On-Line Obstacle Avoidance at High Speeds

Zvi Shiller and Sanjeev Sharma
The Paslin Laboratory for Robotics and Autonomous Vehicles
Ariel University Center of Samaria, Israel

Abstract This paper presents an efficient algorithm for on-line obstacle avoidance that accounts for robot dynamics and actuator constraints. The robot trajectory (path and speed) is generated on-line by avoiding obstacles, optimally, one at a time. The trajectory is generated recursively using a *basic* algorithm that plans trajectory segments to intermediate goals. The use of intermediate goals ensures safety and convergence to the global goal. This approach reduces the original problem of avoiding m obstacles to m simpler problems of avoiding *one* obstacle each, producing a planner that is linear, instead of exponential, in the number of obstacles.

1 Introduction

This paper presents an algorithm for obstacle avoidance at high speeds. It generates trajectories that satisfy robot dynamics and actuator constraints, and are guaranteed to reach the goal from any admissible state. The incremental generation of the trajectories and the relatively low computational requirements at each step make it suitable for on-line applications.

The time-optimal control problem for robotic manipulators has been addressed in numerous studies over the past twenty years (e.g. Shiller and Dubowsky (1989)). This problem is inherently *off-line*, as it requires the solution of a two point boundary value problem. The typically nonlinear and coupled robot dynamics make such solutions computationally extensive. Adding obstacles makes the computational challenge even harder. Recent works have addressed the problem of on-line trajectory planning and high speed obstacle avoidance, using sampling-based methods (Kuwata et al., 2008; Knepper and Mason, 2009), and mixed-integer linear programming (MILP) (Vitus et al., 2008). Sampling-based methods are usually inefficient for tightly spaced environments with narrow passages. The MILP based method is computationally intensive and hence limited to very simple cases.

The on-line algorithm presented in this paper was motivated by the observation that the effect of an obstacle on the value function (the global cost-to-go function) is local. Focusing on one obstacle, while ignoring the rest,

effectively approximates the multi-obstacle problem of avoiding m obstacles by m simpler sub-problems of avoiding one obstacle each. The trajectory is thus generated *on-line*, incrementally, one step at a time, requiring a low computational effort at each step relative to the original, inherently off-line, problem.

Convergence to the goal is guaranteed, subject to a few assumptions, by selecting at each step the obstacle with the highest cost. When those assumptions are not satisfied, the original problem is subdivided by intermediate goals to smaller problems for which the assumptions are satisfied. The algorithm is demonstrated in a numerical example for a planar point robot moving among many (70) tightly spaced circular obstacles.

2 Problem Formulation

We wish to minimize the cost function

$$\min_u \int_0^{t_f} 1 \; dt, \tag{1}$$

subject to system dynamics

$$\ddot{x} = f(x,u) \; ; x,u \in \mathbb{R}^n, \tag{2}$$

where $x \in \mathbb{R}^n$ is the vector of axis displacements, and $u \in \mathbb{R}^n$ is a vector of actuator efforts, subject to the actuator constraints, obstacle constraints and the boundary constraints:

$$|u_i| \leq 1 \; , \qquad i \in \{1, \cdots, n\} \tag{3}$$

$$g(x) \geq 0 \; ; \qquad g \in \mathbb{R}^m \tag{4}$$

$$x(0) = x_0; \; x(t_f) = x_f; \quad \dot{x}(0) = \dot{x}_0; \; \dot{x}(t_f) = \dot{x}_f \tag{5}$$

where t_f is free, and m is the number of obstacles. We assume that the obstacles (4) do not overlap with each other and with the goal x_f.

Problem (1) is difficult to solve because of the general nonlinear robot dynamics (2) and the obstacle (state) constraints (4). In addition, the obstacles introduce multiple local minima by creating multiple distinct pathways. The size of the problem can be measured by the number of such local minima. In a planar environment, each obstacle introduces two local minima (one on each side). The total number of local minima for m obstacles is therefore 2^m, which demonstrates the inherent difficulty of the time-optimal avoidance problem. This represents a computational challenge for on-line, as well as for off-line, avoidance. We settle for an approximate solution

by reducing the original problem to many simpler sub-problems that avoid obstacles optimally one at a time.

3 Optimal Avoidance of a Single Obstacle

The optimal avoidance of a single obstacle is relatively simple. We solve it for the following point mass model:

$$\ddot{x} \;=\; u_1 \;;\; |u_1| \;\leq\; 1 \qquad \ddot{y} \;=\; u_2 \;;\; |u_2| \;\leq\; 1 \tag{6}$$

where (x, y) and (u_1, u_2) represent the position and the actuator efforts, respectively, along the x and y axes. We first compute the *unconstrained* trajectory, for states not affected by the presence of the obstacle.

3.1 The Unconstrained Optimal Trajectory

The unconstrained optimal trajectory is the solution to the minimum time problem from any point in the state-space to the target state for the obstacle-free problem. For the decoupled system (6), it is determined by the minimum time of the *slowest* axis.

Consider first a single axis, represented by the double integrator

$$\dot{x}_1 = x_2 \qquad \dot{x}_2 = u; \quad |u| \leq 1 \tag{7}$$

Denoting, $x = (x_1, x_2)$, $x_0 = (x_{10}, x_{20})$ and $x_f = (x_{1f}, x_{2f})$, it can be shown that the time-optimal control for system (7) from any state x to the target x_f is bang-bang with at most one switch. The minimum *time-to-go* from any x to x_f can be computed analytically and will be skipped for the sake of brevity.

Since the minimum time trajectory for a single axis has only one switch (excluding initial states on the switching curves), reaching the target at time T greater than the minimum time, t_f, using bang-bang control, would require more than one switch (Sundar, 1994). The trajectories that reach the target state in a specified time ($T > t_f$) are not unique since the number of switches and their timing are not unique. They are, however, bounded by two bang-bang trajectories with only two switches each (Sundar, 1994).

The single axis extremal controls, $u_{max}(t)$ and $u_{min}(t)$, that generate the two extremal trajectories that reach the target at a specified time $T > t_f$, can be computed analytically:

$$u_{\max}(t) = \begin{cases} 1 & \text{if } t \in [0, t_{s1}] \\ -1 & \text{if } t \in [t_{s1}, t_{s2}] \\ 1 & \text{if } t \in [t_{s2}, T] \end{cases} \quad u_{min}(t) = \begin{cases} -1 & \text{if } t \in [0, t_{s3}] \\ 1 & \text{if } t \in [t_{s3}, t_{s4}] \\ -1 & \text{if } t \in [t_{s4}, T] \end{cases} \tag{8}$$

where

$$\begin{aligned} t_{s1} &= \frac{1}{2\alpha}(x_{1f} - x_{10} + 2\alpha T - x_{20}T - \frac{T^2}{2} - \alpha^2) \\ t_{s2} &= t_{s1} + \alpha \\ \alpha &= \frac{(T + x_{20} - x_{2f})}{2}, \end{aligned} \tag{9}$$

$$\begin{aligned} t_{s3} &= \frac{1}{2\beta}(x_{1f} - x_{10} - 2\beta T - x_{20}T + \frac{T^2}{2} + \beta^2) \\ t_{s4} &= t_{s3} + \beta \\ \beta &= \frac{(T - x_{20} + x_{2f})}{2}. \end{aligned} \tag{10}$$

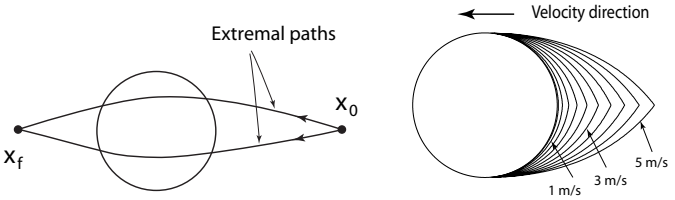
For the two axes system (6), the minimum motion time for the unconstrained problem from any state $x = (x_1, x_2, y_1, y_2)$ to the target state $x_f = (x_{1f}, x_{2f}, y_{1f}, y_{2f})$ is determined by the optimal motion time t_f of the slowest axis. The time-optimal trajectory is thus obtained by driving the slowest axis optimally, and driving the other axis so that it reaches the target at the same final time, t_f. The trajectory of the slowest axis is optimal, and unique with one switch. The trajectory of the other axis reaches the goal at a non-optimal time, and has therefore *at least* two switches, and is not unique. It follows that the time-optimal path (the projection of the trajectory to the configuration space) between the end points is not unique. The set of all time-optimal paths is bounded by two *extremal* paths, generated by the *extremal* trajectories (8) of the slowest axis.

3.2 The Constrained Optimal Trajectory

The extremal trajectories are used to determine whether an obstacle is avoidable using the *unconstrained* solution. If both extremal trajectories intersect the obstacles (see Figure 1a), then it must be avoided using the *constrained* solution discussed next. We refer to the set of points from which both extremal trajectories intersect the obstacle as the Dynamic Obstacle Shadow.

The time optimal trajectory that avoids the obstacle is computed numerically, using a line search over the terminal time. In this search, the terminal motion time is increased from the unconstrained optimal time t_f until at least one extremal avoids the obstacle. The optimal constrained trajectory has two switches in both axes.

The dynamic obstacle shadow may include *infeasible* states from which the obstacle is *unavoidable*, i.e. the entire set of attainable positions, generated by the controls satisfying (3), intersects the obstacle at some time



(a) A point in the dynamic obstacle shadow

(b) Obstacle hole

Figure 1: Obstacle shadow and obstacle hole.

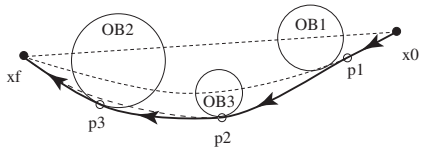


Figure 2: The avoidance procedure demonstrated for three obstacles.

$t \in [0, t_f]$. The set of all infeasible states forms, what we call, an *Obstacle Hole*, shown in Figure 1b for a circular obstacle for velocities pointing left. The *obstacle hole*, shown in Figure 1b, is represented by curves of constant speeds, generated at $0.5m/s$ increments up to $5m/s$. Each curve represents positions from which the sharpest avoidance maneuver, initiated at the corresponding velocity, is tangent to the obstacle.

4 Multi-Obstacle Avoidance

The avoidance of a single obstacle is now used to avoid multiple obstacles, *one at a time*. It remains to determine which obstacle is to be avoided at any given time. The *current* obstacle to be avoided is selected as the one that requires the longest time to avoid to the goal from the current state. This ensures that the obstacles requiring the largest deviation from the unconstrained trajectory to the goal will be considered first. This choice forms the *basic* algorithm, which under a few assumptions guarantees convergence to the goal, as discussed next.

4.1 The Basic Algorithm

The avoidance of one obstacle at a time is illustrated schematically in Figure 2 for three obstacles. The initial state, x_0, is assumed to be in the dynamic shadow of OB_1 and OB_2, hence both extremal trajectories from

that point (marked by dotted lines) cross both obstacles, as shown. Let the unconstrained optimal time from x_0 to x_f be, for example, $1s$, the constrained optimal time avoiding OB_1 be $2s$, and the constrained optimal time avoiding OB_2 be $1.5s$. OB_1 is, therefore, the maximum cost obstacle, and hence selected as the *current* obstacle. This obstacle is avoided until some point p_1, from which avoiding OB_3 takes longer than avoiding OB_1 or OB_2 to the goal. At that point, OB_3 is the maximum cost obstacle, selected to be avoided next. OB_3 is then avoided until some point p_2, from which OB_2 becomes the maximum cost obstacle. From p_3, x_f is reachable by an unconstrained optimal trajectory. This procedure is summarized in the following *basic* algorithm.

Algorithm 1: *The Basic Algorithm*

Initialize Set $x = x_0$. Select the termination condition ϵ, and time step Δt.

Step 1. Given x and x_f, determine the *current* obstacle, OB_k, to x_f. If $k = 0$, go to Step 2. Compute the optimal trajectory avoiding OB_k to x_f.

Step 2. Follow the optimal trajectory for some time step Δt. Update x. If $||x - x_f|| \leq \epsilon$, STOP.
Go to Step 1.

The basic algorithm can be proven to converge to the goal if the cost along the trajectory avoiding each obstacle decreases monotonically. This is generally the case if the following assumptions are satisfied: (i) the obstacles are convex and do not overlap; (ii) the trajectory does not enter an obstacle hole; and (iii) no two obstacles have equal cost at the same time.

The last two assumptions are generally difficult to satisfy unless the obstacles are spaced far apart. A potential violation of assumption (ii) is countered by looking one step ahead to warn of a possible fall into an obstacle hole. In the case a trajectory is about to collide with an obstacle, that obstacle becomes the *current* obstacle and is avoided next, often causing a local increase in the cost-to-go. A violation of Assumption (iii) may cause the trajectory to follow a curve that is not necessarily optimal for any obstacle, thus causing the cost along the trajectory to increase rather than decrease. This is remedied by defining an intermediate goal on the boundary of the current obstacle, and applying the *basic* algorithm to that goal. When the intermediate goal is reached, the trajectory is planned again to the global goal. This ensures that obstacles with a cost to the goal equal to that of the *current* obstacle are no longer competitive when the goal lies on the boundary of the *current* obstacle. The *basic* algorithm is thus applied recursively to each intermediate goal, until reaching the global goal.

4.2 Convergence

Convergence can be proven if assumptions (1-3) are satisfied, since the cost along the trajectory reduces monotonically by virtue of the properties of the value function. In case assumption (3) is violated, each intermediate goal subdivides the trajectory from the current state to the goal into two segments. Since every division does not involve previously avoided obstacles, the number of such divisions is finite as the number of obstacles is assumed finite. Hence, the trajectory will terminate at the final goal at a finite time after following a final number of trajectory segments.

5 Example

Figure 3 shows a trajectory of a point mass robot, computed online using Algorithm 1, that avoids 70 circular planar obstacles. It starts at the bottom right point at zero speed, and terminates at the top left corner at zero speed. The trajectory is marked by small dots, spaced at equal time intervals. The intermediate goals selected by the algorithm along the trajectory are shown as hollow circles. The large number of intermediate goals (14) reflects the tight spaces through which the robot has to maneuver. The travel time for this trajectory is $39.8s$, and its travel distance is $89.0m$ (compared to the straight line distance of $81.4m$), resulting in an average speed of $2.2m/s$. The average computation time per-step was $8ms$, for a total computation time of $3.8s$, with $\Delta T = 0.1s$, implemented in MATLAB. This example demonstrates the algorithm's successful handling of narrow passages. The algorithm was tested for 500 cases, with end points selected randomly in the same environment as in Figure 3. All 500 cases succeeded reaching the goal, demonstrating the robustness of this algorithm for a challenging environment.

6 Conclusions

An efficient approach for on-line sub-time optimal obstacle avoidance has been presented. It considers robot dynamics while attempting to minimize motion time. The algorithm is based on avoiding obstacles optimally one at a time, thus reducing the complex problem of time-optimal avoidance of m obstacles to m simpler problems, each avoiding optimally a single obstacle. This reduces the complexity of the original problem from exponential to linear in the number of obstacles. This significant reduction and the incremental nature of this approach make it suitable for on-line applications, such as mobile robots moving amongst many obstacles and in changing environments. Despite its simplicity, this approach converges to the goal from any feasible state, provided that the obstacles are avoided in the order of their

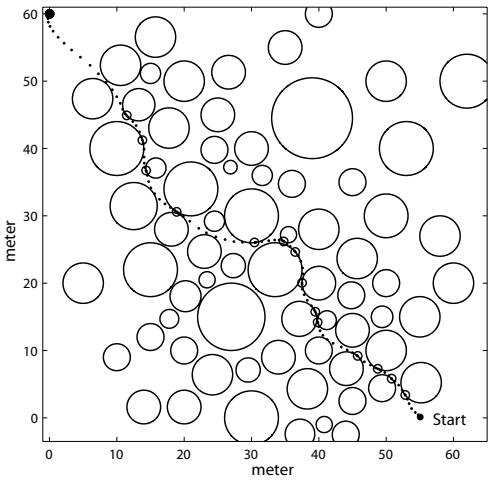


Figure 3: Navigation through 70 obstacles

cost. The algorithm was demonstrated to produce high speed trajectories through very crowded and tight spaces.

Bibliography

R.A. Knepper and M.T. Mason. Path diversity is only part of the problem. In *ICRA*, 2009.

Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How. Motion planning for urban driving using RRT. In *ICRA*, 2008.

Z. Shiller and S. Dubowsky. Time-optimal path-planning for robotic manipulators with obstacles, actuator, gripper and payload constraints. *IJRR*, 8(6):3–18, 1989.

S. Sundar. *Near-Optimal Feedback Control of Robotic Manipulators*. Ph.D. Dissertation, Dept. of Mechanical, Aerospace and Nuclear Engineering, University of California, Los Angeles, CA, 1994.

M. P. Vitus, V. Pradeep, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin. Tunnel-milp: Path planning with sequential convex polytopes. In *AIAA Guidance, Navigation and Control Conf.*, 2008.