiment. From choosing the optimal set of sensors to ensure reliable estimation of a position, to moving robots in a coordinated manner and making them converge on a desired point, consensus algorithms have demonstrated their ability to reduce the time and distance traveled to complete a search mission. We explored different types of cooperation and information sharing behaviors as strategies for direct guidance to a specific target and to avoid useless returns to already visited spaces. We discussed the limitations of the models used and proposed solutions to overcome them. We introduced modifications to established models to make them suitable for our specific case. We proposed a suitable hardware implementation consisting of the mobile robots and the communication infrastructure. As a future development of this work, we propose to autonomously build the environment map under the influence of other sensor noise. For example, adding a real robot model with accelerometers, gyroscopes, among others will make the system more realistic; a final development step could deal with the complete removal of the room sensors altogether.

References

[1] Elisa Bassoli and Loris Vincenzi. Parameter calibration of a social force model for the crowd-induced vibrations of footbridges. *Frontiers in Built Environment*, 7, 2021.

[2] eu.robotshop.com. `https://eu.robotshop.com/fr/products/kit-robotique-compacte-3-roues-omnidirectionnelles-compatible-arduino`, 2023. Accessed: 2024-06-06.

[3] Daniele Fontanelli. slides_b4_rte_and_wireless, 2023/2024.

[4] Daniele Fontanelli. slides_d3_distributed_estimation, 2023/2024.

[5] Lin Huang, Jianhua Gong, Wenhang Li, Tao Xu, Shen Shen, Jianming Liang, Quanlong Feng, Dong Zhang, and Jun Sun. Social force model-based group behavior simulation in virtual geographic environments. *ISPRS International Journal of Geo-Information*, 7(2), 2018.

[6] Joel George Manathara and Debasish Ghose. Rendezvous of multiple uavs with collision avoidance using consensus. *Journal of Aerospace Engineering*, 25(4):480–489, 2012.

[7] Giuseppe Notarstefano and Francesco Bullo. Distributed consensus on enclosing shapes and minimum time rendezvous. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 4295–4300. IEEE, 2006.

[8] Hyongju Park and Seth A. Hutchinson. Fault-tolerant rendezvous of multirobot systems. *IEEE Transactions on Robotics*, 33(3):565–582, 2017.

[9] Varun Amar Reddy and Gordon L. Stüber. Multi-user position estimation and performance trade-offs in ieee 802.11az wlans. 2023.

[10] Craig Reynolds. Flocks, herds and schools: A distributed behavioral model. *Computer Graphics*, 1987.

[11] Jonathan Segev and Roy Want. Newly released ieee 802.11az standard improving wi-fi location accuracy is set to unleash a new wave of innovation, 2023. Accessed: 2024-06-06,

AUTHORS FINAL STATEMENTS

The authors have the following addresses:

Italy, jacopo.endrizzi-1@studenti.unitn.it.

Italy, thadekilian.pfluger@studenti.unitn.it

This report is the final document for the course of “Intelligent Distributed Systems”.

APPENDIX

A. Other distributions

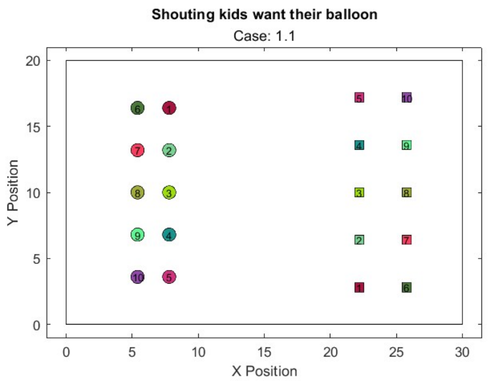


Figure 24: Grid-grid distribution

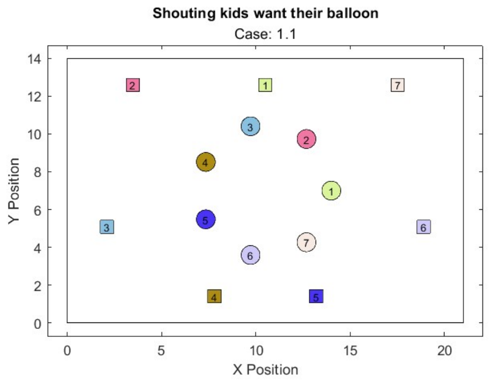


Figure 26: Circular distribution

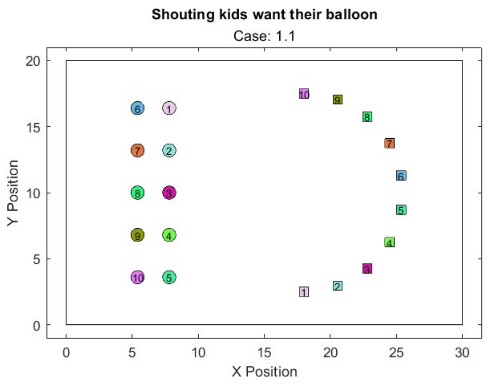


Figure 25: Grid-arc distribution

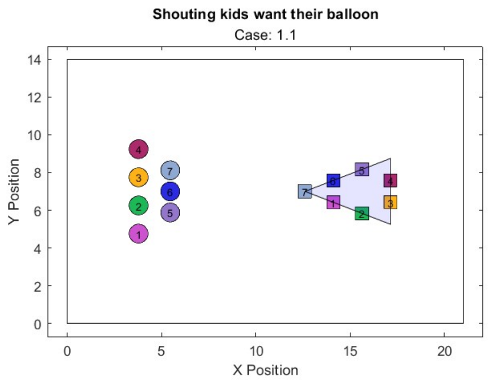


Figure 27: Triangular distribution

B. From fastest to slowest kid

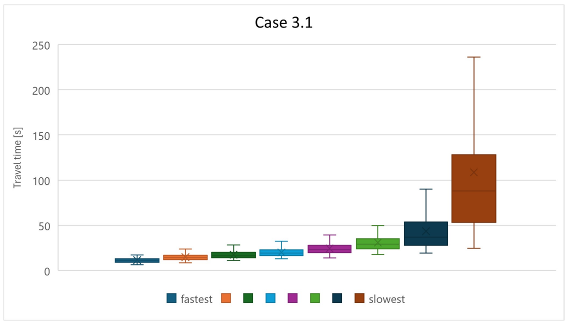


Figure 28: Case 3.1

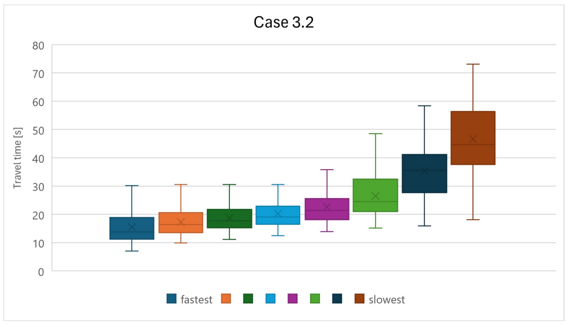


Figure 29: Case 3.2

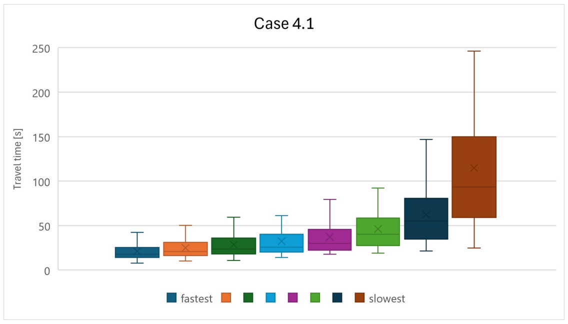


Figure 30: Case 4.1

C. Histograms

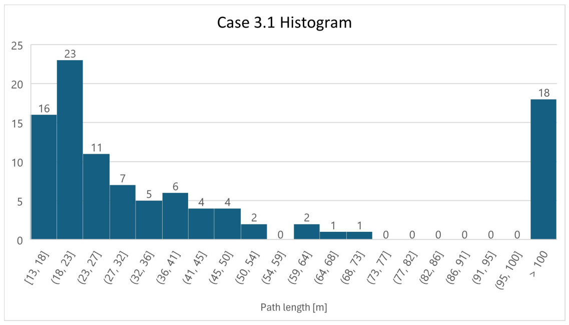


Figure 31: Histogram Case 3.1

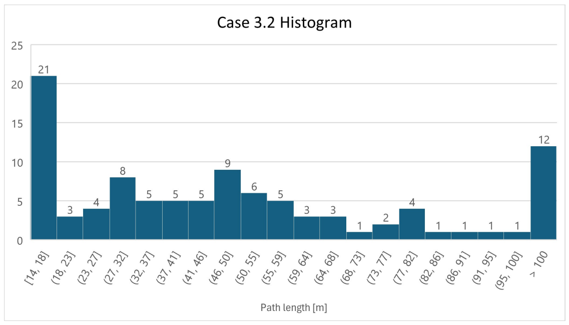


Figure 32: Histogram Case 3.2