THE BEAM-CURVATURE METHOD: A NEW APPROACH FOR IMPROVING LOCAL REAL-TIME OBSTACLE AVOIDANCE

J.A. Benayas, J.L. Fernández, R. Sanz and A.R. Diéguez

Dept. de Ingeniería de Sistemas, Universidad de Vigo, Campus Lagoas-Marcosende, 36200 Vigo, Spain
Phone Int.: +34 986 812222, E-mail: {jalbertos,joaquin,rsanz,amador}@aisa.uvigo.es

Abstract: This paper describes a new local avoidance method for indoor mobile robots. The method uses a directional method named the Beam Method to improve the performance of a local obstacle avoidance approach called Curvature Velocity Method (CVM). The proposed Beam Method employs radial distances provided by the robot sensors to calculate the best one-step headings. The CVM uses this information to calculate the optimal translational and rotational velocities.

Our approach incorporates the local information in an intuitive way, similar to human vision, by taking advantage of the natural disposition of the mobile robot sensor systems. Compared with other methods it shows more efficient performances, being able to navigate through openings very efficiently way. Several experiments in populated and dynamic environments with our mobile robot RATO have proved to be very successful.

Copyright © 2002 IFAC

Keywords: Obstacle avoidance, robot navigation, mobile robots, robotics, robot control, real time.

1. INTRODUCTION

This paper deals with reactive avoidance of obstacles in indoor environments. Although this topic has been broadly considered in the literature, it is evident that not all the proposed methods are able to give an appropriate answer to every situation. For example, in very crowded and dynamic environments it is very important to find obstacle-free paths without causing dangerous situations for people.

Different obstacle avoidance methods have been proposed over the years. Some of them employed global information provided by a complete model of the robot’s environment. These approaches are appropriate for path planning but are inadequate for real-time obstacle avoidance since the environment model is generally incomplete and inaccurate. Local methods, on the other hand, use only the information provided by proximity sensors to produce robot commands. These methods give faster responses but the robot can remain blocked in many situations. Some of local methods only provide a direction (for example, Borenstein and Koren, 1991), rather than more recent methods that can also integrate the robot dynamics. Although these obstacle avoidance algorithms work acceptably in general, they sometimes make undesirable command selections. These drawbacks are due to their purely local nature since they do not take into account short-term future motion. Combining a local approach with a motion planner can solve some of these limitations. Nevertheless, these frameworks require a priori knowledge about the environment in order to select the best motion commands.

The new local obstacle avoidance method that we present in this paper, called the Beam Method (BM), avoids the use of local planning. It has been inspired by our experiments with CVM and LCM approaches developed by Simmons (1996) and Ko and Simmons (1998). Though these methods produce reliable and smooth navigation in indoor environments, they have some limitations.

The algorithm described in this paper overcomes problematic situations discovered after extensive testing of the above mentioned local obstacle avoidance methods.

The remainder of this paper is organized as follows. In section 2 some related methods are briefly
presented. Section 3 describes our approach in detail. Experimental results with our mobile robot are summarized in section 4. Finally, some conclusions are given in section 5.

2. OTHER RELATED OBSTACLE AVOIDANCE METHODS

There is a large number of local obstacle avoidance methods proposed in the literature. Most of the earlier real-time obstacle avoidance approaches were based on Artificial Potential Fields (Khatib, 1986). For example, Borenstein and Koren (1989) developed the Vector Field Histogram (VHF) method by integrating the concept of potential fields with certainty grid. VHF is less likely to get in local minima and allows the robots to navigate at faster speeds without colliding with obstacles. Although the original VHF method ignores the dynamics and kinematics of the robot it has become very popular due to its advantages. New extensions of the original methods that take into account the dynamics and kinematics constraints have been proposed recently (Ulrich and Borenstein, 1998; Ulrich and Borenstein, 2000). Other researchers, probably influenced by VHF, have developed algorithms based on the Velocity Space approaches that incorporate vehicle dynamics by choosing rotational velocity along with translational velocity. The Dynamic Window approach (Fox et al., 1997; Brock and Khatib, 1999) and the Curvature-Velocity Method (Simmons, 1996) are two of the most significant frameworks. These methods consider kinematics constraints by directly searching the velocity space of the robot. The search space is a set of tuples of translational and rotational velocities that are achievable by the robot. The constraints represent both the presence of obstacles and physical limitations on robot’s velocities and accelerations.

The Curvature-Velocity Method (CVM) chooses a point in translational-rotational velocity space, which satisfies some constraints and maximizes an objective function. The objective function is proportional to the arc distance that the robot can go before hitting a set of obstacles, the normalized arc distance by some limiting distance and the normalized error in goal heading defined as the difference between the commanded heading and the heading the robot will achieve if it turns at the rotational velocity for some time constant. Also, it adds an extra constraint to avoid backward motion.

The CVM assumes that the robot always travels along circular arcs. These trajectories work well from the point of view of the reactive control transforming Cartesian space obstacles into velocity space. Though it produces reliable and smooth navigation, it has some limitations. For example, at an intersection of corridors, it fails to guide the robot into an open corridor toward the goal direction (figure 1). This problem derived from the fact that CVM chooses commands based on the collision-free length of the arcs assumed to be robot’s trajectories. It does not consider that the robot may be on that arc for just a short distance, and will soon be turning again.

Ko and Simmons (1998) realized these limitations and developed a method to guide the CVM and a way to correct these drawbacks. Their method is called Lane-Curvature Method (LCM). The LCM enhances the velocity space approach by considering collision free direction as well as the collision free arc length. It combines the Curvature-Velocity Method (CVM) with a directional method called the Lane Method that divides the environment into lanes oriented in the direction of the desired goal heading, and then selects the best lane to optimize travel along a desired heading. A local heading is next calculated for going through and following the best lane. CVM uses this heading to determine the optimal translational and rotational velocities, considering the heading direction, physical limitations, and environmental constraints.

By combining both directional and velocity space methods, LCM yields safe collision-free motion as well as smooth motion, taking the dynamics of the robot into account. Nevertheless, the LCM still has some other problems encountered in practice because the lane calculation modifies the vision of the environment not always guaranteeing to get good trajectories (figure 2). Again, there are other limitations due to the nature of the method itself caused by the lack of radial projected vision, not seeing clear spaces with enough distance free of collision (figure 2). Also, it sometimes detects false openings when the robot detects obstacles that hide other blocked paths (figure 3).

![Fig. 1. Example of an opening ignored by CVM.](image1)

![Fig. 2. Some radial openings ignored by LCM.](image2)
Fig. 3. Example of a true and a false openings detected by LCM.

All these problems are basically because the method calculates lanes in the goal direction based on the expected free space. In addition, like many other local obstacle avoidance techniques, LCM does not take into account further steps.

Our approach takes the CVM method as a basis because it is quite efficient and has been used in a safe way over the last five years in Xavier, a robot controlled from the Web [Simmons, 1999]. However, in spite of safety, in this work we have intended to improve its efficiency in a similar way to LCM.

Our method, like LCM, generates motion commands in two steps (figure 4). In the first step a desired local goal heading is determined by using a directional approach that we have called Beam Method. In the second step, the steering commands yield a motion in the desired local direction by using CVM method and take into account the dynamic constraints of the robot. The local trajectory toward the objective is calculated in real time for each sampling interval. This two-step approach improves navigation because it is able to find collision free openings that could not be determined by LCM. Indeed, our method uses robot sensor information as is. The obstacle information is centered around the robot’s current position and it is built from data gathered by the proximity sensors.

3. THE BEAM METHOD

The purpose of this new method is to get a local heading from global goal heading that guides the robot to the best path.

The BCM obtains a divergent radial projection model of the environment based on the sensors’ natural disposition. This model is later simplified and, after that, a local heading target is calculated by maximizing an objective function. In the following, a brief description of the main steps is given.

3.1 Calculation of the beam associated to the obstacle.

As in other methods, we approximate every obstacle to a circle, since computing the obstacle distance can be very complex for arbitrarily shaped obstacles. For each obstacle the associated radial beam, \( B \), is calculated. This is defined by the parameters \( B(\rho_1, \rho_2, d) \), where \( \rho_1 \) and \( \rho_2 \) are the limit angles from the tangent beam to the obstacle, and \( d \) the beam minimum impact distance. The calculation of these parameters is defined next.

From figure 5 we can deduce the following relations to find the angles:

\[
\theta_0 = \tan(x_{obs}/y_{obs}) \\
\theta = \sin(r_{obs}/d_{obs})
\]

where the angles \( \rho_1 \) and \( \rho_2 \) are easily obtained from the following relations:

\[
\rho_1 = \theta_0 - \theta \\
\rho_2 = \theta_0 + \theta
\]

Then, a beam associated to an obstacle \( O(x_0, y_0, r_{obs}) \) is defined in the following way:

\[
d = \sqrt{x_{obs}^2 + y_{obs}^2} - r_{obs} \\
\rho_1 = \theta_0 - \frac{r_{obs}}{\sqrt{x_{obs}^2 + y_{obs}^2}} \\
\rho_2 = \theta_0 + \frac{r_{obs}}{\sqrt{x_{obs}^2 + y_{obs}^2}}
\]

Fig. 4. Beam Method approach is used to improve CVM.

Fig. 5. Beam parameters and obstacle distance calculation.
3.3. Selection of the best beam.

After filtering and constructing the final set of beams, the best is selected for guiding the robot. Each beam $B(\rho_1, \rho_2, d)$ is rated according to the following objective function:

$$f(\rho_1, \rho_2, d) = \alpha (d \cos(\varepsilon) / \delta_{max}) - \beta \varepsilon$$  \hspace{1cm} (7)

Where $d \cos(\varepsilon) / \delta_{max}$ is the projected distance over the goal direction; $\varepsilon$ is the smallest angular error between the goal direction and every analyzed beam; and $\alpha, \beta$ are weight constants experimentally set ($\alpha+\beta=1$). If we call $\Phi_o$ the angle between the robot’s current orientation and the goal direction, $\varepsilon$ is defined as:

$$\varepsilon = \begin{cases} 0 & \text{if } (\rho_1 < \Phi_o) \text{ and } (\rho_2 > \Phi_o) \\ \Phi_o - \rho_2 & \text{if } (\rho_2 < \Phi_o) \\ \rho_1 - \Phi_o & \text{otherwise} \end{cases}$$  \hspace{1cm} (8)

The beam with the highest rate according to the value of the objective function is selected.

Where:

$$\theta_o = \begin{cases} \pi - \tan(x_{obs} / y_{obs}) & \text{if } (y_{obs} < 0) \text{ and } (x_{obs} > 0) \\ \pi + \tan(x_{obs} / y_{obs}) & \text{if } (y_{obs} < 0) \text{ and } (x_{obs} < 0) \\ \tan(x_{obs} / y_{obs}) & \text{otherwise} \end{cases}$$  \hspace{1cm} (6)

3.2 Beam overlapping.

All the beams produced in the previous step must be reduced to a final set, so that every point in the space must belong to only one beam. A minimum fusion algorithm has been developed to analyze the different overlapped beams and to select those that have minimum impact distance to the obstacle. The impact distance is the distance that the robot needs in order to stop before it reaches the obstacle.

The beams, already processed and inserted in the final set, can be modified every time a new beam is evaluated and added to the set. In order to integrate a new beam, the method considers four types of relations between the new beam $B(\rho_1, \rho_2, d)$ and each of the already integrated beams $B(\rho_1, \rho_2, d)$ (see Figure 6): a) Both two beams are disjoint (obstacles 1 and 9); b) the new beam contains the existent one (the beam of obstacle 8 includes beam 5); c) the new beam is contained in the existent beam (the beam of obstacle 5 is included in beam 8), or d) Two limit beam lines are overlapped.

3.3. Selection of the best beam.

Fig. 6. Different cases of beam overlapping.

3.4. Goal heading calculation.

Once the best beam has been chosen, it is necessary to determine the desired goal heading. CVM will try to follow this heading if it is possible, though this could result in passing excessively close to the obstacles. To avoid this, the process to obtain the goal heading takes into account a certain security distance.

The safe beam $B(\rho_{safe1}, \rho_{safe2})$ represents the collision-free angular interval, around the objective beam, that guarantees a set of headings that keep the robot traveling away from any obstacle at least minimum distance $d_{safe}$. Approximating this safety distance to an arc of a circle with a minimum impact distance, $d$, the cautious angle $\rho_{safe}$ is:

$$\rho_{safe} = d_{safe} / d$$  \hspace{1cm} (9)

The safe beam is determined as a function of the best beam $B(\rho_{best1}, \rho_{best2}, d_{best})$:

$$\rho_{safe1} = Max \{ \rho_{best1} + d_{safe} / d \}$$  \hspace{1cm} (10)

$$\rho_{safe2} = Min \{ \rho_{best2} - d_{safe} / d \}$$

The “Max” in the first function is calculated over all the beams between $-\pi$ and the best beam $B_{best}(\rho_{best1}, \rho_{best2}, d_{best})$ and the “Min” for all the beams between the best beam and $\pi$.

If $d << d_{safe}$, then the above formulation yields high security angles (increasing to infinite when $d = 0$), which produces sudden steering if a close obstacle is in front of the robot. In contrast, a near lateral obstacle requires a smaller change of heading command to avoid it. This is the reason why we reduce the security angle associated to the lateral obstacles with respect to the central ones. With this limitation, we obtain a new formulation:
\( \rho_{1\text{safe}} = \max_{B_i(\rho_1, \rho_2, d_i)} \left( \rho_2 + \min(d_{\text{safe}} / d_i, \rho_2 + \pi / 2) \right) \)

\( \rho_{2\text{safe}} = \min_{B_i(\rho_1, \rho_2, d_i)} \left( \rho_1 - \min(d_{\text{safe}} / d_i, \rho_1 + \pi / 2) \right) \)

Defining:

\( \rho_{1\text{lim}} = \max(\rho_{\text{best}}, \rho_{1\text{safe}}) \)

\( \rho_{2\text{lim}} = \min(\rho_{2\text{best}}, \rho_{2\text{safe}}) \)

Finally, for obtaining the local goal heading \( \phi_{\text{goal}} \) it is necessary to consider two situations:

1. \( \rho_{2\text{lim}} \geq \rho_{1\text{lim}} \)

\( \phi_{\text{goal}} = \begin{cases} \phi_o & \text{if } (\phi_o \in [\rho_{1\text{lim}}, \rho_{2\text{lim}}]) \\ \rho_{1\text{lim}} & \text{if } (\phi_o \in (-\infty, \rho_{1\text{lim}}]) \\ \rho_{2\text{lim}} & \text{if } (\phi_o \in (\rho_{2\text{lim}}, \infty]) \end{cases} \)

(13)

2. \( \rho_{2\text{lim}} < \rho_{1\text{lim}} \)

\( \phi_{\text{goal}} = \rho_{2\text{lim}} + (\rho_{2\text{lim}} - \rho_{2\text{lim}}) \alpha (d_{2\text{near}} / (d_{1\text{near}} + d_{2\text{near}})) + \beta (\Delta \rho_{1\text{safe}} / (\Delta \rho_{1\text{safe}} + \Delta \rho_{2\text{safe}})) \)

(14)

where

\( \Delta \rho_{1\text{safe}} = |\rho_{1\text{safe}} - \rho_{1\text{near}}| \)

\( \Delta \rho_{2\text{safe}} = |\rho_{2\text{safe}} - \rho_{2\text{near}}| \)

(15)

\( \rho_{1\text{near}} \) and \( d_{1\text{near}} \) are the angle (\( \rho_1 \)) and distance (\( d_1 \)) associated to the beam that minimizes the function to calculate \( \rho_{1\text{safe}} \) (eq. 11). In a similar way, \( \rho_{2\text{near}} \) and \( d_{2\text{near}} \) are the angle (\( \rho_2 \)) and distance (\( d_2 \)) associated to the beam that minimizes the function to calculate \( \rho_{2\text{safe}} \) (eq. 11).

4. EXPERIMENTS AND RESULTS

The BCM method has been tested on our mobile robot called RATO, based on a B21 mobile robot platform manufactured by RWI. For obstacle avoidance it is equipped with 24 sonar proximity sensors and a range laser. RATO’s software is designed with a layered architecture, consisting of a supervisor, task scheduling, path planning, navigation, and obstacle avoidance components, each of which relies on the abstraction provided by the previous level. The robot architecture is implemented as a collection of asynchronous process. System integration is performed using the Task Control Architecture (Simmons, 1994).

The BCM method has been extensively tested in our School of Engineering main building, where the robot needs to run through narrow corridors and pass through doors. For comparison, different experiments with CVM, LCM and BCM methods were done. Figure 7 shows three executions when the robot is going through a narrow door. The translational velocity is set to 45 cm/sec. Notice that the BCM obtains a better orientation and subsequently a smoother path than LCM and CVM. Figure 8 shows the results of crossing close to a circular wall in our lab with an obstacle. CVM is unable to find the opening since it turns to the right. BCM method is able to find an opening before LCM and then the resulting path always has a better orientation to cross the narrow opening.

Fig. 7. Experiments entering into a narrower door: a) CVM; b) LCM; b) BCM.

Fig. 8. Experiments crossing a narrower area with an obstacle: a) CVM; b) LCM; c) BCM.
In other experiments at high velocities LCM was blocked because it does not have enough time to react when there were obstacles in the goal direction even though there were openings detected by the sensors.

Although our method (like the CVM and LCM methods) does not take into account dynamic obstacles, in practice, it has shown good results with obstacles that move slowly (like people moving around).

5. CONCLUSIONS

This paper presented the Beam-Curvature Method (BCM) for local obstacle avoidance, which incorporates the Curvature Velocity Space Method (CVM) with a radial directional method (Beam Method). The Beam Method determines a local direction for a collision-free space. BCM has been tested in different environment situations and in general better paths have been obtained compared with LCM.

Although BCM and LCM are local directional methods that use CVM as the reactive component, they differ in many aspects:

a) BCM assumes that there are not openings behind the obstacles, whereas LCM could find openings in the goal direction.

b) The BCM environment model is obtained with radial projection, which matches the information gathered by robot’s sensors.

c) Both methods search a best local heading, however LCM stops when the robots is in the goal lane while BCM is continuously replanning the local heading.

d) Obstacle security distances are considered by the BCM method.

e) Overlapped obstacles are detected earlier with radial projection. For such a reason, BCM can find openings before LCM, resulting in an increment in the response time to the robot.

These improvements result greater reliability when the robot is moving in cluttered and dynamic environments.

The BCM method has solved some limitations observed in LCM. However, it is important to notice that the correct selection of the parameters is a crucial point in order to obtain the desired behavior of both methods. In this work the parameters of both methods have been adjusted according to our robot’s characteristics.

The computation time for the three methods have not been considered in the comparative experiences because it is much less than the period of the sensor readings.

ACKNOWLEDGEMENTS

This work has been partially supported by the Comisión Interministerial de Ciencia y Tecnología, CICYT; project TAP98-0955-C03-01.

We would like to thank Prof. Reid Simmons of the Robot Learning Lab in CMU for his influences in this work.

REFERENCES


