The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

A Minimalist Approach to Path Following among Unknown Obstacles

Matteo Campani, Francesco Capezio, Alberto Rebora, Antonio Sgorbissa, Renato Zaccaria

Abstract— **The article proposes a feedback control system for path following in presence of obstacles that is an extension of previous work and is made of two components: (i) a sensor-based, real-time model that generates and periodically updates the path on–line in order to avoid both known and unforeseen obstacles, and (ii) a feedback-control model that is capable of driving a unicycle vehicle along the collision free path. The system has some unique characteristics, among which it requires very few computational resources as a consequence of its extreme simplicity.**

I. Introduction

The article proposes a feedback control system for path following in presence of obstacles. The approach is an extension of the work proposed in [1], however it has many differences with the original work.

This article consider an indoor AGV with unicycle kinematics moving in a partially known environment (e.g., a hospital, a warehouse, etc.), which has to follow a prescribed path but can encounter unforeseen obstacles: in particular, it can happen that the prescribed path cannot be followed due to objects left unattended, pieces of furniture, other vehicles, people, and finally errors in localization. Under these conditions, the approach used by commercial AGVs to avoid obstacles in real–world situations (e.g., in a corridor crowded with people) is often *not to avoid them at all*[1]: experience tells that if the vehicle slows down or stops, warns people, and waits, it definitely has more probability of success that trying to find a path to the goal by all means, with the risk of ending up in a configuration from which it is difficult to come out. However, it appears reasonable that slight deviations from the nominal path should be taken into considerations to guarantee a safe and efficient behaviour: for example, even a small error in localization can prevent the AGV from passing through a narrow door or enter an elevator, a situation which could be easily handled by allowing deviations from the path.

In literature, the problem of obstacle avoidance is often investigated separately from path following [2][3][4][5]. Some of the most successful approaches to obstacle avoidance [6][7] rely on the idea of computing "artificial" repulsive forces which are exerted on the robot by surrounding obstacles, and an attractive force exerted by the goal. All of these forces are then summed up to produce a resulting force vector that is used to control the motion of the robot. Artificial Potential Fields (APF) and similar approaches have well known problems [8], among which the presence of local minima in the field (e.g., in correspondence of narrow passages). In addition, if *occupancy grids* are used to store sensor data, APF can produce oscillation in the resulting velocity vector (and hence on the robot path) due to the finite resolution of the grid. The problem is well described in [9], which argues that the latter two problems are both due to the fact that a huge amount of sensor data are summarized in just one force vector, and proposes the Vector Field Histogram (VFH) as a solution. In VFH, the concept of "resulting force" is substituted with the concepts of "valleys" towards which the robot is allow to navigate. VFH has been applied both in indoor and outdoor robotic applications, and improvements are described in [10][11]. Recently, obstacle avoidance in very dense, complex and cluttered scenario has been considered in [14]. In spite of the many important differences, all of the previous approaches share a common characteristic: they face obstacle avoidance as a totally separate problem from path following, therefore requiring to transform the velocity vector into commands to actuators in a separate phase, which turns out to be a complex task in presence of nonholomic and other kinematics constraints.

This problem is explicitly considered in [15], which presents the Curvature-Velocity Method (CVM) that formulates obstacle avoidance as a problem of constrained optimization in velocity space. Physical limitations (velocities and accelerations) and the configuration of obstacles place constraints on the translational and rotational velocities of the robot. The robot chooses velocity commands that satisfy all the constraints and maximize an objective function that trades off speed, safety and goal-directedness. A variant of the approach is proposed, e.g., in [16]. In a similar spirit, [17] presents the Dynamic Window approach (DW), which relies on the idea of performing a local search for admissible velocities which allow to avoid obstacles while meeting kinematics constraints. A theoretical treatment of the algorithm's convergence properties is proposed in [18]. CVM and the DW allow to deal with kinematics constraints, however they still have the problem of local minima. To avoid them, a solution is proposed in [19] by introducing a planning stage in DW which produces collision-free local paths with a given velocity profile.

An extension of the traditional APF in order to deal with kinematics constraints is proposed in [20][21], where the whole path is ideally deformed on the basis of forces exerted by surrounding obstacles. In particular, the initial path is augmented by a set of paths homotopic to it, represented implicitly by a volume of free space in the work space

M. Campani, F. Capezio, A. Rebora, A. Sgorbissa, R. Zaccaria are with DIST, Department of Communication, Computer and System Sciences, University of Genoa, Via Opera Pia 13, 16145, Genoa, Italy. Corresponding Author email:{antonio.sgorbissa}@unige.it.

[1]See the description of a successful case study in http://www.swisslog.com/hcs-agv-memorialhermann.pdf

which describes the maximum allowed deviation from the path. During execution, reactive control algorithms are used to select a valid path from the set of homotopic paths, using proximity to the environment in a sense very similar to [6]. The concept of path deformation is considered also in [22], in which the current path is described as a mapping from an interval of real numbers into the configuration space of the robot, and iteratively deformed in order to get away from obstacles and satisfy the nonholonomic constraints of the robot. The approach has been shown to work with complex non-holomic systems (e.g., a trailer) with complex shapes. Approaches based on path deformation avoid local minima since the path connection is preserved during deformation. However, they have high computational and memory requirements, because they require either to memorize a set of alternative paths, or to compute the path deformation in run–time.

Obstacle avoidance is fully integrated with path following in [23]: path following is achieved by controlling explicitly the rate of progression of a "virtual target" to be tracked along the path [4][5], and obstacle avoidance relies on the deformable virtual zone principle, that defines a safety zone around the vehicle, in which the presence of an obstacle drives the vehicle reaction. However, as stated by authors, the combination of path following with a reactive - local - obstacle avoidance strategy has a natural limitation coming from the situation where both controllers yield antagonist system reactions. This situation leads to a local minimum, where a heuristic switch between controllers is necessary.

Finally, even if the problem is not explicitly addressed in this work, it is worth remarking that obstacle motion prediction has been performed in [26][25], and more recently in [28][29][24][27]. All of these approaches increase robustness at a price of a higher computational complexity.

In this general scenario, the contributions of this work are the following.

First, it proposes a method for obstacle avoidance which is highly integrated with path following, and produces paths which are directly executable by an AGV with unicycle kinematics [31] (a similar goal is shared by [15][17][20][23], as well as by our maze–solving algorithm [30]). This approach is completely different from standard APF, which returns a force vector [6][7] which must be properly transformed into a velocity profile to be fed to actuators. Deadlock situations are avoided by allowing only slight deviations from the nominal path, which appears reasonable in many indoor applications: however, there are situations in which a path to the goal is not found even if it theorically exist.

Second, the approach offers a new solution to the well–known dilemma that one has to face when dealing with multiple sensor readings, i.e., whether it is better, to summarize a huge amount of sensor data, to consider only the closest sensor reading [6], to consider all sensor reading separately to compute the composite force vector [7], or to build a local map or histogram to cluster data in some way [9], [32]. In this work, similarly to APF, all sensor readings returned by proximity sensors are considered separately, as if they would describe a cloud of "point–like" obstacles. Next, a simple rule is used to compute a motion law that avoids all point–like obstacles which can cause a collision, while keeping the AGV as close as possible to the nominal path. In particular, this is achieved by *artificially* reducing or increasing the position error (i.e., the distance to the nominal path) through simple, real–time computations that consider only dangerous point–like obstacles in the immediate surroundings. The approach requires very little memory and computational resources with respect, e.g., to path–deformation approaches [20][21][22], and for this reason it can be used in any kind of unknown or changing environment, even by very simple robots.

Section II describes the system. Section III shows experimental results. Conclusions follow.

II. Obstacle Avoidance

A. General Ideas

Consider a robot which is given a sequence of waypoints to be reached, either they have been specified by the user, or they have been produced by a high–level path planner on the basis of a model of the environment. Consider, as an additional constraint, that waypoints have been chosen in such a way as to guarantee that, in absence of obstacles and in case of perfect localization, it is always possible to navigate from a via point to the next one by following a straight path: in the following this straight path is referred to as the *nominal path*. This work focuses on the problem of locally modifying the *nominal path* between two waypoints in order to guarantee that, even in presence of unforeseen obstacles and of big errors in localization, the robot will still be able to find a way to the next via point given that a path exists.

Under these assumptions, the *nominal path* between subsequent waypoints can be described through its implicit equation in the form:

$$f(X,Y) = aX + bY + c = 0. \tag{1}$$

Without loosing generality, through a proper coordinate transform, it is possibile to define a local Cartesian Frame $\mathcal{F}$ such that the path lies along the X-axis of $\mathcal{F}$. The implicit equation of the straight line in $\mathcal{F}$ simply becomes:

$$f(X,Y) = Y = 0. \tag{2}$$

Assume now a robot with unicycle kinematics, whose position and orientation in $\mathcal{F}$ are described through a vector $\eta \equiv [X_r Y_r \psi_r]^T$. The motion of the robot can be described through the following state equations:

$$\begin{aligned} \dot{X}_r &= u_1 \cos\psi_r \\ \dot{Y}_r &= u_1 \sin\psi_r \\ \dot{\psi}_r &= u_2, \end{aligned} \tag{3}$$

where inputs u_1 and u_2 are – respectively – the translational and the rotational velocities.

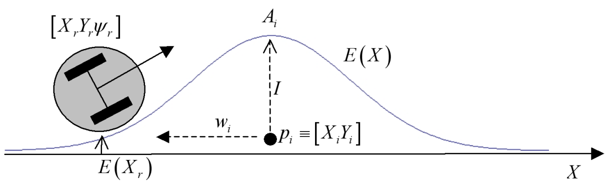

Fig. 1. A *point obstacle* p_i and the corresponding error function $E(X)$.

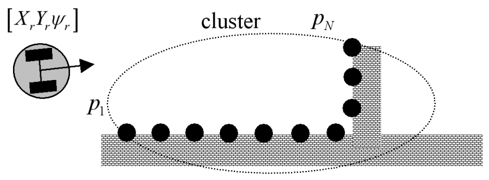

Fig. 2. Gray circle: robot. Black circles: sensor readings. Dashed ellipse: cluster of sensor readings modelled as a single obstacle.

In [31] a method has been presented which allows to achieve convergence to a generic curve expressed through its implicit equation in the form $f(X,Y)=0$ by setting:

$$\begin{array}{l} u_1 = U_1 \\ u_2 = K(-\|\nabla f\| f - \frac{d}{dt} f) + \dot{\psi}_c, \end{array} \tag{4}$$

where $f_X = \frac{\partial f(X,Y)}{\partial X}$, $f_Y = \frac{\partial f(X,Y)}{\partial Y}$ are the partial derivatives of $f = f(X,Y)$, and $\|\nabla f\| = \|\nabla f(X,Y)\|$ is the norm of the gradient of f. The angle ψ_c is the orientation of the tangent to the level curve of $f(X,Y)=0$ in (X_r, Y_r), which can either be computed as $\psi_c = \tan^{-1}(-f_X/f_Y)$ or $\psi_c = \cot^{-1}(-f_Y/f_X)$; $\dot{\psi}_c$ is the derivative of ψ_c with respect to time. The term U_1 is a positive constant[2].

In order to avoid obstacles detected by sensors, the general idea is that of choosing an error function $E(X,Y)$ which is subtracted from $f(X,Y)$, and to use the new quantity $f'(X,Y) = f(X,Y) - E(X,Y) = 0$ to substitute $f(X,Y)$ in (4). If the error function is properly chosen such as $E(X,Y) > 0$ in the close proximity of obstacles and $E(X,Y) \approx 0$ otherwise, the new path expressed as $f'(X,Y)=0$ almost overlaps with the nominal path while locally avoiding obstacles.

Consider for example an ideal *point obstacle* $p_i \equiv [X_i Y_i]^T$ which has been detected at time t and lies on the robot's path (Figure 1). After having decided a steering direction for avoiding the obstacle (i.e., left or right), it turns out that, if the *nominal path* is a straight line aligned with the X-axis, a good choice for $E(X,Y)$ is:

$$E(X,Y) = E(X) = A_i e^{-\frac{(X-X_i)^2}{2w_i^2}}, \tag{5}$$

which returns the following expression for $f'(X,Y)=0$:

$$f'(X,Y) = Y - E(X) = 0. \tag{6}$$

Equation (6) describes a Gaussian aligned with the $X-$ axis. The parameter A_i represents the height of the Gaussian, and must be chosen in order to allow the vehicle to safely avoid the *point obstacle* p_i. Towards this end, it is possible to set:

$$A_i = Y_i + I, \tag{7}$$

where the absolute value I depends on the dimensions of the robot and on a safety distance arbitrarily chosen, and its sign depends on the steering direction which has been chosen to avoid the obstacle.

[2]The method has been recently updated to consider a generic speed profile [33], but the work has not been yet published.

The parameter w_i determines at which distance from X_i the contribution of $E(X)$ becomes negligible. That is, by varying w_i it is possible to vary the time instant in which a robot in $[Y_r X_r \psi_r]^T$ must leave the *nominal path* in order to avoid the obstacle.

Remark 1. Since the error in (5) is computed only for the current robot position and on the basis of sensor data returned at a time t, there is never a waste of computational resources: the motion planner never re–plans a whole path to the goal, but computes just what is needed "here and now". To stress this fact, the error is always written as $E(X_r)$ in the following. □

Obstacles detected by sensors at time t usually do not consist in a single point p_i, but in a cloud of points $p_i \in \mathcal{P}$, with $i = 1 \dots N$. Therefore, it is necessary to compute an error $E(X_r)$ which depends on all sensor data which have been returned. The rest of the paper deals with this problem, which plays a fundamental role in all reactive–methods which are required to deal with a huge amount of sensorial data.

B. Obstacle Model

Given a cloud of points $p_i \in \mathcal{P}$, with $i = 1 \dots N$, describing the environment around the robot, i.e., corresponding to sensor readings returned by a proximity sensor, it is necessary to decide how to interpret them.

A possible approach is that of clustering nearby points which could belong to the same obstacle [1]. This approach, however, has some limitations. Consider Figure 2, showing an extreme situation in which a cloud of points distributed along a wall have been modelled as a single ellipsoidal obstacle: it is straightfoward to notice that the corresponding ellipse is too big to allow avoiding the obstacle in a reasonable way. Different geometric primitives, e.g., line segments, can be used to model obstacles. However, two major problems remain: i) clustering is computationally expensive; ii) the shape of obstacles is necessarily approximated by the geometric primitive adopted.

In this work a different approach is proposed, which share some similarities with reactive–planning approaches based on Artificial Potential Fields but also overcome some of their limitations. The basic idea is that of introducing the concept of *dangerous points*. A point $p_i \equiv [X_i, Y_i]$ (corresponding to a sensor reading) is considered dangerous for navigation *if and only if* one of the two following conditions is true:

- the point is closer to the *nominal path* than the robot (*initialization of dangerous points*);

- the point is close to another *dangerous point* less than D_{max}, a threshold value which takes into account the dimensions of the robot (*propagation of dangerous points*).

More in details, given a set $p_i \in \mathcal{P}$ of points returned by the laser at time t, and radially ordered according to their azimuth direction with respect to the laser, it is possible to identify *dangerous points* through the following algorithm.

Algorithm 1 Find *dangerous points*

```
Require: $[X_r Y_r \psi_r]$ {robot position}
    $I$ {used for initialization}
    $D_{max}$ {used for propagation }
    $\mathcal{P}$ {radially ordered set of $N$ points}
    $\mathcal{B}$ {ordered buffer of $M$ points}
Ensure: $\mathcal{D}$ {ordered set of dangerous points}
1:  if $Y_r \geq 0$ then
2:    for all $p_i \in \mathcal{P}$ do
3:      if $-I \leq Y_i \leq Y_r + I$ then
4:        PUSH($p_i$,$\mathcal{D}$) {initialization}
5:        PUSH($p_i$,$\mathcal{B}$) {the point is buffered}
6:      else
7:        for all $b_j \in \mathcal{B}$ do
8:          if $|p_i - b_j| \leq D_{max}$ then
9:            PUSH($p_i$,$\mathcal{D}$) {propagation}
10:           PUSH($p_i$,$\mathcal{B}$) {the point is buffered}
11:           break
12:         end if
13:       end for
14:     end if
15:   end for
16: end if
17: return $\mathcal{D}$
```

The algorithm assumes that robot steers to the left when approaching an obstacle, therefore being always in the half plane with $Y_r \geq 0$ (line 1). A symmetrical check should be performed if the robot is allowed to steer to the right.

Line 2 considers points p_i, $i = 1 \dots N$ according to their radial disposition around the robot: this allows to simplify the following computations, since the search for *dangerous points* operates on an ordered set. Line 3 checks whether the current point p_i is closer to the *nominal path* than the robot or not. If so, p_i is initialized as *dangerous* (line 4) and buffered (line 5). If not, *dangerous points* in the buffer are considered in inverse order, and the algorithm checks whether one of these points b_j is closer to p_i that D_{max} (line 8). If so, b_j propagates to p_i, which is classified as dangerous (line 9) and buffered (line 10).

Remark 2. According to Line 2 and 7, for every point $p_i \in \mathcal{P}$ it is necessary to check the distance from all dangerous points which have been stored in the buffer so far. The complexity of the algorithm can be reduced to $O(NM)$ by limiting the dimension of the buffer to a finite value M (in experiments it has been set $M = 1$). Since the algorithm memorizes only points with correspond to the last laser scan (i.e., without building any map of the environment), it requires a minimal amount of storage memory and computational power, thus being executable in real–time even in less powerfull microcontroller.

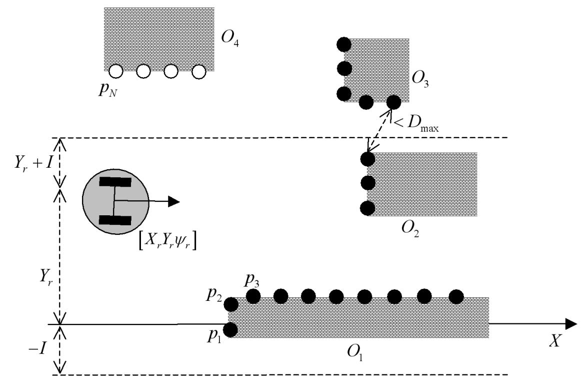

Fig. 3. Gray circle: robot. Black circles: dangerous points. White circles: not dangerous points.

In Figure 3 it is possible to notice points which are classified as *dangerous* by the algorithm (black circles). Points belonging to objects O_1 and O_2 are classified as *dangerous* since they are closer to the path than the robot (line 3 in the algorithm). Points belonging to object O_3 are classified as *dangerous* since they are closer to other *dangerous points* than D_{max} (line 8 in the algorithm). Finally, points belonging to object O_4 do not meet any of the previous conditions, and therefore they are classified as *not dangerous*. Points are checked in their radial order starting from point p_1 to p_N, i.e., the order in which they were stored during laser scanning.

When M is high, the computational complexity increases, but the efficiency of the algorithm increases as well. To verify this, consider Figure 4 in which it has been set $M = 1$. The robot is moving along the path by avoiding obstacles O_1 and O_2, but the points belonging to O_3 have not yet been classified as *dangerous*. This happens because, when considering point p_9, the algorithm checks the distance with the only *dangerous point* in the buffer, i.e., point $p_8 \in \mathcal{B}$. Since the distance between these latter two points is greater than D_{max}, point p_9 is classified as *not dangerous*. However, the distance between point p_9 and point $p_6 \in \mathcal{D}$ is less than D_{max}, which would have been easily detected by increasing the dimension of the buffer, e.g., by setting $M \geq 3$.

Even by setting $M = 1$, the robot still manages to deal with the difficult situation. In fact, while the robot moves in order to avoid obstacles O_1 and O_2, Y_r necessarily increases. If the distance between point p_6 and p_9 is smaller than I, sooner or later the algorithm will classify point p_9 as a *dangerous point* in subsequent scans by applying the *initialization* condition in line 3. Obviously, an early detection of obstacles improves the run–time behaviour of the robot, since it allows to start steering earlier, therefore producing paths with a lower curvature. The limit situation $M = N$ guarantees to detect all *dangerous points* as long as obstacles appear in the field of view, at the price of a

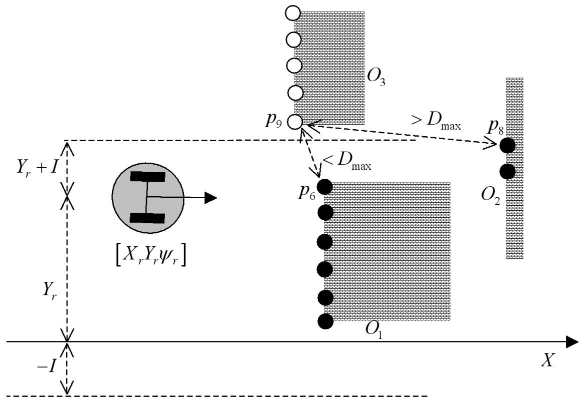

Fig. 4. Gray circle: robot. Black circles: dangerous points. White circles: not dangerous points.

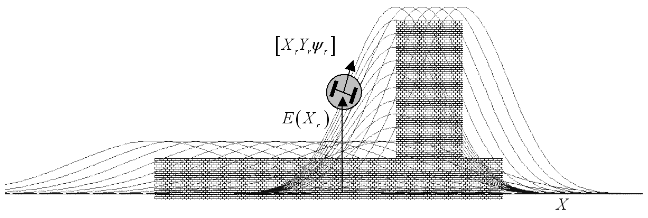

Fig. 5. Error fuctions generated by dangerous points.

compexity $O(N^2)$.

Once *dangerous points* have been detected, it is necessary to choose a proper error $E(X_r)$ to be added to X_r. Differently from previous work, the approach proposed here is to define an error $E_i(X_r)$ for every dangerous point $p_i \in \mathcal{D}$ with $i = 1 \dots K$ using the same expression in (5), i.e.:

$$E_i(X_r) = A_i e^{-\frac{(X_r - X_i)^2}{2w_i^2}}. \quad (8)$$

Finally,

$$E(X_r) = \max_i E_i(X_r), \quad i = 1 \dots K, \quad (9)$$

that is the error is computed as the maximum among all contributions given by *dangerous points*.

An example is shown in Figure 5. By observing the Figure one could expect that, as X_r varies, the robot moves along a wavy profile due to the finite angular resolution of the laser sensors, which produce a finite number of Gaussian curves $E_i(X)$ even in presence of straight obstacles (e.g., walls). However, it is important to remark that the final path followed by the robot cannot be computed using (9) by varying X_r. In fact, even in absence of moving obstacles, the position of *dangerous points* $p_i \in \mathcal{D} = \mathcal{D}(t)$ at time t, as well as their number $K = K(t)$ depends on the configuration of the robot itself $\eta(t) = [X_r(t) Y_r(t) \psi_r(t)]$.

Therefore, the profile followed by the robot turns out to be, at every t:

$$T(t) = \max_i A_i e^{-\frac{(X_r(t) - X_i(\eta(t)))^2}{2w_i^2}}, \quad i = 1 \dots K(t). \quad (10)$$

This concept can be clarified by assuming a robot that moves parallel to an ideal *infinite wall*: as the robot moves, the number $K(t)$ of *dangerous points* and their positions relative to the robots $p_i \in \mathcal{D}(t)$ do not vary with t, thus necessarily producing a constant profile for $T(t)$. This is compatible with (10) as t varies, but different from what can be expected from (9) as X_r varies.

Remark 3. The profile $T(t)$ is never stored in memory. At every control cylce, path following requires only the value of $E(X_r)$ in X_r to compute controls as defined in (6) and (4). To stress this point, the dependence on time has not been written explicitly in (8) and (9). $\square$

Finally, a mechanism is introduced to overcome limitations of real sensors, in particular on the maximum sensing angle[3]. If the sensing angle is less than 360^o, it can happen that the set of points $p_i \in \mathcal{D}(t)$ (which depends on $\eta(t)$) changes dramatically at time $t+1$ simply because a big obstacle has disappeared behind the robot. To avoid this effect, the system memorizes at time t the parameters $X_j(t)$, $A_j(t)$, $w_j(t)$ of the Gaussian $E_j(X_r(t))$ that has been selected to contribute to $E(X_r(t))$, that is:

$$j(t) = \arg\max_i E_i(X_r(t)), \quad i = 1 \dots K(t). \quad (11)$$

At time $t+1$, and *if and only if* $X_r(t+1) > X_j(t)$ (i.e., the robot is on the descending side of the Gaussian), $X_j(t)$, $A_j(t)$, $w_j(t)$ are used to compute an additional Gaussian which is then compared with the $K(t+1)$ Gaussians derived from *dangerous points* $p_i \in \mathcal{D}(t+1)$. This guarantees that, even if the obstacle which produced $E_j(X_r(t))$ suddenly disappears behind the robot, the robot still smoothly converges to the *nominal path* along the same profile.

III. Experimental Results

In order to validate the model, many tests have been performed in a real indoor environment. Towards this end, a LabMate robot with unicycle kinematics has been used, equipped with a SICK laserscanner with a maximum sensing angle of 180^o and a sensing range of about 16 meters. Experimental runs shown here have been classified into 3 typologies of tests, which represent typical situations encountered in real application domains. Tests have been performed for different values of the translational speed, ranging from 0.15 to 0.6 m/s. Finally, in case that the obstacle avoidance mechanism described is not sufficient to find a path to the goal (which can happen, for example, in the case of a very cluttered or crowded environment) an additional safety mechanism is implemented which stops the robot, turns it towards the goal, and possibly asks people for help. In a complete architecture for robot navigation, this situation could require the intervention of a high–level path planner which finds an alternative solution. This problems is not faced here.

Figure 6 shows the first test typology. The robot is requested to follow a *nominal path* which lies on the border of a wall. This can happen as a consequence of a localization

[3]The robot used for experiments uses a SICK with a 180^o angular width

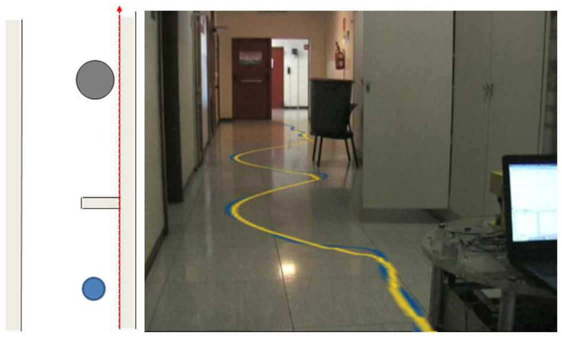

Fig. 6. Test 1: moving along a corridor with a localization error.

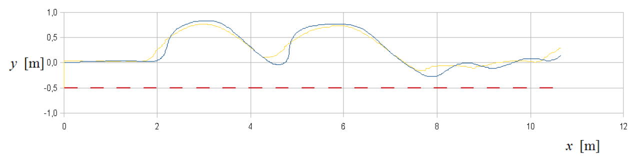

Fig. 7. Plot of $(X_r(t), T(t))$ and $(X_r(t), Y_r(t))$ for test 1.

error. Even if the *nominal path* was defined in order to stay in the middle of the corridor at a safety distance from both walls, the robot has a wrong estimate of its own position: in absence of an obstacle avoidance strategy, it would try to converge to the path on its right, thus incurring in a collision. The test validates the performance of the system to keep the robot at a safety distance from the wall, while avoiding additional obstacles on its path.

Figure 7 shows a plot of the imposed and the actual path at a speed of 0.3 m/s. The X-axis is oriented along the corridor, with a *nominal path* expressed as $Y = -0.5$ (dashed line). The yellow curve shows the plot of $(X_r(t), T(t))$: this corresponds to the imposed profile that the robot is instructed to follow by the obstacle avoidance algorithm. The curve in blue shows the actual profile $(X_r(t), Y_r(t))$ followed by the robot, provided by odometry. The average and the standard deviation of the difference $D_r = |T(t) - Y_r(t)|$ between the imposed and the actual profile are computed. Notice that the actual path does not asymptotically converge to the imposed path: this is due to the fact that the imposed path changes every time t depending on sensor readings, according to (10).

Table I reports the statistics of D_r, averaged over different runs of test 1, for different speeds.

Figure 8 shows the second test typology, which consists in passing through a door. The passage is quite narrow compared with the robot's dimensions (it is only 25cm wider than the robot), and therefore this test is performed only at the lower speed of 0.15 m/s for safety reasons. Figure 9 shows a plot of the imposed and the actual path. Statistics are shown in Table I. The system proves to perform well even in a situation which, in APF–based approaches, is very likely to produce local minima in the potential function (a problem which requires carefully tuning of all parameters to be avoided).

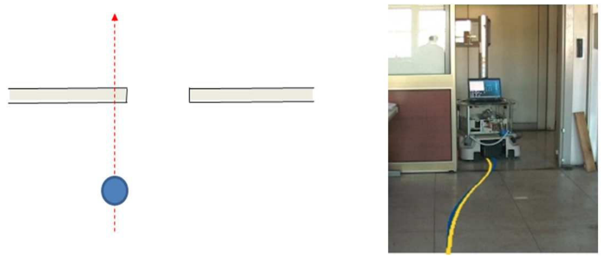

Fig. 8. Test 2: passing through a door with a localization error.

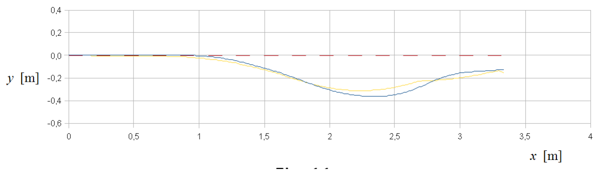

Fig. 9. Plot of $(X_r(t), T(t))$ and $(X_r(t), Y_r(t))$ for test 2.

Figure 10 shows the third test typology, which consists in avoiding two subsequent obstacles that intersect the *nominal path*. In this experiment, a method has been implemented for automatically choosing the steering direction (left or right) on the basis of some heuristics. This allows to find a path which could not be found otherwise. Figure 11 shows a plot of the imposed and the actual path at a speed of 0.3 m/s. Statistics are shown in Table I.

Experiments with moving persons have been performed as well, but are not reported here since are hardly repeatable and therefore it is not easy to draw quantitative results about theme. In spite of this, and from a qualitative point of view, the system proves to perform properly even in presence of dynamically moving obstacles.

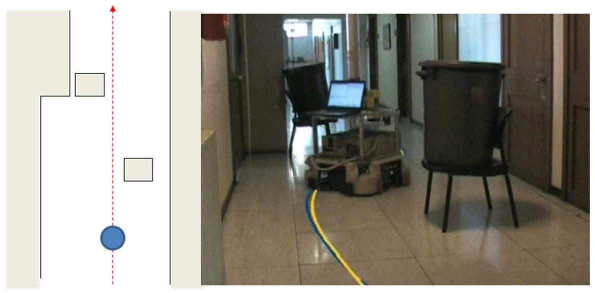

Fig. 10. Test 3: avoiding two obstacles by choosing the steering direction.

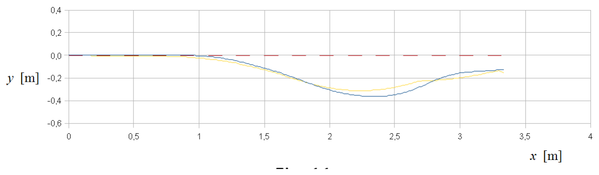

Fig. 11. Plot of $(X_r(t), T(t))$ and $(X_r(t), Y_r(t))$ for test 3.

TABLE I
RESULTS OF EXPERIMENTS

Test Type	Speed [m/s]	$\bar{D}_r$ [m]	σ_{D_r} [m]
test 1	0.15	0.034	0.037
test 1	0.3	0.08	0.033
test 2	0.15	0.021	0.019
test 3	0.15	0.042	0.079
test 3	0.3	0.064	0.101
test 3	0.6	0.094	0.098

IV. CONCLUSIONS

The article describes a novel approach to obstacle avoidance which shares some similarities with Artificial Potential Fields and methods based on the online deformation of the path, but offer some advantages.

First, it proposes a method for obstacle avoidance which is highly integrated with path following, and produces paths which are directly executable by an AGV with unicycle kinematics. Second, it proposes an original and efficient approach to deal with a huge amount of sensor data.

Experiments with a real robot has been performed up to a speed of 0.6m/s, both with statics and dynamic obstacles, allowing to validate theoretic results. In the close future this work will be extended in order to consider the case that the *nominal path* is not a straight line, but is represented as a generic curve in $2D$ expressed through its implicit equation in the form $f(X, Y) = 0$ [31].

REFERENCES

[1] A. Sgorbissa, A. Villa, A. Vargiu, and R. Zaccaria, A Lyapunov-stable, sensor-based model for real-time path-tracking among unknown obstacles, 2009 IEEE/RSJ Int. Conf. on Intelligent RObots and Systems, October 11 – 15, 2009, St. Louis, MO, USA

[2] Samson, C. and Ait-Abderrahim, K., Mobile Robot Control Part 1: Feedback Control of A Non-Holonomic Mobile Robots, Technical Report No. 1281, INRIA, Sophia-Antipolis, France, June 1991.

[3] C. Canudas de Wit, H. Khennoul, C. Samson, and O. J. Sordalen, Nonlinear control design for mobile robots, in Recent Trends in Mobile Robots, ser. Robotics and Automated Systems, Y. F. Zheng, Ed. World Scientific, 1993, ch. 5, pp. 121-156.

[4] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino. Closed Loop Steering of Unicyle-Like Vehicles via Lyapunov Techniques. IEEE Robotics and Automation Magazine, 1995.

[5] D. Soetanto, L. Lapierre, A. Pascoal, Nonsingular path-following control of dynamic wheeled robos with parametric modeling uncertanity. Proceedings of the 11th International Conference on Advanced Robotics, ICAR 2003, Coimbra, Portugal.

[6] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. of Robotics Research, vol. 5, no. 1, 1986.

[7] R. C. Arkin: Motor Schema-Based Mobile Robot Navigation. Int. J. of Robotic Researcg. 8(4): 92–112, 1989.

[8] Y. Koren and J. Borenstein, Potential Field Methods and Their Inherent Limitations for Mobile Rabat Navigation, Proc. IEEE Conf. on Robotics and Automation, Sacramento, CA, pp. 1398-1404, Apr. 7–12, 1991.

[9] J. Borenstein and Y. Koren, The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots, IEEE Journal of Robotics and Automation, Vo1.7, No.3, June 1991, pp.278–288.

[10] I. Ulrich and J. Borenstein, J., VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots, IEEE Int. Conf. on Robotics and Automation, May 1998, pp. 1572–1577.

[11] I. Ulrich and J. Borenstein, VFH*: Local Obstacle Avoidance with Look-Ahead Verification, IEEE Int. Conf on Robotics and Automation, April 2000, pp. 2505–2511.

[12] L. Linhui, Z. Mingheng, G. Lie, Z. Yibing, Stereo Vision Based Obstacle Avoidance Path-Planning for Cross-Country Intelligent Vehicle, Sixth Int.. Conf. on Fuzzy Systems and Knowledge Discovery, FSKD '09, pp. 463 – 467 2009.

[13] R. Kurozumi, T. Yamamoto, Implementation of an obstacle avoidance support system using adaptive and learning schemes on electric wheelchairs, 2005 IEEE/RSJ Int.. Conf. on Intelligent Robots and Systems, pp. 1108 – 1113, 2005.

[14] J. Minguez, The obstacle-restriction method for robot obstacle avoidance in difficult environments, 2005 IEEE/RSJ Int.. Conf. on Intelligent Robots and Systems. pp. 2284 - 2290, 2005.

[15] R. Simmons, The Curvature-Velocity Method for Local Obstacle Avoidance. In IEEE Int. Conf. on Robotics and Automation, pp. 3375-3382, Minneapolis, USA, 1996.

[16] F. Zhang, A. O'Connor, D. Luebke, P.S. Krishnaprasad, Experimental study of curvature-based control laws for obstacle avoidance, 2004 IEEE Int. Conf. on Robotics and Automation, pp. 3849 – 3854, 2004.

[17] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. IEEE Robotics and Automation Magazine, 4(1), 1997.

[18] P. Ogren, N.E. Leonard, A convergent dynamic window approach to obstacle avoidance, IEEE Transactions on Robotics, Volume: 21, Issue: 2, pp. 188 – 195, 2005.

[19] K.O. Arras, J. Persson, N. Tomatis, R. Siegwart, Real-time obstacle avoidance for polygonal robots with a reduced dynamic window, IEEE Int. Conf. on Robotics and Automation, pp: 3050 – 3055, 2002.

[20] O. Brock and O. Khatib, Elastic strips: a framework for motion generation in human environments, Int. J. of Robotics Research, vol. 21, no. 12, pp. 1031–1052, Dec. 2002.

[21] S. Quinlan and O. Khatib. Elastic Bands: Connecting Path Planning and Control. In IEEE Int. Conf. on Robotics and Automation, volume 2, pages 802807, Atlanta, USA, 1993.

[22] F. Lamiraux, D. Bonnafous, and O. Lefebvre, Reactive path deformation for nonholonomic mobile robots, IEEE Trans. on Robotics and Automation, vol. 20, no. 6, pp. 967–977, Dec. 2004.

[23] L. Lapierre, R. Zapata, P. Lepinay, Simultaneous Path Following and Obstacle Avoidance Control of a Unicycle-type Robot, 2007 IEEE Int. Conf. on Robotics and Automation, pp. 2617 - 2622, 2007.

[24] Y. Li, C. Li, R. Song, A new hybrid algorithm of dynamic obstacle avoidance based on dynamic rolling planning and RBFNN, IEEE Int. Conf. on Robotics and Biomimetics, ROBIO '07, pp. 2064 – 2068, 2007.

[25] Y. S. Nam, B. H. Lee, M. S. Kim, View-time based moving obstacle avoidance using stochastic prediction of obstacle motion, IEEE Int. Conf. on Robotics and Automation, pp. 1081 – 1086, 1996.

[26] Zhu, Q., Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation, IEEE Transactions on Robotics and Automation, Vol. 7 , No. 3, pp 390 – 397, 1991.

[27] C. Fulgenzi, A. Spalanzani, C. Laugier, Dynamic Obstacle Avoidance in uncertain environment combining PVOs and Occupancy Grid, 2007 IEEE Int. Conf. on Robotics and Automation, pp: 1610 - 1616, 2007.

[28] G. Bianco, P. Fiorini, Visual avoidance of moving obstacles based on vector field disturbances, IEEE Int. Conf. on Robotics and Automation, pp. 2704 - 2709, 2001.

[29] G. Zong, L. Deng, W. Wang, A Method for Robustness Improvement of Robot Obstacle Avoidance Algorithm, IEEE Int. Conf. on Robotics and Biomimetics, ROBIO'06. pp. 115 – 119, 2006.

[30] F.Mastrogiovanni, A. Sgorbissa, and R. Zaccaria, Robust navigation in an unknown environment with minimal sensing and representation, 2009 IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Volume 39, Issue 1, pp. 212–229, 2009.

[31] A. Sgorbissa and R. Zaccaria, A Minimalist Feedback Control for Path Tracking in Cartesian Space, 2009 IEEE/RSJ Int. Conf. on Intelligent RObots and Systems, October 11 – 15, 2009, St. Louis, MO, USA

[32] M. Piaggio and A. Sgorbissa, Ai-cart: an algorithm to incrementally calculate artificial potencial fields in real-time. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 238-243, Monterey, CA, 1998.

[33] A. Morro, A. Sgorbissa, and R. Zaccaria, Path Following for Unicycle Robots with Arbitrary Path Curvature, Technical Report, 2010. Available online: http://www.robotics.laboratorium.dist.unige.it/index.php?section=5