Robust Supervisory-Based Control Strategy for Mobile Robot Navigation

Michele Furci, Roberto Naldi, Andrea Paoli and Lorenzo Marconi

Abstract This work introduces a novel control strategy to allow a class of mobile robots to robustly navigate in a dynamic and potentially cluttered environment. The proposed approach combines a high-level motion planner, designed considering the supervisory control theory, and a low-level stabilizing feedback control law. Taking advantage of a symbolic description of the vehicle dynamics and of the environment, the supervisor reactively selects the current motion primitive to be executed so as to reach the desired target location optimally with respect to a given index cost. Sufficient conditions ensuring boundedness of the tracking error are derived in order to handle the interaction between the discrete-time dynamics of the supervisor and the continuous-time dynamics of the low-level control loop in charge of tracking the desired reference. The resulting approach allows to employ supervisory control tools online without affecting the stability properties of the continuous-time low-level control loop. The results are demonstrated by considering, as application, the kinematic model of an aerial vehicle navigating in a cluttered environment.

Keywords Planning · Supervisory control · Switching systems · Tracking

1 Introduction

The employment of mobile robot systems in real-world applications—ranging from data harvesting, border patrolling or search and rescue operations [14]—requires to design advanced planning and control algorithms [12] in order to accomplish the

M. Furci (✉) · R. Naldi · A. Paoli · L. Marconi
CASY-DEI, University of Bologna, 40124 Bologna, Italy
e-mail: michele.furci@unibo.it

R. Naldi
e-mail: roberto.naldi@unibo.it

A. Paoli
e-mail: andrea.paoli@unibo.it

L. Marconi
e-mail: lorenzo.marconi@unibo.it

E. Menegatti et al. (eds.), *Intelligent Autonomous Systems 13*,
Advances in Intelligent Systems and Computing 302,
DOI 10.1007/978-3-319-08338-4_10

desired mission optimally with respect to a given index cost and robustly in the presence of possible disturbances affecting the vehicle's dynamics. When real-world populated environments and accurate vehicle's dynamics are taken into account, planning the motion of the vehicle optimally may require to solve an optimization problem with a high computational load [12]. Existing approaches often rely on a discrete description of the environment and of the vehicle dynamics so as to reduce the admissible region that optimization tools have to explore. In [8, 9, 11], for instance, feasible paths toward a desired target are obtained by sampling a map of the environment. In [1], the planning task has been described in terms of languages that can be described formally by means of automata. In [5], the complex nonlinear dynamics of the vehicle is decomposed into a finite number of motion primitives. This approach has also been extended in [17] where stability and robustness of the optimal sequence has been taken into account. As far as the environment is concerned, recent approaches have considered the problem of optimizing accurate 3D maps extracted by means of vision algorithms. For instance in [16] and [7], the problem of obtaining a compact representation of a 3D map, based on the idea of employing suitable graphs, has been considered.

This work focuses on the design of a control strategy to robustly steer a mobile robot toward a desired target location while navigating in a populated and dynamic environment. The vehicle dynamics consists of a bidimensional double integrator modeling, the lateral/longitudinal dynamics of a vertical takeoff and landing (VTOL) aerial vehicle, e.g., a quadrotor [15]. Drawing inspiration from maneuver-based motion planning [5], the vehicle is controlled to perform a finite number of elementary movements, i.e., the *primitives*. The current primitive to be executed is selected in real time by a high-level supervisor that is designed as a discrete-event system [2]. To decide the optimal sequence of primitives, the supervisor considers an optimization problem in which it takes into account for the current discrete-event model of the environment and of the vehicle. This fact allows the high-level controller to take decisions as soon as new events occur. With the sequence of maneuvers at hand, a low-level stabilizing controller is proposed whose goal is to maintain the tracking error bounded. The presence of two stabilizing control loops, namely the high-level controller based on supervisory control theory and the low-level continuous-time feedback law, requires to define proper conditions so as to ensure the desired stable behavior of the overall system. This fact is investigated formally by considering stability tools driven by the switching systems theory [13].

The paper is organized as follows. In Sect. 2 the notation used in the paper is defined and some mathematical tools to understand the proposed method are present. The results, in particular, take into account for practical tracking of piecewise continuous references for linear switching systems in which a condition on dwell time of the trajectory pieces is given to ensure a bound on the tracking error. In Sect. 3 the statement of the problem and the solution of the planning problem with the proposed method is presented. In Sect. 4 the control law in terms of low-level control is defined and supervisor and the simulations are presented in Sect. 5. Finally in Sect. 6 the conclusion about the presented work sums up the key points of the paper and explains possible extension or future works on the topic.

2 Notation and Preliminaries

2.1 Notation

The following notation will be used through the whole paper. We define the real positive numbers with $\mathbb{R}^+$. With $t \in \mathbb{R}^+$ we define the time. For a matrix, A^T defines the transpose of the matrix A. I_n is the identity matrix of dimension $n \times n$.

2.2 Tracking of Piecewise Continuous References

Let us consider the following continuous-time single-input time-invariant linear system:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1}$$

where $x = \left[x_1,\, x_2,\, \cdots,\, x_n\right]^T \in \mathbb{R}^n$, $u \in \mathbb{R}$, $A \in \mathbb{R}^{n\times n}$ given by

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

and $B \in \mathbb{R}^{n\times 1}$ defined as

$$B = \left[0,\, \ldots,\, 0,\, 1\right]^T$$

Assume now that the goal of the control design is to track a desired time reference trajectory given by

$$x_R(t) = \left[x_{1R}(t),\, x_{2R}(t),\, \ldots,\, x_{(n+1)R}(t)\right]^T$$

with $x_{iR}(t) = \dot{x}_{(i-1)R}(t),\ \forall i = 2, 3, ..., n+1$. For sake of convenience, we compactly denote as $\overline{x}_R(t)$ the vector which corresponds to the desired trajectory together with the derivatives up to the $(n-1)$-th so as

$$\overline{x}_R(t) = \left[x_{1R}(t),\, x_{2R}(t),\, \ldots,\, x_{nR}(t)\right]^T .$$

The desired trajectory is assumed to be a piecewise continuous function of time with arbitrary switch time and piecewise class C^n on the piece intervals. We define t_j the

time when a trajectory switch occurs, namely when the reference is switching from the trajectory piece *j-th* to trajectory piece $(j+1)-th$, with t_j^- the time just before the j-th reference switch and with t_j^+ the time just after the j-th reference switch, with $j \in \mathbb{N}$. The j-th trajectory piece is continuous and class C^n on the interval $\left[t_{j-1}^+,\, t_j^-\right]$. We suppose that $\left|x_{iR}(t_j^+) - x_{iR}(t_j^-)\right| \leq d_i \;\; \forall i = 1, 2, ..., n, \;\; \forall j$, with $d_i \in \mathbb{R}^+$ (i.e., there are bounds on the difference of trajectory and its derivatives after a switch occurs). Finally, we define $\hat{d} = \sqrt{d_1^2 + d_2^2 + \cdots + d_n^2}$. One can now write the error dynamics of the system by defining $e(t) \triangleq x(t) - \overline{x}_R(t)$, and accordingly

$$\begin{aligned} e_1(t) &\triangleq x_1(t) - x_{1R}(t) \\ e_2(t) &\triangleq x_2(t) - x_{2R}(t) \\ &\cdots \\ e_n(t) &\triangleq x_n(t) - x_{nR}(t) \end{aligned}$$

obtaining, from (1),

$$\dot{e}(t) = Ae(t) + B(u(t) - x_{(n+1)R}(t)) \tag{2}$$

The error system results in a time-dependent switching system due to the presence of the piecewise continuous reference $x_R(t)$. In fact, at every switch in the reference signal it corresponds a jump in the state e of the error system.

The goal of the control law is to track the desired reference trajectory and to achieve

$$\|e(t)\| \leq q, \quad \forall t \geq 0 \tag{3}$$

with $q > \hat{d}$, namely to maintain the tracking error bounded despite jumps in the reference signal.

Inspired by [13], we introduce a dwell time τ, i.e., a minimum time between two switches in the reference trajectory $x_R(t)$, so as (3) can be achieved. More formally, the following result can be stated.

Lemma 1 *With the control input $u(t) = B^T Pe(t) + x_{(n+1)R}(t)$, with P solution of the Riccati equation $A^T P - 2PBB^T P + PA = -aI_n$, and with a Dwell time $\tau \geq -\frac{\bar{\lambda}_P}{a} log\left(\frac{q^2-\hat{d}^2}{q^2}\right)$, with $\bar{\lambda}_P$ the largest eigenvalue of the matrix P, the overall error system (2) is stable in the sense of Lyapunov (see [13]) and $\|e(t)\| \leq q, \;\; \forall t \geq 0$.*

Proof Choosing as common Lyapunov function, the quadratic function $V = e^T Pe$, the control law $u(t) = B^T Pe(t) + x_{(n+1)R}(t)$ with P solution to the Riccati equation given in the statement of the lemma makes the matrix $(A + BK)$ Hurwitz and ensures that

$$\dot{V}(t) = -a \|e\|^2 (t) \tag{4}$$

By applying the *comparison lemma* [10], it also holds

$$\underline{\lambda}_P \|e(t)\|^2 \leq V(t) \leq \bar{\lambda}_P \|e(t)\|^2 \tag{5}$$

From (4) and (5) we have

$$-a\frac{V(t)}{\underline{\lambda}_P} \leq \dot{V}(t) \leq -a\frac{V(t)}{\bar{\lambda}_P} \tag{6}$$

The time between two consecutive switches, namely the dwell time, is given by $\tau = t_j^- - t_{j-1}^+$. We have that, because of the switch

$$\left\|e(t_j^+)\right\|^2 \leq \left\|e(t_j^-)\right\|^2 + \hat{d}^2 \tag{7}$$

and then to achieve (3), a necessary condition is given by

$$\left\|e(t_j^+)\right\|^2 \leq q^2 \tag{8}$$

which, by considering the Lyapunov function (4) and the inequalities in (5), (7), and (8) could be rewritten as

$$V(t_j^-) \leq \bar{\lambda}_P(q^2 - \hat{d}^2) \tag{9}$$

From (9), it is possible to write the evolution of the Lyapunov function based on the bounds (6) on its derivatives

$$V(t_j^-) = V(t_{j-1}^+)e^{-\frac{a}{\bar{\lambda}_P}\tau} \leq \bar{\lambda}_P(q^2 - d^2) \tag{10}$$

Then, by considering the worst case of $V(t_{j-1}^+) = \bar{\lambda}_P q^2$, note that the inequality (10) holds true when the dwell time is chosen as in the statement of the lemma. □

3 Problem Statement

Let us consider a mobile robot modeled as a fully actuated double integrator system defined over a two-dimensional Euclidean space

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = u_1 \\ \dot{y}_1 = y_2 \\ \dot{y}_2 = u_2 \end{cases} \tag{11}$$

in which x_1 represents the longitudinal position in a 2D plane and y_1 the lateral position. The above model can be employed to model, for instance, the kinematics of a holonomic vehicle including the lateral/longitudinal dynamics of VTOL aerial vehicle [4].

Goal of the control law to be designed can be then summarized as follows:

- **Trajectory Tracking**: to allow the vehicle to track a desired trajectory $x_R(t)$, $y_R(t)$ maintaining a bounded tracking error despite the presence of jumps or asymptotically track the desired reference when no jump occurs;
- **Trajectory Planning**: to build the (sub)optimal trajectory for the robot to reach a given target in a map (defined over a two-dimensional Euclidean space) while respecting constrains modeling the environment, e.g., obstacles, and the robot characteristics, e.g., kinematic constraints.

3.1 Trajectory Tracking

The trajectory tracking problem for (11) can be defined as follows. Given the system (11), a trajectory $x_{1R}(t)$, $y_{1R}(t)$, and its derivatives up to the second order, build a control law $u_s = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T$ such that the error system $\dot{e}_s(t) = A_s e_s(t) + B_s u_s(t) + \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^T \ddot{x}_{1R}(t) + \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T \ddot{y}_{1R}(t)$ is asymptotically stable in the special case no jump in the reference occurs. The error system is derived from (11) by defining:

$$e_s(t) = \begin{bmatrix} e_{s1}(t) \\ e_{s2}(t) \\ e_{s3}(t) \\ e_{s4}(t) \end{bmatrix} \triangleq \begin{bmatrix} x_1(t) - x_{1R}(t) \\ x_2(t) - \dot{x}_{1R}(t) \\ y_1(t) - y_{1R}(t) \\ y_2(t) - \dot{y}_{2R}(t) \end{bmatrix} \tag{12}$$

so as

$$A_s = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad B_s = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

The goal could be achieved by a control law

$$u_s(t) = K_s e_s(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \ddot{x}_{1R}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ddot{y}_{1R}(t) \tag{13}$$

with $K_s = B_s^T P_s$, with P_s solution of the Riccati equation $A_s^T P_s - 2P_s B_s B_s^T P_s + P_s A_s = -aI_4$.

Remark: The above result follows as in the proof of Lemma 1 by considering system (11) as two disjoint single-input systems, namely the lateral and the longitudinal dynamics. □

3.2 Trajectory Planning: Supervisor

The high-level control problem of trajectory planning is managed by the supervisor who is in charge of generating the trajectory for the low-level control. Goal of the supervisor is to build online the optimal trajectory $x_R(t)$ and $y_R(t)$ to reach a desired goal in a map, respecting the constrain of the environment (for instance avoiding obstacles). The constrains of the map and the target could change dynamically, i.e., they could change while the robot is already performing a previously defined trajectory tracking task. Following [6], the supervisor is composed of a discrete-event system (DES) supervisor, a graph optimization tool, and a trajectory generator. The purpose of the DES supervisor is to evaluate the feasibility (or sub-feasibility) of a given mission in terms of reaching a target, respecting the constrains of the environment, and return the optimal trajectory in terms of primitives [6]. The trajectory generator task is to form, from the primitives string, a trajectory in terms of $x_R(t)$, and $y_R(t)$. A graph optimization tool is then used on supervisor graph to obtain the optimal trajectory. The trajectory is given in terms of sequence of primitives (elementary or complex maneuvres of the robot) which are then converted in a trajectory of time in terms of $x_R(t)$, $y_R(t)$ and its derivatives. The conversion could be a simple concatenation of the primitives, or it could include a smoother to improve performance of low-level control at a cost of a more complex supervisor.
The stability of the interconnection between the low-level control and the supervisor is guaranteed by Lemma 1; as far as the condition of the minimum dwell time is respected, the overall system is stable in the sense of Lemma 1 and Eq. (3) ensures a bound on the error tracking. The reference generated by the supervisor could be seen indeed as a sequence of maneuvres (primitives) which shapes a piecewise continuous trajectory as the setting in Sect. 2.2.

4 Control Law

We define the control law for the system (11) considering a 2D limited environment. The control law is defined with the low-level control and the supervisor.

4.1 High-Level Control: Supervisor

The supervisor is built from the map automaton, the specification automaton, and the agent automaton.

The map automaton is built from the discretization of the 2D map. Every state of the automaton represents a quantum of the space of a given dimension and the events represent ideal movements in this space. We suppose as example that the environment space is discretized into 7×7 squares of 2m^2 each. On top of that,

we add to states $3-6$ and $4-6$ an uncontrollable event *wind_up* which could represent a zone in the map in which a big upward wind is present. *wind_up* event is modeled as an uncontrollable movement of the agent toward $-y$ direction. This proves that the supervisor could handle even uncontrollable or unobservable events in the map automaton. Uncontrollable events could model, among others, wind or uncontrollable forces of the environments, interactions with the environments. On the other hand, unobservable events could model, among others, zones in which GPS signal is not present.

The specification automaton is dynamically built from map automaton, by specifying the forbidden states (obstacles, forbidden areas, and dangerous areas) and the target (marked state). In this simulation, we consider this automaton static so as both the target and the forbidden states are fixed, but the validity could be extended to a dynamic case in which the target and the forbidden states change online. The forbidden states for this simulation are: $3-2$, $3-3$, $3-4$, $3-5$, $4-5$, the target is set in the state $1-5$, and the initial state (initial position of the robot) is set to $5-4$. The map automaton and the specification automaton resulting from the map and the specifications are depicted in Fig. 1.

The primitives (possible maneuvres, events) of the agent are eight: *go_north*, *go,_n-w*, *go_n-e*, *go_east*, *go_west*, *go_south*, *go_s-w*, *go_s-e*. They represent straight lines at 45° each (i.e., the eight directions north, north-west, west, south-west, etc). We assume that after a primitive, it is possible to have a primitive which differs only 45° to the previous one. This will ensure that the final trajectory will be a broken line with only 45° of difference between consecutive lines. This could be a limitation

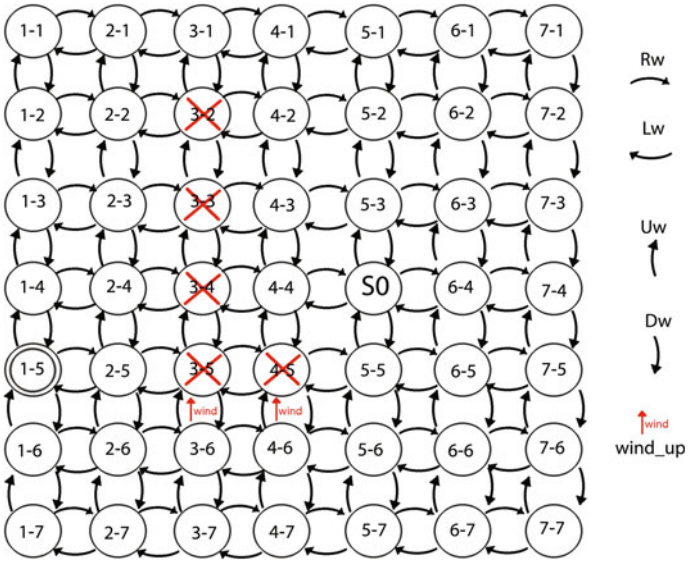

Fig. 1 Specification automaton

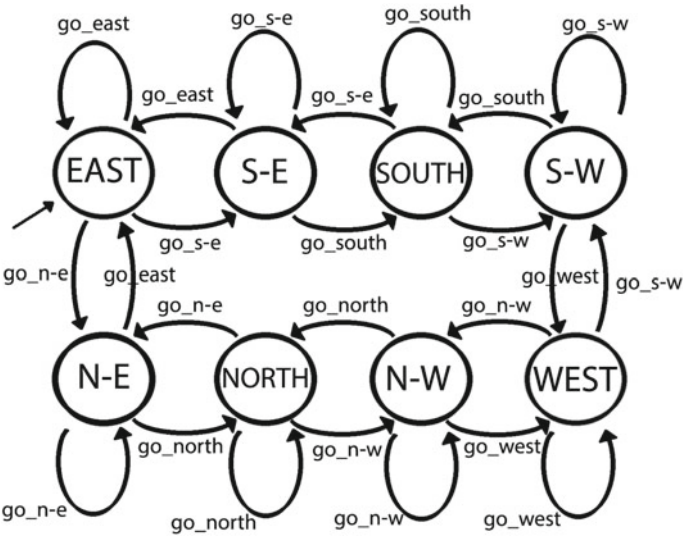

Fig. 2 Agent automaton

for a fully actuated model but ensures a smoother trajectory with less constrains on derivative bounds. A simplified automaton of the agent is depicted in Fig. 2. The real agent automaton should include the interaction with the map, while this represents only the behavior in terms of primitives. The primitives are taken as linear motion with constant speed in the eight directions. In particular, the speed on every direction is taken as 1.2 m/s. For sake of compactness, we use the index $i = 1, 2, ..., 8$ to identify the primitives so as, *go_north*$\Rightarrow i = 1$, *go_n-e*$\Rightarrow i = 2$, *go_east*$\Rightarrow i = 3$, *go_s-e*$\Rightarrow i = 4$, *go_south*$\Rightarrow i = 5$, *go_s-w*$\Rightarrow i = 6$, *go_west*$\Rightarrow i = 7$, *go,_n-w*$\Rightarrow i = 8$. With the discretization of 2 m and assuming that every maneuver is completed in $\frac{2}{1.2}s$ the position, velocity and acceleration of every time-trajectory mapping the primitive are defined as

$$\ddot{x}_{1R,i}(t) = 0 \ \forall i \qquad \ddot{y}_{1R,i}(t) = 0 \ \forall i$$

$$\dot{x}_{1R,i}(t) = \begin{cases} 1.2 & i = 2, 3, 4 \\ 0 & i = 1, 5 \\ -1.2 & i = 6, 7, 8 \end{cases} \qquad \dot{y}_{1R,i}(t) = \begin{cases} 1.2 & i = 4, 5, 6 \\ 0 & i = 3, 7 \\ -1.2 & i = 1, 2, 8 \end{cases}$$

$$x_{1R,i}(t) = \dot{x}_{R,i}(t)t + x_{R0} \ \forall i \qquad y_{1R,i}(t) = \dot{y}_{R,i}(t)t + y_{R0} \ \forall i$$

where $x_{1R,i}(t)$ and $y_{1R,i}(t)$ denote the references for the x and y position for the i-th primitive and x_{R0}, y_{R0} represent the position references before the switch, i.e., the final position of the previous trajectory piece, so as the position reference is continuous.

The initial state is *EAST* state, meaning that the initial primitive could be only *go_east*, *go_n-e* or *go_s-e*.

The supervisor is then built with classical DES supervisor theory, basically performing a parallel composition between specification automaton and agent automaton, requesting in a controllable supervisor. The optimization tool is then used to find the optimal string between the feasible strings. The supervisor (parallel composition) is evaluated online every time a change in the map occurs; so when an obstacle changes position, an obstacle appears/disappears in the map or the target changes position.

4.2 Low-Level Control

Low-level control is defined with matrix K_s. We need to ensure dwell time condition of Lemma 1 for the goal (3) and to make the system stable. By having defined the primitives of the supervisor, one can define the parameter d. In particular, by noticing that two consecutive concatenated trajectory will have a difference in position equal to 0 and a maximum difference in velocity equal to 1.2, one can pick $d = 1.2$. Defined d, one can choose $q > d$. In our case, we pick $q = 1.4$ to impose the maximum tracking error. By choosing the regulator parameter $a = 40$, we obtain

$$K_s = \begin{bmatrix} 4.4721 & 4.9469 & 0 & 0 \\ 0 & 0 & 4.4721 & 4.9469 \end{bmatrix}, \quad P_s = \begin{bmatrix} 44.2467 & 4.4721 & 0 & 0 \\ 4.4721 & 4.9469 & 0 & 0 \\ 0 & 0 & 44.2467 & 4.4721 \\ 0 & 0 & 4.4721 & 4.9469 \end{bmatrix} \tag{14}$$

with a minimum dwell time condition $\tau \geq 1.4844$ which is respected by the $\frac{2}{1.2}$s switch time of the supervisor.

5 Simulations

A simulation was run to test the proposed control and to verify that the conditions for the reference trajectory respect (3). The resulting trajectory is obtained applying the optimization on the supervisor graph, in particular in this case the Dijkstra algorithm [3] was applied to obtain the minimum distance trajectory. In particular, every state of the supervisor was handled as a node of the graph and every event as an arc of the graph and weighting every primitive with the related distance so as the four primitives $i = 1, 3, 5, 7$ were weighted 1 while the other four primitives $i = 2, 4, 6, 8$ were weighted $\sqrt{2}$. The reference generated by the supervisor and the real position of the system are depicted in Fig. 3.

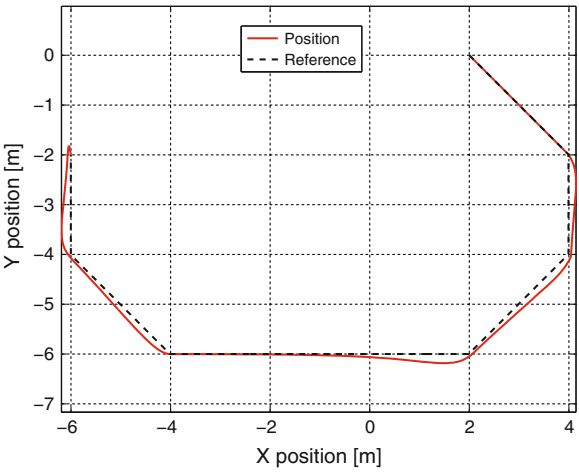

Fig. 3 Position reference generated by the supervisor and real position of the system in the x-y plane

The supervisor generated the expected trajectory, following the constrains of the agent automaton, choosing the shortest path, avoiding obstacles, and avoiding uncontrollable events.

Figures 4 and 5 depict, respectively, x and y position and velocity trajectory tracking. The goal (3) is respected for the whole trajectory.

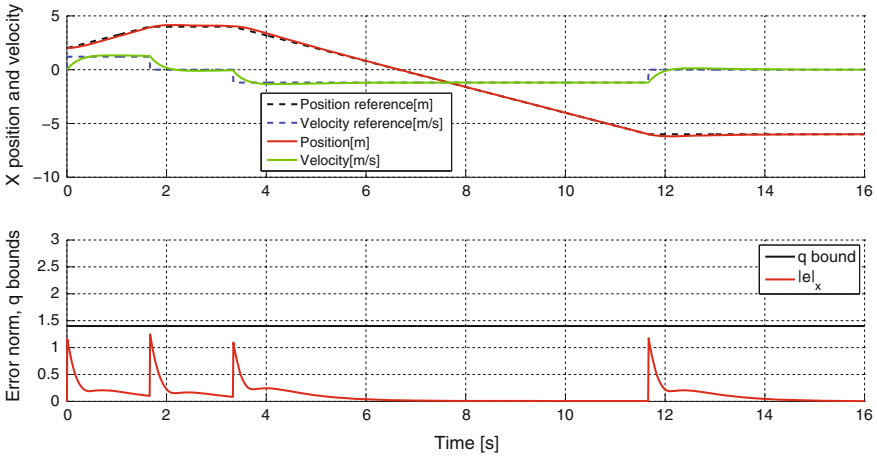

Fig. 4 First plot includes the x position and velocity reference and the real x position and velocity. Second plot includes the norm of the x error and the bound given by the parameter q: the bound is respected for the whole time

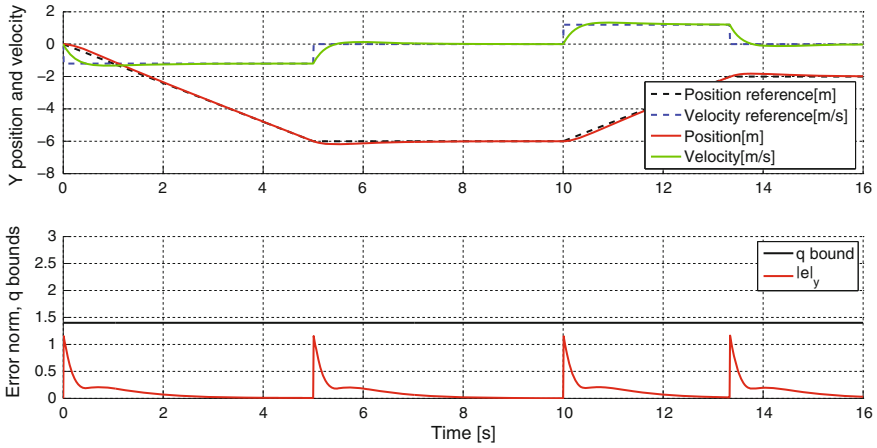

Fig. 5 First plot includes the y position and velocity reference and the real y position and velocity. Second plot includes the norm of the y error and the bound given by the parameter q: the bound is respected for the whole time

6 Conclusion and Future Works

This work proposed a novel control law to navigate robots in a dynamic environment. The control tool is composed by a low-level closed-loop control law and a high-level supervisor. The advantages of this strategy is to fully exploit the capabilities of the supervisor while mantaining a robust practical tracking of the low level control, ensuring an upper bound on the tracking error. The application to a model of the kinematic of an aerial vehicle was shown to prove the effectiveness.

References

1. C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G.j. Pappas. Symbolic planning and control of robot motion: Finding the missing pieces of current methods and ideas. *IEEE Robotics & Automation Magazine*, pages 61–70, March 2007.
2. C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event System*. Springer, 2008.
3. E.W. Dijkstra. A note on two problems in connexion with graph. *Numerische Mathematik*, 1:269–271, 1959.
4. E. Frazzoli. Robust hybrid control for autonomous vehicle motion planning. Ph.D. Thesis, Massachusetts Institute Of Technology, 2001.
5. E. Frazzoli, M.A. Dahlel, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transaction on Robotics*, 21(6):1077–1091, december 2005.
6. M. Furci, R. Naldi, and A. Paoli. A supervisory control strategy for robot-assisted search and rescue in hostile environments. *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, September 2013.
7. A. Hornung, K.M. Wurm, M. Bennewitz, Stachniss C, and W. Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octress. *Autonomous Robots*, 34:189–206, 2013.

8. S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30:846–894, 2011.
9. L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transaction on Robotics and Automation*, August 1994.
10. H.K Khalil. *Nonlinear System*. Prentice Hall, 2002.
11. S.M. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Computer Science Dept., Iowa State University*, October 1998.
12. S.M. LaValle. Motion planning: The essentials. *IEEE Robotics & Automation Magazine*, 18:79–89, 2011.
13. Daniel Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applications. Birkhauser, Boston, MA, June 2003.
14. R.R. Murphy. Humans, robots, rubble and research. *Interactions*, 12:37–39, 2005.
15. P. Pounds, R. Mahony, and P. Corke. Modelling and control of a large quadrotor robot. *Control Eng. Pract*., 18(7):691–699, 2010.
16. J. Ryde and H. Hu. 3d mapping with multi-resolution occupied voxel list. *Autonomous Robots*, 28:169–185, 2010.
17. R.G. Sanfelice and E. Frazzoli. A hybrid control framework for robust maneuver-based motion planning. *American Control Conference*, pages 2254–2259, 2008.