The Dissertation Committee for Shilpa Gulati
certifies that this is the approved version of the following dissertation:

A Framework for Characterization and Planning of Safe, Comfortable, and Customizable Motion of Assistive Mobile Robots

Committee:

Benjamin J. Kuipers, Supervisor

Raul G. Longoria, Supervisor

Dragan Djurdjanovic

S.V. Sreenivasan

Peter Stone

A Framework for Characterization and Planning of Safe, Comfortable, and Customizable Motion of Assistive Mobile Robots

by

Shilpa Gulati, B.Tech., M.Tech.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2011

A Framework for Characterization and Planning of Safe, Comfortable, and Customizable Motion of Assistive Mobile Robots

Publication No. ___________________

Shilpa Gulati, Ph.D.
The University of Texas at Austin, 2011

Supervisors: Benjamin J. Kuipers
Raul G. Longoria

Assistive mobile robots, such as intelligent wheelchairs, that can navigate autonomously in response to high level commands from a user can greatly help people with cognitive and physical disabilities by increasing their mobility. In this work, we address the problem of safe, comfortable, and customizable motion planning of such assistive mobile robots.

We recognize that for an assistive robot to be acceptable to human users, its motion should be safe and comfortable. Further, different users should be able to customize the motion according to their comfort. We formalize the notion of motion comfort as a discomfort measure that can be minimized to compute comfortable trajectories, and identify several properties that a trajectory must have for the motion to be comfortable. We develop a motion planning framework for planning safe, comfortable, and customizable trajectories in small-scale space. This framework removes the limitations of existing methods for planning motion of a wheeled mobile robot moving on a plane, none of which can compute trajectories with all the properties necessary for comfort.

We formulate a discomfort cost functional as a weighted sum of total travel time, time integral of squared tangential jerk, and time integral of squared normal jerk. We then define the problem of safe and comfortable motion planning as that of minimizing this discomfort such that the trajectories satisfy boundary conditions on configuration and its higher derivatives, avoid obstacles, and satisfy constraints on curvature, speed, and acceleration. This description is transformed into a precise mathematical problem statement using a general nonlinear constrained optimization approach. The main idea is to formulate a well-posed infinite-dimensional optimization problem and use a conforming finite-element discretization to transform it into a finite-dimensional problem for a numerical solution.

We also outline a method by which a user may customize the motion and present some guidelines for conducting human user studies to validate and/or refine the discomfort measure presented in this work.

Results show that our framework is capable of reliably planning trajectories that have all the properties necessary for comfort. We believe that our work is an important first step in developing autonomous assistive robots that are acceptable to human users.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mobility impairments result in functional limitations and a reduced sense of physical and emotional well-being for many people [34, 72]. Recent surveys have concluded that many users with mobility impairments find it difficult or impossible to operate existing power wheelchairs because they lack the necessary motor skills or cognitive abilities [18, 85]. Assistive mobile robots such as smart wheelchairs and scooters that can navigate autonomously can enormously benefit such users by increasing their mobility [18]. The number of users who would benefit from at least part-time use of assistive mobile robots was estimated to be between 2.6 to 3.9 million for 2010 [85].

Significant advances have been made toward solving scientific and technical problems crucial for developing assistive mobile robots [84]. From a hardware point of view, this includes the development of suitable standards for evaluation of wheelchair platforms and design of platforms that meet these standards [75]. From an autonomous agent point of view, this involves research into sensory inference (perception) [93], representation of spatial knowledge [47], spatial inference [7, 92], autonomous navigation [83], and human-robot interaction [89, 99].

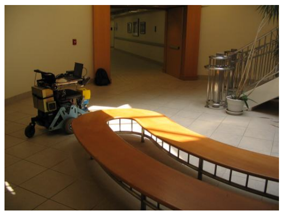
This research focuses on autonomous navigation in small-scale space, that is, space within the robot's sensory horizon [48]. Specifically, this research focuses on safe, comfortable, and customizable motion planning of an assistive mobile robot in small-scale space.

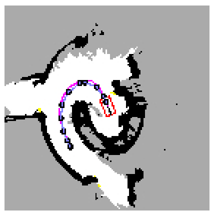
1.1 Background and Motivation

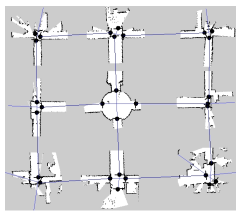
One of the long-term objectives of research in autonomous navigation of assistive robots is to develop methods that will enable a robot to navigate autonomously in response to high-level commands from a user (e.g. "take me to my office"). To navigate autonomously in response to such commands, a robot should be able to construct a representation of space and should be able to use this representation to compute the appropriate motions.

The Hybrid Spatial Semantic Hierarchy (HSSH) [6, 7, 47, 48], a hierarchical framework for representing spatial knowledge, draws a distinction between small-scale and large-scale space. Small-scale space is space whose structure is within the robot's sensory horizon. Large-scale space is space whose structure is beyond the sensory horizon. There exist many methods that enable a robot to efficiently construct and update a metrically accurate map of small-scale space as it moves [93]. This map provides information about free space and occupied and hazardous regions, and can be used to plan local motions when the goal is within the map region.

While many algorithms have been proposed for accurate metrical mapping of large-scale space, they suffer from several structural ambiguity prob-


(a)


(b)


(c)

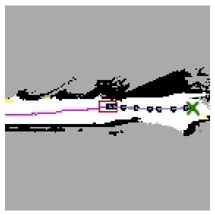

(d)

Figure 1.1: Autonomous navigation in small-scale and large-scale space. (a) A wheelchair in an indoor environment. (b) For navigation in small-scale space, a goal configuration is provided by a user or a high-level planner. A locally accurate metrical map constructed from the wheelchair's sensory data forms a representation of small-scale space. This map shows accessible regions (in white) and obstacles (in black). A motion planning algorithm uses this representation to compute local motions of the wheelchair. (c) A global topological map consisting of places and paths is constructed from a set of local maps. (d) Autonomous navigation in large-scale space is achieved by providing a sequence of intermediate states such that each subsequent state is within the robot's current sensory horizon. The waypoint corresponding to one such state is shown (the cross). The motion planning algorithm of (b) plans a trajectory to reach this state. A sequence of such local trajectories leads the wheelchair to its eventual goal. Figures (a), (b), and (d) borrowed from [67]. Figure (c) borrowed from [48].

lems [7, 48]. An alternative way to represent large-scale space is in the form of topological maps. These maps concisely represent large-scale space as a graph of places and paths [7]. Autonomous navigation in large-scale space involves moving from one place to another via a path. This can be achieved by first computing a sequence of intermediate states such that each subsequent position is within the robot's sensory horizon, and then using a local motion planning method to navigate to this intermediate state. See Figure 1.1.

In this work, we are interested in planning local motions in small-scale space. Motion planning in small-scale space is the problem of determining the appropriate motions of a robot, given a representation of the world, so that it arrives at a goal within its sensory horizon while satisfying task requirements.

Motion planning is a challenging problem and has received significant attention. Many practical methods for planning motion of mobile robots and manipulator arms can be found in literature [14, 52, 57]. However, most of the existing motion planning methods have been developed for autonomous robots that do not perform any assistive function. Hence, many issues pertinent to assistive robots that transport a human user have not been considered. We recognize that one of the issues that determine the human acceptability of an assistive robot is its ability to plan motions that are comfortable for the user. This issue becomes even more critical for people with serious injuries, such as those with spinal chord injuries. Moreover, since the notion of comfort is subjective, any motion planning method for assistive robots should allow different users to customize the motion according to their comfort.

1.1.1 Comfort

Comfort - What is it? Comfort has both psychological and physiological components, but it involves a sense of subjective well-being and the absence of discomfort, stress or pain [76].

Studies have shown that comfort is one of the factors that influences choice of mode of transportation [69] and determines acceptability of a mode of transportation for human users [15]. Hence, many studies have been conducted to characterize comfort in ground vehicles including automobiles and trains. The feeling of comfort in a vehicle is affected by various characteristics of the vehicle environment including dynamic factors (such as acceleration and jerk), ambient factors (such as temperature and air quality), and spatial factors (such as seat quality and leg room) [76].

Although many other factors can contribute to the feeling of comfort for a human user of an assistive robot, in this work we focus on comfort due to dynamic factors alone. Here we briefly discuss what comfort means in terms of dynamic factors. For a fuller discussion, see Section 3.3. Studies of passenger discomfort in automobiles and trains have shown that discomfort increases as the magnitudes of acceleration and jerk increase [13, 24, 37, 70, 91]. Various comfort measures are used in industry, but almost all are either functions of acceleration or jerk (or both) or prescribe maximum permissible values of these quantities [12, 24, 37, 70]. This makes intuitive sense since acceleration is proportional to force, and bounded acceleration implies bounded force. A high rate of change of acceleration (jerk) means that forces rapidly change in magnitude or direction or both.

Thus, smooth and bounded acceleration is necessary for comfort. Further, for planning in large-scale space, we must be able to join one motion sequence in small-scale space to the next without causing discomfort. This means that we must ensure continuity of acceleration at the joining of two motion sequences. This is possible only when the planned motion attains a specified velocity and acceleration at its start and end. Additionally, for assistive robots, the robot should reach a specified goal position and orientation exactly. This is important for many tasks such as docking at a desk. It is also desirable that the robot travel from its initial state to a goal state without stopping in between. This means that the geometric path of the robot should necessarily have curvature continuity. The planned path should also have a specified curvature at the ends so that, when two paths in small-scale space are joined, the combined path retains curvature continuity. It is also desirable that the motion of the robot is such that the passenger faces the direction of motion most of the time.

Thus, any motion planning method for assistive mobile robots should, at the very least, produce motion that attains a specified velocity and acceleration at the start and end point, has smooth and bounded acceleration, and results in geometric paths that avoid obstacles, have curvature continuity, and attain a specified curvature at the ends. While many of the existing motion planning algorithms developed for mobile have been applied to navigation of intelligent wheelchairs [30] in an experimental setting, we will see below in Section 1.1.2 that all of these have limitations that preclude their straightforward application to planning comfortable motion.

1.1.2 Motion Planning

There exists a large body of work on robot motion planning. Before reviewing this work, we define some terms. The space of all possible positions and orientations of a robot is called *configuration space.* The space of all possible positions and orientations and their first derivatives is called *state space*. The space of input velocities (or forces or torques, as appropriate) is called *control space*. The physical space in which the robot moves is called *task space*. A *trajectory* is a function that describes the configuration of the robot at every instant of time during the robot's motion. A *control trajectory* is a function that describes the control inputs at every instant of time during the robot's motion. The differential equation relating the robot's configuration, its first derivatives, and control inputs, is called the *kinematic model* of the robot.

Motion planning is the problem of finding either a trajectory, or a control trajectory, or both, given the initial and final configuration, and possibly both initial and final velocity and acceleration, such that the geometric path of the robot does not intersect any obstacles in its task-space, and its trajectory satisfies *kinematic* and *dynamic constraints*. Kinematic constraints refer to constraints on configuration and dynamic constraints refer to constraints on velocity and its higher derivatives). These constraints arise from physics, engineering limitations, or comfort requirements.

This problem has received attention from multiple directions. One set of methods, called *path planning* methods, treat robot motion planning as a purely geometric problem and disregard dynamics. Another set of methods, stemming from differential geometric control theory, focus on controllability issues. The primary focus of these methods has been on computing control inputs that steer a robot to a specified position and orientation or that make a robot follow a specified path. These methods usually do not consider obstacle avoidance. Another set of methods, referred to as *kinodynamic motion planning* methods, consider both dynamics and obstacles. See [14, 52, 57] for excellent presentation of all three kinds of methods, [54] for differential geometric control methods, and [16, 25, 33, 59] for kinodynamic planning.

Motion planning has been variously defined in literature and is sometimes used interchangeably with the purely geometric problem of path planning. In this work, we use motion planning in the sense of [16], that is, we speak of kinodynamic motion planning.

Broadly speaking, there are two major paradigms of motion planning. One paradigm is to decouple motion planning into path planning and velocity planning [57]. First, dynamic constraints are ignored and a geometric collision-free path is computed (path planning). Second, the path is transformed into a new path such that it is possible for the robot to follow the path without violating dynamic constraints (path transformation) [19, 78]. Third, a velocity on the path is computed such that dynamic constraints are satisfied [9], some performance measure is optimized [9] or moving obstacles are avoided [39] (velocity planning).

One of the advantages of using such a decoupled approach is that any of the existing efficient path planning algorithms can be used in the path planning step. However, since dynamic constraints are ignored in the path planning phase, a collision-free path computed by a path planning algorithm may not be dynamically realizable by the robot. In the next step, when the collision free path is transformed into a path that satisfies dynamic constraints, the transformed path may result in collisions with obstacles. Hence, it is often necessary to also consider obstacle avoidance at this stage. This usually requires an iterative procedure where the path is transformed using some heuristic, the transformed path is checked for collisions, and if there are collisions, another transformation is carried out [53]. Further, this decoupling makes it difficult to achieve optimality of the trajectory with respect to a performance measure because it is non-trivial to design performance measures for each step such that when these are individually optimized, the resulting trajectory optimizes the overall performance measure.

The other motion planning paradigm is to plan trajectories in one step while considering all dynamic constraints and avoiding obstacles. We will refer to these methods as direct trajectory planning methods. Two approaches that have been found to be of practical are sampling-based methods and optimization-based methods.

Sampling-based methods find a trajectory by sampling the state space [16, 59] or state-time space [25, 33] to iteratively construct a search graph rooted at the initial state. Earlier methods [16, 25] first discretize the space into a grid. The set of admissible controls is also discretized and a fixed time interval is chosen. The differential equations of the system are integrated over the chosen time interval for each of the discretized controls, starting at the initial state, to obtain a set of reachable states. Each trajectory segment connecting the initial state to a newly reached state is checked for collisions with obstacles. If the trajectory segment is collision free, the grid cell that contains the new state is determined and marked visited. This grid cell is not visited again. The new state is added to the tree as a vertex and the trajectory segment is added as an edge. This process of generating and adding new states is repeated for all the newly added vertices. The search graph continues to grow till a state that is within a specified tolerance of the goal is reached. A graph search algorithm is then used to find a trajectory from the initial state to the goal state such that the trajectory is optimal with respect to some criterion such as minimum-time. These grid-based approaches become computationally expensive as the dimension of the state space increases (for example, in the case of manipulator arms). To alleviate this problem, methods that construct a search graph by randomized sampling of state space [59] or state-time space [33] were developed and have found many practical applications. Randomized sampling methods, however, sacrifice optimality for efficiency and find feasible but non-optimal paths.

Since all of the sampling-based methods discretize the time, the state space or the control space or both, the accuracy of reaching a goal increases as the resolution of discretization is increased. A goal state cannot be reached exactly in finite time. If it is desired to reach a goal state exactly, then a two-point boundary value problem has to be solved where the goal state reached by the algorithm is the initial state and the exact goal state is the final state. This is a non-trivial problem since the solution must avoid obstacles and satisfy dynamic constraints. Since a fixed value of control input is applied for a finite length of time to compute a trajectory segment, the paths planned by these methods for wheeled robots usually consist of a sequence of straight-line segments and circular arcs. These paths lack curvature continuity and have to be smoothed in a post-processing step so that they can be followed by a wheeled robot without frequent stopping and reorienting wheels.

In general, there exist infinitely many trajectories that satisfy boundary conditions at the start and end points and possess properties such as satisfying dynamic constraints and avoiding obstacles. Optimal control methods have traditionally been used for computing the "best" trajectory for systems subject to dynamic constraints [11, 96]. These methods find a trajectory that minimizes a cost functional. A *functional* is an operator that maps a function to a real or complex number. These methods either solve the first order differential equations derived from the cost functional using Pontryagin's maximum principle, or use various approximations methods such as the Ritz method or the Finite Element Method to directly minimize the cost functional in a finite-dimensional space.

Optimal control formulations for trajectory planning of wheeled mobile robots have focused primarily on minimum-time trajectories. These trajectories result in geometric paths that consist of a sequence of straight-line and arc segments [4, 17, 74]. These paths do not have curvature continuity and the robot cannot be driven on these paths without stopping and changing orientation while stopped. Thus, minimum-time trajectories, while important from a theoretical perspective, are of little practical use for assistive robots where comfort is important. Several other cost functionals have been used for trajectory planning, such as the integral of L^2 norm of the controls [2, 21, 97], or a weighted sum of travel time and integral of L^2 norm of controls [79]. All of these formulations make several limiting assumptions, such as known travel time, or known path, or boundary conditions on configuration but not its derivatives. To the best of our knowledge, there exists no optimal control formulation of trajectory planning for mobile robots that has produced demonstrable results with obstacle avoidance constraints.

Thus, none of the existing motion planning approaches can plan trajectories that satisfy all the requirements for user comfort as described in Section 1.1.1.

1.2 Overview of Approach

We develop a motion planning framework that removes the limitations of existing work and plans safe, comfortable, and customizable motion for a wheeled mobile robot moving on a plane. The main idea is to characterize user discomfort in terms of dynamic properties such as jerk (time derivative of acceleration) of the robot, formulate this discomfort as a mathematically meaningful cost functional, and minimize this cost functional to find a trajectory. We incorporate dynamic and obstacle-avoidance constraints into the optimization problem and impose the appropriate boundary conditions. The steps involved in our approach are described below.

- *Formulate user discomfort as a mathematically meaningful cost functional.* Based on existing literature, and making the assumption that the user would like to reach the goal as fast as is consistent with comfort, we define a measure of discomfort as a weighted sum of the following three terms: total travel time, time integrals of squared tangential jerk and squared normal jerk.

  Each weight used in the discomfort measure to add different quantities is a product of two factors. The first factor has physical units so that the physical quantities with different dimensions can be added together. It is a fixed function of known length and velocity scales. The second factor is a dimensionless parameter that can be varied according to user preferences. The dimensional part is derived using the standard technique of dimensional analysis [51].

- *Define the problem.* We formulate our motion planning problem as follows: "Given the start and end boundary conditions on pose, speed and acceleration, the values of bounds on curvature, velocity and acceleration, the weights in the cost functional, and the locations of obstacles, find a trajectory that minimizes the user discomfort measure such that bounds are not violated and the geometric path does not pass through obstacles". This description is transformed into a precise mathematical problem statement using a general nonlinear constrained optimization approach.

- *Choose a parameterization of the trajectory.* Mathematically, one can use different functions to fully describe a trajectory. For example, one way is to provide the position vector as a function of time and the final time [28]. Another way is to represent the path separately, using either position vector or orientation as function of arc-length. The speed is provided separately. Both ways are equivalent in that all physical quantities can be written in terms of the selected primary unknowns. We have found that expressing the trajectory solely in terms of speed and orientation leads to a relatively simple expression for discomfort. We use a scaled arc-length parameterization where the scaling factor is an additional scalar unknown to be solved for. This is necessary since we don't know the arc-length until the problem is solved.

- *Analyze the boundary conditions.* A complete analysis of boundary conditions shows that for the optimization problem to be well-posed, we need to impose boundary conditions on position, orientation, curvature, speed, and tangential acceleration on each end. Further, we find that three different types of boundary conditions on velocity and acceleration on each end describe all types of motion tasks of interest such as starting/ending at rest or not.

- *Choose a representation of obstacles.* To incorporate obstacle avoidance, we make the assumption that each obstacle can be modeled as a star-shaped domain with a boundary that is a piecewise smooth curve with continuous second order derivative. If an obstacle within the map is not star-shaped, our framework can still handle it if it can be expressed as a finite union of piecewise smooth star-shaped domains. It is assumed that a representation of each obstacle is known in polar coordinates where the origin lies in the interior of the kernel of the star-shaped domain. Since each obstacle is assumed star-shaped, the constraint that the robot not collide with obstacles can be easily cast as an inequality.

For efficient incorporation of obstacle avoidance constraints, we have to introduce position on the path as an additional unknown. This leads to a sparse Hessian of constraint inequalities, which otherwise would be dense. The position as an unknown is redundant in that it can be computed from the two primary unknowns (orientation and speed). Hence that relation is included as an extra equality constraint.

- *Discretize the problem.* We use finite elements to convert the infinite-dimensional minimization problem to a finite dimensional one. For discomfort to be mathematically meaningful and bounded, both speed and orientation must have square-integrable second derivatives. We use a uniform mesh and cubic Hermite polynomial shape functions on each element for speed and orientation. Starting or stopping with zero velocity is a special case that requires that speed have an infinite derivative (with respect to scaled arc-length) with a known strength on the corresponding boundary point. In this case we use singular shape functions for speed only on elements adjacent to the corresponding boundary.

In the non-discretized version of the optimization problem the obstacle avoidance constraint can be expressed as the condition that each point on the trajectory should be outside each obstacle. We discretize this into a finite dimensional set of inequalities by requiring that some fixed number of points on the trajectory be outside each obstacle.

- *Compute a good initial guess.* A good initial guess is necessary for efficiently solving any nonlinear optimization problem. In general, there exist infinitely many paths between any given pair of start and end configuration. Based on our analysis of this non-uniqueness, we compute a set of four good quality initial guesses by solving another, simpler, optimization problem. These initial guesses do not incorporate obstacle-avoidance constraints. Four discomfort minimization problems, corresponding to these four initial guesses, are solved to find four trajectories. The lowest cost trajectory can be chosen as the final solution.

- *Implement and solve.* We use Ipopt, a robust large-scale nonlinear constrained optimization library [98] written in C++ to solve the discretized problem.

1.3 Contributions

To the best of our knowledge, this is the first work that addresses the problem of planning comfortable and customizable motion for assistive mobile robots. Our main contributions are described below.

- We recognize that for a robotic wheelchair to be acceptable to human users, its motion should not only be safe, it should also be comfortable. Based on analysis of user comfort studies in other disciplines, we develop a measure of discomfort for users of assistive mobile robots. This discomfort measure is a sum of total travel time and time integrals of squared tangential jerk and squared normal jerk. The weights can be changed by a user to change the relative contributions of travel time and jerk. We expect that future studies with human subjects will lead to validation and/or refinement of this measure of discomfort.

- We develop a framework for safe and comfortable motion planning of assistive robots that removes the limitations of existing work. We present a precise mathematical formulation of kinodynamic motion planning of a wheeled mobile robot moving on a plane as a nonlinear constrained optimization problem. This includes an in-depth analysis of conditions under which the cost-functional is mathematically meaningful, and analysis of boundary conditions. Such a formulation of kinodynamic motion planning for wheeled robots is absent from the literature. The closest existing work to ours is for manipulator trajectory planning [97]. The trajectories planned by our framework have several useful properties – they exactly satisfy boundary conditions on position, orientation, curvature, velocity and acceleration, satisfy kinematic and dynamic constraints, and avoid obstacles while minimizing discomfort. Further, this framework is capable of planning a family of trajectories between a start and a goal state and can be customized by different users to obtain a trajectory that satisfies their comfort requirements.

Our motion-planning framework is not limited to assistive robots. It applies equally well to motion planning of other wheeled robots including robotic cars.

- We represent obstacles as star-shaped domains with piecewise C^2 boundary. This choice allows treatment of non-convex obstacles without subdividing them into a union of convex shapes. This reduces the number of constraints imposed due to obstacles and leads to a faster optimization process. This is unlike most collision-detection modules used with sampling-based algorithms that assume polygonal obstacles, and detect collisions between non-convex polygons by subdividing them into convex polygons [61, 65, 73].

- We use the Finite Element Method to discretize the above infinite-dimensional problem into a finite dimensional problem. Using the Finite Element Method along with other careful choices (See Chapter 3) results in a nice sparsity structure of the Hessian, making the problem amenable to fast numerical solution by an optimization algorithm. The use of Finite Element Method to solve such optimization problems is rare in robotics, and to our knowledge, has only been done for manipulator trajectory planning [97].

Part of the work in this dissertation has been presented in [28, 29, 68].

Chapter 2

Background and Related Work

Chapter 1 provided an overview of existing motion planning methods that are relevant to our work. This chapter reviews the rich body of literature on robot motion planning in some more detail.

2.1 Fundamental Concepts

This section introduces the concepts of *configuration space* and *phase space* fundamental to understanding the motion planning literature.

2.1.1 Configuration Space

Consider an object moving in a two or three dimensional world, or *task space*, that has one or more obstacle regions. The configuration of an object is a specification of all points comprising the object in a global frame of reference. For example, the configuration of a rigid body that can translate and rotate in $\mathbb{R}^2$ can be specified by the position $(x, y) \in \mathbb{R}^2$ and orientation $\theta \in [0, 2\pi)$ of a body-fixed frame with respect to the world frame (Figure 2.1). The configuration space $\mathcal{C}$ is the set of all possible configurations. Thus the configuration space $\mathcal{C}$ of the rigid body is $\mathbb{R}^2 \times \mathbb{S}^1$.

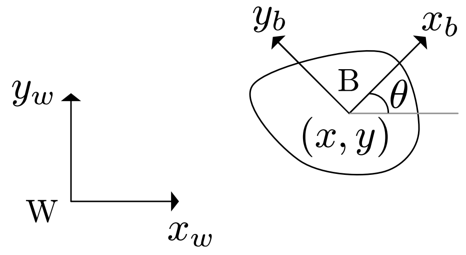


Figure 2.1: The position of any point on a rigid body in $\mathbb{R}^2$ can be determined if the position and orientation of body-centered coordinate frame B in a world coordinate frame W is known. Thus, the configuration of the rigid body can be completely specified by specifying the position (x, y) and orientation θ of B with respect to W.

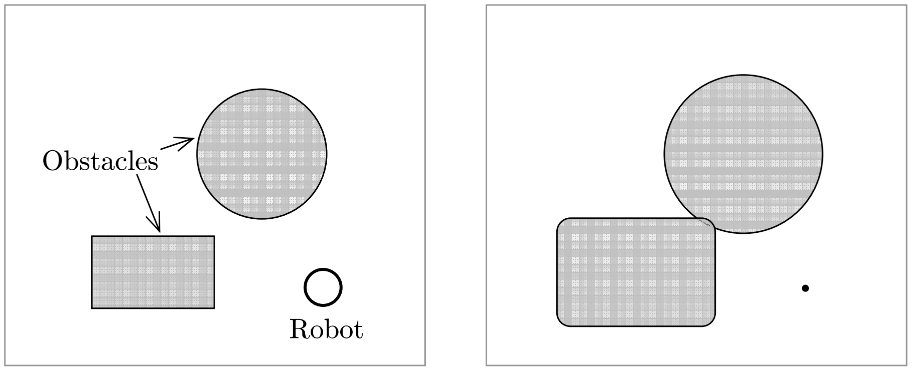
The configuration space obstacle region $\mathcal{C}_{obs}$ is the set of all configurations that intersect with an obstacle in the task space. The set of the remaining configurations is the free configuration space $\mathcal{C}_{free}$. The configuration space for a circular translating robot in $\mathbb{R}^2$ is shown in Figure 2.2.

Let $\mathbf{q} \in \mathcal{C}$ represent the configuration of a robot. The *kinematic model* is given by the state transition equation

$$\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \mathbf{u}, t), \tag{2.1}$$

where the dot represents the derivative with respect to time, $\mathbf{u} \in U(\mathbf{q})$ is the control or input velocity vector, $U(\mathbf{q})$ is the set of all controls, $\mathbf{f}$ is smooth function and t is time. The set $U(\mathbf{q})$ is usually assumed to be state independent. Hence we will drop the notation $U(\mathbf{q})$ and use U for the control set.

A *trajectory* is a function, $\mathbf{q}(t)$, that describes the configuration of the robot at every instant of time during the robot's motion. A *control trajectory*



(a) (b)

Figure 2.2: (a) A circular translating robot in a two dimensional workspace. (b) The configuration space of the robot. The circular robot shrinks to a point while the obstacles have "grown" by the radius of the robot. The white space is $\mathcal{C}_{free}$ and the gray space is $\mathcal{C}_{obs}$. The mapping from workspace to configuration space for arbitrary shaped robot and obstacles where the robot can also rotate is much more complicated.

is a function, $\mathbf{u}(t)$, that describes the control inputs at every instant of time during the robot's motion

A *holonomic constraint* can be expressed purely as a function of the configuration variables and is of the form

$$g(\mathbf{q}, t) = 0.$$

A holonomic constraint reduces the dimension of the configuration space by one. A *nonholonomic constraint* is a constraint involving velocities and is of the form

$$g(\dot{\mathbf{q}}, \mathbf{q}, t) = 0.$$

A nonholonomic constraint cannot be integrated to yield a constraint involving configuration variables alone and does not reduce the dimension of the configuration space. One example of a system with nonholonomic constraints is a disk rolling on plane (Figure 2.3).

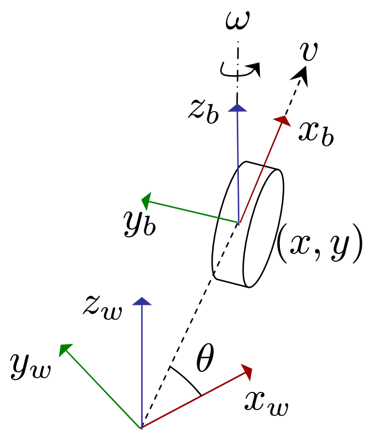


Figure 2.3: A disk rolling on a plane. The body centered frame has axes x_b, y_b, and z_b. The disk can translate along x_b and rotate about z_b, but cannot translate sideways along y_b. This is an example of a nonholonomic constraint. Suppose that a translational speed v and a rotational speed ω are applied as inputs. Then the kinematics of the disk are given by Equation 2.2

.

The configuration of the disk is given by $\mathbf{q} = [x, y, \theta]^T$, and external inputs in the form a linear speed v and angular speed ω can be applied to it. Thus $\mathbf{u} = [v, \omega]^T$. The kinematic model of the rolling disk is

$$\begin{aligned} \dot{x} &= v \cos\theta \\ \dot{y} &= v \sin\theta \\ \dot{\theta} &= \omega, \end{aligned} \tag{2.2}$$

where v and ω are the linear and angular velocity inputs respectively. This disk is subject to the nonholonomic constraint

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0. \tag{2.3}$$

This constraint means that the disk can roll forward and backward but cannot move sideways. Many wheeled mobile robots, including most wheelchairs are governed by this kinematic model.

2.2 Phase Space

To fully model the motion of a rigid body, its momentum must also be taken into account. This gives rise to second-order differential equations.

A *dynamic* model of the system is

$$\ddot{\mathbf{q}} = \mathbf{f}(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{u}, t),$$

where $\mathbf{u} \in U(\mathbf{q})$ is the control or input force vector, $U(\mathbf{q})$ is the set of all controls, $\mathbf{f}$ is smooth function and t is time. Like the kinematic model, the control set will be assumed to be state-independent.

Constraints on second-order derivatives can be converted to constraints on first-order derivatives by introducing the phase space. The phase space $\mathcal{X}$ is the set of all possible values of $[\mathbf{q}, \dot{\mathbf{q}}]^T$. If the dimension of the configuration space is n, the dimension of the phase space is $2n$.

Let $\mathbf{x} \in \mathcal{X}$. Then $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T$, and constraints on $\ddot{\mathbf{q}}$ in configuration space become constraints on $\dot{\mathbf{x}}$ in phase space. $\mathbf{x}$ is called the *state* of the system. The dynamic model in phase space is

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t). \tag{2.4}$$

2.3 Motion Planning and Control

Motion planning for an autonomous robot refers to the problem of computing either a trajectory, or a control trajectory, or both, given the initial and final configurations and possibly their higher derivatives such that the geometric path of the robot does not intersect any obstacles in its task-space. In addition, the trajectory must satisfy *kinematic* and *dynamic constraints.* Kinematic constraints refer to constraints on configuration and dynamic constraints refer to constraints on first and higher derivatives of configuration.

Let $\mathbf{q}_d(t)$ $t \in [0, \tau]$ be a planned or *desired* trajectory where $\tau \in \mathbb{R}$ is the motion duration. An open-loop control trajectory for executing this state trajectory can be computed from the inverse model of the system:

$$\mathbf{u} = \mathbf{g}(\mathbf{q}_d, \dot{\mathbf{q}}_d, t) \tag{2.5}$$

It is usually not possible to model a system exactly. Hence, open loop control may lead to a trajectory that is significantly different from the desired trajectory $\mathbf{q}_d(t)$. Therefore, feedback control is used for most real systems. The feedback controller uses an error measure between the current state and the desired state along with the inverse model to compute closed-loop control inputs. Some methods such as optimal control incorporate a forward model of the system in the planning stage and compute both an open-loop control trajectory and a state trajectory.

2.4 Decoupled Motion Planning

Decoupled planning approaches break the motion planning problem into two components – path planning and velocity planning. This is also referred to as the path-velocity decomposition approach [39]. First, a path-planning algorithm is used to find a collision free path in configuration space. Then, this path is modified to satisfy dynamic constraints so that it is possible for the robot to follow the path. Finally, a velocity on the path is computed such that dynamic constraints are satisfied [9], some performance measure is optimized [9] or moving obstacles are avoided [39]. The trajectories found in this way are not optimal, in general, because while the path, and velocity on the path may separately be optimal, their composition may not be optimal.

We discussed some of the advantages and disadvantages of this approach in Section 1.1.2. Here we discuss some of the more influential path-planning algorithms.

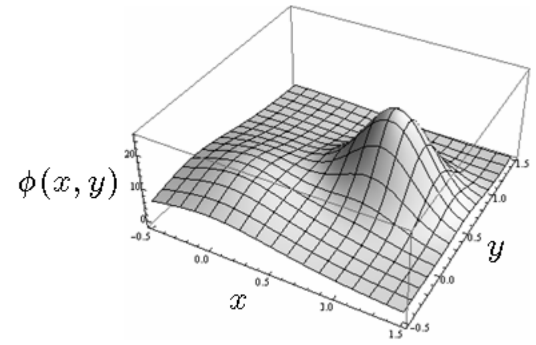
2.4.1 Path Planning

A basic path planning problem that does not consider dynamic constraints is defined as follows: Given (i) the geometry of a robot moving in a two or three dimensional world (ii) the geometry and position of obstacles in the world (iii) the initial and final configuration of the robot, find a continuous sequence of configurations (path) connecting the initial and final configurations, such that the interior of the robot's body does not intersect with any obstacle at any configuration specified by the path. It is assumed that the obstacles are stationary. Despite the simplification achieved by neglecting dynamic constraints, this is a challenging problem and has received significant attention.

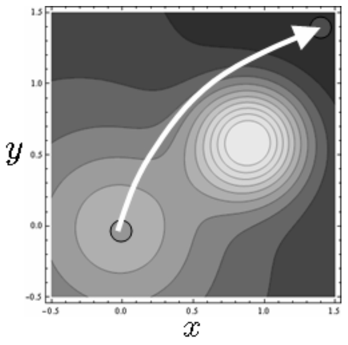
The problem of finding a path for an object with dimensions can be transformed into that of finding a path for a point in an appropriate space using the concept of configuration space [52, 63]. With this transformation, the path planning problem becomes that of finding a continuous curve in $\mathcal{C}_{free}$ that connects the initial and final configurations in $\mathcal{C}_{free}$. Path planning in configuration space has been a very active field and some excellent references are [14, 52, 57]. We discuss two approaches that have been more useful in practice.

2.4.1.1 Potential Functions

Potential functions have been one of the more influential path planning techniques. These were first introduced as a reactive obstacle avoidance method [42] but were later extended to path planning [5]. A potential function is a differentiable real valued function $\phi : \mathcal{C}_{free} \mapsto \mathbb{R}$ and is designed such that the goal configuration is a minimum. For example, one may specify a potential function as the sum of an attractive potential that decreases as the distance from the goal decreases, and a repulsive potential around each obstacle that



(a)



(b)

Figure 2.4: (a) Potential function over a two dimensional configuration space. The goal is to the far right and is the minimum of the potential function in the domain. The "hill" is the repulsive potential around an obstacle. (b) Contour plot of the potential function. One possible path to the goal from an initial configuration is shown.

increases as the configuration comes close to the obstacle. Figure 2.4 illustrates one such potential function. One might also specify a potential function that assigns a cost to each point such that the cost is minimum at the goal [57].

There are many variants of algorithms that use potential functions for path planning. In one formulation, similar to gradient descent optimization, a step is taken in the direction of the negative gradient of the potential function to reach a new point in the configuration space [14]. This process is continued till the goal configuration is reached (to within a specified tolerance). In another formulation, the potential function is thought to induce a vector field on the configuration space from which the velocity at any point $\mathbf{x} \in \mathcal{C}_{free}$ is computed as $\dot{\mathbf{q}} = -\nabla\phi$ [57]. In yet another formulation, the gradient of the potential function is combined with dynamics of the system to compute the forces that must be applied [42, 77].

A potential function can have many local minima causing a planning algorithm to become stuck. Randomized potential fields (RPPs) [5] alleviate this by performing random walks to escape the minima. However, it is easy to construct examples where RPPs fail to find paths [40]. Navigation functions [77] were also introduced as a solution to the problem of local minima. Navigation functions are defined so that there is only a single minimum at the goal, except at a few saddle-point configurations. However, navigation functions have been found to be difficult to construct in practice for domains with arbitrary shaped robot and obstacles or high dimensional configuration spaces [41, 44].

While the potential function approach appears attractive due to its apparent mathematical simplicity, it presents several theoretical and practical difficulties. This approach requires many heuristics in specifying a potential function [52]. For example, defining a repulsive potential around an obstacle requires choosing an appropriate form of the function and parameters that determine the region of influence of the obstacle. This becomes difficult if the number of obstacles is large or if the obstacles have arbitrary shapes. In addition, various heuristics are needed to escape from the local minima of the potential function. If the potential function is used to compute control velocities or forces, the resulting values may not be physically meaningful. This is because the potential function is an artificially defined function and has no relation to the dynamics of the body. This leads to the need for scaling these values using multiplicative constants which have to be tuned. Some practical difficulties were also identified after experiments with a mobile robot in [45] and include oscillations while moving between two closely-spaced obstacles and oscillations near the goal.

2.4.1.2 Sampling-based Planning

In general, it is not easy to explicitly obtain a representation of $\mathcal{C}_{obs}$ for obstacles that are not polygonal or polyhedral in shape, especially in high dimensional configuration spaces. Hence, many algorithms have been developed that do not need an explicit representation of $\mathcal{C}_{obs}$ [33, 40, 41, 43, 56]. These algorithms build a representation of the free configuration space in the form of a topological search graph by incrementally sampling the space, checking for collisions, and adding collision-free configurations to the search graph. Collision checking is done in the task space by means of an independent collision-detection module.

Sampling-based algorithms follow two paradigms – *multiple-query* and *single-query*. In the multiple-query paradigm, algorithms are designed to answer multiple path planning queries. The algorithms consist of two steps: the roadmap construction phase and the query phase [40]. In the roadmap construction phase, the configuration space is preprocessed to build a roadmap representing the connectivity of $\mathcal{C}_{free}$. The roadmap is an undirected graph $G = (V, E)$ in the configuration space where the vertices V are configurations in $\mathcal{C}_{free}$ and the edges E are paths between pairs of free configurations. The roadmap is usually constructed by sampling the configuration space to obtain a set of collision-free configurations and connecting each configuration to k-nearest configurations by collision-free paths. In the query phase, the roadmap is used to find paths between specified configurations. Once a roadmap has been constructed, multiple queries can be quickly answered by searching the roadmap. Multiple query algorithms are efficient for static environments since once a roadmap has been constructed, it is computationally cheap to find a path.

In the single-query paradigm, algorithms are designed to find a path between a pair of configurations only once. These algorithms start constructing a topological graph when such a planning query is received. Let (q_i, q_f) be the initial and goal configurations in a new query. The search graph G begins with the initial configuration q_i as the only vertex. The configuration space is sampled for a new collision-free configuration q_{new} and a local planner computes a path between q_{new} and an existing vertex q_{curr} in the graph. A collision-detection algorithm checks the path and if the path is collision free, an edge (q_{curr}, q_{new}) is added to the graph. Sampling is biased towards configurations that are close to the initial and goal configurations. The graph continues to grow till a solution path is found. The single-query paradigm is, in general, more efficient for cases where the environment changes because extra computational effort is not wasted in constructing a roadmap through the entire configuration space.

Many different sampling-based algorithms have been proposed, the key difference being the sampling strategy. Grid based sampling [43, 80] is one the earliest sampling techniques. The configuration space is discretized into a grid and samples are chosen from these grid points. Here, the main difficulty is in choosing a suitable resolution of the discretization. If the resolution is too fine, the search in the query phase may take a long time since the graph will consist of a large number of vertices. If the resolution is too coarse, the solution may never be found since the goal configuration may be too far from any vertex in the graph [81].

The Rapidly Exploring Random Tree (RRT) [56, 60] algorithm does not discretize the configuration space into a grid but chooses new configurations by randomized sampling. The sampling scheme may also be biased toward regions of the configuration space near the initial or goal configurations. The RRT algorithm has been found to be very efficient and RRT and its variants have been used in many applications involving high degree of freedom manipulator arms [60, 82]. RRT sacrifices optimality of paths for efficiency and feasibility.

RRT was originally developed for motion planning with dynamic constraints. This version of RRT, called Rapidly Exploring Dense Trees (RDTs), is discussed later in Section 2.5.1.

2.5 Direct Trajectory Planning

This section discusses methods that directly find a trajectory without going through the intermediate step of planning a path. The trajectory planning problem is defined as follows: Given (i) the geometry of a robot moving in a two or three dimensional world (ii) the geometry and position of obstacles in the world, and (iii) the initial and final configuration and possibly higher derivatives of configuration, find a continuous trajectory connecting the initial and final configurations, such that the interior of the the robot's body does not intersect with any obstacles at any time. Dynamic constraints should be satisfied. Moving obstacles may be considered, and some performance measure may be optimized. Two trajectory planning approaches that have found practical applications are discussed below.

2.5.1 Sampling-based Planning

Sampling-based planning methods of Section 2.4.1.2 can be extended to planning in phase space. The planning problem is to find a path in phase space such that its projection in configuration space does not intersect any obstacles. The general framework is similar to that of sampling-based path planning in configuration space. The key difference is in the way new vertices are added to the graph. Any new sampled point in phase space cannot be added to the graph since it may be impossible to reach the point while also satisfying dynamic constraints.

Sampling-based methods find a trajectory by sampling the state space [16, 59] or state-time space [25, 33] to iteratively construct a search graph rooted at the initial state. We described these methods in Section 1.1.2. Like the grid-based methods of Section 2.4.1.2, these methods discretize the space into a grid and construct a search graph. These approaches suffer from the curse of dimensionality and become computationally expensive as the dimension of the state space increases. To alleviate this problem, methods that construct a search graph by randomized sampling of state space [59] or state-time space [33] were developed.

We describe one such randomized sampling algorithm, the Rapidly Exploring Dense Tree (RDT) algorithm here. A new vertex is added as follows (i) a sample point q_{new} is chosen from a randomized sequence (ii) a vertex q_{curr} in the graph that is closest to the sample point, according to a distance metric, is selected (iii) all controls from a set of discretized controls are applied to q_{curr} and the system is allowed to evolve for a time Δt (iv) out of all the new points that can be reached via collision-free trajectories satisfying differential constraints, the point nearest q_{new} is chosen and added to the the graph. An example application of RDT for a mobile robot with dynamic constraints is shown in Figure 2.5.

There are several challenges to applying sampling-based methods for trajectory planning. First, defining a good distance metric is difficult because it is not easy to choose meaningful weights for dimensionally different terms such as position, orientation, linear and angular velocity, to make up a weighted sum. Second, the dimension of the phase space is twice that of the configuration space which increases the time taken for obtaining a dense coverage. Third, the choice of a suitable time step Δt and a suitable discretization of the action space may involve many problem-specific heuristics.

Because of discretization, a goal state cannot be reached exactly in

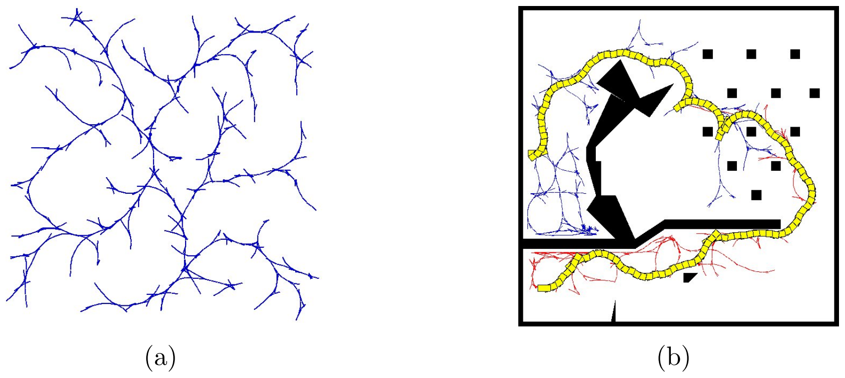


Figure 2.5: (a) An example of a path generated by RDT for a car-like robot with dynamic constraints. The robot can move forwards or backwards but cannot move sideways. The maximum radius of curvature is bounded because of the limited steering angle of the car. (b) Path found by RDT for the car-like robot in an obstacle strewn environment. Notice that the path involves many backing-up and "wavy" maneuvers because the each of the path segments is obtained by applying a constant control for a fixed interval of time. Figures reproduced with permission from [58].

finite time. The accuracy of reaching a goal increases as the resolution of discretization is increased. To reach the goal exactly, a two-point boundary value problem has to be solved to compute a trajectory that connects the goal state to the search graph. This is non-trivial problem since this trajectory must avoid obstacles and satisfy dynamic constraints. Bidirectional search strategies grow the search graph simultaneously from both the start and goal configurations. In this case a boundary value problem has to be solved to connect the two graphs [57].

Since a constant control input is applied for a finite length of time to compute a trajectory segment, the paths planned by these methods for wheeled robots usually lack curvature continuity. These paths have to be smoothed in a post-processing step so that they can be followed by a wheeled robot without frequent stopping.

2.5.2 Optimal Control Methods

Optimal control methods have been traditionally used to plan trajectories for systems subject to dynamic constraints. The formulation consists of constructing a cost functional representing the cumulative cost over the duration of motion and minimizing the cost functional to find a desired state trajectory or control trajectory or both. Optimal control methods can be thought of as planning in continuous space as opposed to sampling methods that plan in an artificially discretized space.

Optimal control methods have been widely applied to trajectory planning in aerospace engineering and control-systems engineering. The general formulation of an optimal control problem is as follows:

Determine a state-control trajectory $\{\mathbf{x}(t), \mathbf{u}(t) : 0 < t < \tau\}$, and possibly the final time τ that minimize the cost functional

$$J(x(.), u(.), \tau) = h(\mathbf{x}(\tau), \tau) + \int_0^\tau l(\mathbf{x}(t), \mathbf{u}(t), t)dt,$$

given the boundary conditions

$$\mathbf{x}(0) = \mathbf{x}_0, \qquad \mathbf{x}(\tau) = \mathbf{x}_\tau,$$

$$\mathbf{u}(0) = \mathbf{u}_0, \qquad \mathbf{u}(\tau) = \mathbf{u}_\tau,$$

subject to the dynamic constraints

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t),$$

and other constraints on state and control variables

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \leq 0.$$

Sufficient conditions for a solution of this system are given by the Hamilton-Jacobi-Bellman (HJB) equation. HJB is a second-order partial differential equation with end-point boundary conditions. Analytic solutions of the HJB equation for linear systems with quadratic cost have long been known [11]. In general, most real systems are nonlinear and the HJB equation has to be solved numerically.

Necessary conditions for optimality are derived using Pontryagin's principle and consist of a set of first-order ordinary differential equations. These differential equations convert the optimization problem into a two-point boundary value problem. The system of differential equations can either be solved analytically (where possible) or numerically using methods such as the shooting method or finite-difference methods.

Analytical solution to the problem of finding minimum length paths for Dubins [17] car and Reeds and Shepp [74] car (see [87]) was found using such an approach. Dubins car is only allowed to move forward while Reeds and Shepp car is also allowed to move backward. These paths are comprised of straight line and arc segments and minimize the distance traveled by the mid-point of the rear axle. Each path segment is traversed at a fixed speed, so the trajectories corresponding to these paths are also time-optimal for a given speed. More recently, shortest paths for a differential drive wheeled robot were developed by including a rotation cost in the cost functional [4] (since a differential drive robot can turn in place). Such minimum-time paths, while of theoretical interest, are of little practical use in motion planning for wheeled mobile robots because they lack curvature continuity and require frequent stopping and reorienting of wheels.

In general, more complex problems require a numerical solution. One frequently used numerical method is the "shooting method" where the two point boundary value problem is converted into an initial value problem. The objective now is to guess the initial state and improve upon this guess till the actual final state is close to the specified final state. Numerically, this is accomplished by discretizing the control function $\mathbf{u}(t)$, estimating the state at the end point by integrating the state-transition equation, and perturbing the unknown control parameters and the initial state to minimize the distance between actual final state and specified final state. This nonlinear optimization problem is solved using well-known techniques such as Newton's method. Shooting methods have been used for trajectory planning for nonholonomic mobile robots [20, 32]. However, in shooting methods, it is extremely challenging to specify a good initial guess of the unknown parameters that produce a final state reasonably close to the specified state. This problem arises because the final state is obtained by integrating the state equations starting from the initial state and hence it can be very sensitive to changes in the initial state.

Instead of of solving the differential equations representing necessary conditions, approximation methods that discretize the infinite-dimensional problem into a finite-dimensional one and optimize the cost functional directly in this finite-dimensional space can be used. Either only the control function $\mathbf{u}(t)$ or both the state and control functions $\mathbf{x}(t)$ and $\mathbf{u}(t)$ can be discretized. Such methods have been used for planning optimal trajectories of mobile robots. In [21], control inputs that minimize total control energy to travel between a given pair of boundary states are computed. Here Fourier basis functions are used for discretization. In [97], trajectories that minimize the integral of square of L^2 norm of end-effector jerk and the square of L^2 norm of time derivatives of joint torque vector, subject to torque constraints, are computed. Here a finite-element discretization is used. Other discretization are also possible, such as B-spline [10] and spectral [90] discretization.

Very few of the existing optimal control approaches include obstacle-avoidance. Obstacle-avoidance is achieved in [80], but by adopting a path-velocity decomposition approach. First the configuration space is discretized into a grid and an obstacle-free path is found by searching this grid, then a time-optimal path in the neighborhood of this path is computed, and finally a time-optimal velocity on the path is determined. Obstacles were included as hard constraints for a two-dimensional translating robot in [95].

Thus, optimal control methods have primarily been used for trajectory planning in the absence of obstacles. Further, while some special problems such as that of finding time-optimal trajectories have been solved, a comprehensive formulation of kinodynamic motion planning problem for wheeled mobile robots is absent.

2.6 Collision Detection and Obstacle Avoidance

Obstacle avoidance is inherent in the definition of path planning. The planning algorithms discussed in the previous sections deal with obstacles in two ways. One set of algorithms, such as the potential function methods, map the obstacles to configuration space to obtain a representation of C_{obs}. Then the representation of C_{obs} is used to define a repulsive potential function that is added to the overall potential function. The other set of algorithms, such as sampling-based algorithms, do not construct an explicit representation of C_{obs}. They generate a path (or trajectory) segment and test it for collisions using a collision detection module that tests for collisions in the task-space (instead of the configuration space). Many collision detection algorithms developed in the field of computational geometry have been adopted in robotics. An obstacle is modeled as a shape (e.g. polyhedron) or hierarchy of primitive shapes (e.g. as in constructive solid geometry) or as a parametric or implicit curve. The robot is modeled in a similar way. Then, the robot model is tested for intersection with the obstacle models. Some excellent references on collision detection algorithms are [38, 61, 62, 65]. In the optimal control approach, obstacles can be included as hard constraints, or added to the cost functional in the form of barrier functions.

2.7 Summary

There are two main approaches to trajectory planning that have been found to be of practical use for kinodynamic motion planning – sampling-based methods and optimal control methods. Existing formulations of these methods cannot compute trajectories that satisfy all requirements of comfortable motion. While sampling-based methods are computationally efficient, the trajectories computed by these methods cannot reach a goal exactly in finite time, lack curvature continuity, and generally sacrifice optimality for efficiency. Optimal control methods compute trajectories that exactly satisfy boundary conditions, but there is no comprehensive formulation of the full kinodynamic motion planning problem for wheeled mobile robots, and none of the optimal control formulations show demonstrable results with obstacle avoidance.

Chapter 3

Formulating Motion Planning as a Constrained Optimization Problem

This chapter presents the mathematical foundation of our motion-planning framework for planning safe, comfortable, and customizable motion of assistive wheeled mobile robots. The main idea is to formulate a mathematically meaningful measure of discomfort and pose the motion-planning problem as that of minimizing this discomfort subject to appropriate boundary conditions and constraints.

We begin by an analysis of motion of a wheeled mobile robot moving on a plane. We then provide a brief introduction to parametric curves and arc-length parameterization of curves. Next, we review literature from various disciplines to formulate a measure of discomfort. We then present a mathematical formulation of the motion-planning problem as a constrained optimization problem.

3.1 Motion of a Wheeled Mobile Robot on a Plane

We saw in Section 2.1.1 that the configuration of a rigid body moving on a plane at any time t can be completely specified by specifying the position vector $\mathbf{r}(t) = \{x(t), y(t)\}$ and orientation $\theta(t)$ of a body-fixed frame with respect to a fixed reference frame. Suppose the rigid body starts from an initial configuration at time $t = 0$ and reaches a final configuration at time $t = \tau$. To fully specify the motion of the body it is necessary to specify the functions $x(t), y(t)$ and $\theta(t)$ on $I = [0, \tau]$. If this body is a physical system, it cannot change its position instantaneously. Further, since forces of infinite magnitude cannot be applied in the real world, the acceleration of the body must be finite. Hence $x(t), y(t)$, and $\theta(t)$ must be at least C^1 on I.

If this rigid body has directional wheels, its motion should obey the following nonholonomic constraint

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0. \tag{3.1}$$

Here dot, $(\dot{})$, represents derivative with respect to t. For motion planning, it is common to model a wheeled mobile robot as a wheeled rigid body, and we will do the same. A motion of such a body can be specified by specifying a travel time τ and a trajectory $\mathbf{r}(t)$ for $t \in [0, \tau]$. The orientation $\theta(t)$ can be computed from Equation 3.1. Essentially, $\theta(t) = \text{arctan2}(\dot{\mathbf{r}}(t))$. If $\dot{\mathbf{r}}(t)$ is zero, which means the velocity is zero, then this equation cannot be used. If the instantaneous velocity is zero at $t = t_0$, and non-zero in a neighborhood of t_0, then $\theta(t_0)$ can be defined as a $\lim_{t \to t_0} \text{arctan2}(\dot{\mathbf{r}}(t))$.

We now present a brief introduction to parametric curves and the arc-length parameterization since it is relevant to our formulation ahead. The reader can refer to any book on differential geometry of curves for more details.

3.2 Parametric Curves and the Arc-length Parameterization

Let $q_a < q_b$ and $I = [q_a, q_b] \subset \mathbb{R}$. A planar parametric curve is a mapping $\mathbf{r} : I \mapsto \mathbb{R}^2$. If components of $\mathbf{r}$ are of class C^1, the vector space of functions with continuous first derivatives, the tangent vector at $\mathbf{r}(q)$ for $q \in [q_a, q_b]$ is $\mathbf{r}'(q)$. In this section, we denote derivatives with respect to the parameter q by a prime ($'$).

Let the length of a curve be denoted by λ, where

$$\lambda = \int_{q_a}^{q_b} ||\mathbf{r}'(q)||\, dq. \tag{3.2}$$

Define a function $s = s(q)$, which is the length of the curve between $[q_a, q]$. Then,

$$s(q) = \int_{q_a}^{q} ||\mathbf{r}'(q)||\, dq. \tag{3.3}$$

Note that the integrand $||\mathbf{r}'(q)||$ is non-negative throughout I. We make an assumption that it is zero only at a finite number of q's in I. If q were time t, the physical interpretation is that the velocity is equal to zero only at a finite number of discrete instants in time. This assumption implies that s is an increasing function of q. That is, if $q_2 > q_1$, then $s(q_2) > s(q_1)$. This, in turn, means that for any given $s \in [0, \lambda]$, a unique $q = q(s)$ can be found that corresponds to that s. If components of $\mathbf{r}$ are of class C^1, then $||\mathbf{r}'(q)||$ is continuous, and thus $s = s(q)$ is also in C^1. Thus, $\frac{ds}{dq}$ is defined and is a continuous function. Obviously, $\frac{ds}{dq} = ||\mathbf{r}'(q)||$.

With the assumption above that $||\mathbf{r}'(q)||$ can be zero only at a finite number of q's, it is possible to introduce the arc-length parameterization. For $s \in [0, \lambda]$ define

$$\widehat{\mathbf{r}}(s) = \mathbf{r}(q) \text{ where } s = s(q). \tag{3.4}$$

The function $\widehat{\mathbf{r}}$ is well-defined because for each $s \in [0, \lambda]$ a unique q can be found. Using the chain-rule for differentiation,

$$\frac{d\widehat{\mathbf{r}}}{ds} = \frac{d\mathbf{r}}{dq}\frac{dq}{ds}.$$

Now $\frac{d\mathbf{r}}{dq}$ exists and is continuous and $\frac{dq}{ds} = \frac{1}{\frac{ds}{dq}} = \frac{1}{||\mathbf{r}'(q)||}$ also exists (and is continuous) if $||\mathbf{r}'(q)||$ is not zero. Thus, at points where $||\mathbf{r}'(q)|| > 0$,

$$\left|\left|\frac{d\widehat{\mathbf{r}}}{ds}\right|\right| = ||\mathbf{r}'(q)|| \,/\, ||\mathbf{r}'(q)|| = 1.$$

On points where $||\mathbf{r}'(q)|| = 0$, $\left|\left|\frac{d\widehat{\mathbf{r}}}{ds}\right|\right|$ cannot be computed by the expression above. However, the choice that makes it continuous for all s is 1. This is analogous to computing the limiting value of the orientation when velocity is zero as shown earlier in this section.

Symbolically, the curve has been parameterized by the arc-length. Since $\left|\left|\frac{d\widehat{\mathbf{r}}}{ds}\right|\right| = 1$, the tangent vector computed in the new parameterization is a unit vector. The tangent vector is $\mathbf{T}(s)$ and the unit normal vector is $\mathbf{N}(s)$, where

$$\begin{aligned} \mathbf{T}(s) &= \frac{d\widehat{\mathbf{r}}}{ds} \\ \mathbf{N}(s) &= \frac{\frac{d\mathbf{T}}{ds}}{\left|\left|\frac{d\mathbf{T}}{ds}\right|\right|} \end{aligned} \tag{3.5}$$

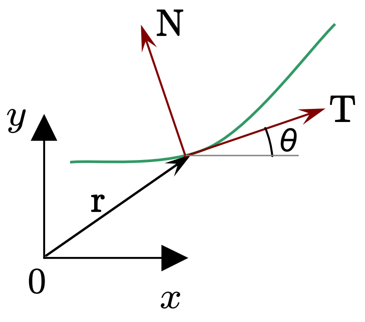


Figure 3.1: Tangent and Normal to a curve

See Figure 3.1. The signed curvature $\kappa(s)$ is defined as

$$\kappa(s) = \frac{d\theta}{ds} \tag{3.6}$$

where $\theta(s)$ is the tangent angle.

3.3 A Characterization of Discomfort

To characterize comfortable motion, we review literature from transportation design, elevator design, robot motion planning, and neuroscience. Studies of passenger discomfort in automobiles and trains have shown that discomfort increases as the magnitude of acceleration increases [24, 37, 70]. Two separate components of acceleration effect discomfort – tangential component along the direction of motion and normal component perpendicular to the direction of motion [24, 37, 70]. The normal component is zero in a straight line motion but becomes important when traversing curves. The actual values of comfortable bounds of the two components may be different [91], may vary across people, may depend on the mode of transportation, and may depend on the passenger's position [24, 70]. Hence, guidelines for ground transportation design prescribe maximum values of accelerations [13, 36, 91], or maximum values of comfort indices that are functions of accelerations [12, 35]. Studies also show that discomfort increases as the magnitude of jerk increases [24, 70]. Upper bounds on jerk for comfort have been proposed for road [13] and railway vehicles [91]. In elevator design, motion profiles are designed for user comfort by choosing profiles with smooth accelerations and low jerk [31, 46, 88]. In neuroscience, studies of point-to-point human arm movements show a velocity profile that is consistent with minimizing time integral of squared jerk over the motion duration [23, 94]. Later a similar model was shown to replicate grasping actions of fingers [86].

From a geometric standpoint, it has been known for more than a century that sharp changes in curvature of roads and railway tracks can be dangerous and can cause passenger discomfort [27, 50, 55]. In robotics, planning continuous curvature paths for mobile robots has received significant attention and has primarily been motivated by the desire to drive the robot with non-zero speed from start to goal [8, 9, 26, 49, 71]. This is because a discontinuity in curvature requires the speed to go to zero for continuity of speed at that point.

Thus, we can conclude that for motion comfort, it is necessary to have continuous and bounded acceleration along the tangential and normal directions. It is possible that the actual values of the bounds on the tangential and normal components are different. It is also desirable for paths to have curvature continuity so that the robot can travel on the path without having to stop and reorient wheels.

Based on the above analysis, we define a measure of discomfort as a weighted sum of the following three terms: total travel time, time integral of squared tangential jerk and time integral of squared normal jerk. Travel time is included because we make the justifiable assumption that a user would prefer to reach a goal as fast as is consistent with comfort. Thus, longer travel time implies greater discomfort. Jerk is included because we saw above that discomfort increases with the magnitude of jerk [24, 70]. We separate jerk into components – tangential and normal and give them separate weights because the actual values of these two components that cause discomfort may be different, just as the bounds on tangential and normal acceleration may be different [24, 37, 70, 91]. If studies show later that these values need not be different, both the weights can be set equal. We will see later in Section 3.8 that this cost functional is mathematically meaningful only when both tangential and normal acceleration are continuous. Thus, we get continuous accelerations by construction. To keep accelerations within comfortable bounds, we impose explicit constraints on the maximum and minimum values.

3.4 The Discomfort Cost Functional

We construct a cost functional J to precisely define the above discomfort measure as follows:

$$J = \tau \ + \ w_{T} \int_0^{\tau} (\dddot{\mathbf{r}} \cdot \mathbf{T})^2 \, dt \ + \ w_{N} \int_0^{\tau} (\dddot{\mathbf{r}} \cdot \mathbf{N})^2 \, dt. \tag{3.7}$$

Here τ is the total travel time and r is the position of robot at time $t \in [0, \tau]$. $\dddot{\mathbf{r}}$ represents the jerk. $\dddot{\mathbf{r}} \cdot \mathbf{T}$ and $\dddot{\mathbf{r}} \cdot \mathbf{N}$ are the tangential and normal components of jerk respectively. We assume that $\mathbf{r}(t)$ is smooth enough for the cost functional to be well-defined. This means (at least) that the acceleration vector is continuous and normal and tangential components of jerk are square integrable.

The term τ is necessary. If it is not included in the functional, the optimal solution is to reach the destination at $\tau = \infty$ traveling at essentially zero speed in the limit (except perhaps at the end-points where the speed is already specified). Thus, minimizing just the integral terms will not lead to a good solution.

The weights (w_{T} and w_{N}) are non-negative known real numbers. We separate tangential and normal jerk to allow a choice of different weights (w_{T} and w_{N}).

The weights serve two purposes. First, they act as scaling factors for dimensionally different terms. Second, they determine the relative importance of the terms and provide a way to adjust the robot's performance according to user preferences. For example, for a wheelchair, some users may not tolerate high jerk and prefer traveling slowly while others could tolerate relatively high jerks if they reach their destination quickly. The typical values of weights will be chosen using dimensional analysis.

3.5 Dimensional Analysis of Cost Functional and Determination of Characteristic Weights

Choosing the weights in an *ad hoc* manner does not provide weights that lead to similar comfort levels independent of the input (the boundary conditions). Moreover, since the different components of the total discomfort are different physical quantities, choice of weights should reflect this. In other words, for the total discomfort to make physical sense, the weights cannot be dimensionless numbers but should have physical units. We determine the weights using dimensional analysis [51]. If the weights are chosen without the dimensional analysis step, the optimal trajectory will be different just by specifying the input in different physical units. In addition, using the same numerical weights for different tasks will not lead to similar quantitative discomfort level.

All the physical quantities in the cost functional (time, jerk) depend on only two units − length L and time T. From Equation 3.7 we see that J has dimensions L^0T^1 due to the first term (τ). Thus w_T should have dimensions T^6/L^2. Similarly, the dimensions of w_N is T^6/L^2. Alternatively, since $T = L/V$, w_T and w_N has dimensions L^4/V^6.

We now determine the base values of weights analytically. The main idea behind determining the base values is that the correct base values should keep the maximum speed below the maximum allowable speed. A user can then customize the weights by multiplying the base values by a dimensionless constant that indicates user preference.

3.5.1 Weight for Tangential and Normal Jerk

We first determine w_T. Consider a one dimensional motion with a trajectory that starts from origin and travels a distance $L > 0$ in an unknown time $\tau > 0$. The starting and ending speeds and accelerations are zero. We choose the exact form of the trajectory to be a quintic polynomial in time $t \in [0, \tau]$. This choice uniquely determines the trajectory. The reason we have chosen a quintic is that it minimizes integral of squared jerk (a third derivative), just like a cubic spline minimizes integral of squared second derivative. Additionally, we choose the quintic to satisfy the boundary conditions.

Let $s(t)$ be the distance traveled in time t. It is easily seen that the quintic

$$s(t) = \frac{Lt^3}{\tau^5}\left(6t^2 - 15t\tau + 10\tau^2\right)$$

satisfies all the boundary conditions. For such a trajectory, the discomfort functional is

$$J = \tau + w_T \int_0^\tau \dddot{s}(t)^2 dt = \tau + \frac{720L^2 w_T}{\tau^5}.$$

We do not know τ and w_T yet. We first choose a τ that minimizes J for all w_T. This means

$$\tau = \left(3600L^2 w_T\right)^{1/6}.$$

Obviously, choosing a large value of w_T will increase τ, which is natural because doing so penalizes jerk and would slow down the motion. We now choose a w_T so that the maximum speed during the motion is V, a dimensional velocity scale. It can be seen that the maximum speed occurs at $t = \tau/2$ and it is

$$\left(\frac{225}{2048}\right)^{1/3}\left(\frac{L^4}{w_T}\right)^{1/6}.$$

Hence we choose

$$w_T = \left(\frac{225}{2048}\right)^2 \frac{L^4}{V^6}. \tag{3.8}$$

The base value for the weight corresponding to the normal jerk (w_N) is chosen to be the same. We emphasize that both w_T and w_N will be present in a real problem and the maximum speed constraint is imposed explicitly rather than relying on weights. The analysis done here is to get dimensional dependencies of the base weight and reasonable proportionality constants using a simple problem that can be treated analytically.

3.5.2 Factoring the Weights for Customization

In the preceding discussion, we determined the base values of weights using simple analytical problems. We will refer to these base values as $\widehat{w}_T$ and $\widehat{w}_N$. Let R_* be the minimum turning radius of the robot. For any given input, we determine the characteristic length L_* as $\max(\Delta L, \pi R_*)$ where ΔL is the straight line distance between the start and end points. The characteristic speed V_* is the maximum allowable speed of the robot. The base values of weights are then computed as

$$\widehat{w}_T = \widehat{w}_N = \left(\frac{225}{2048}\right)^2 \frac{L_*^4}{V_*^6}. \tag{3.9}$$

The weights for the actual problem are chosen as a multiple of these base weights where the multiplying factors f_T and f_N are chosen by a user.

$$\begin{aligned} w_T &= f_T \widehat{w}_T, \\ w_N &= f_N \widehat{w}_N. \end{aligned} \tag{3.10}$$

3.6 Problem Statement

We formulate the problem of planning safe and comfortable motion planning as a constrained optimization problem as follows: Given the start and end conditions on position, orientation, speed, and acceleration, the values of bounds on curvature, speed and acceleration, the locations and representation of obstacles, the weight factors f_T and f_N (Equation (3.9)), find a trajectory that minimizes the cost functional of Equation (3.7) such that bounds are not violated and the geometric path does not pass through obstacles.

We model the robot as a wheeled rigid body moving on a plane and assume that the robot moves with non-zero speed except at a finite number of points. Let the robot start from $\mathbf{r}_0$ at $t = 0$ and reach $\mathbf{r}_\tau$ in time τ (Figure 3.2). From the discussion in Section 3.1, we see that to fully specify the motion of the robot, we need only to specify a curve $\mathbf{r}(t)$ on $t \in [0, \tau]$ such that the curve is at least C^1 continuous. Henceforth, in this chapter, we will use *trajectory* to refer to a function of robot position with respect to time.

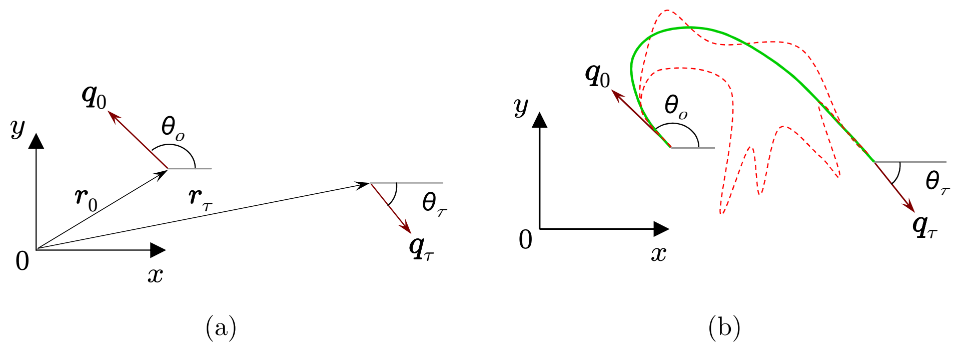


Figure 3.2: Illustration of the optimization problem.
(a) The initial configuration of the robot at time $t = 0$ is given by the position $\mathbf{r}_0$ and orientation θ_0. The final configuration at time $t = \tau$ is given by the position $\mathbf{r}_\tau$ and orientation θ_τ. The speed at an end point, when non-zero, is necessarily along the vector $\mathbf{q}$. (b) There exist infinitely many trajectories that satisfy boundary conditions and respect constraints, illustrated by the solid and dotted curves. Infinitely many of such trajectories will not result in comfortable motion, illustrated by the dotted curves. Our objective is to find a trajectory $\mathbf{r}(t)$ that additionally minimizes the cost functional of Equation 3.7 and results in comfortable motion. Such a trajectory is illustrated by the solid curve.

We now transform the above problem description into a precise mathematical problem statement using a general nonlinear constrained optimization approach. Our objective is formulate a mathematically meaningful infinite-dimensional optimization problem with a complete analysis of boundary conditions and constraints.

The steps involved are: (i) choosing an appropriate parameterization of the trajectory (Section 3.7), (ii) choosing the function space to which the trajectory should belong for the cost functional to be well-defined (Section 3.8), (iii) analysis of boundary conditions to determine the boundary conditions that should be imposed for the problem to be well-posed (Section 3.9), (iv) choosing a representation of obstacles and imposing constraints for obstacle avoidance (Section 3.10), and finally, (v) formulating the full infinite-dimensional constrained optimization problem (Section 3.11).

3.7 Parameterization of the Trajectory

Mathematically, one can use different primary variables to describe a trajectory. For example, assuming the trajectory starts at zero time, one way to describe a trajectory is to provide the final time and the position vector as a function of time in between. Another way is to provide the final time and specify the orientation and velocity as functions of time. Another way is to represent the geometric path separately, using either position vector or orientation as a function of arc-length. The velocity at each point on the path is provided separately in this case.

We have found that making the assumption that speed be non-zero except at boundaries and expressing the trajectory solely in terms of *speed* and *orientation* as functions of a *scaled* arc-length parameter leads to relatively simple expressions for all the remaining physical quantities (such as accelerations and jerks). We shall see below, that with this parameterization, the primary variables (speed and orientation) and their derivatives enter the cost functional polynomially. This would not have been the case if everything were expressed in terms of $\mathbf{r}$ as a function of time as we did in our previous work [28].

In the following discussion, we implicitly assume that all the quantities being differentiated have sufficient smoothness for differentiation to be mathematically meaningful. In some cases, the derivatives appear not as point-wise values but inside an integral sign. In such a case we will assume that the integrands belong to an appropriate space of functions so that the integrals are well-defined. We explicitly state the requirements on the regularity when posing the optimization problem later in Section 3.8.

3.7.1 Scaled Arc-length Parameterization

Let $u \in [0, 1]$. The trajectory is parameterized by u. The starting point is given by $u = 0$ and the ending point is given by $u = 1$. Let $\mathbf{r} = \mathbf{r}(u)$ denote the position vector of the robot in the plane. Let $v = v(u)$ be the speed. Both $\mathbf{r}$ and v are functions of u. Let λ denote the length of the trajectory. Since only the start and end positions are known, λ cannot be specified in advance. It has to be an unknown that will be found by the optimization process.

Let $s \in [0, \lambda]$ be the arc-length parameter. We choose u to be a scaled arc-length parameter where $u = \frac{s}{\lambda}$ so that the unknown constant λ is not used in defining an unknown sized interval (as would be the case if u was chosen as the arc-length parameter).

In the following discussion we will see that the trajectory, $\mathbf{r}(t), t \in [0, \tau]$ is completely specified by the *trajectory length* λ, the *speed* $v = v(u)$, and the *orientation* or the tangent angle $\theta = \theta(u)$ to the curve. λ is a scalar while speed and orientation are functions of u. These are the three unknowns which will be determined by the optimization process.

Since speed is the rate of change of arc length, we have

$$v(u) = \frac{ds}{dt}. \tag{3.11}$$

Using $u = \frac{s}{\lambda}$ in the above equation, we get

$$\frac{du}{dt} = \frac{v(u)}{\lambda}. \tag{3.12}$$

This gives,

$$t = t(u) = \int_0^u \frac{\lambda}{v(u)} du. \tag{3.13}$$

If $v(u)$ is zero only at a finite number of points in $[0, 1]$, then $t(u)$ is well defined for all $u \in [0, 1]$.

Equation 3.13 is a key relation and gives us the means to convert between the time domain and scaled arc-length domain. We now introduce the third unknown – the orientation or the tangent angle to the curve $\theta = \theta(u)$. Using the results of Section 3.2, we can show that

$$||\mathbf{r}'(u)|| = \lambda. \tag{3.14}$$

The tangent vector $\mathbf{r}'(u)$ to the curve $\mathbf{r}(u)$ is given by

$$\mathbf{r}'(u) = ||\mathbf{r}'(u)||\, \mathbf{T}(u) = \lambda \mathbf{T}(u) \tag{3.15}$$

where $\mathbf{T}(u)$ is the tangent function.

$$\mathbf{T}(u) = \left\{\cos(\theta(u)), \sin(\theta(u))\right\}. \tag{3.16}$$

The braces $\{\}$ enclose the components of a 2D vector.

Thus, $\mathbf{r}(u)$ can be computed via the following integrals.

$$\mathbf{r}(u) = \mathbf{r}(0) + \lambda \left\{ \int_0^u \cos\theta(u)\, du, \int_0^u \sin\theta(u)\, du \right\}. \tag{3.17}$$

Now, if $\theta(u)$ is known, $\mathbf{r}(u)$ can be computed from Equation (3.17). If $v(u)$ and λ are known, $t(u)$ can be computed from Equation (3.13). Using these two, we can determine the function $\mathbf{r}(t), t \in [0, \tau]$.

We now have all the basic relations to use chain-rule to derive expressions for all the physical quantities needed to pose the constrained optimization problem. We drop explicit references to u as a function parameter to keep the expression concise.

We compute first, second, and third derivatives of $\mathbf{r}$ with respect to time. These expressions are easily derived in one or two steps of algebra and so we do not present the intermediate steps in detail.

$$\begin{aligned}
\dot{\mathbf{r}} &= v\{\cos\theta, \sin\theta\} && (3.18)\\
\ddot{\mathbf{r}} &= \frac{v}{\lambda}(v'\{\cos\theta, \sin\theta\} + v\theta'\{-\sin\theta, \cos\theta\}) && (3.19)\\
\dddot{\mathbf{r}} &= \frac{v}{\lambda^2}\left((v'^2 + vv'' - v^2\theta'^2)\{\cos\theta, \sin\theta\}\right)\\
&+ \frac{v}{\lambda^2}\left((3vv'\theta' + v^2\theta'')\{-\sin\theta, \cos\theta\}\right) && (3.20)
\end{aligned}$$

From the equations above, the expressions for tangential acceleration a_T and normal acceleration a_N are

$$a_T = \ddot{\mathbf{r}} \cdot \mathbf{T} = \frac{vv'}{\lambda}. \tag{3.21}$$

$$a_N = \ddot{\mathbf{r}} \cdot \mathbf{N} = \frac{v^2\theta'}{\lambda} \tag{3.22}$$

The tangential jerk $j_{\scriptscriptstyle T}$ is

$$j_{\scriptscriptstyle T} = \dddot{\mathbf{r}} \cdot \mathbf{T} = \frac{v}{\lambda^2}(v'^2 + vv'' - v^2\theta'^2) \tag{3.23}$$

and the the normal jerk $j_{\scriptscriptstyle N}$ is

$$j_{\scriptscriptstyle N} = \dddot{\mathbf{r}} \cdot \mathbf{N} = \frac{v^2}{\lambda^2}(3v'\theta' + v\theta''). \tag{3.24}$$

Here $\mathbf{N}$ is the direction normal to the tangent (rotated $\frac{\pi}{2}$ anti-clockwise). The signed curvature is given by

$$\kappa(u) = \frac{\theta'}{\lambda} \tag{3.25}$$

We can use the Equations 3.23 and 3.24 to express the total discomfort

$$J(\mathbf{r}, \tau) = \int_0^\tau dt + w_{\scriptscriptstyle T} \int_0^\tau (\dddot{\mathbf{r}} \cdot \mathbf{T})^2 dt + w_{\scriptscriptstyle N} \int_0^\tau (\dddot{\mathbf{r}} \cdot \mathbf{N})^2 dt \tag{3.26}$$

in terms of v, θ, and λ. First, we express the travel time τ in terms of the primary unknowns.

$$\tau = \int dt = \int_0^1 \frac{dt}{du} du = \int_0^1 \frac{\lambda}{v} du. \tag{3.27}$$

Using a similar change of variables in the integration ($t \to u$), the total discomfort can be written as

$$J(v, \theta, \lambda) = \int_0^1 \frac{\lambda}{v} du + w_{\scriptscriptstyle T} \int_0^1 \frac{v}{\lambda^3}(v'^2 + vv'' - v^2\theta'^2)^2 du + w_{\scriptscriptstyle N} \int_0^1 \frac{v^3}{\lambda^3}(3v'\theta' + v\theta'')^2 du. \tag{3.28}$$

The first integral (J_τ) is the total time, the second integral ($J_{\scriptscriptstyle T}$) is total squared tangential jerk, and the third integral ($J_{\scriptscriptstyle N}$) is total squared normal jerk.

Note that except for the term due to total travel time, the primary variables v and θ and their derivatives enter the total discomfort expression polynomially.

The discomfort J is now a function of the primary unknown functions v, θ, and a scalar λ, the trajectory length. All references to time t have disappeared. However, once the unknowns are found via optimization, we must compute a mapping between t and u. This can be done using Equation (3.13).

3.8 Conditions on v and θ for a Finite Discomfort

Now that we have a concrete expression for the discomfort J in Equation (3.28), it can be used to define the function spaces to which v and θ can belong so that the discomfort is well-defined (finite). This will, in turn, lead to conditions on the physical quantities for safe and comfortable motion. We have two distinct cases depending on whether the speed is zero at an end-point on not.

3.8.1 Conditions for Positive Speeds

Let $\Omega = [0,1]$ and $H^2(\Omega)$ be the Sobolev space of functions on Ω with square-integrable derivatives of up to order 2. Let $f : \Omega \to \mathbb{R}$. Then

$$f \in H^2(\Omega) \overset{\text{def}}{\iff} \int_\Omega \left(\frac{d^j f}{dx^j} \right)^2 dx < \infty \; \forall \; j = 0, 1, 2. \tag{3.29}$$

First, we show that if $v, \theta \in H^2(\Omega)$, then the integrals of squared tangential and normal jerk are finite. Using the Sobolev embedding theorem [1] it can be shown that if $f \in H^2(\Omega)$, then $f' \in C^0(\Omega)$ and by extension $f \in C^1(\Omega)$. Here $C^j(\Omega)$ is the space of functions on Ω whose up to j^{th} derivatives are bounded and continuous. Thus, if $v, \theta \in H^2(\Omega)$, then all the lower derivatives are bounded and continuous. Physically this means that quantities like the velocity, acceleration, and curvature are bounded and continuous $-$ all desirable properties for a smooth and comfortable motion.

Expanding all the jerk related terms in Equation (3.28), bounding all the non-second derivative terms by a constant using the results from the Sobolev embedding theorem, we immediately see that the jerk part of discomfort is finite if $v, \theta \in H^2(\Omega)$. This is a sufficient condition only and not a necessary one as we shall see below.

We also need that the inverse of v be integrable so that J_τ is finite. This is trivially true if v is uniformly positive, that is, $v \geq \overline{v} > 0$ for some constant positive $\overline{v}$ throughout the interval $[0, 1]$. However, v can be zero at one or both end-points because of the imposed conditions. Section 3.9 analyses the boundary conditions in detail. Here we assume that speed on both end-points is positive. The cases with zero end-point speed are treated below in Section 3.8.2.

Thus, consider the case that v is positive on both end-points. Since v is speed and always non-negative, it can approach zero from above only. We make a justifiable assumption that v can be zero only at end-points if at all and not in the interior. Otherwise, the wheelchair would stop and then start again. This is costly for discomfort since it increases travel time and leads to acceleration and deceleration. Of course, we *can* choose a motion in which $v = 0$ in the interior and it can still be a valid motion with finite discomfort. The assumption is that the trajectory that actually minimizes discomfort will not have a halt in between. Thus, if $v > 0$ on end-points it remains uniformly positive in the interior the discomfort is finite.

3.8.2 Conditions for Zero Speed on Boundary

Consider the case in which $v(0) = 0$. The case $v(1) = 0$ can be treated in a similar manner. If $v(0) = 0$, $\frac{1}{v}$ must not blow up faster than $\frac{1}{u^p}$ where $p < 1$. This is to keep J_τ finite. This can be seen as follows. Lets assume $v(u) = u^p$ for some $p > 0$ (so that $v(0) = 0$). This implies that $J_\tau = \frac{\lambda}{1-p}$ provided $p < 1$, otherwise it is not defined.

For simplicity, assume a 1D motion so that $\theta(u) \equiv 0$. Then $J_T = \frac{1}{\lambda^3}\frac{(1-2p)^2p^2}{5p-3}$ provided $p > \frac{3}{5}$. Taking all conditions into account, if $v(0) = 0$, the discomfort is finite if $v(u)$ behaves like u^p where $\frac{3}{5} < p < 1$. However, in such a case, $\int_0^1 v''^2 du = \frac{(-1+p)^2p^2}{2p-3}$ is defined and finite only if $p > 3/2$. This conflicts with the assumption that $v \in H^2(\Omega)$. Thus, we can have a finite discomfort even if $v \notin H^2(\Omega)$. We see that the reason for this is the zero speed boundary condition which leads to $\int_0^1 v^3 v''^2 du$ being finite for $\frac{3}{5} < p < 1$ even though $\int_0^1 v''^2 du$ (which is the highest order term in J_T) is not finite for such a range of p.

If we look at the integral $J_T = \frac{1}{\lambda^3}\frac{(1-2p)^2p^2}{5p-3}$ carefully, we see that it can be finite even if $p < \frac{3}{5}$, provided $p = \frac{1}{2}$. This is a special case because $vv'' + v'^2$

is identically zero for such a p and tangential jerk discomfort is finite for a 1-D motion.

For a mathematically meaningful problem we must treat zero speed boundary conditions separately from non-zero speed boundary condition. This analysis will be done in more detail in Section 3.9 and Section 4.1.3 which are focused on boundary conditions and appropriate singular finite elements respectively.

3.8.3 Summary

To summarize, the total discomfort is finite if $v, \theta \in H^2(\Omega)$ and the inverse of v is integrable. Inverse of v is integrable if v is uniformly positive in $[0, 1]$. If zero speed boundary conditions are imposed, we will have to choose v outside $H^2(\Omega)$. In such a case, at $u = 0$, it is sufficient that v approaches zero as u^p where $\frac{3}{5} < p < 1$ or $p = \frac{1}{2}$. For the right end point, where $u = 1$, replace u with $(1-u)$ in the condition. We do not lose higher regularity of v throughout the interval Ω just because $v \notin H^2(\Omega)$. Assume $v > 0$ in the interior, as justified above. Then $v \geq \overline{v} > 0$ in $\Omega_\delta \overset{\text{def}}{=} [\delta, 1-\delta]$ where $\delta = \delta(\overline{v}) > 0$. Thus $v \in H^2(\Omega_\delta)$ is necessary to keep total discomfort finite. This implies continuity and boundedness of velocity and acceleration in Ω_δ $\forall \delta > 0$. For zero speed boundary condition, a similar division of the interval $[0, 1]$ into pieces will be necessary to create the finite element mesh.

3.9 Analysis of Boundary Conditions

The expression for the cost functional J in Equation (3.28) shows that the highest derivative order for v and θ is two. Thus, for the boundary value problem to be well-posed we need two boundary conditions on v and θ at each end-point $-$ one on the function and one on the first derivative.

We also have to impose that the robot move from a specific starting point to a specific ending point. This condition is a set of two equality constraints on λ and θ based on Equation (3.17). If the motion is from positions $\mathbf{r}_0$ to $\mathbf{r}_\tau$, then

$$\mathbf{r}_\tau - \mathbf{r}_0 = \lambda \left\{ \int_0^1 \cos\theta \, du, \int_0^1 \sin\theta \, du \right\}. \tag{3.30}$$

We now relate the mathematical requirement on v and θ boundary values above to expressions of physical quantities. We do this for the starting point only. The ending point relations are analogous.

3.9.1 Positive Speed on Boundary

First, consider the case when $v > 0$ on the starting point. The speed v needs to be specified, which is quite natural. The u-derivative of v, however, is not tangential acceleration. The tangential acceleration is the t-derivative and is given by Equation (3.21). It is $\frac{vv'}{\lambda}$. Here v is known but λ is not. Thus specifying tangential acceleration gives us a constraint equation and not directly a value for $v'(0)$. This is imposed as an equality constraint. Similarly, fixing a value for θ on starting point is natural. We "fix" the values of $\theta'(0)$

by fixing the signed curvature $\kappa = \frac{\theta'}{\lambda}$. As before, this leads to an equality constraint relating $\theta'(0)$ and λ if $\kappa \neq 0$. Since choosing a meaningful non-zero value of κ is difficult, it is natural to impose $\kappa = 0$. In this case $\theta'(0) = 0$ can be imposed easily.

3.9.2 Zero Speed on Boundary

We now discuss the $v = 0$ case. If $v(0) = 0$, then, as seen in Section 3.8.2, $v(u)$ must behave like u^p for $\frac{3}{5} < p < 1$ or $p = \frac{1}{2}$ near $u = 0$ and $v'(u) \sim u^q$ for $-\frac{2}{5} < q < 0$ or $q = -\frac{1}{2}$ respectively. This means the $\lim_{u \to 0} v'(u)$ is infinite. This leads to a difficulty in analyzing the expression for the tangential acceleration $(\frac{vv'}{\lambda})$ without using limits. We prove that if $v \sim u^p$ at boundary, then the tangential acceleration is 0 if $\frac{3}{5} < p < 1$ and it is finite but non-zero if $p = \frac{1}{2}$. If $v(u) \sim u^p$, then, $vv' \sim u^{2p-1}$. If $\frac{3}{5} < p < 1$, it means $\frac{1}{5} < 2p - 1 < 1$. Thus as $u \to 0$, $vv' \to 0$ because of the allowable range of p. If $p = \frac{1}{2}$, vv' behaves like a positive constant as $u \to 0$. Hence $p = \frac{1}{2}$ corresponds to non-zero tangential acceleration.

We still have to decide with what strength does $v'(u)$ tend to infinity at an end-point. If $v = 0$ and $a \neq 0$, it is clear that $v'(u) \sim u^{-1/2}$. If $v = 0$ and $a = 0$, The analysis above has only shown that $\lim_{u \to 0} v(u)v'(u) = 0$, and $\lim_{u \to 0} v'(u) = \infty$. In this case, we need to use the time domain. The reason we have such a singularity is because of working in the arc-length domain. Consider starting from origin with zero velocity and acceleration at zero time (t) in 1D. Expanding the distance traveled (s) as a function of time, we see that

$$s(t) = 0 + 0t + \frac{1}{2}0t^2 + \frac{1}{6}jt^3 + \dots.$$

Here $j > 0$ is the jerk at $t = 0$. We ignore the higher order terms. Then, to the lowest power of t, the velocity as a function of t is

$$v(t) = \frac{1}{2}jt^2.$$

Eliminating t to relate v and s, we get

$$v = \frac{6^{2/3}}{2}j^{1/3}s^{2/3}.$$

Now $s = \lambda u$ because u is the scaled arc-length parameter. Using this we get $v = Cu^{2/3}$, where all the constants are absorbed in C. Thus, $v(u) \sim u^p$ for $p = \frac{2}{3}$. This value of p is within the acceptable range of p, the open interval $(\frac{3}{5}, 1)$. This also tells us that

$$v'(u) \sim u^{-1/3} \tag{3.31}$$

is the appropriate strength of the singularity. This will be crucial in designing the singular finite elements on the boundary in Section 4.1.3.

3.9.3 Summary

To summarize, one must specify the starting and ending poses, orientations, and curvatures. For the motion, one must specify the speeds. If a specified speed is non-zero, the tangential acceleration must be specified. If the speed is zero, the tangential acceleration can be zero. If tangential acceleration is non-zero, it must be be positive if it is starting point or must be negative if it is the ending point.

3.10 Obstacle Avoidance

For safe motion, it is necessary that the robot avoid obstacles while navigating. Simply speaking, obstacles are regions in the plane of motion through which the geometric path must not pass. This simple notion can be translated to mathematically posed constraints in a variety of ways. For example, convex polygons, rectangular cells, simple closed shapes like ellipses, or level sets of implicitly defined simple functions of two arguments are some possibilities.

3.10.1 Modeling Obstacles as Star-shaped Domains

We have chosen to model the "forbidden" region formed by the obstacles as a union of star-shaped domains with boundaries that are closed curves with piecewise continuous second derivative. A set in $\mathbb{R}^n$ is called a star-shaped domain if there exists at least one point $\mathbf{x}_0$ in the set such that the line segment connecting $\mathbf{x}_0$ and $\mathbf{x}$ lies in the set for all $\mathbf{x}$ in the set. Intuitively this means that there exists at least one point in the set from which all other points are "visible". We will refer to such a point $\mathbf{x}_0$ as a *center* of the star-shaped domain.

The choice of using star-shaped domains is made so that each point on the boundary of an obstacle can be treated as coming from a well-defined function in polar coordinates centered within the particular obstacle. See Figure 3.3. This also allows treatment of non-convex obstacles without subdividing them into a union of convex shapes. A big advantage is that we reduce the number of imposed constraints since the number of inequality constraints is proportional to the number of obstacles. This leads to a faster optimization process.

This approach is a special case of using level sets of an implicitly defined function as an obstacle boundary. What is different here is that given the description of the boundary in polar coordinates, which is easy to specify for common shapes, we *construct* an implicit function (see the following section). This is done based on the assumption that the boundary encloses a star-shaped region. The piecewise smoothness property is required to impose the obstacle constraint in a numerical optimization method. Since up to second derivative of constraint can be required, the obstacle boundary should also be smooth to that order (or at least piecewise smooth).

If an obstacle within the map is not star-shaped, our framework can still handle it if it can be expressed as a finite union of piecewise smooth star-shaped domains. Efficient algorithms to decompose any polygon into a finite number of star-shaped polygons exist [3], but it is unknown if any star-shaped domain can be decomposed in such a way.

3.10.2 Incorporating Constraints for Obstacle Avoidance

We now derive a function for the inequality constraint that a given point in the plane is not inside the boundary of one star-shaped obstacle. It is easy to extend this to multiple points and multiple obstacles by just repeating the inequality with different parameters.

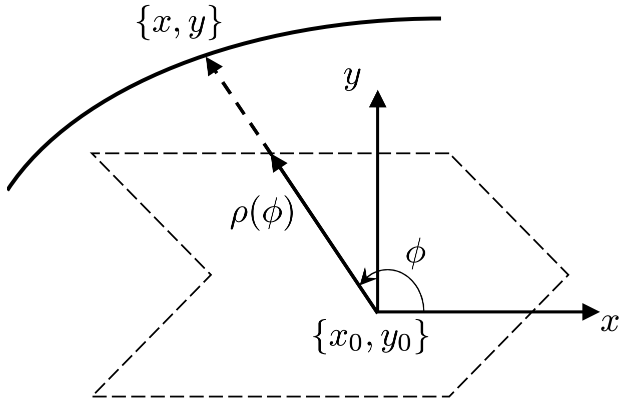


Figure 3.3: Notation for star-shaped obstacles.
A non-convex star-shaped obstacle is shown with its "center" $\{x_0, y_0\}$ and a distance function $\rho = \rho(\phi)$. The distance function gives a single point on the boundary for $\phi \in [0, 2\pi]$. The robot trajectory must lie outside the obstacle.

Let an obstacle be specified by its boundary in polar coordinates that are centered at $\mathbf{r}_0 = \{x_0, y_0\}$. Each $\phi \in [0, 2\pi)$ gives a point on the boundary using the distance $\rho(\phi)$ from the obstacle origin. The distance function ρ must be periodic with a period 2π. See Figure 3.3.

Suppose we want a point $\mathbf{r} = \{x, y\}$ to be outside the obstacle boundary. Define $C(\mathbf{r})$ as

$$C(\mathbf{r}) = ||\mathbf{r} - \mathbf{r}_0||_2 - \rho(\text{arctan2}(\mathbf{r} - \mathbf{r}_0)) \tag{3.32}$$

where the subscript 2 refers to the Euclidean norm. It is obvious that $C(\mathbf{r}) \geq 0 \iff$ the point $\mathbf{r}$ is outside the obstacle. This can be seen using a 1D graph of $\rho(\phi)$. For example, let an obstacle be represented as shown in Figure 3.4(a). Figure 3.4(b) shows the same obstacle flattened out as a 1D curve. Then $C(\mathbf{r})$ is positive in the top region and negative below. The star-shaped property leads to a single-valued curve $\rho(\phi)$ when flattened like this. The vector $\mathbf{r}$ is related to the primary variables in trajectory optimization problem using Equation (3.17).

3.10.3 Derivatives of Obstacle Avoidance Constraint

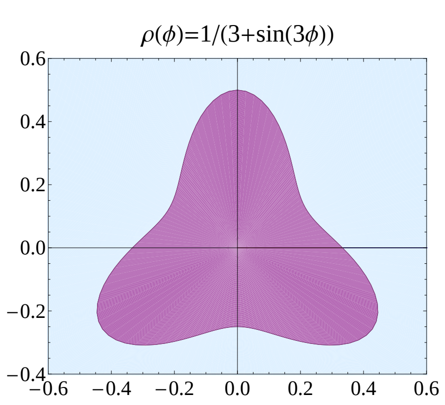
We will need derivatives of $C(\mathbf{r})$ with respect to $\mathbf{r}$ for incorporating $C(\mathbf{r}) \geq 0$ as a constraint in the trajectory optimization problem. Here $\mathbf{r}$ is any point on the path that we want to lie outside a given obstacle. We can derive the following expressions for first and second derivatives of $C(\mathbf{r})$. The derivatives of ρ below are evaluated at $\phi = \text{arctan2}(\mathbf{r} - \mathbf{r}_0)$. To simplify the expressions, $x, y, \mathbf{r}$ refer to the offsets from obstacle origin $\mathbf{r}_0$ instead of absolute positions in the plane.

$$\frac{\partial C}{\partial \mathbf{r}} = \frac{\mathbf{r}}{||\mathbf{r}||_2} - \rho'(\phi)\frac{\{-y, x\}}{||\mathbf{r}||_2^2} \tag{3.33}$$

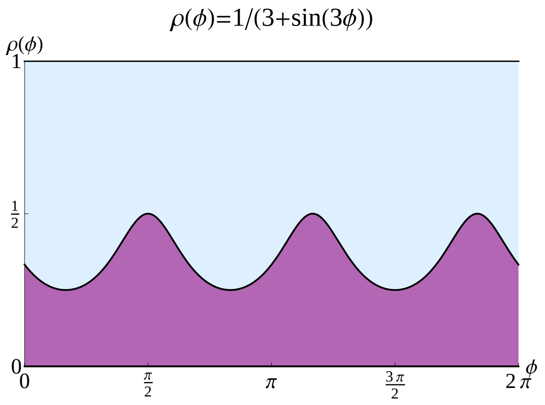
$$\frac{\partial^2 C}{\partial \mathbf{r}^2} = \frac{1}{||\mathbf{r}||_2^3}\left(1 - \frac{\rho''(\phi)}{||\mathbf{r}||_2}\right)\begin{bmatrix} y^2 & -xy \\ -xy & x^2 \end{bmatrix} - \frac{\rho'(\phi)}{||\mathbf{r}||_2^4}\begin{bmatrix} 2xy & y^2 - x^2 \\ y^2 - x^2 & -2xy \end{bmatrix} \tag{3.34}$$

Obviously, the second derivative is a 2×2 matrix.

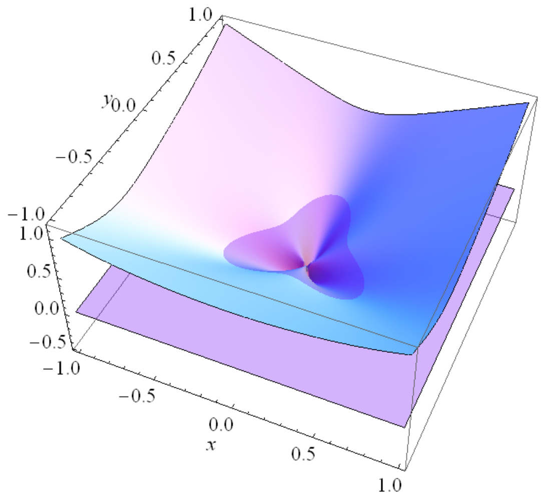
The constraint function $C(\mathbf{r})$ is piecewise differentiable for all $\mathbf{r}$ except at a single point $\mathbf{r} = \mathbf{r}_0$. If $\mathbf{r} = \mathbf{r}_0$ by chance, which is easily detectable, we know that the $\mathbf{r}$ is inside the obstacle and can perturbed to avoid this undefined behavior. Note that $C(\mathbf{r})$ remains bounded inside the obstacle. It is the derivatives that are not bounded as $\mathbf{r} \to \mathbf{r}_0$ Figure 3.4(c) shows a surface



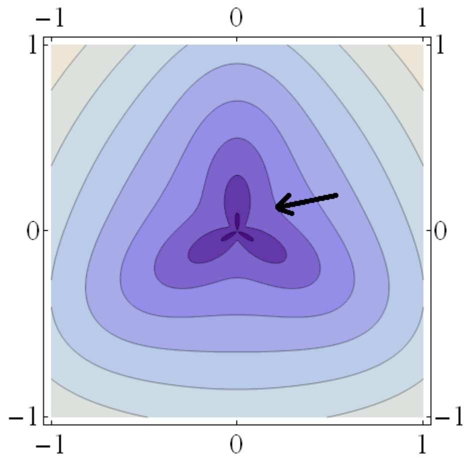
(a) Obstacle shape



(b) Obstacle as a 1-D curve



(c) Surface plot of constraint



(d) Level sets of constraint

Figure 3.4: Obstacle and constraint plots.
The figures show an obstacle in polar coordinates in (a), and its 1-D representation in (b). The region with darker shade is the interior and a feasible trajectory must not pass through it. The surface plot of the corresponding constraint function $C(\mathbf{r})$ of Equation (3.32) is shown in (c) and its level set is shown in (d). The arrow marks the zero level set, which is the obstacle boundary.

plot of $C(\mathbf{r})$ for the obstacle shown in Figure 3.4(a). The contours of constant values are shown in Figure 3.4(d).

3.10.4 Incorporating Robot Shape

The discussion on obstacle avoidance constraints so far has assumed a point robot. In reality, the robot is not a point. To impose obstacle avoidance constraints in this case, the robot can be modeled as a closed curve that encloses the projection of its boundary in the plane of motion. We can choose a set of points on this curve and impose the constraint that all these points be outside all obstacles. The distance between any pair of points can be smaller than the smallest obstacle. We have currently not implemented this and this is part of future work.

3.11 The Full Nonlinear Constrained Optimization Problem

We now summarize the nonlinear and constrained trajectory optimization problem taking into account all input parameters, all the boundary conditions, and all the constraints. This is the "functional" form of the problem (posed in function spaces). We will present an appropriate discretization procedure valid for all input combinations in the next chapter.

Minimize the discomfort functional J, where

$$J(v,\theta,\lambda) = \int_0^1 \frac{\lambda}{v} du + w_T \int_0^1 \frac{v}{\lambda^3}(v'^2 + vv'' - v^2\theta'^2)^2 du + w_N \int_0^1 \frac{v^3}{\lambda^3}(3v'\theta' + v\theta'')^2 du,$$

given the following boundary conditions for both starting point and ending point

- position $(\mathbf{r}_0,\, \mathbf{r}_\tau)$,
- orientation $(\theta_0,\, \theta_\tau)$,
- signed curvature $(\kappa_0,\, \kappa_\tau)$,
- speed $(v_0 \geq 0,\, v_\tau \geq 0)$,
- tangential acceleration $(a_{T,0},\, a_{T,\tau})$,

and constraints on allowable range of

- speed $(v_{min} = 0, v_{max})$,
- tangential acceleration $(a_{T,min}, a_{T,max})$,
- normal acceleration $(a_{N,min}, a_{N,max})$,
- angular speed $(\omega_{min}, \omega_{max})$,
- curvature, if necessary $(\kappa_{min} = 0, \kappa_{max})$,

and

- number of obstacles N_{obs},
- locations of obstacles $\{\mathbf{c}_i\}_{i=1}^{N_{obs}}$

- representation of obstacles that allows computation of $\{\rho_i(\phi)\}_{i=1}^{N_{obs}}$, for $\phi \in [0, 2\pi)$

and

- an initial guess for $(v(u), \theta(u), \lambda)$, in $u \in [0, 1]$,
- weights $w_T > 0$ and $w_N > 0$.

The constraint on starting and ending position requires that

$$\mathbf{r}_T - \mathbf{r}_0 = \lambda \left\{ \int_0^1 \cos\theta \, du, \int_0^1 \sin\theta \, du \right\}$$

Staying outside all obstacles requires that

$$||\mathbf{r}(u) - \mathbf{c}_i||_2 - \rho_i(\text{arctan2}(\mathbf{r}(u) - \mathbf{c}_i)) \geq 0 \;\forall\; i \in 1, \ldots, N_{obs}, \text{ and } \forall\; u \in [0, 1]$$

where

$$\mathbf{r}(u) = \mathbf{r}(0) + \lambda \left\{ \int_0^u \cos\theta \, du, \int_0^u \sin\theta \, du \right\}.$$

As a post-processing step, we compute time t as a function of u using

$$t = t(u) = \int_0^u \frac{\lambda}{v(u)} du$$

and convert all quantities ($v, \theta, \mathbf{r}$, and their derivatives) from u domain to t domain.

Chapter 4

A Finite Element Discretization of the Trajectory Optimization Problem

In the previous chapter, we posed a constrained nonlinear optimization problem to compute a safe, comfortable, and customizable motion of a wheelchair moving in a plane. We showed that we must be able to impose two kinds of boundary conditions. In the first kind, the problem is set in the Sobolev space of functions whose up to second derivatives are square-integrable. In the second kind, we must allow functions that are singular at the boundary (with a known strength) but still lie in the same Sobolev space in the interior.

This optimization problem is infinite dimensional since it is posed on infinite dimensional function spaces. This means we must discretize it as a finite dimensional problem before it can be solved numerically. Keeping the problem setting and requirements mentioned above in mind, it is natural to use the Finite Element Method (FEM) to discretize it.

In this chapter, we show how to use an appropriate finite dimensional subspace for both kinds of boundary conditions. We also show the sparsity structure of the Hessian of the global problem. Some of the discretized inequality and inequality constraints, if computed naively, lead to a dense global Hessian. We avoid this and keep the global Hessian sparse by introducing auxiliary variables.

4.1 Finite Element Discretization

We first discuss the case when there is no singularity in the speed v. This is the case when the given boundary speeds are positive. In this case, $v \in H^2(\Omega)$ as shown in Section 3.8.2, where $\Omega = [0,1]$. We assume that $\theta \in H^2(\Omega)$ always (whether v is singular or not) because it is sufficient for the discomfort to be finite. Thus, to discretize the problem, it is natural to use the basis functions in $C^1(\Omega)$, the space of functions that are continuous and have continuous first derivatives. This makes the second derivative of v and θ discontinuous but its square is still integrable.

4.1.1 Basis Functions

We minimize the discomfort and satisfy all the constraints in a finite dimensional subspace of $C^1(\Omega)$. We make the following choice.

$$v^h(u) = \sum_{i=1}^{N} \alpha_i^v \chi_i(u) \tag{4.1}$$

$$\theta^h(u) = \sum_{i=1}^{N} \alpha_i^\theta \chi_i(u) \tag{4.2}$$

Here $\chi_i(u) \in C^1(\Omega)$ are basis functions for this problem. The symbol h traditionally denotes a measure of the "mesh width" to distinguish the approximate solution from the "exact" infinite dimensional solution. The unknown scalar values α_i^v and α_i^θ for $i = 1, \ldots, N$ are the degrees of freedom (DOFs) which are to be determined via "solving" the optimization problem of Section 3.11. For reasonable choices of $\chi_i(u)$, as N increases the finite dimensional solution approaches the exact solution.

Usually, one chooses $\chi_i(u) \in \Omega$ that are easy (and cheap) to compute, and have local support. Piecewise polynomial functions (that have sufficient differentiability) are good candidates. By having local support, we mean the functions are non-zero only over a limited interval and not on whole Ω. This has two main advantages. First, while performing integration to compute J, the product of χ_i and χ_j is zero over most of the interval. This leads to $O(n)$ rather than $O(n^2)$ interactions. Second, the global Hessian matrix is sparse rather than being dense. Typically for 1D problems, the matrix has a small constant band-width.

For our problem, we first divide the interval $[0, 1]$ into n equal-size intervals of length $h = \frac{1}{n}$. Each of these intervals is an *element*. Thus we have n elements and $n + 1$ equidistant points or *nodes*. The collection of elements and nodes is the finite element *mesh*.

At each node we define two piecewise polynomial functions that are non-zero only on the two elements surrounding the node. This is the standard cubic Hermite basis for problems posed in the H^2 space in 1D. See Figure 4.1 for both the functions and their first and second derivatives. The first kind of basis function is 1 on the node with which it is associated and has zero derivative there. The second function is zero on the corresponding node and has unit derivative there. Additionally, both are zero with first derivatives also zero on two surrounding nodes. Thus, in all, we have $2(n+1)$ basis functions (and unknown scalars) each for v and θ. Because of the above-mentioned properties each basis functions belongs to $C^1(\Omega) \subset H^2(\Omega)$. Note that the basis functions on the boundary points are slightly different. They are not extended outside the interval. Figure 4.2 shows 5 basis functions of each kind for $n = 4$. As seen, the boundary basis functions are truncated. It would not matter anyway what their values outside the interval are since no integration is performed outside. The other basis functions are translations of each other.

4.1.2 Element Shape Functions

In FEM practice, we define basis functions in terms of "shape functions". The shape functions are defined only on a single reference element and multiple shape functions placed on neighboring elements are joined to create a single basis function. This requires that shape functions have appropriate values (and derivative) on reference element boundary so that the basis functions are valid for the problem.

Figure 4.3 shows the four cubic Hermite shape functions on a single

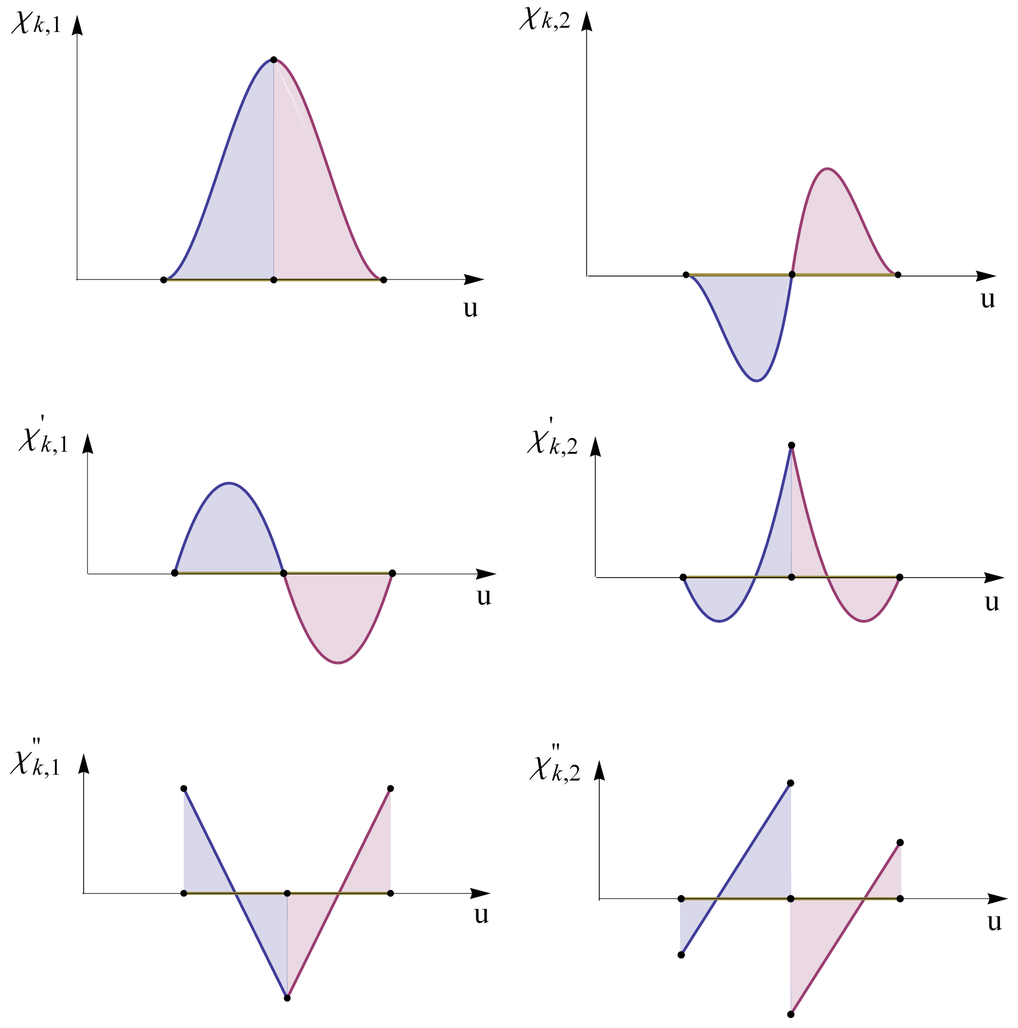


Figure 4.1: Cubic Hermite basis functions and their first and second derivatives.
Two kinds of basis functions are defined at each node such that each is zero everywhere except on the two elements sharing the node. The first kind, $\chi_{k,1}$ has a value of 1 and a zero derivative at the associated node k. Its value and derivatives are zero on both adjacent nodes. The second kind, $\chi_{k,2}$ has a value of zero and a unit derivative at the associated node k. Its value and derivatives are zero on both adjacent nodes. Both $\chi_{k,1}$ and $\chi_{k,2}$ are square integrable on $u \in [0, 1]$.

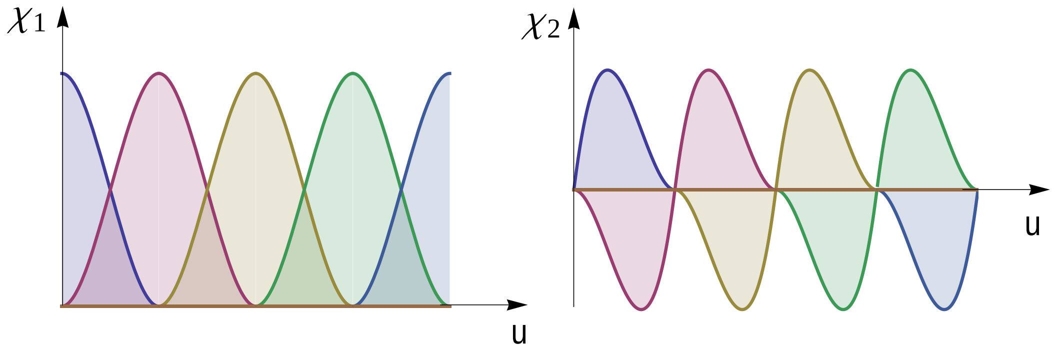


Figure 4.2: Cubic Hermite basis functions on five consecutive nodes. The two kinds of basis functions on any node are translations of the respective basis function of Figure 4.1. The basis functions on a boundary node are truncated.

reference element $[0, 1]$. The shape functions are

$$\begin{aligned}\phi_1(x) &= (x-1)^2(1+2x)\\ \phi_2(x) &= (x-1)^2x\\ \phi_3(x) &= (3-2x)x^2\\ \phi_4(x) &= (x-1)x^2\end{aligned}$$

For maintaining basis function continuity, each ϕ_i satisfies $\phi_i(0) = \phi_i'(0) = \phi_i(1) = \phi_i'(1) = 0$, except $\phi_1(0) = \phi_2'(0) = \phi_3(1) = \phi_4'(1) = 1$.

4.1.3 Singular Shape Functions at Boundary

For the case when either one or both the boundary points have zero speed specified, $v(u)$ is singular at the corresponding boundary with $v'(u)$ infinite with singularity $u^{-1/3}$ when acceleration is also zero and $u^{-1/2}$ if accel-

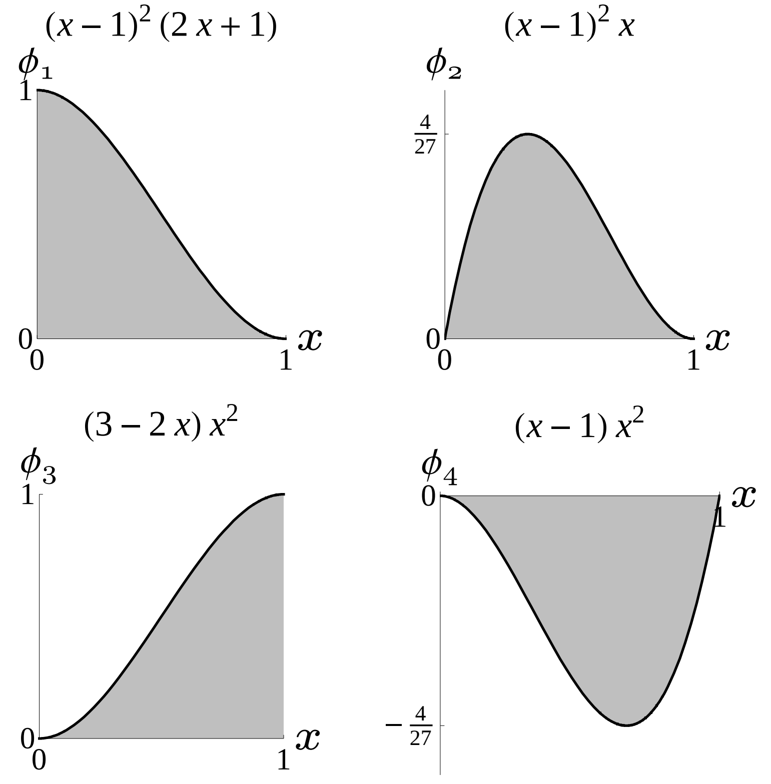


Figure 4.3: Cubic Hermite shape functions on a reference element. Four shape functions are defined on each element. Each is a cubic polynomial on $x = [0, 1]$ where x is the local coordinate on the element.

eration is non-zero. Thus, v as a function does not belong to $H^2(\Omega)$. However, v does belong to H^2 in the interior as shown in Section 3.8.2.

After a FEM mesh is decided, we take this into account and do not use the above-mentioned regular shape functions for v on the element(s) near the boundary with zero speed. For the interior elements, however, no change is done and the regular shape functions are used.

We now derive the new singular shape functions for the left boundary ($u \in [0, h]$) element and the singular shape functions for the right boundary element can be derived using symmetry.

We need at least two shape functions so that the two shape functions coming from the $[h, 2h]$ element can be matched. Denote them by ψ_1^L and ψ_2^L. The function value and the function derivative both must be matched at h. For a reference element shape function, this means $\psi_1^L(1) = 1, \psi_1^{L\prime}(1) = 0, \psi_2^L(1) = 0, \psi_2^{L\prime}(1) = 1$. Both ψ_1^L and ψ_2^L must be zero on $u = 0$ because the speed is zero there. Thus, the Dirichlet boundary condition is imposed explicitly. Finally, as $x \to 0$, one of the functions must behave as x^p, for $p = \frac{2}{3}$ or $p = \frac{1}{2}$, to match the singularity in Equation (3.31). It can be seen that the choice

$$\begin{aligned}\psi_1^L(x) &= x^p + p(1-x)x \\ \psi_2^L(x) &= (x-1)x\end{aligned}$$

satisfies all these requirements. Figure 4.4 shows the four shape functions, two for the left element and two for the right element for the case $p = \frac{2}{3}$. Figure 4.5 shows that ψ_1^L and ψ_1^R are singular when approaching the boundary (shown for $p = \frac{2}{3}$ only). The other two functions ψ_2^L and ψ_2^R are smooth.

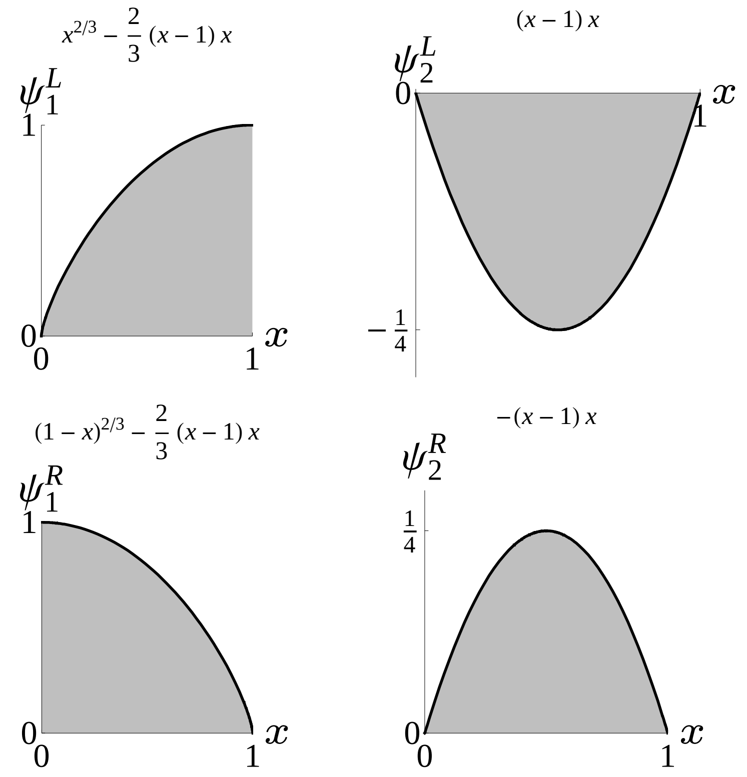


Figure 4.4: Singular shape functions on boundary elements for zero speed and zero acceleration boundary conditions.

Two singular shape functions are defined on a boundary element. Consider zero speed condition on left element. The first shape function ψ_1^L has a value of 1 and zero derivative on the right to match the shape function ϕ_1 of Figure 4.3 on the next element. The second shape function ψ_2^L has a value of 0 and a unit derivative on the right to match the shape function ϕ_2 of Figure 4.3 on the next element. Both ψ_1^L and ψ_2^L have a value of zero on the left because speed is zero. The singular shape functions for zero speed boundary conditions on right are similarly defined.

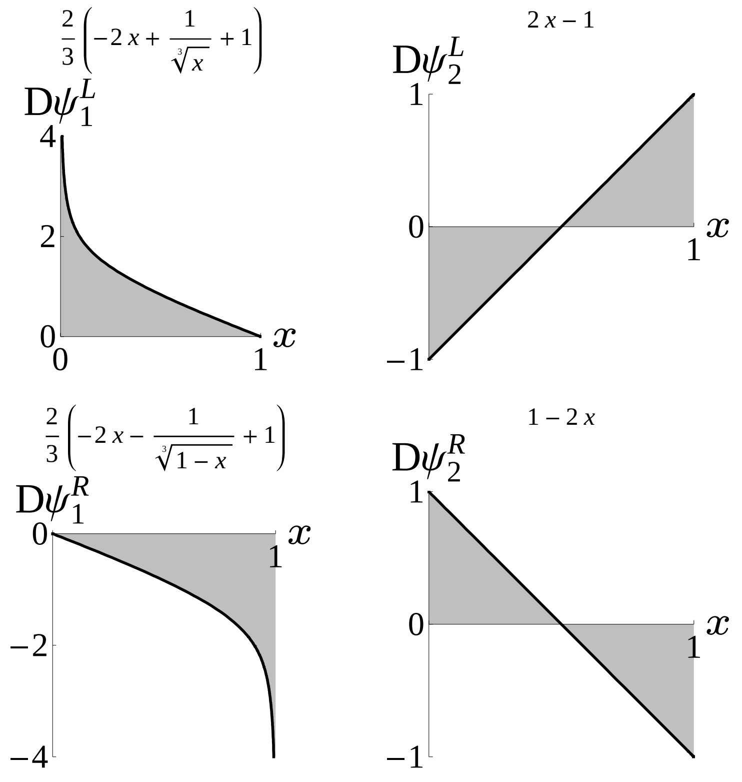


Figure 4.5: First and second derivatives of singular shape functions for zero speed and zero acceleration boundary conditions.
ψ_1^L and ψ_1^R tend to infinity as $x^{-\frac{2}{3}}$ as they approach the boundary. ψ_2^L and ψ_2^R are smooth.

4.2 The Finite-dimensional Optimization Problem

With the choice of basis functions described above, we can express $v^h(u)$ and $\theta^h(u)$ as:

$$v^h(u) = \sum_{i=1}^{n+1} v_i \chi_{i,1}(u) + \sum_{i=1}^{n+1} v_i' \chi_{i,2}(u) \quad (4.3)$$

$$\theta^h(u) = \sum_{i=1}^{n+1} \theta_i \chi_{i,1}(u) + \sum_{i=1}^{n+1} \theta_i' \chi_{i,2}(u) \quad (4.4)$$

where v_i, v_i', θ_i, and θ_i' for $i = 1, \ldots, n$ are the (unknown) nodal values and $\chi_{i,1}(u)$ and $\chi_{i,2}(u)$ are the two kinds of basis functions described in the previous section.

For optimization, the values of cost, its gradient and Hessian, the values of constraints, and the gradient and Hessian of each constraint are required. For efficiency, it is desirable that cost and constraint Hessians be sparse. We will see later that for the Hessian of obstacle avoidance constraints to be sparse, it is useful to introduce $2N$ additional unknowns in the form of position $\mathbf{r}_i = \{x_i, y_i\}_{i=1}^{N}$ at N points.

Thus, our objective now is to determine the values of these unknowns and the unknown path length λ that minimize the cost functional and satisfy the boundary conditions and constraints described in (Section 3.11).

4.2.1 Numerical Integration for Computing the Integrals in the Cost Functional and Constraints

We use Gauss quadrature formulas to compute the integrals in the cost functional and constraints. When using m integration points in an interval, the formulas are accurate for polynomials of degree up to $2m - 1$. For the regular C^1 basis functions, which have maximum polynomial degree 3, it is easily seen that the square tangential jerk is a polynomial of degree 23, and the squared normal jerk is a polynomial of degree 17. See Equation (3.28). These are polynomials in u and not t. Hence, 12 Gauss points will give exact integrals up to floating point accuracy. Of course, the integrands being polynomials, the integrals corresponding to J_T and J_N can be evaluated without using Gauss points (if one is ready to work with complex algebraic expressions). But the other integrals, for J_τ and those relating $\mathbf{r}$ to θ (Equation (3.17)), must be evaluated numerically. Hence, we use 12 Gauss points to evaluate all integrals.

4.3 Imposing Constraints

We need to impose multiple equality and inequality constraints while minimizing the cost functional. Some of the equality constraints affect a single DOF each and hence they can be used to eliminate the particular unknown. The others relate multiple DOFs and must be imposed as an equality explicitly. The equality constraints are described below.

- Fix end-point positions ($\mathbf{r}_0$, $\mathbf{r}_\tau$) by eliminating the unknowns x and y on the first and last nodes.
- Fix end-point orientations (θ_0, θ_τ) by eliminating the unknown θ on the first and last nodes.
- Relate start and end position $\mathbf{r}_\tau - \mathbf{r}_0 = \lambda \left\{ \int_0^1 \cos\theta \, du, \int_0^1 \sin\theta \, du \right\}$ (Equation (3.30)) by computing the integrals as described in Section 4.3.1, Equation (4.5).
- Fix end-point speeds ($v_0 \geq 0$, $v_\tau \geq 0$) by eliminating the unknown v on the first and last nodes.

- If speed on an end-point is positive, tangential acceleration a_T must be specified at that end-point. Impose $\frac{vv'}{\lambda} = a_T$ on that end point. Otherwise, this constraint will be automatically imposed by using singular shape functions.

- Impose specified end-point curvature κ by imposing $\theta' = \lambda\kappa$ on each end-point.

The inequality constraints that are not related to obstacles avoidance are as follows. We must maintain

- velocity in $[v_{min} = 0, v_{max}]$,

- tangential acceleration in $[a_{T,min}, a_{T,max}]$,

- normal acceleration in $[a_{N,min}, a_{N,max}]$, and

- angular velocity in $[\omega_{min}, \omega_{max}]$.

- curvature in $[\kappa_{min} = 0, \kappa_{max}]$,

Note that these must be maintained for each $u \in [0, 1]$ in the infinite dimensional optimization problem. For the discretized version, we choose the Gauss integration points and impose that these quantities remain in the specified range *only* on those points. Thus, for a mesh with n elements, each inequality above results in $12n$ constraints (assuming 12 points are used as discussed in Section 4.2.1). The values of these physical quantities on each Gauss point is a function of the DOFs on element nodes. Thus, these constraints are local. They are not affected when DOFs of non-element nodes change.

There are two important reasons to keep the values within range on Gauss points as opposed to on some other, say, uniform set of points. First, since we use v at the Gauss point to compute the integrals, it is more important that v remain non-negative there to avoid problems of large negative values of J. Second, since v and θ are already computed there it saves extra computation.

4.3.1 Obstacle Avoidance Constraints

Staying outside obstacles, if present, requires additional inequality constraints. For this we pick N uniformly separated points in the interval $[0, 1]$ and impose the constraint that each of $\mathbf{r}$ on the N points remain outside each of N_{obs} obstacles. This leads to $N \times N_{obs}$ constraints. In our implementation we make $N = nM + (n+1)$, so that if the distribution is uniform, each element has M such points in the interior and each node is a point too. Two of these N points are the boundary points which must be outside all obstacles for the optimization problem to have a feasible solution. If the robot boundary is not circular and we choose P points on the robot's boundary, then the number of constraints is $N \times N_{obs} \times P$.

We come back to obstacle related constraint relating a single obstacle and a single point on the trajectory. One could simply relate the position at the point with $\theta(u)$ and λ using Equation (3.17), and use Equation (3.32) to impose conditions on $\theta(u)$ and λ. However, because of the structure of Equation (3.32) and because $\mathbf{r}(u)$ depends on *all* θ DOFs of nodes that are before u, the Hessian of this constraint is not sparse. This would lead to efficiency problems when doing iterations in the numerical optimization process. Even computing the dense Hessian would be very costly as N_{obs} and N increase. We must work around this elimination approach of imposing the obstacle related constraints.

To avoid the dense Hessian of obstacle constraint, we make two changes to the simplistic approach. First, we do not use Equation (3.30) for eliminating $\mathbf{r}$ but keep $\mathbf{r}$ as an unknown function. Second, we relate adjacent $\mathbf{r}$'s via Equation (3.17) as follows.

$$\mathbf{r}(u_j) - \mathbf{r}(u_{j-1}) = \lambda \left\{ \int_{u_{j-1}}^{u_j} \cos\theta \, du, \int_{u_{j-1}}^{u_j} \sin\theta \, du \right\}. \tag{4.5}$$

Here j goes from 1 to $N-1$. What this change does is that, as long as adjacent $\mathbf{r}(u_j)$ and $\mathbf{r}(u_{j-1})$ belong to a maximum two adjacent elements, the equation above relates only a small number of local DOFs. Secondly, since each $\mathbf{r}(u_j)$ is now a legitimate unknown, it can be used to impose the inequality constraint Equation (3.32) without θ being involved.

This new approach does have a price, however. We have increased the number of unknowns and hence increased the size of the gradient vector and Hessian matrix. But this is a small price to pay considering that the sparsity is still maintained, the amount of computation does not grow, and equations are local in nature. We explore the sparsity pattern more in Section 4.4 ahead.

4.4 Element and Global Gradient and Hessian

An important choice in the FEM discretization of any variational problem is the ordering of all the unknowns when forming the global Hessian matrix. A good choice simplifies the assembly process as well as could lead to useful structural sparsity.

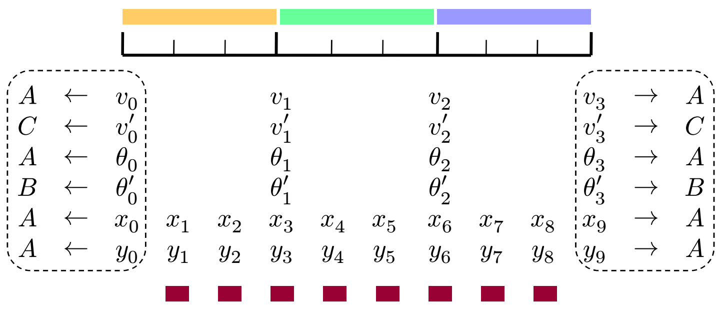
We have four kinds of DOFs. For simplicity, we discuss the regular case and where boundary conditions are not yet imposed. The singular case differs in minor details only that does not affect the ordering process. The four kinds are as follows.

- four unknowns each on $n + 1$ node $-v, v', \theta, \theta'$
- N x and N y unknowns
- a scalar unknown λ

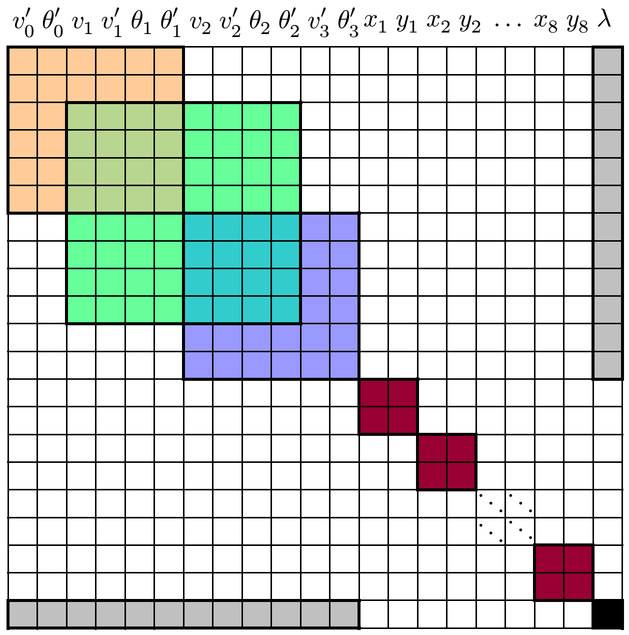
The unknowns are ordered in the same sequence shown above starting from $u = 0$ and going to $u = 1$.

The ordering chosen above means that each DOF except λ interacts with DOFs on two elements. The scaled arc-length parameter, λ, is global by its nature and interacts with all other DOFs. Hence, the global Hessian matrix is sparse. Some interactions lead to linear equations, so they do not affect the Hessian. This is the case for x, y, and λ interactions in Equation (4.5).

We now describe the structure of the finite dimensional optimization problem using a small mesh with $n = 3$ elements and $N = 10$ points for obstacle constraints as shown in Figure 4.6(a). In FEM, each element provides a small Hessian, typically dense, that relates all the DOFs present in that element. We have eight v and θ DOFs on each element except for the singular corner elements that have six. Figure 4.6(b), shows the global connectivity structure of the problem after boundary conditions are imposed on boundary $v, \theta, x,$ and y DOFs. These are marked A in Figure 4.6(a). The three element matrices are added to their appropriate positions. The DOFs marked B (for θ') are constrained via equality constraints. The DOFs marked C (for v') are constrained via equality constraints if speed is non-zero. Otherwise, it is infinite and is taken care using singular elements. The x and y DOFs do not enter the expression of cost, hence all corresponding rows and columns are empty (zero). The Hessian of obstacles constraints does contain non-zero 2×2 blocks relating x and y of the same point.



(a) A finite element mesh with 3 elements along with $N = 10$ $\{x, y\}$ pairs for obstacle avoidance.



v_0' θ_0' v_1 v_1' θ_1 θ_1' v_2 v_2' θ_2 θ_2' v_3' θ_3' x_1 y_1 x_2 y_2 $\ldots$ x_8 y_8 λ

(b) Connectivity structure

Figure 4.6: Global connectivity structure of the finite dimensional optimization problem.
(a) Some boundary element DOFS and the first and last $\{x, y\}$ pairs, are set equal to the appropriate boundary conditions and removed from the list of unknowns (A). Some boundary element DOFS are related to boundary conditions by equality constraints (B). Some are either related to boundary conditions via equality constraints if speed is non-zero, or taken care of by singular elements(C). (b) All unknowns on a node interact with unknowns on only two neighboring nodes. Each $\{x, y\}$ pair interacts only with itself. All DOFS on a node interact with λ.

Chapter 5

Initial Guess for the Optimization Problem

In Chapter 3, we described a nonlinear constrained optimization problem to find an optimal trajectory that results in a small discomfort. Because of the nonlinearity and presence of both inequality and inequality constraints, it is crucial that a suitable initial guess of the trajectory be computed and provided to an optimization algorithm.

Many packages can generate their own "starting points", but a good initial guess that is within the feasible region can easily reduce the computational effort (measured by number of function and derivative evaluation steps) many times. Not only that, reliably solving a nonlinear constrained optimization problem without a good initial guess can be extremely difficult. Because of these reasons, we invest considerable mathematical and computational effort to generate a good initial guess of the trajectory.

5.1 Overview

As described in Chapter 3, a trajectory can be completely described by its length λ, orientation $\theta(u)$, and the speed $v(u)$ for $u \in [0, 1]$. Our optimization problem is to find the scalar λ and the two functions θ and v that minimize the discomfort. We compute the initial guess of trajectory by computing λ and θ first and then computing v by solving a separate optimization problem. We emphasize that the initial guess computation process must deal with arbitrary inputs and reliably compute the initial guesses.

Before we discuss the initial guess of θ, we must discuss a genuine non-uniqueness issue. It is obvious that there exist infinitely many paths for a given pair of initial and final orientations. There exist at least two different kinds of non-uniqueness. The first kind of non-uniqueness exists because multiple numerical values of an angle correspond to a single "physical" orientation. The second kind of non-uniqueness exists because even for the same numerical values of initial and final angles, one can end up in one of multiple local minima after optimization. We now discuss these in detail.

5.2 Multiplicity of Paths

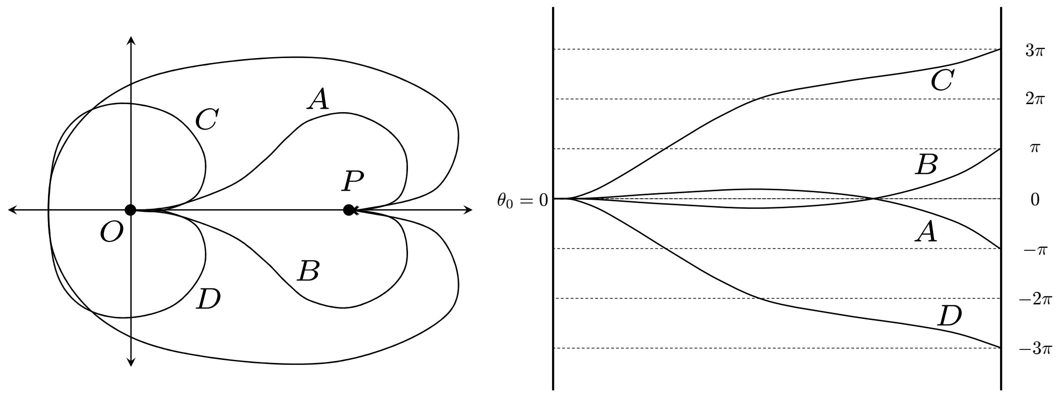
Since the trajectory orientation θ is an angle, a single θ value is completely equivalent in physical space to $\theta \pm 2n\pi \ \forall n \in \mathbb{N}$. However, consider a trajectory that starts with a given angle θ_0, and stops at orientation θ_τ (where τ denotes final time). Such a trajectory will be different than a trajectory that starts off with the same orientation but stops at $\theta_\tau \pm 2n\pi$. This is because θ is continuous and cannot jump to a different value in between. Of course, the boundary condition will still be satisfied. Thus, even though the original trajectory optimization problem is specified using a single stopping orientation, we must consider multiple stopping orientations, differing by 2π, when computing the initial guess as well as solving the original discomfort minimization problem. We have called this a "parity" problem. Note that the same logic of parity applies to the starting orientation, but what matters is the difference and we have chosen to vary only the ending orientation by choosing different values of n.

Figure 5.1 shows a few examples of this parity. It shows four paths corresponding to different n each sharing a common starting angle, but reach the destination at $\{-3\pi, -\pi, \pi, 3\pi\}$. Of course, we could create more paths by increasing the 2π difference but doing so makes the paths more convoluted and self-intersecting in general. This is because when n is too large in magnitude, $\theta(u)$ has to vary rapidly at least around some u values in $[0, 1]$ to satisfy the larger difference in boundary conditions. For optimization purposes, we assume that the starting and ending orientations are given between $[0, 2\pi)$ and we choose just three end-point orientations that give the least difference $|\theta_\tau - \theta_0|$.

Apart from the multiplicity of θ curves due to parity, the optimization problem discussed Section 5.3 to compute θ as well as the full discomfort minimization problem can lead to multiple solutions even when parity remains unchanged. This occurs due to the nonlinearity of constraints in Equation (3.30). Figure 5.2(a) and (b) shows two such paths A and B that start and end at the same numerical orientation but are qualitatively different. We do observe such multiple minima in practice. If we observe B more carefully, it is seen that it can be continuously deformed into $B*$ shown in Figure 5.2(c) and (d).

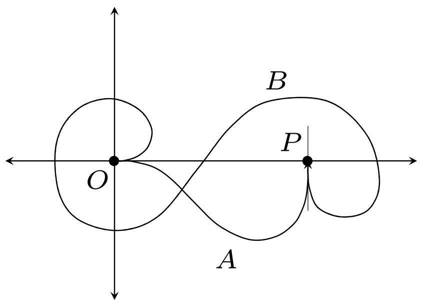
This figure clearly shows that the reason B is qualitatively different from A is because of two self-intersections − first in anti-clockwise direction and second in clockwise direction. Both "loops" cancel each others' changes in orientation. We suspect that this topological difference is the cause of multiple local minima.

Of course, this argument can be carried further and one can introduce an equal number of clockwise and anti-clockwise loops in arbitrary order and the final orientation will remain unchanged. Thus, we believe that there can be infinitely many local minima. Obviously, doing so would increase the discomfort in general and such a path will not be desirable. We try to avoid this problem by setting bounds on maximum and minimum θ when we compute the initial guess of θ. However, it is important to not ignore multiple minima

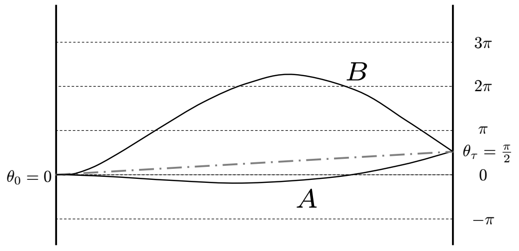


(a) Four paths (b) Four θ curves

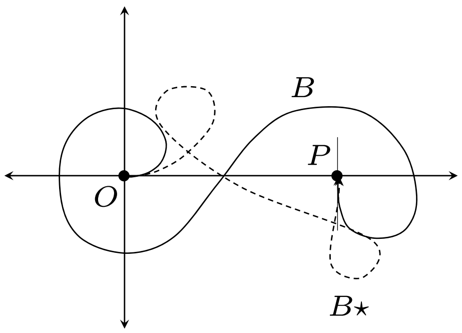
Figure 5.1: Four paths with different parity
The paths A, B, C, and D start from O and reach P at identical physical angles but, looked as θ curves, their ending angles differ by integer multiples of 2π.



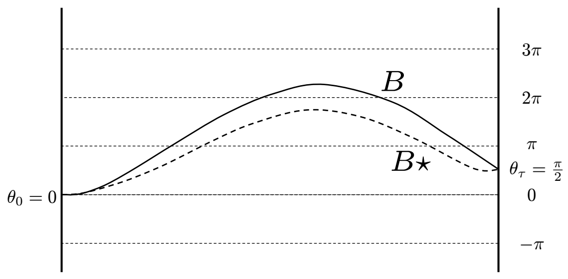
(a) Two paths



(b) Two θ curves



(c) Deformed path



(d) Deformed θ curve

Figure 5.2: (Two local minima for same boundary conditions
(a)(b) The paths A and B are different but both minimize discomfort compared to neighboring paths. (c)(d) B_* is obtained by a continuous deformation of B. The θ curves of B and $B*$ are similar. The corresponding paths show that $B*$ contains two self intersections and is topologically different from A that does not contain self intersections.

if they are found within these bounds. If obstacles are present so that A is infeasible, B might be chosen even though it is longer and has more turns.

Thus, because of these two kinds of multiplicities, we use more than one initial guess when minimizing the discomfort and choose the one that has the minimum discomfort and satisfies the constraints. We discuss the details in the following section.

5.3 Initial Guess of Path

We compute initial guesses for λ and $\theta(u)$ using two different methods. The first method computes a $\theta(u)$ such that the trajectory has a piecewise constant curvature. This is a computationally inexpensive method and does not satisfy many of the constraints exactly. The output of this method can be used to solve the full discomfort minimization problem.

The second method computes a $\theta(u)$ and λ by solving an auxiliary (but simpler) nonlinear constrained optimization problem. Of course, now we need an initial guess for this new optimization problem! The output of the "constant curvature" method mentioned above is used as the initial guess. Unlike the first method, the output of this second method leads to trajectories that have continuous and differentiable curvature and also satisfy boundary conditions and maximum curvature constraint exactly.

5.3.1 Piecewise Constant Curvature Path

In the full discomfort minimization problem, the orientation $\theta(u)$ has to satisfy the boundary conditions and Equation (3.30). In total, there are four constraints – two linear (those due to boundary conditions) and two non-linear (those of Equation (3.30)).

For computing initial guess of θ, we modify the inputs of the full optimization problem using a rotation such that initial and final position have the same y coordinate value. The initial and final orientations are also modified appropriately. Once we find an initial guess for the transformed input, it can be easily transformed back to the original configuration by the inverse rotation. This is done to allow efficient storage of precomputed θ guesses for various end-point conditions.

Thus, the inputs to the initial guess generation problem are the initial and final positions, x_0 and x_τ, and orientations, θ_0 and θ_τ, in the rotated frame. The output will be a path length λ and a function $\theta(u)$.

We begin by choosing the value of path length λ as $\max(R, 2 * \Delta L)$, where R is the minimum turning radius of the robot and $\Delta L = ||\mathbf{r}_\tau - \mathbf{r}_0||$. Using this maximum takes care of the case when initial and final positions are very close to each other. In such a case, the path length is decided by the minimum turning radius constraint.

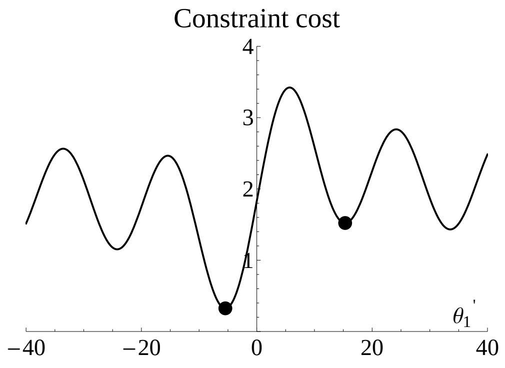
Ideally, an initial guess of $\theta(u)$ should obey the following constraints so that the constraints of the full optimization problem are satisfied:

$$\begin{aligned} \theta(0) &= \theta_0, \\ \theta(1) &= \theta_\tau, \end{aligned} \tag{5.1}$$

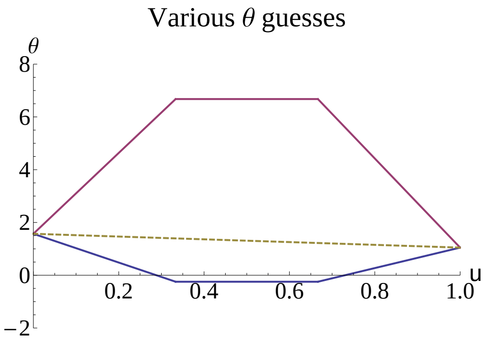
and transformed Equation (3.30)

$$\begin{aligned} \lambda \int_0^1 \cos\theta \, du &= x_\tau - x_0, \\ \lambda \int_0^1 \sin\theta \, du &= 0, \end{aligned} \tag{5.2}$$

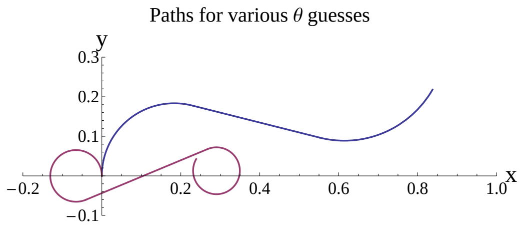
Consider a piecewise linear function that looks like the solid curves in Figure 5.3(b).



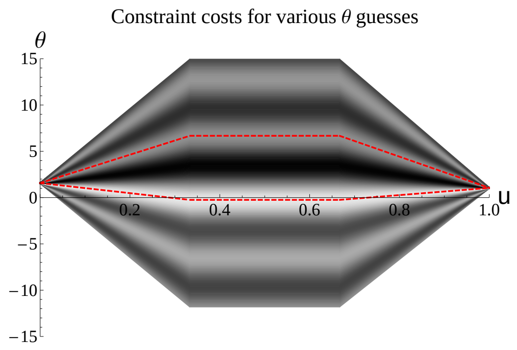
(a) Cost from Equation (5.4)



(b) Two constant curvature optimizers



(c) Paths corresponding to (b)



(d) Multiple local minima

Figure 5.3: Piecewise constant curvature initial guesses
(a) Multiple local minima in graph of J_{cc} of Equation (5.4). Two minima closest to the maxima are highlighted. (b) Piecewise constant curvature paths (not dashed) corresponding to highlighted minima in (a). (c) Paths corresponding to the θ curves in (b). Both start at $\frac{\pi}{2}$ and end at $\frac{\pi}{3}$. (d) Lighter shade represents minima and darker shade represents maxima. The two optimizing paths of (b) are shown.

Essentially, each such curve is defined on $[0, 1]$, is continuous, and is made of three line segments in $[0, \frac{1}{3}]$, $[\frac{1}{3}, \frac{2}{3}]$, and $[\frac{2}{3}, 1]$. The middle line segment has zero derivative. The values at 0 and 1 are known and the only variable is the function value on the middle segment. Equivalently, one can use the slope of first line segment as the variable. Let this slope be denoted by θ_1'. Then, we define $\theta(u)$ as

$$\theta(u) = \begin{cases} \theta_0 + \theta_1' u & \text{if} \quad 0 \leq u < \frac{1}{3}; \\ \theta_0 + \frac{1}{3}\theta_1' & \text{if} \quad \frac{1}{3} \leq u < \frac{2}{3}; \\ \theta_0 + \frac{1}{3}\theta_1' - 3(\theta_0 - \theta_1 + \frac{1}{3}\theta_1')(u - \frac{2}{3}) & \text{if} \quad \frac{2}{3} \leq u \leq 1. \end{cases} \tag{5.3}$$

If we use such a curve for $\theta(u)$, it will result in a circular arc, a tangent line segment, and another circular arc tangential to the middle segment, in that order. This, in turn, implies that the resulting path will have a piecewise constant curvature.

To determine $\theta(u)$, we need to to determine the value of the unknown slope θ_1'. Since only one value cannot satisfy two constraints of Equation (5.2), we minimize

$$J_{cc}(\theta_1') = \left(\int_0^1 \cos\theta \, du - 1 \right)^2 + \left(\int_0^1 \sin\theta \, du \right)^2 \tag{5.4}$$

to find θ_1'. Figure 5.3(a) shows the plot of J_{cc} as a function of θ_1'. Depending on the boundary conditions, the shape of J_{cc} changes but qualitatively it has the behavior as shown $-$ oscillatory with a maximum not too far from zero. We find this maximum using a table lookup and the neighboring two minima to compute the initial guess. Figure 5.3(c) shows two paths using this method where $\theta_0 = \frac{\pi}{2}$ and $\theta_\tau = \frac{\pi}{3}$. The path length is 1. As seen, the curve end-point is not too far from $x-$axis, and the curve satisfies the boundary condition on θ. Figure 5.3(d) shows the cost J_{cc} for various constant curvature paths. The lighter shade corresponds to the minima.

5.3.2 Optimization Approach for Initial Guess of Path

In this second method to compute the initial guess of the path, we minimize

$$J(\theta, \lambda) = \lambda + w \int_0^1 \theta''^2 du \tag{5.5}$$

where $w := \max(\Delta L, R)$, and θ must satisfy the boundary conditions, the two equality constraints of Equation (3.30), and the curvature constraint

$$|\theta'(u)| \leq \lambda \kappa_{\max} \; \forall u \in [0, 1].$$

We do not impose the obstacle related constraints in this problem. This problem is related to the concept of "Minimum Variation Curves" [66] which have been proposed for curve shape design. We add the curve length λ so that in the presence of multiplicities, discussed in Section 5.2 earlier, the curves with smaller lengths are preferred. This optimization problem is discretized using C^1 finite elements as described in Chapter 4, and the initial guess is the piecewise constant curvature function from Section 5.3.1. Paths computed using this approach are shown in the next chapter in Section 6.2.

5.4 Initial Guess of Speed

Computing the initial guess of v is relatively simpler. We solve a convex quadratic optimization problem with linear inequality box constraints to compute the initial guess. Because of convexity of the functional and the convex shape of the feasible region, this problem has a unique solution and an initial guess is not necessary to solve it. Any good quality optimization package can find the solution without an initial guess. Of course, because of the simplicity of box constraints, we can and do provide a feasible initial guess for this auxiliary problem.

First consider the case when both end-points have non-zero speed. We minimize

$$J(v) = \int_0^1 v''^2 du \tag{5.6}$$

subject to boundary constraints $v(0) = v_0 > 0$, $v(1) = v_1 > 0$, $v'(0) = \frac{a_0\lambda}{v_0}$, $v'(1) = \frac{a_1\lambda}{v_1}$ and inequality constraints $v_{\min}(u) \leq v(u) \leq v_{\max}(u)$ and $A_{\min}(u) \leq v'(u) \leq A_{\max}(u)$. The expressions for $v'(0)$ and $v'(1)$ come from the relation in Equation (3.21). The length λ is computed when the initial guess for θ is computed. Here we choose $v_{\min}(u) = \min(v_0, v_1)/2$ and $v_{\max}(u)$ is a constant that comes from the hardware limits. The function $A_{\min}(u)$ is chosen to be the constant $10a_{\min}\lambda/\min(v_0, v_1)$ where $a_{\min}$ is the minimum allowed physical acceleration. $A_{\max}(u)$ is chosen similarly using $a_{\max}$.

This optimization problem is discretized using C^1 finite elements as described in Chapter 4 and leads to a convex programming problem that is easily solved. Of course, this method does not take care of cases in which one or both points have zero speed boundary conditions.

If both end-points have zero speeds, the function

$$v(u) = v_{\max} \left(4u(1-u)\right)^{2/3} \tag{5.7}$$

satisfies the boundary conditions and singularities and has a maximum value of $v_{\max}$. This case doesn not require any optimization.

If only one of the end-points has a zero speed boundary condition, we split the initial guess for v into a sum of two functions. The first one takes care of the singularity and the second takes care of the non-zero speed boundary condition on the other end-point. We now maintain only the $v_{\max}$ constraint because $v'(u)$ is unbounded and $v_{\min} = 0$ naturally. If the right end-point has zero speed, we choose

$$v(u) = v_{\text{singular}}(u) + v_{\text{non-singular}}(u)$$

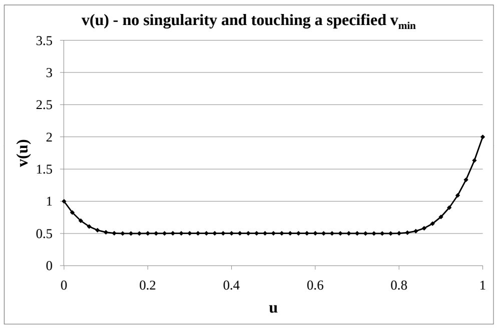
where

$$v_{\text{singular}}(u) = \frac{16}{9} 2^{1/3} v_{\max} u^2 (1-u)^{2/3}.$$

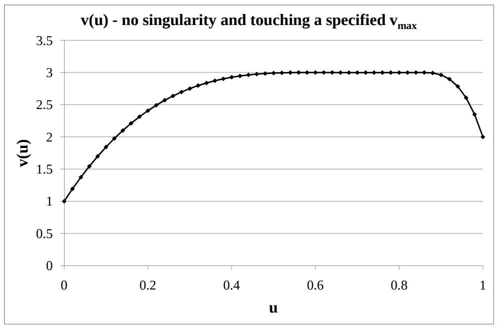
This function has the correct singularity behavior and its maximum value is $v_{\max}/2$. The non-singular part is computed via optimization so that the sum is always less than $v_{\max}$. For the other case, when left end-point has zero speed, the singular part (using symmetry) is

$$\frac{16}{9} 2^{1/3} v_{\max} (1-u)^2 u^{2/3}.$$

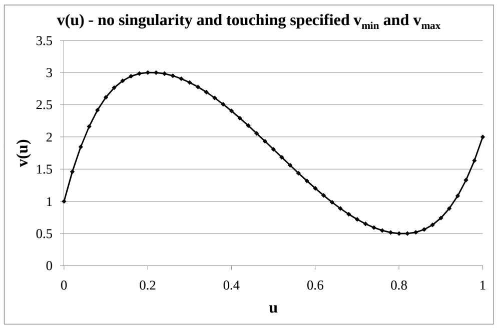
Figure 5.4 shows these different cases. All the imposed bounds are maintained and the initial guesses of v are smooth curves for all kinds of boundary conditions. For non-zero boundary speed, the values are 1 on the starting point and 2 on the ending point. Maximum speed is 3. Where imposed, $A_{\min} = -50$ and $A_{\max} = 50$.



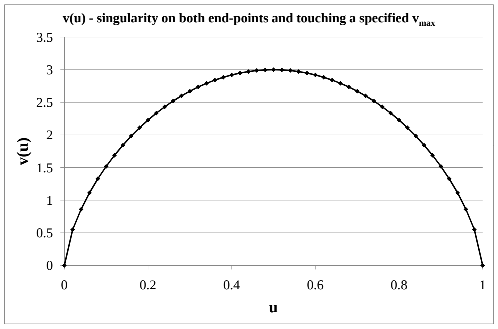
(a) $v_{\min} = 0.5$ is active



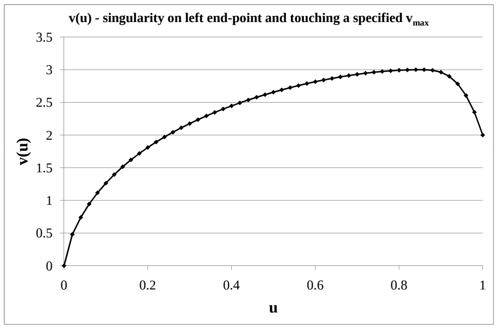
(b) $v_{\max} = 3$ is active



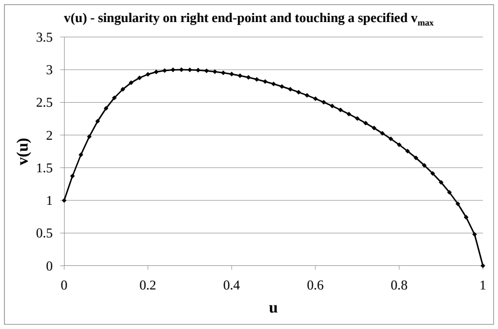
(c) Both $v_{\min}$ and $v_{\max}$ active



(d) Both ends singular and $v(u) = v_{\max} \left(4u(1-u)\right)^{2/3}$



(e) $v_{\max} = 3$ is active, singular start



(f) $v_{\max} = 3$ is active, singular end

Figure 5.4: Initial guesses for speed

Chapter 6

Evaluation and Results

The motion planning framework described in earlier chapters is expected to reliably plan trajectories for different types of boundary conditions. These trajectories should satisfy dynamic constraints and the geometric paths should not intersect obstacles. Further, it should be possible to reliably compute trajectories between a given pair of boundary conditions for a range of weights, w_T and w_N, so that users can customize the motion by changing these weights.

The trajectories are computed by solving a constrained optimization problem that minimizes the cost functional

$$J = \tau \ + \ f_T \widehat{w}_T \int_0^\tau (\dddot{\mathbf{r}} \cdot \mathbf{T})^2 \, dt \ + \ f_N \widehat{w}_N \int_0^\tau (\dddot{\mathbf{r}} \cdot \mathbf{N})^2 \, dt \tag{6.1}$$

subject to dynamic and obstacle-avoidance constraints. Here $\widehat{w}_T$ and $\widehat{w}_N$ are the dimensional weights that are automatically computed from the length and velocity scales of the task as described in Section 3.5. The positive dimensionless factors f_T and f_N can be varied by a user for balancing jerk discomfort and travel time.

This problem is discretized into a finite dimensional problem using the Finite Element Method (FEM) and an optimization package (Ipopt) is used to solve it numerically. In the following discussion, we refer to this optimization problem as the "discomfort minimization problem".

We begin by describing the input to the discomfort minimization problem and how it is determined. Some quantities in the input such as dynamic bounds are fixed, while others such as boundary conditions and obstacle locations and shapes are problem dependent. We describe how parameters such as the number of elements in the finite element discretization are determined. We also provide some implementation details.

Next, we present illustrative examples showing the various steps of the solution method, and demonstrate some of the strengths of our method such as the ability to plan trajectories for a wide variety of boundary conditions and obstacle shapes.

We then analyze how varying the weight factors f_T and f_N affect the solution trajectory. Our objective is to find qualitative relationships between these weight factors and each of the terms in the discomfort measure (total travel time, integral of squared tangential jerk, and integral of squared normal jerk). These relationships should provide guidelines for user customization.

Next, to evaluate the reliability of our method, we construct a large data set of problems with different geometry and boundary conditions and find the success rate. We also analyze the run-time and number of iterations to compute the initial guess as well as the solution to the discomfort minimization problem.

6.1 Experimental setting

The input to the discomfort minimization problem described in Section 3.11 consists of:

1. Number of elements, n, for finite element discretization. We choose $n = 32$ based on a numerical experiment based on convergence to the "exact" solution of the infinite dimensional optimization problem as the maximum finite element size is reduced (see below, Section 6.3).

2. Number of intervals per element M, to compute the $\{x, y\}$ pairs for imposing obstacle constraints (see Section 4.3.1). We choose $M = 20$ when obstacles are present, otherwise the choice is irrelevant.

3. Values of bounds on curvature, speed, angular speed, tangential acceleration, and normal acceleration (See Section 3.11). Curvature bounds should be determined from the robot's geometry. While we assume a point robot and do not consider robot shape for obstacle-avoidance, we do include curvature constraints based on the dimensions of a typical wheelchair. All other bounds should be chosen for comfort. In the absence of relevant comfort studies for assistive robots, we choose linear and angular speed based on our expectation of typical values of these quantities for an assistive robot. We choose values of acceleration bounds based on studies of comfort in ground vehicles (see Section 3.3). All these values are shown in Table 6.1.

Quantity	**Lower Bound**	**Upper Bound**
Curvature (1/m)	-1.8	1.8
Speed (m/s)	0.0	3.0
Angular speed (rad/s)	-1.57	1.57
Tangential acceleration (m/s^2)	-1.0	1.0
Normal acceleration (m/s^2)	-1.0	1.0

Table 6.1: Lower and upper bounds on curvature, speeds, and accelerations used in experiments.
Curvature bounds are based on a minimum turning radius of 0.55 m.

4. Non-dimensional multiplying factors for weights, $f_T > 0$ and $f_N > 0$. Both these values are set to 1 unless mentioned otherwise.

5. Representation of obstacles as star-shaped domains with piecewise C^2 boundary (see Section 3.10.3). In our experiments, we use circular, elliptical, and star-shaped polygonal obstacles. See Figures 6.6 and 6.7.

6. Boundary conditions on position, orientation, curvature, speed and tangential acceleration (see Section 3.9). These are problem specific and we describe these for each of the experiments.

We have implemented our code in C++. We use Ipopt, a robust large-scale nonlinear constrained optimization library [98] written in C++ to solve the optimization problem. We explicitly compute gradient and Hessian for the optimization problem in our code instead of letting Ipopt compute these using finite-differences. This leads to greater robustness and faster convergence. We set the Ipopt parameter for relative tolerance as 10^{-8} and set the maximum number of iterations to 500.

After optimization, the outputs are the nodal values of v, v', θ, and θ', and the curve length λ (see Section 4.2). The functions $v(u)$ and $\theta(u)$, $u \in [0,1]$ are known using Equation 4.3. We use Equation 3.13 to construct a table of u values for $u \in [0,1]$ and the corresponding t values for $t \in [0,\tau]$. The value of any of the quantities of interest (orientation, speed, etc.) at any time $t \in [0,\tau]$ is computed using this table by linear interpolation.

6.2 Illustrative examples

We begin by presenting an example that illustrates the optimization process. In Figure 6.1, the initial position is $\{0,0\}$ and final position is $\{-1,-4\}$. The initial and final orientations are both zero. The speed and tangential acceleration at both ends are also zero.

First, an initial guess of path ($\theta(u), u \in [0,1]$ and λ) is computed. Using this value of λ, and initial guess of speed ($v(u), u \in [0,1]$) is computed. To compute initial guess of path, we choose three $\{\theta_0, \theta_\tau\}$ pairs: $\{0,0\}$, $\{0,2\pi\}$, and $\{0,-2\pi\}$. For the $\{0,0\}$ pair, we compute two piecewise constant curvature paths (Section 5.3.1) by choosing two minima of J_{cc} (Equation 5.4) as shown in Figure 5.3. For the $\{0,-2\pi\}$ and $\{0,2\pi\}$ pairs, we compute one piecewise constant curvature path each. This results in *four* piecewise constant curvature paths. These paths serve as initial guesses for the optimization problem of Section 5.3.2 which computes four initial guesses of path.

The four paths computed above serve as initial guesses for the discomfort minimization problem. Figure 6.1 shows four initial guesses of path. The

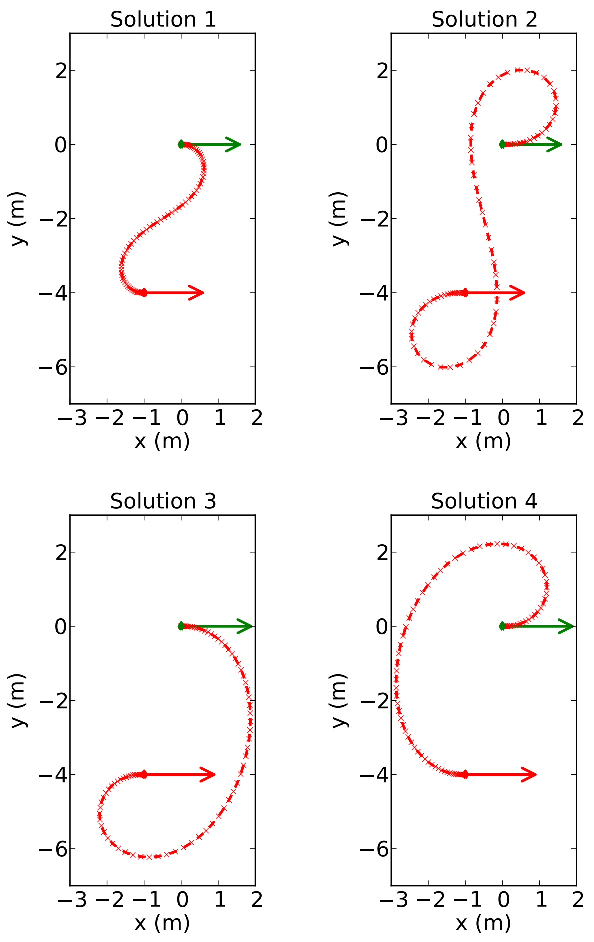


Figure 6.1: Four initial guess of path.

Problem input is as follows: initial position = $\{0, 0\}$, orientation = 0, speed = 0, tangential acceleration = 0; final position = $\{-1, -4\}$, orientation = 0, speed = 0, tangential acceleration = 0. The four initial guesses of path are computed using the method described in Sec 5.3.2 so that final orientation in (a),(b),(c) and (d) is 0, 0, -2π and 2π respectively. Initial position is shown by a green marker and initial orientation is indicated by the direction of the green arrow. Final position and orientation are similarly indicated in red. While the path is parameterized by u, for ease of visualization, we show markers at equal intervals of time. Thus distance between markers is inversely proportional to speed.

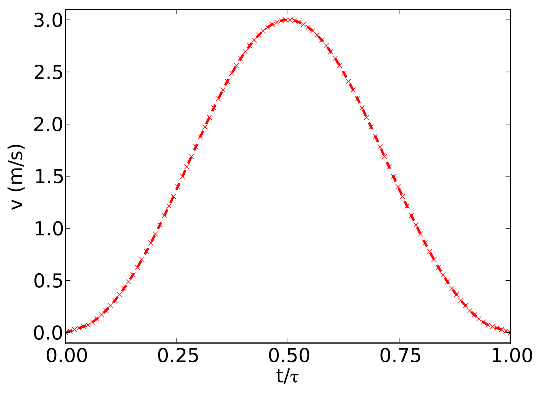


Figure 6.2: Initial guess of speed for problem of Figure 6.1.
In this case, because of zero speed boundary condition on both ends, the same initial guess of speed is produced for each path guess. When speed is non-zero on one or both ends, four distinct guesses of speed may be produced.

first two solutions have $\theta_\tau = 0$, the third solution has $\theta_\tau = -2\pi$, and the fourth solution has $\theta_\tau = 2\pi$.

An initial guess of speed, $v(u)$, is computed as described in Section 5.4. In this example (Figure 5.4, speed at both ends is zero and hence $v(u)$ is computed using Equation 5.7. Thus we get the same function $v(u)$ for all guesses of path. If speed is non-zero at either end, then we solve an optimization problem to compute $v(u)$. In this case, the curve length λ, from each of the four path guesses is input to the optimization problem, and we may get different guesses of speed corresponding to each path guess.

The discomfort minimization problem is solved for each of these four initial guesses. The four solution paths that minimize discomfort are shown in Figure 6.3.The travel time and costs for the four solution paths are shown

Solution Number	**Travel time (s)**	**Total cost (s)**
1	6.3	6.5
2	10.0	11.0
3	7.9	8.0
4	7.9	8.0

Table 6.2: Travel time and total cost for problem of Figure 6.1.

in Table 6.2. The path corresponding to Solution 1 has the minimum cost, and is thus in agreement with our intuitive notion of the best path amongst these four. Notice the circular arcs at the start and end of the path of Solution 2. These arcs have a constant radius equal to the minimum turning radius of the robot because of curvature constraints. If curvature constraints are not imposed, these arcs have a smaller radius and the path has a smaller length. Note that it is not always true that all four solutions are distinct since two or more problems starting from different initial guesses may converge to the same solution.

The solution speeds are shown in Figure 6.4. The final speeds in Solution 1 and Solution 2 are symmetric about $t = \frac{\tau}{2}$ because of the inherent "symmetry" due to zero orientation, speed, and acceleration at both ends. The final speeds in Solution 3 and Solution 4 are mirror images of each other about $t = \frac{\tau}{2}$ because the final orientations in these two are -2π and 2π respectively. The figures also show that the initial guesses of the paths and speeds are quite good, which is important for nonlinear optimization.

In Figure 6.5, we introduce five elliptical obstacles for the same bound-

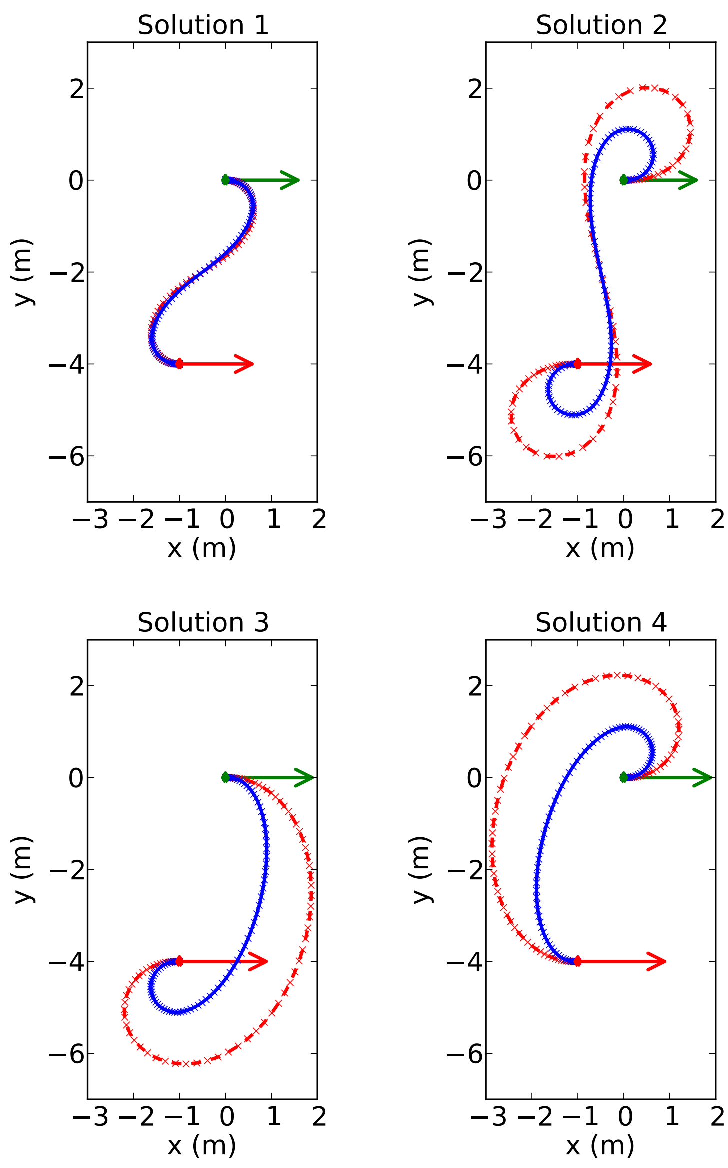


Figure 6.3: Solution paths of the problem of Figure 6.1.
Final (optimal) path for each solution is shown as solid blue curve. Initial guess is shown as dashed red curve. The number of DOFS for the discomfort minimization problem were 1403 and number of constraints were 3232. The total cost and travel time for the four solutions are shown in Table 6.2.

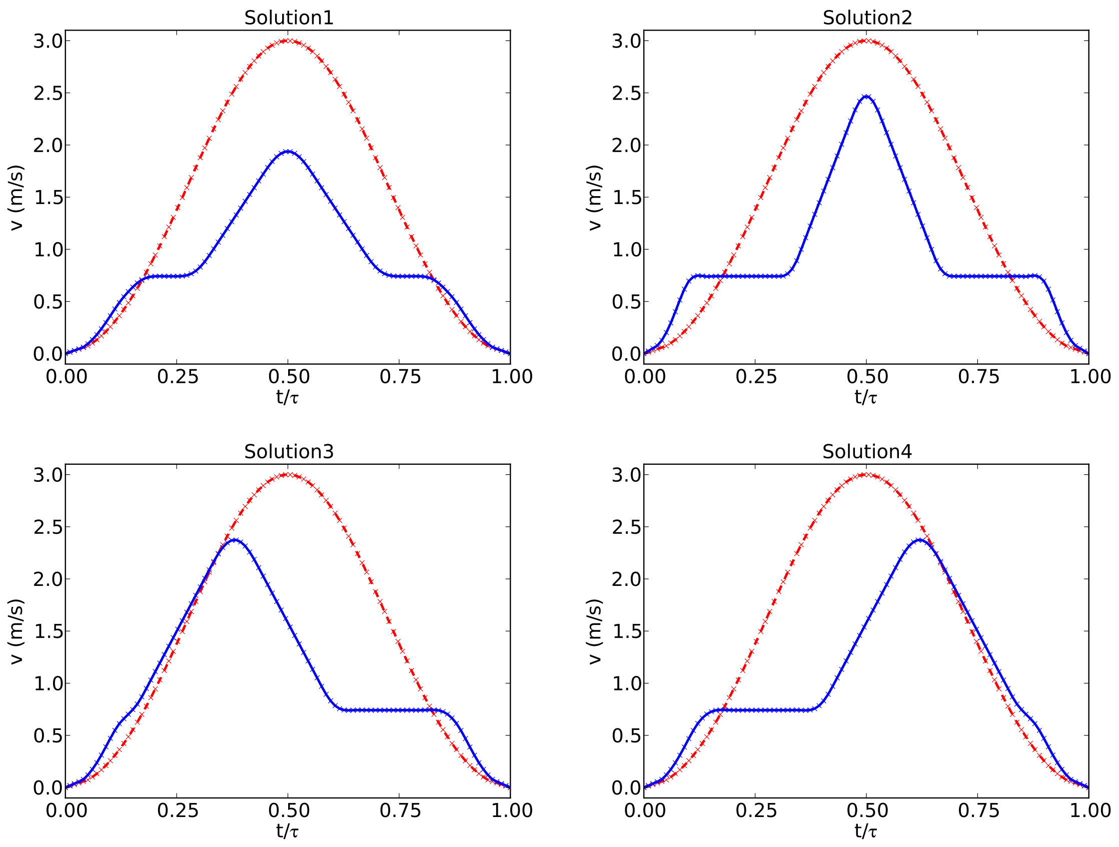


Figure 6.4: Solution speeds of the problem of Figure 6.1.
Final (optimal) speed for each solution is shown as solid blue curve. Initial guess is shown as dashed red curve.

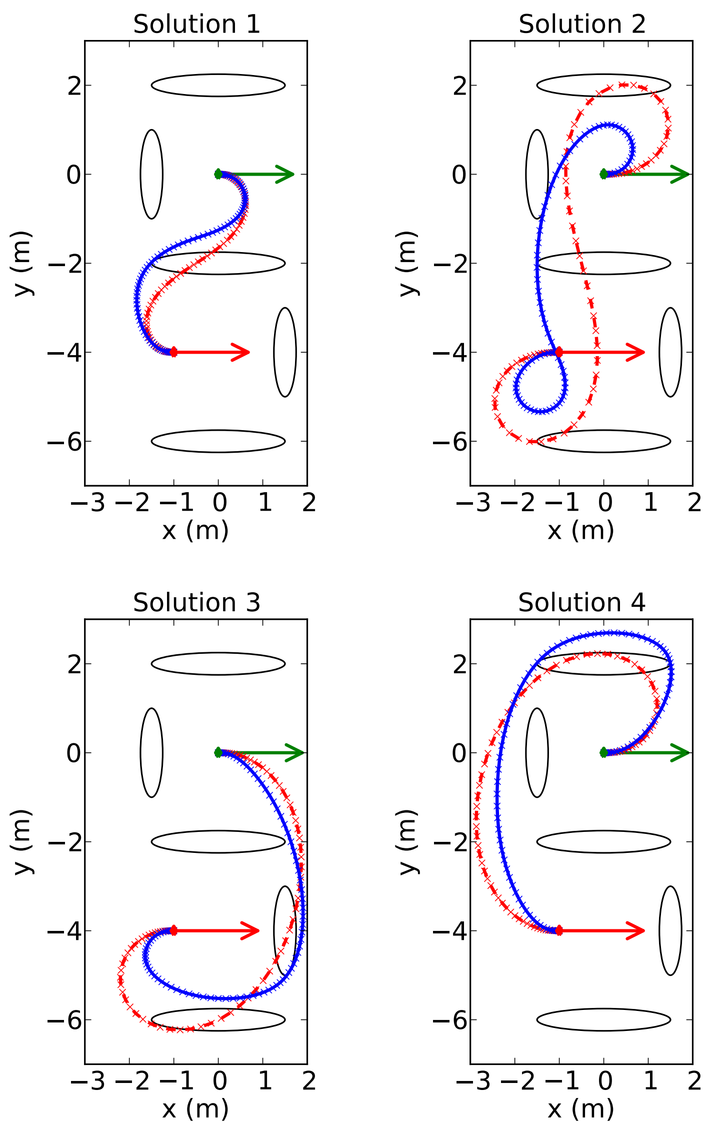


Figure 6.5: Solution paths to a problem with five elliptical obstacles. The boundary conditions of this problem are identical to the problem of Figure 6.1. Four distinct solution paths in the neighborhood of the four initial guesses are found. This problem had 3195 constraints for obstacle-avoidance in addition to the constraints in Figure 6.1. The total cost and travel time for the four solutions are shown in Table 6.3.

Solution Number	**Travel time (s)**	**Total cost (s)**
1	7.0	7.1
2	11.7	11.9
3	9.1	9.4
4	10.3	10.4

Table 6.3: Travel time and total cost for problem of Figure 6.5.

ary conditions. All four initial guesses of path and solution paths are shown. The initial guesses of path and speed do not consider obstacles and hence are identical to those in Figures 6.3 and 6.4 respectively. Four distinct solution paths are found. The travel time and total cost for all four solutions is shown in Table 6.3, and is greater than for the problem of Figure 6.3 (see Table 6.2). The minimum cost path is that of Solution 1 which again agrees with our intuition. Notice how the path of Solution 3 passes above the lowermost elliptical obstacle, while the path of Solution 4 passes below the uppermost elliptical obstacle. Our experience with this and other examples shows that that once the optimization algorithm takes a step that brings an iterate to one side of the obstacle, further iterations keep it on the same side. We believe that this is because paths passing an obstacle on different "sides" belong to disjoint feasible regions. Since two such paths cannot be transformed to each other via a continuous deformation of the path, the two paths are in disjoint feasible regions. The iterates in the optimization process also cannot jump from one feasible region to a different feasible region in general.

Figure 6.6 show an example where the initial and final speeds are both non-zero. This scenario exemplifies one of the common navigation tasks for an assistive robot – that of navigating in a corridor or sidewalk. We show only one solution out of four in this case. Figure 6.6(a) has two rectangular obstacles, signifying a wall. In the sequence Figure 6.6(b)–(f), one obstacle is added at a time, and each time a path is found that avoids all the obstacles.

Figure 6.7 shows an example when the initial speed is non-zero and the initial acceleration is positive. There are four rectangular and two star-shaped obstacles. This is a particularly difficult case because it involves a non-zero speed and high acceleration(0.5 m/s^2, half the maximum allowable acceleration) at the beginning and a narrow passage between obstacles. In this case, only one of the four initial guesses resulted in a solution. Notice the loop in the path near the start. This is because the initial speed and acceleration are non-zero, and hence a sharp 90 degree left turn is not possible without violating dynamic bounds. If dynamic bounds are removed, another path, without a loop, starting from another initial guess is also found as a solution. This path does not have a loop. Also notice how the path just touches the vertices of obstacles so that its length is as small as is consistent with comfort.

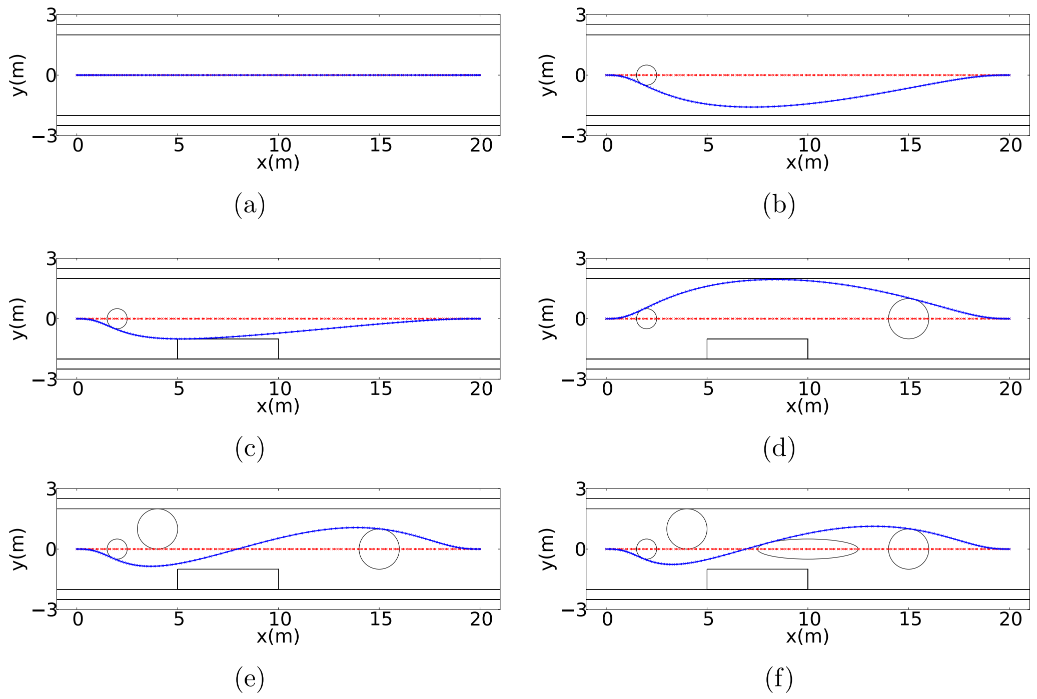


Figure 6.6: Obstacle avoidance in a corridor-like setting with non-zero speed at both ends.
Problem input is as follows: initial position = $\{0, 0\}$, orientation = 0, speed = 1, tangential acceleration = 0; final position = $\{20, 0\}$, orientation = 0, speed = 1, tangential acceleration = 0. One of the four solution paths is shown. Initial guess is shown as dashed red curve while solution is shown as solid blue curve. (a) Only two rectangular obstacles, comprising the corridor walls are present. The solution path is a straight line. (b) Addition of a circular obstacle results in a path that passes below the obstacle. Another solution path that passes above the obstacle and is symmetric to this path about the centerline would also be a solution with same cost. (c),(d),(e),(f) One more obstacle is added and the same problem is solved starting from the same initial guess as in (a). All quantities have appropriate units in terms of meters and seconds.

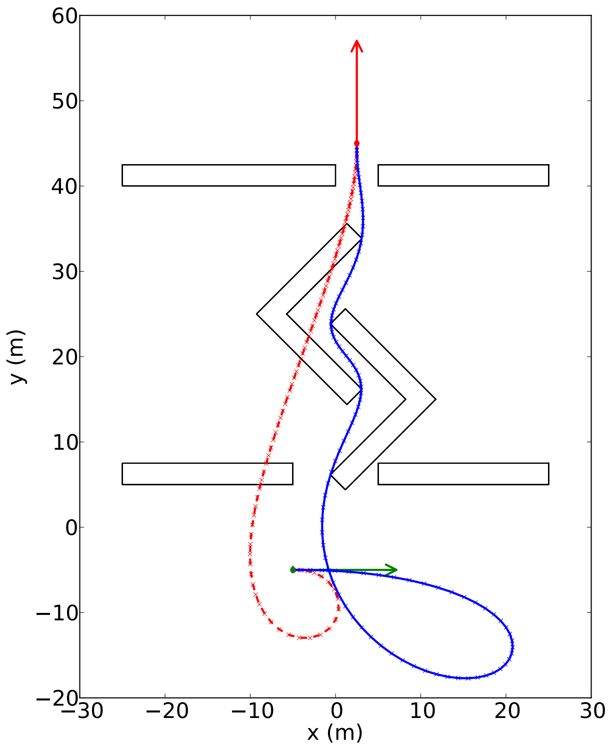


Figure 6.7: Illustrative example showing passage through narrow space between star-shaped obstacles with non-zero speed at both ends and high positive acceleration at start.
Problem input is as follows: initial position $= \{-5, -5\}$, orientation $= 0$, speed $= 1$, tangential acceleration $= 0.5$; final position $= \{2.5, 45\}$, orientation $= \pi/2$, speed $= 1$, tangential acceleration $= 0$. Four rectangular and two star-shaped obstacles are present. The loop at the beginning of the path is because the initial acceleration is high and hence it is not possible to make a sharp turn without violating dynamic constraints. All quantities have appropriate units in terms of meters and seconds.

6.3 Convergence on Decreasing Mesh Size

To analyze the effect of mesh size on convergence, we construct two examples of straight line motion, each with start and end positions as $\{0, 0\}$ and $\{10, 0\}$, and start and end orientations as 0. In the first example, speed at both ends is 0. In the second, speed at both ends is 1. Tangential acceleration at both ends is 0. We vary the number of elements from 2 to 128 in multiples of 2 and each time solve the discomfort minimization problem for one initial guess. As the number of elements increases, the optimum cost found by the optimization process decreases. This is natural because increasing the mesh size means we're minimizing a function in a superset of degrees of freedom. As the number of elements increases, the relative change in minimum cost decreases. We compare all costs with the cost corresponding to 128 elements. We see that the 32 elements give a cost that within 0.01% of cost for 128 elements (when $v > 0$ on end-points). The curve for $v = 0$ shows a lower convergence rate and we believe that the reason behind this is using standard Gauss-Legendre quadrature for the singular elements. A more precise procedure would use specially designed quadrature scheme keeping in mind the form of singularity at end-points. We have kept this as part of future work.

6.4 Effect of Weights on Discomfort

In this section we analyze how the two dimensionless factors f_T and f_N affect the individual terms comprising the cost functional (travel time, integral of squared tangential jerk, and integral of squared normal jerk) as shown in

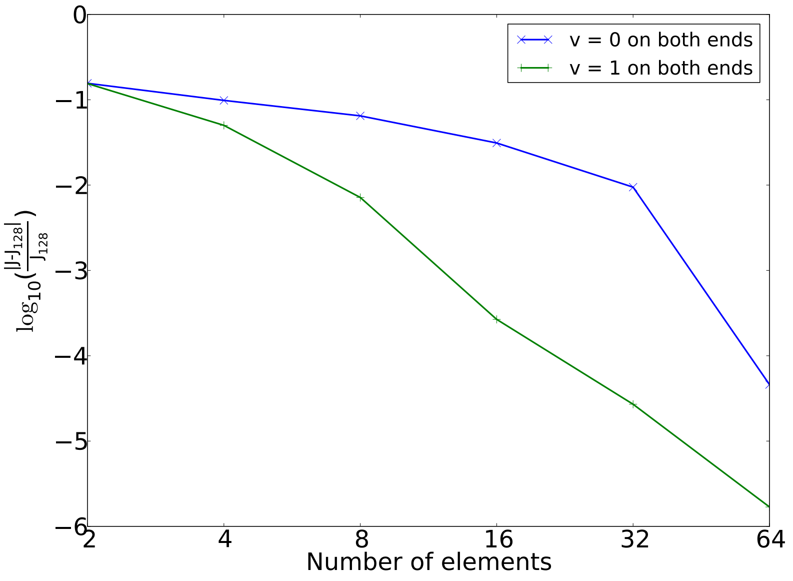


Figure 6.8: Convergence on decreasing mesh size. $\log_{10}(\frac{|J-J_{128}|}{J_{128}})$ on $y-$axis against number of elements on $\log_2$ scale on $x-$axis. The curve on top is for zero speed boundary conditions on both ends and the curve on bottom is for unit speed boundary conditions on both ends.

Equation (6.1). This analysis provides us with guidelines for choosing the values of weights for customization by human users. Henceforth, for conciseness, we will refer to the three terms – travel time, integral of squared tangential jerk, and integral of squared normal jerk as τ, J_T and J_N respectively. Thus, the cost functional of Equation (6.1) is

$$J = J_\tau + f_T J_T + f_N J_N$$

For this experiment, we construct a problem with identical boundary conditions as that of the example in Figure 6.1. In order to delineate the effect of weights, we remove all constraints and solve the unconstrained problem for a range of factors f_T and f_N for each of the four initial guesses. f_T is varied from 2^{-13} to 2^{13} in a geometric sequence, each subsequent value being obtained by multiplying the current value by 10. For each value of f_T, f_N is varied from 2^{-13} to 2^{13} in a similar manner. Thus each weight roughly ranges between 0.0001 and 10000. This results in $4 \times 27 \times 27 = 2916$ problems out of which 97% were successfully solved. We show plots corresponding to only one of these four solutions. Plots for the remaining solutions are similar, although the number of problems that converge is different for each initial guess.

Figures 6.9, 6.10, and 6.11 show τ, J_T and J_N respectively. In each figure, part (a) shows log of the respective quantity as a function of f_T and f_N on a log-log-log scale. Part (b) is a top view of the surface plot above. Part (c) shows slices of this surface plot at $f_N = 1$ and $f_T = 1$ respectively.

The "holes" in the surface plots correspond to the problems that did not converge to a solution. In general, the surfaces are rougher and there are more failures when f_N is much larger than f_T. This indicates that the problem becomes less "stable" as the weight factors are too imbalanced. (In reality there are more holes in the surfaces than there are non-convergent problems. This is an unfortunate artifact of the plotting software that we use. In the surface plot, a vertex corresponds to a problem rather than a cell. Thus, one non-convergent problem causes all the cells that share that vertex to be removed. The actual non-convergent problems correspond to the empty cells of Figure 6.12).

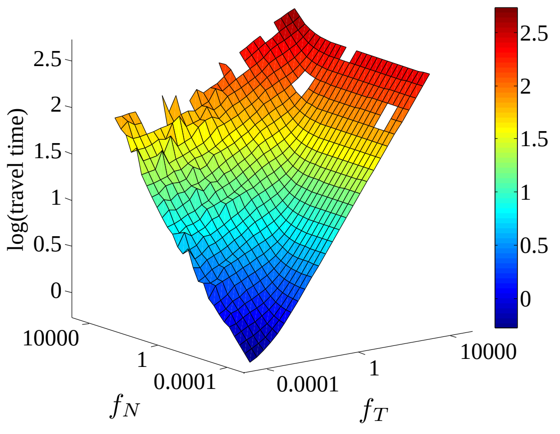
In this experiment, the ratio of tangential jerk weight to normal jerk weight has been varied by nearly 8 orders of magnitude and we get solutions in almost all cases.

From Figure 6.9, we see that the travel time increases with increase in weights. This is expected since large weights mean that the contribution of travel time to total discomfort is relatively low compared to the contribution of the terms due to jerk. We also see that τ monotonically increases with f_T. For low values of f_N, τ does not change appreciably with f_N. As the value of f_N increases beyond a threshold, τ monotonically increases with f_N. The rate of increase of τ with respect to f_T is higher than it is with respect to f_N.

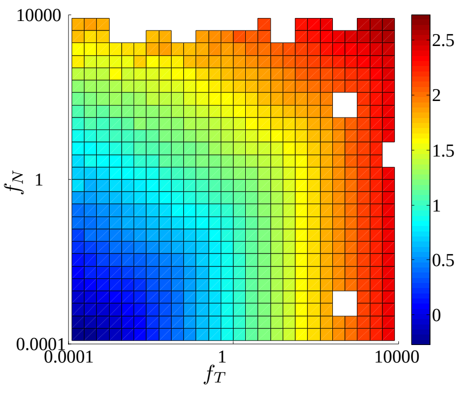
From Figure 6.10, we observe that $\log J_T$ decreases linearly with $\log f_T$ while it is almost constant with respect to $\log f_N$. Thus, the integral of squared tangential jerk, J_T, is related to f_T by a power law.

From Figure 6.11 we see that for low values of f_T, J_N does not change appreciably with f_T. As the value of f_T increases beyond a threshold, J_N monotonically decreases with f_T. A similar behavior is observed with respect to f_N although the threshold value appears lower than that for f_T. Once the values exceed the threshold, the rate of change of J_N with respect to both f_N and f_T is almost the same.

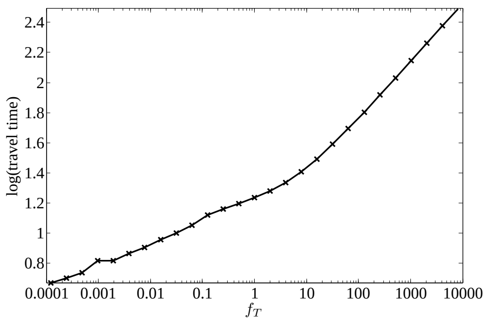
Thus, we see that the integral of squared tangential jerk, J_T is a function of f_T alone, and travel time changes more rapidly by changing f_T compared to f_N. Integral of squared normal jerk, J_N is a function of both f_T and f_N.



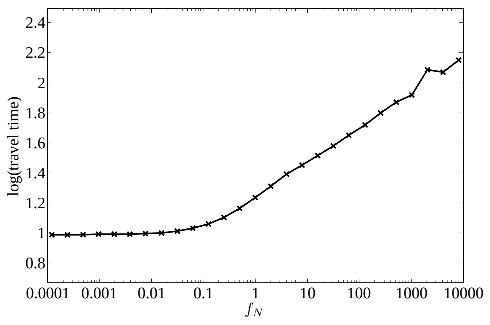
(a)



(b)

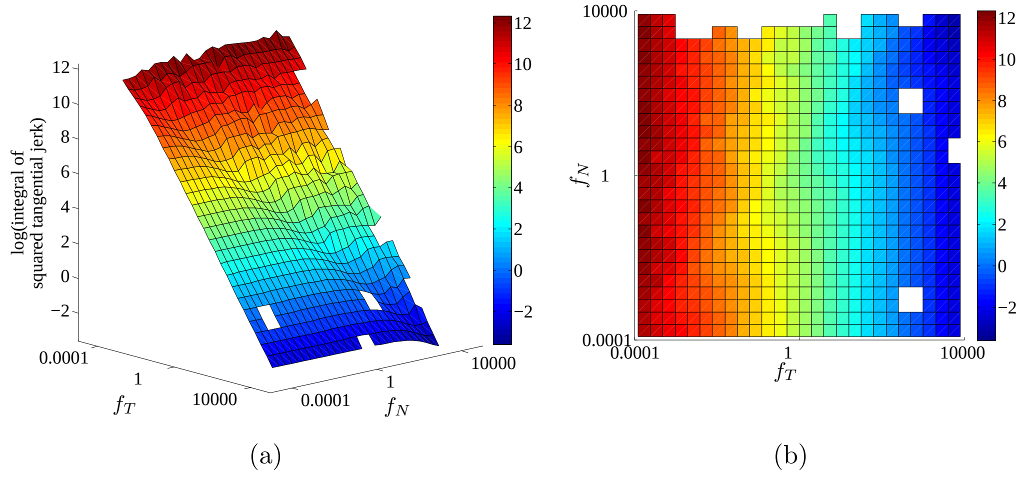


(c)



(d)

Figure 6.9: Effect of weights on travel time.
(a) Surface plot of $\log \tau$ as a function of f_T and f_N on a log-log scale. (b) Top view of the surface plot. (c) Slice of the surface plot at $f_N = 1$. (d) Slice of the surface plot at $f_T = 1$.



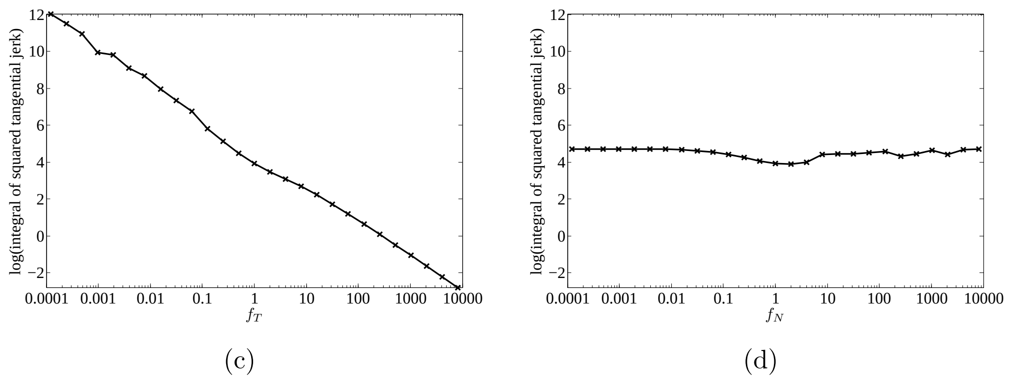


Figure 6.10: Effect of weights on integral of squared tangential jerk.
(a) Surface plot of log(integral of squared tangential jerk) as a function of f_T and f_N on a log-log scale. (b) Top view of the surface plot. (c) Slice of the surface plot at $f_N = 1$. (d) Slice of the surface plot at $f_T = 1$.

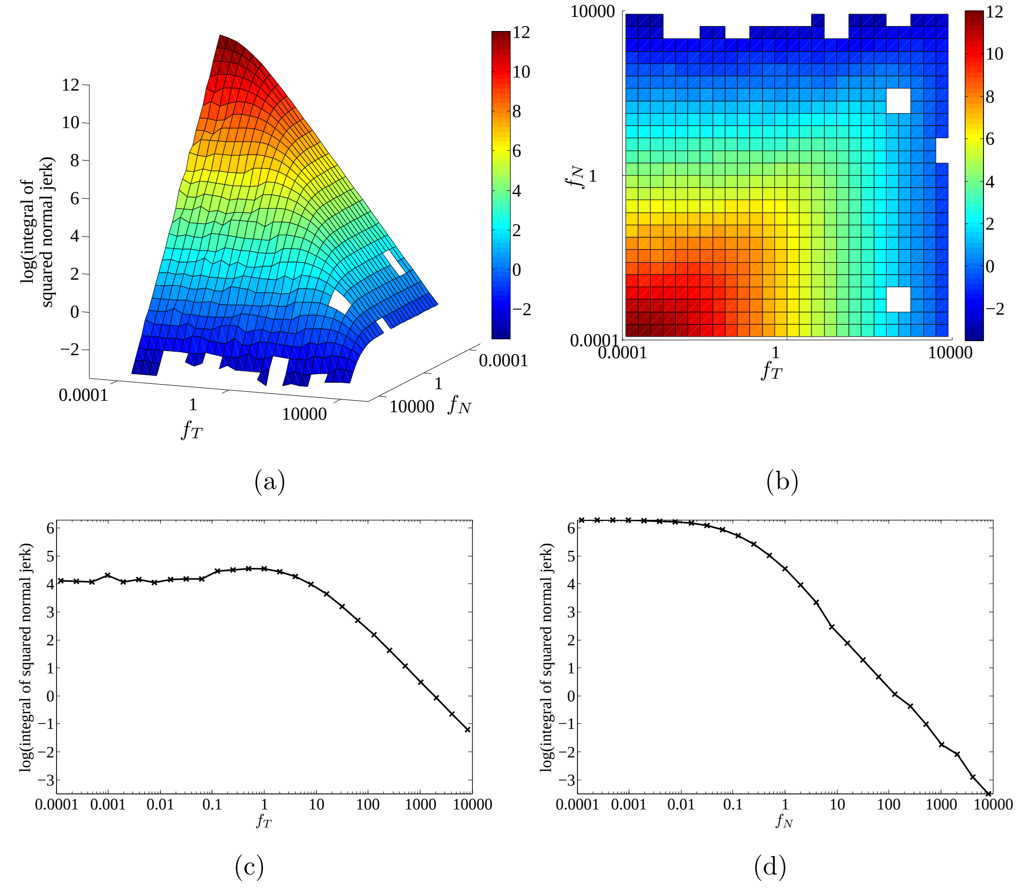
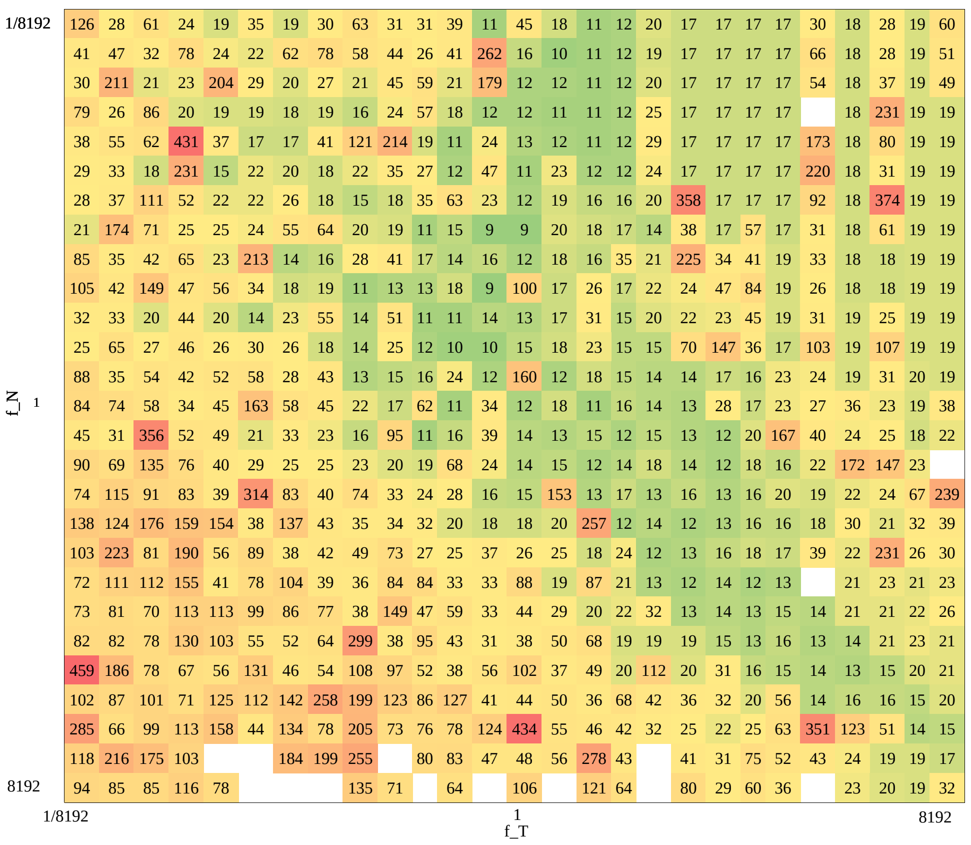


Figure 6.11: Effect of weights on integral of squared normal jerk. (a) Surface plot of log(integral of squared tangential jerk) as a function of f_T and f_N on a log-log scale. (b) Top view of the surface plot. (c) Slice of the surface plot at $f_N = 1$. (d) Slice of the surface plot at $f_T = 1$.

Whenever a relationship exist between f_T or f_N and any of the quantities travel time, integral of squared tangential jerk, and integral of squared normal jerk, it is of the form of a power law.

From this analysis, we can draw some useful guidelines for customizing weights for comfort even though the effect of weight on discomfort is nonlinear. Since J_T is a function of f_T alone, we can devise experiments that allow a user to choose f_T that keeps tangential jerk to an acceptable level. For example, we can devise experiments that consist primarily of straight line motion, and has zero speeds on both ends. In such a motion, normal component of jerk will make none or minimal contribution to discomfort. Hence, it would be easy to set f_T. Next, we can devise experiments that consist of at least some curved segments. The user can choose f_N to keep normal jerk during this curved motion to an acceptable level. Because of power law relationships, the weights should be varied in a geometric manner rather than a linear manner for faster customization.

Figure 6.12 shows the number of iterations taken by Ipopt to find a solution. Apart from a few isolated outliers that require large number of iterations, it is clear that the number of iterations is small in the region where f_T is not too small compared to f_N and both factors are not too small either. If f_N is much larger than f_T, the problems still converge in most cases but require many iterations. Most of the failures are when f_N is too large compared to unity. Hence, we recommend that for customization f_N should not be too large compared to f_T and both should be not too small compared to unity.



	1/8192													1													8192
1/8192	126	28	61	24	19	35	19	30	63	31	31	39	11	45	18	11	12	20	17	17	17	17	30	18	28	19	60
	41	47	32	78	24	22	62	78	58	44	26	41	262	16	10	11	12	19	17	17	17	17	66	18	28	19	51
	30	211	21	23	204	29	20	27	21	45	59	21	179	12	12	11	12	20	17	17	17	17	54	18	37	19	49
	79	26	86	20	19	19	18	19	16	24	57	18	12	12	11	11	12	25	17	17	17	17		18	231	19	19
	38	55	62	431	37	17	17	41	121	214	19	11	24	13	12	11	12	29	17	17	17	17	173	18	80	19	19
	29	33	18	231	15	22	20	18	22	35	27	12	47	11	23	12	12	24	17	17	17	17	220	18	31	19	19
	28	37	111	52	22	22	26	18	15	18	35	63	23	12	19	16	16	20	358	17	17	17	92	18	374	19	19
	21	174	71	25	25	24	55	64	20	19	11	15	9	9	20	18	17	14	38	17	57	17	31	18	61	19	19
	85	35	42	65	23	213	14	16	28	41	17	14	16	12	18	16	35	21	225	34	41	19	33	18	18	19	19
	105	42	149	47	56	34	18	19	11	13	13	18	9	100	17	26	17	22	24	47	84	19	26	18	18	19	19
	32	33	20	44	20	14	23	55	14	51	11	11	14	13	17	31	15	20	22	23	45	19	31	19	25	19	19
	25	65	27	46	26	30	26	18	14	25	12	10	10	15	18	23	15	15	70	147	36	17	103	19	107	19	19
	88	35	54	42	52	58	28	43	13	15	16	24	12	160	12	18	15	14	14	17	16	23	24	19	31	20	19
1	84	74	58	34	45	163	58	45	22	17	62	11	34	12	18	11	16	14	13	28	17	23	27	36	23	19	38
	45	31	356	52	49	21	33	23	16	95	11	16	39	14	13	15	12	15	13	12	20	167	40	24	25	18	22
	90	69	135	76	40	29	25	25	23	20	19	68	24	14	15	12	14	18	14	12	18	16	22	172	147	23	
	74	115	91	83	39	314	83	40	74	33	24	28	16	15	153	13	17	13	16	13	16	20	19	22	24	67	239
	138	124	176	159	154	38	137	43	35	34	32	20	18	18	20	257	12	14	12	13	16	16	18	30	21	32	39
	103	223	81	190	56	89	38	42	49	73	27	25	37	26	25	18	24	12	13	16	18	17	39	22	231	26	30
	72	111	112	155	41	78	104	39	36	84	84	33	33	88	19	87	21	13	12	14	12	13		21	23	21	23
	73	81	70	113	113	99	86	77	38	149	47	59	33	44	29	20	22	32	13	14	13	15	14	21	21	22	26
	82	82	78	130	103	55	52	64	299	38	95	43	31	38	50	68	19	19	19	15	13	16	13	14	21	23	21
	459	186	78	67	56	131	46	54	108	97	52	38	56	102	37	49	20	112	20	31	16	15	14	13	15	20	21
	102	87	101	71	125	112	142	258	199	123	86	127	41	44	50	36	68	42	36	32	20	56	14	16	16	15	20
	285	66	99	113	158	44	134	78	205	73	76	78	124	434	55	46	42	32	25	22	25	63	351	123	51	14	15
	118	216	175	103			184	199	255		80	83	47	48	56	278	43		41	31	75	52	43	24	19	19	17
8192	94	85	85	116	78				135	71		64		106		121	64		80	29	60	36		23	20	19	32

Figure 6.12: Number of iterations for the range of weight factors. The green cells indicate smaller number of iterations compared to red cells. It is clear that the least number of iterations are taken in the region where both factors are greater than 1/32 and one factor is roughly within 1/16 to 16 times the other. The empty cells correspond to problems that failed to converge.

6.5 Reliability

To evaluate the reliability of our method, we construct a set of 7500 problems with different boundary conditions and solve the full constrained optimization problem corresponding to each of the 4 initial guesses for each problem. We do not include obstacles in this test.

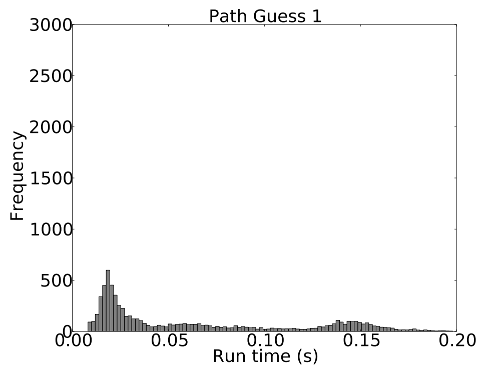
We generate the problem set as follows. Fix the initial position as $\{0, 0\}$ and orientation as 0. Choose final position at different distances along radial lines from the origin. Choose 10 radial lines that start from 0 degrees and go up to 180 degrees in equal increments. The distance on the radial line is chosen from the set $\{1, 2, 4, 8, 16\}$. The angle of the radial line and the distance on the line determines the final position. Choose 30 final orientations starting from 0 up to 360 degrees (360 degrees not included) in equal increments. The speed, v, and tangential acceleration, a_T, at both ends are varied by choosing $\{v, a_T\}$ pairs from the set $\{\{0, 0\}, \{1, -0.1\}, \{1, 0\}, \{1, 0.1\}, \{3, 0\}\}$. Thus we have 10 radial lines, 5 distances on each radial line, 30 orientations, 5 $\{v, a_T\}$ pairs, resulting in $10 \times 5 \times 30 \times 5 = 7500$ cases.

Each problem has 189 degrees of freedom, 2018 constraints, out of which 66 are equality constraints and 1952 are inequality constraints. For computation of initial guess of path, we set the maximum number of iterations to 100. For discomfort minimization problem we set the maximum number of iterations to 200. An average of 3.6 solution paths were found for each problem. This average would be higher if we set the maximum number of iterations even higher. However, since we wanted to evaluate how reliably our method performed in a reasonable amount of computation time, we kept the maximum number of iteration as 200.

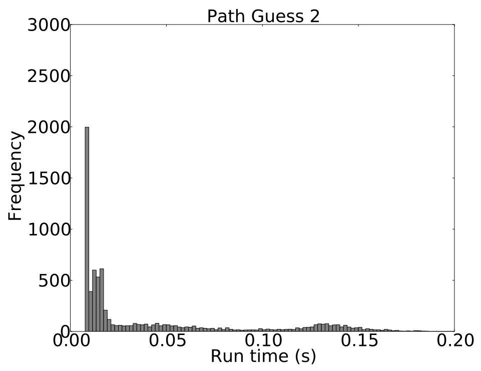
All the problems were solved on a laptop with Intel Core i7 CPU running at 2.67 GHz, 4 GB RAM, and 4 MB cache size. Histograms of run-time for computing initial guess of speed, initial guess of path, and solution to the discomfort minimization problem are shown in Figures 6.14, 6.13, and 6.15 respectively. Histograms for all four initial guesses and all four discomfort minimization problems are shown. In all these histograms, we have removed 1% or less of cases that lie outside the range of the axis shown for better visualization. All histograms show both successful and unsuccessful cases.

From Figure 6.13 we see that than 99% or more of initial guesses of path are computed in less than 0.2 s. From Figure 6.14 we see that 99% or more of initial guesses of speed are computed in less than 0.12 s. From Figure 6.15 we see that 99% or more of the solutions of the full problem are computed in less than 4 s. This is further visualized in Figure 6.16 that shows a normalized cumulative histogram.

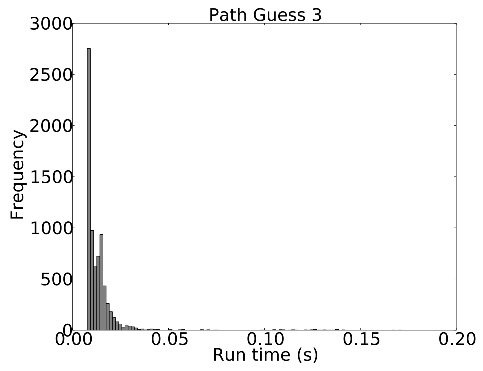
Histograms of number of iterations for computing final solution are shown in Figure 6.17. On average, 90% all four solutions were computed in 100 iterations or less.



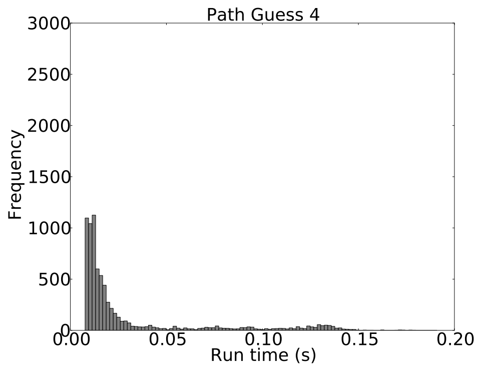
(a) 99% solved within 0.2 s.



(b) 99% solved within 0.2 s.

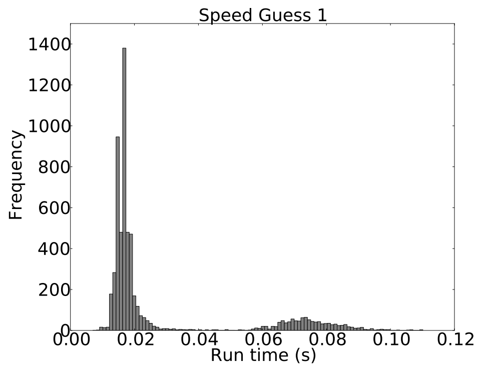


(c) 100% solved within 0.2 s.

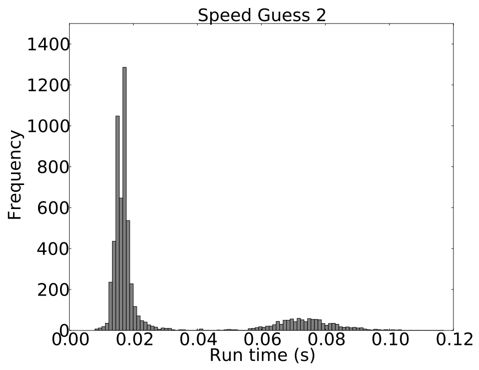


(d) 100% solved within 0.2 s.

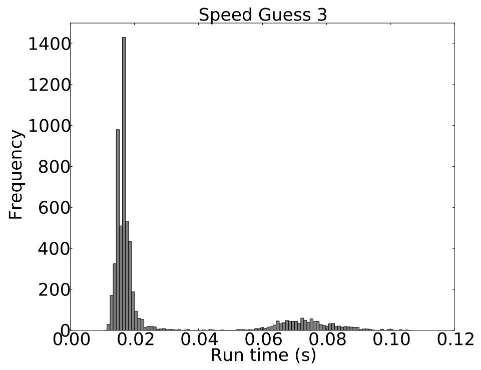
Figure 6.13: Histogram of time taken to compute initial guess of path. This includes both successful and unsuccessful cases. Total 7500 cases.



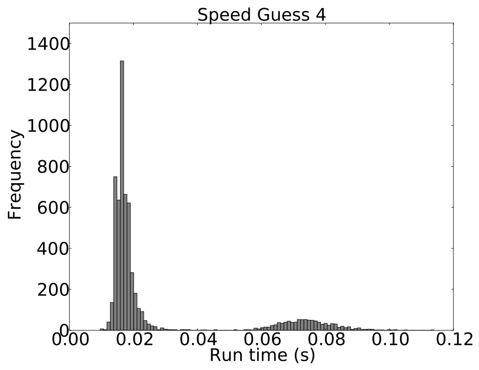
(a) 99% solved within 0.12 s.



(b) 99% solved within 0.12 s.

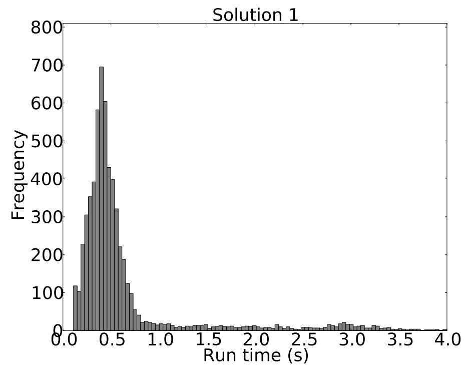


(c) 99% solved within 0.12 s.

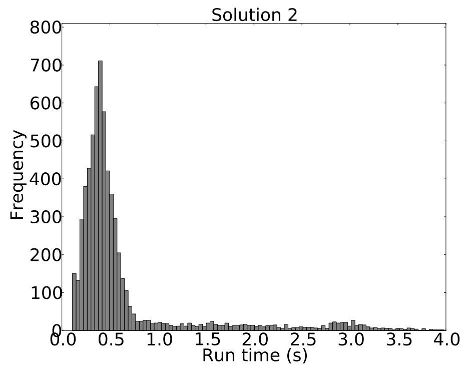


(d) 99% solved within 0.12 s.

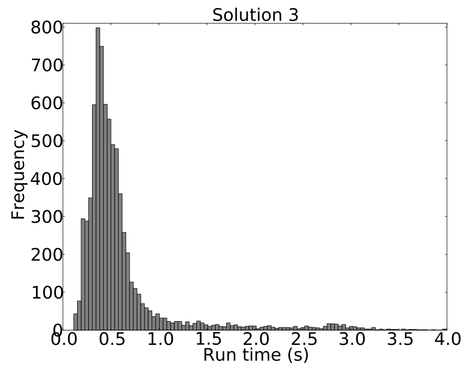
Figure 6.14: Histogram of time taken to compute initial guess of speed. This includes both successful and unsuccessful cases. Total 7500 cases.



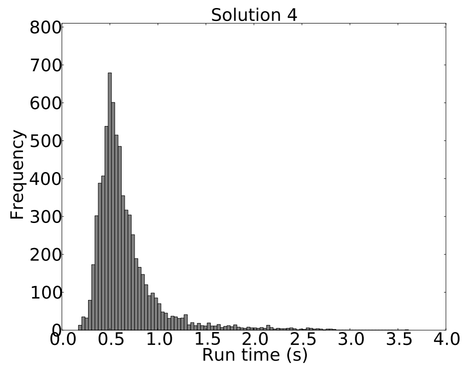
(a) 99% solved within 4 s.



(b) 99% solved within 4 s.

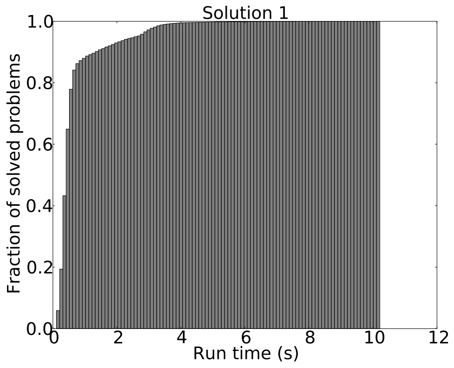


(c) 99% solved within 4 s.

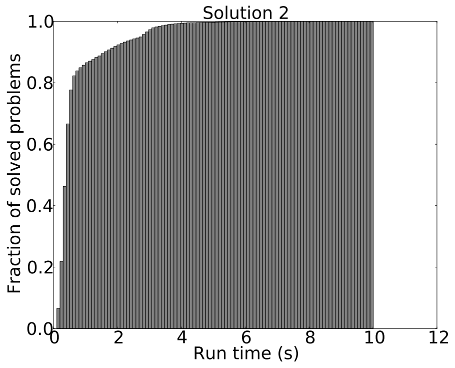


(d) 100% solved within 4 s.

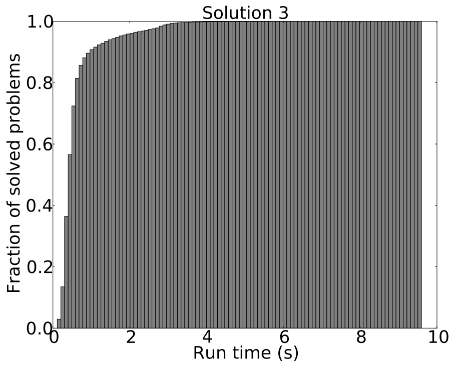
Figure 6.15: Histogram of time taken to compute solution of discomfort minimization problem.
This includes both successful and unsuccessful cases. Total 7500 cases.



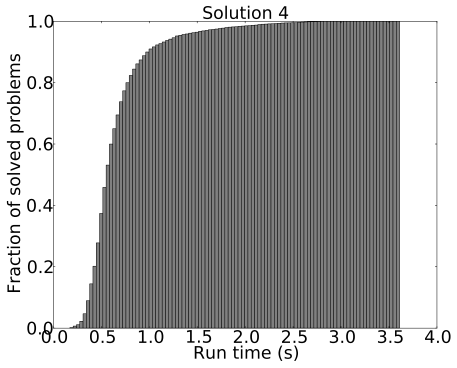
(a) 99% solved within 4 s.



(b) 99% solved within 4 s.

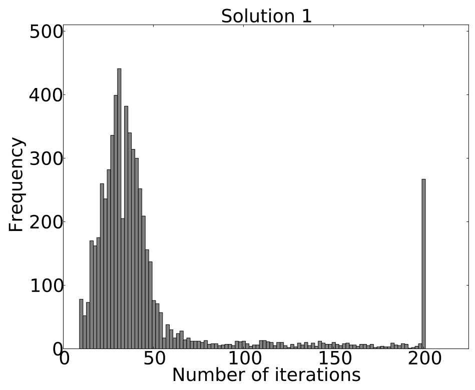


(c) 99% solved within 4 s.

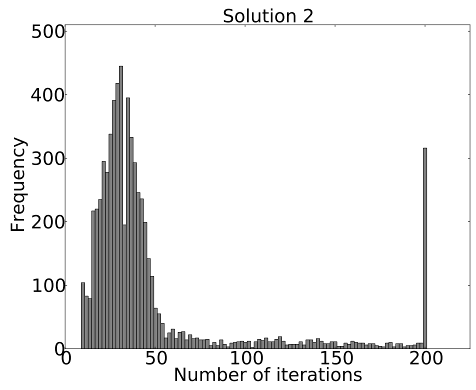


(d) 100% solved within 4 s.

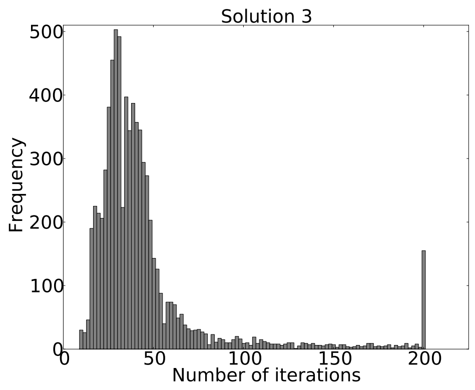
Figure 6.16: Normalized cumulative histogram of time taken to compute solution of discomfort minimization problem.
This includes both successful and unsuccessful cases. Total 7500 cases.



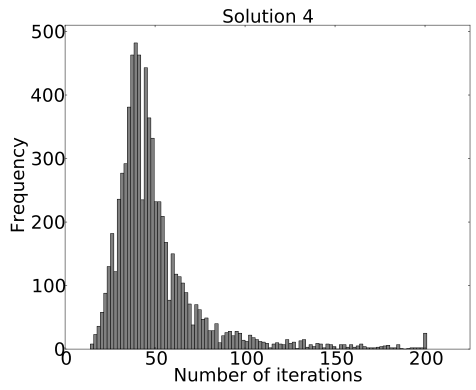
(a) 89% solved in 100 iterations or less



(b) 88% solved in 100 iterations or less



(c) 93% solved in 100 iterations or less



(d) 94% solved in 100 iterations or less

Figure 6.17: Histogram of number of iterations to compute solution of discomfort minimization problem.
Total number of problems is 7500. The peak at 200 iterations is due to failed cases since maximum number of iterations was set to 200.

6.6 Discussion of Results and Limitations

Results show that our framework is capable of reliably planning trajectories between a large variety of boundary conditions and for a range of weights. 97% of 2916 unconstrained problems for a fixed boundary condition but varying weights were solved successfully when weights were varied by 8 orders of magnitude. Out of a set of 7500 examples with varying boundary conditions, and all dynamic constraints imposed, 3.6 solution paths, on average, were found per example. The time taken to compute the solution to the discomfort minimization problem was less than 10 seconds for all the cases, 99% of all problems were solved in less than 4 seconds, and roughly 90% were solved in less than 100 iterations.

We also saw that our framework can plan trajectories with a variety of boundary conditions that avoid obstacles. We presented concrete examples for circular, elliptical, and star-shaped obstacles.

Thus our framework, with some more speedups in run-time, can be implemented for efficient and robust motion-planning of assistive mobile robots. We will discuss possible way of achieving speedups in computational time in Section 7.1. One of the limitations of our framework, in its current implementation, is that if the initial guess of path passes through obstacles, it may take a large number of iterations for the optimization algorithm to converge to a solution, and sometimes a solution may not be found. We have observed this on some example cases and this will need a more careful analysis in the future. One way to deal with this issue is to generate initial guesses of path that are obstacle free, and is part of future work.

There are many tasks in which an assistive robot must back up and then move towards the goal. For example, if a robot is positioned at a user's desk, it cannot move forward. To go anywhere it must back up first. Such tasks can be handled with the help of a high-level planner that breaks this sequence into two and provides a set of two boundary conditions in sequence to our framework – one for backing up and one for the goal. The intermediate waypoint can also be chosen by an optimization process.

In our method, we impose obstacle avoidance constraints on a discrete set of points on the path. Thus, we cannot guarantee that segments of the path between these points will not intersect obstacles. In practice, if the points are chosen to be close enough, so that distance between these points is smaller than most obstacles, the path would be collision-free. Even so, sharp pointed corners of obstacles can intersect the path. This can be resolved in three ways. First, when we incorporate robot's body for obstacle-avoidance an extra margin of safety can be added. Second, we can implement an efficient collision checker that checks the final optimal path for collisions with obstacles by dividing it into small segments. A robot should execute this trajectory only if the collision checker finds no collisions. Third, obstacles can be represented with a piecewise smooth boundary curve that enloses the obstacle shape such that sharp corners are smoothed out.

Chapter 7

Concluding Remarks and Directions for Future Research

We make two main contributions in this work. First, we recognize that for an autonomous assistive robot to be acceptable to a human user, its motion should not only be safe, it should be also be comfortable. We formalize the notion of motion comfort in a way that can be used to compute trajectories for an autonomous wheeled mobile robot in an optimization framework. Among the various contributing factors to comfort, we focus on dynamic factors. For comfortable motion, a trajectory should have the following properties – it should boundary conditions on speed and tangential acceleration at the start and end points, have smooth and bounded accelerations, the geometric path should avoid obstacles, have curvature continuity, and should satisfy boundary conditions on curvature. In the absence of relevant comfort studies for assistive robots, we developed a characterization of discomfort based on comfort studies for ground vehicles and studies of human arm motion. While human user studies are required to validate this measure of discomfort, we believe that we have taken an important first step in formalizing the notion of motion comfort for assistive robots.

Second, we develop a novel motion planning framework to plan trajectories in small-scale space such that the trajectories minimize discomfort and have all the properties described above. Our framework removes the limitations of existing motion planning methods, none of which can plan trajectories that have all the properties necessary for comfort. To the best of our knowledge, this is the first comprehensive formulation of kinodynamic motion planning for wheeled mobile robots that includes all of the following – a careful analysis of boundary conditions and continuity requirements on trajectory, dynamic constraints, obstacle avoidance constraints, and a robust numerical method that computes solution trajectories in a few seconds.

One of the strengths of our framework is that it is easy to incorporate additional kinematic and dynamic constraints, and additional terms can also be incorporated in the discomfort functional. Of course, care has to be taken to keep the problem mathematically meaningful. While this motion planning framework was developed for assistive mobile robots, it can be applied to motion planning of other classes of wheeled mobile robots, including robotic cars.

Results show that our framework is capable of reliably planning trajectories for a large variety of boundary conditions. For application to real-world robotic systems, some important extensions to our framework will be required. First, our current implementation achieves obstacle avoidance for a point robot. We have described a method for incorporating robot shape, and this will have to be implemented. Second, our results show that time taken to find a solution is of the order of seconds. This will have to be reduced a hundred-fold to make this framework feasible to be implemented for real-time planning. We discuss these, and several other extensions, below.

7.1 Directions for Future Research

Incorporating robot shape for obstacle avoidance. We described a general method to incorporate arbitrary shaped robot body in Section 3.10.4. This method consists of modeling the robot as a closed curve that encloses the projection of its boundary in the plane of motion, choosing a set of points on this curve, and imposing the constraints that all these points be outside all obstacles. If m points are chosen on the boundary and there are n obstacles, this method will result in $m \times n$ constraints. A more efficient approach may be possible when the robot can be modeled by a simple shape such as a circle or a convex polygon. Since most mobile robots, in practice, have simple shapes, it is worthwhile to explore these shapes as special cases for obstacle avoidance.

Incorporating moving obstacles. One way to incorporate moving obstacles is to frequently update a map of the world and use this updated map to re-plan a new trajectory starting from the current state. Since our method plans trajectories in small-scale space, and there exist efficient methods for computing and updating a local map, moving obstacles can be avoided if the trajectories can be planned fast enough. We used such an approach in our previous work [68].

For comfort of a human user, it might be useful to develop models that estimate a moving obstacle's trajectory, and use this trajectory during planning. This could result in paths that have fewer changes in direction (compared to those found by fast-re planning) and are perceived to be more comfortable. Such obstacle models have been previously employed for motion planning [22].

Culling obstacles intelligently. In our method, we choose a set of points on the path, and impose the constraint that all obstacles be outside all points on the path. In our earlier approaches, we have experimented with culling these obstacles intelligently so that the number of obstacle constraints is reduced. If the trajectory is well-behaved, that is, if the geometric path does not have too many self intersections, and if one iterate does not vary too wildly from the previous, then we may be able to achieve a reduction in the number of constraints.

First, we can remove, in advance, all obstacles that are too far from the initial guess of path. Second, for every point, we impose the constraint that it be outside obstacles within its "neighborhood" rather than being outside all obstacles. Under the above described conditions, if a point is outside obstacles in its neighborhood, it can be expected to be outside all other obstacles that are far from it. In our experiments with our current approach, we have observed that the above conditions hold if the initial guess of path is outside obstacles.

Reducing computational time. For real-time implementation, it would be necessary to achieve at least a 10–fold or preferably a 100–fold reduction in the computational time so that the problem is solved in one hundredth of a second. Many steps can be taken to achieve this.

First, we have observed that when an initial guess of path is inside an obstacle, it takes longer for the optimization algorithm to converge to a solution. Therefore, it would be worthwhile to invest some effort in generating an initial guess of path that is outside obstacles. This would reduce the number of iterations required to find a solution.

Second, intelligently culling obstacles and efficiently implementing obstacle avoidance constraints for special robot shapes, as discussed earlier, could result in significant reduction in the number of constraints and faster computations in every iteration.

Third, a multi-step optimization procedure can be tried. A coarser finite element mesh with fewer elements can be used to find a solution which would serve as an initial guess for a problem with a finer mesh.

Finally, parallelism inherent in the problem can be exploited and parts of the program can be executed on a GPU. For example, computation of constraint values, gradients and Hessians can be parallelized. Other such parallelisms should also be exploited. In addition, many other code optimizations can also be implemented.

Evaluating the "goodness" of discomfort measure. We have formulated a measure of discomfort based on comfort studies in ground vehicles such as automobiles and trains. To the best of our knowledge, no such studies have been conducted for assistive robots. Since discomfort is subjective, the best way to assess comfort is to ask a user. Hence, to validate this discomfort measure, human user studies should be conducted with enough users to yield statistically significant data. We provide some guidelines on how such a study may be conducted in Section 7.2 below.

Motion planning for ramps and non-planar surfaces. The motion planning framework presented in this work was developed for planning trajectories for a wheeled mobile robot moving on a plane. This assumption holds, for the most part, in indoor environments. For navigating in an urban outdoor environment, an assistive robot is often required to move up and down ramps. Since a ramp is a planar surface, a relatively simple extension of our framework may enable motion planning for moving up and down on ramps. Navigating sideways on ramps, and on other undulating surfaces, such as parks, would likely require a more significant extension.

7.2 Implementation of the Motion Planning Framework for Human Users

Once robot shape has been incorporated for obstacle avoidance and a reasonable reduction in computational time has been achieved, this framework can be implemented on an assistive robot and a study with human users can be performed. The purpose of such a study would be to either confirm that the measure of discomfort is good by showing that multiple human users can achieve comfort after choosing the weights, or failing that, to provide additional insight into what might be missing. Below are some guidelines on how to implement the framework on an assistive robot and how to conduct such a study.

- Our motion planning framework requires a representation of small-scale space to plan trajectories. An occupancy-grid based representation of small-scale space can be used. In such a representation, obstacles are represented as occupied cells in the grid. See [93] for a detailed discussion of such a representation. For efficient motion planning, these cells should be grouped together, where possible, into a single polygonal obstacle. When such a grouping yields an obstacle that is not star-shaped, it should be decomposed into a union of star-shaped polygons. An efficient algorithm for doing so can be found in [3].
- A goal state consisting of position, orientation, curvature, speed, and magnitude of tangential acceleration, is required as input to the motion planning framework. Position and orientation may be provided by a human user through some input device (e.g by clicking on a map as in [68]). Curvature should be set to zero. Speed may be specified as zero if it is desired to stop at the final position, otherwise is should

be a speed that is typically found comfortable by the user. Tangential acceleration should be set to zero. For navigating in large-scale space, a high-level planner such as that used in [68] could be used for generating intermediate way points. Such a planner usually provides only position and orientation. The rest of the quantities can be provided according to the guidelines above.

- All necessary bounds should also be provided as input. The bounds in Table 6.1 may be used as a start.

- A controller that can track the planned trajectory should be implemented. We have achieved good tracking accuracy, in our previous work [68], with a feedback-linearization based controller described in [64]. The trajectory tracking accuracy of this controller should be carefully evaluated.

- Before performing human user experiments, the framework should be comprehensively tested in the environment in which the users will evaluate it. If the environment is likely to have moving obstacles, fast re-planning should be implemented. This requires trajectories to be computed in at most a tenth of second. A relatively safe indoor environment with no drop-offs and other hazards should be chosen and common failure cases should be identified via experimentation.

- In the first step of the study, a user should be asked to manually operate the assistive robot on a variety of tasks. A speed that the user typically operates at should be determined from these tasks.

- Although a more detailed study than that described in Section 6.4 could yield an empirical relationship between weights and the individual terms in our discomfort measure, such a study is not an absolute prerequisite to performing human user studies. The two dimensionless factors corresponding to the weights for integral of squared tangential jerk and squared normal jerk are the parameters that should be varied in the experiments.

- First, the weight factor for tangential jerk should be determined. To do this, the following experiment can be conducted. Set start and end boundary conditions such that motion is along a straight line. Set initial and final speed and acceleration to zero. Use the motion planning framework to plan trajectories for this task for a range of weight factors for tangential jerk. Ask the user to compare discomfort for every pair of weights. This comparison should include subjective questions on overall comfort as well as questions comparing the level of tangential jerk, and asking whether the time of travel was satisfactory. Vary the total length of the path and repeat the experiment for multiple lengths. Based on these experiments, fix a value of this weight factor.

- Next, the weight factor for normal jerk should be determined. To do this, the following experiment can be conducted. Set start at end boundary conditions such that most of the motion is along a curved path. One way to achieve this is by choosing final position very close to the start position such that the robot has to travel along a curve to reach the goal. Follow a procedure similar to the one described above (for tangential jerk) to determine the weight factor for normal jerk.

- Once the weight factors are determined, a set of motion tasks with a variety of boundary conditions should be performed and user should be asked to rate comfort.

- If the motion for the above tasks is found to be comfortable, then it can be concluded that the measure of discomfort, in fact, captures user discomfort. If not, a set of questions designed to learn what might be missing should be asked.

- In all cases, all quantitative information such as speed, acceleration, jerk, travel time, length of path etc., should be collected.

7.3 Summary

In this work, we formalized the notion of motion comfort for assistive mobile robots. We developed a motion planning framework for kinodynamic motion panning for a wheeled mobile robots moving on a plane that minimizes user discomfort and plans safe, comfortable, and customizable trajectories. We have outlined a method by which a user may customize the motion and presented some guidelines for conducting human user studies to validate and/or refine the measure of discomfort presented in this work. We believe that our work is an important step in developing autonomous assistive robots that are acceptable to human users.

Bibliography

[1] R. A. Adams and J. F. Fournier. *Sobolev spaces.* Elsevier, 2003.

[2] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz. An optimality principle governing human walking. *IEEE Transactions on Robotics*, 24:5–14, 2008.

[3] D. Avis and G. Toussaint. An efficient algorithm for decomposing a polygon into star-shaped polygons. *Pattern Recognition*, 13(6):395–398, 1981.

[4] D. J. Balkcom and M. T. Mason. Time optimal trajectories for differential drive vehicles. *International Journal of Robotics Research*, 21(3):199–217, 2002.

[5] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10:72–89, 1990.

[6] P. Beeson, M. MacMahon, J. Modayil, A. Murarka, B. Kuipers, and B. Stankiewicz. Integrating multiple representations of spatial knowledge for mapping, navigation, and communication. In *Symposium on Interaction Challenges for Intelligent Assistants*, AAAI Spring Symposium Series, pages 1–9, 2007. AAAI Technical Report SS-07-04.

[7] P. Beeson, J. Modayil, and B. Kuipers. Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy. *International Journal of Robotics Research*, 29(4):428–459, 2010.

[8] C. Guarino Lo Bianco and M. Romano. Smooth motion generation for unicycle mobile robots via dynamic path inversion. *IEEE Transactions on Robotics*, 20(5):884 – 891, 2004.

[9] C. Guarino Lo Bianco and M. Romano. Bounded velocity planning for autonomous vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 685–690, 2005.

[10] J. E. Bobrow, B. Martin, G. Sohl, E. C. Wang, F. C. Park, and J. Kim. Optimal robot motions for physical criteria. *Journal of Robotic Systems*, 18(12):785–795, 2001.

[11] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control : Optimization, Estimation, and Control.* Hemisphere Publishing Corporation, 1975.

[12] CEN. Railway applications - ride comfort for passengers - measurements and evaluation. ENV 12299, 1999.

[13] P. Chakroborty and A. Das. *Principles of Transportation Engineering.* PHI Learning Pvt. Ltd., 2004.

[14] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, 2005.

[15] D. W. Conner. Passenger comfort technology for system decision making. *Human factors in transport research*, 20:51–57, 1980.

[16] B.R. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the ACM*, 40(5):1048–1066, 1993.

[17] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[18] L. Fehr, W. E. Langbein, and S. B. Skaar. Adequacy of power wheelchair control interfaces for persons with severe disabilities: A clinical survey. *Journal of Rehabilitation Research and Development*, 37(3):353–360, 2000.

[19] P. Ferbach. A method of progressive constraints for nonholonomic motion planning. In *IEEE International Conference on Robotics and Automation*, pages 2949–2955, 1996.

[20] D. Ferguson, T. M. Howard, and M. Likhachev. Motion planning in urban environments: Part I. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1063–1069, 2008.

[21] C. Fernandes, L. Gurvits, and Z. X. Li. A variational approach to optimal nonholonomic motion planning. In *IEEE International Conference on Robotics and Automation*, pages 680–685, 1991.

[22] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(2):760–772, 1998.

[23] T. Flash and N. Hogan. The co-ordination of arm movements: an experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5:1688–1703, 1985.

[24] J. Förstberg. *Ride comfort and motion sickness in tilting trains: Human responses to motion environments in train experiment and simulator experiments*. PhD thesis, KTH Royal Institute of Technology, 2000.

[25] T. Fraichard. Dynamic trajectory planning with dynamic constraints: A state-time space approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1394–1400, 1996.

[26] T. Fraichard and A. Scheuer. From Reeds and Shepp's to continuous-curvature paths. *IEEE Transactions on Robotics and Automation*, 20(6):1025–1035, 2004.

[27] J. Glover. Transition curves for railways. In *Minutes of Proceedings of the Institution of Civil Engineers*, pages 161–179, 1900.

[28] S. Gulati, C. Jhurani, B. Kuipers, and R. Longoria. A framework for planning comfortable and customizable motion of an assistive mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4253–4260, 2009.

[29] S. Gulati and B. Kuipers. High performance control for graceful motion of an intelligent wheelchair. In *IEEE International Conference on Robotics and Automation*, pages 3932–3938, 2008.

[30] K. Z. Haigh and H. Yanco. Automation as caregiver: A survey of issues and technologies. In *AAAI workshop on Automation as Caregiver*, pages 39–53, 2002.

[31] D. L. Hall, R.C. Loshbough, and G. D. Robaszkiewicz. Jerk, acceleration, and limited pattern generator for an elevator system. U.S. Patent number: 3523232, 1970.

[32] T. M. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *The International Journal of Robotics Research*, 26(2):141–166, 2007.

[33] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *Int. J. Robotics Research*, 21(3):233–255, 2002.

[34] L. I. Iezzoni, E. P. McCarthy, R. B. Davis, and H. Siebens. Mobility difficulties are not only a problem of old age. *Journal of General Internal Medicine*, 16(4):235–243, 2001.

[35] ISO. Mechanical vibration and shock – evaluation of human exposure to whole body vibrations - part 1: General requirements. ISO 2631-1.2(E), 1997.

[36] S. Iwnicki. *Handbook of Railway Vehicle Dynamics.* CRC Press, 2006.

[37] I. D. Jacobson, L. G. Richards, and A. R. Kuhlthau. Models of human comfort in vehicle environments. *Human factors in transport research*, 20:24–32, 1980.

[38] P. Jiménez, F. Thomas, and C. Torras. Collision detection algorithms for motion planning. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 1–53. Springer-Verlag, Berlin, 1998.

[39] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *International Journal of Robotics Research*, 5(3):72–89, 1986.

[40] L. E. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for path planning: Articulated robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1764–1772, 1994.

[41] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[42] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5:47–431, 1986.

[43] K. Kondo. Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration. *IEEE Transactions on Robotics and Automation*, 7:267–277, 1991.

[44] K. Konolige. A gradient method for realtime robot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 639 – 646, 2000.

[45] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE Conference on Robotics and Automation*, pages 1398–1404, 1991.

[46] K. J. Krapek and J. Bittar. Elevator motion profile selection. U.S. Patent number: 5266757, 1993.

[47] B. Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

[48] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the Hybrid Spatial Semantic Hierarchy. In *IEEE International Conference on Robotics and Automation*, pages 4845–4851, 2004.

[49] F. Lamiraux and J.-P. Laumond. Smooth motion planning for car-like vehicles. *IEEE Transactions on Robotics and Automation*, 17(4), 2001.

[50] R. Lamm, B. Psarianos, and T. Mailaender. *Highway design and traffic safety engineering handbook*. McGraw-Hill, 1999.

[51] H. L. Langhaar. *Dimensional Analysis and Theory of Models.* Wiley, 1951.

[52] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Press, 1991.

[53] J.-P. Laumond, P. E. Jacobs, Michel Taïx, and R. M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 10(5), 1994.

[54] J.-P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 1–53. Springer-Verlag, Berlin, 1998.

[55] Laundhart. *Theory of the alignment.* Schmorl & von Seefeld publishing house, 1887.

[56] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Computer Science Department, Iowa State University, October 1998.

[57] S. M. LaValle. *Planning Algorithms.* Cambridge University Press, 2006.

[58] S. M. LaValle. RRT page: Photo and animation gallery. `http://msl.cs.uiuc.edu/rrt/gallery_car.html`, 2009.

[59] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[60] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A. K. Peters, Wellesley, MA, 2001.

[61] M. C. Lin and D. Manocha. Collision and proximity queries. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 787–807. Chapman and Hall/CRC Press, New York, 2004.

[62] M. C. Lin, D. Manocha, J. Cohen, and S. Gottschalk. Collision detection: Algorithms and applications. In J.-P. Laumond and M. H. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 129–142. A.K. Peters, Wellesley, MA, 1997.

[63] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Trans. on Computers*, 32(2):108–120, 1983.

[64] A. De Luca, G. Oriolo, and C. Samson. Feedback control of a nonholonomic car-like robot. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 171–253. Springer-Verlag, Berlin, 1998.

[65] B. Mirtich. Efficient algorithms for two-phase collision detection. In K. Gupta and A.P. del Pobil, editors, *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, pages 203–223. Wiley, New York, 1998.

[66] Henry Packard Moreton. *Minimum Curvature Variation Curves, Networks, and Surfaces for Fair Free-Form Shape Design.* PhD thesis, EECS Department, University of California, Berkeley, Mar 1993.

[67] A. Murarka, S. Gulati, P. Beeson, and B. Kuipers. Towards a safe, low-cost, intelligent wheelchair. In *Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV)*, pages 42–50, 2009.

[68] A. Murarka and B. Kuipers. A stereo vision based 3D mapping algorithm for detecting ramps, drop-offs, and obstacles for safe local navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, to appear.

[69] B. B. Myers and B. Marshall. The influence of comfort on passenger modal choice in western Canada. *Human factors in transport research*, 20:42–50, 1980.

[70] R. D. Pepler, E. D. Sussman, and L. G. Richards. Passenger comfort in ground vehicles. *Human factors in transport research*, 20:76–84, 1980.

[71] A. Piazzi, C. Guarino Lo Bianco, and M. Romano. η^3 splines for the smooth path generation of wheeled mobile robots. *IEEE Transactions on Robotics*, 23(5), 2007.

[72] A. M. Popse and A. R. Tralov. *Disability in America: Toward a national agenda for prevention.* National Academy Press, 1991.

[73] S. Quinlan. Efficient distance computation between non-convex objects. In *IEEE International Conference on Robotics and Automation*, pages 3324 – 3329, 1994.

[74] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[75] A. J. Rentschler, R. A. Cooper, S. G. Fitzgerald, M. L. Boninger, S. Guo, W. A. Ammer, M. Vitek, and D. Algood. Evaluation of selected electric-powered wheelchairs using the ANSI/RESNA standards. *Archives of Physical Medicine and Rehabilitation*, 85:611–619, 2004.

[76] L. G. Richards. On the psychology of passenger comfort. *Human factors in transport research*, 20:15–23, 1980.

[77] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5), 1992.

[78] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars. Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *International Journal of Robotics Research*, 17:840–857, 1998.

[79] Z. Shiller. Time-energy optimal control of articulated paths with geometric path constraints. In *International Conference on Robotics and Automation*, 1994.

[80] Z. Shiller and S. Dubowsky. On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, 7(6):785–797, 1991.

[81] Z. Shiller and Y.-R. Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7(2):241–249, 1991.

[82] A. Shkolnik and R. Tedrake. Path planning in 1000+ dimensions using a task-space voronoi bias. In *IEEE International Conference on Robotics and Automation*, 2009.

[83] R. Siegwart and I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.

[84] R. C. Simpson. Smart wheelchairs: a literature review. *Journal of Rehabilitation Research and Development*, 42(4):423–436, 2005.

[85] R. C. Simpson, E. F. LoPresti, and R. A. Cooper. How many people would benefit from a smart wheelchair? *Journal of Rehabilitation Research and Development*, 45(1):53–72, 2008.

[86] J. B. Smeets and E. Brenner. A new view on grasping. *Motor Control*, 3:237–271, 1999.

[87] P. Souères and J.D. Boissonnat. Optimal trajectories for nonholonomic mobile robots. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 93–170. Springer-Verlag, Berlin, 1998.

[88] H.-K. J. Spielbauer and M. Peters. Elevator start jerk removal. U.S. Patent number: 5424498, 1995.

[89] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich. Common metrics for human-robot interaction. In *ACM SIGCHI/SIGART Conference Human-Robot Interaction*, 2006.

[90] J. Strizzi, I. M. Ross, and F. Fahroo. Towards real-time computation of optimal controls for nonlinear systems. In *AIAA Guidance, Navigation, and Control Conference*, 2002.

[91] H. Suzuki. Research trends on riding comfort evaluation in japan. *Proceedings of the Institution of Mechanical Engineers – Part F – Journal of Rail and Rapid Transit*, 212(1):61–72, 1998.

[92] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[93] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* MIT Press, 2005.

[94] E. Todorov and M. Jordan. Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements. *Journal of Neurophysiology*, 80(2):696–714, 1998.

[95] H. Tominaga and B. Bavarian. Global robot path planning using exact variational methods. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 617–619, 1990.

[96] J. L. Troutman. *Variational Calculus and Optimal Control: Optimization with Elementary Convexity.* Springer, 2 edition, 1995.

[97] M. Žefran. *Continuous Methods for Motion Planning.* PhD thesis, University of Pennsylvania, Philadelphia, PA, 1996.

[98] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[99] H. A. Yanco and J. L. Drury. A taxonomy for human-robot interaction. In *AAAI Fall Symposium on Human-Robot Interaction*, pages 111–119, 2002. AAAI Technical Report FS-02-03.