Robotics and Autonomous Systems 86 (2016) 113–127

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

A hybrid approach for autonomous navigation of mobile robots in partially-known environments

Madjid Hank *, Moussa Haddad [1]

Ecole Militaire Polytechnique, BP 17, Bordj El-Bahri, 16111 Algiers, Algeria

HIGHLIGHTS

- Our proposal includes an offline task-independent preprocessing phase, based on near-time-optimal trajectory planning.
- It combines trajectory–tracking and reactive–navigation modes to execute a task.
- It includes a transition module ensuring smooth transitions between navigation modes.
- It can circumvent unexpected obstacles when tracking a trajectory.
- Results show the effectiveness of our proposal to achieve short execution times.

ARTICLE INFO

Article history:
Received 14 December 2015
Accepted 16 September 2016
Available online 23 September 2016

Keywords:
Hybrid navigation
Mobile robots
Trajectory planning
Trajectory tracking
Reactive navigation
Fuzzy logic

ABSTRACT

We propose a hybrid approach specifically adapted to deal with the autonomous-navigation problem of a mobile robot which is subjected to perform an emergency task in a partially-known environment. Such a navigation problem requires a method that is able to yield a fast execution time, under constraints on the capacity of the robot and on known/unknown obstacles, and that is sufficiently flexible to deal with errors in the known parts of the environment (unexpected obstacles). Our proposal includes an off-line task-independent preprocessing phase, which is applied just once for a given robot in a given environment. Its purpose is to build, within the known zones, a roadmap of near-time-optimal reference trajectories. The actual execution of the task is an online process that combines reactive navigation with trajectory tracking and that includes smooth transitions between these two modes of navigation. Controllers used are fuzzy-inference systems. Both simulation and experimental results are presented to test the performance of the proposed hybrid approach. Obtained results demonstrate the ability of our proposal to handle unexpected obstacles and to accomplish navigation tasks in relatively complex environments. The results also show that, thanks to its time-optimal-trajectory planning, our proposal is well adapted to emergency tasks as it is able to achieve shorter execution times, compared to other waypoint-navigation methods that rely on optimal-path planning.

1. Introduction

In the present work, we are interested in the autonomous-navigation problem of a mobile robot subjected to perform an emergency task to respond to an accident in a partially-known environment. Such an environment is often defined in the literature as a structured environment that possibly includes unexpected static/dynamic obstacles. In our case, the environment is composed of two types of zone: *disaster* and *safe* zones. For the first type, we assume that we know the position and orientation of the entry/exit, but the structure will be considered unknown as any prior information may no longer be reliable due to the accident. For the second type, the structure will be considered known, but unexpected obstacles are not excluded.

Navigation in this type of environments has important potential applications, all of which require a fast execution of the task (for example, an emergency intervention in case of fire). In order to handle efficiently such a navigation problem, it is necessary to exploit fully: (i) the perception/decision capability of the system; (ii) the information already available about the known parts of the environment; and (iii) the capacity of the robot at hand.

Approaches that are based solely on reactive navigation, although they will benefit from (i), still have obvious limitations. They may lead to prohibitively long execution time as they do not consider (ii) nor do they take full advantage of (iii). Much more

* Corresponding author. Fax: + 213 21 863 204.
E-mail addresses: madjid.hank@gmail.com (M. Hank), haddadmoussa2003@yahoo.fr (M. Haddad).
[1] Fax: + 213 21 863 204.

http://dx.doi.org/10.1016/j.robot.2016.09.009
0921-8890/

interesting are hybrid approaches that combine, for example, a reactive-navigation method (in the disaster zones) with an optimal path-planning method (in the safe zones). However, although a path-planning certainly will help in reducing the execution time, it may still remain insufficient as it usually accounts *consecutively* for (ii) and (iii) rather than *simultaneously*. Indeed, optimal-path planning, even if subsequently followed by optimal-motion planning, does not lead generally to a time-optimal solution.

Here, we propose a hybrid approach that combines a reactive-navigation method with a near-time-optimal *trajectory-planning* method which, in contrast to a *path-planning* method, considers (ii) and (iii) simultaneously. This approach includes an offline task-independent preprocessing phase, which is applied just once for a given robot in a given environment and thus before any accident occurs. Its purpose is to build, beforehand, a roadmap of near-time-optimal reference trajectories interconnecting a set of feature points in the known parts of the environment. Then, given a task, an online execution phase will move the robot from start to goal. In the safe zones, it will benefit from relevant portions of the roadmap by tracking reference trajectories pre-computed in there; whereas in the disaster zones, it will resort to reactive navigation.

The key features of our proposal can be summarized as follows. First and foremost, we take full advantage of the known parts of the environment and the capacity of the robot by using *time-optimal* trajectory planning under constraints instead of optimal path planning. This allows us to obtain shorter execution times, which make our proposal well adapted to emergency tasks requiring a rapid intervention. Second, we use a specific trajectory-planning method that allows us to reduce drastically memory storage and that thus can be very useful in the case of complex environments that require to save a large roadmap of trajectories in the safe zones. Third, the trajectory tracking module is fed with the complete time-history information, including not only a path to be followed but also a valid motion that already accounts for known obstacles and for the capacity of the robot at hand. Finally, we use a specific transition module both to handle unexpected obstacles and to toggle smoothly between trajectory tracking and reactive navigation, which further improves the time required to execute the task.

The rest of this paper is organized as follows. Section 2 is devoted to some related works. Section 3 defines the navigation problem. Section 4 describes the proposed hybrid approach. Section 5 presents and discusses some simulation results, obtained using the *Powerbot* robot, and some preliminary experimental results using the Pioneer P3-AT robot.

2. Related work

Autonomous navigation problem of mobile robot in different types of environments is largely studied in the literature. Depending on available information about the navigation environment and the specifications of the navigation task, several types of approaches are proposed.

Classical deliberative approaches, based on path or trajectory planning according to a performance criterion (distance, travel time, energy consumption ...) in completely known environments, ensure safe and fast solution navigation. Reactive navigation approaches have received a large amount of attention in order to deal with an important sub-problem of mobile robot navigation which is obstacles avoidance as those are unknown, unexpected or dynamic. The majority of reactive navigation approaches are based on combining a set of competitive behaviors. According to the used arbitration method between behaviors, two major categories are proposed in the literature: the competitive Brook's Subsumption architecture and the cooperative Motor-Schema architectures [1,2]. However, such approaches suffer from the problems of local minima (non convergence to a feasible solution). In addition, the robot traveled trajectory is not optimal in terms of distance and/or travel-time, due to lack of a global vision on the environment.

Given the drawbacks of the two types of methods, nowadays, many researchers are interested in hybrid approaches combining the two previous types in order to take advantage of both of them [3]. This choice allows safer and faster navigation solutions, from one hand, by taking into account prior knowledge on the environment to minimize the risk of blockage in local minima and reduce the travel time and, from another hand, by considering local perception of the environment to adapt to unexpected situations.

In the literature, several hybrid approaches have been developed to solve the problem of mobile robot navigation in a partially-known environment [2–7]. However, all these approaches deal with the problem of avoiding unexpected static/dynamic obstacles in structured environments. Some works combine a reactive-navigation method with the following of a path defined from a set of waypoints [7–15]. Sometimes, the waypoints are obtained using a path-planning method [16–18].

In some other cases, limits on velocity and acceleration are taken into account during the reactive-navigation phase [19] by using the dynamic window approach (DWA) to execute the path defined by a set of waypoints. An example of these approaches is the one implemented in ROS which incorporates both global and local path planners [20–22]. Basic algorithms are defined for the global path planner such as A*, Dijkstra, and carrot planners in order to define the shortest path between start and goal. The local path planner is based mainly on the Trajectory Rollout [23] and the DWA [24] algorithms that take into account kinematics and/or dynamic constraint while executing the pre-planned path composed of a set of waypoints.

Other works combine a reactive-navigation method with the following of a predefined path and the robot motion on this path is defined via a motion-planning method at constant speed [25] or via a trapezoidal velocity profile [26]. Another class of hybrid approaches combines reactive navigation with trajectory-planning. For instance, Castro et al. [27] and Santos et al. [28] propose multi-loop modular-navigation architecture for mobile robot navigation in partially-modeled environments. An optimization algorithm provides a path between sub-goals. Then, an online trajectory-planning module generates smooth trajectories based on pairs of clothoid curves. Reactive-navigation or tele-operation modes are used to deal with unexpected situations. Mouad et al. [29] propose a method composed of a planning and re-planning module, which determines an initial path based on potential fields. If an unexpected obstacle is detected while tracking a reference trajectory, the navigation system switches to a reactive navigation based on the so-called orbital obstacle avoidance algorithm.

3. Problem statement

Fig. 1 shows an example of the navigation environment. It is composed of several zones with well-defined boundaries.

Lightly-shaded areas constitute accident-prone critical places requiring careful attention: any of which may need an emergency intervention in the future. The structure of these potential-disaster zones will be considered unknown. Obstacles in there are shown in dark gray. In contrast, areas painted in white are passageways, hereafter called safe zones, whose structure will be considered known. Most obstacles in there are supposed to be identified (shown in black), but occasional unexpected obstacles (shown in dark gray) are not excluded.

We assume we have sufficient information about the environment to build a set of n feature points, denoted $\boldsymbol{P}_k = [x_k, y_k]^T$ where $k = 1 \ldots n$. This set includes the entry/exit of the unknown zones as well as any additional relevant point, such as roundabout,

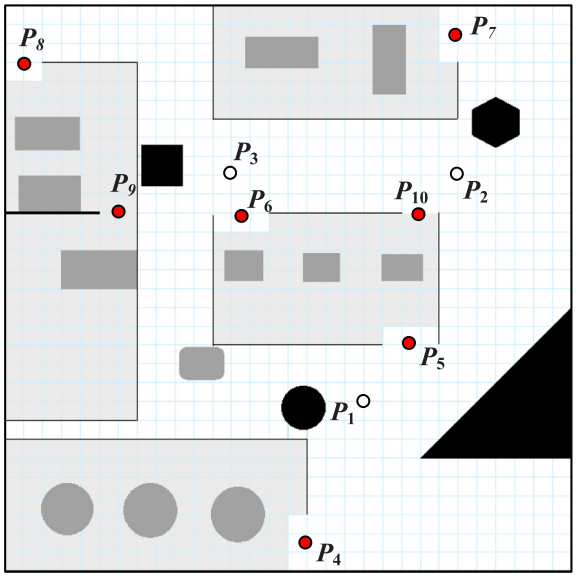


Fig. 1. Example of a partially-known environment. Known obstacles are shown in black. Unknown/unexpected obstacles are shown in dark gray. Lightly-shaded areas indicate zones with unknown structure. Two types of feature points $\boldsymbol{P}_k$ can be distinguished. Those shown as small filled circles are the entry/exit of the unknown zones. The others (empty circles) are user-defined feature points which can be positioned strategically in the known zones and which would later help in guiding the navigation.

road intersection, etc., which we might want to pre-select in the safe zones and which would later help guide the navigation of the robot. Note that these additional feature points need not be mutually visible. In fact, the fewer, the better; as their role is mainly to indicate the places where multiple potential pathways exist for the robot.

The emergency task will consist of moving the robot from a given start configuration $\boldsymbol{S}$ to a given goal configuration $\boldsymbol{G}$. We assume that, at each instant, the robot has information about its position/orientation (provided by a localization system) and distances to obstacles (provided by a perception system). The problem is to compute the linear and angular velocities $v(t)$ and $w(t)$ so as to minimize the *total travel time* from $\boldsymbol{S}$ to $\boldsymbol{G}$ under constraints: (i) on the task (boundary conditions on position, orientation, and velocity); (ii) on the environment (avoidance of known, unknown, and unexpected obstacles); and (iii) on the capacity of the robot.

4. Proposed hybrid approach

4.1. Overview

The proposed hybrid approach combines trajectory-tracking (TT) and reactive-navigation (RN) modes. It also includes smooth transitions between these two types of modes so as to ensure motion continuity and to improve performance. The tracked trajectories are near-time-optimal reference trajectories. They are pre-computed off-line, in the known parts of the environment, taking into account the capacity of the robot and known obstacles. Our proposal includes thus two distinct phases. The first is an off-line task-independent preprocessing, which is performed once for a given robot in a given environment. The second handles online the actual execution of a given task after the accident occurs.

The purpose of the preprocessing phase is to build an initial graph (Fig. 2(a)) whose nodes coincide with the feature points and whose edges can be grouped in two categories, depending on whether they belong to a safe zone or not. If an edge of the graph belongs to a safe zone, then a reference trajectory linking its two end-nodes will be computed and stored for future use. If no such a trajectory is feasible for the robot at hand, then this edge is simply deleted. Otherwise, it will be kept and will represent the path of a reference trajectory and its cost will be the *travel time* of this trajectory. If an edge does not belong to a safe zone, then no corresponding trajectory will be computed. The cost (travel time) of such an edge will be roughly estimated simply based on the straight-line distance between nodes and on the capacity of the robot, including however a multiplicative penalty factor. The penalty must be heavy enough so that this type of edge will be considered only in extreme cases to avoid deadlocks or too costly alternatives.

The whole graph is stored in memory to be used whenever needed to execute a given task.

The execution phase is an online process that deals with the task which consists of moving the robot, as fast as possible, from a given start $\boldsymbol{S}$ to a given goal $\boldsymbol{G}$. First, the graph is augmented (Fig. 2(b)) by incorporating $\boldsymbol{S}$ and $\boldsymbol{G}$, linking them via a straight line to their respective closest-neighbor node of the graph. The newly added edges being compulsory, their cost is irrelevant and, as such, can be set to any positive value. Second (Fig. 2(c)), a standard graph optimization algorithm (such as that of Dijkstra [30]) is used to get the optimal edge succession that minimizes the estimated total travel time of moving from $\boldsymbol{S}$ to $\boldsymbol{G}$. Finally, depending on the nature of each edge, TT or RN modes are activated to perform the task. The resulting executed path is shown as a dashed curve in Fig. 2(d).

Note that a smooth transition will be ensured at each junction (feature point), so that the executed path will not go necessarily through every node of the graph. Moreover, when tracking a reference trajectory, if an unexpected obstacle is encountered, the robot will try to avoid it before resuming the tracking.

Next, we present some details of the proposed hybrid approach. As shown in Fig. 3, which gives an overview of the navigation system, our proposal includes several separate modules. To ease the implementation, we propose for each module a simple method, but more sophisticated alternatives could be considered as well to improve our proposal.

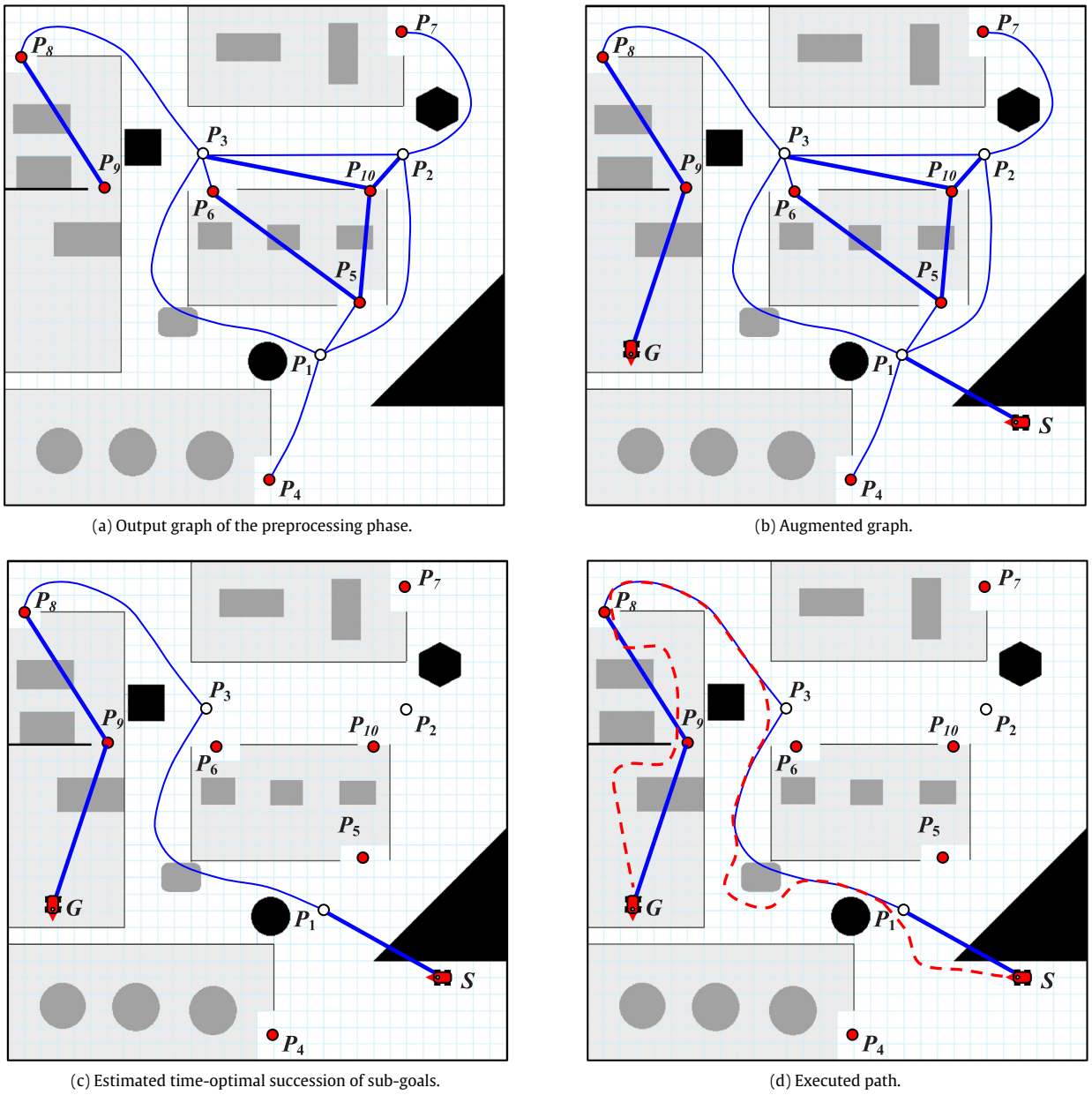
4.2. Planning reference trajectories

In order to pre-compute near-time-optimal reference trajectories between interconnected nodes of the graph in the safe zones, we use the Random Profile Approach (RPA) [31,32]. With RPA, a trajectory is split into a path and a motion on this path, both of which are modeled as parametric functions via a small number (typically <20) of control points. Then, the whole trajectory-planning problem is converted to finding the optimal position of all these control points [31].

One of the most attractive features of RPA is that it helps in reducing drastically memory requirements for storing pre-computed reference trajectories. For instance, there will be no need to store detailed time-history information on positions, velocities or torques at regular time intervals. Indeed, once a trajectory is obtained via RPA, any future request about this trajectory can be easily extracted *online* directly from the corresponding optimal control points.

Another interesting feature of RPA is that it is able to provide reference trajectories of good quality without requiring any prior simplification of the robot dynamic model. In fact, it is able to account for: (i) geometric constraints (obstacle avoidance, boundary conditions on position and orientation); (ii) kinematic constraints (limits in speed and acceleration); and if necessary (iii) dynamic constraints (limits on torques, dynamic stability of the robot).

Moreover, in our case, a reference trajectory computed between sub-goals $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$ will be the same as that between $\boldsymbol{P}_j$ and $\boldsymbol{P}_i$. Although the directions of the robot at these end-points must be pre-defined by the user, the reference trajectory thus computed will not be tracked completely.



(a) Output graph of the preprocessing phase.

(b) Augmented graph.

(c) Estimated time-optimal succession of sub-goals.

(d) Executed path.

Fig. 2. Main steps of the proposed hybrid approach. (a) Thin solid curves show the path of reference trajectories pre-computed in the known zones; (b) Augmented graph obtained by linking the start and goal points to their respective closest-neighbor node; (c) Estimation of the best node succession leading from start to goal; (d) The actual executed path is shown as a dashed curve.

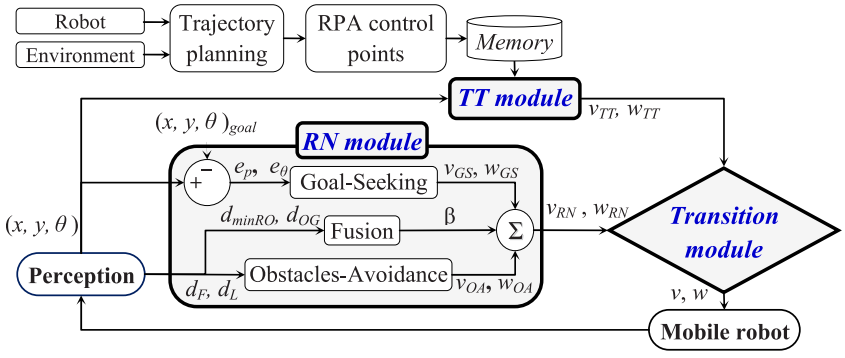
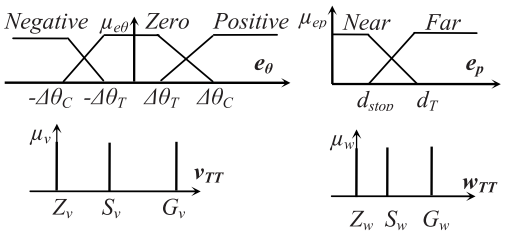
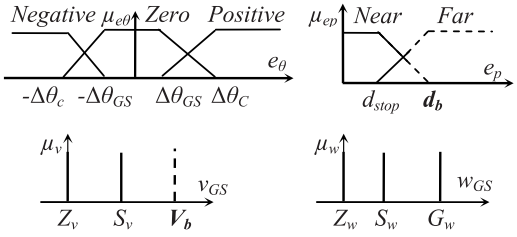


Fig. 3. Global view of the navigation system. The navigation system is composed of three main modules. The TT module is used for tracking pre-computed reference trajectories. The RN module is used for reactive navigation. The transition module is used for ensuring a smooth switch between different navigation modes and includes the procedure to circumvent unexpected obstacles encountered while tracking a reference trajectory. Further details are provided in Section 4 of the main text.



e_p \ e_θ		Negative	Zero	Positive
Near	v_{TT}	Z_v	Z_v	Z_v
	w_{TT}	S_w	Z_w	$-S_w$
Far	v_{TT}	S_v	G_v	S_v
	w_{TT}	G_w	Z_w	$-G_w$

Fig. 4. Fuzzy-rule base and output membership functions of the TT controller. The inputs are the tracking errors in position and orientation (e_p, e_θ). The outputs are the linear and angular velocities (v_{TT}, w_{TT}). The predefined parameters of this controller are the thresholds (tracking and stopping distances and angles d_T, d_{stop}, $\Delta\theta_T$ and $\Delta\theta_c$) and the singleton values (Z_v, S_v, G_v, Z_w, S_w, G_w). The rule consists in keeping e_p and e_θ within the thresholds. If $e_p > d_T$, the robot will try to catch up the virtual robot with a maximum speed; whereas if $e_p < d_{stop}$, it will stop. Moreover, if $e_\theta < \Delta\theta_T$, the robot will move straightly; whereas if $e_\theta > \Delta\theta_c$, the orientation must be corrected.



e_p \ e_θ		Negative	Zero	Positive
Near	v_{GS}	S_v	Z_v	S_v
	w_{GS}	$-S_w$	Z_w	S_w
Far	v_{GS}	V_b	V_b	V_b
	w_{GS}	$-G_w$	Z_w	G_w

Fig. 5. Fuzzy-rule base and output membership functions of the Goal-Seeking sub-module. The threshold values are, d_{stop}, the stopping distance, and ($\Delta\theta_{GS}$, $\Delta\theta_c$) the Goal-Seeking and correction angles. The braking distance d_b and the maximum velocity V_b are adaptive parameters computed via Eq. (1). The fuzzy-rule base is established using the same principle as that of the TT module (see Fig. 4).

4.3. Trajectory-tracking module

The role of the TT module is to track a reference trajectory which had been pre-computed between two given sub-goals in a safe zone. A suitable choice for the tracking method is dictated by the following objectives: (i) the execution time must be as close as possible to that of the reference trajectory; (ii) the method must be sufficiently robust to handle different types of modeling errors (robot and environment); and (iii) the method must be sufficiently flexible to adapt to unexpected obstacles.

To meet these objectives, we use a TT module based on the tracking of a virtual robot [33] which will be moving along the reference trajectory. Such a method is characterized by its simplicity of implementation. When tracking a trajectory, it strives to keep the tracking errors in position and orientation within predefined limits. In our case, this criterion will be easily satisfied as the virtual robot will be moving along a reference trajectory which had been computed under constraints on the capacity of the real robot.

For the implementation, we use a Takagi–Sugeno fuzzy controller [34], which only needs an implicit description of the robot behavior by the fuzzy rule. The controller generates the necessary commands to minimize errors in position (e_p) and in orientation (e_θ) between real and virtual robots. The calculation steps are the classic stages of a fuzzy controller [34]: fuzzification, fuzzy inference and defuzzification. The inputs are (e_p, e_θ). The outputs are (v_{TT}, w_{TT}) the linear and angular velocities. As detailed in Fig. 4, the predefined internal parameters of this controller are the thresholds for decision making (d_T, d_{stop}, $\Delta\theta_T$, $\Delta\theta_c$) and the singleton values (Z_v, S_v, G_v, Z_w, S_w, G_w) to be used for the outputs.

4.4. Reactive-Navigation module

The role of the RN module is to move the robot inside an unknown zone from a given sub-goal to another, while avoiding encountered obstacles. We have selected a method which is based on a behavioral architecture of Motor-Schema type [35–38]. As previously shown in Fig. 3, we use three fuzzy-logic sub-modules: *Goal-Seeking*, *Obstacle-Avoidance*, and *Fusion*. Using a weighting coefficient, determined via the *Fusion* sub-module, this method allows the robot to reach the next sub-goal as fast as possible. The modularity of this system allows the robot to adapt easily to various situations.

4.4.1. Goal-Seeking sub-module

The *Goal-Seeking* sub-module is implemented using a simplified version of the method proposed in [36]. The membership functions and the fuzzy-rule base are illustrated in Fig. 5. The inputs are the errors (e_p, e_θ), which are now measured relatively to the current sub-goal. The outputs are the linear and angular velocities (v_{GS}, w_{GS}).

The principle is the same as that of the TT module previously described in Fig. 4, but the setting of the internal parameters is different. Moreover, the previous thresholds d_T and $\Delta\theta_T$ are now replaced, respectively, by the braking distance d_b and the goal-seeking angle $\Delta\theta_{GS}$; whereas, the previous singleton G_v is now replaced by the braking velocity V_b. Note also that d_b and V_b are no longer user-defined, but are computed as follows:

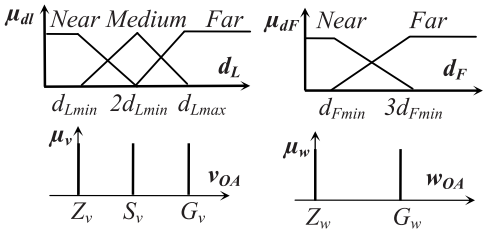
$$\begin{aligned} &d_{\max} = v_{\max}^2/(2\times a_{\max}) \\ &(V_b = v_{\max} \quad \text{and} \quad d_b = d_{\max}) \quad \text{if} \quad e_p \geq d_{\max} \\ &(V_b = \sqrt{e_p \times a_{\max}} \quad \text{and} \quad d_b = e_p/2) \quad \text{else.} \end{aligned} \tag{1}$$

Here, $v_{\max}$ and $a_{\max}$ are the maximum linear velocity and acceleration. The linear velocity output, v_{GS}, will be trapezoidal in shape, thereby reducing the time to reach a given sub-goal.

4.4.2. Obstacle-Avoidance sub-module

Obstacle avoidance is a standard feature present in all navigation systems. We have chosen a *wall-following* method. The fuzzy controller is of Takagi–Sugeno type [34]. As shown in Fig. 6, the inputs are the lateral and frontal distances (d_L, d_F). The outputs are the linear and angular velocities (v_{OA}, w_{OA}). The predefined parameters are the thresholds (d_{Lmin}, d_{Lmax}, d_{Fmin}) and the singletons values (Z_v, S_v, G_v, Z_w, G_w).

We use a single fuzzy controller for right and left wall following. For example, in the case of the left wall following, the behavior is



d_F \ d_L		Near	Medium	Far
Near	v_{OA}	Z_v	S_v	S_v
	w_{OA}	$-G_w$	Z_w	$-G_w$
Far	v_{OA}	G_v	G_v	G_v
	w_{OA}	Z_w	G_w	G_w

Fig. 6. Fuzzy-rule base and output membership functions of the Obstacle-Avoidance sub-module.

managed as follows:

$$\begin{aligned} &w_{OA} \geq 0 \quad \text{if} \quad (d_L > d_{Lmax}) \text{ and } (d_F > d_{Fmin}) \\ &w_{OA} = 0 \quad \text{if} \quad (d_{Lmin} \leq d_L \leq d_{Lmax}) \text{ and } (d_F > d_{Fmin}) \\ &w_{OA} < 0 \quad \text{if} \quad (d_L < d_{Lmin}) \text{ or } (d_F < d_{Fmin}). \end{aligned} \tag{2}$$

A similar procedure is used for right wall-following by reversing the sign of the angular velocity w_{OA}. The choice between left or right is made according to the minimum of the left and right distances to obstacle.

4.4.3. Fusion sub-module

The *Fusion* sub-module is based on the principle of concurrence between the *Goal-Seeking* and *Obstacle-Avoidance* sub-modules [36,38]. This method determines the appropriate weighting between the competing sub-modules depending on the situation that arises (free or occupied space).

The fuzzy controller is of Takagi–Sugeno type [34]. As detailed in Fig. 7, the inputs are (d_{minRO}, d_{OG}). The output is a weight coefficient $0 \leq \beta \leq 1$. The predefined internal parameters are the threshold d_s, which is a safety distance, and the singletons values (Z_β, U_β).

The coefficient β is computed as follows:

$$\begin{aligned} &\beta = 0 \quad \text{if} \quad (d_{minRO} < d_s) \text{ and } (d_{OG} < 1.5 d_s) \\ &\beta = 1 \quad \text{else.} \end{aligned} \tag{3}$$

The inferred actions of *Goal-Seeking* and *Obstacle-Avoidance* sub-modules are weighted to compute the output of the RN module as follows:

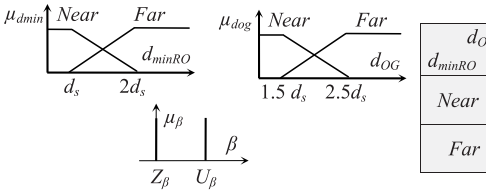
$$\begin{aligned} v_{RN} &= \beta\, v_{GS} + (1-\beta) v_{OA} \\ w_{RN} &= \beta\, w_{GS} + (1-\beta) w_{OA}. \end{aligned} \tag{4}$$

Provided that the parameters of the controller are chosen appropriately, this method allows a reactive and fast conflict management between sub-modules.

4.5. Transition module

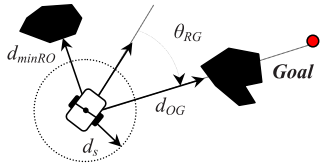
The transition module plays a key role in improving the performance of the proposed hybrid approach. This module, which acts as a supervisor, is responsible for ensuring a smooth transition between different navigation modes. Switching from one mode to another is performed according to a transition radius R_T, which is pre-defined according to the information available on the robot and the environment. We distinguish five types of transitions, one of which handles the avoidance of unexpected obstacles while tracking a reference trajectory.

Fig. 8 illustrates the case of a TT–RN transition. The robot, which is in TT mode, should normally track the reference trajectory completely to the point ***A***. Then, it would stop at this point before switching to the RN mode to join the sub-goal ***B***. The transition module helps improve the execution of such a task by launching the RN mode earlier at ***C***, thus preventing the robot to move till ***A***,



d_{minRO} \ d_{OG}	Near	Far
Near	Z_β	U_β
Far	U_β	U_β

(a) Fusion sub-module.



(b) Input signification.

Fig. 7. Fuzzy-rule base and output membership functions of the Fusion sub-module.

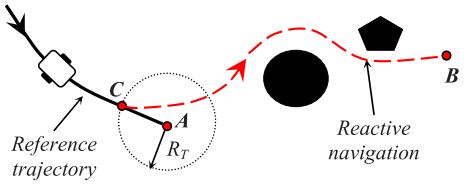


Fig. 8. TT–RN transition.

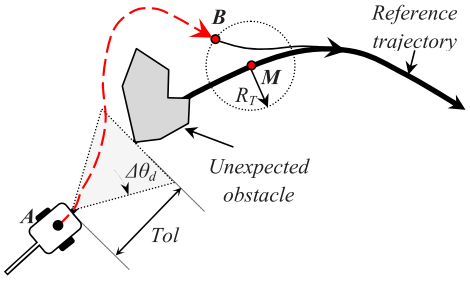
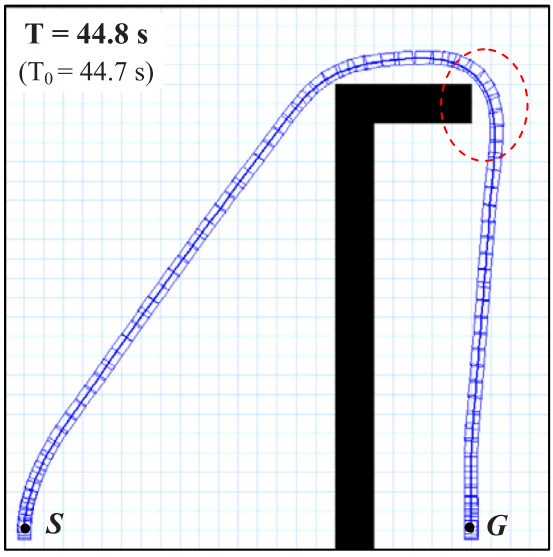


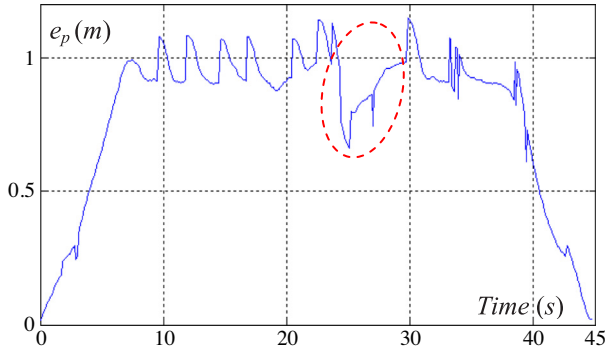
Fig. 9. AUO transition.

avoiding thus an unnecessary stop there. This switching is made as soon as the robot enters a transition zone centered on ***A*** and of radius R_T.

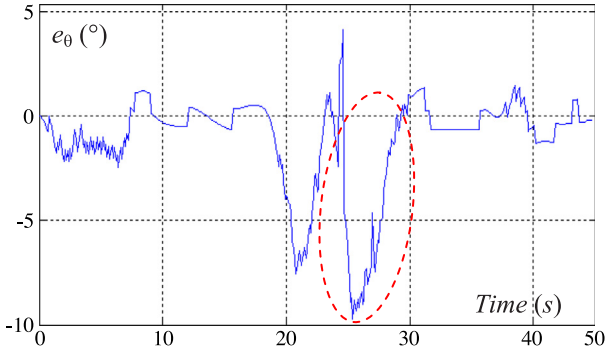
The RN–TT transition concerns the opposite situation (see for example in Fig. 2(d) the switching from $\boldsymbol{S}$–$\boldsymbol{P_1}$ to $\boldsymbol{P_1}$–$\boldsymbol{P_3}$). Here, however, the switching decision is not based solely on the transition radius, but also requires that the beginning of the reference trajectory be visible.



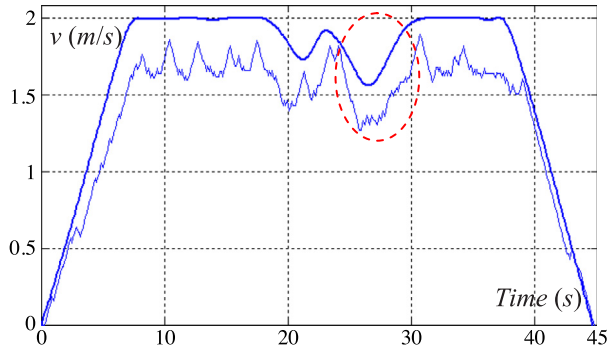
(a) Executed path by the TT module.



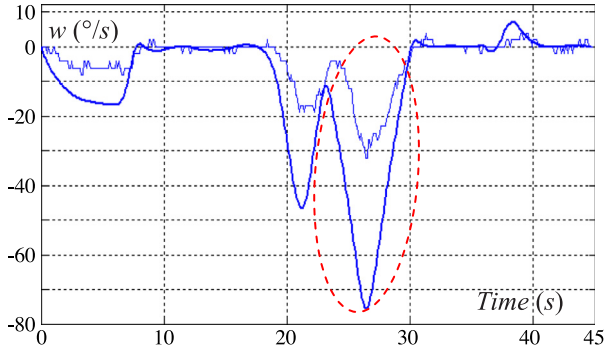
(b) Tracking errors in position (m).



(c) Tracking errors in orientation (°).



(d) Linear velocity (m/s).



(e) Angular velocity (°/s).

Fig. 10. Results obtained for the example 1. (a) The path of the reference trajectory (computed via RPA) is shown as a solid line while the resulting executed path (by the TT module) is shown as a sequence of robot positions taken at regular time interval; (b) and (c) present tracking errors in position and orientation, which are maximal at the place indicated by a dashed ellipse; (d) and (e) present the linear and angular velocities. Curves shown in bold correspond to velocities of the reference trajectory.

The TT–TT transition is activated to toggle between the tracking of two successive reference trajectories (see for example in Fig. 2(d) the transition from $\boldsymbol{P_1}$–$\boldsymbol{P_3}$ to $\boldsymbol{P_3}$–$\boldsymbol{P_8}$). As soon as the robot enters the transition zone, of radius R_T and centered on the junction $\boldsymbol{P_3}$ of the two trajectories, it will leave the tracking of the current trajectory to join the next one.

The RN–RN transition concerns the switching between two reactive navigation modes within a *same* unknown zone. This special situation may happen when we have prior knowledge that the zone is split into two distinct parts with a known compulsory common access (see for example, $\boldsymbol{P_9}$ in Fig. 2). When the robot enters the transition zone of radius R_T and centered on $\boldsymbol{P_9}$, and if this sub-goal $\boldsymbol{P_9}$ is visible, then it will be replaced by the next sub-goal.

The last transition concerns the avoidance of unexpected obstacles (AUO) when tracking a reference trajectory. The AUO transition is composed, in fact, of a TT–RN transition to quit the current tracking in order to avoid the obstacle, followed by a RN–TT transition to resume the tracking once the obstacle is circumvented.

As illustrated in Fig. 9, the TT–RN transition is launched at point ***A***, at time t_A, when (within some predefined angle $\Delta\theta_d$) the minimum distance d_{minRO} separating the robot to the obstacle is less than a given tolerance *Tol*. Then, the robot switches to the RN mode with a new sub-goal ***M*** which must be forecasted, at time $t_A + \Delta t$, on the path of the currently-tracked reference trajectory. The value of Δt must be large enough so that ***M*** would end-up being located beyond the unexpected obstacle. It is evaluated as follows: $\Delta t = 2(Tol + R_{AUO})/v(t_A)$, where R_{AUO} is a predefined estimated maximum radius of the unexpected obstacle and $v(t_A)$ is the robot speed at point ***A***. Finally, a standard RN–TT transition will be launched at point ***B*** when the robot enters a transition zone centered on ***M*** and of radius R_T.

Note that if the unexpected obstacle happens to be located near a feature point, then the value of $t_A + \Delta t$ may exceed the total travel time of the currently-tracked reference trajectory. If this feature point is a TT–TT junction then the remaining time will be used to forecast the position of ***M*** on the path of the next reference trajectory. If the feature point is a TT–RN junction then ***M*** will simply coincide with the sub-goal of the next RN phase.

5. Results and discussion

In this section, we present some simulations to test the performance of the proposed hybrid approach. We use *MobileSim* software for simulating *MobileRobots/ActivMedia* platforms and their environments [39]. In this simulator, the updated pose estimate is calculated by projecting the robot along its current basic trajectory (velocities) from the previous pose estimate, plus some static-systematic error on position and orientation [39].

These environments are represented by 2D maps using Mapper3 tools. The use of ARIA, which is a C++ library for all *MobileRobots/ActivMedia* platforms [39], allows to control dynamically the actuators and to receive current operating data sent by the robot. In order to localize the robot and detect obstacles, we use, respectively, an odometer and a laser range finder, both of which are supported by ARIA.

The mobile robot used here is the *Powerbot* robot by Adept *MobileRobots* [39]. The main characteristics of this robot are summarized in the Appendix. We have also regrouped in the Appendix the setting of all the predefined parameters used in the simulations presented in this section.

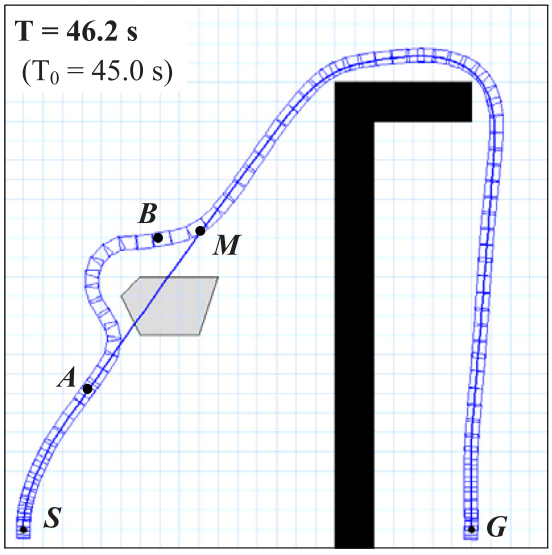
In each of the following examples, we denote by T the total travel time of the task executed from start to goal by the proposed hybrid approach. For comparison, we also give systematically the estimated idealistic lower bound T_0, which is the total travel time of the ideal *single* trajectory obtained for the whole task via RPA, but with *full* knowledge of the environment, which accounts therefore for unknown as well as unexpected obstacles.

5.1. Example 1 (performance of the TT module)

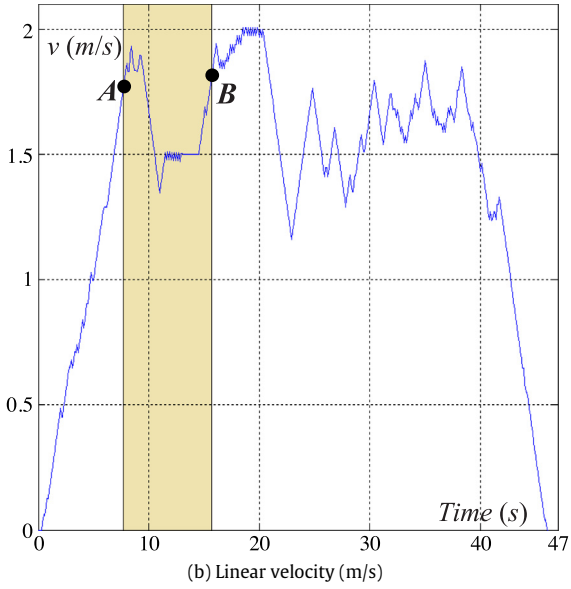
We consider the simplest case of a fully-known (28×28 m^2) environment shown in Fig. 10(a). The purpose is to assess the performance of the TT module. The task is to move the robot, as quickly as possible, from ***S*** [1, 1, 90°] to ***G*** [24, 1, −90°]. We have fixed at 0.3 m/s^2 the maximum linear acceleration allowed for the robot.

The path of the reference trajectory, obtained via RPA, is shown as a solid line in Fig. 10(a) with a score $T_{RPA} = 44.7$ s (which, in this example, coincides with T_0 as the environment is fully known). The execution of this task by the *TT* module is shown in the same figure as a sequence of robot positions taken at regular time intervals. The final score, $T = 44.8$ s, just slightly exceeds T_0.

Fig. 10(b)–(c) show, respectively, the tracking errors in position and in orientation. We note that these errors do not exceed the predefined limits d_T and $\Delta\theta_T$ (Table A.2 of the Appendix). Errors are maximal at the place indicated by a dashed ellipse. Fig. 10(d)–(e) show, respectively, the linear and angular velocities. Curves in bold correspond to those of the reference trajectory and are given there for comparison.



(a) Resulting executed path.

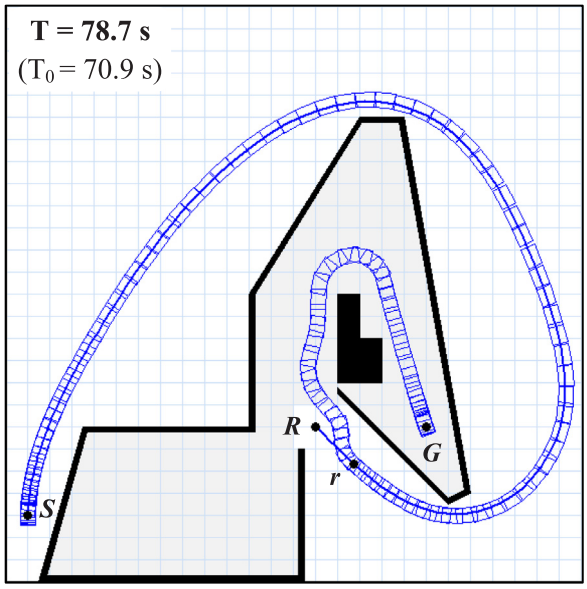


(b) Linear velocity (m/s)

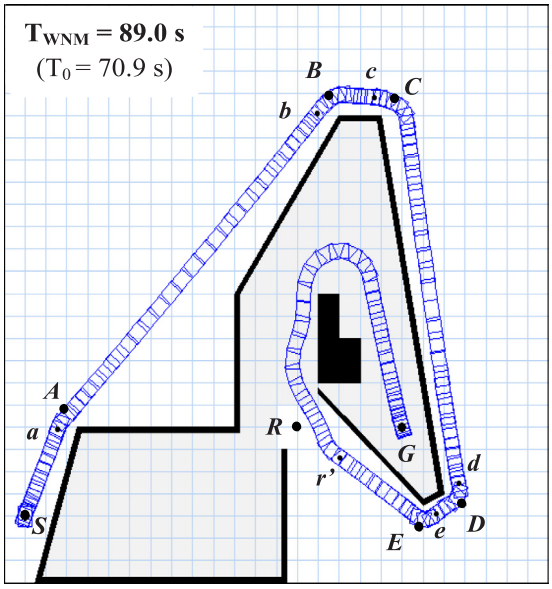
Fig. 11. Results obtained for the example 2.

5.2. Example 2 (avoidance of unexpected obstacles)

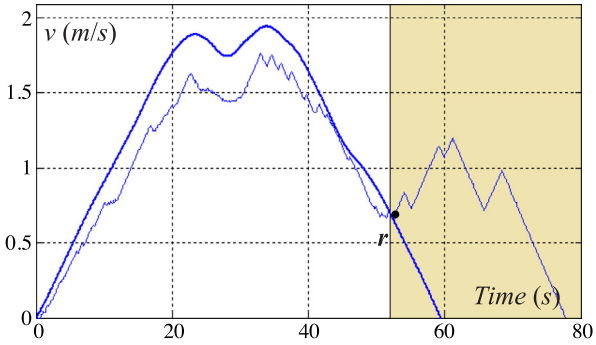
The purpose of this example is to show the ability of the proposed hybrid approach to deal with unpredicted situations when tracking a reference trajectory. The task and maximum acceleration are the same as those of Example 1. The environment is also the same, except that it now includes an unexpected obstacle (Fig. 11(a)), which had not been accounted for when planning the reference trajectory. As shown in this figure, the execution of the task is composed of three successive segments: ***S***–***A***; ***A***–***B*** and ***B***–***G***. The first and third are handled by the *TT* module. The second is handled by the AUO transition (Section 4.5), which computes



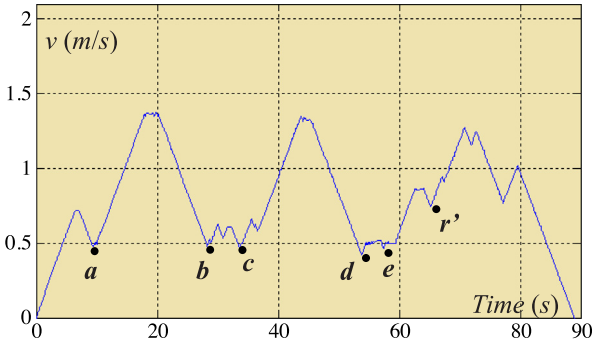
(a) Executed path (proposed approach).



(c) Executed path (waypoint navigation).



(b) Linear velocity (proposed approach).



(d) Linear velocity (waypoint navigation).

Fig. 12. Results obtained for the example 3.

and uses the sub-goal $\boldsymbol{M}$. Fig. 11(b) presents the linear velocity corresponding to these three segments.

As expected, the new execution time is higher ($T = 46.2$ s). However, it remains close to the idealistic score ($T_0 = 45.0$ s) which would have been obtained via RPA had the unexpected obstacle been known beforehand. This result shows the flexibility of the proposed approach to adapt efficiently to some situations not accounted for in the preprocessing phase (trajectory planning).

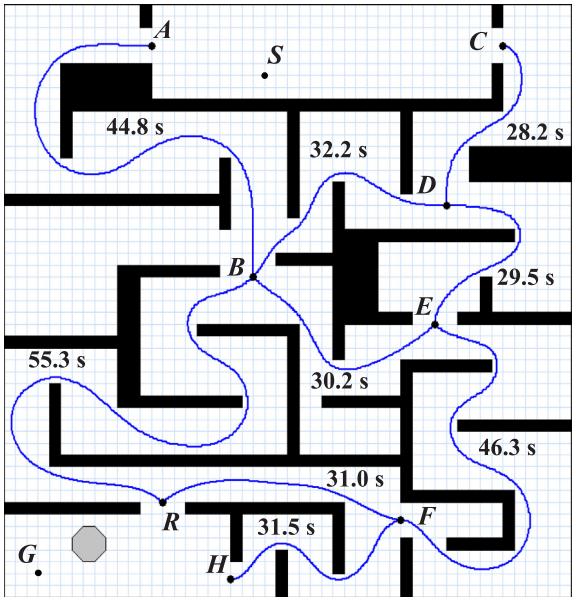
5.3. Example 3 (comparison with a waypoint-navigation method)

In this example we compare the proposed approach with a simple method that combines a reactive navigation with a waypoint path planning. We consider a (26×26 m^2) partially-known environment (Fig. 12(a)). The disaster zone is identified by its access $\boldsymbol{R}$ [14, 7, 135°]. The task is to move from $\boldsymbol{S}$ [1, 3, 90°] to $\boldsymbol{G}$ [24, 7, −75°]. We also impose a more stringent constraint on the maximum linear acceleration allowed for the robot, which is now fixed at 0.1 m/s^2. Here, the idealistic travel time is $T_0 = 70.9$ s.

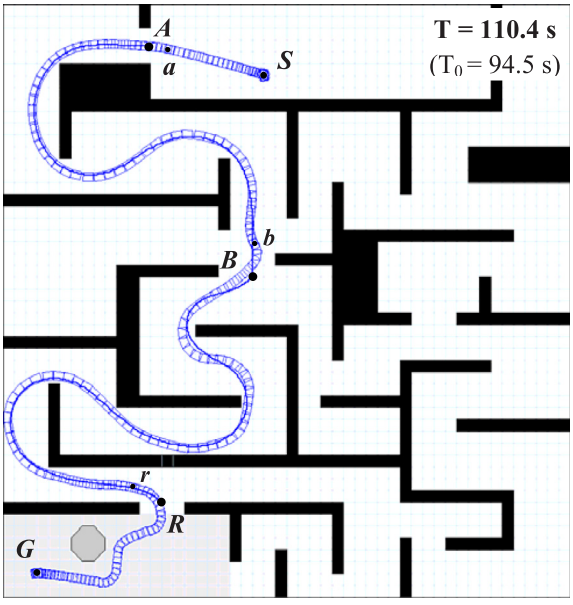
The path of the reference trajectory, obtained via RPA in the safe zone between $\boldsymbol{S}$ and $\boldsymbol{R}$, is shown as a solid line in Fig. 12(a). Its score ($T_{RPA} = 59.5$ s) is given here for completeness, but is irrelevant as this reference trajectory will not be tracked completely. Indeed, the TT module performs the tracking only from $\boldsymbol{S}$ to $\boldsymbol{r}$, at which point a TT–RN transition is launched to join the goal $\boldsymbol{G}$. The execution time to complete the whole task is $T = 78.7$ s. The velocity profile is shown in Fig. 12(b). The curve shown in bold is the velocity profile of the reference trajectory.

The above result can be compared with that of a very simple waypoint-navigation method (WNM). As shown in Fig. 12(c), this method relies on a set of waypoints ($\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{E}$) which can be positioned strategically in the safe zone in order to guide the navigation towards the access $\boldsymbol{R}$ of the disaster zone. Here, the waypoints are placed so as to ensure the shortest path while circumventing known obstacles with a safety radius $R_s = 1.1$ m. The method uses the same adaptive navigation module (Section 4.4) to move the robot, as fast as possible, between two successive sub-goals. It also uses the same transition module (Section 4.5) to switch from one sub-goal to the next. The switching points are labeled ($\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{e}$, and $\boldsymbol{r}'$) in Fig. 12(c). Note that $\boldsymbol{r}'$ does not coincide with $\boldsymbol{r}$ of Fig. 12(a) as the new solution is different. Its score is $T_{WNM} = 89.0$ s and the corresponding velocity profile (Fig. 12(d)) exhibits a trapezoidal shape between successive sub-goals.

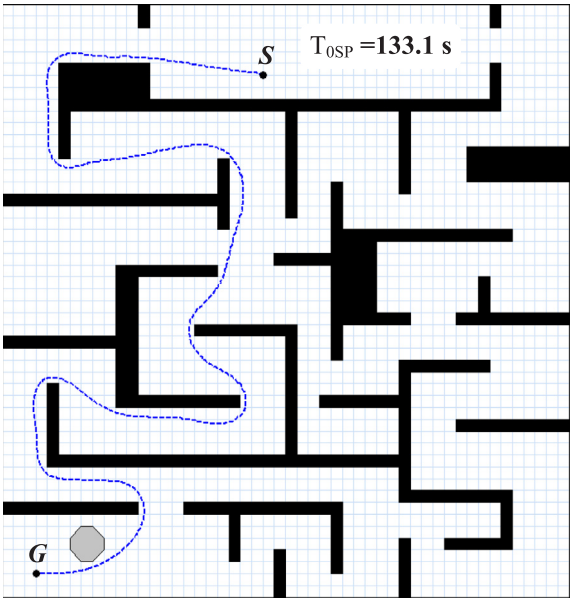
This example shows that, in simple environments, a straightforward method based on a path planning via waypoints, although it will yield a poorer execution time, may remain somewhat competitive considering that it does not require a trajectory-planning preprocessing phase as does the proposed hybrid approach. However, the next example will show that the waypoint-based method can be largely outperformed in more complex environments.



(a) Output graph of the preprocessing phase.



(b) Executed path (hybrid approach).



(c) Optimal-motion planning on the shortest-path.

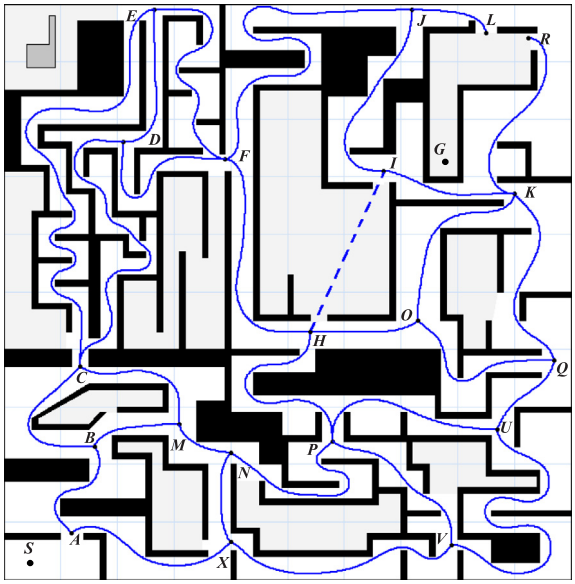
Fig. 13. Results obtained for the example 4.

5.4. Example 4 (case of a more complex environment)

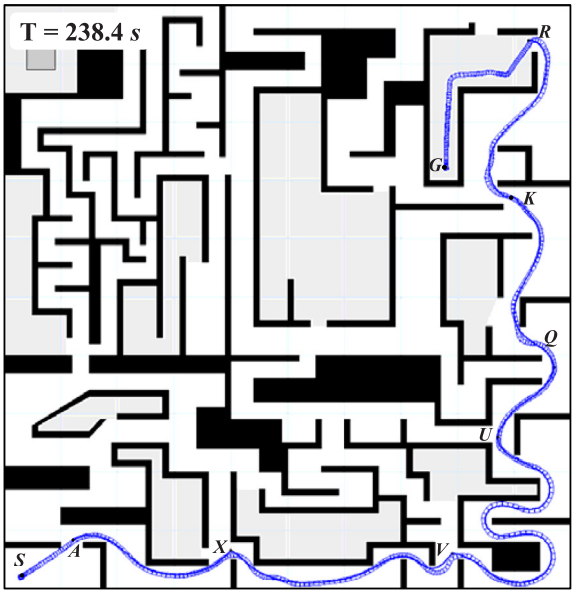
We consider in this example the (50×50 m^2) partially-known environment shown in Fig. 13. The disaster zone is identified by its access $\boldsymbol{R}$ [14, 8, −90°]. The task is defined by $\boldsymbol{S}$ [23, 44, 90°] and $\boldsymbol{G}$ [3, 2, 180°]. The maximum linear acceleration allowed for the robot is set at 0.1 m/s^2. The idealistic travel time to complete the task is $T_0 = 94.5$ s.

Fig. 13(a) shows the output graph of the preprocessing phase. The nine edges of this graph are paths of time-optimal reference trajectories pre-computed via RPA in the safe zones. The travel time of each of these nine trajectories is indicated in the same figure. This graph is augmented by adding the start and goal points ($\boldsymbol{S}$ and $\boldsymbol{G}$) and the least-cost sequence of nodes yielding the shortest travel time is the sequence $\boldsymbol{S}$–$\boldsymbol{A}$–$\boldsymbol{B}$–$\boldsymbol{R}$–$\boldsymbol{G}$. The execution of the proposed hybrid approach along this sequence yields the solution shown in Fig. 13(b). We observe the smoothing of the path by the transition module which automatically avoids the intermediate points $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{R}$ and uses instead the transition points $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{r}$. The score of the resulting solution is $T = 110.4$ s.

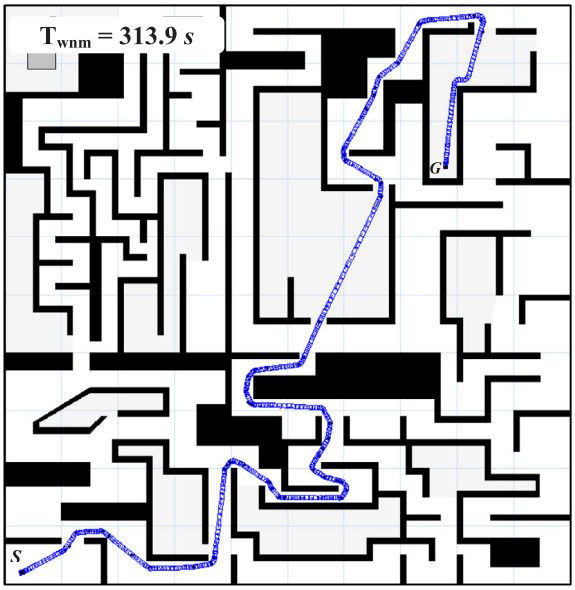
The execution of the same task by the previously-described waypoint-based method, on the shortest path between $\boldsymbol{S}$ and $\boldsymbol{G}$ (computed via the probabilistic road-map method (PRM) [40]), is not shown but would yield a much poorer score, $T_{WNM} = 160.9$ s. We show instead in Fig. 13(c) the *idealistic* solution that would be obtained with *full* knowledge of the environment but via the optimal *motion* planning (computed via RPA) on the *shortest path* between $\boldsymbol{S}$ and $\boldsymbol{G}$. The score T_{0SP} of this idealistic shortest-path-based solution is $T_{0SP} = 133.1$ s. This score represents an approximate lower bound for any method that uses the shortest path instead of the path of the time-optimal trajectory. For instance, the execution of the same task by a method based on VFH [41]



(a) Output graph of the preprocessing phase.



(b) Executed path (Hybrid approach).



(c) Executed path (waypoint navigation).

Fig. 14. Results obtained for the example 5.

associated to a path-tracking algorithm of type *Follow-the-Carrot* [42], on the shortest path between ***S*** and ***G*** yields a score, $T_{VFH} = 140.7$ s.

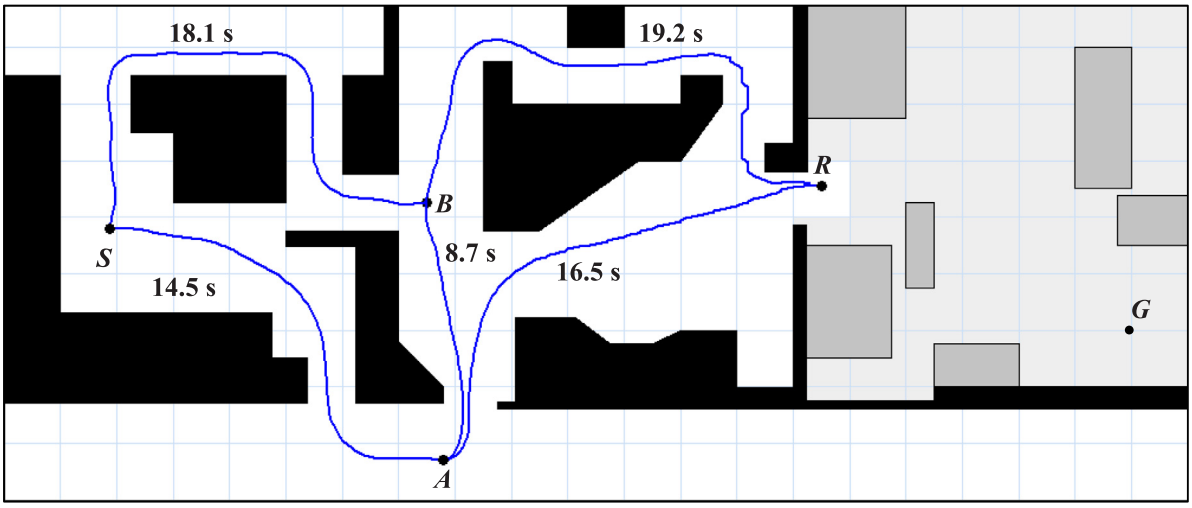
5.5. Example 5 (case of a complex and large environment)

The purpose of this last example is to stress further the important difference between optimal path planning and time-optimal trajectory planning for emergency tasks. We consider the large partially-known environment ($100 \times 100\,\text{m}^2$) shown in Fig. 14. The disaster zone is identified by two distinct accesses ***L*** [85, 96, −90°] and ***R*** [92.5, 94, −180°]. The task is defined by ***S*** [3, 2, 0°] and ***G*** [77.5, 74, −90°]. The maximum linear acceleration allowed for the robot is set at $0.1\ \text{m/s}^2$.

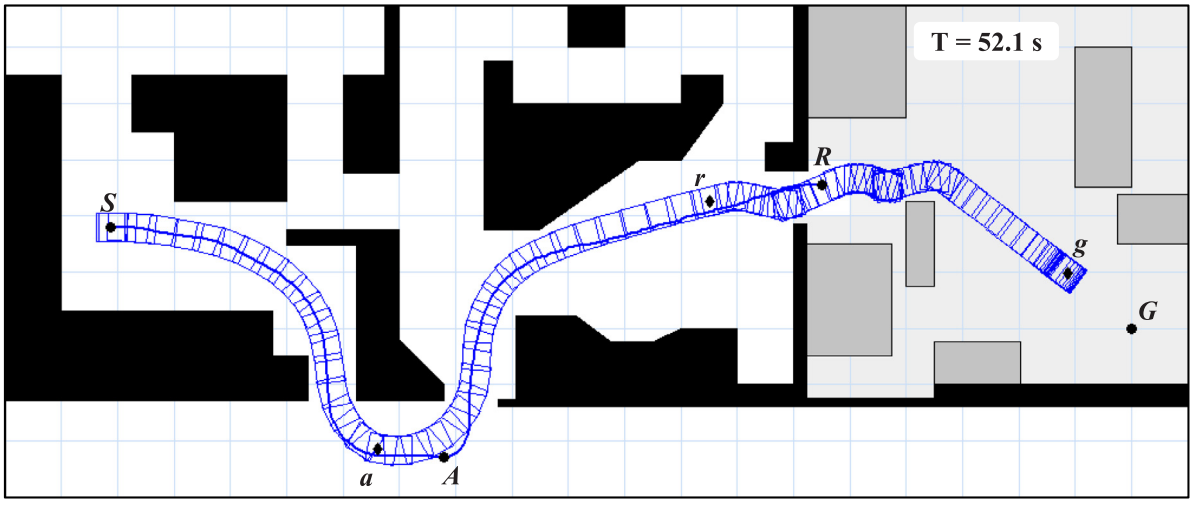
The output of the preprocessing phase is the initial graph presented in Fig. 14(a) which shows the path of thirty pre-computed reference trajectories. Their respective scores are listed in Table 1. The least-cost sequence of nodes from ***S*** to ***G*** is the sequence ***S***–***A***–***X***–***V***–***U***–***Q***–***K***–***R***–***G***. The execution of the task on this sequence by our proposal is presented Fig. 14(b) with a score $T = 238.4$ s.

Here again, we have computed the shortest path between ***S*** and ***G*** using the probabilistic road-map method implemented in Matlab with 2×10^4 nodes and a maximum connection distance equal to 100 m. The execution of the task on this shortest path by the previously-described waypoint-based method is shown in Fig. 14(c) with a score $T_{WNM} = 313.9$ s. We note that this new solution uses a distinct corridor and that its path goes across an unknown zone (segment ***H***–***I***). However, we have assumed a free path between ***H*** and ***I*** in order to avoid penalizing further this solution.

This example demonstrates that distance-based planning and time-based planning can lead to very different solutions and that this difference can be crucial in the case of emergency tasks.



(a) Output graph of the preprocessing phase.



(b) Executed path (Hybrid approach).

Fig. 15. Experimental results.

Table 1
Cost of each edge of the output graph of example 5.

Edge	Score (*s*)	Edge	Score	Edge	Score	Edge	Score	Edge	Score
AB	30.4	CM	32.4	HI	30.5	KO	37.5	OQ	37.8
AX	36.7	DE	33.5	HO	28.7	KQ	39.1	PU	38.3
BC	33.5	DF	37.1	HP	41.7	KR	38.8	PV	36.3
BM	26.7	EF	46.8	IK	31.7	MN	21.9	QU	26.2
CD	65.9	FH	42.5	IJ	39.5	NP	42.8	UV	56.3
CE	87.2	FJ	61.5	JL	25.0	NX	26.7	VX	45.2

5.6. Example 6 (experimental results)

In order to test the proposed approach on a real robot, we consider in this example the ($8.75 \times 21\ m^2$) partially-known environment shown in Fig. 15, which represents a part of the EMP Robotics Laboratory. We use in this example the Pioneer P3-AT produced by *ActivMedia*. It is a four-wheel skid-steering mobile robot with maximum translation and rotation velocities fixed, respectively, at 600 mm/s and 140°/s . Different sensors are embedded to the robot. In our case, we use the SICK laser rangefinder integrated with the robot platform for obstacles detection and the odometer sensor for localization.

The unknown zone is identified by its access $\boldsymbol{R}$ [14.5, 5.5, 0°]. The task is defined by $\boldsymbol{S}$ [1.9, 4.8, 0°] and $\boldsymbol{G}$ [20, 3, −45°]. The maximum linear acceleration allowed for the robot is set at 0.1 m/s^2. The output of the preprocessing phase is the initial graph presented in Fig. 15(a) which shows the path of five pre-computed reference trajectories with their respective score. We note here that obstacles were augmented by a security margin of 0.1 m. In Fig. 15(b) we observe that the estimated least-cost sequence of feature points from $\boldsymbol{S}$ to $\boldsymbol{G}$ is the sequence $\boldsymbol{S}$–$\boldsymbol{A}$–$\boldsymbol{R}$–$\boldsymbol{G}$. The execution of the task by the proposed hybrid approach gives a path that goes through the transition points ($\boldsymbol{a}$ and $\boldsymbol{r}$) and yields a final score $T = 52.1$ s.

This example illustrates the effect of the localization error introduced by the odometer sensor. We observe a difference that exceeds one meter between the requested goal (**G**) and the one

given by the odometer (**g**). This error is due mainly to the severe maneuvers made by the robot to follow the two curved trajectories and to avoid encountered obstacles.

Clearly, the execution of a navigation task strongly depends on the accuracy of the localization system. Failure of the localization module is a crucial problem for any navigation system. In general, when the localization fails, the whole approach fails. A partial solution is to act on the localization module. This problem is not addressed in this work, but several methods are proposed in the literature to improve the accuracy of positioning systems for autonomous navigation of a mobile robot both in indoor and outdoor environment.

5.7. Discussion

We have proposed a hybrid approach for mobile-robot navigation in partially-known environments. This approach is adapted to deal with tasks that require a rapid intervention. Its efficiency strongly depends on the cost-effectiveness of the preprocessing phase which pre-computes near-time-optimal reference trajectories in the safe zones. Clearly, if the known part of the environment is small (relatively to the unknown part) and if its structure is not too complex, then simpler methods based on online path planning (rather than off-line trajectory planning) can be competitive. Thus, the effectiveness of the proposed approach is expected to increase with the size and complexity of the safe zones. In the extreme case of a complex but fully-known environment, our proposal can still be applied without modification and will remain useful as there will be no need for any further trajectory planning besides the ones of the preprocessing phase.

The strongest features of our proposal lie, first, in the use of RPA in the preprocessing phase, which has helped in reducing memory requirements while providing reference trajectories of good quality (near-optimal travel time under constraints on the capacity of the robot and on known obstacles). Second, the transition module has ensured a smooth switching between navigation modes and has helped in improving the execution time by avoiding unnecessary stops. Third, thanks to the flexibility of the TT module based on a virtual robot, the tracking of reference trajectories has not required exactitude while still providing execution times that have remained close to reference times. Fourth, *small* errors in the known parts of the environment, that can be caused by the environment model or the localization system, can be viewed as unexpected obstacles and, as such, can be efficiently handled via the AUO transition, which further adds to the robustness of the proposed approach.

One of the weakest features of our proposal concerns particular complications which can be induced by the presence of unexpected obstacles encountered while tracking a reference trajectory. For example, a poor choice of the direction of avoidance (left or right), which depends on the selected navigation method, may yield a longer than necessary execution time. An extreme case, that can yield a much poorer execution time, is the one in which an unexpected obstacle provokes a local deadlock. In that case, the currently-tracked edge of the graph will have to be deleted. Moreover, the execution phase will need to be re-initialized by replacing $\boldsymbol{S}$ with the current robot configuration and by re-computing a new workable edge succession (if available) to guide the navigation to the goal $\boldsymbol{G}$. Sometimes, such a deadlock can be prevented, *at the preprocessing phase*, by penalizing any edge known to go through a corridor that happens to be too narrow relatively to the robot size and the predefined value of R_{AUO}. However, it is not always straightforward to select the appropriate penalty factor.

Furthermore, the issue of local minima is a very common and well-known problem of *reactive navigation* systems which are used generally in completely *unknown* environment. In the literature several works already provide some solutions to this problem. Hence, more sophisticated state-of-the-art navigation controllers can replace our implementation and can only improve our proposal.

Table A.1
Powerbot parameters.

L_w (m)	L_g (m)	r_w (m)	$v_{\max}$ (m/s)	$w_{\max}$ (rad/s)
0.34	0.90	0.12	2	5.2

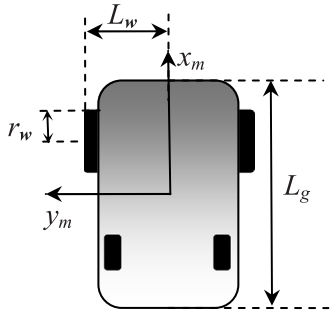


Fig. A.1. Geometric description of the Powerbot mobile robot.

6. Conclusion

We have proposed a hybrid approach, which combines trajectory-tracking and reactive-navigation modes, to deal with the autonomous navigation of a mobile robot in a partially-known environment. Simulation results have shown that our proposal is able to achieve short execution times, which makes it adequate for emergency tasks.

Preliminary experimental results have been presented in an indoor environment in order to test the proposed hybrid approach. However, additional experimental tests still need to be done, both in indoor and outdoor environments using more accurate localization systems. Future work will focus on incorporating and testing more sophisticated modules for reactive navigation and on dealing with dynamic obstacles.

Acknowledgments

The authors would like to thank Professor H.E. Lehtihet for his collaboration, orientations, and valuable help. They also would like to thank Dr. Abdelouahab Bazoula and Dr. Abdelkarim Nemra, from Robotics Laboratory (EMP), for their help in realizing the experimental results.

Appendix

A.1. Characteristics of the powerbot robot

We give here the geometric description of the *Powerbot* mobile robot (Fig. A.1). It is a 2WD differential robot with two idler wheels. The geometric parameters and kinematic limits are listed in Table A.1.

A.2. Setting of the parameter values of the controllers

The following Table A.2 summarizes, for the RN and TT modules, the setting of the predefined parameters (threshold and singletons values) used in the simulations presented in Section 5 of the main text.

The transition module uses the parameter $R_T = 2.2$ m. The AUO transition uses the following predefined parameters: the tolerance

Table A.2
Setting of the predefined parameters of the TT and RN modules.

Modules		Parameters										
		Threshold values				Singleton values						
TT module		Unit	(m)		(°)		(m/s)			(°/S)		
		Label	d_{stop}	d_T	$\Delta\theta_T$	$\Delta\theta_C$	Z_v	S_v	G_v	Z_w	S_w	G_w
		Value	0.05	1.1	3	25	0	1.55	2	0	10	50
RN module	Goal Seeking Sub-module	Unit	(m)		(°)		(m/s)			(°/S)		
		Label	d_{stop}		$\Delta\theta_{GS}$	$\Delta\theta_C$	Z_v	S_v		Z_w	S_w	G_w
		Value	0.1		1	10	0	0.5		0	10	40
	Obstacle avoidance sub-module	Unit	(m)				(m/s)			(°/S)		
		Label	d_{Lmin}	d_{Lmax}	d_{Fmin}		Z_v	S_v	G_v	Z_w	G_w	
		Value	0.9	2.5	1.1		0	0.3	2	0	40	
	fusion sub-module	Unit	(m)				/					
		Label	d_s				Z_β			U_β		
		Value	1.1				0			1		

Table A.3
Modified parameters setting of the RN module adapted to an AUO transition.

Modules		Parameters						
		Threshold values				Singleton values		
RN module	Goal-Seeking Sub-module	Unit				(m/s)		(°/S)
		Label				S_v		G_w
		value	/			0.6		45
	Obstacle-avoidance sub-module	Unit	(m)			(m/s)		(°/S)
		label	d_{Lmin}	d_{Lmax}	d_{Fmin}	S_v	G_v	G_w
		value	1.2	4.8	1.6	1.2	1.5	45
	Fusion sub-module	Unit	(m)					
		label	d_s					
		value	2			/		

$Tol = 3$ m and the detection angle $\Delta\theta_d = 3°$. In addition, it uses a distinct RN module which is specifically adapted to deal with high speeds while tracking a reference trajectory. This RN module uses the same parameters as those previously listed in Table A.2, except for the following modifications listed in Table A.3.

References

[1] L. Adouane, Hybrid and safe control architecture for mobile robot navigation, in: 9th Conference on Autonomous Robot Systems and Competitions, 2009.

[2] D. Nakhaeinia, S.H. Tang, S.M. Noor, O. Motlagh, A review of control architectures for autonomous navigation of mobile robots, Int. J. Phys. Sci. 6 (2) (2011) 169–174. http://dx.doi.org/10.5897/IJPS10.540.

[3] C.P. Lee-Johnson, P. Chand, D.A. Carnegie, Applications of a adaptive hierarchical mobile robot navigation system, in: Proc. Australasian Conference on Robotics and Automation, vol. 1, 2007.

[4] R. Huq, G.K. Mann, R.G. Gosine, Mobile robot navigation using motor schema and fuzzy context dependent behavior modulation, Appl. Soft Comput. 8 (1) (2008) 422–436. http://dx.doi.org/10.1016/J.ASOC.2007.02.006.

[5] S.F. Alves, H. Ferasoli Filho, J.M. Rosario, L.K. Rincón, R.A. Yamasaki, Conceptual Bases of Robot Navigation Modeling, Control and Applications, INTECH Open Access Publisher, ISBN: 97 8-953-307-346-0, 2011.

[6] U.A. Sheikh, M. Jamil, Y. Ayaz, A comparison of various robotic control architectures for autonomous navigation of mobile robots, in: International Conference on Robotics and Emerging Allied Technologies in Engineering, 2014, pp. 239–243. http://dx.doi.org/10.1109/iCREATE.2014.6828372.

[7] N. Vuković, Z. Miljković, New hybrid control architecture for intelligent mobile robot navigation in a manufacturing environment, FME Trans. 37 (1) (2009) 9–18.

[8] K.H. Low, W.K. Leow, M.H. Ang Jr., A hybrid mobile robot architecture with integrated planning and control, in: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1, AAMAS'02, 2002, pp. 219–226. http://dx.doi.org/10.1145/544741.544797.

[9] C. Urdiales, E.J. Perez, F. Sandoval, J. Vázquez-Salceda, A hybrid architecture for autonomous navigation in dynamic environments, in: Proc. of the IEEE/WIC International Conference on Intelligent Agent Technology, IAT 2003, Halifax, Canada, 2003, pp. 225–232.

[10] C. Chen, D. Dong, Grey system based reactive navigation of mobile robots using reinforcement learning, Int. J. Innov. Comput., Inf. Control 6 (2) (2010) 789–800.

[11] R. González, F. Rodríguez, J. Sánchez-Hermosilla, J.G. Donaire, Navigation techniques for mobile robots in greenhouses, Appl. Eng. Agric. 25 (2) (2009) 153.

[12] F. Tian, S.R. Ge, H. Zhu, A navigation control strategy with hybrid architecture for rescue robot, in: IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS, vol. 1, 2010, pp. 675–680. http://dx.doi.org/10.1109/ICICISYS.2010.5658730.

[13] Y. Zhu, T. Zhang, J. Song, X. Li, A new hybrid navigation algorithm for mobile robots in environments with incomplete knowledge, Knowl.-Based Syst. 27 (2012) 302–313.

[14] A. Meléndez, O. Castillo, F. Valdez, J. Soria, M. Garcia, Optimal design of the fuzzy navigation system for a mobile robot using evolutionary algorithms, Int. J. Adv. Robot. Syst. 10 (139) (2013). http://dx.doi.org/10.5772/55561.

[15] P. Natthariyth, Fuzzy logic based control of mobile robot navigation: A case study on iRobot Roomba Platform, Sci. Res. Essays 8 (2) (2013) 82–94. http://dx.doi.org/10.5897/SRE12.586.

[16] L.C. Wang, L.S. Yong, M.H. Ang Jr., Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment, in: IEEE International Symposium on Intelligent Control. Proceedings of the 2002, 2002, pp. 821–826. http://dx.doi.org/10.1109/ISIC.2002.1157868.

[17] J. Minguez, L. Montano, Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios, Robot. Auton. Syst. 52 (4) (2005) 290–311.

[18] V.G. Junior, S.P. Parikh, J. Okamoto, Hybrid deliberative/reactive architecture for human–robot interaction, in: Proceedings of the ABCM Symposium Series in Mechatronics, vol. 2, 2006, pp. 563–570.

[19] W. Chung, S. Kim, M. Choi, J. Choi, H. Kim, C.B. Moon, J.B. Song, Safe navigation of a mobile robot considering visibility of environment, IEEE Trans. Ind. Electron. 56 (10) (2009) 3941–3950. http://dx.doi.org/10.1109/TIE.2009.2025293.

[20] Koubaa Anis (Ed.), in: Robot Operating System (ROS):The Complete Reference., vol. 1, Springer, ISBN: 978-3-319-26054-9, 2016. http://dx.doi.org/10.1007/978-3-319-26054-9.

[21] M. Suchi, M. Bader, M. Vincze, Meta-Heuristic search strategies for Local Path-Planning to find collision free trajectories, in: Proceedings of the Austrian Robotics Workshop, ARW-14, Linz, Austria, 2014, pp. 36–41.

[22] http://www.ros.org/.

[23] B.P. Gerkey, K. Konolige, Planning and control in unstructured terrain, in Workshop on Path Planning on Costmaps, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 2008.
[24] D. Fox, W. Burgard, S. Thrun, The dynamic window approach to collision avoidance, IEEE Robot. Autom. Mag. 4 (1) (1997) 23–33.
[25] E. Garcia, P.G. de Santos, Hybrid deliberative/reactive control of a scanning system for landmine detection, Robot. Auton. Syst. 55 (6) (2007) 490–497. http://dx.doi.org/10.1016/j.robot.2006.12.001.
[26] R.L. Williams, J. Wu, Dynamic obstacle avoidance for an omnidirectional mobile robot, J. Robot. (2010). http://dx.doi.org/10.1155/2010/901365. Article ID 901365.
[27] J. Castro, V. Santos, M.I. Ribeiro, A multi-loop robust navigation architecture for mobile robots, in: IEEE International Conference on Robotics and Automation, vol. 2, 1998, pp. 970–975. http://dx.doi.org/10.1109/ROBOT.1998.677213.
[28] V.M. Santos, J.P. Castro, M.I. Ribeiro, A nested-loop architecture for mobile robot navigation, Int. J. Robot. Res. 19 (12) (2000) 1218–1235. http://dx.doi.org/10.1177/02783640022068048.
[29] M. Mouad, L. Adouane, D. Khadraoui, P. Martinet, Mobile robot navigation and obstacles avoidance based on planning and re-planning algorithm, in: 10th International IFAC Symposium on Robot Control, SYROCO, vol. 12, 2012 http://dx.doi.org/10.3182/20120905-3-HR-2030.00170.
[30] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1) (1959) 269–271. http://dx.doi.org/10.1007/BF01386390.
[31] M. Haddad, T. Chettibi, S. Hanchi, H.E. Lehtihet, A random-profile approach for trajectory planning of wheeled mobile robots, Eur. J. Mech. A Solids 26 (3) (2007) 519–540. http://dx.doi.org/10.1016/J.EUROMECHSOL.2006.10.001.
[32] M. Haddad, S. Hanchi, H.E. Lehtihet, Point-to-point trajectory planning of wheeled mobile manipulators with stability constraint. Extension of the random-profile approach, Eur. J. Mech. A Solids 28 (3) (2009) 477–493. http://dx.doi.org/10.1016/J.EUROMECHSOL.2008.06.008.
[33] M. Egerstedt, X. Hu, A. Stotsky, Control of mobile platforms using a virtual vehicle approach, IEEE Trans. Automat. Control 46 (11) (2001) 1777–1782. http://dx.doi.org/10.1109/9.964690.
[34] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cybern. 1 (1985) 116–132. http://dx.doi.org/10.1109/TSMC.1985.6313399.
[35] R.C. Arkin, Behavior-Based Robotics, MIT press, ISBN: 9780262011655, 1998.
[36] C. Ye, D. Wang, Behavior learning and behavior fusion in autonomous navigation of mobile robots, in: John X. Liu (Ed.), Book Chapter (Invited), Mobile Robots: New Research, Nova Science Publishers, 2005 ISBN: 1-59454.
[37] A. Souici, H. Rezine, Learning goal seeking and obstacle avoidance using the FQL algorithm, in: Proc. World Cong. on Engineering and Computer Science, 2007.
[38] C. Ye, D. Wang, A novel navigation method for autonomous mobile vehicles, J. Intell. Robot. Syst. 32 (4) (2001) 361–388. http://dx.doi.org/10.1023/A:1014224418743.
[39] MobileRobots *Powerbot*, 2015. URL: http://www.mobilerobots.com/.
[40] L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Autom. 12 (4) (1996) 566–580. http://dx.doi.org/10.1109/70.508439.
[41] I. Ulrich, J. Borenstein, VFH: Reliable obstacle avoidance for fast mobile robots, in: Proceedings. IEEE International Conference on Robotics and Automation, 1998, pp. 1572–1577.
[42] M.J. Barton, Controller Development and Implementation for Path Planning and Following in an Autonomous Urban Vehicle (Doctoral dissertation) The University of Sydney, 2001.

Madjid Hank received his engineering and M.S. degrees, respectively, in 2005 and 2009, from the *École Militaire Polytechnique* (EMP) at Algiers, Algeria. Since 2009, he has been a researcher with the department of automatic at EMP. His research interests include embedded positioning systems, control architectures and navigation systems for mobile robots.

Moussa Haddad received his engineering degree in 1996 from the *Institut National de Génie Mécanique* (INGM), Boumerdès, Algeria and his M.S. and Ph.D. degrees, respectively, in 1999 and 2008 from the *École Militaire Polytechnique* (EMP), Algiers, Algeria. He obtained his Academic Habilitation in 2011 from the *École Nationale Polytechnique* (ENP), Algiers, Algeria. He is currently a Professor and a research scientist at the department of applied mechanics at EMP. His research interests include trajectory planning for robotized systems, modeling and optimal design of mechanical systems.