in: Proc. 52nd Int. Scientific Colloquium (IWK), 2007, Technische Universität Ilmenau, Germany, Vol. I, pp. 425-430, Universitätsverlag 2007

C. Schröter, M. Höchemer, H.-M. Gross

A Particle Filter for the Dynamic Window Approach to Mobile Robot Control

Abstract

In this paper we present an anticipative local navigation algorithm for an autonomous mobile robot. The purpose of local navigation is to move the robot according to a specified goal, like a planned path to a target, and avoiding collisions with obstacles during operation. The robot is perceiving its immediate surroundings by laser and sonar range scanners and by a stereo camera. All the sensor information is represented in a local map. In order to chose the best action, a number of possible trajectories are then evaluated. The trajectories are modelled as clothoid curves, a parametric curve which is well suited for moving vehicles. A fitness function that takes into account the likelihood of collisions, the compliance with the navigation goal and the speed that can be achieved selects the best trajectory, which is then translated into motion commands for the drive system.

1 Introduction

Autonomous navigation is one of the key features of a mobile robot. Typically, an autonomous system is possessing a map of the environment, which is either given by the designer or built from sensory data during operation. Furthermore, the robot has some means of determining and tracking its own position in the environment. In order to reach a certain position within that map a 2 step process is conducted: First, the robot has to plan a series of actions to move from its current position A to the desired target position B. Then, it has to translate the actions into concrete motion commands to be executed by the drive system. In most cases it is not possible to plan the entire sequence of motion steps that would move the robot from A to B because of the occurrence of positional errors during execution as well as discrepancies between the map and the actual environment, caused e.g. by dynamic obstacles such as people appearing in the are or unpredictable motion of obstacles. Therefore, typically the planner just generates a shortest path from A to B which is a sequence of positions, while a controller/local navigator is trying to sequentially generate motion commands that make the robot follow that path, with respect to the current position and the updated state of the environment as perceived by the robot's external sensors. In this paper we will consider a new implementation of the local navigator only. The global map, the planning of a shortest path from the robot's position to its target as well as the self-localization of the robot within the environment are taken for granted and not discussed here. The local navigation approach we propose is an anticipatory behaviour because it not only tries to avoid obstacle collisions but does so by evaluating the results of a number of possible actions using a limited foresight into the future. To this purpose, actions are coded as parametrical trajectory curves and managed by a scheme similar to the way particle filters are used for state estimation problems.

2 Local Navigation - Related Work

Existing obstacle avoidance aproaches can be roughly divided into two classes: reactive and anticipatory. A very simple reactive method is the potential field approach, which works by assigning virtual repelling forces to obstacles close to the robot, and attracting forces to navigation goals. However, this method often fails to pass narrow passages like doors. An improvement is the Vector Field Histogram proposed by Borenstein et.al. and its various enhancements [1], [2]. In this group of algorithms, the robot explicitely distinguishes between free and blocked directions and chooses a free direction that is closest to the navigation goal. In contrast to these reactive methods, anticipative approaches explicitely evaluate the consequences of certain actions and try to choose the one yielding the highest return or lowest cost. In the Dynamic Window Approach [3], a number of circular trajectories are tested for the distance they keep to the obstacles around the robot. An enhancement to this is the Global Dynamic Window Approach, which additionally incorporates global navigation goals in the cost function. Our approach here basically is a modification of the Global Dynamic Window Approach, where we use clothoid instead of circular trajectories. Furthermore, instead of re-generating and evaluating all the possible trajectories from scratch in each time step, in analogy to a particle filter, the hypotheses are sampled from the best trajectory of the last time step, imposing an implicit smoothness constraint.

3 Clothoids

In order to choose the best action in the current situation, given the current state of the robot and the local environment as well as the overall target, we need to generate and evaluate a number of possible local motion trajectories. For the representation of the trajectories a form of parametrized curves called clothoids are used. The definition of a clothoid is a curve with linearily changing bending

$$c(l) = c_0 + c_1 * l \tag{1}$$

where the bending c is the inverse of the curve radius r. Clothoids are used in road construction because the linearily changing curvature in turn means a linear change of lateral force, avoiding a jump in the force imposed on vehicles following the road. Obviously, for the same reason they are a good model for robot motion.

Independently of the actual drive system, the robot's motion is usually seen as a superposition of translational and rotational velocity, denoted as v and w respectively. For constant translation velocity v and rotation velocity w, the robot will move on a circle with radius

$$r = \frac{v}{w} \tag{2}$$

Therefore, if the robot is currently moving at a velocity (v, w), the initial curvature c_0 is fixed

$$c_0 = \frac{w}{v} \tag{3}$$

Furthermore, there are limitations on the change rate of the curvature c_1, which reflect the physical properties of the robot such as mass and motor power. The actual sequence of positions described by the clothoid trajectories is then given by

$$x(l) = x_0 + \int_0^l cos(\phi(l))dl \tag{4}$$

$$y(l) = y_0 + \int_0^l sin(\phi(l))dl \tag{5}$$

$$\phi(l) = \phi_0 + \int_0^l c(l)dl \tag{6}$$

where (x_0, y_0, ϕ_0) is the current pose of the robot, containing position and orientation.

4 Trajectory Evaluation

When the robots navigator module receives a new target, it plans a path to the target position using the Dijkstra algorithm on the global map. During the path calculation, a potential field is generated which holds for each position of the global map the distance to the target, assuming a shortest path motion. This target distance will be used, together with other costs, in evaluating possible trajectories.

While the robot is moving, the external sensor measurements are continuously integrated into a local 2D map. This local map holds information about traversable and blocked space in a local vicinity of the robot. Due to the reliable perception distance of the sensors, the local map has a radius of about 3 meters around the current robot position.

The navigator is using the local map to generate motion commands for the drive system in intervals of 100 ms (at a maximum speed of 1m/s, this corresponds to a maximum driven distance of 0.1m). In order to determine the best local trajectory, a number of candidate clothoid trajectories are generated. Each clothoid is described by parameters c_0 and c_1. As exxplained in section 3, c_0 is equal for all possible clothoids, determined by the current translational and rotational robot speed, which is reported by the drive system. However, c_1 is sampled from a random distribution. When no best trajectory was selected in the previous loop run, e.g. at the very beginning of autonomous motion, the distribution is just a Gaussian with mean 0 and a fixed variance. When a previous best trajectory is already known, the new candidates are sampled with c_1^{old} as mean value. Together with $cost_{change}$ (see below), a behaviour of permanent alternating is suppressed in situations where 2 possible

trajectories are approxametely equally good (e.g. an obstacle in the center of a hallway that could be passed to the left or right), imposing an implicit smoothness constraint.

Each trajectory is assigned a cost that is a weighted sum of a number of costs, each one representing a certain objective:

- $cost_{closest\_obstacle}$: Along the trajectory, normal vectors are calculated in regular intervals. Along each normal line, the closest obstacle (blocked cell) is searched. If an obstacle is found, the cost is $(1.0 - d'_{traj}) * (1.0 - d'_{norm})$, where d_{traj} is the distance along the trajectory, d_{norm} is the distance from the trajectory along the normal line. d' denominates normalization by dividing by the maximum trajectory/normal line length respectively. The maximum cost for a single found obstacle determines $cost_{closest\_obstacle}$

- $cost_{sum\_obstacles}$: The summed obstacle cost sums the values over all normal lines. In contrast to $cost_{closest\_obstacle}$ it does not only consider the most extreme obstacle approach, but the overall distance keeping to obstacles along the entire trajectory.

- $cost_{bending}$: In order to enforce straight motion of the robot when possible, a high bending of the trajectory is punished with high cost. $cost_{bending}$ is directly proportional to the trajectory parameter c_1

- $cost_{target}$: While the robot must avoid collisions, it is still expected to follow a path that will take it to the target position. This is reflected by $cost_{target}$. The cost is proportional to the decrease of the target distance (when following the optimal path) between the current position and the trajectory end point..

- $cost_{change}$: This cost is proportional to the difference between the current trajectories parameter vector (c_0, c_1) and the previously selected trajectories parameter vector $(c_0, c_1)^{old}$ (see above).

Bending cost as well as change cost also depend on the current robot speed: at low speeds, a strong bending and a faster change of bending are less punishable than when driving at maximum speed. The overall cost is then given by

$$cost = \alpha * cost_{closest\_obstacle} + \beta * cost_{sum\_obstacles} + \gamma * cost_{bending} + \delta * cost_{target} + \epsilon * cost_{change}$$

with $\alpha = 10.0$, $\beta = 50.0$, $\gamma = 0.5$, $\delta = 8.0$ and $\epsilon = 0.2$ Obviously, when obstacles are near, they determine the cost mainly. Only in free space situations the bending and change cost have any significant influence.

Finally, the trajectory with the lowest overall cost determines the motion command. By choosing a certain c_1, the desired change of the curve bending and, with a fixed update cycle, a bending c to be reached till the next step is given. From eq. (3) follows that only the relation between v and w is determined by the bending c. Therefore, in order to find specific values for v and w, additional rules are needed. One possibility would be to always keep a constant translational velocity $v = v_0$. However, for safety reasons we prefer to slow down if we get closer to obstacles, therefore v depends on the trajectory obstacle cost too.

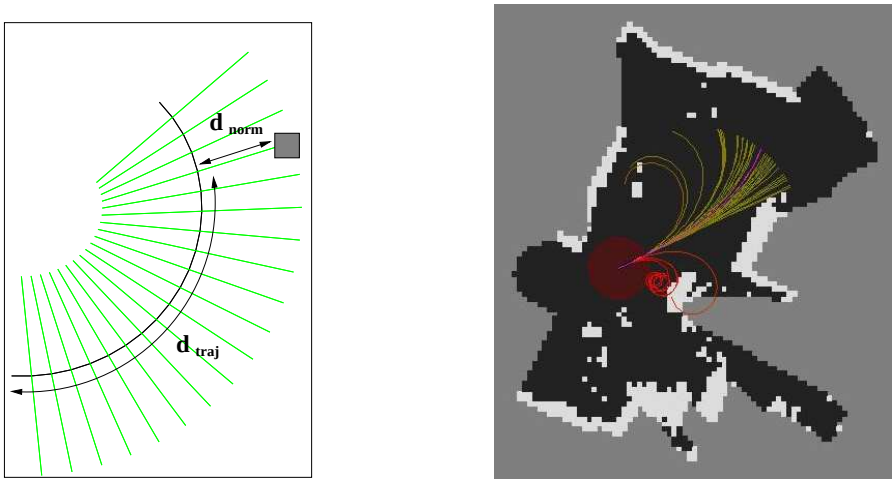

Figure 1: Left: The obstacle cost for a trajectory depends on the distance of the obstacle to the robot (along the trajectory) and the distance to the trajectory itself.
Right: The figure shows the robot and the local environment, which is perceived using a laser scanner and a stereo camera. White areas are obstacles, while black areas are free space. Grey indicates areas which have not been seen by the robot. A number of trajectories are shown, where the color shows the cost associated with each trajectory. Green colors mean low costs, while red colors show high costs. The preferred trajectory, which determines the motion command, is marked magenta.

5 Results

To compare the new navigation algorithm presented here to an implementation of the Vector Field Histogram (this is actually an enhanced version of VFH that has been our standard local navigation approach for years), we show results of a test run where the robot's task was to go down a hallway and turn into an adjacent room, crossing a very narrow door (only a few cm space to either side of the robot). In both cases the maximum robot speed was limited to 0.5 m/s. The plots show that the robot moves significantly faster and smoother using the new algorithm for local navigation (Fig. 2). While with VFH the robot took 36 seconds for the path (average velocity 0.25 m/s), it arrived 40% faster with the new algorithm.

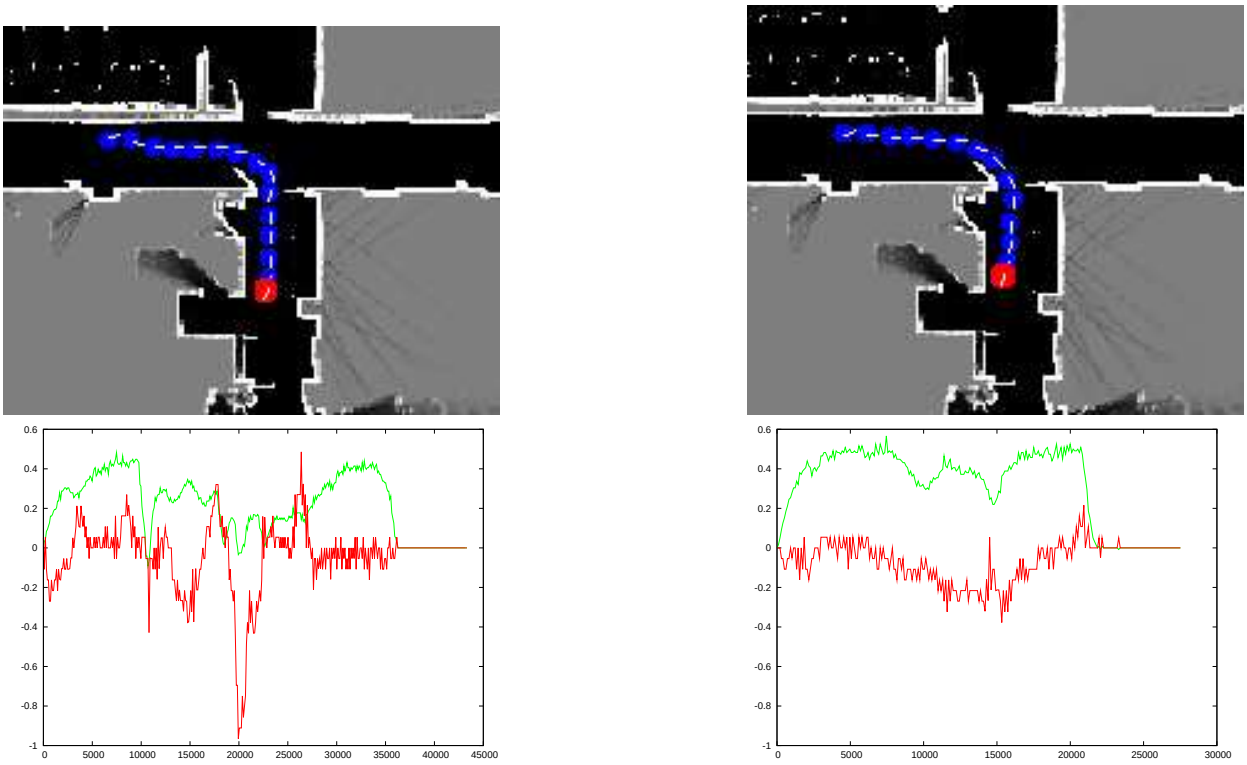

Figure 2: Test run results - left: results using an implementation of the Vector Field Histogram, right: using the Trajectory Particle Filter. The upper row shows the robot path for both algorithms respectively. The path is slightly smoother using the new algorithm. Results are more obvious from the velocity plots (2nd row). Here, translational (green) and rotational (red) velocity are drawn along the path. With the new algorithm, there are less changes in translational as well as rotational velocity, the behaviour gives a much smoother impression to an observer. The average translation velocity also is significantly higher, the robot reaches the target position about 40% faster.

References

[1] J. Borenstein and Y. Koran, "The Vector Field Histogram - fast obstacle avoidance for mobile robots", IEEE Journal of Robotics and Automation, Vol. 7, Num. 3, pp. 278-288, 1991.

[2] I. Ulrich, J. Borenstein, "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots", Proc. 1998 IEEE Intl. Conf. on Robotics and Automation (ICRA98), pp. 1572 - 1577, 1998

[3] D. Fox and W. Burgard and S. Thrun, "The Dynamic Window Approach to Collision Avoidance", Technical Report, University of Bonn, IAI-TR-95-13, 1995.

[4] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach", Proc. 1999 IEEE Intl. Conf. on Robotics and Automation (ICRA99), 1999

Author Information:

Christof Schröter, Matthias Höchemer, Horst-Michael Gross
Department of Neuroinformatics and Cognitive Robotics
Faculty of Computer Science and Automation
Ilmenau Technical University
PO Box 10 05 65
98684 Ilmenau
Tel: +49 3677 69 1306
Fax: +49 3677 69 1665
E-mail: christof.schroeter@tu-ilmenau.de