A Lane Based Obstacle Avoidance Method for Mobile Robot Navigation

Nak Yong Ko*
Department Information, Control, and Instrumentation Engineering and Factory Automation Center for Parts of Vehicles, Chosun University, Korea
Reid G. Simmons
School of Computer Science, Carnegie Mellon University, USA
Koung Suk Kim
Department Mechanical Information Engineering, Chosun University, Korea

This paper presents a new local obstacle avoidance method for indoor mobile robots. The method uses a new directional approach called the Lane Method. The Lane Method is combined with a velocity space method i.e., the Curvature-Velocity Method to form the Lane-Curvature Method(LCM). The Lane Method divides the work area into lanes, and then chooses the best lane to follow to optimize travel along a desired goal heading. A local heading is then calculated for entering and following the best lane, and CVM uses this local heading to determine the optimal translational and rotational velocities, considering some physical limitations and environmental constraint. By combining both the directional and velocity space methods, LCM yields safe collision-free motion as well as smooth motion taking the physical limitations of the robot motion into account.

Key Words : Mobile Robot, Robot Control, Lanes, Curvature, Goal Heading, Local Heading, Heading Command, Obstacle Avoidance

1. Introduction

A Local motion planning is one of the key issues of mobile robot control. Especially, an issue of practical importance is the local obstacle avoidance method which guides a robot through a collision-free space to a given goal heading, or to a goal location, in unknown or partially known environment. For fast and smooth robot movement, the method should be efficient for real-time implementation, and take the dynamics and physical limitations of the robot into account.

* Corresponding Author,
E-mail : nyko@mail.chosun.ac.kr
TEL : +82-62-230-7108; **FAX** : +82-62-224-1987
Depertment Information, Control, and Instrumentation Engineering and Factory Automation Center for Parts of Vehicles, Chosun University, Korea. (Manuscript **Received** March 17, 2003; **Revised** July 25, 2003)

There have been two major approaches for the local obstacle avoidance : directional approaches and velocity space approaches. Though many of them efficiently yield commands guiding the robot through a collision-free path, they often do not address the dynamics of the robot, and result in slow or jerky movement. This is the case for many of the directional approaches. As for the velocity space approaches, they often fail to guide a robot through collision-free space which is obviously easy to find.

The directional approaches compute the direction for a robot to head in, in Cartesian space or configuration space. The V-graph search method (Kant and Zucker, 1986), artificial potential field methods (Khatib, 1986 ; Hwang and Ahuja, 1992), and Vector Field Histogram (VFH) method (Borenstein and Koren, 1991) belong in this category. Though the potential field based approach is simple and easily extensible, they

have some drawbacks : problem of local minima, difficulty in doorway passage, oscillatory movement in a narrow corridor. Also, they are not adequate for taking the robot dynamics into account. One of the improved potential field approach, VFH method (Borenstein and Koren, 1991) achieved smoother navigation and more successful traveling through narrow openings (Kortenkamp et al., 1998). And then, there have been many efforts to improve these drawbacks of the potential field approach (Cho et al., 2001 ; Louste and Liegeois, 2002) until these days. And yet, since the approach pays major attention to collision-free direction, 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 driving the robot through collision-free space, and can incorporate vehicle dynamics (Simmons, 1996 ; Buhmann et al., 1995 ; Feiten et al, 1994 ; Fox et al., 1995 ; Kelly, 1995) into account. They typically presume that the robot travels along a trajectory of arc segments (Simmons, 1996) or parabolas (Ihn, 1996 ; Kwon, 1997). Sometimes nonholonomic property of wheeled mobile robots is taken into account by constructing the robot path with straight line segments connected by circular arcs (Esquivel and Chiang, 2002).

The Curvature-Velocity Method (CVM) chooses a point in translational-rotational velocity space which satisfies some constraint and maximizes an objective function (Simmons, 1996). The constraint represents both the presence of obstacles and physical limitations on robot's velocity and acceleration. CVM is used in the robot Xavier (Simmons et al., 1997) as a local collision avoidance method at CMU. 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 a narrow open corridor toward the goal direction. It also passes by some entrances toward the paths which are at right angles to the current robot orientation. Also, it sometimes lets a 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. In this respect, we devised the method LCM and implemented it on Xavier.

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 is a two-step approach to navigation. First, given a desired goal heading, a directional approach, called the Lane Method, chooses a "lane" for the robot to be in, taking obstacle avoidance, motion efficiency, and goal directedness into consideration. Then, the Lane Method calculates a local heading that will guide the robot either into, or along, the selected lane. Since the Lane Method alone cannot account for the physical constraint of the robot motion, the local heading is supplied to CVM as a heading command. Based on this local heading, CVM produces translational and rotational velocity commands, taking the physical constraint of the robot into consideration.

As the first step of LCM, the Lane Method concerns primarily about obstacle avoidance. The CVM step works to produce commands taking account of the physical limitations of the robot motion, as well as obstacle avoidance. Though obstacle avoidance is considered in both steps, they consider obstacle avoidance in different aspects. While the CVM counts on collision-free arc length for obstacle avoidance, Lane Method uses collision-free straight-line distance and collision-free lane width. Combining these two steps, LCM considers more aspects for obstacle avoidance than does either CVM-only or Lane Method.

Since the LCM uses CVM to yield final commands, it maintains the advantages of CVM over potential field approaches (Khatib, 1986) and Vector Field Histogram method (Borenstein and

Koren, 1991). Potential field approaches use vector sum of repulsive and attractive features to compute a local heading. Speed control is sometimes handled by choosing velocity proportional to the magnitude of the potential vector, or proportional to the distance to obstacles ahead. So, the potential field method yields less smooth path than the CVM, as shown in the reference (Simmons, 1996). Though there are some other methods studied in the context of off-line path planning, taking vehicle dynamics and non-holonomic constraints into account (Jacobs and Canny, 1989; Latombe, 1991), they generally require more computations than the LCM.

The Lane Method chooses the direction to a wide collision-free opening since it decides a local heading based on the collision-free distance and width of lanes. On the other hand, the directional method chooses a direction to an 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 directional method.

The rest of the paper is organized as follows. Section II describes the LCM which consists of the Lane Method and CVM. It explains how the work area is divided into lanes, how to select the best lane to follow, and the method determining the local heading direction to get into the lane. Finally it briefly reviews CVM which calculates translational and rotational velocity with reference to the local heading direction. In Section III, experimental results and discussions of implementing the LCM are shown. Also, the results are compared with those of experiments using the CVM, to present the improvements in collision avoidance motion. Then in Section IV, we draw some concluding remarks.

2. The Lane-Curvature Method (LCM)

The proposed method consists of two steps. First, the Lane Method calculates local heading direction for obstacle avoidance taking the goal direction and mobility of the robot into account. And then the CVM finds the translational and rotational velocity. The robot cannot change its heading direction instantaneously. So we cannot command the robot to follow the local heading direction immediately. Instead, the local heading direction obtained from the Lane Method is fed to the CVM. CVM calculates translational velocity and rotational velocity based on the local heading direction and considering the physical limitations of robot motion and collision avoidance.

Our approach takes care of collision avoidance twice, once in Lane Method and once in CVM. While the Lane Method considers collision avoidance assuming that the robot trajectory consists of straight line segments, CVM considers collision avoidance on the assumption that the trajectory consists of arcs. Therefore, the LCM, combining these two methods, results in much safer motion than the CVM only. Figure 1 depicts the outline of the LCM.

To find a safe local 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 local heading to enter, or continue along, the selected lane. CVM uses the local heading to find translational and rotational

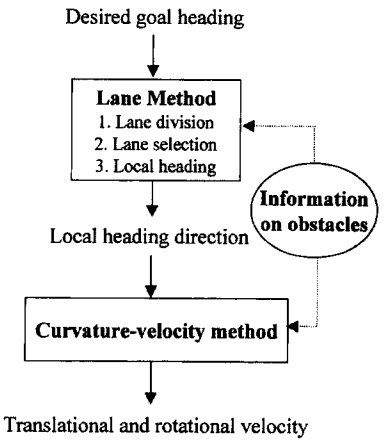


Fig. 1 Outline of the LCM

velocity.

2.1 Lane division

The 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 Cartesian space obstacles into 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 view angle $va(k)$, 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 2.

In Figure 2, the number of lanes N_L is six. The working area is divided into lanes within the range of the maximum obstacle sensing distance (set to 4 meters in our experiments). In determining lanes, we ignore the obstacles that are "behind" the robot. Since the robot is continually moving forward, we actually determine what obstacles are "behind" the robot as those whose angular distance from the desired goal heading is beyond some predefined angle limit. The angular limits for the clockwise and counter clockwise direction are determined individually. In our implementation, the blocking angle limit, ba for each direction is set to:

$$|ba| = \begin{cases} 90°, & \text{if there is no obstacle on the starting line} \\ 55°, & \text{if there are some obstacles on the starting line} \end{cases} \quad (1)$$

The collision-free distance of the k-th lane, $d(k)$ is defined as the distance the robot can go through the k-th lane before hitting obstacles, from the starting line.

The view angle for the k-th lane, $va(k)$ is the minimum angle from the desired goal heading gd to the collision-free direction to the k-th lane. In determining $va(k)$, it is assumed that the i-th lane ($i=0.1, \cdots, N_L-1$) is blocked at the distance $d(i)$ from the starting line.

A lane with very narrow lane width is merged to a neighboring lane by the following rule: If $w(h) \le w_{\min}$ and $d(h) > Min\{d(h-1), d(h+1)\}$ for some $1 \le h \le N_L-2$, then merge the h-th lane into the neighboring lane with the shorter collision-free distance. In the experiment, $w_{\min}$ is set to be 2.0 cm. Also, two lanes with similar collision-free distances are merged together using the following rule: If $|d(h)-d(h+1)| \le \triangle d_{\min}$, $0 \le h \le N_L-2$, then merge the lane with longer collision-free distance into the other lane. We set $\triangle d_{\min}=2.5$ cm in the experiment.

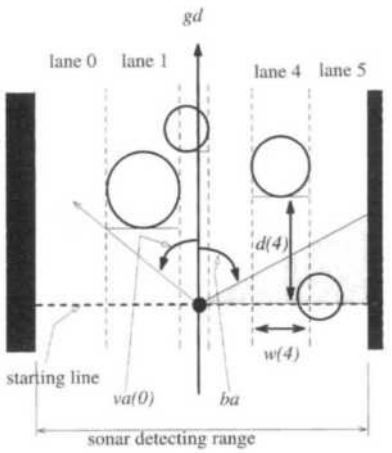


Fig. 2 Lanes and their describing parameters

2.2 Lane selection

Once lanes are constructed, the Lane Method chooses the best lane to be in for efficient and collision-free movement. For safe, collision-free movement, it is desired to go through a lane with longer collision-free distance and wider lane width. For efficient steering, smaller change of heading command is desired. Also, abrupt change of heading command due to noisy sonar reading can be prevented by keeping the change of heading command as small as possible. For fast and efficient robot motion, heading command closer to the current robot orientation o_r is preferred. To address the above discussions, we choose the following linear function $f_s(k)$ as a lane selection function.

$$f_s(k) = \beta_1 \cdot \bar{d}(k) + \beta_2 \cdot \bar{w}(k) - \beta_3 \cdot \overline{ad}_{va,c}(k) - \beta_4 \cdot \overline{ad}_{va,o}(k) \quad (2)$$

$$\bar{d}(k) = Min\{ d(k),\ D_{limit} \} / D_{limit} \quad (3)$$

$$\bar{w}(k) = Min\{ w(k),\ W_{limit} \} / W_{limit} \quad (4)$$

$$\overline{ad}_{va,c}(k) = Min\{ | va(k) - c_p |,\ C_{limit} \} / C_{limit} \quad (5)$$

$$\overline{ad}_{va,o}(k) = Min\{ | va(k) - o_r |,\ O_{limit} \} / O_{limit} \quad (6)$$

c_p : heading command of the previous sampling time

o_r : current oriention of the robot

Finding k maximizing the $f_s(k)$ selects a wide, collision-free, and motion-efficient lane. Since the view angle $va(k)$ is the minimum collision-free angular deviation from the goal heading gd to the k-th lane, we use it as a guiding direction to the k-th lane in the selection function (2).

Here, each term in $f_s(k)$ is limited and normalized by the corresponding maximum values, D_{limit}, W_{limit}, C_{limit}, and O_{limit}. The distance to an obstacle in a lane $d(k)$ is limited by the limit distance D_{limit}. If $d(k)$ is beyond D_{limit}, the lane is regarded as collision-free and the longer collision-free distance doesn't guarantee safer motion. Likewise, if the width of the lane is wider than the limit width W_{limit}, then the lane is wide enough to collision-free motion, and also, the wider lane width doesn't guarantee safer motion. In equation (5), the angular difference from the view angle to previous heading command is limited to C_{limit}. If the angular difference is beyond the limit, it is considered to have the same value C_{limit}. In equation (6), the angular difference from the view angle to the current robot orientation is limited to O_{limit}. If the angular difference is beyond the limit, it is considered to have the same value O_{limit}.

In equation (2) the terms $\overline{ad}_{va,c}(k)$ and $\overline{ad}_{va,o}(k)$ plays the role of both reducing the effect of noisy sonar reading of obstacle location and smoothing robot motion. Ultra sonic sensors used in our work are liable to yield wrong distance information due to noise, multiple reflection, and so on (Kang and Lim, 1999). Non-structural sonar noise is often random and disappears soon. So, even if there is no obstacle, sonar may detect false obstacles temporarily. In this case, sonar misreading may result in abrupt change of steering direction both before and after the misreading. The term $\overline{ad}_{va,c}(k)$ indicates preference for smaller change of heading command. Similarly, the term $\overline{ad}_{va,o}(k)$ indicates preference for heading command closer to the current robot orientation. Due to these two terms, the robot prefers to change its direction as small as possible from current orientation and current steering command. So, they can reduce the influence of false detection of obstacles due to sonar misreading.

The β values are the weights to be given to each term of the selection function and they are all positive. In our experiment, they are set to be $\beta_1 : \beta_2 : \beta_3 : \beta_4 = 6 : 1 : 6 : 1$. β_1 and β_2 are the weights for collision-free distance and lane width respectively. β_3 and β_4 are the weights for angular deviation of robot steering direction from previous heading command and from robot orientation respectively. If β_1 and β_2 are set to high, the robot selects a lane with wider width and longer distance to obstacles. If β_3 and β_4 are set to high, the trajectory becomes smoother. However, too high β_3 and β_4 doesn't allow the robot to change its course fast enough even if obstacles are detected in front of the robot motion. If they are two low, robot responds too sensitive to sonar detection of obstacles, and thus results in oscillatory robot trajectory. Besides, it may result in unnecessary avoidance motion in response to false detection of obstacles due to sonar misreading.

2.3 Local heading

If the robot is already in the best lane, the original desired goal heading is sent to CVM, and the CVM uses the goal heading to calculate translational and rotational velocity command. Otherwise, a local heading is calculated that causes CVM to transfer lanes to the best lane. 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

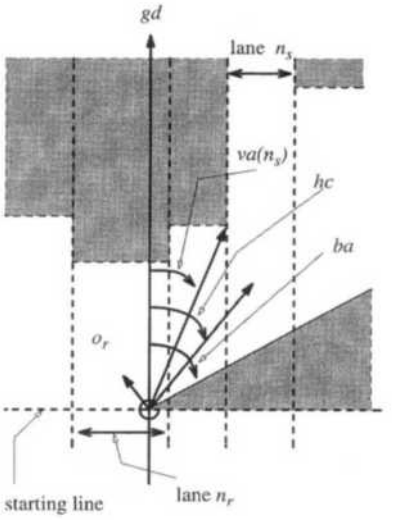


Fig. 3 Determination of local heading

heading hc becomes:

$$hc=\begin{cases} gd, \text{ if } n_r=n_s \\ va(n_s)+\delta\cdot(ba-va(n_s)), \text{ if } n_r\neq n_s \end{cases} \quad (7)$$

where, n_r is the number of the lane where the robot is in, and $0.0\leq\delta\leq1.0$.

The value δ determines how far the local heading is from the view angle of the selected lane. If $\delta=0$, then the local heading is just the view angle, and there is no clearance for safe motion. If $\delta=1.0$, local heading 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 local heading hc, view angle $va(n_s)$, and blocking angle ba is shown in the Figure 3.

Since the robot is not able to change its heading direction abruptly, local heading direction can not be used as a command directly to the robot. To command the robot to steer toward the local heading continuously, we use the curvature based method CVM which yields translational and rotational velocity command. In addition, the CVM considers collision avoidance again assuming that the robot trajectory consists of arcs.

2.4 Translational and rotational velocity-CVM

To produce optimal translational velocity and rotational velocity, the CVM (Simmons, 1996) requires heading command as an input. The local heading obtained by the lane method is fed to the CVM as an input of heading command.

The CVM formulates 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 $f(tv, rv)$:

$$f(tv, rv)=\alpha_1\cdot dist(tv, rv)+\alpha_2\cdot head(rv)+\alpha_3\cdot speed(tv) \quad (8)$$

$$dist(tv, rv)=d(tv, rv, OBS)/L \quad (9)$$

$$head(rv)=1-|\theta_c-rv\cdot T_c|/\pi \quad (10)$$

$$speed(tv)=tv/tv_{\max} \quad (11)$$

$d(tv, rv, OBS)$ 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). The $head(rv)$ is the normalized error in goal heading. It is defined to be the normalized difference between the heading command θ_c (in the robot's local reference frame) and the heading the robot will achieve if it turns at the rotational velocity rv for some time constant T_c (T_c is the command issuing period). In other words, the objective function tries to have the robot achieve fast movement close to the heading command, while traveling longer before hitting the obstacles.

In the equation (8), the α_1 is the weight for long collision-free arc length. α_2 is the weight for steering close toward the heading command. α_3 is the weight for fast motion of the robot. If α_1 becomes higher, the robot gives more attention to collision avoidance while it moves slowly and deviates more from heading command. If α_2 becomes higher, the robot steers more close to heading command while it moves slowly and the possibility of collision becomes higher. If α_3 becomes higher, the robot moves faster at the expense of higher possibility of collision and larger deviation from the heading command. α values of the CVM were determined through a number of empirical trials as the values resulting in best safe,

smooth, and efficient robot movement.

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

$$\begin{aligned} &0 \le tv \le tv_{\max}, \quad -rv_{\max} \le rv \le rv_{\max} \\ &rv \ge rv_{cur} - (ra_{\max} \cdot T_{accel}) \\ &rv \le rv_{cur} + (ra_{\max} \cdot T_{accel}) \\ &tv \ge tv_{cur} - (ta_{\max} \cdot T_{accel}) \\ &tv \le tv_{cur} + (ta_{\max} \cdot T_{accel}) \end{aligned} \tag{12}$$

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 backward. Here, T_{accel} is the time interval with which commands are issued, and is set to be the same as the T_c of (10).

As a whole, CVM finds a point in translational-rotational velocity space satisfying the constraints (12), and maximizing the objective function (8). 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. Though it doesn't completely consider dynamics of robot motion, it prevents abrupt change of robot motion and yields smoother and faster motion than the directional approaches.

3. Experiments and Results

The LCM algorithm has been implemented and extensively tested on the Xavier mobile robot (Figure 4) (Simmon et al., 1997). 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 ($T_c = T_{accel} = 1/8$ sec $= 125$ msec). The LCM algorithm runs on an on-board 200 MHz Pentium-Pro computer.

In the experiment, obstacles are modeled as circles. Sometimes, multiple circular obstacles of different size form an obstacle. The radius of a circular obstacle is determined using the distance data fed from the sonars or laser range sensor. If a sonar detects an obstacle, it is assumed that a circular obstacle occupies 15 degrees of space. So, as the distance to the obstacle is large, the radius of the obstacle becomes large. The Figure 5 explains how the radius is calculated.

In the Figure 5, an obstacle is detected at the distance l from the center of the sonar array ring. $\Delta\theta$ is the angular space between the directions of sonar radiation — in the experiments, 15 degrees. Since $(l+r)\sin(\Delta\theta/2) = r$, we get the radius r of the obstacle as the following.

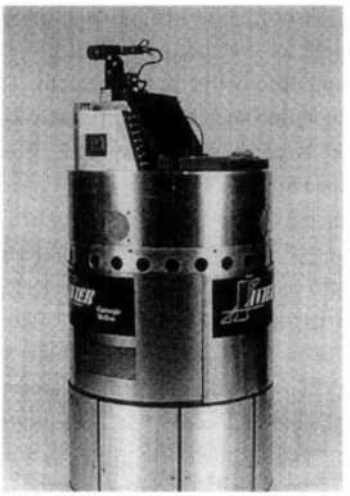


Fig. 4 The Xavier mobile robot

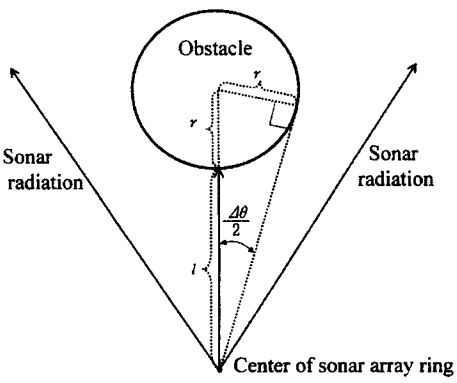


Fig. 5 The method calculating the radius of an obstacle

$$r = l \frac{\sin(\Delta\theta/2)}{1-\sin(\Delta\theta/2)}$$

As in the experiment, if $\Delta\theta = 15°$ for the sonar array, the equation becomes $r = 0.501 \cdot l$.

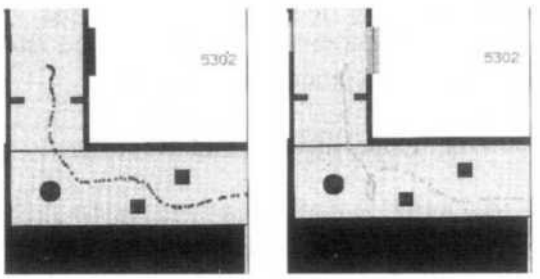
If the obstacle is within the angular range of Nomadics laser range sensor (30° in the paper), we can detect the configuration of the environment more precisely, since the laser range sensor scans the range data much more closely than the sonar array. Typically the angular space $\Delta\theta$ for the case of laser range sensors is $\Delta\theta = 0.5°$. The method of calculating the radius of the obstacles is the same as for the case of sonar, as described in the equation above.

The α values of the CVM objective function (8) were determined through a number of empirical trials as the values resulting in 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 (Simmons, 1996). 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, the α_2, which dictates the importance of steering close toward the heading command, is set higher in LCM, to force the robot to adhere more closely to the local heading (or heading command) that is issued by the Lane Method.

To show the improvement of obstacle avoidance performance, the results of the LCM and CVM only are compared for the four environments: (1) turning at a corner with three obstacles, (2) going through a corridor with an obstacle, (3) entering to a narrower corridor, and (4) turning right into a narrow entrance. The maximum translational and rotational velocities are set to be $tv_{max} = 50$ cm/sec, $rv_{max} = 60°$/sec. Note that in all the environments, the robot has no prior knowledge of the environments and it is just provided with the desired goal heading gd. The information on environments is given during robot motion from the sonars.

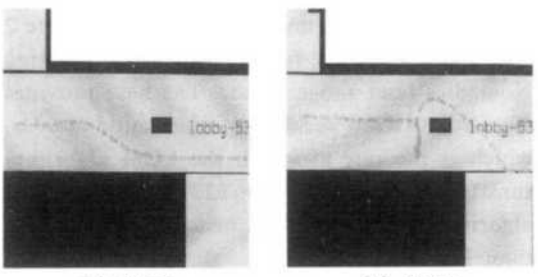
The results for the first environment are shown in the Figure 6. In this experiment, the robot starts to move downward from the top, and goal heading gd is $-90°$. That is, the robot is commanded to find and go through a collision-free path in the direction $-90°$ from its initial orientation (that is, from the top to the lower right in the figure). There are two possible collision-free paths: One is over the second obstacle, and the other is below the second obstacle which is narrower than the other. While LCM finds the wider collision-free path successfully, the 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 7 shows the results for the second environment. The robot starts from left to right, and goal heading is $gd = 0°$ (that is, to the right in the



(a) LCM (b) CVM

Fig. 6 Turning at a corner avoiding obstacles



(a) LCM (b) CVM

Fig. 7 Avoiding an obstacle in a corridor

figure). The LCM forces the robot to steer away from the obstacle earlier than does CVM. CVM lets the robot head towards the obstacle until the robot 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.

Figure 8 shows results for the third environment. The robot starts to move upward from the bottom, and goal heading is $gd=0°$ (that is, heading up in the figure). Though there is a narrow open corridor in the direction of gd, CVM guides the robot straight towards the wall, turning late to avoid it. With LCM, the robot notices the long open corridor, and steers toward the corridor fairly early.

In Figure 9, results for the fourth environment are shown. The robot starts from the left to right, and goal heading here is $gd=90°$ (that is, turning downward in the figure). LCM smoothly guides the robot through the entrance into the correct corridor, while CVM fails to find the perpendicular entrance to the corridor 90° apart from its way, and continues straight (later turning down the next corridor). This result is similar to the result for the first environment, where the CVM passes over a wide opening and fails to find collision-free path. In this environment, the width of the entrance to the perpendicular corridor is 120 cm. If the width of the entrance increases by 20 cm, CVM can also find the perpendicular corridor.

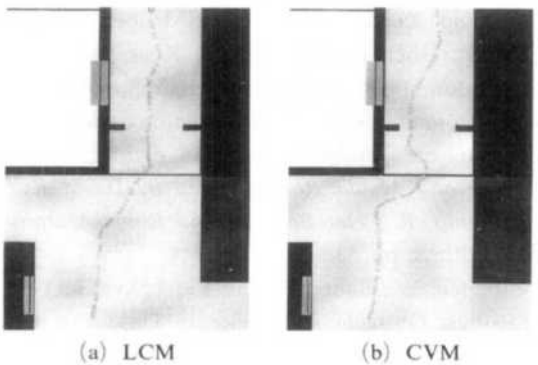


Fig. 8 Entering into a narrower corridor

Through experiments, we found that proper selection of α values is critical to have the LCM to overcome the shortcomings of CVM. As the α_2 decreases below 0.6 and α_1 increases above 0.1, the LCM produces similar paths as the CVM-only does. Also, it is noticeable that since the α values in LCM differ from those for CVM-only case, the CVM stage in LCM pays less attention to obstacle avoidance than does in CVM-only case.

Though the arrangements of the work area shown above are of extreme cases, they show typical circumstances where CVM fails. The failure of CVM for these experiments can be explained using Figure 10. Since the CVM prefers longer collision-free arc length, and α_1 is much

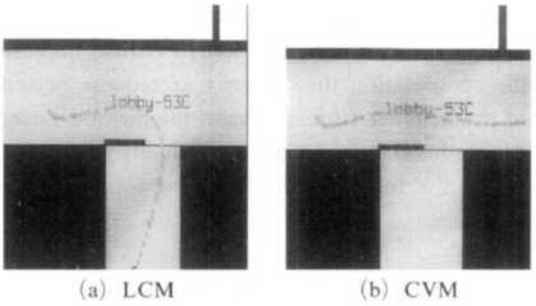


Fig. 9 Turning right through a narrow entrance

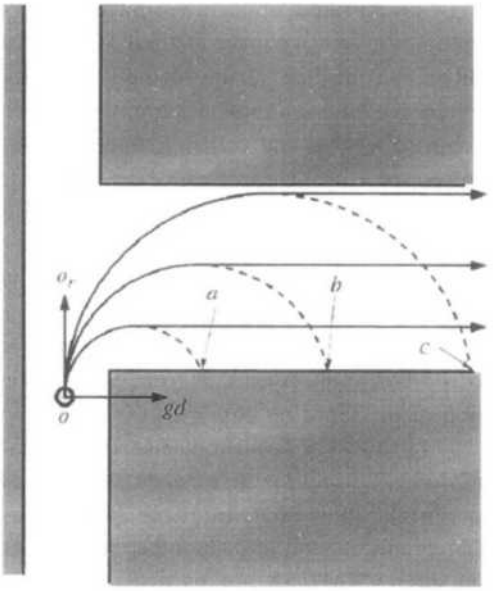


Fig. 10 Problems of CVM

greater than α_2, it prefers the path through the arc oc rather than 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, because the Lane Method supplies CVM with a local heading (heading command) that will avoid obstacles under the assumption of straight-line motion.

As shown in these results, LCM overcomes some of the CVM's shortcomings by incorporating the Lane Method, while maintaining advantages of CVM. The disadvantage of the proposed method is that it requires more computations than the CVM. As it may, it is not so critical because the computation time doesn't exceed the period of obstacle detection. At least, the method issues translational and rotational velocity command before the robot updates sensor information on obstacles.

4. Conclusions

This paper presents 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 a robot to a wide and collision-free lane. So, it resolves some problems of using the CVM-only, such as proceeding past a collision-free corridor in the goal direction. By using CVM to actually choose velocity commands, LCM simultaneously controls the speed (translational velocity) and heading (rotational velocity) of the robot and incorporates some constraint from physical limitation of the robot motion.

The method has been implemented and tested on Xavier, a synchro-drive robot. Our extensive experiments show that, in many critical cases, LCM produces safer and smoother robot motion than does CVM-only. In particular, LCM can often guide the robot along collision-free path which CVM misses.

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

In our implementation, a ring of 24 sonars is used for obstacle detection. As well known, sonars are subject to noise and sporadic false readings. The inaccurate range data deteriorates reliable and fast robot motion even though the obstacle avoidance algorithm is good enough. To make an obstacle avoidance algorithm work well, a method to obtain reliable, fast, and accurate information on obstacles should be employed for safe and efficient navigation.

Acknowledgment

This study was supported by research funds from Chosun University, 2000. Thanks to Greg Armstrong for helping with many of the experiments.

References

Borenstein, J. and Koren, Y., 1991, "The Vector Field Histogram-Fast obstacle avoidance for mobile robots," *IEEE Trans. Robotics Automat.*, Vol. 7, No. 3, pp. 278~288.

Buhmann, J., Burgard, W., Cremers, A. B., Fox, D., Hofmann, T., Schneider, F., Strikos, J. and Thurn, S., 1995, "The mobile robot Rhino," *AI Magazine*, Vol. 16, No. 2, pp. 278~288.

David Kortenkamp, Marcus Huber, Charles Cohen, Ulich Raschke, Frank Koss, and Clare Congdon, 1998, "Integrating High-Speed Obstacle Avoidance, Global Path Planning and Vision Sensing on a Mobile Robot," in *Artificial Intelligence and Mobile Robots edited by David Kortenkamp, R. Peter Bonasso and Robin Murphy*, MIT Press, pp. 53~71.

Feiten, W., Bauer, R. and Lawitzky, G., 1994, "Robust Obstacle Avoidance in Unknown and Cramped Environment," In Proc. IEEE Int. Conf. Robotics and Automation, pp. 2412~2417.

Fox, D., Burgard, W. and Thrun, S., 1995, "The Dynamic Window Approach to Collision Avoidance," Tech Report IAI-TR-95-13, CS department, University of Bonn.

Hwang, Y. K. and Ahuja, N., 1992, "A Potential Field Approach to Path Planning," *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 1, pp. 23~32.

Hye-Kyung Cho, Young-Jo Cho and Bum-Jae You, 2001, "Integration of Schma-Based Behavior and Variable-Resolution Cognitive Maps for Stable Indoor Navigation," Proc. *IEEE International Conference on Robotics and Automation*, pp. 3618~3623, Seoul, Korea.

Ihn, N. G., 1996, "A Global Collision-Free Path Planning Using Parametric Parabola Through Geometry Mapping of Obstacles in Robot Work Space," *KSME International Journal*, Vol. 10, No. 4, pp. 443~449.

Jacobs, P. and Canny, J., 1989, "Planning Smooth Paths for Mobile Robots," In Proc. *IEEE Int. Conf. on Robotics and Automation*, Scottsdale AZ, pp. 2~7.

Kang, S. K. and Lim, J. H., 1999, "Sonar Based Position Estimation System for an Autonomous Mobile Robot Operating in an Unknown Environment," *KSME International Journal*, Vol. 13, No. 4, pp. 339~349.

Kant, K. and Zucker, S. W., 1986, "Toward Efficient Trajectory Planning : The Path-Velocity Decomposition," The Int. *Journal of Robotics Research*, Vol. 5. No. 3, pp. 72~89.

Kelly, A., 1995, "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.

Khatib, O., 1986, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," The Int. *J. Robotics Research*, Vol. 5. No. 1.

Latombe, J. C., 1991, "Robot motion planning," Kluwer Academic Publishers.

Louste, C. and Liégeois, A., 2002, "Path Planning for Non-Holonomic Vehicles : A Potential Viscous Fluid Field Method," *Robotica*, Vol. 20, No. 3, pp. 291~298.

Simmons, R. G., 1996, "The Curvature-Velocity Method for Local Obstacle Avoidance," *IEEE International Conference on Robotics and Automation*, Minneapolis MN.

Simmons, R., Goodwin, R., Zita Haigh, K., Koening, S. and O'Sullivan, J., 1997, "A Layered Architecture for Office Delivery Robots," *Proc. Autonomous Agents '97*, pp. 245~252, Marina del Rey, CA.

Tai Hun Kwon, 1997, "A Collision-Free Path Planning Using Linear Parametric Curve Based on Geometry Mapping of Obstacles," *Transactions of KSME A*, Vol. 21, No. 12, pp. 1992~2007.

Wilson D. Esquivel and Luciano E. Chiang, 2002, "Nonholonomic Path Planning Among Obstacles Subject to Curvature Restrictions," *Robotica*, Vol. 20. No. 1, pp. 49~58.