

Time Optimal Trajectory Planning in Dynamic Environments

Paolo Fiorini* and Zvi Shiller†

* Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109 USA

† Department of Mechanical, Nuclear and Aerospace Engineering
University of California, Los Angeles
Los Angeles, CA 90024 USA

Abstract

This paper presents a direct method for computing the time optimal trajectory for a robot among stationary and moving obstacles, subject to robot's dynamics and actuator constraints. The motion planning problem is first formulated as an optimization problem, and then solved numerically using a gradient descent. The initial guess for the optimization is generated using a method based on the concept of Velocity Obstacles. The method is demonstrated for a 2-DOF planar manipulator moving in static and dynamic environments.

1. Introduction

Motion planning is central to the operation of autonomous robots. It concerns the generation of a trajectory from start to goal that satisfies objectives, such as minimizing path distance or motion time, while avoiding obstacles in the environment and satisfying the robot mechanics (kinematics and dynamics). We distinguish between *planning* and *control* in that the former generates a nominal trajectory, whereas the latter tracks that trajectory. Robot motion planning is generally too complex to be handled by on-line feedback controllers due to the nonlinear state constraints introduced by the obstacles and the highly nonlinear and coupled nature of robot mechanics.

Traditionally, motion planning has been treated as a *kinematic* problem, i.e. determining the path that avoids obstacles without concern to robot speeds. This was first extensively addressed for articulated robots by transforming the problem into the *configuration space*, in which the robot reduces to a point and the obstacles map into C-space obstacles [25,29]. The focus in this body of work has centered on computational complexity and completeness (the ability of the algorithm to find a path if one exists). More recently the *kinematic* problem was extended to car-like robots, which are subject to *non-holonomic* kinematic constraints due to the assumption of no slip between the wheels and ground. Here the focus has centered on obstacle avoidance [27] and on minimizing path distance [26].

While solving a problem fundamental to robotics, *kinematic* motion planning ignores the important effects of robot dynamics which become significant at all but the lowest speeds. For example, non-holonomic motion planning of a car is useful for parking [31], which is usually done

at very low speeds, but is all but meaningless for high speed emergency maneuvers [36]. Similarly, obstacle-free paths computed using robot kinematics only may be dynamically infeasible at even moderate speeds, causing the robot to deviate from the *kinematic* path due to its dynamics and limited actuator efforts. This gave rise to *dynamic* motion planning¹, which produces a *trajectory* in the state space rather than just a *path* in the configuration space. Planning in the state space, while computationally more extensive, allows one to minimize *dynamic* cost functions, such as time or energy. These problems have been treated previously for both articulated [34,35] and mobile robots [37].

We distinguish between motion planning in *static* and in *dynamic* environments. In static environments, the obstacles are static, and the robot is the only one that moves, whereas in dynamic environments, both robot and obstacles move. Typical examples of dynamic environments include manufacturing tasks in which robot manipulators track and retrieve parts from moving conveyers, and intelligent vehicles negotiating freeway traffic.

Motion planning in dynamic environments was originally addressed by adding the time dimension to the robot's configuration space, assuming bounded velocity and known trajectories of the obstacles [8,18,32]. Reif and Sharir [32] solved the planar problem for a polygonal robot among many moving polygonal obstacles by searching a visibility graph in the configuration-time space. Erdmann and Lozano-Pérez [8] discretized the configuration-time space to result in a sequence of configuration space slices at successive time intervals. This method essentially solves the static planning problem at every slice and joins adjacent solutions. Fujimura and Samet [18] used a cell decomposition to represent the configuration-time space, and joined empty cells to connect start to goal.

Another approach to dynamic motion planning is to decompose the problem into smaller problems: path planning and velocity planning. This method first computes a feasible path among the static obstacles, and represents it as a parametric curve in the arc length. Then, the intersections of the moving obstacles with the path are represented as forbidden regions in an arc length-time plane. The velocity along the path is chosen to avoid the forbidden regions [14,15,17,19,24,28]. Kant and Zucker [24] selected both path and velocity profile using a visibility graph approach. Lee and Lee [28] developed independently a similar approach for two cooperating robots, and compared the effects of delay and velocity reduction on motion time. Fraichard [14] considered acceleration bounds, and used a search in a state-time space (s, \dot{s}, t) to compute the velocity profile yielding a minimum-time trajectory. Fraichard and Laugier [15] considered adjacent paths that could be reached from the nominal path when the nominal path becomes blocked by a moving obstacle. Fujimura [17] considered the case of a robot moving on a fixed time-dependent network, whose nodes could be temporarily occluded by moving obstacles.

A different approach consists of generating the accessibility graph of the environment, which is an extension of the visibility graph [19,20]. Fujimura and Samet [19] defined it as the locus of points on the obstacles which are reachable by the robot moving at maximum speed. These points form the *collision front*, and can be linked together to construct a path from start to goal. The accessibility graph has the property that, if the robot moves faster than the obstacles, the path computed by searching the graph is time-minimal. This concept was extended in [16] to the case of slowly moving robots and transient obstacles, i.e. obstacles that could appear and disappear in the environment.

None of the previous methods considered the non-linear robot dynamics, and none produced time-optimal motions. Time-optimal motions have obvious benefits in industrial applications by reducing cycle times and thus increasing the productivity of automated manufacturing systems.

¹Others use *dynamic* motion planning to denote motion planning in *dynamic* environments [25], which is a subset of our definition.

Other application domains, such as intelligent vehicles and air traffic control, may benefit from time-optimal motions by minimizing the recovery time from emergency situations and when defining emergency maneuvers.

The time-optimal motion planning problem in static environments has been treated previously, beginning with the work by Kahn and Roth [23], who solved the problem for a linearized robot model, using the Pontryagin's Minimum Principle (PMP). The full robot model was used in [30], assuming bang-bang control and using a steepest descent over the switching times, derived to satisfy the necessary conditions of optimality stated by the PMP. However, the most efficient methods to date seem to consist of parameter optimizations over the trajectory [1,22,35], which are similar to the Differential Inclusions introduced in [33], and the Inverse Dynamic Optimization introduced by Bryson [3].

In this paper, we present a method for computing the time optimal trajectories of a robot moving in a dynamic environment. To make the problem computationally tractable, we restrict the treatment to the plane and assume circular robot and obstacles. We also assume a full knowledge of the environment.

Central to this approach is the computation of the initial guess for the optimization. This is done by utilizing the concept of Velocity Obstacle [9], which maps the dynamic environment into the robot velocity space. The velocity obstacle is the first-order approximation of the robot's velocities that would cause a collision with an obstacle at some future time, within a given time horizon. Feasible avoidance maneuvers are computed simply by selecting velocities outside the velocity obstacle, and satisfying additional velocity constraints computed from robot dynamics and actuator constraints. The initial guess of the optimal trajectory is computed by a global search over a tree of feasible avoidance maneuvers, generated at discrete time intervals so as to minimize time to the goal.

The optimal trajectory is computed using a steepest descent algorithm over the admissible controls [5–7], modified to consider time varying state inequality constraints. The state inequality constraints due to the moving obstacles are considered by transforming them into state-dependent control constraints. The method was implemented for intelligent vehicles negotiating freeway traffic [38], and for a planar SCARA robot, considering its full nonlinear dynamics and moving circular obstacles [9]. Examples of the latter are presented in this paper.

The paper is organized as follows. Section 2 formulates the motion planning problem as a minimum time problem and presents the numerical method for computing the optimal solution satisfying state inequality constraints and state-dependent control constraints. Then, Section 3 addresses the problem of generating the nominal trajectory for the numerical optimization. Finally, examples of optimal trajectories of a SCARA robot avoiding fixed and moving obstacles are presented in Section 4.

2. The Dynamic Optimization

The dynamic motion planning problem in the context of this paper consists of determining the trajectory between two specified boundary conditions that avoids all static and moving obstacles and minimizes motion time. This is formulated as an optimization problem with time-varying state constraints, and is solved numerically using the steepest descent method [7], as discussed next.

2.1. Problem Formulation

The motion planning problem can be formulated as follows: Find the control $\mathbf{u}^*(t) \in \mathbf{U}$ in $t_0 \leq t \leq t_f$, which minimizes the cost function J :

$$\min_{u(t) \in \mathbf{U}} J = \min_{u(t) \in \mathbf{U}} \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t)) dt = \phi_{min}(\mathbf{x}(t_f), t_f) = t_f \quad (1)$$

where t_f is free, subject to robot dynamics

$$\dot{\mathbf{x}} = \mathcal{F}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \quad (2)$$

admissible controls

$$\mathbf{U} = \{\mathbf{u} \mid \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max}\} \quad (3)$$

initial conditions

$$\mathbf{x}(t_0) = \mathbf{x}_0 \quad (4)$$

terminal manifold

$$\Omega(\mathbf{x}(t_f), t_f) = 0 \quad (5)$$

and state inequality constraints due to the moving obstacles:

$$\Psi : \bigcup_{i=1}^n [S_i(\mathbf{x}(t), t) \geq 0] \quad (6)$$

where $S_i(\mathbf{x}(t), t)$ represents the time-varying boundaries of the moving obstacles.

The original problem calls for a fixed final point. However, we assume instead a terminal manifold (a hyper-sphere around the final point) so that we can use influence functions to compute the initial conditions of the Lagrange multipliers, and thus avoid using the more sensitive shooting method [2].

State inequality constraints are generally difficult to satisfy although necessary conditions for optimality have been developed for such problems [21,39]. One way to consider state inequality constraints is to transform them into state-dependent control *equality* constraints, active only when the robot slides along the obstacle boundary [6,7].

To demonstrate the treatment of the state inequality constraints, we consider a single obstacle:

$$\Psi : S(\mathbf{x}(t), t) \geq 0 \quad , \quad S(x) \in \mathbb{R}^m \quad (7)$$

where m is the dimension of the position space. Differentiating (7) with respect to time p times until it becomes explicit in the control \mathbf{u} , and assuming an active constraint, yields the state-dependent control constraint

$$S^{(p)}(\mathbf{x}, \mathbf{u}) = 0 \quad (8)$$

where $S^{(p)}$ denotes the p th derivative of S , with p being the order of the constraint.

A solution satisfying (8) does not necessarily satisfy (7), unless it passes through at least one point satisfying (7) and all the derivatives of order less than p . We choose this point to be the initial entry point of the constrained arc, at time $t_1 > t_0$. The inequality constraint (7) is

thus replaced by the tangency point condition, Ψ_1 , and the state-dependent equality constraint, Ψ_2 :

$$\Psi_1 : \begin{pmatrix} S(\mathbf{x}(t_1), t_1) = 0 \\ \dot{S}(\mathbf{x}(t_1), t_1) = 0 \\ \vdots \\ S^{(p-1)}(\mathbf{x}(t_1), t_1) = 0 \end{pmatrix} \quad (9)$$

$$(10)$$

$$\Psi_2 : S^{(p)}(\mathbf{x}(t), \mathbf{u}(t), t) = 0 \quad t_1 \leq t \leq t_2 \quad (11)$$

where t_1 is the entry time, and t_2 is the exit time of the constrained arc. This also modifies the admissible controls (3) to:

$$\mathcal{U} : \begin{cases} \mathbf{U} : \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max} \\ S^{(p)}(\mathbf{x}(t), \mathbf{u}(t), t) = 0 \quad \text{for } t \in (t_1, t_2] \end{cases} \quad (12)$$

The addition of the tangency constraint, Ψ_1 , thus transforms the original Two Point Boundary Value Problem (1) into a Three Point Boundary Value Problem (for a single moving obstacle), which is solved numerically using the method discussed next.

Note that this treatment of the state inequality constraints may over-constrain the problem since the trajectory is forced to satisfy the state constraint as an equality along a finite arc. Consequently, this approach cannot find solutions that touch the state constraint at multiple isolated points [21]. This, however, has been shown to affect only constraints of order higher than two, and is hence not an issue for the circular obstacles treated here [21].

2.2. Numerical Computation

We apply the steepest descent method, which rigorously satisfies a set of necessary optimality conditions. This method was originally developed in [5], modified to include state dependent control inequality constraints in [7], and modified to consider bang-bang controls in [30].

The steepest descent method iteratively computes the optimal controls by following the negative gradient of the augmented cost function with respect to the controls and the final time. The gradient is derived by adjoining the differential of the cost function with the differentials of the terminal manifold and the tangency-point constraint, as discussed below.

2.2.1. The Differential of the Performance Index

Following the classic approach to constrained optimization [6], system dynamics and control constraints are adjoined to the performance index J using two arrays of Lagrange functions $\lambda_\phi(t) \in \mathfrak{R}^n$ and $\mu(t) \in \mathfrak{R}^k$, where n is the dimension of the state space, and k is the number of active state-dependent control constraints. This leads to the performance index \tilde{J} :

$$\tilde{J} = \phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \left[\lambda_\phi^T (\mathcal{F}(\mathbf{x}, \mathbf{u}) - \dot{\mathbf{x}}) + \mu^T \varphi(\mathbf{x}, \mathbf{u}) \right] d\tau \quad (13)$$

where

$$\varphi = \begin{cases} 0 & t \notin (t_1, t_2) \\ S^{(p)} & t \in (t_1, t_2) \end{cases} \quad (14)$$

and μ is a vector of Kuhn-Tucker multipliers [6]

$$\mu = \begin{cases} 0 & \text{when } \varphi = 0 \\ -\Lambda^T \mathbf{g}(\mathbf{x}) \left(\frac{\partial \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \right)^{-1} & \text{otherwise} \end{cases} \quad (15)$$

By defining the Hamiltonian as:

$$\mathcal{H}(\lambda_\phi, \mathbf{x}, \mathbf{u}) = \lambda_\phi^T \mathcal{F}(\mathbf{x}, \mathbf{u}) + \mu^T \varphi(\mathbf{x}, \mathbf{u}) \quad (16)$$

and by choosing:

$$\dot{\lambda}_\phi(t) = - \left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}} \right)^T \quad (17)$$

$$\lambda_\phi(t_f) = \left(\frac{\partial \phi}{\partial \mathbf{x}} \right)_{t_f} \quad (18)$$

we reduce the differential $d\tilde{J}$ to:

$$d\tilde{J} = \int_{t_0}^{t_f} \frac{\partial \mathcal{H}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \left(\frac{\partial \phi}{\partial t} + \mathcal{H} \right)_{t_f} dt_f \quad (19)$$

This establishes the relations between variations in the independent variables, \mathbf{u} and t_f , and variations in the cost function for the unconstrained problem.

2.2.2. The Differential of the Terminal Constraint

The differential of the terminal constraint Ω is:

$$(d\Omega)_{t_f} = \left(\frac{\partial \Omega}{\partial x} dx + \frac{\partial \Omega}{\partial t} dt \right)_{t_f} \quad (20)$$

Following the derivation in Appendix A, and choosing multipliers $\lambda_\Omega \in \mathbb{R}^n \times \mathbb{R}^l$ (where l is the number of terminal constraints) to satisfy:

$$\dot{\lambda}_\Omega(t) = - \left(\frac{\partial \mathcal{F}}{\partial x} \right)^T \lambda_\Omega(t) \quad (21)$$

$$\lambda_\Omega(t_f) = \left(\frac{\partial \Omega}{\partial x} \right)_{t_f} \quad (22)$$

the differential $d\Omega$ (20) reduces to:

$$(d\Omega)_{t_f} = \int_{t_0}^{t_f} \lambda_\Omega^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \left(\frac{\partial \Omega}{\partial x} \dot{x} + \frac{\partial \Omega}{\partial t} \right)_{t_f} dt_f \quad (23)$$

2.2.3. The Differential of the Point Constraint

Similarly, the differential of the intermediate tangency constraints, Ψ_1 , at time t_1 , is:

$$(d\Psi_1)_{t_1} = \left(\frac{\partial \Psi_1}{\partial x} dx + \frac{\partial \Psi_1}{\partial t} dt \right)_{t_1} \quad (24)$$

Following the derivation in Appendix A, and choosing the Lagrange functions $\lambda_\Psi \in \mathbb{R}^n \times \mathbb{R}^k$ (where k is the number of constraints Ψ_1)

$$\dot{\lambda}_\Psi(t) = - \left(\frac{\partial \mathcal{F}}{\partial x} \right)^T \lambda_\Psi(t) \quad (25)$$

$$\lambda_\Psi(t_1) = \left(\frac{\partial \Psi}{\partial x} \right)_{t_1} \quad (26)$$

the differential $d\Psi$ (24) reduces to:

$$(d\Psi_1)_{t_1} = \int_{t_0}^{t_1} \lambda_\Psi^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \left(\frac{\partial \Psi}{\partial x} \dot{x} + \frac{\partial \Psi}{\partial t} \right)_{t_1} dt_1 \quad (27)$$

2.2.4. Discontinuity of the Lagrange Functions

The Lagrange functions λ_ϕ , and λ_Ω are integrated through the entry point of the constrained arc at t_1 , where they are discontinuous. This discontinuity is computed as a function of the jump in the acceleration (for a second order system) across the entry point to the constrained arc [4] (see also Appendix B):

$$\lambda_\phi^T(t_1^-) = \lambda_\phi^T(t_1^+) \left(I - \frac{\dot{\mathbf{x}}(t_1^-) - \dot{\mathbf{x}}(t_1^+)}{S^{(p)}(t_1)} \frac{\partial S^{(p-1)}}{\partial \mathbf{x}} \Big|_{t_1} \right) \quad (28)$$

$$\lambda_\Omega^T(t_1^-) = \lambda_\Omega^T(t_1^+) \left(I - \frac{\dot{\mathbf{x}}(t_1^-) - \dot{\mathbf{x}}(t_1^+)}{S^{(p)}(t_1)} \frac{\partial S^{(p-1)}}{\partial \mathbf{x}} \Big|_{t_1} \right) \quad (29)$$

2.2.5. The Differential of the Augmented Performance Index

The differential of the augmented performance index $d\underline{J}$ consists of the differentials (23) and (27), appended to the differential $d\tilde{J}$ with the constant multipliers η and ν :

$$\begin{aligned} d\underline{J} = & \left[\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \left(\frac{\partial \phi}{\partial \mathbf{x}} + \nu^T \frac{\partial \Omega}{\partial \mathbf{x}} \right) \dot{\mathbf{x}} + \mu^T \varphi(\mathbf{x}, \mathbf{u}) \right]_{t_f} dt_f \\ & + \int_{t_0}^{t_1^-} \left[\frac{(\lambda_\phi^T + \nu^T \lambda_\Omega^T + \eta^T \lambda_\Psi^T) \mathcal{F} + \mu^T \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \right] \delta \mathbf{u} d\tau \\ & + \int_{t_1^+}^{t_f} \left[\frac{(\lambda_\phi^T + \nu^T \lambda_\Omega^T) \mathcal{F} + \mu^T \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \right] \delta \mathbf{u} d\tau \end{aligned} \quad (30)$$

Note that the multipliers λ_Ψ are defined only between t_0 to t_1 , since Ψ_1 is not affected by the states after t_1 . Setting $\lambda_\Psi(t) = 0$ for $t > t_1$ we define an augmented Lagrange function, Λ :

$$\Lambda^T = \lambda_\phi^T + \nu^T \lambda_\Omega^T + \eta^T \lambda_\Psi^T \quad (31)$$

which yields the Hamiltonian:

$$\mathcal{H}(\Lambda, \mathbf{x}, \mathbf{u}) = \Lambda^T \mathcal{F}(\mathbf{x}, \mathbf{u}) + \mu^T \varphi(\mathbf{x}, \mathbf{u}) \quad (32)$$

and reduces (30) to:

$$d\underline{J} = \left(\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \mathcal{H} \right)_{t_f} dt_f + \int_{t_0}^{t_1^-} \mathcal{H}_u \delta \mathbf{u} d\tau + \int_{t_1^+}^{t_f} \mathcal{H}_u \delta \mathbf{u} d\tau \quad (33)$$

This establishes the relations between variations in the independent variables, \mathbf{u} and t_f , and variations in the cost function for the *constrained* problem, including the terminal manifold, the tangency point, and the state-dependent control constraint. Assuming bang-bang control, we use these relations to compute the variations in the switching times that would zero the differential of the augmented cost function.

2.2.6. The Bang-Bang Solution

It is easy to show that the solution for minimum-time problems consists of bang-bang controls for systems linear in the controls, excluding singular arcs [6] [40]. By assuming bang-bang control we reduce the functional optimization to a parameter optimization over the switching times. The number of switches is approximated from the initial guess, as discussed later, and the singular arcs are approximated by a finite number of switches [30].

For bang-bang controls, the variations δu_i in (33) are replaced with:

$$\delta u_i = (\alpha_m - \alpha_m) \operatorname{sgn}(dt_{ij}) \quad (34)$$

where sgn is the signum function, and dt_{ij} is the change of the j th switching time for control u_i . Note that $\delta u_i \neq 0$ only at the switching times where u_i switches between the extremes. Therefore δu_i is represented by

$$\delta u_i = (-1)^{j-1} \Delta \alpha dt_{ij} \quad (35)$$

where

$$\Delta \alpha = \alpha_M - \alpha_m \quad (36)$$

Using (35) we now discretize the augmented cost function $d\underline{J}$ of (33) as a function of the switching times:

$$\begin{aligned} d\underline{J} = & \sum_{i=1}^m \sum_{j=1}^{s_{1,i}} (\mathcal{H}_{u_i})_{t_{ij}} \Delta \alpha dt_{ij} + \sum_{i=1}^m \sum_{j=1}^{s_{2,i}} (\mathcal{H}_{u_i})_{t_{ij}} \Delta \alpha dt_{ij} \\ & + \sum_{i=1}^m \sum_{j=1}^{s_{3,i}} (\mathcal{H}_{u_i})_{t_{ij}} \Delta \alpha dt_{ij} + \left(\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \mathcal{H} \right)_{t_f} dt_f \end{aligned} \quad (37)$$

where s_1 represents the segment of the trajectory before the obstacle, s_2 represents the constrained arc, and obstacle, s_3 is the segment of the trajectory from the obstacle to the target, and m is the dimension of \mathbf{u} . Since the second term in (37) corresponds to the constrained arc, the corrections dt_{ij} are computed only for the controls not determined from $S^{(p)}(\mathbf{x}, \mathbf{u}, t) = 0$.

The objective now is to determine the variations dt_{ij} that would minimize the differential $d\underline{J}$. This can be done by following the negative gradient of $d\underline{J}$ defined by the coefficients of the dt_{ij} in (37). The step size of each move is determined by adding a quadratic term in dt_{ij} and dt_f to (37) [30]:

$$\begin{aligned} d\hat{J} = & \sum_{i=1}^m \sum_{j=1}^{s_{1,i}} ((\mathcal{H}_{u_i})_{t_{ij}} \Delta \alpha dt_{ij} + \frac{1}{2} w_{ii} \Delta \alpha_i^2 dt_{ij}^2) \\ & + \sum_{i=1}^m \sum_{j=1}^{s_{2,i}} (\mathcal{H}_{u_i})_{t_{ij}} \Delta \alpha dt_{ij} + \frac{1}{2} w_{ii} \Delta \alpha_i^2 dt_{ij}^2 \\ & + \sum_{i=1}^m \sum_{j=1}^{s_{3,i}} (\mathcal{H}_{u_i})_{t_{ij}} \Delta \alpha dt_{ij} + \frac{1}{2} w_{ii} \Delta \alpha_i^2 dt_{ij}^2 \\ & + \left(\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \mathcal{H} \right)_{t_f} dt_f + \frac{1}{2} b (dt_f)^2 \end{aligned} \quad (38)$$

where b is a positive constant, and w_{ii} are the elements of a diagonal positive definite matrix.

The step size that minimizes (38) is given by:

$$dt_{ij} = - \frac{(\mathcal{H}_{u_i})_{t_{ij}}}{w_{ii} \Delta \alpha} \quad (39)$$

$$dt_f = - \frac{1}{b} \left(\mathcal{H} + \frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} \right)_{t_f} \quad (40)$$

The values of dt_{ij} and dt_f in equations (39) and (40) depend on the multipliers η and ν , which are computed by back-substituting (39) and (40) in (23) and (27), and by multiplying

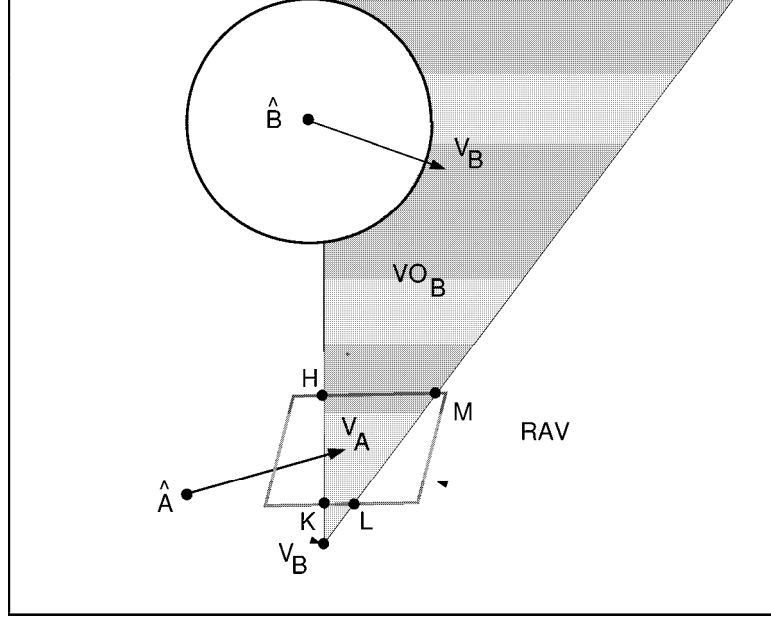


Figure 1: The feasible avoidance velocities RAV.

$d\Psi_1(t_1)$ and $d\Omega(t_f)$ by $-\epsilon$, with ϵ a small positive constant. This scales the improvements in η and ν to satisfy the first-order necessary conditions of optimality.

With this substitution, equations (23) and (27) yield:

$$\begin{aligned} \eta &= -I_{\Psi\Psi}^{-1}(-\epsilon d\Psi_{t_1} + I_{\Psi\Omega}\nu + I_{\Psi\phi}) & (41) \\ \nu &= -\left({}^1I_{\Omega\Omega} + {}^1I_{\Omega\Psi} {}^1I_{\Psi\Psi}^{-1} {}^1I_{\Psi\Omega} + {}^2I_{\Omega\Omega} + {}^3I_{\Omega\Omega} + \frac{1}{b} \left(\frac{d\Omega}{dt} \frac{d\Omega^T}{dt} \right)_{t_f} \right)^{-1} \\ &\quad \left(-\epsilon d\Omega_{t_f} + {}^1I_{\Omega\phi} - {}^1I_{\Omega\Psi} {}^1I_{\Psi\Psi}^{-1} (-\epsilon d\Psi_{t_1} + {}^1I_{\Psi\phi}) + {}^2I_{\Omega\phi} + {}^3I_{\Omega\phi} + \frac{1}{b} \left(\frac{d\Omega}{dt} \frac{d\phi}{dt} \right)_{t_f} \right) \end{aligned}$$

where the terms ${}^lI_{hk}$ are defined as:

$${}^lI_{hk} = \sum_{i=1}^m \sum_{j=1}^{s_{l,i}} (\lambda_h^T \frac{\partial \mathcal{F}}{\partial u_i} w_{ii}^{-1} \frac{\partial \mathcal{F}^T}{\partial u_i} \lambda_k)_{t_{ij}} \quad (42)$$

with $h = \Psi_1, \Omega$, $k = \Psi_1, \Omega, \phi$, $l = 1, 2, 3$, representing *before*, *on* and *after* the state constraint, and i indicating the independent controls.

This procedure reduces the differential defined in (33) to zero, which also satisfies the necessary conditions of optimality stated by the Pontryagin Minimum Principle, as discussed in [9].

3. The Initial Guess

The dynamic optimization discussed earlier converges only to a local minimum, which depends on the initial guess. Since the dynamic motion planning problem is generally not convex, i.e. it has multiple local minima, selecting the appropriate initial guess would determine

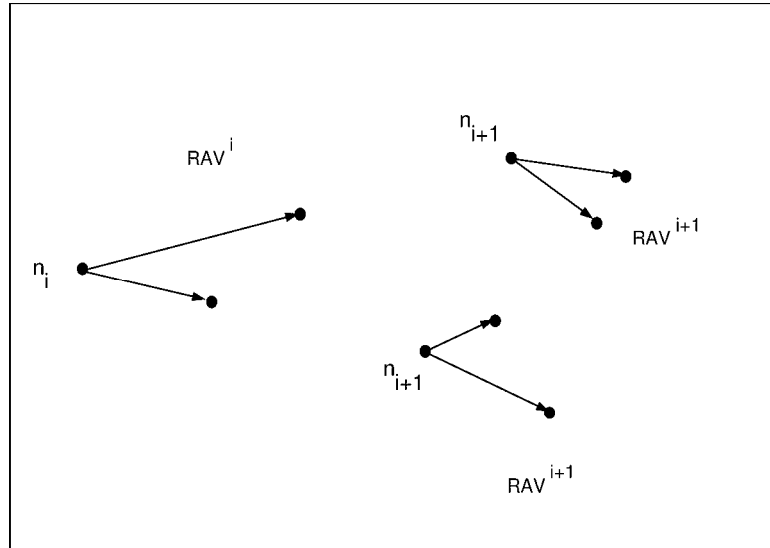


Figure 2: Tree representation for the global search.

the quality of the solution. While it is generally desirable to compute the global minimal trajectory, it is equally important to obtain a trajectory specified in terms of the sequence of avoidance and the side from which each obstacle is being avoided. Selecting an initial guess in dynamic environments is in itself a dynamic motion planning problem, as discussed earlier in the Introduction. Imposing a desired structure makes the problem only harder.

An efficient method for solving both problems has been recently developed [12]. It generates trajectories that are both collision-free and dynamically feasible. Below, we first briefly summarize this approach, and then compute a bang-bang approximation for the controls.

3.1. Generating the Trajectory

The method for generating feasible trajectories in dynamic environments is based on the concept of velocity obstacles, which is a first-order approximation of the robot velocities that would cause a collision with some obstacle at some future time [9,11]. Collision is avoided by selecting velocities outside the union of the velocity obstacles due to all moving and static obstacles.

To ensure that the selected maneuver is also dynamically feasible, we impose additional velocity constraints due to robot dynamics and actuator constraints, as shown in Figure 1. Figure 1 shows the velocity obstacle of \hat{B} , moving at some velocity \mathbf{v}_B , with respect to a point robot, \hat{A} . Also shown are the feasible velocities RAV, which for a planar robot are represented by a parallelogram. The feasible avoidance velocities are confined to the set defined by the difference between the feasible avoidance velocities and the velocity obstacle.

An avoidance maneuver consists of a velocity vector and a time interval over which that velocity is applied. Maneuvers can be selected to minimize a global cost function, such as motion time, or to satisfy local objectives, such as passing an obstacle from the front rather than from the rear.

A trajectory consists of a sequence of avoidance maneuvers. A trajectory that minimizes motion time can be generated by searching over a tree of feasible avoidance maneuvers, generated at discrete time intervals. Figure 2 shows two branches of the tree, rooted in node n_i at time i

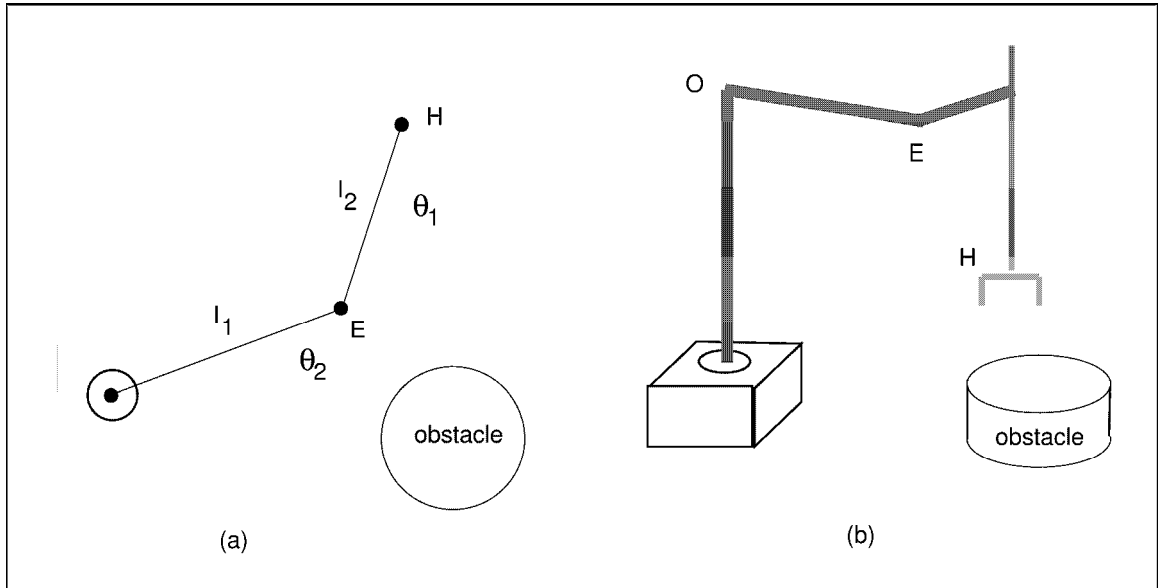


Figure 3: Planar 2-dof Manipulator: a) top view, b) side view

and reaching nodes n_{i+1} . The feasible avoidance velocities at times i and $i + 1$ are represented by RAV^i and RAV^{i+1} . A trajectory generated by this search is a good initial guess for the dynamic optimization, since it is quasi optimal, and it has the desired topological properties (i.e. sequence of avoidance and type of maneuvers). A drawback of this trajectory is that its velocity profile is discontinuous, and hence cannot be differentiated to compute the nominal controls. This is resolved by first smoothing the trajectory using Hermite splines, as discussed next.

3.2. Generating the Controls

To compute the controls, we first smooth the trajectory, consisting of a sequence of avoidance maneuvers, using a spline interpolation. First, the path is smoothed by joining the mid-points of every consecutive path segments with a third order Hermite spline that matches the slopes of the path segments [13]. Then, the velocity profile along the resulting path is smoothed using a cycloid between the mid-points of consecutive velocity segments, given by:

$$v(t) = \frac{\omega t - \sin(\omega t)}{2.0\pi} \quad (43)$$

where $\omega = \frac{2.0\pi}{T}$, and T is the motion time between the two mid-points.

Using inverse dynamics, we now compute the controls associated with the smoothed trajectory. The resulting actuator efforts are approximated by bang-bang controls by choosing the switching times at the zero crossings of the smooth controls, with a dead-band to avoid chatter [9,11].

4. Examples

Here we present examples for the two degree-of-freedom planar manipulator shown in Figure 3. The problem is greatly simplified by assuming a planar SCARA manipulator with two

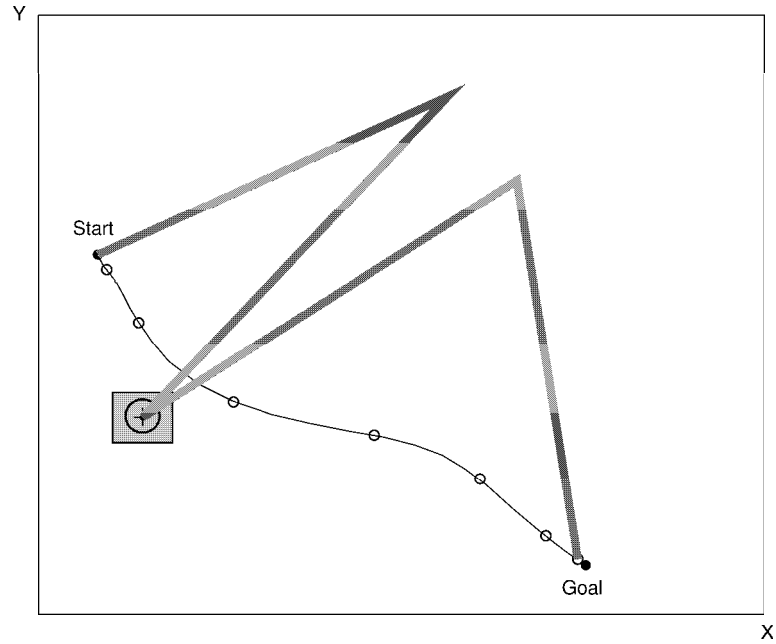


Figure 4: Optimal trajectory in the free environment

uniform links and with only the end effector reaching the plane of the obstacles. Then, only the end effector trajectory among the obstacles needs to be computed. The parameters of the arm are: $l_1 = 1.5$ m, $l_2 = 1.3$ m, $m_1 = 10.0$ Kg, $m_2 = 10.0$ Kg, $\tau_1 = 10.0$ Nm, $\tau_2 = 3.0$ Nm.

4.1. Single Obstacle

The objective in the following examples is to move the end-effector from rest at the starting position $\mathbf{x} = (-.15$ m, $.55$ m), to rest at the goal position $\mathbf{x} = (1.5$ m, $-.5$ m), in minimum time.

First, the optimal path, computed with no obstacles, is shown in Figure 4. The actuator

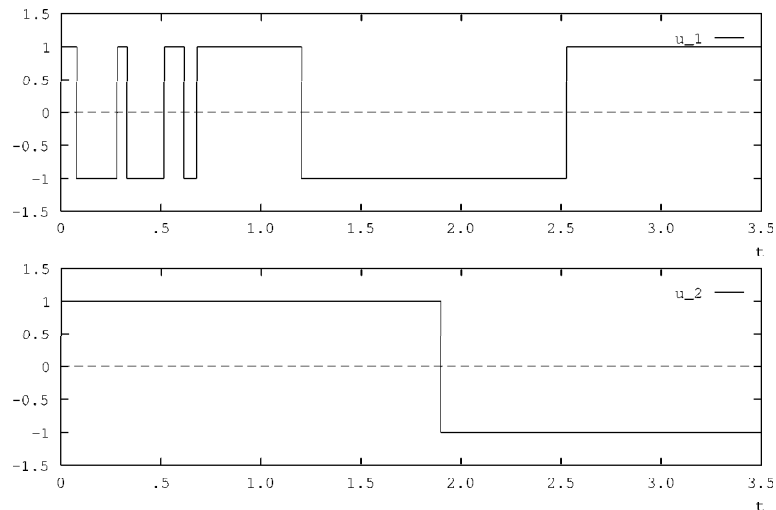


Figure 5: Optimal controls in the free environment

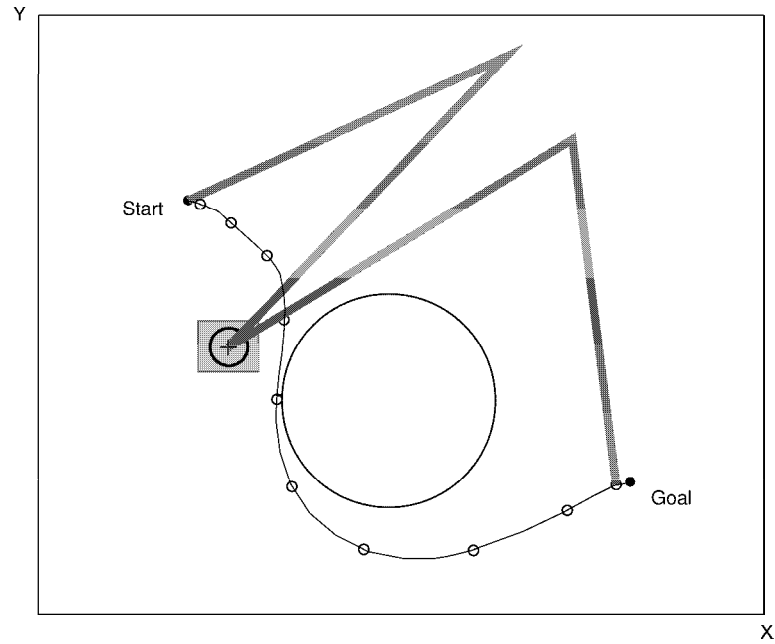


Figure 6: Optimal trajectory with a static obstacle

torques for this solution are shown in Figure 5. For this case, the second joint has one switch, whereas the first joint has two switches and a possible singular arc (multiple switches) near the start point. This singular arc may be explained by the smaller angular rotation of the first joint compared to the rotation of the second joint. This solution closely satisfies the necessary conditions of optimality, and is similar to the solution computed by the parameter optimization presented in [35]. The optimal time for this case is 3.59 s.

The second case considers a static obstacle, represented by a circle of radius $r = .4 m$,

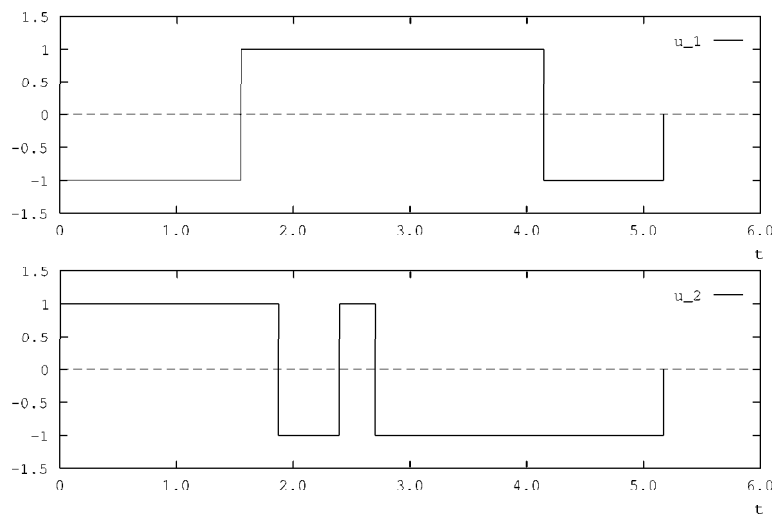


Figure 7: Optimal controls with a static obstacle

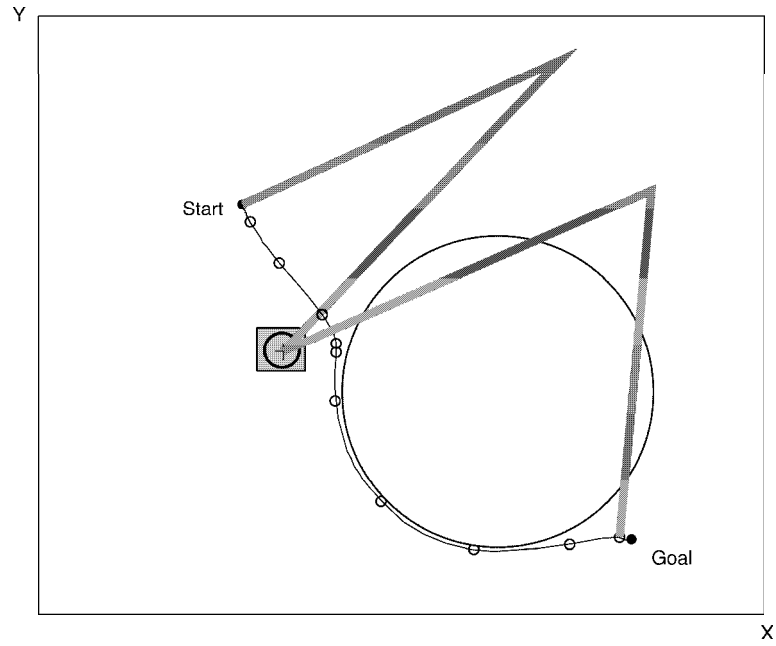


Figure 8: Optimal trajectory with a large static obstacle

centered at $\mathbf{C} = (.6 \text{ m}, -.2 \text{ m})$. The constraints Ψ_1 and Ψ_2 due to this obstacle are:

$$\Psi_1 : \begin{pmatrix} (x - x_o)^2 + (y - y_o)^2 - r^2 = 0 \\ (x - x_o)v_x + (y - y_o)v_y = 0 \end{pmatrix} \quad t = t_1$$

$$\Psi_2 : \quad v_x^2 + (x - x_o)a_x + v_y^2 + (y - y_o)a_y = 0 \quad t_1 \leq t \leq t_2$$

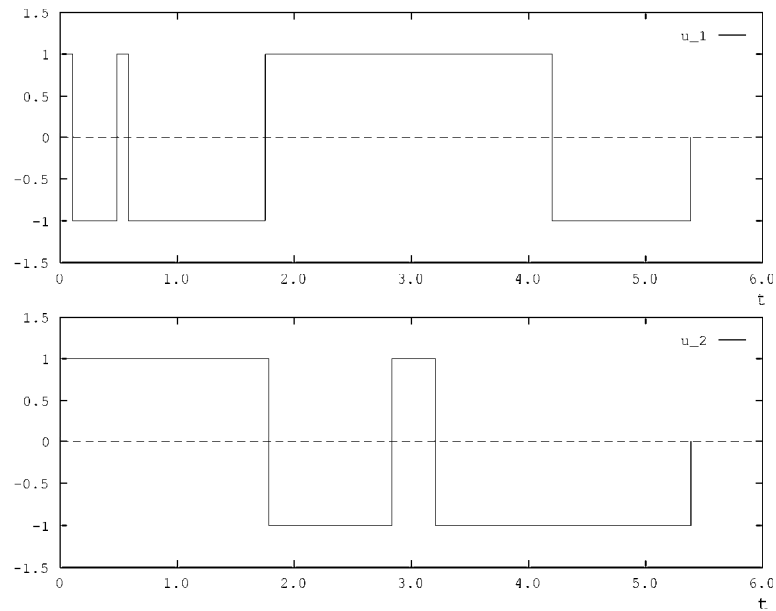


Figure 9: Optimal controls for a large static obstacle

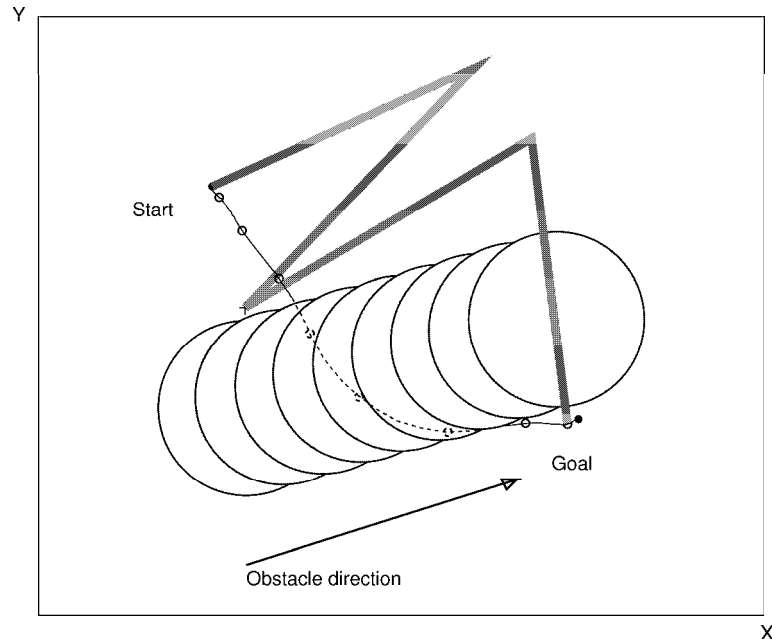


Figure 10: Optimal trajectory with a moving obstacle

The optimal path for this case is shown in Figure 6, and the actuator torques are shown in Figure 7. Here the path grazes the obstacle at one point, and does not follow the obstacle because of its high curvature. The optimal time for this case is 5.17 s .

This case was repeated with a larger obstacle, as shown in Figure 8, where the path follows the obstacle boundary. Here, the obstacle is of radius $r = .6\text{ m}$, located at $\mathbf{C} = (.8\text{ m}, -.15\text{ m})$. The optimal time for this case is 5.38 s , and the controls are shown in Figure 9.

Finally, the third case considers a moving obstacle, as shown in Figure 10. The constraints

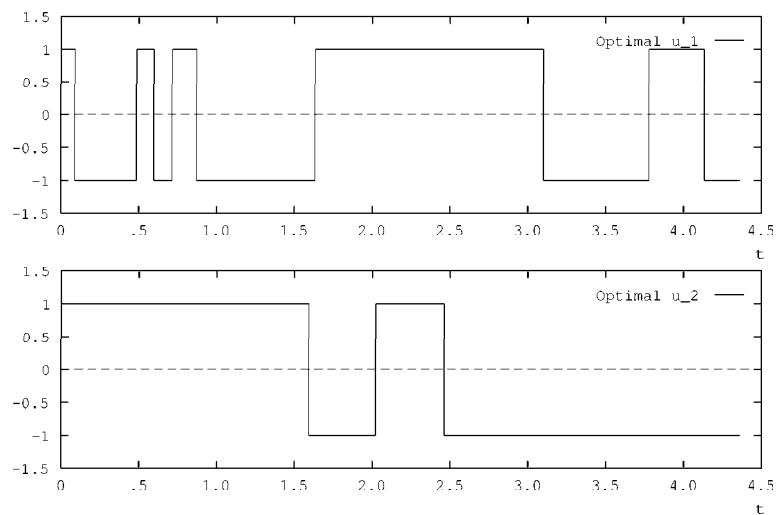


Figure 11: Optimal controls with a moving obstacle

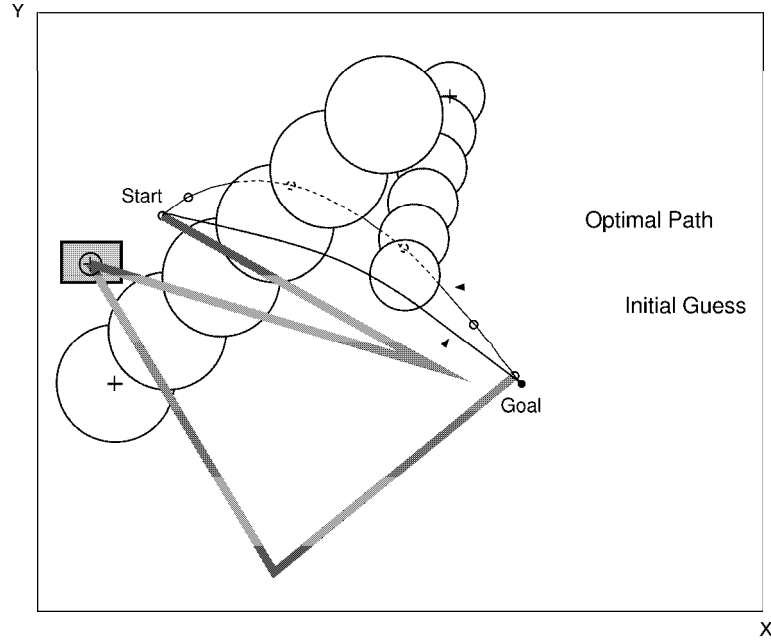


Figure 12: Optimal trajectory with two moving obstacles.

Ψ_1 and Ψ_2 are now:

$$\Psi_1 : \begin{pmatrix} (x - (v_{ox}t + x_o))^2 + (y - (v_{oy}t + y_o))^2 - r^2 = 0 \\ (x - (v_{ox}t + x_o))(v_x - v_{ox}) \\ + (y - (v_{oy}t + y_o))(v_y - v_{oy}) = 0 \end{pmatrix} t = t_1$$

$$\Psi_2 : \begin{pmatrix} ((v_x - v_{ox})^2 + (x - (v_{ox}t + x_o))a_x + \\ ((v_y - v_{oy})^2 + (y - (v_{oy}t + y_o))a_y = 0 \end{pmatrix} t_1 \leq t \leq t_2$$

The optimal path for this case, shown in Figure 10, slides along the moving obstacle. The actuator torques for this case are shown in Figure 11. The motion time for this case is $t = 4.36$ s, which is longer than the unconstrained time, but shorter than the time with a fixed obstacle.

4.2. Multiple Obstacles

In this example, the optimal trajectory is computed for two moving obstacles, using the SCARA manipulator as in the previous examples. The obstacles are moving at constant velocities: obstacle 1 at $(.045, .045)$ m/s and obstacle 2 at $(-.007, -.03)$ m/s, starting at time t_0 from the positions $(.1, -.5)$ m and $(1.15, .7)$ m, respectively. The end-effector starts at rest from $(.3, .2)$ m, and ends at rest at $(1.5, -.5)$ m.

The initial guess for this case is shown in Figure 12, with the motion time of 4.81 s. The bang-bang controls approximated for this trajectory are shown in Figure 13. Optimizing from this initial guess resulted in the path shown in Figure 12, and the actuator torques shown in Figure 14. The optimal motion time for this case is 2.6 s.

The improvement in motion time of the optimal trajectory compared to the initial guess is due to the fact that avoiding the velocity obstacles produces conservative trajectories, i.e. trajectories consisting of velocity segments that are guaranteed to avoid both obstacles at *all* times [10]. For this reason, the initial guess passes both obstacles from behind. The optimal trajectory, on the other hand, passes both obstacles from the front, which explains the significant reduction in motion time.

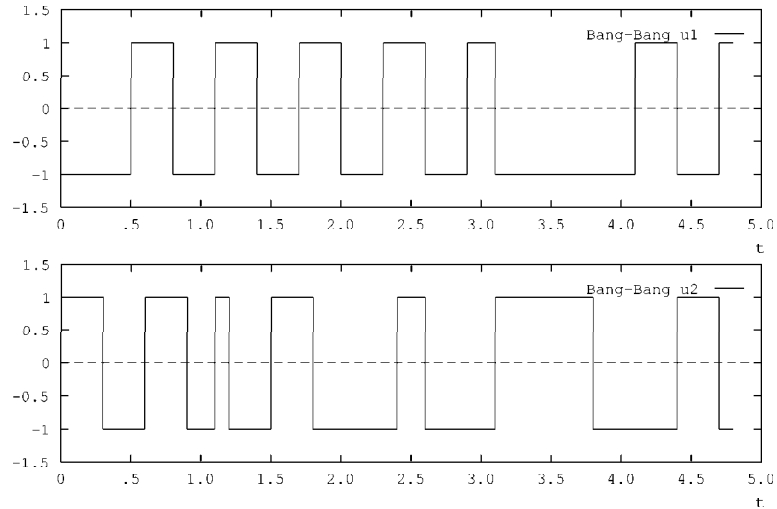


Figure 13: Bang-bang controls for the initial guess shown in Figure 12.

5. Summary

This paper presented a method for computing the time-optimal trajectories of a manipulator moving in dynamic environments, subject to system dynamics and actuator constraints. Formulating the problem as a time-minimization, the state inequality constraints due to the moving obstacles are transformed to state-dependent control constraints and a tangency point constraint at the entry point of the constrained arc. Assuming bang-bang controls, this optimization problem is solved numerically as a parameter optimization over the switching times and the final time, using a steepest descent algorithm. The initial guess for the optimization is computed using the previously developed concept of the *Velocity Obstacle* [9]. The velocity obstacles allow one to select an initial guess that has a desirable structure, i.e. a desirable sequence of avoidance and a desirable side from which each obstacle should be avoided. The

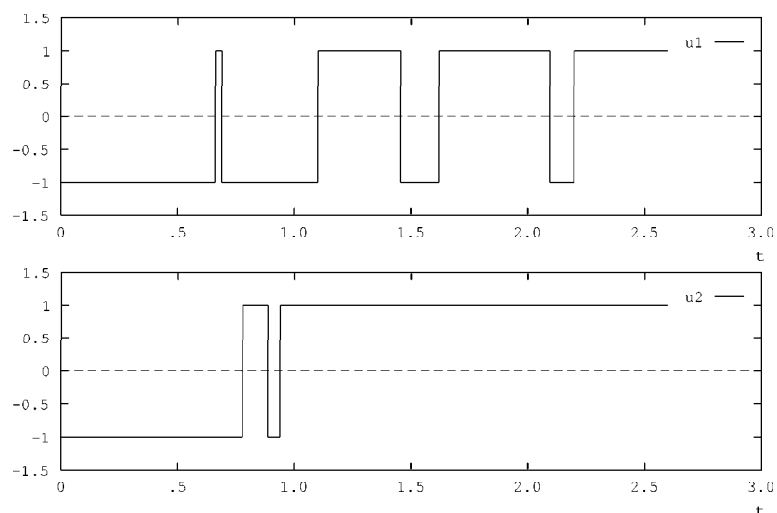


Figure 14: Bang-bang controls for the optimal path shown in Figure 12.

method is demonstrated in several examples for a 2 DOF planar manipulator moving amongst static and moving circular obstacles.

This method is meant for off-line computations, and is thus applicable to repetitive tasks, such as manipulators operating between moving conveyor belts, or manipulators operating off moving platforms. A more efficient method for on-line planning (with no guarantee of optimality) in dynamic environments has been presented in [11].

6. Acknowledgment

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

REFERENCES

1. J.E. Bobrow. Optimal robot path planning using the minimum time criterion. *IEEE Transaction on Robotics and Automation*, 4(4):443–450, 1988.
2. A.E. Bryson. *Dynamic Optimization*. Stanford University, Palo Alto, CA, March 23 1992.
3. A.E. Bryson. Inverse Dynamic Optimization. MANE Seminar, UCLA, 1995.
4. A.E. Bryson and S.E. Denham, W.F. and Dreyfus. Optimal programming problems with inequality constraints *i*: Necessary conditions for extremal solutions. *AIAA Journal*, 1(11):2544–2550, November 1963.
5. 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.
6. A.E. Bryson and Y.C. Ho. *Applied Optimal Control*. Hemisphere Publishing Corp., New York, NY, 1975.
7. 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.
8. M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, (2):477–521, 1987.
9. P. Fiorini. *Robot Motion Planning among Moving Obstacles*. PhD thesis, University of California, Los Angeles, January 1995.
10. 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.
11. P. Fiorini and Z. Shiller. Robot motion planning in dynamic environments. In G. Girald and G. Hirzinger, editors, *International Symposium of Robotic Research*, pages 237–248, Munich, Germany, 20-24 October 1995. Springer-Verlag.
12. P. Fiorini and Z. Shiller. Time optimal trajectory planning in dynamic environments. In *IEEE International Conference of Automation and Robotics*, volume 2, pages 1553–1558, Minneapolis, MN, 22-28 April 1996.
13. J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics*. Addison-Wesley Publishing Company, Reading, MA, 1990.
14. T. Fraichard. Dynamic trajectory planning with dynamic constraints: a state-time space approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1393–1400, Yokohama, Japan, 26-30 July 1993.
15. T. Fraichard and C Laugier. Path-velocity decomposition revisited and applied to dynamic trajectory planning. In *IEEE International Conference of Automation and Robotics*, volume 1, pages 40–45, Atlanta, GA, 2-6 May 1993.
16. K. Fujimura. Motion planning amid transient obstacles. *International Journal of Robotics Research*, 13(5):395–407, October 1994.
17. K. Fujimura. Time-minimum routes in time-dependent networks. *IEEE Transaction on Robotics and Automation*, 11(3):343–351, June 1995.
18. K. Fujimura and H. Samet. A hierarchical strategy for path planning among moving obstacles. *IEEE Transaction on Robotics and Automation*, 5(1):61–69, February 1989.
19. K. Fujimura and H. Samet. Time-minimal paths among moving obstacles. In *IEEE International Conference on Robotics and Automation*, pages 1110–1115, Scottsdale, AZ, May 1989.
20. 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.
21. D.H. Jacobson, M.M. Lele, and J.L. Speyer. New necessary conditions of optimality for con-

- control problems with state-variable inequality constraints. *Journal of Mathematical Analysis and Applications*, 35(2):255–284, 1971.
22. 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.
 23. B. Kahn, M.E. and Roth. The near-minimum time control of open loop kinematic chains. *Transactions of ASME*, Series G(93):164–172, 1971.
 24. 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.
 25. J.C. Latombe, editor. *Robot Motion Planning*. Kluwer Academic Publisher, Boston, MA, 1991.
 26. J.P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In L.O. Hertzberger and F.C.A. Groen, editors, *Intelligent Autonomous Systems*, pages 346–354, New York, NY, 1987. North Holland.
 27. J.P. Laumond, P.E. Jacobs, M. Taix, and Murray R.M. Motion planner for non-holonomic mobile robots. *IEEE Transactions on Robotics and Automation*, RA-10(5):573–593, 1994.
 28. 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):21–32, January-February 1987.
 29. T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.
 30. E.B. Meier. *An Efficient Algorithm for Bang-Bang Control Systems Applied to a Two-Link Manipulator*. PhD thesis, Stanford, December 1987.
 31. R.M. Murray and S.S. Sastry. Nonholonomic motion planning: steering with sinusoids. *IEEE Transactions on Automatic Control*, AC-38(5):700–716, 1993.
 32. J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *25th IEEE Symposium on the Foundation of Computer Science*, pages 144–154, 1985.
 33. H. Seywald. Trajectory optimization based on differential inclusion. *AIAA Journal of Guidance, Control and Dynamics*, 17(3), 1994.
 34. Z. Shiller. Time-energy optimal control of articulated systems with geometric path constraints. *ASME Journal of Dynamic Systems, Measurements and Control*, 118(1):139–143, March 1996.
 35. Z. Shiller and S. Dubowsky. Robot path planning with obstacles, actuators, gripper and payload constraints. *The International Journal of Robotics Research*, 8(6):3–18, December 1989.
 36. Z. Shiller and P. Fiorini. Autonomous negotiation of freeway traffic. In A.C. Chachich, editor, *SPIE Conference on Collision Avoidance and Traffic Management Sensors*, volume 2592, pages 83–92, Philadelphia, PA, October 1995.
 37. Z. Shiller and R.Y. Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions of Robotics and Automation*, 7(2):241–249, April 1991.
 38. Z. Shiller and S. Sundar. Emergency maneuvers of autonomous vehicles. In *The 13th World Congress, IFAC'96*, volume Q, pages 393–398, San Francisco, CA, July 1996.
 39. J. Speyer and A.E. Bryson. Optimal programming problems with a bounded state space. *AIAA Journal*, 6(8):1488–1491, August 1968.
 40. A. Weinreb and A.E. Bryson. Optimal control of systems with hard control bounds. *IEEE Transactions on Automatic Control*, AC-30(6), November 1985.

Appendix

A. Derivation of the Terminal Differential

The differential of the terminal constraint Ω can be computed using [5]:

$$(d\Omega)_{t_f} = \left(\frac{\partial\Omega}{\partial x} dx + \frac{\partial\Omega}{\partial t} dt \right)_{t_f} \quad (44)$$

Using $dx = \delta x + \dot{x}dt$ it follows that

$$(d\Omega)_{t_f} = (\delta\Omega)_{t_f} + \dot{\Omega}_{t_f} dt_f \quad (45)$$

The variation δx satisfies the first order perturbation equation:

$$\delta\dot{x} = \frac{\partial\mathcal{F}}{\partial x} \delta x + \frac{\partial\mathcal{F}}{\partial u} \delta u \quad (46)$$

Therefore, there exists a state transition matrix $\Phi(t, \tau)$ expressing the variation $(\delta x)_{t_f}$ [6]. The variation $\delta\Omega$ is then:

$$(\delta\Omega)_{t_f} = \left. \frac{\partial\Omega}{\partial x} \right|_{t_f} \left(\Phi(t_f, t_0) \delta x(t_0) + \int_{t_0}^{t_f} \Phi(t_f, \tau) \frac{\partial\mathcal{F}}{\partial u} \delta u(\tau) d\tau \right) \quad (47)$$

This expression can be simplified by defining a multiplier $\lambda_\Omega \in \mathfrak{R}^n \times \mathfrak{R}^l$ as:

$$\lambda_\Omega^T(t) = \left(\frac{\partial\Omega}{\partial x} \right)_{t_f} \Phi(t_f, t) \quad (48)$$

where n is the dimension of \mathbf{x} and l is the number of terminal constraints. Taking advantage of the properties of the state transition matrix Φ [6], a set of adjoint equations for λ_Ω can be written as:

$$\dot{\lambda}_\Omega(t) = - \left(\frac{\partial\mathcal{F}}{\partial x} \right)^T \lambda_\Omega(t) \quad \lambda_\Omega(t_f) = \left(\frac{\partial\Omega}{\partial x} \right)_{t_f} \quad (50)$$

Therefore, using

$$(\delta\Omega)_{t_f} = \int_{t_0}^{t_f} \lambda_\Omega^T \frac{\partial\mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \delta\Omega_0 \quad (51)$$

in equation (45), and assuming fixed initial conditions, the total differential of Ω becomes:

$$d\Omega(t_f) = \int_{t_0}^{t_f} \lambda_\Omega^T \frac{\partial\mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \left(\frac{\partial\Omega}{\partial x} \dot{x} + \frac{\partial\Omega}{\partial t} \right)_{t_f} dt_f \quad (52)$$

B. Effect of the Point Constraint on the Multipliers

The co-state equations for λ_{phi} , λ_Ω used in the previous Sections do not take into account the effects of the constraints Ψ_1 and Ψ_2 given by:

$$\mathbf{\Lambda}_{t_1^-}^T = \mathbf{\Lambda}_{t_1^+}^T + \eta^T \frac{\partial \Psi_1}{\partial \mathbf{x}(t_1)} \quad (53)$$

This discontinuity affects the co-state equations, as illustrated in the following using multipliers λ_Ω [7].

The unknown η is computed by relating the value of λ_Ω at t_1^- , i.e. just before reaching the constraint Ψ_1 , to its value at t_1^+ , i.e. just after reaching Ψ_1 . To do this, $\lambda_\Omega(t_1^+)$ and $\lambda_\Omega(t_1^-)$ are first computed independently of each other, using the expressions for $d\Omega$ at t_f and t_1 .

The value of $\lambda_\Omega(t_1^+)$ is computed from the expression of the changes in $d\Omega(t_f)$ due to the variation of \mathbf{x} , $\delta\mathbf{x}(t_1^+)$:

$$d\Omega(t_f) = \lambda_\Omega^T \delta\mathbf{x}(t_1^+) = \lambda_\Omega^T (d\mathbf{x} - \dot{\mathbf{x}} dt_1)_{t_1^+} \quad (54)$$

that can be rewritten as:

$$d\Omega(t_f) = \left(\frac{\partial \Omega}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial \Omega}{\partial t} dt \right)_{t_1^+} \quad (55)$$

from which:

$$\frac{\partial \Omega}{\partial \mathbf{x}_1} = \lambda_\Omega^T(t_1^+) \quad (56)$$

$$\frac{\partial \Omega}{\partial t_1} = -\lambda_\Omega^T(t_1^+) \dot{\mathbf{x}}(t_1^+) \quad (57)$$

The value of $d\Omega$ at t_1^- is computed using:

$$d\Omega(t_1^-) = (\delta\Omega)_{t_1^-} + \dot{\Omega}_{t_1^-} dt_1^- \quad (58)$$

where:

$$(\delta\Omega)_{t_1^-} = (\lambda_\Omega \delta\mathbf{x})_{t_0} + \int_{t_0}^{t_1^-} \lambda_\Omega^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta\mathbf{u} d\tau$$

Since $dS^{(p-1)}(\mathbf{x}) = 0$, the value of dt_1 is:

$$dt_1 = \frac{1}{\dot{S}^{(p-1)}} \left[-(\lambda_{S^{(p-1)}}^T \delta\mathbf{x})_{t_0} - \int_{t_0}^{t_1} \lambda_{S^{(p-1)}}^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta\mathbf{u} dt \right] \quad (59)$$

By replacing dt_1 in $d\Omega$ of equation (58) with (59), and since $\dot{S}^{(p-1)}$ and $\dot{\Omega}$ are both independent of the integration variable, $d\Omega(t_1^-)$ becomes:

$$d\Omega(t_1^-) = (\lambda_\Omega^T \delta\mathbf{x})_{t_0} + \int_{t_0}^{t_1^-} \left(\lambda_\Omega^T - \frac{\dot{\Omega}}{\dot{S}^{(p-1)}} \lambda_{S^{(p-1)}}^T \right) \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta\mathbf{u} d\tau - \frac{\dot{\Omega}}{\dot{S}^{(p-1)}} \Big|_{t_1^-} (\lambda_{S^{(p-1)}}^T \delta\mathbf{x})_{t_0} \quad (60)$$

The desired expression of λ_Ω at t_1^- , satisfying $d(S^{(p-1)}) = 0$ is then:

$$\lambda_{\Omega, S^{(p-1)}}^T(t_1^-) = \left(\lambda_\Omega^T - \frac{\dot{\Omega}}{\dot{S}^{(p-1)}} \lambda_{S^{(p-1)}}^T \right)_{t_1^-} \quad (61)$$

This equation can be further simplified by replacing $\dot{\Omega}(t_{-1})$ with:

$$\dot{\Omega}(t_1^-) = \left. \frac{\partial \Omega}{\partial \mathbf{x}} \right|_{t_1^-} \dot{\mathbf{x}}(t_1^-) - \lambda_{\Omega}^T(t_1^+) \dot{\mathbf{x}}(t_1^+) \quad (62)$$

Since the differentials $d\mathbf{x}(t_1)$ and dt_1 are the same at t_1^- and t_1^+ , equation (56) gives:

$$\left. \frac{\partial \Omega}{\partial \mathbf{x}} \right|_{t_1^-} = \lambda_{\Omega}^T(t_1^+) \quad (63)$$

and similarly

$$\lambda_{S^{(p-1)}}^T(t_1^-) = \left. \frac{\partial S^{(p-1)}}{\partial \mathbf{x}} \right|_{t_1} \quad (64)$$

By using equations (63), (64), and (62) in (61), the discontinuity of λ_{Ω} at t_1 becomes:

$$\lambda_{\Omega, S^{(p-1)}}^T(t_1^-) = \lambda_{\Omega}^T(t_1^+) \left(I - \frac{\dot{\mathbf{x}}(t_1^-) - \dot{\mathbf{x}}(t_1^+)}{S^{(p)}(t_1)} \left. \frac{\partial S^{(p-1)}}{\partial \mathbf{x}} \right|_{t_1} \right) \quad (65)$$

which is equivalent to the necessary condition (53) if the multiplier η is equal to:

$$\eta^T = -\lambda_{\Omega}^T(t_1^+) \left(\frac{\dot{\mathbf{x}}(t_1^-) - \dot{\mathbf{x}}(t_1^+)}{S^{(p)}(t_1^-)} \right) \quad (66)$$