2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Hybrid Motion Planner Integrating Global Voronoi Diagrams and Local Velocity Obstacle Method

Nicola Piccinelli[1] and Riccardo Muradore[1]

Abstract— **Global and local path planning are a classic research field in robotics and are key elements for the development of automated warehouse. Velocity Obstacle (VO) and Voronoi Diagrams (VD) are well known methods in autonomous navigation. VO provides for each autonomous vehicle a set of available velocities ensuring collision-free trajectory for multi-robot systems, whereas VD is a geometrical space subdivision used to compute paths furthest away from obstacles. In this paper we will focus on the design of a distributed navigation architecture aiming at integrating to integrate VO, as a local planner and VD as a global planner, to provides efficient collision-free and safe maneuver for each agent belonging to the fleet. The approach is validated in simulation using ROS and Gazebo as a virtual robot development environment.**

I. INTRODUCTION

In mobile autonomous vehicle navigation, path planning is of paramount importance. The planning problem can be addressed from a global or a local point of view and different approaches can be implemented. In general, global path planning can be used to find the optimal path only if the environment is know, static and if only one vehicle is considered. These approaches are computationally expensive and cannot be used in real-time applications. A*, Rapidly exploring Random Tree (RRT) [11] or Voronoi Diagram [17] are some of the most popular algorithms.

Local path planning, as opposed to global planning, can be used both for single and multi-robot systems and when a partial knowledge about the environment is available. In general this knowledge is provided by on-board sensors and is dynamically improved and expanded. These planning methods are computationally more efficient than the global ones and can be implemented in real-time applications enabling re-planning to promptly react to environment changes and/or no a priori known obstacles [2], [5], [9], [18]. A subset of local planners which may be expressed as a direct mapping between sensors and control without memory are called reactive methods [15].

In the literature, there are also approches suitable for both local and global path planning such as the Artificial Potential Field (APF) [7] and more recently, the Model Predictive Control (MPC) [14]. Global path planning, like RRT, and collision avoidance methods, like Velocity Obstacle (VO) [3], are less conservative than MPC when high order and/or nonlinear vehicles (e.g. UAV) are taken into account [8].

This work was partly supported by the project MIUR "Department of Excellence" 2018-2022.

[1]Nicola Piccinelli and Riccardo Muradore are with the Department of Computer Science, Altair Laboratory, University of Verona, Strada le Grazie 15 - 37134 Verona, Italy. nicola.piccinelli@univr.it, riccardo.muradore@univr.it

To choose one possible solution over another, many quality measures or combination of them are used: the most common is the minimum distance. For differential drive wheeled vehicle a minimum wheeled rotation criteria is used, especially when fine movements are required. For MPC based approaches the minimum distance metric is often substituted with the minimum time metric, taking also into account kinematic and dynamic constraints of the robotic system [8].

Such as in hybrid navigation systems like [24] in this paper we will focus on the integration of a Voronoi Diagram based global planning with a local collision avoidance method based on VO to maximize the clearance to the obstacles. The contributions of our research are:

- The seamless integration of a global planner (VD) with a local planner (VO) for generating collision-free trajectories for multi-agent systems.
- A localization and identification agent-obstacle system based on laser scanner and fiducial markers using on-board sensors.
- A policy to modify the maximum speed of the agent according to the scenario (e.g. other agents and/or obstacles along the nominal path).

The paper is organised as follows. In Section II we introduce the system architecture and explain how the global planner and local planner cooperate. In Section III we provide the implementation details and present the experimental results. Finally in Section IV conclusions are drawn together with our future work plan.

II. PLANNER

The proposed method aims to integrate a local collision avoidance system, based on the Velocity Obstacle approach, with a global path planning algorithm, based on the VD for holonomic Unmanned Ground Vehicle (UGV) fleet. The system architecture is shown in Fig. 1 and is subdivided in two main modules. The first module (in yellow) provides a representation of the free and occupied space by encoding the static and known planimetry information of the environment. This information is precomputed during the setup phase and made available to all agents at run time.

The second module (in light blue) runs motion planning and collision avoidance algorithms during the autonomous navigation by exploiting the representation created by the first module and the data gathered on-line by a set of on-board sensors.

A low level controller is in charge to apply the Cartesian velocity commands generated by the navigation system to the robot motors.

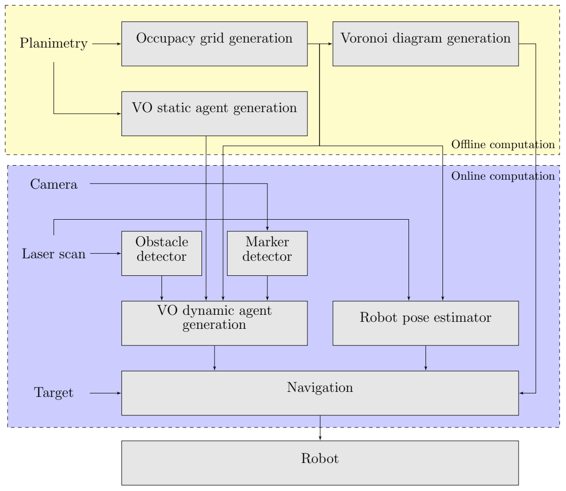


Fig. 1. System architecture overview.

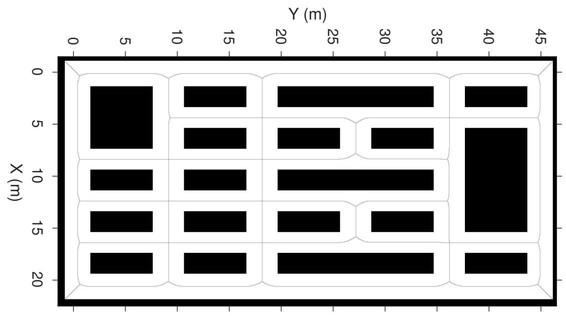


Fig. 2. Offline computation results. The dark boxes represent the occupancy grid while the grey lines are the maximum clearance paths computed by the Voronoi Diagram.

This architecture is similar to the one proposed in [21] even though the algorithm in the modules are different.

A. *Offline computation*

The proposed planning architecture is based on the a priori knowledge of the environment planimetry. We can then generate offline an occupancy grid to represent the static obstacles in the scene. Before proceeding with the computation of the VD on the free space of the map, we have to take into account the robot size (assumed the same for the fleet) because the robot geometry is abstracted as a point.

To do that, we can increase the occupied space around the edges with the robot size and then proceeds as shown in Fig. 2 to compute all the available paths. A further required step consists in representing the obstacles as circles [12] to fill the list of static obstacles used by the VO algorithm taking care of the robot size. Once the offline computation step is concluded, the agents can start the navigation towards their target points.

1) Voronoi Diagram: Given a set S of k points laying on an environment plane E, the Voronoi Diagram is a partition of E into k regions where at each point corresponds a region containing all points closer to that point than to any other in the set [1]. For a generic point $s \in S$ the related plane region is computed as:

$$reg(s) = \bigcap_{q \in S \setminus \{s\}} \{x \in \mathbb{R}^2 | \|x - p\| \leq \|x - q\|\}$$

where $\|\cdot\|$ is the Euclidean distance. Generalized Voronoi Diagram (GVD) are an extension of these diagrams which allows to calculate the regions for primitives like points, lines, polygons, curves or surfaces [22] and can be used for path planning. VD are suitable for autonomous navigation because moving along the edges of a Voronoi Diagram implies that the robot is as far away as possible from the neighboring obstacles [17] and maximizes the clearance.

B. *On-line computation*

The on-line computation is in charge to provide collision-free trajectory, robot localization and motion planning. Collision avoidance and motion planning are based on a Velocity Obstacle extensions called Optimal Reciprocal Collision Avoidance (ORCA) [20]. When the target position is provided and the current robot state is correctly estimated, the path planning along the Voronoi Diagram edges is computed using Dijkstra shortest path algorithm. This is possible because the VD is computed on the occupancy grid, the available paths (i.e. the region edges) are discretized on the cells, and the diagram can be represented as a connectivity graph among the cells containing the region edges. If target and/or current robot position are not part of the diagram edges they are connected with the closest point on the edges and then the shortest path can be computed.

While the original VO works with a single target point to reach, we consider the movement of the robot from the source point to the target point as a motion along temporary waypoints belonging to the VD edges. In this way we maximize the distance of the agent to the obstacles within the static map.

Let P be the set of points obtained by spatially sampling the Voronoi path according to the maximal speed of the robot and $P^\star$ a subset of it.

$$P = \bigcup_{i=1}^{n} p_i, \qquad P^\star = \bigcup_{i=1}^{m} p_i^\star$$

As shown in Fig. 3, the elements of $P^\star$ are obtained downsampling according to the curvature of the path (i.e. the higher the curvature, the larger the number of selected points) and are provided to the VO as temporary waypoints. Each agent will have a different set $P^\star$ related to the specific starting point and final destination.

Given a sequence of n points $p_i \in P$, the sequence of $m \leq n$ target points $p_i^\star \in P^\star$ for the VO algorithm are selected using two criteria:

1) the maximum angle between two consecutive points $(p_i^\star, p_{i+1}^\star)$ has to be smaller than a threshold α,
2) the maximum distance between two consecutive points $(p_i^\star, p_{i+1}^\star)$ has to be smaller than a threshold β.

The thresholds α and β are application-dependent and have to be chosen according to the agents and the environment.

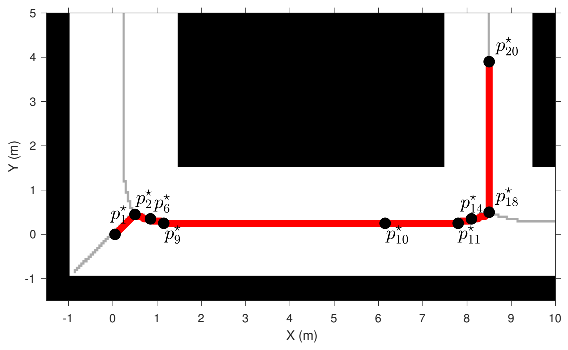


Fig. 3. In red an example of a precomputed Voronoi path for the i-th agent. The dots are a subset of the waypoints in $P^\star$; the waypoint density is proportional to the curvature of the path. Point $p_{10}^\star$ is an example of additional point added to split long straight portion of the path in shorter segments according to the criterion explained in Eq. 1.

The set of intermediate target points for the VO is then

$$\begin{aligned} P^\star \; = \; & \{p_1\} \\ & \cup\{p_i \in P | \angle(p_i - p_{pre(p_i)}, p_{i+1} - p_i) > \alpha, \\ & \quad \forall i = 2, \dots, n-1\} \\ & \cup\{p_i \in P | \| p_i - p_{pre(p_i)} \| > \beta, \forall i = 2, \dots, n-1\} \\ & \cup\{p_n\}. \end{aligned} \tag{1}$$

where the function $pre(p_i)$ gives the previous element index in $P^\star$.

The current waypoint $p_i^\star$ is updated to the next $p_{i+1}^\star, i = 1, \dots, n-1$ according to the Euclidean distance $d(t)$ between $p_{i+1}^\star$ and the robot position $R(t)$ at time t. A distance threshold r_n is introduced to trigger the update of the target point if $d(t) < r_n$. For the last point $p_n^\star$, for which a next point does not exist a second threshold r_e is used to smooth the robot velocity to zero. Choosing $r_n > r_e$ the nominal velocity without obstacles on the planned trajectory will decrease from a maximum speed v_{max} only when the agent is close to the last target point $p_n^\star$. The speed adaptation follows a raised-cosine function for guaranteeing a smooth slowdown to zero. The speed of the agent is:

$$v_n(t) = \begin{cases} v_{max}, & \text{if } d(t) \geq r_e \\ \frac{cos(\pi + (\frac{d(t)}{r_e}\pi)) + 1}{2} v_{max}, & \text{if } d(t) < r_e \end{cases}$$

The value v_{max} is the maximum speed used by the Velocity Obstacle. Unlike ORCA where the maximum robot speed v_{max} is fixed, our approach modifies this value at run time according to the detected agents and obstacles during the navigation. The three maximum speeds v_1, v_2, v_3 with $v_1 > v_2 > v_3$ are chosen according to the following rules:

v_1	is adopted by the i-th agent as maximal speed when no other agents and obstacles are detected.
v_2	is adopted by the i-th agent as maximal speed when the detected objects are agents sharing the Velocity Obstacle policy.
v_3	is adopted by the i-th agent as maximal speed when at least one detected object is an unknown obstacle.

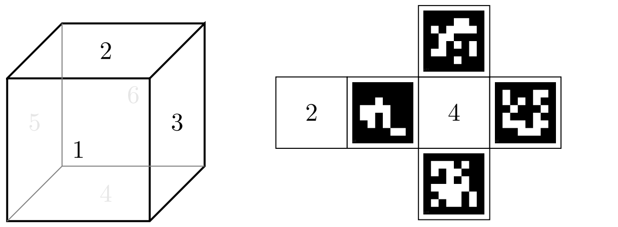


Fig. 4. Four different markers are used to identify from which direction the robot is seen, the marker size is 16.7 cm.

1) Robot position and velocity estimation: VO assumes that each agent knows its and others position and velocity in its own reference frame. Using a laser scanner mounted on the front of the UGV it is possible to estimate the robot position in a known environment using Adaptive Monte Carlo Localisation (AMCL) algorithm [4]. The robot velocity can be estimated directly from the robot odometry or from a sequence of measured robot positions.

2) Obstacles identification and classification: The obstacle identification is a crucial part of the proposed planning architecture and is carried out using the laser scanner. We implemented the approach for detecting and tracking of 2D obstacles presented in [13]. A point cloud from the laser scanner is acquired and a set of segments extracted. Using a heuristic procedure a subset of the segments is selected to represent the object profile and enclosed in bounding circle. These circles represent the detected objects (agents and obstacles) and their position and velocity are estimated by a set of Kalman filters (one for each object).

These objects could also be agents because the profile shape is not sufficient to determinate if an object is a member of the fleet (sharing the VO collision avoidance strategy) or an obstacle. To correctly recognize the agents a second detector strategy has been used as shown in Fig. 1. Each vehicle belonging to the fleet is equipped with a set of calibrated [23] rgb cameras and a set of four fiducial markers (with a specific identification code) applied on top of the UGV. Fig. 4 shows the cube with four ArUco markers on the lateral faces. Using the cameras, a list of detected markers is retrieved and using a static transformation related to the identification codes, a list of detected agents position and velocity is created. It is worth remarking that in case an object lies inside the bounding circle of an agent, such object is removed from the obstacle list because the agent detection takes precedence over the obstacles detector.

Once this check is accomplished, the remaining objects are checked against the static map to remove the obstacles already taken into account. At the end the agents and the obstacles are collected into two lists and it is then possible to generate the dynamic environment used by the local navigation algorithm.

3) Velocity Obstacle and ORCA: VO approach generates collision-free manoeuvres for a multi-robot system in unknown environment without communication among the agents and in our case we use the VO as a local planner for collision avoidance because even if the static environment information is shared among the robots, the motion of other agents and obstacles has to be collected at run time by each agent. Velocity Obstacle methods are based on the following assumptions:

1) Agents and static/dynamic obstacles are modelled as circles (in 2D): this approximation greatly decreases the computational effort.
2) The agents can move in any direction at any time (holonomic robots).
3) Speed and position measurements of other entities (agents and obstacles) are known without uncertainty.
4) All the agents in the shared environment use the same planning algorithm.

In real environments the third assumption is not guaranteed.

Each agent has to estimate the position and speed of every other robot in the fleet using its on-board sensing system. The VO algorithm chooses at run time the actual velocity as close as possible to the nominal velocity over a *Time Horizon* τ.

An extension of VO called Reciprocal Velocity Obstacle is presented in [19] which overcomes the limitation of representing obstacles as set of circles and the oscillation problem present in [3] sharing the responsibility to avoid collisions between agents when they adopt the VO algorithm. If some obstacle is static or adopts a different policy, the responsibility is completely in charge of the involved agent. In our system we integrated a more recent and optimized solution of the VO approach called ORCA [20].

III. SIMULATION RESULTS

The proposed software architecture has been implemented and tested in a simulated environment using the Robot Operating System framework (ROS) and Gazebo simulator [10]. This will allow us to easily implement the proposed approach in a real environment with real robots.

The agents are youBot, a holonomic mobile robot by Kuka which is a common research platform (http://www.youbot-store.com/). The forward and inverse kinematic between wheels and robot base velocities are:

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix} = \frac{1}{r}\begin{bmatrix}1 & -1 & -(l_x+l_y)\\ 1 & 1 & (l_x+l_y)\\ 1 & 1 & -(l_x+l_y)\\ 1 & -1 & (l_x+l_y)\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ \omega_z\end{bmatrix},$$

$$\begin{bmatrix}v_x\\ v_y\\ \omega_z\end{bmatrix} = \frac{r}{4}\begin{bmatrix}1 & 1 & 1 & 1\\ -1 & 1 & 1 & -1\\ -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} & -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y}\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix}$$

where ω_n is the n-th wheel speed, r is the radius (the same for all the wheels), l_x and l_y are the longitudinal and transversal lenght of the robot, and v_x, v_y and ω_z are the linear and angular velocities in the robot body frame, respectively [16].

The robot is equipped with a set of sensors used to identify obstacles, agents and to localise the robot in the environment. The simulated laser scanners are the commercial Hokuyo URG-04LX-UG01 laser range finder with a maximum detection distance of 5 m. Four of them are mounted all around the robot base and they are used for the recognition of agents/obstacles. To detect agents around the robot a set of generic rgb cameras is used to detected ArUco [6] fiducial markers of size 0.167 m. The simulated cameras have a far plane at 5 m and a resolution of 640×480 pixels.

As shown in Fig. 2 the simulated environment is taken from a planimetry of a warehouse with dimension 24x48 m with corridors 2.5 m wide. We test our autonomous navigation system in this environment over three different scenarios:

1) One agent is moving alone in the environment.
2) One agent is moving in the environment and a obstacle is placed along its pre-planned path.
3) Two agents are moving in the environment on overlapping paths.

In our case study the values for the maximum speeds are set to: $v_1 = 0.5$, $v_2 = 0.4$ and $v_3 = 0.3$ m/s due to the limitation of the simulated kinematic model of the youBot where the maximum speed is 0.473 m/s. The robot desired longitudinal and transversal velocities v_x and v_y are computed by ORCA while the angular velocity ω_z is controlled by a local proportional velocity controller so that the front of the robot always points towards the current waypoint.

A. Single agent

Fig. 5(a) shows the first scenario where only one agent is moving in a known environment. All obstacles in the scene are taken into account by the precomputed path. The maximum displacement from the nominal trajectory is 0.14 m and the root mean square error (RMSE) between the Voronoi path and the actual position of the agent is 0.045 m. This is due to the systematic error on localisation and to the robot state estimation.

The maximum speed is always fixed at the highest value v_1 as shown in Fig. 5(b) during the motion, but the planner reduces its velocity to a lower speed in the bends. This is due to the presence of at least a static obstacle over the time horizon of the robot (i.e. the corner of the environment). This is taken into account by ORCA as possible future collision and consequently the speed is reduced to a safer value. This is shown in Fig. 5(b) in positions 1 and 2 when the agent turns right and in positions 13 and 14 when the agent turns left.

B. Single agent and obstacle

In the static environment we added a box of size 0.5 m along the precomputed path visible as shown in Fig. 6(a). When the robot detects the obstacle with the laser scanner it starts moving away from the nominal trajectory enough to avoid the collision with the obstacle. Once the obstacle is overcame and no longer detected by the laser scanner the agent gets back to the nominal path until the target is reached.

In Fig. 6(b) shows how the maximum speed is modified by the planner. At the beginning of the pre-planned trajectory

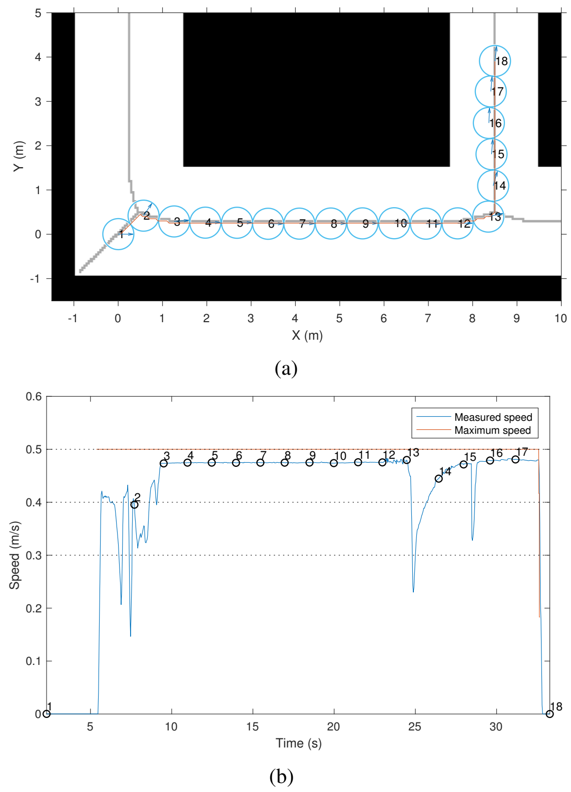


Fig. 5. (a) Cartesian trajectory of an agent moving from starting point (0, 0) to the target point (8.5, 4) m; only static obstacles are present in the environment. (b) Actual robot speed and maximum speed chosen by the planner. The dotted lines are the three maximum speeds (v_1, v_2, v_3).

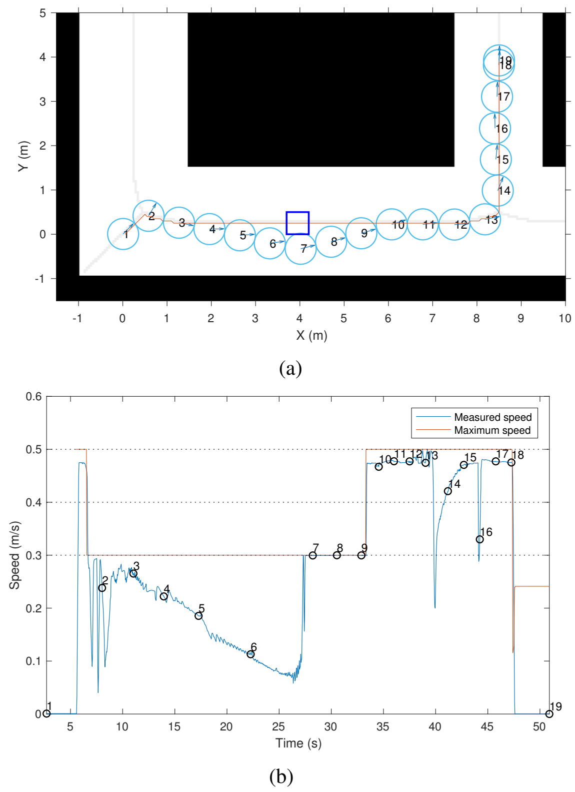


Fig. 6. (a) Cartesian trajectory of an agent moving from starting point (0, 0) to the target point (8.5, 4) m. An unaccounted box (blue square) is placed on the precomputed path. (b) Actual robot speed and the maximum speed chosen by the planner.

(positions 1, 2, 3) the robot is not close enough to detect the obstacle and the planner sets the maximum speed equal to v_1. When the obstacle is detected the maximum speed decreases to the lower value v_3 (positions 4-9). Once the obstacle is avoided and no longer detected, the planner sets the v_{max} again to v_1 (positions 10-19).

C. Multiple agents

In the last scenario we test the classification of other agents. Starting from the static environment derived from the planimetry, two agents are added to the simulation and their target points are chosen in a manner to have the precomputed paths overlapped and so to create a possible collision event during the navigation.

As shown in Fig. 7(b) and Fig. 7(d) the maximum speed is modified from an initial maximum value v_1 to the lower value of v_3 until one of the cameras is able to detect the fiducial markers on the other agent. The marker detection event pushes out the previous detected object from the obstacle list and includes it in the agent list. Then the maximum speed is set to v_2. When the collision is avoided, the agents set their maximum speed again to v_1 and complete the navigation toward their targets.

In Fig. 7(b) it is possible to see at simulation time 10 s some spikes on the maximum speed signal. These are due to the agent detection system that can't recognize correctly the fiducial markers and so the agent is momentarily classified as obstacle. Fig. 7(a) and Fig. 7(c) show the trajectories of the agents (solid blue line) and the position of the other agent at the same time (dashed gray line). In positions 4, 5, 6 the avoidance responsibility should be shared equally between the involved agents because they share the same VO policy. However systematic error makes the agent 2 to move away from the nominal trajectory more than necessary.

IV. CONCLUSION

In this paper we presented an approach for autonomous distributed navigation of a robot fleet which integrates a global path planning (Voronoi Diagram) and a local path planning (collision avoidance algorithm ORCA). Obstacles and agents are detected using on-board sensors (laser scanner and rgb camera). A new policy which shapes the maximum speed and the time horizon of the agent has been integrated to increase the smoothness and the safety of the motion.

We tested the proposed approach in simulation using the ROS framework and Gazebo in three different scenarios: single agent and multiple agents, with and without obstacles. In the future we will test the hybrid local/global planner in a real environment using two youBot robots that we have in the Altair robotic lab at the University of Verona.

References

[1] Franz Aurenhammer. Voronoi diagrams - A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

[2] Johann Borenstein and Yoram Koren. The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[3] P Fiorini and Z Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(Copyright 1998, IEE):760–772, 1998.

[4] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *Proceedings of the national conference on Artificial intelligence*, (Handschin 1970):343–349, 1999.

[5] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997.

[6] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[7] SS Ge and YJ Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, 13(3):207–222, 2002.

[8] Michael Hoy, Alexey S. Matveev, and Andrey V. Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33(03):463–497, 2015.

[9] David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. Randomized Kinodynamic Motion Planning with Moving Obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

[10] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 3:2149–2154, 2004.

[11] S M LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. *In*, 129:98–11, 1998.

[12] Joseph O'rourke and Norman Badler. Decomposition of Three-Dimensional Objects into Spheres. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(3):295–305, 1979.

[13] M. Przybyla. Detection and tracking of 2d geometric obstacles from lrf data. In *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*, pages 135–141, July 2017.

[14] A. Richards and J. P. How. Robust distributed model predictive control. *International Journal of Control*, 80(9):1517–1531, 2007.

[15] Andrey V. Savkin and Michael Hoy. Reactive and the shortest path navigation of a wheeled mobile robot in cluttered environments. *Robotica*, 31(02):323–330, 2013.

[16] Hamid Taheri, Bing Qiao, and Nurallah Ghaeminezhad. Kinematic Model of a Four Mecanum Wheeled Mobile Robot. *International Journal of Computer Applications*, 113(3):975–8887, 2015.

[17] O. Takahashi and R. J. Schilling. Motion planning in a plane using generalized voronoi diagrams. *IEEE Transactions on Robotics and Automation*, 5(2):143–150, Apr 1989.

[18] R.B. Tilove. Local obstacle avoidance for mobile robots based on the method of artificial potentials. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 566–571, 1990.

[19] Jur Den Van Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1928–1935, 2008.

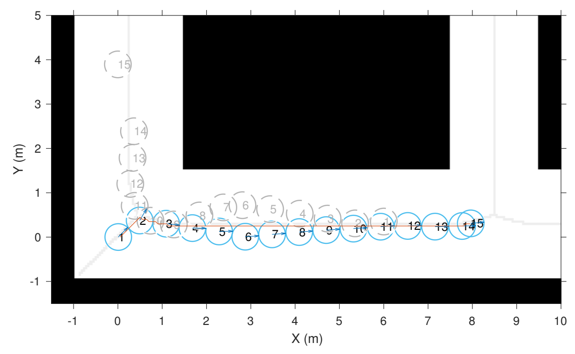
[20] Jur Van Den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Springer Tracts in Advanced Robotics*, volume 70, pages 3–19, 2011.

[21] Jur van den Berg, Sachin Patil, Jason Sewall, Dinesh Manocha, and Ming Lin. Interactive navigation of multiple agents in crowded environments. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, 2008.

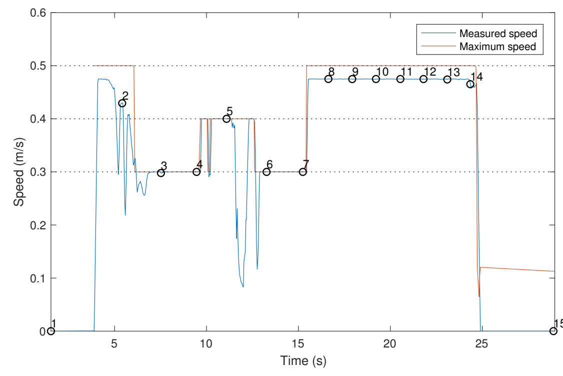
[22] Dave Watson. Spatial tessellations: concepts and applications of voronoi diagrams. *Computers & Geosciences*, 19(8):1209–1210, 1993.

[23] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

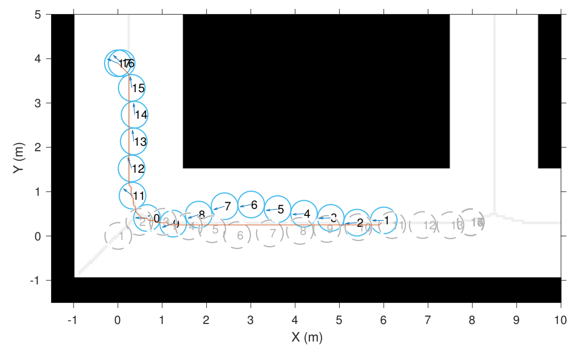
[24] Yi Zhu, Tao Zhang, Jingyan Song, and Xiaqin Li. A hybrid navigation strategy for multiple mobile robots. *Robotics and Computer-Integrated Manufacturing*, 29(4):129–141, 2013.



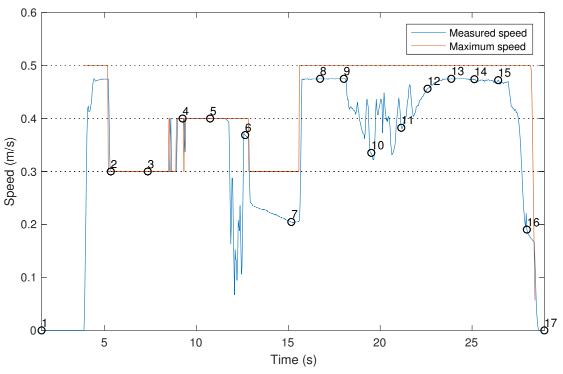
(a) Agent 1



(b) Agent 1 speed



(c) Agent 2



(d) Agent 2 speed

Fig. 7. (a),(c) Cartesian trajectories of the two agents during the autonomous navigation (solid blue circles). The grey circles represent the position of the other agent at the same simulation time. (b),(d) Actual speed of the two agents and the maximum speeds chosen by the planner.