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, 1.0s Angeles
Los Angeles, California 90024

Abstract

This paper presents a method for planning the motion of a robot in a dynamic environment by computing a trajectory that avoids all obstacles and that satisfies the robot dynamics and its actuator constraints. This method consists of two steps: the computation of the trajectory and its refinement with a dynamic optimization.

The trajectory consists of a sequence of avoidance maneuvers that are computed at discrete time intervals using the concept of Velocity Obstacle, i.e. selecting those velocities that avoid future collisions and satisfy the dynamic constraints. This trajectory is determined by a global search over a tree of avoidance maneuvers, computed at discrete time intervals to satisfy the constraints represented by the velocity obstacles and the robot dynamics. This trajectory is then smoothed and further refined using a dynamic optimization that accounts for the robot dynamics, actuator (wrist raints and the static and moving obstacles.

The dynamic optimization uses the smoothed trajectory as its initial guess, eliminates the effects of the discretization, and minimizes motion time, subject to the dynamics of the robot, its actuator limits and the state inequality constraints due to the moving obstacles. The optimization is based on a steepest descent method in which the state inequality constraints are transformed into state-dependent control constraints.

This approach is demonstrated for planning the trajectory of a simple 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, and air, sea, and land traffic, where aircraft, vessels and vehicles avoid each other while moving towards their destination.

Motion planning in dynamic environments is considerably more difficult than the widely studied static problem, since it requires the simultaneous solution of the path planning and of the velocity planning proi)-1(111s. Path planning involves the computation Of a collision free path from start to goal without considering robot dynamics. Velocity planning, on the other hand, involves the computation of the velocity profile along a given path, satisfying system dynamics and actuator constraints.

Previous methods consisted of a graph search in a position-time configuration space of the robot [7], and checking for intersection of the swept volumes of the robot and obstacles in a Cart(wian-time space [4]. Velocity constraints are considered in [25] for the solution of the asteroid problem, and acceleration constrains are satisfied in [5] and in [23] for planning for a point Ills.ss.

Approximate dynamic constraints are satisfied in [15] by decomposing the planning problem into path and velocity planning, and by limiting slope, direction and curvature of the trajectory in a space-time plane. Dynamic constraints are used in [12] to define the collision front, and in [11] to introduce the concept of rail.s)t,nl obstacles, also applicable to dynamic planning.

The time-optimal motion planning problem in static environments has been treated previously using parameter optimizations, representing the trajectory as a polynomial in time and accounting for obstacles using a penalty function [14], and representing the path by a cubic spline [28], [1], computing the motion time along the path with an efficient method developed in [1] and in [29]. Time optimal motion planning

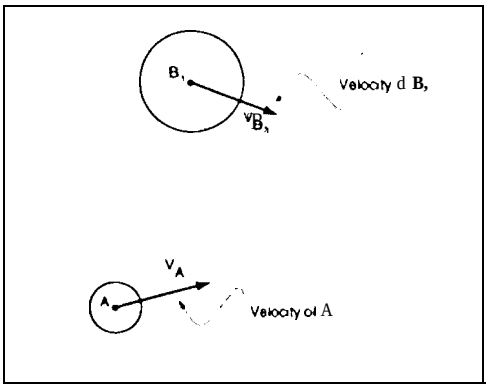

Figure 1: The robot and a moving obstacle.

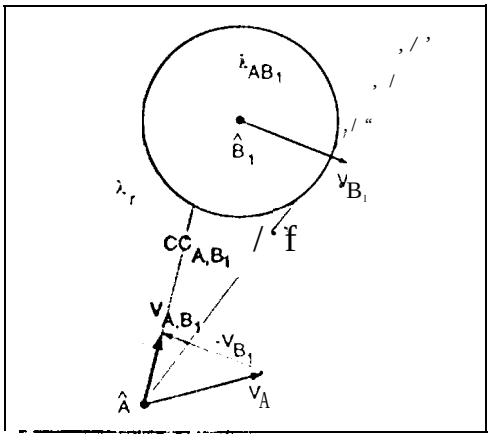

Figure 2: The Collision Cone CC_{A,B_1}.

of cooperating robots moving along specified paths has been studied in [17], [29].

In this paper, we develop an efficient method to compute the dynamically feasible trajectories of a robot moving in a time-varying environment. It utilizes the concept of Velocity Obstacle (VO), which represents the robot's velocities that would cause a collision with an obstacle at some future time. Thus, an avoidance maneuver is computed by selecting velocities that are outside of the velocity obstacle. To ensure that the maneuver is dynamically feasible, robot dynamics and actuator constraints are mapped into the robot velocity space. A trajectory consists of a sequence of such avoidance maneuvers. These trajectories are then refined by using a dynamic optimization. The method that we use is a steepest descent [2], [6], modified to include the state-dependent inequality constraints due to the moving obstacles. The initial guess of the optimization is the approximate trajectory generated by the global search over the maneuver tree.

This paper is organized as follows. Section 2 defines the Velocity Obstacle, the avoidance velocities, and describes the representation used to compute the trajectories. Section 3 describes the dynamic optimization used to refine the trajectories. An examples of a trajectory and its time optimal refinement is presented in Section 4.

2 The Velocity Obstacle

The **Velocity Obstacle** (VO) is an extension of the Configuration Space Obstacle [20] to a time-varying environment. It consists of the velocities of the robot that will cause a collision between the robot and the obstacles at some future time. Although this concept is valid for general robots and obstacles, in this paper we restrict our analysis to circular robots and obstacles in the plane.

In this section we first define the VO concept and then we combine it with the dynamic constraints of the robot to compute avoidance velocities that are also dynamically feasible. The corresponding avoidance maneuvers are used to build a tree that represents, within the given temporal resolution, all the avoidance trajectories generated to satisfy the constraints represented by the velocity obstacles.

2.1 Definition of Velocity Obstacle

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. The velocities and positions of A and B_1 were chosen so that A and B_1 will collide at some time t_1, ($t_1 > t_0$), provided that $\mathbf{v}_A$ and $\mathbf{v}_{B_1}$ do not change.

To compute the VO, we first represent B_1 in the *Configuration Space* of A by reducing A to the point $\hat{A}$, and enlarging B_1 by the radius of A to the circle $\widehat{B_1}$ [19], and then we attach the velocity vectors to the position of $\hat{A}$ 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 $\hat{A}$ and $\widehat{B_1}$ will occur at some future time $t_1 > t_0$ if the line λ_{A,B_1} of the relative velocity $\mathbf{v}_{A,B_1}$ intersects $\widehat{B_1}$. In fact, any relative velocity that lies between the two tangents to

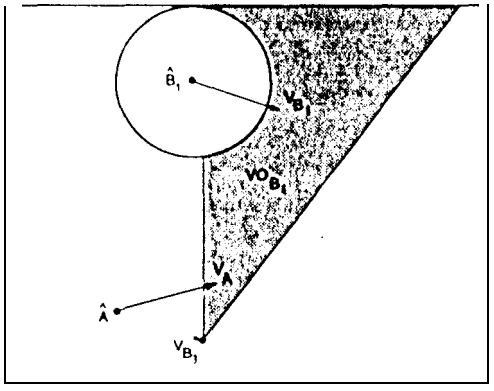

Figure 3: The velocity obstacle VO.

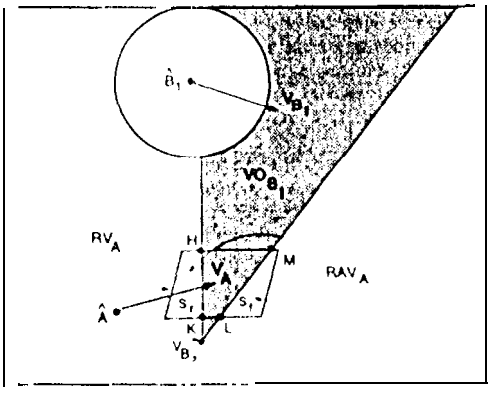

Figure 4: The reachable avoidance velocities RAV.

$\widehat{B_1}$ λ_f and λ_r will cause a collision between A and B_1. Therefore, we define the *Collision Cone*, CC_{A,B_1}, as the set of colliding relative velocities between $\widehat{A}$ and $\widehat{B_1}$ or:

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

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

The collision cone thus partitions the space of *relative* velocities into colliding and avoiding velocities. The relative velocities, $\mathbf{v}_{A,B_1}$, lying on the boundaries of CC_{A,B_1} represent *tangent* maneuvers that would graze the obstacle B_1.

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 simply by adding the velocity of B_1, $\mathbf{v}_{B_1}$, to each velocity in CC_{A,B_1} or, equivalently, by translating the collision cone CC_{A,B_1} by $\mathbf{v}_{B_1}$, as shown in Figure 3 [9]. The *Velocity Obstacle* VO is then defined as:

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

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 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's of each obstacle are combined into a single velocity obstacle:

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

where m is the number of obstacles. The VO assumes that the velocity of B_1 remains constant. To account for variable velocities, VO is recomputed at specified time intervals.

2.2 The Avoidance Maneuvers

The velocities reachable by robot A at a given state over a given time interval Δt 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:

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

where $f(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u})$ represents the dynamics of the robot, $\mathbf{u}$ 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 [27].

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

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

The set of *reachable avoidance* velocities, RAV, is defined as the difference between the reachable velocities 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. A maneuver avoiding obstacle B_1 is thus computed by selecting any velocity in RAV. Figure 4 shows schematically the reachable velocity set RAV consisting of two disjoint closed sets, S_f and S_r. For multiple obstacles, the RAV may consist of multiple disjoint subsets.

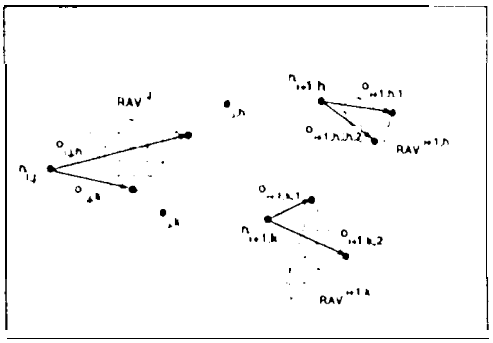

Figure 5: Tree representation for the global search.

2.3 The Avoidance Trajectories

The trajectory is computed as a discrete sequence of avoidance maneuvers, selected by a global search over the tree of all feasible maneuvers at specified time intervals.

We represent the state space of the robot 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 [24, 22]. The search tree is then defined as follows:

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

where $n_{i,j}$ if the jth 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 lth 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}.

Since the avoidance maneuvers are selected based on the velocity obstacles VO, they are never on a collision course with any of the considered obstacles. This excludes trajectories that might, part of the time, be on a collision course with some of the obstacles. However, such trajectories may be generated by either considering only obstacles with imminent collisions, or by refining the trajectory using a dynamic optimization, as discussed next.

3 The Dynamic Optimization

The refinement of the approximate trajectory for off-line applications is carried out by formulating the planning problem as a dynamic optimization, using Mayer notation for the performance index J:

$$\min_{u(t)\in \mathbf{U}} J \;, \quad \text{with} \quad J = t_f = \phi(\mathbf{x}(t_f), t_f) \tag{7}$$

subject to the following kinematic and dynamic constraints:

$$
\begin{aligned}
\Gamma(\mathbf{x}(t_0), t_0) &= 0 && (8)\\
\Omega(\mathbf{x}(t_f), t_f) &= 0 && (9)\\
\Psi &: \bigcup_{i=1}^{n} [S_i(\mathbf{x}(t), t) = 0] && (10)\\
\dot{\mathbf{x}} &= \mathcal{F}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} && (11)\\
\mathbf{U} &= \{\mathbf{u} \mid u_i(min) \le u_i \le u_i(max)\} && (12)
\end{aligned}
$$

where (8) is the initial manifold, (9) is the terminal manifold, (10) represents the time-varying obstacles, (11) is the robot dynamics, and (12) represents the admissible controls.

State constraints due to the presence of obstacles (10) are differentiated with respect to time until they become explicit in the controls u, and then appended as state dependent control constraint to the Hamiltonian [3], [6]. The number of differentiations of each constraint represents the *order* p of that constraint. This approach requires an additional tangency constraint at the entry point of the obstacle [30]. This may over-constrain the problem for constraints of order higher than two [13].

For example, in the case of a single obstacle, the state constraint (10) is replaced by the tangency condition, denoted Ψ_1, and the control equality constraint, denoted Ψ_2:

$$\Psi_1 : \begin{pmatrix} S(\mathbf{x}(t), t) = 0 \\ \dot{S}(\mathbf{x}(t), t) = 0 \\ \vdots \\ S^{(p-1)}(\mathbf{x}(t), t) = 0 \end{pmatrix} \quad t = t_1 \tag{13}$$

$$\Psi_2 : S^{(p)}(\mathbf{x}(t), u(t), t) = 0 \quad t_1 \le t \le t_2 \tag{14}$$

Then, the admissible control set $\mathcal{U}$ for the optimal control $u^*(t)$ becomes:

$$\mathcal{U} : \begin{cases} \mathbf{U} : \mathbf{u}_{min}(\mathbf{x}) \le \mathbf{u} \le \mathbf{u}_{max}(\mathbf{x}) \\ S^{(p)}(\mathbf{x}(t), u(t), t) = 0 \quad \text{for} \quad S(\mathbf{x}, t) = 0 \end{cases} \tag{15}$$

3.1 The Necessary Optimality Conditions

The optimal control $\mathbf{u}^*(t)$ in the interval $t_0 \le t \le t_f$, (hat generates the optimal solution, x-(f). minimizes J. and satisfies the fixed terminal manifolds Γ and Ω, is computed by satisfying the necessary conditions of Pontryagin Minimum Principle [3].

The Hamiltonian function $\mathcal{H}$ for this problem is defined as [3] [1 S]:

$$\mathcal{H}(\Lambda, \mathbf{x}, \mathbf{u}) = \Lambda^T(\mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}) + \mu^T\varphi(\mathbf{x}, \mathbf{u}) \quad (16)$$

where Λ and μ 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 given by:

$$\dot{\Lambda} = -\left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right)^T \qquad \Lambda^T(t_f) = \left(\frac{\partial \phi}{\partial x} + \nu^T \frac{\partial \Omega}{\partial x}\right)_{t_f} \quad (17)$$

which exhibit the discontinuity at the entry point of the constraint given by [6]:

$$\Lambda^T_{t_1^-} = \Lambda^T_{t_1^+} + \eta^T \frac{\partial \Psi_1}{\partial x(t_1)} \quad (18)$$

When the constraint (14) is active, the minimization of the Hamiltonian for an autonomous system is equivalent to satisfying:

$$\mathcal{H}^*_u(\Lambda(t), \mathbf{x}^*(t), \mathbf{u}^*(t)) = 0 \qquad t_0 \le t \le t_f \quad (19)$$

and adjoint equations for the constrained arcs become:

$$\dot{\Lambda}^T = -\Lambda^T \left[\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} + \frac{\partial \mathbf{g}(\mathbf{x})\mathbf{u}}{\partial \mathbf{x}} - \mathbf{g}(\mathbf{x}) \left(\frac{\partial \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right)^{-1} \frac{\partial \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right] \quad (20)$$

In summary, the trajectory minimizing the performance index $J = t_f$ is characterized by the following: *(i)* its controls $\mathbf{u}^*$ are in the admissible set $\mathcal{U}$, *(ii)* the states satisfy the terminal manifolds Γ at t_0 and Ω at t_f, *(iii)* the co-state equations are described by (17) on the free arcs and by (20) on the constrained arcs, with a discontinuity at the junction of the two given by (18), and *(iv)* the Hamiltonian is minimized over the entire interval. These conditions are satisfied by the trajectory computed by the numerical method described next.

3.2 Numerical Computation

The optimal trajectory is computed numerically by appending the constraints to the performance index J via appropriate arrays of Lagrange multipliers, and by computing the corrections to the controls that drive to zero the differential of the augmented performance index $d\hat{J}$ using a steepest {i['set']lt. method [S]. This differential is then derived as function of the variations in the control switches, assuming a bang-bang solution [31]. Singular arcs on the solution are tilt.!l approximated by a finite number of switches generating a quasi-optimal solution [21].

The differential of the augmented performance index is formed by computing the differentials of J, Ω and Ψ_1 independently, then including the effects of Ψ_1 on dJ and $d\Omega$, and finally combining all into $d\hat{J}$. The final form of the augmented performance index is:

$$d\hat{J} = \left(\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \mathcal{H}\right)_{t_f} dt_f + \int_{t_0}^{t_1} \mathcal{H}_u \delta u d\tau + \int_{t_1^+}^{t_f} \mathcal{H}_u \delta u d\tau \quad (21)$$

where the Hamiltonian is given by $\mathcal{H} = \Lambda^T\mathcal{F} + \mu\varphi$ as in equation (16): the Lagrange multiplier Λ is defined as $\Lambda^T = \lambda^T_\phi + \nu^T\lambda^T_\Omega + \eta^T\lambda^T_\Psi$, since tile constraints Ψ_1 are not affected by the State after t_1: and η, and ν are two constant Lagrange multipliers.

Bang-bang controls can only assume one Of two values α_M or α_m and the variation in $\mathbf{u}$ is rewritten as $\delta u_i = (\alpha_M - \alpha_m)\, sgn(dt_i)$, where sgn is the signum function.

If $s_{s,r}$ indicates the number of switches in the s segment of the trajectory for the rth input, and m indicates the number of elements in u. then dJ for bang-bang controls becomes:

$$\begin{aligned} d\hat{J} = & \sum_{i=1}^{m}\sum_{j=1}^{s_{1,i}} (\mathcal{H}_{u_i})_{t_{ij}} \delta u_i dt_{ij} \\ & + \sum_{i=1}^{m}\sum_{j=1}^{s_{2,i}} (\mathcal{H}_{u_i})_{t_{ij}} \delta u_i dt_{ij} \\ & + \sum_{i=1}^{m}\sum_{j=1}^{s_{3,i}} (\mathcal{H}_{u_i})_{t_{ij}} \delta u_i dt_{ij} \\ & + \left(\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \mathcal{H}\right)_{t_f} dt_f \end{aligned} \quad (22)$$

This differential is minimized by choosing the steepest descent increments as [3]:

$$dt_{ij} = -\frac{(\mathcal{H}_{u_i})_{t_{ij}}}{w_{ii}\delta u_i} \quad (23)$$

$$dt_f = -\frac{1}{b}\left(\mathcal{H} + \frac{\partial\phi}{\partial t} + \nu^T \frac{\partial\Omega}{\partial t}\right)_{t_f} \qquad (24)$$

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

The values of dt_{ij} and dt_f depend on multipliers η and ν. Their values are computed by requiring that differentials $d\Psi(t_1)$ and $d\,\Omega(t_f)$ be reduced by given quantities ϵ and θ. The values for the multipliers ν and η are:

$$\begin{aligned}
\eta &= -I_{\Psi\Psi}^{-1}(\epsilon + I_{\Psi\Omega}\nu + I_{\Psi\phi}) \qquad (25)\\
\nu &= -\Big({}^1I_{\Omega\Omega} + {}^1I_{\Omega\Psi}\,{}^1I_{\Psi\Psi}^{-1}\,{}^1I_{\Psi\Omega} + {}^2I_{\Omega\Omega}\\
&\quad + {}^3I_{\Omega\Omega} + \frac{1}{b}\left(\frac{d\Omega}{dt}\frac{d\Omega}{dt}^T\right)_{t_f}\Big)^{-1}\\
&\quad \Big(\theta + {}^1I_{\Omega\phi} - {}^1I_{\Omega\Psi}\,{}^1I_{\Psi\Psi}^{-1}\left(\epsilon + {}^1I_{\Psi\phi}\right) +\\
&\quad {}^2I_{\Omega\phi} + {}^3I_{\Omega\phi} + \frac{1}{b}\left(\frac{d\Omega}{dt}\frac{d\phi}{dt}\right)_{t_f}\Big)\\
{}^lI_{h,k} &= \sum_{i=1}^{m}\sum_{j=1}^{s_{l,i}}\left(\lambda_h^T \frac{\partial\mathcal{F}}{\partial u_i} w_{ii}^{-1} \frac{\partial\mathcal{F}}{\partial u_i}^T \lambda_k\right)_{t_{ij}}
\end{aligned}$$

where $h = \Psi_1, \Omega$, $k = \Psi_1, \Omega, \phi$, and $l = 1, 2, 3$. The index l indicates the segments of the trajectory before, on and after the state constraint. Index i on the constraint boundary represents only the independent controls.

3.3 Generating the Initial Guess

The trajectory computed by the heuristic search consists of a sequence of straight line segments and velocities assigned to each segment. Since the velocity changes abruptly between segments, it is not possible to use this trajectory as the initial guess of the optimization, but it must first be smoothed by a spline interpolation. Each trajectory segment i is divided into the two halves i_1 and i_2, and the spline interpolation is applied to the trajectory segments i_2 and $(i+1)_1$. This replaces the velocity discontinuities by a polynomial blend, chosen as a third order Hermite spline [10].

The bang-bang controls for the spline interpolation are computed as follows. First, the actuator efforts are computed from the accelerations of the spline interpolation using the inverse system dynamics. Then, the smooth actuator efforts are approximated by bang-bang controls, so that the trajectory obtained by the integrating the controls has the same shape of the trajectory computed by the search. The optimal trajectory is the local minimum, in the vicinity of the heuristic trajectory, obtained by improving the position of the switches determined by the threshold.

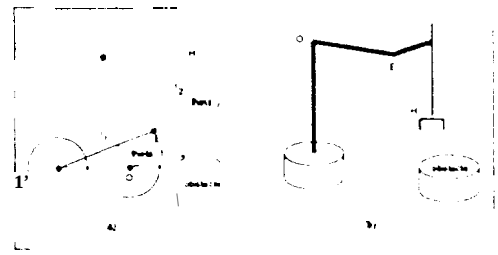

Figure 6: Planar 2-dof Manipulator: a) top view, b) side view

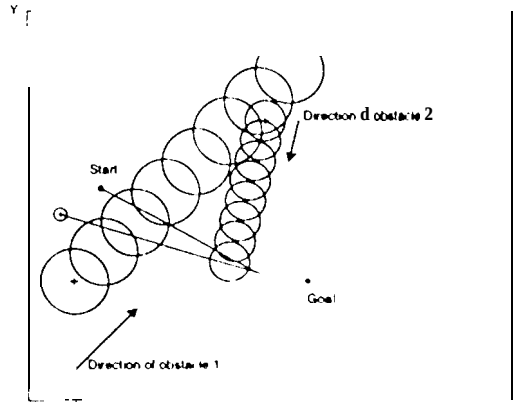

Figure 7: Planning example.

4 Example

In this example, the computation of a trajectory is demonstrated for computing the collision free trajectory of the end effector of a planar manipulator, treated as a SCARA robot with the links placed above the plane of the obstacles. Only the tip of the manipulator reaches the plane of the obstacles by means of a prismatic joint in the z-axis direction, as shown in Figure 6-b.

The time interval T between nodes in the search tree is chosen arbitrarily as $T = 1s$. The circles representing the robot end-effector and the obstacles are displayed at $.5\,s$ intervals.

The initial scenario of this example is shown in Figure 7. The base of the robot is marked by the sign + at its origin. The two obstacles move with constant velocity, starting from the positions marked by the sign +. Obstacle 1 starts from position $(x = .1\,m,\ -.5\,m)$, with velocity $(v_x = .045\,m/s,\ v_y = .045\,m/s)$, and

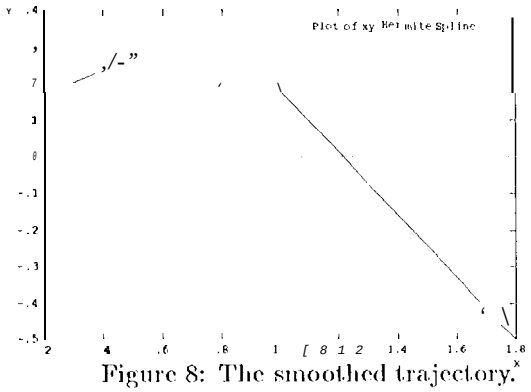

Figure 8: The smoothed trajectory.

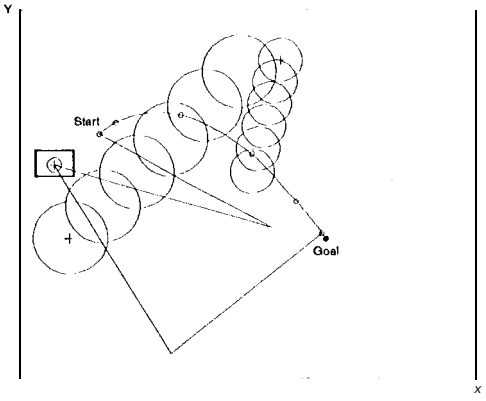

Figure 9: Optimal trajectory.

obstacle 2 starts from position ($x = 1.15\ m,\ .7\ m$) with velocity ($v_x = -\ .007\ m/s, v_y = -\ .03\ m/s$). The robot starts with the hand located at ($x = .3\ m,\ .2\ m$) and zero velocity, and is required to reach the goal marked by the black dot.

The spline interpolation of the trajectory computed by the search is shown in Figure 8, and the resulting trajectory has a motion time of is 4.81 s. This is the nominal trajectory used by the dynamic optimization. The solution computed by the dynamic optimization is shown Figure 9, and it has a motion time of 2.6 s.

5 Conclusion

A novel method for planning the motion of a robot moving in a time-varying environment, the Velocity Obstacle approach, has been presented. It is significantly different from currently available planning algorithms, since it simultaneously computes the path and velocity profile that avoid all static and moving obstacles and satisfy the robot's dynamic constraints.

The method Consists of computing, for every obstacle, its corresponding velocity obstacle, which is the set of colliding velocities between the obstacle and the robot. **'I '11(' 11.** by subtracting it from the reachable velocities of the robot. The set of reachable avoidance velocities is formed, which consists of all the velocities that avoid the obstacles and satisfy the robot's dynamic constraints. A search space is then formed by representing the state space of the robot by a tree of avoidance maneuvers. A global search over the tree yields trajectories that minimize motion time. The solution is then refined by a dynamic optimization to compute the time optimal solution in the vicinities of the approximate trajectory. The method is demonstrated for planning the trajectory of a simple manipulator.

The main advantages of the velocity obstacle approach include the efficient geometric representation of maneuvers avoiding any number of moving and static obstacles, and the simple consideration of robot dynamics and actuator constraints. This approach, therefore, makes it possible to compute safe and feasible trajectories for robots in dynamic environments.

6 Acknowledgment,

The research described in this paper has been partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautic 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 W.F. Denham. A steepest-ascent method for solving optimum programming problems. *ASME Journal of Applied Mechanics*, (29):247 257, June 1962.

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

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

[5] J. Canny, J. Reif, B. Donald, and P. Xavier. On the complexity of kinodynamic planning. In *29th IEEE Symposium on Foundation of Computer Science*, 1988.

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

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

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

[9] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using the relative velocity paradigm. In *IEEE International Conference of Automation and Robotics*, volume 1, pages 560 566, May 1993.

[10] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics*. Addison-Wesley Publishing Company, Reading, MA. 1990.

[11] K. Fujimura. Motion planning amid transient obstacles. *International Journal of Robotics Research*. 13(5):395 407, October 1994.

[12] K. Fujimura and H. Samet. Motion planning in a dynamic domain. In *IEEE International Conference on Robotics and Automation*, pages 324 330, Cincinnati, OH, May 1990.

[13] D.H. Jacobson, M.M. Lele, and J.L. Speyer. New necessary conditions of optimality for control problems with state-varial~lc inequality constraints. *Journal of Mathematical Analysis and Applications*, 35(2):255 284, 1971.

[14] D.W. Johnson and E.G. Gilbert. Minimum time robot planning in the presence of obstacles. In *IEEE Conference on Decision and Control*, pages 1748 1753, Ft. Lauderdale, FL, December 1985.

[15] K. Kant and S.W. Zucker. Towards efficient trajectory planning: the path-velocity decomposition. *The International Journal of Robotic Research*, 5(3):72 89, Fall 1986.

[16] O. Khatib. A unified approach for motion and force control of robot manipulators: the operational space formulation. *IEEE Transaction on Robotics and Automation*, 3:43 53, 1987.

[17] B.H. Lee and C.S. G. Lee. Collision-free motion planning of two robots. *IEEE Transactions on System Man and Cybernetics*, SMC-17(1):2132, January-February 1987.

[18] G. Leitman. *An Introduction to Optimal Control*. McGraw-Hill Book Co., New York, NY. 1966.

[19] T. Lozano-Pérez. Spatial planning: a configuration space approach. *IEEE Transaction on Computers*, c 32(2):1 08 120, February 1983.

[20] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collisim-free paths among polyhedral obstacles. *Communications of the ACM*. 22(10):560 570. October 1979.

[21] E.B. Meier. *An Efficient Algorithm for Bang-Bang Control Systems Applied to a Two-Link Manipulator*. PhD thesis, Stanford, December 1987.

[22] N Nilsson. *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA, 1980.

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

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

[25] J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *25th IEEE Symposium on the Foundation of Computer Science*, pages 144 153, 1985.

[26] J.C. Sanborn and J.A. Hendler. A model of reaction for planning in dynamic environments. *International Journal of Artificial Intelligence in Engineering*, 3(2):95 101, April 1988.

[27] Z. Shiller and Dubowsky. Time optimal paths and acceleration lines of robotic manipulators. In *The 26th IEEE Conference on Decision and Control*, Los Angeles, CA, December 1987.

[28] Z. Shiller and S. Dubowsky. Robot path planner with obstacles, actuators, gripper and payload constraints. *The International Journal of Robotics Research*, 8(6):3 18, December 1989.

[29] K.G. Shin and N.D. McKay. Open loop minimum time control of mechanical manipulators. In *American Control Conference*, pages 1231 1236, June 1984.

[30] J. Speyer. *Optimization and Control of Nonlinar Systems with Inflight Constraints*. PhD thesis, Harvard, February 1968.

[31] A. Weinreb and A.E. Bryson. Optimal control of systems with hard control bounds. *IEEE Transactions on Automatic Control*, AC-30(d116), November 1985.