Dynamic speed planning for safe navigation

A. Mandow, V.F. Muñoz, R. Fernández, and A. García-Cerezo

Departamento de Ingeniería de Sistemas y Automática
University of Málaga, 29071 Málaga, Spain
e-mail: amandow@uma.es

Abstract - The paper deals with speed control for autonomous mobile robots. First a trajectory planner attaches a speed component to the postures of a path by considering speed limitations introduced by the vehicle and by task specifications. In order to provide robustness during task execution in the real world, environment feedback is used to dynamically adjust the speed of the vehicle to the presence of unexpected obstacles, both static and mobile. This is accomplished by combining the planned speed profile with the outputs of two reactive controllers. The system has been successfully implemented within the control architecture of the RAM-2 mobile robot.

Keywords: Mobile robots; velocity control; fuzzy logic; obstacle avoidance; laser range-finder; path planning; reactive navigation; moving object detection.

This document is a self-archiving copy of post-print IEEE-copyrighted work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author’s copyright. This work may not be reposted without the explicit permission of the copyright holder.

The published article is available in: http://dx.doi.org/10.1109/IROS.1997.649059

How to Cite:


@INPROCEEDINGS{Mandow1997IROS,
  title={Dynamic speed planning for safe navigation},
  booktitle={IEEE International Conference on Intelligent Robots and Systems},
  year={1997},
  pages={231-237},
  doi={10.1109/IROS.1997.649059},
  address={Grenoble, France},
}

© Copyright 1997 by the authors.
Dynamic Speed Planning for Safe Navigation

A. Mandow, V. F. Muñoz, R. Fernández, and A. García-Cerezo

Departamento de Ingeniería de Sistemas y Automática. Universidad de Málaga.
Plaza el Ejido s/n, 29013 Málaga (Spain).
Fax: (+34) 5 213-14-13; Tel: (+34) 5 213-14-07; E-mail: amandow@ctima.uma.es

Abstract

The paper deals with speed control for autonomous mobile robots. First a trajectory planner attaches a speed component to the postures of a path by considering speed limitations introduced by the vehicle and by task specifications. In order to provide robustness during task execution in the real world, environment feedback is used to dynamically adjust the speed of the vehicle to the presence of unexpected obstacles, both static and mobile. This is accomplished by combining the planned speed profile with the outputs of two reactive controllers. The system has been successfully implemented within the control architecture of the RAM-2 mobile robot.

1. Introduction

Real world tasks for autonomous mobile robots demand efficient and robust navigation systems for coping with dynamic environments. Efficiency is greatly improved when navigation time is optimized for executing a particular mission, while robustness refers to the capability of accomplishing a given task despite uncertainties and unexpected obstacles, which demands active sensing and reactivity.

The combination of geometric trajectory planning and tracking has proven to achieve efficient navigation based on a map of the environment [3]. Mobile robot trajectories are planned by considering kinematic and dynamic constraints on the vehicle motion, which results in a smooth path with quasilinear curvature variations and associated speeds. Such plans provide ideal conditions for high precision path tracking, usually by means of odometry.

However, these techniques alone do not guarantee safe navigation in real environments, since they lack reactive capabilities to cope with unexpected events. To deal with real world tasks, mobile robots use an external sensor system that provides information about its surroundings. This valuable information can be processed in many different ways in order to dynamically add robustness to preplanned tasks by coping with dead reckoning errors, changes in the environment, mobile obstacles, local replanning, etc.

The paper addresses the control of the vehicle's speed while executing a planned trajectory. The plan incorporates speed commands for each point along the path, which have been computed by considering a number of operational and vehicle speed restrictions. When the robot is actually performing the task, these planned speed commands may have to be dynamically replanned according to run-time restrictions, mainly concerned with safety, detected through the sensor system.

Two controllers have been incorporated into the tracker to deal with unexpected obstacles on the vehicle's way and to adapt the speed behaviour to the presence of other mobile objects in the working area. Range sensor data, odometric estimations, geometric computations and fuzzy rules are used by these controllers, whose outputs, together with the planned speed, are compared to obtain the dynamic speed limitation at each control step. Fuzzy logic has been chosen for the design of the dynamic speed controllers [4] because it provides tools for fusing different control commands, produces smooth transitions between states, and is suitable to easily model non-linearities.

The whole system has been implemented on the RAM-2 robot, a wheeled octagonal platform 1.10 metres wide equipped with a 180º radial laser scanner (see Fig.1).

The paper is organized as follows. Section 2 introduces the different types of speed constraints to be considered for robot navigation. Section 3 presents the

Fig. 1. RAM-2 mobile robot.
2. Speed Constraints

The reasons which determine the necessity of speed planning along a path arise in order to satisfy a set of practical constraints. These constraints can be divided into three categories, which are detailed in the following subsections.

2.1. Speed limitations introduced by vehicle features.

They are due to the kinematic and dynamic behaviour of the mobile robot. These restrictions impose upper bounds to speed and acceleration according to the peculiarities of the vehicle and the path to be followed. The origin of these constraints can be classified into the points shown below:

- **Vehicle mechanics.** The features of the robot locomotion system, such as the driver motors and the traction chain, set a maximum speed and acceleration.
- **Vehicle kinematics.** The vehicle kinematic model limits the maximum speed and acceleration of the vehicle's guide point as a function of the speed of each wheel.
- **Vehicle dynamics** involve the dynamic behaviour of the robot driver and steering systems, as well as the forces that act over the robot in motion and deflect its desired trajectory. For instance, it is possible to define a top speed which provides an admissible maximum lateral force.

2.2. Operational speed limitations.

The robot performs its task as an element of a more complex system. Consequently, the robot should adapt its speed depending on the current mission. The possible presence of human operators or mobile obstacles introduces limitations of this kind. These are described in the following points:

- **Distance to a goal point or known obstacle.** The closeness to an obstacle or to the goal point imposes a safety speed value so that the vehicle can stop before getting into a zone with collision hazard. This speed is a function of the maximum deceleration that provides the brake system and the current distance to the obstacle or goal point.
- **Safety speed.** This is a set of speed constraints which depend exclusively on the robot working environment. For safety reasons or synchronization with other elements in the working environment (i.e. avoidance of known mobile obstacles [2]), it is necessary that the robot navigates with a given speed along a path segment. This kind of constraints are defined as a function that assigns a constant top speed \( V_i \) to each portion of the path defined by \([s_{i-1}, s_i]\). These speed constraints are modelled in expression (1).

\[
V(s) = \begin{cases} 
V_1 & 0 \leq s \leq s_1 \\
... & \\
V_N & s_{N-1} < s \leq s_N
\end{cases}
\]

2.3. Dynamic speed limitations.

The speed constraints related in subsections 2.1 and 2.2 are based on facts known at planning time and are supposed to remain unchanged during task execution. The first set of limitations is determined by the vehicle models and the second one by the specification of the task (i.e. desired speed at subgoals, environment map, etc.). Therefore, it is possible use this knowledge for planning a speed profile which optimizes a heuristic function, like minimizing robot's navigation time.

However, other speed limitations can arise dynamically at navigation time, due to scenarios which cannot be taken into account beforehand, like unexpected static obstacles or uncontrolled moving elements. The information provided by the on-board external sensors of the robot must be processed in real time to define these dynamic speed limitations, which are needed in order to stop the vehicle smoothly or change its planned speed according to the presence of unexpected obstacles.

3. Global Speed Planning Methodology.

A robot path is defined as a set of evenly spaced postures \( Q = \{q_1, ..., q_n\} \), which are to be executed by the path tracking algorithm. A posture \( q_i \) is composed of five basic elements: \( x_i, y_i, \theta_i, \kappa_i \) and \( s_i \). The first two elements are position components, the third is the heading with respect to a global work frame, the fourth is the curvature component, and the last one is the distance along the path from the starting posture to the current one.

In order to convert a path \( Q \) into a trajectory \( \hat{Q} \) it is necessary to append a speed component to each posture of the path. In other words, the trajectory conversion process must turn each \( q_i = (x_i, y_i, \theta_i, \kappa_i, s_i) \) into \( \hat{q}_i = (x_i, y_i, \theta_i, \kappa_i, s_i, v_i) \), where \( v_i \) is the posture speed component. This transformation is made by the definition of a parametric arc length speed function \( V(s) \). Such a curve is defined in the space-speed plane [5], in which the upper speed limits for each posture \( q_i \) of the path \( Q \) are represented. These limits
are obtained by taking into account the speed constraints introduced by the vehicle features and operational speed limitations. Thus, \( V(s) \) is specified in such a way that it preserves all the posture speed limits, in order to obtain a speed profile with good tracking conditions. That means that \( V(s) \) must lie inside a safety area of the space-speed plane defined by the speed limits functions (see Fig. 2).

The speed function \( V(s) \) definition is made in two steps: speed planner and speed generation processes.

### 3.1. Speed planner process.

This stage chooses a set of control path postures \( C = \{ q_1, ..., q^{P-1} \} \) which divides the path into a set \( S = \{ S_1, ..., S_p \} \) of path segments, where \( S_i \) is composed of the path postures sequence between \( q_i \) and \( q^{i+1} \). The choice of the elements which will be belong to \( C \) is made depending on the nature of the speed limitations:

- Kinematic and dynamic considerations are a function of the path curvature \( \kappa \). Therefore, it is necessary to consider the variation law of this magnitude. In this way, the postures which have the local maximum or minimum curvature values are added to \( C \) set. Hence, the path is divided into a set of path segments whose curvatures follow either an increasing, decreasing or approximately constant law.
- Secondly, new path postures are chosen in order to satisfy the operational speed limitations. The posture closer to the goal point or known obstacle which satisfies the distance to a goal point speed limitation is selected. Moreover, in order to prove the safety speed considerations, path postures that define the portion of the path in expression (1) are added to the control path postures \( C \).

A top speed \( v_i \) is assigned to each member \( q_i \) of \( C \) by using the minimum speed value provided by the speed limitations introduced by vehicle features and operational speed constraints. This operation sets up a speed control set \( V = \{ 0, v_1, ..., v_{P-1}, 0 \} \) for the path \( Q \), whose first component (always null) is the starting speed for \( S_1 \) and the remaining components \( v_i \) are speed boundary conditions between segments \( S_{i-1} \) and \( S_i \). Sets \( C \) and \( V \) resulting from the speed planning process, can be represented in the space-speed plane, and will be used by the speed profile generator process for building \( V(s) \).

### 3.2. Speed generation process.

The \( V(s) \) is modelled through a spline curve defined as a set of space-time functions \( \sigma_i(t) \). The \( i \)th component of this curve is assigned to the path segment \( S_i \), and it is determined as follows:

\[
\sigma_i(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \alpha_3 t^3
\]

The coefficients \( \alpha_i \) are computed in order to satisfy the constraints given by the sets \( C \) and \( V \), as it is shown in the following expression:

\[
\begin{bmatrix}
0 & 0 & 1 & \sigma_0 \\
1 & 2 & t & \sigma_1 \\
0 & 0 & 1 & \sigma_2 \\
3 & 2 & t^2 & \sigma_3 \\
\end{bmatrix}
\begin{bmatrix}
\alpha_0 \\
\alpha_1 \\
\alpha_2 \\
\alpha_3 \\
\end{bmatrix} = 
\begin{bmatrix}
0 \\
\sigma_1 \\
\sigma_2 \\
\sigma_3 \\
\end{bmatrix}
\]

where \( \sigma_1 \) and \( \sigma_{i+1} \) are the starting and ending speed assigned to the current path segment; \( s_i \) be its length and \( t_i \) the navigation time. Finally, the speed profile \( V(s) \) is defined in the space-speed plane by using expression (4).

\[
V(s) = \bigcup_{i=1}^{p} \frac{d}{dt} \sigma_i^{-1}(s)
\]

### 4. Dynamic Speed Planning

The trajectories generated by the algorithms presented in the preceding section can be safely executed by means of dead reckoning tracking. However, their utility is limited to static environments that must match exactly with the map used for the planning process. This section discusses how the speed component of the planned trajectory can be dynamically treated when further speed constraints are detected through the sensor system, without the need to modify the position and orientation components of the task. First, a basic controller is introduced that sets a speed command to slow down the vehicle in the presence of obstacles on the way. Then, a more complex controller is presented, which adapts the speed command to the presence and trajectories of other mobile objects.

### 4.1. Unexpected obstacles.

One simple way for coping with obstacles found unexpectedly on the robot's way is to adapt its speed according to their measured distance. This is a basic safety controller that eventually stops the vehicle if the obstacle is not removed before close vicinity is reached. The robot will resume its motion when the obstacle moves away or is removed from its hindering position.

This is a low cost solution that can actually be implemented by means of a few fuzzy rules and a very simple sensor system, like a couple of ultrasonic sensors pointing forward (see Fig. 3.a).

If a more sophisticated sensor, like a radial laser
scanner, is available, a simple processing of the data is performed by establishing a rectangular security area at the front of the vehicle, and using the shortest detected range within that frame (as in Fig. 3b). When several sensors are used, their outputs are and-ed by choosing the minimum of them (the closest reading) as input to the controller.

Fig. 3. Detection of unexpected obstacles.

This basic controller has been implemented by using two sonars mounted on the RAM-2 robot which can range up to 1.3 metres. The controller consists of just the 3 rules shown in Fig. 4. The input (left side) is the measured distance to the object, and the output the commanded speed.

Fig. 4. Rules for speed control in the presence of unexpected obstacles.

Note that the output of the rules related with closer distances (i.e. the most dangerous) have been given a greater weight. The performance of this controller is shown in fig. 5, where speed is modified according to the lowest distance measured by the two sonars.

Fig. 5. Performance of the unexpected obstacle controller.

This figure shows a case in which a fixed obstacle appears, and illustrates how the speed is controlled to smoothly stop the vehicle. The central graph depicts the readings from two front sonars, and the lower shows the speed command and the real speed.

These safety rules can also be used for mobile objects, allowing to adapt the speed of the robot to the distance of a moving object driving ahead in the same direction, or to slow down and stop if a mobile obstacle crosses the robot's path. However, in such cases results are merely adequate and characterized by sharp-profiled control curves, since the controller is only triggered when the obstacle lies within the security area.

4.2. Mobile obstacles.

A smoother behaviour can be obtained if a more comprehensive sensor system, like a laser scanner, can be used to detect mobile objects, identify their trajectory and estimate if, when and where they will cross the robot's path.

The application presented here makes use of an on-board laser scanner which produces a two-dimensional description of the environment in polar coordinates, providing 90 points over a 180º field of view, every 0.3 seconds. Although it can reach up to 50 metres, only measures under 8 metres have been considered because greater ranges are affected by a worse resolution and also because objects at longer distances pose little danger to the robot.

The following algorithm is used to identify the presence and trajectory of mobile objects around the robot:

\[
\begin{align*}
M & = \{ \} \quad \text{(Set of matched points)} \\
U & = \{ \} \quad \text{(Set of unmatched points)} \\
NU & = \{ \} \quad \text{(Set of New Unmatched points)} \\
NM & = \{ \} \quad \text{(Set of New Matched points)} \\
\text{Every Scan Interval do,} \\
& \text{For every scan measure } s, \text{ such that } s < 8m \text{ do,} \\
& \quad \text{Transform } s \text{ into world cartesian coordinates } s_w \\
& \quad \text{If } s_w \in U \text{ then} \\
& \quad \quad \text{Add } s_w \text{ to } NM \\
& \quad \text{If } s_w \in M \text{ and } s_w \notin U \text{ then} \\
& \quad \quad \text{Add } s_w \text{ to } NU \\
& \quad \text{end (for)} \\
& \quad \text{Add } NM \text{ to } M \\
& \quad \text{Remove } NM \text{ from } U \\
& \quad \text{NM} = \{ \} \\
& \text{Update } U \text{ by adding } U \\
& \text{For every cluster in } NU, \\
& \quad \text{Compute its Centre of Gravity (CoG)} \\
& \quad \text{Relate with CoGs from latest scans in order to identify object trajectories} \\
& \text{end (for)} \\
& \text{NU} = \{ \} \\
\end{align*}
\]

Basically, the algorithm detects clusters of points that were not detected in previous scans (NU points), whereas points which have been detected before are added to a set of matched points (M points), which implicitly defines the static map of the environment. This can be intuitively seen.
on Fig. 6, which shows all points (closer than 8 metres) obtained along a short path with the RAM-2 robot. Here, points repeatedly appear where they define walls, columns and other static items, whereas “one-scan” occurrences define the paths of what can possibly be two moving objects crossing the path of the robot (in fact, two people walking).

![Fig. 6. All measured scans from a short path.](image)

Membership (ε) of new scan point to the sets M or U has been computed in terms of closeness, which, considering sensor resolution, varies depending on the range at which the point is measured. The set M is restricted to a maximum of 300 points, so that the problem is computationally tractable. Besides, since local polar coordinates are transformed to world cartesian based on odometry, older points accumulate odometric uncertainty. Thus, the system “forgets” points which are no longer useful. Moreover, new scan points that are already in M are discarded.

Clusters of each scan’s new unmatched points, may correspond to moving obstacles. These are represented by their centre of gravity in order to simplify the problem, and most important, to minimize the uncertainty about the actual size, shape and volume of detected obstacles. Each of these centers is then compared with the ones from the 5 latest scan intervals in order to update the trajectories of mobile objects. Particularly, the latest 3 centers of an obstacle are considered to fix a straight line as its estimated trajectory.

The results obtained with the application of this method along the path shown in Fig. 6 can be appreciated in Fig. 7, where the paths of two moving objects have been clearly defined. The figure also shows how the set M contains only the essential points necessary to define the static environment.

By using this method, spurious mobile objects can be obtained. This usually happens when the robot reaches a new area (like arriving to a room after following a corridor), when most points are considered to be new, and processed as mobile by the algorithm. Nevertheless, the importance of these false mobiles is scarce, since they follow the contour of natural objects, already contemplated in the plan. Furthermore, in those cases two consecutive scans suffice to update the M back to normal, so scans with high percentages of NU points are not used to update obstacles in the implemented version of the algorithm.

![Fig. 7. Processed sensor data, with the estimation of obstacle trajectories.](image)

The distance along the robot’s trajectory at which the obstacle is expected to cross (dc) is obtained geometrically, considering the next 10 metres of planned trajectory. The estimated speed of the obstacle is used to compute the time that is left for it to actually cross the trajectory (te), as shown in Fig. 8.

![Fig. 8. Input variables to the controller.](image)

Parameters dc and te are inputs to the fuzzy controller defined by the following rule table to produce a speed command.

<table>
<thead>
<tr>
<th>Table 1: Mobile Obstacle rulebase</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.2</td>
</tr>
<tr>
<td>0.15</td>
</tr>
<tr>
<td>0.1</td>
</tr>
</tbody>
</table>

Each column on the table corresponds to a fuzzy set defined over the distance to the robot (Zero, Almost Zero,
Middle, Far and Very Far), and the rows represent the Time left (Zero, Almost Zero, Middle, Long, Very Long). The outputs are the speed commands in m/sec produced by each combination of antecedent data. These rules adapt the speed of the vehicle to give way to the obstacle; thus, the cells at the bottom-left of the table, which correspond to objects that will cross a close point in the trajectory after a long period of time, produce very slow commands. When more than one obstacle is processed, the lowest speed reference is considered.

5. Implementation and Experiments

5.1. Autonomous navigation control architecture.

The system has been integrated in the control architecture for the RAM-2, a mobile robot designed and built for research in indoor and outdoor industrial applications. The control architecture follows a hierarchical decomposition in three levels: User, Executive, and Low Level Control, as can be seen on Fig. 9. The system is implemented using the real-time operative system Lynx. The boxes depicted in the figure are independent threads, which are special processes that can communicate with each other by means of events and entry points to their global variables.

At the top of the architecture, the user can start-up, shutdown, choose different strategies of actuation, or record execution data through the User Level, which incorporates a graphical user interface.

The Executive Level is coordinated by the Supervisor, which manages events from the rest of the modules and changes the operation mode; it also initializes and configures all threads at the two lower levels. Moreover, the Trajectory Planner and the Tracker (which includes the Dynamic Planner described above) are part of this level. A vigilance module checks the correct operation of the low level systems (sensors and motors) and issues an emergency event to the Supervisor if a failure is detected.

The Low Level Control is dedicated to collecting the data from external sensor system, controlling the actions of the motors, and odometric estimations.

5.2. Integration of Trajectory Planning and Tracking.

All three subsystems presented in the paper produce speed commands for the vehicle. The actual speed command, issued to the Vehicle Control by the Tracker, must be obtained by appropriately combining those values.

The simplest and most coherent way of doing so is by adding run-time restrictions to the space speed plane of Fig. 2. This is accomplished by applying the minimum operation (the typical fuzzy "and") to the three possible values, so that the most restrictive one is applied.

Since the Mobile Obstacle controller always yields priority to other moving objects, no matter how slow or distant they are, it is preferable that the Tracker only considers its output if the crossing point is not going to be surpassed by the robot (with a reasonable security margin, which has been established as 1.5 metres) according to the planned speed.

5.3. Experimental results.

Figures 10 to 12 show actual results of the proposed method obtained with the RAM-2 mobile robot at the hall of our building. First, a path was generated from point A to point B, as illustrated on Fig. 10, to which a speed component was added according to the procedures discussed in Section 3. The execution of the planned path is shown in the figure by a dotted line. In particular, pure pursuit with parameter lookahead at 1.0 m was the method used by the path tracker. During execution, a person walked through the environment, crossing the path of the robot at point C. The dynamic speed planning controllers detected its presence, as can be appreciated on Fig. 11, and issued speed commands that modified the planned one, as can be seen on Fig. 12. This dynamic speed control avoids stopping the vehicle by adapting its speed to the trajectory of the moving obstacle.

The reactive nature of this controller accounts for the oscillations produced by its commands (asterisks on Fig. 12). Obviously, the trajectory followed by the obstacle is not a straight line, and being a human, the shape of the obstacle varies notably from one scan to another as it moves its arms, legs, etc. On account of this, outputs from the mobile obstacle avoidance controller are filtered by issuing the lowest command of the latest five control intervals. Once the obstacle has safely crossed the robot’s
path, the planned speed commands are activated again.

Fig. 10. Planned and executed paths.

Fig. 11. Perceived trajectory of the mobile obstacle at run time.

Fig. 12. Planned speed profile against the executed one.

6. Conclusions and future work.

This paper presents a solution to the mobile robot speed planning problem by using a global path planner procedure and a dynamic speed planning based on environment feedback.

The global path planner builds an admissible speed profile by taking into account the vehicle's motion features, operational limitations and the path properties. The method is composed of two steps (speed planning and speed profile generation) and provides a safe global speed plan according to the task specification.

The dynamic planner consists of two controllers: the Unexpected Obstacle Controller and the Mobile Obstacle Avoidance Controller. First, the Unexpected Obstacle Controller simply prevents collision with any obstacle which is actually on the robot's path, becoming the basic safety controller. A short distance ahead of the robot is considered, so that natural landmarks are not considered as obstacles. Its performance could be improved if longer ranges were considered. In this way, a mechanism would have to be devised to differentiate between expected and unexpected objects. Another possible solution would be adapting the shape of the security area to that of the path, considering the robot's width.

The second one, the Mobile Obstacle controller is predictive, so it allows adapting the speed of the vehicle to the collision hazard posed by the surrounding moving obstacles, which in most cases avoids stopping the robot abruptly and wait once an obstacle is actually crossing. An algorithm for detecting the presence and the trajectory of moving obstacles from laser scans has been presented, which has been used to predict when and where they will cross the robot's path. A real time speed command is obtained from a fuzzy rulebase, whose parameters have been empirically adjusted. If an obstacle changes its speed while crossing or is bigger than usual (since the actual size is unknown), robustness is achieved by combination with the Unexpected Obstacle controller. Future work will involve the estimation of the size of the obstacle and consideration of the robot's width.

7. Acknowledgements

This work has been done within the framework of the projects TAP96-1184-C04-02 and TAP96-0763 of the C.I.C.Y.T. (Spain).

8. References