

# Navigation Architecture for Mobile Robots with Temporal Stabilization of Movements

Jorge Silva, Cristina Santos and João Sequeira

**Abstract**—Path modulation and generation are classical issues in navigation architectures for autonomous mobile robots. However, a relevant issue arises in the path planning problem if temporal stabilization of the robot's movement is considered, *i.e.*, all movements of the robot should be compensated when disturbances accelerate or decelerate the robot so that the total mission time reaches a target value.

This work extends previous work by the authors on navigation architectures by including a global path planning level into the architecture.

Simulations using the Webots software demonstrate the ability of the architecture to generate paths with temporal stabilization while avoiding obstacles detected in the environment. The experiments suggest that this extension preserves good formal properties such as stability which can be identified with the ability of the control architecture to drive the robot successfully to the goal.

## I. INTRODUCTION

Path modulation and generation are two fundamental issues in the path planning problem for autonomous mobile robots. Also a relevant issue related to path planning is the temporal stabilization of the robot's movements. In dynamic environments where trajectories are evolving in real time, this is not a trivial task.

Temporal stabilization means being in accordance with the planned movement time, despite varying environmental conditions or perturbations. All movements of the robot should be compensated when disturbances accelerate or decelerate the movement. Temporal stabilization is important for timed action sequences, in which subsequent actions must be initiated and terminated after previous actions have terminated, at specific time instants, or even in human-robot interaction scenarios since humans exhibit strong temporal stabilization in their movements.

To address the problem of temporal stabilization applied to autonomous navigation, the control architecture should include a set of different models, namely, world representation, global planning, local planning and the corresponding timing model.

Several works have been focused on the generation of timed movements for mobile robots [1], [2], [3], [4]. The local nature of these architectures can make the robot fall into dead-end situations or cyclic motions, making it unable to finish the mission.

This work extends the architecture proposed in [4] with a global path planning layer that solves the aforementioned local planning problems. In addition, stability and performance of the architecture are verified following the formal results in [4].

Section II presents a brief review of literature related to navigation architectures and generation of temporal movements for mobile robots. Section III details the subsystems of the architecture. The simulations in section IV show the generation of trajectories under temporal stabilization. Section V concludes the paper with a brief remark about the evolution of this architecture.

## II. RELATED WORK

In general, architectures addressing autonomous mobile navigation involve hybrid path planning, wherein a combination of local and global navigation is required, in addition to an associated module responsible for modeling the environment [5].

Architectures focused on robustness, flexibility, and reliability, making robots navigating in cluttered environments while avoiding obstacles have long been studied. See for instance [6], [7], [8], [9].

A topological representation for global planning and metric representation for local planning which simplifies the design of the architecture was proposed by [10]. The work in [11] showed that path planning based on topological representations leads to path lengths only a few percent longer than the metric representations. Moreover, these considerations are also similar to those in [10]. In [12], metric and topological planning alternate according to the precision needed.

However, while such works have focused on the integration between local and global path planning, temporal stabilization of the robot's movements has not been attempted or considered as a requirement for the success of the mission.

Typical approaches on temporal stabilization of the robot's movements are based on nonlinear dynamical systems and cover several scopes, [13], [14], [15], [16], [17]. The framework in [18] on mobile navigation extended the attractor dynamics approach of behavior generation to the timing domain. This framework was used in a mobile robot to generate temporal coordinated movements [1]. However, [2] argued that the temporal stabilization mechanism of previous works do not guarantee invariant movement time and proposed several changes to improve the framework. Controlling the robot's velocity to achieve temporal stabilization of movements has suffered some novel adaptations when [3] proposed to explore the intrinsic properties of the Landau-Stuart oscillator, by exploring the bifurcation theory to switch the qualitative dynamics of the oscillator, instead of switching among different dynamical systems.

Previous work by the authors (see [4]) focused on the formal analysis of the architecture able of local path planning namely, by identifying stability with mission success. In this work the global planner extension is empirically verified to still hold the good properties that ensure mission success.

### III. DYNAMICAL SYSTEMS BASED ARCHITECTURE

A block diagram (see fig. 1) can be used to represent the global system of this work, through three main blocks each one representing a different agent on the system:  $f_{\text{robot}}$  stands for the robot,  $f_{\text{supervisor}}$  stands for a system that controls the robot, and the block  $K$  represents a mapping of the output of  $f_{\text{robot}}$  into the input of  $f_{\text{supervisor}}$ .

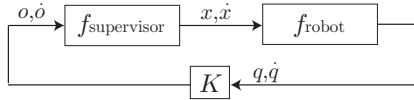


Fig. 1. A graphical representation of the global system.

Hereafter, we assume that  $f_{\text{supervisor}}$  and  $f_{\text{robot}}$  can be described by dynamical systems of the form,

$$\dot{x} = f_{\text{supervisor}}(x, P_g, o), \quad (1)$$

$$\dot{q} = f_{\text{robot}}(q, x) \quad (2)$$

$$o = K(q, \dot{q}) \quad (3)$$

where the righthand sides of these equations are  $C^1$  and  $P_g$  stands for the goal mission.

In this section we describe the overall architecture of our system (fig. 2). It is hierarchically divided into three subsystems of control: global, local and timing. Each subsystem contains several blocks working in parallel and exchanging information among them.

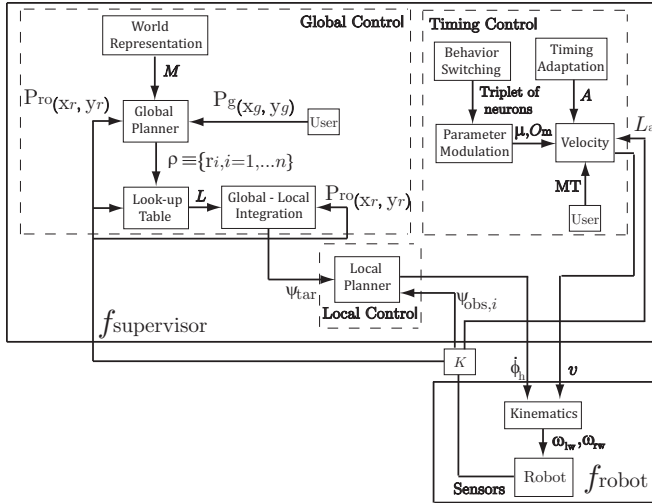


Fig. 2. Schematic of the overall architecture.

The architecture receives as external parameters (sent by the user) the information about the mission, specifically the goal location  $P_g(x_g, y_g)$  and the time frame  $MT$  assigned

to the respective mission. Also, the information about the environment topology, encoded in an incidence matrix  $M$ , is provided *a priori*, and the robot provides the sensorial information  $\psi_{\text{obs},i}$  and  $L_a$  at each time step to the architecture.

The Global Control provides the direction  $\psi_{\text{tar}}$  that the robot should follow during a mission to reach the goal location  $P_g$ . The Local Control merges the direction  $\psi_{\text{tar}}$  and the sensorial information  $\psi_{\text{obs},i}$  in order to provide the angular velocity for the robot  $\dot{\phi}_h$ . The Timing Control generates the adequate linear velocity  $v$  which allows the robot to complete its mission in the time frame,  $MT$ .

The kinematics module receives from the architecture the angular velocity  $\dot{\phi}_h$  and the linear velocity  $v$ , and converts them into the angular velocities for each robot's wheel according to,

$$\omega_{lw} = \frac{1}{R_{\text{wheel}}} \left( v - \dot{\phi}_h \left( \frac{D_{\text{wheel}}}{2} \right) \right), \quad (4)$$

$$\omega_{rw} = \frac{1}{R_{\text{wheel}}} \left( v + \dot{\phi}_h \left( \frac{D_{\text{wheel}}}{2} \right) \right), \quad (5)$$

where  $\omega_{lw}$  and  $\omega_{rw}$  are respectively the angular velocity for the left and right wheels.  $R_{\text{wheel}}$  is the radius of the robot's wheels and  $D_{\text{wheel}}$  is the distance between the two robot's wheels. Note that the architecture is independent of the robot's configuration, however, the robot is fundamental to the mission success [4].

#### A. Global Control

The Global Control consists of four blocks: World Representation, Global Planner, Look-up Table and Global-Local Integration.

In the World Representation block, a common robotics strategy used to represent the environment through a topological map composed by a set of regions, where each corridor and each room of the environment are identified with a region  $r_i$  (see [8]). More generally, the regions  $r_i$  can result from a tessellation procedure on the environment. The transition function  $M$ , useful for practical navigation needs, defines the neighboring relations.

The Global Planner block finds a sequence of regions,  $\rho$ , connecting the perceived position of the robot  $P_{ro}(x_r, y_r)$  to the goal location  $P_g$ . In practical terms, the position of the robot  $P_{ro}$  is perceived with a maximum error  $\tau$  and it can be represented as a neighborhood of the robot's real position  $P_r$  with radius  $\tau$ . Note that we consider that the robot reaches  $P_g$  when it is within the neighborhood of the goal  $\mathcal{B}(P_g, \varepsilon + \tau)$ , where  $\varepsilon$  defines a safety distance.

The sequence of regions  $\rho \equiv \{r_i, i = 1, \dots, n_a\}$  connecting  $P_{ro}$  to  $P_g$ , is found by selecting the path with the minimum cost, where the minimization is achieved by using Dijkstra's algorithm and  $n_a$  is the number of regions  $r_i$  in the sequence  $\rho$ .

The Look-up Table block is a feedthrough map that ensures the correct transition between the regions in  $\rho$  by providing the line segments  $L$ , which defines the border lines between any two neighbor regions in  $\rho$ . The selected line segment  $L$  contains the local point  $P_i(x_i, y_i)$  that the

robot has to track in order to move from a region  $r_i$  to the next region  $r_{i+1}$  in the optimal sequence  $\rho$ . Note that each pair of neighbor regions has an associated border line  $l_{i,i+1}$ , whose extremities are the points  $P_{1i}(x_{1i}, y_{1i})$  and  $P_{2i}(x_{2i}, y_{2i})$  obtained from the environment.

The Global-Local Integration block receives the line segment  $L$  and calculates the local goal  $P_i(x_i, y_i)$  at each instant of time through a projection of  $P_{ro}$  onto  $L$ . When the robot traverses regions, abrupt changes in parameters  $P_{1i}, P_{2i}$  will occur, and without losing generality one may replace  $P_i(x_i, y_i)$  by two smooth functions interpolating a path between  $P_{1i}(k), k = 0, \dots, n-1$  and  $P_{2i}(k), k = 0, \dots, n-1$ , and hence the overall map that generates point  $P_i$  is  $C^1$ . Then, point  $P_i(x_i, y_i)$  is the solution of the following dynamical system,

$$\dot{x}_i = \lambda_{tr}(x_i - x_{1i} + u(x_{2i} - x_{1i})), \quad (6)$$

$$\dot{y}_i = \lambda_{tr}(y_i - y_{1i} + u(y_{2i} - y_{1i})), \quad (7)$$

where  $u$  is the distance between  $P_i$  and  $P_{ro}$  and  $\lambda_{tr}$  defines the relaxation rate of the dynamical system. The local goal  $P_i$  is then used to calculate the direction  $\psi_{tar}$  (rad), that the robot should follow, in order to move across the border line between regions,

$$\psi_{tar} = \arctan\left(\frac{y_i - y_r}{x_i - x_r}\right). \quad (8)$$

### B. Local Control

The Local Control consists on a single block responsible for ruling the robot's heading direction  $\phi_h$  by setting the angular velocity  $\omega_r = \dot{\phi}_h$  through a nonlinear dynamical system (see [19]) defined as,

$$\begin{aligned} \dot{\phi}_h &= \sum_{i=1}^{n_s} \left( -\lambda_{obs,i}(\phi_h - \psi_{obs,i}) \exp\left[-\frac{(\phi_h - \psi_i)^2}{2\sigma_i^2}\right] \right) \\ &- \lambda_{tar} \sin(\phi_h - \psi_{tar}), \end{aligned} \quad (9)$$

where

$$\sigma_i = \arctan\left(\tan\left(\frac{\Delta\theta}{2}\right) + \frac{R_{robot}}{R_{robot} + d_i}\right). \quad (10)$$

where  $\Delta\theta$  is the sensor angular resolution,  $R_{robot}$  is the radius of the robot,  $d_i$  is the distance to obstacles detected by sensor  $i$ ,  $n_s$  is the number of sensor readings,  $\lambda_{tar}$  and  $\lambda_{obs,i}$  define the strength of attraction and repulsion respectively. To detect obstacles in the environment, we used a range finder laser with an angular range of  $240^\circ$  and an angular resolution of  $0.352^\circ$ , resulting in  $n_s = 682$ .

We motivate the choice of this local planner, because its properties allow for an integration of sensory-motor feedback, providing a closed-loop control.

A potential function (see [19]) obtained by integrating the obstacle force-lets verifies if the presence of obstacles is sufficient to change the robot's heading direction  $\phi_h$ ,

$$U(\phi_h) = \sum_{i=1}^{n_s} \left( -\lambda_{obs,i} \sigma_i^2 \exp\left[-\frac{(\phi_h - \psi_i)^2}{\sigma_i^2}\right] - \frac{\lambda_{obs,i} \sigma_i^2}{\sqrt{e}} \right). \quad (11)$$

This function is then used to reduce the amplitude of the oscillator,  $A$ , when obstacles strongly obstruct the path of the robot. If  $U(\phi_h) = 0$ , there are no obstacles in the surroundings of the robot that make the robot to change its direction. If  $U(\phi_h) < 0$ , the robot's heading direction  $\phi_h$  is changed in order to circumnavigate obstacles in the path.

### C. Timing Control

The Timing Control is responsible for generating the velocity of the robot  $v$ , fundamental to the success of the mission. This module is composed by four blocks: Timing Adaptation, Behavior Switching, Parameter Modulation and Velocity.

Velocity block sets in a straightforward manner the velocity  $v$  as the solution  $m$  generated by the Landau-Stuart oscillator (see [1] [2]). This oscillator generates a single oscillation cycle, adapted according to external conditions in order to accelerate or decelerate the robot,

$$\dot{m} = \alpha(\mu - (m - O_m))(m - O_m) - \omega n, \quad (12)$$

$$\dot{n} = \alpha(\mu - (m - O_m))n + \omega(m - O_m), \quad (13)$$

where  $m$  and  $n$  are the state variables,  $\omega$  specifies the frequency of oscillations,  $O_m$  controls the  $m$  solution offset and  $\mu$  encodes the amplitude of the oscillations (see [3] for details). We motivate the choice of this oscillator, because it enables to explicitly modulate the generated movements, according to small parameter changes while keeping the general features of the original movements.

The Timing Adaptation block calculates at each time step the amplitude  $A$ , and the frequency  $\omega$  of the Landau-Stuart oscillator based on the remaining distance that the robot has to cover,  $D$ , to reach  $P_g$ . The robotic mission is subdivided into three time intervals,  $(T_1, T_2, T_3)$  such their sum equals the movement time  $MT$  of the mission. The frequency of oscillation is defined for each time interval as  $\omega_1 = \frac{\pi}{2T_1}$ ,  $\omega_2 = \frac{\pi}{T_2}$  and  $\omega_3 = \frac{\pi}{2T_3}$ . The amplitude of the oscillator is also subdivided as,

$$A_1 = \frac{D}{\frac{\frac{\pi}{2}-1+\sin(\omega_1 t)}{\omega_1} + \frac{\pi+2}{\omega_2} + \frac{\frac{\pi}{2}-1}{\omega_3} - t}, \quad 0 < t \leq t_1, \quad (14)$$

$$A_2 = \frac{D}{\frac{\frac{\pi}{2}}{\omega_1} + \frac{\pi+1+\cos(\omega_2(t-T_1))}{\omega_2} + \frac{\frac{\pi}{2}-1}{\omega_3} - t}, \quad t_1 < t \leq t_2, \quad (15)$$

$$A_3 = \frac{D}{\frac{\frac{\pi}{2}}{\omega_1} + \frac{\pi}{\omega_2} + \frac{\frac{\pi}{2}-\cos(\omega_3(t-T_1-T_2))}{\omega_3} - t}, \quad t_2 < t \leq t_3. \quad (16)$$

The velocity profile is modulated in amplitude and frequency by simply changing both  $A$  and  $\omega$  parameters respectively according to the current state as follows,

$$\begin{aligned} A'(\omega) &= \frac{A_1(\omega_1)}{(1 + e^{b(m-O_m)})(1 + e^{bn})} \\ &+ \frac{A_2(\omega_2)}{1 + e^{-b(m-O_m)}} + \frac{A_3(\omega_3)}{(1 + e^{b(m-O_m)})(1 + e^{-bn})}, \end{aligned} \quad (17)$$

where the value of  $b$  controls the alternation speed between these values ( $b = 500$ ).

If the robot is in a repulsion area where obstacles are obstructing the path of the robot, its velocity  $v$  should be reduced to ensure a safe circumnavigation by adapting the amplitude of the oscillator,  $A$ ,

$$A = A'(-aU(\phi_h)^2 + 1), \quad (18)$$

where parameter  $a$  controls the influence of the obstacle repulsion in the robot's velocity.

#### IV. SIMULATIONS

This section presents a set of simulations in an environment using a Pioneer 3-DX robot with on board laser range finder in Webots simulator.

The simulations were designed to mimic a realistic environment. Noise in the robot's dynamics include both slippage and encoders noise, and an error component of  $\pm 25\%$  is added to the command for each simulation step. The noise is different for each wheel and has a uniform distribution. Laser range finder has an error of 3% of its real value.

During the missions, the robot knows *a priori* the map through  $M$  and only the goal position,  $P_g$ , and the time frame of the mission,  $MT$ , are provided to the robot. The uncertainty on the robot's localization is assumed as  $\tau = 0.5m$ , and  $\varepsilon = 0.3m$  is a sufficient distance that allows the robot to safely stop when it reaches the goal location. When the estimate of the error of the robot's localization reaches  $\tau$ , a reset on the robot's odometry is realized. At each sensorial cycle, information is acquired, dynamic equations are integrated using an Euler method with a time step of 0.1s. The robot is able to reach a maximum velocity of approximately  $1.2ms^{-1}$ .

##### A. Simulation 1

Fig. 3 shows a snapshot of the hospital environment generated by the mobile robot using the onboard laser range finder. The initial position of the robot is marked by a red circle while the goal position  $P_g$  is indicated by a red cross. The distance between both positions is approximately 28m. Red dashed lines indicate the border lines  $L$  and blue continuous line depicts the path followed by the robot (odometry values). In this first simulation, the robot must reach the goal location within the specified time frame,  $MT = 80s$ .

Fig. 4 depicts a sequence of snapshots obtained during the simulation, where several problematic situations for mobile robots are identified: obstacle avoidance (panel A), passage in a narrow corridor (panel B) and obstacle avoidance of dynamical obstacles (panel C).

Fig. 5 (top) depicts the evolution of the velocity,  $v$  (blue continuous line) obtained through odometry, and the amplitude of the oscillator,  $A$  (green dashed line). At instants of time  $t \approx 19s$ ,  $t \approx 23s$  and  $t \approx 35s$  (vertical dashed black lines), the amplitude  $A$  was decreased to reduce the robot's velocity as a consequence of obstacle detection through (18). Within the time interval,  $43s < t < 47s$ , it is noticeable the increase of the velocity to compensate for the delay provoked by the obstacles circumnavigation. Fig. 5 (bottom) depicts the distance  $D$  between the robot's position  $P_{ro}$  and the goal

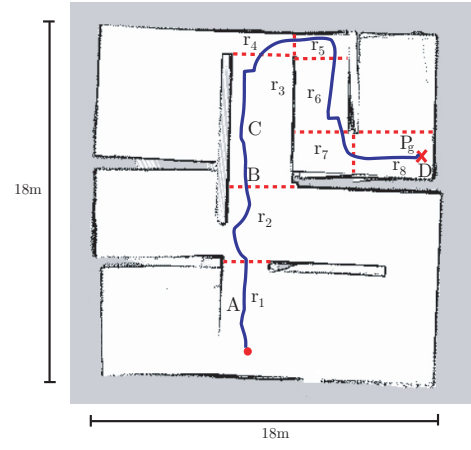


Fig. 3. Visualization of the environment generated through the onboard laser range finder while running the robot across the simulated hospital environment.

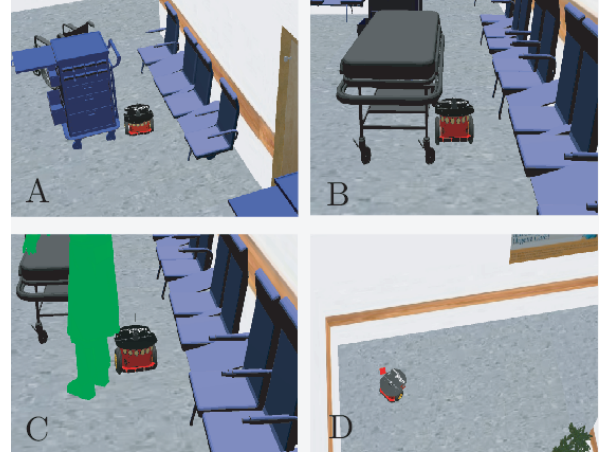


Fig. 4. Snapshots of the simulation, where the robot has to move inside a hospital environment between two different locations.

location  $P_g$ . It is noticeable in the interval,  $43s < t < 47s$ , (shadow blue region), that the robot does not approximate the goal location, since it was circumnavigating obstacles. However, such obstacles are not sufficient to reduce the robot's velocity. At the end of the mission, the distance between the robot and  $P_g$  was approximately 0.4m, which means that the robot was in the neighborhood of the goal location, defined by the radius  $\tau + \varepsilon$ . The robot successfully reached  $P_g$  within the specified time frame for this mission,  $MT = 80s$ .

In fig. 6 (top) the potential function  $U(\phi_h)$  indicates the presence of obstacles when  $U(\phi_h) < 0$ . Fig. 6 (bottom) illustrates the direction  $\psi_{tar}$  (red dashed line) that the robot should follow to reach  $P_g$ , and the robot's heading direction,  $\phi_h$ , calculated through odometry. As expected when  $U(\phi_h)$  is negative, the robot changes its direction to avoid obstacles. Note for instance during the interval,  $19s < t < 25s$ , identified by vertical dashed lines, that there are two negative values of  $U(\phi_h)$  created by obstacles, that forces the robot to change

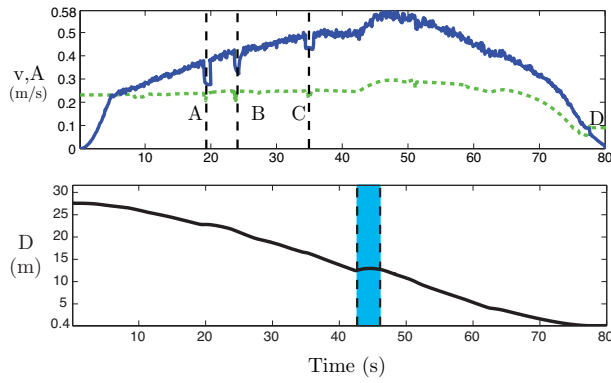


Fig. 5. Top) Velocity performed by the robot,  $v$ , (continuous blue line) and amplitude of the oscillator,  $A$ , (dashed green line). Letters identify the correspondent panels of fig. 4. Bottom) Distance between the robot  $P_{To}$  and the goal position  $P_g$ .

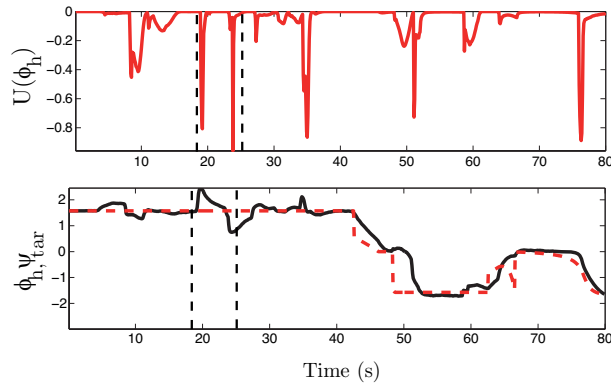


Fig. 6. Top) Potential function  $U(\phi_h)$ . Bottom) Direction followed by the robot,  $\phi_h$ , (continuous black line) and direction that the robot has to follow to reach the goal location,  $\psi_{tar}$ , (dashed red line).

its direction.

The coordinates of the local goal  $P_i(x_i, y_i)$  calculated through (6) and (7) are shown in fig. 7. As expected, the values of  $x_i$  and  $y_i$  change when the robot crosses to another region  $r_i$ . For instance, when the robot is in region  $r_2$ , we verify the adaptation of the value  $y_i$  as a consequence of the obstacles circumnavigation in the interval of time  $19s < t < 25s$ , previously identified in fig. 6 (bottom).

Stability properties are measured using an upper bound in the open loop gain associated to the control architecture (see [4]). This value must be much smaller than 1 in order to ensure stability (and hence mission success). Figure 8 shows the evolution of the stability index along the simulation.

### B. Simulation 2

The second simulation consists in a mission in which the environment presents several populated regions  $r_i$ , where the robot has to circumnavigate several obstacles located in the path, and a high robot's velocity is required to accomplish the timing constraints of the mission. This simulation mimics a mission in a real hospital environment in which the robot moves through crowded regions and regions without perturbations. Regions with many obstacles make the robot to reduce its velocity to ensure safe circumnavigation and

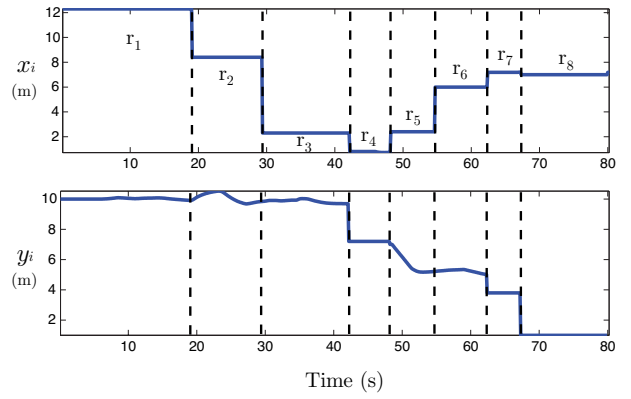


Fig. 7. Coordinates of the local goals  $P_i(x_i, y_i)$  at each instant of time during the mission.

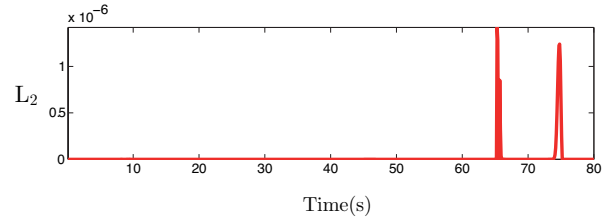


Fig. 8.  $L_2$  gain for the control architecture during simulation 1.

to cover more distance than expected. Herein, the robot's velocity has to increase during the mission to compensate for the delay provoked by the decrease in the velocity and to increase the distance covered.

The mission must be performed within the time frame  $MT = 50s$ . Fig. 9 depicts a sequence of snapshots obtained during the simulation. Note that panels A, B and C correspond to populated regions  $r_i$  while panel D depicts an area without obstacles.

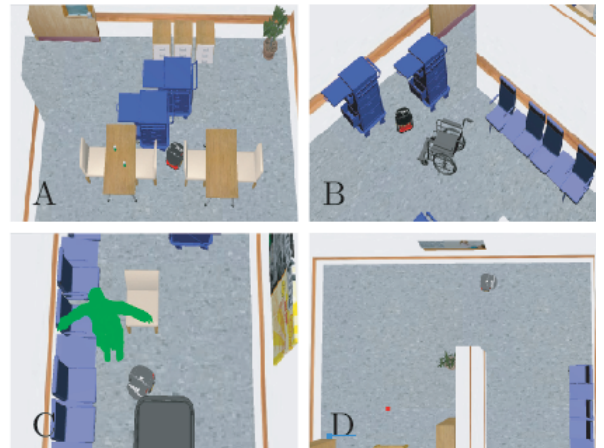


Fig. 9. Snapshots of the simulation, where several regions of the environment are populated with many obstacles.

Fig. 10 (top) depicts the evolution of the velocity  $v$  (blue continuous line) and the amplitude of the oscillator,  $A$  (green dashed line). Note that at several instants of time



the amplitude  $A$  decreases to reduce the robot's velocity. However, in order to finish the mission within the time frame there are intervals of time that the required velocity is higher than the robot's maximum velocity. Thus, in such intervals of time,  $33s < t < 36s$ , and  $45s < t < 47s$  (blue shaded areas), the robot moves at its maximum velocity  $1.2(\text{ms}^{-1})$ .

It is noticeable that the amplitude of the oscillator  $A$ , increases during the mission, except when obstacles force the robot to decrease its velocity, to compensate the delay created by the increase of the path's length. Despite the strong obstacle contribution, the architecture generates the suitable velocity for the robot to reach the goal location  $P_g$  within the time frame  $MT$ .

Fig. 10 (bottom) shows the stability index, also well below 1 during all mission.

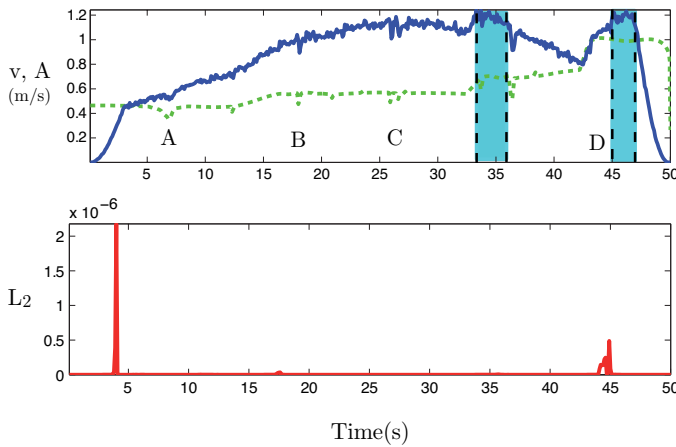


Fig. 10. Top) Velocity performed by the robot (continuous blue line) and amplitude of the oscillator (dashed green line). Bottom)  $L_2$  gain for the control architecture during simulation 3.

## V. CONCLUSIONS

This paper presents an architecture for mobile robots based on a mesh of nonlinear dynamical systems and feedthrough maps able to control mobile robots in missions under timing constraints. The architecture is divided into three subsystems of control: global, local and timing.

The results demonstrate the ability of the robot to follow a collision free path between two locations, in a cluttered and dynamic environment, within the temporal bound specified. Moreover, the evolution of the stability index along the missions clearly suggests that extending the architecture with the global planner level still preserves the good properties of the baseline architecture.

Future work includes the deployment in a real hospital delivery robot whose goal is to move between different locations within a hospital, while keeping a timing constraint on its tasks. In addition, we intent do develop a localization system in order to demonstrate the concepts developed in the paper in more complex missions.

## ACKNOWLEDGMENTS

This work was partially supported by FCT projects PEst-OE/EEI/LA0009/2011 and FCOMP-01-FEDER-0124-022674, and grant SFRH/BD/68805/2010

## REFERENCES

- [1] C. Santos. Generating timed trajectories for an autonomous vehicle: a non-linear dynamical systems approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3741 – 3746, 2004.
- [2] M. Tuma, I. Iossifidis, and G. Schöner. Temporal stabilization of discrete movement in variable environments: An attractor dynamics approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 863 – 868, 2009.
- [3] J. Silva, C. Santos, and V. Matos. Timed trajectory generation for a toy-like wheeled robot. In *36th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pages 1645 – 1650, 2010.
- [4] J. Sequeira, C. Santos, and J. Silva. Dynamical systems in robot control architectures: A building block perspective. In *12th International Conference on Control, Automation, Robotics and Vision, ICARCV*, 2012.
- [5] J. Minguez, L. Montesano, and L. Montano. An architecture for sensor-based navigation in realistic dynamic and troublesome scenarios. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2750 – 2756, 2004.
- [6] K. Low, W. Leow, and M. Ang Jr. A review of control architectures for autonomous navigation of mobile robots. In *International Joint Conference on Autonomous Agents and MultiAgent Systems*, 2002.
- [7] D. Nakhaeinia, S. H. Tang, S.B. Mohd Noor, and O. Motlagh. A review of control architectures for autonomous navigation of mobile robots. In *International Journal of the Physical Sciences*, 6(2):169 – 174, 2011.
- [8] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. In *Artificial Intelligence*, volume 99, pages 21 – 71, 1998.
- [9] A. Ozkil, Z. Fan, J. Xiao, J. Kristensen, S. Dawids, K. Christensen, and H. Aanaes. Practical indoor mobile robot navigation using hybrid maps. In *IEEE International Conference on Mechatronics, Istanbul, Turkey*, 2011.
- [10] K. Konolige, E. Marder-Eppstein, and B. Marthi. Navigation in hybrid metric-topological maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3041 – 3047, 2011.
- [11] Z. Zivkovic, B. Bakker, and B. J. A. Kröse. Hierarchical map building and planning based on graph partitioning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 803 – 809, 2006.
- [12] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Simultaneous localization and map building: a global topological model with local metric maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 421 – 426, 2001.
- [13] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *IEEE International conference on robotics and automation (ICRA)*, pages 763 – 768, 2009.
- [14] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems 15*, pages 1547–1554, 2002.
- [15] K. Hiroshi, Y. Fukuoka, and A. H. Cohen. Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts. *Int. Journal of Robotics Research*, 26(5):475–490, 2007.
- [16] R. Ronsse, N. Vitiello, T. Lenzi, J. van den Kieboom, M. Carrozza, and A. Ijspeert. Human-robot synchrony: flexible assistance using adaptive oscillators. *IEEE Transactions on Biomedical Engineering*, 58:1001 – 1012, 2010.
- [17] C. Santos and V. Matos. Gait transition and modulation in a quadruped robot: A brainstem-like modulation approach. *Robot. Auton. Syst.*, 59:620 – 634, 2011.
- [18] G. Schöner and C. Santos. Control of movement time and sequential action through attractor dynamics: A simulation study demonstrating object interception and coordination. In *9th International Symposium on Intelligent Robotic Systems (SIRS)*, 2001.
- [19] E. Bicho, P. Mallet, and G. Schöner. Target representation on an autonomous vehicle with low-level sensors. *The International Journal of Robotics Research*, (210):424–447, 2000.