Proceedings of the 1996 IEEE
International Conference on Robotics and Automation
Minneapolis, Minnesota - April 1996

# Time Optimal Trajectory Planning in Dynamic Environments

**Paolo Fiorini**

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

**Zvi Shiller**

Department of Mechanical, Aerospace
and Nuclear Engineering
University of California, Los Angeles
Los Angeles, California 90095

## Abstract

*This paper presents a method for motion planning in dynamic environments, subject to robot dynamics and actuator constraints. The time optimal trajectory is computed by first generating an initial guess using the concept of velocity obstacle. The initial guess, computed by a global search over a tree of avoidance maneuvers, is then optimized using a dynamic optimization. This method is applicable to repetitive tasks in known dynamic environments, as is demonstrated for a planar robot manipulator.*

## 1 Introduction

This paper addresses the problem of motion planning in dynamic environments. Typical examples of dynamic environments include manufacturing tasks in which robot manipulators track and retrieve parts from moving conveyers.

Motion planning in dynamic environments requires the simultaneous computation of a collision free path from start to goal, and of the velocity profile along the path, satisfying system dynamics and actuator constraints.

Previous methods for motion planning in dynamic environments consist of a graph search in the position-time configuration space [5], and in the Cartesian-time space [3]. These methods assume no limits on the robot's velocity and acceleration, producing trajectories that might be dynamically infeasible. Velocity constraints are considered in [13], and acceleration constrains are satisfied in [11] for planning for a point mass. Approximate dynamic constraints are satisfied in [9]. The collision front is used in [8] to compute trajectories satisfying approximate constraints on the robot acceleration. None of the previous methods considered the non-linear robot dynamics, and none produced time optimal motions.

The time-optimal motion planning problem in static environments has been treated previously using parameter optimization in [1, 15, 16].

In this paper, we develop a method for computing the time optimal trajectories of a robot moving in a time-varying environment. We assume circular obstacles traveling along known trajectories, and a Scara robot.

Central to this method is the computation of the initial guess for the optimization, that would closely approximate the time optimal trajectory. The method utilizes the concept of Velocity Obstacle (VO), representing the robot's velocities that would cause a collision with any obstacle at some future time. Feasible avoidance maneuvers are computed by selecting velocities outside the velocity obstacle, and satisfying additional velocity constraints computed from robot dynamics and actuator constraints.

An initial guess of the optimal trajectory is computed by a global search over a tree of feasible avoidance maneuvers at specified time intervals so as to minimize time and reach to goal. This trajectory is then used to compute the initial guess of the controls for the dynamic optimization. The optimal trajectory is computed using a steepest descent algorithm over the admissible controls [4]. This optimization is applicable to any robot dynamics. However, the velocity obstacle is most suitable for circular robots and obstacles.

## 2 The Velocity Obstacle

The **Velocity Obstacle** (VO) consists of the velocities of the robot that would cause a collision between

0-7803-2988-4/96 $4.00 

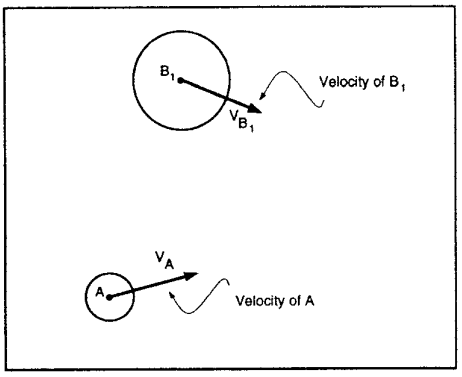


Figure 1: The robot and a moving obstacle.

the robot and an obstacle at some future time.

The VO is illustrated using the scenario shown in Figure 1, where two circular objects, $A$ and $B_1$, are shown at time $t_0$ with velocities $\mathbf{v}_A$ and $\mathbf{v}_{B_1}$. Circle $A$ represents the robot, and circle $B_1$ represents the obstacle. To compute the VO, we first represent $B_1$ in the *Configuration Space* of $A$, and then attach the velocity vectors to the position of $\widehat{A}$ and to the center of $\widehat{B_1}$, respectively.

By considering the relative velocity $\mathbf{v}_{A,B_1} = \mathbf{v}_A - \mathbf{v}_{B_1}$, and by assuming that $A$ and $B_1$ maintain their current velocities, a collision between $\widehat{A}$ and $\widehat{B_1}$ will occur at some future time $t_1 > t_0$ if the line $\lambda_{A,B_1}$ of the relative velocity $\mathbf{v}_{A,B_1}$ intersects $\widehat{B_1}$. We define the *Collision Cone*, $CC_{A,B_1}$, as the set of colliding relative velocities between $\widehat{A}$ and $\widehat{B_1}$ or:

$$CC_{A,B_1} = \{\mathbf{v}_{A,B_1} \mid \lambda_{A,B_1} \cap \widehat{B_1} \neq \emptyset\} \quad (1)$$

This cone is the planar sector with apex in $\widehat{A}$, bounded by the two tangents $\lambda_f$ and $\lambda_r$ from $\widehat{A}$ to $\widehat{B_1}$, as shown in Figure 2.

The collision cone is specific to a particular pair of robot/obstacle. To consider multiple obstacles, it is useful to establish an equivalent partition of the *absolute* velocities of $A$. This is done by adding the velocity of $B_1$, $\mathbf{v}_{B_1}$, to the set $CC_{A,B_1}$, as shown in Figure 3 [7]. The *Velocity Obstacle* $VO$ is then defined as:

$$VO = CC_{A,B_1} \oplus \mathbf{v}_{B_1} \quad (2)$$

where $\oplus$ is the Minkowski vector sum operator.

Thus, the VO partitions the absolute velocities of $A$ into *avoiding* and *colliding* velocities. Velocities on the boundaries of $VO$ would result in $A$ grazing $B_1$, since the corresponding relative velocities lie on the boundary of the collision cone $CC_{A,B_1}$. Note that the

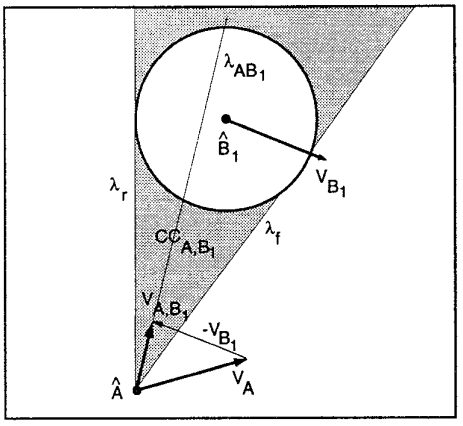


Figure 2: The Collision Cone $CC_{A,B_1}$.

VO of a stationary obstacle is identical to its relative velocity cone, since then $\mathbf{v}_{B_1} = 0$.

To avoid multiple obstacles, the $VO$of all obstacles are combined into a single velocity obstacle:

$$VO = \cup_{i=1}^{m} VO_i \quad (3)$$

where $m$ is the number of obstacles. Since the VO assumes a linear extrapolation of the current velocities of the obstacles, it is periodically computed to update the current positions and velocities of the obstacles.

## 3 The Avoidance Maneuvers

The velocities reachable by robot $A$ at a given state over a given time interval $\Delta t$ are computed by transforming the dynamic constraints of the robot into bounds on its acceleration. The set of *feasible accelerations* at time $t_0$, $FA(t_0)$, is defined as

$$FA(t_0) = \{\ddot{\mathbf{x}} \,|\, \ddot{\mathbf{x}} = f(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}), \mathbf{u} \in U\} \quad (4)$$

where $f(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u})$ represents the robot dynamics, $\mathbf{u}$ are the actuator efforts, U is the set of admissible controls, and $\mathbf{x}$ is the position vector defined earlier. Note that the feasible acceleration range of a two degree-of-freedom system with decoupled actuator limits is a parallelogram [14].

The set of *reachable* velocities, $\mathrm{RV}(t_0 + \Delta t)$, over the time interval $\Delta t$, is thus defined as:

$$\mathrm{RV}(t + \Delta t) = \{\mathbf{v} \,|\, \mathbf{v} = \mathbf{v}_A(t_0) \oplus \Delta t \cdot FA(t_0)\} \quad (5)$$

The set of *reachable avoidance* velocities, RAV, is defined as the difference between the reachable veloc-

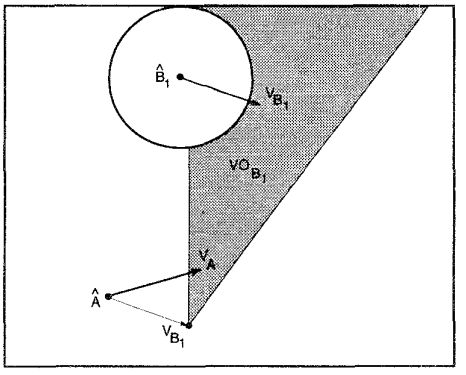


Figure 3: The velocity obstacle $VO$.

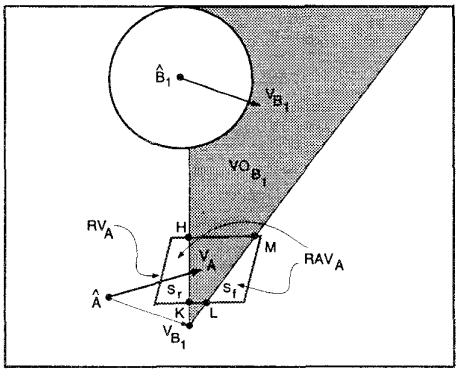


Figure 4: The reachable avoidance velocities RAV.

ities and the velocity obstacle:

$$\mathrm{RAV}(t_0 + \Delta t) = \mathrm{RV}(t_0 + \Delta t) \ominus VO(t_0) \qquad (6)$$

where $\ominus$ denotes the operation of set difference. Any velocity in RAV represents a maneuver avoiding obstacle $B_1$.

Figure 4 shows schematically the reachable velocity set RAV consisting of two disjoint closed sets, denoted $S_f$ and $S_r$. For multiple obstacles, the RAV may consist of multiple disjoint subsets.

It is important to note that the velocity obstacle accounts for all future collisions, and as a result, it excludes maneuvers that are on a collision course with any obstacle at some future time. This, in turn, may result in RAV being empty. However, this can be avoided by considering only the obstacles with the most imminent collision time.

## 4 Generating the Initial Guess

Generating the initial guess for the optimization involves the computation of a feasible trajectory to the goal, using a tree search, and the computation of the corresponding controls. For on-line applications the global search can be replaced by a heuristic search and the optimization step can be omitted [6].

### 4.1 The Tree Search

The state space of the robot is represented by a tree of avoidance maneuvers at discrete time intervals. The *nodes* on this tree correspond to the positions of the robot at discrete times $t_i$. The *operators* expanding a node at time $t_i$ into its successors at time $t_{i+1} = t_i + T$ are the velocities in the reachable avoidance velocity set RAV. The *edges* correspond to the avoidance maneuvers at those positions [12]. The search tree is then defined as follows:

$$\begin{aligned} n_{i,j} &= \{\mathbf{x}_{i,j} = (x_j(t_i), y_j(t_i)), \\ &\quad \mathbf{v}_{i,j} = (v_{j,x}(t_i), v_{j,y}(t_i))\} \\ o_{i,j,l} &= \{\mathbf{v}_l \mid \mathbf{v}_l \in \mathrm{RAV}_j(t_i)\} \\ e_{j,k} &= \{(n_{i,j}, n_{i+1,k}) \mid n_{i+1,k} = n_{i,j} + (o_{i,j,l}\,\Delta t)\} \end{aligned} \qquad (7)$$

where $n_{i,j}$ if the $j$th node at time $t_i$, $\mathrm{RAV}_j(t_i)$ is the reachable velocity set computed for node $n_{i,j}$, $o_{i,j,l}$ is the $l$th operator on node $j$ at time $t_i$, and $e_{j,k}$ is the edge between node $n_j$ at time $t_i$, and node $n_k$ at time $t_{i+1}$. The trajectory is computed using a Depth First Iterative Deepening Search method [10] which efficiently generates the time-minimum solution.

### 4.2 Generating the Controls

The trajectory computed by the tree search consists of a sequence of straight line segments and velocities assigned to each segment. This trajectory might

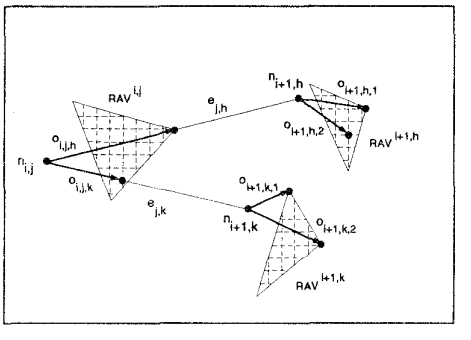


Figure 5: Tree representation for the global search.

not be dynamically feasible because of the discontinuities of the velocity at the tree nodes. The trajectory must then be smoothed by a spline interpolation. This replaces the velocity discontinuities by a polynomial blend, chosen as a third order Hermite spline. The control variables associated with this trajectory are computed using inverse dynamics. These controls are then approximated by bang-bang controls as discussed next.

## 5 The Dynamic Optimization

The trajectory is optimized for off-line applications by solving the following optimization problem:

$$\min_{u(t)\in\mathbf{U}} J = \phi_{min}(\mathbf{x}(t_f), t_f) = t_{f,min} \tag{8}$$

subject to robot dynamics

$$\dot{\mathbf{x}} = \mathcal{F}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \tag{9}$$

admissible controls

$$\mathbf{U} = \{\mathbf{u} \mid u_i(min) \le u_i \le u_i(max)\} \tag{10}$$

initial manifold

$$\Gamma(\mathbf{x}(t_0), t_0) = 0 \tag{11}$$

terminal manifold

$$\Omega(\mathbf{x}(t_f), t_f) = 0 \tag{12}$$

and time-varying obstacles

$$\Psi : \bigcup_{i=1}^{n} [S_i(\mathbf{x}(t), t) = 0] \tag{13}$$

State constraints due to the obstacles are treated by differentiating (13) with respect to time, until they become explicit in the controls $\mathbf{u}$, and then appended as state dependent control constraint to the Hamiltonian [2], [4].

In the case of a single obstacle, the state constraint (13) is replaced by the tangency condition, denoted $\Psi_1$, and the control equality constraint, denoted $\Psi_2$:

$$\Psi_1 : \begin{pmatrix} S(\mathbf{x}(t), t) = 0 \\ \dot{S}(\mathbf{x}(t), t) = 0 \\ \vdots \\ S^{(p-1)}(\mathbf{x}(t), t) = 0 \end{pmatrix} \quad t = t_1 \tag{14}$$

$$\Psi_2 : \quad S^{(p)}(\mathbf{x}(t), u(t), t) = 0 \quad t_1 \le t \le t_2 \tag{15}$$

This transforms the admissible control set $\mathcal{U}$ for the optimal control $u^*(t)$ to:

$$\mathcal{U} : \begin{cases} \mathbf{U} : \mathbf{u}_{min}(\mathbf{x}) \le \mathbf{u} \le \mathbf{u}_{max}(\mathbf{x}) \\ S^{(p)}(\mathbf{x}(t), u(t), t) = 0 \quad \text{for} \quad S(\mathbf{x}, t) = 0 \end{cases} \tag{16}$$

The optimal control $\mathbf{u}^*(t)$ is computed by satisfying the necessary optimality conditions stated by Pontryagin's Minimum Principle [2].

The optimal controls minimize the Hamiltonian $\mathcal{H}$, defined as [2]:

$$\mathcal{H}(\Lambda, \mathbf{x}, \mathbf{u}) = \Lambda^T(\mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}) + \mu^T\varphi(\mathbf{x}, \mathbf{u}) \tag{17}$$

where $\Lambda$ and $\mu$ are vectors of Lagrange multipliers, and $\varphi(\mathbf{x}, \mathbf{u})$ is the set of active control constraints at time $t$, $t_0 \le t \le t_f$.

The adjoint equations for the Lagrange multipliers are:

$$\dot{\Lambda} = -\left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right)^T \qquad \Lambda^T(t_f) = \left(\frac{\partial \phi}{\partial x} + \nu^T \frac{\partial \Omega}{\partial x}\right)_{t_f} \tag{18}$$

with a discontinuity at the entry point of the constrained arc given by [4]:

$$\Lambda^T_{t_1^-} = \Lambda^T_{t_1^+} + \eta^T \frac{\partial \Psi_1}{\partial x(t_1)} \tag{19}$$

where $\eta$ and $\nu$ are constant multipliers.

The optimal controls along the constrained arc are computed by satisfying:

$$\mathcal{H}^*_u(\Lambda(t), \mathbf{x}^*(t), \mathbf{u}^*(t)) = 0 \quad t_0 \le t \le t_f \tag{20}$$

and the adjoint equations on the constrained arcs are:

$$\begin{aligned} \dot{\Lambda}^T \;=\; & -\Lambda^T \left[\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} + \frac{\partial \mathbf{g}(\mathbf{x})\mathbf{u}}{\partial \mathbf{x}} - \right. \\ & \left. \mathbf{g}(\mathbf{x}) \left(\frac{\partial \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right)^{-1} \frac{\partial \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right] \end{aligned} \tag{21}$$

The numerical algorithm assumes bang-bang controls so that the adjustments of the control variables become variations on the switching times [2]. Bang-bang controls transform the differential of the performance index

$$\begin{aligned} d\widehat{J} \;=\; & \left(\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \mathcal{H}\right)_{t_f} dt_f \\ & + \int_{t_0}^{t_1^-} \mathcal{H}_u \delta u d\tau + \int_{t_1^+}^{t_f} \mathcal{H}_u \delta u d\tau \end{aligned} \tag{22}$$

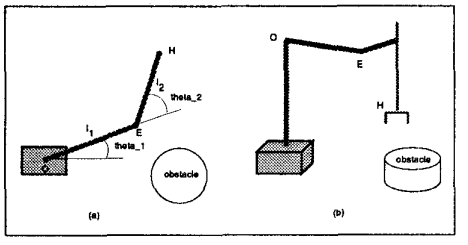


Figure 6: Planar 2-dof Manipulator: a) top view, b) side view

into an explicit function of the variations of the switching times $dt_{ij}$:

$$d\hat{J} = \sum_{i=1}^{m}\sum_{j=1}^{q_1}\mathcal{H}_{u_i}\delta u_i dt_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{q_2}\mathcal{H}_{u_i}\delta u_i dt_{ij} \quad (23)$$
$$+ \sum_{i=1}^{m}\sum_{j=1}^{q_3}\mathcal{H}_{u_i}\delta u_i dt_{ij} + \left(\frac{\partial\phi}{\partial t} + \nu^T\frac{\partial\Omega}{\partial t} + \mathcal{H}\right)dt_f$$

where $\delta u_i = (\alpha_M - \alpha_m)\, sgn(dt_i)$, $sgn$ is the signum function, and $\alpha_M$ and $\alpha_m$ are the upper and lower values of the controls, $m$ indicates the number of independent controls, and $q_i$ indicates the number of switches in each segment of the trajectory, i.e. before, on, and after the state constraint.

Differential (23) is minimized by choosing the steepest descent increments as [2]:

$$dt_{ij} = -\frac{(\mathcal{H}_{u_i})_{t_{ij}}}{w_{ii}\delta u_i} \quad (24)$$

$$dt_f = -\frac{1}{b}\left(\mathcal{H} + \frac{\partial\phi}{\partial t} + \nu^T\frac{\partial\Omega}{\partial t}\right)_{t_f} \quad (25)$$

with $w_{ii}$ and $b$ being suitable positive values.

The values of $dt_{ij}$ and $dt_f$ depend on multipliers $\eta$ and $\nu$, that are computed by requiring that differentials $d\Psi(t_1)$ and $d\Omega(t_f)$ be reduced by given quantities $\epsilon$ and $\theta$ at each iteration. The optimal controls are computed by integrating the state equations forward in time, and the co-state equations backward in time. The variations of the terminal states are used to compute the initial conditions of the co-states for the backward integration. The gradient of the Hamiltonian is then used to adjust the controls. This process is terminated when the boundary conditions on the states fall within specified bounds.

## 6 Example

In this example, the optimal motion planning method is demonstrated for computing the collision free trajectory of the end effector of the SCARA robot shown schematically in Figure 6.

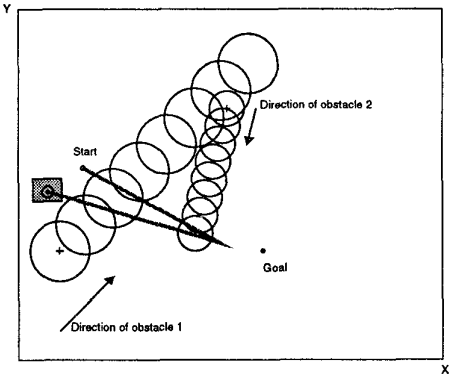


Figure 7: Planning example.

The environment of this example consists of two obstacles moving with known velocities, as shown in Figure 7. Obstacle 1 starts from position $P = (.1, -.5)\ m$, with velocity $v = (.045, .045)\ m/s$, and obstacle 2 starts from $Q = (1.15, .7)\ m$ with velocity $w = (-.007, -.03)\ m/s$. At the initial time, the robot end-effector is at rest at $S = (.3, .2)\ m$, and the goal is point $G = (1.5, -.5)\ m$. In Figure 7 the obstacles are displayed at .5 $s$ intervals.

The optimal trajectory is computed by first building the tree of avoidance maneuvers satisfying the velocity obstacles at intervals of $T = 1\ s$. Searching over the tree yields the time optimal trajectory avoiding the obstacles and reaching goal $G$, with a motion time of 4.81 $s$.

This trajectory is the initial guess used by the dynamic optimization. The solution computed by the optimization is shown in Figure 8. Its motion time

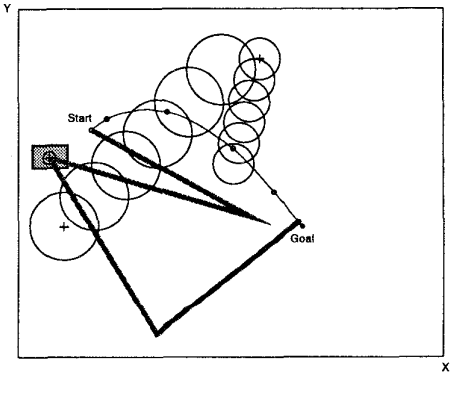


Figure 8: Optimal trajectory.

is 2.6 $s$. The robot end-effector is represented by the small circles on the trajectory. The end-effector and the obstacles are displayed at .5 $s$ intervals.

This trajectory is the initial guess used by the dynamic optimization. This trajectory has the same structure as the initial guess, i.e. it avoids both obstacles by crossing their paths in front of them, but it has a much lower motion time. This improvement in motion time is due to the relaxation of the velocity obstacle constraint, that allows the use of colliding velocities for small segments of the trajectory. Since obstacle collisions are checked explicitly by the dynamic optimization, the colliding velocities are safely included into the trajectory and achieve the faster motion time.

## 7 Conclusion

A method for optimal planning the motion of a robot in time-varying environments has been presented. It computes simultaneously the path and velocity profile that avoid static and moving obstacles and satisfies the robot's dynamic constraints. The obstacles are assumed to be circular, moving along general trajectories.

The method first computes a feasible trajectory using a global search over a tree of avoidance maneuvers. The avoidance maneuvers are computed by selecting velocities outside the velocity obstacle, and within the reachable velocity set. This trajectory is then used as the initial guess for a dynamic optimization, that minimizes motion time to the goal satisfying robot dynamics and actuator constraints. The method is demonstrated for a Scara robot.

The initial trajectory, computed by the global search, is conservative since it consists of maneuvers that include velocities that are never on a collision course with any moving obstacles. The dynamic optimization relaxes this constraint, and therefore reduces the motion time as it was demonstrated in the example. It is also possible to generate less conservative initial trajectories by considering only obstacles with a collision time within a specified time horizon. This method is applicable to Scara manipulators operating on moving conveyer belts.

## 8 Acknowledgment

The research described in this paper was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

## References

[1] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, Fall 1985.

[2] A.E. Bryson and Y.C. Ho. *Applied Optimal Control.* Hemisphere Publishing Corp., New York, NY, 1975.

[3] S.A. Cameron. *Modelling Solids in Motion.* PhD thesis, Edinburgh, UK, 1984.

[4] W.F. Denham and A.E. Bryson. Optimal programming problems with inequality constraints *ii*: Solution by steepest ascent. *AIAA Journal*, 2(2):25–34, January 1964.

[5] M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, (2):477–521, 1987.

[6] P. Fiorini. *Robot Motion Planning among Moving Obstacles.* PhD thesis, University of California, Los Angeles, January 1995.

[7] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using the relative velocity paradigm. In *IEEE ICRA*, volume 1, pages 560–566, May 1993.

[8] K. Fujimura and H. Samet. Motion planning in a dynamic domain. In *IEEE ICRA*, pages 324–330, Cincinnati, OH, May 1990.

[9] K. Kant and S.W. Zucker. Towards efficient trajectory planning: the path-velocity decomposition. *IJRR*, 5(3):72–89, Fall 1986.

[10] R. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *AI*, (27):97–109, 1985.

[11] C. Ó'Dúnlaing. Motion planning with inertial constraints. Technical Report TR-230, NYU Robotics Laboratory, 1986.

[12] J. Pearl. *Heuristics.* Addison Wesley Publishing, Co., Reading, MA, 1985.

[13] J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *25th IEEE Symp. Found. of CS*, pages 144–153, 1985.

[14] Z. Shiller and Dubowsky. Time optimal paths and acceleration lines of robotic manipulators. In *The 26th IEEE CDC*, Los Angeles, CA, December 1987.

[15] Z. Shiller and S. Dubowsky. Robot path planner with obstacles, actuators, gripper and payload constraints. *IJRR*, 8(6):3–18, December 1989.

[16] K.G. Shin and N.D. McKay. Open loop minimum time control of mechanical manipulators. In *ACC*, pages 1231–1236, June 1984.