Multi-robot Operation System with Conflict Resolution

Eduardo Ferrera[1], Angel R. Castaño[2], Jesus Capitán[1], Pedro J. Marrón[1], and Aníbal Ollero[2]

[1] University of Duisburg-Essen, Bismarckstr. 90, 47057, Duisburg, Germany
[2] University of Seville, Av. de los Descubrimientos S/N, 41092, Seville, Spain

Abstract. Applications with large teams of robots are becoming more and more useful. If the scenario is very crowded or very dynamic, conflict resolution when using a shared workspace is a challenging problem. In this paper, an scalable, decentralized and reactive approach for collision avoidance is presented. The robots can navigate in a 2D environment avoiding each other and without high computational requirements. In addition to the conflict resolution algorithm, a multi-robot simulator is presented. The system is flexible and can be used to simulate different algorithms with realistic robots. Finally, an extension of the simulator is proposed in order to operate real robots in a multi-robot testbed. Results of the collision avoidance approach are shown with both real and simulated robots.

Keywords: Multi-robot coordination, conflict resolution, testbeds.

1 Introduction

The use of multi-robot teams shows a clear trend in the last decade. Advances in swarm robotics make it possible to use large teams of robots in different applications such air-traffic control, multi-UAV missions, logistics, etc. If the scenarios are highly dynamic or the number of robots is variable, the problem of conflict resolution becomes challenging. In particular, this paper focuses on multi-robot systems for conflict resolution in crowded environments. We consider teams that can vary dynamically during operation, adding new robots and removing others.

Although there are centralized algorithms that can tackle collision avoidance with multiple vehicles [4], they are not usually suitable for dynamic applications, since they take many resources when computing the solution (online recomputation becomes infeasible). This paper opts for a decentralized conflict resolution approach in which robots can reach their goals in a reactive manner, sharing as little information as possible and requiring low computational resources. In particular, robots detect potential collisions with local information and solve them assuming that the others will follow the same rules. Each robot only needs to know the position of its neighbouring robots, but not their velocities, orientations or goals. Therefore, the algorithm scales well with the number of robots and adapts when robots are missing.

This paper includes several contributions. First, the paper proposes a decentralized state machine to control the robots and avoid collisions. In particular, the state machine is designed for robots with differential drive which can rotate *in situ*. The approach is decentralized and scalable because each robot runs its own state machine accessing only local information. Second, a simulation framework with Matlab/Simulink has been developed in order to test multi-robot systems in realistic scenarios. The framework is general and flexible and allows us to deal with different kinds of robots and algorithms but, in this paper, it is illustrated to simulate the proposed conflict resolution approach. This tool is quite relevant in order to reduce the gap between theory and real multi-robot systems, since models of real robots and controllers can be used together with the conflict resolution scheme. Third, a framework to operate multiple robots in a real testbed has been developed. This framework allows users to run collision avoidance experiments with a team of Pioneer robots in a testbed scenario.

Regarding related work, many authors solved collision avoidance in the past by means of mathematical programming or control theory [4,1,8,11]. In particular, [1] provides a thorough review of these kind of approaches. Although these methods can obtain optimal solutions, they are complex and in many cases, do not allow for online re-planning. Navigation functions based on potential field can also be used to solve multi-robot collision avoidance [13,6]. If the navigation functions are decentralized, some approaches [12] propose to use *rules of the road* to assign priorities to the vehicles. Others achieve collision-free trajectories by reasoning with velocity profiles [5,2]. In [14], *Optimal Reciprocal Collision Avoidance* is introduced to select in a decentralized fashion velocities that guarantee collision-free movements. The algorithm is extended in [3] to cope with certain kinds of non-holonomic vehicles. Most approaches based on velocities assume that the velocities of the other robots can be shared or sensed. Finally, the work in this paper is similar to that in [10], but it offers advantages against their *Generalized Roundabout Policy* (GRP). Although the GRP can be applied to any kind of vehicles (it was originally developed for airplanes), it is not efficient for vehicles that can stop and turn at zero forward speed, like differential drive robots, helicopters or quad-rotors. Our method takes advantage of this kind of mobility and reduces the area required by the robots to navigate avoiding each other. Also, we present an implementation in a multi-robot testbed of our approach, showing the algorithm with real robots.

The paper is organized as follows: Section 2 presents the conflict resolution approach; Section 3 details the simulation framework with multiple robots; Section 4 explains the multi-robot system in a real testbed; and Section 5 includes conclusions and future work.

2 Collision Avoidance Approach

2.1 Preliminaries

This section presents a decentralized algorithm to avoid collisions among multiple robots working in a shared environment. The main objective is the following: *Given a team of robots in a 2-D workspace, and given their initial positions and their goal destinations, they should be able to navigate to the final configuration avoiding collisions.* Mathematically, there is a set of robots $\mathcal{N} = \{1, \ldots, n(t)\}$ with positions $\{p_i(t) | p_i(t) \in \mathbb{R}^2\}_{i=1,\ldots,n}$ and headings $\{\theta_i(t) | \theta_i(t) \in [0, 2\pi]\}_{i=1,\ldots,n}$. Besides, there is an initial configuration $\mathcal{I} = \{p_1(0), \ldots, p_n(0)\}$, a goal configuration $\mathcal{G} = \{g_1, \ldots, g_n\}$, and a set of static obstacles $\mathcal{S} = \{1, \ldots, m(t)\}$. Each robot has an omnidirectional perception system that allows it to detect obstacles (static or moving) in a circle of radius R_s. Note that the number of robots $n(t)$ can vary throughout time, since new robots could appear or some robots could stop working and become static obstacles. The algorithm makes the following assumptions:

1. Robots are able to localize themselves globally in the scenario, so each robot i has its position $p_i(t)$ available.
2. Robots have differential drive, so they can rotate *in situ* with zero linear velocity.
3. Obstacles around each robot can be sensed and also positioned. The sensing radius is R_s for each robot.
4. Each robot has access to the positions of its neighbouring robots (those within its sensing radius R_s). These positions could be transmitted or sensed with the local perception system.

In order to operate, each robot needs to define two regions around itself: a *Safety Disk* and a *Reserved Disk*.

Definition 1. *The Safety Disk is a circle of radius R_{sd} around each robot that determines the area in which that robot could stop safely (assuming maximum velocity) without hitting a hypothetical obstacle. For obstacles beyond this circle, the robot can assure a safe stop before colliding. R_{sd} can be defined depending on the robot dynamics and size. Therefore, a collision is considered to take place whenever two or more safety disks overlap.*

Definition 2. *The Reserved Disk is a circle of radius R_{rd} around each robot such that $R_{rd} > R_{sd}$, and it is used to detect conflicts. A conflict is considered to take place whenever two or more reserved disks overlap. Moreover, we define the conflictive neighbours of a robot as the subset of neighbours that are in conflict with it.*

It is important to note that R_s must be greater than $2R_{rd}$ in order to be able to detect all possible conflicts in the system. Otherwise, obstacles in conflict may be out of the sensing range. The idea of using a safety disk to define a collision allows the robots to operate safely even under uncertainties in their positions. Moreover, artificial safety and reserved zones can be defined for all the static obstacles, so that moving robots and other obstacles are treated seamlessly.

2.2 State Machine

This section describes the decentralized state machine that defines the behaviour of the robots. Each robot runs the state machine locally and tries to reach its goal reactively, without having information about others' goals, orientations or velocities. In order to detect and solve conflicts, only obstacles in the neighbourhood are considered. In the following, the behaviours for the different modes of the state machine (see Fig. 1) are described.

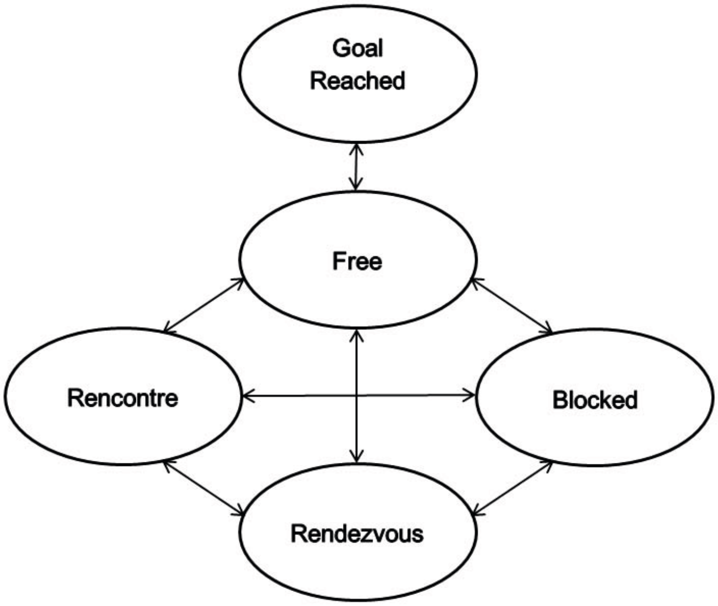
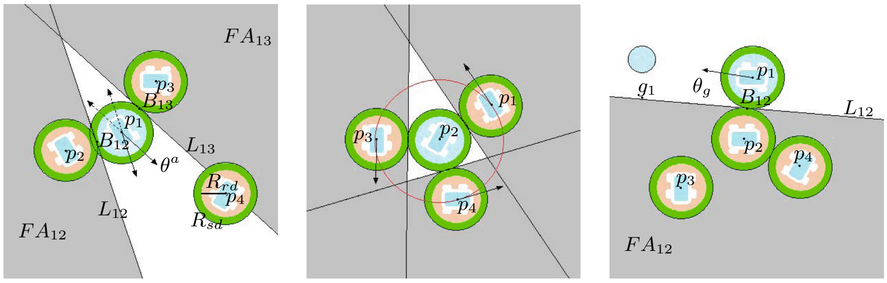


Fig. 1. The robot behaviour is defined by a state machine with five states

`Goal-reached`**.** When the robot gets to its destination point, the state switches to `Goal-reached`. In this state, the robot stops and waits until a new goal is assigned. Moreover, since the positioning systems of the robots have always some error, a distance threshold d_{th} is defined: a robot is considered to be at its goal if its distance to it is lower than d_{th}. Otherwise, robots may not converge to the destination due to minor inaccuracies. This threshold is usually lower than R_{sd} and its value depends on the accuracy of the positioning system of the robot.

`Rencontre`**.** In this state[1], the robot detects one or more conflicts that need to be solved, so it stops immediately its movement in order to prevent possible

[1] *Rencontre* is a word derived from French that means meeting someone unexpectedly.



(a) (b) (c)

Fig. 2. Different conflictive situations. The safety and reserved disks are drawn around the robots. (a) `Rencontre` situation for robot 1, which has conflicts with robots 2 and 3. Its navigable area is shown in white. (b) Robot in the middle is in `Blocked` state. The other robots will surround it counter-clockwise until it is released. (c) Robot 1 is in `Free` state. Despite the conflicts, the robot can still drive towards the goal without collisions with heading θ_g.

collisions. Then, an *avoidance* heading is necessary in order to surround the obstacles. For that, all conflictive obstacles around the robot must be analysed.

For a given robot i and conflictive obstacle j, the space is split into two semi-planes defined by a separator line L_{ij}. If B_{ij} is the closest point to p_i of the intersection of the two reserved disks, L_{ij} is the line that contains B_{ij} and is perpendicular to the vector $\overrightarrow{p_i B_{ij}}$. The semi-plane defined by L_{ij} that contains the obstacle j is the *forbidden* area FA_{ij} for the robot, while the other is the *navigable* area NA_{ij}. After processing all the obstacles causing conflicts, the total forbidden area of the robot i is defined by the union of all the individual forbidden areas $FA_i = \bigcup_{j \in \mathcal{C}_i} FA_{ij}$. The remaining area is the total navigable area, since the robot can move in it without provoking collisions. This partition of the workspace is depicted in Fig. 2a. Note that only positions of the obstacles in the robot local coordinate frame are needed for the computation, but not their velocities nor orientations.

If the navigable area is an open space (infinite area), the robot can find an *avoidance* direction that allows it to surround the current obstacles preventing collisions. In particular, for each line L_{ij} two possible avoidance directions can be proposed, those parallel to the line. The robot only considers avoidance directions that drive it into the navigable area, and selects the one that will allow it to surround the associated obstacle counter-clockwise. For instance, in Fig. 2a each obstacle for robot 1 defines an avoidance direction moving the robot into the navigable area (vectors with solid lines) and one moving it into the forbidden area (vectors with dashed lines). In this case, θ^a is selected as avoidance heading because it also allows robot 1 to surround robot 3 counter-clockwise.

In the `Rencontre` state, the robot performs an *in-situ* turn in order to orientate towards its selected avoidance direction θ^a without moving into the forbidden area. Once it is oriented correctly, the avoidance manoeuvre is safe and the robot switches to the `Rendezvous` state.

`Rendezvous.` If the robot is performing an avoidance manoeuvre with any conflictive obstacle, it is in this state[2]. This means that a conflict has occurred but the navigable area is open, which allows the robot to look for an avoidance heading. Indeed, after the `Rencontre` state, the robot is oriented to an avoidance heading, so it can start the `Rendezvous` operation safely. The robot will move forward keeping the heading θ^a, which will vary in real time as the navigable area does.

`Blocked.` In case that the conflicts produce a navigable area that is a closed polygon (finite area), the robot is `Blocked` and surrounded by obstacles (see Fig. 2b). In this state, the robot stops and waits until any of the moving obstacles is gone. It is important to note that each robot assumes that the others will follow the same rules, so they will move away eventually if there is free space in the scenario.

`Free.` This is the normal operation mode, where the robot can go freely towards its goal. This situation will only happen if there are no conflicts or there are conflicts but the goal is still in the navigable area. Therefore, the robot will be commanded to go to the goal in a straight line without avoiding obstacles (see Fig. 2c). If the robot reaches its goal, it switches to the `Goal-reached` state, but if the goal gets out of the navigable area an obstacle avoidance will be required.

2.3 Discussion

All the static obstacles in the environment can be surrounded by a safety and a reserved zone as it was explained before. Therefore, collisions will always be avoided. Regarding the moving obstacles, in addition to the other robots in the scenario, there may be external agents that are not controlled by the system, such as other vehicles and people. In this case, the algorithm is no longer guaranteed to find the solution. The reason is that those external obstacles do not behave according to the system rules, so they may trap robots and lead them to deadlocks. However, robots stop whenever they detect collisions, so a safe navigation is assured, at least from our system's side.

Since we are considering robots that could fail and stop anywhere, there may be situations in which they occupy the goals of some other robots or block others' paths. To solve these situations, the inclusion of a higher level planner will be considered for future works. Deadlocks could be detected and solved by sending new destinations to the robots involved.

So far, only homogeneous robots with similar reserved and safety disks have been considered. However, the approach would work even in the case of heterogeneous robots. In that case, different robots would have different safety and

[2] *Rendezvous* is a word derived from French that means meeting someone at a specified time and place.

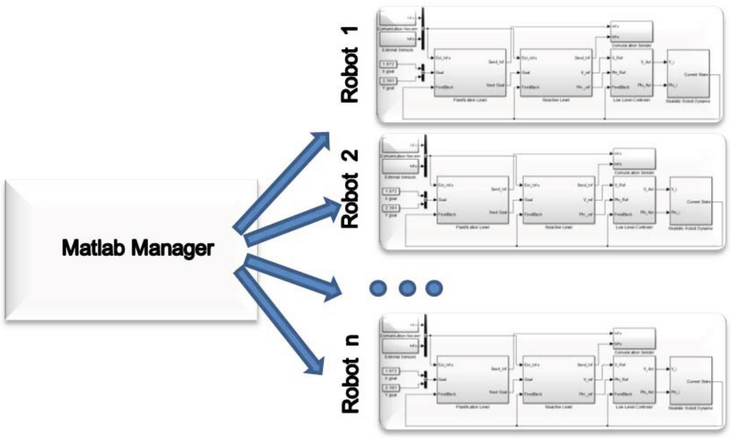


Fig. 3. Block diagram of the multi-robot simulator. A Matlab Manager is in charge of managing the individual Simulink blocks for each robot.

reserved disks, each of them according to their characteristics. Besides, neighbouring robots would also have to communicate that information, since others would need it to discover and solve possible conflicts.

3 Multi-robot Simulator

In order to test the capabilities of the system for different scenarios, a multi-robot simulator has been developed with Matlab/Simulink. The simulator is generic, so the algorithm proposed in this paper as well as alternative approaches can be tested easily. For the design, the following specifications were taken into account:

1. The simulator should consider large teams of robots coexisting in the same workspace.
2. All the simulated robots should have realistic dynamics.
3. The system should be flexible, allowing the user to configure new dynamic models or different control schemes.
4. The simulator should provide a friendly human-machine interface.

In the architecture, a Matlab Manager plays the relevant role of controlling the coordination among individual simulations for each robot (see Fig. 3), which are run in parallel. For a multi-robot experiment, the simulation is divided into time steps of T_{step} seconds and each robot is simulated individually with a Simulink model that is initialized with the last saved state. The simulator also implements a communication system that allows robots to share messages. In particular, a maximum delay of T_{step} in the reception of the messages is considered.

The Simulink model used to model each robot (see Fig. 4) adds some flexibility to the system, since it is prepared to include different dynamics. The only

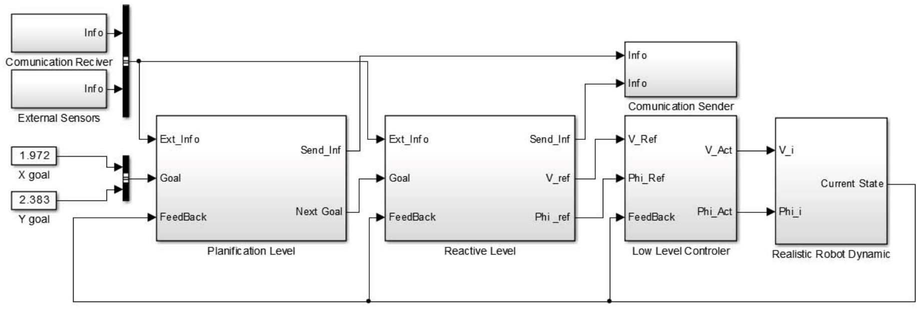


Fig. 4. Simulink block diagram of one of the simulated robots. Dynamics and low-level controllers for the motors are modelled, as well as the reactive state machine.

restriction is that the model should include the possibility of being initialized in a point p with an orientation θ and a speed v at any time. The architecture allows the user to change the whole configuration of the robot, giving to the system enough flexibility to emulate all kind of ground robots and control systems. Moreover, the Matlab Manager is in charge of allocating memory for all the Simulink modules and initializing them. The resulting state for each time step is saved in a logging file so that the experiment can be replayed later.

A friendly user interface was implemented to help the user to design fully controlled scenarios, challenging scenarios or randomly crowded scenarios. The user is also allowed to define a queue of simulations that can be combined and executed sequentially in a predefined or random order. This feature is quite useful for experiments with many trials, such as Monte-Carlo simulations.

In order to illustrate the simulator, an experiment evaluating the performance of the conflict resolution approach presented in this paper is shown. In particular, multiple simulations were run ($T_{step} = 0.1s$) with initial configurations drawn from a uniform distribution in a scenario of size $14 \times 14m$ (only initial configurations without conflicts were considered). The parameters of the algorithm for this experiment were $R_s = 5m$, $R_{sd} = 0.4m$, $R_{rd} = 0.6m$ and $d_{th} = 0.25m$.

The results of the simulations in terms of travelled distance are depicted in Fig. 5, where teams with different number of robots are considered. For each team 200 samples of distances are taken. All distances are normalized with respect to the unconstrained case, in which each robot can travel alone in the workspace straight to its goal. If the scenario is not very crowded, the travelled distances are close to the unconstrained case. When the number of robot increases the distances also do, but a nice degradation can be observed.

Finally, it is important to note that many other simulations were run with the system, testing complex initial configurations and crowded environments. They are not shown in the paper due to space constraints. Indeed, in a stress test for the simulator, it was able to run successfully simulations of up to 100 robots coexisting in a crowded workspace.

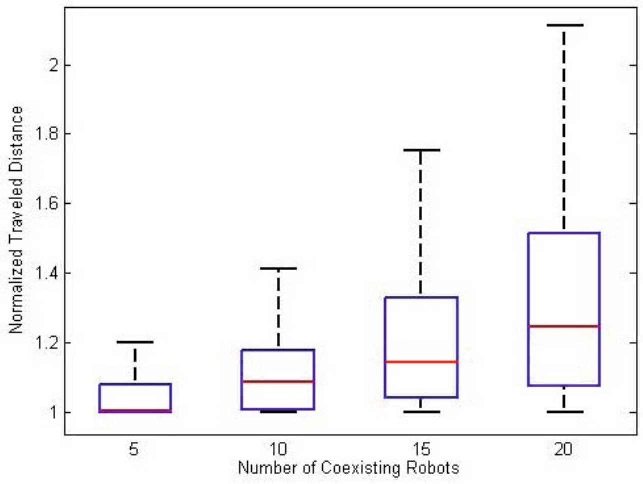


Fig. 5. Normalized travelled distances of simulations in a workspace of $14 \times 14m$, varying the number of robots. Each box plot represents the median (red line), and the first and third quartiles (blue box) for 200 simulations.

4 Multi-robot Testbed

In addition to the simulations, we have performed a number of real experiments using a testbed with multiple robots. For this purpose, an extension to the simulation tools has been developed in order to use the user interface of the simulator with the multi-robot integrated testbed of CONET[3] [9]. The CONET testbed is installed at the University of Seville (Spain) and was designed to test and validate collaborative algorithms with different levels of decentralization, including fully distributed and centralized systems. The testbed also allows users to operate experiments remotely by the Internet.

The testbed is set in a room of more than $500m^2$ ($22{\times}24m$) with three columns in the middle and an IP camera that provides remote users a general view. There are 5 skid-steered holonomic robots (Pioneer 3-AT). The robots are equipped with several sensors including an Hokuyo 2D laser and one Microsoft Kinect RGB-D sensor. A Netbook PC with an Intel Atom processor, 1,024 MB SDRAM and a IEEE 802.11 a/b/g/n Wireless bridge provides enough computational capacity at each robot to perform experiments.

The software architecture of the testbed is based on the open-source middleware Player [7]. The Matlab Manager in Section 3 is extended in order to interact with the testbed. In particular, a VPN direct connection is established with Player in order to command the Pioneers 3-AT. The same user

[3] Cooperating Objects Network of Excellence (INFSO-ICT-224053). Available online: `http://www.cooperating-objects.eu/`

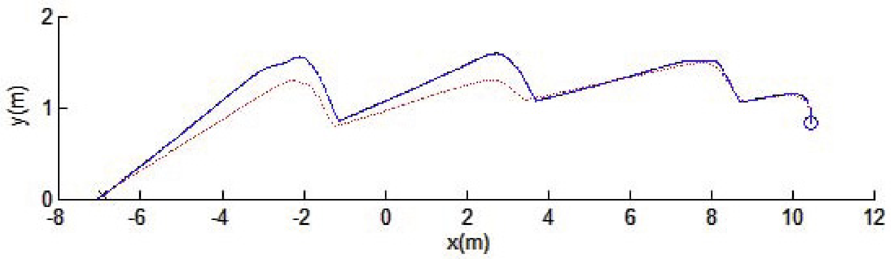


Fig. 6. In blue: Path of a Pioneer 3-AT using the proposed algorithm to avoid three static obstacles. In red: Path of a simulated robot initialized with the same conditions in the simulator. The circle is the initial position and the cross the goal.

interface as before can be used here to define the scenario and the experiments. Then, the Manager is in charge of managing the Player connections with the robots, initialize them and run the experiments autonomously. Moreover, after each experiments data are logged and the robots are sent to their idle positions. It is important to note that this architecture allows the user to perform real experiments in the testbed remotely, without the need for physical presence.

A test of the reactive conflict resolution algorithm presented in this paper was performed with a real robot in the testbed.[4] The experiment consists of a robot that has to cross the room avoiding the three columns in the middle, that are static obstacles. The same experiment was simulated with the system in Section 3 to compare the results, which are shown in Fig. 6. In both cases, the robot reaches the goal avoiding the obstacles, and it can be seen that the behaviour of the simulated robot is quite similar to the real one. The parameters of the algorithm for this experiment were $R_s = 5m$, $R_{sd} = 0.4m$, $R_{rd} = 0.6m$ and $d_{th} = 0.25m$.

A second test using five real robots was also performed. The experiment consists of five robots placed forming a circle in the free part of the room. The goal of each robot is placed at its antipodal position, enforcing a crowded confrontation in the center of the circle. The results of this experiment are shown in Fig. 7. It can be seen how each robot reaches its goal avoiding any collision with the others. This experiment uses the same parameters as above except for R_{rd}, that was increased to $R_{rd} = 0.7m$ in order to make the avoidance more conservative.

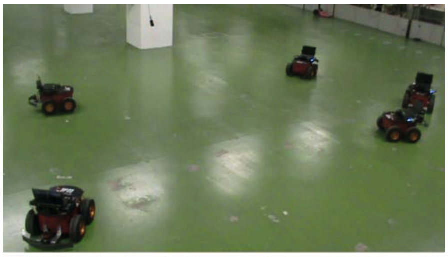
[4] In all the experiments, the robots were localized using a map of the testbed and the Hokuyo readings. Also, robots shared their positions to each other.

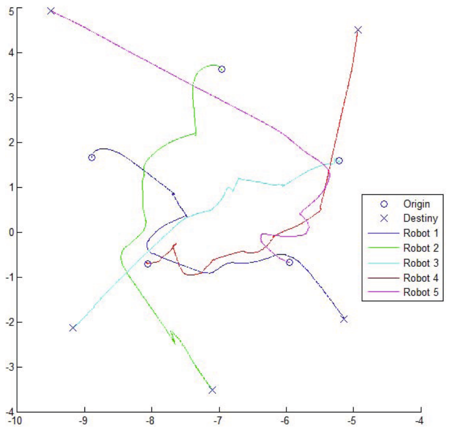
(a) Starting positions $t = 0s$.

(b) Going to the conflictive point $t = 4s$.

(c) Avoiding the confrontation $t = 29s$.

(d) Ending the confrontation $t = 47s$.


(e) Final positions $t = 59s$.



(f) Paths of the robots.

Fig. 7. Five Pioneer 3-AT using the proposed algorithm to solve a conflictive situation

5 Conclusions

This paper has presented a multi-robot system for conflict resolution. A decentralized and reactive algorithm for collision avoidance in crowded scenarios is proposed. The computational requirements for the robots are low, so the approach can be run in small robots in dynamic environments that require online recomputation. Besides, the algorithm is decentralized and scalable with the number of robots in the team. Although the paper focuses on robots with differential drive, future work will extend the method to Ackermann vehicles, including forward and backward manoeuvres.

A multi-robot simulator based on Matlab/Simulink is also presented. The idea is to exploit its flexibility in order to test different kinds of robots and controllers in a realistic manner. To complete the system, the simulator architecture is extended to be able to test algorithms in a real testbed. Experiments with real robots and simulations are shown to illustrate the performance of the collision avoidance method.

Future work will consider the adaptation of the multi-robot system to ROS[5], in order to be able to interact with other testbeds and make the system more portable. Moreover, using distributed computers to run the simulated robots in parallel can be of interest.

Acknowledgements. This work was partially supported by the DECOV project (P10-TIC-6220), the European Center for Ubiquitous Technologies and Smart Cities (UBICITEC) and the FP7 EC-SAFEMOBIL Project [288082].

References

1. Abichandani, P., Ford, G., Benson, H.Y., Kam, M.: Mathematical programming for multi-vehicle motion planning problems. In: IEEE International Conference on Robotics and Automation, pp. 3315–3322 (2012)
2. Alejo, D., Díaz-Báñez, J., Cobano, J., Pérez-Lantero, P., Ollero, A.: The velocity assignment problem for conflict resolution with multiple aerial vehicles sharing airspace. Journal of Intelligent & Robotic Systems 69(1-4), 331–346 (2013)
3. Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., Siegwart, R.: Optimal reciprocal collision avoidance for multiple non-holonomic robots. In: Martinoli, A., Mondada, F., Correll, N., Mermoud, G., Egerstedt, M., Hsieh, M.A., Parker, L.E., Sty, K. (eds.) Distributed Autonomous Robotic Systems. Springer Tracts in Advanced Robotics, vol. 83, pp. 203–216 (2013)
4. van den Berg, J., Snoeyink, J., Lin, M., Manocha, D.: Centralized path planning for multiple robots: Optimal decoupling into sequential plans 2 (2009)
5. Claes, D., Hennes, D., Tuyls, K., Meeussen, W.: Collision avoidance under bounded localization uncertainty. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1192–1198 (2012)
6. Dimarogonas, D.V., Loizou, S.G., Kyriakopoulos, K.J., Zavlanos, M.M.: A feedback stabilization and collision avoidance scheme for multiple independent non-point agents. Automatica 42(2), 229–243 (2006)

[5] `http://www.ros.org`

7. Gerkey, B.P., Vaughan, R.T., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proceedings of the 11th International Conference on Advanced Robotics, pp. 317–323 (2003)
8. Hoy, M., Matveev, A.S., Savkin, A.V.: Collision free cooperative navigation of multiple wheeled robots in unknown cluttered environments. Robotics and Autonomous Systems 60(10), 1253–1266 (2012)
9. Jimenez-Gonzalez, A., Martinez-de Dios, J.R., Ollero, A.: An integrated testbed for cooperative perception with heterogeneous mobile and static sensors. Sensors 11(12), 11516–11543 (2011), `http://www.mdpi.com/1424-8220/11/12/11516`
10. Pallottino, L., Scordio, V., Bicchi, A., Frazzoli, E.: Decentralized cooperative policy for conflict resolution in multivehicle systems. IEEE Transactions on Robotics 23(6), 1170–1183 (2007)
11. Peng, J., Akella, S.: Coordinating multiple robots with kinodynamic constraints along specified paths. The International Journal of Robotics Research 24(4), 295–310 (2005)
12. Roussos, G., Kyriakopoulos, K.J.: Decentralized and prioritized navigation and collision avoidance for multiple mobile robots. In: Martinoli, A., Mondada, F., Correll, N., Mermoud, G., Egerstedt, M., Hsieh, M.A., Parker, L.E., Sty, K. (eds.) Distributed Autonomous Robotic Systems. Springer Tracts in Advanced Robotics, vol. 83, pp. 189–202 (2013)
13. Tanner, H.G., Kumar, A.: Formation stabilization of multiple agents using decentralized navigation functions. In: Robotics: Science and Systems, pp. 49–56 (2005)
14. Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics Research, pp. 3–19. Springer (2011)