Robotics and Autonomous Systems 144 (2021) 103849

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

A modular functional framework for the design and evaluation of multi-robot navigation ☆

Cyrill Baumann *, Alcherio Martinoli

Distributed Intelligent Systems and Algorithms Laboratory (DISAL), School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne (EPFL), CH 1015, Switzerland

ARTICLE INFO

Article history:
Received 14 November 2020
Received in revised form 3 April 2021
Accepted 21 July 2021
Available online 4 August 2021

Keywords:
Distributed control algorithms
Multi-robot systems
Benchmarking
Performance evaluation
Control software design

ABSTRACT

In this work, we address the design of tightly integrated control, estimation, and allocation algorithms allowing a group of robots to move collectively. For doing so, we leverage a modular framework that allows us to define precisely the needed functional components and thus consider and compare multiple algorithmic solutions for the same module. We demonstrate the effectiveness of such a framework through multiple spatial coordination challenges carried out both in simulation and reality and leveraging different distributed control laws (graph-based and behavior-based controllers).

Moreover, we investigate the impact of different localization and communication constraints as well as that of real-time switching of control laws on selected coordination metrics. Finally, we also introduce additional algorithmic components for demonstrating further the modularity of the framework.

We find that defining the modularity based on functionality is a very effective way to enable algorithm benchmarking and discover possible improvements of the overall software stack while at the same time being agnostic to the underlying hardware and middleware resources. This is an especially welcome feature in case of severely resource-constrained multi-robot systems. Moreover, an important benefit of such design process is that the resulting distributed control algorithms are very robust to the considered noise sources and amplitudes as well as to the diverse types of challenges considered.

1. Introduction

Collective movements are of interest for many multi-robot applications including robotic search and rescue scenarios [1], space applications [2], or car platooning [3]. One way to classify the related control algorithms is to consider the resulting spatial topologies: rigid topologies (e.g., static configurations, formations), or loose, structure-less topologies (e.g., aggregates, flocks). In formation control, as an example of the rigid category, the multi-agent system is tasked to achieve and maintain a given (geometrically defined) formation. The goal of flocking, on the other hand, is more comparable with a flock of birds or a herd of sheep, with the agents tasked to keep the fleet of robots in a certain vicinity. This usually translates into keeping a given inter-neighbor distance and aligning the direction of movement with the neighbors. In this paper, for demonstration purposes, as it is more generally applicable to engineering problems, we will primarily focus on navigation in formation, and secondarily on flocking.

Formation control has been an actively studied topic in the last two decades, and as such there exists a large amount of prior work, as mentioned in relatively recent surveys [4,5]. In literature, two main coordination schemes have been used for this topic: Leader–Follower (LF) and Virtual Structure (VS).

In the LF scheme, one agent (or multiple agents [6,7]) is (are) designated as leader. All other agents, the followers, are then tasked to keep a certain relative position (and more rarely orientation) with respect to the leader, resulting in the formation shape retention. This method has first been introduced in [8] and was improved upon ever since. For example, in [9] a formation shape variation was introduced to account for obstacles. In [10], the authors propose a less traditional LF method, where agents maintain a formation by keeping a desired angle with a specific, predefined neighbor. It is worth noting that the LF scheme has also been applied to model less classical formations, such as, for example, the information (and thus control) relationship between a patroller and an intruder in [11].

The VS scheme has first been proposed in [12]. Its principle consists out of three steps: aligning and assigning the agents to spots within the virtual structure, moving the virtual structure,

☆ This work has been financially supported by the Mitsubishi Electric Corporation, Japan.

* Corresponding author.
E-mail addresses: cyrill.baumann@epfl.ch (C. Baumann), alcherio.martinoli@epfl.ch (A. Martinoli).

https://doi.org/10.1016/j.robot.2021.103849

and moving the agents towards their new positions according to the virtual structure. This scheme has since been applied to numerous platforms such as the mobile robots in the original work [12] or spacecraft [2].

One of the most commonly used control methods within both the LF and the VS framework is the graph-based Laplacian control. A very insightful introduction to it can be found in [13]. Its principle consists out of building an interaction graph between the robots, adding biases in order to create a specific formation topology, and finally controlling the robots based on the previously constructed graph. In [14], Laplacian-based formation control is applied to spatially coordinate robots using only local range and bearing measurements. [15] uses communication to improve the connectedness of the underlying graph, especially for the case of a blocked line of sight. In [16], the authors leverage a complex Laplacian to enable the formation to be scalable geometrically based on the relative positions of two leaders.

While all these works are algorithmically interesting, they all focus on a specific control component of the system and therefore do not explicitly report, in a detailed way, a number of additional algorithmic design choices that had to be made in order to obtain a functional system. Moreover, some contributions validate their findings exclusively in simulation (in the best case using high-fidelity simulators), some use powerful real robots enabling the deployment of dedicated middleware such as the Robot Operating System (ROS, ROS2 [17,18]) while others propose ad-hoc solutions for extremely resource-constrained robots. Finally, most of the contributions consider simplified or idealized environmental conditions (e.g., no obstacles, no robot failures, accurate absolute localization systems, and flawless communication channels) that result in algorithmic contributions operating reliably only in such conditions.

Since our intention is to evaluate the performance of algorithmic solutions on a targeted robotic platform in reality, we felt that the design of a modular framework based on functionality was needed as a first key element paving the way towards such benchmarking effort. Particularly as, to the best of our knowledge, currently no such framework targeted to the design of multi-robot navigation strategies exists.

Thus, in this paper, we propose a modular framework able to accommodate the various functional elements needed to achieve robust multi-robot navigation, in formation or as a flock. Therefore, thanks to the functional abstraction, any of the published solutions mentioned above should be easily cast in our framework and possibly benefit of further improvement by exchanging some of its algorithmic components. Such process might lead to, for example, additional robustness to different communication and localization conditions and might provide new insights on the interactions of the components which could not be achieved without such a structured approach. It could also increase the reuseability of code and simplify the portability from simulation to real robots, as only the algorithmic parts interacting with the simulation engine, respectively the underlying hardware, have to be adapted. Furthermore, it would ease proper calibration of simulation tools as sensory reading and actuator commands are encapsulated in their respective modules and can thus be calibrated separately. Finally, it is worth noticing that the resulting software stack would be agnostic to the existence of a dedicated, potentially modular middleware since it guides the control design and optimization at a more abstract level, namely that of the functionalities needed to obtain the overall targeted spatially coordinated behavior.

In summary, the main contribution of this paper is to propose a modular functional framework for the design, implementation, and evaluation of multi-robot navigation strategies and illustrate its benefits through multiple collective movement variations. In particular, we use the flexibility of the framework to investigate the robustness of a specific algorithm for navigation in formation under different communication and localization constraints, compare parallel and sequential inclusion of collision avoidance into the collective movement, compare an hybrid algorithm consisting of both loose and tight coordination components with one based purely on tight elements, and extend one of the algorithms with an additional component focusing on heading control.

The remaining of this paper is organized as follows: first, the tackled problem is described and broken down into sub-problems effectively introducing our modular functional framework. Then, we describe the methods we adopted for solving each sub-problem as well as our experimental setup. Finally, we detail the algorithmic investigations undertaken and discuss the results obtained in simulation and reality, and report some conclusive remarks.

2. Problem statement

The collective movement problem considered here consists of convening N homogeneous differential wheeled robots into an ensemble obeying to specific requirements, and move them towards a common goal or along a given trajectory while reactively avoiding collisions. As shown in Fig. 1, the robots are endowed with both relative and absolute positioning systems as well as an omni-directional proximity sensing system. Inter-robot communication capabilities are available as well. It is worth noting that the communication range is assumed to be larger than the relative localization range, which is again assumed to be larger than the proximity sensing range.

In order to showcase the applicability of the modular functional framework to different types of collective movement, we have chosen two specific collective movement problems for our experiments:

1. The creation and maintenance of a team of robots in a desired formation while driving around a pre-established trajectory under different localization and communication constraints. Both the target formation's orientation as well as the target positions of each robot within the formation are not defined in advance (i.e. the individual robots' positions are interchangeable).
2. The creation and maintenance of a flock of robots using two of the three rules proposed by Reynolds [19], attraction and repulsion, following the very same pre-established trajectory mentioned above.

Furthermore, all the following requirements must be fulfilled: all the coordination laws are fully distributed, all the algorithms use exclusively on-board computational resources, and all robots are identically programmed.

3. Modular functional framework and algorithms

Independently of the exact formulation of the collective navigation mission and the algorithms used, there are multiple Functional Elements (FEs) needed for every robot to converge towards the requested spatial coordination. These FEs, listed in Table 1, form the modular framework proposed in this work, which, to our knowledge, is the first of its kind. Hierarchically, the framework is situated above a dedicated middleware, as illustrated in Fig. 2. The middleware could be a powerful, modular, and hardware-agnostic one such as the Robotic Operating System (ROS) or a simpler one, specific for a single platform, such as the Khepera IV Toolbox [20], perhaps augmented by specific open-source libraries required for the compilation of some of the FEs. As indicated through the information flow in Fig. 2, the framework

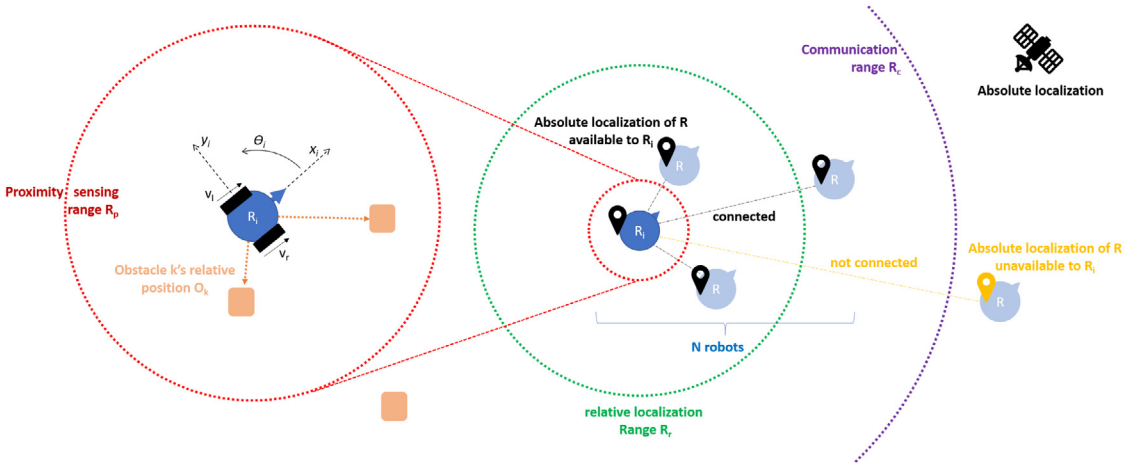


Fig. 1. Schematic representation including the main variables and parameters used in this work.

couples the individual FEs together into a single-threaded controller, where all previously gathered information is processed and made available to the next element (note that FEs 4 and 5 can also, but do not have to, be executed in parallel).

The availability of a dedicated middleware such as ROS would be highly advantageous for the framework, as it could take advantage of all its main components: 'plumbing' for the inter-element communications, 'tools' for visualization and debugging, and 'capabilities', which correspond to ready-made algorithms for each FE. As a consequence, if ROS is available, the framework will consist of ROS nodes and Table 1 would represent a sort of recipe to design and concatenate them. Generally speaking, thanks to the functional abstraction, the framework is formulated in a way that provides useful guidance with and without the availability of a dedicated, more or less powerful middleware.

Table 1 reports the FEs required for both types of movement, navigation in formation and flocking, as well as their input and output variables (see the following dedicated sections for a detailed description of these variables and the associated algorithms within each FE). We note that the first difference between the two movements is that flocking does not need an explicit consensus on the robots' position within the ensemble. Furthermore, FE 4 is specific to each algorithm; there exist therefore two variants, 4.1 and 4.2, which are specified hereafter. In this section the algorithms used for the various components of Table 1 are defined generally, concrete parameters used in our implementation can be found in Table 2. It is further worth noting that in this work, for every FE, one or multiple well-established algorithms have been implemented in an attempt to provide intuitive examples for the framework, but without necessarily aiming at the highest possible performance in the resulting combination.

3.1. State estimation

Depending on the situation, robots have different localization data available for their own pose and that of their teammates (e.g., odometry, relative or absolute positioning combined with communication channels). The implementation of this FE is highly dependent on the capabilities of the target platform. As shown in Fig. 1, we consider a robot equipped with communication capabilities, both absolute and relative localization functionalities, as well as an omni-directional proximity sensory system.

If absolute localization is available, the odometry data on our robots are fused through an Extended Kalman Filter (EKF) with the absolute pose information provided for instance by (an emulation of) a Global Navigation Satellite System (GNSS). The knowledge of the own pose is then broadcasted by the robots to their teammates via a dedicated narrow-band communication channel (e.g., WiFi). If relative localization measurements are available, these are fused by a noise-weighted average [21] with the received broadcasts of neighboring robots to get a higher (though less accurate) update rate of their relative positions.

To estimate the position of obstacles, any omni-directional proximity sensing system (e.g., a belt of sensors with overlapping fields of view) is suitable. When considering, for example, a belt of proximity sensors, the coordinates of an obstacle can easily be obtained through calibration of the sensor characteristics using the detecting sensor's angle and measured distance, resulting in the obstacle k's position O_k relative to the robot, as illustrated in Fig. 1. Of course, this simple obstacle detection algorithm can readily be exchanged for a more powerful obstacle mapping algorithm leveraging similar or different sensors (e.g., LIDAR).

3.2. Orientation consensus

As the formation's orientation is not established a priori, a consensus protocol is needed. In a situation where robots can leverage a common reference frame, this consensus is rather trivial; one possible solution is to implement a simple Laplacian rendez-vous controller. However, the robots do not necessarily have a common reference frame in all cases: for instance, in a GNSS-denied environment, such a frame should not be taken for granted. When no absolute positioning information is available, a consensus through communication can still be achieved, as long as relative localization information and communication are available. Considering the very same control law mentioned above based on a Laplacian rendez-vous, the implementation would be as follows.

Given robot i's bearing measurement of robot k $\mathcal{M}_{ik} \forall k \in r$ with $r \subset \mathcal{R}$ a subset of robots visible to robot i with $r \neq \varnothing$ and $\mathcal{R}$ being the set of all robots, and $\mathcal{M}_{ki}$, the measurement for robot i from every robot in r, as well as the orientation consensus bearing beliefs $B_{ll} \forall l \in (\{i\} \cap r)$, we can express robot k's orientation consensus belief B_{kk} in i's reference frame as:

$$B_{ik} = B_{kk} + \mathcal{M}_{ik} - \mathcal{M}_{ki} - \pi \tag{1}$$

Applying a Laplacian on all B_{ik} is straightforward, where some attention has to be brought to the fact that the orientation has a cyclic value.

Based on the orientation consensus obtained here, a common reference frame can be established. For example, by taking the orientation axis as the x-axis and using calibrated sensors for scale.

Table 1
FEs required for navigation in formation and flocking. The numbering on the left specify the FE number and corresponds to the dedicated sections below further illustrating the algorithms chosen for a given FE.

Functional Elements (FE)	Input	Output
1. Estimate pose of itself and team members, estimate position of obstacles	Sensing data	$\mathcal{M}, \mathcal{B}, \mathcal{X}, \mathcal{O}$
2. Reach consensus on orientation of the ensemble	$\mathcal{M}, \mathcal{B}$	$\mathcal{B}$
3. Allocate position within the ensemble (formation only)	$\mathcal{X}, b$	b
4.1 Create temporary target from trajectory to specific position within the formation	$\mathcal{X}, \mathcal{B}, b$	$\mathcal{X}^{tar}$
4.2 Create temporary target from trajectory to specific position within the flock	$\mathcal{X}$	$\mathcal{X}^{tar}$
5. Add movement toward common goal to target position	$\mathcal{B}, \mathcal{X}^{tar}$	$\mathcal{X}^{tot}$
6. Add reactive collision avoidance[a]	$\mathcal{O}, \mathcal{X}^{tot}$	$\mathcal{X}^{res}$
7. Move robot toward temporary target position	$\mathcal{X}^{res}$	Motor commands

[a]A reactive collision avoidance component is not needed in idealized conditions, but becomes necessary for realistic ones involving, for example, navigation through obstacles, malfunctioning robots, etc. Furthermore, it is worth noting that for this FE orthogonal sensory input is used (e.g., a dedicated proximity sensor belt, c.f. Fig. 1) instead of the positioning information already shared among the robots for collective navigation purposes.

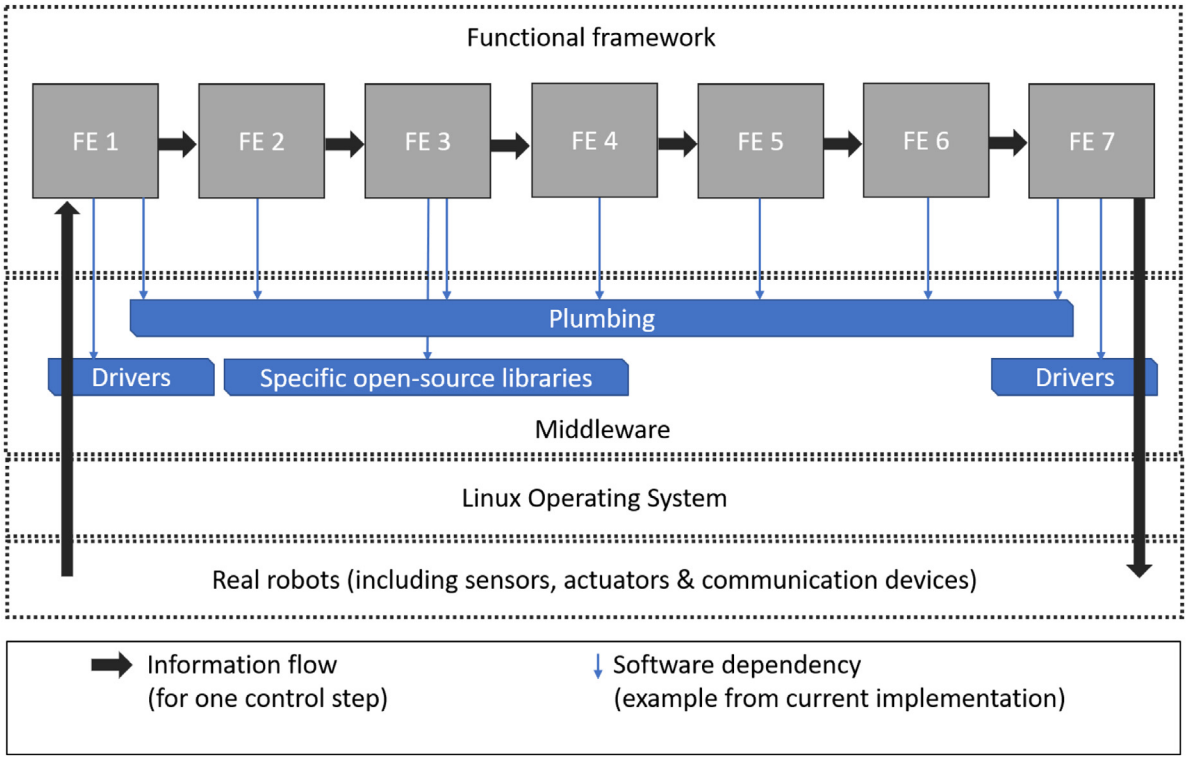


Fig. 2. Schematic representation of the framework with the software dependencies on the middleware layer. In an attempt to increase the readability of the figure, the implementation details of the different FEs are not represented here.

3.3. Target position assignment

For the target position assignment within the formation we chose to use a Hungarian algorithm [22], which solves the problem optimally (in terms of overall distances between the robots and their target position) in $\mathcal{O}(N^3)$ even for the case of partial assignments where we have more robots than positions in the formation or the contrary. Assuming perfect knowledge of all robots positions (i.e. no noise) and given that the solution obtained by the Hungarian algorithm is optimal, we obtain the same solution on every robot running the algorithm. This allows for a fully decentralized (although redundant) implementation, where every robot calculates the assignments for itself and thus remains fully functional even in scenarios without communication.

Though this cannot be guaranteed anymore for realistic situations involving noisy beliefs about the relative localization of neighboring robots, their distributions are assumed to be Gaussian with zero mean. Given the slow convergence of each robot toward its target position (compared to the update frequency of the measurements), it is therefore safe to assume that occasional momentarily wrong assignments of some robots due to large measurement noise will not influence the general outcome of the assignment in practice.

3.4. Self-positioning within an ensemble

In order to control the individual robots to their relative target positions within the ensemble, numerous algorithms exist. We consider two different ones, a graph-based Laplacian control for formation control as well as an algorithm based on vector summation for flocking. Furthermore, we also consider a combination of them.

3.4.1. Self-positioning within a formation

We use the standard graph-based Laplacian feedback method for a single integrator kinematic model as in [13] in order to

control the individual robots towards a formation:

$$\dot{X}_i = -\mathcal{L}X_i + b \tag{2}$$

where $\mathcal{L}$ is the Laplacian matrix of the underlying connectivity graph, X_i the N-dimensional state vector of the ith robot in its local coordinates (x_i, y_i, θ_i), which is centered on the robot with the x axis pointing forward, and b the bias vector corresponding to the relative positions within the formation (also in the robot's local coordinates). Two robots are 'connected' in the graph if they have knowledge of their absolute or relative positions. It is worth noting that a consensus on the heading angle θ_i is not necessary to obtain a formation (and thus not used by our standard robot control explained in Section 3.7), but allows the robots to align themselves with respect to each other.

Given that every robot is always centered at {0,0} in its local coordinate frame, we can express the temporary target position of robot i as:

$$X_i^{tar} = -\dot{X}_i dt \tag{3}$$

with dt a numerical parameter (c.f., Table 2).

3.4.2. Self-positioning within a flock

In order to control the individual robots within the flock, simple attraction and repulsion rules have been implemented:

$$X_i^{tar} = \frac{1}{N}\sum_{j=1, j\neq i}^{N}\left(X_j a_{i,j}^{att} - X_j a_{i,j}^{rep}\right) \tag{4}$$

Where

$$a_{i,j}^{att} = \begin{cases} 0 & \text{if } \|X_i - X_j\| \leq Q + \epsilon \\ 1 & \text{if } \|X_i - X_j\| > Q + \epsilon \end{cases} \tag{5}$$

and

$$a_{i,j}^{rep} = \begin{cases} 0 & \text{if } \|X_i - X_j\| \geq Q - \epsilon \\ 1 & \text{if } \|X_i - X_j\| < Q - \epsilon \end{cases} \tag{6}$$

with Q and ϵ numerical parameters (c.f., Table 2).

Note that for the sake of simplicity, for every robot, complete knowledge of the flock state is assumed here (e.g., through global communication). However, using the very same rules, flocking can also be achieved with partial knowledge, as is analyzed for example in [23].

3.4.3. Generalized self-positioning within a formation or a flock

Both flocking and navigation in formation have their respective advantages and disadvantages. By alternating their use depending on the current situation, we aim to leverage the best of both options. This alternation is done in FE 4 of Table 1 through the use of a finite state machine:

$$X_i^{tar} = \begin{cases} \frac{1}{N}\sum_{j=1, j\neq i}^{N}\left(X_j a_{i,j}^{att} - X_j a_{i,j}^{rep}\right) & \text{if } nav\_state == flocking \\ -(\mathcal{L}X_i + b)dt & \text{if } nav\_state == formation \end{cases} \tag{7}$$

with implementation details as specified in Sections 3.4.1 and 3.4.2 . In fact, due to the functional abstraction, the very same implementations could be used. While for this implementation the distance from the closest obstacle is used as the criterion for switching the navigation state (nav_state) for each robot, a more elaborated criterion, for instance based on the obstacle density around the group of robots, could be used for a synchronous switching of the navigation state of the whole ensemble.

3.5. Integration of a common goal

In our case, the common goal is given as a temporal trajectory in a common reference frame based on $\mathcal{B}_{ii}$. Using $\mathcal{B}_{ii}$, we can then simply transform it into T_i, a temporal trajectory in robot i's coordinates and add the common goal to the individual one:

$$X_i^{tot} = X_i^{tar} + cT_i(t) \tag{8}$$

with c being a scaling coefficient (c.f., Table 2 for the chosen numerical value) and t the current time.

It is worth noting that while we adopted a vector summation approach in this work, other approaches, for example based on optimization techniques, can easily be used as substitute.

3.6. Integration of reactive collision avoidance

In most robotic implementations not everything is fully predictable. For example, obstacles may appear unexpectedly, robots may malfunction and suddenly become obstacles to be avoided, or the environmental situation leads to degraded navigation behavior (e.g., collisions or near misses among robots). In this work, two possible solutions for reactive Collision Avoidance (CA) were considered and combined with the collective navigation algorithms using two different methods.

Consistently with the idea of the functional framework, the CA algorithms do not directly use the raw values of the sensory system dedicated to this functionality (different from that used for collective navigation), but use M vectors corresponding to the coordinates of the detected obstacles O_k with $k \in [1, M]$ which are obtained via FE 1 reported in Table 1. Recall also that, as illustrated in Fig. 1, we assume a noisy but omni-directional proximity sensory system.

3.6.1. Laplacian collision avoidance

Collision avoidance within a graph-based Laplacian formation can easily be achieved by adding obstacles as negative weights in the graph, as was done for example in [24]. However, in this work, the influence of the obstacle avoidance is strictly separated from the influence of the formation control Laplacian, and only combined with the latter in a second step (c.f. Section 3.6.3), also because the sensory systems used for the two FEs, collective navigation and collision avoidance, are different. Given the input of the coordinate vectors to M detected obstacles O_k with $k \in [1, M]$, the resulting CA motion vector can be expressed as:

$$X_L^{CA} = \frac{-1}{M} \cdot \sum_{k=1}^{M} w(\|O_k\|) \cdot O_k \tag{9}$$

where $w(x)$ is a nonlinear weighting function (c.f., Table 2).

3.6.2. Braitenberg collision avoidance

One of the most commonly used collision avoidance behaviors is the purely reactive Braitenberg algorithm [25]. As a second example of collision avoidance we thus decided to implement a slightly modified version of this algorithm. In the original work, the algorithm is applied directly to wheel speeds. To cast it in our framework, we transformed these wheel speeds into a motion vector for collision avoidance, which can then be combined with the other motion vectors as described in Section 3.6.3.

The Braitenberg algorithm is based on readings coming from a suite of S_B discrete sensors. Therefore the M vectors O_k with $k \in [1, M]$ containing the position of detected obstacles need to be transformed into a set of readings s_i with i $\in [1, S_B]$. This can be done using idealized sensors, for example:

$$s_i = max\left(\left(1 - \frac{O_{k,range}}{R_p}\right)\frac{1}{\sigma\sqrt{2\pi}}e^{-\left(O_{k,bearing} - \mu_i\right)^2/2\sigma^2}, k \in [1, M]\right) \tag{10}$$

with $\sigma = \frac{\pi}{S_B}$, $\mu_i = \frac{2\pi i}{S_B}$ and S_B being the total number of sensors and R_p their range, as illustrated in Fig. 1.

Then, a version of the original algorithm is applied for every independently controllable wheel of the robot as follows:

$$v_{B,j} = \frac{1}{T_B} \cdot \sum_{i=1}^{S_B} W_{B,j} \cdot (1 - \frac{s_i}{0.5}) \tag{11}$$

with numerical parameters W_B, a Braitenberg weight-set and its sum T_B per dimension.

For a differential wheeled vehicle as considered by the Braitenberg algorithm, it follows that:

$$u_B = C_B \cdot (v_{B,l} + v_{B,r})$$

$$\omega_B = C_B \cdot (v_{B,r} - v_{B,l})$$

$$r_B = \frac{u_B}{2 \cdot \omega_B}$$

with $v_{B,l}$ and $v_{B,r}$ the resulting speeds of the Braitenberg algorithm for the left and right wheel, respectively, and C_B a scaling coefficient, and r_B a help variable.

The motion vector angle can then be calculated through

$$\theta_B = \frac{\pi}{2} \cdot \exp\left(\frac{{r_B}^2}{{\sigma_B}^2}\right) \cdot sign(\omega_B) \tag{12}$$

resulting in a motion vector of

$$X_B^{CA} = \begin{cases} a \cdot cos(\theta_B) \cdot sign(u_B) \\ a \cdot sin(\theta_B) \end{cases} \tag{13}$$

With σ_B a numerical parameter and a being defined as: $a = max(\|u_B\|, \|X^{tot}\|)$.

3.6.3. Combination of behaviors

In this work, two methods to combine the resulting motion vectors of the previous sections X^{tot} and X^{CA} are considered. Following a Sequential Behavior Coordination (SBC) approach, similar to the Subsumption Architecture proposed by [26], the resulting X^{res} is:

$$X^{res} = \begin{cases} X^{tot} & if \|X^{CA}\| = 0 \\ X^{CA} & if \|X^{CA}\| > 0 \end{cases} \tag{14}$$

Following a Parallel Behavior Coordination (PBC), similarly to the Motor Schema approach proposed by [27], the resulting X^{res} is:

$$X^{res} = X^{CA} + X^{tot} \tag{15}$$

3.7. Robot control

The resulting combined motion vector for the ith robot, X_i^{res}, can be transformed into polar coordinates:

$$\begin{cases} e_i &= \sqrt{X_{x,i}^{res2} + X_{y,i}^{res2}} \\ \alpha_i &= atan2\left(X_{x,i}^{res}, X_{y,i}^{res}\right) \end{cases} \tag{16}$$

Where $X_{x,i}^{res}, X_{y,i}^{res}$ correspond to the x and y components of X_i^{res} respectively.

We can then use the single integrator control law given by:

$$\begin{cases} u_i &= c_1 \cdot e_i^{\frac{3}{2}} \cdot cos(\alpha_i) \\ \omega_i &= c_1 \cdot sin(\alpha_i) \cdot cos(\alpha_i) + c_2 \cdot \alpha_i \end{cases} \tag{17}$$

Where u_i corresponds to the forward movement of robot i and ω_i to its rotational movement. Numeric values for c_1 and c_2 can be found in Table 2.

This control law improves on the one initially proposed in [28] by adding an additional damping term. This allows the robot to better cope with unfiltered sensor values without oscillations.

Lemma 1. *If the underlying graph of $\mathcal{L}$ is connected and no biases are present, then the control law of Eq. (17) will stabilize the system to a final common value for all c_1, c_2 where $\frac{c_2}{c_1} > 0.19893$ and $c_1, c_2 > 0$.*

The proof is based on a Lyapunov function which can be analyzed for local maxima, similar to [28] and is thus omitted here.

3.7.1. Robot control with heading alignment

Controlling a single vehicle to its target position while at the same time aligning its heading into a given orientation has been done, for example in [29]. However, this algorithm does only work if the goal position and orientation are known well in advance. In combination with a reactive graph-based Laplacian control law, its performance is therefore unsatisfactory. This can be explained intuitively with the fact that we aim at controlling a three dimensional pose (x, y, θ) using two controllable variables v_r and v_l, the speeds of the right and left wheel, respectively. Therefore, the system is only controllable if one dimension of the pose can be expressed as a function of the other two, which is possible through the expression $\theta = atan(\frac{\dot{y}}{\dot{x}})$. In order for this expression to be true, the final speed of the robot $(\dot{x}, \dot{y})$ before attaining the desired pose needs to be in the direction of the desired θ, which is only possible if the desired pose is known well enough in advance in order to generate a path complying with this condition. Unfortunately, this is not the case for a graph-based Laplacian formation control since at every step the positions of all robots are taken into account to update the desired final position.

Therefore, in order for this control law to be compatible with a graph-based Laplacian one, we consider a two-stage approach here: initially, the robots are controlled towards their desired position as before; then, once the distance to the desired position becomes smaller, the alignment factor becomes more prominent. Therefore, Eq. (17) gets transformed into:

$$\begin{cases} u_i &= c_1 \cdot e_i^{\frac{3}{2}} \cdot cos(\alpha_i) \\ \omega_i &= (1 - C) \cdot (c_1 \cdot sin(\alpha_i) \cdot cos(\alpha_i) + c_2 \cdot \alpha_i) + C \cdot c_5 \cdot X_{\theta,i}^{res} \end{cases} \tag{18}$$

with

$$C = 1 - \frac{1}{1 + exp(-c_3 \cdot (e_i - c_4))} \tag{19}$$

Where $X_{\theta,i}^{res}$ corresponds to the goal orientation as seen from the robot's coordinate frame. The numerical values of the coefficients c_3, c_4, c_5 can be found in Table 2.

4. Experiments

In Section 4.1, we first introduce the general experimental set-up, including the robotic platform. Then we present the three sets of experiments which have been designed to challenge the proposed functional framework. Their results are also presented in their respective Sections 4.2–4.4 . Each experiment aims at shedding light on a specific research question on its own while focusing on challenging one or multiple FEs.

Table 2
Numerical parameters chosen for the various algorithmic components, organized by FE.

Parameter	Numerical value	Unit
Robot diameter	0.14	[m]
Arena size	3 x 3	[m]
Proximity sensor range R_p	0.2	[m]
Relative sensing range R_r	2	[m]
Communication range R_c	$\gg$ arena size	[m]
Timestep dt	50	[ms]
Ideal flocking range Q	0.6	[m]
Flocking tolerance ϵ	0.15	[m]
Scaling coefficient c	0.5	[−]
Weighting function $w(x)$	$\begin{cases} 21 & \forall\ x \in [0, 16] \\ 21-(x-16)^2 & \forall\ x \in]16, 20] \\ 0 & \forall\ x \in]20, \infty] \end{cases}$	[−]
Proximity sensor count S_B	8	[−]
Braitenberg coefficients W_B	$\begin{bmatrix} -7, -3, -2, -0.5, -0.5, +4, +5, +7 \\ +7, +5, +4, -0.5, -0.5, -2, -4, -6 \end{bmatrix}$	[−]
$\sum W_B$ per dimension, T_B	29	[−]
Scaling coefficient C_B	10	[−]
Scaling coefficient σ_B	1.5	[m]
Robot control parameter c_1	3	[−]
Robot control parameter c_2	13	[−]
Robot control parameter c_3	1000	[−]
Robot control parameter c_4	0.08	[−]
Robot control parameter c_5	100	[−]

4.1. Experimental set-up

All experiments were conducted using both simulated and real Khepera IV robots [20] enhanced by a custom-built range-and-bearing module [30] for relative localization and an active marker module featuring two LEDs for enabling accurate tracking with the SwisTrack software [31] (see Fig. 3). Khepera IV robots are differentially driven vehicles with a diameter of 14 cm and a maximal speed of $\sim$ 81 cm/s. They are equipped with eight infrared (IR) proximity sensors with a range R_p of 20 cm with noise increasing linearly from a standard deviation of 0% at 0 cm to 10% at 20 cm. They are equally distributed around the robot, resulting in an angle between two sensors of 45°. The obtained obstacle positions O_k include thus noise with standard deviation up to 10% of the distance, and discretization errors of up to 22.5° on the bearing. Obstacles consist in walls of the arena in addition to fellow robots being within the proximity sensor range. The range-and-bearing module with maximal range R_r of 2 m suffers, as shown in Fig. 4, from a 10% noise ratio in the estimation of the distance and 0.1 radians ($\sim$ 6°) noise in the relative angle. Its refresh rate is set to 50 ms. Communication between the robots is realized using UDP/IP and assumed to have a range R_c way larger than the dimensions of the enclosed experimental arena. Simulations have been carried out in Webots [32], an open-source, high-fidelity robotics simulation platform. For the simulated robots, sensors and actuators were calibrated to match those of the real robots.

For the physical experiments, SwisTrack [31] is used in combination with a GigE color camera to both track the robots in order to calculate the evaluation metrics, as well as to emulate absolute localization such as that provided by a GNSS module in indoor settings. SwisTrack is calibrated using the algorithm proposed by Tsai et al. [33] which, in our settings, results in a spatial error of 1.06 +/−0.65 cm and a neglectable temporal error of less than 1 mm amplitude. Emulated GNSS information is sent to the robots via UDP/IP at a rate of 10 Hz.

For real experiments, every scenario is repeated ten times, and simulated experiments are repeated 20 times. During every run, the positions of the robots are monitored, the evaluation metrics calculated and logged. When available, communication is assumed to be global over the operational area and without added noise or packet loss. We report, however, approximately 2% of intrinsic packet loss for real robot experiments due to the selected communication protocol.

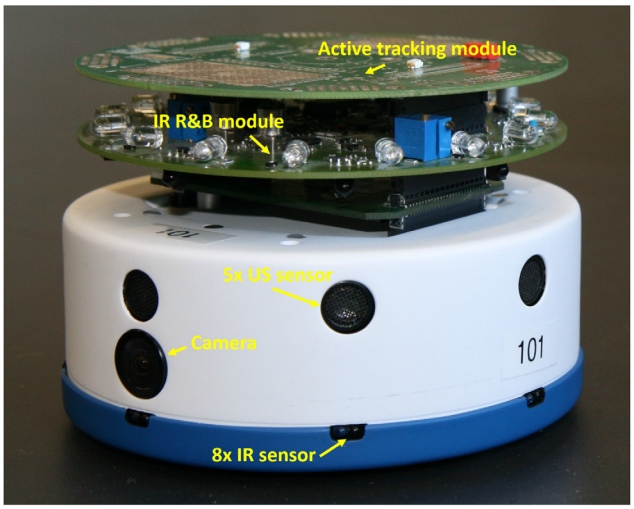


Fig. 3. Khepera IV robot with the custom built range-and-bearing module as well as an additional module featuring two LEDs for tracking purpose. The available Ultra-Sound (US) and camera sensors were not used in this work.

4.2. Communication and localization robustness

The following experiments have been designed to challenge and assess diverse design solutions for FE1. More specifically, the sensing and communication channels exploitable for estimating the own pose and that of the other robots are varied.

4.2.1. Experiment description

Four Khepera IV robots are operating in a 3 m × 3 m arena (see Fig. 5). The robots are tasked to assume a diamond formation and drive twice through an 8-shape in 80 s at an average speed of $\sim$ 15 cm/s while being subject to different sensory and communication constraints: non-communicating robots (Disconnected) provided with Absolute localization information (DA), Networked with Absolute localization (NA), Networked with Relative localization (NR), Networked with Absolute and Relative localization (NAR). Under conditions NA, NR, and NAR, robots are placed randomly within 1.5 m × 1.5 m area at the beginning of the experiment, where under condition DA robots have a fixed orientation with an accuracy of approximately 10° within the same area. In condition DA, robots have only their absolute pose available; in condition NA, they dispose of the same absolute pose, but have communication to enable each robot to broadcast the belief of its pose. In condition NR, the robots have access to relative localization as well as communication. Condition NAR combines the conditions NA and NR: robots have access to both absolute and relative localization as well as communication. A summary of the resources available per condition is reported in Table 3.

The common goal in the form of an 8-shape trajectory is given to every robot as temporal changing speed offset ($T(t) = \{T_x(t), T_y(t)\}$) with respect to the orientation consensus direction.

The offset can be expressed in robot i's coordinates as:

$$\begin{bmatrix} T_{x,i}(t) \\ T_{y,i}(t) \end{bmatrix} = \begin{bmatrix} cos(\theta_i(t)) & sin(\theta_i(t)) \\ -sin(\theta_i(t)) & cos(\theta_i(t)) \end{bmatrix} \cdot \begin{bmatrix} T_x(t) \\ T_y(t) \end{bmatrix} \quad (20)$$

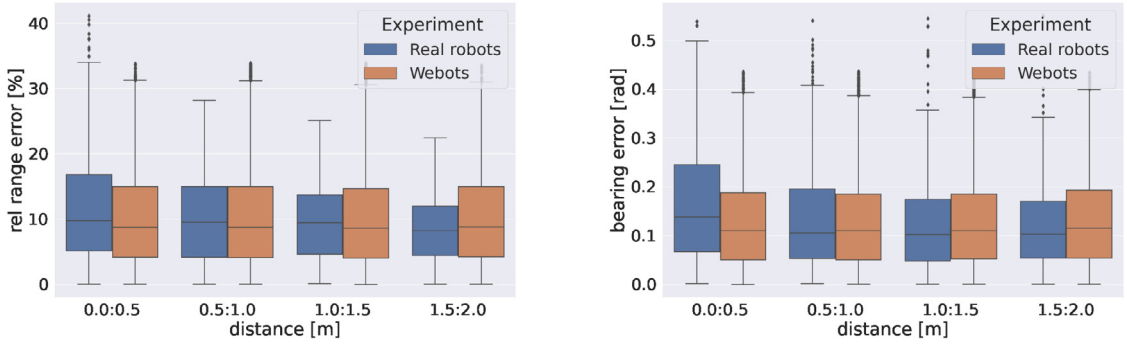


Fig. 4. Range-and-bearing module estimation noise on real Khepera IV robots and in Webots.

Fig. 5. Four Khepera IV robots in a diamond formation in the 3 m × 3 m arena superimposed by the 8-shape they are tasked to drive in reality (left) and Webots (right).

Table 3
Resources available in every condition.

	DA	NA	NR	NAR
Absolute localization	X	X		X
Networking		X	X	X
Relative localization			X	X

With $\theta_i(t)$ being the current consensus on the formation's orientation in robot i's coordinates.

Since this is a scenario centered around collective navigation in the environment depicted in Fig. 5, no collision avoidance algorithm is needed. Therefore, FE 6 is not present in this experiment.

The cumulative formation error over time, normalized by the number of robots serves as the common evaluation metric for this experiment. For each timestep t this error E is defined as:

$$E(t) = \sum_{r=1}^{N} \frac{\|P_r(t) - F_r(t)\|}{N} \tag{21}$$

where P_r corresponds to the actual absolute position of robot r and F_r to the target position of robot r, given an optimal alignment of the formation onto the robots. The optimal alignment during the 8-shape trajectory is found through post-experiment optimization of the target formation alignment angle. The formation error can thus be calculated for each timestep using the ground truth recorded by a non interfering supervisor. A zero error in Eq. (21) would imply a perfect alignment of the robots to their target formation.

Additionally, in order to further show the modularity potential of the proposed framework and robustness of the overall navigation algorithm, we carried out experiments affected by on-line switching between the sensing and communication constraints.

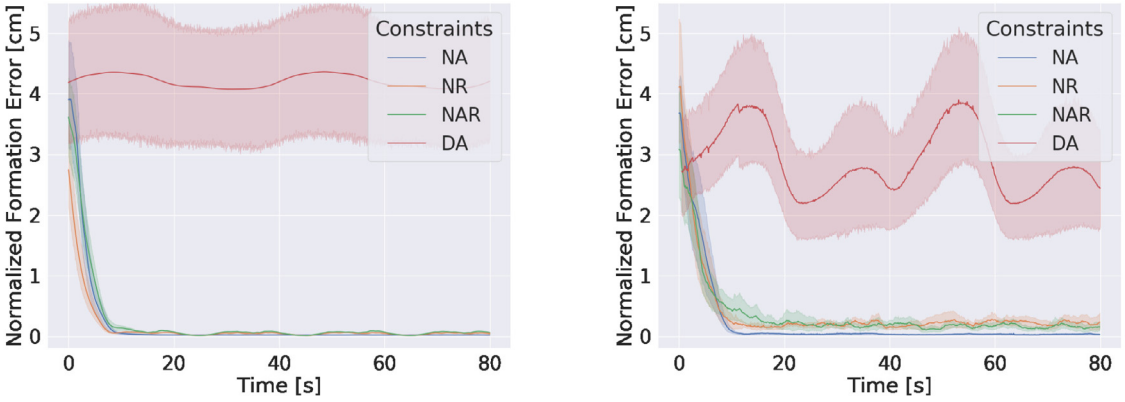
4.2.2. Results

Fig. 6(a) depicts the results of the simulated runs under the different scenarios, while Fig. 6(b) those of the real runs under the same scenarios. As it can be seen in Fig. 6, the resulting performance remains stable independent of the localization information available as long as the robots are connected through communication. The differences between simulation and reality are mostly negligible except for the DA condition. This is explainable by the fact that, for this condition, the performance depends highly on the randomized initialization of the experiment.

The oscillations in performance observable in NR and NAR conditions can be explained by the fact that the robots are tasked to drive an 8-shape. During the lateral edges of the 8-shape, when the range-and-bearing estimates tend to be noisier due to turning, errors grow more easily than during the straight segments, resulting in the oscillatory performance. These oscillations do not occur for the NA condition as there (almost) no noise is present and, additionally, the information given through (simulated) GNSS signals is not noisier during turns either. Finally, the oscillatory behavior of the DA condition is due to another effect. Due to the lack of communication, the robots cannot reach a perfect consensus of the direction of the 8-shape. Thus depending on their position within the shape, the formation error changes.

In Fig. 7, the performance during the on-line switching of localization and communication conditions shows that the combination of algorithms used are highly resistant to such variable constraints. In fact, the performance of the overall algorithm remains comparable in all conditions. It is worth noting that the performance of the system in the DA condition is significantly better in this on-line switching experiment than in the separate evaluations (Fig. 6). This is due to the fact that the error in the DA condition is a function of the initial conditions, and here the robots are almost perfectly aligned at the beginning of the DA condition thanks to the previous condition. They continue to rely on the information gathered before connection blackout and are thus able to perform relatively well.

Through this comparison of formation error under different sensor and communication constraints, we have shown that an implementation according to our framework can achieve almost complete independence of the underlying localization (or more



(a) Results in simulation (b) Results in reality

Fig. 6. Resulting formation error under different localization and communication constraints.

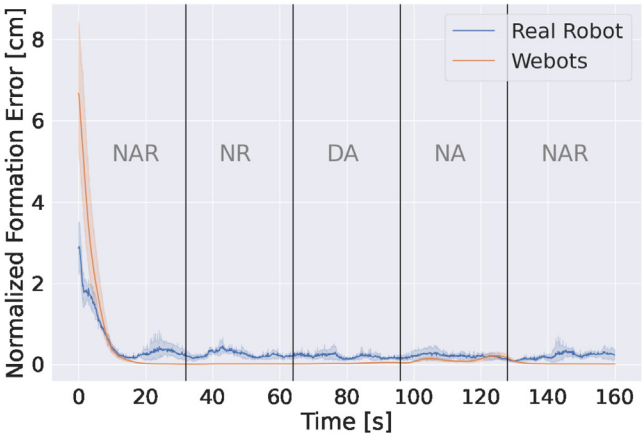


Fig. 7. Simulated and real Khepera IV robots driving an 8-shape in a diamond formation and switching localization and communication conditions during run time.

generally speaking sensing) technology, allowing also for online switching between different conditions serving the same functionality.

4.3. Collective navigation in presence of obstacles

The following experiments have been designed to challenge and assess different combinations of collective navigation and collision avoidance algorithms concerning FE 3 and FE 6, respectively. More specifically, they are evaluated for their performance in presence of obstacles, including an algorithm choosing online between formation control and flocking, effectively dynamically changing FE 4.

4.3.1. Experiment description

Four Khepera IV robots are placed randomly within the starting location of an arena featuring a narrow passage, as shown in Fig. 8. The length of the passage is 3 m and the width changes from 3 m to 0.5 m over the first 2 m of the passage. The robots are tasked to assume a line formation (orthogonal to the direction of motion) and pass through the narrow passage. This setup allows the robots to initially form the prescribed formation, which is then increasingly impossible to maintain given the slanted walls. As a result, the performance of the interaction of CA and a given spatial coordination algorithm becomes apparent. The robots are either using the graph-based Laplacian control law or the control law consisting of a finite state machine choosing between navigation in formation or flocking. In this work, the criterion for switching to flocking is a detected obstacle closer than 15 cm. However, this criterion could easily be substituted with a more elaborated one, for instance based on the obstacle density perceived around the ensemble, as mentioned before.

Both the formation control and the flocking algorithm presented in Section 3.4 are augmented by the CA algorithms presented in Section 3.6. Both behavioral combination mechanisms presented in Section 3.6.3 are considered and compared for this scenario.

It is worth noting that the combination of SBC for arbitrating between CA and a mixed flocking-formation controller does not make sense, as in the vicinity of obstacles the robots switch to CA state and thus completely ignore the flocking controller. The behavior of it is thus identical to that having only the formation control component. Table 4 summarizes the algorithmic combinations constituting the resulting control software stack.

The time needed to traverse the narrow passage, as well as a performance metric related to collision avoidance are used as common evaluation metrics for all six combinations. The CA metric is defined as follows:

$$E_{CA}(t) = \text{mean}(D(t)) \; \forall \; t \in [0, T] \mid D(t) \leq 20cm \tag{22}$$

Where D(t) corresponds to the distance towards the closest obstacle and T is the time needed to pass the narrow passage. It is worth noting that only distances shorter than 20 cm are taken into account, which corresponds to the maximal range of the Khepera IV's infra-red sensors, as mentioned above. Therefore, this metric truly corresponds to the performance of the algorithm once an obstacle is effectively detectable.

4.3.2. Results

As can be seen in Fig. 9(a), the PBC algorithms perform significantly better in simulation but struggle to keep such performance in reality (note the high failure rate, represented by red crosses, for all their variants in reality, including the 100% failure rate for the combination with Laplacian CA and formation control). The SBC versions, on the other hand, perform similarly in reality and simulation.

Fig. 9(b) shows the obstacle avoidance metric for the same experiments. We note the smaller distance towards obstacles that are kept in simulation for the PBC algorithms. In fact due to the non-ideal situation in reality, this margin seems not to be enough for real robots. As a result, they bump into obstacles and get stuck, which explains the poor behavior of these algorithms in

Table 4
Algorithmic combinations evaluated for collective navigation in presence of obstacles.

Label	Navigation algorithm	CA algorithm	Behavioral coordination
Lapl PBC, formation only	Formation (Section 3.4.1)	Laplacian (Section 3.6.1)	Parallel (Section 3.6.3)
Lapl PBC, formation & flocking	Formation and flocking (Section 3.4.3)	Laplacian (Section 3.6.1)	Parallel (Section 3.6.3)
Braiten PBC, formation only	Formation (Section 3.4.1)	Braitenberg (Section 3.6.2)	Parallel (Section 3.6.3)
Braiten PBC, formation & flocking	Formation and flocking (Section 3.4.3)	Braitenberg (Section 3.6.2)	Parallel (Section 3.6.3)
Lapl SBC, formation only	Formation (Section 3.4.1)	Laplacian (Section 3.6.1)	Sequential (Section 3.6.3)
Braiten SBC, formation only	Formation (Section 3.4.1)	Braitenberg (Section 3.6.2)	Sequential (Section 3.6.3)

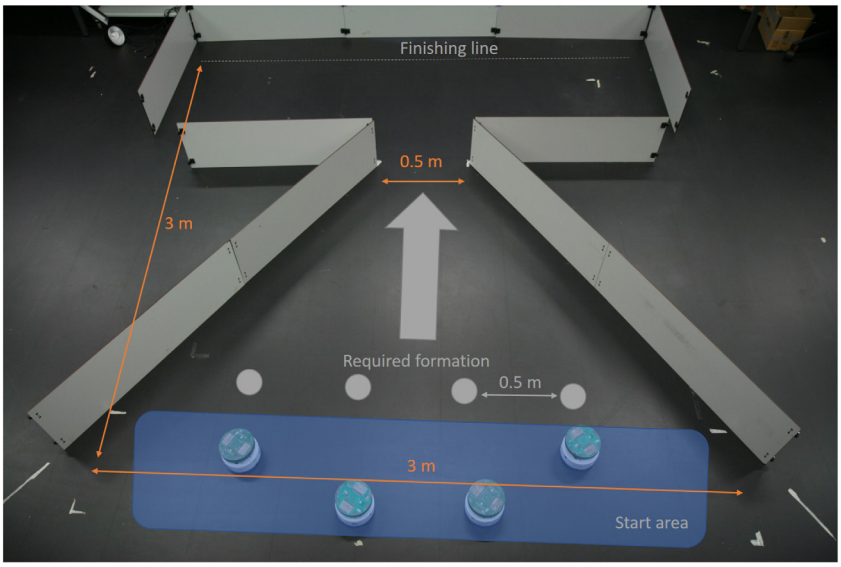
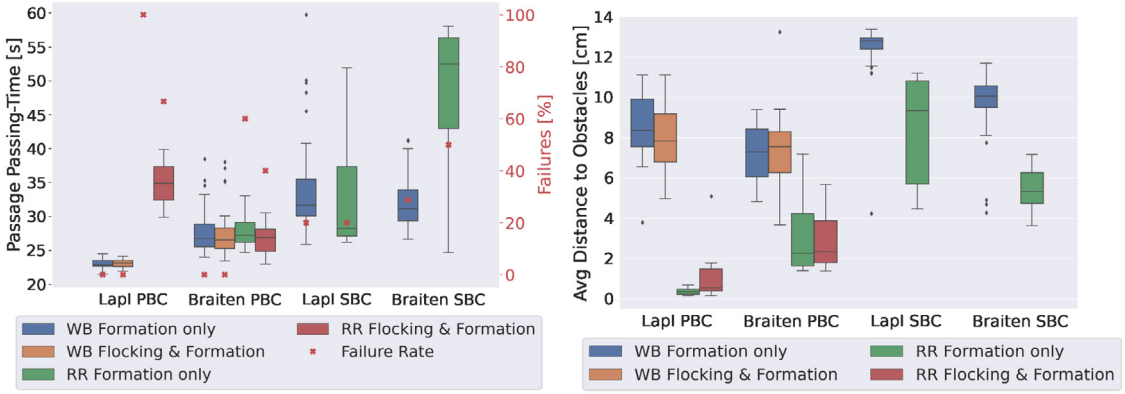


Fig. 8. The narrow passage setup in reality with overlaid distances, required formation, start area and finishing line.



(a) Speed performance **(b)** Collision avoidance performance

Fig. 9. Results for different algorithmic combinations in both simulation and reality of the collective navigation scenario in presence of obstacles (in this case a narrow passage as depicted in Fig. 8); WB = Webots; RR = real robots; flocking and formation bars are missing in combination with SBC for the reason mentioned above.

reality. The SBC versions, on the other hand, cope better with this additional noise due to their larger maneuvering margin.

In Fig. 9(b) the difference between the mixture of flocking and formation control compared to the pure flocking is almost nonexistent for a given CA algorithm. However, in Fig. 9(a), especially for the real robots, the mixture seems to perform better overall (e.g., the Lapl PBC's failure rate decreases by 40%), which corresponds to our expectations due to the nature of flocking being more flexible to shape deformation.

We can therefore conclude that, for more constraint spaces, a mixture of simple flocking and basic formation control seems to be more ideal for real world imperfections than a basic formation control. However, in future, the mixture should not only take into account more sophisticated flocking algorithms, but also compare to (or leverage) work including dynamically changing swarm algorithms such as in [34] or more flexible formation control such as in [35].

4.4. Heading alignment

The following experiments have been designed to challenge and assess diverse design solutions for FE7.

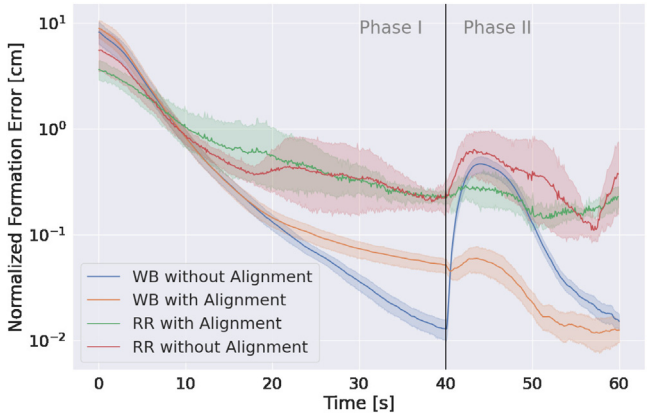


Fig. 10. Comparison of control with and without heading alignment in simulation and reality; WB: Webots; RR: real robots.

4.4.1. Experiment description

Similar to Section 4.2, four Khepera IV robots are placed randomly within a 1.5 m × 1.5 m area in the middle of a 3 m × 3 m arena. The robots are then tasked to assume a diamond formation (Phase I) and, after a delay of 40 s, start to drive around an 8-shape (Phase II). The robots are set in the NAR condition and use either the control law of Section 3.7 not taking into account the heading, or the control law proposed in Section 3.7.1 which includes heading alignment. The cumulative formation error over time (cf. Eq. (21)) serves as the common evaluation metric.

4.4.2. Results

Fig. 10 reports the results of the comparison of the control with and without heading alignment. In simulation, the achieved formation error prior to movement is lower for the control not taking into account heading alignment (Phase I). However, the formation error during the first moments of movement is significantly smaller for the control taking into account the heading alignment (Phase II). The real experiments, on the other hand, show that the larger error due to the heading alignment algorithm prior movement is negligible in comparison to the larger formation errors due to the non-perfect reality (Phase I). Also in reality, the heading alignment algorithm helps keeping the formation error low while moving (Phase II).

We therefore conclude that while an error-dependent parameter for balancing heading alignment and positional control seems to be a trade-off between static and start-of-movement formation errors in simulation, the additional error introduced by this modified control law is negligible in reality when compared to other noises present. Such an algorithm might therefore be preferable for any graph-based Laplacian formation generation where subsequent movement has to be expected.

5. Conclusion

In this work, we have introduced a modular functional framework that guides the implementation and evaluation of multiple concatenated Functional Elements (FEs) illustrated in Table 1. Such framework has allowed not only to increase our code reusability, but also to easily compare and consider multiple algorithms for each FE, as well as the impact of specific combinations on the resulting overall navigation performance. In order to demonstrate this, we have undertaken four comparison experiments, modifying different FEs:

- For FE 1, we compared the formation error of a graph-based Laplacian formation control under different sensor and communication constraints. We also showed that online modifications of the constraints are handled smoothly.
- By changing FE 4, we compared pure formation control with a mixed formation-flocking control.
- We compared two reactive CA algorithms as well as two combination schemes integrating these algorithms into the collective movement (FE 6).
- By changing FE 7, we compared a control law leveraging an error-dependent parameter for balancing heading alignment and positional control against a classical control law without heading alignment.

All four comparisons have been done both in simulation and on real robots. The real robotic platform we leveraged for this study, the Khepera IV, while able to run a Linux operating system on board, can only be endowed with a simple middleware (the Khepera IV Toolbox [20]). Implementing our functional framework on this platform required some significant coding effort, but the modularity and portability of the resulting code was certainly worth the effort, as demonstrated by the variety of the experiments reported in this paper. Moreover, the framework has eased the transition to real robots as interactions with hardware are concentrated into only two FEs (FE 1 for sensing and communication and FE 7 for actuating). Of course, if compatible with the targeted hardware, the use of a more powerful middleware such as ROS or ROS2 [18], with freely available packages for the individual algorithms needed by the FEs, as well as for the interfaces between them (e.g., through custom topics), further accelerates the development process. However, we are strongly convinced that, even in situations where hardware resources allows for such powerful middleware deployment, our modular functional framework offers complementary guidance to the software design process, thanks to its functional abstraction.

In the future, we would like to demonstrate such complementarity by leveraging ROS or ROS2 as middleware. This will allow us to benefit of the vast amount of available algorithms to efficiently compare possible combinations of them to generate collective movement. Moreover, such effort will demonstrate further the generality and portability of our framework, while at the same time offering an opportunity to disseminate its source code to a way larger palette of robotic platforms. We are highly confident that the proposed functional framework is fully scalable in terms of number of robots since it is built for distributed algorithms without direct dependence on the robot numbers, but this should be verified experimentally. Finally, we further intend to leverage the functional framework for exploring the design of more complex collective movements, possibly involving additional nonlinear and deliberative components in their underlying control laws.

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.

Acknowledgments

We would like to thank Masaki Haruna and Masahiko Kurishige of the Advanced Technology R&D Center, Mitsubishi Electric Corporation, Amagasaki City, Japan for the fruitful technical discussions all along this work.

References

[1] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, J. Spletzer, Distributed search and rescue with robot and sensor teams, in: S. Yuta, H. Asama, E. Prassler, T. Tsubouchi, S. Thrun (Eds.), Field and Service Robotics: Recent Advances in Reserch and Applications, in: Springer Tracts in Advanced Robotics, Springer, 2006, pp. 529–538, http://dx.doi.org/10.1007/10991459_51.

[2] R.W. Beard, F.Y. Hadaegh, Constellation templates: an approach to autonomous formation flying, in: World Automation Congress, 1998.

[3] S. Santini, A. Salvi, A.S. Valente, A. Pescapè, M. Segata, R.L. Cigno, Platooning maneuvers in vehicular networks: a distributed and consensus-based approach, IEEE Trans. Intell. Veh. 4 (1) (2019) 59–72, http://dx.doi.org/10.1109/TIV.2018.2886677.

[4] K.-K. Oh, M.-C. Park, H.-S. Ahn, A survey of multi-agent formation control, Automatica 53 (2015) 424–440, http://dx.doi.org/10.1016/j.automatica.2014.10.022.

[5] Y. Liu, R. Bucknall, A survey of formation control and motion planning of multiple unmanned vehicles, Robotica 36 (7) (2018) 1019–1047, http://dx.doi.org/10.1017/S0263574718000218.

[6] A. Widyotriatmo, E. Joelianto, A. Prasdianto, H. Bahtiar, Y.Y. Nazaruddin, Implementation of leader-follower formation control of a team of nonholonomic mobile robots, Int. J. Comput. Commun. Control 12 (6) (2017) 871–885, http://dx.doi.org/10.15837/ijccc.2017.6.2774.

[7] W. Ren, N. Sorensen, Distributed coordination architecture for multi-robot formation control, Robot. Auton. Syst. 56 (4) (2008) 324–333, http://dx.doi.org/10.1016/j.robot.2007.08.005.

[8] P.K.C. Wang, Navigation strategies for multiple autonomous mobile robots moving in formation, J. Robot. Syst. 8 (2) (1991) 177–195, http://dx.doi.org/10.1002/rob.4620080204.

[9] J.P. Desai, J. Ostrowski, V. Kumar, Controlling formations of multiple mobile robots, in: IEEE International Conference on Robotics and Automation (Cat. No.98CH36146), Vol. 4, 1998, pp. 2864–2869 vol.4, http://dx.doi.org/10.1109/ROBOT.1998.680621.

[10] J. Fredslund, M. Matarić, A general algorithm for robot formations using local sensing and minimal communication, IEEE Trans. Robot. Autom. 18 (5) (2002) 837–846, http://dx.doi.org/10.1109/TRA.2002.803458.

[11] N. Basilico, N. Gatti, F. Amigoni, Leader-follower strategies for robotic patrolling in environments with arbitrary topologies, in: International Conference on Autonomous Agents and Multi-Agent Systems, 2009, pp. 57–64, http://dx.doi.org/10.1145/1558013.1558020.

[12] K.-H. Tan, M.A. Lewis, Virtual structures for high-precision cooperative mobile robotic control, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1, 1996, pp. 132–139, http://dx.doi.org/10.1109/IROS.1996.570643.

[13] M. Mesbahi, M. Egerstedt, Graph theoretic methods in multiagent networks, in: Princeton Series in Applied Mathematics, 2010, http://dx.doi.org/10.1515/9781400835355.

[14] S. Gowal, A. Martinoli, Bayesian Rendezvous for distributed robotic systems, in: IEEE International Conference on Intelligent Robots and Systems, 2011, pp. 2765–2771, http://dx.doi.org/10.1109/IROS.2011.6094976.

[15] R. Falconi, S. Gowal, A. Martinoli, Graph-based distributed control of nonholonomic vehicles endowed with local positioning information engaged in escorting missions, in: IEEE International Conference on Robotics and Automation, 2010, pp. 3207–3214, http://dx.doi.org/10.1109/ROBOT.2010.5509139.

[16] Z. Han, L. Wang, Z. Lin, R. Zheng, Formation control with size scaling via a complex Laplacian-based approach, IEEE Trans. Cybern. 46 (10) (2016) 2348–2359, http://dx.doi.org/10.1109/TCYB.2015.2477107.

[17] Stanford Artificial Intelligence Laboratory, et al., Robotic operating system, URL https://www.ros.org.

[18] Y. Maruyama, S. Kato, T. Azumi, Exploring the performance of ROS2, in: International Conference on Embedded Software, Association for Computing Machinery, 2016, pp. 1–10, http://dx.doi.org/10.1145/2968478.2968502.

[19] C.W. Reynolds, Flocks, herds and schools: a distributed behavioral model, in: Annual Conference on Computer Graphics and Interactive Techniques, ACM, 1987, pp. 25–34, http://dx.doi.org/10.1145/37401.37406.

[20] J.M. Soares, I. Navarro, A. Martinoli, The Khepera IV mobile robot: performance evaluation, sensory data and software toolbox, in: Robot 2015: Second Iberian Robotics Conference, Vol. 417, Springer International Publishing, 2015, pp. 767–781, http://dx.doi.org/10.1007/978-3-319-27146-0_59.

[21] D. Fraser, J. Potter, The optimum linear smoother as a combination of two optimum linear filters, IEEE Trans. Automat. Control 14 (4) (1969) 387–390, http://dx.doi.org/10.1109/TAC.1969.1099196.

[22] H.W. Kuhn, The hungarian method for the assignment problem, Nav. Res. Logist. Q. 2 (1–2) (1955) 83–97, http://dx.doi.org/10.1002/nav.3800020109.

[23] E. Soria, F. Schiano, D. Floreano, The influence of limited visual sensing on the Reynolds flocking algorithm, in: Third IEEE International Conference on Robotic Computing, 2019, pp. 138–145, http://dx.doi.org/10.1109/IRC.2019.00028.

[24] R. Falconi, L. Sabattini, C. Secchi, C. Fantuzzi, C. Melchiorri, Edge-weighted consensus-based formation control strategy with collision avoidance, Robotica 33 (2014) 1–16, http://dx.doi.org/10.1017/S0263574714000368.

[25] V. Braitenberg, Vehicles: Experiments in Synthetic Psychology, MIT Press, Cambridge, MA, 1984.

[26] R. Brooks, A robust layered control system for a mobile robot, IEEE J. Robot. Autom. 2 (1) (1986) 14–23, http://dx.doi.org/10.1109/JRA.1986.1087032.

[27] R.C. Arkin, Motor schema — based mobile robot navigation, Int. J. Robot. Res. 8 (4) (1989) 92–112, http://dx.doi.org/10.1177/027836498900800406.

[28] R. Falconi, S. Gowal, J. Pugh, A. Martinoli, Graph-based distributed control for non-holonomic vehicles engaged in a reconfiguration task using local positioning information, in: Second International Conference on Robot Communication and Coordination, 2009, pp. 1–6, http://dx.doi.org/10.4108/ICST.ROBOCOMM2009.5865.

[29] M. Aicardi, G. Casalino, A. Bicchi, A. Balestrino, Closed loop steering of unicycle like vehicles via Lyapunov techniques, IEEE Robot. Autom. Mag. 2 (1) (1995) 27–35, http://dx.doi.org/10.1109/100.388294.

[30] J. Pugh, X. Raemy, C. Favre, R. Falconi, A. Martinoli, A fast onboard relative positioning module for multirobot systems, IEEE/ASME Trans. Mechatronics 14 (2) (2009) 151–162, http://dx.doi.org/10.1109/TMECH.2008.2011810.

[31] T. Lochmatter, P. Roduit, C. Cianci, N. Correll, J. Jacot, A. Martinoli, SwisTrack - a flexible open source tracking software for multi-agent systems, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 4004–4010, http://dx.doi.org/10.1109/IROS.2008.4650937.

[32] O. Michel, WebotsTM: professional mobile robot simulation, Int. J. Adv. Robot. Syst. 1 (2004) http://dx.doi.org/10.5772/5618.

[33] R. Tsai, A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, IEEE J. Robot. Autom. 3 (4) (1987) 323–344, http://dx.doi.org/10.1109/JRA.1987.1087109.

[34] M. Wahby, J. Petzold, C. Eschke, T. Schmickl, H. Hamann, Collective change detection: adaptivity to dynamic swarm densities and light conditions in robot swarms, Artif. Life Conf. Proc. (31) (2019) 642–649, http://dx.doi.org/10.1162/isal_a_00233.

[35] A. Wasik, J.N. Pereira, R. Ventura, P.U. Lima, A. Martinoli, Graph-based distributed control for adaptive multi-robot patrolling through local formation transformation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2016, pp. 1721–1728, http://dx.doi.org/10.1109/IROS.2016.7759276.

Cyrill Baumann received a M.Sc in Microengineering from the Swiss Federal Institute of Technology in Lausanne (EPFL) in 2018. Currently, he is a Ph.D. student in the Distributed Intelligent Systems and Algorithms Laboratory at EPFL. His research focuses on modeling and control design for multi-robot collective movement.

A. Martinoli has a M.Sc. in Electrical Engineering from the Swiss Federal Institute of Technology in Zurich (ETHZ), and a Ph.D. in Computer Science from the Swiss Federal Institute of Technology in Lausanne (EPFL). He is currently an Associate Professor at the School of Architecture, Civil, and Environmental Engineering and the head of the Distributed Intelligent and Algorithms Laboratory. Before joining EPFL he carried out research activities at the Institute of Biomedical Engineering of the ETHZ, at the Institute of Industrial Automation of the Spanish Research Council in Madrid, Spain, and at the California Institute of Technology, Pasadena, U.S.A. His research interests focus on methods to design, control, model, and optimize distributed cyber–physical systems, including multi-robot systems, sensor and actuator networks, and intelligent vehicles.