High Speed On-Line Motion Planning in Cluttered Environments

Zvi Shiller and Sanjeev Sharma

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. This reduces the original problem from one with *m* obstacles to *m* simpler problems with *one* obstacle each, thus resulting in a planner that is linear, instead of exponential, in the number of obstacles. While this approach is quite general and applicable to any cost function and to any robot dynamics, it is treated here for minimum time motions, a point mass robot, and planar circular obstacles.**

I. Introduction

This paper presents an algorithm for near-optimal obstacle avoidance of a mobile robot moving in static cluttered environments. It generates trajectories that satisfy robot dynamics and actuator constraints, and can be proven to reach the goal when assuming zero terminal speeds. The incremental generation of the trajectories and the relatively low computational requirement at each step make this algorithm suitable for on-line applications. The avoidance of one obstacle at a time allows the algorithm to generate trajectories through very cluttered environments.

The time-optimal control problem for robotic manipulators has been addressed in numerous studies over the past twenty years (see for example [2], [6], [11], [14], [15]). 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.

Off-line computation of time-optimal trajectories may suffice in applications involving repeatable tasks, but is somewhat useless for "on-line" applications, when the next move is determined during motion (on-line). Obviously, *waiting* for a time-optimal trajectory to be computed defeats the purpose of minimizing time.

One approach to solving the time optimal control problem *on-line* is to derive a time-optimal control law that drives the system time-optimally to the goal from any initial state. To derive such a control law requires solving the Hamilton-Jacobi-Bellman (HJB) equation, which in turn requires the derivation of the *value* function [1]. The globally optimal trajectory is then generated by selecting the controls that minimize the time derivative of the value function. Although the theoretical framework exists for deriving optimal feedback controllers, it is impractical to derive a time-optimal control law for a typical obstacle avoidance problem that accounts for robot dynamics.

Recent works have addressed online motion planning, using RRT [8], local path-set generation [7] and mixed-integer linear programming (MILP) [10]. However, they are *inefficient* for crowded and tightly spaced environments with narrow passages. Online avoidance can be achieved by navigating in the robot's velocity space using Velocity Obstacles [9]. While this approach is applicable to static and moving obstacles, it requires that the time horizon be carefully determined, for otherwise the robot may avoid passing through tight spaces. The selection of the proper time horizon is still unresolved.

In this paper, we address the online obstacle avoidance problem in static environments. Motivated by the observation that the effect of an obstacle on the value function (the global cost-to-go function) is local, we solve the multi-obstacle problem by avoiding obstacles one at a time. Computationally, this transforms the multi-obstacle problem with *m* obstacles to *m* simpler sub-problems with one obstacle each, thus reducing the size of the problem from exponential to linear in the number of obstacles. As a result, this approach produces an *on-line* planner, i.e. the trajectory is generated incrementally, one step at a time, requiring a low computational effort at each step relative to the original, inherently off-line, problem. The algorithm is demonstrated in several examples for a planar point robot moving among many (70) obstacles.

II. Optimal Avoidance of a Single Obstacle

The time optimal avoidance of a single obstacle in the plane is relatively simple. It can be computed using a global optimization [12], or by running a local optimization [11] twice (one for each side of the obstacle for a planar problem). For simplicity, we choose to represent the robot by a point mass model in the plane. This simplification is for computational reasons, and is in no way a limitation of this approach.

Consider the following point mass model:

$$\begin{aligned} \ddot{x} &= u_1 \; ; \; |u_1| \leq 1 \\ \ddot{y} &= u_2 \; ; \; |u_2| \leq 1 \end{aligned} \qquad (1)$$

where $(x,y)^T \in \mathbb{R}^2$ and $(u_1,u_2)^T \in \mathbb{R}^2$ represent the positions and actuator efforts along the x and y axes, respectively . Define the state vector for system (1) as $\mathbf{x} = (x_1, x_2, y_1, y_2)^T$, where $(x_1,x_2) = (x,\dot{x})$ and $(y_1,y_2) = (y,\dot{y})$.

This work was performed at the Paslin Laboratory for Robotics Autonomous Vehicles at the Ariel University Center, Israel.

Zvi Shiller is with the Department of Mechanical Engineering and Mechatronics, Ariel University Center of Samaria, Israel (shiller@ariel.ac.il)

Sanjeev Sharma is a visiting scholar at the Paslin Laboratory for Robotics Autonomous Vehicles at the Ariel University Center, Israel

We first derive the *unconstrained* trajectory, for states not affected by the presence of the obstacle.

A. The Unconstrained Trajectory

The unconstrained trajectory for the decoupled system (1) is determined by the minimum motion time of the *slowest* axis.

Consider first a single axis, represented by the double integrator

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \; ; \; |u| \le 1. \end{aligned} \quad (2)$$

Using optimal control theory [3], it is easy to show that the time-optimal control for system (2) is bang-bang with at most one switch. In the following, we denote $\mathbf{x} = (x_1, x_2)$ and $\mathbf{x}_f = (x_{1f}, x_{2f})$:

The minimum *time-to-go* from any state $\mathbf{x}$ to $\mathbf{x}_f$ can be computed analytically [4]:

$$t_f(\mathbf{x},\mathbf{x}_f) = -x_2 - x_{2f} + 2\sqrt{-x_1 + x_{1f} + \frac{x_2^2}{2} + \frac{x_{2f}^2}{2}}, \quad if \mathbf{x} \in \mathbf{R}$$
$$x_2 + x_{2f} + 2\sqrt{+x_1 - x_{1f} + \frac{x_2^2}{2} + \frac{x_{2f}^2}{2}}, \quad otherwise$$

where

$$\mathbf{R} = \{(\mathbf{x}) \,|\, S_1(\mathbf{x}) > 0, S_2(\mathbf{x}) < 0\}, \quad (3)$$

and

$$\begin{aligned} S_1(\mathbf{x}) &= x_2^2 - 2(x_1 - x_{1f} + \frac{x_{2f}^2}{2}) = 0, \\ S_2(\mathbf{x}) &= x_2^2 + 2(x_1 - x_{1f} - \frac{x_{2f}^2}{2}) = 0. \end{aligned} \quad (4)$$

Since the minimum time trajectory has only one switch (excluding initial states on the switching curves), reaching the target at a time greater than the minimum time, t_f, using bang-bang control, requires more than one switch, as stated in the following Lemma, ommitting the proof for brevity.

Lemma 2.1: Let t_f be the optimal time from some state $\mathbf{x}$, that is not on one of the switching curves, to $\mathbf{x}_f$. Any bang-bang trajectory from $\mathbf{x}$ to $\mathbf{x}_f$ that takes time $t > t_f$ has at least two switches.

The trajectories that reach the target state in a specified time (greater than the optimal time 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. We call the two-switch trajectories the *extremal* trajectories.

The extremal controls, u_{max} and u_{min}, that generate the two extremal trajectories can be computed analytically:

$$u_{\max}(t) = \begin{cases} 1 & \text{if } t \in [0, t_{s1}] \\ -1 & \text{if } t \in [t_{s1}, t_{s2}] \\ 1 & \text{if } t \in [t_{s2}, T] \end{cases} \quad (5)$$

$$u_{min}(t) = \begin{cases} -1 & \text{if } t \in [0, t_{s3}] \\ 1 & \text{if } t \in [t_{s3}, t_{s4}] \\ -1 & \text{if } t \in [t_{s4}, T] \end{cases} \quad (6)$$

where $T > t_f$ is specified, and

$$\begin{aligned} t_{s1} &= \frac{1}{2\alpha}(x_{1f} - x_{10} + 2\alpha T - x_{20}T - \frac{T^2}{2} - \alpha^2) \\ t_{s2} &= t_{s1} + \alpha \\ \alpha &= \frac{(T + x_{20} - x_{2f})}{2}, \end{aligned} \quad (7)$$

$$\begin{aligned} t_{s3} &= \frac{1}{2\beta}(x_{1f} - x_{10} - 2\beta T - x_{20}T + \frac{T^2}{2} + \beta^2) \\ t_{s4} &= t_{s3} + \beta \\ \beta &= \frac{(T - x_{20} + x_{2f})}{2}. \end{aligned} \quad (8)$$

Returning to the two axes system (1), the unconstrained trajectory from any state $\mathbf{x} = (x_1, x_2, y_1, y_2)$ to the target state $\mathbf{x}_f = (x_{1f}, x_{2f}, y_{1f}, y_{2f})$ is determined by the optimal motion time of the slowest axis. Henceforth, we assume, without loss of generality, that the faster axis from any initial state $\mathbf{x}_0 = (x_{10}, x_{20}, y_{10}, y_{20})$ to the target state $\mathbf{x}_f$ is the y-axis. The time-optimal trajectory is thus obtained by driving the x-axis optimally, using (3), and driving the y-axis so that it reaches the target at the same final time, t_f. The trajectory of the x-axis is unique with one switch. The trajectory of the faster y-axis has *at least* two switches, and is thus not unique. It follows that the time-optimal path 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 of the slowest axis. Note that if the optimal motion times of both axes are identical, then the time-optimal trajectory is unique.

The extremal trajectories are needed to determine whether an obstacle is avoidable by an *unconstrained* trajectory. The *unconstrained* trajectory can be used as long as at least one extremal trajectory avoids the obstacle. Otherwise, the obstacle must be avoided using the *constrained* trajectory discussed next.

B. The Constrained Trajectory

The constrained trajectory applies to points in the state-space from which *all* unconstrained time-optimal trajectories to the target intersect the obstacle, as shown schematically in Fig. 1. We refer to the set of such points as the Obstacle Shadow. In the kinematic case [16], the shadow corresponds to the shadow created behind the obstacle by a point light source at the target. The physical analogy for the dynamic problem is not as obvious. In the following, we consider one planar obstacle.

Definition 2.1: Obstacle Shadow. The Obstacle Shadow, D, of obstacle OB is the set of initial states from which *all* unconstrained time-optimal paths (the projections of the trajectories on the position space) intersect the obstacle.

The intersection of all the *extremal* time-optimal paths with the obstacle implies the intersection of all *unconstrained* optimal paths. Therefore, determining if a state belongs to

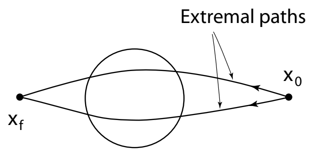

Fig. 1. A point in the obstacle shadow.

the shadow of a given obstacle requires only to determine if both extremal trajectories intersect the obstacle. For points lying in the obstacle shadow, we compute the optimal time of the trajectory that avoids the obstacle, numerically, using a line search (one parameter optimization).

The set of all feasible trajectories for system (1) that reach the goal at a specified time is bounded by two extremal trajectories, corresponding to combinations of the extremal trajectories for the individual axes (5,6). Thus, the obstacle *cannot* be avoided in time $t_f + \delta$ if and only if both extremal trajectories intersect the obstacle. It is easy to show that a sufficient increase in δ should yield an avoiding trajectory. We therefore compute the constrained time-optimal trajectory by selecting δ such that *at least one* extremal trajectory from the current state to the target does *not* intersect the obstacle. The computation of the constrained trajectory for avoiding one obstacle is, therefore, a single dimensional optimization, *independent* of the dimension of the state-space, and of the particular shape of the obstacle.

So far we have considered states from which the presence of the obstacle affects the optimal motion time to the target. The obstacle shadow may include *infeasible* states from which the obstacle is *unavoidable*. We call it the *Obstacle Hole*, as discussed next.

C. Infeasible States: The Obstacle Hole

A state from which the entire set of attainable positions, generated by controls satisfying (1) , intersects the obstacle at some time $t \in [0, t_f]$ is an *infeasible* state, i.e., a state from which all feasible trajectories intersect the obstacle. The set of all infeasible states forms the Obstacle Hole. The obstacle hole is similar to the ICS (Inevitable Collision States) [5].

Definition 2.2: Obstacle Hole. A given state $\mathbf{x}$ is said to lie in the obstacle hole OH of a planar obstacle OB if all feasible trajectories from $\mathbf{x}$ intersect OB at some time $t > 0$.

Fig. 2 shows the obstacle hole for a circular planar obstacle for motion at velocities pointing in the left direction. Each curve represents the boundary of the set of positions from which the obstacle is unavoidable at a specific velocity.

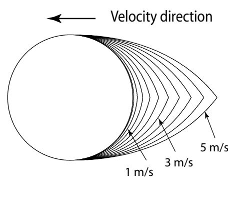

Fig. 2. The obstacle hole for a circular planar obstacle for velocities pointing leftwards.

III. Multi-Obstacle Avoidance

The optimal avoidance of one obstacle is relatively simple, and is hence suitable for on-line computation. We use it to solve the multi-obstacle problem by avoiding obstacles *one at a time*. Key to this approach is the selection of the *current* obstacle to be avoided at any given time, as discussed next.

A. The Current Obstacle

We choose the *current* obstacle to be the one that takes the longest time to avoid from the current state to the goal. Selecting at each step the obstacle with the highest cost to the goal produces a trajectory that is close to optimal.

The current obstacle is thus selected by first determining all obstacles with shadows containing the current state $\mathbf{x}$, then computing the constrained trajectories avoiding each obstacle to the goal, and selecting the one with the largest motion time. If $\mathbf{x}$ does not lie in the shadow of any obstacle, then the cost of all obstacles equals to the unconstrained trajectory to the goal and none is selected to be avoided. One of the unconstrained trajectories is then selected for navigation .

B. The Avoidance Algorithm

The avoidance algorithm assumes convex and non-overlapping obstacles (in the configuration space). It selects the current obstacles to be avoided, computes the time optimal trajectory that avoids that obstacle, and repeats the process until the final goal is reached. To avoid chatter between obstacles of equal cost, the algorithm selects an intermediate goal on the boundary of the current obstacle. A few intermediate goals might be selected, repeating the algorithm recursively to each one. The use of intermediate goals ensures convergence to the global goal in a finite time, as discussed later.

```
Algorithm 1: Avoidance using intermediate goals
Initialize Set current state $x = x_0$, current goal $x_g(i) = x_f$, index of current goal $i = 0$. Select the termination condition $\varepsilon$, and time step $\Delta t$.
Step 1. Determine the index $k$ of the current obstacle, $OB_k$, to $x_g(i)$.
If $k = 0$, go to Step 2.
Compute the optimal trajectory $x_c(t)$ avoiding $OB_k$ to $x_g(i)$.
Increment index of intermediate goals $i = i + 1$.
Select an intermediate goal $x_g(i)$ on the boundary of $OB_k$.
If $x_g(i)$ is not in the obstacle hole of any obstacle, go to Step 2.
Reduce the speed of $x_g(i)$ (lower speed while retaining position and direction) until the modified $x_g(i)$ is not in the hole of any obstacle.
Step 2. Follow the optimal trajectory for some time step $\Delta t$.
Update $x$.
If $|x - x_g(i)| \leq \varepsilon$, $i = i - 1$. If $i = -1$, STOP.
Go to Step 1.
```

Algorithm 1 progresses to the goal via a series of selected intermediate goals. Each intermediate goal $x_g(i)$ $(i \geq 1)$

is selected along the constrained trajectory $x_c(t)$ from the current state x to the current goal $x_g(i-1)$ at a point where $x_c(t)$ is tangent to the current obstacle OB_k. Usually, there is just one such point. In case $x_c(t)$ follows the obstacle for some time, the point closest (in time) to the goal $x_g(i-1)$ is selected.

Algorithm 1 may fail if the shadows of two obstacles intersect because the trajectory will chatter between the two obstacles until reaching a dead end at the intersection point. This is remedied by reducing the target velocity at the intermediate goal to the maximum velocity from which the trajectory can circle the current obstacle without violating the acceleration constraints. Denoting this velocity as the *curvature* velocity, it can be easily proven that the *curvature* velocity does not lie in the obstacle hole of any obstacle, as stated in the following Lemma.

Definition 3.1: Curvature Velocity. The *curvature* velocity, v_c, is defined as:

$$v_c = \sqrt{u_{max} R} \tag{9}$$

where u_{max} is the maximum acceleration, and R is the radius of the obstacle.

Lemma 3.1: Under the assumption that the obstacles are convex and do not overlap, the *curvature* velocity, as defined in (9), from a position on the boundary of an obstacle of radius R, is guaranteed to be outside the obstacle hole of any other obstacle.

C. Convergence

We can prove convergence only for zero terminal speeds. Even if the final state (position and velocity) is reachable from the initial state for the obstacle-free problem, it may not be reachable for the obstacle avoidance problem because of the reduction in speed caused by the need to pass through tight spaces between obstacles. Furthermore, the loops created when the terminal speed is too high or too low (at least one axis moves in the opposite direction to pick up, or reduce, speed) may not be feasible because of the presence of obstacles.

Theorem 3.2: Consider the initial and final points $\mathbf{x}_{p0}$ and $\mathbf{x}_{pf}$, which are the projections to the configuration space of the initial and final states $\mathbf{x}_0$ and $\mathbf{x}_f$, respectively. The trajectory generated by Algorithm 1 terminates at the goal $\mathbf{x}_{pf}$ in a finite time, for all initial points $\mathbf{x}_{p0}$, assuming that $\mathbf{x}_{pf}$ and $\mathbf{x}_{p0}$ are connected.

Proof: Algorithm 1 progresses recursively by moving to a sequence of intermediate goals. To prove convergence, it is sufficient to show that the path (the projection of the trajectory to the configuration space) generated by Algorithm 1 reaches the goal $\mathbf{x}_{pf}$ from any point $\mathbf{x}_{p0}$, and is of finite length.

With the generation of each intermediate goal, the algorithm subdivides the avoidance problem, from the current state $\mathbf{x}$ to the current goal $\mathbf{x}_g(i-1)$, into two smaller problems: 1) from the current state $\mathbf{x}$ to the intermediate goal $\mathbf{x}_g(i)$, and 2) from the intermediate goal $\mathbf{x}_g(i)$ to the current goal $\mathbf{x}_g(i-1)$. Both problems have a kinematic solution (path connecting the end points), since by assumption the obstacles are convex and non-overlapping. Each introduction of an intermediate goal subdivides the problem into smaller segments until the intermediate goal is reachable by an unconstrained trajectory. The size of the smallest segment is bounded by the longest time optimal trajectory between any two obstacles. The number of such segments is bounded by the number of obstacles, which is assumed to be finite. □

D. Optimality

The trajectory generated by Algorithm 1 is not necessarily optimal, since each step is only locally optimal. Anecdotal examples show that the paths (projection of the trajectory to the configuration pace) generated by Algorithm 1 are close to the global time optimal paths computed by a global search [13]. The motion time along the on-line trajectory is always higher than the global optimal motion time due to the curvature velocity (9) imposed at the intermediate goals.

E. Computational Issues

The consideration of the obstacle constraints one at a time reduces the original computationally exponential problem with m obstacles to m simple sub-problems with one obstacle each. That the original problem is exponential in the number of obstacles m becomes obvious if we simply count the number of potential local minima, which is $2^m, m \geq 1$ (each planar obstacle contributes 2 local minima).

The cost for this reduction is the loss of optimality, and the need to check at each time step if *all* obstacles intersect the unconstrained optimal path from the current state, and for those that do, solve the single obstacle problem. This may seem excessive, but the alternative (solving the original exponential problem) is much worse. Our approach generates the trajectory incrementally, unlike the original problem that requires a complete solution before making the first move. In fact, for problems with many obstacles (see the following examples with 70 obstacles), the on-line (heuristic) solution may be the only viable alternative. It is important to note that the computation at each time step is identical for each obstacle, and can hence benefit from parallel computing.

IV. Examples and Experiments

Algorithm 1 was implemented in MATLAB on an Intel core-i7 desktop computer. Its efficiency is demonstrated for a planar environment, consisting of 70, tightly spaced circular obstacles. In all experiments, the speeds at the intermediate goals are upper bounded by the curvature velocity (9).

A. Experiment 1

Algorithm was tested for 500 different start-goal configurations, selected randomly in the free space, as shown in Figure 3. Also shown are the initial and final points as red and blue dots, respectively. The time-step ΔT was set to $0.1s$. The algorithm succeeded in reaching the goal-state in all the 500 start-goal configurations. The average speed along the 500 trajectories was $2.3m/s$ with a standard deviation of $0.43 m/s$. Repeating this experiment without obstacles, using the same start-goal configurations as previously, and computing the trajectory between every pair analytically, resulted in an average speed along the 500 trajectories of $4.3 m/s$ with a standard deviation of $0.83 m/s$. Comparing the average speed of the on-line trajectories with and without obstacles suggests that, while the algorithm is not optimal, it produces fast (54% of the average obstacle free optimal time) trajectories in a very challenging environment.

B. Example 1

This example shows an on-line trajectory that avoids 70 obstacles, from the initial state $(x_1, x_2, y_1, y_2) = (10.46, 0.001, 58.26, -0.001)m$ to the target state $(x_{1f}, x_{2f}, y_{1f}, y_{2f}) = (52.55, 0, 7.33, 0)m$, as shown in Figure 4. The time step Δt in was $0.1s$. The motion time along the online trajectory, shown Figure 4 with black dots, is $35.2s$, with a top speed of $3.4 m/s$ and an average speed of $2.1 m/s$. The intermediate goals are shown as empty circles along the trajectory. There were 12 intermediate goals generated for this trajectory. The total computation time was $4.3s$, with an average computation time of $11ms$ per-step. The speed along the trajectory, as a function of distance traveled, is shown in Figure 5. The oscillation in the speed is produced due to the curvature velocity imposed at the intermediate goals.

C. Example 2

This example demonstrates the on-line planner in the same 70 obstacle environment, from the initial state $(x_1, x_2, y_1, y_2) = (55, 0.1, 0, 0)m$ to the target state $(x_{1f}, x_{2f}, y_{1f}, y_{2f}) = (0, 0, 60, 0.1)m$. The time step $\Delta t = 0.1s$. The on-line trajectory shown in Figure 6 was completed in $39.8s$, with a top speed of $4.8 m/s$ and an average speed of $2.2 m/s$. The total computation time was $3.8s$, with an average computation time of $8ms$ per step. The speed along the trajectory, as a function of distance traveled, is shown in Figure 7.

D. Example 3

This example compares the on-line motion planner with the global planner [12]. Figure 8 shows the trajectories generated by the on-line planner (black dots) and the global planner (red dots) from the start state $(x_1, x_2, y_1, y_2) = (51, 0, 6, 0)$ to the goal state $(x_{1f}, x_{2f}, y_{1f}, y_{2f}) = (30, 0, 91, 0)$. The on-line and globally optimal trajectories have similar topologies as they pass between the same obstacles. The velocity profiles along both trajectories are shown in Figure 9. The motion time along the on-line was $28.9s$ over a total distance of $93.8m$, with an average speed of $3.2 m/s$, compared to the global optimal motion time of $20.7s$ over a total distance traveled of $99m$, and an average speed of $4.8 m/s$. This is not a significant increase, considering that the global optimal trajectory is smoother, allowing higher speeds than the on-line trajectory, and that the speeds along the on-line trajectory is reduced by the curvature velocities (9) at the intermediate points.

An experiment with a real differential-drive robot, comparing the online and global optimal trajectories, for the avoidance of 5-obstacles can be seen in the video attached to this paper.

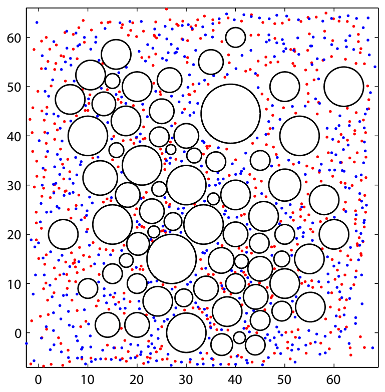
Fig. 3. The 500 random start-goal configurations for Experiment 1.

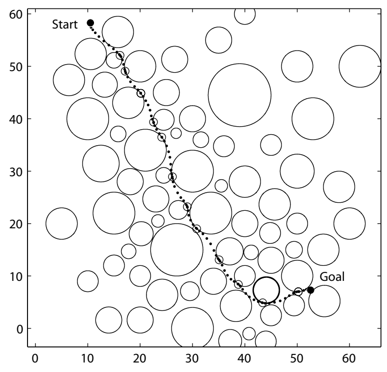

Fig. 4. Trajectory generated on-line in a tightly spaced environment with 70 circular obstacles for Example 1.

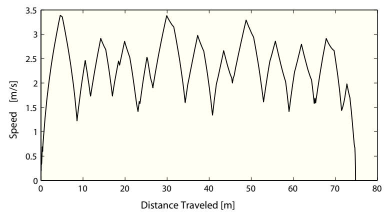

Fig. 5. Speed as a function of distance traveled along the online trajectory of Example 1.

V. Conclusions

An efficient motion planner for on-line near-time optimal obstacle avoidance, in cluttered environments, was presented. It avoids obstacles optimally one at a time, thus reducing the complex problem of time-optimal avoidance of m obstacles to m simpler problems, thereby reducing the computational complexity of the original problem from exponential to linear in the number of obstacles. This significant reduction in the computational effort and the incremental nature of this approach make it suitable for on-line applications, such as mobile robots moving amongst many obstacles. By following the intermediate goals, which break the navigation problem

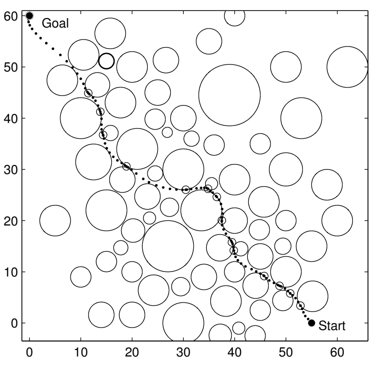

Fig. 6. Online trajectory avoiding 70 obstacles in Example 2.

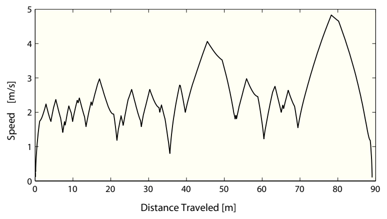

Fig. 7. Speed as a function of distance traveled during the online trajectory of Example 2.

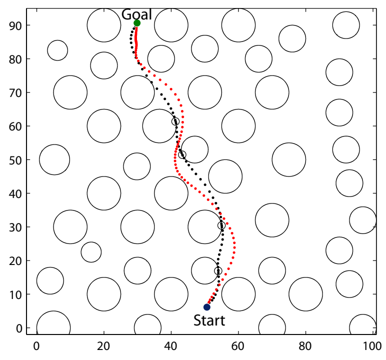

Fig. 8. The trajectories generated by global (red) and online (black) planners, among 48 obstacles, in Example 3.

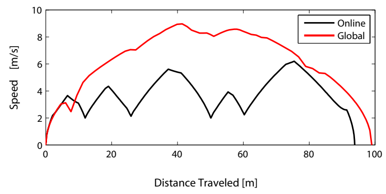

Fig. 9. Speed as a function of distance traveled for the online and global trajectories in Example 3.

into many simpler problems, the algorithm converges to the goal from any feasible state, despite the local increase in the cost produced by lowering the speed at the intermediate goals. The approach is applicable to any robot dynamics and to general obstacles. The implementation presented in this paper, however, has focused on a point mass model and planar obstacles. This simplification is necessary for computational reasons. Computation time of $11ms$ per-step were demonstrated for very challenging environments with 70 obstacles, suggesting a re-planning frequency of 90 Hz, which makes it suitable for planning in changing environments. This approach is applicable to known environments, or to environments that can be detected by global sensors, such as overhead cameras identifying all obstacles in a given space. With minor modifications, it can be used for sensor-based planning by considering only the obstacles within the visibility range of the on-board sensor. This would entail adding constraints on the robot speed so that it can stop at the boundary of its visibility range. Although this approach does not guarantee optimality, numerical examples demonstrate close correlation between the on-line solution and the global optimal trajectories.

VI. Acknowledgments

This work was motivated by the research conducted by Dr. Satish Sundar during his doctoral studies at the University of California Los Angeles. The generous support of Dr. David Paslin for the Paslin Laboratory for Robots and Autonomous Vehicles is gratefully acknowledged.

References

[1] M. Athans. *Optimal Control: An Introduction to the Theory and It's Applications*. Academic Press, New York, 1965.
[2] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators. *IJRR*, 4(3), 1985.
[3] A.E. Bryson and Y.C. Ho. *Applied Optimal Control*. Blaisdell Publishing Co., Cambridge, MA, 1969.
[4] S. Dreyfus. *Dynamic Programming and the Calculus of Variations*. Academic Press, New York, 1965.
[5] T. Fraichard and H. Asama. Inevitable collision states. a step towards safer robots? *Advanced Robotics*, 18(10), 2004.
[6] M.E Khan and B. Roth. The near-minimum time control of open loop articulated kinematic chains. *J. Dyn. Sys. Meas. Ctrl.*, 93(3):164–172, Sept. 1971.
[7] R.A. Knepper and M.T. Mason. Path diversity is only part of the problem. In *Internationa Conference on Robotics and Automation*, 2009.
[8] Y. Kuwata, G.A. Fiore, J. Teo, E. Frazzoli, and J.P. How. Motion planning for urban driving using RRT. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1681–1686, 2008.
[9] Fiorini P. and Shiller Z. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17:760–772, 1998.
[10] T. Schouwenaars, J. How, and E. Feron. Receding horizon path planning with implicit safety guarantees. In *American Control Conf.*, pages 5576 – 5581, 2004.
[11] Z. Shiller and S. Dubowsky. Time-optimal path-planning for robotic manipulators with obstacles, actuator, gripper and payload constraints. *IJRR*, 8(6):3–18, Dec. 1989.
[12] Z. Shiller and S. Dubowsky. On computing the global time optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, 7(6):785–797, Dec. 1991.
[13] Z. Shiller and Y.R. Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotocis and Automation.*, 7(2):241–249, April 1991.
[14] Z. Shiller and H.H. Lu. Computation of path constrained time-optimal motions with dynamic singularities. *ASME Jnl. Dyn. Sys. Meas. Ctrl.*, 114(1):34–40, March 1992.
[15] K.G. Shin and N.D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Trans. Aut. Ctrl.*, AC-30(6):531–541, June 1985.
[16] 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.