

Dynamic Window Approach for Omnidirectional Robots with Polygonal Shape

Andreas Lawitzky, Daniel Althoff, Dirk Wollherr and Martin Buss

Abstract—This work presents an extension of the Dynamic Window Approach. The reactive collision avoidance algorithm is generalized to the case of omni-directional kinematics for any polygonal shaped, mobile robot. This paper includes a superior implementation to former realizations and has already been successfully tested with an omni-directional robot. The implementation is efficient and produces collision-free trajectories even in narrow and crowded environments.

I. INTRODUCTION

Mobile robots are about to find their way into everyday life e.g. as shopping assistants, in nursing or as household helpers. In their field of application they have to solve different high-level duties, but a necessity for these tasks is a collision-free movement.

One representative is the potential fields method [1], inspired by the electrical attraction force. It uses an analogy in which the robot is a particle that moves in the configuration space under the influence of a force field. But it is far from creating optimal trajectories around obstacles for all information is compressed into one single resultant force.

A subsequent approach which faced these shortcomings is the Vector Field Histogram [2], which determines the velocity of the next time step in two steps. First it computes a set of candidate motion directions and afterwards selects one of them.

Further algorithms are described in [3], [4]. Nevertheless, all these algorithms for reactive collision avoidance suffer from problems of high complexity, unreliable collision-avoidance or poor trajectories, because these approaches do not take the robot's shape, its kinematics or its dynamics explicitly into account.

The Dynamic Window Approach (DWA) was proposed by Fox et al. in [5]. In various tests, the DWA figured out to be reliable even in highly crowded areas. The advantage of the DWA over other low-level collision avoidance algorithms is the low complexity even at high speeds. This is done by explicitly respecting the kinematics of the robot. Therefore the DWA became an often used algorithm for collision avoidance. In a nutshell, it plans the velocity for the next time step in the velocity space, while using only local information of the environment. All variables are strictly calculated in the local robot frame. The algorithm discretizes the search space i.e. the whole velocity sphere, where the robot is able to drive

respecting the robot's kinematic constraints. The dynamic constraints are explicitly taken into account and represented by the dynamic window. The velocities the robot can reach within a single time step T are only a small subset of the whole velocity sphere, called the dynamic window because in its simplest implementation it is a rectangle.

A. PROBLEM FORMULATION

While former implementations of the DWA were built for robots with either circular shape or differential drive, this work presents a superior implementation. It features collision avoidance for robots with omni-directional drive, too. Also, the extension of the DWA can handle polygonal shapes of robots while both convex and concave forms are possible.

II. METHOD

The calculation of the time-to-collision is the key challenge for the DWA implementation. For simplicity the calculation is reduced to the basic equations.

In the case of angular velocity $\omega = 0$ the trajectory, the robot describes is a straight line in a global frame. Hence, the obstacle's trajectory in the robot's frame is also a straight line $P = P_0 - vt$. The robot's polygonal hull can be separated into lines, regarding $A-B$, the time-to-impact of the obstacle P with that line is calculated with $P_0 - vt = A + \mu(B - A)$. Only if $\mu \in [0, 1]$ and $t > 0$ a collision with the regarded line will occur. The resulting collision time for the whole robot is the minimum of the values of all lines.

In case $\omega \neq 0$, the trajectory of the robot is a circle in the global frame. To describe the trajectory of an obstacle in the robot's frame, a rotation \mathcal{R} of the coordinate system is useful with angle $\alpha = -\text{atan2}(v_y, v_x)$. With this, the coordinate system can be rotated to have the combined speed vector of v_x and v_y aligned with the new x' -axis. Any vector a is rotated like $\mathcal{R} : a' = R \cdot a$. This rotation allows to treat the omni-directional robot as one with differential-drive.

In this coordinate system the robot's trajectory R is described by a circle with

$$\begin{aligned} R'(t) &= \frac{v_{x'}}{\omega} (\sin(\omega t), (1 - \cos(\omega t)))^T \\ r^2 &= (x'_R(t) - x'_M)^2 + (y'_R(t) - y'_M)^2 \\ r &= \pm \sqrt{\frac{v_{x'}^2}{\omega^2}} = \pm \frac{v_{x'}}{\omega} \end{aligned}$$

The radius of the circle is $r = \frac{v_{x'}}{\omega}$ with the center M at $(0, \frac{v_{x'}}{\omega})^T$. Hence, all points of the robot describe a circle around M with angular velocity ω

The authors are with the Institute of Automatic Control Engineering (LSR). Dirk Wollherr is also with Institute for Advanced Study (IAS). Both institutes are with the Technische Universität München, D-80290 München, Germany. alawitzky@lsr.ei.tum.de, {da, dw, mb}@tum.de

The distance $M—P$ is $r_P = \sqrt{x_P'^2 + (y_P' - \frac{v_x'}{\omega})^2}$. In robot's frame the obstacle describes a circle around M , radius r_P , and angular velocity $-\omega$.

1) *Collision Requirement*: One requirement that a collision could occur can be formulated at this point as $y_M' - r_{out} \leq r_P \leq y_M' + r_{out}$, with r_{out} is the circumcircle's radius of the robot's hull (see Fig. 1).

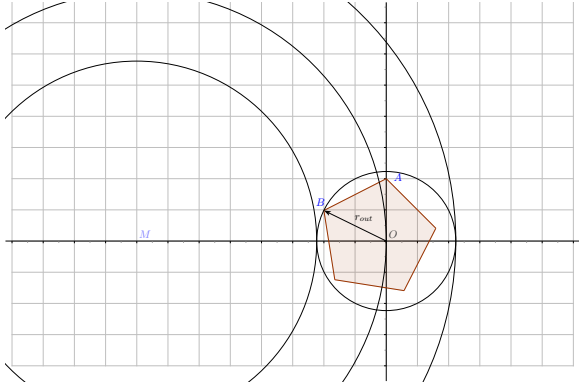


Fig. 1. Calculating the collision time of obstacles and polygonal shapes.

2) *Time to collision*: For a single line $A—B$ of the robot, K is the point of the collision, which can be calculated using $a = B - A$. b is defined with $\langle a, b \rangle = 0$ normal to a . Now the system of equations $A + \lambda a^0 = M + \mu b^0$ can be solved. Collisions only occur for $|\mu| < r_P$. Define k with $r_P^2 = \mu^2 + k^2$ then only solutions where $0 < k < \text{length}(AB)$ represent collisions at $K_i = M + \mu \cdot \vec{b}^0 + k_i \cdot \vec{a}^0$, $i = 1, 2$.

W.l.o.g. $\omega > 0$, let

$$\begin{aligned} \varphi_{K,i} &= \text{atan2}(y_{K,i}' - y_M', x_{K,i}') \\ \varphi_{Obstacle} &= \text{atan2}(y_{Obstacle}' - y_M', x_{Obstacle}') \\ \Delta\varphi_i &= \varphi_{Obstacle} - \varphi_{K,i} \end{aligned}$$

With normalized $\Delta\varphi$, the time-to-collision is $t = \frac{\Delta\varphi}{\omega}$.

Using the algorithm described above a collision-free motion can be assured. To reach a goal autonomously, one velocity is chosen among all possible ones using heuristics i.e. superposition of utility functions.

III. EXPERIMENTS

To test the algorithm, the omni-directional platform of the research robot of the CoTeSys (Cognition for Technical Systems) project “Multi-Joint Action of Cognitive Systems” (MuJoA) was used (see Fig.2). The collision times are calculated on-line, so the algorithm can use adjustable robot hulls which can be configured and changed according to the current configuration of the robot and his arms. The advantage of this implementation is the higher flexibility, because the robot hull can be adjusted during operation and still assures collision-free driving.

The MuJoA research robot has been used for intensive testing of the implementation with different scenarios, from simple straight line movement to narrow corridors and also crowded rooms. The implementation figured out to offer reliable collision avoidance in all situations for hours of

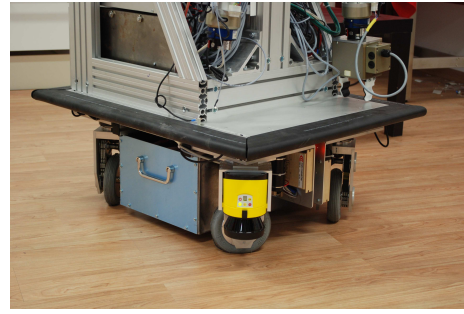


Fig. 2. The rectangular shaped platform of the MuJoA research robot.

testing. The code of the library has been tested to run on different standard Linux PCs. On all computers the workload produced by the collision avoidance process during driving was below 5% at an update frequency of 10 Hz.

IV. DISCUSSION

This work presented a superior implementation of the Dynamic Window Approach, suitable for omni-directional driven robots with any polygonal shape. Besides this extension for this class of robots, the algorithm is suitable for calculating the collision times on-line. This allows high flexibility e.g. to change the robot's arm configuration according to the current needs while driving. But an off-line calculation of the collision times is still possible to reduce the work load further. The practicability of this new generation of Dynamic Window Approach implementations was proofed on the MuJoA research robot. Nevertheless, the Dynamic Window Approach suffers from shortcomings like discretization and local minima.

V. ACKNOWLEDGMENTS

The authors gratefully acknowledge partial financial support of this work by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the excellence initiative research cluster *Cognition for Technical Systems – CoTeSys* (www.cotesys.org), and the EU-STREP project *Interactive Urban Robot (IURO)*, www.iuro-project.eu.

REFERENCES

- [1] O. Khatib, “Real-time Obstacle Avoidance for Manipulators and Mobile Robots,” *1985 IEEE International Conference on Robotics and Automation (IROS 1985), Proceedings*, vol. 2, pp. 500 – 505, March 1985.
- [2] J. Borenstein and Y. Koren, “The vector field histogram — fast obstacle avoidance for mobile robots,” *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278 – 288, June 1991.
- [3] J. Minguez, “The Obstacle-Restriction Method (ORM) for Robot Obstacle Avoidance in Difficult Environments,” in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2005.
- [4] O. Brock and O. Khatib, “High-Speed Navigation Using the Global Dynamic Window Approach,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, 1999, pp. 341 – 346, 10 – 15 May 1999.
- [5] D. Fox, W. Burgard, and S. Thrun, “The Dynamic Window Approach to Collision Avoidance,” *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.