Speed-Dependent Obstacle Avoidance by Dynamic Active Regions

Hans-Ulrich Kobialka and Vlatko Becanovic

Fraunhofer Institute for Autonomous Intelligent Systems, Sankt Augustin, Germany

Abstract. An obstacle avoidance approach is introduced that has dynamic active regions. The dynamic regions are adapted to the current speed of the robot and there are different active regions used, one for speed reduction and one of turning away from obstacles. The overall strategy of this approach is that the robot can drive with high speed which will be reduced in front of an obstacle in order to do a sharper turn.

1 Introduction

Successful obstacle avoidance has to consider the abilities of the vehicle to slow down and turn. When driving at high speed, a robot needs more time to stop and making sharp curves becomes impossible. So if the robot slows down, the amount of possible trajectories increases. In presence of an obstacle, it has to be decided whether the robot should slow down and turn more sharply closer to the obstacle, or keep its speed but deviate more from the desired path.

The potential field method [8, 6] is commonly used for autonomous mobile robots in recent years. Basically it builds up an artificial potential field consisting of repulsive forces which pull the robot away from obstacles, and an attractive force directed towards a target direction. If the position of obstacles and target(s) is well-known and stable, the potential field can be used for planning a trajectory towards the target, e.g. [15]. In case of unknown environments where the robot can sense obstacles only in its near vicinity, the potential field method can only be used for local obstacle avoidance, i.e. compute the next direction to go for. The same holds true for dynamic environments, like RoboCup [1], where obstacles (e.g. robots) change their positions continuously.

The potential field method regards the robot to be holonomic, i.e. it can drive at any moment in any direction. Obviously this is not true for robots depending on their speed and mass. Non-holonomic, e.g. differentially steered, robots have to turn first before they can move in the desired direction. Obstacle avoidance methods, like VFH+ [14] and the Curvature Velocity method [11], consider this by cost functions. Using these cost functions the costs for each direction is computed. The dynamics and kinematics of a robot are considered as only feasible directions are evaluated using the cost function. A differentially steered robot can drive only small curvatures at high speed, so only these directions are considered. Finally, the direction with minimal cost is selected.

In VFH+, cost increases if a direction leads towards obstacles, deviates from the goal direction, or deviates from the current steering direction. The robot only slows down if every direction is blocked. So if a robot should drive along a corridor and turn into a narrow door, it may miss it because the speed isn't reduced. The same problem has been experienced for the Curvature Velocity method [9], and in [12] for the Dynamic Window approach [5].

The Lane Curvature method [9] chooses a free lane by minimizing an objective function which has to implement the trade-off between avoiding obstacles, heading for a target position, and maximizing speed. Among these three aspects, speed has the lowest priority, so the direction with minimum costs may demand to slow down the robot. This is similar to the effects of our approach.

In our approach, collisions are avoided by reducing the linear speed which again enables the robot to turn away from obstacles and towards the target direction. Both, speed reduction and turning, influence each other:

- speed reduction enables the robot to turn sharper, and
- turning to free space enables the robot to increase its speed again.

Speed dependency is mainly introduced by enlarging/shrinking the active region. The active region is an area that moves with the robot, usually a circle around the robot, see e.g. [6, 13]. In [12], a channel around a path (produced by a A* planner) is used. Only obstacles within the active region are considered for obstacle avoidance. An obstacle outside the active region is ignored as it is not regarded as a possible hazard.

Speed reduction and turning use different active regions. This is another outstanding characteristic of our approach. This brings the advantage that they can be tuned independently.

Speed reduction and turning are implemented as distinct behavior modules. In compliance with the Dual Dynamics architecture [7, 10, 3, 2], each module has a so called activation dynamics and a target dynamics. The target dynamics computes the motor commands by which the module wants to pursue its goals. The activation dynamics computes the activation value of this behavior, ranging in from 0 to 1. The activation value is used as a weight in order to superimpose concurrently active behavior modules. An activation value of 0 means that the module is not active, i.e. it should not influence the motors in any way. The activation of an obstacle avoidance module expresses the danger of having a collision; when approaching an obstacle it rises continuously from 0 to 1 and, after turning away, back to 0 again.

We will describe the speed reduction module *SlowDown* (section2) and the module *TurnAway* which makes the robot turning away from an obstacle (section 3). Section 4 tells how the outcome of these two modules is superimposed with the motor values of the goal-oriented behaviors. Our experience with speed-dependant obstacle avoidance is sketched in section 5.

2 Speed Reduction

2.1 Active Region

The active region of the module *SlowDown* is in front of the robot regarding its current direction of movement. In case of a differentially steered robot this active region is either at its front, or, if it moves backwards, at its back. The active region is shaped like a rectangle because obstacles within this region would collide with the robot as it moves forward. The length and the width of the active region is computed as follows:

$$\begin{aligned} length_{AR} &= speed_{Robot} \cdot \Delta t + mDist2Obstacle \\ width_{AR} &= width_{Robot} + mDist2Obstacle \end{aligned} \tag{1}$$

where $width_{Robot}$ is the robot's width, $speed_{Robot}$ is the robot's current speed, $\triangle t$ is some time span (e.g. 1 sec) and *mDist2Obstacle* is the minimum distance within which the robot should react on obstacles.

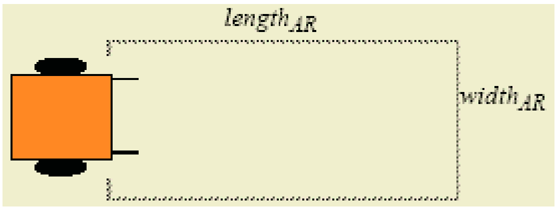

Fig. 1. The active region of *SlowDown*

2.2 Collision Danger

While the robot senses its environment (e.g. using ultrasonic distance measurement), it may detect a number of obstacle points. The possible danger of colliding with an obstacle point which is measured inside the active region, decreases exponentially depending on its distance $Dist_i$ to the robot.

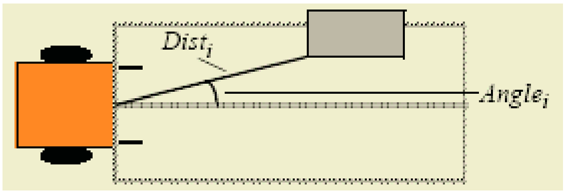

Fig. 2. An obstacle point is viewed with $Angle_i$ and $Dist_i$

$$F_{repulsive_i} = \begin{cases} e^{(-K_1 \cdot Dist_i)} \cdot weight_i & \text{if } Obstacle_i \text{ is in active region} \\ 0 & \text{else} \end{cases} \tag{2}$$

The weight of the obstacles is computed during several stages. First, it is proportional to the cosine of the obstacle's angle $Angle_i$.

$$weight_i^{**} = K_P \cdot \cos(Angle_i) + w_{\min} \tag{3}$$

where $w_{\min}$ denotes another safety margin.

If the robot drives at high speeds, the obstacles at the side (having low values for $\cos(Angle_i)$) should gain more importance. This is achieved by the following step

$$weight_i^* = \begin{cases} \max(weight_{critical}, weight_i^{**}) & \text{if } Dist_i \leq RobotSpeed K \\ weight_i^{**} & \text{else} \end{cases} \tag{4}$$

Finally, the weight is multiplied with a constant depending on how the obstacle has been classified. For instance, in the RoboCup domain there used to be a wall which has to be touched in order to get a ball away from it. So this kind of obstacle should not repel the robot as much as, for instance, other robots. If the ball lies between the robot and its own goal, it has to be avoided in order not to shoot self goals; so in this situation K_{ball} is very high, otherwise zero.

$$weight_i = \begin{cases} K_{ball} \cdot weight_i^* & \text{if } Obstacle_i \text{ belongs to the ball} \\ K_{wall} \cdot weight_i^* & \text{if } Obstacle_i \text{ belongs to a wall} \\ K_{robot} \cdot weight_i^* & \text{else} \end{cases} \tag{5}$$

2.3 Activation Dynamics

The activation dynamics of *SlowDown* computes its activation value which expresses the need for speed reduction for the current situation. Basically, it is the maximum of all repulsive "forces" clipped to 1.

$$a_{target} = \min(1, \max_i(F_{repulsive_i})) \tag{6}$$

This target value is low-passed by an ordinary differential equation

$$\dot{a}_{SlowDown} = T \cdot (a_{target} - a_{SlowDown}) \tag{7}$$

$$T = \begin{cases} T_{reactNow} & \text{if } \quad RobotSpeed \geq 70 \\ T_{reactFast} & \text{if } 70 > RobotSpeed \geq 40 \\ T_{reactModerate} & \text{else} \end{cases}$$

where the time constant T (which expresses how fast $a_{Slowdown}$ should follow a_{target}) depends on the current speed of the robot.

2.4 Target Dynamics

The target dynamics of *SlowDown* computes the maximum speed which is admissible for a certain situation. Basically it reduces the maximum Speed $Speed_{max}$ depending on $a_{Slowdown}$; especially if $a_{Slowdown}$ is zero, $Velocity_{SlowDown}$ equals $Speed_{max}$. This is done by a weighted sum where $w_{SlowDown} >> w_{Goal}$ ensures speed reduction in case of high values of $a_{Slowdown}$. In situations when turning is dangerous, i.e. will lead to collisions, the robot has to drive backwards for a short while.

$$Velocity_{SlowDown} = \begin{cases} \dfrac{w_{Goal} \cdot Speed_{max} + w_{SlowDown} \cdot a_{SlowDown} \cdot 0}{w_{Goal} + w_{SlowDown} \cdot a_{SlowDown}} & \text{if } a_{SlowDown} < 1 \\ 0 & \text{if } a_{SlowDown} = 1 \\ -b & \text{if turning is dangerous} \end{cases} \quad (8)$$

3 Turning away from an Obstacle

3.1 Active Region

The active region of the module *TurnAway* is oriented towards the current target direction.This region differs from the active region of *SlowDown*; especially it is independent of the current orientation of the robot.

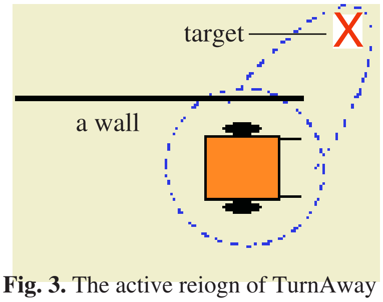

Fig. 3. The active reiogn of TurnAway

Also the shape of the active region differs from the one of *SlowDown*: first, there is a circle around the robot; if there are no obstacle point within, the robot can turn, otherwise not. Second, there is an active region oriented towards the target. It is shaped like a parabola of a width (near the robot) and a length defined like

$$length_{AR} = \min(speed_{Robot} \cdot \Delta t, dist2destination) \quad (9)$$
$$width_{AR} = width_{Robot} + mDist2Obstacle$$

Similar to the active region of *SlowDown*, the length of the parabolic active region depends on the robot's speed. In addition, the length is clipped to *dist2destination*, the distance between the robot and its target. This is because the robot should not care about obstacles being behind its target. Otherwise such obstacles would cause the robot to turn away before reaching its target. For instance, in RoboCup, the target may be the ball and there may be other robots behind the ball. These should not be avoided if a robot approaches the ball.

For the same reason the parabolic shape was chosen for the active region: the robot should not turn away (but only slow down), if there are other robots in the vicinity of the ball. This example illustrates that the active region should be shaped in an application specific way. For other applications than RoboCup, other shapes could be advantageous.

3.2 Collision Danger and Activation Dynamics

The collision danger and activation dynamics of *TurnAway* are computed in the same way as for *SlowDown* (see equations (2) to (7)). However it should be noted that

- a different active region is used, and
- the used constants are different.

This way, the modules *TurnAway* and *SlowDown* can be tuned independently (e.g. scope, reaction time, the strength of reaction) which helped us a lot while tuning the system.

3.3 Target Dynamics

The target dynamics of TurnAway computes the direction which is recommended to the robot by the obstacle avoidance module. Here we use a potential field approach: Each obstacle point has, beside $Dist_i$ and $Angle_i$, a $direction_i$ which is a unit vector pointing from the obstacle to the front of the robot.

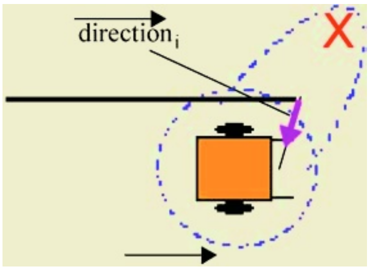

Fig. 4. Direction$_i$ the repulsive unit vector

When computing the weighted vector sum, using $Frepulsive_i$ as weights, we get again a unit vector $direction_{TurnAway}$, the direction where *TurnAway* recommends to turn.

$$\overrightarrow{direction}_{TurnAway} = \frac{\sum F_{repulsive_i} \cdot \overrightarrow{direction}_i}{\sum F_{repulsive_i}} \tag{10}$$

4 Superposition of Goal-Oriented Behaviors and Obstacle Avoidance

After the obstacle avoidance modules have computed their output (activation values of *SlowDown* and *TurnAway* together with the recommended direction and the maximal admissible linear speed), we have to tell how the final motor commands result from this.

In the computation of the maximal admissible linear speed, the activation value of *SlowDown* is already used. So, the only thing left to be done is to clip the speed of the goal-oriented behavior(s) by this speed.

$$Velocity = \min(Velocity_{Goal}, Velocity_{SlowDown}) \tag{11}$$

Compromising between $direction_{TurnAway}$ and the direction wanted by the goal-oriented behaviors ($direction_{Goal}$) is done again by a weighted sum of vectors. Giving $direction_{Goal}$ the weight 1, $Weight_{TurnAway}$ has to be >> 1 in order to let $direction_{TurnAway}$ dominate if $a_{TurnAway}$ increases. This gives *TurnAway* the power to suppress $d_{desired}$ if obstacles are very close (i.e. if $a_{TurnAway}$ gets close to 1).

$$\overrightarrow{direction} = \frac{1 \cdot \overrightarrow{direction}_{Goal} + Weight_{TurnAway} \cdot a_{TurnAway} \cdot \overrightarrow{direction}_{TurnAway}}{1 + Weight_{TurnAway} \cdot a_{TurnAway}} \tag{12}$$

5 Results

The dynamic active region approach has been used since April 2001 by our RoboCup robots participating in the middle-size league. Each robot is equipped with 5 infrared distance sensors, two of them can sense obstacles up to 80 cm in front of the robot while the other three have a range of 50 cm. Obstacle points are remembered up to 1.5 seconds in an occupacy grid; as the robot turns, it gets a scan of obstacle points in its vicinity. Thus, the robot is quite short-sighted and the sensory input is sometimes incomplete, i.e. obstacles are not sensed. Even with this limited sensory input, our approach worked even in very crowded and dynamic situations. We limited the maximum speed of our robots to 160 cm per second because of the limited scope of our sensors and due to the fact that other robots also move fast. Under other conditions we expect our approach to work at higher speeds.

Figure 5 shows a scenario where a robot wants to drive behind the ball but finds its way obstructed by obstacles. The resulting trajectory is shown at the right side of figure 5: the points are drawn in equidistant time steps; it can be seen that the robot slows down in the vicinity of obstacles.

6 Conclusions

A novel obstacle avoidance approach is introduced that has dynamic active regions. The dynamic regions are adapted to the current speed of the robot. There are different active regions used, one for speed reduction and one of turning away from obstacles. The overall strategy of this approach is that the robot can drive with high speed which will be reduced in front of an obstacle in order to do a sharper turn.

The obstacle avoidance module is placed in the system architecture in a way that it can not be overruled by the behavior system. Also, the behavior system may choose among several possible paths by using the obstacle avoidance module as an oracle. This creates a highly flexible system that has shown good results in experiments, even with low dimensional sensor data of limited range.

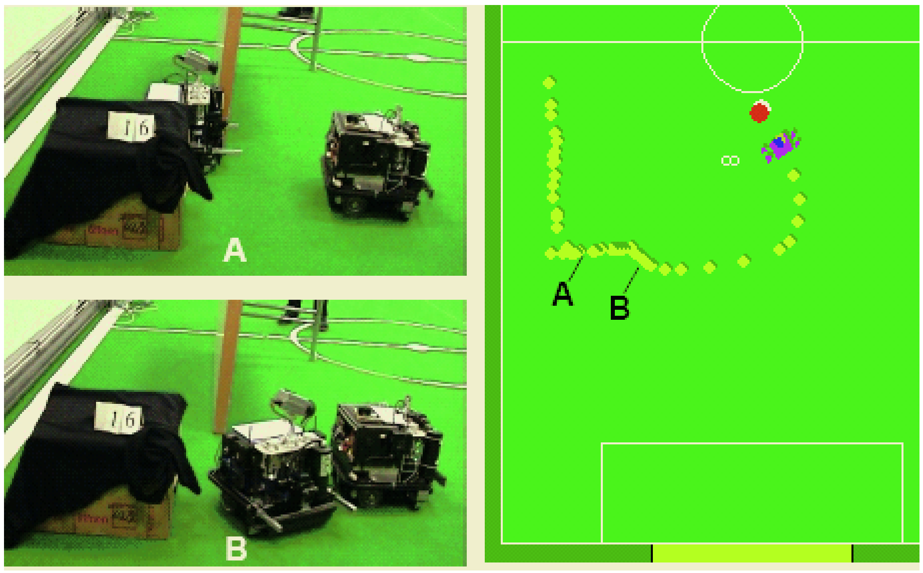

Fig. 5. A robot going for a ball while avoiding obstacles. The two snapshots at the left show the robot at positions (A and B) marked in the robot's trajectory shown at the right hand side

References

1. http://www.robocup.org
2. S. Behnke, R. Rojas, G. Wagner, 'A Hierarchy of Reactive Behaviors handles Complexity', *Workshop 'Balancing Reactivity and Social Deliberation in Multi-Agent Systems'* at the 14th European Conference on Artificial Intelligence (ECAI), 2000.
3. A. Bredenfeld, H.-U. Kobialka, 'Team Cooperation Using Dual Dynamics' *Balancing reactivity and social deliberation in multi-agent systems,* Lecture notes in computer science 2103, pp. 111 - 124, Springer, 2001.
4. A. Bredenfeld, G. Indiveri, 'Robot Behavior Engineering using DD-Designer', *Proc IEEE International Conference on Robotics and Automation (ICRA 2001),* pp. 205-210, 2001.
5. D. Fox ,W. Burgard, S. Thurn, 'The Dynamic Window Approach to Collision Avoidance' *IEEE Transactions on Robotics and Automation*, 4:1, 1997.
6. S.S. Ge, Y.J. Cui, 'Dynamics Motion Planing for Mobile Robots Using Potential Field Method', *Autonomous Robots* 13, 207-222, 2002.
7. H. Jaeger, T. Christaller, 'Dual Dynamics: Designing behavior systems for autonomous robots', *Artificial Life and Robotics*, 2:108-112, 1998.
8. O. Khatib, 'Real-time obstacle avoidance for manipulators and mobile robots', *The International Journal of Robotics Research* 5(1):90-98, 1986.
9. N.Y. Ko, R. Simmons, 'The Lane-Curvature Method for Local Obstacle Avoidance', *Proc. Intelligent Robots and Systems (IROS)*, 1998.
10. H.-U. Kobialka, H. Jaeger, 'Experiences Using the Dynamical System Paradigm for Programming RoboCup Robots', Proc. *AMiRE 2003* (2nd International Symposium on Autonomous Minirobots for Research and Edutainment), pp 193-202, 2003.
11. R. Simmons, 'The Curvature-Velocity Method for Local Obstacle Avoidance', *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA 1996)*, pp. 3375-3382, 1996.

12. C. Stachniss, W. Burgard, 'An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments', *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, 2002.
13. I. Ulrich, J. Borenstein, 'VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots', *IEEE Int. Conf. Robotics and Automation (ICRA 1998)*, pp. 1572-1577, 1998.
14. I. Ulrich, J. Borenstein, 'VFH*: Local Obstacle Avoidance with Look-Ahead Verification', *IEEE Int. Conf. Robotics and Automation (ICRA 2000)*, pp. 2505-2511, 2000.
15. T. Weigel, J.-S. Gutmann, M. Dietl, A. Kleiner, B. Nebel, 'CS-Freiburg: Coordinating Robots for Successful Soccer Playing', *IEEE Transactions on Robotics and Automation* 18(5):685-699, October 2002.