Decentralized Safe Conflict Resolution for Multiple Robots in Dense Scenarios

Eduardo Ferrera[a,*], Jesus Capitán[b], Angel R. Castaño[b], Pedro J. Marrón[a]

[a] *University of Duisburg-Essen, Essen, Germany*
[b] *University of Seville, Seville, Spain*

Abstract

Multi-robot conflict resolution is a challenging problem, especially in dense environments where many robots must operate safely in a confined space. Centralized solutions do not scale well with the number of robots in dynamic scenarios: a centralized communication can cause bottlenecks and may not be robust enough when channels are unreliable; the complexity of algorithms grows with the number of robots, making online re-computation too expensive in many situations. In this work, we propose a decentralized approach for conflict resolution where robots show reactive and safe behaviors, avoiding collisions with both static and dynamic objects, even under unreliable communication conditions and with low resources. They detect conflicts with neighboring obstacles locally and then apply rules to surround them in a roundabout fashion, assuming that others will follow the same policy. The method is designed for unicycle robots with range-finder sensors, and it is able to cope with noisy sensors and second-order dynamic constraints, ensuring always collision-free navigation. Besides, a set of metrics and scenarios for benchmarking in multi-robot collision avoidance are proposed. We also compare our method with others from the state of the art through extensive simulations. Experiments with real robots are also presented in order to show the feasibility of the system.

Keywords: Dense conflict resolution, decentralized multi-robot systems, safety-enhanced and reactive behaviors, benchmarking, collision avoidance

1. Introduction

Multi-robot systems are becoming more common in the last decades thanks to recent advances in communication, control and perception technologies. The use of cooperative teams of multiple robots, both aerial and ground, is of interest in many applications such as surveillance, traffic management, industrial robots,

*Corresponding author
Email addresses: eduardo.ferrera@uni-due.de (Eduardo Ferrera), jcapitan@us.es (Jesus Capitán), castano@us.es (Angel R. Castaño), pjmarron@uni-due.de (Pedro J. Marrón)

etc. [1, 2]. However, when the number of robots cooperating becomes large, scenarios (mainly indoors) may turn into dense environments. In those highly dynamic scenarios with many robots operating in a confined space, collision avoidance becomes a challenging problem. Robots need to consider trajectories from many other teammates, additional obstacles, and so on; so efficient and safe conflict resolution algorithms become essential. This is even more important as we move into the realm of cooperating aerial vehicles, where a collision no matter how small becomes critical.

Centralized approaches take into account information from all teammates to plan collision-free routes for each robot. However, they do not scale with the size of the team, since they rely on central communication facilities and their complexity usually increase with more robots. Decentralized conflict resolution methods are more adequate for dense teams in cluttered setups, since they are more robust under communication constraints and shared resources. We focus on decentralized approaches where robots can show myopic (or reactive) and safe behaviors even with low resources on board and unreliable communication conditions. This is particularly relevant in highly dynamic scenarios where robots need to react constantly to situations, and re-planning over a horizon time may consume too many resources, or sticking to an initial fixed plan is infeasible.

We propose to detect potential conflicts with neighboring obstacles using local information, and then apply some rules so that robots can avoid each other orderly in a roundabout fashion. The conflict resolution works on the basis that all robots in the team follow the same rules. The underlying idea is somehow similar to the *Generalized Roundabout Policy* [3], which we tailor to unicycle robots (i.e., those that can move in any direction but with a rotation rate associated). Thus, the way each robot reserves space around to avoid collisions is different, since we exploit that unicycle robots can rotate in place, while [3] forces them to circulate in roundabouts without stopping. Besides, we address arbitrary static obstacles, and cope with noisy sensors and second-order dynamic constraints (velocities do not change instantaneously when accelerating).

We extend our previous work [4], where we already proposed a decentralized conflict resolution algorithm for multiple robots. A similar state machine for local robot behavior is still used, but we generalize the detection of potential conflictive obstacles around by assuming range-finder sensors on board. Constant communication among the robots is not assumed, but position information shared by neighbors is used only when available. However, robots never share their orientations, velocities nor goals, and a global map is not required.

Additionally, there have been remarkable efforts recently in benchmarking approaches for robot navigation [5, 6]. However, this is still an open issue, mainly for multi-robot systems. In this paper, we propose a set of metrics and scenarios to benchmark the performance and scalability of multi-robot collision avoidance approaches. We also run extensive simulations to compare different methods in the state of the art, including our proposed algorithm. Moreover, we show experimental results of our system working in a real multi-robot testbed.

In summary, the contributions of our work are the following:

- A novel decentralized algorithm for conflict resolution in dense multi-robot teams based on collision detection with range-finder sensors. Robots require only local information to navigate reactively and safely (without collisions), even with uncertain sensors, unreliable communications and dynamic constraints. We name our algorithm *SWAP: safety-enhanced avoidance policy*.

- The algorithm only requires local information to perform obstacle avoidance. Global positioning information from other robots can be integrated when available. This is considered to make the system more general, but it is not a requirement.

- The system copes with static and dynamic obstacles. Those are integrated in an equivalent manner and robots are able to avoid them without livelocks/deadlocks independently of their shape.

- A thorough discussion on the properties of the algorithm in terms of safety, liveness, robustness, computational complexity and scalability.

- A set of metrics and scenarios for benchmarking in multi-robot collision avoidance. Some of the metrics and scenarios are extracted from the literature and adapted to the multi-robot paradigm. Others are novel. Extensive simulations are presented to compare alternative methods from the state of art within those benchmarking scenarios.

The remainder of the paper is organized as follows: Section 2 surveys related work; Section 3 presents the problem definition; Section 4 describes our algorithm for conflict resolution; Section 5 discusses further the algorithm properties; Section 6 presents benchmarking scenarios and extensive simulation results; Section 7 describes experimental results with real robots; and Section 8 includes conclusions and future work.

2. Related Work

In the literature, the problem of collision-free robot navigation has been widely studied and different classifications are possible [7]. In particular, this paper focuses on dynamic scenarios where there are multiple robots involved. Most related works are based on mathematical programming and control theory, velocity obstacles or potential fields; and hence, a review of these families of methods is presented in the remainder of this section.

Mathematical programming and control theory

Multi-robot path planning has been approached from a control-theoretic point of view. A thorough review of methods based on mathematical programming is provided in [8]. Many of these methods achieve optimal solutions, but they involve complex computation that makes them less suitable for real-time

collision avoidance. Moreover, many of them do not consider other obstacles apart from the robots themselves.

A centralized solution is proposed in [9], splitting the problem into subproblems with less robots that are later executed sequentially (hence, it is not optimal in terms of arrival time). In [2], they use a method for tessellation of the underlying motion area in a number of cells, which are treated as shared resources in a distributed resource allocation problem. The decentralized *Generalized Roundabout Policy* is presented in [3] for conflict resolution, using reserved areas and traffic rules for the vehicles. Under certain assumptions, they guarantee safety and even liveness (i.e., vehicles will reach their destinations in a finite time).

Additionally, it has been reported relevant to consider robots dynamics and to plan over a time horizon (reasoning over future situations) [10]. Even though these methods avoid better local minima, they also require higher computation time and are less adequate for real-time planning. A distributed kinodynamic planner is presented in [10], guaranteeing safe navigation for vehicles with second-order dynamics. Others apply *Model Predictive Control* to achieve collision-free trajectories for multiple robots [11]. In [12], techniques from optimal control and mathematical programming are used to generate continuous velocity profiles that satisfy dynamic constraints, avoid robot collisions, and minimize the task completion time. They show results with non-holonomic robots, but the paths to follow are fixed a priori.

Velocity obstacles

Many authors achieve collision-free trajectories by reasoning about velocity obstacles, i.e., they define regions in the velocity space where collisions are prevented given the current robot velocities. These methods have proven to be very efficient in practise with simulations of large teams of vehicles, but they assume that robots can share or sense velocities and positions of obstacles (and usually shapes). Moreover, most of them do not give theoretical proof of convergence, and do not assure the absence of collisions or livelocks/deadlocks under noisy sensors or second-order dynamics.

The algorithms *Reciprocal Velocity Obstacles* (RVO) [13] and *Optimal Reciprocal Collision Avoidance* (ORCA) [14] introduce the concept of reciprocal velocity obstacles to avoid undesirable oscillations when robots avoid each other (all robots follow a similar behavior). *AccelerationVelocity Obstacles* (AVO) [15] extend RVO, reasoning over second-order dynamics. All these algorithms can select velocities reactively for large teams in a decentralized fashion. ORCA guarantees collision-free trajectories except for densely packed conditions, but given perfect knowledge of positions and velocities of others. Moreover, they prevent certain oscillatory velocity profiles (conflicting robots varying their velocities in a loop because of each other) [13] by selecting velocities in the boundaries of the velocity obstacles. This solution is sensitive to noisy sensors, since observing velocities erroneously may lead robots to collide. Therefore, further extensions are required to deal with inaccurate localization and sensing. In [16], the uncertainties in robot positions are modeled by convex hulls. They report

that using a disk representation is not appropriate in certain situations where the position uncertainty is not typically circular, such as corridors. The *Hybrid Reciprocal Velocity Obstacles* [17] also consider uncertainties in positions and velocities, apart from dealing with non-collaborative obstacles by means of a high-level roadmap. However, collisions and livelocks are still possible.

Additionally, there are extensions of ORCA to cope with certain types of non-holonomic vehicles [18]; or to navigate in 3D environments with multiple aerial robots [1, 19, 20].

Potential fields and vector fields

Approaches based on artificial potential fields are also commonplace in robot navigation. Basically, the robot is kept at a safe distance from obstacles by repulsive forces, while being drawn towards the goal by attractive forces. The *Vector Field Force* (VFF) [21] is one of the first algorithms considering this idea, but they do not show results with multiple robots nor moving obstacles. Potential field methods have low computational requirements and are suitable for real-time collision avoidance, but they do not usually achieve optimal paths, and can present problems with local minima and densely packed environments.

Nonetheless, there also exist the *navigation functions*, which are special types of artificial potential fields that can be designed to have no local minima. In particular, decentralized navigation functions are applied to multi-robot collision avoidance in [22, 23]; whereas [24] propose new theoretical insights by reformulating some assumptions in the previous works that did not hold for multi-robot systems. In [25], they propose *rules of the road* to assign priorities to the vehicles. In [26], the authors contribute with a version of the *Dynamic Windows Approach* (dynamic and kinematic constraints of robots are considered) based on artificial potential fields whose convergence can be proved, but they do not extend the method to multiple robots.

In addition, approaches based on vector field histograms are well known. The original *Vector Field Histogram* (VFH) [27] and its successors VFH+ [28] and VFH* [29], build a polar histogram with local obstacle measurements and look for directions heading to open spaces. They could be applied to multi-robot teams, but they rarely account for robot dynamic nor kinematic constraints, and do not study convergence properties.

Other reactive approaches

In terms of reactive algorithms that only use local information to avoid collisions, there are more approaches that cannot be included in the above categories. This paper focuses on approaches with low computational requirements that can work in real time and scale better with large teams of robots. In this sense, the *bug* algorithms represent a simple approach to navigate toward a goal surrounding static obstacles. A review of these types of algorithms can be found in [30]. They are purely reactive and can converge to the final goal, but they cannot handle dynamic obstacles. The *Bubble* algorithm [31] is another purely reactive method to drive robots around, and it is to some extend similar to the

VFH. Again, the algorithm is not prepared to work with dynamic obstacles, let alone densely packed environments.

3. Problem Definition

We address multi-robot conflict resolution in densely packed environments. In particular, the team of robots must navigate in a 2D scenario with static obstacles of arbitrary shapes and unknown positions. Let $\mathcal{N} = \{1, \ldots, n\}$ be a set of robots with positions $\{p_i(t) = (x_i(t), y_i(t))\}_{i=1,\ldots,n}$ and orientations $\{\theta_i(t) | \theta_i(t) \in [0, 2\pi]\}_{i=1,\ldots,n}$. Let $\mathcal{I} = \{p_1(0), \ldots, p_n(0)\}$ be the initial configuration and $\mathcal{G} = \{g_1, \ldots, g_n\}$ the goal configuration, such that $g_i \in \mathbb{R}^2$, $\forall i$. The problem to solve is the following: *Given the initial configuration $\mathcal{I}$ of the set $\mathcal{N}$, they should be able to navigate to their goal configuration $\mathcal{G}$ without collisions.*

We assume a *unicycle* model for the robots, which means that the direction of their velocity vector is determined by their angular orientation [7]. Orientation changes are limited by a turning rate constraint, but robots can rotate *in situ*, i.e. they can modify their orientation without varying their 2D position (e.g., differential-drive vehicles). Moreover, we consider second-order dynamics in the linear and angular variables, so the robots cannot stop instantly at any time if required. In particular, the kinematic model used for each robot is the following:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix}, \qquad \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{-1}{\tau_v} & 0 \\ 0 & \frac{-1}{\tau_\omega} \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{k_v}{\tau_v} & 0 \\ 0 & \frac{k_\omega}{\tau_\omega} \end{bmatrix} \cdot \begin{bmatrix} u_v \\ u_\omega \end{bmatrix}, \quad (1)$$

where the linear and angular velocities, v and ω, can be controled with the acceleration-based control inputs u_v and u_ω, respectively. The parameters τ_v, k_v, τ_ω and k_ω determine the robot dynamics. Also, the control inputs of the system saturate, in such a way that the linear and angular velocities cannot exceed v_{max} and ω_{max}, respectively.

In terms of sensing capabilities, we assume range-finder sensors for the robots, which allow them to know the distance to obstacles in a finite number of directions around (beams). Therefore, each robot has a polar representation of the obstacles in its neighborhood. Moreover, each robot can localize itself. For this purpose, robots could use the onboard range-finder sensor or any other localization system. We assume noisy sensors, so robots can only observe their positions with a bounded error: we define an uncertainty disk of radius e_l around the observed position where the actual position should lie. The range-finder sensor also produces distance measurements to obstacles up to a maximum range of r_{max} and with a bounded error e_r. We denote the range-finder field of view as fov and the number of beams as n_b.

Besides, there may be a communication system between the neighboring robots to transmit their positions. This information can be incorporated in others' polar representation of obstacles when available. However, our system does not rely on those data and must still work with no communication, since we

assume noisy channels with possible delays, failures, occlusions, etc. The robots do not share any information about their orientations, velocities nor goals.

Our approach does not impose explicitly specific optimization criteria, such as the traveled distance or the time to reach the goal. Instead, we prioritize safety, which means that robots should never collide. We try to achieve that by using a reactive and decentralized method with low computational requirements.

4. Safety-enhanced Avoidance Policy

In order to address the above conflict resolution problem, we propose *SWAP: safety-enhanced avoidance policy*, a decentralized algorithm which prioritizes safety for the robots (no collisions), and is able to drive them reactively toward their goals using only local information. In particular, the robots use the information provided by their range-finder sensors to detect conflicts and resolve them modifying their trajectories, assuming that others will follow the same policy. For that, they build a polar representation of their surroundings and define angular sectors of orientations that would allow them to navigate safely. Then, some rules are defined so that the robots can manoeuvre avoiding each other in a counter-clockwise fashion and stop when completely surrounded. In the following sections, we explain how the conflicts are detected, and how they are solved by means of a set of navigation behaviors, which are activated according to local Finite State Machines running on each robot.

4.1. Conflict detection

Collisions are defined using a circular approximation of the shape of each robot, which we name *Safety Region*. Each robot uses also a polar representation of its surroundings given by the readings of its range-finder sensor. The obstacles within this polar representation are inflated with an *Inflation Region* to determine which are conflictive.

Definition 1. *The Safety Region (SR) of a robot is a circle of radius r_{sr} that is centered on its middle and encompasses its whole contour. Formally, the Safety Region of a robot i is defined as $SR_i = \{d \in \mathbb{R}^2 : ||d - p_i|| \leq r_{sr}\}$. The value of r_{sr} can vary from one robot to another depending on their shape and size. Then, a collision is considered to take place in two cases: (1) when the SR of two or more robots intersect, i.e., $\exists i \neq j : SR_i \cap SR_j \neq \emptyset$; (2) when any other external object enters the SR of a robot. We consider here as external objects static features of the scenario and dynamic obstacles not controlled by our system.*

Definition 2. *The Inflation Region (IR) of a robot is the region created by inflating all the obstacles in its field of view. Let $\mathcal{R} = \{range(k)\}_{k=1,\ldots,n_b}$ be the set of range-based readings of a robot. If the previous readings are inflated a distance r_{ir} towards the robot, the previous set can be transformed into $\mathcal{R}' = \{range'(k)\}_{k=1,\ldots,n_b} = \{range(k) - r_{ir}\}_{k=1,\ldots,n_b}$. Thus, the Inflation Region is the area included between $\mathcal{R}$ and $\mathcal{R}'$. Moreover, a conflict is considered to take place for a robot when its Safety Region intersects with its Inflation Region.*

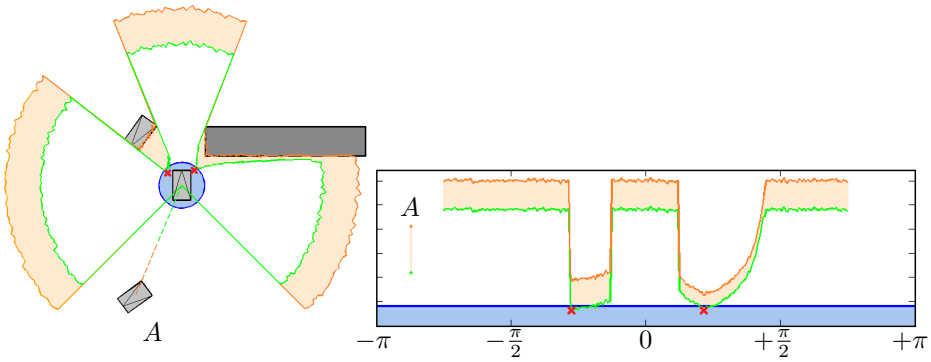

Figure 1: (Left) Regions of a robot surrounded by two other robots and a static obstacle, from a top view. (Right) Corresponding polar graph centered in the robot. In blue, the SR of the central robot; in dark orange, the noisy readings from its range-finder sensor; in light orange, its IR. Robot A is out of the fov, but included thanks to the communication system. The overlaps between the IR and the SR represent the conflictive areas, and the red crosses indicate the direction angle for each conflict, which corresponds to the closer point of each conflictive area with respect to the robot.

Figure 1 depicts an example of a robot with some static and dynamic obstacles around, where the above regions are represented. The obstacles are ignored until they become a conflict, which implies a potential collision that needs to be avoided. For a safe operation, the *Inflation Region* must be such that a robot can detect a conflict and still stop in time before a collision occurs. This idea is illustrated in Figure 2.

First, let d_{br} be the braking distance of the robot, i.e., the maximum distance required to fully stop assuming that it was moving at full speed v_{max}. Then, Figure 2 depicts the worst case scenario of two robots moving against each other at full speed. Once they detect the conflict, they will try to avoid the obstacle (as it will explained later) and, in case they cannot do it, they will still brake not to collide. If we set $r_{ir} = 2d_{br} + e_r + \gamma$, the conflict will be detected in time to stop the collision:

- The term $2d_{br}$ represents the necessary distance so that both robots can stop without colliding in the worst case.
- The term e_r is the maximum possible error of the range sensor, and it is included to consider the fact that the detected obstacles may actually be closer than measured.
- The parameter γ is an offset that can be designed to take into account other potential errors, communication delays, etc. Moreover, if the external shape of the robot could not be measured exactly (e.g., the sensor ranger is mounted on the top of the robot and does not see properly others' contours), this variable may be used to compensate those errors by inflating the readings.

Furthermore, note that if the robots could distinguish between static or dynamic obstacles (e.g., by having a map with the static obstacles or advanced sensing capabilities), the previous r_{ir} may be adjusted more by not considering the d_{br} term twice for static obstacles. When no extra information is available, the system is conservative and all obstacles are assumed to be dynamic.

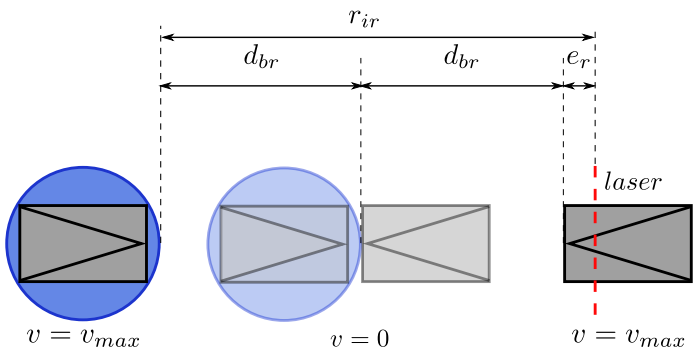

Figure 2: Worst case scenario of two robots travelling at maximum speed against each other. The SR and the required inflation distance are shown for the left robot. It can be seen how both robots will be able to stop in time without colliding after detecting the conflict.

Additionally, robots could share their positions by means of unreliable communication if they are within range. Robots work assuming no communication, but they use it when available. In particular, each robot can use the positions received by its m neighbors $\mathcal{P} = \{p_j\}_{j=1,\dots,m}$ to populate with obstacles areas that are out of their fov (for instance, see Figure 1). For those, $r_{ir} = 2d_{br} + 2e_l + r_{sr} + \gamma$:

- The term $2d_{br}$ represents the necessary distance so that both robots can stop without colliding in the worst case.
- The term $2e_l$ considers the maximum error in the positioning systems. Both robots could be actually closer than they measure.
- The term r_{sr} is included because each robot inflates the received other's position with the SR to detect collisions.
- The parameter γ is an offset that can be designed to take into account other potential errors, communication delays, etc.

Each intersection between the *Safety* and *Inflation Regions* of a robot is treated as a different conflict, and the robot must manoeuvre to resolve all its conflicts. We denote $\mathcal{C} = \{C_k\}_{k=1,\dots,|\mathcal{C}|}$ as the set of conflictive regions of a robot.

Finally, note that we assume that r_{max} is enough to detect conflictive obstacles. Also, the *Safety* and *Inflation Regions* could vary from one robot to another depending on their size and dynamics. We assume here homogeneous robots for simplicity and discuss the heterogeneous case later.

4.2. Computing forbidden directions

Robots analyze locally their conflicts to define which are the headings that will ensure safe navigation without collisions. For each conflict $C_k \in \mathcal{C}$, the robot defines the direction angle of the conflict φ_k as the orientation of the closer point of C_k with respect to the robot [1]. Figure 1 shows an example with the direction angles for different conflicts. For each conflict, the robot defines also a sector of *forbidden* directions $(\varphi_k - \pi/2, \varphi_k + \pi/2)$, which means that navigating towards a direction out of this interval, the robot will be safe. This is depicted geometrically in Figure 3, where a robot computes the *forbidden* directions for its two conflicts. Basically, we can draw a line perpendicular to the direction of the conflict with respect to the robot and navigate in that direction safely, without approaching to the obstacle. At some point, the robot will leave the obstacle aside and will be able to surround it.

After analysing all its conflicts, a robot could have a sector of safely navigable directions (non-colored sector in Figure 3). In that case, an *avoidance angle* is defined in order to resolve the conflicts by surrounding obstacles.

Definition 3. *The avoidance angle φ_a is the orientation of a robot such that: (1) it lies in the boundary of its forbidden sector; and (2) it allows the robot to leave this forbidden sector on its left side.*

By setting the same rule for selecting the *avoidance angles* for all robots, they will resolve conflicts by avoiding each other in a counter-clockwise manner. For instance, in Figure 3 there are two possible robot orientations in the boundary of the *forbidden* sector, but only one (hence selected as φ_a) will allow it to surround an obstacle counter-clockwise, leaving the *forbidden* sector to its left.

4.3. Navigation behaviors

Robots try to reach their goal positions safely by resolving conflicts that they encounter. This is done by following some navigation behaviors encoded into a Finite State Machine (FSM). Each robot runs locally in a decentralized fashion the same FSM, which is in charge of setting different speed and orientation commands depending on the state. Thus, all robots follow the same navigation behaviors, which allows them to solve conflictive situations. Each navigation FSM has the next states:

- `Free`: This is the normal navigation behavior, where the robot can go freely towards its goal. Let φ_g be the goal orientation in the local polar coordinate system of the robot. Then, the robot is `Free` if there are no conflicts or there are conflicts but two conditions are met: (1) φ_g does not belong to the *forbidden* sector; and (2) the goal is not behind the robot, i.e., $\varphi_g \in [-\pi/2, \pi/2]$. In this state the robot navigates at full speed towards its goal, by commanding linear velocity $v_{ref} = v_{max}$ and orientation $\theta_{ref} = \varphi_g$.

[1] In order to avoid noisy measurements when computing the minimum distance of a conflictive obstacle, a low-pass filter is first applied to the readings of the range sensor.

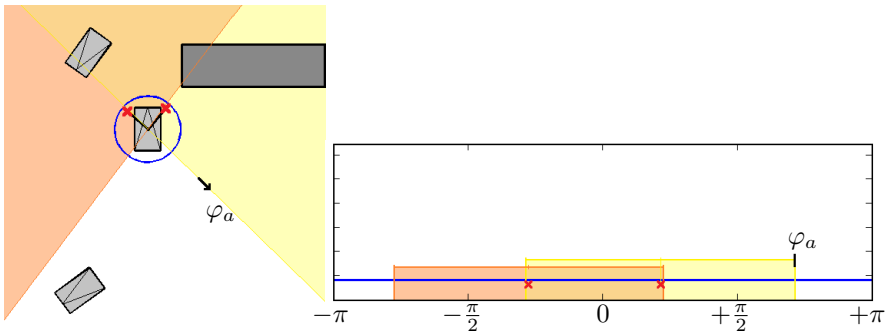

Figure 3: (Left) Forbidden and navigable areas of a robot with two conflicts, from a top view. (Right) Corresponding polar graph centered in the robot. In blue, the boundary of the SR of the central robot. For each conflict, its direction is marked with a red cross and its corresponding *forbidden* sector in a different color. The robot has two possible orientations in the boundary of its *forbidden* sector, but only the one leaving it to its left is selected as the *avoidance angles* φ_a.

- `Blocked`: The robot enters this state when it detects conflicts around and there are no possible orientations out of the *forbidden* sector (there does not exist φ_a). This means that it is `Blocked` and surrounded by obstacles, so the robot stops ($v_{ref} = 0$) and waits until any of the moving obstacles is gone and an escape orientation φ_a appears.

- `Rencontre`: The robot enters this state when it detects conflicts that must and can avoid before navigating toward its goal, i.e., it is not `Free` but not `Blocked` either. In this situation the robot needs to resolve the conflicts escaping through the *avoidance angle*, so it starts stopping to prevent collisions ($v_{ref} = 0$) at the same time that tries to reach the orientation $\theta_{ref} = \varphi_a$. Once it is oriented correctly, the avoidance manoeuvre is safe and the robot can switch to the `Rendezvous` state. Note that the robot may be oriented correctly before stopping completely.

- `Rendezvous`: This state occurs when a robot is performing a manoeuvre to avoid a conflictive obstacle. The robot is oriented to φ_a and can surround the obstacle counter-clockwise. During this operation, a linear velocity $v_{ref} = v_a \leq v_{max}$ is commanded for safety reasons, and a heading $\theta_{ref} = \varphi_a$ maintained. As the robot moves, φ_a varies, which could cause an oscillatory behavior with the robot switching continuously between `Rendezvous` and `Rencontre` to decelerate, reorientate and go forward again. This is prevented with a tolerance error φ_{th} in the robot orientation, i.e., the robot does not leave the `Rendezvous` state as long as its orientation holds $|\theta - \varphi_a| \leq \varphi_{th}$.

In this FSM, which is depicted in Figure 4, transitions are possible among all states. However, there are some typical storylines for the navigation behaviors. For instance, a robot would usually travel toward its goal in `Free` state, and

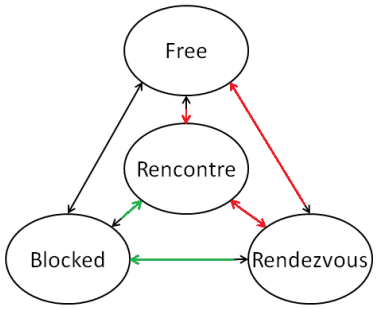

Figure 4: Finite State machine with the navigation behaviors that rule SWAP. In red, the typical transitions while avoiding one or more obstacles. In green, the typical transitions during a blocking situation. Other transitions (in black) are possible but less common.

then it may encounter a conflict. If the conflict is in front of the robot (not ignorable), it will switch to the `Rencontre` state and eventually to `Rendezvous`, surrounding the conflictive obstacle counter-clockwise until it is `Free` again. If two robots encounter each other and perform this operation simultaneously, they will *swap* their positions. If multiple robots converge to the same point, those in the middle become surrounded and `Blocked`, until the external ones release them.

Finally, Algorithm 1 describes all steps to run an iteration of SWAP in a decentralized manner at a given robot. First, the IR is computed from the readings of the range-finder sensor as in Section 4.1 (lines 1-2). If the robot receives information through the communication system, that is incorporated into the IR too (lines 3-8). Then, the *avoidance angle* is calculated as in Section 4.2 (lines 9-13); and the FSM selects the current navigation behavior and the corresponding references in linear velocity and orientation (line 14).

5. Discussion

In this section, we provide further discussion to analyze better the properties of our approach, as well as its limitations.

5.1. Safety

Safety is one of the main objectives in this work. The idea is to ensure multi-robot navigation with no collisions at all. In Section4, it has been explained how SWAP is designed so that robots can always avoid collisions, stopping in time in the worst case scenario. Of course, this comes at a cost, since designing parameters like d_{br}, e_r or e_l may lead to conservative policies, where robots

Algorithm 1 SWAP$(p, \theta, \mathcal{R})$

$r_{ir} \leftarrow 2d_{br} + e_r + \gamma$
IR $\leftarrow$ `inflate`$(\mathcal{R}, r_{ir})$
`sendInfo`(p)
$\mathcal{P} \leftarrow$ `receiveInfo`$()$
if $\mathcal{P} \neq \emptyset$ **then**
  $r_{ir} \leftarrow 2d_{br} + 2e_l + r_{sr} + \gamma$
  IR $\leftarrow$ IR $\cup$ `inflate`$(\mathcal{P}, r_{ir})$
end if
$\mathcal{C} \leftarrow$ IR $\cap$ SR
for all $C_k \in \mathcal{C}$ **do**
  $\varphi_k \leftarrow$ `closestPointAngle`(C_k)
end for
$\varphi_a \leftarrow$ `computeAvoidanceAngle`$(\{\varphi_k\}_{k=1,\dots,|\mathcal{C}|})$
$v_{ref}, \theta_{ref} \leftarrow$ `FSM`$(\varphi_a, \varphi_g, \theta)$
return v_{ref}, θ_{ref}

prioritize safety over performance. Nonetheless, this seems to be adequate in crowded situations, where the algorithm would ensure safe navigation even with sensor uncertainties and no communication at all.

In addition, our robots never move backwards, so they can only provoke conflicts with obstacles in front of them. This entails that robots do not need to detect all possible conflicts around with a full field of view. A field of view of at least 180° would suffice though, which is more common in real robotic systems. For instance, a common situation can be that of two robots driving toward the same direction, one behind the other and faster. In that case, the robot behind will overtake the slower one, which will ignore the conflict at first. The robot behind will enter into `Rencontre`, slowing down as it tries to leave the other at its left. In the middle of the process, the slower robot will detect the other and start to slow down or even stop in order to let it pass.

SWAP would also work for heterogeneous robots with different *Safety Regions* (r_{sr}), braking distances (d_{br}) or even sensor noise (e_r, e_l). Those parameters could be communicated among neighboring robots in order to resolve conflicts more efficiently, or in the worst case, upper bounds could be used. Moreover, nothing precludes the algorithm from using non-circular shapes for the *Safety Regions*, adapting better to different robot shapes.

5.2. U-shaped and non-collaborative obstacles

A single robot following SWAP can always reach its destination as long as there is a conflict-free trajectory in the scenario, regardless of the shapes of the static obstacles. Once the robot encounters an obstacle, it proceeds to surround it counter-clockwise until it finds again free space toward the goal, acting as a *bug* algorithm [30].

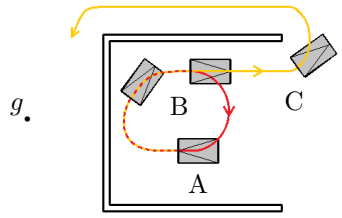

Figure 5: Robot trying to avoid a U-shaped obstacle to reach its goal g. In red, the livelock situation: the robot detects a wall in front of it (A); it starts to surround it counter-clockwise (B); and it gets free eventually (C), turning back to its goal again. In yellow, the path that would be followed by SWAP: the robot keeps surrounding the wall (C), since its goal is behind and it does not switch to `Free`.

Some reactive algorithms (e.g., our previous work [4]) present problems with U-shaped obstacles (convex contours); livelocks where robots never overcome an obstacle completely could happen. An example of this situation is depicted in Figure 5. A robot finds a U-shaped obstacle and starts to surround it counter-clockwise to resolve the conflict. However, at some point it sees no more conflicts in the direction of its goal and turns back to it, ending up into a livelock.

SWAP addresses these situations with the combination of two different mechanisms. First, recall that robots with conflicts only become `Free` again if their goals are not behind them or all conflicts disappear. This requirement of not having the goal behind would avoid livelock situations like the one in Figure 5, forcing the robot to keep surrounding the obstacle instead of turning back when the goal gets out of the *forbidden* sector.

However, imprecisions in the control systems when surrounding an obstacle could still make robots separate too much from it, becoming `Free` (no conflict anymore) before the surrounding is complete and getting into livelocks. Therefore, we include a control mechanism so that robots try to keep a constant distance to the obstacle they are surrounding when they are in `Rencontre` or `Rendezvous`, preventing undesired transitions to `Free` due to momentary separations from the obstacle. Moreover, this mechanism is also useful to avoid collisions with moving obstacles that get closer to the robot while surrounded.

Additionally, there may be *non-collaborative* obstacles in some scenarios, such as other vehicles not following the rules in SWAP. In that case, the non-collaborative vehicles could enclose others preventing them to reach their destinations. SWAP could only ensure *passive* safety in this case, which means that robots would not provoke collisions with others, but they could be hit by the

non-collaborative ones.

A final comment is that, though SWAP can drive robots to their destinations, solutions are not necessarily optimal. For instance, two or more robots could encounter each other within a corridor without place for all of them to pass through. In that case, all robots would turn back to resolve their conflicts by following the walls of the corridors backwards. Even though they would be able to surround the static obstacle eventually and find a path again to their goals, the optimal solution would be that one of the robots passes through the corridor while the others go backward. These and other situations such as avoiding U-shaped obstacles in a *bug* manner can be alleviated by using a higher-level path planner on top of our reactive SWAP. In that configuration, SWAP would act as a local navigation system receiving waypoints from a global planner. Nonetheless, note that, although not optimally, SWAP is still able to drive robots safely toward their goals standalone.

5.3. Liveness

Liveness is a property that guarantees that the robots will reach their destinations in a finite time, not getting stuck into deadlocks/livelocks [3]. In this section, the liveness of SWAP is discussed. Previously, we explained how a single robot is able to reach its goal with SWAP, surrounding any kind of static obstacle without getting into livelocks (as long as there exist a conflict-free path). However, proving that SWAP allows all robots to reach their goals in a generic multi-robot scenario is complex. For instance, if two goals are too close and a robot is occupying one of them, the other could never be occupied in order to prevent collisions. Therefore, liveness will be studied here for a strict set of assumptions, although the discussion could be valid qualitatively for a wider range of situations.

Theorem 1. *Given a set of n robots following SWAP from an initial configuration $\mathcal{I}$ with no conflicts, their goal configuration $\mathcal{G}$ must hold $||g_i - g_j|| \geq d_g = 2r_{ir} + 4r_{sr}$, $\forall i \neq j$ in order to ensure there will be no livelocks/deadlocks.*

Proof. If $\mathcal{G}$ does not fulfill the above property, there exist the possibility of having robots that will never reach their goals. Imagine, for instance, some robots that reached their goals and stay there static. Imagine also that they surround the goal of another robot in the scenario. If the separation between the static robots is lower than d_g, the remaining robot may not be able to navigate through them to reach its goal without being stopped by conflicts (a robot closer than $d_g/2$ would be conflictive). □

The question now is whether the above condition is sufficient or not. This is not true for all cases, but we can draw that the condition suffices for a wide variety of practical setups. In the following, we further discuss this.

Robots that get blocked by others will become free eventually as their neighbors go away toward their goals and release them. However, a deadlock situation in our algorithm appears when n_i robots are `Blocked` (*blocked* set) by other n_e robots (*blocking* set), whose goals are *occupied* by the former. In that case, the

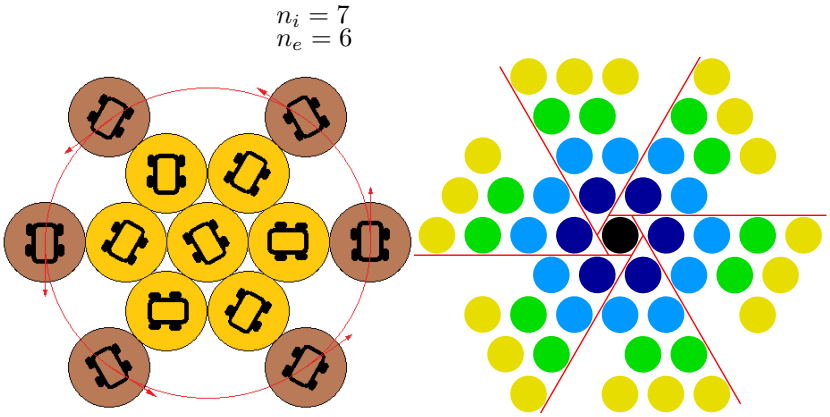

Figure 6: (Left) Blocking situation. Seven robots are blocked (yellow) by six blocking robots (brown). The robots are represented as non-overlapping circles of radius $d_g/2$, provoking conflicts to each other. Following the constraint in Theorem 1, no more than 2 goal points could be occupied by the blocked robots. (Right) Circles forming a centered hexagonal lattice. Each layer l is represented with a different color.

external robots would never be able to reach their goals (livelock), whereas the internal ones would get into a deadlock (see Figure 6, left).

If we have a *blocked* set of robots, they may only be occupying a finite number of goal points n_g according to Theorem 1; and those could *belong* to external robots. We need to check whether $n_g \geq n_e$, and hence there may exist enough external robots to surround the n_i robots and force an infinite blocking situation. Otherwise, the internal robots would end up getting released and navigating to their goals. Therefore, we want to find the minimum number of blocking robots necessary to block n_i robots. The most favorable situation will be when the blocked robots are as packed as possible, since there will be less in the outer layer and it will be easier to surround them. The problem of packing circular forms as densely as possible is a well-known geometrical problem called *circle packing*; and it has been proved [32] that the densest distribution is achieved by forming a hexagonal lattice (see Figure 6, right). In order to be `Blocked`, robots need to be at *conflictive* distance, so they could be represented as non-overlapping circles with an inter-distance of $d_g/2$ (see Figure 6, left).

Given a packed situation like the one in Figure 6, the total number of circles in the centered hexagonal lattice can be computed as $n_i = 3(l)(l-1)+1$, with $l \in \mathcal{N}$; being the ones in the last layer $6(l-1)$. In that case, note that only the blocked robots in the outer layer are relevant to determine the number of required blocking robots. Therefore, we would need $n_e \geq 6(l-1)$. Moreover, if the condition in Theorem 1 holds, and we try to place as many goals as possible within the blocked set, goals would form another 6-connected lattice

with point-to-point distances equal to d_g. In that case, it can be proved that:

$$n_g \leq \begin{cases} h_{\frac{l}{2}} + l - 1 & \text{if } l \text{ is even} \\ h_{\frac{l+1}{2}} & \text{if } l \text{ is odd} \end{cases}, \tag{2}$$

where $h_l = 3(l)(l-1) + 1$. For example, in Figure 6 (left) $n_i = 7$, $n_e = 6$ and $n_g \leq 2$. The 6 external robots would be surrounding the 7 internal forever only if their goal positions were occupied by the blocked robots. Since we can only fit 2 goal points as maximum, the other 4 blocking robots would leave away eventually. Then, 7 robots could not be blocked with only 2 blocking robots.

It is important to remark that Theorem 1 would not be sufficient for very large sets of robots. In particular, for $n_i = 127$, we could place 37 goal points within the blocked area and we would only need 36 external robots to block. However, having so many robots packed at the same place and being surrounded synchronously by so many others is a very unlikely situation for practical setups as the ones proposed in this paper. Moreover, in Section 6, we test our algorithm probabilistically for random configurations, showing liveness in all cases.

5.4. Complexity

SWAP is computationally not expensive and scales well with the number of robots, since this was one of its primary objectives. SWAP runs Algorithm 1 at each robot in a decentralized and local fashion. Algorithm 1 needs to analyze all incoming inputs and process the conflicts around, so it is $O(k)$, where $k = |\mathcal{C}|$ is the number of conflictive obstacles for a given robot. Note that the complexity of the algorithm does not depend on the total number of robots n, but only on the local conflicts, which makes it scalable.

5.5. Robustness and other issues

SWAP is robust against uncertainties in the sensors, second-order dynamics and unreliable communication, providing safe solutions. As we reduce parameters such as d_{br}, e_r or e_l, robots would become less conservative, being able to go through narrower situations; but at certain point, the likelihood of collisions would stop being zero and increase gradually.

Besides, it is important to remark that robots can work asynchronously. Each robot deals with its conflicts locally and regardless of what others are doing, i.e., there are no synchronous avoidance manoeuvres. Furthermore, robots do not need to stop to avoid a collision. When they detect conflicts, they start reducing their speed as they reorientate to their avoidance headings. Full stops will only take place at extreme situations very crowded, where many robots could get blocked; or when the dynamics of the linear velocity are much faster than those of the orientation (i.e., $\tau_\omega >> \tau_v$).

The algorithm also provides some mechanisms to be robust against non-ideal robot dynamics, avoiding unnecessary oscillations. Robots enter in `Rendezvous` when they reach the right orientation to perform the avoidance manoeuvre. However, they cannot follow that orientation perfectly due to their dynamic

constraints. Therefore, PID controllers are used to regulate their headings and the threshold φ_{th} is set to prevent them from switching continuously between `Rencontre` and `Rendezvous`, avoiding repetitive decelerations and accelerations.

Finally, SWAP is designed to work with unicycle robots. Collision avoidance is performed mainly by means of heading manoeuvres, but robots are not always required to stop and turn *in situ*. In contrast, they follow an approach of *stopping while turning*. This gives some room to adapt the algorithm to other kinds of vehicles with non-holonomic constraints, although SWAP was primarily thought to operate more efficiently in packed environments with unicycle robots.

6. Simulations

In this section, we present simulated results that compare our approach with others from the state of the art. There have been some recent work in the literature for benchmarking in robot navigation [5, 6]. Nonetheless, this is still an open and critical issue, mainly for multi-robot conflict resolution. Therefore, we put some effort in this paper to propose benchmarking scenarios and metrics that allow the community to compare their multi-robot collision avoidance approaches. First, we describe these metrics and scenarios for benchmarking. Then, we present experimental results in extensive simulations comparing our algorithm with other representative ones from the state of the art.

6.1. Metrics

There is a high variety of metrics that can be used to compare approaches for multi-robot collision avoidance. Safety, performance and scalability are critical issues, so we have selected a set of metrics that can evaluate those features in a significant manner.

Failure rate. The failure rate (F_r) measures the percentage of robots that fail to reach their goal configuration safely. We consider that a robot has failed when it gets stuck in a deadlock/livelock situation or collides with an obstacle. This metric is relevant to evaluate the safety and liveness of the algorithm.

Collision rate. The collision rate (C_r) complements the F_r and measures the percentage of robots that collide with other obstacles, assessing the safety of the algorithm.

Normalized traveled distance. This distance, denoted as NTD, is the distance that is traveled by a robot to reach its goal from its initial position. The distance is normalized with respect to the shortest collision-free path. This metric is relevant to evaluate the efficiency of the algorithm. The closer to 1, the better.

Normalized traveled time. This metric, denoted as NTT, is analogous to the distance traveled, but measuring time. The normalizing time is that of a robot travelling through the shortest collision-free path at maximum velocity. There are algorithms where robots could travel distances close to optimal, but spend much time stopped instead. Therefore, measuring traveled distance and time together is key to assess the performance of the algorithm. As for the NTD, the closer to 1, the better.

Relative rotational energy. This metric estimates the amount of power that a robot consumes by rotating while travelling to its goal. The total energy consumed can be split into two terms: translational energy E_t and rotational energy E_r; accounting for energy associated with linear and rotational movements, respectively. First, we define $E_t = \int |v| dt$ and $E_r = \int |\omega| dt$; and then, we measure the relative rotational energy as $E_\eta = \frac{E_r}{E_r + E_t}$. This metric is relevant, since some algorithms may be more adequate for certain types of vehicles depending on the amount of rotation they impose.
Cycle execution time. This metric (T_c) measures the time that takes the algorithm to execute an iteration on a robot. The lower this execution time is, the higher the frequency at which the algorithm can be run on the robots. This metric is relevant for the scalability of the system. Of course, the metric is highly dependant on the machine used, but under similar conditions, it gives an idea of the level of computational load for different algorithms.

6.2. Benchmarking scenarios

In order to assess the performance of the algorithms, it is key to select a representative set of scenarios where the robots encounter a good range of potential complex situations for collision avoidance. First, we seek for densely packed situations where second-order dynamics and noisy sensors become more relevant to avoid the collisions. These scenarios are also interesting to evaluate scalability issues. Second, we pursue scenarios with different types of static obstacles (concave shapes, convex shapes, free space). The following scenarios are proposed as benchmarks:
Free random configurations[2]. In this scenario, there is a free space with no obstacles apart from the robots themselves, and random initial and goal configurations are drawn in order to test the algorithms within a variety of situations. If we keep constant the size of the arena as we increase the number of robots, we can evaluate how the system behaves with higher densities of robots. Particularly, we draw configurations uniformly distributed in the space but imposing some constraints: (1) the robots do not start colliding; (2) there is enough separation between goals to let other robots pass; (3) initial and goal points must be separated a minimum distance, so that the robots have to travel significantly along the scenario, avoiding useless configurations with robots starting too close to their final goals.
Circular configurations. In this scenario, there is also a free space with no obstacles apart form the robots themselves. However, the initial configuration is always a circle with all robots at equidistant distance, and each robot must travel to its antipodal position in the circle (see Figure 7). Thus, we provoke a highly conflictive situation at the center of the circle, where all robots will try to pass through. Previous works on velocity obstacles [13, 14] have also proposed this scenario for similar reasons.

[2]For simplicity, in the remainder of the paper, we denote the benchmarks as *Free-space*, *Circular*, *Square* and *H-shaped*

Square configurations. This scenario has also been proposed previously [13] to test RVO. It is a symmetric scenario with four static square obstacles and square initial and goal configurations for the robots. An example can be seen in Figure 7, where each robot needs to reach is antipodal position in the diagonally opposed square. Thus, the system is tested with the existence of static obstacles and narrow corridors.

H-shaped obstacles. Some previous works [13, 4] have reported problems with certain types of convex obstacles. Due to this, we include this scenario with an H-shaped obstacle. The robots are distributed on the symmetry axis at both sides of the H and have to reach the opposite side, so they must overcome an U-shaped obstacle to reach their destination (see Figure 7). We consider that this scenario is relevant to check the ability of the system to cope with convex obstacles where robots could get trapped.

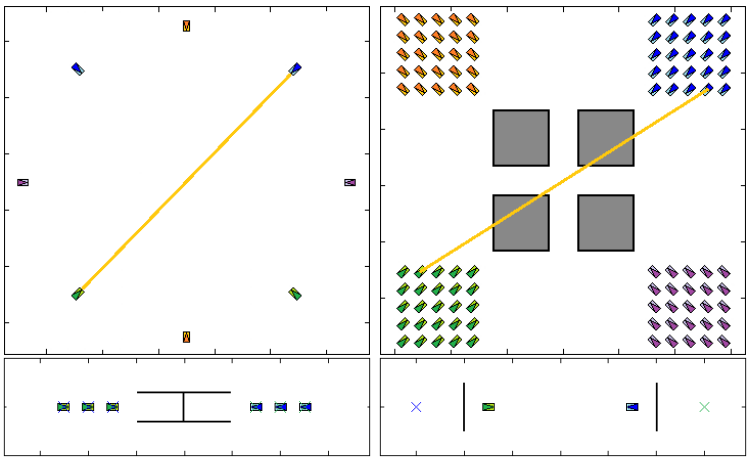

Figure 7: Scenarios for the simulations with examples of initial and goal configurations. Static obstacles are shown in black/grey, robots are placed at their initial positions and their goals are represented as crosses. The yellow lines connect the initial and goal positions of example pairs of robots. (Top, left) *Circular* benchmark. (Top, right) *Square* benchmark. (Bottom, left) *H-shaped* benchmark. (Bottom, right) Scenario for tuning parameters of the algorithms.

6.3. Simulation setup

In order to compare algorithms for multi-robot conflict resolution, we developed a MATLAB simulator implementing all the benchmarking scenarios from the previous section. In the simulator, 2D scenarios with static obstacles modeled as polygons are allowed. The robots are represented by rectangular shapes of $0.5 \times 0.7m$, and follow the dynamic model in Section 3, where the parameters have been set to $\tau_v = 0.5s, \tau_\omega = 0.2s, k_v = 1, k_\omega = 1, v_{max} = 1m/s$ and $\omega_{max} = 1rad/s$. All robots have PID controllers implemented to reach smoothly their commanded orientation and linear velocities.

Robots have a simulated laser range-finder pointing forward with $fov = 270°$, $n_b = 270$ and $r_{max} = 30m$. A Gaussian noise $\mathcal{N}(0, \sigma_r)$ is added to the dis-

tance measurements of the laser. Also, each robot can observe its own position and orientation, but with an additional Gaussian noises $\mathcal{N}(0, \sigma_p)$ and $\mathcal{N}(0, \sigma_\phi)$, respectively. If required by the collision avoidance algorithm (SWAP does not require it), robots can also observe their velocities with another Gaussian noise $\mathcal{N}(0, \sigma_v)$. The values of all these noises vary for each experiment and will be detailed later.

A communication system without delays nor failures is simulated for the robots. With this system, robots can broadcast their noisy positions and velocities (direction and module) depending on the needs of the tested collision avoidance algorithm. Besides, the simulator emulates that the collision avoidance approach is run locally at each robot in a synchronized fashion, at a frequency of 10 Hz. We used a single 32-bit computer with a dual core CPU@3GHz and 4GB of RAM to run all simulations.

We compare our approach for multi-robot conflict resolution with some representative algorithms from the state of the art, including instances from velocity obstacles (ORCA), pure potential fields (VFF), vector field histograms (VFH, VFH+), and the Bubble algorithm. These are decentralized and reactive algorithms without high computational load, since they only use local information and do not plan for future steps [3]. The information shared by the robots differs depending on the case: Bubble, VFF, VFH, VFH+ and SWAP use laser measurements and do not share robot positions (except for SWAP) nor velocities; ORCA does not use the laser sensors though, and robots have to share their positions and velocities. ORCA also requires a previous knowledge of the positions and shapes of the static obstacles, while the others can use the laser sensor instead. The metrics in Section 6.1 were computed for all the algorithms without taking into account the failing robots (except for F_r and C_r) [4].

6.3.1. Parameter tuning

Each of the tested algorithms has different parameters associated. In order to tune the values of those parameters and evaluate the algorithms in similar conditions, we used the tuning scenario in Figure 7 (bottom, right). This scenario consists of two robots starting at opposite positions that have to avoid each other and then a static wall before reaching their destinations. Given the dynamics of the robots, the size of the scenario was prepared so that they could reach full speed before detecting each other and before detecting the walls, making the avoidance more complex. We set the simulator noise parameters to $\sigma_r = 0.05m$, $\sigma_p = 0.05m$, $\sigma_\phi = 0.087rad$, $\sigma_v = 0.05m/s$; and repeated the experiment 1000 times for each algorithm. The parameters of each algorithm were handcrafted to ensure $F_r \leq 1\%$ in the tests[5]. The obtained parameters that were used for each algorithm during our simulations are depicted in Table 1.

[3] VFH* is excluded because it is not purely local.

[4] In our experiments, a robot was considered into a deadlock when it was stopped for a time twice longer than its normalizing time (see Section 6.1); and into a livelock when its $NTD > 15$.

[5] We were only able to achieve a $F_r \leq 7\%$ for the VFF algorithm.

Algorithm	Parameters
SWAP	$r_{sr} = 0.33m$, $e_r = 0.1m$, $e_l = 0.1m$, $d_{br} = 0.48m$, $\gamma = 0.015m$, $\varphi_{th} = \pi/15rad$, $v_a = 0.5m/s$
ORCA [14]	$V^{max} = 1m/s$, $V^{pref} = 1m/s$, $r = 0.33m$, $\tau = 5s$
Bubble [31]	$K_i = 14$, $N = 180$, $\delta_t = 0.1$
VFF [21]	$F_{cr} = 0.1N$, $F_{ct} = -0.012N$, $K_s = 22.47$, $w = 0.1$
VFH [27]	$W_s = 4$, $n = 54$, $\alpha = \pi/36rad$, $a = 2.83$, $b = 1$, $l = 7$, $threshod = 6$
VFH+ [28]	$W_s = 4$, $n = 54$, $\alpha = \pi/36rad$, $a = 2.83$, $b = 1$, $r_r = 0.33m$, $d_s = 0.5$, $\tau_{low} = 10$, $\tau_{high} = 20$

Table 1: Values of the parameters for each algorithm. Those were obtained with a tuning procedure to achieve a safe behavior with each algorithm in a typical scenario with noise and varied obstacles. The meaning of each parameter can be seen in the corresponding original paper.

6.4. Simulations for Free-space scenarios

In these experiments, we used the *Free-space* scenario of size $50 \times 50m$, increasing the size of the team up to 70 robots. For each size, 100 simulations were run, drawing random initial and goal configurations from a uniform distribution. In particular, we imposed a minimum distance between initial positions of $2m$, $5m$ between goal positions (Theorem 1 holds), and $10m$ from the initial to the goal position of each robot. Moreover, there was no noise for the laser and positioning systems, so robots could localize themselves and sense the obstacles accurately.

Figure 8 shows the results of the experiments for the different algorithms. SWAP is the only one able to ensure zero F_r and C_r for high densities of robots, while the others get worse performance as the number of robots increases, as they do not consider second-order dynamics. It can be seen that VFH+ also keeps the robots safe, and its failure rate is due to livelocks. The worst performance is achieved by VFF, due to local minimum on the potential fields that lead the robots to collide.

SWAP, VFH+ and Bubble increase slightly the NTD in denser scenarios while the others do not; showing VFH+ a worse profile than SWAP and Bubble. In terms of NTT, ORCA and VFH keep it almost constant for higher numbers of robots; while SWAP, VFH, VFH+ and Bubble show a linear increase due to the more frequent stops of the robots. Some relatively good results here (e.g., NTD for VFF) can be explained by the fact that the metrics are only computed for the successful robots, which are not many in some cases. Moreover, it can be drawn that VFH and ORCA make the robots drive closer to each other, augmenting the chances of collisions, but reducing their traveled distances.

Regarding E_η, it can be seen that VFH and VFH+ impose clearly more rotational movements for the robots. As expected, this metric is also increasing for SWAP, but for low densities of robots, its value is similar to those from ORCA, Bubble and VFF. T_c is constant for all algorithm except for SWAP and ORCA, where it increases linearly. In these simulations, the communication system is assumed to be perfect, i.e., robots get information from all others

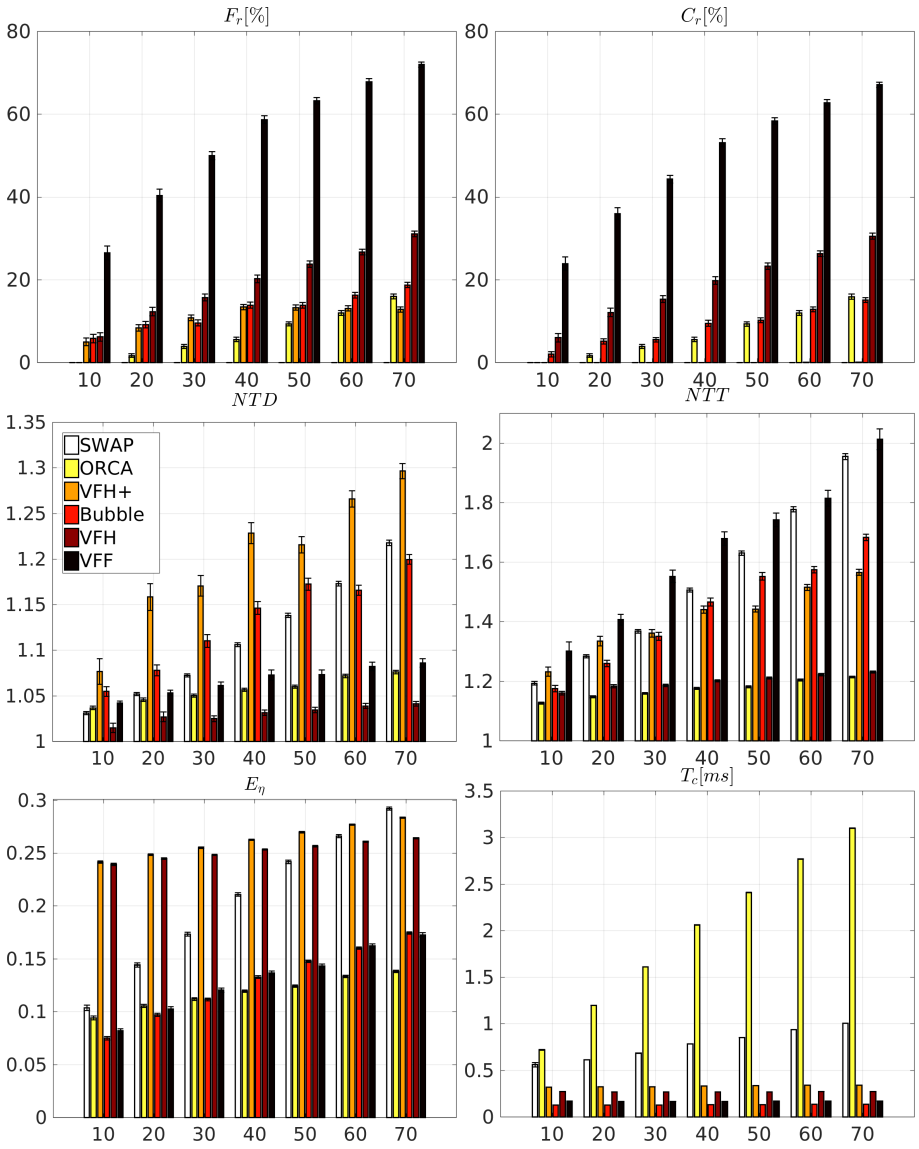

Figure 8: Comparison of the different metrics for the *Free-space* simulations. The horizontal axes show the number of robots; the vertical axes show mean and standard error (whiskers) for 100 runs.

in the scenario. This means that, as the number of robots increases, they need to process more information. Mainly ORCA, where all robots share their positions and velocities. Nonetheless, in real setups, only communication among neighbors would be required.

As a summary, Bubble, VFF and VFH do not work well for multi-robot

scenarios, while VFH+ performs well in terms of safety and could be combined with a higher-level planner to avoid livelocks. In scenarios not very packed, ORCA seems to be a good choice achieving good values of arrival time and distance traveled. Our algorithm SWAP is the only one achieving safety and liveness in packed situations, and at the same time it presents a performance similar to ORCA for low densities of robots.

6.5. Simulations for Free-space scenarios with noise

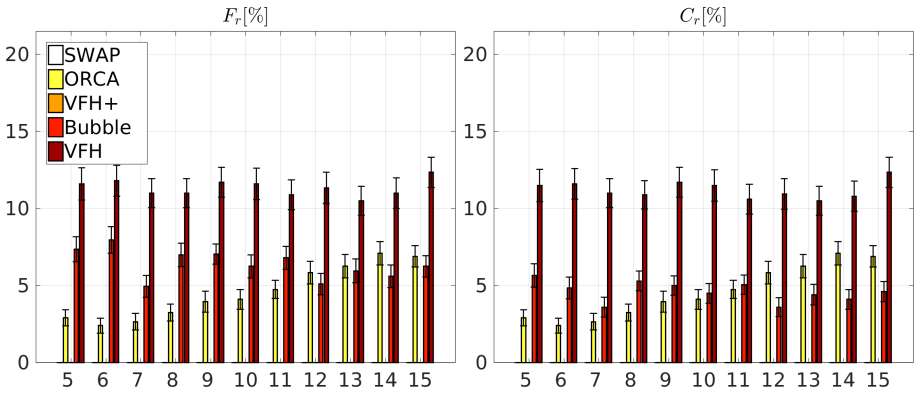

Figure 9: Comparison of F_r and C_r for *Free-space* simulations with noisy sensors. The horizontal axes show the level of noise (σ_b); the vertical axes show mean and standard error (whiskers) for 100 runs.

In these experiments, we tested the robustness of the algorithms under noisy conditions. In particular, we used the *Free-space* scenario with the same initial and goal configurations indicated in the previous section for 20 robots. However, we increased gradually the level of noise from *usual* values in real robots to *high* values. We set a base standard deviation σ_b and computed the noisy variables as $\sigma_p = \sigma_v = \sigma_r = 0.01\sigma_b$ and $\sigma_\phi = [(6\sigma_b/10 - 2)]\pi/180$.

Figure 9 shows the resulting F_r and C_r for the different algorithms[6]. SWAP is able to deal with noisy sensors, keeping a zero failure rate. This happens despite having noise levels higher than those considered by the algorithm (the highest level is $\sigma_r = \sigma_p = 0.15m$, while $e_r = e_l = 0.1m$). VFH+ also behaves without collisions. Indeed, the noise *helps* the algorithm, since it allows robots to escape from the livelock situations. ORCA does not ensure safe trajectories for the robots, but it still performs well for low levels of noise. Recall that ORCA works selecting velocities in the boundaries of the velocity obstacles. This is sensitive to noisy sensors, since observing velocities erroneously may lead robots to collide.

[6]VFF is not included because it was not able to manage noisy situations at all.

6.6. Simulations for Circular scenarios

In these experiments, we used the *Circular* scenario with a radius of $50m$ and no noisy measurements. The scenario was populated with robots increasingly, until reaching the packed situation of 40 robots going toward the center. Table 2 shows the results of the experiments for the different algorithms. Since some metrics are only computed considering the successful robots, the cases without successful robots have the value *None* for the corresponding metrics.

In this symmetric scenario, Bubble, VFH and VFF do not behave well, failing in most cases. SWAP is the only algorithm able to perform all experiments without failures. In between, ORCA and VFH+ show an interesting pattern where the results do not necessarily get worse with more robots (e.g., ORCA is able to achieve a 100% of success for 5 and 15 robots, but it enters on a livelock with 10 robots). We do believe that these algorithms were very sensitive to the initial angles between each robot and its two neighbors in this symmetric circular configuration. Those initial angles depended on the number of robots in the scenario and led to different subdivisions of the space for the robots, and hence to varied results.

Regarding the NTD, SWAP, ORCA and VFH+ present similar results, not increasing the minimum traveled distances significantly. Regarding the NTT, ORCA achieves better results than VFH+ and SWAP, where the robots take longer to resolve the conflictive situation in the center. Also, as expected, ORCA requires less rotational energy than VFH+ and SWAP, since it resolves the conflicts varying robots' velocities instead of with turns. Finally, as it was explained before, the execution time for ORCA grows with the number of robots because we used in these experiments a centralized communication facility, so ORCA needs to process more information as there are more robots in the scenario.

6.7. Simulations for Square scenarios

In these experiments, we used the *Square* scenario, consisting of four squares of $12 \times 12m$ placed symmetrically with a separation of $20m$. The four corridors were $8 \times 12m$ and the robots' initial and goal configurations were squares with an inter-robot distance of $4m$. A single simulation with 100 robots and without noise was run. Table 3 summarizes the results for the different algorithms.

As it can be seen, the scenario is really challenging, due to the combination of static and moving obstacles, a highly packed situation, and the narrow corridors[7]. All algorithms presented high rates of failure except for SWAP, which was able to cope with the scenario safely. VFH+ was also able to keep the robots safe, but they still got into livelocks. It is also noticeable that SWAP presented a high value of NTT. This is because the robots had to follow the walls to overcome the static obstacles and got stuck in the middle long while resolving the crowded situation.

[7] A video of the experiment for SWAP can be seen at `https://project.nes.uni-due.de/videos/ferrera/SquaresSimulationSWAP.avi` or on the attachment "SquaresSimulationSWAP.avi"

Scenario		*Circular* scenario with radius $50m$					
Alg.	robots	$F_r(\%)$	$C_r(\%)$	NTD	NTT	E_η	$T_c(ms)$
SWAP	5	0.00	0.00	1.02 ± 0.000	1.10 ± 0.000	0.07 ± 0.000	0.47 ± 0.014
	10	0.00	0.00	1.04 ± 0.000	1.19 ± 0.000	0.13 ± 0.000	0.49 ± 0.003
	15	0.00	0.00	1.06 ± 0.000	1.27 ± 0.000	0.17 ± 0.000	0.53 ± 0.004
	20	0.00	0.00	1.10 ± 0.002	1.44 ± 0.007	0.25 ± 0.003	0.59 ± 0.002
	25	0.00	0.00	1.09 ± 0.006	1.40 ± 0.018	0.21 ± 0.005	0.64 ± 0.003
	30	0.00	0.00	1.09 ± 0.003	1.46 ± 0.010	0.23 ± 0.003	0.71 ± 0.003
	35	0.00	0.00	1.15 ± 0.004	1.65 ± 0.008	0.31 ± 0.002	0.71 ± 0.002
	40	0.00	0.00	1.09 ± 0.001	1.50 ± 0.009	0.23 ± 0.003	0.81 ± 0.002
ORCA	5	0.00	0.00	1.00 ± 0.000	1.02 ± 0.001	0.01 ± 0.001	0.40 ± 0.024
	10	100.00	0.00	None	None	None	0.90 ± 0.006
	15	0.00	0.00	1.01 ± 0.004	1.10 ± 0.011	0.05 ± 0.007	0.97 ± 0.008
	20	0.00	0.00	1.02 ± 0.003	1.09 ± 0.006	0.05 ± 0.006	1.23 ± 0.009
	25	28.00	24.00	1.03 ± 0.005	1.18 ± 0.021	0.08 ± 0.006	1.86 ± 0.013
	30	26.67	26.67	1.04 ± 0.007	1.16 ± 0.023	0.08 ± 0.006	2.15 ± 0.015
	35	51.43	40.00	1.04 ± 0.006	1.20 ± 0.015	0.08 ± 0.007	2.85 ± 0.018
	40	20.00	20.00	1.03 ± 0.004	1.14 ± 0.014	0.07 ± 0.006	2.99 ± 0.019
VFH+	5	0.00	0.00	1.02 ± 0.001	1.10 ± 0.002	0.22 ± 0.001	0.28 ± 0.005
	10	40.00	0.00	1.03 ± 0.004	1.19 ± 0.006	0.21 ± 0.003	0.32 ± 0.002
	15	66.67	0.00	1.06 ± 0.043	1.26 ± 0.052	0.22 ± 0.007	0.32 ± 0.001
	20	65.00	0.00	1.03 ± 0.009	1.29 ± 0.014	0.23 ± 0.008	0.32 ± 0.001
	25	48.00	0.00	1.15 ± 0.082	1.35 ± 0.082	0.23 ± 0.004	0.32 ± 0.000
	30	50.00	0.00	1.10 ± 0.042	1.33 ± 0.050	0.23 ± 0.004	0.32 ± 0.000
	35	42.86	0.00	1.09 ± 0.020	1.40 ± 0.028	0.25 ± 0.005	0.32 ± 0.000
	40	52.50	12.50	1.10 ± 0.023	1.35 ± 0.031	0.25 ± 0.004	0.32 ± 0.000
Bubble	5	0.00	0.00	1.03 ± 0.000	1.10 ± 0.000	0.05 ± 0.000	0.13 ± 0.020
	10	100.00	100.00	None	None	None	0.11 ± 0.011
	15	100.00	100.00	None	None	None	0.11 ± 0.006
	20	100.00	100.00	None	None	None	0.11 ± 0.002
	25	100.00	100.00	None	None	None	0.11 ± 0.002
	30	100.00	100.00	None	None	None	0.11 ± 0.003
	35	97.14	97.14	1.04 ± 0.000	1.25 ± 0.000	0.08 ± 0.000	0.11 ± 0.001
	40	100.00	100.00	None	None	None	0.11 ± 0.003
VFH	5	100.00	0.00	None	None	None	0.25 ± 0.015
	10	90.00	90.00	1.02 ± 0.000	1.10 ± 0.000	0.22 ± 0.000	0.25 ± 0.002
	15	100.00	100.00	None	None	None	0.23 ± 0.001
	20	100.00	100.00	None	None	None	0.23 ± 0.001
	25	100.00	92.00	None	None	None	0.23 ± 0.001
	30	100.00	86.67	None	None	None	0.23 ± 0.001
	35	100.00	85.71	None	None	None	0.23 ± 0.000
	40	100.00	97.50	None	None	None	0.23 ± 0.000
VFF	5	100.00	80.00	None	None	None	0.16 ± 0.027
	10	100.00	40.00	None	None	None	0.14 ± 0.002
	15	100.00	93.33	None	None	None	0.14 ± 0.002
	20	90.00	90.00	1.02 ± 0.013	1.25 ± 0.015	0.08 ± 0.002	0.14 ± 0.001
	25	92.00	84.00	1.00 ± 0.000	1.33 ± 0.053	0.06 ± 0.006	0.14 ± 0.001
	30	100.00	96.67	None	None	None	0.14 ± 0.001
	35	80.00	80.00	1.03 ± 0.005	1.35 ± 0.062	0.08 ± 0.010	0.14 ± 0.001
	40	95.00	92.50	1.02 ± 0.012	1.41 ± 0.017	0.08 ± 0.003	0.14 ± 0.000

Table 2: Results of the different metrics for the simulations with the *Circular* scenario. Each value represents the mean $\pm$ the standard error among all robots (except for F_r and C_r, which only have a single value per case).

Sce.	Alg.	$F_r(\%)$	$C_r(\%)$	NTD	NTT	E_η	$T_c(ms)$
Square	SWAP	**0.00**	**0.00**	1.57 ± 0.023	3.87 ± 0.100	0.37 ± 0.007	1.26 ± 0.001
	ORCA	91.00	91.00	$\mathbf{1.11 \pm 0.009}$	$\mathbf{1.43 \pm 0.057}$	$\mathbf{0.07 \pm 0.005}$	2.12 ± 0.017
	VFH+	21.00	0.00	1.35 ± 0.019	2.03 ± 0.038	0.34 ± 0.005	0.36 ± 0.002
	Bubble	75.00	73.00	1.41 ± 0.031	2.55 ± 0.090	0.25 ± 0.011	$\mathbf{0.12 \pm 0.002}$
	VFH	83.00	70.00	1.20 ± 0.017	1.47 ± 0.031	0.30 ± 0.008	0.23 ± 0.001
	VFF	100.00	98.00	None	None	None	0.14 ± 0.002
H-shaped	SWAP	**0.00**	**0.00**	1.22 ± 0.082	2.09 ± 0.217	0.27 ± 0.03	0.81 ± 0.016
	ORCA	100.00	0.00	None	None	None	1.98 ± 0.032
	VFH+	66.67	0.00	$\mathbf{1.05 \pm 0.000}$	$\mathbf{1.24 \pm 0.000}$	$\mathbf{0.25 \pm 0.00}$	0.49 ± 0.005
	Bubble	66.67	66.67	1.07 ± 0.000	1.85 ± 0.000	0.28 ± 0.00	$\mathbf{0.14 \pm 0.017}$
	VFH	100.00	100.00	None	None	None	0.24 ± 0.004
	VFF	100.00	100.00	None	None	None	0.17 ± 0.025

Table 3: Results of the different metrics for the simulations with the *Square* and the *H-shaped* scenarios. Each value represents the mean ± the standard error among all robots (except for F_r and C_r, which only have a single value per case).

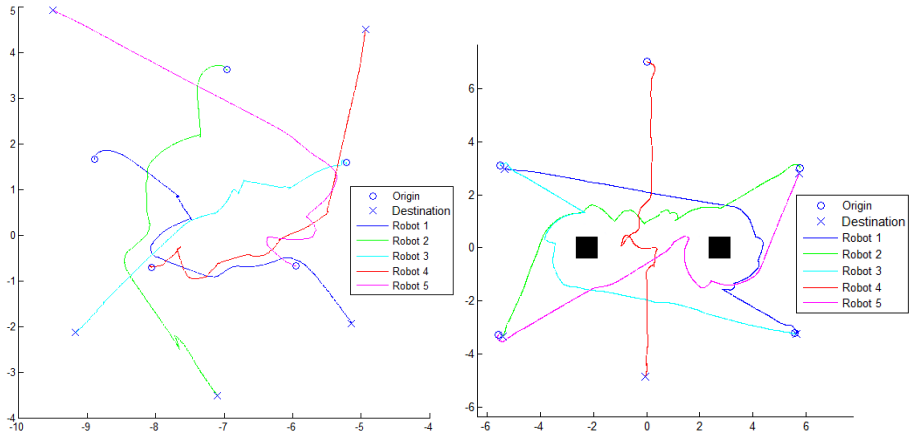

Figure 10: Paths followed by the five Pioneers in Experiment A (left) and Experiment B (right). In Experiment B there were static columns of the testbed in the middle, which are depicted as black squares.

6.8. Simulations for H-shaped scenarios

In this experiment, we used the *H-shaped* scenario with an H of $8m$ of height and $2m$ of width, and three robots at each side with an inter-robot separation of $2m$. A single simulation for each algorithm was run without noise, obtaining the results depicted in Table 3. SWAP, ORCA and VFH+ were able to keep the robots without collisions, but only SWAP was able to drive all robots to their goals overcoming the convex obstacle by surrounding it.

7. Real Experiments

We also tested our conflict resolution approach with real robots[8], showcasing the feasibility of the system under realistic conditions. For that, we performed a number of experiments using an indoor testbed with multiple ground robots. In particular, we used the CONET multi-robot testbed [33], which is installed at the University of Seville (Spain) and covers an area of more than $500m^2$ ($22 \times 24m$). The testbed has five skid-steer robots (Pioneer 3-AT), each of them with a Hokuyo 2D laser.

In our experiments, each robot was able to localize itself by means of an Adaptive Monte-Carlo Localization algorithm, which combined measurements from the Hokuyo laser with a given static map of the scenario. The accuracy of the localization was around $0.20m$. Robots were sharing with others their localization by an IEEE 802.11 wireless bridge, and they were always within communication range for this setup. In order to detect the conflictive obstacles around, each robot was using its Hokuyo laser and the positioning information shared by others. Besides, we run our algorithm for conflict resolution in Section 4 on board each robot at $2Hz$, using the same parameters as in the simulations[9].

A first experiment (Experiment A) consisted of five robots forming a circle in a part of the testbed without static obstacles. The goal of each robot was located at its antipodal position, enforcing a crowded confrontation at the center of the circle. In a second experiment (Experiment B), the robots were placed again forming a circle, but this time there were two static obstacles in the middle. Those were two square columns of $0.8m \times 0.8m$. Figure 10 shows how the robots reached their goals without colliding, even with static obstacles close to the confrontation area[10]. Note that the paths depicted in the figure come from the noisy robot localizations, but the actual paths followed were smoother.

Additional results are summarized in Table 4, which shows that the conflictive situations were solved without increasing traveled distances more than 1.41. NTT is also shown to prove that the time that robots were delayed is bounded and acceptable. Moreover, the average linear speed for Experiment A was a 72.67% of the maximum velocity specified for the robots (v_{max}), whereas for Experiment B it was a 69.2%. This means that robots did not need to decrease their velocities dramatically in order to avoid the confrontation.

8. Conclusions

This paper presents SWAP, an algorithm for multi-robot conflict resolution. The algorithm is decentralized, reactive and focuses on achieving safe behaviors, mainly in packed situations with many robots in confine spaces. With the use

[8] The source code is available at `https://github.com/multirobot-ferr/swap`

[9] We set $v_{max} = 0.15m/s$ to avoid serious damage in case of potential collisions.

[10] A video of the experiment B can be seen at `https://project.nes.uni-due.de/videos/ferrera/RealExperimentSWAP.avi` or on the attachment "RealExperimentSWAP.avi"

	Robot	$p(0)(m)$	$g(m)$	NTD	NTT
Experiment A	1	$(-8.89, 1.68)$	$(-5.03, -2.08)$	**1.41**	1.84
	2	$(-6.96, 3.64)$	$(-7.04, -3.66)$	1.28	1.50
	3	$(-5.22, 1.59)$	$(-9.33, -2.27)$	1.15	**1.98**
	4	$(-8.06, -0.70)$	$(-4.93, 4.67)$	1.33	1.75
	5	$(-5.95, -0.67)$	$(-9.62, 5.00)$	1.28	1.67
Experiment B	1	$(5.58, -3.23)$	$(-5.50, 2.99)$	1.33	**1.97**
	2	$(5.77, 3.02)$	$(-5.46, -3.37)$	1.26	1.88
	3	$(-5.52, 3.09)$	$(5.79, -3.37)$	1.30	1.75
	4	$(0.01, 7.02)$	$(0, -4.95)$	1.23	1.77
	5	$(-5.58, -3.29)$	$(5.74, 2.94)$	**1.35**	1.89

Table 4: Results for real experiments with 5 robots. Initial and final positions are given together with NTD and NTT. Robots do not increase traveled distances significantly (maximum $NTD = 1.41$ only for one case). The NTT values are slightly higher, but still reasonable and in line with the simulated results.

Figure 11: Screenshots of the experiment B, where 5 Pioneer 3-AT running SWAP are commanded to simultaneously cross a room with two static obstacles in the middle. From left to right, the temporal evolution of the experiment.

of local range-finder measurements and low resources, our results show that the method is scalable; and it is able to reach goal configurations safely with noisy sensors, second-order dynamics, different types of static obstacles, and highly conflictive situations.

We propose some benchmarks for multi-robot collision avoidance, including varied scenarios and adequate metrics. Then, we run extensive simulations to compare SWAP with other algorithms from the state of the art, demonstrating empirically liveness and safety for complex situations. We also run our algorithm with real robots in order to show the feasibility of the method.

The paper focuses on unicycle robots and safety is achieved at the cost of increasing rotational movements and the average time to reach the goal. Alternative methods such as ORCA work with velocity profiles instead of orientation changes (improving traveled distance and time), but robots still collide in packed situations and need to sense others' velocities and positions.

Future work will consider Ackermann vehicles. In that case, robots would perform forward and backward manoeuvres within their reserved spaces in order to reorientate in `Rencontre`. Also, we will consider to combine the system with a higher-level planner in order to optimize traveled distances and time foreseeing potential conflictive situations.

Acknowledgements

This work was partially supported by the European Commission H2020-ICT Programme under the project MULTIDRONE (731667).

[1] D. Alejo, J. A. Cobano, G. Heredia, A. Ollero, A reactive method for collision avoidance in industrial environments, Journal of Intelligent & Robotic Systems (2016) 1–14doi:10.1007/s10846-016-0359-7.
URL http://dx.doi.org/10.1007/s10846-016-0359-7

[2] S. A. Reveliotis, E. Roszkowska, Conflict resolution in free-ranging multivehicle systems: A resource allocation paradigm, IEEE Transactions on Robotics 27 (2) (2011) 283–296. doi:10.1109/TRO.2010.2098270.

[3] L. Pallottino, V. Scordio, A. Bicchi, E. Frazzoli, Decentralized cooperative policy for conflict resolution in multivehicle systems, Robotics, IEEE Transactions on 23 (6) (2007) 1170–1183. doi:10.1109/TRO.2007.909810.

[4] E. Ferrera, A. Rodriguez-Castano, J. Capitan, A. Ollero, P. Marron, Decentralized Collision Avoidance for Large Teams of Robots, in: Proceedings of the International Conference on Advanced Robotics, Montevideo, Uruguay, 2013.

[5] F. Amigoni, E. Bastianelli, J. Berghofer, A. Bonarini, G. Fontana, N. Hochgeschwender, L. Iocchi, G. Kraetzschmar, P. Lima, M. Matteucci, P. Miraldo, D. Nardi, V. Schiaffonati, Competitions for benchmarking: Task and functionality scoring complete performance assessment, IEEE Robotics Automation Magazine 22 (3) (2015) 53–61. doi:10.1109/MRA.2015.2448871.

[6] C. Sprunk, J. Röwekämper, G. Parent, L. Spinello, G. D. Tipaldi, W. Burgard, M. Jalobeanu, Experimental Robotics: The 14th International Symposium on Experimental Robotics, Springer International Publishing, 2016, Ch. An Experimental Protocol for Benchmarking Robotic Indoor Navigation, pp. 487–504. doi:10.1007/978-3-319-23778-7_32.
URL http://dx.doi.org/10.1007/978-3-319-23778-7_32

[7] M. Hoy, A. S. Matveev, A. V. Savkin, Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey, Robotica 33 (2015) 463–497. doi:10.1017/S0263574714000289.
URL http://journals.cambridge.org/article_S0263574714000289

[8] P. Abichandani, G. Ford, H. Y. Benson, M. Kam, Mathematical programming for multi-vehicle motion planning problems, in: IEEE International Conference on Robotics and Automation, 2012, pp. 3315–3322. doi:10.1109/ICRA.2012.6225141.

[9] J. van den Berg, J. Snoeyink, M. Lin, D. Manocha, Centralized path planning for multiple robots: Optimal decoupling into sequential plans, in: Robotics: Science and systems, Vol. 2, 2009.

[10] K. E. Bekris, D. K. Grady, M. Moll, L. E. Kavraki, Safe distributed motion coordination for second-order systems with different planning cycles, The International Journal of Robotics Research 31 (2) (2012) 129–150. arXiv:http://ijr.sagepub.com/content/31/2/129.full.pdf+html, doi:10.1177/0278364911430420.
URL http://ijr.sagepub.com/content/31/2/129.abstract

[11] M. Hoy, A. S. Matveev, A. V. Savkin, Collision free cooperative navigation of multiple wheeled robots in unknown cluttered environments, Robotics and Autonomous Systems 60 (10) (2012) 1253 – 1266. doi:http://dx.doi.org/10.1016/j.robot.2012.07.002.

[12] J. Peng, S. Akella, Coordinating multiple robots with kinodynamic constraints along specified paths, The International Journal of Robotics Research 24 (4) (2005) 295–310.

[13] J. Van den Berg, M. Lin, D. Manocha, Reciprocal velocity obstacles for real-time multi-agent navigation, in: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, IEEE, 2008, pp. 1928–1935.

[14] J. Van Den Berg, S. J. Guy, M. Lin, D. Manocha, Reciprocal n-body collision avoidance, in: Robotics Research, Springer, 2011, pp. 3–19.

[15] J. van den Berg, J. Snape, S. J. Guy, D. Manocha, Reciprocal collision avoidance with acceleration-velocity obstacles, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011, pp. 3475–3482. doi:10.1109/ICRA.2011.5980408.

[16] D. Claes, D. Hennes, K. Tuyls, W. Meeussen, Collision avoidance under bounded localization uncertainty, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 1192–1198.

[17] J. Snape, J. van den Berg, S. Guy, D. Manocha, The hybrid reciprocal velocity obstacle, Robotics, IEEE Transactions on 27 (4) (2011) 696–706. doi:10.1109/TRO.2011.2120810.

[18] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, R. Siegwart, Optimal reciprocal collision avoidance for multiple non-holonomic robots, in: Distributed Autonomous Robotic Systems, Vol. 83 of Tracts in Advanced Robotics, Springer, 2013, pp. 203–216. doi:10.1007/978-3-642-32723-0_15.

[19] E. Lalish, K. A. Morgansen, Distributed reactive collision avoidance, Autonomous Robots 32 (3) (2012) 207–226. doi:10.1007/s10514-011-9267-7.
URL http://dx.doi.org/10.1007/s10514-011-9267-7

[20] J. Alonso-Mora, T. Naegeli, R. Siegwart, P. Beardsley, Collision avoidance for aerial vehicles in multi-agent scenarios, Autonomous Robots 39 (1)

(2015) 101–121. doi:10.1007/s10514-015-9429-0.
URL http://dx.doi.org/10.1007/s10514-015-9429-0

[21] J. Borenstein, Y. Koren, Real-time obstacle avoidance for fast mobile robots, Systems, Man and Cybernetics, IEEE Transactions on 19 (5) (1989) 1179–1187.

[22] H. G. Tanner, A. Kumar, Formation stabilization of multiple agents using decentralized navigation functions, in: Robotics: Science and Systems, 2005, pp. 49–56.

[23] D. V. Dimarogonas, S. G. Loizou, K. J. Kyriakopoulos, M. M. Zavlanos, A feedback stabilization and collision avoidance scheme for multiple independent non-point agents, Automatica 42 (2) (2006) 229 – 243. doi:http://dx.doi.org/10.1016/j.automatica.2005.09.019.

[24] H. G. Tanner, A. Boddu, Multiagent navigation functions revisited, IEEE Transactions on Robotics 28 (6) (2012) 1346–1359. doi:10.1109/TRO.2012.2210656.

[25] G. Roussos, K. J. Kyriakopoulos, Decentralized and prioritized navigation and collision avoidance for multiple mobile robots, in: Distributed Autonomous Robotic Systems, Vol. 83 of Tracts in Advanced Robotics, Springer, 2013, pp. 189–202. doi:10.1007/978-3-642-32723-0_14.

[26] P. Ogren, N. E. Leonard, A convergent dynamic window approach to obstacle avoidance, IEEE Transactions on Robotics 21 (2) (2005) 188–195. doi:10.1109/TRO.2004.838008.

[27] J. Borenstein, Y. Koren, Real-time obstacle avoidance for fast mobile robots in cluttered environments, in: Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on, IEEE, 1990, pp. 572–577.

[28] I. Ulrich, J. Borenstein, Vfh+: Reliable obstacle avoidance for fast mobile robots, in: Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on, Vol. 2, IEEE, 1998, pp. 1572–1577.

[29] I. Ulrich, J. Borenstein, VFH*: local obstacle avoidance with look-ahead verification, in: IEEE International Conference on Robotics and Automation, Vol. 3, 2000, pp. 2505–2511. doi:10.1109/ROBOT.2000.846405.

[30] J. Ng, T. Bräunl, Performance comparison of bug navigation algorithms, Journal of Intelligent and Robotic Systems 50 (1) (2007) 73–84.

[31] I. Susnea, A. Filipescu, G. Vasiliu, G. Coman, A. Radaschin, The bubble rebound obstacle avoidance algorithm for mobile robots, in: Control and Automation (ICCA), 2010 8th IEEE International Conference on, IEEE, 2010, pp. 540–545.

[32] H.-C. Chang, L.-C. Wang, A simple proof of Thue theorem on circle packing, http://arxiv.org/abs/1009.4322v1 (2010).

[33] A. Jimenez-Gonzalez, J. R. Martinez-de Dios, A. Ollero, An integrated testbed for cooperative perception with heterogeneous mobile and static sensors, Sensors 11 (12) (2011) 11516–11543. doi:10.3390/s111211516.