The Lane-Curvature Method for Local Obstacle Avoidance

Nak Yong Ko

Control and Instrumentation Eng.
Chosun Univ.
Kwang-Ju, 501-759 Korea
nyko@soback.kornet.nm.kr

Reid G. Simmons

School of Computer Science.
Carnegie Mellon Univ.
Pittsburgh, PA 15213
reids@cs.cmu.edu

Abstract

The Lane-Curvature Method (LCM) presented in this paper is a new local obstacle avoidance method for indoor mobile robots. The method combines the Curvature-Velocity Method (CVM) with a new directional method called the Lane Method. The lane method divides the environment into lanes, and then chooses the best lane to follow to optimize travel along a desired heading. A local heading is then calculated for entering and following the best lane, and 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.

Introduction

A local obstacle avoidance method for indoor mobile robots in unknown or partially known environments is investigated. The method should guide a robot through a collision free space along a given goal heading, or to a goal location, as fast as possible. For fast and smooth robot movement, it should be efficient for real-time implementation, and take the dynamics and physical limitations of the robot into account. Though many approaches efficiently yield commands guiding the robot through a collision free path, they often do not address the dynamics of the robot, and results in slow or jerky movement.

The directional approaches compute a direction for robot to head in, in Cartesian space or configuration space. The V-graph search methods [1], potential field methods [2, 3], and Vector Field Histogram method [4] belong in this category. Though they are simple and efficient in producing a directional command for collision free movement, they are not adequate for taking the robot dynamics into account. The Vector Field Histogram [4] method is able to achieve smoother navigation and is more successful in traveling through narrow openings. However, it is still not adequate to deal with vehicle dynamics, which can cause problems in cluttered environments.

Velocity space approaches, on the other hand, choose rotational velocity along with translational velocity, and can incorporate vehicle dynamics [5, 6, 7, 8, 9]. They typically presume that the robot travels along arcs of circles. The Curvature-Velocity Method (CVM) chooses a point in translational-rotational velocity space which satisfies some constraints and maximizes an objective function [5]. The constraints represent both the presence of obstacles and physical limitations on robot's velocities and accelerations. Though it produces reliable, smooth, and speedy navigation in office environments, it has some shortcomings. Often, at an intersection of corridors, it fails to guide the robot into an open corridor toward the goal direction. It often passes over some paths which are at right angles to the current robot orientation. Also, it sometimes lets the robot head towards an obstacle until the robot gets near the obstacle, even if there is a clear space around the obstacle. These problems all stem 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. In short, CVM pays less attention to collision free *directions* than do the directional approaches.

The Lane-Curvature Method (LCM), described in this paper, improves the velocity space approach by considering collision free direction as well as the collision free arc length. It uses a two-step approach to navigation. First, given a desired goal heading, a di-

rectional approach, called the Lane Method, chooses a "lane" for the robot to be in, taking into consideration obstacle avoidance, motion efficiency, and goal directedness. Then, the Lane Method calculates a local heading that will guide the robot either into, or along, that lane. Since, the Lane Method alone cannot account for the physical constraints of the robot motion, the local heading is supplied to CVM. Based on this heading, CVM produces translational and rotational velocity commands, taking into consideration the physical constraints of the robot.

The Lane Method chooses the direction to a wide and collision free opening since it decides heading direction based on the collision free distance and width of lanes. On the other hand, the VFH method chooses a direction to the opening with wide collision free *angular* range rather than an opening with wide width. So, it may force a robot into a narrow opening near the robot because even a narrow opening can offer wide collision free angular range to a robot if the opening is close to the robot. In this respect, the lane method can provide safer heading commands to CVM than the VFH.

The Curvature-Velocity Method

CVM formulates the local obstacle avoidance problem as one of constrained optimization in the velocity space of the robot. It determines translational velocity tv and rotational velocity rv, maximizing the objective function $f(tv, rv)$:

$$\begin{aligned} & f(tv, rv) = \alpha_1 \cdot dist(tv, rv) + \alpha_2 \cdot head(rv) \\ & + \alpha_3 \cdot speed(tv) \\ & dist(tv, rv) = d(tv, rv, OBS)/L \\ & head(rv) = 1 - |\theta_c - rv \cdot T_c|/\pi \\ & speed(tv) = tv/tv_{max} \end{aligned} \tag{1}$$

$d(tv, rv, OBS)$ is the arc distance that the robot can go with the curvature $c = rv/tv$ before hitting a set of obstacles OBS. The arc distance $d(tv, rv, OBS)$ is normalized to $dist(tv, rv)$ by some limiting distance L(three meters, in our implementation). $head(rv)$ is the normalized error in goal heading. It is defined to be the difference between the commanded heading θ_c (in the robot's local reference frame) and the heading the robot will achieve if it turns at rv for some time constant T_c. In other words, the objective function tries to have the robot achieve high speed movement close to the command heading direction, while traveling longer before hitting the obstacles.

The constraints maintaining the robot motion within its physical limitations are the followings:

$$\begin{aligned} & 0 \le tv \le tv_{max}, -rv_{max} \le rv \le rv_{max} \\ & rv \ge rv_{cur} - (ra_{max} \cdot T_{accel}) \\ & rv \ge rv_{cur} + (ra_{max} \cdot T_{accel}) \\ & tv \ge tv_{cur} + (ta_{max} \cdot T_{accel}) \end{aligned} \tag{2}$$

These constraints limit the robot's translational velocity, rotational velocity, translational acceleration, and rotational acceleration within the maximum values tv_{max}, rv_{max}, ta_{max}, and ra_{max}, respectively. The constraint $0 \le tv$ prohibits the robot from moving backwards. Here, T_{accel} is the time interval with which commands are issued.

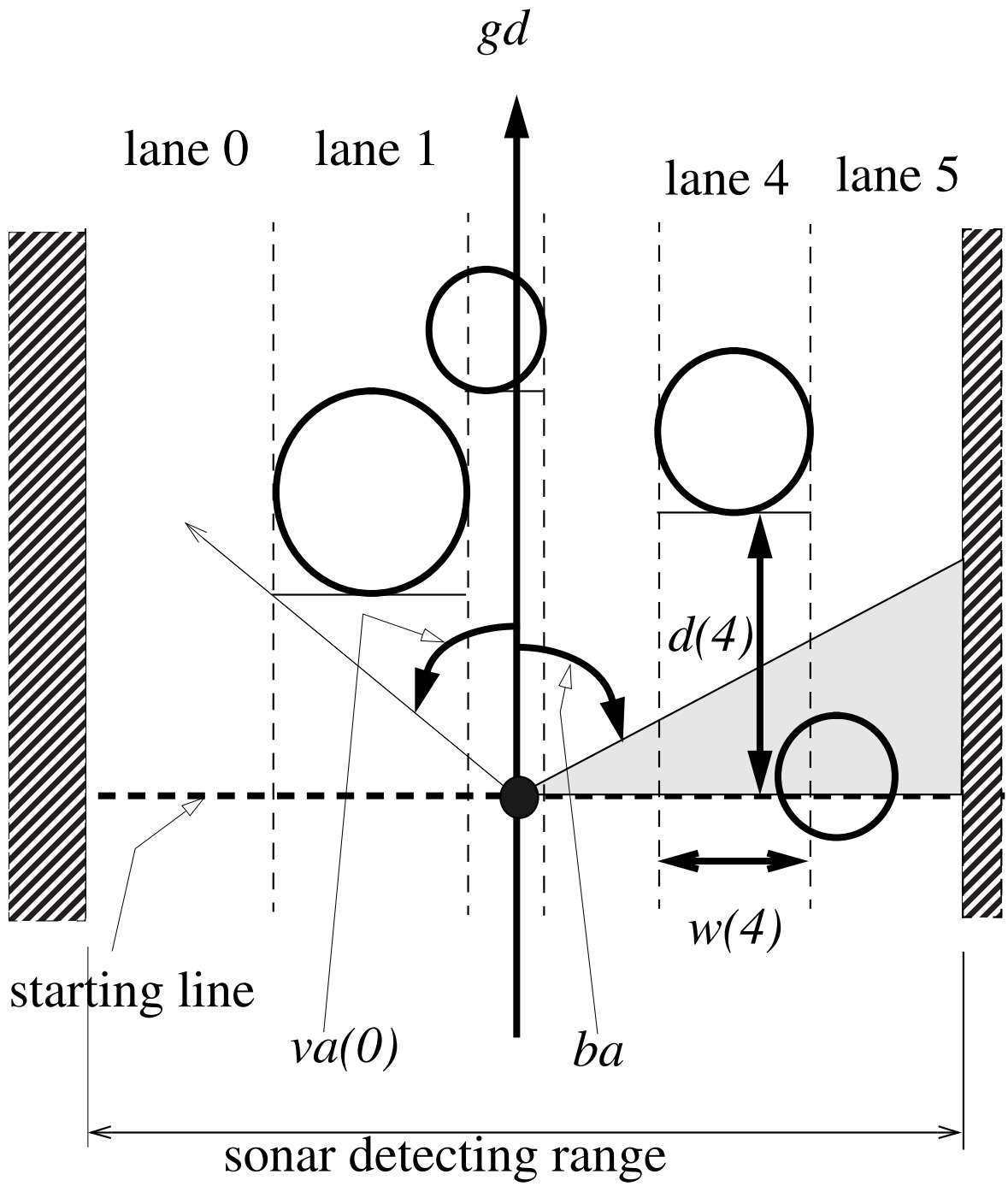
As a whole, CVM finds a point in translational-rotational velocity space satisfying the constraints (2), and maximizing the objective function (1). This produces rotational and translational commands that move the robot through a safe and goal directed path as fast as possible, within the robot's physical driving ability.

The Lane Method

To find a heading direction for collision free movement, the lane method divides the environment into lanes oriented in the direction of the desired goal heading. Then it selects the best lane for collision free and efficient motion. Finally, it calculates a heading direction to enter, or continue along, the selected lane.

Lanes

Lanes are constructed by determining the maximum collision-free distance to obstacles along the desired goal heading. Adjacent lanes with similar collision-free distances are merged. To facilitate the lane determination, and to match the implementation of CVM, the obstacles are approximated as circles, represented by their locations and radii. The radii of the obstacles are increased by the radius of the robot to convert from Cartesian space to configuration space obstacles, since our robot is also circular. The parameters describing the k-th lane are lane width $w(k)$, collision-free distance $d(k)$, and viewing direction $va(k)$, which is the angle at which a line from the robot to the lane passes through only collision-free areas. These parameters are depicted graphically in Figure 1.


gd
lane 0
lane 1
lane 4
lane 5
d(4)
w(4)
starting line
va(0)
ba
sonar detecting range

In our experiments, they are set to be $\beta_1 : \beta_2 : \beta_3 : \beta_4 = 6 : 1 : 6 : 1$. Maximizing the $f_s(k)$ selects a wide, collision-free, and motion-efficient lane.

Local Heading

If the robot is already in the best lane, CVM is sent the original desired goal heading, and uses this to command the robot. Otherwise, a local heading is calculated that will cause CVM to transfer lanes. Assume the n_s-th lane is selected as the best. Since the view angle $va(n_s)$ is the minimum collision-free angle to the n_s-th lane, the local heading hc should be $|va(n_s)| \leq |hc|$. Also, we confine the local heading to be within the blocking angle ba, that is $|hc| \leq |ba|$. So, the local heading hc becomes:

$$hc = va(n_s) + \delta * (ba - va(n_s))$$
$$\text{where,} \qquad (5)$$
$$0 \leq \delta \leq 1.0$$

The value δ determines how far the local heading is from the viewing angle of the selected lane. If $\delta = 0$, then the heading command is just the viewing angle, and there is no clearance for safe motion. If $\delta = 1.0$, heading command always directs to the extreme left hand side or right hand side. In our experiments, δ is set to 0.5. The relationship between the heading command, viewing angle, and blocking angle is shown in the Figure 2.

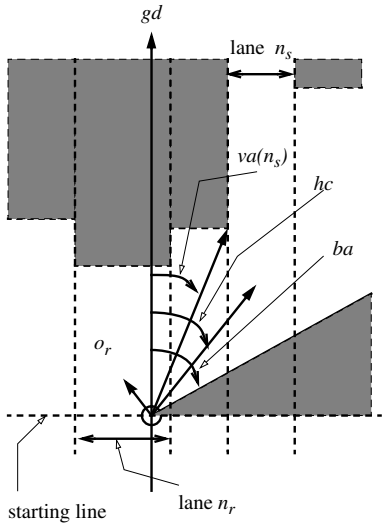


Figure 2: Determination of local heading

Experiments and Results

The LCM algorithm has been implemented and extensively tested on the Xavier mobile robot (Figure 3) [10]. Xavier is built on a four-wheel synchro-drive base, produced by RWI, and has independent control over translational and rotational velocities. For obstacle detection, it uses a ring of 24 sonars (data rate 2 Hz) and a 30 degree field of view front-pointing Nomadics laser range sensor. The base provides Xavier with dead-reckoning information at 8 Hz, which is the rate at which the LCM algorithm runs. The LCM algorithm runs on an on-board 200 MHz Pentium-Pro computer.

The α values of the CVM objective function (1) were determined through a number of empirical trials as the values resulting in the best safe, smooth, and efficient robot movement. The values used in the combined LCM approach differ from those used when CVM is the only obstacle avoidance mechanism. In LCM, they are set to be $\alpha_1 = 0.1$, $\alpha_2 = 0.6$, $\alpha_3 = 0.3$, while they are set to be $\alpha_1 = 0.6$, $\alpha_2 = 0.1$, $\alpha_3 = 0.3$ if CVM alone is used for obstacle avoidance. While α_1, which dictates the importance of long, collision-free arcs, is set high for obstacle avoidance in the CVM-only case, it is lowered in LCM because obstacle avoidance is fully addressed by the Lane Method. On the other hand, α_2, which dictates the importance of staying close to the goal heading, is set higher in LCM, to force the robot to adhere more closely to the heading command that is issued by the Lane Method.

For comparison, the results of CVM and LCM are shown for four environments: (1) turning a corner with three obstacles, (2) going through a corridor with an obstacle, (3) entering to a narrow corridor, and (4) turning right through a narrow entrance. The maximum translational and rotational velocities are set to be $tv_{max} = 50cm/sec$, $rv_{max} = 60°/sec$.

The results for the first environment are shown in the Figure 4 (note that in all the experiments, the robot has no initial knowledge of the environment – it is just provided with the desired heading gd). In this experiment, the desired goal heading gd is $-90°$. That is, the robot is commanded to find and go through a collision-free path in the direction $-90°$ from its initial orientation (that is, to the right in the figure). There are two possible collision-free paths: One is over the

Figure 3: The Xavier mobile robot

(a) LCM (b) CVM

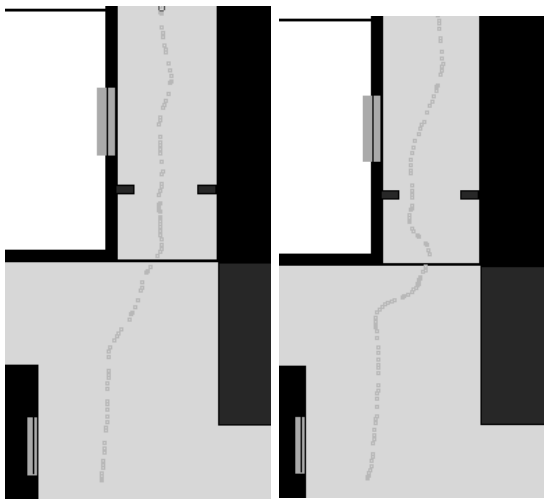
Figure 4: Turning at a corner avoiding obstacles

second obstacle, and the other is below the second obstacle which is narrower than the other. LCM finds the wider collision-free path successfully, but CVM first tries to find a collision-free path below the second obstacle. As the robot gets closer, it discovers that the collision-free path below the second obstacle is too narrow, and so CVM directs it back, and eventually finds the collision-free space. In this case, at first the CVM misses the wider collision-free path.

Figure 5 shows the results for the second environment. The goal direction is $gd = 0°$ (that is, to the right in the figure). The LCM forces the robot to steer away from the obstacle sooner than does CVM. CVM lets the robot head towards the obstacle until it gets too close to turn smoothly. This is because the CVM prefers longer collision-free distance of arc, rather than collision-free space itself. On the other hand, LCM can detect wide collision-free lane from earlier stage, and the avoidance motion begins earlier

(a) LCM (b) CVM

Figure 5: Avoiding an obstacle in a corridor



(a) LCM (b) CVM

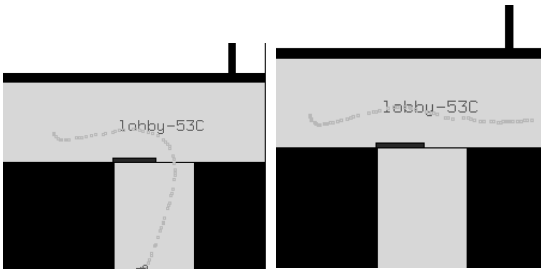
Figure 6: Entering into a narrower corridor

(in fairness to CVM, it usually handles such situations much more smoothly – this is just an extreme case).

Figure 6 shows results for the third environment. The goal direction is $gd = 0°$ (that is, heading up in the figure). Though there is a corridor in the direction gd, CVM guides the robot straight towards the wall, turning late to avoid it. With LCM, the robot notices the long open corridor, and enters that lane fairly early.

In Figure 7, results in the fourth environment are shown. The goal direction here is $gd = 90°$ (that is, heading down in the figure). LCM smoothly guides the robot into the correct corridor, while CVM fails to find the perpendicular corridor 90°, and continues straight (later turning down the next corridor). This result is similar to the result for environment 1, where the CVM passes over an opening and fails to find collision-free path.

The failure of CVM for these experiments can be explained using Figure 8. Since CVM prefers longer collision-free arc lengths, and α_1 is much greater than α_2, it prefers the path through the arc oc rather than



(a) LCM (b) CVM

Figure 7: Turning right through a narrow entrance

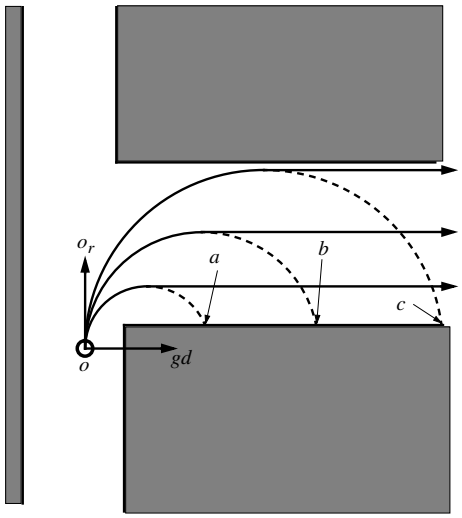


Figure 8: Problems of CVM

the path through the arc *oa* or *ob*. However, the path through *oc* is not better than the other paths, regarding the obstacle avoidance and motion efficiency. In the CVM-only case, it does not help to make α_1 less than α_2, however, since then CVM will be reluctant to turn the robot to avoid obstacles, preferring to keep heading in the goal direction. This is not a problem in the combined LCM approach, since the Lane Method supplies CVM with a local heading that will avoid obstacles (under the assumption of straight-line motion).

Conclusions

We have presented the Lane-Curvature Method (LCM) for local obstacle avoidance, which incorporates a velocity space method (CVM) with a directional method (Lane Method). The Lane Method determines a local heading which directs robot to a wide and collision-free lane. So, it resolves some problems of using the CVM only for obstacle avoidance, such as passing over a collision-free corridor in the goal direction. By using CVM to actually choose commands, LCM simultaneously controls the speed and heading of the robot and incorporates constraints from the robot dynamics.

The method has been implemented and tested on Xavier, a synchro-drive robot. Our extensive experiments show that, in many cases, LCM produces safer and smoother robot motion than does CVM (in many other cases, not shown here, their behavior is essentially identical). In particular, LCM can often guide the robot along collision-free paths that CVM misses.

This work shows that by combining the directional and curvature-based velocity-space methods, we can obtain an efficient and reactive local navigation algorithm which produces smooth and speedy, as well as safe, collision-free movement.

Acknowledgments

This work is supported by RRC of Chosun Univ., and KOSEF (Korea Science and Engineering Foundation). Thanks to Greg Armstrong for helping with many of the experiments. Richard Goodwin and Lonnie Chrisman provided the original inspiration for a lane-based method of local obstacle avoidance.

References

[1] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The International Journal of Robotics Research*, Vol. 5. No. 3, pp. 72-89, Fall 1986.

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, Vol. 5. No. 1, Spring 1986.

[3] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 1, pp. 23-32, February 1992.

[4] J. Borenstein and Y. Koren, "The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots," *IEEE Transactions on Robotics and Automation*, pp. 278-288, Vol. 7, No. 3, June 1991.

[5] R. G. Simmons, "The Curvature-Velocity Method for Local Obstacle Avoidance," In *Proc. International Conference on Robotics and Automation*, Minneapolis MN, April 1996.

[6] J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Stricos, and S. Thrun, "The mobile robot Rhino," *AI Magazine*, Vol. 16, No. 2, pp. 278-288, Summer 1995.

[7] W. Feiten, R. Bauer, and G. Lawitzky, "Robust obstacle avoidance in unknown and cramped environment," In *Proceedings IEEE International Conference on Robotics and Automation*, pp. 2412-2417, San Diego, CA, May 1994.

[8] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, Vol. 4, No. 1, pp. 23-33, 1997.

[9] A. Kelly, "An intelligent predictive control approach to the high speed cross country autonomous navigation problem," Tech Report CMU-CS-TR-95-33, School of Computer Science, Carnegie Mellon University, 1995.

[10] R. Simmons, R. Goodwin, K. Zita Haigh, S. Koenig and J. O'Sullivan, "A layered architecture for office delivery robots," In *Proc. Autonomous Agents '97*, pp. 245-252, Marina del Rey, CA, February 1997.