BotArena, a new affordable experimentation platform for multiple mobile robots

Franck Mercier, Rémy Guyonneau, Alain Godon

Mechatronics 106 (2025) 103295

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

BotArena, a new affordable experimentation platform for multiple mobile robots☆

Franck Mercier [a,*], Rémy Guyonneau [a], Alain Godon [b]

[a] Univ Angers, LARIS, SFR MATHSTIC, F-49000 Angers, France
[b] Univ Angers Polytech, F-49000 Angers, France

ARTICLE INFO

Dataset link: https://gitlab.u-angers.fr/botarena/ba_conception

Keywords:
Swarm robotics
Mobile robotic
Multi-agent
Hardware in the loop
Robotarium
Swarm arena

ABSTRACT

This paper presents the design of an experimental hardware-in-the-loop platform for multiple mobile robots that meets three criteria: cost reduction, modularity, and versatility. Inspired by structures like the Robotarium, the BotArena aims to be a low-cost, open source, and open hardware version of an arena for multiple mobile robot experiments. The arena was mainly designed with a focus on research applications, to allow the implementation of algorithms and their experimental evaluations. However, it was also designed with applicability to educational applications, for the training of future engineers in swarm robotics techniques and methods. Finally, it was also designed for use in science popularization events. In a nutshell, this platform is intended to be inclusive and appealing for introducing multiple mobile robot concepts. The paper details the design, realization, and underlying principles of this tool. Finally, the performance of the arena (response time and localization accuracy) is quantified and discussed, and an ant-inspired application is presented.

1. Introduction

1.1. Context

Swarm robotics is a field of robotics that draws inspiration from the self-organized behaviors of social organisms such as animals and insects. Through simple rules and local interactions, swarm robotics aims to achieve complex, robust, and flexible behaviors via the coordination of a large number of robots [1]. The underlying idea is that tasks can be collectively accomplished that would be unattainable for an individual robot. Some bio-inspired examples, such as the ability of ants to find the shortest path [2] or the organization of schools of fish, demonstrate "collective intelligence", based on simple local rules of interaction [3].

Swarm robotics is a relatively recent field of research inspired by swarm intelligence applied to robotics [4]. Although research in this topic has increased over the last two decades with significant progress, real applications of robot swarms are still rare, as are experiments with actual robots [5]. That is, applications can be observed [6] in medical robotics, meteorology, or in the field of entertainment where swarms of drones give rise to new large-scale entertainment.[1] Nevertheless, the number of applications remains small compared to the numerous applications for which swarm robotics is interesting:

- Tasks that require monitoring across a geographical area, such as forest fires, pollutant leaks, or agricultural crop inspection;
- Tasks that are too dangerous for humans or for individual, complex, and costly robots, such as clearing a mine corridor by allowing robots to exhibit sacrificial behavior;
- Tasks that require scalability or redundancy.

Whether for research or educational purposes, there are few platforms allowing experimentation for swarm robotics approaches [7,8]. A reference for this type of installation is the Robotarium [9,10], an enclosure dedicated to research, which researchers from all countries can use remotely. This is a platform with a swarm of simple and inexpensive robots (around fifty dollars per robot) whose positions are processed by a set of cameras. The robots communicate via Wi-Fi (Wireless Fidelity) and perception information is provided by a supervisor. However, although the robots are cheap, the Robotarium required a significant investment of $2.5 million[2] and remains restricted to the world of research. Faced with the development of swarm robotics, the need for an experimental platform and the lack of available solutions, it was decided to create a new low-cost and modular system for experimenting with multiple mobile robots. This new platform, named "BotArena", is not designed for experiments with hundreds of robots (but rather small

☆ This paper was recommended for publication by Associate Editor Oliver Sawodny.

* Corresponding author.
E-mail addresses: franck.mercier@univ-angers.fr (F. Mercier), remy.guyonneau@univ-angers.fr (R. Guyonneau), alain.godon@univ-angers.fr (A. Godon).

[1] https://youtu.be/44KvHwRHb3A

[2] https://www.atlantamagazine.com/news-culture-articles/georgia-tech-robotarium-shining-beacon-robotic-awesomeness/

https://doi.org/10.1016/j.mechatronics.2025.103295
Received 17 June 2024; Received in revised form 26 December 2024; Accepted 13 January 2025
Available online 23 January 2025

dozens), so it will be referred to as a multiple mobile robot system instead of a swarm robot system.

This paper presents a proof of concept of a low-cost multi-robot arena that can be used for education, popularization, and research. The objective of this work is to enable research institutions to equip themselves at low cost with an experimental platform that allows for the implementation and evaluation of multiple mobile robot algorithms. The authors believe that this platform can help to stimulate research and demonstrations in the field of robotics, particularly in swarm robotics. Furthermore, the authors contend that it is important to train the next generations of engineers in this area and to raise their awareness of swarm robotics. Finally, the authors assert that the popularization of a given topic has a positive impact on the associated research field [11]. The availability of a demonstration platform that can be easily moved and installed serves as an effective tool for participation in popularization events. To the best of the authors' knowledge, such a multi-robot platform represents a novel contribution in comparison to existing platforms.

The following section presents the different constraints identified during preliminary thinking and the adopted solutions.

1.2. Constraints and architecture of the arena

The project was built around three main constraints:

- Cost: The BotArena must remain financially accessible to most research and academic institutions;
- Modularity: Given the diverse contexts in which the BotArena will be deployed (e.g., scientific experiments, educational activities, demonstrations), it is crucial that the system's dimensions can be easily adjusted. Furthermore, as these use cases may occur in different locations, the arena, either in whole or in part, must be easily transportable;
- Versatility: Given the wide range of potential use cases, the system must be capable of addressing various challenges in mobile robotics (e.g., exploration, mapping, swarm intelligence). The robots must therefore be adaptable.

The overall system is based on a principle called hardware-in-the-loop (HIL) [12,13]. HIL is a crucial methodology in engineering and technology. In swarm robotics it integrates physical robotic agents with computer simulations to form a dynamic and realistic testing environment. This enables researchers and engineers to evaluate the performance of robot swarms in simulated scenarios, encompassing complex multi-agent coordination and environmental interactions. HIL not only enhances the development and testing phases of swarm robotic systems but also enables the validation of control algorithms and hardware components. Additionally, it serves as an initial step toward ensuring the reliability and efficiency of robotic swarms before deployment in real-world applications. More details about the HIL aspect of the BotArena are presented in Section 2.3.

To satisfy the cost constraint, it was decided, similarly to the Robotarium, to create "simple" robots. These robots are presented in Section 2.1, although it is noteworthy that they do not have any sensors but are equipped with visual markers allowing cameras to identify and locate them. To locate the robots, the system relies on inexpensive camera systems (Raspberry Pi.[3] based) and free software tools (OpenCV, Flask). The approach to identify and locate the robots is detailed in Section 2.2 To satisfy the modularity constraint by allowing an easy reconfiguration of the arena a simple and classical calibration process is detailed in Section 2.2. Finally, a supervisor (another Raspberry Pi) is used to manage everything. It processes the commands of the robots according to simulated sensor data and the expected behavior for each robot. This supervisor is presented in Section 2.3.

The structure of the paper is as follows: Section 2 describes the arena, the robots, and the localization process. Section 3 covers experiments conducted with the BotArena to test the system performances and present an use case, and Section 4 concludes the paper.

2. The arena: design and operation

The platform consists of three main components: the robots, the cameras, and the supervisor. The robots are described in Section 2.1, the cameras are discussed in Section 2.2, and the supervisor is addressed in Section 2.3.

The general concept of the arena is as follows: the cameras detect the robots' positions and orientations and transmit this data to the supervisor. The supervisor then generates control commands and transmits them to each robot. All system components communicate via a Wi-Fi network.

Fig. 1(a) illustrates the hardware configuration. In the figure, only two cameras are depicted, but this does not limit the system: cameras can be added to scale the arena at will, in accordance with the modularity criteria defined in Section 1.2. Fig. 1(b) provides an overview of the general operation of the arena. This is a HIL approach as the sensor data for the robots are simulated, and the behavior of the robots is computed by the supervisor and not by the robots themselves as depicted in Fig. 2. Thus, there is a simulated part and a hardware part.

2.1. Robots: design and operation

It has been decided to design new robots to meet the objectives of the arena. Here are the defined constraints for the robots:

- Low cost: when considering multiple mobile robots, the individual cost of each robot is therefore relevant;
- Reduced size: similar to cost, the individual size of each robot is important when planning to deploy several robots in the same arena;
- Incorporating a top-mounted marker: necessary for detection by overhead cameras;
- Differential kinematics: enabling maximum mobility in a restricted space;
- Wi-Fi communication to facilitate interactions between the robots and the supervisor.

The following sections detail the design process to meet these constraints. The interested reader can check the dedicated GitLab repository.[4] The chosen materials (references and suppliers), mechanical design (SolidWorks CAD - Computer-Aided Design), and electronic design (Eagle CAD) are freely available. Fig. 3(a) shows the result of the CAD design.

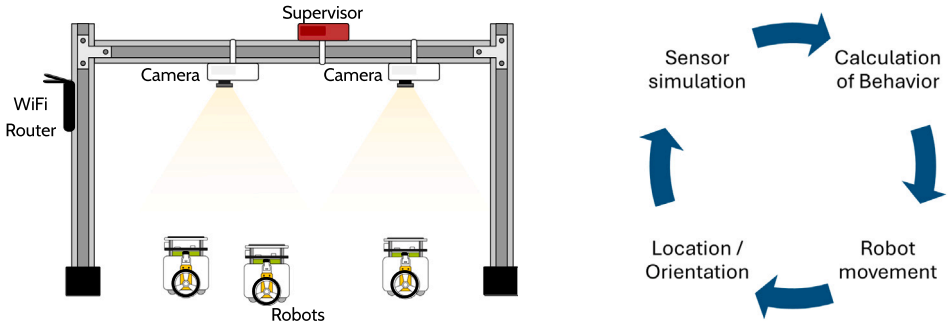
2.1.1. Hardware design

The robots are built around an ESP32 micro-controller, which integrates a standard micro-controller along with Wi-Fi and Bluetooth communication capabilities. This choice was made to minimize the final size of the robot. To facilitate programming the micro-controller, a programmer was integrated directly into the robot, although the use of an external programmer could have simplified the on-board electronics. The academic context of the BotArena justified this choice, similar to the approach taken with the Arduino platform, for example. Besides, it can be noticed that the ESP32 can be programmed with the Arduino IDE (Integrated Development Environment), as it can be for the robots of the BotArena.

To fine-tune robot movements, encoders are used on each of the two motors, and their data are processed by a digital PID controller (Proportional–Integral–Derivative) implemented on the ESP32. Several

[3] https://www.raspberrypi.org/

[4] https://gitlab.u-angers.fr/botarena/ba_conception



(a) Hardware setup of the BotArena (b) Operation process of arena

Fig. 1. The BotArena overview: The cameras locate the robots, send the poses information to the supervisor, the supervisor simulates sensor data, computes the robot's commands, and sends them to the robots, the robots move accordingly to the commands.

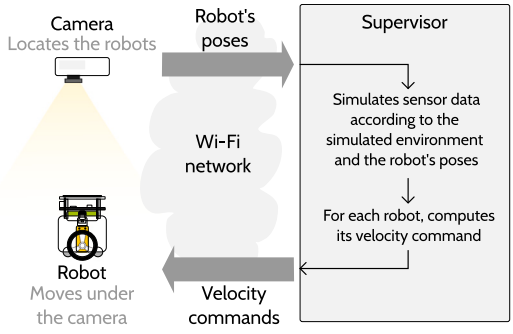


Fig. 2. The HIL architecture of the Botarena.

methods exist to adjust the parameters of this type of controller; the method used here is inspired by Åström and Hägglund [14].

For the choice of batteries to power the robots, a 900 mAh nickel-metal hydride (NiMh) solution has been chosen and provides a minimum of 3 h of operation. The design team's experience in mobile robotics prompted the decision to discontinue the use of Lithium polymer (LiPo) technology due to its instability over time, particularly with respect to morphological changes, as highlighted in [15], which are primarily caused by prolonged periods of inactivity. A detailed overview and comparison of battery technologies can be found in [16]. When using a battery, a management system is necessary [17], here the battery voltage is brought back via a divider bridge to an analog-digital conversion input of the micro-controller.

Several non-optimal choices from a cost point of view were made in order to increase the robustness of the robot. This is the case of the H-bridges for controlling the motors which are oversized, as well as the choices of a very robust USB connector, or the battery previously mentioned, which slightly increase the cost of the robot while remaining correct. However, as the arena is open hardware, anyone could update the robot components to meet their need.

To conclude with the mechanical design, it is possible to note the use of angle gear motors in order to limit the bulk and facilitate the positioning of the magnetic encoders. These motors are sized for a maximum linear robot speed of 30 $cm\ s^{-1}$ and angular speed of approximately 100 rpm. The differential kinematics of the robot is modeled as:

$$\begin{bmatrix} v_r \\ v_l \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} V \\ \omega \end{bmatrix} = \frac{1}{2R} \begin{bmatrix} 2 & L \\ 2 & -L \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix}, \tag{1}$$

with v_r and v_l respectively the rotation speeds of the right and left wheels, V and ω the linear and angular speeds of the robot, L the distance between the wheels and R the radius of the wheels (L = 10 cm and R = 3 cm, schema on Fig. 3(b)). $\mathbf{J}$ corresponds to the Jacobian of the system [18]. It is this relation (1) that is implemented in the low-level functions for the mobility of the robot.

2.1.2. Embedded software

The commands of the robots are computed by the supervisor as detailed in Section 2.3. Thus, the only information the robots receive are linear and angular speeds. This behavior makes it possible to simplify the programming of robot behaviors, but could very well be modified to directly integrate the decision algorithms into the robots (this is not the purpose of this article).

At the robot level, the software can be seen as an extremely simple loop composed of two operations depicted in algorithm 1.

Algorithm 1: Robot main loop

```
1 initialization;
2 while 1 do
3 |   botNetwork();
4 |   botOrganizer();
```

The function $botNetwork()$ gets the speed command over the network via TCP sockets.

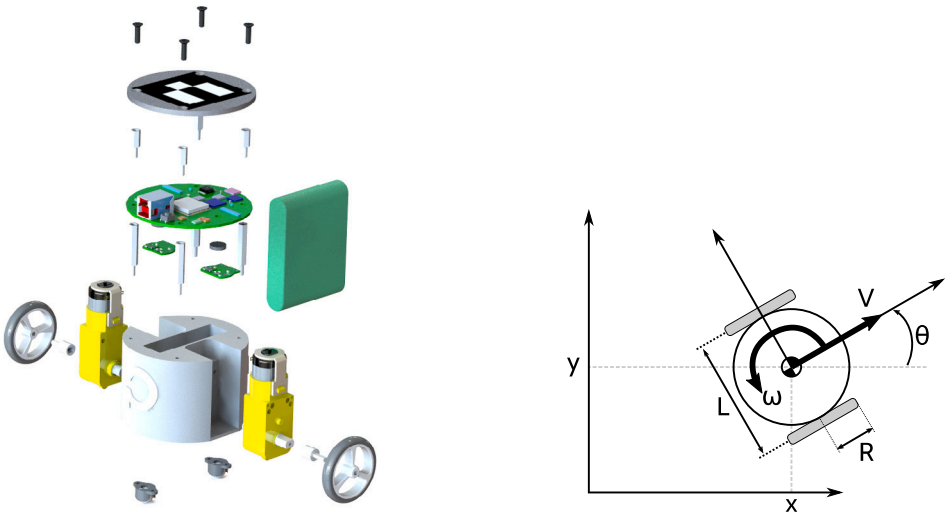
The commands received are not executed directly but are stored in a linked list which is processed by the $botOrganizer()$ function. The main advantage of this architecture is its ability to easily create scenarios that are executed according to a predefined timing.

This operation is non-blocking and reduces the individual cycle time of the different functions, ensuring good responsiveness of the robots. Furthermore, this allows "stacking" orders for further processing.

To the aforementioned architecture are added three low-level functions, managed via interrupts to ensure accurate execution times. These functions can be activated by an on/off type request and associated parameters can be modified. Implemented using interrupts, these functions remain fully transparent to the user:

- A function managing the speed control (the most complex, and thus presented here);
- A function that enables a speed ramp with adjustable acceleration and deceleration coefficients for smooth movement, avoiding jolts that unnecessarily strain the mechanics;
- A *watchdog*, a security feature to stop the robot in the event of a loss of connection.

To implement the speed control function, accurate motor rotation speed measurements are required. These are gathered through the encoder edges, which require two interrupts per motor. Each encoder has two channels (A and B), and an interrupt is triggered on each edge. The direction of rotation is determined by the state of the channels,



(a) Exploded CAD view (b) Kinematics

Fig. 3. The robot.

while the speed is measured using a counter whose value corresponds to the frequency of interruptions.

Calling the function that directly commands the servo-motors is done through a timer generating an interrupt. The accuracy of this time interval is crucial, as it is used in calculations for angular speed and derivatives.

The angular velocity is computed according to Eq. (2):

$$\omega_{lr} = \frac{\text{counter} \times \text{angular resolution} \times \pi}{\mathrm{d}t \times 180} \tag{2}$$

With ω_{lr} in deg s^{-1}, and lr the right or left wheel index. Angular velocity is converted into linear velocity with Eq. (3):

$$V_{lr} = \omega_{lr} \times R \tag{3}$$

With V the linear speed in m s^{-1}. The error ec is thus defined by the difference with the speed reference which allows the resolution of the corrector Eq. (4):

$$C_{pid} = K_p \times \left(ec + \frac{1}{T_n} \int ec \times \mathrm{d}t + T_v \frac{\mathrm{d}(ec)}{\mathrm{d}t} \right), \tag{4}$$

with $K_i = \frac{K_p}{T_n}$, $K_d = K_p \times T_v$, C_{pid} the command (output of the PID corrector) and ec the error (difference between the expected speed – the setpoint – and the measured speed). The main difficulty when implementing a PID corrector it the adjustment of the coefficient values K_p, K_i and K_d. There are many methods to evaluate those parameters; the one chosen here is the CHR method (Chien, Hornes and Reswick) [14] that evaluates each coefficient from the index response of the system, an example is implemented in [19].

2.2. Localization system

2.2.1. Hardware and software architecture

In order to meet the low-cost and modularity constraints, a decision was made to design a multi-camera system using low-cost cameras. Raspberry Pi paired with PiCam[5] cameras were selected. Each connected camera (that is, a Raspberry Pi board with a PiCam) is placed high up to detect the markers attached to the top of the robots. Consequently, the detected position and orientation of the markers corresponds to the robots. Each camera module is connected to the Wi-Fi network and runs a TCP/IP socket server (based on Flask). On request, the cameras send the positions and orientations of the visible robots in JSON (JavaScript Object Notation) format.

Tracking via coded targets is a method frequently used in augmented reality applications and robot localization that, in general, outperforms methods based on natural descriptor extraction. It is easy to implement, and many libraries such as ARTool-Kit [20], very popular in the scientific literature, Cybercode [21], InterSense [22], ARTag [23], ALVAR, or ArUco [24] provide the ability to detect these targets, identify them, and easily estimate their pose (position and orientation). ArUco markers were chosen for being more recent and natively implemented in Open-CV, the library used here. Furthermore, according to [25], ArUco markers provide an effective balance between speed, flexibility, and accuracy in position and orientation detection.

The configuration of the cameras requires three typical steps: the evaluation of the intrinsic matrix (to be done once for each camera), the computation of the camera's position relative to the global arena reference (to be done each time the camera is moved), and the detection and localization of the robots in the arena frame. These three steps are detailed in the following subsections.

2.2.2. Calibration — intrinsic matrix and extrinsic matrix

The camera calibration involves two steps: the estimation of the intrinsic matrix and the computation of the position of the camera relative to the global arena reference (extrinsic matrix). The global arena frame will be referred to as the world frame in the following sections.

To compute the intrinsic matrix (as well as the distortion coefficients), the classic approach of OpenCV based on a chessboard is used.[6] Since this calibration depends solely on the camera and not on the scene, it can be performed once for each camera.

According to the pinhole model [26], the relationship between the camera pixels and the world frame can be expressed as:

$$\mathbf{p} = \mathbf{A}[\mathbf{R}_{w,c}|\mathbf{t}_{w,c}]\mathbf{P}_w, \tag{5}$$

where $\mathbf{p} = (u, v)$ are the coordinates of the pixel, $\mathbf{A}$ is the intrinsic matrix, $[\mathbf{R}_{w,c}|\mathbf{t}_{w,c}]$ is the transformation matrix from the world frame to the camera frame (with $\mathbf{R}_{w,c}$ being the rotation matrix and $\mathbf{t}_{w,c}$ the translation vector), and $\mathbf{P}_w$ represents the coordinates of the object in the world frame. Computing the extrinsic matrix (the pose of the camera in the world frame) involves evaluating $[\mathbf{R}_{w,c}|\mathbf{t}_{w,c}]$.

[5] https://www.raspberrypi.com/documentation/accessories/camera.html

[6] https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

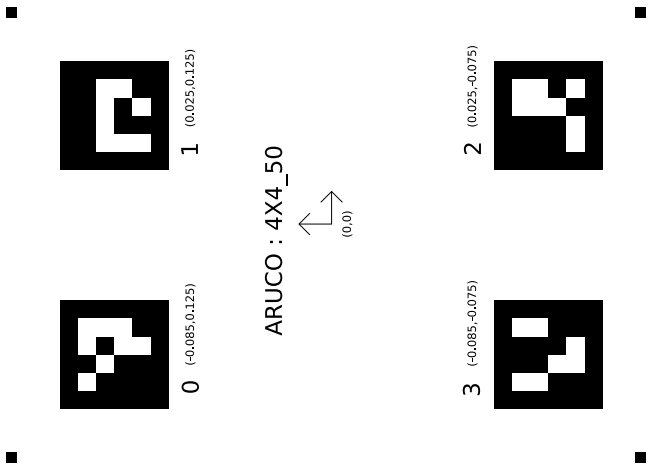


Fig. 4. Example of pattern to compute the extrinsic matrix.

This estimation can be achieved through several approaches. For instance, the poses of the camera could be predetermined using computer-aided design (CAD) software and precise camera placement, or one could measure those poses using precise external tools like laser measurements. However, as previously mentioned, modularity is a crucial criterion for the Botarena, as it must be adaptable in size depending on user requests, location, or context of use. Therefore, changes must be simple and quick: extensive calibration steps cannot be considered. It has been decided to implement a fast estimation of the camera poses by utilizing a printed marker pattern with known poses (this can easily be done using drawing software like Inkscape, for instance). An example of a pattern with 4 markers is shown in Fig. 4. Thus, it is only necessary for a camera to locate 4 markers to estimate its pose (position and orientation). Larger patterns with more markers can be designed to calibrate cameras over a broader environment, as long as the marker positions are known in the world frame and each camera can locate at least 4 markers. This approach facilitates camera calibration: the cameras are mounted facing downward to detect the markers, the reference pattern is then placed in the arena so that each camera can see at least 4 markers, and calibration is carried out for each camera. Thanks to this method, the entire arena setup can be completed in about 15 min, from mounting the arena structure to localizing the robots in the world frame.

2.2.3. Estimate the robot's positions

Once the calibration has been completed and the extrinsic matrix has been computed, it is then possible to locate the robots (i.e., the corresponding ArUco markers) in the world frame.

To achieve this, the first step is to detect the ArUco markers in the image and estimate their positions. This is accomplished using built-in OpenCV functions that provide a rotation vector and a translation vector for each detected marker. It is assumed that a marker is at the origin of its own reference frame: $\mathbf{P}_m^j = (0,0,0,1)^T$, where $\mathbf{P}_m^j$ denotes the position of marker j in its own frame. Using OpenCV, it is also possible to compute $\mathbf{R}_{m,c}^j$ and $\mathbf{t}_{m,c}^j$, which represent the rotation matrix and the translation matrix, respectively, that allow transformation from the marker frame to the camera frame (utilizing the results of the calibration):

$$\mathbf{P}_c^j = [\mathbf{R}_{m,c}^j|\mathbf{t}_{m,c}^j]\mathbf{P}_m^j, \tag{6}$$

with $\mathbf{P}_c^j$ the coordinates of the marker j in the camera frame. Using the extrinsic matrix it is possible to compute:

$$\mathbf{P}_w^j = (x_w, y_w, z_w)^T = [\mathbf{R}_{w,c}^j|\mathbf{t}_{w,c}^j]^{-1}[\mathbf{R}_{m,c}^j|\mathbf{t}_{m,c}^j]\mathbf{P}_m^j, \tag{7}$$

with $\mathbf{P}_w^j$ the coordinates of the marker j in the world frame and

$$[\mathbf{R}_{w,c}^j|\mathbf{t}_{w,c}^j]^{-1}[\mathbf{R}_{m,c}^j|\mathbf{t}_{m,c}^j] = \begin{bmatrix} r_{m,w}^{x,x} & r_{m,w}^{x,y} & r_{m,w}^{x,z} & t_{m,w}^x \\ r_{m,w}^{y,x} & r_{m,w}^{y,y} & r_{m,w}^{y,z} & t_{m,w}^y \\ r_{m,w}^{z,x} & r_{m,w}^{z,y} & r_{m,w}^{z,z} & t_{m,w}^z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{8}$$

with $[\mathbf{R}_{w,c}^j|\mathbf{t}_{w,c}^j]$ the transform from the world to the camera frame and $[\mathbf{R}_{m,c}^j|\mathbf{t}_{m,c}^j]$ the transform from the marker to the camera frame. Thus it is possible to compose the pose of the marker in the world frame (and therefore the pose of the robot):

$$\begin{bmatrix} x^j \\ y^j \\ \theta^j \end{bmatrix} = \begin{bmatrix} x_w \\ y_w \\ \text{atan2}(r_{m,w}^{y,x}, r_{m,w}^{x,x}) \end{bmatrix} \tag{9}$$

with (x^j, y^j) the position of the robot attached to the marker j in the world frame and θ^j its orientation.

2.3. Supervisor

The supervisor is the central element of the platform. Fig. 5 depicts the functional schematic of the Botarena. This centralized architecture enables the platform to simultaneously meet the three constraints of low cost, modularity, and versatility. The supervisor manages three main tasks:

- Camera administration;
- Experiment configuration;
- Robot sensors and codes (i.e., the computation of the robot commands).

The reader will notice that the BotArena has a centralized architecture that may differ from some decentralized concepts of swarm robotics. Indeed, the supervisor controls the entire arena. However, this configuration makes it possible to simulate a decentralized behavior for the robots: within the supervisor, the commands of each robot are processed separately as they would be in each robot. This choice has been made to simplify robot programming while meeting the low-cost constraint.

To facilitate the use of the arena, the supervisor relies on a web server implemented with Flask.[7] This server runs on a dedicated Raspberry Pi (referred to as the supervisor) and allows access to the various functionalities of the system through any web browser, eliminating the need for a heavy client. The three following subsections detail the three points managed by the supervisor.

2.3.1. Camera administration

One of the requirements for the platform is modularity: it must be possible to easily modify the configuration of the arena, including its surface area, the number of cameras, and their placement. The supervisor provides access to the cameras' functionalities, such as the calibration process detailed in Section 2.2.

To achieve this, the supervisor facilitates image retrieval from a specified camera, records the necessary images for the calibration process, computes the intrinsic matrix, stores the image to position the camera within the world reference frame, computes the extrinsic matrix, visually verifies the consistency of the computed matrices, and checks the robots detected by the camera. All of this is done via TCP/IP sockets.

[7] https://flask.palletsprojects.com

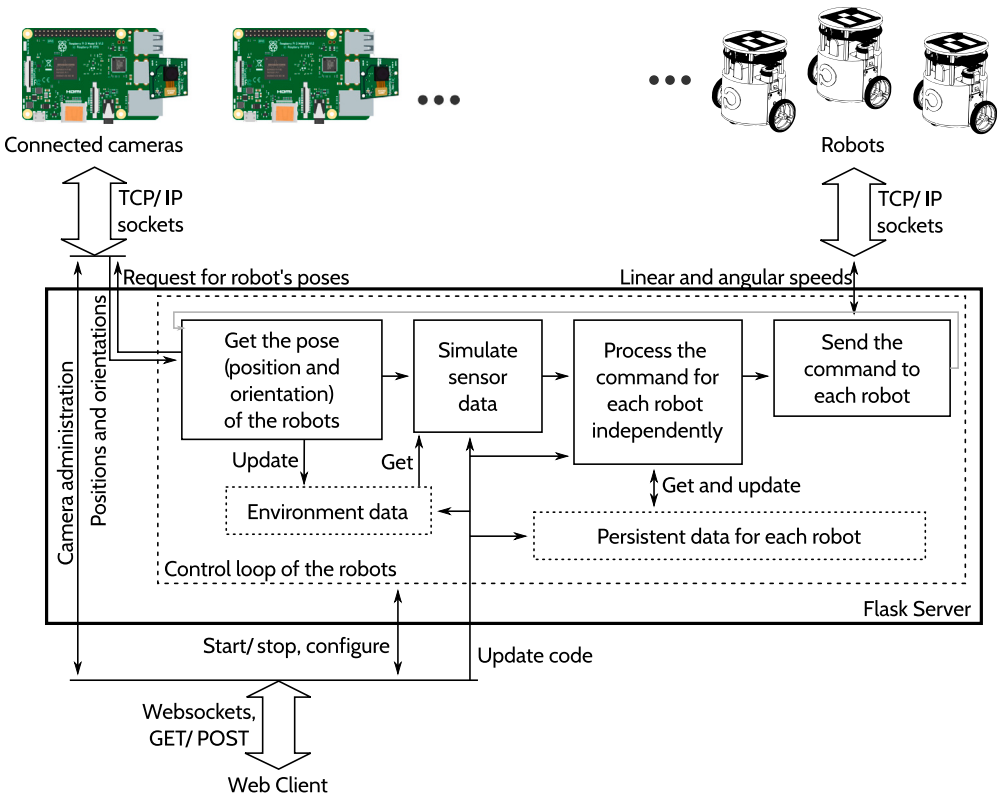


Fig. 5. Functional schematic of the BotArena. The cameras locate the robots, send the poses information to the supervisor. The supervisor processes those poses to simulate sensor data and to compute the robot's commands. The commands are sent to the robots and the robots move accordingly.

2.3.2. Experiment setup

To meet the constraint of modularity, it is essential to add or remove cameras and robots with ease. The supervisor operates on configuration files that specify the cameras in use, including their IP addresses and port numbers, as well as the robots, such as their identification, IP address, and port number. These configuration files are editable through the web interface.

To satisfy the constraint of versatility, the robots are categorized into *groups*, where each group has a defined behavior. The objective is to enable multiple packs of robots with different behaviors to coexist within the same application.

2.3.3. supervisor control loop and data management

Algorithm 2 presents the tasks done by the supervisor during an experimentation run.

For each camera, the position and orientation of detected robots are retrieved. Subsequently, this information is merged to generate an overview of all the robots in the arena. This step corresponds to the "Location/Orientation" in Fig. 1(b) and provides as output a table with robot ID, location (x, y) and orientation θ specified in the arena reference frame. According to the position and orientation of the robots, the sensor data for each robot are calculated (sensor simulation in Fig. 1(b)).

Sensor simulation is not a pre-implemented callable function, but rather a feature that must be customized by the user, allowing for tailored adjustments. To simulate sensor data, the user must implement a Python function which takes as input the marker identifier of the considered robot, the positions and orientations of all the detected markers, and the environment configuration. The function returns in output a dictionary containing all the sensor data for a given robot. For instance, to simulate a 2D LiDAR (Light Detection And Ranging), the function could compute the intersection between the robot's simulated beams (given its position and orientation) and the other markers. A similar approach can be taken with conic beams to simulate ultrasonic sensors.

In addition to sensor data, each robot can maintain persistent data (e.g., the values of a PID regulator or a local cost map). This data is stored in a dictionary specific to each robot, which can be updated during the processing of commands. Each group of robots may possess different persistent data; however, they are initialized identically for each robot within the same group.

Once the sensor data has been calculated, it is utilized together with any necessary persistent data to compute the command for each robot. This computation is performed using source code that returns the linear

Algorithm 2: Supervisor experiment loop

```
5  foreach Camera do
6      Open a TCP/IP socket;
7  foreach Robot do
8      Open a TCP/IP socket;
9  while Experiment is running do
10     foreach Camera do
11         Get the position and orientation of all the detected
             robots;
12     foreach Robot do
13         Generate the sensor data;
14     foreach Robot do
15         Compute the command based on sensor data and the
             robot's data;
16         Update the robot's data if needed;
17     foreach Robot do
18         Send the command to the robot;
```

Table 1
Main robots for swarm robotics compared to BotArena — mainly based on [30].

Robot/Feature	Introduction date	Developer	Commercially available	Open hardware	Cost
Khepera IV	2015	K-Team (EPFL spin-off)	✓	–	USD 3200
E-puck	2004	Ecole Polytechnique Fédérale de Lausanne (EPFL)	✓	–	USD 1000
SMARTmBot	2022	Purdue University	–	✓	USB 210 (parts)
Thymio	2011	Mobsya (EPFL spin-off)	✓	✓	USD 173
Kilobot	2010	Harvard University	✓	✓	USD 130
Mona	2019	University of Manchester	–	✓	USD 129
BotArena	2022	University of Angers	–	✓	USD 110 (parts)
HeRoSwarm	2023	University of Georgia	–	✓	USD 100 (parts)
GRITSBot	2015	Georgia Tech	–	✓	USD 50 (parts)
Colias	2014	University of Lincoln	–	✓	USD 32
WsBot	2019	Universidade Tecnologica Federal do Parana	–	✓	USD 17 (parts)

and angular speeds to be applied to. While this code may vary across different groups of robots, it remains consistent for each robot within the same group.

Finally, the computed linear and angular velocities are transmitted to their respective robots (TCP/IP sockets).

All these steps are illustrated in Fig. 5.

2.4. Synthesis

The BotArena is an open-source and open-hardware platform that facilitates reproduction and modification. While it serves as an additional platform for multi mobile robots, it possesses unique characteristics compared to existing systems. Some reviews of existing systems can be found in [9,27,28]. It appears that the criteria for choosing a test bench are multifaceted, as detailed in [29]. It is often necessary to make trade-offs among various parameters. A system performing with high localization accuracy (within centimeters) will be more expensive due to the use of an opti-track system, for instance. Some systems feature low-cost robots; however, they often utilize expensive cameras. Furthermore, many do not consider the size of the experimental area or the ease of transporting the test bench. In the following, the evaluations carried out on the arena will therefore be difficult to compare with existing arenas as the source data are fragmented and, in many cases, absent from the original publications. The BotArena is positioned within a distinctive cost segment (arena and robot), and its originality lies in the ease with which the arena can be relocated or the experimental area size can be adjusted. In addition to research applications, the BotArena is also valuable for communication and demonstration events. The ability to dismantle and reassemble it within a quarter of an hour provides a significant advantage and facilitates the promotion of this emerging segment of robotics. Representing an exhaustive comparison of platforms in an orderly manner is beyond the scope of this paper; however, a summary of the principal robots is provided in Table 1, positioning the BotArena robot within the current landscape of swarm robotics. It is noteworthy that this table is derived from [30] and supplemented with more recent robots [31,31,32].

Overall, it is affordable by most institutes and can be reproduced in numbers to be representative of swarm robotics [33]. The remainder of this paper focuses on quantifying arena performance.

3. Experiments

This section assesses the performance of the arena. Initially, the accuracy of the robot's localization will be examined, followed by an analysis of the network and processing times. Finally, a brief application simulating the behavior of an ant colony will be implemented.

3.1. Localization evaluation

Due to cost optimization, a Raspberry Pi SBC and its dedicated camera have been selected; however, concerns regarding performance are valid. The following experiments were conducted with a setup of three cameras with overlapping areas. A reference pattern, visible in Fig. 6, was printed in A0 format. All positions, orientations, and distances between markers are known, establishing a ground truth for comparison with the data collected by the system.

3.1.1. Global localization performance

In this experiment, three cameras are positioned to detect the reference pattern. None of the cameras can view the entire reference pattern; in particular, the first and third cameras do not share any overlapping fields of view.

Fig. 7 has been generated using MATLAB's boxplot tool. The operational canvas is as follows (excerpt from documentation[8]): for each box, the central mark indicates the median, while the bottom and top edges represent the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points that are not classified as outliers, while outliers are represented individually using the '+' marker symbol.

A jitter is observed in each position marker, which was anticipated. Fig. 7 has been constructed to quantify the localization inaccuracy. The localization jitter is computed based on the Euclidean distance from the ground truth to the detected marker position.

From Fig. 7(a), the localization error is bounded within the interval $[+1; -1.5]$ cm. When compared to the arena dimensions (1.3 x 1.6 m), this constitutes a reasonable range; however, considering the robot's size (8 cm), this interval corresponds to $\approx 32\%$ of the robot's size, which is a significant parameter. Given these inaccuracies and the robot's form factor, specifically the gap between the wheels and the body, the robots should not operate too close to each other, in order to prevent unintended physical locking. In terms of orientation (Fig. 7(b)), the error is bounded within the interval $[+6.7; -8.1]$ deg, primarily due to one marker.

The following section will focus on each camera and analyze their respective fields of view.

3.1.2. Decomposed location performance

In Fig. 8, overlapping areas are clearly visible: the spread measurements are wider in these regions. To characterize this, the problem will be addressed using clustering tools.

Many algorithms are available to evaluate cluster spreading. As the number of clusters is known (number of markers) and the clusters are distinct, K-means [34] appears to be a suitable algorithm for this application. The algorithm output is the sum of the Euclidean distance (d) between the centroid and each point of the cluster[9] defined as:

$$BSS = \sum_{i=1}^{N_c} \sum_{j=1}^{p_i} \left| X_j - \bar{C}_i \right| \tag{10}$$

With i denoting the cluster index, N_c denoting the number of clusters, $\bar{C}_i$ denoting the cluster centroid, X_j denoting the current sample

[8] https://fr.mathworks.com/help/stats/boxplot.html

[9] BSS - Between Cluster Sum of Squares.

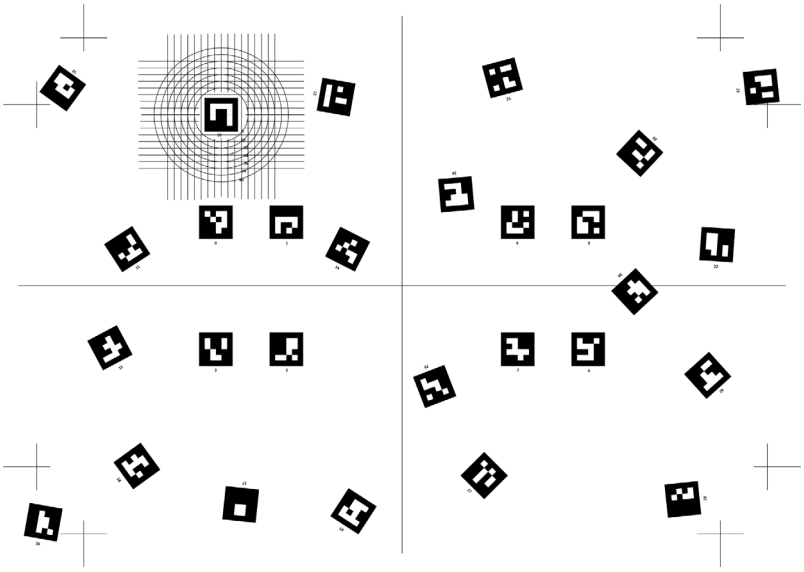
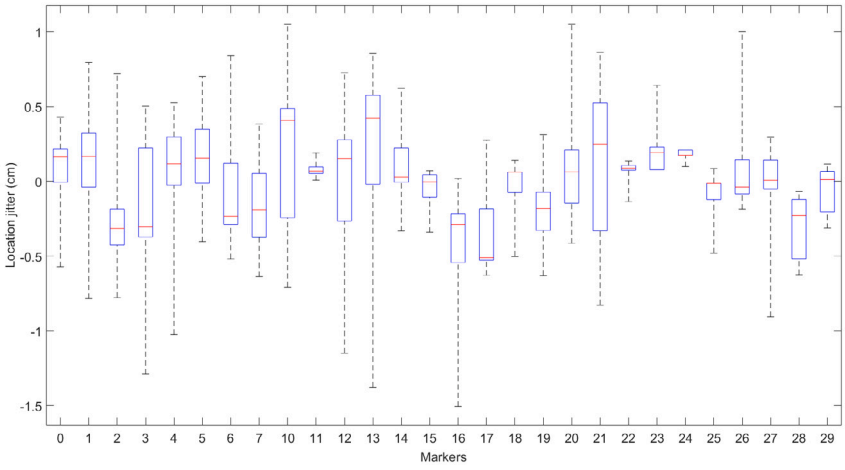
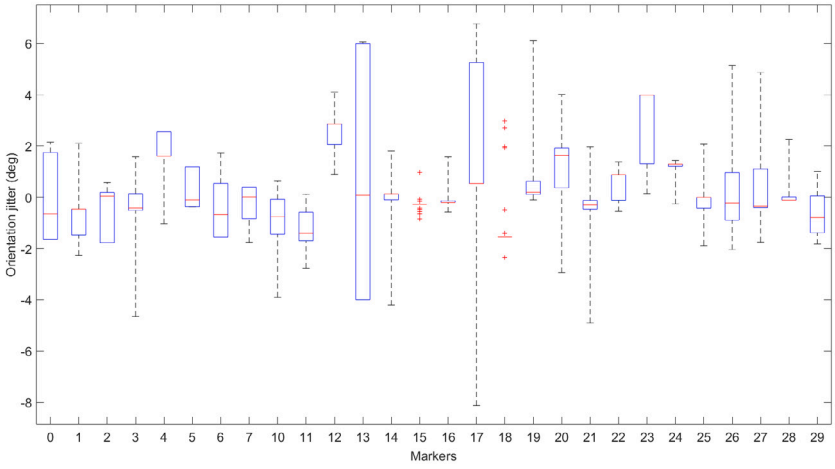


Fig. 6. A0 Pattern for localization evaluation.



(a) Localization jitter of markers



(b) Orientation jitter of markers

Fig. 7. Localization and orientation jitter of markers.

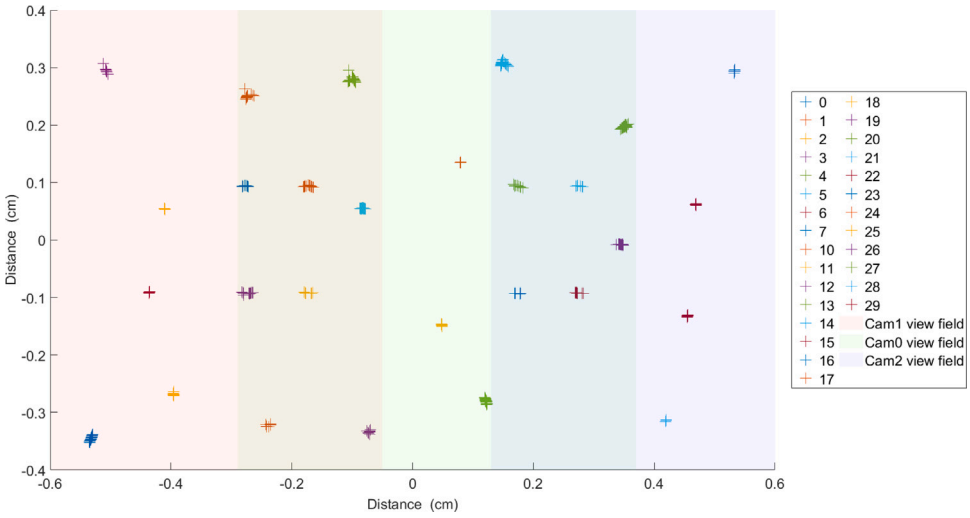


Fig. 8. Localization of markers by all system.

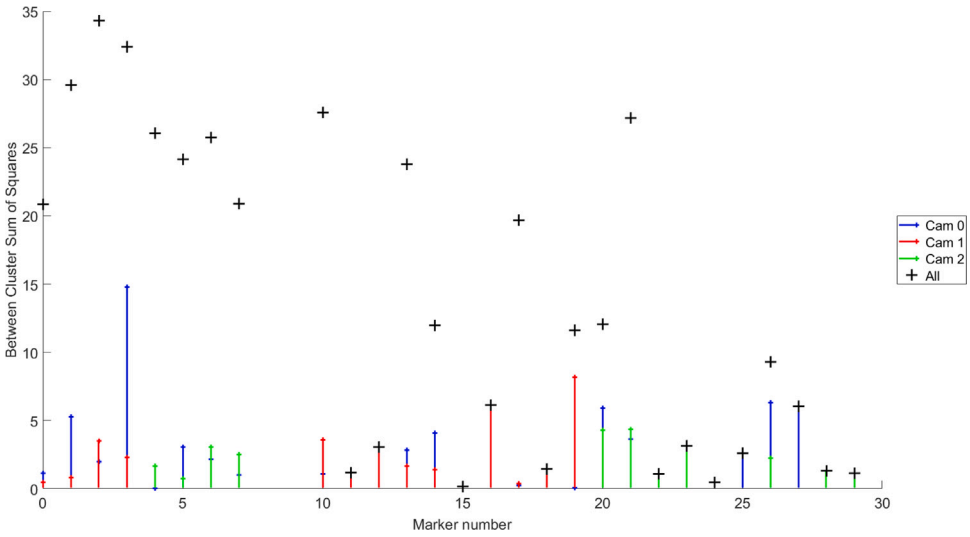


Fig. 9. Similarity comparison — Between Cluster Sum of Squares.

from cluster i, and p_i denoting the number of samples in the cluster. Between Cluster Sums of Squares (BSS) will provide a quantification of cluster dispersion. Fig. 9 presents the BSS of each marker according to each camera's detection and the overall detection.

When the marker is seen by only one camera, the overall BSS for that marker matches that of the camera. That corresponds to 12 markers. For the rest, we can clearly see that the entire system has a greater dispersion than the cameras taken individually. For applications, robots are submitted to the entire system, so finally, we can conclude that overlapping areas are less precise than others. To improve the precision, one could update the calibration algorithm that computes the extrinsic matrix.

The intrinsic matrix can also be evaluated with respect to the BSS dispersion of a single camera. The analysis of individual view fields reveals a constant: the fact that in the middle of the image, the dispersion of markers is low while the periphery where it is high. To try to highlight this, the order box is sorted by distance to the center of the image.

Fig. 10 illustrates the sorted markers and demonstrates a trend of increasing amplitude, particularly evident in the linear regression of BSS data. Optical aberration are documented for raspberry camera [35]; however, localization jitters are the result of the compromise between price, simplicity, efficiency and, performance. That is, despite the jitters, the arena can still be regarded as suitable for a wide range of applications.

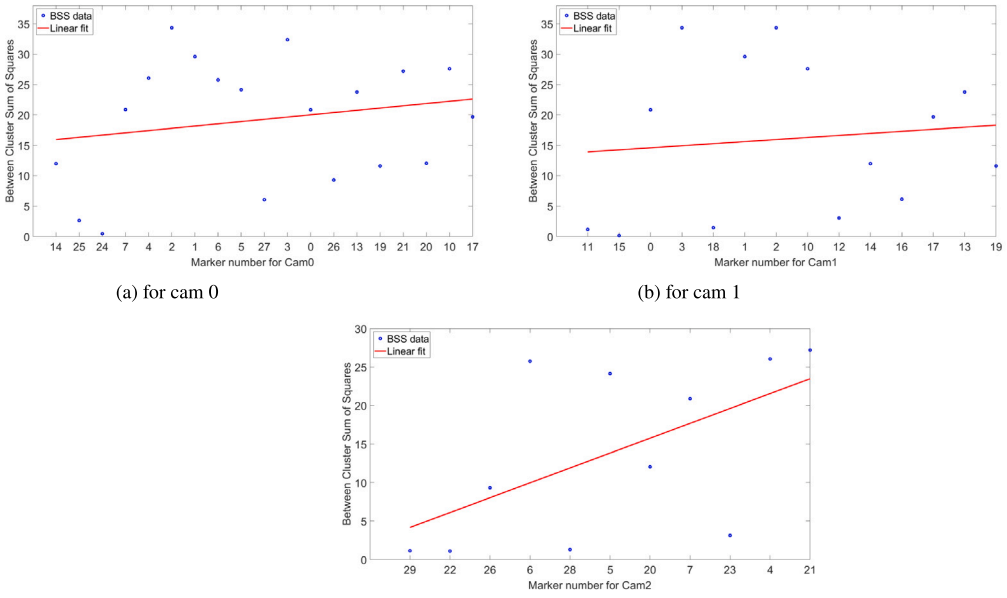
3.2. Timing evaluation

The BotArena is made of three parts: the cameras, the supervisor, the robots. All of these components are evaluated in the subsequent sections.

3.2.1. Camera processing time

During an experiment, the cameras continuously compute the poses of the robots and, upon request, transmit the most recent known poses to the supervisor. This section estimates the image processing time required to compute the poses of the robots. To evaluate the processing time of a given camera, time measurements were recorded on the camera module based on the number of markers within the field of view. For each quantity of markers, between 200 and 325 measurements were conducted. Fig. 11 summarizes these findings.

It can be noticed that the number of robot does not significantly impacts the processing time. It is mostly contained between 60 to 80 µs. Some unexplained outliers are observed but always in negligible time for user.



(a) for cam 0

(b) for cam 1

(c) for cam 2

Fig. 10. Sorted jitter of markers position for cameras.

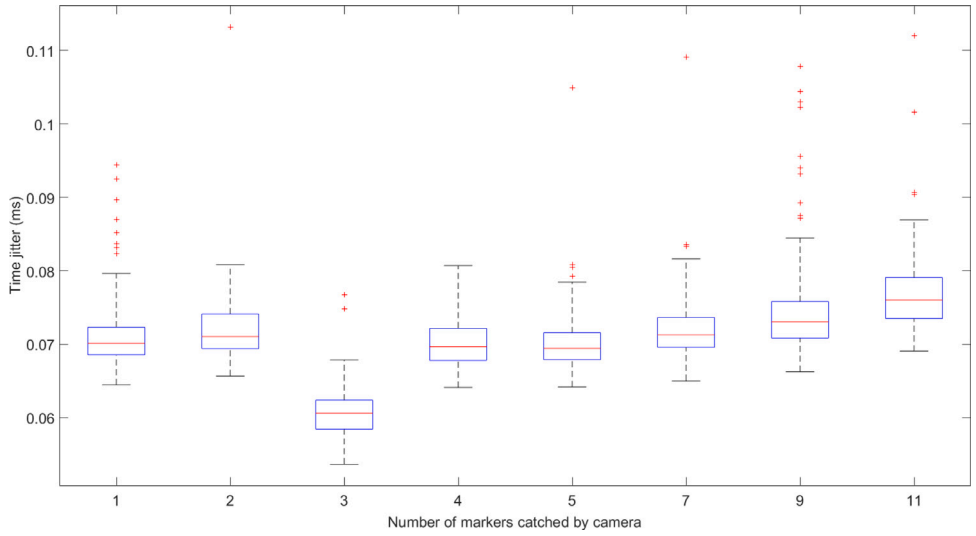


Fig. 11. Jitter representation.

3.2.2. *Supervisor response time*

As depicted in Section 2.3, the supervisor manages several steps:

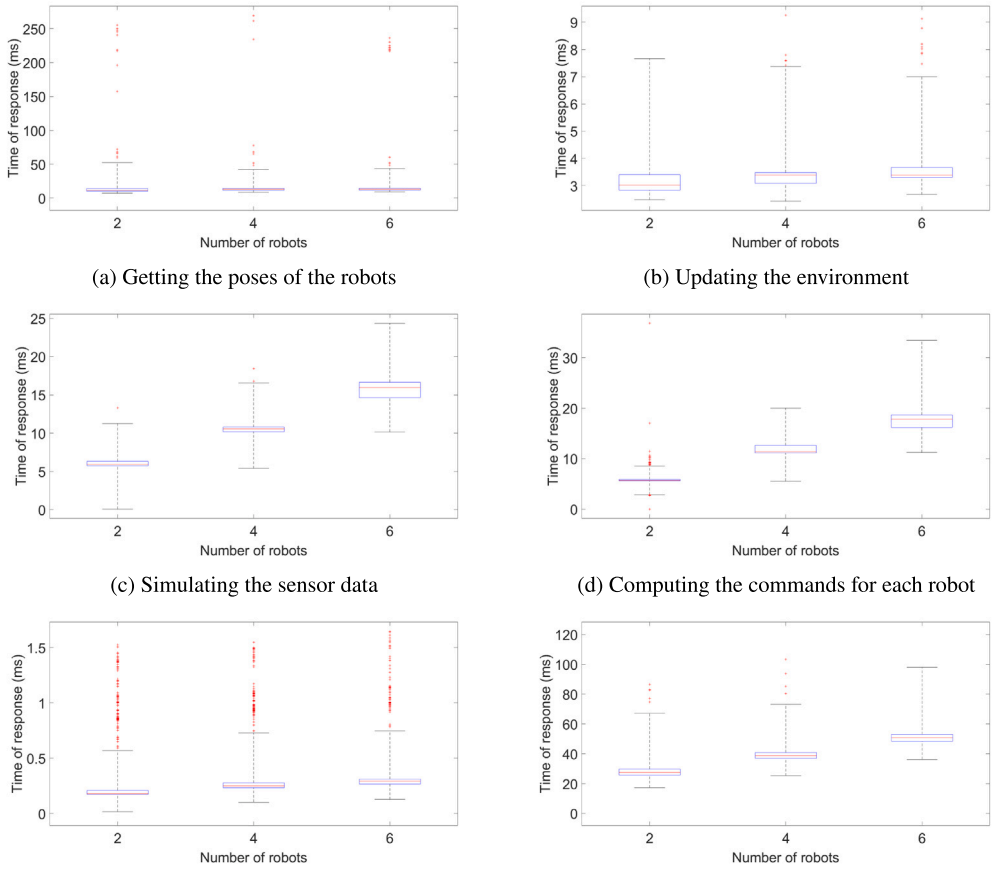
- Acquiring the poses of the robots from the cameras;
- Updating the environment;
- Simulating the sensor data for each robot;
- Computing the commands for each robot and, if necessary, updating the persistent data;
- Finally, sending the commands to the robots.

For each of these steps, a time evaluation is conducted over 1500 samples for three different numbers of robots: 2, 4, 6 robots. This setup utilizes three cameras to locate the robots. The results are depicted in Fig. 12.

Here are some interpretations of the results:

- The time measurement in Fig. 12(a) does not appear to be affected by the number of robots. This is expected, as the image processing is conducted independently to the supervisor; thus, the measured time corresponds to the duration required to send the last updated poses.
- From Fig. 12(f), it can be observed that the mean response time is approximately 50 ms (for 6 robots), resulting in a command rate of 20 Hz for the robots, which is more than sufficient to ensure responsive behavior.
- The overall processing time is dependent on the number of robots. This was predictable, as the supervisor must compute the sensor data and the command for each robot. Consequently, the command rate decreases as the number of robots increases.

In addition to this, outliers can be noted in the results: some supervisor iterations took longer than others and could even be considered excessively long. Although this does not significantly impact the average processing time, it could pose an issue during the experiment: the robots may not receive command updates in a timely manner. To prevent hazardous behavior, a watchdog has been implemented on the



(a) Getting the poses of the robots

(b) Updating the environment

(c) Simulating the sensor data

(d) Computing the commands for each robot

(e) Sending the command to the robots

(f) Overall supervisor response time

Fig. 12. Detailed supervisor response time.

robots, as said in Section 2.1. These outliers seem to be primarily due to network latency, occurring between the supervisor and the camera, but mainly between the supervisor and the robots. This will need to be investigated in the future.

Finally, based on these data, the maximum number of robots that the arena can support can be extrapolated. Indeed, 12 robots were constructed and considered in various experiments. However, the maximum number of robots that the arena can accommodate has not been experimentally determined. That is, based on the results in Fig. 12(f), utilizing a linear regression of the median values allows for the determination of a trend. The intersection of this curve with the value of the robots' watchdog provides a theoretical estimate regarding the maximum number of robots: 40 robots in this case. In order to accommodate 40 robots, it is assumed, based on the findings in [9], that each robot requires an 8 cm safety buffer. Consequently, at least an arena with dimensions of 2.60×1.04 m^2 is necessary. Given that each camera possesses a field of view of 1 m^2, only three cameras are required under these conditions. This suggests that the current configuration is feasible for this use case. However, to facilitate robot movement, it may be advisable to increase the size of the arena by adding additional cameras. Expanding the number of cameras will, in turn, increase the duration of the pose retrieval process (Fig. 12(a)). This process is presumed to scale linearly with the number of cameras, as it involves a simple request/response mechanism via a TCP/IP socket, with each camera transmitting the most recent detected poses. To explore the possibility of using that many robots, an experiment with 40 markers has been done (Fig. 13). Three cameras have been used to locate the markers, and they managed to provide the detected markers positions at a frequency around 16 Hz. From those positions, it has been possible to send 40 velocity commands at a 10 Hz frequency each, that is below the watchdog. Even if those result supports the hypothesis of using 40 robots, further investigation is required to validate the possibility of using that many actual robots.

It can be noticed that the theoretical limitation of 40 robots is not particularly restrictive, as numerous recent studies in the field of multi-robot systems do not involve such a larger number of robots [36–39].

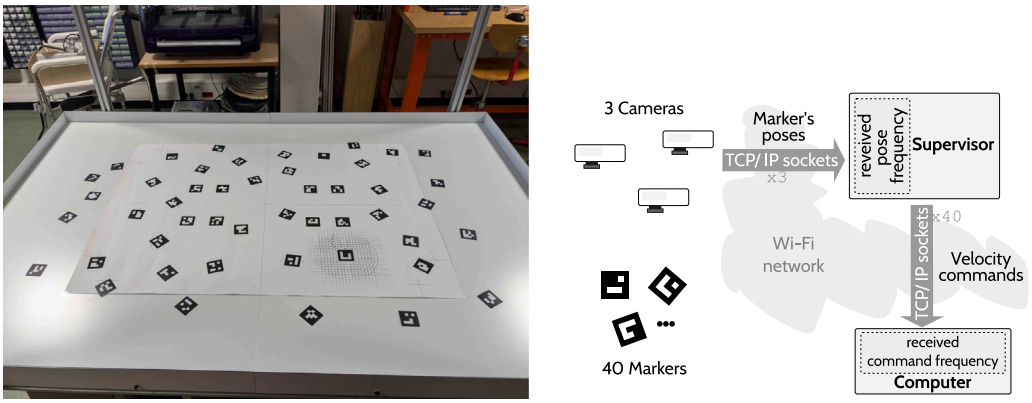
3.2.3. Robot timing

Finally, the robot's cycle time is evaluated. It has been done by simply including two time readings (Arduino *micros()* function) and subtracting them. This gives an average value of 28 µs. Has the measurement where highly constant, it has been decided not to plot the results.

3.3. Application

The results presented in the previous section show that the localization precision and timing of the system should allow the development of applications involving multiple mobile robots.

One popular paper behind swarm robotics is the shortest path problem observed in ants [2]. This experiment was reproduced with robots 20 years later using light markers [40], and this is a key milestone in swarm robotics, as group-level behavior was observed. The goal here is to reproduce a version of it with the BotArena.



(a) The 40 markers. (b) An overview of the experiment.

Fig. 13. Experimental setup with 40 markers.

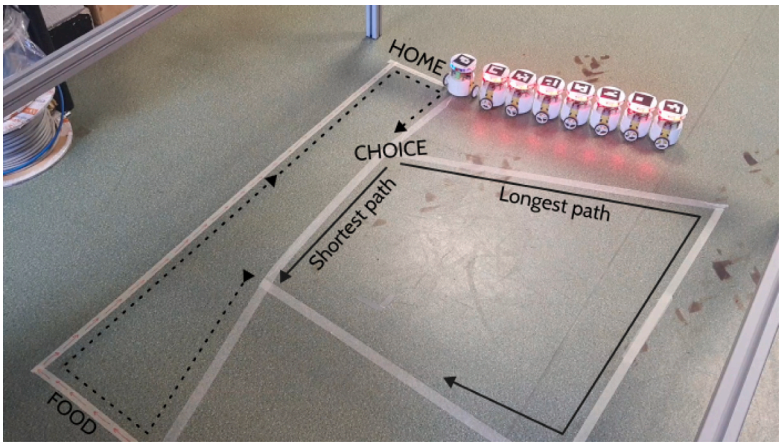


Fig. 14. Ant test bench.

This experiment involves an ant colony that has 2 possible paths to reach a food supply (Fig. 14). One of the paths is shorter than the other, but the ants (robots) have no prior information about the shortest one. At the crossroad, the ants randomly choose a path. Without any additional information, the ants have a 50% chance of choosing the shortest path.

While moving along the path, the ants deposit pheromones along their route. This causes the random choice at the crossroad to be influenced by the amount of pheromones on the path: *i.e.*, the ants will more often choose the path with the most pheromones.

The expected behavior is that, thanks to the deposited pheromones, the ants will increasingly choose the shortest path.

With the BotArena, it is possible to set up such an experiment using environment data to simulate the pheromone deposits. Fig. 14 represents the initial position of the test bench.

Fig. 15 shows two moments of the experiment. In this figure, the black ants depict the pose of the robots (a group of 8 robots is considered), and the blue cells represent the quantity of pheromone on the path (the bluer the more pheromone). In Fig. 15(a), the ants are using both paths equally, whereas in Fig. 15(b), the ants tend to prefer the shortest path.

Fig. 16 depicts 55 choices made by robots during the experiment. It appears that 69% of robots chose the shortest path, which is better than the 50% expected without involving pheromones and thus demonstrates swarm behavior.

Obviously, there is much more to be said regarding how to implement this example, but that is not the topic of this paper, which remains focused on the arena itself and its possibilities. The interested reader can find some references on the Keller–Segel model for chemotaxis in [41].

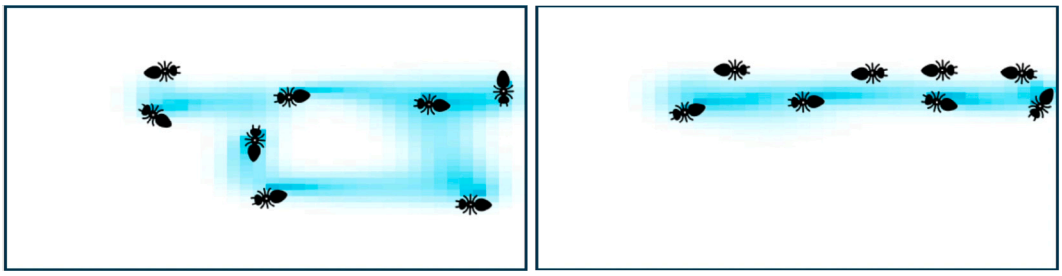
4. Conclusions and perspectives

As mentioned above, the objective of the BotArena is to provide the community with an affordable tool for experimenting in the field of mobile robot behavior. This arena integrates three primary criteria that facilitate its adoption by a wide range of institutions: cost, modularity, and versatility 1.2. This article delineates the architecture, operation, and design of the arena and the robot. Through appropriate experiments, the localization precision was quantified, along with an evaluation of the operational and cycle times of the arena. This enables readers to understand the arena's limitations.

Eventually, the arena has been used in an experimental setup to demonstrate its usefulness. The ant experiment has been showcased at multiple science festivals (Fig. 17). Undoubtedly, this aspect of robotics captivates the interest of the general public. The promotion of bio-inspired, yet unpredictable behavior, elicits inquiries and stimulates curiosity among a diverse audience.

Here is the future work for that platform:

- Testing the Stag markers and comparing them with AruCO markers within the arena context. The Stag markers may offer improved localization accuracy without increasing processing time;



(a) Both paths are equally chosen (b) The shortest path is preferred

Fig. 15. Distribution of ants during the experiment.

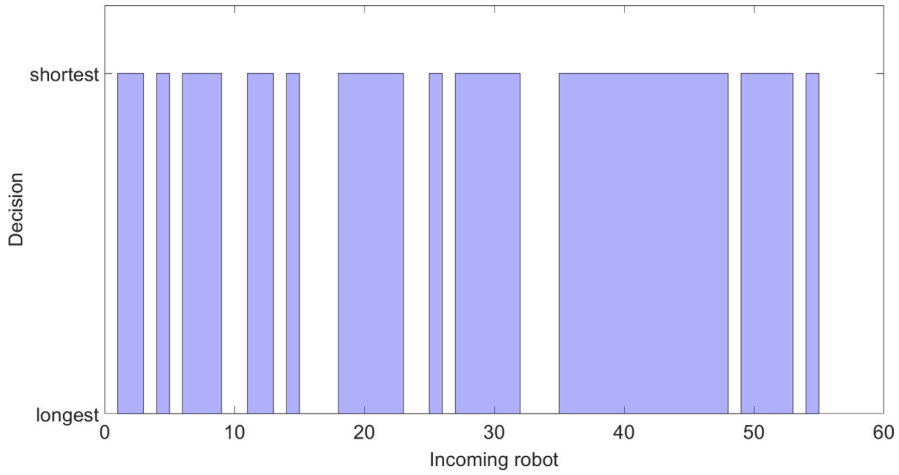
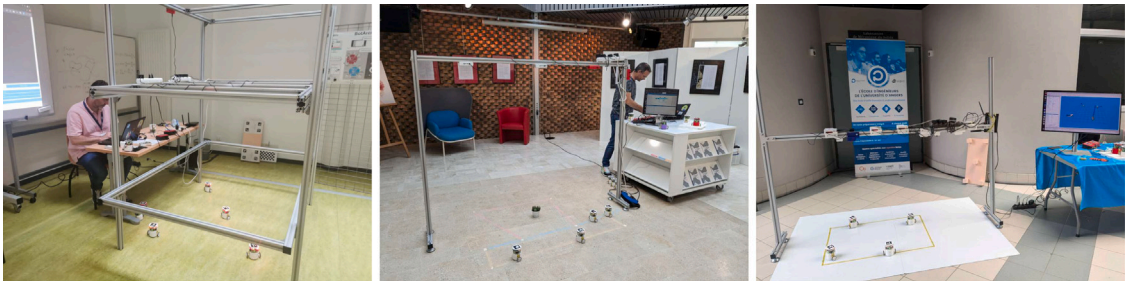


Fig. 16. Ant path choice.



(a) Automatic Demo Days (2022) (b) Science festival (2023) (c) Science festival (2024)

Fig. 17. Three events where the BotArena has been used to present multi-robots behaviors.

- Investigating the erratic network latency that punctually slow down the communication between the robots and the supervisor.
- Offering comprehensive sensor simulation capabilities out-of-the-box. This functionality can be implemented either as a plugin for the user interface or provided as directly accessible source code to simulate standard sensors such as LiDAR, IMU (Inertial Measurement Unit), and others.
- Testing the arena with a larger number of robots. To date, the arena has been tested with a maximum of eight robots simultaneously. There are plans to build additional robots and test the limits of the platform. As for the robots, the arena has been tested with at most 3 cameras, there are plans to double the number of cameras;
- Integrating the ROS2 framework.[10] ROS2 is widely adopted in academic robotic and its inclusion will facilitate the use of the arena by the broader research community. Moreover, ROS2 offers several tools for visualization and simulation that could be useful for this platform;
- Introducing fully holonomic robots. This would allow for the simulation of challenges related to maritime environments, including elements such as wind and ocean currents;
- Implementing online recovery in the event of network disconnection is currently lacking. At present, if a network disconnection occurs, the robot's watchdog is triggered, causing the robots to

[10] https://docs.ros.org/en/humble/index.html

stop. However, the system does not offer a mechanism to resume the experiment once the network reconnects. To restore normal operation, a complete system reboot is required. This will be updated in the future;
- Using the arena to validate theoretical results. The team is currently developing novel approaches for robot dispersion and plans to use this platform to test those algorithms.

The arena is already usable, all designs, codes, tutorials are available in open access.[11] The team will be happy to have feedback in case of reproduction.

CRediT authorship contribution statement

Franck Mercier: Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation. **Rémy Guyonneau:** Writing – review & editing, Validation, Supervision, Software, Methodology, Conceptualization. **Alain Godon:** Writing – review & editing, Software, Resources, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All data is available on the gitlab link given in the article https://gitlab.u-angers.fr/botarena/ba_conception.

References

[1] Brambilla M, Ferrante E, Birattari M, Dorigo M. Swarm robotics: a review from the swarm engineering perspective. Swarm Intell 2013;7(1):1–41.

[2] Goss S, Aron S, Deneubourg J-L, Pasteels J. Self-organized shortcuts in the Argentine Ant. Naturwissenschaften 76: 579-581. Naturwissenschaften 1989;76:579–81. http://dx.doi.org/10.1007/BF00462870.

[3] Duan H, Huo M, Fan Y. From animal collective behaviors to swarm robotic cooperation. Natl Sci Rev 2023;10(5):nwad040. http://dx.doi.org/10.1093/nsr/nwad040, arXiv:https://academic.oup.com/nsr/article-pdf/10/5/nwad040/49848073/nwad040.pdf.

[4] Bonabeau E, Theraulaz G, Dorigo M, Theraulaz G, Marco DdDF, et al. Swarm intelligence: from natural to artificial systems. Oxford University Press; 1999.

[5] Dorigo M, Theraulaz G, Trianni V. Reflections on the future of swarm robotics. Science Robotics 2020;5(49):eabe4385.

[6] Barca JC, Sekercioglu YA. Swarm robotics reviewed. Robotica 2013;31(3):345–59. http://dx.doi.org/10.1017/S026357471200032X.

[7] Jdeed M, Schranz M, Elmenreich W. A study using the low-cost swarm robotics platform spiderino in education. Comput Educ Open 2020;1:100017. http://dx.doi.org/10.1016/j.caeo.2020.100017, URL: https://www.sciencedirect.com/science/article/pii/S2666557320300033.

[8] Vitanza A, Rossetti P, Mondada F, Trianni V. Robot swarms as an educational tool: The Thymio's way. Int J Adv Robot Syst 2019;16(1):1729881418825186. http://dx.doi.org/10.1177/1729881418825186.

[9] Pickem D, Glotfelter P, Wang L, Mote M, Ames A, Feron E, et al. The robotarium: A remotely accessible swarm robotics research testbed. In: 2017 IEEE international conference on robotics and automation. ICRA, IEEE; 2017, p. 1699–706.

[10] Wilson S, Glotfelter P, Wang L, Mayya S, Notomista G, Mote M, et al. The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems. IEEE Control Syst Mag 2020;40(1):26–44.

[11] Zhang T, Tan F, Yu C, Wu J, Xu J. Understanding relationship between topic selection and academic performance of scientific teams based on entity popularity trend. Aslib J Inf Manag 2023;75(3):561–88.

[12] Bullock D, Johnson B, Wells RB, Kyte M, Li Z. Hardware-in-the-loop simulation. Transp Res C 2004;12(1):73–89. http://dx.doi.org/10.1016/j.trc.2002.10.002, URL: https://www.sciencedirect.com/science/article/pii/S0968090X03000792.

[13] Bacic M. On hardware-in-the-loop simulation. In: Proceedings of the 44th IEEE conference on decision and control. 2005, p. 3194–8. http://dx.doi.org/10.1109/CDC.2005.1582653.

[14] Åström KJ, Hägglund T. PID controllers: theory, design, and tuning. ISA-The Instrumentation, Systems and Automation Society; 1995.

[15] Zhang SS. Insight into the gassing problem of Li-ion battery. Front Energy Res 2014;2:59.

[16] Hasan K, Tom N, Yuce MR. Navigating battery choices in IoT: An extensive survey of technologies and their applications. Batteries 2023;9(12). http://dx.doi.org/10.3390/batteries9120580, URL: https://www.mdpi.com/2313-0105/9/12/580.

[17] Xiong R, Li L, Tian J. Towards a smarter battery management system: A critical review on battery state of health monitoring methods. J Power Sources 2018;405:18–29.

[18] Siegwart R, Nourbakhsh IR, Scaramuzza D. Introduction to autonomous mobile robots. MIT Press; 2011.

[19] Guyonneau R, Mercier F. IstiABot ou la Conception d'un Robot Libre pour l'Éducation et la Recherche. J3eA 2021;20:0002.

[20] Kato H, Billinghurst M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In: Proceedings 2nd IEEE and ACM international workshop on augmented reality. IEEE; 1999, p. 85–94.

[21] Rekimoto J, Ayatsuka Y. CyberCode: designing augmented reality environments with visual tags. In: Proceedings of DARE 2000 on designing augmented reality environments. 2000, p. 1–10.

[22] Naimark L, Foxlin E. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In: Proceedings. International symposium on mixed and augmented reality. IEEE; 2002, p. 27–36.

[23] Fiala M. ARTag, a fiducial marker system using digital techniques. In: 2005 IEEE computer society conference on computer vision and pattern recognition, vol. 2, IEEE; 2005, p. 590–6.

[24] Garrido-Jurado S, Muñoz-Salinas R, Madrid-Cuevas FJ, Marín-Jiménez MJ. Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognit 2014;47(6):2280–92.

[25] Kalaitzakis M, Cain B, Carroll S, Ambrosi A, Whitehead C, Vitzilaios N. Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers. J Intell Robot Syst 2021;101:1–26.

[26] Dawson-Howe KM, Vernon D. Simple pinhole camera calibration. Int J Imaging Syst Technol 1994;5(1):1–6.

[27] Jiménez-González A, Martinez-de Dios JR, Ollero A. Testbeds for ubiquitous robotics: A survey. Robot Auton Syst 2013;61(12):1487–501. http://dx.doi.org/10.1016/j.robot.2013.07.006, URL: https://www.sciencedirect.com/science/article/pii/S0921889013001322.

[28] Casan GA, Cervera E, Moughlbay AA, Alemany J, Martinet P. ROS-based online robot programming for remote education and training. In: 2015 IEEE international conference on robotics and automation. IEEE; 2015, p. 6101–6.

[29] Tonneau A-S, Mitton N, Vandaele J. How to choose an experimentation platform for wireless sensor networks? A survey on static and mobile wireless sensor network experimentation facilities. Ad Hoc Netw 2015;30:115–27.

[30] Calderón-Arce C, Brenes-Torres JC, Solis-Ortega R. Swarm robotics: Simulators, platforms and applications review. Computation 2022;10(6). http://dx.doi.org/10.3390/computation10060080, URL: https://www.mdpi.com/2079-3197/10/6/80.

[31] Starks M, Gupta A, OV SS, Parasuraman R. Heroswarm: Fully-capable miniature swarm robot hardware design with open-source ros support. In: 2023 IEEE/sICE international symposium on system integration. IEEE; 2023, p. 1–7.

[32] Jo W, Kim J, Wang R, Pan J, Senthilkumaran RK, Min B-C. SMARTmBOT: A ROS2-based low-cost and open-source mobile robot platform. 2022, arXiv preprint arXiv:2203.08903.

[33] Winfield AF, Nembrini J. Safety in numbers: fault-tolerance in robot swarms. Int J Model Ident Control 2006;1(1):30–7. http://dx.doi.org/10.1504/IJMIC.2006.008645, arXiv:https://www.inderscienceonline.com/doi/pdf/10.1504/IJMIC.2006.008645, URL: https://www.inderscienceonline.com/doi/abs/10.1504/IJMIC.2006.008645.

[34] Likas A, Vlassis N, J. Verbeek J. The global k-means clustering algorithm. Pattern Recognit 2003;36(2):451–61. http://dx.doi.org/10.1016/S0031-3203(02)00060-2, URL: https://www.sciencedirect.com/science/article/pii/S0031320302000602, Biometrics.

[35] Miikki K, Karakoç A, Rafiee M, Lee DW, Vapaavuori J, Tersteegen J, et al. An open-source camera system for experimental measurements. SoftwareX 2021;14:100688. http://dx.doi.org/10.1016/j.softx.2021.100688, URL: https://www.sciencedirect.com/science/article/pii/S2352711021000339.

[36] Zhang S, Lei X, Duan M, Peng X, Pan J. A distributed outmost push approach for multi-robot herding. IEEE Trans Robot 2024.

[37] Ebel H, Rosenfelder M, Eberhard P. Cooperative object transportation with differential-drive mobile robots: Control and experimentation. Robot Auton Syst 2024;173:104612.

[11] https://gitlab.u-angers.fr/botarena

[38] Zhao R, Li F, Lu X, Lyu S. Multi-objective cooperative transportation for reconfigurable robot using isomorphic mapping multi-agent reinforcement learning. Mechatronics 2024;101:103206. http://dx.doi.org/10.1016/j.mechatronics.2024.103206, URL: https://www.sciencedirect.com/science/article/pii/S0957415824000710.

[39] Dulce-Galindo J, Santos MA, Raffo GV, Pena PN. Distributed supervisory control for multiple robot autonomous navigation performing single-robot tasks. Mechatronics 2022;86:102848. http://dx.doi.org/10.1016/j.mechatronics.2022.102848, URL: https://www.sciencedirect.com/science/article/pii/S0957415822000769.

[40] Garnier S, Combe M, Jost C, Theraulaz G. Do ants need to estimate the geometrical properties of trail bifurcations to find an efficient route? A swarm robotics test bed. PLoS Comput Biol 2013;9:1–12. http://dx.doi.org/10.1371/journal.pcbi.1002903.

[41] Schroeder A, Ramakrishnan S, Kumar M, Trease B. Efficient spatial coverage by a robot swarm based on an ant foraging model and the Lévy distribution. Swarm Intell 2017;11. http://dx.doi.org/10.1007/s11721-017-0132-y.

Franck Mercier is an electronics engineer. He joined the CNRS in 2004, where he developed experiments in the field of combustion for safety, and space propulsion. Since 2015 he has worked at the University of Angers where he specialized in robotics. The various projects have led him to approach many fields of robotics, mobile robotics, humanoid, quadruped or swarm robotics. His position as a research engineer places him at the interface between engineering and science, requiring broad skills in electronics, mechanics, mechatronic design, sensors, but also software implementation, particularly with ROS and ROS2. He is confronted with production issues such as 3D printing, digital milling, or laser cutting.