



**HAL**  
open science

## Safe proactive plans and their execution

K. Madhava Krishna, Rachid Alami, Thierry Simeon

► **To cite this version:**

K. Madhava Krishna, Rachid Alami, Thierry Simeon. Safe proactive plans and their execution. Robotics and Autonomous Systems, 2006, 54 (3), pp.244-255. hal-01976206

**HAL Id: hal-01976206**

**<https://laas.hal.science/hal-01976206>**

Submitted on 9 Jan 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Safe Proactive Plans and their Execution

K. Madhava Krishna<sup>a</sup>

R. Alami<sup>b</sup>

T. Simeon<sup>b</sup>

a: International Institute of Information Technology, Hyderabad - India

b: LAAS-CNRS, 31077 Toulouse Cedex - France

**Abstract:** We present in this paper a methodology for computing the maximum velocity profile over a trajectory planned for a mobile robot. Environment and robot dynamics as well as the constraints of the robot sensors determine the profile. The planned profile is indicative of maximum speeds that can be possessed by the robot along its path without colliding with any of the mobile objects that could intercept its future trajectory. The mobile objects could be arbitrary in number and the only information available regarding them is their maximum possible velocity. The velocity profile also enables to deform planned trajectories for better trajectory time. The methodology has been adopted for holonomic and non-holonomic motion planners. An extension of the approach to an online real-time scheme that modifies and adapts the path as well as velocities to changes in the environment such that both safety and execution time are not compromised is also presented for the holonomic case. Simulation and experimental results demonstrate the efficacy of this methodology.

## 1 Introduction

Several strategies exist for planning collision free paths in an environment whose model is known [9]. However during execution, parameters such as robot and environment dynamics, sensory capacities need to be incorporated for safe navigation. This is especially so if the robot navigates in an area where there are other mobile objects such as humans. For example in Figure 1, the robot would require to slow down as it approaches the doorway, in anticipation of mobile objects to emerge from there, even if it does not intend to make a turn through the doorway.

A possible means to tackle the above problem at the execution stage is to always navigate the robot at very low speeds. In fact, reactive schemes such as the nearness diagram approach [11] operate the robot at minimal velocities throughout the navigation. However incorporating the computation of a velocity profile at the planning stage would circumvent not only the problem of conservative velocities throughout navigation but would also allow for a modification of the trajectory to achieve lower time (Fig. 2).

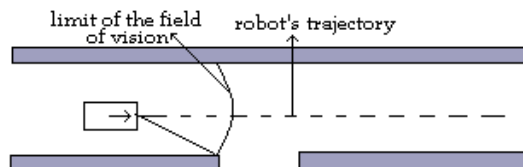


Figure 1: A safe robot has to slow down while approaching the doorway

We present in this paper a novel pro-active strategy that incorporates robot and environment dynamics as well as sensory constraints into a collision free motion plan. By pro-active we mean that the robot is always in a state of expectation regarding the possibility of a mobile object impinging onto its path from regions invisible to its sensor. This pro-active state is reflected in the velocity profile of the robot, which guarantees that in the worst case scenario, the robot will not collide with any of the moving objects that can interfere with its path. The ability of the algorithm to compute a-priori velocities for the entire trajectory accounting for moving objects moving in arbitrary directions is the essential novelty of this effort.



Figure 2: A longer path can be faster due to higher speed.

As is always the case, planned paths and profiles need constant modification at the execution stage due to changes in the environment. For example a profile and path that was planned for an environment with a closed doorway needs to be modified during real-time if the doorway is found open. Also addressed in this article the problem portrayed in Figure 3. Given an initial trajectory planned for a particular environment how does the robot modify its trajectory while new objects (not necessarily intersecting the robot's

trajectory) are introduced into the environment such that the basic philosophy of ensuring safety as well as reducing time lengths of the path continue to be respected. Simulation and experimental results are presented to indicate the efficacy of the scheme. In [1] we had reported how the maximum velocity profiles can be computed for any generic planner and in [8] we presented initial simulation and experimental results of the reactive version of [1].

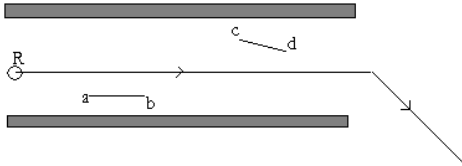


Figure 3: How does the robot adapt its path in the presence of new segments  $(a, b)$  and  $(c, d)$  while maintaining safe velocities?

Related work can be cited in the areas of modifying global plans using sensory data obtained during execution for overcoming uncertainty accumulated during motions [3] and those that try to bridge the gap between planning and uncertainty [10] or planning and control [7][2]. The velocity obstacle concept [13][5] bears resemblance to the current endeavor in that they involve selection of a robot velocity that avoids any number of moving objects. The difference is that in the present approach the only information about the mobile object available is the bound on velocity. The direction of motion and the actual velocities are not known during computation of the velocity profile. The work of Stachniss [14] also involves considering the robot's pose and velocities at the planning phase. A path is determined in the  $(x, y)$  space and a subgoal is chosen. A sequence of linear and angular velocities,  $(v, w)$ , is furnished till the subgoal is reached. In [12] a policy search approach is presented that projects a low dimensional intermediate plan to a higher dimensional space where the orientation and velocity are included. As a result better motion plans are generated that enable better execution of the plan by the robot. The current effort has similarities to [12], at the planning level but also extends it to a suitable reactive level in the presence of new obstacles encountered during execution. Similarly the dynamic window approach [16] and the global dynamic window method of Brock et.al. [17] both incorporate the dynamics and the kinematics of the robot for a reactive collision avoidance system. Incorporating the dynamics and searching in the space of velocities overcome the problems of purely geomet-

ric methods. However these methods do not speak of modifying the path in order to reduce its time-length and the dynamics of the environment does not affect the computation of the velocity profile, which places our approach as different from those mentioned above.

## 2 Problem Definition

The following problems are addressed in the paper, given:

- A robot  $\mathcal{R}$  modelled as a disc and equipped with an omnidirectional sensor having a limited range  $R_{vis}$ . We call  $\mathcal{C}_{vis}$  the visibility circle, centered at robot's position with radius  $R_{vis}$ . The paths of  $\mathcal{R}$  are sequences of straight segments or straight segments connected with circular arcs of radius  $\rho$  in case of a non-holonomic robot. The robot's motion is subject to dynamic constraints simply modelled by a bounded linear velocity  $v \in [0, v_{rm}]$  and a bounded acceleration  $a \in [-a_{-m}, a_m]$ . The maximum possible deceleration  $a_{-m}$  need not equal the maximum acceleration  $a_m$ .
- A workspace cluttered by static polygonal obstacles  $\mathcal{O}_i$ . The static obstacles can hide possible mobile objects whose motions are not predictable; the only information is their bounded velocity  $v_{ob}$ .

**Problem 1:** Given a robot's path  $\tau(s)$  computed by a standard planner [9], determine the maximal velocity profile  $v_\tau(s)$  such that, considering the constraints imposed by its dynamics, the robot can stop before collision occurs with any of the mobile objects that could emerge from regions not visible to the robot at position  $s \in \tau(s)$ . For example the velocity profile dictates that the robot in Figure 1 slow down near the doorway in expectation of mobile objects from the other side. We call  $MP = (\tau(s), v_\tau(s))$  a **robust motion plan**. The velocity profile allows us to define the time  $T(\tau)$  required for the robust execution of path  $\tau$ :

$$T(\tau) = \int_0^L \frac{ds}{v_\tau(s)}$$

**Problem 2:** Modify the planned trajectory such that the overall trajectory time  $T(\tau)$  is reduced. For example, the path of Figure 2, albeit longer than the one of Figure 1 is traversed in a shorter time.

**Problem 3** Adapt the path and velocities reactively in the presence of new objects not a part of the original workspace such that the criteria of safe velocities and reduced time of path continue to be respected. This is illustrated in Figure 3.

### 3 From path to robust motion plan

The procedure for computing the maximum velocity profile  $v_r(s)$  delineated in Sections 3.1, 3.2 and 3.3 addresses the first problem. The constraints imposed by the environment on the robot's velocity are due to two categories of mobile objects. The first category consists of mobile objects that could appear from anywhere outside the boundary of the visibility circle  $C_{vis}$ . The second category involves mobile objects that could emerge from shadows created in  $C_{vis}$  due to stationary objects.

#### 3.1 Velocity constraints due to the environment

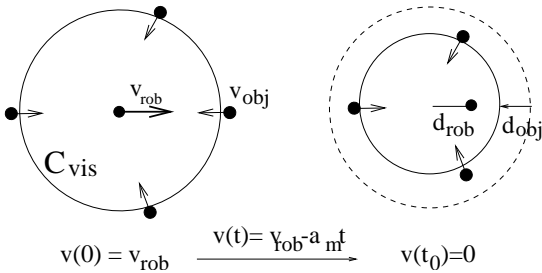


Figure 4: Mobile objects may appear anywhere on  $C_{vis}$ 's contour.

**No obstacles in  $C_{vis}$**  In the simple case where the robot's position is such that no static obstacle lies inside  $C_{vis}$ , a moving object may appear (at time  $t = 0$ ) anywhere on  $C_{vis}$ 's boundary (Fig. 4). Let  $V_{rb}$  denote the maximum possible robot velocity due to a mobile object at the boundary. At time  $t_0 = v_{rb}/a_{-m}$  (i.e., when the robot is stopped), the distance crossed by the object is  $d_{obj}(v_{rb}) \leq v_{ob}v_{rb}/a_{-m}$ . Avoiding any potential collision imposes that  $R_{vis} \geq d_{rb}(v_{rb}) + d_{obj}(v_{rb})$ , where  $d_{rb} = v_{rb}^2/2a_{-m}$ . The condition relates  $v_{rb}$  to the sensor's range  $R_{vis}$  as:

$$v_{rb} = -v_{ob} + \sqrt{v_{ob}^2 + 2a_{-m}R_{vis}} \quad (1)$$

**Influence of shadowing corners** Static obstacles lying inside  $C_{vis}$  may create shadows (e.g., see the grey region of Figure 5) which contain mobile objects. The worst-case situation occurs when the mobile object remains unseen until it arrives at the *shadowing corner* of a polygonal obstacle. Since the mobile object's motion direction is not known it is best modeled for a worst case scenario as an expanding circular wave of radius  $v_{ob}t$  centered at  $(d, \theta)$

$$(X(t) - d \cos \theta)^2 + (Y(t) - d \sin \theta)^2 = v_{ob}^2 t^2$$

Let us first consider that the robot's path  $\tau$  is a straight segment. Considering that the intersections between the circular wave and the robot's segment path, should never reach the robot before it stops at time  $t_0 = v_{rs}/a_{-m}$  yields the following velocity constraint:

$$v_{rsv}^4 - 4(a_{-m}d \cos \theta + v_{ob}^2)v_{rsv}^2 + 4a_{-m}^2d^2 \geq 0 \quad (2)$$

Here  $v_{rsv}$  is the maximum possible robot velocity due to the shadowing vertex under consideration. The solution of Eq. 2 gives  $v_{rsv}$ , as a function of  $(d, \theta)$ .

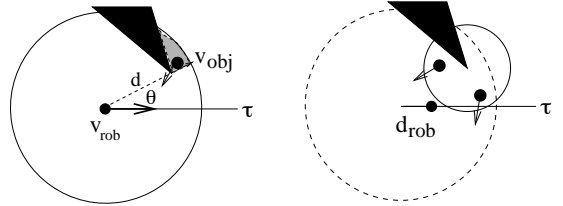


Figure 5: Mobile objects may also appear from the shadows of static obstacles

This solution only exists under the condition  $v_{ob} > \sqrt{a_{-m}d(1 - \cos \theta)}$ , i.e., when the object's velocity  $v_{ob}$  is sufficiently high to interfere with the robot's halting path. Otherwise, the shadowing corner does not constrain the robot's velocity which can be set to  $v_{rm}$ , the maximum bound on robot's velocity.

Similar reasoning can be applied to the case where the robot traverses a circular arc path of radius  $\rho$ . This case however leads to a nonlinear equation that needs to be solved numerically to derive the maximal velocity [4]. The expression that needs to be solved for computing the maximum velocity at a given point on a circular arc is of the form :

$$\begin{aligned} & ((v_{rsv}^2 v_{ob}^2)/a_{-m}^2) + 2\rho^2 \cos(v_{ob}^2/2a_{-m}\rho) + \\ & 2d\rho \sin((v_{ob}^2/2a_{-m}\rho) - \theta) \\ & = d^2 + 2\rho^2 - 2d\rho \sin \theta \end{aligned} \quad (3)$$

#### 3.2 Computing the shadowing corners

The problem of determining the set of shadowing corners needed for the velocity computation in Section 3.1 is the problem of extracting those vertices of the polygonal obstacle to which a ray emitted from the robot's center is tangential (Figure 6). The set of shadowing corners can be easily extracted from an algorithm that outputs the visibility polygon [15] as a sorted list of vertices.

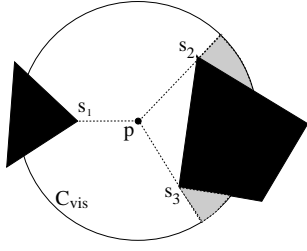


Figure 6: Shadowing corners: among the three vertices of  $\mathcal{V}(p)$ , only  $s_2$  and  $s_3$  create shadows (the line going through  $s_1$  is not tangent to the left obstacle).

### 3.3 Computing the velocity profile $v_\tau(s)$

While the methodology for computing the maximum velocity profile delineated here is essentially for a holonomic path, its extension to the non-holonomic case is not very difficult.

1. A holonomic path  $\tau$ , consisting of a sequence of straight line segments  $ab, bc, cd$  (Fig. 7) is deformed into a sequence of straight lines and clothoids to ensure continuity of velocities at the bends [6]. The maximum deviation from an endpoint to its clothoidal arc (depicted as  $e$  in Figure 7) is dependent on the nearest distance to an object from the endpoint under consideration.

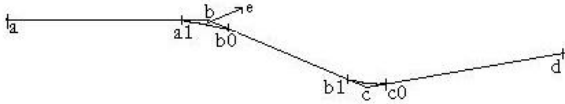


Figure 7: A holonomic path deformed into a sequence of straight segments and clothoidal arcs.

2. The linear velocity along a clothoid is a constant and the maximum possible linear velocity considering robot dynamics alone is calculated for each of the clothoidal arc  $a1b0, b0c0$  according to [6] and is represented as  $v_c(a1), v_c(b0)$ .
3. The straight segment  $aa1$  is discretized into  $M$  equally spaced points, excluding the endpoints of the segment, viz.  $a$  and  $a1$ . We denote the first such point as  $a_1$  and the last such point as  $a_M$ . The point of entry into the clothoid, viz.  $a1$  is also denoted as  $a_{M+1}$ .
4. For each of the  $N$  points,  $a_i$ , the steps 4a to 4e are repeated.
  - 4a Maximum possible velocity that a robot could have such that it can come to a halt before colliding with objects that enter into the robot's field of

vision from the boundary is computed as  $v_{rb}(a_i)$  according to Equation 1.

- 4b Velocity of the robot due to stationary obstacles inside the robot's field of vision that create shadows is computed as  $v_{rsv}(a_i)$  according to Equation 2. The minimum of all the velocities due to such vertices is found and denoted as  $v_{rs}(a_i)$ .
- 4c The maximum possible velocity of the robot at  $a_i$  due to environment is then computed as

$$v_{re}(a_i) = \min(v_{rb}(a_i), v_{rs}(a_i)) \quad (4)$$

- 4d Velocity of the robot at  $a_i$  due to its own dynamics is given by

$$v_{rd}(a_i) = \sqrt{v_r^2(a_{i-1}) + 2a_m s(a_i, a_{i-1})} \quad (5)$$

The above equation is computed if  $v_{re}(a_i) > v_r(a_{i-1})$ . Here  $s(a_i, a_{i-1})$  represents the distance between the points  $a_i$  and  $a_{i-1}$ .  $a_m$  represents the maximum acceleration of the robot.

- 4e The eventual velocity at  $a_i$  is given by

$$v_r(a_i) = \min(v_{rd}(a_i), v_{re}(a_i), v_{rm}) \quad (6)$$

Here  $v_{rm}$  represents the maximum robot velocity permissible due to servo motor constants.

5. The velocity at the endpoint  $a1$  is computed as  $v_r(a1) = \min(v_r(a1), v_c(a1))$  and this would be the linear velocity with which the robot would traverse the clothoid.
6. Steps 6a and 6b are performed by going backwards on each of the  $N$  points from  $a_N$  to  $a_1$ .
  - 6a If  $v_r(a_i) > v_r(a_{i+1})$  then the modified maximum possible velocity at  $a_i$  is computed as

$$v_{rd}(a_i) = \sqrt{v_r^2(a_{i+1}) + 2a_{-m} s(a_i, a_{i+1})} \quad (7)$$

- 6b Finally the maximum safe velocity at  $a_i$  is given as  $v_r(a_i) = \min(v_r(a_i), v_{rd}(a_i))$ .
7. Repeat steps 3 to 6 for all the remaining straight segments to obtain the maximal velocity profile over a given trajectory  $\tau$  as  $v_\tau(s) = \{v_r(a), v_r(a_1), \dots, v_r(a_N), v_r(a1), v_r(b1), \dots, v_r(d)\}$ .

### 3.4 Modifying planned trajectory for lower time

The knowledge of the maximum velocity profile over a trajectory is utilized to tackle the problem posed in Section 2 of reducing the overall trajectory time of the path. The procedure for reducing trajectory time at the planning stage involves random deformation of the planned path and evaluating time along this path. The modified path becomes the new trajectory if time

along it is less than along the original trajectory. The process is continued till over a finite number of attempts no further minimization of trajectory time is possible. Prior to delineating the algorithm it is to be noted that the set of all collision free space of the workspace is denoted as  $C_{free}$  and the current trajectory of the robot as  $\tau_c(s)$ . A point of discretization on a trajectory discretized into  $N$  parts is denoted as  $p(s_i), i \in \{1, 2, \dots, N\}$ . The corresponding configuration of the robot at those points is denoted by  $q(s_i)$ . The algorithm is given as Algorithm 1.

---

**Algorithm 1** Globally reducing trajectory time

---

```

1:  $N_{try} \leftarrow 0$ 
2: while  $N_{try} < N_{attempts}$  do
3:   Discretize current trajectory  $\tau_c(s)$  into  $N_p$  parts
   where  $N_p$  is selected based on minimum discretization
   distance between two points.
4:   Set  $flag \leftarrow 0$ 
5:   for  $i = 1$  to  $N_p$  do
6:     Compute minimum velocity at  $s_i$  due to shadowing
     vertices as  $v_{rmin}(s_i)$ 
7:     if  $v_{rmin}(s_i) < v_{rm}$  then
8:       Find a configuration  $q(s_p) \in C_{free}$  and
        $s_p \notin \tau_c(s_k), k \in \{1, \dots, N_p\}$  such that
        $q(s_p)$  is reachable from  $q(s_i)$ .
9:       Find a point  $s_r$  on the remaining part of the
       trajectory,  $s_r \in \tau_c(s_j); i < j \leq N_p$  such
       that  $q(s_r)$  is reachable from  $q(s_p)$ .
10:      Form a new trajectory through  $s_i, s_p, s_q$ 
      and denote it as  $\tau_n(s)$ 
11:      if  $T(\tau_n) < T(\tau_c)$  then
12:        discretize  $\tau_n$  into  $N_q$  points.
13:         $\tau_c \leftarrow \tau_n$ 
14:         $N_p \leftarrow N_q$ 
15:        Set  $flag \leftarrow 1$ 
16:      end if
17:    end if
18:  end for
19:  if  $flag = 0$  then
20:     $N_{try} \leftarrow N_{try} + 1$ 
21:  end if
22: end while

```

---

Step 8 of the algorithm is carried out by searching for a collision free configuration which would displace the path away from the shadowing vertex responsible for the lowest velocity at  $s_i$ . Step 11 adapts the displaced path as the new current path if its trajectory time is less than the current path.  $N_{attempts}$ , is the number of unsuccessful attempts at minimizing trajectory time before the algorithm halts.

### 3.5 Remembering Sensor Information

The computation of the velocity profile at a given point on the robot's trajectory incorporates the robot's field of vision at that point. This field can change appreciably between two successive instances of computation. For example in Figure 8 the robot at position  $a$  has full field of vision of the corridor that is transverse to the robot's trajectory. However at position  $b$  the robot is blind to the zone shown in darker shade of gray. Hence it needs to slow down as it moves further down to  $c$  since it envisages the possibility of a moving object approaching it from the corners of the stationary objects. These corners are the starting areas of the robot's blindzone at  $b$ .

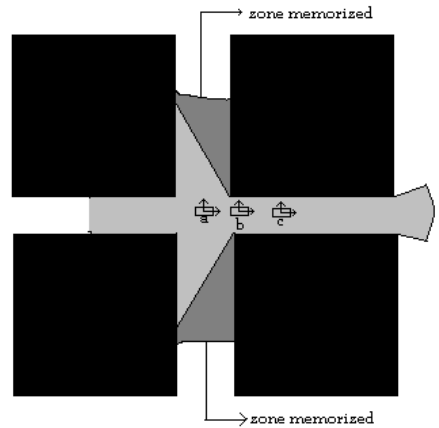


Figure 8: Memorization of previous scenes

However, if the robot could remember the earlier scene it could use this when computing its velocity profile during execution of the planned path. In such a case, if the robot did not see any moving objects in close proximity at  $a$  it can make use of this information at  $b$  to have a velocity profile from  $b$  that is greater than the one computed in the absence of such information. Figure 8 shows (in darker shade) the zone remembered by the robot. The contour of the remembered area represents the blindzone of the robot at  $b$ , from where mobile objects can emanate. The area in lighter shade of gray is the visibility polygon for the robot at  $b$ . With the passage of time the frontier of the remembered area shrinks due to the advancement of the imagined mobile objects from the initial frontier. The details of this scheme are given below.

Remembering is fruitful when a non-shadowing vertex begins to cast a shadow thereby hiding regions which were previously visible. The set of all ver-

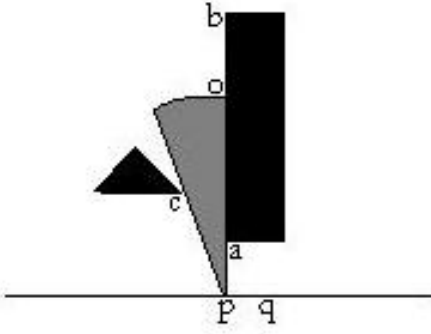


Figure 9: Three categories of blind vertices.

tices that are currently visible, shadowing and were at some prior instant visible, non-shadowing is denoted by  $Vsns$ . For every vertex  $ve \in Vsns$  a corresponding vertex is associated and called the blind vertex. The blind vertices are of three categories explained in Figure 9 where the vertex  $a$ , non-shadowing for the robot at  $p$  becomes shadowing when the robot is at  $q$ . Correspondingly the vertex  $c$  of the triangular obstacle which was visible and shadowing when the robot was at  $p$  becomes invisible when the robot moves to  $q$ . Simultaneously one of the other end-points of  $a$ , viz.  $b$ , would also become inevitably invisible at  $q$ . Vertices like  $b$  fall in the second category. If  $b$  was already outside  $C_{vis}$  at  $p$  the intersection of  $C_{vis}$  with the segment  $ab$ , namely  $o$  is identified as the third category of blind vertex. The set of all such vertices is denoted by  $Vbs$ . These vertices are advanced by a distance  $v_{ob}\Delta t$  where  $\Delta t$  is the time taken by the robot between  $p$  and  $q$  to new virtual locations along the line that connects those vertices to  $a$ . At  $q$  the velocity is computed due to the closest of the vertices in the set  $Vbs$  at their virtual locations instead of  $a$ , which is otherwise the vertex for which equation(2) is computed.. Such a trend continues till the distance between the robot to the closest hypothetical vertex is less than the actual distance of the robot to  $a$ .

The remembering part of the algorithm is given in algorithm 2. The set of all visible shadowing vertices is denoted by  $Vsh$ .

#### 4 From Plan to Execution

The velocity profile,  $v_\tau(s)$ , is a sequence of maximum velocities calculated at discretized locations along the trajectory  $\tau(s)$ . The locations at which the velocity profile at the execution stage is computed are not the same locations as where the profile was computed during planning, due to odometric and motor

---

#### Algorithm 2 Remembering effects on velocity

---

```

1: for each vertex  $v_e \in Vsh$  do
2:   if  $v_e \in Vsns$  then
3:     for each vertex  $v_b \in Vbs$  associated with  $v_e$  do
4:       Advance  $v_b$  by  $v_{ob}\Delta t$ 
5:     end for
6:     Denote the distance from the robot's current location,  $s_c$ , to the closest of all advanced vertices,  $v_{bc}$  as  $d_{cvb}$ 
7:     if  $d(s_c, v_e) < d_{cvb}$  then
8:       Compute velocity due to the virtual vertex  $v_{bc}$  through Equation 2
9:     else
10:      Compute velocity due to the actual vertex  $v_e$  through Equation 2
11:    end if
12:  end if
13: end for

```

---

constraints. Moreover, if there are changes in the environment it entails modifying the trajectory and hence the velocities. During execution it is computationally expensive to compute the profile for the entire remaining trajectory, hence the profile is computed for the next finite distance, given by,  $d_{safe} = d_{max} + nd_{samp}$ , where  $d_{max} = v_{rm}^2 / (2 * a_{-m})$ , represents the distance required by the robot to come to a halt while it moves with the maximum permissible velocity afforded by motor constants. And  $d_{samp} = v_{rm}t_{samp}$  is the maximum possible distance that the robot can move between two successive samples (time instants) of transmitting motion commands, where time between two samples is  $t_{samp}$ .

The main issue here is what should be the distance over which the velocity profile needs to be computed during execution such that it is safe. A velocity command is **not** considered safe if it is less than the current velocity and **not** attainable within the next sample. The velocity is constrained by the environment as well as robot's own dynamics and hence their roles are studied below.

**Effect of Environment** Mobile objects that can emerge from corners in a head-on direction cause the greatest change in velocity over two samples. Figure 10 shows one such situation, where the rectangular object casts a shadow and is susceptible to hide mobile objects. Let the current velocity of the robot at  $a$  due to the object be  $v_1$ . Let the velocity at a distance,  $s$ , from  $a$ , at  $b$  (Fig. 10) due to the object be  $v_2$ .

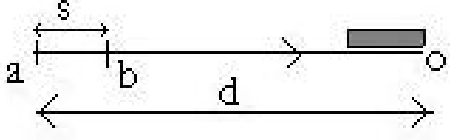


Figure 10: Effect of an obstacle on the robot's velocity, possibly hiding mobile objects at locations  $a$  and  $b$ .

The velocities at  $a$  and  $b$  are given by

$$v_a = -v_{ob} + \sqrt{v_{ob}^2 + 2a_{-m}d} \quad (8)$$

$$v_b = -v_{ob} + \sqrt{v_{ob}^2 + 2a_{-m}(d-s)} \quad (9)$$

Hence

$$v_a^2 - v_b^2 = 2a_{-m}s + 2v_{ob}(\sqrt{v_{ob}^2 + 2a_{-m}(d-s)} - \sqrt{v_{ob}^2 + 2a_{-m}d}) \quad (10)$$

Evidently the second term on the right hand side of Equation 10 is negative, since the second square root term is more positive than the first. Hence  $v_a^2 - v_b^2 \leq 2a_{-m}s$ . Therefore the velocity at  $b$ ,  $v_b$  can be attained from the velocity at  $a$ ,  $v_a$  under maximum deceleration,  $d_m$ , irrespective of the maximum velocity of the mobile object or the robot's own motor constraints. This was intuitively expected since the robot's velocity at any location is the maximum possible velocity that guarantees immobility before collision; its velocity at a subsequent location permitted by the environment would be greater than or equal to the velocity at the same location obtained under maximum deceleration from the previous location. In other words for safeness of velocity going purely by environmental considerations it would suffice to calculate the velocity, for the next sampling distance alone, for without loss of generality,  $d = d_{samp}$ .

**Effect of robot's dynamics** The robot needs to respect the velocity constraints imposed while nearing the clothoidal arcs and eventually while coming to the target. The robot can reach zero velocity from its maximum velocity over a distance of  $d_{max}$ , computed before. Hence  $d_{max} + d_{samp}$  represents the safe distance over which the velocities need to be computed.

#### 4.1 Online path adaptation for better trajectory time

The third of the problems outlined in Section 2 is tackled here. During navigation the robot in general comes across objects hitherto not a part of the map.

The robot reacts to these new objects in line with the basic philosophy of safety as well as time reduced paths. The adaptation proceeds by finding locations over a finite portion of the future trajectory where drops in velocity occur and pushing the trajectory away from those vertices of the objects that caused these drops to areas in free space where higher velocities are possible. A search is made through the newly found locations of higher velocities for a time reduced path.

**Generalized Procedure** The generalized procedure for adapting the path in the presence of new objects is delineated through Figure 11.

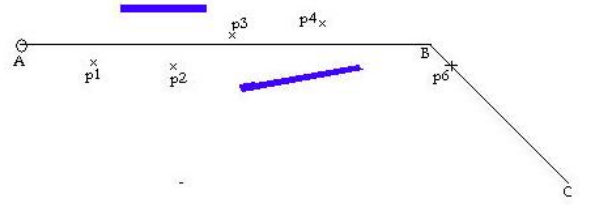


Figure 11: A trajectory in the presence of new objects. The points marked with crosses represent locations through which a path is searched for reduced time of trajectory.

1. On the trajectory segment that is currently traversed,  $AB$  in Figure 11, enumerate the vertices of objects that reduce the velocity of the robot.
2. The positions are found on  $AB$  where the influence of vertices is likely to be maximal.
3. These positions are pushed by distances  $d_p = k(v_l - v_r)$ , where  $v_l$  and  $v_r$  are the velocities at that location on the path due to the most influential vertices on the left and right of the path. These new locations are denoted as  $p1, p2, p3, p4$  (Fig.11) and maintained as a list provided the velocity at the new locations is higher than the original ones.  $p6$  is the farthest point on the robot's trajectory visible from its current location at  $A$ .
4. On this set of locations  $A, p1, p2, p3, p4, p5, p6$  starting from the current location at  $A$ , find a trajectory sequence shorter in time than the current sequence of  $A, B, p6$  if it exists.
5. The steps 1 to 4 are repeated until the robot reaches the target.

It should be noted that when a collision with an object is detected, a collision free location is first found that connects the current location with another location on the original trajectory and this new collision



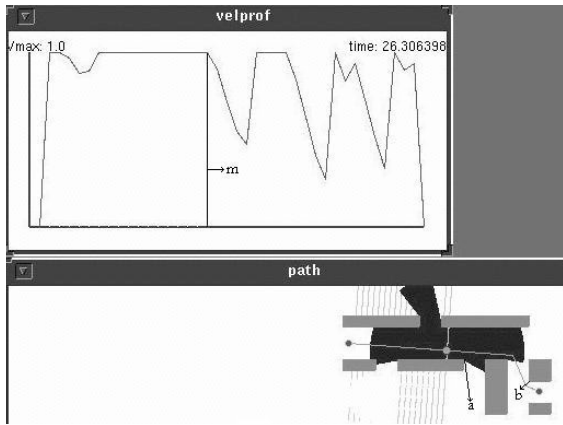


Figure 12: Path computed by a typical planner and its velocity profile shown on the top. The robot's velocity corresponding to its location on the trajectory is shown by a vertical line on the profile and labeled as  $m$ .

free path is further adapted for a time-reduced path if it exists. Also note that while the velocities are computed over a distance  $d_{safe}$ , that part of the remaining trajectory that is visible from the current location is considered for adapting to a better time-length.

## 5 Planning Results and Analysis

In this section the results of incorporating the velocity profile computation as a consequence of considering robot and environment dynamics and sensor capacities at the planning stage and the subsequent adaptation of paths to better time of trajectory is analyzed. Figure 12 shows the path computed by a typical holonomic planner [9] and its corresponding velocity profile. The velocity corresponding to the robot's location on the trajectory (shown as a small circle) is marked by a straight line labeled  $m$  on the profile. The dark star-shaped polygon centered at the robot depicts the visibility of the robot at that instant and is called the visibility polygon. The figure is a snapshot of the instant when the robot begins to decelerate to a velocity less than half the current velocity as it closes down on the vertex  $a$  marked in the figure. Evidently from the visibility polygon the vertex  $a$  casts a shadow and the closer the robot gets to it, the slower the velocity must be.

Figure 13 is the time reduced counterpart of Figure 12. The snapshot is once again at a location close to vertex  $a$ . Staying away from  $a$  permits nearly maximum velocity. The dip observed in the profile due to vertex  $a$  is negligible. Similarly staying away from other vertices such as  $b$  allows for a trajectory time of 21.79s compared to 26.30s for Figure 12. Mod-

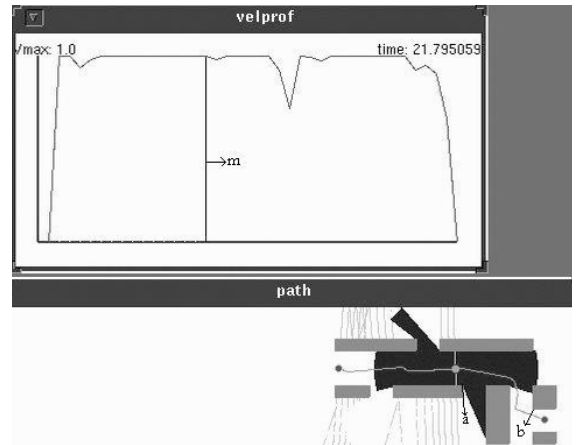


Figure 13: Path obtained after adaptation to reduced time-length.

ification of the trajectory for shorter time proceeds along the lines of Section 3.4. For the two examples discussed, the robot's maximum acceleration and deceleration was fixed at  $1m/s^2$ , maximum velocity at  $1m/s$  and the sensor range at  $7m$ . The maximum bound on the object's velocities was  $1.5m/s$ .

Figures 14 and 15 depict the planned trajectory and velocity profiles before and after reduction of trajectory time for our laboratory environment. The time reduced trajectory is shorter by more than 8 seconds as it widens its field of view by moving away from the bends while turning around them.

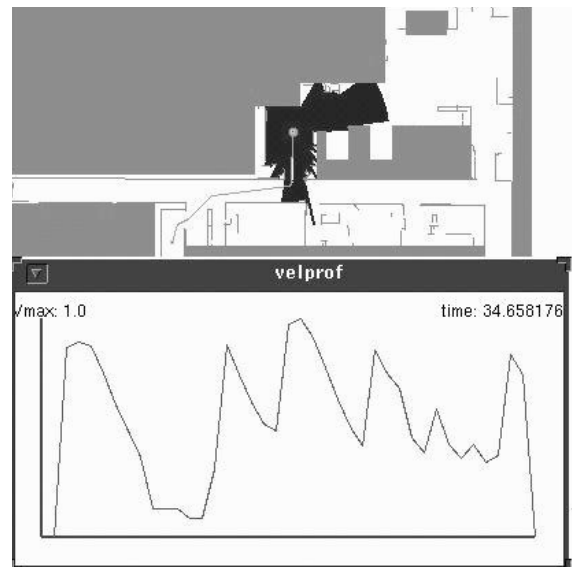


Figure 14: Planned trajectory before adaptation to a reduced time.

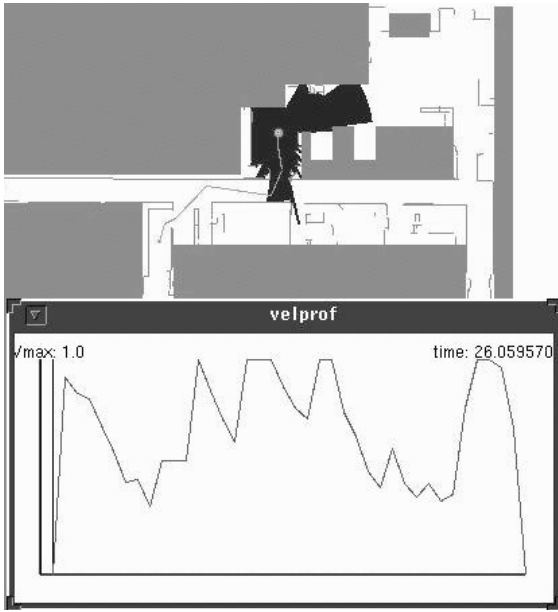


Figure 15: Time reduced trajectory at planning stage.

### 5.1 Effect of remembering on trajectory time

Figure 16 shows an environment with four corridors named 1, 2, 3 and 4 with planned path obtained by minimizing time. It also portrays the robot's field of vision as it enters corridor 3. The velocity profile for the above path is shown in Figure 17. The location of the robot corresponding to its location in Figure 16 is shown through the vertical line. The locations of the robot as it decelerates when its field of view of each of the corridors vanishes is also marked with the respective numbers on the profile.

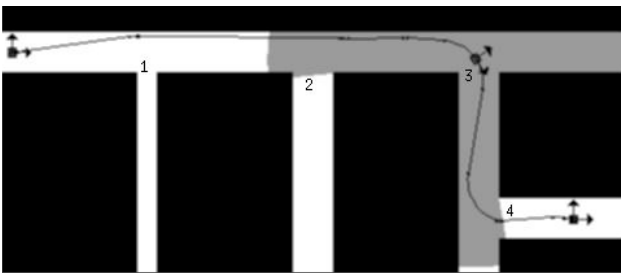


Figure 16: Robot's field of view as it enters corridor 3.

Though the path of Figure 16 is minimized in time its velocity profile still shows decelerations in the vicinity of the corridors. This is due to the phenomenon discussed in Section 3.5 where the robot becomes blind to many parts of the environment it had seen at the preceding instant. Figure 18 shows the robot's field

of vision at an instant after the one shown in figure 16. There is a marked decrease in its field of vision at the latter instant that results in the robot reducing its velocity in anticipation of moving objects from the blindzones depicted in the velocity profile.

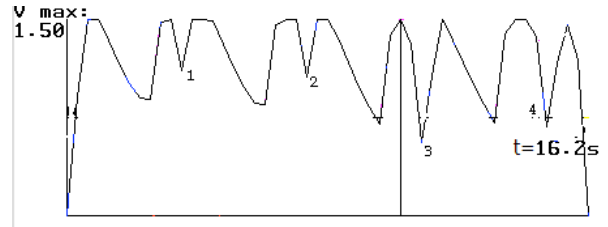


Figure 17: Velocity profile for the Figure 16. Corresponding position of the robot shown in vertical line. Decelerations near the corridors are also marked with the same numbers.

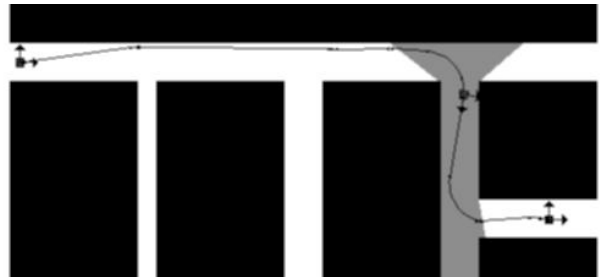


Figure 18: Robot's field of vision at an instant that immediately follows the instance of Figure 16.

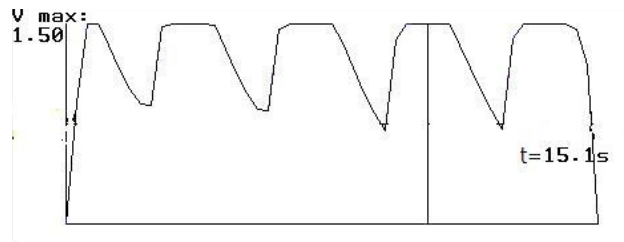


Figure 19: Velocity profile obtained after incorporation of memory

However, when the robot is able to remember the previous images, the need to decelerate is nullified and the trajectory time is further reduced. Figure 19 illustrates this where the decelerations shown in the velocity profile of Figure 17 at locations 1, 2, 3 and 4 are now absent.

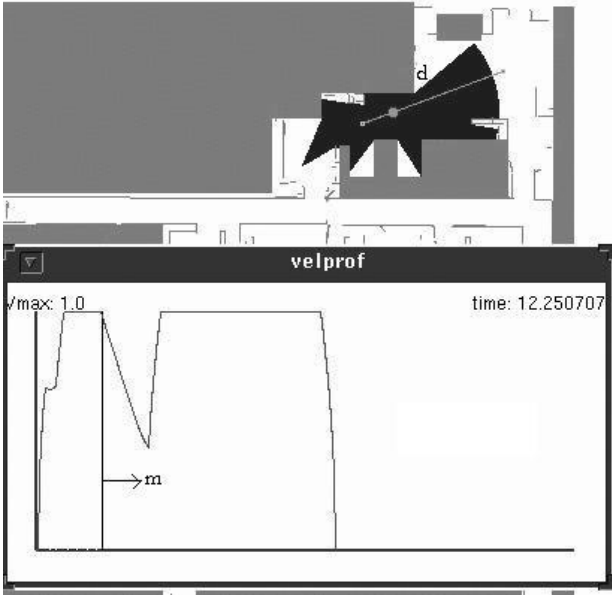


Figure 20: A simple planned trajectory and its velocity profile.

## 6 Experimental Results

### 6.1 Velocity profiles

In this section the velocity profiles obtained during the planning and execution stages are compared in the absence of any new objects during execution. Figure 20 shows a simple planned trajectory and the corresponding velocity profile for our lab environment. Some of the obstacles are filled in gray and others are shown as segments (in gray). The robot is shown as a small circle and the star shaped polygon in black represents the field of vision of the robot at that location. The vertical line, marked  $m$  in the velocity profile represents the velocity of the robot corresponding to its position on the trajectory. The profile shows a subsequent drop in velocity, a consequent of robot getting closer to region marked,  $d$ , to which it is blind.

Figure 21 compares the planned and executed (in simulation) velocity profile. The executed trajectory tallied to a time of  $12.28s$  in comparison with  $12.25s$  for the planned profile. These figures illustrate that the executed profiles and execution times are close to the planned profiles and times while there are no changes in the environment.

Figures 23 and 24 show the execution by the Nomad XR4000 (Fig. 22) of paths computed by a standard planner. Figure 23 corresponds to the original path computed by the planner and Figure 24 is its time reduced counterpart.

The velocity profiles during execution of the two

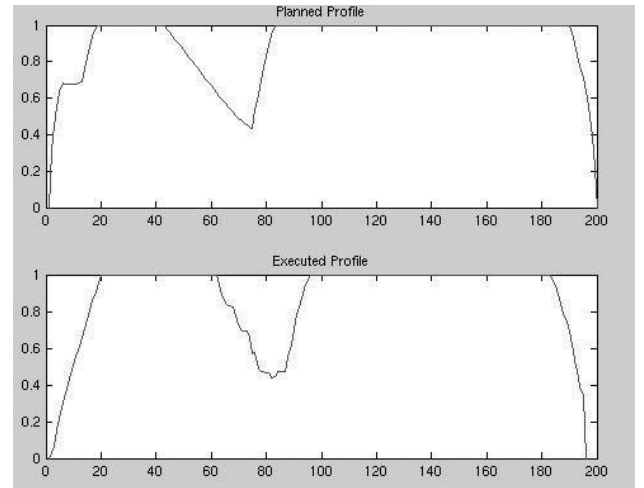


Figure 21: The planned and executed velocity profile in simulation. The ordinate measures velocity in  $m/s$  and abscissa time in seconds.



Figure 22: The Nomad XR4000 used in our experiments at LAAS.

paths are shown in Figure 25. Some of the bigger drops in the unreduced profile are absent in the reduced profile as the robot avoids turning close to the obstacles that form the bends. The path of Figure 24 got executed in  $12.9s$  while the path in Figure 23 was executed in  $13.98s$ . The Figures are meant as illustrations of the theme that trajectories deformed to shorter time-lengths at planning stage are also executed in shorter time during implementation than their unreduced versions.

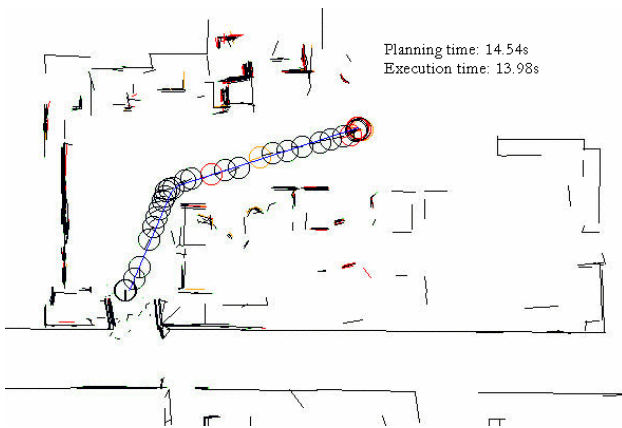


Figure 23: Execution of the original planned path.

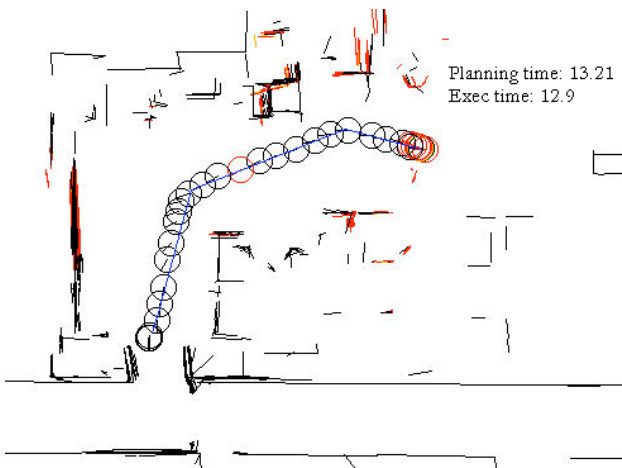


Figure 24: Execution of the time-reduced path.

## 6.2 Online adaptation of paths for better trajectory time

This section presents results of the algorithm in the presence of newly added objects that affect the velocities of the robot in real-time. Figure 26 shows a path where the robot avoids the two new segments  $S1$  and  $S2$  intersecting the original planned trajectory but does not adapt its path for better time. The velocity profile for the same is shown in Figure 27. Figure 28 is the counterpart of Figure 29 where the robot adapts its path to a better time-length reactively. The big dips in the velocity profile of Figure 27 are filtered in Figure 29 considerably as the robot avoids the obstacles with larger separation. The time reduced execution tallied to 10.9s while the unreduced version was executed in 12.5s. The trajectory time at planning was 7.9s. The above graphs are those obtained in simulation.

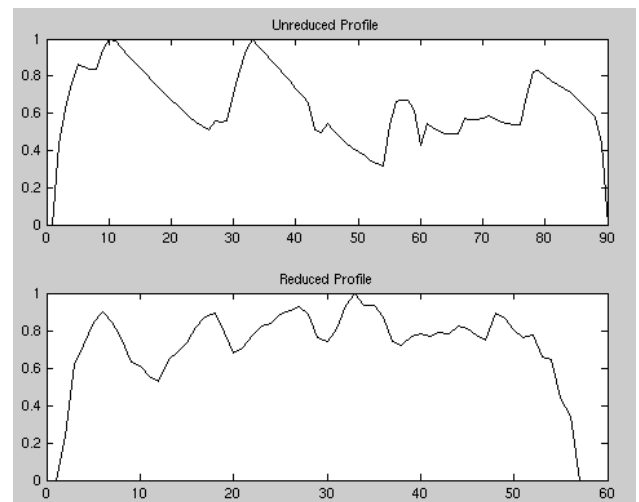


Figure 25: The top profile corresponds to the path executed in Figure 23 and the bottom to Figure 24. The planned and executed velocity profile in simulation. The ordinate measures velocity in  $m/s$  and abscissa time in seconds.

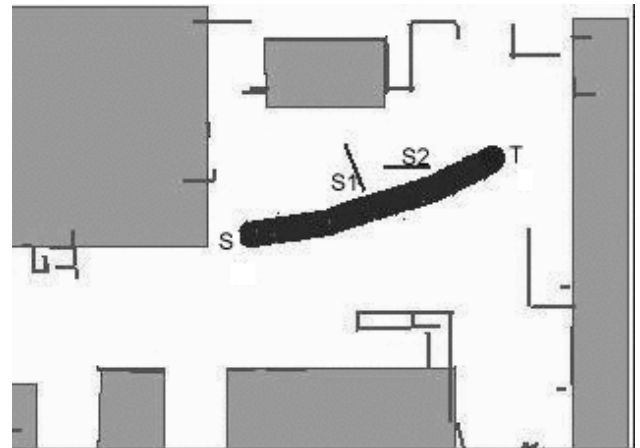


Figure 26: A simulated execution in the presence of two new segments  $S1$  and  $S2$  along with the corresponding velocity profile. The path is not adapted to better time-length. Start and goal locations marked as  $S$  and  $T$ .

Figure 30 shows the unreduced executed path by the XR4000 Nomadic robot in our laboratory at LAAS. The obstacles in the original map are shown by black lines, while the segments perceived by the SICK laser are shown in lighter shades of gray. Some of these segments get mapped to the ones in the map and the others are considered new segments. This is done by a segment based localization algorithm. The segments of concern here are those which form a box shaped obstacle marked  $B$  in Figure 30. The vertex  $d$  of this

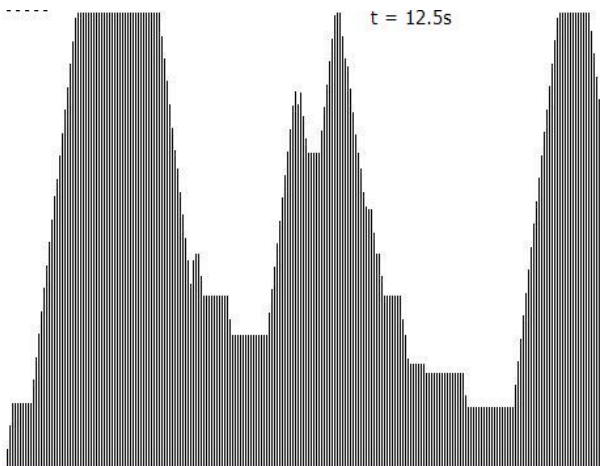


Figure 27: Velocity profile for the execution of Figure 26.

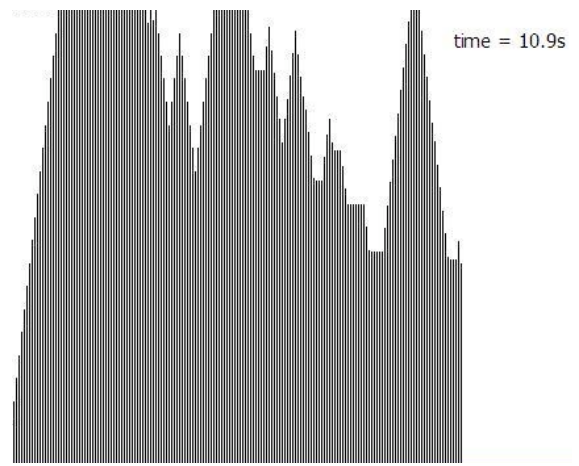


Figure 29: Velocity profile for the execution of Figure 28.

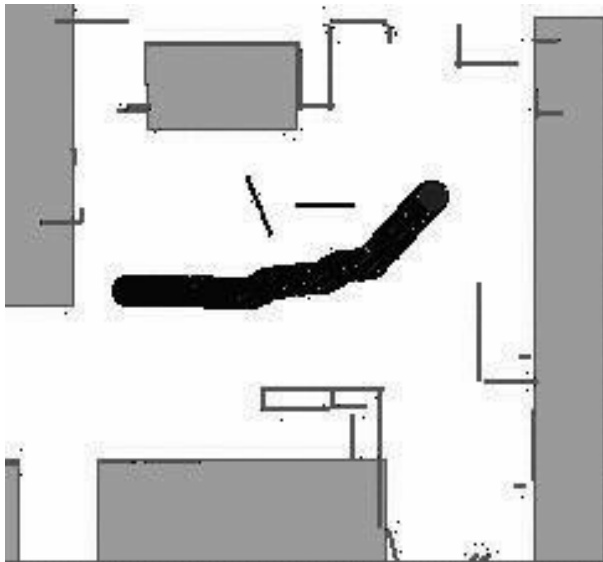


Figure 28: Path of Figure 26 adapted to better time-length.

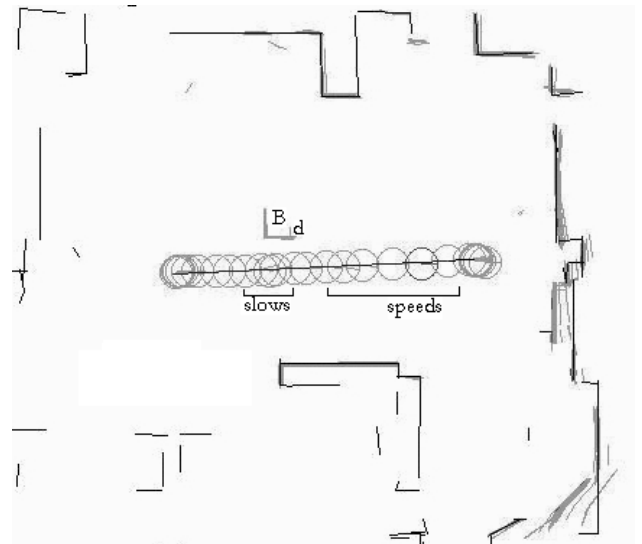


Figure 30: Unreduced path executed by the Nomad XR4000. The vertex  $d$  of the new box shaped object  $B$  forces a slow down near it.

obstacle casts a shadow on the robot's sensory field, which forces it to slow down at those locations due to Equation 2. The execution time for this unreduced path is  $10.6s$ .

The time reduced counterpart is shown in Figure 31 that tallied to  $9.6s$ . The original planning time was  $8.8s$  in the absence of the box shaped object. The corresponding velocity profile is shown in figure 32.

## 7 Conclusions and Scope

A proactive safe planning algorithm and its reactive version that facilitates real-time execution has been presented. The proactive nature of the algo-

rithm stems from the computed velocity profile,  $v_r(s)$ , that guarantees immobility of the robot before collision with any of the possible mobiles that could interfere its future trajectory from regions blind to its sensor. The proactivity does not however come at the cost of robot's velocity or trajectory time. The knowledge of  $v_r(s)$  computed over the trajectory  $\tau(s)$  further facilitates reduction of the over all trajectory time  $T(\tau)$  by adaptation of the initially planned path. Analysis of the scheme at the planning stage depict that the robot can have a velocity profile that achieves its maximum possible velocity for a sustained duration without many dips provided it stays away from doorways

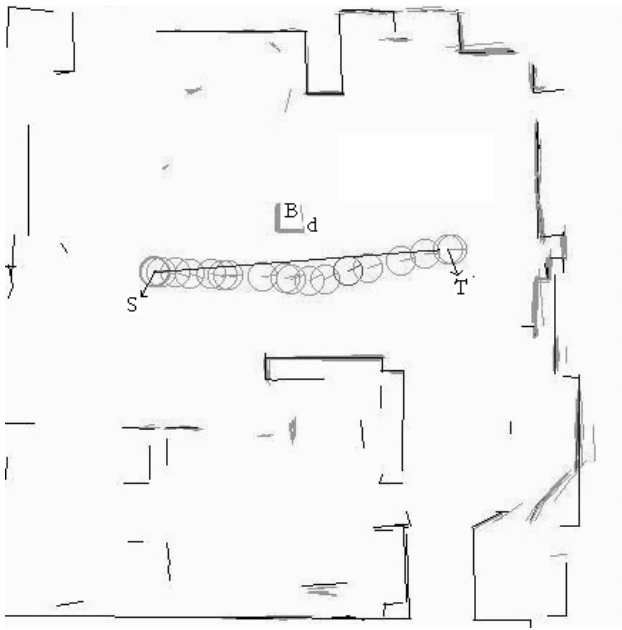


Figure 31: Time reduced path executed by the Nomad XR4000. Increasing linear and angular separation from vertex  $d$  facilitates a higher speed.

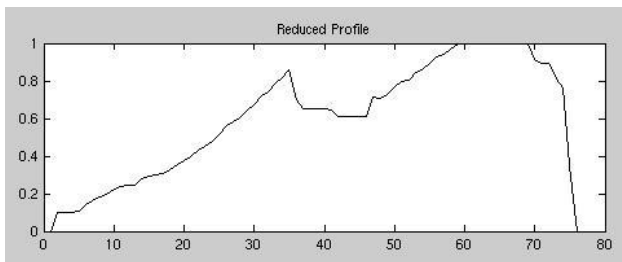


Figure 32: Velocity profile for the path executed by the Nomad in Figure 31. The planned and executed velocity profile in simulation. The ordinate measures velocity in  $m/s$  and abscissa time in seconds.

and narrow passages along its path. Remembering of previous scenes also enhances the robot's performance through reduced trajectory time and a more uniform velocity profile.

A reactive extension of the scheme that facilitates real-time simulation and implementation is also presented. The scheme maintains the underlying philosophy of computing safe velocities and modification of paths for better trajectory time. Simulation and experimental results at real-time corroborate our earlier results obtained at the planning stage (that by keeping away from vertices of objects that could hide mobiles the robot could move at higher velocities and obtain better time-lengths) and thus the efficacy of

overall strategy is vindicated. The minimum distance over which the velocities need to be computed on the remaining trajectory during real-time such that the computed velocities are safe is theoretically established. This avoids repetitive computation of velocities over the entire remaining trajectory for every motion command, thereby reducing computational intensity and facilitating for real-time implementation. The methodology could be useful in the context of personal robots moving in areas where interference with mobile humans especially aged ones are generally expected.

Immediate scope of this work involves in incorporating the memory phenomena at the reactive level such that higher speeds are possible. The methodology needs to be validated in the presence of mobile objects that actually impinge on the path from blindzones with a provision for the robot to avoid the objects without halting, continuing to respect safety considerations as well as minimizing trajectory time.

## Acknowledgments

The work described in this paper was conducted within the EU Integrated Project COGNIRON ("The Cognitive Companion") and was funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020 and by the French National Program ROBEA.

## References

- [1] R. Alami, T. Siméon, and K. Madhava Krishna. On the influence of sensor capacities and environment dynamics onto collision-free motion plans. *IEEE/RSJ International Conference on Intelligent Robots and Systems, EPFL, Switzerland*, 2002.
- [2] J.C. Alvarez, A. Skhel, and V. Lumelsky. Accounting for mobile robot dynamics in sensor-based motion planning: experimental results. *IEEE International Conference on Robotics and Automation, Leuven (Belgium)*, 1998.
- [3] B. Bouilly, T. Siméon, and R. Alami. A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty. *IEEE International Conference on Robotics and Automation, Nagoya (Japan)*, 1995.
- [4] D. Cruzel. Planification de mouvements sous contraintes de perception. Master's thesis, LAAS-CNRS, 1998.

- [5] P. Fiorinin and Z. Schiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, 1998.
- [6] S. Fleury, P. Soueres, and J.P. Laumond. Primitives for smoothing mobile robot trajectories. *IEEE Transactions on Robotics and Automation*, 11(3):441–448, 1995.
- [7] M. Khatib, B. Bouilly, T. Siméon, and R. Chatila. Indoor navigation with uncertainty using sensor-based motions. *IEEE International Conference on Robotics and Automation, Albuquerque (USA)*, 1997.
- [8] K.Madhava Krishna, R. Alami, and T. Siméon. Moving safely but not slowly - reactively adapting paths for better trajectory times. *IEEE International Conference on Advanced Robotics, Quimbra, Portugal*, 2003.
- [9] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic, 1991.
- [10] A. Lazanas and J.C. Latombe. Motion planning with uncertainty: a landmark approach. *Artificial Intelligence*, pages 287–315, 1995.
- [11] J. Minguez and L. Montano. Nearness diagram navigation. a new real-time collision avoidance approach. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.
- [12] N. Roy and S. Thrun. Motion planning through policy search. *IEEE/RSJ International Conference on Intelligent Robots and Systems, EPFL, Switzerland*, pages 2419–2425, 2002.
- [13] Z. Schiller, F. Large, and S. Sekhavat. Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. *IEEE International Conference on Robotics and Automation.*, pages 3716–3721, 2001.
- [14] C. Stachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems, EPFL, Switzerland*, 2002.
- [15] S. Suri and J. O’Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. *ACM Symp. on Computational Geometry*, 1986.
- [16] D Fox, W Burgard and S Thrun The Dynamic Window Approach to Collision Avoidance *IEEE Robotics and Automation Magazine*, 1997.
- [17] Brock, Oliver and O Khatib High Speed Navigation using the Global Dynamic Window Approach *IEEE International Conference on Robotics and Automation*, 1997.