Proceedings of The 10th Mediterranean Conference on Control and Automation - MED2002
Lisbon, Portugal, July 9-12, 2002.

# VISUAL-BASED PLANNING AND CONTROL FOR NONHOLONOMIC MOBILE ROBOTS

**A. De Luca, G. Oriolo, L. Paone, P. Robuffo Giordano, M. Vendittelli**

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Eudossiana 18, 00184 Roma, Italy
e-mail: {deluca,oriolo,vendittelli}@dis.uniroma1.it

**Keywords:** Visual feedback, nonholonomic mobile robots, motion planning, nonlinear control

## Abstract

An integrated visual-based approach to motion planning and control of a nonholonomic wheeled mobile robot is presented. Vision is used to localize the obstacles and the robot and for real-time feedback control of the vehicle motion. An $A^*$-based motion planner is used to search the discretized robot configuration space for collision-free paths. Nonholonomic constraints are taken into account by properly defining cell adjacency in the discretized configuration space and in the choice of the heuristic functions. Path planning results for unicycle and car-like kinematics are presented and compared in simulated environments. The whole system is tested with the mobile robot SuperMARIO in an unknown indoor environment. In the experiments, tracking of the planned trajectories is achieved via visual feedback from a fixed camera and using a dynamic linearizing control law.

## 1 Introduction

Mobile robot autonomy requires the integration of sensing, planning, and control capabilities. Typical sensors include rangefinders (infrared and sonar) and more powerful vision systems [5]. Their (combined) use allows to build a map of an unknown environment [18] and/or to localize the mobile robot [2, 9]. Once a geometric description of the environment is available, the motion planner define a feasible collision-free path joining the start and goal robot postures. For wheeled mobile robots, nonholonomic constraints due to the perfect rolling assumption of the wheels on the ground can be taken into account in the planning phase, as well as other physical or task-based constraints such as maximum allowed path curvature [13, 15, 21]. During motion execution, the robot controller has to be designed in order to guarantee path following and regulation to the final configuration [6]. Typically, a combination of feedforward and feedback commands are needed and the availability of a single control law for both motion tasks is highly desirable. In order to overcome the control limitation due to nonholonomy, time-varying [20], discontinuous [1], or dynamic [7] state feedback have been used. Since proprioceptive (i.e., odometric) feedback is sensitive to slipping disturbances, the use of visual information for real-time motion control is becoming more popular [4, 11, 12, 16].

In this paper, we present an integrated approach to motion planning and control of a wheeled mobile robot based on visual information. The 2D indoor environment is completely unknown and a camera mounted on the ceiling is used to determine the position of the obstacles and to localize the robot. This information, together with a given goal posture, is directly processed by a motion planner that generates an optimal nonholonomic and collision-free path, using the $A^*$ algorithm on a discretized robot configuration space as in [17]. The objective function is related to the path length while different heuristics can be used, depending on the robot kinematic constraints.

Collision-checking is efficiently performed in the cartesian space. The mobile robot SuperMARIO used in our experiments is a two-wheel differentially driven vehicle and its kinematics is equivalent to that of a unicycle. However, feasible paths may be generated also taking into account additional curvature constraints, i.e., assuming that the robot cannot rotate on place and thus covering also the case of a vehicle with car-like kinematics [14]. Path smoothing is then performed using clothoids [10] and a timing law is assigned so as to comply with bounds on vehicle velocity and acceleration. The resulting trajectory is passed to the robot controller, which is based on the recently developed dynamic feedback linearization technique [19], with the current robot state measured through the fixed camera as in [8].

The paper is organized as follows. In the next section we briefly describe our experimental setup, including the robot and the vision system, and the visual processing for obstacle localization and robot feedback control. In Sect. 3, we present the nonholonomic motion planner and compare the performance with different heuristics on simulated environments, both for unicycle and car-like kinematics. In Sect. 4, we report on the experimental results with SuperMARIO.

## 2 System setup

The mobile robot SuperMARIO is a two-wheel differentially-driven vehicle (see Fig. 1). The driving wheels have radius $r = 9.93$ cm and their axle is of length $d = 29$ cm; a small passive off-centered wheel (castor) is placed near the vehicle front. Incremental encoders are mounted on the two wheels' motors. SuperMARIO communicates via radiomodem with a 300Mhz PC Pentium II, where a library of C++ control algorithms is installed. The vision system is made by a digital 1/2" camera with $768 \times 576$ pixels, fixed on the laboratory ceiling at an height of 2.9 m, and a Matrox Meteor frame grabber on the PC. The camera output signal (RGB or CCIR) is sent to the frame grabber with a 25 Hz frame rate in CCIR mode. The vision area (i.e., the workspace) dimensions are $2.90 \times 2.10$ m. As a result, 1 pixel $\approx 3.7$ mm. A more detailed description of the whole system is given in [19].

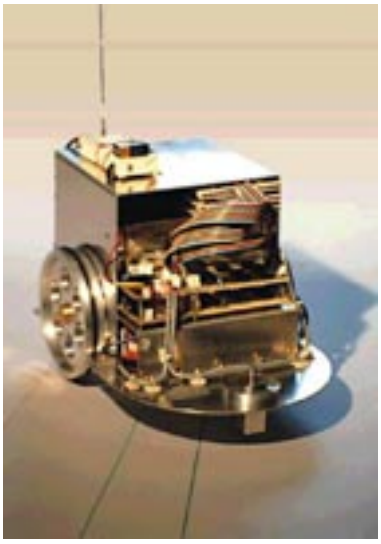

Figure 1: The mobile robot SuperMARIO

### 2.1 Kinematic model

The kinematic model of nonholonomic robot SuperMARIO is

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega, \end{aligned} \tag{1}$$

where the robot reference point $(x, y)$ is the cartesian position of the wheel axle midpoint, $\theta$ is the vehicle orientation w.r.t. the $x$-axis, $v$ and $\omega$ are, respectively, the linear and angular robot velocity. The actual input commands are the angular velocities $(\omega_R, \omega_L)$ of the right and left wheels; these are one-to-one related to $(v, \omega)$ by

$$v = r\,\frac{\omega_R + \omega_L}{2}, \qquad \omega = r\,\frac{\omega_R - \omega_L}{d}. \tag{2}$$

Although SuperMARIO can rotate on place, and is thus kinematically equivalent to a unicycle, one may impose a car-like behavior by introducing a constraint on the inputs. This is related to the maximum curvature value $\kappa$ allowed for paths to be followed by the robot reference point, i.e.,

$$|\omega| \le \kappa\,|v|. \tag{3}$$

The inequality (3) leads to linear constraints also on the actual commands $(\omega_R, \omega_L)$.

### 2.2 Visual feedback

Visual feedback is used for determining the start and current robot posture and to localize the obstacles in the workspace.

For reconstructing the robot posture, we have mounted on SuperMARIO a black surface with three leds, located at the vertices of an isosceles triangle pointing in the forward direction having the upper vertex position in the robot reference point $(x, y)$. In order to localize this triangle in the image, we proceed as follows. A binary image is created first from the original 256-level grayscale image, by using a fixed threshold (set to 240). Using a *dilation* operator, a more significant image is obtained from which a list of light *blobs* (in the range of $30 \div 120$ pixels each) is extracted. Upper and lower bounds on the blob area are used to discard false reflections from the floor and the robot chassis (especially, wheels)[1]. Next, an appropriate algorithm based on relative distances deletes from the list all the blobs that cannot be candidate vertices (within some tolerance), and builds with the remaining blobs all isosceles triangles with consistent side length. Once a single triangle is determined, let $(x_u,\ y_u)$, $(x_l,\ y_l)$, and $(x_r,\ y_r)$ be the coordinates of, respectively, the upper, lower-left, and lower-right vertex. The estimate of the robot reference point is simply $\hat{x} = x_u$, $\hat{y} = y_u$. The center $(x_c,\ y_c)$ of the triangle and its base midpoint $(x_m,\ y_m)$ are then computed from the three vertices. The estimate $\hat{\theta}$ of the robot orientation is finally computed as

$$\hat{\theta} = \frac{\theta_1 + \theta_2 + \theta_3}{3},$$

with

$$\begin{aligned} \theta_1 &= ATAN2\{y_u - y_c, x_u - x_c\} \\ \theta_2 &= ATAN2\{y_u - y_m, x_u - x_m\} \\ \theta_3 &= ATAN2\{y_c - y_m, x_c - x_m\}. \end{aligned}$$

[1]This is done using the MIL Libraries that allow blob detection and fast analysis of basic blob features, such as center of mass, area, etc.

This averaging strategy is able to reduce the effects of image noise.

When using visual feedback during motion execution, the control sampling time is $T_c = 55$ ms, including frame acquisition, elaboration and robot-server communication. The small increase (5 ms) in sampling time with respect to odometric feedback control [8] has been limited by making frame acquisition asynchronous from other control routines and by restricting the above triangle search to a $300 \times 300$ window centered around the previous robot posture estimate. A full window search would have led to 80 ms sampling time.

For detecting the (static) obstacles, the robot is removed from the image by first bounding its 3D shape with a box centered on the robot reference point, then projecting this box onto the cartesian plane, and finally saturating the pixel values in the obtained area. Pixels with gray level below a given threshold (set to 55) are assumed to belong to obstacles.

## 3 Motion planner

The path planning problem consists in finding a path between a start configuration $q_S = (x_S, y_S, \theta_S)$ and a goal $q_G = (x_G, y_G, \theta_G)$, which avoids workspace obstacles (*collision-free*) and respects the nonholonomic constraints of the vehicle (*feasible*).

The workspace is represented by a bitmap with $r \times c$ square cells of side $\delta$. Each cell typically contains about one hundred pixels. The cell is labeled as an obstacle if at least one pixel is an obstacle.

The path planner searches for collision-free paths over the (discretized) configuration space. However, in order to increase efficiency, configuration space obstacles are never built but collision is checked at runtime in the workspace only for candidate robot configurations selected by the planner. For this collision check, SuperMARIO is represented by its bounding rectangle (see Fig. 2).

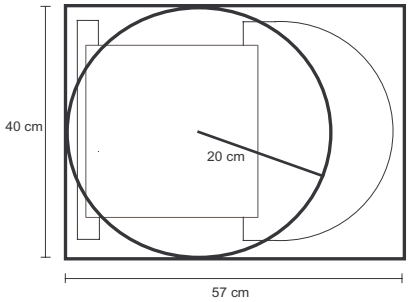


Figure 2: Bounding rectangle and inscribed circle used for SuperMARIO within the motion planner

### 3.1 Cell adjacency

To take into account the nonholonomic constraints, adjacency between cells in the configuration space representation is defined by considering a discrete set of velocity inputs [15]:

$$v = \{\pm v_0, 0\} \quad \omega = \{\pm\omega_0, 0\}. \tag{4}$$

The adjacent successors of a given configuration $q_i$ are generated by integrating the kinematic equations (1) on a unitary time interval, with $v$ and $\omega$ taking values in the finite velocity set (4). In particular, we have chosen $v_0 = \delta$ and $\omega_0 = \pi/8$ to guarantee that the next configuration $q_{i+1}$ will not be farther away than one neighboring cell [17]. Note that, in the case of our differentially-driven robot, it is sufficient to consider combinations of inputs in which either $v$ or $\omega$ is zero, while in order to emulate a car-like behavior $\omega$ is forced to zero if $v = 0$.

A feasible collision-free path $P$ will then be a sequence of adjacent robot configurations $\{q_0, \ldots, q_N\}$ joining $q_S$ with $q_G$ (i.e., $q_0 = q_S$ and $q_N = q_G$) and such that, in these configurations, the robot does not collide with obstacles in the environment.

### 3.2 Search algorithm

Search in the discretized configuration space is performed with the well-known $A^*$ algorithm (see, e.g., [15]), which returns, if it exists, a path minimizing a given cost function $g(P)$. The search is made efficient by the use of an heuristic function $h(q)$ estimating the cost of the optimal path from the configuration $q$ to the goal $q_G$.

#### 3.2.1 Cost function

The cost of a path $P$ is defined as

$$g(P) = \sum_{i=0}^{N-1} w(q_i, q_{i+1}),$$

where $w(q_i, q_{i+1})$ is a measure of the space traveled by the robot wheels. In particular we have set: $w(q_i, q_{i+1}) = v_0$, for linear displacements $(v = \pm v_0, \omega = 0)$; $w(q_i, q_{i+1}) = \omega_0 d/2$, for a reorientation on place $(v = 0, \omega = \pm\omega_0)$; $w(q_i, q_{i+1}) = v_0 + \omega_0 d/2$, for motion along an arc of a circle of radius $v_0/\omega_0 = 1/\kappa$ $(v = \pm v_0, \omega = \pm\omega_0)$. The latter expression is a monotonic function of the space traveled by the robot reference point.

#### 3.2.2 Heuristic function

The simplest heuristic function to be used within our application of $A^*$ is the distance $h_e$ to the goal, i.e., $h_e(q_i)$ is the euclidean distance between $(x_i, y_i)$ and $(x_G, y_G)$. However, this heuristic leads the search toward the goal without taking into account *i)* the nonholonomic nature of the robot, and *ii)* the presence of obstacles.

For the first issue, better results may be obtained by using, in the definition of the heuristic function, the length of the time-optimal trajectories computed in [3] for a differentially-driven robot moving in free space. In fact, when $q$ is sufficiently far from $q_G$, the optimal trajectory is composed by two rotations on place (one at the beginning and the other at the end) and a motion along the linear segment joining the respective cartesian positions. This also minimizes the length of the path traveled by either one of the two wheels [2].

To incorporate in this framework the workspace obstacles, the heuristic function $h_n$ replaces, in the computation of the total length of the path-to-go, the length of the linear segment (euclidean distance) with the value of the numerical *navigation function* obtained by a cartesian wavefront

[2] By considering the cartesian path traced by the wheels, we can also obtain an homogeneous measure of combined angular and linear quantities.

expansion from the goal position $(x_G, y_G)$ to the position $(x, y)$ [15, p. 318].

If $(x, y)$ is closer than about $d$ to the goal position, the sequence of optimal maneuvers in [3] becomes more complex and results in a computational load which does not justify the benefits of getting a more informed heuristic. Thus, when the robot is very near to the goal, the heuristic function $h_n$ will be simply given by the value of the navigation function.

The efficiency of the planner can be further improved by preliminarily *growing* the workspace obstacles by the radius of the circle inscribed in the SuperMARIO bounding rectangle (see again Fig. 2) and then performing wavefront expansion. Below this minimum distance from the robot reference point there will be certainly collision with the obstacles.

Note finally that both $h_e$ and $h_n$ are admissible and locally consistent [15] heuristics. Hence, the resulting $A^*$ instances will be resolution complete and will gain the lowest $n \log n$ complexity (being $n$ the total number of cells). Indeed, refining the cell thickness $\delta$ will make the length of the resolution-optimal path closer to the real (continuous) optimum.

### 3.3 Smoothing with clothoids

The $A^*$ algorithm outputs a path $P$ made by a sequence of configurations $q_i = (x_i, y_i, \theta_i)$ which projects into a polygonal line on the $xy$ plane. Depending on the local sequence of $\theta_i$ around the vertices of this polygonal line, it may be convenient to join contiguous cartesian segments in a smooth way. This will avoid practical problems when the robot needs to follow the resulting smoothed path with a continuous timing law (trajectory tracking). In order to smooth corners, when needed, we have used clothoid curves that guarantee the following properties [10]:

- The smoothed curve never lies farther away than a threshold $\varepsilon$ from the original cartesian segments (this condition enables to preserve collision avoidance also for the smoothed curve).
- The total change of orientation along the curve equals the rotation at the corner between the two segments.
- The velocities of the robot wheels along the curve can be made continuous and never zero at the same time.

A typical clothoid smoothing is shown in Fig. 3, where $\varepsilon = |CP|$ can be regulated at will by the choice of the clothoid parameters. For contiguous segments to be executed with the same robot motion direction (forward or backward), the clothoid allows the use of a timing law with non-vanishing velocity. In the presence of a cusp (motion reversal), the linear velocity should indeed go to zero.

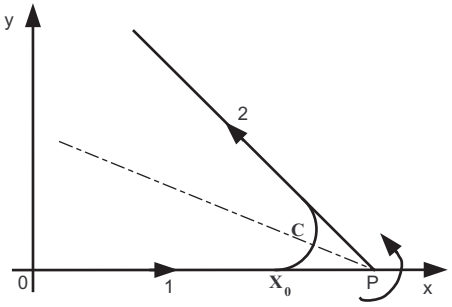


Figure 3: Example of clothoid smoothing

### 3.4 Planning results

We present numerical results on path planning using the $A^*$ algorithm with the two proposed heuristics and the two wavefront expansion strategies. We compare computational times in the following three cases, for the unicycle kinematics or a car-like behavior of SuperMARIO:

C1 Heuristic function $h_e$ (euclidean distance)

C2 Heuristic function $h_n$, with wavefront expansion on the cartesian obstacles

C3 Heuristic function $h_n$, with wavefront expansion on the grown obstacles

Figure 4 shows the test environment and the considered motion planning problem; note that a

'trap' for case C2 is present in the upper obstacle. The workspace is discretized into cells having $\delta = 4.4$ cm. If taken into account (car-like behavior), the maximum allowed curvature is $\kappa = \pi/(8\delta) \approx 0.088$ cm$^{-1}$.

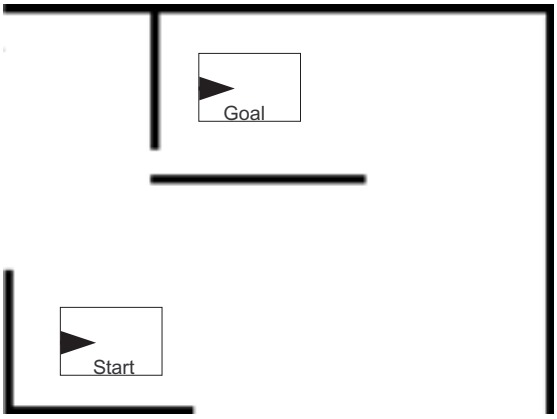


Figure 4: Test environment

Figures 5 and 6 show the wavefront expansions for cases C2 and C3, respectively. For case C2 (Fig. 5), the wave expansion is fooled by the trap and the $A^*$ planner search will be slowed down accordingly. In case C3, the preliminary obstacle growing closes the trap and so the heuristic $h_n$ will provide a more correct information.

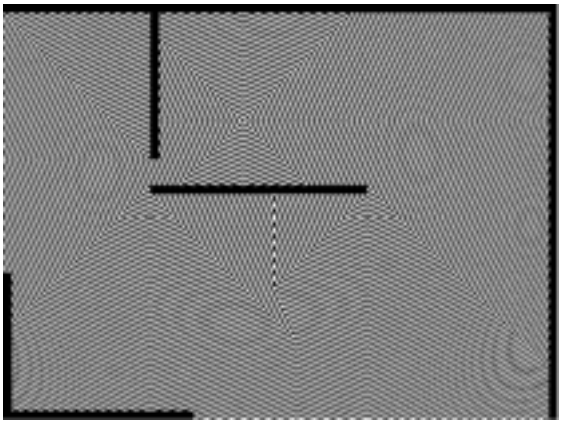

Figure 5: Wavefront expansion in case C2

Tables 1 and 2 report the $A^*$ computational times in the three cases, using the cell adjacency for the unicycle and, respectively, for the car-like kinematics. T1 is the wavefront expansion time (and the obstacle growing for case C3) while T2 is the search time (both expressed in seconds on the Pentium II PC governing SuperMARIO).

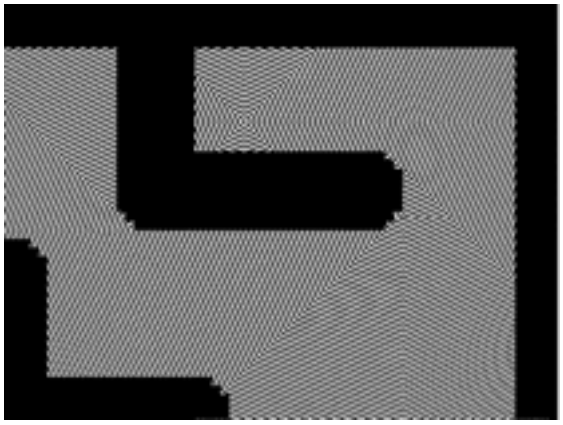

Figure 6: Wavefront expansion in case C3

| **Heuristic** | **T1** | **T2** | **Total** |
|---|---|---|---|
| C1 | 0 | 156 | 156 |
| C2 | 2 | 107 | 109 |
| C3 | 6 | 13 | 19 |

Table 1: Computational times with $A^*$ and different heuristics: unicycle case

| **Heuristic** | **T1** | **T2** | **Total** |
|---|---|---|---|
| C1 | 0 | 102 | 102 |
| C2 | 2 | 89 | 91 |
| C3 | 6 | 38 | 44 |

Table 2: Computational times with $A^*$ and different heuristics: car-like case

The euclidean heuristic $h_e$ used in C1 requires the longest time, even if no wavefront expansion is performed. In case C2, the heuristic $h_n$ improves $A^*$ performance but it is affected by the obstacle trap, while the best results are obtained in case C3 with an overall significant improvement, more pronounced indeed for the unicycle kinematics. The resulting motions generated by the planner, after the clothoid smoothing, are presented in Figs. 7 (unicycle) and 8 (car-like). Only the bounding rectangle is shown for SuperMARIO.

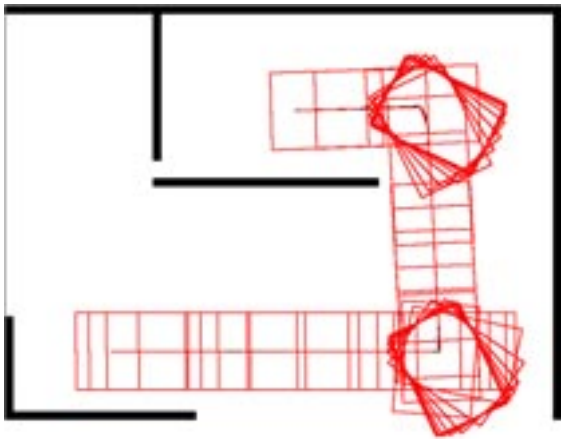

Figure 7: Planned motion: unicycle case

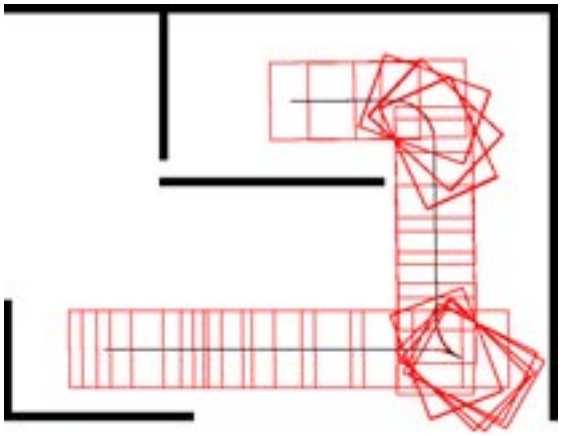

Figure 8: Planned motion: car-like case

## 4 Experimental results

In this section we present the results of an experiment in our Laboratory with the real robot SuperMARIO. Figure 9 shows the initial situation captured by the camera on the ceiling. After the preliminary elaborations, the obstacles can be easily localized (Fig. 10).

The path planner is invoked under the same maximum curvature constraint (car-like behavior) of Sect. 3, preliminary growing the obstacles and using the heuristic $h_n$ (case C3). The resulting path is smoothed with clothoids. In order to generate the reference trajectory used by the controller, a timing law is associated to the path, so as to start the robot from rest, reach the goal at rest, stop at cusps along the path (for motion inversion), and satisfy the following bounds on the robot actuators angular velocity and acceleration:

$$\begin{aligned} |\omega_i| &\leq \omega_{\max} = 3.52 \text{ rad/s} \\ |\dot{\omega}_i| &\leq \dot{\omega}_{\max} = 8.35 \text{ rad/s}^2, \end{aligned}$$

with $i \in R, L$ (right and left wheels).

Figure 11 shows the motion executed by SuperMARIO. Three motion inversions occur along the path generated by the planner. Accurate tracking of the trajectory is achieved using the following dynamic feedback linearizing control law (see [8]):

$$\begin{aligned} \dot{\xi} &= u_1 \cos\theta + u_2 \sin\theta \\ v &= \xi \\ \omega &= \frac{1}{\xi}\left[u_2 \cos\theta - u_1 \sin\theta\right], \end{aligned} \tag{5}$$

where $\xi$ is the (scalar) compensator state and $(u_1, u_2)$ is the auxiliary command input. It can be easily shown that the closed-loop system (1) and (5) is equivalent to two independent chains of two integrators between $(u_1, u_2)$ and $(x, y)$. Therefore, the auxiliary command input can be defined as a linear PD stabilizing feedback on the cartesian reference trajectory $(x_d(t), y_d(t))$

$$\begin{aligned} u_1 &= \ddot{x}_d + K_{p1}(x_d - x) + K_{d1}(\dot{x}_d - \dot{x}) \\ u_2 &= \ddot{y}_d + K_{p2}(y_d - y) + K_{d2}(\dot{y}_d - \dot{y}), \end{aligned} \tag{6}$$

with $K_{pi} > 0$, $K_{di} > 0$ $(i = 1, 2)$. The initial compensator state in eq. (5) must be $\xi_0 \neq 0$ and this condition should be preserved also during motion, in order to avoid control singularity [19]. The actual values of $(x, y, \theta)$ needed in eqs. (5–6) are evaluated by visual feedback every $T_c = 55$ ms, while $(\dot{x}, \dot{y})$ in eq. (6) can be determined as $(\xi\cos\theta, \xi\sin\theta)$, i.e., using the compensator state.

The evolution of the cartesian tracking error norm during motion is given in Fig. 12. The peaks $(2 \div 3$ cm) occurr in correspondence to zero velocity points, due to the filtering effect of the control saturation. The total length of the cartesian path is approximately 3.2 m and the execution time is rather long (65 s), just for safety reasons.

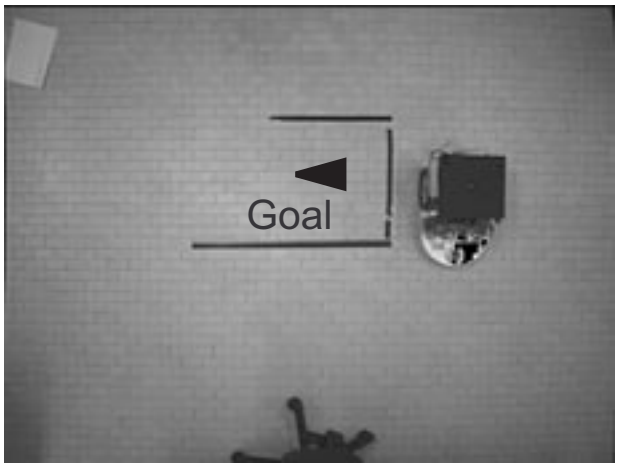


Figure 9: Initial image captured by the camera

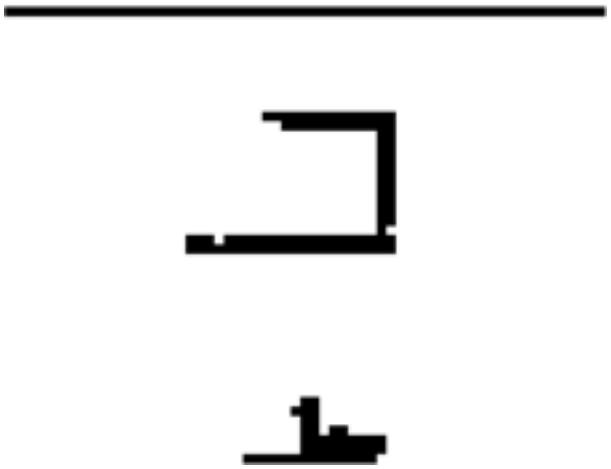

Figure 10: Detected obstacles (the robot is localized first and the removed from the picture)

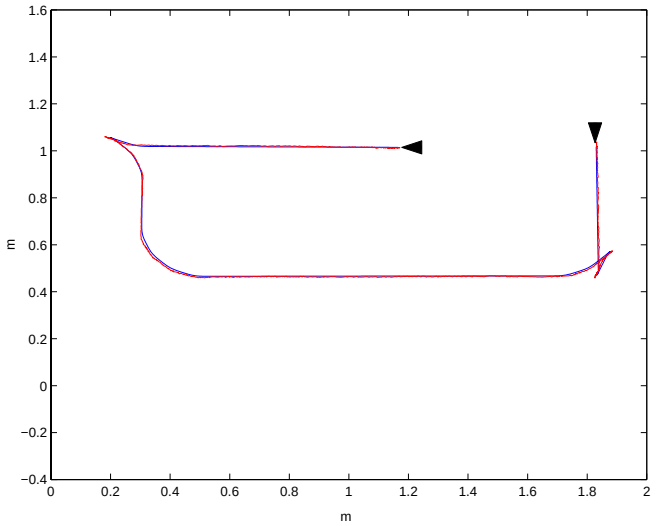


Figure 11: Motion execution: trace of the robot reference point $(x, y)$

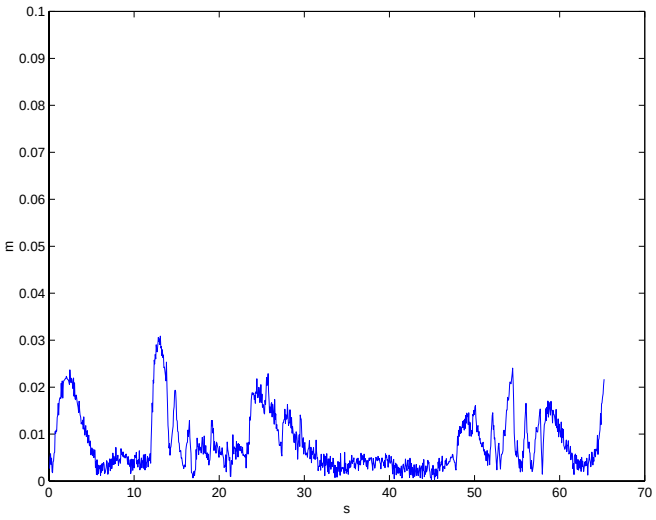


Figure 12: Motion execution: norm of the cartesian tracking error

## 5 Conclusions

An integrated visual-based approach to motion planning and control of a nonholonomic wheeled mobile robot has been developed and experimentally tested on the laboratory prototype robot SuperMARIO. The motion planner is based on the $A^*$ algorithm, defining cell adjacency and locally consistent heuristic functions so as to take into account the robot nonholonomic constraints (unicycle or car-like kinematics) as well as the obstacle locations, that are reconstructed by the vision system. The combination of these ideas provides a considerable improvement in the overall computational load. The resulting smoothed collision-free path (with an associated timing law) is then accurately executed in real-time, by using a trajectory tracking controller based on visual feedback.

Future work includes the consideration of dynamic obstacles, both at the planning and control level. We are also pursuing the use of an on-board pan-tilt camera. In this case, one should develop incremental visual-based planning methods and execute motion by visual servoing, i.e., with the task/trajectory and the associated errors directly defined in terms of features in the image plane.

## Acknowledgements

Work supported by MURST (Italian Ministry of University and Research) project *MISTRAL*.

## References

[1] A. Astolfi. "Discontinuous control of nonholonomic systems", *Systems & Control Lett.*, **27**, pp. 37–45, (1996).

[2] S. Atyia, G. D. Hager. "Real-time vision-based robot localization", *IEEE Trans. on Robotics and Automation*, **9**(6), pp. 785-800, (1993).

[3] D. J. Balkcom, M. T. Mason. "Time optimal trajectories for bounded velocity differential drive robots", *2000 IEEE Int. Conf. on Robotics and Automation*, pp. 2499–2504, (2000).

[4] F. Conticelli, B. Allotta, P. K. Khosla. "Image-based visual servoing of nonholonomic mobile robots", *38th IEEE Conf. on Decision and Control*, pp. 3496–3501, (1999).

[5] P. I. Corke. *Visual Control of Robots: High-performance Visual Servoing*, Research Studies Press, Taunton, UK, (1996).

[6] A. De Luca, G. Oriolo, C. Samson. "Feedback control of a nonholonomic car-like robot", in *Robot Motion Planning and Control*, J.-P. Laumond (Ed.), Lecture Notes in Control and Information Sciences, **229**, pp. 171–253, Springer Verlag, London, (1998).

[7] A. De Luca, G. Oriolo, M. Vendittelli. "Stabilization of the unicycle via dynamic feedback linearization", *6th IFAC Symp. on Robot Control*, pp. 397–402, (2000).

[8] A. De Luca, G. Oriolo, L. Paone, P. Robuffo Giordano. "Experiments in visual feedback control of a wheeled mobile robot", *2002 IEEE Int. Conf. on Robotics and Automation*, to appear, (2002).

[9] H. F. Durrant-Whyte. "Where am I? A tutorial on mobile vehicle localization", *Industrial Robot*, **21**(2), pp. 11-16, (1994).

[10] S. Fleury, P. Souères, J.-P. Laumond, R. Chatila. "Primitives for smoothing mobile robots trajectories", *IEEE Trans. on Robotics and Automation*, **11**(3), pp. 441-448, (1995).

[11] K. Hashimoto, T. Noritsugu. "Visual servoing of nonholonomic cart", *1997 IEEE Int. Conf. on Robotics and Automation*, pp. 1719–1724, (1997).

[12] B. Kwolek, T. Kapuscinski, M. Wysocki. "Vision-based implementation of feedback control of unicycle robots", *1st Work. on Robot Motion and Control*, pp. 101–106, (1999).

[13] F. Lamiraux, S. Sekhavat, J.-P. Laumond. "Motion planning and control for Hilare pulling a trailer", *IEEE Trans. on Robotics and Automation*, **15**(4), pp. 640–652, (1999).

[14] F. Lamiraux, J.-P. Laumond. "Smooth motion planning for car-like vehicles", *IEEE Trans. on Robotics and Automation*, **17**(4), pp. 498–502, (2001).

[15] J.-C. Latombe. *Robot Motion Planning*, Kluwer Academic Publishers, Norwell, MA, (1991).

[16] Y. Masutani, M. Mikawa, N. Maru, F. Miyazaki. "Visual servoing for nonholonomic mobile robots", *1994 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1133–1140, (1994).

[17] G. Oriolo, M. Vendittelli, G. Ulivi. "Path planning for mobile robots via skeletons on fuzzy maps", *Intelligent Automation and Soft Computing*, **2**(4), pp. 355–374, (1996).

[18] G. Oriolo, G. Ulivi, M. Vendittelli. "Real-time map building and navigation for autonomous robots in unknown environments planning", *IEEE Trans. on Systems, Man, and Cybernetics – Part B*, **28**(3), pp. 316–333, (1998).

[19] G. Oriolo, A. De Luca, M. Vendittelli. "WMR control via dynamic feedback linearization: Design, implementation and experimental validation", *IEEE Trans. on Control Systems Technology*, to appear, (2002).

[20] C. Samson. "Control of chained systems. Application to path following and time-varying point-stabilization of mobile robots", *IEEE Trans. on Automatic Control*, **40**(1), pp. 64–77, (1995).

[21] S. Sekhavat, J.-P. Laumond. "Topological property for collision-free nonholonomic motion planning: the case of sinusoidal inputs for chained form systems", *IEEE Trans. on Robotics and Automation*, **14**(4), pp. 671–680, (1998).