Proceedings of the 1998 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
Victoria, B.C., Canada • October 1998

Fast Local Obstacle Avoidance under Kinematic and Dynamic Constraints for a Mobile Robot

C. Schlegel
Research Institute for Applied Knowledge Processing (FAW)
PO-Box 2060, D-89010 Ulm, Germany
schlegel@faw.uni-ulm.de

Abstract

This paper presents an efficient approach for reactive collision avoidance taking into account both vehicle dynamics and nonholonomic constraints of a mobile robot. Motion commands are generated by searching the space of actuating variables. Vehicle dynamics are considered by restricting the search space to values which are reachable within the next time step. The final selection among admissible configurations is done by an objective function which trades off speed, goaldirectedness and remaining distance until an obstacle is hit when moving along the chosen path. The presented approach differs from previous ones in the selective use of precalculated lookup tables. These are the key to efficiency, and they especially allow the use of any-shaped robot contours. Furthermore, obstacle information from different sources can easily be considered without preprocessing. Extensive experiments on different robots have shown robust operation in dynamic and unprepared indoor environments with speed up to 1m/s.

1 Introduction

Indoor mobile service robots have to be able to reliably carry out orders even in partially unknown and cramped environments. Safe operation is becoming more and more important as many of the applications are in busy environments, which has important impacts on suitable algorithms. Many approaches ignored vehicle dynamics and kinematic constraints or assumed circular robots. Thus, these models are often not suited to be used in allday indoor environments or at least they allow only slow motions in environments not too cramped.

In many robots the lowest level of motion control consists of a local obstacle avoidance which receives goal directives from other modules. The most promising approaches of local obstacle avoidance operate either in velocity space or in the space of steering angles. Periodically only a short time interval is considered when computing the next motion command. During such a time interval constant values of the actuating variables are assumed. This results in trajectories approximated by geometrical primitives such as circular arcs. A crucial point is the calculation of the remaining free path length under the current obstacle configuration when driving with a certain setting (fig.1). This length e.g. constrains the allowed maximum speed and thus the possible paths for the next cycle.

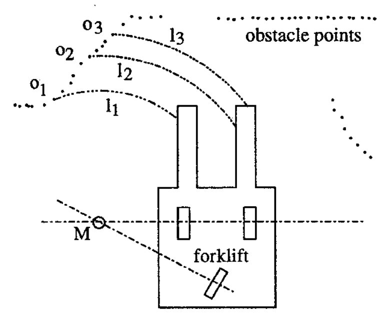

Figure 1: Problem of calculating free path length, e.g. forklift with tricycle kinematic and laser scan. For some selected obstacle points o_x the remaining path length l_x is shown when driving with a certain steering angle.

Since the available path length varies depending on the obstacle configuration it has to be recalculated at each cycle. In realtime, this can be done only for circular robots, even if the actuacting variables are discrete and not too many obstacles are around.

The main contribution of the presented method is the consequent exploitation of precalculated lookup tables which significantly reduces the number of required calculations during one cycle. This allows one to avoid the usually made simplifications concerning the shape of the robot, the vehicle dynamic, and its kinematic. Furthermore, many different obstacle information such as raw laserscan data and different map information can be considered.

The remainder of this paper is organized as follows. After discussing some previous work settled in this area in section 2, the use of lookup tables is presented in section 3. Implementation details and experimental results are summarized in section 4, followed by a discussion of further research issues.

2 Related Work

Many well-known local obstacle avoidance methods are based on the computation of a direction into which the robot shall move. Prominent examples are potential field methods or vector field histograms [4]. Typically these approaches do not take into account that vehicles move along arcs or have complex shapes. This is especially mandatory in cluttered environments.

Recently, local methods have been reported which incorporate vehicle dynamics by selecting e.g. steering commands rather than travel direction. The steering angle field method [1] is one example which has been applied to rectangular robots with tricycle or equivalent kinematic. All steering angles are rejected that move the robot along a path colliding with an obstacle within a speed-dependent time-interval. Different selection functions can be used to determine the final steering angle. Within this approach no dynamic aspects are considered and speed control is an iterative process between the steering angle field module and a pilot module.

The dynamic window approach operates directly in velocity space [2]. Dynamic constraints are used to reduce the velocity search space to values reachable within the next cycle. An objective function selects those settings which e.g. combine high velocity with close heading to the goal. To allow realtime operation, the method's implementation assumes a circular robot. Furthermore, obstacles are only described by an obstacle line field. Although our method is based on the same basic ideas the lookup table implementation avoids these restrictions.

The curvature velocity method [5] is very similar but formulates the local obstacle avoidance problem as one of constrained optimization. This approach also makes several simplifying assumptions to speed up distance calculation.

3 The Method

The search for commands controlling the motion of the robot is carried out directly in the space of actuating variables. Using e.g. a synchrodrive this is the space of velocities. The search space which is considered within one cycle is limited to a window whose values can be reached from the current state under the dynamic constraints of the vehicle. Furthermore, only those values are considered which are safe with respect to obstacles. Among the remaining values the final setting for the next cycle is chosen based on an objective function.

3.1 Calculation of Motion Commands

The space of actuating variables V_s of a synchrodrive is defined by the translational velocity v and rotational velocity w. For a differential drive, V_s is given by the velocity of the left and the right wheel. For a tricycle kinematic it is defined by the speed of the steering wheel and the steering angle. All these kinematics have in common that the resulting trajectories are composed of circular arcs. They can be described by a curvature c. In the case of a synchrodrive the curvature is given by the ratio w/v (for $v = 0$ a special curvature value is assigned which describes rotation without translation)[1]. The admissible velocities V_r are given as the intersection $V_s \cap V_a \cap V_d \cap V_o \cap V_f$ (fig.2). V_a constrains the search to those values which are reachable within the next cycle. V_d selects those velocities which allow the robot to stop without collision. V_o constrains the maximum velocity depending on the side clearance. V_f forbids those settings which would result in unsafe operation. For example it is not allowed to drive too fast through narrow curves.

As an example we take a look at a synchrodrive. v_c denotes the current translational velocity, w_c the rotational velocity, a_v and a_w the maximum translational and rotational accelerations and $\mathcal{O}$ is the set of obstacles. $d(c, \mathcal{O})$ is the collision free path length when driving with curvature c and obstacles $\mathcal{O}$. $vel_v(\mathcal{O})$ and $vel_w(\mathcal{O})$ denote maximum allowed velocities depending on side clearance (see section 3.3 for details). Then V_a, V_d and V_o are calculated as follows [2].

$$\begin{aligned} V_a = \{(v,w) \mid \quad & v \quad \in [v_c - a_v \cdot \Delta t, v_c + a_v \cdot \Delta t], \\ & w \quad \in [w_c - a_w \cdot \Delta t, w_c + a_w \cdot \Delta t]\} \end{aligned}$$

[1]Differential drive: $v = \frac{v_l - v_r}{(v_r - v_l} \cdot \frac{b}{2}$, $w = \frac{v_r - v_l}{b}$, b is lateral wheelbase of robot. Tricycle: $c = \frac{1}{r} = \frac{\sin\alpha}{d}$, α is steering angle, d is the wheelbase

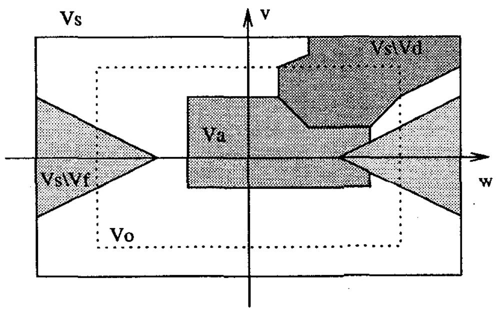

Figure 2: Velocity space of synchrodrive

$$\begin{aligned} V_d = \{(v,w) \mid \; & |v| \leq \sqrt{2 \cdot d(c,\mathcal{O}) \cdot a_v}, \\ & |w| \leq \sqrt{2 \cdot d(c,\mathcal{O}) \cdot a_w}\} \\ V_o = \{(v,w) \mid \; & |v| \leq vel_v(\mathcal{O}), \\ & |w| \leq vel_w(\mathcal{O})\} \end{aligned}$$

In the case of a tricycle the constraints look very similar. They are simpler because the curvature depends only on the steering angle α. The change of the steering angle within one time step is restricted by the maximum steering velocity v_α.

$$\begin{aligned} V_a = \{(v,\alpha) \mid \; & v \in [v_c - a_v \cdot \Delta t, v_c + a_v \cdot \Delta t], \\ & \alpha \in [\alpha_c - v_\alpha \cdot \Delta t, \alpha_c + v_\alpha \cdot \Delta t]\} \\ V_d = \{(v,\alpha) \mid \; & |v| \leq \sqrt{2 \cdot d(c,\mathcal{O}) \cdot a_v}\} \\ V_o = \{(v) \mid \; & |v| \leq vel_v(\mathcal{O})\} \end{aligned}$$

3.2 Objective Function

For the remaining values V_r of the actuating variables an objective function $G(v,w,\mathcal{O})$ is calculated. The objective function favours high speed and curvatures that can be followed longer before hitting obstacles. Furthermore, it supports heading towards the goal [5]:

$$\begin{aligned} G(v,w,\mathcal{O}) &= \alpha_1 \cdot speed(v) + \alpha_2 \cdot distance(c,\mathcal{O}) \\ &+ \alpha_3 \cdot heading(w) \\ speed(v) &= v/v_{max} \\ distance(c,\mathcal{O}) &= d(c,\mathcal{O})/L \\ heading(w) &= 1 - \mid \Theta_g - w \cdot \Delta t \mid /\pi \end{aligned}$$

The different terms are normalized to $[0,1]$. L is the maximum considered free path length, Θ_g is the goal heading relative to the robot's orientation. The α values allow relative weighting of the terms.

3.3 The Use of Lookup Tables

To use lookup tables, the space of actuating variables and the local environment relative to the robot are mapped on a discrete grid. The following lookup tables are used.

- $T_{command}[i_v, i_w]$ provides a curvature index i_c which is used to access all the values related to this curvature. V_f is included into this lookup table by using a distinguished value to indicate whether (v,w) is an allowed setting.
- $T_{distance}[i_x, i_y, i_c]$ provides the distance along the path with curvature i_c considering an obstacle at $[i_x, i_y]$. Only obstacles within a certain distance to the robot are considered. The local environment is called the cartesian space.
- $T_{vel}[i_x, i_y]$ is the lookup table for V_o. For each grid cell in cartesian space, the allowed maximum velocities are stored. These values are chosen in such a way that obstacles close to the robot's contour slow down the robot.

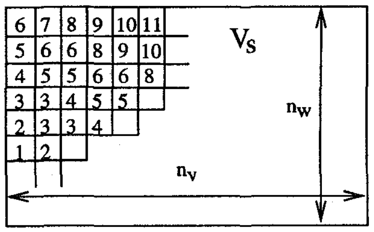

Figure 3: $T_{command}$ of synchrodrive. Each cell provides a curvature index. Notice that there are far less curvature indices than cells which is important for the size of the distance lookup table.

The calculation of the lookup table $T_{command}$ depends on the kinematic of the robot. For a tricycle, $T_{command}(v,\alpha)$ is directly derived from the steering angle α because the curvature is not affected by the velocity v. For a synchrodrive and a differential drive the two-dimensional space of actuating variables is mapped onto the one-dimensional space of curvatures. In the case of a synchrodrive if the discretization of V_s results in $n_v \cdot n_w$ different commands, only $n_c = (2(n_v + n_w) - 4 + 2)$ different curvatures are distinguished (fig.3). They are defined by the boundary cells of V_s including two separate values indicating forbidden settings as defined by V_f and stalling. For inner cells of V_s the index of the closest matching curvature out of the set defined by the boundary cells is entered.

Now the lookup table $T_{distance}$ can be precalculated. For every curvature index i_c and every considered cell $[i_x, i_y]$ in cartesian space the remaining path length is determined by calculating the intersection of the robot's contour with circle C_{i_x,i_y}. C_{i_x,i_y} is the circle with center point M through the obstacle cell $[i_x, i_y]$. M is the center of rotation associated with curvature i_c (fig.1). The arc length is stored in $T_{distance}[i_x, i_y, i_c]$. This calculation is very time consuming for an any-shaped robot, however, it must be done only when the robot's shape changes. This is e.g. the case when items are picked up or a manipulator is deployed.

3.4 Online Use of Lookup Tables

We now turn to the description of the steps to be performed online within one cycle. The current values of the actuating variables with the dynamic constraints define a window in the search space V_s.

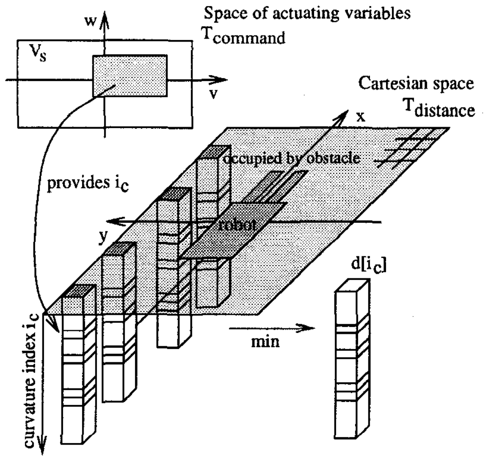

Figure 4: Lookup of distance $d(c, \mathcal{O})$ for a forklift robot. Occupied cells in cartesian space select curvature dependent distances, indices in V_r determine which curvatures have to be considered.

For each cell in $V_s \cap V_a$ the curvature index i_c is read from $T_{command}$. If the index indicates a forbidden actuating variable setting, these values are not considered anymore because they do not belong to V_r due to V_f. Otherwise, the distance $d(c, \mathcal{O})$ has to be determined. All known obstacle information such as raw laserscans and map data is mapped into cartesian space (fig.4). The collision free path length $d[i_c]$ considering curvature i_c is the minimum of all distance values $T_{distance}[i_x, i_y, i_c]$ with indices $[i_x, i_y]$ selecting all occupied grid cells in cartesian space. Within this step, the maximum allowed velocities concerning side clearance as defined by V_o are also looked up. The maximum allowed velocities are given by the minimum of $T_{vel}[i_x, i_y]$ for all occupied cells. Since only few curvature indices have to be considered within one cycle and a minimum operation is not very time consuming, the lookup of the distance values can be done very fast.

Now the values are checked against V_o and V_d. If the cell belongs to V_r, the objective function G is calculated. The motion control commands corresponding to the best value of the objective funtion are executed within the next cycle.

The following pseudocode illustrates the different steps. Designators are for synchrodrive, *index()* maps from real values to indices.

```
$v = 0,\ w = 0$
for $i_v \in [index(v_c - a_v \cdot \Delta t), index(v_c + a_v \cdot \Delta t)]$ {
  for $i_w \in [index(w_c - a_w \cdot \Delta t), index(w_c + a_w \cdot \Delta t)]$ {
    $i_c = T_{command}[i_v, i_w]$
    if ($i_c$ == FORBIDDEN) {
      (ignore these values)
    } else {
      if $d[i_c]$ not already calculated {
        $d[i_c] = \min_{o \in \mathcal{O}} T_{distance}[index(o_x), index(o_y), i_c]$
        if first time going through all obstacles $\mathcal{O}$ {
          calculate $vel_v(\mathcal{O}), vel_w(\mathcal{O})$
        }
      }
      check $V_o$ and $V_d$
      if both conditions hold {
        calculate objective function $G$
        if new maximum store $v, w$
}}}}
$v, w$ contain the new control command
```

4 Implementation and Results

The approach has been tested with different robots in our institute's indoor environment. Speeds up to 1 m/s were achieved even in peopled environments. The local obstacle avoidance module is only one part of an overall architecture. Due to the power of the local obstacle avoidance a grid based wave front expansion algorithm is sufficient to generate intermediate way points to provide global goal information [3]. Even if the planner generates the next way point in such a way that it can be reached only by a sudden sharp turn, the considered dynamic constraints ensure that only safe paths are chosen.

4.1 The Robot Smart^{Ulm}

Smart^{Ulm} is a synchrodrive RWI B21 robot (fig.5). Since no negative translational speed is considered, the number of curvature indices is $n_c = 2 \cdot n_v + n_w - 2 + 2$. Currently the following settings are used:

$v_{min,max}$, resolution	$0 - 900\frac{mm}{s}$, $10\frac{mm}{s}$
$w_{min,max}$, resolution	$\pm 70\frac{deg}{s}$, $1\frac{deg}{s}$
a_v, a_w	$300\frac{mm}{s^2}$, $50\frac{deg}{s^2}$
cartesian space (x, y)	$\pm 3000mm$
resolution	$100mm$
cycle time	$125ms$
curvatures	323
cartesian space	3721 cells
$T_{distance}$	1201883 cells
$T_{command}$	12831 cells
number of laser scan points	361 each cycle

Figure 5: Smart^{Ulm}

Since each distance value in $T_{distance}$ is represented by two bytes, the distance lookup table size is around 2.5 MB which is acceptable with the present computing power on board of mobile systems. Figure 6 shows a typical part of a run with 800 mm/s ($\alpha_{1,2,3} = 1.0, 1.0, 2.0$).

4.2 Forklift with Tricycle Kinematic

The forklift (fig.7) was built using a three-wheeled industrial vehicle (AGV) as a platform. As it has a tricycle kinematic, actuating variables are steering angle and translational velocity. Depending on the loaded items, the shape of the forklift changes. Thus each time an item is picked up or set down, the distance

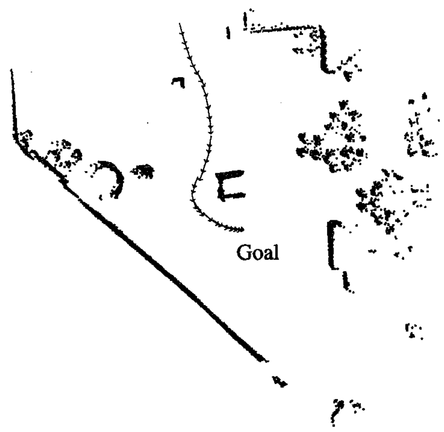

Figure 6: Part of run in cafeteria with 800 mm/s

lookup table has to be recalculated. With this algorithm the forklift correctly approaches palettes to pick them up. This is not possible with an algorithm approximating the shape with a circle, because collision detection prevents insertion of the fork below the palette (fig.8).

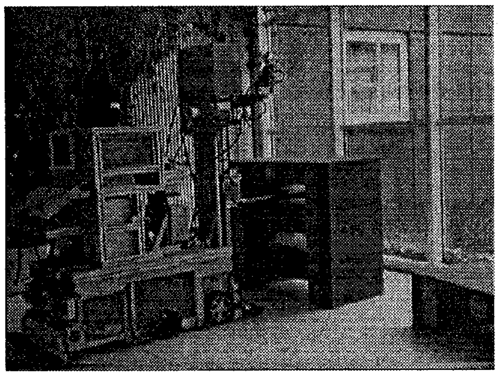

Figure 7: The forklift.

On this vehicle a laser scanner providing 8 scans/s (with 441 scanpoints each) is used. For this application, we use a different heading term within the objective function. Since the forklift has a tricycle kinematic and it has to reach the goal position G with a certain heading, the desired curvature is given by the circle C tangential in G through R as shown in figure 9. If the robot is within a certain distance to the goal, the prefered steering angle α is defined by C and an additional term proportional to the heading error φ. The heading term of the objective function prefers steering angles close to α. Furthermore, the translational speed is varied according to the goal dis-

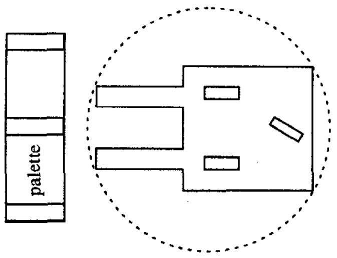

Figure 8: With a circular robot shape the collision with the palette's feed would prevent insertion of the fork.

tance. Since α is determined within each cycle, the forklift smoothly approaches G with the correct heading without colliding with obstacles.

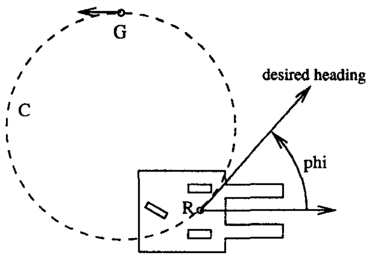

Figure 9: Curvature for approaching goal with given heading.

4.3 Wheelchair

Another possible application is to use the algorithm as an assisting system on a wheelchair. The person sitting in the wheelchair only selects the desired driving direction. The finally executed commands are calculated by the obstacle avoidance. By different objective functions the wheelchair characteristic can be adapted to different needs. This allows to operate safely even for an unexperienced user.

5 Conclusion

In this paper a local collision avoidance method has been described which allows the use of an any-shaped robot within a varying and cluttered environment. The approach takes into account the dynamic and kinematic constraints of the robot. By the selective use of precalculated lookup tables the computational load of the online part of the algorithm can be reduced significantly. This is the key to realtime performance. In contrast to other approaches no problematic simplifying assumptions about the shape of the robot or its kinematic and dynamic constraints are made. Since the online lookup phase is very fast, even raw laserscans can be considered without further reduction of the number of distance values. An objective function trades off different aims like goaldirectedness, speed and safety. By variations of the weights of the terms of the objective function, different collision avoidance strategies can be realized. The use of lookup tables makes it possible to integrate advantages of different similar approaches avoiding individually made simplifying assumptions.

The used objective function works very well and the appropriate weighting factors can be easily determined with a few trial runs. Nevertheless, it is worth examining the power of more sophisticated objective functions. Different objective functions and weighting parameters allow to implement different strategies like exact goal approaching or driving with high speed. Another topic of future research is the full exploration of the effects of different grid sizes.

Acknowledgments

This work is part of project C3 (http://www.uni-ulm.de/SFB527/Projects/c3.html) of the SFB527 sponsored by the Deutsche Forschungsgemeinschaft.

References

[1] W. Feiten, R. Bauer, and G. Lawitzky. Robust obstacle avoidance in unknown and cramped environments. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 2412–2417, San Diego, May 1994.

[2] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1), 1997.

[3] M. Knick and C. Schlegel. AMOS: Active perception of an autonomous system. In *Proc. IROS International Conference on Intelligent Robots and Systems*, pages 281–289, Munich, Sept. 1994.

[4] J. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[5] Reid Simmons. The curvature-velocity method for local obstacle avoidance. In *Proc. IEEE Int. Conference on Robotics and Automation*, pages 3375–3382, Minneapolis, April 1996.