Proceedings of the 2nd International Symposium
on Adaptive Motion of Animals and Machines,
Kyoto, March.4-8, 2003

On-Line Dynamic Obstacle Avoidance

Zvi Shiller

Mechanical Engineering-Mechatronics, College of Judea and Samaria

Ariel, 44837, Israel, shiller@yosh.ac.il

Abstract

This paper presents an efficient algorithm for on-line dynamic obstacle avoidance. The robot trajectory (path and speed) is generated on-line by avoiding obstacles optimally one at a time. The resulting planner converges globally to the goal, and is applicable to any cost function and to any robot models. It is demonstrated here for shortest paths and minimum time motions for a point robot moving in the plane.

1. Introduction

To avoid obstacles, humans follow what appears to be the negative gradient of a potential field by treating the goal as an attractor, and obstacles as repellors [14]. They select trajectories that are consistent with an on-line "dynamic control strategy" [14], meaning that their motions are smooth and respect some system dynamics. This recent result validates on-line strategies for obstacle avoidance that have been used by the robotics community for quite some time since the inception of the potential field approach [3, 2, 8], and the more recent incorporation of robot dynamics in the on-line obstacle avoidance problem [12].

In this paper we review an on-line obstacle avoidance strategy that combines a potential field approach with robot dynamics, and seems applicable to on-line trajectory planning of complex systems such as humanoids moving through cluttered environments. The simple trajectories followed by humans despite the high complexity of their dynamics suggests that similar simplifications can be used for on-line obstacle avoidance of mobile robots, as well as of humanoids.

The avoidance strategy presented here is based on following the direction of steepest descent along the value function, which represents the *cost-to-go* to the goal from any given state [1]. The value function reflects the cost function being considered, has a unique minimum at the goal, generates globally optimal trajectory, and is hence an ideal potential function.

The value function is difficult, if not impossible, to compute for a large number of obstacles and for "dynamic" cost functions (that require system dynamics) such as motion time and energy. We circumvent this difficulty by a) decomposing the problem into the avoidance of one obstacle at a time, and b) by computing the optimal direction locally without actually deriving the value function. It is shown that despite these simplifications, the resulting "potential filed" has a unique minimum regardless of the cost function being considered and the number of obstacles.

In contrast, traditional potential field methods [2, 3, 4, 5, 6, 8, 13] typically do not attempt to optimize a specific cost function, and some suffer from local minima [3].

There are currently no comparable on-line planners that consider robot dynamics *and* attempt to minimize a "dynamic" cost function. The closest approach is Lumelsky's sensor-based planner that accounts for robot dynamics [10]. This planner moves dynamically (considering robot dynamics) between via points, selected kinematically using a simple bug algorithm.

We make no attempt to quantify the optimality of this planner as it is problem specific and hence meaningless as a measure for general applications.

Anecdotal numerical experiments show close correlation between the on-line and the global optimal trajectories. Although the on-line planner does not generate the optimal trajectory, it does generate a "good" alternative to the exact solution of the original problem.

2. Problem Formulation

We wish to minimize the cost function:

$$\min_{u} \int_0^{t_f} L(x,u)\ dt + b(d) \tag{1}$$

subject to the autonomous (time-invariant) system dynamics:

$$\dot{x} = f(x,u)\ ,\ x \in \mathcal{R}^n\ ,\ u \in \Omega \subset \mathcal{R}^p \tag{2}$$

actuator constraints:

$$|u_i| \leq 1\ ,\ \ i\ \in\ \{1,\cdots,p\} \tag{3}$$

state dependent constraints:

$$g(x) \geq 0\ ,\ \in \mathcal{R}^m \tag{4}$$

and boundary conditions:

$$x(0)\ \ =\ \ x_0;\ \ x(t_f)\ \ =\ \ x_f \tag{5}$$

where t_f is free, $L(x,u) \in \mathcal{R}$ is a positive cost functional, $f(\cdot,\cdot)$ is differentiable in both arguments, the set of feasible controls, Ω, is bounded and convex, m is the number of obstacles, $b(\cdot)$ is a *barrier* function of the form

$$b(d) = -\mu log(d), \tag{6}$$

d is the minimum distance to any obstacle along the optimal path, and $\mu > 0$ is a small penalty factor. The barrier function is used to ensure that the path does not graze any obstacle, and to repel it from potential traps (regions from which collision is unavoidable).

3. The Hamilton-Jacobi-Bellman Equation

A sufficient condition for the time-optimal control problem (1), known as the Hamilton-Jacobi-Bellman equation, is stated in the following theorem:

Theorem 1 [7]: The control $u^*(x)$ is the solution to the optimal control problem for (1) if it satisfies, on $\mathcal{R}^n - \{x_f\} - \mathcal{I}$, the HJB equation:

$$\min_{u \in \mathbf{\Omega}} \{v_t(x,t) + \langle v_x(x,t), f(x,u)\rangle\} = -L(x,u), \tag{7}$$

subject to (4), where $v(x,t)$ is a continuous scalar function, which is piecewise C^2 on $\mathcal{R}^n - \{x_f\} - \mathcal{I}$, satisfying

$$v(x_f,t) = 0 \tag{8}$$

$$v(x,t) > 0\ \ ,x\ \neq\ x_f \tag{9}$$

and $\mathcal{I}$ is the set of infeasible states:

$$\mathcal{I}\ \ =\ \ \{x : g(x) < 0\}. \tag{10}$$

The subscripts x and t represent partial derivatives with respect to x and t, respectively. For autonomous systems and fixed terminal conditions, v is not an explicit function of t. Hence, $v_t = 0$.

The value function $v(x)$ represents the minimum *cost-to-go* from x to x_f. The optimal trajectory is generated by selecting the control u that minimizes the time-derivative, $\dot{v}(x) = \langle v_x(x), f(x,u)\rangle$, of the value function, where $< \cdot,\cdot >$ denotes the inner product on $\mathcal{R}^n$. This trajectory is globally optimal, and is guaranteed to reach the goal, x_f, since similar to Lyaponov functions, $\dot{v}(x(u^*))$ cannot vanish anywhere but at x_f.

Note that $\dot{v}(x)$ is minimized when $f(x,u) = \dot{x}$ points opposite of $v_x(x)$, the negative gradient of the value function. This is similar to a potential field approach, which generates the trajectory by following the negative gradient of the potential function. The value function can thus be viewed as a potential function that generates the *optimal* trajectory.

Treating the obstacle avoidance problem in the context of the HJB equation allows us to prove convergence using the properties of the value function. It also allows us to generalize this approach to any cost function because these properties apply to any functional optimization problem. So far, we have used this approach for the shortest path [11, 9] and the minimum time [12] problems.

4. The Obstacle Avoidance Strategy

Observing that the effect of an obstacle on the value function is local, we treat the multi-obstacle problem by avoiding obstacles optimally, one at

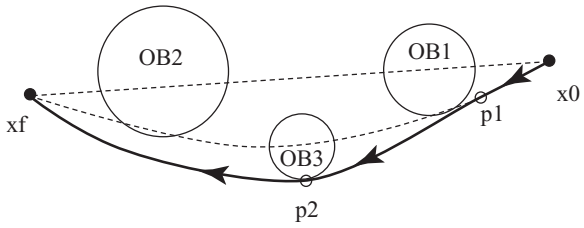

Figure 1: The avoidance procedure

a time. The avoidance procedure is simple: attempt to go straight to the goal. If the goal is obstructed by obstacles, pick the "largest" (in term of the cost function), avoid it until it no longer obstructs the goal, or until another obstacle become the "largest." Switch to the "largest" obstacle and repeat until reaching the goal.

This procedure is illustrated in Figure 1 for the avoidance of three obstacles, where the goal is obstructed from the initial state, x_0, by two obstacles: OB_1 and OB_2. Let the obstacle-free optimal time from x_0 to x_f be 1 s, the optimal time avoiding OB_1 be 2 s, and optimally avoiding OB_2 be 1.5 s. OB_1 is, therefore, the "largest" obstacle, and hence selected to be avoided first. This obstacle is avoided until reaching 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 becomes the "largest"' obstacle, and is avoided next until reaching some point p_2, from which $\mathbf{x}_f$ is reachable by an unconstrained optimal trajectory.

This procedure is summarized in the following algorithm.

Algorithm 1: *On-line obstacle avoidance*

> **Initialize** Set $x = x_0$. Select the termination condition ϵ, and time step Δt.
> **Step 1**. Given x and x_f, determine the "largest" obstacle, OB_k. If $k = 0$, go to Step 3. 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. □

Algorithm 1 assumes that the "largest" obstacle obstructing the goal is known at any given time. This restriction is not unreasonable since it is unrealistic to avoid obstacles dynamically (at various speeds) without knowledge of the obstacles ahead, as it is unrealistic to expect a human to run through a cluttered room blindfolded. The avoidance strategy of a "blind" robot would dictate kinematic avoidance at low speeds that passes through selected via points along the obstacle boundaries [9].

5. Implementation

The implementation of Algorithm 1 requires several computational routines:

- computation of the unconstrained optimal trajectory from any state to the goal,
- determination of an obstructing obstacle from an

6. Convergence

Global convergence of Algorithm 1 can be proven for a point robot and general obstacles, satisfying the following assumptions:
Assumption 1: The free-space is connected (no obstacles with holes).
Assumption 2: The obstacles are compact sets, do not intersect one another, and do not include the goal.
Assumption 3: The minimum distance between any two obstacles is greater than some constant $\epsilon > 0$.
Assumption 4: The number of obstacles is finite.

Assumptions 1 and 2 guarantee existence of a kinematic solution; Assumption 3 establishes a minimum distance, or time, to collision after having avoided an obstacle; and Assumption 4 ensures that the worst case number of switches is finite.

Theorem 2: The trajectory generated by Algorithm 1 is guaranteed to terminate at x_f in a finite time for all initial states $x_0 \in \mathcal{R}^n - \{x_f\} - \mathcal{I}$ ($\mathcal{I}$ defined in (10)).

Proof: The proof is based on the properties of the value function. Recall that the value function for a single obstacle satisfies the HJB equation (7). Rewriting (7) yields:

$$\min_{u \in \mathbf{\Omega}} \dot{v}(x, u, t) = -L(x, u) \qquad (11)$$

or

$$\dot{v}(x, u^*(t)) = -L(x, u) < 0 \qquad (12)$$

where u^* denotes the optimal control. Since $v(x, u, t)$ is positive and finite for all feasible states $x \neq x_f$, (12) implies convergence to x_f, where $v(x_f) = 0$. In other words, the optimal trajectory avoiding a single obstacle monotonically reduces the value function until it reaches x_f. The convergence time to x_f is by definition the value of the value function at the current state.

The trajectory generated by Algorithm 1, $x(t)$, switches to the value functions of the *largest* obstacles, selected globally to attain the maximum of all value functions at each point along $x(t)$. Since each value function monotonically decreases along $x(t)$, it remains to show that the number of switches and the time to each switch are finite.

The selection of the *largest* obstacle as soon as it exceeds the value function of the former obstacle ensures that the value function along $x(t)$ does not increase at the switching points. If Δt is the time between two consecutive switches, then by the properties of the individual value function, the augmented value function $\varphi(x(t), x_f)$ monotonically decreases:

$$\varphi(x(t + \Delta t), x_f) < \varphi(x(t), x_f) \qquad (13)$$

The size of each Δt is a function of the distance between the obstacles and robot speed. Assumption 3, and assuming a bounded speed, implies $\Delta t > 0$. That Δt is not zero and (12) precludes chatter without a reduction in cost. Δt is bounded by the maximum cost (the time to avoid the costliest obstacle) at the initial point. The number of switches is bounded by the number of obstacles, which is assumed to be finite. Note that loops are not possible since any loop would violate (13). This establishes that the number of switches and the time between switches are finite.

Since by construction, $\varphi(x(t), x_f)$ is positive and finite for all feasible states $x \neq x_f$, it follows that the augmented value function, $\varphi(x(t), x_f)$, vanishes after a finite number of steps:

$$\varphi(x(t + \alpha\Delta t), x_f) = 0 \qquad (14)$$

where α is a real integer. Since $\varphi(x(t), x_f)$ vanishes only at x_f, (14) implies $x(t+\alpha\Delta t) = x_f$, or that Algorithm 1 reaches the goal in a finite time. □y state to the goal,

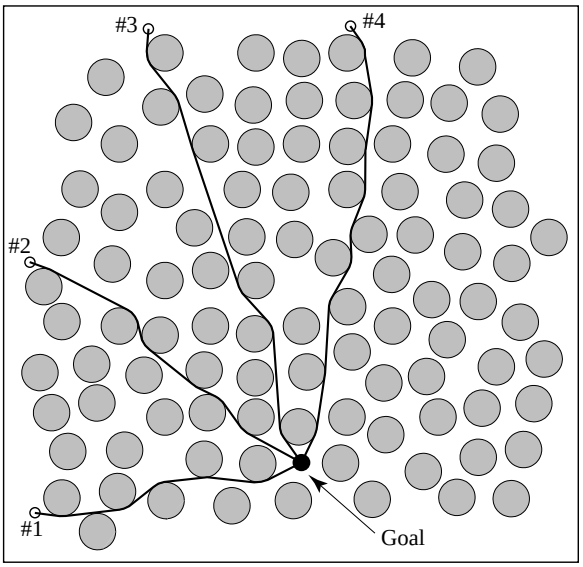

Figure 2: Near-shortest paths avoiding 100 circular obstacles

- computation of the constrained optimal trajectory that avoids a single obstacle from any state to the goal,
- determination of the "largest" obstacle.

The most challenging part of this approach is the optimal avoidance of a single obstacle. It is used to avoid an obstacle, and to evaluate the "largest" obstacle at every time step. Optimally avoiding an obstacle for the shortest path problem is a simple geometric task. Optimally avoiding an obstacle for the minimum time problem is not as trivial, but it can be formulated as a line search over one parameter (for details see [12]).

7. Examples

The following examples demonstrate Algorithm 1 for the kinematic (shortest path) avoidance of convex and general polygonal obstacles, and dynamic (minimum time) avoidance of convex planer obstacles.

7.1. Example 1

Figure 2 shows four paths computed on-line using the shortest path cost function through a cluttered space containing 100 circular obstacles. The computation times for each path varied around

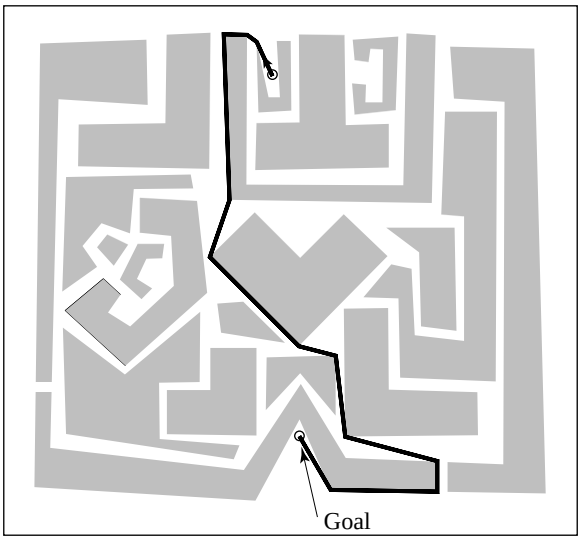

Figure 3: Near-shortest paths avoiding planer polygonal obstacles

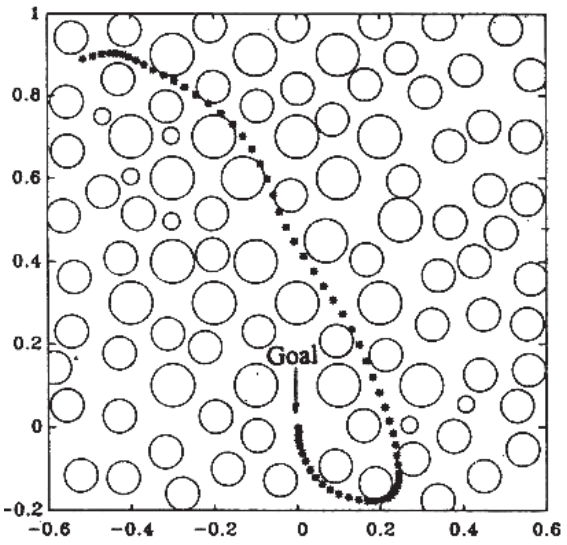

Figure 4: Dynamic avoidance of 100 circular obstacles

$10 - 20ms$. The paths are very close to the straight lines from each initial point, and are therefore near-optimal. They also pass between closely spaced obstacles, which would be infeasible for typical potential functions. Further, increasing the number of obstacles had very little effect on the computation time.

7.2. Example 2

Figure 3 shows 18 planar polygonal obstacles and a path computed on-line using the shortest path cost function. It was assumed that the robot recognizes the "largest" obstacle from every point.

7.3. Example 3

Figure 4 shows the dynamic (minimum time) avoidance of 100 circular obstacles by a point mass robot with actuator constraints. The spacing between the dots represents the speed along the path. The total computation time for the path was about 7 s, approximately 0.1 s per step.

8. Conclusions

This paper reviews an on-line planner that is similar to potential field methods, except that it follows the value function, which we view as the "optimal" potential function, it has no local minima, and is applicable to any cost function and any robot dynamics. It is currently the only on-line planner that considers robot dynamics while attempting to minimize a dynamic cost function. The planner is based on decomposing the multi-obstacle avoidance problem into the optimal avoidance of many single obstacles. The planner was demonstrated for a point robot avoiding general polygonal obstacles, and a point mass robot avoiding a large number (100) of circular obstacles, using the shortest distance and minimum time cost functions, respectively. Although this approach does not guarantee optimality, numerical examples demonstrate close correlation between the on-line solution and the global optimal paths. This approach is applicable to mobile robots moving at moderate to high speeds, and for generating reference trajectories for more complex systems such as humanoids moving through cluttered environments.

References

[1] L. Cesari. *Optimization - Theory and Applications: Problems with Ordinary Differential Equations.* Springer-Verlag, New York, 1983.

[2] C. I. Connoly, J. B. Burns, and R. Weiss.

Path planning using laplace's equation. *IEEE Conf. on Robotics and Automation, Cincinnati, OH*, 1:2102–2106, 1991.

[3] O. Khatib. Real time obstacle avoidance for manipulators and mobile robots. *Int'l Journal of Robotics Research*, 1:65–78, 1986.

[4] P. Khosla and R. Volpe. Superquadric artificial potentials for obstacle avoidance approach. *Proc. IEEE Int'l Conference on Robotics and Automation*, 3:1778–1784, 1988.

[5] J. O. Kim and P. Khosla. Real time obstacle avoidance using harmonic potential functions. *Proc. IEEE Int'l Conference on Robotics and Automation*, 1:790–796, 1991.

[6] D. E. Koditschek and E. Rimon. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, 11:412–442, 1990.

[7] A. I. Moskalenko. Bellman equations for optimal processes with constraints on the phase coordinates. *Automation and Remote Control*, 4:1853–1864, 1967.

[8] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Trans. on Robotics and Automation*, 8:501–518, 1992.

[9] Z. Shiller. On-line sub-optimal obstacle avoidance. *International Journal of Robotics Research*, 19(5):480–497, May, 2000.

[10] A.M. Shkel and V.J. Lumelsky. The jogger's problem: Control of dynamics in real-time motion planning. *Automatica*, 33(7):1219–1233, 1997.

[11] S. Sundar and Z. Shiller. Optimal obstacle avoidance based on sufficient conditions of optimality. *IEEE Transactions of Robotics and Automation*, 13(2):305–310, 1997.

[12] S. Sundar and Z. Shiller. Time-optimal obstacle avoidance. *IEEE Int. Conf. on Robotics and Automation*, pages 3075–3080, May 1995.

[13] R. Volpe and P. Khosla. Artificial potentials with elliptical isopotential contours for obstacle avoidance. *Proc. IEEE Int'l Conf. on Robotics and Automation*, 3:1778–1784, 1987.

[14] D. Belcher W. Warren, B. Fajen. Behavioral dynamics of steering, obstacle avoidance and route selection. *Journal of Vision*, 1(3), 2001.