UNIVERSITY OF CALIFORNIA

SANTA CRUZ

Autonomous Ground Vehicle Path Planning for Autocross Tracks: Optimal vs an Efficient Bézier Curve Path

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

with an emphasis in ROBOTICS & CONTROL

by

John P. Ash

September 2015

The Thesis of John Ash
is approved:

Table of Contents

List of Figures

List of Tables

Abstract

Autonomous Ground Vehicle Path Planning for Autocross Tracks

by

John Ash

This thesis proposes a computationally efficient path planning algorithm for an autonomous ground vehicle. A Bézier curve solution is proposed that maintains G^2 continuity throughout the track. A dynamic programming algorithm plans two initial paths through the course. The first path minimizes the maximum curvature (MMC), while the second path minimizes the distance traveled. By blending the MMC and shortest paths a pseudo-optimal path is calculated based on the vehicle dynamics. The pseudo-optimal path achieves a shorter lap time than either the MMC or shortest paths.

The improved Bézier pseudo-optimal path is compared to a direct optimal control solution found using pseudospectral methods. This comparison reveals previously unrecognized potential for improvement of the dynamic programming algorithm. The dynamic programming algorithm is shown to be highly dependent on the placement of gates throughout the course and it is found that by adding extra gates along the entrance and exits of complex curves the track time can be greatly improved while keeping computation time low. The solution given in this thesis maintains a linear increase in computation time while approaching the optimal track time.

Acknowledgments & Dedications

I would like to thank Dr. Gabriel Elkaim for all of his advice and time, as well as his accepting me into the autonomous systems lab and allowing me to work on the autonomous ground vehicle path planning algorithm. Dr. Elkaim has helped shape me into the engineer I am today. Thank you also to Dr. Renwick Curry for helping shape this project into a wonderful learning experience. His advice and lessons are the reason I was able to complete this project. I would also like to thank Dr. Qi Gong, who was willing and able to help me gain a better understanding of the pseudospectral method.

A special thank you to Dr. Bryant Mairs for his assistance in turning my thesis into the work it is. Bryant taught me many lessons throughout graduate school, and has become a dear friend. Finally a thank you to my friends, family, and partner/finace/wife for supporting me through graduate school.

Chapter 1

Introduction

1.1 Overview

The design of a computationally efficient path planning algorithm is a complex but important problem. Path planning research has many applications from autonomous vehicles to space missions. Decades of research have been used to improve paths from simple straight line paths to fully optimal racing lines. Much research has been focused on optimal path planning, however the computation time associated with calculating the optimal path is intractable for any on-line application. For an on-line system, the vehicle must be able to rapidly reprogram a path that is both feasible and near optimal, while minimizing the computation time such that the solution remains viable (i.e.: vehicle has not traveled far). While algorithms exist that are capable of finding paths through different complex environments, none are fast enough to run in real time or along a path that is continuous. This thesis continues to build on the research initially done by Choi, who developed the initial near-optimal path planning algorithm, and Fiebich, who extended that work [6, 14].

The computationally-efficient algorithm described in this thesis uses a simple straight line primitive path planning algorithm to plan an initial minimized maximum curvature path and a shortest distance path through a given course. A blending of these two straight line paths is calculated using either a sequential quadratic programming method or a particle swarm optimization method. The resultant path minimizes the time through the race course. The straight line path is than smoothed using multiple fourth order Bézier curves to smooth the final path, while maintaining continuity in the path and curvature, allowing the path be continuous in curvature and acceleration, and therefore drivable by a vehicle.

The final algorithm results are compared to a direct optimal control path that provides a theoretically optimal path for the vehicle. This comparison allows previously unrecognized potential for improvement within the blended path. The results of the direct optimal approach allow for emergent heuristics resulting in upgrading the computationally efficient algorithm to allow the path to more closely follow the direct optimal control path.

The blended computationally-efficient path has proven itself to be fast and accurate when calculating a path through an autocross course. The resultant paths are within 15% of the direct optimal paths, and an improvement of up to 60% upon previous work. This performance demonstrates that the algorithm can be used to quickly and efficiently plan a path through a complex environment (such as an autocross course).

1.2 Motivation

Auto racing is asport that began in the late 1800's. Since that time the sport has grown into an enormous undertaking with many different styles of racing. Many of the advancements that have been made to standard vehicles first appeared on the race track. These competitive events allow for the testing and showcasing of engineering innovations that then moved to commerica vehicles, such as the placment of camera on cars, which in turn allows for improvements to the autmization of vehicle safety systems.

In this day and age society is approaching a time when autonomous vehicles will become commonplace. This encroaching inevitability has led racing teams to look for ways to use autonomous vehicles to their advantage. Just as racing has pushed the mechanical performance of cars with rapidly evolving computing and sensor technology, so too will racing technology extend to include the actual driving of the vehicle. Current racing drivers can benefit from an autonomous vehicles repeatability and reliability in performing complex maneuvers. An inexperienced driver can learn how to properly handle complex maneuvers while inside an autonomous car. There are major advantages to this, as race car drivers are very expensive to train. Furthermore, as demonstrated by such venues as the Bob Bondurant School of Racing Driving, ordinary drivers could benefit a great deal from additional high performance driver training, but as stated before, the cost is usually prohibitive.

A professional race car driver must be able to make split second decisions about the optimal race line their vehicle can handle. This is a skill that is only gained after hundreds of hours behind the wheel of a vehicle and is correspondingly expensive. With the use of on-line path planning algorithms racing teams are optimizing their training process. By having an inexperienced driver ride in the vehicle while the car does the driving, the learning process is sped up by having the driver see the "optimal" path and feel where the key points to brake and accelerate are, where to corner hard, etc. The driver experiences firsthand what it feels like to deal with the G-forces instead of learning in a virtual reality system, or on non-optimal path heuristics (such as hit the apex of the curve at your highest velocity). These virtual reality systems—although useful in the training process—cannot provide the same experience of a real vehicle driving on a real course.

Many race car drivers learn to drive on short one kilometer courses in a race style known as autocross. Autocross racing is generally considered an entry point to professional racing because the results of the race are much more dependent on the drivers abilities to handle the vehicle than on the vehicle's abilities themselves. Autocross laps are time-trials, meaning that only one car races at a time on the course, allowing the vehicle and driver to dispense with worry about managing collision avoidance. Autocross courses are popular because of how inexpensive they are to run; the races are generally performed on an empty parking lot with the course marked by a set of cones.

With the ever growing popularity of autocross it is important that path planning algorithms be able to plan on-line paths through these courses. Current optimal path planning algorithms are unable to plan on-line paths through an autocross course due to two main problems: (1) the algorithm cannot be run on any on-line system due to very long computation times, and (2) the resultant path is not smooth making it an unsuitable path for vehicle guidance without some sort of ancillary software to make it driveable (such as a driver emulator). If the path is not smooth the vehicle will be unable to stay along the prescribed path at the points of discontinuity. This research presents an algorithm that is both computationally efficient and returns continuous state and control variables usable directly for vehicle guidance.

1.3 Literature Review

While autocross racing has been around for a long time, it is only recently that vehicle autonomy has become capable of driving at high speeds along arbitrary race tracks. In order for an autocross path generation algorithm to be effective, it must be able to quickly generate a path comparable to that of one driven by a professional. A great deal of research has been focused on problems similar in nature. Most research into path planning algorithms can be grouped into two categories: potential fields [20], and cell decomposition [5].

The potential field technique [18] involves discretizing the field into points and giving a weight to each of the points based on their potential, the weighting is distributed such that near obstacles the cost is hight, while near the goal the cot is low. Once the potential field is created a path through the course is generated by determining a set of points that have the lowest overall gradient, thereby finishing at the end point, and staying away from obstacles.

This technique works well in situations where the exact potential of an area varies over time or is not fully known, and is therefore often employed in real world and highly-dynamic situations. The process of statistically generating the potential of each of the points along an autocross course would be intractable in any kind of on-line path planning system.

The cell decomposition method segments the course into independent cells. This differs from the potential fields method which treats the course as a large plane. The cell decomposition method utilizes a connectivity graph to determine all possible paths to traverse from the start to the end, based on the adjacent relationship between the cells. Once all paths have been determined, the most efficient path is selected based on a cost function. For example, Suh [29] uses a cell decomposition method to divide a course into a number of different cells. By discretizing the edge of the cells into a number of points, as shown in Fig. 1.1, the final optimal path is based on one point from each of the edges.

The work by Choi [6] is a variant of Suh's work. Choi uses cell decomposition to divide a course in corner cells and straight cells. A number of points are placed within each of the corner cells. By selecting a point from each of the corner cells, a path is created, such that it traverse from each corner cell passing through the straight cells.

A straight line path is created by connecting a point from each corner cell. Since the final race path is a sequence of connected straight line segments, the final path is only G^0 continuous. Driving a path that is only G^0 continuous (continuous in position only) would require instantaneous velocity changes, meaning infinite acceleration – impossible for any physically real vehicle. As such, the path must be

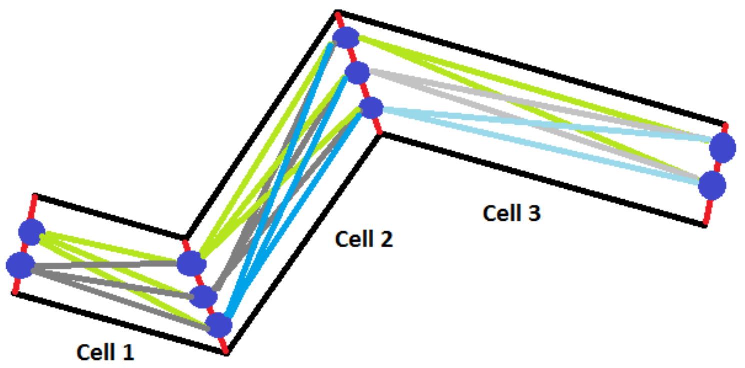


Figure 1.1: A short race course is shown which has been broken into 3 cells by the red lines. These lines are discretized into several points. Based on these points a path through the race course can be quickly determined using a connectivity graph, represented by the lines connecting each point.

smoothed to make it realizable. Dubin developed a method by which to smooth the path using constant-curvature circular arcs and straight line segments [11]. Dubin's algorithm inserts constant-curvature circular arcs in turns. This makes the paths G^1 continuous, and allows the path to be driven by a vehicle (though a Dubin's path requires instantaneous steering angle changes, which again is not physically realizable).

Improvements exist to account for more exotic curves/splines: Choi [7, 9] used Bézier curves to smooth a straight line primitive path through a course. Bézier curves are a spline based on Bernstein polynomials developed in [13]. A Bézier curve of degree n is based on $n+1$ control points, allowing Bézier curves to be easily parameterized Bézier curves have several advantages over other splines:

- They are a continuous functions whose derivatives are always known. The derivatives of a Bézier curve are themselves another Bézier curve of degree N-1.
- They start and end at their first and last control point.
- The path of a Bézier curve lies within the convex hull of its control points.

- They begin and end tangentially to the line formed when connecting the control points $0 \rightarrow 1$ and $(N-1) \rightarrow N$.

By using Bézier curves to smooth the discontinuous path through each corner cell; a vehicle is able to follow the prescribed trajectory.

Other researchers have used Bézier curves to plan complex path through dynamic environment. [16] used Bézier curves to plan a continuous path through a course and was able to calculate the velocity of the vehicle along the entire path. This path planning algorithm proved to be efficient at generating paths that sufficiently traverse a complex environment faster than previously achieved. Choi was able to minimize the curvature of these Bézier curves allowing for a higher velocity path through the course [8].

Choi's method of Bézier curve path planning allowed multiple curves to be planned depending upon a set of weighting parameters: (1) the minimized maximum curvature of the path (2) the minimized distance of the path, and (3) the maximum distance from the corridors of the path. The minimized maximum curvature path was considered to be the fastest track time path in [14]. Braghin, however proposed that the optimal racing line lies between the minimized maximum curvature path and the shortest path [2]. Braghin blended the minimized maximum curvature path and the shortest paths using a single coefficient, chosen such that it minimizes the time through the course depending upon the vehicle's dynamics. Based on this research Cardamone proposed a method which utilized multiple coefficients throughout the track [3]. For each section where the minimized maximum curvature path and the shortest path intersect, a different value was selected, by a genetic algorithm.

This method however caused discontinuities in the final path such that it was only G^0 continuous. These discontinuities were smoothed by an analytical driver model, which proved to be computationally inefficient and removed the solution further from the true optimal. This thesis proposes a method of blending the minimized maximum curvature path and the shortest path while still maintaining full continuity (in path, velocity, and steering angles).

This thesis compares this computationally efficient blending method to a direct optimal control method—similar to other optimal path planning algorithms. An example of a typical optimal control formulation can be found in Casanova, where they proposed a nonlinear programming problem to solve for the optimal lap time of a Formula One vehicle [4]. This research sparked interest in using high powered computation to search for the minimum track time of different race courses. Casanova's optimal path research involves strong computation software. Limebeer, Perantoni, and Rao most recently devised a non-linear programming problem that solved for the optimal trajectory of a Formula One car energy recovery system [23]. They used a highly dynamic full car model of the vehicle to get full calculations of the forces acting on each of the tires.

Other optimal path planning research is restricted to vehicle maneuvers that utilize the convex nature of the problem to solve for the optimal solution. Lipp and Boyd utilized a change of variables on a vehicle traveling a fixed path to exploit the convexity of the problem [24]. This change of variables allowed the velocity profile to be produced in a real time environment. Timings and Cole were able to frame a short maneuver as a convex optimization problem [30]. By changing the method by which the vehicle's path is calculated, they were able to create a convex optimization over short paths. This research, though good at calculating the paths of a vehicle, only works on a very small subset of problems and is therefore not as useful in actually generating full trajectories.

A vehicle model must be used that guarantees that the generated path is drivable for a 4 wheeled vehicle while still remaining computationally efficient. Most vehicle models are separated into three categories: point-mass models, half-car models, and full car models. Point mass systems are often used when computational efficiency matters. Jolly used a point-mass model to emulate a two wheeled robot [16]. In contrast, [32] showed that the point-mass model has some undesirable characteristics where loss of controllability can occur when the trajectory is driven by a four wheeled vehicle. However [31] showed that the undesirable characteristics were removed by upgrading to a half-car model.

The half-car model was a term coined by Krtolica [19]. This model considers the front and back half the vehicle independently, and as only a single tire each, which allows the radial acceleration of the vehicle to be more accurately modeled. Jeon utilized a half-car model to exploit the abilities of a vehicle in a dynamic and unstructured environment [15].

The full-car model is often used in direct optimal control problems. Kelly, [17], utilizes a seven-degree-of-freedom model that emulates the forces on each of the tires independently. This model takes into account wheel load transfers, engine torque, gear ratios, and limited slip differentials. These models are actively used by Formula One racing teams to test racing lines. These models are the most accurate, but they require extensive testing of the vehicle to determine the model parameters and are much slower to run.

1.4 Contributions

The following contributions were made by this thesis:

1. The design and implementation of a direct optimal control problem suitable for use on typical autocross courses.
2. The implementation of a new acceleration constraint system that emulates an ellipse with a cut off top, to better emulate a standard car.
3. Upgrade of the current dynamic programming algorithm to handle situations where gates are placed along the straight segments of the course, allowing the

vehicle to change its angle across a given straight segment.

4. A blending of two computationally efficient paths to locate a new path with a shorter overall race time.

5. A comparison of the blended path to the direct optimal path, both on common race segments, and full courses resulting in improvements to the computationally efficient blended path algorithm.

1.5 Organization

The following four chapters detail the two path planning algorithms and the evaluation of the systems. These chapters aim to provide substantial mathematical details to be used as reference for future work on the system.

Chapter 2 describes the mathematical problem of the system, including vehicle model, constraints, and cost function. Chapter 3 fully describes the two solutions to the path planning problem. The first being the direct optimal control problem, and the second the computationally efficient method. These algorithms are compared in Chapter 4, which examines the algorithms performance on a number of common race elements, and race tracks. Chapter 5 provides the concluding summary and details the future work for the algorithms.

Chapter 2

Problem Statement

2.1 Introduction

Auto racing is a sport which places several vehicles in competition against each other in order to achieve the fastest race time in a set number of laps. This style of racing has been popular since the late 1800's. Researchers and car companies are directly interested in auto racing because often it shapes the future direction of car engines, tires etc... There are many popular styles of auto racing including: NASCAR, Formula One, Rally, and Autocross. Each of these different racing styles differs in the types of vehicles driven and the platforms on which they race. This work focuses on autocross racing, which is often considered a gateway to other racing styles.

Autocross courses focus on cornering maneuvers rather than large straight segments, this allows inexpensive (compared to general race cars) cars to participate in the sport without the requirements for a vehicle with a large engine. Autocross is becoming more popular every year, due to it's low-cost of entry. Entrance fees to autocross races are able to stay inexpensive because it is cheap and easy to setup autocross courses almost anywhere. Unlike other racing styles which require large and expensive arenas or tracks, autocross is generally performed on a parking lot with the course marked by traffic cones, allowing for modularity and multi-track use.

As autocross becomes more popular, more racing teams are focusing on training their drivers on autocross courses. Autocross provides the ability to setup and restructure complex racing maneuvers, which allows a driver to practice a specific maneuver repeatedly. Autocross allows inexperienced drivers the ability to quickly and cheaply to showcase themselves against other drivers.

As a method of decreasing the cost of training inexperienced race car drivers, autonomous cars can be used to supplement training from experienced race car drivers which is generally expensive. Autonomous cars that can repeatedly and quickly navigate a course allow a driver to rapidly learn the proper ways in which to perform several complex course maneuvers.

Many path planning algorithms exist that are able to plan an optimal path through a given race course. However, these optimal paths are computationally intractable for use with any on-line system. Even by reducing the resolution/optimality of the solution the computation time for a full course is still unable to perform in real time. Therefore a computationally-efficient method must be used which provides a near-optimal solution while remaining tractable for on-line execution. The algorithm must be able to plan a path through a given course any time, and do so quickly such that the vehicle has not significantly moved during re-computation. This allows the vehicle to re-plan a new course if the vehicle strays from the path due to any disturbances internal or external.

The optimization problem that arises from an autocross race is a standard control problem that has constraints from the vehicle dynamics as well as the boundary constraints of the path. These constraints lead into a final cost function that is then solvable using standard control techniques to determine the final time of the vehicle.

2.2 Dynamics

Several vehicle models exist that can be used to model a four wheeled race car (bicycle, half-car, full-car, and 7-degree of freedom model). However these models are unnecessarily complex. A simple point mass model can be used to generate the accelerations on the vehicle. The point mass model often over simplifies the non-holonomic nature of the vehicle by allowing the vehicle to move side to side instead of only straight and along an arc. Instead of using a more complex model, this thesis emulates the vehicles dynamics within the velocity and acceleration constraints of the vehicle.

The standard point mass vehicle model has kinematics which are controlled by the accelerations (radial and tangential) of the vehicle, and thereby completely control a vehicles position, and velocity. These factors are the basis of the vehicle model used in this thesis. However the vehicle model used is ideal meaning tire slipping and throttle issues are not taken into account within the model. Instead these factors are taken into account when constraining the vehicle's acceleration and velocity.

Autocross courses generally take place on a parking lot surface, and therefore the vehicle model used in this work is constrained to two dimensions. As seen in Fig. 2.1 the vehicles positions is specified by the mid-point of the rear axle as that is the center turning point for a vehicle with no slip. The mass distribution of the vehicle is then considered a point mass located at this center of rotation, allowing the vehicle to be modeled as a point-mass system.

For this algorithm, the only state necessary to track the vehicle is its position and velocity. Therefore the *state vector* contains the vehicles position, x, y, and the velocity of the vehicle along the x and y axis, v_x and v_y, respectively. By using the point mass model the *control vector* contains the tangential acceleration of the vehicle, a_t, and the radial acceleration which points towards the center of the arc on which the vehicle is traveling, a_r. Based on these kinematics the vehicle dynamics

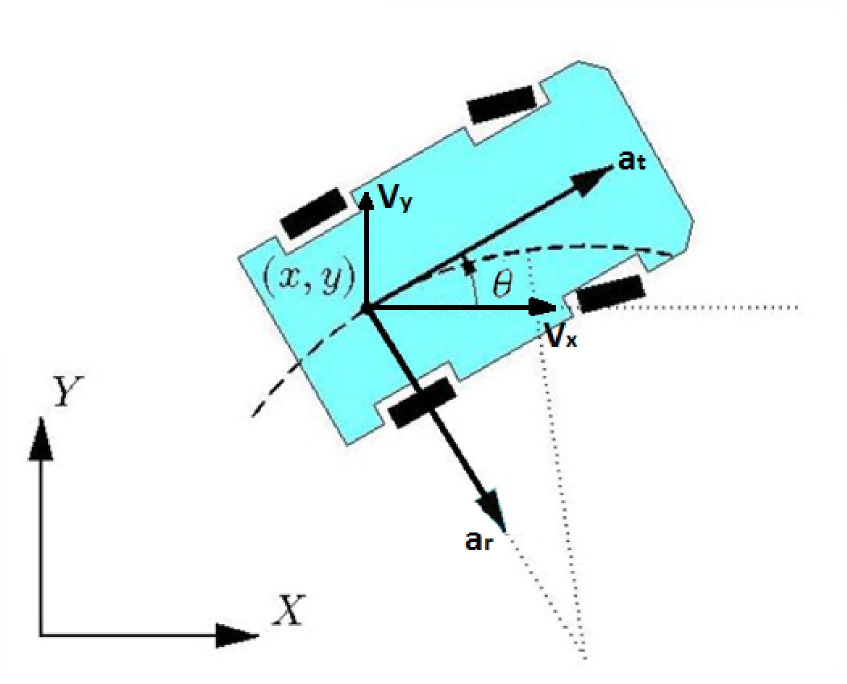


Figure 2.1: The race car as it follows a curve trajectory in the two-dimensional plane of the system model. The position of the vehicle is indicated by x,y and is the center of the rear axle. As the vehicle travels along a curve a radial acceleration, a_r, is generated that points towards the center of the arc it is traveling along. The tangential acceleration, a_t, is tangent to that same arc. The velocity of the vehicle is based on the velocity along the X axis and Y axis, v_x and v_y, respectively. θ dictates the vehicles course over ground, or it's direction of travel.

(gas and steering) can be backed out as control parameters for the vehicle. Therefore the state and control vectors are defined as $\mathbf{X} = (x, y, v_x, v_y)^T$ and $\mathbf{U} = (a_r, a_t)^T$, respectively.

The equations connecting the state and control variables define the vehicles' kinematics as it travels along a path. The vehicle kinematics are given in Eq. 2.1. These equations link the state vector and the control vector, and by using these equations the point mass model can be used to calculate how hard one pushes on the gas pedal or turns the wheel of the car. The equations are based on the radial and tangential accelerations of the vehicle. As the radial accelerations magnitude increases or decreases the turn rate of the car increases and decreases. The tangential acceleration is directly related to the forward acceleration of the vehicle such that as the tangential accelerations magnitude increases or decreases the vehicle will speed up or slow down.

$$
\begin{aligned}
\theta &= \arctan\left(\frac{v_y}{v_x}\right) \\
V &= \sqrt{v_x^2 + v_y^2} \\
\dot{x} &= v_x \\
\dot{y} &= v_y \\
\dot{v}_x &= a_t * \cos(\theta) - a_r * \sin(\theta) \\
\dot{v}_y &= a_t * \sin(\theta) + a_r * \cos(\theta) \\
\dot{\theta} &= \frac{a_r}{V}
\end{aligned}
\tag{2.1}
$$

These vehicle dynamics do not account for a situation where the velocity, V, approaches zero; the car must always be moving. If the velocity approaches zero, the point mass model allows the vehicle to turn instantaneously in any direction. This is a known problem with using a point mass vehicle model, as discussed in Section 1.3 and must be directly accounted for within the algorithms constraints. The benefits of using a simple point mass model however far outweigh the costs.

2.3 Constraints

All vehicles have different abilities and attributes, which include differences in their maximum speed, turning radius, and acceleration profiles. It is important to consider these constraints when solving for the optimal path, as there is not a single path that is optimal for all vehicles. For example, a vehicle with higher slip differential/radial acceleration limit is able to take a path with sharper turns, i.e. a shorter distance path, where as a vehicle with a lower radial acceleration limit will be forced to slow down to follow the same path, thereby losing time.

A system can be described utilizing three sets of constraints: state, control, and path. The state, control, and path constraints guarantee that the vehicle is in a position that is possible, with command outputs that are possible, along a path that is optimal and inbounds. These constraints grantee that the vehicle path traversed is one that is feasible to drive and remains within the boundaries of the course.

2.3.1 State Constraints

There exist limits on all state attributes for the vehicle because of its operating limitations. It has maximum and minimum velocities, generally dictated by the size of its engine, the vehicle's weight, and the tire grip. There's no need to model the vehicle to that level of detail, so abstracting it to just a v_{min} and v_{max} is sufficient, as shown in Eq. 2.3.1. This abstraction accounts for both the vehicles maximum speed, and does not allow the vehicle's velocity to approach 0. The only constraint of the vehicle's x, and y position are the race course boundaries discussed in Section 2.3.3.

$$v_{min} \leq \sqrt{v_x^2 + v_y^2} \leq v_{max}$$

2.3.2 Control Constraints

Every vehicle has limits on its acceleration, when the gas pedal is depressed all the way, there is only so fast that the car will accelerate, based on the vehicles weight and engine torque. There are also limits on how hard and fast a vehicle can turn without losing traction, based on the vehicle's tire grip. These limits are defined by the vehicle's acceleration limit, which are defined by three values:

1. $a_{t_{max}}$: The fastest the vehicle can accelerate along a straight line.
2. $a_{t_{min}}$: The minimum(negative) acceleration when the vehicle is braking. (larger than $a_{t_{max}}$)
3. $a_{r_{max}}$: The maximum radial acceleration the vehicle can handle before the wheels begin to slip.

Vehicles are generally capable of braking faster than they can accelerate, therefore the following equation is assumed to hold:

$$a_{t_{max}} < |a_{t_{min}}| \tag{2.2}$$

The acceleration limits are based on a simple ellipse with a cut off top and are therefore defined by the following equations:

$$\frac{a_t^2}{a_{t_{min}}^2} + \frac{a_r^2}{a_{r_{max}}^2} = 1 \tag{2.3}$$

$$a_t \leq a_{t_{max}} \tag{2.4}$$

Fig. 2.2 shows a two dimensional plot of the maximum acceleration of the vehicle model. There is an area centered around 0,0 that describes the accelerations (a_r, a_t) that the vehicle can endure with no slippage or other losses. All other loses are considered noise and will be accounted for in a dynamic driver model of the vehicle. For a commanded acceleration to be possible the point, (a_r, a_t), must be inside the area marked by the blue lines. This means that for any possible acceleration point (a_r, a_t), it can mathematically be determined whether it is within the acceleration limits or not.

It was noticed in the optimal solution that the radial acceleration, a_r never settled to zero. Instead it would chatter near zero, this was due to the flat curve that is the $a_{t_{max}}$ limit. Without a direct benefit from converging to zero the solver struggled to converge, which forced the tangential acceleration to also chatter and thereby the resultant velocity of the vehicle to be smaller, as can be seen in Fig. 2.3. In order to avoid chattering within the optimal solutions acceleration, it is necessary that the top of the ellipse have a slight curve to it. This curve is defined in such a way that the vehicle has a maximum tangential acceleration only when the radial acceleration is equal to zero, therefore the vehicle is traveling in a straight line. This curve optimizes the path through the course when the vehicle is traveling straight.

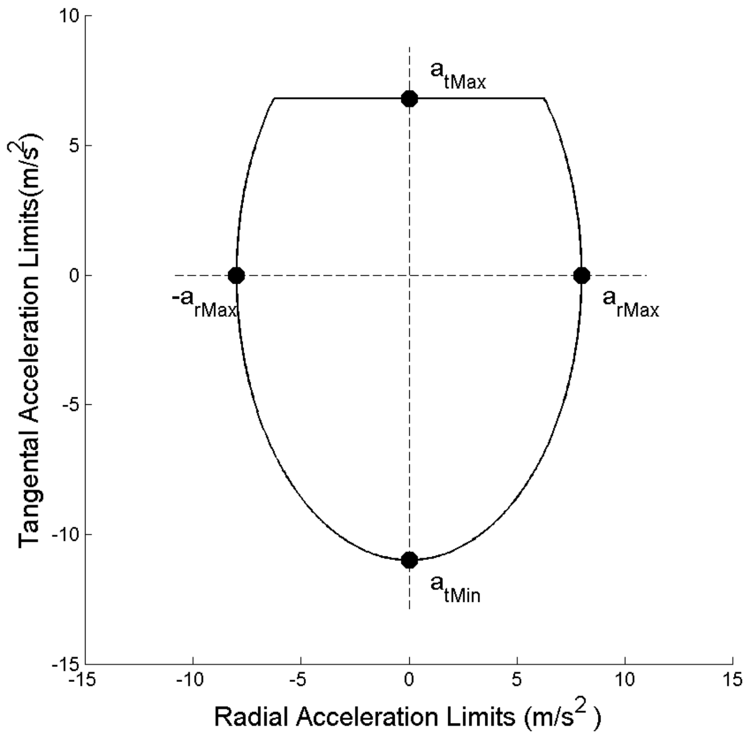

Tangential Acceleration Limits(m/s^2)
Radial Acceleration Limits (m/s^2)
a_{tMax}
$-a_{rMax}$
a_{rMax}
a_{tMin}
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15

Figure 2.2: The acceleration constraints of a vehicle with the following acceleration limits: $a_{r_{max}} = 8$, $a_{t_{min}} = 11$, and $a_{t_{max}} = 6.8$. The ellipse is formed with the two values $a_{t_{min}}$ and $a_{r_{max}}$ while the tangential acceleration, a_t is prevented from exceeding $a_{t_{max}}$.

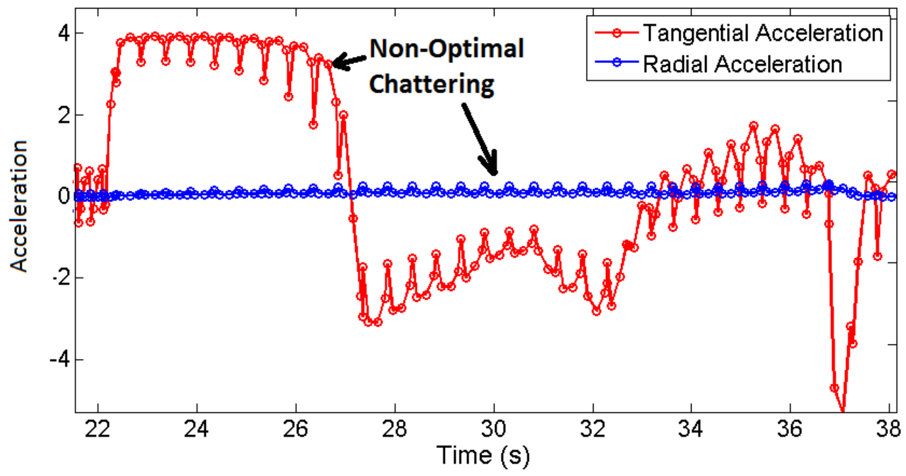


Figure 2.3: A plot of the vehicles acceleration a_t and a_r over time. The plot shows a large amount of chattering on both the tangential and radial acceleration. This chattering was seen when a flat top acceleration constraint was used because the radial acceleration was not optimized to approach 0.

A curved top was added by utilizing a second ellipse that removed the top of the first ellipse, as seen in Fig. 2.4. The intersection of the two ellipses is considered the acceptable acceleration range, while anything outside the intersection is not. A mathematical definition of these ellipses will be defined in the direct optimal control section, 3.2.2.1. This curved top gave a direct benefit to the solver to converge to zero, and thereby achieve a higher tangential acceleration.

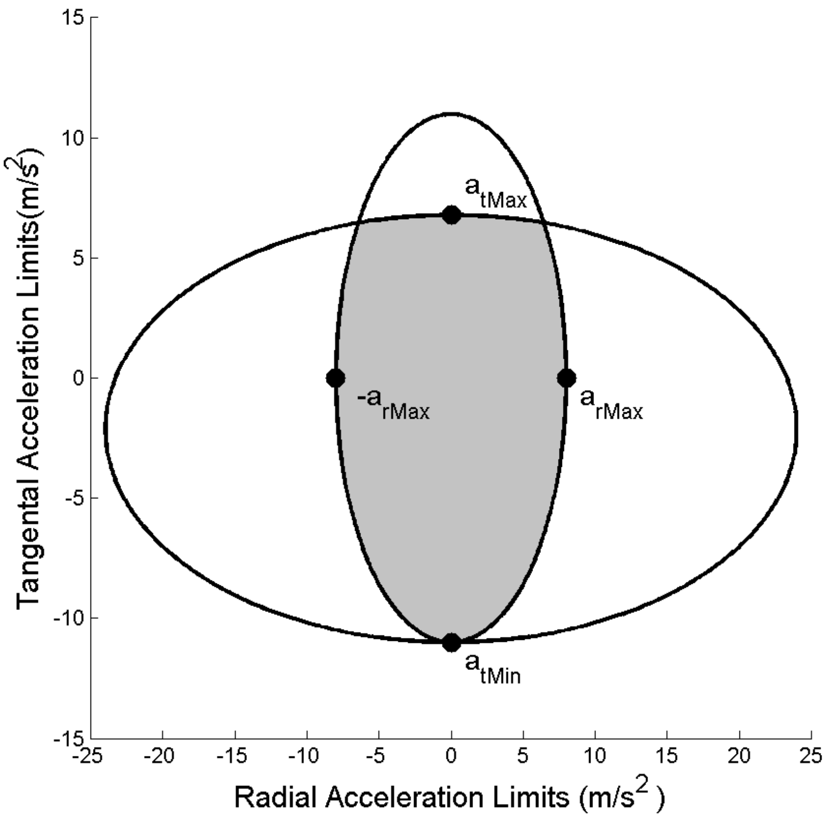


Figure 2.4: Acceleration ellipse that utilizes a second ellipse instead of a flat top. To prevent the optimal solution from chattering during straight line tracking a slight curve along the top of the cut-off ellipse is required. In the figure the two ellipses are traced, while the shaded area is the intersection of those two ellipses, or the area that is considered legal for the vehicles acceleration (a_r, a_t). The vehicles acceleration limits are labeled.

2.3.3 Path Constraints

An autocross course is defined by a set of cones that mark the borders of the course. Generally the cones exist in pairs such that the vehicles path is defined by a cone on the left and a cone on the right. By staying between these cones a complete path through the autocross course can be generated. However, autocross features certain slalom sections that do not consist of paired cones. It is common for an autocross course to feature at least one of these complex sections. A slalom section consists of multiple cones placed in a straight line. The goal of the section is that the vehicle zigzags between the cones.

The path planning algorithms use the autocross cones to define a set of gates that must be traversed in order to complete the course. In a general autocross section the gates are easily defined by the pair of cones as seen in Fig. 2.5. This process becomes more difficult in the slalom section where the cones do not have a matched pair. Instead, a theoretical cone must be added along the path in order for the vehicle to properly traverse the slalom section. The addition of this virtual cone simplifies the algorithm so that slaloms and regular sections do not need to be treated differently.

2.4 Cost Function

The winner of the autocross course is defined as the car that completes the track in the shortest time. Therefore the cost function is defined such that the final track time is minimized. When a path through the race course is found using a direct optimal control algorithm time is generally the independent variable and therefore does not need to be calculated. However, in the case that only a path through the course is planned, the final time must be calculated by integrating the time differentials along the prescribed path:

$$t_f = \int_{\Gamma(0)}^{\Gamma(1)} = \frac{ds}{v(s)} \tag{2.5}$$

This notation allows all paths to be taken into account which start at a single (x, y) start point, $\Gamma(0)$, and end at a final (x, y) position, $\Gamma(1)$. Any values between 0 and 1 define a location on the vehicles path a percentage of the way through the course.

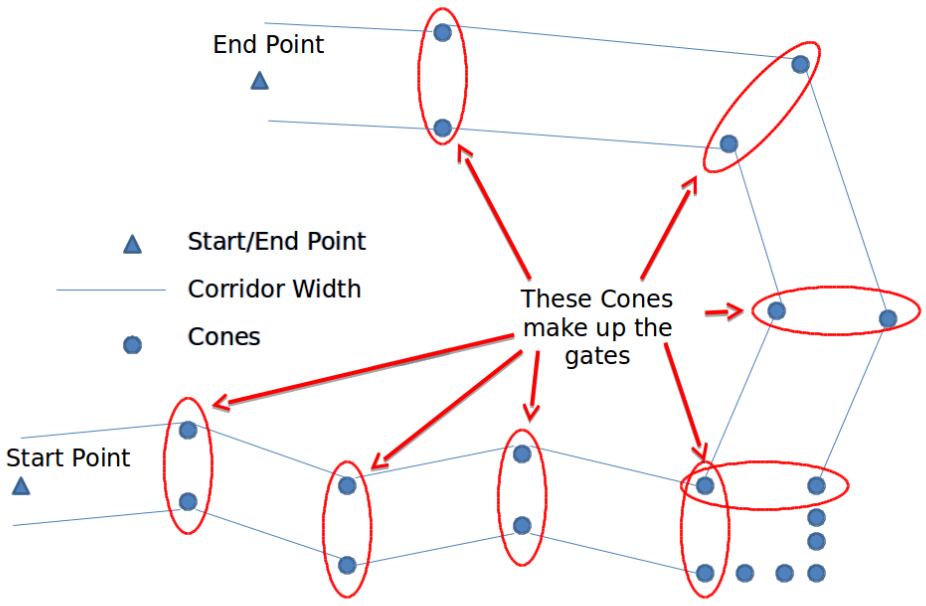


Figure 2.5: An example of an autocross course that uses cones to determine the way points, a pair of matched cones are chosen to represent the gates of the course, as shown above. The vehicle begins at the start point, and traverse through each of the gates, in order, while staying within the corridor, finishing at the end point.

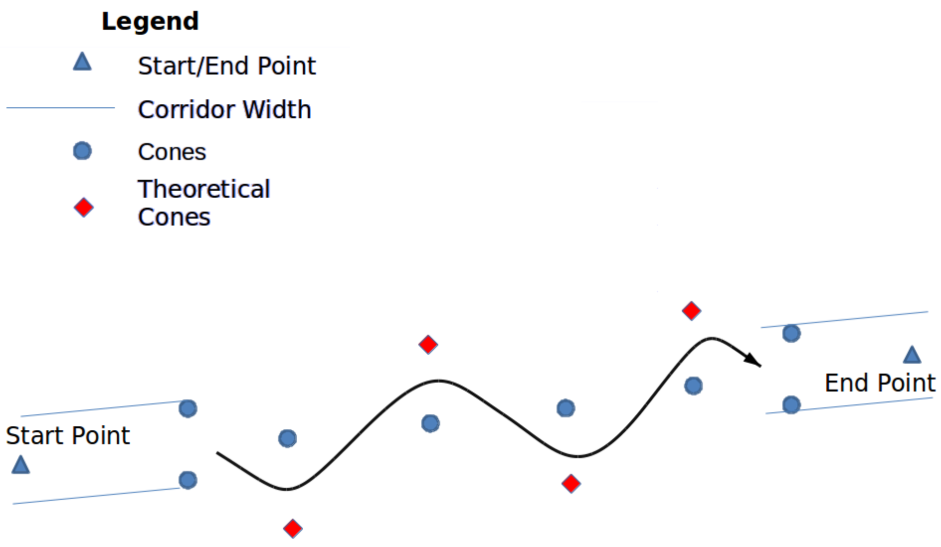


Figure 2.6: An example of a slalom segment of an autocross course. A slalom does not have a matched cone, necessitating a theoretical one be place on the path in such a way that the gates along the slalom are composed of one real and one theoretical cone. The arrow shows the proper method in which the vehicle is suppose to traverse the cones.

Based on Eq. 2.5 the distance between all points in Γ must be known, along with the velocity. To solve for the velocity at each point along the path a velocity profile must be generated which provides the maximum velocity throughout the path. [22] proposed a method for calculating the maximum velocity profile, a description of the method has been included in Appendix A for completeness.

2.5 Conclusion

The goal of this research is to plan a path through an autocross course which minimizes the track time. The track time is the final time through the course and can be calculated based on the vehicle dynamics. These dynamics explain how the vehicle moves in response to pressing the gas pedal or turning the wheel. Along with the dynamics of the vehicle each vehicle also has a set of constraints on its velocity and accelerations. The vehicle constraints are devised to model the vehicle's abilities.

By using the specific vehicle constraints, a path through the autocross course can be generated that is optimized specifically for that vehicle. These are grouped into three sets of constraints: state, control, and path. The state constraints limit the velocity of the vehicle, the control constraints limit the maximum accelerations of the vehicle, and finally the path constraints define the in-bounds and out of bonds area of the course.

In order to optimize the path through the course a cost function must be devised which minimizes the final time through the course. The path through the race course is discretized and the maximum velocity profile is generated. Using the velocity profile the time between each of the points can be calculated, and thereby the final time through the course. With the description of the autocross path planning problem done in this way, existing optimization methods can be applied to it and resultant paths can be generated and examined.

Chapter 3

Solution Approaches

3.1 Introduction

This chapter presents two approaches used to solve the optimal path planning problem. The first approach focuses on a direct optimal control solution to the state and control variables using a pseudospectral method. The second method utilizes a computationally efficient but suboptimal path planning algorithm to find a path through the course by blending the minimized maximum curvature and shortest paths.

By using these algorithms together, the degree of optimization and computational improvement can be compared. This comparison also allows the computationally efficient method to be improved to better match the direct optimal control algorithm, while still allowing the computationally efficient method to be used in an on-line system.

3.2 Direct Optimal Control - Problem Statment

The direct optimal control algorithm calculates a path through a race course such that all constraints are satified and the path is “optimal”. The optimal solution will be one that minimizes the cost function, J, in this case the final time through the course, t_f

$$\begin{aligned}
\text{min:} \quad & J = t_f \\
\text{subject:} \quad & \dot{x} = f(X,U) && \forall t \in [0, t_f] \\
& h(X,U) \leq 0 && \forall t \in [0, t_f] \\
& e(X(0), X(t_f)) = 0
\end{aligned} \tag{3.1}$$

where the vehicles state constraints $\dot{x} = f(X,U)$, the path constraints are $h(X,U) \leq 0$, and the boundary constraints are $e(X(0), X(t_f)) = 0$.

This algorithm utilizes a multiphase approach to solve the nonlinear programming problem. A multiphase solution was used to allow the solution to be broken into pieces based on the curves of the race course. This partitioning of the problem into subproblems relies on the use of gates to divide the track into segments, which can be optimally solved sequentially. This provides the same results as trying to optimize the problem in its entirety while reducing computation time.

By holding the constraints during each phase of the problem, the final assembled path is, guaranteed to fulfill the constraints. The total path constraints are defined as:

$$\begin{aligned}
& \dot{x} = f(X(t), U(t)) && \forall t \in [t_i, t_{i+1}] \\
& h_i(X(t), U(t)) \leq 0 && \forall t \in [t_i, t_{i+1}] \\
& && \forall i = 1, 2, \ldots, M
\end{aligned} \tag{3.2}$$

Each of the sections of the multiphase path begins and ends at a gate. To guarantee this a multiphase terminal condition, $k(i)$, must also be constrained to ending along the prescribed gate, as seen in Eq. 3.3. Where N_i is the final point in phase i.

$$X(t_i(N_i)) = X(t_{i+1}(1)) \quad , \quad k(i) = 0 \tag{3.3}$$

3.2.1 Multiphase Problem Set-up

In Section 2.3.3, the method for encoding an autocross course was presented. This course is composed of a set of gates as shown in Fig. 3.1. The multiphase problem uses these gates to divde the multiphase problem. The goal of the direct optimal control problem is to traverse through the set of gates, G. These gates must be traversed in the set order $G_1, G_2, \ldots, G_M$. By successfully traversing each of these gates in order, a continuous map of the course, Γ, and a set of state and control variables is determined (see Section 2.2).

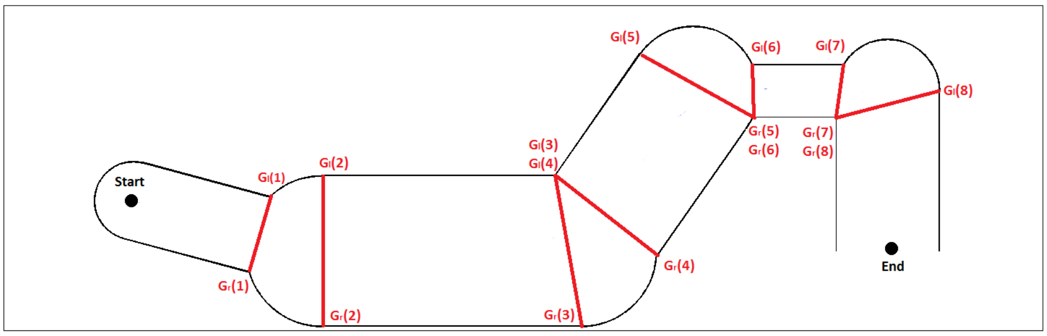


Figure 3.1: Shows a full autocross course from start to finish, with each of the gates labled by its left and right points. The line connecting the two points defines the gate G_i. A complete autocross course must pass through each of these gates in order.

$$\begin{aligned}
\Gamma(0) &= \begin{Bmatrix} x_0 \\ y_0 \\ v_{x0} \\ v_{y0} \end{Bmatrix} = z_{init} \\
\Gamma(j) &= \begin{Bmatrix} x_j \\ y_j \\ v_{x_j} \\ v_{y_j} \end{Bmatrix} \quad , \quad j = \frac{1}{M*N}, \frac{2}{M*N}, \ldots, \frac{M*N}{M*N}
\end{aligned} \tag{3.4}$$

where z_{init} is defined as the start point and inital velocity, M is the number of gates and N is the number of points within each section.

The gate is defined by two sets of (x, y) points $G_l(i)$ and $G_r(i)$, where G_l is the part of the gate which will be on the left side of the vehicle when passed and G_r is on the right side. The whole course can be defined by a start point z_{init}, and a set of gates as shown:

$$
\begin{aligned}
G_l &= \begin{pmatrix} x_{l_1} & y_{l_1} \\ x_{l_2} & y_{l_2} \\ \vdots & \vdots \\ x_M & y_{l_1} \end{pmatrix} \\
G_r &= \begin{pmatrix} x_{r_1} & y_{r_1} \\ x_{r_2} & y_{r_2} \\ \vdots & \vdots \\ x_{r_M} & y_{r_M} \end{pmatrix}
\end{aligned}
\tag{3.5}
$$

Each phase is calculated between two gates. This means that during each phase i the vehicle's position P must end along the gate G_{i+1}. To guarantee this, vectors are used to calculate the distance that point P is away from the gate (see Fig. 3.2). Based on these definitions the terminal phase constraint, $k(i)$, can be calculated using Eq. 3.6. The distance from G_l to P plus the distance from G_r to P must equal the distance from G_l to G_r, this forces the resultant point to be on the line between the gate.

$$k(i) = |G_l(i) - P(i)| + |G_r(i) - P(i)| - |G_r(i) - G_l(i)| \tag{3.6}$$

3.2.2 Continuous Constraints

This section presents the different continuous constraints to the direct optimal control problem. In order to separate the vehicle model from the constraints, the Eq. 3.1 are divided into two parts, $f(X, U)$ and $h(X, U)$. The vehicle dynamics, $\dot{x} = f(X, U)$, was addressed in Section 2.2. The vehicle and path constraints, $h_i(X, U))$, are addressed in the following sections. These constraints must be satisfied at all points along the course.

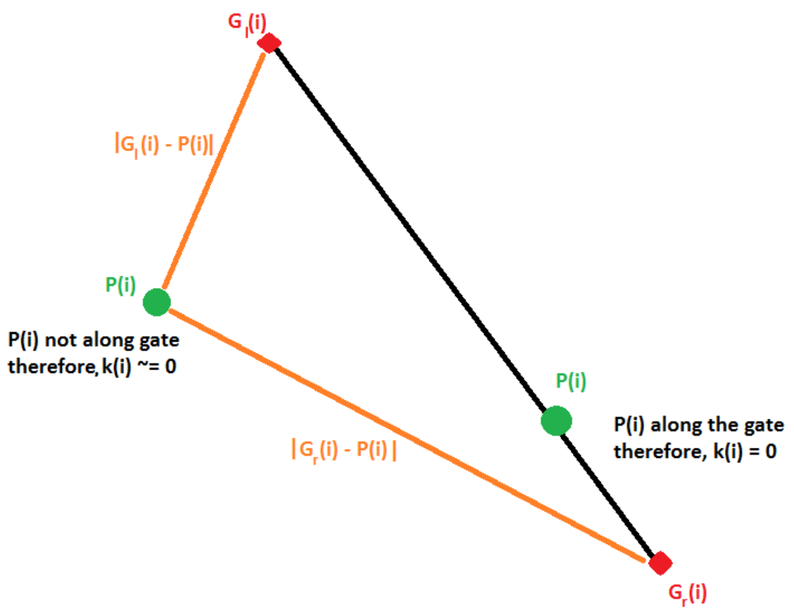

$G_l(i)$
$|G_l(i) - P(i)|$
P(i)
P(i) not along gate therefore, k(i) ~= 0
P(i)
P(i) along the gate therefore, k(i) = 0
$|G_r(i) - P(i)|$
$G_r(i)$

Figure 3.2: A figure that shows each of the vectors when calculating the terminal condition of the multiphase problem. Where G_l and g_r are the left and right points of the gate, and P is a theoretical point of the vehicle.

These constraints were required due to the simplified point mass vehicle model used. As stated in Section 1.3 the point mass model has points of non linearity when the velocity approaches zero. This means limits must be placed on the vehicles velocity, along with the direction of travel to guarantee that the vehicle is always traveling towards the next gate.

3.2.2.1 Vehicle Constraints

Constraints are placed on the vehicle's velocity and acceleration in order to simulate realistic tire loading and grip limits. As explained in Section 2.3 these constraints are used to limit the maximum velocity and acceleration of the vehicle.

As shown in Eq. 2.3.1, the velocity is constrained by the minimum and maximum velocities: v_{min}, v_{max}. These constraints are stated below conforming to the standard direct optimal control form in Eq. 3.1:

$$\begin{aligned} V(j) - v_{max} &\leq 0 \\ -V(j) + v_{min} &\leq 0 \end{aligned} \tag{3.7}$$

Where the vehicle velocity is calculated as:

$$V(j) = \sqrt{v_x(j)^2 + v_y(j)^2} \tag{3.8}$$

The limits placed on the vehicle's radial and tangential acceleration (a_r, a_t) mimic an ellipse with a cut off top, (see Fig. 3.3). This is accomplished by utilizing two ellipses and confirming that the vehicles acceleration is within the intersection of the two ellipses.

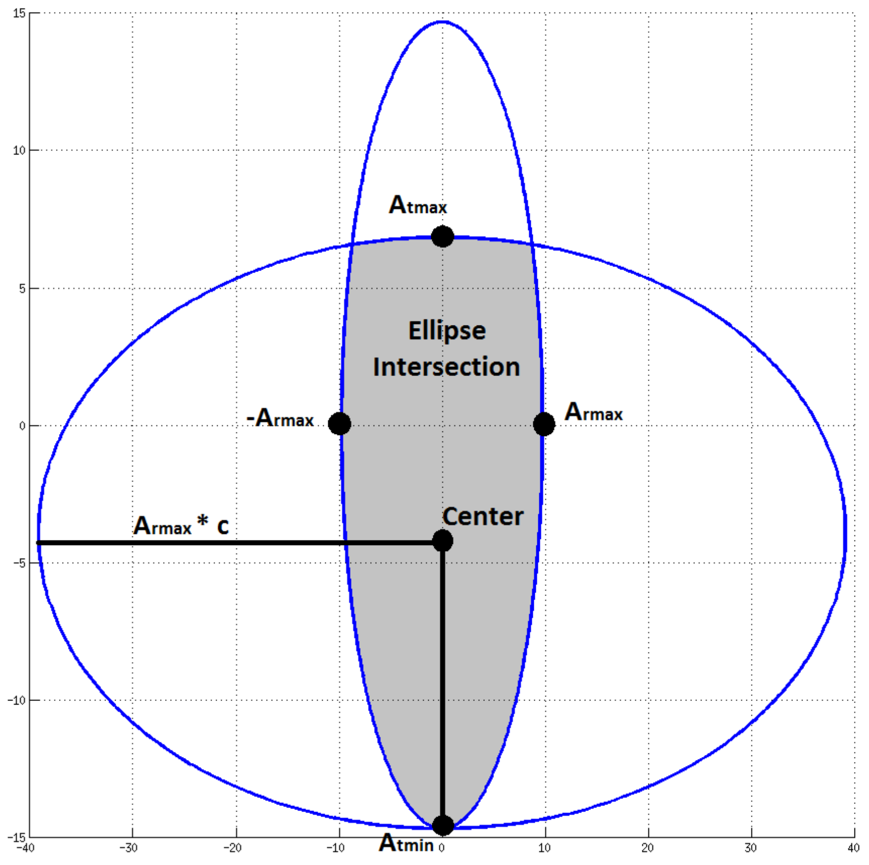


Figure 3.3: This figure shows two ellipses used to limit the acceleration of the vehicle. The shaded area represents a legal vehcile acceleration. The ellipses are calculating using a_{tmax},a_{tmin},a_{rmax} and c, as shown in Eq. 3.9.

The two ellipses can be represented in the form $h(X, U) \leq 0$ as shown below:

$$\begin{aligned} \left(\frac{a_t}{a_{tmin}}\right)^2 + \left(\frac{a_r}{a_{rmax}}\right)^2 - 1 \leq &\quad 0 \\ \left(\frac{a_t - center}{diameter}\right)^2 + \left(\frac{a_r}{a_{rmax} * c}\right)^2 - 1 \leq &\quad 0 \end{aligned} \tag{3.9}$$

Where c is a scaler factor which determines the flatness of the top of the intersection. Diameter and center can be found using the following equations:

$$\begin{aligned} diameter &= \frac{a_{tmax} + a_{tmin}}{2} \\ center &= a_{tmax} - diameter \end{aligned} \tag{3.10}$$

3.2.2.2 Path Constraints

To produce a viable solution it is important to confirm that the vehicle is continually traveling towards the next gate. This can also be thought of as ensuring that the vehicle's velocity vector should always be pointing in the direction of the next gate G_{i+1}, as seen in Fig. 3.4.

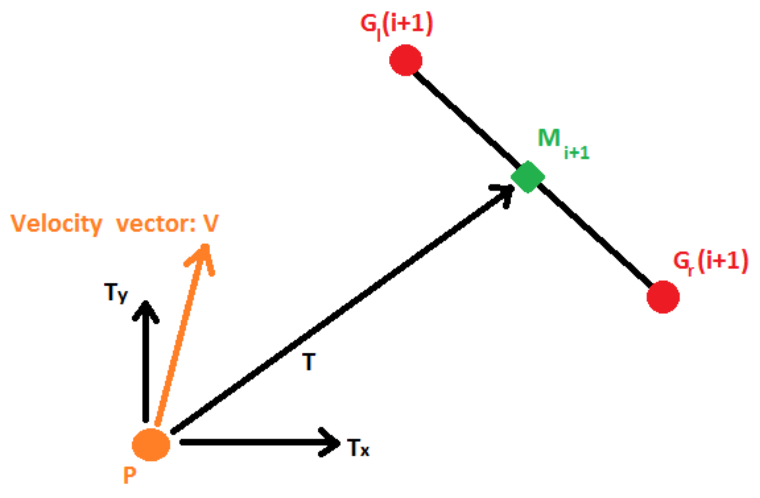


Figure 3.4: This figure shows how the vector T is used to guarantee that the vehicle is always moving towards the gate. T is pointed from the vehicles position, P to the midpoint of the gate G_{i+1}.

To guarantee that the velocity vector V is pointed in the same direction as the gate, G_{i+1} a vector, $\overrightarrow{T}$, is used. $\overrightarrow{T}$ points from the vehicles position, P, to the midpoint of the next gate, M_{i+1}. By constraining the sign of the velocity vector and $\overrightarrow{T}$ to be postive, the vehicle can be forced to always point towards the next gate:

$$M_{i+1} = \frac{G_l(i+1) + G_r(i+1)}{2} \tag{3.11}$$

$$T = M_{i+1} - P \tag{3.12}$$

$$move_{forward}(v_x, v_y, T_x, T_y) = v_x * T_x + v_y * T_y \tag{3.13}$$

this means that the constraint equation is defined in such a way that the following must be true:

$$\forall P_j \quad j = 1, 2, \ldots, m \quad , \quad -1 * move_{forward}(v_x(j), v_y(j), T_x, T_y) \leq 0 \tag{3.14}$$

The vehicle and path constraints, $h(X, U)$ guarantee that the final optimal path is feasible. Without the acceleration constraints a point mass model of the vehicle would attempt to turn too sharply. The acceleration limits guarantee that the path is feasible, while preventing the vehicle from chattering along a straight path. Finally the limits on the direction of the vehicles velocity vector guarantee that the vehicle is always moving towards the next gate.

3.2.3 Receding Horizon

Like most nonlinear programming problems as the number of nodes increases it takes exponentially longer to compute an accurate solution to the problem. For this problem, the computational difficulty rises as the number of gates do, quickly becoming intractable. For even the smallest of real-world race tracks (gate count 25+) the problem is too large. Therefore, for this algorithm to be usable in any real-world scenario, it must be independent of the total number of gates in the track. In order to achieve gate indpendence the algorithm utilizes receding horizon control. Receding horizon control has been shown to be applicable in large nonlinear programming problems [21, 28]. Receding horizon control allows the current gate to be optimized by only taking into account a limited number of subsequent gates.

This is particularly beneficial when calculating tracks with many gates (greater than 10) as it greatly increases the speed of computation. Looking further ahead in order to encompass more gates increases the accuracy of the solution while simultaneously increasing the computation time [26]. This research utilizes a horizon of seven to calculate large tracks because it provided an appropriate trade-off between run-time and performance.

3.3 Blending of Two Paths

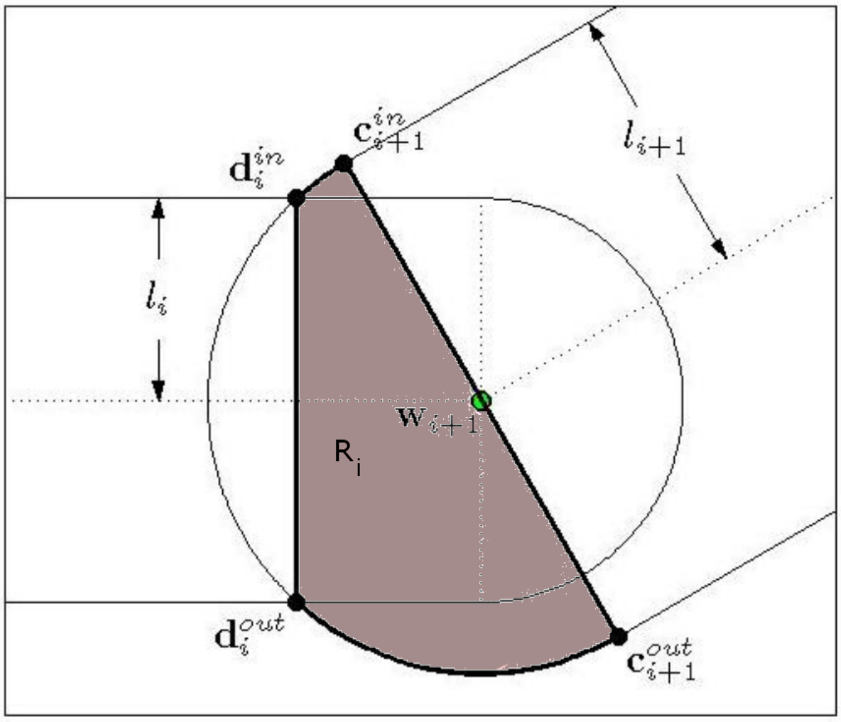
The work contained in this section is based on previous work by Choi and Fiebich [6, 14]. This thesis further improves upon the path planning algorithm originally derived by Choi [6]. The algorithm is based in a cell decomposition method. By dividing the course into corner and straight cells a primitive straight line path is able to be very efficiently planned through the course. This path is then smoothed using forth order Bézier curves that maintain full continuity along the course.

The algorithm is upgraded to handle gates placed along the straight segments, allowing for a wider deviation from the center line, corresponding closely to the optimal path. The algorithm additionally blends two different paths: the minimized maximum curvature and the shortest path. By blending these two paths a faster time through the course is found.

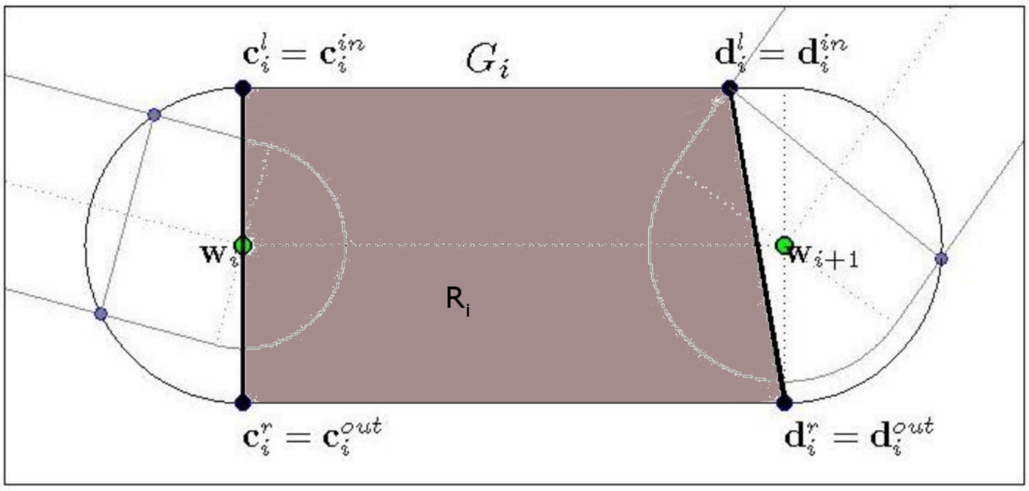
3.3.1 Describing the Course

A race course is described as a set of way points $\boldsymbol{W}= w_1, w_2, \dots, w_N \in \mathbb{R}^2$ and the corridor widths between two points $\boldsymbol{L}= l_1, l_2, \dots l_{N-1} \in \mathbb{R}^+$. An inbounds area,

$\boldsymbol{S}$, is made up of route segments $\boldsymbol{R}= R_1, R_2, \ldots, R_{N-1}$. The route segments are grouped into two categories, *corner cells* and *straight cells*. Straight cells are the straight-path regions between two way points. Corner cells are the regions between the straight cells. Both of these cell types are shown in Fig. 3.5.



(a)



$\mathbf{c}_i^l = \mathbf{c}_i^{in}$
G_i
$\mathbf{d}_i^l = \mathbf{d}_i^{in}$
$\mathbf{w}_i$
$\mathbf{w}_{i+1}$
R_i
$\mathbf{c}_i^r = \mathbf{c}_i^{out}$
$\mathbf{d}_i^r = \mathbf{d}_i^{out}$

(b)

Figure 3.5: The geometry of a (a) corner cell, (b) straight cell. R_i represents each cell of the course, with the barricades of the cell being based on c and d.

While each cell has a precise definition in Section 5.1 of Choi [6], an intuitive description of a cell is as follows:

1. Draw a circle of radius l_{i-1} and l_i around each point w_i, $i = 2, 3....N-1$.
2. Connect each point by a corridor of width $2 * l_i$
3. The two points where the corridor and the large circle intersect define a barricade. See points c and d in Fig. 3.5.
4. These barricades determine the different types of cells: The straight sections between the circles are the straight cells, while the cells that include the circles are the corner cells.

3.3.2 Generating Primitive Path Points

To maintain the computational efficiency of the algorithm, a set of primitive path points are generated in each corner cell. These points are equally spaced along multiple lines within the corner cell, as can be seen in Fig. 3.6. Section 7.2 of Choi [6], references specific equations for spreading the points equally along the corner cell. Within the algorithm the number of points and lines vary based on the size of the corner cell. Fig. 3.6 utilizes 4 lines with a varying number of points 5 to 3, depending on the length of the corner cell. In this way there are never too many or too few point along a line, or within a corner cell.

3.3.3 Optimal Path by Dynamic Programming

Using a dynamic programming (DP) algorithm a primitive path point from each corner cell is selected such that the entire path minimizes the cost function, J. The cost function is based on three weighting parameters: c_l, c_c, and c_k. Where these three weighting parameters are all normalized such that:

$$c_l + c_c + c_k = 1 \tag{3.15}$$

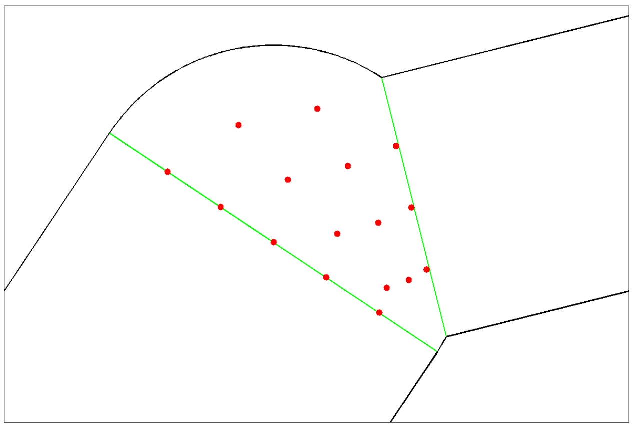


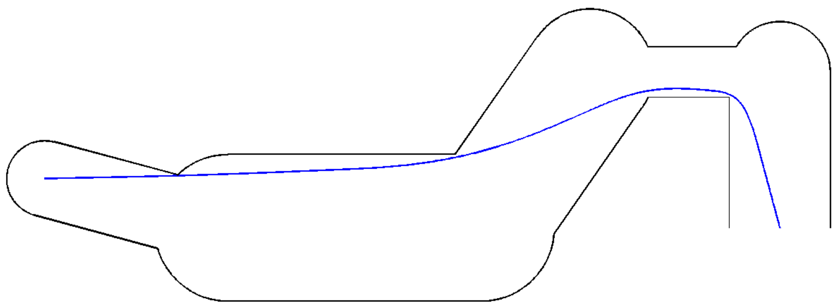
Figure 3.6: This figure shows a set of primitive path points for a single corner cell i. Each of the corner cells contains a certain number primitive path points based on the size of the cell.

The cost function J is calculated based on these weighting parameters, and the specific calculations for L_i,C_i and K_i are described below, while the mathematical equations can be found in Section 7.4 of Choi [6].

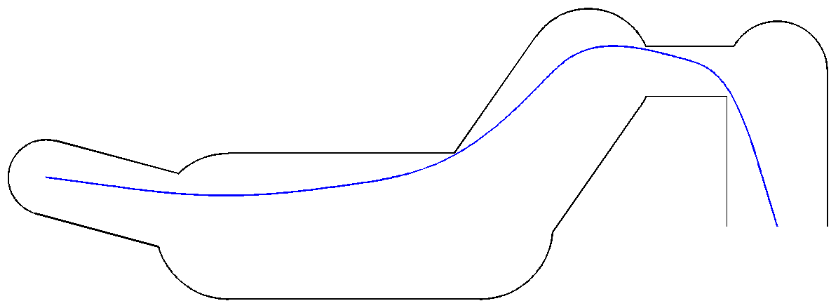
$$J(i) = L_i * c_l + C_i * c_c + K_i * c_k \tag{3.16}$$

L_i, C_i, and K_i each represent a different factor that can be minimized. L_i is the length of the primitive path segment, which can be calculated using the distance between the two primitive path points. C_i is the clearance from the edges of the straight cell. It can be calculated by determining the distance from the corridors and the primitive path points. K_i is the curvature of the path which is determined by using three primitive path points to determine the maximum curvature of a three point Bézier curve [8]. Fig. 3.7 shows three different solutions to the same course based on different weighting parameters.

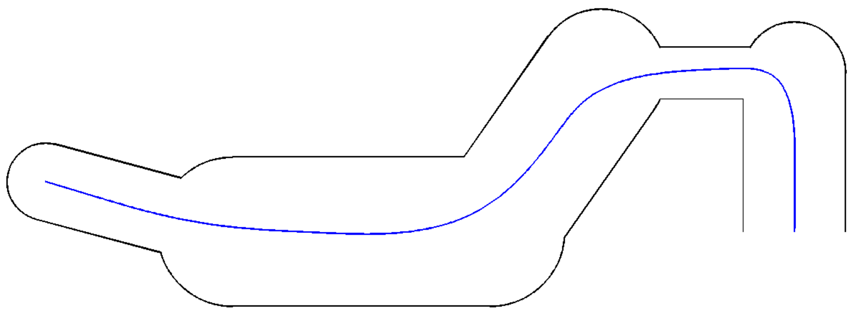
The following is an example of how the dynamic programming algorithm is used to solve for the minimum curvature path. Given a course with two corner cells that serve as a start and end point, the optimal set of primitive path points through the course can be calculated based on the cost function, J.



(a) $c_l = 1 \quad c_k = 0 \quad c_c = 0$



(b) $c_l = 0 \quad c_k = 1 \quad c_c = 0$



(c) $c_l = 0 \quad c_k = 0 \quad c_c = 1$

Figure 3.7: This figure displays three distinct solutions to a course. a) minimize the length of the race line, b) minimize the maximum curvature of the race line, and c) minimize the distance from the center line.

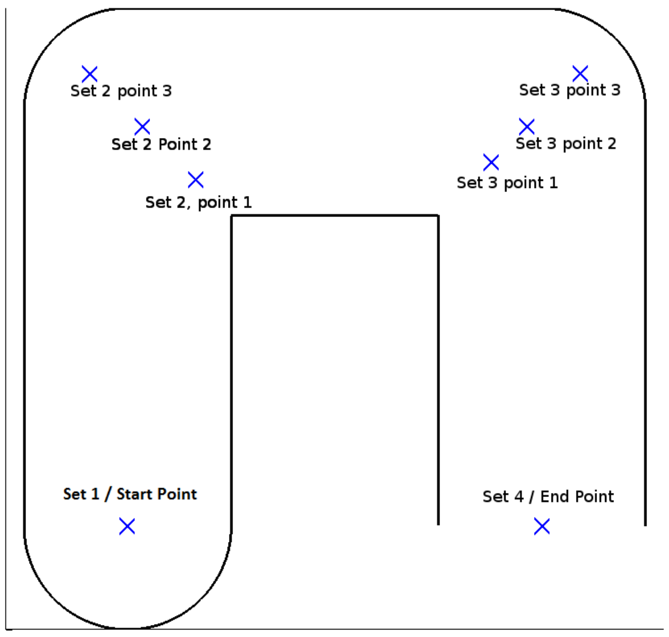


Figure 3.8: A short course with a start point two corner cells and an ending point. Show a limited number of primitive path points at each of the corners of the course.

The dynamic programming algorithm begins by calculating the cost function (penalty) from each of the points in set 3 to the end point (set 4). This penalty is assigned to each of the nodes in set 3 and is used to determine the optimal path.

Primitive Path Points	Curvature Cost
$3_1 \rightarrow$ End	0
$3_2 \rightarrow$ End	0
$3_3 \rightarrow$ End	0

Table 3.1: The curvature cost of traveling from a point in set 3 to the final location(set 4), reference Fig. 3.8. Note that the above calculation from set 3 to point 4 is a straight line and therefore the curvature is 0.

The algorithm now calculates the curvature from all points in set 2 and 3, ending at 4th set. The table below contains hypothetical data for the DP algorithm.

Primitive Path Points	Curvature Cost	Previous Cost
$2_1 \rightarrow 3_1 \rightarrow$ End	10	0
$2_1 \rightarrow 3_2 \rightarrow$ End	15	0
$2_1 \rightarrow 3_3 \rightarrow$ End	20	0
$2_2 \rightarrow 3_1 \rightarrow$ End	11	0
$2_2 \rightarrow 3_2 \rightarrow$ End	10	0
$2_2 \rightarrow 3_3 \rightarrow$ End	12	0
$2_3 \rightarrow 3_1 \rightarrow$ End	3	0
$2_3 \rightarrow 3_2 \rightarrow$ End	7	0
$2_3 \rightarrow 3_3 \rightarrow$ End	10	0

Table 3.2: The cost, J, of the the vehicle to travel from any point in set 2, to any point in set 3, ending at the 4th set. Note that for each start point 2_1,2_2, and 2_3 the minimum curvature path is highlighted.

As can be seen from the table, each of the points in set 2 is compared to all points in set 3. Based on the cost function, the minimum point in set 3 is chosen with respect to each point in set 2. These points have been highlighted for clarity.

The DP algorithm concludes by calculating the cost function as determined by the start point. Each of the points in set 2 is compared, along with the optimal point in gate 3.

Primitive Path Points	Curvature Cost	Previous Cost	Total Cost
Start $\rightarrow 2_1 \rightarrow 3_1$	9	10	19
Start $\rightarrow 2_2 \rightarrow 3_2$	10	10	20
Start $\rightarrow 2_3 \rightarrow 3_1$	15	3	18

Table 3.3: The cost function, from the start point through each of the points in set 2, ending at the point in set 3 that minimizes the cost function for each point in set 2, reference Table 3.2. The final minimum path through the 3 points is highlighted.

In the end, the path which minimizes the total cost function is chosen. In this case that path is $Start \to 2_3 \to 3_1 \to End$.

3.3.4 Improvements to Computationally Efficient Method

This method for generating a path through a race course by placing a gate within the corner cells works efficiently but can be further improved by placing a gate along certain straight segments, the overall track time can be improved by allowing the vehicle to approach each turn with lower curvature, as shown in Fig. 3.9. The previous dynamic programming algorithm was unable to handle this situation because if any three gates were in a straight line, the minimized maximum curvature path would also be a straight line. However, this would increase the instantaneous curvature along another part of the curve.

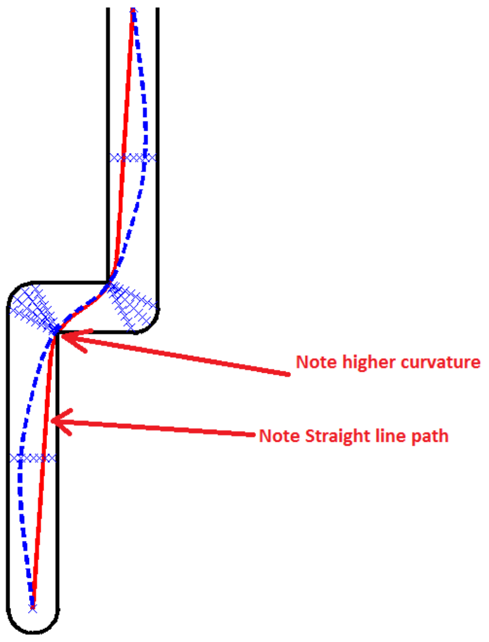


Figure 3.9: An example course with two corner cells,demonstrating how the new path selection improvements (blue dashed) compares to the original algorithm (red). While the 'X 's signify the location of gates along the course.

The new dynamic programming algorithm is able to detect when a gate has been placed along a straight segment. If this is the case the algorithm calculates the cost function through the current set of primitive path point. This cost will be near 0 when minimizing the maximum curvature, because the paths chosen will be straight lines. Since this analysis provides no new information, an optimal path through the gates is not chosen. Instead a set of 4 gates is examined. These gates should now form a curve such that the algorithm can properly calculate a minimized maximum curvature path, which will not be a straight line.

3.3.5 Minimizing the Maximum Curvature of a Bézier Curve

Let the primitive path be represented by $\mathscr{G}$ where $\mathscr{G} = (s, r_2, r_4 \ldots, r_{M-1}, t)$, and s and t are the start and finish points respectively. This section reviews a method from Choi for smoothing the straight line primitive path [6].

Choi proposed minimizing the curvature of a Bézier curve by changing the distance (not the angle) of a 2nd-order Bézier curves control points, while still remaining inside the point p, as seen in Fig. 3.10 [8]. The equation for a quadratic Bézier curve is given in Eq. 3.17.

$$Q(\lambda) = (1-\lambda)^2 q_0 + 2\lambda(1-\lambda)q_1 + \lambda^2 q_2 \tag{3.17}$$

Where q_0, q_1, and q_2 are the control points of the Bézier curve, with the middle point being placed at one of the primitive path points along a gate. A coordinate frame transformation simplifies the problem where q_1 is at the point (0,0), the line q_1q_0 is along the x-axis, and q_1q_2 is pointed in the positive y direction. With respect to this coordinate frame, the control points can be written as:

$$\begin{aligned} q_0 &= (\alpha_3, 0) \\ q_1 &= (0, 0) \\ q_2 &= (-\beta_3 \cos\theta, \beta_3 \sin\theta) \end{aligned} \tag{3.18}$$

Where α_3 represents the distance from the first to second control point and β_3 is the distance from the second to third control point, see Fig. 3.10.

Fiebich adapted the 3-point Bézier curve into a 5-point Bézier curve in order

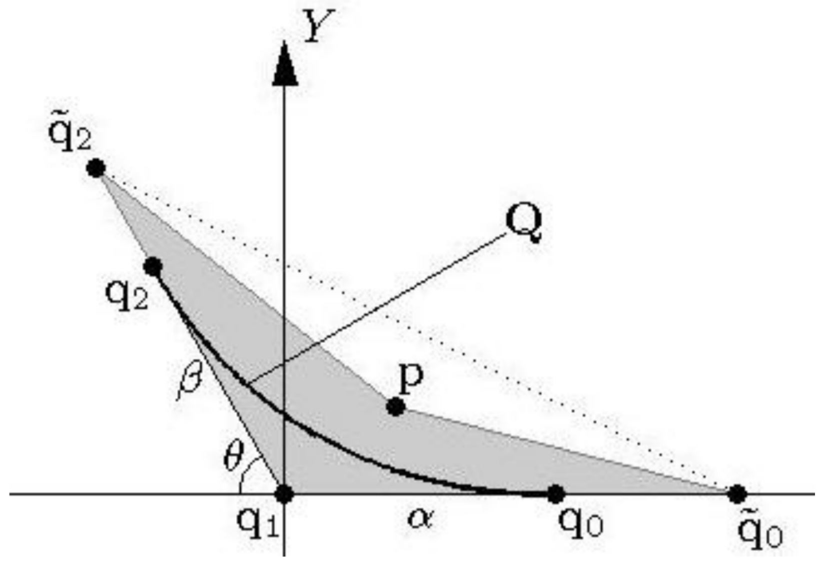


Figure 3.10: A reproduction of Fig. 6.3 from Choi [6]. It is a visual representation of the coordinate frame used to solve for α and β given a 3-point Bézier curve. α and β are chosen such that they minimize the curvature of the Bézier curve while still remaining inside the corner of the course p.

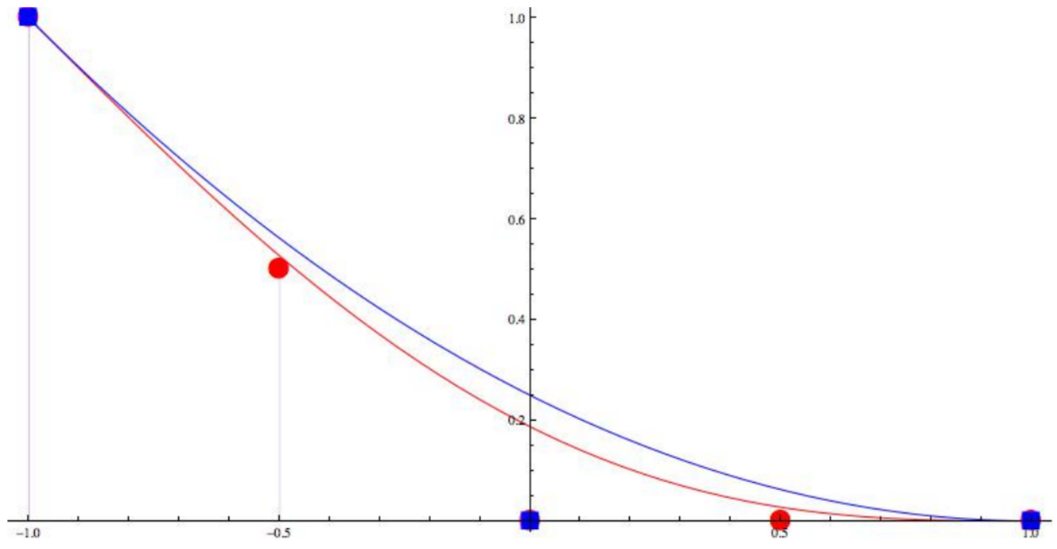
to force the curvature at the end points to zero [14]. This modification allows multiple Bézier curves to be placed end-to-end without abrupt changes in the velocity or acceleration, making the joining G2 continuous. In order to drive the curvature κ to zero at the end points, two new points should be place at the mid-points of α and β.

In Fig. 3.11 it can be seen that by keeping the same α and β values of the 3 point Bézier curve the Bézier curves no longer match well, and the curvature is no longer minimized. This can be resolved by setting the α_5 and β_5 terms as shown in Eq. 3.19. The 4th order curves can be made to match the second-order Bézier curve [14].

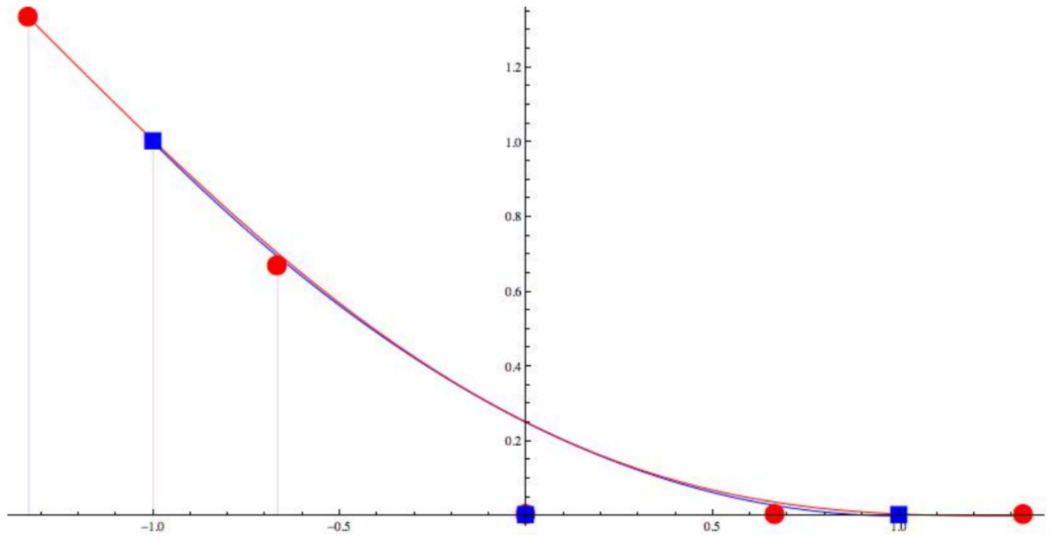
$$\begin{aligned} \beta_5 &= \frac{4\beta_3}{3} \\ \alpha_5 &= \frac{4\alpha_3}{3} \end{aligned} \tag{3.19}$$

The difference between the two sets of curves is minimal as can be seen in

Fig. 3.11.



(a)



(b)

Figure 3.11: A reproduction of figure 5.2 and 5.5 from [14]. A 3-point Bézier curve over laid on a 5-point Bézier. The red circles represent the 5 control points of the fourth-order Bézier curve, while the blue squares represent the 3 control points of the second-order Bézier curve. (a) Utilizes the same locations for α_3, β_3 and α_5, β_5. (b) Places α_5 and β_5 based on Eq. 3.19. Notice how the Bézier curves match in (b) closer than in (a).

3.3.6 Improvements to Path

The existing planner did not explicitly handle vehicle dynamics; in an effort to improve performance the vehicle acceleration limits were embedded directly into the algorithm in order to generate an improved constrained path. In [3] and [2] a blended pseudo-optimal path (POP) was developed. This blended path has been proven to provide a shorter race time and is bounded by the minimized maximum curvature(MMC) and shortest path (SP), as follows:

$$POP = (1 - \epsilon) * MMC + \epsilon * SP \qquad \epsilon = [0, 1] \tag{3.20}$$

Braghin assumed that there was a single ϵ value that could optimize the entire track and, therefore, provide a better race time than either the MMC or shortest paths alone [2]. Cardamone refined this by choosing multiple ϵ values for different subsections of the race course, specifically for those subsections where the MMC and shortest paths intersected (known as knot points) [3]. The increased dimensionality of multiple ϵ's allows better performance by breaking the problem into multiple optimizations. This comes at increased computational difficulty when matching continuity at the knot points. The resultant solutions provided better performance by following closer to the shortest path in straight sections and closer to the MMC path during high velocity sections.

This solution improved track times around many tracks. However at the inflection points where two different ϵ values met the vehicle was forced to make an instantaneous change in the direction of travel (see Fig. 3.12). In order to account for this discontinuity an analytical driver model is used to smooth the discontinous path at run time. This method is computationally inefficient and removes the solution further from the true optimal.

The main contribution of this thesis is a method for blending the MMC and shortest paths while retaining continuity in the higher derivatives in the final POP

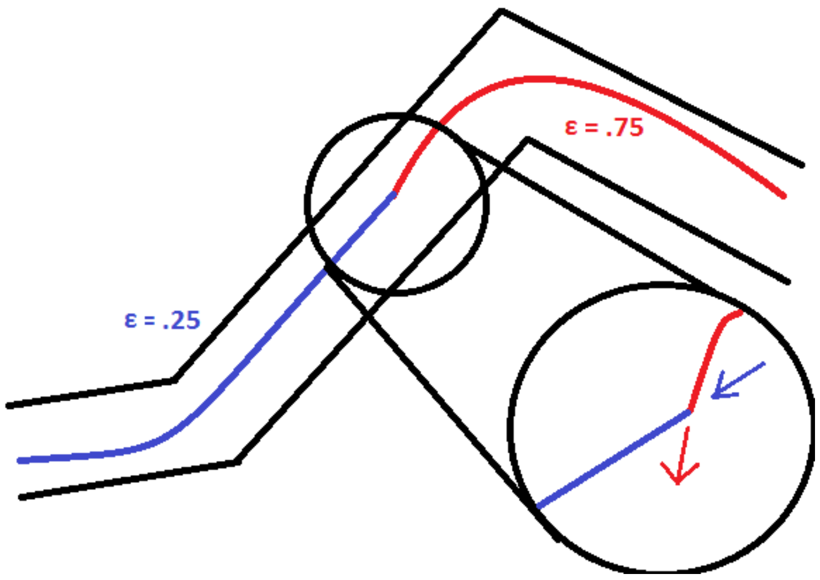


Figure 3.12: This figure shows the how the path is discontinuous at the inflection point between two different ϵ paths. At these points the vehcile is required to make an instanteous change in angle, which is not possible.

solution. In short, this is accomplished by blending the primitive straight line paths of the MMC and shortest paths, then smoothing the resultant straight line path, calculating track time, and using Bézier curves while constraining the control points to guarantee G^2 continuity. This allows for better performance than a single ϵ blending value, as shown in [3].

3.3.7 Setup of Parameter Optimization Problem

As shown in Fig. 3.13, the path is assembled from straight line segments connecting sampled points within *corner cells*, which are then smoothed to construct the final path. This formulation has advantages in terms of computational efficiency. That is, each sampled point within the corner cells has a cost that can be quickly reevaluated based on the cost of the entire path.

The improved algorithm blends the two selected points within the corner cell (corresponding to the MMC and shortest path points) using a single parameter ϵ for each pair of points in each corner cell. The ϵ moves the point along the line

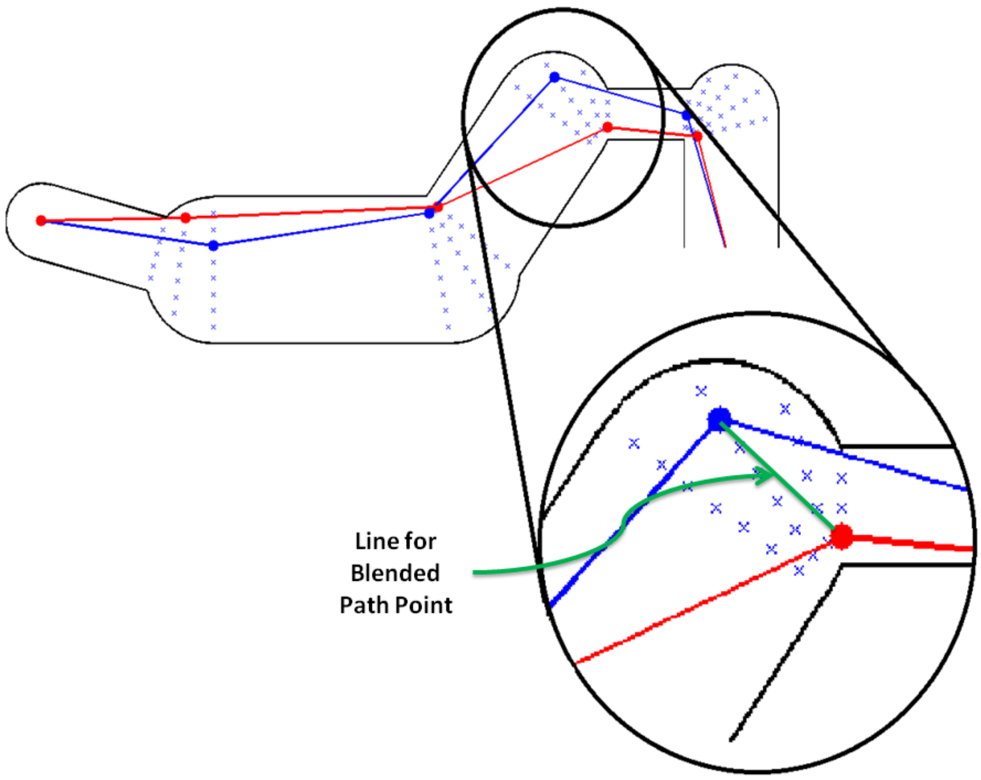


Figure 3.13: This figure shows the primitive path points of both the MMC and shortest paths. A parameter search is performed to find a point along the blended line that minimizes the time through the total path.

connecting the MMC and shortest path's primitive path points. While it is unlikely that the optimal point will lay on the line connecting these two points, the computational improvement gained from making this assumption is worth the trade-off.

To construct the entire optimized path trajectory a set of $n - 2$ ϵ blending values are chosen to mix the MMC and shortest paths, where n is the total number of primitive path points, using Eq. 3.20; the first and last points of the primitive path do not require any blending as they are the same for both the MMC and shortest path algorithms.

After the blended straight line primitive path is determined for a specific ϵ, the algorithm then smooths this path using Bézier curves so that the course is G^2 continuous. A cost function for the complete smooth path must be constructed, from the set of optimized points which includes the maximum velocity obtainable along the path. See section A for details of the construction of the complete continuous path cost function.

Based on this information, a parameter optimization algorithm can be utilized to solve for the best set of ϵ values that will minimize the time through the course while at the same time maintain total longitudinal and lateral acceleration within their respective constraints. Many different algorithms can be used to solve for optimal parameter points ϵ_k, including genetic algorithms, Sequential Quadratic Programming(SQP), and Particle Swarm Optimization(PSO). This research focuses on SQP and PSO to solve for a set of ϵ_k parameters that minimize the cost function.

3.3.7.1 Sequential Quadratic Programming

The sequential quadratic programming (SQP) method was first developed by Wilson [33]. Since its development, the SQP method has proven to be one of the most successful methods for finding a numerical optimal solution to a constrained nonlinear optimization problem. Consider a nonlinear programming (NLP) problem of the form:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & h(x) = 0 \\ & g(x) \leq 0 \end{aligned} \tag{3.21}$$

Where f is the objective function $f : \mathbb{R}^n \to \mathbb{R}$, h and g are the constraint functions where $h : \mathbb{R}^n \to \mathbb{R}^m$, and $g : \mathbb{R}^n \to \mathbb{R}^p$.

The SQP method models the NLP problem as an approximate solution, x^k, using a quadratic programming subproblem. A better approximation, x^{k+1}, is developed using the solution to this subproblem. By repeating this process, a sequence of approximations are found that will converge to the solution, x^*. Often the SQP method can be viewed as an extension of Newton's method.

This iterative approach means the SQP method shares characteristics with Newton's methods, with rapid convergence when the approximations are close to the solution, but possibly erratic behavior when the approximations are far off. However, the SQP method calculates x^{k+1} differently from Newton's method due to the quadratic subproblem that must be solved. The subproblem is found in such a way that the local properties of the original problem remain. A quadratic subproblem is used because the problems are relatively easy to solve and yet their results reflect nonlinearities in the original problem. The details for solving the quadratic subproblem will not be dealt with here, although the resolution of the solver will affect the performance of the SQP algorithm [1].

Though at first lace SQP appears as a direct replacement for Newton's method it has some caveats that limit its applications:

1. SQP does not guarantee that any individual iteration is feasible, including the initial point, x^k.
2. SQP relies heavily on the accuracy of the solution of the quadratic subproblem.

For a more in-depth overview of SQP methods, see [1]. This thesis utilizes Matlab's fmincon() function for its SQP algorithm implementation.

3.3.7.2 Particle Swarm Optimization

A direct optimal control method such as SQP is advantageous in a number of situations. However in many situations a multiple shooting method can be preferable. This research compared a multiple shooting method, Particle Swarm Optimization (PSO), to the well-known optimization algorithm SQP to see which would be advantageous under the specific constraints given in the race track optimization problem.

PSO has become well known for its use on large scale optimization problems. PSO differs from many multiple shooting method because each particles interacts and shares information on where the most optimal positions are [12, 27]. A number of particles are spread throughout the search space of the function. Each of the particles is composed of three $N\epsilon$-dimensional vectors (where $N\epsilon$ is the dimensionality of the search space). The vectors are: the current position, $\vec{x}_i$, the previous best position, $\vec{p}_i$, and the velocity, $\vec{v}_i$.

The objective function is evaluated at each of the particle's current positions. Each particle then determines its next position/velocity by combining aspects of the particles history, its current best-fitness location, and the positions of other particles. The next iteration takes place after all particles have been moved. Eventually the particles swarm to an optimum of the fitness function.

This iterative approach is utilized to move the particles throughout the search space, while updating each particle's "best position." However the general algorithm also tracks the "best position" for all of the particles, resulting in a known best function input when the algorithm completes.

3.4 Conclusion

The direct optimal control algorithm proved to be computationally intractable to use as an on-line path planning system. Through the use of receding horizon control the algorithms calculation time was decreased, though not enough to be considered a viable candidate for an on-line model. However, the final output is considered optimal for the current vehicle model. Although, improvements could be made to the constraints and non-linear programming solver, the results in Chapter 4 show that the direct optimal algorithm is adequate in solving large courses, just with the computation time required for an on-line system.

In contrast the computationally efficient algorithm does not suffer from run time issues. The algorithm is quickly able to plan multiple paths through the course.

These paths can than be blended to decrease overall track time. Chapter 4 shows that the advances in the algorithm place the blended path with comparible performance to the optimal path.

The final results of the work discussed in this chapter is two comprehensive algorithms that are able to handle complex and large race courses in a number of scenarios. Using these algorithms, a basis for improvement was developed that successfully tracked the improved changes in the computationally efficient method.

Chapter 4

Results

4.1 Introduction

The addition of gates along straight segments has been shown to generate a path which more closely emulates the direct optimal solution. However a standard position for these gates is not yet known. By using a simple grid search to calculate the track time with different gate positions a standard process can be created for when, where, and how to use straight cell gates. A grid search allows for several locations to be easily examined and compared.

Two algorithms have been used in these results: a baseline algorithm that provides a direct optimal solution and a more computationally-efficient algorithm. With respect to the computationally-efficient method, Section 3.3.7 proposed two algorithms for blending the minimized maximum curvature path and the shortest path. The two algorithms sequential quadratic programming and particle swarm optimization differ in their approach to minimizing the cost function, and therefore provide different benefits. To determine which is more beneficial for minimizing the track time of an autocross course, a direct comparison of these algorithm is run on thirteen representative tracks.

4.2 Course Catalog

In order to determine a good placement of gates along straight sections of the course it is important to examine several standard course maneuvers found within autocross courses. Generally an autocross course can be decomposed into three standard maneuvers: chicane, slalom, and hairpin turns. An autocross course generally features at least one of these maneuvers. By comparing the optimization algorithms through each of these turns in isolation, it provides a good representation of how the algorithms compare over an entire course.

Unlike other forms of racing, autocross courses generally avoid large straight sections. These sections are limited as to not give a large advantage to vehicles that have a high acceleration ability (that is, more expensive vehicles with larger horsepower engines). Straight sections are often also drastically limited by the space allowances for an autocross course. Lastly, there is no functional difference in either algorithm on a straight section, both ride the maximum longitudinal acceleration to the maximum velocity allowed.

To determine the optimal gate placements for the standard maneuvers (chicane, slalom, and hairpin), a grid search is performed on the placement of gates along the initial and final straight sections. 100 equally spaced gate positions are chosen along the straight sections. Each position of the gates along the initial straight section are compared to the 100 gate positions along the end, as can be seen in Fig. 4.1. This means that 10,000 different gate positions are tested on each of the standard maneuvers. This grid search method is than used to test multiple different starting angles and velocities. A note should be made than when testing the vehicle with an initial start angle the first 10 gates positions along the straight section will be utilized to set the direction of the vehicle, as seen in Fig. 4.2. Due to the nature of the maneuvers this does not change the resultant optimal gate positions.

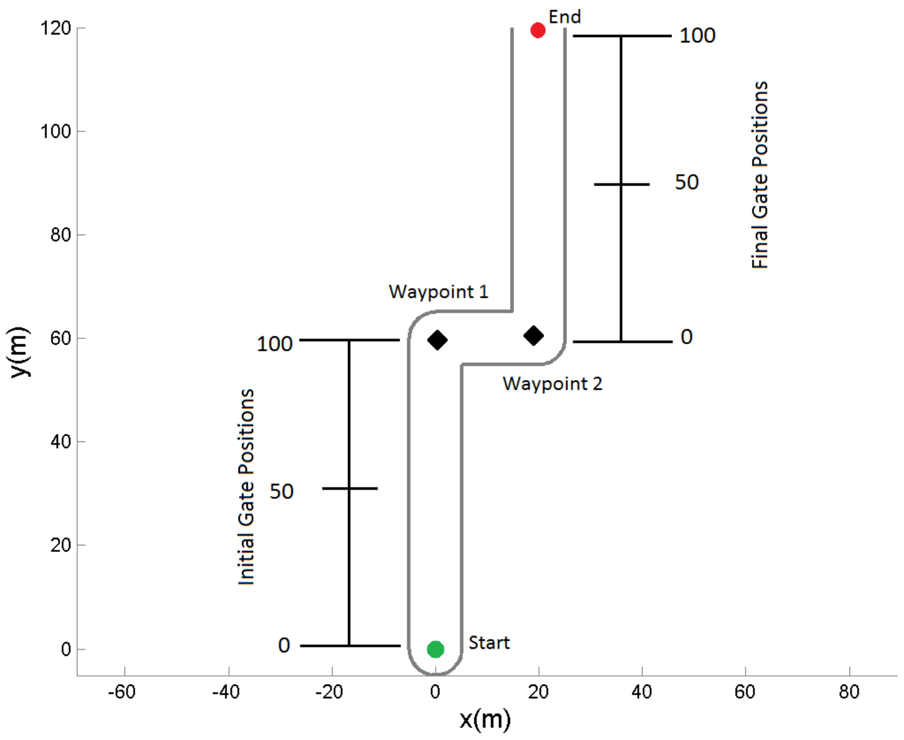


Figure 4.1: A simple autocross maneuver showing how the grid search for the placement of the initial and final gates is done. The start point of this maneuver is labeled in Green, and the finish point in Red. The course initially has two corner cell gates formed by waypoints 1 and 2.

4.2.1 Chicane

A chicane is a common vehicle maneuver, also known as a lane change. Fig. 4.3, compares the original blended path to the direct optimal path. It is clear that the blended path solution is not currently able to match the optimal solution, because the optimal path swings wide before and after the manuever, while the blended path appears to take the turns very tightly. Section 3.3.4 showed that the addition of gates along the straight section can allow the vehicle to approach the curve at a higher angle, and thereby approach the direct optimal solution.

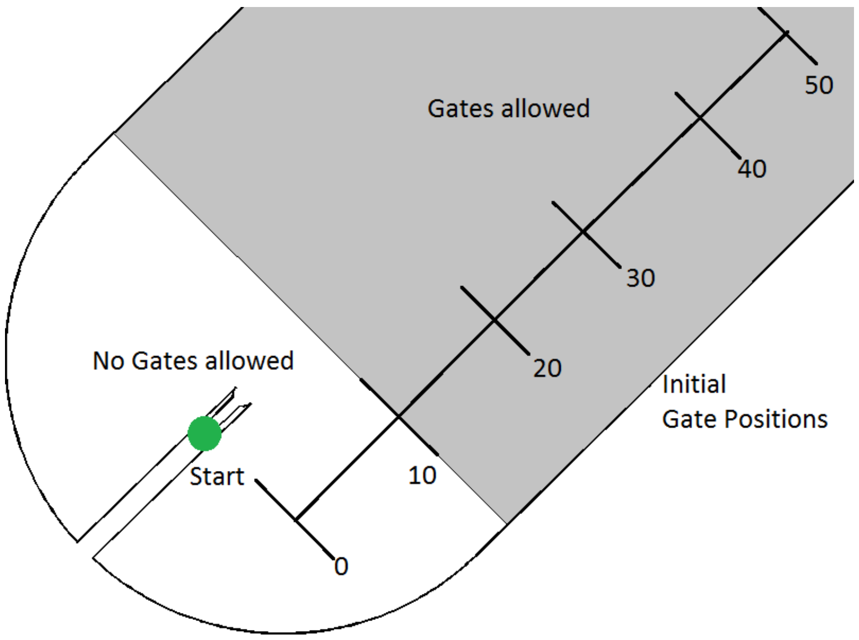


Figure 4.2: The start of an autocross maneuver where an initial angle has been set. In these cases the first 10% of the straight section will fail when a gate is placed there. The forced angle would be too high causing the vehicle to exceed it's acceleration constraints.

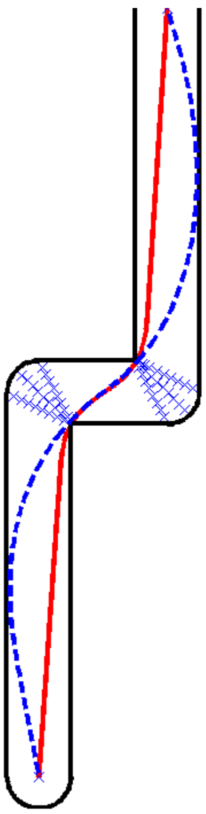


Figure 4.3: A comparison of the blended path (red) with the direct optimal control path (blue dashed). No gates have been added along the straight sections, the only gates are marked as blue X's.

The previously described grid search is used to choose the location of the optimal gates, by examining the final track times of all possible gate positions along both the initial and final straight sections. This grid search is applied to a standard chicane maneuver with the initial velocities of 5, 10, 15, and 20 m/s, which showcase a range of velocities at different points throughout the autocross course. Fig. 4.4 shows the resultant blended paths compared to a direct optimal paths for an initial velocity of 15 m/s. This showcases the general trend when changing the initial velocity. The optimal gate for the initial velocities appears to remain around 40 to 60% of the way through the straight section. It can be seen that although this addition does not directly match the optimal path, it does go a long ways towards decreasing the high point of maximum curvature at each of the original corner cells.

Fig. 4.5 shows the locations of the optimal gates for multiple different initial velocities. The final gate position appears not to change with the starting velocity; this is due in part because the vehicle is forced to slow down as it proceeds through the curves which means that it will exit the final curve at a single consistent speed, regardless of how fast it enters. However, the initial gate position does change based on the starting velocity. As the velocity increases the gate moves closer to the start point of the course which corresponds to allowing the vehicle to swing out wider in the straight section.

When a course designer is attempting to quickly encode a course with gate positions that are close to optimal it is important that they know a general placement of the gates that minimizes the track time. Fig. 4.6 shows the results of the grid search in a heat map form. It is clear that there appears to always be a good choice about 50% through the first straight cell and 42% through the final cell. This figure also shows that the optimal locations for gates are all located near each other. Fortuitously, the optimization is robust to small variation in gate placement.

Fig. 4.7 shows the final track times of the course for the blended and direct

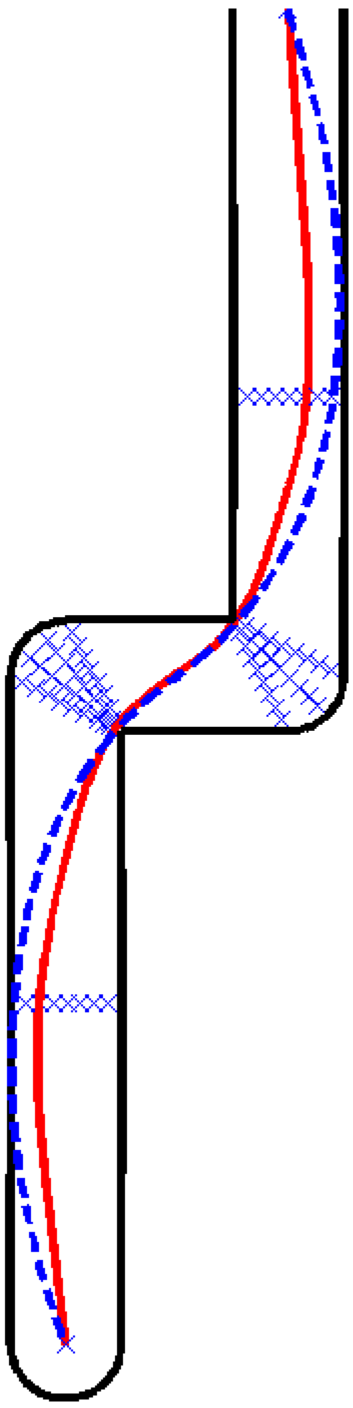


Figure 4.4: The optimal path (red) compared to the direct optimal path (blue dashed) along a standard chicane turn without a starting angle, at a stating velocity of 15 m/s.The final optimal gates are shown as blue X's.

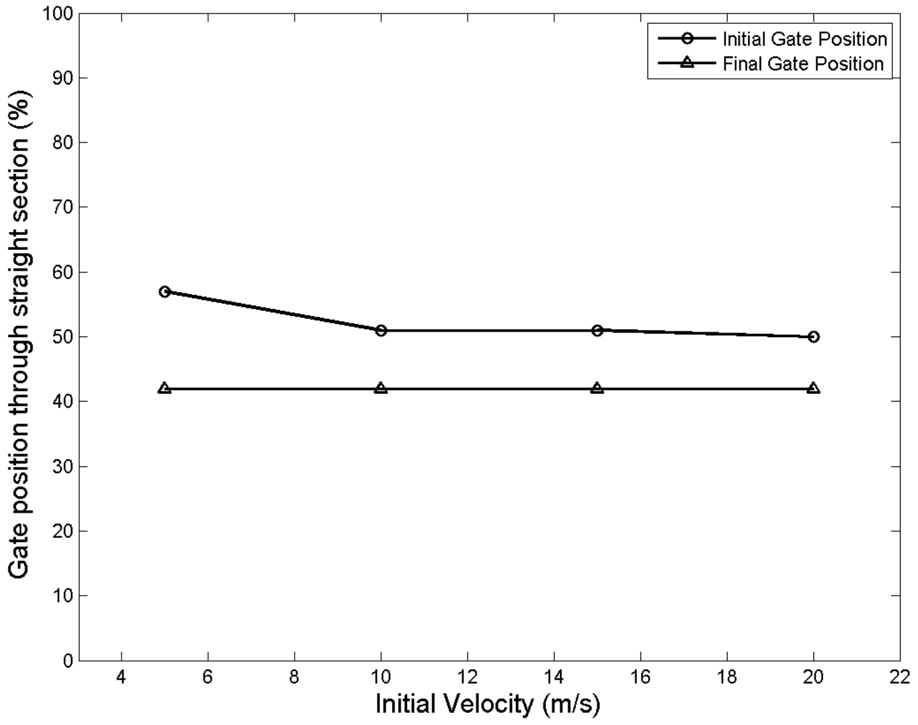
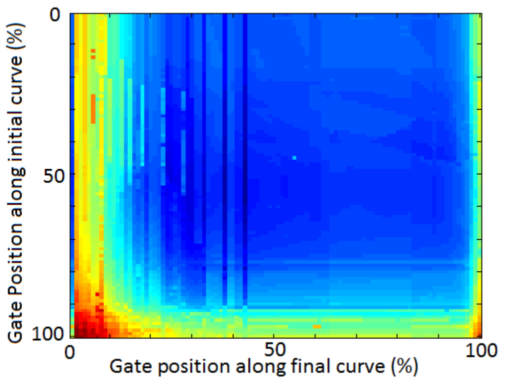


Figure 4.5: The position of the gates as a percentage through their respectice straight cells shown as the velocity changes.

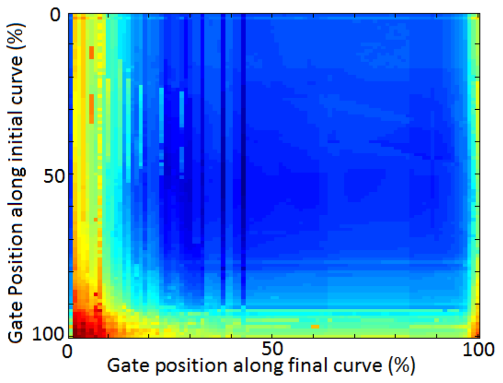
optimal solution. This figure shows that the direct optimal control solution is unable to be matched by the addition of an extra gate along the straight section. This is due impart because the direct optimal control solution is not derived from a set of Bézier curves unlike the blended path solution. Although, the optimal solution cannot be matched perfectly, the track time is still within 7-8.5% of the optimal.

In most cases, the vehicle will not be able to enter a maneuver at the optimal angle. Therefore a similar grid search is performed when an approach angle has been specified along the entrance to the maneuver. The following angles are chosen to represent standard course angles: 0, 20, and -20 degrees. These examinations will assist course designers in selecting gate positions based on the possible entrance angles to the curves.

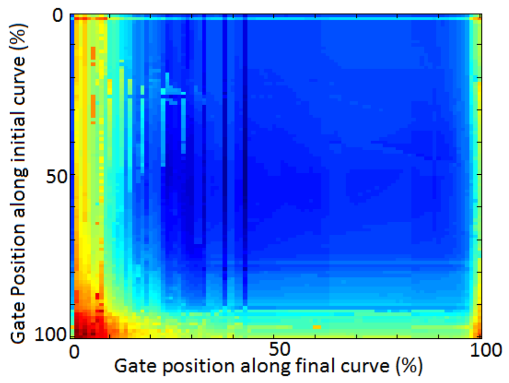
Fig. 4.8 uses an initial vehicle angle of 0 degrees. As when no starting angle



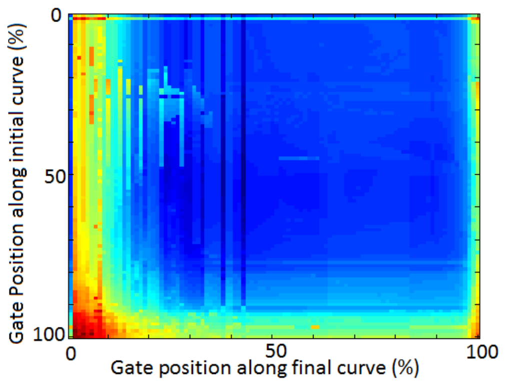
(a) 5 m/s



(b) 10 m/s



(c) 15 m/s



(d) 20 m/s

Figure 4.6: A heat map showing the total course time as a function of the gate position within the first and last chicane segments. This is a more comprehensive data set than that shown in Fig. 4.5 The x-axis is the gate positions along the final straight section. The y-axis is the gate positions along the initial straight section. The redder or hotter an area is the worse the time through the course is.

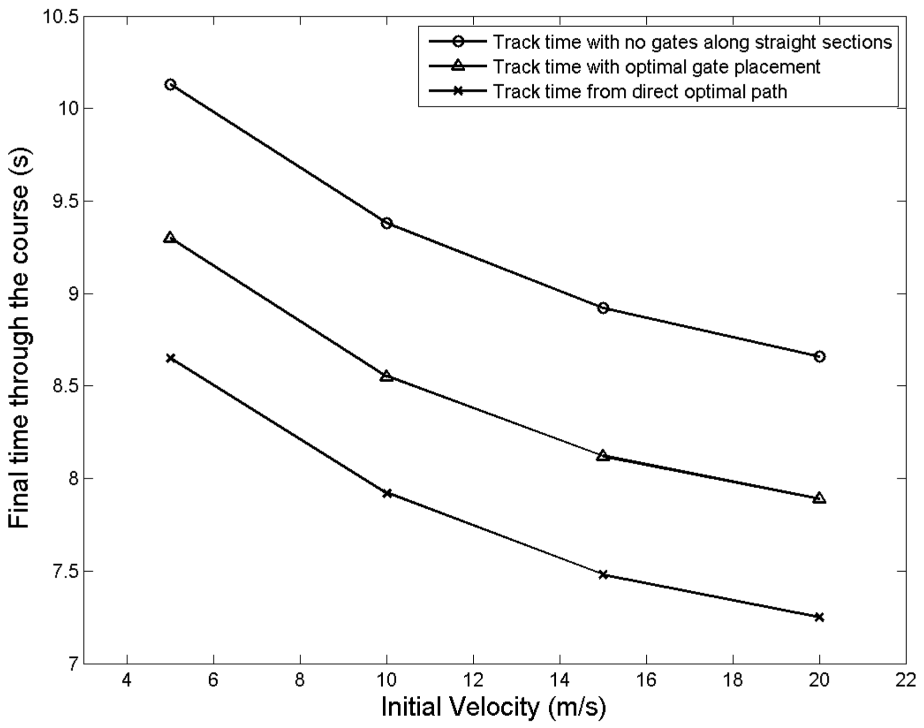
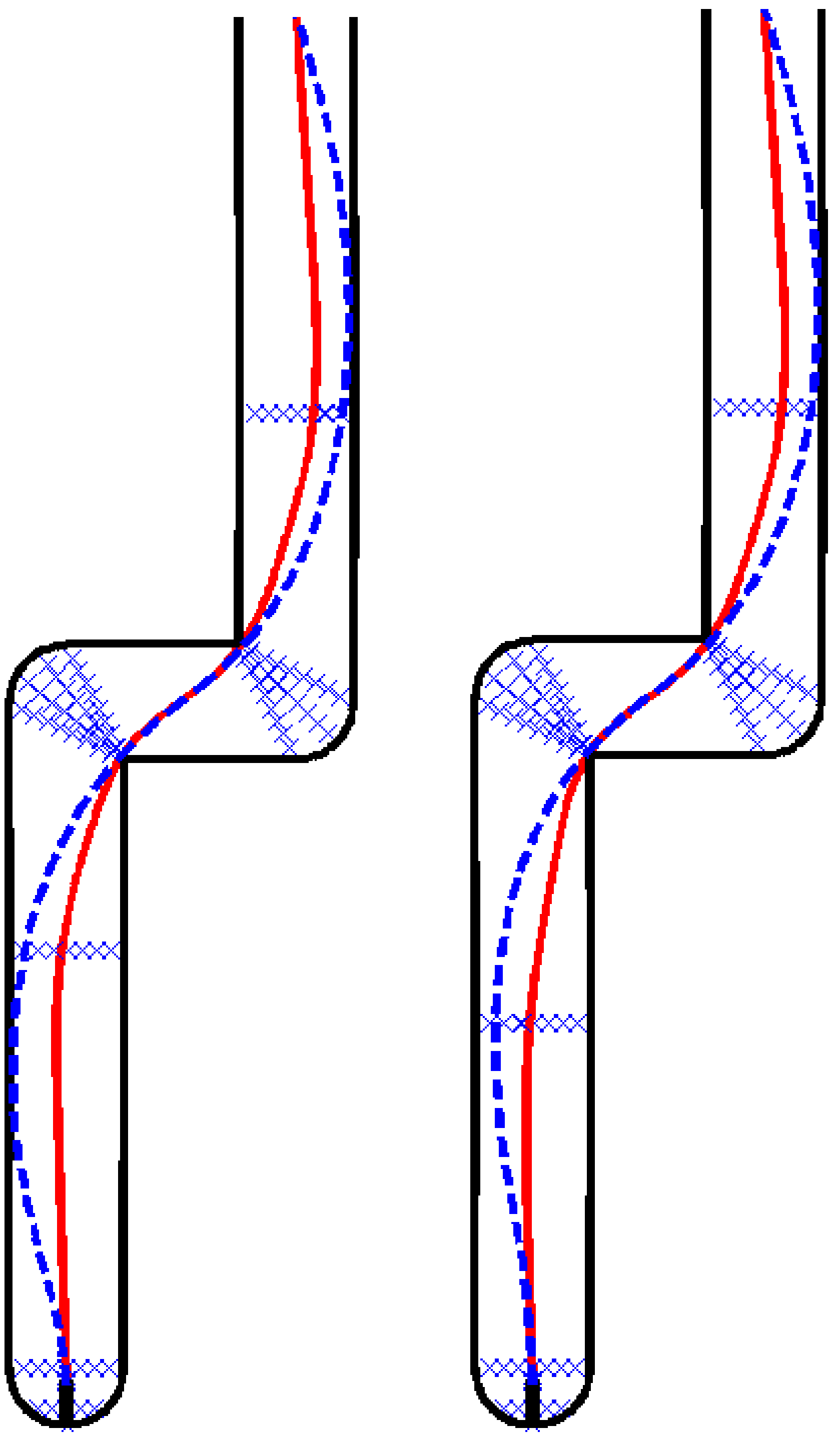


Figure 4.7: The track times for the vehicle given different starting velocities along a course with no extra gates, a course with the optimal gate position, and a course based on the direct optimal control algorithm.

was specified, the blended path solution does not match the direct optimal control solution because when the blended path attempts to swing wide it causes a large increase in the initial curvature. It can also be seen that by specifying the angle the difference between the optimal and blended paths has been increased. This is partly because the optimal path is able to quickly correct its approach angle, while the blended path cannot due to Bézier curve constraints.

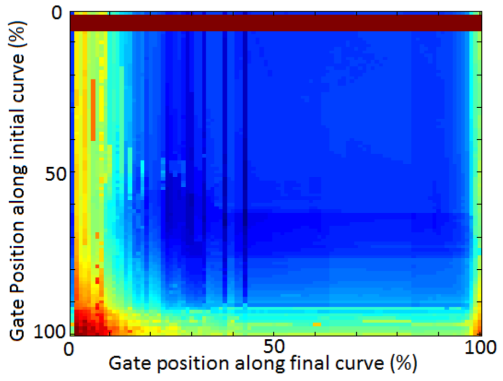
By examining Fig. 4.10 one can see that at lower velocities the position of the gates are similar. However, when the velocity increases to 20 m/s one sees a drastic change. In Fig. 4.9 it can be seen that at 20 m/s there are large portions of the heat map that are dark red. This indicates that these gate positions are not reachable by the blended path algorithm, because at these positions the vehicle would exceed its radial acceleration constraint.



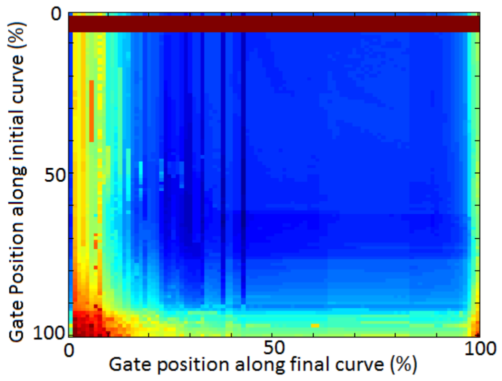
(a) 10 m/s

(b) 20 m/s

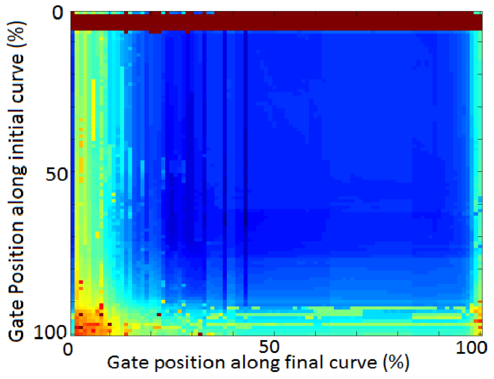
Figure 4.8: The optimal path (red) compared to the direct optimal path (blue dashed) along a standard chicane turn with a starting angle of 0 degrees, at a starting velocity of (a) 10 m/s and (b) 20 m/s.The final gates are shown as blue X's.



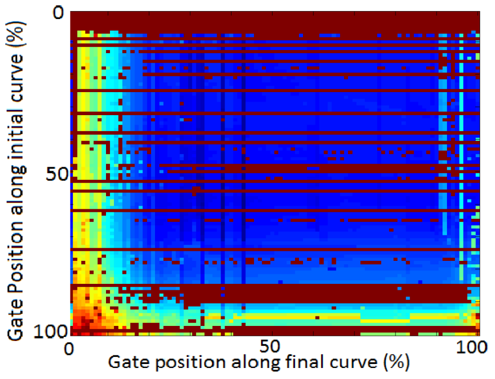
(a) 5 m/s



(b) 10 m/s



(c) 15 m/s



(d) 20 m/s

Figure 4.9: Heat maps of a chicane starting at an angle of 0 degrees for multiple initial velocities. Areas marked in red were not tested or could not be solved.

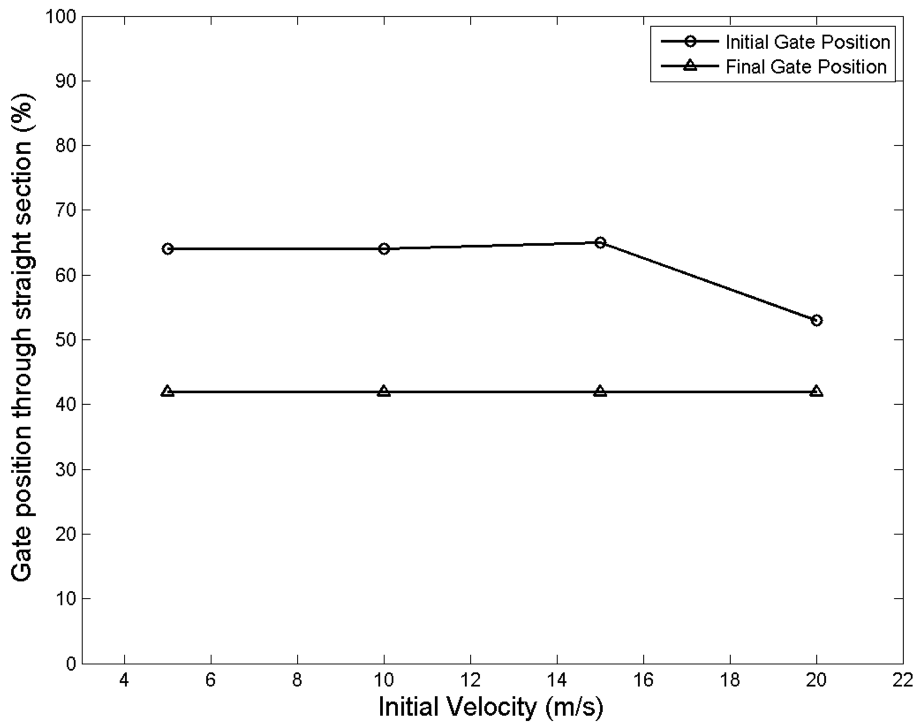


Figure 4.10: The position of the gates as a percentage through their respective straight cells shown as the velocity changes. For a chicane with a starting angle of 0 degrees.

Fig. 4.11 and Fig. 4.13 show the same result but from runs with an initial angle of 20 degrees and -20 degrees respectfully. At such drastic angles the vehicle has a hard time compensating to stay on the course at all. As the velocity increases it can be seen that the number of viable positions for the gates becomes limited, shown by the large amount of invalid/unsolvable starting positions in the heatmap (indicated by red).

The optimal gate positions for the chicanes with a starting angle of 20 and -20 degrees are shown in Fig. 4.14. The corresponding track times for these courses are shown in Fig 4.12. When the initial velocity of the vehicle reaches 10 m/s the track time suddenly increases from 5 m/s. This shows the non-optimality of the solution, due to constraints on the acceleration limits.

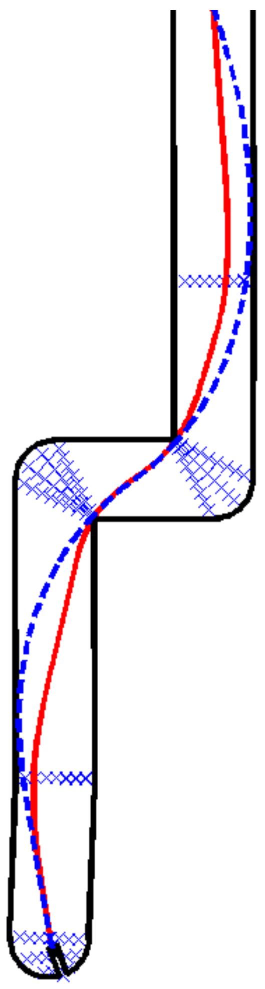
Although the blended path is unable to match the direct optimal paths track time, the blended path does improve greatly with the use of gates along the straight sections. The addition of the straight cell gates allowed the curve to more closely emulate the optimal solution. By examining the track times for the chicane maneuver it can be seen that although the blended path is unable to match the times of the direct optimal control path, it does greatly improve by 40 to 60% upon previous attempts.

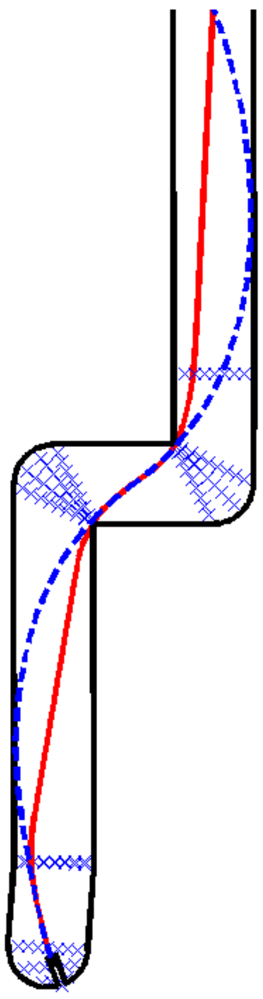
The position of the initial extra gate is highly dependent on how the vehicle enters the maneuver. However, due to the nature of the chicane, the position of the final gate should be 42% through the straight section. By using the above heat maps to determine a placement of the final gates, a decreased final track time should be expected.

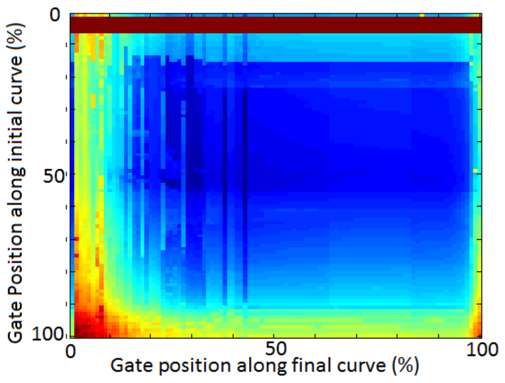
4.2.2 Slalom

Another common course maneuver is a slalom turn. A slalom is marked by a set of cones in a straight line, the goal being to maneuver between the cones without hitting them. Fig. 4.15 shows an empty slalom segment an the implied course through them. An autocross course will generally specify which direction the vehicle must pass the initial cone on.

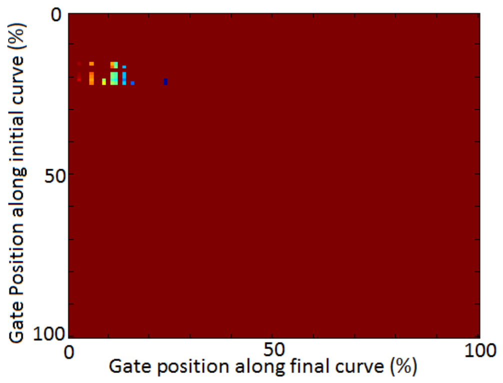
In Fig. 4.16 the results of the blended and direct optimal paths are shown with no additional gates along the straight segments. It can be seen that the blended path is unable to approach the first corner from a wide angle (similar to the chicane), therefore it can benefit from the addition of a gate along the first and last straight cells. The additional gates would allow the vehicle to enter and exit the slalom at a wider angle, thereby increasing the speed through the course. The solutions for the rest of the curve however look nearly identical to the direct optimal path.


(a) 5 m/s


(b) 10 m/s

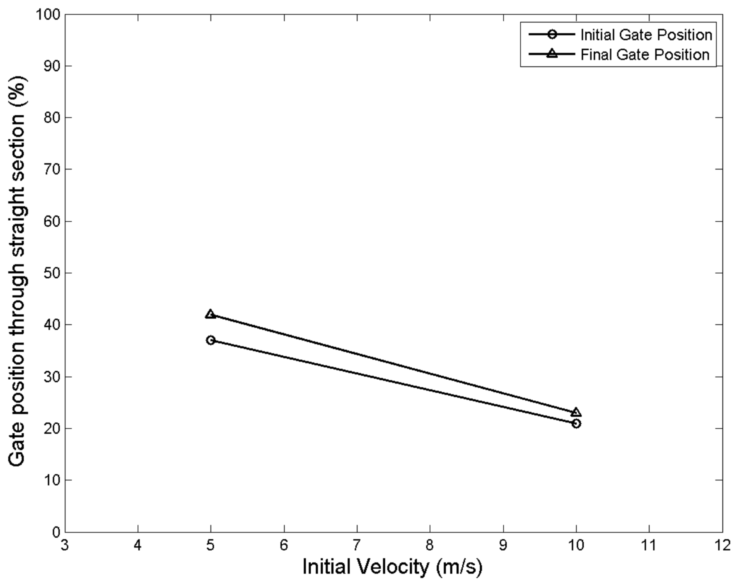


(c) 5 m/s Heat map

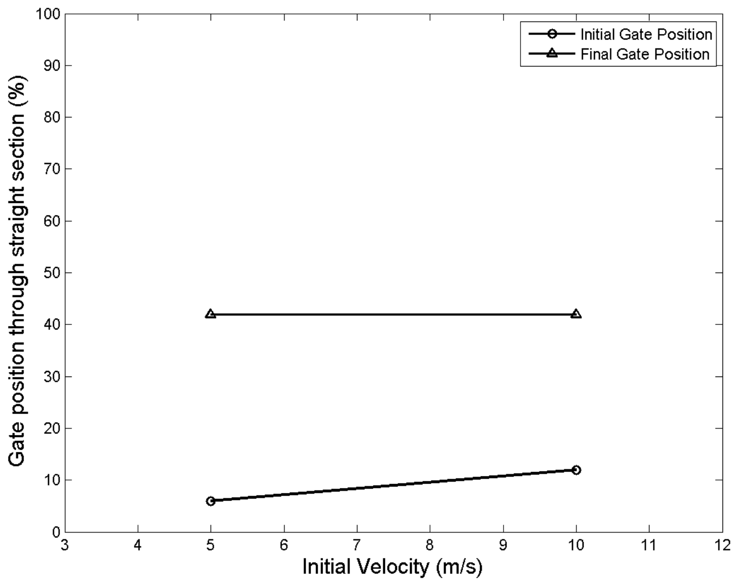


(d) 10 m/s Heat map

Figure 4.11: A comparison of the optimal path (blue dashed) to the blended path(red solid) with optimal gate placement for different velocities, at an initial angle of 20 degrees. At such a drastic angle it becomes very difficult for the the blended path to stay within the course at higher velocities. At 10 m/s there are few placements of the gates that will work.



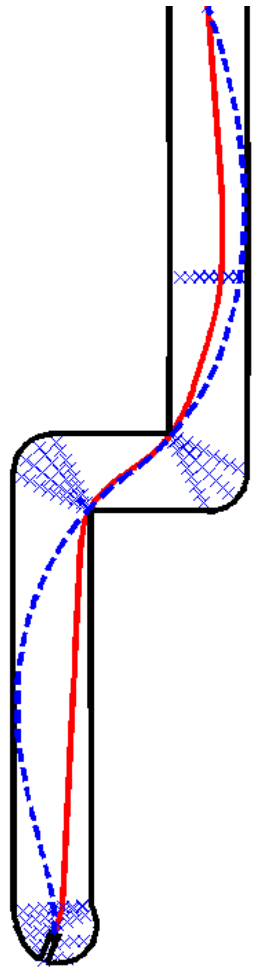
(a) 20 degree starting angle

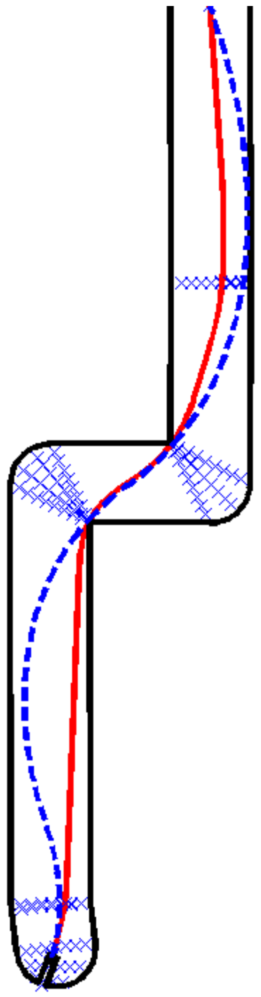


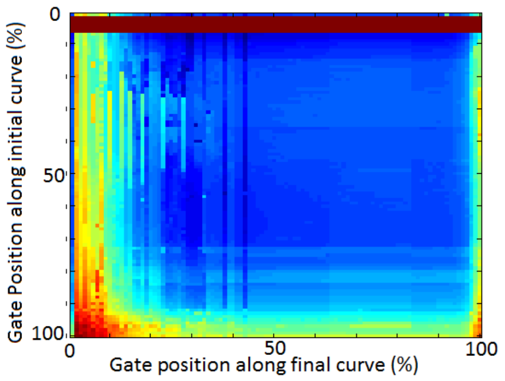
(b) -20 degree starting angle

Figure 4.12: The position of the gates as a percentage through their respective straight cells shown as the velocity changes. For a chicane with a starting angle of (a) 20 degrees and (b) -20 degrees.

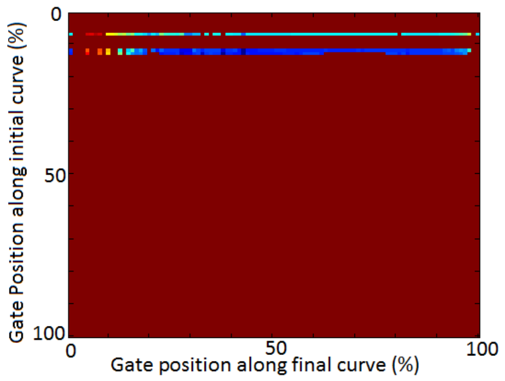
As previously done for the chicane maneuver, a selection of initial angles and velocities are selected for comparison. The resultant optimal paths of each of the initial angles are shown in Fig. 4.17 (no specified angle), 4.18 (0 degree entrance


(a) 5 m/s


(b) 10 m/s

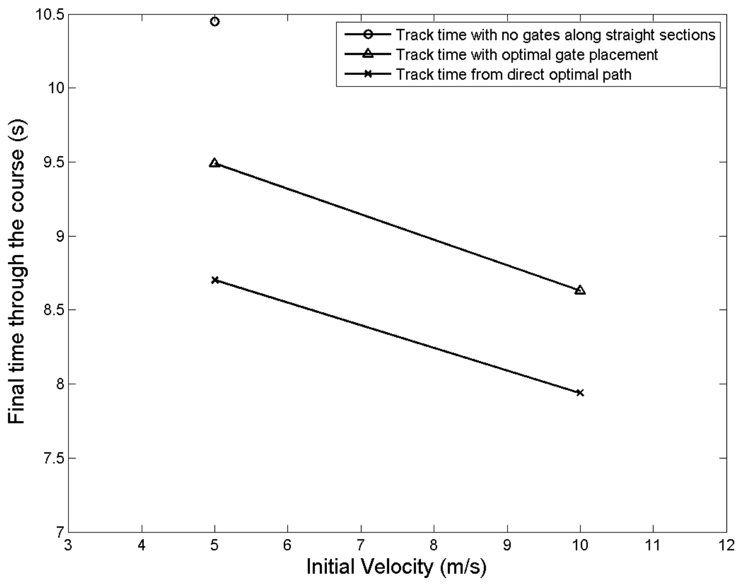


(c) 5 m/s heat map

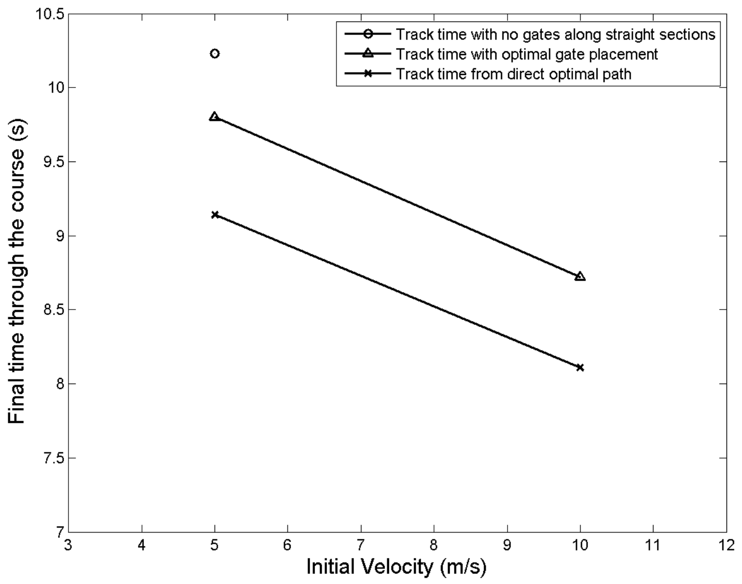


(d) 10 m/s heat map

Figure 4.13: A comparison of the optimal path (blue dashed) to the blended path(red solid) with optimal gate placement for different velocities, at an initial angle of -20 degrees. At such a drastic angle it becomes very difficult for the the blended path to stay within the course at higher velocities. At 10 m/s there are few placements of the gates that will work.



(a) 20 degree starting angle



(b) -20 degree starting angle

Figure 4.14: The track time through the chicane maneuver as the velocity changes. For a chicane with a starting angle of (a) 20 degrees and (b) -20 degrees.

angle), and 4.19 (5 degree entrance angle). It can be seen that the addition of the extra gates has increased the angle of approach of the vehicle, thereby matching the optimal solution more closely.

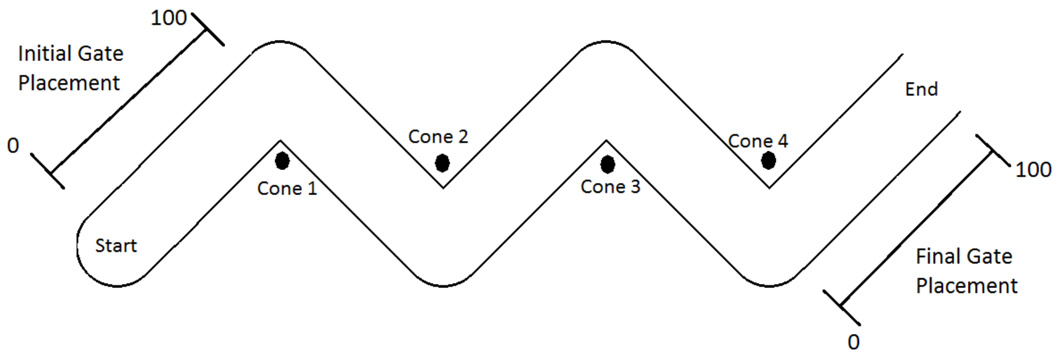


Figure 4.15: An example of a standard slalom turn. Each of the dots represent a cone on the race course. The course is generated in such a way that the vehicle will swerve between the cones without leaving the course boundaries. A grid search is performed to find the optimal straight cell gate placement along the initial and final straight cells.

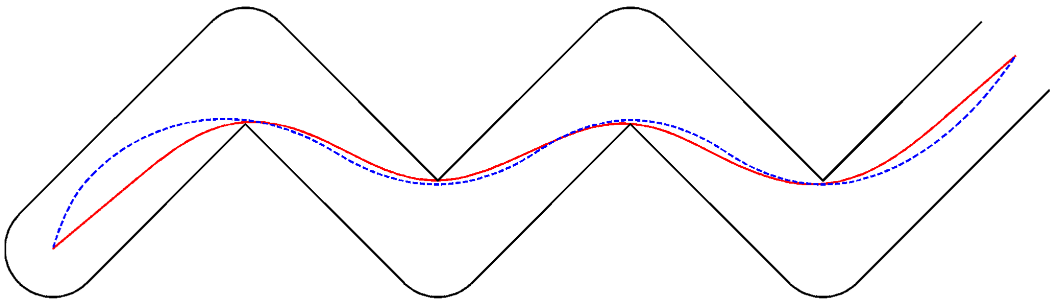
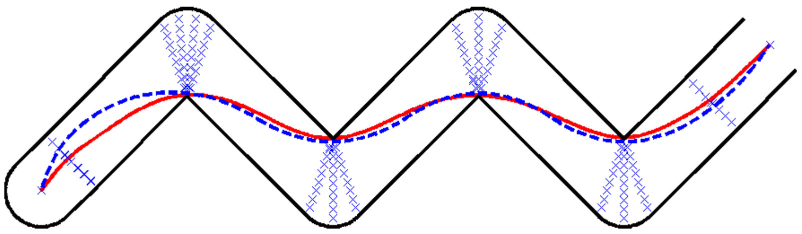


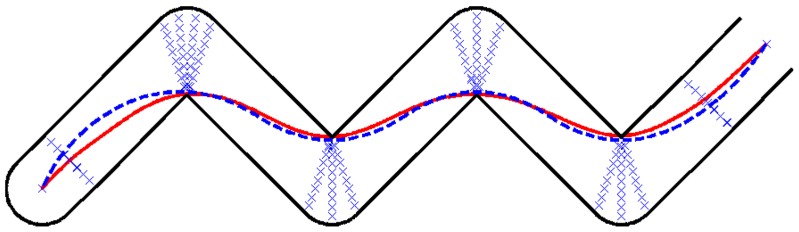
Figure 4.16: The dashed blue line represents the direct optimal solution while the red solid line represents the blended solution through a standard slalom section. Without the addition of a gate the blended path is unable to approach the first corner at a wide angle.

By examining the optimal positions of the additional gates, it can be seen that independent of angle or speed, specific locations fair better than others. The final gate is between 56% and 60%. While the initial gate should be placed between 19 to 30%. Unlike the previous chicane segment these gate positions are all towards the front, instead of the middle. This is partially due to the shorter distance from the start point to the initial curve.

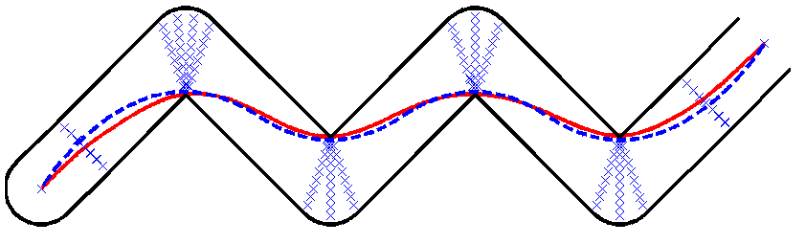
Fig. 4.20 highlights the track times for a slalom section. The direct optimal path was unable to return accurate times for the tracks with a required starting angle or high velocities, and have therefore been removed from the table. However, from the data that was properly generated it is easy to see that the addition of the



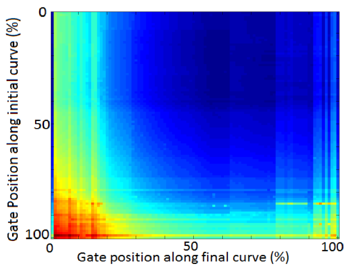
(a) 5 m/s



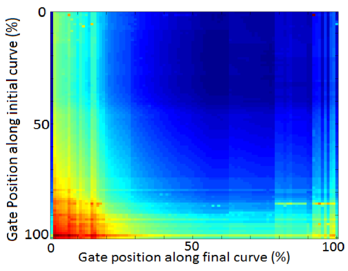
(b) 10 m/s



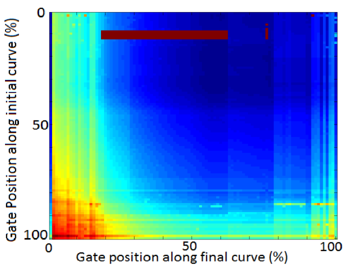
(c) 15 m/s



(d) 5 m/s heat map

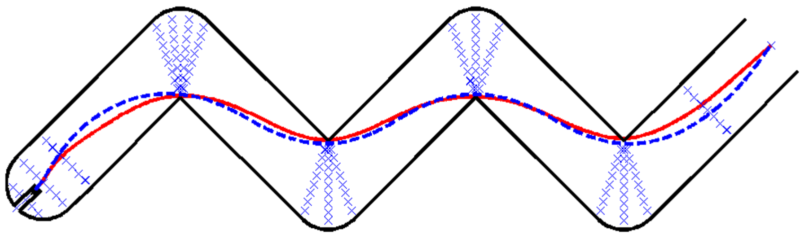


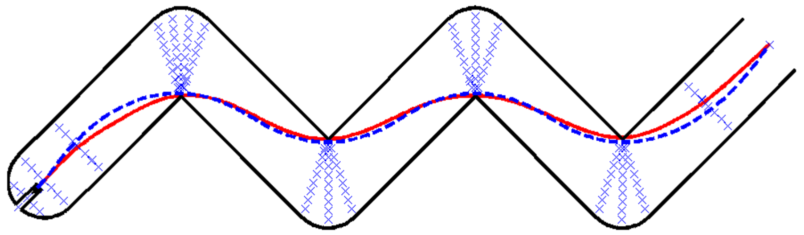
(e) 10 m/s heat map

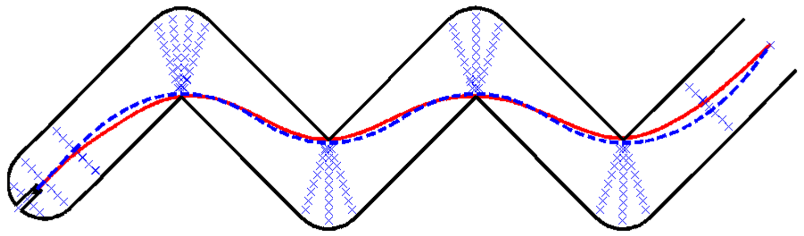


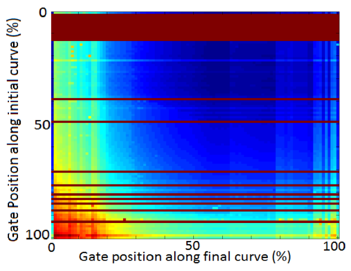
(f) 15 m/s heat map

Figure 4.17: blended path (red) and direct optimal(blue dashed) solutions to a standard slalom, with the addition of gates(blue x) along the first and last straight segments. Without a specified starting angle, at a variety of speeds, along with the associated grid search for the optimal gate placement displayed as a heat map.

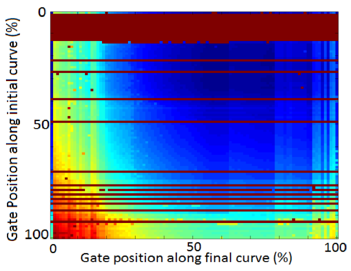

(a) 5 m/s


(b) 10 m/s

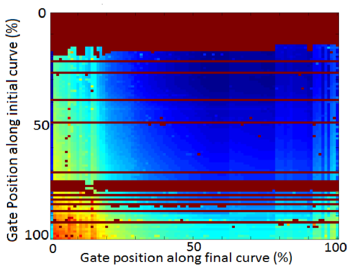

(c) 15 m/s



(d) 5 m/s heat map

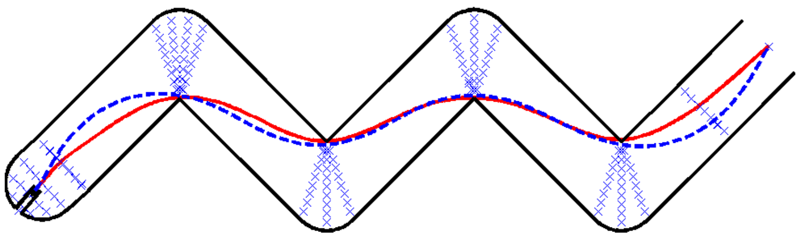


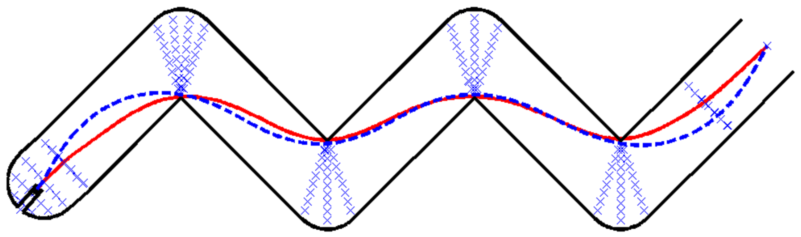
(e) 10 m/s heat map

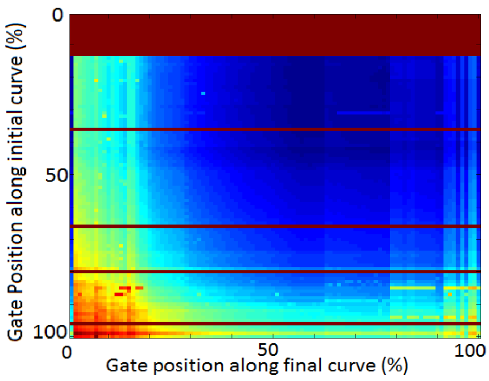


(f) 15 m/s heat map

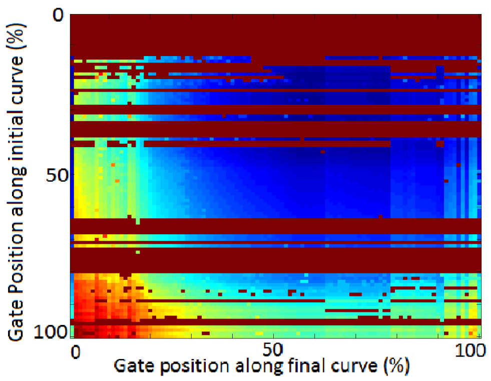
Figure 4.18: blended path (red) and direct optimal(blue dashed) solutions to a standard slalom, with the addition of gates(blue x) along the first and last straight segments. The vehicle used an initial starting angle of 0 degrees, at a variety of speeds, along with the associated grid search for the optimal gate placement displayed as a heat map.


(a) 5 m/s


(b) 10 m/s



(c) 5 m/s heat map



(d) 10 m/s heat map

Figure 4.19: blended path (red) and direct optimal(blue dashed) solutions to a standard slalom, with the addition of gates(blue x) along the first and last straight segments. The vehicle used an initial starting angle of 5 degrees, at a variety of speeds, along with the associated grid search for the optimal gate placement displayed as a heat map.

gates along the straight sections does decrease the final track time.

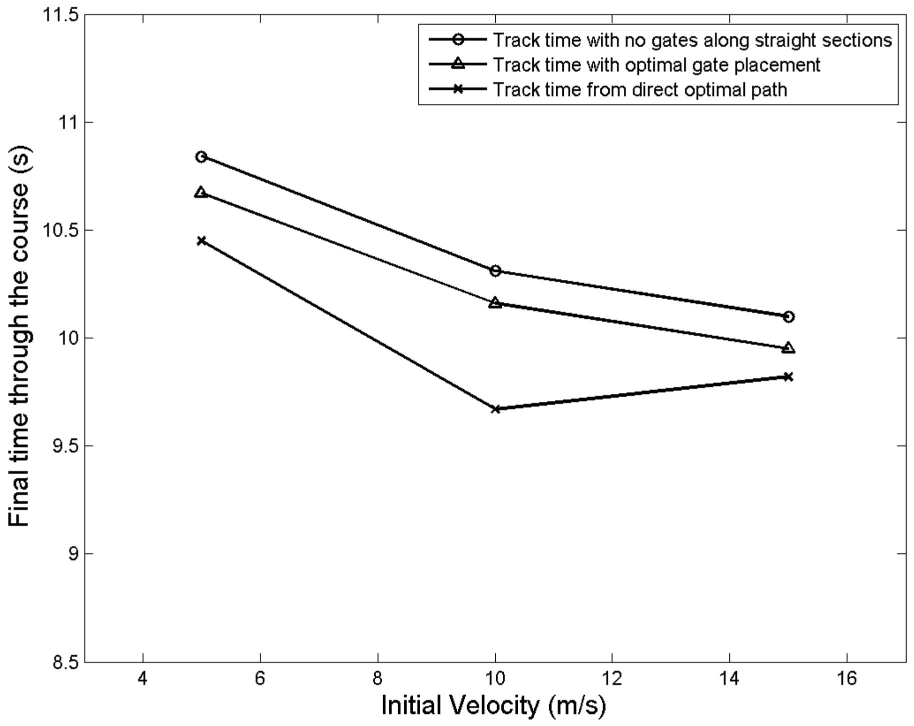
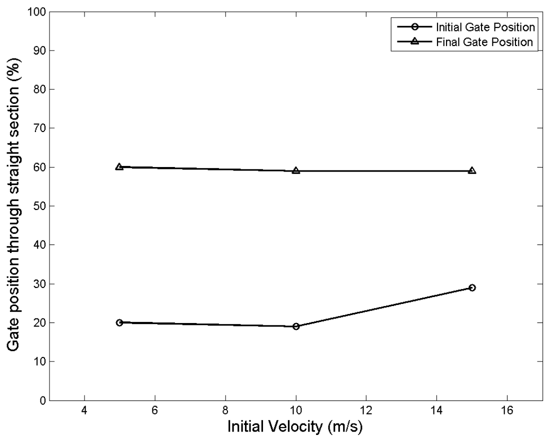


Figure 4.20: The track times for the vehicle given different starting velocities along a slalom maneuver with no extra gates, a course with the optimal gate position, and a course based on the direct optimal control algorithm

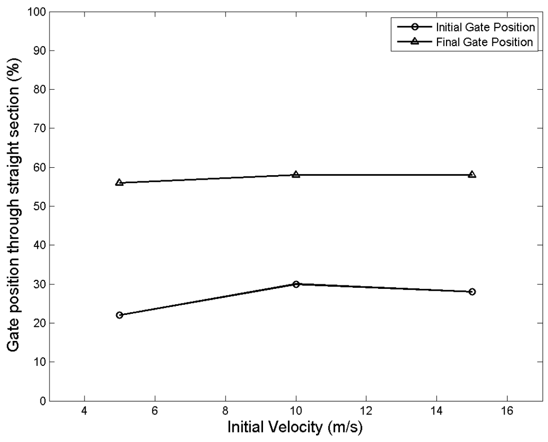
4.2.3 Hairpin

The final standard course segment is the hairpin turn. The angle of a hairpin turn can vary anywhere from 90 degrees to 180 degrees. Fig. 4.22 shows the solution to a hairpin turn without the addition of gates along the straight sections. It is clear that the path does not match the optimal path shown.

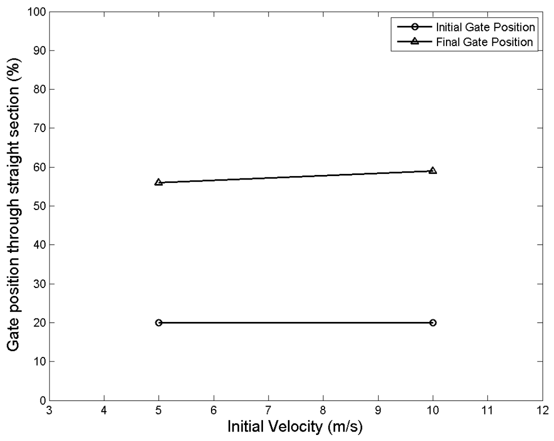
The hairpin turn can benefit from additional gates along the straight sections. Unfortunately, due to the curve being so drastic the blended path was unable to calculate paths for multiple different starting angles. Fig. 4.25 compares the blended path to the direct optimal path with the addition of the added gates along the entrance and exit straight segments.



(a) No required starting angle



(b) 0 degree starting angle



(c) 5 degree starting angle

Figure 4.21: The position of the gates as a percentage through their respective straight cells shown as the velocity changes. For a slalom with a starting angle of (a) no starting angle, (b) 0 degrees and (c) 5 degrees.

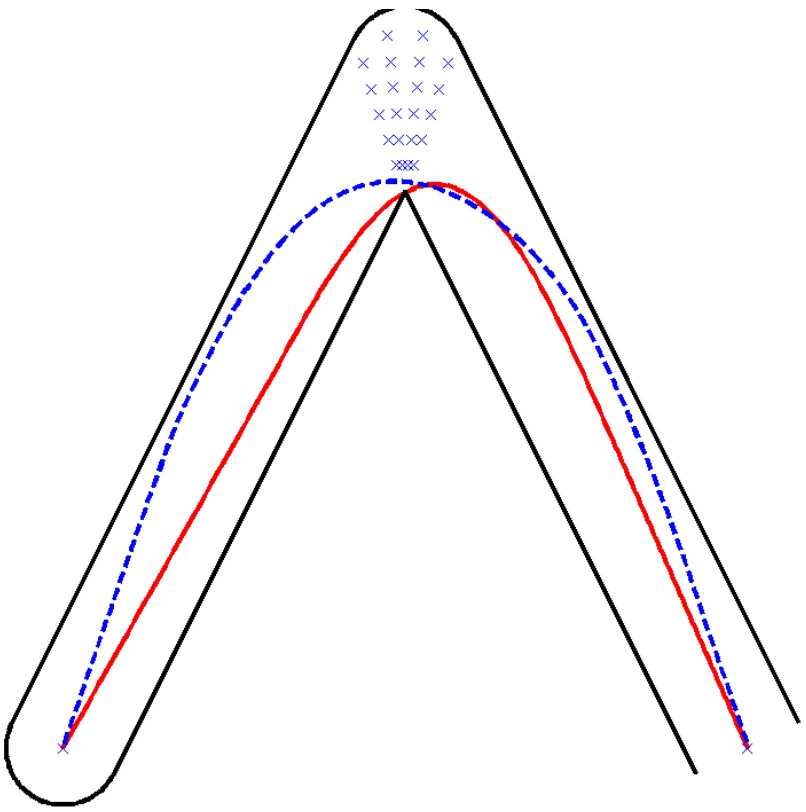


Figure 4.22: A comparison of the direct optimal solution (blue dashed) and the blended solution (red), along a hairpin turn without the use of gates along the straight sections. Where the blue 'X's signify the only gates along the course.

By examining the graphs and gate positions it can be seen that the gate positions do not change, and that the solution to the course remains the same no matter the starting velocity. Comparing the results and the heat maps it appears that placing the initial gate towards the beginning of the curve is an optimal position, while placing the final gate between 56 to 60% is optimal.

Fig. 4.24 shows the track times for each of the paths shown in Fig. 4.25, compared to the blended solutions with no gates along the straight section. The figure once again shows that although the blended solution with gates along the straight section does not match the optimal solution, the blended path solution improves when compared to the course without the extra gates.

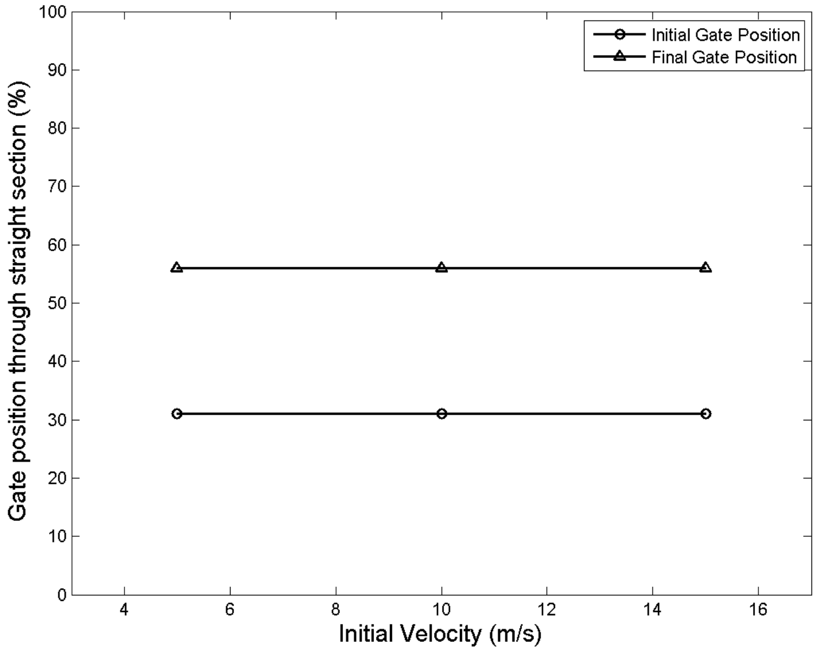


Figure 4.23: The position of the gates as a percentage through their respective straight cells shown as the velocity changes. For a hairpin turn with no starting angle.

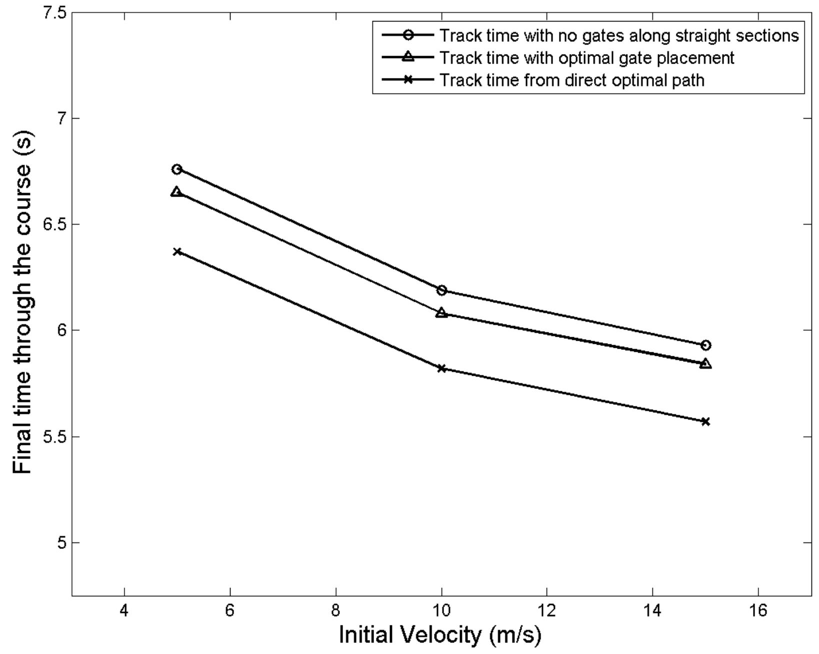
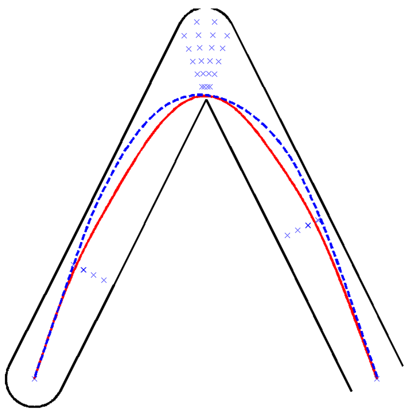
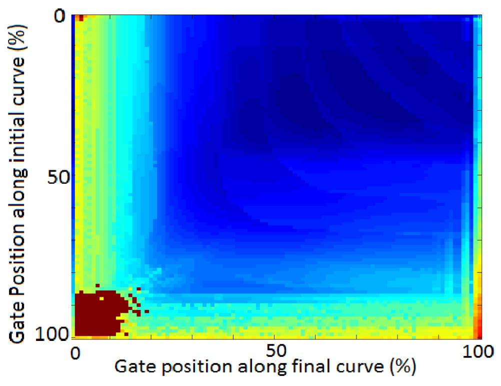
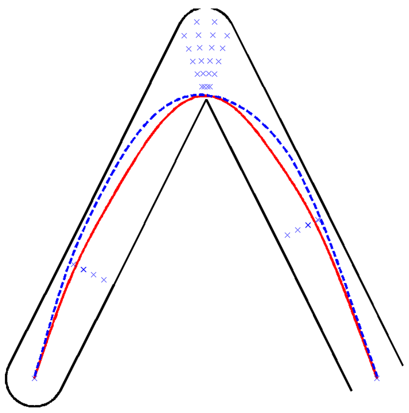


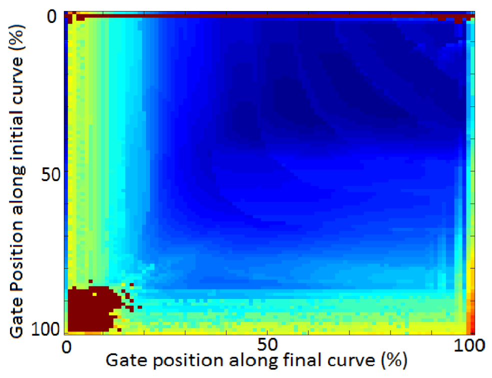
Figure 4.24: The track time through the hairpin turn as the velocity changes. For a course with no starting angle.


(a) 5 m/s

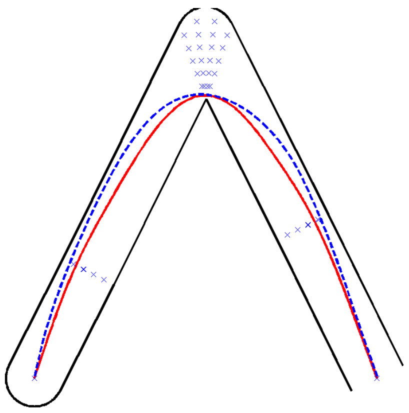


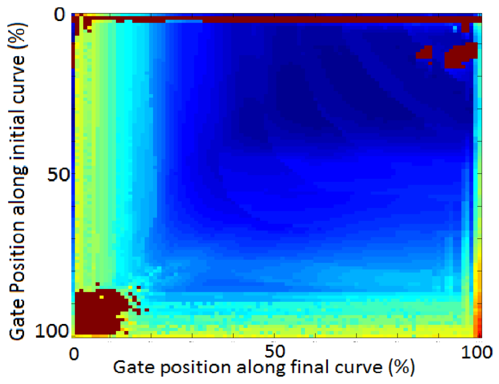
(b)


(c) 10 m/s



(d)


(e) 15 m/s



(f)

Figure 4.25: blended path (red) and direct optimal (blue dashed) solutions to a standard hairpin turn, with the addition of gates(blue x) along the first and last straight segments. The grid search results for gate placements are included alongside the path results.

4.3 Real Course Comparison

Though analyzing the individual turns common to autocross provides a clear and narrow view of algorithm performance, full courses require testing as well. It is only when validated on real courses and compared to actual driver times that these algorithms demonstrate their worth, as the goal is for performance to be comparable, if not better than, human drivers.

A vehicle model is first necessary to compare the algorithm to human drivers. Equipe racing [10] has a set of videos that users have generated from accelerometer data to track the G forces on the car. These videos display the velocity, the current lateral acceleration, and the current longitudinal acceleration. The vehicle model used within this comparison is based on this data. The vehicle model is shown below:

$$\begin{aligned} a_{rMax} &= G * .9 \quad m/s^2 \\ a_{tMax} &= G * .6 \quad m/s^2 \\ a_{tMin} &= G * .8 \quad m/s^2 \\ V_{max} &= 80 \quad km/h \end{aligned} \tag{4.1}$$

The above model was used when evaluating a set of different autocross courses. Fig. 4.26, shows a course designed by Equipe Racing [10]. This course contains all of the elements of a standard autocross course including multiple: chicane, slalom, and hairpin turns. This makes it a good course for benchmarking the blended path and direct optimal control algorithms.

This course was hand encoded by measuring the distance from each of the cones

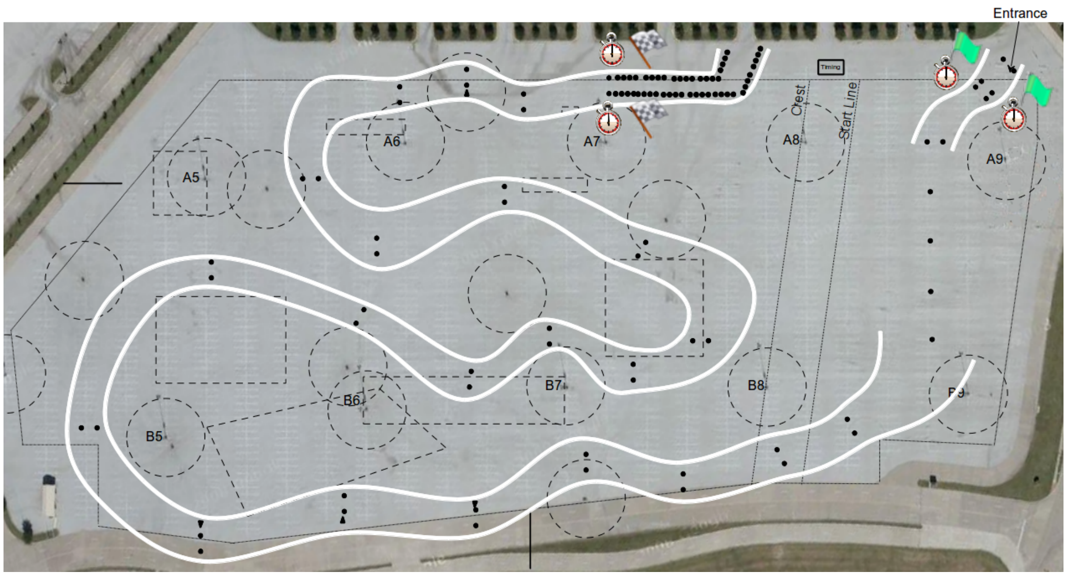


Figure 4.26: A photo of an autocross course from Equipe Racing. The course is overlaid on a parking lot with markings of how the race will be run. However, the black points represent the cones that must be passed in order to complete the track. The white line is a corridor width which limits how wide a car can turn when driving. The start and end points are marked by time clocks to signify the start and stop of the time trial.

marked in the figure, such that it would properly traverse all gates along the course. Gates were added such that the approach to a section could be optimized, based on the results of Section 4.2. However, in most parts of the course, the race lines were tight and the addition of gates did not decrease the track time.

The blending of the MMC and shortest paths can be accomplished using several different algorithms. This work focuses on comparing a multiple shooting method, particle swarm optimization(PSO), to sequential quadratic programming (SQP). These algorithms are both used to plan a blended path through the Challenge Course 1. The results of both the computation time, and the final track times are given in Fig. 4.27. The results show that both the PSO and SQP algorithms are within seconds of each other, but that the computation time for the algorithms is a full minute different. This large computational difference means the SQP algorithm is unable to be used as an on-line path planning algorithm.

When the track times are compared to the actual results, provided by Equipe Racing from their race in 2014, the generated paths come in the top 10 places out of 75 different racers. The actual track times ranged from 62.18 seconds to 85.636 seconds, which places the SQP programming method 3.6 seconds away from first place.

As a comparison to blended algorithms, a solution for the course was also found using the direct optimal control method. As described in Section 3.2.3, receding horizon control was used to look seven way-points ahead at a time. Using the same vehicle model above a path through the course is calculated, the resultant path can be seen in Fig. 4.28. It can be seen that in many points along the course the direct optimal solution swings wider than the blended curve solution, allowing the vehicle to reach a higher velocity.

The resultant track time for the optimal course is 58.62 seconds, which is 3.5 seconds faster than the person who took first place at the Equipe race. This shows that the direct optimal path algorithm has the ability to properly generate a path that is better than any human driver. However it took the algorithm 7+ hours to generate a solution, making the solution intractable for use with any kind of on-line system.

4.4 Direct Algorithm Comparison

When designing an on-line path planning algorithm it is important that the algorithm be both computationally efficient and provide a minimal race time. These two factors are in competition, an algorithm rarely is both optimal and computationally efficient. Two blending algorithms are compared in this thesis. In the previous section we directly compared the compuation time and track time of the sequential quadratic programming (SQP) algorithm and the particle swarm optimization (PSO) method. The following section more directly compares the two algorithms to determine which is the optimal trade off for an on-line path planning algorithm. It has already been shown that the direct optimal path is intractable for use on any on-line system and due to computational constraints was not run on the following paths.

To properly compare the algorithms it is important to use a large sample size of race courses, and so 13 real-world autocross courses were evaluated. Table 4.1 contains a figure of each of the tracks, and notes how many gates were required to create the track. A large set of courses were chosen with different amounts of gates to show how to algorithms perform on small and large tracks.

As a basis of comparison for the two algorithms the track time of the MMC path and the shortest paths track times are calculated through each of the courses. Table 4.2 contains the final times for each of the courses. It can be seen that in 8 of the 13 courses the MMC path has a faster track time.

The PSO and SQP algorithms were each run on the 13 autocross courses. These two algorithms track times are directly compared in Table 4.4 which takes the track time minus the minimum of the MMC or shortest path; in 9 of the 13 courses SQP has the best final track time.

Although in most of the courses the SQP algorithm performs best. The final race times are always within within 1-2% of each other. This result shows that although SQP generally out performs PSO, the PSO algorithm can still provide a good solution to a course. Since track time is not the only important detail when selecting a path blending algorithm the computation time is also compared in 4.5.

The computation times for the courses shows a large difference in the amount of time it takes SQP and PSO to compute a solution. For 11 of the 13 courses PSO obtains its solution faster than SQP as seen in Table 4.3. Fig. 4.29 shows clear correlation that as the number of gates increases so does the computation time of

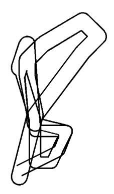
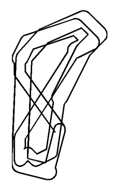
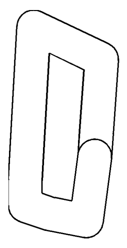
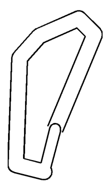
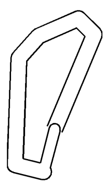
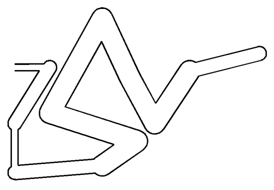
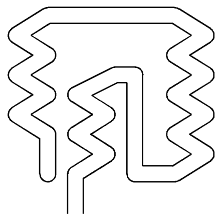
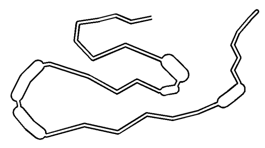
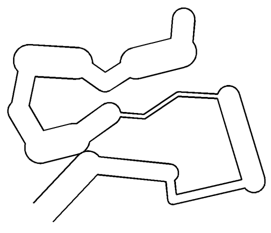
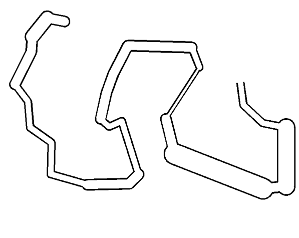
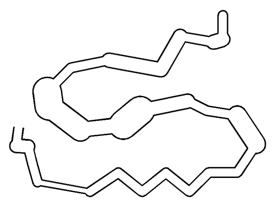
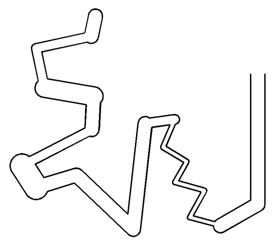
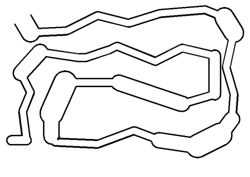
 (a) Shoreline 1 – 21	 (b) Shoreline 2 – 22	 (c) Shoreline 3 – 9
 (d) Shoreline 4 – 8	 (e) Shoreline 4 with additional gates – 10	 (f) Top Hairpin – 12
 (g) Field Test Long –27	 (h) Challenge Cup 1 – 44	 (i) Challenge Cup 5 – 18
 (j) Challenge Cup 6 –29	 (k) Challenge Cup 7 – 28	 (l) Challenge Cup 8 – 20
 (m) ER Event 7(2011) – 37		

Table 4.1: Diagrams of the thirteen autocross courses used for algorithm evaluation. The number of gates that define them is also listed.

Track	MMC (s)	Shortest Path (s)
Shoreline 1	**49.86**	49.99
Shoreline 2	52.08	**51.64**
Shoreline 3	17.37	**16.76**
Shoreline 4	24.31	**23.82**
Shoreline 4 with additional gates	23.30	**22.53**
Top Hairpin	**20.31**	21.04
Field Test Long	28.93	**28.12**
Challenge Cup 1	**67.95**	71.01
Challenge Cup 5	**28.34**	29.85
Challenge Cup 6	**43.00**	45.30
Challenge Cup 7	**47.11**	50.96
Challenge Cup 8	**36.58**	38.19
ER Event 7 (2011)	**56.23**	56.74

Table 4.2: Lap times for the Minimized Maximum Curvature (MMC) and the Shortest paths. Shortest times for a course are bolded.

Track	SQP (s)	PSO (s)
Shoreline 1	23.93	5.66
Shoreline 2	17.32	5.92
Shoreline 3	1.88	1.75
Shoreline 4	0.66	1.45
Shoreline 4 with additional gates	2.87	1.90
Top Hairpin	1.91	2.68
Field Test Long	21.38	5.62
Challenge Cup 1	108.75	16.04
Challenge Cup 5	5.87	4.48
Challenge Cup 6	52.26	8.81
Challenge Cup 7	42.99	8.21
Challenge Cup 8	8.72	4.92
ER Event 7 (2011)	78.47	12.31

Table 4.3: Computation time differences between the SQP and PSO algorithms, on thirteen autocross courses.

each algorithm. The figure shows that as the number of gates increases the PSO's computation time increases linearly, while SQP's increase with a much larger slope. This causes large problems on courses with a large number of gates like the ER Event 7 course which took the SQP algorithm 78.47 seconds to calculate, which is more than five times the 12.31 seconds for the PSO algorithm.

Track	**SQP (s)**	**PSO (s)**
Shoreline 1	-2.7800	-2.2600
Shoreline 2	-2.4500	-2.2500
Shoreline 3	-0.6300	-0.6300
Shoreline 4	-0.1200	-0.1200
Shoreline 4 with additional gates	-0.3800	-0.3800
Top Hairpin	-0.6200	-0.6300
Field Test Long	-0.0400	-0.0200
Challenge Cup 1	-2.6700	-1.8900
Challenge Cup 5	-1.5800	-1.4100
Challenge Cup 6	-1.0500	-0.6700
Challenge Cup 7	-2.7200	-2.3200
Challenge Cup 8	-0.8700	-0.6500
ER Event 7 (2011)	-4.0000	-3.4700

Table 4.4: Track times for blending algorithms compared to the minimum of the MMC or shortest paths.

Plotting the computation time versus the track time shows a similar result for the time difference based on the different tracks, as seen in Fig. 4.30. It is seen that as the courses become larger it becomes more difficult for SQP to efficiently calculate the track. This is partially due to the fact that SQP does not have a definitive end condition, where as PSO ends after a certain number of generations.

Finally a graph of the time differences between PSO and SQP can be seen in Fig. 4.31. This shows how close the SQP and PSO track times are because although SQP was able to obtain the best final track time, that PSO was always close behind. Both algorithms also showed that they improved greatly over the standard minimized maximum curvature path or the shortest path.

4.5 Conclusion

While the algorithms have been shown to be sound in theory, this chapter evaluated multiple algorithms in both specific situations and over entire courses. These evaluations examined both the algorithm run-time as well as the optimality of the final

Track	**SQP (s)**	**PSO (s)**
Shoreline 1	23.93	**5.66**
Shoreline 2	17.32	**5.92**
Shoreline 3	1.88	**1.75**
Shoreline 4	**0.66**	1.45
Shoreline 4 with additional gates	2.87	**1.90**
Top Hairpin	**1.91**	2.68
Field Test Long	21.38	**5.62**
Challenge Cup 1	108.75	**16.04**
Challenge Cup 5	5.87	**4.48**
Challenge Cup 6	52.26	**8.81**
Challenge Cup 7	42.99	**8.21**
Challenge Cup 8	8.72	**4.92**
ER Event 7 (2011)	78.47	**12.31**

Table 4.5: Computation time of the blending algorithms, with the minimal computation time highlighted for each for each of the tracks.

solution, as both are relevant for the domain in which this work seeks to be applied.

Previously the only position for gates was along the corner cells of the course. This proved to be a non-optimal decision because the planned path was unable to take advantage of the entrance and exit angles from a large turn. With the addition of gates along straight sections, the planned paths more closely emulated the direct optimal solution. The placement of these gates however is highly dependent upon the course.

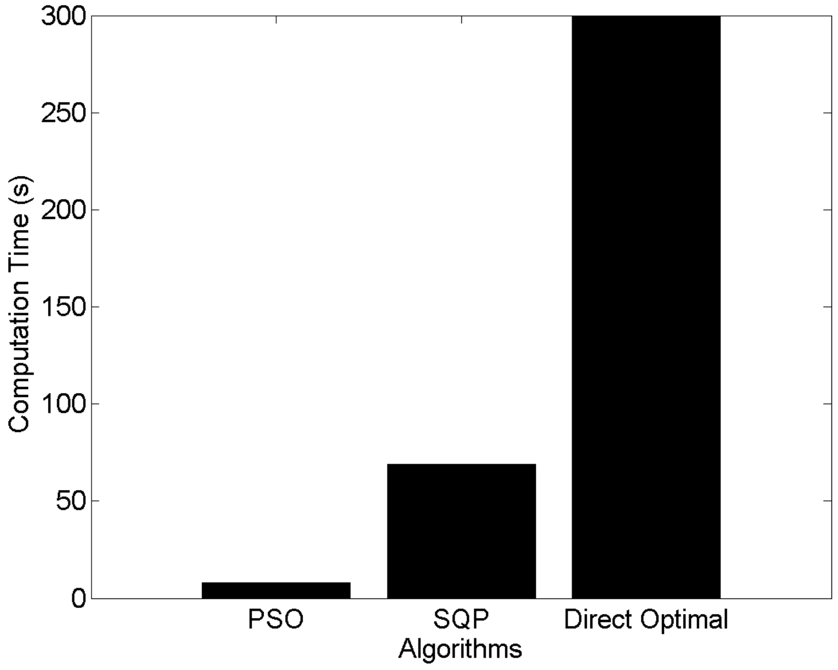
However, by studying common race maneuvers: the chicane, slalom, and hairpin turns, generalizations can be made about the placement of these gates. In general the gate placement should not be too close to either the previous or next gates. As the vehicle gains a higher velocity the gates should be moved closer to the previous gate. This thesis found that the optimal placement should be at least 15 meters away from the next gate, but no more than 35 meters away. A similar assumption can be made for gates placed along the exit of curves.

These distances, although not guaranteed to be the optimal position for the gate, will still provide an advantage and allow the vehicle to achieve a shorter track time. However, it should be noted that as the number of gates increase so will the computation time. These measures must be weighted against each other when designing a course.

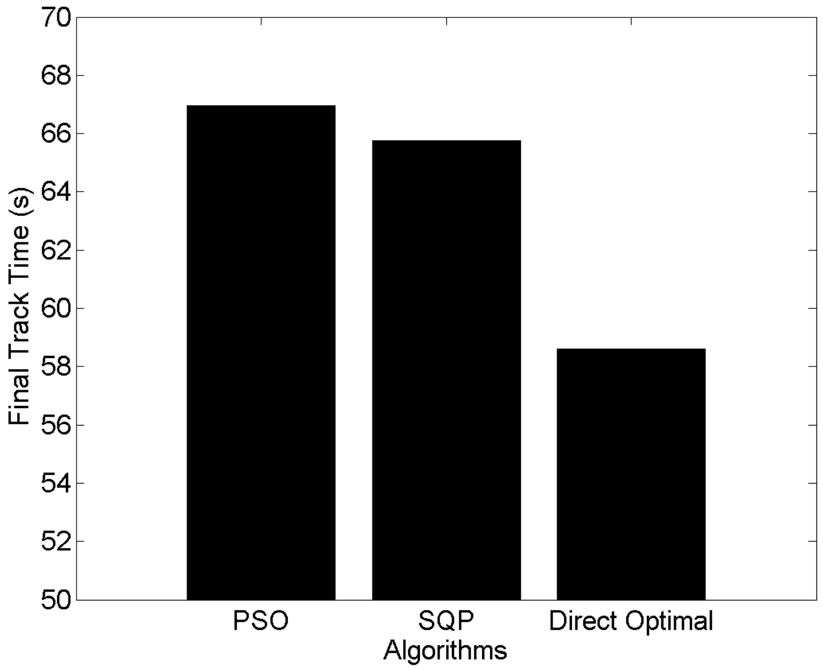
The planned paths through the courses closely emulates real vehicles on a race course, as was seen in Challenge Course 1, Section 4.3. This is a good sign that both the vehicle model and path planning algorithms were created under reasonable assumptions. It was also shown that the direct optimal solution was solvable on a large real-life course, although the computation took hours.

SQP and PSO each differ in their approach to solving non-linear problems, and were compared across a number of tracks. The results from this comparison showed that SQP will have better track times compared to PSO, but comes at a large cost of computational efficiency. On average, the PSO algorithm generated track times within 1-2% of the SQP algorithm (and in turn were approximately 8-10% worse than the direct optimal control).

Though these algorithms were close on many of the tested courses, as the number of gates increased for a course, the computation time increased faster for SQP than it did for PSO. This means that PSO is a better fit for large problems with many gates while SQP is a better fit for smaller problems that can be calculated quickly.

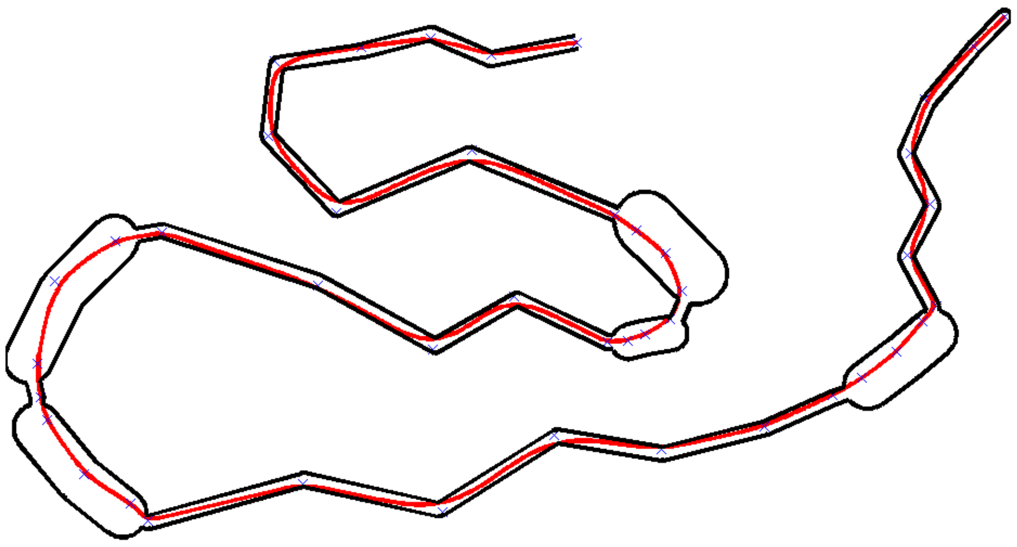


(a)

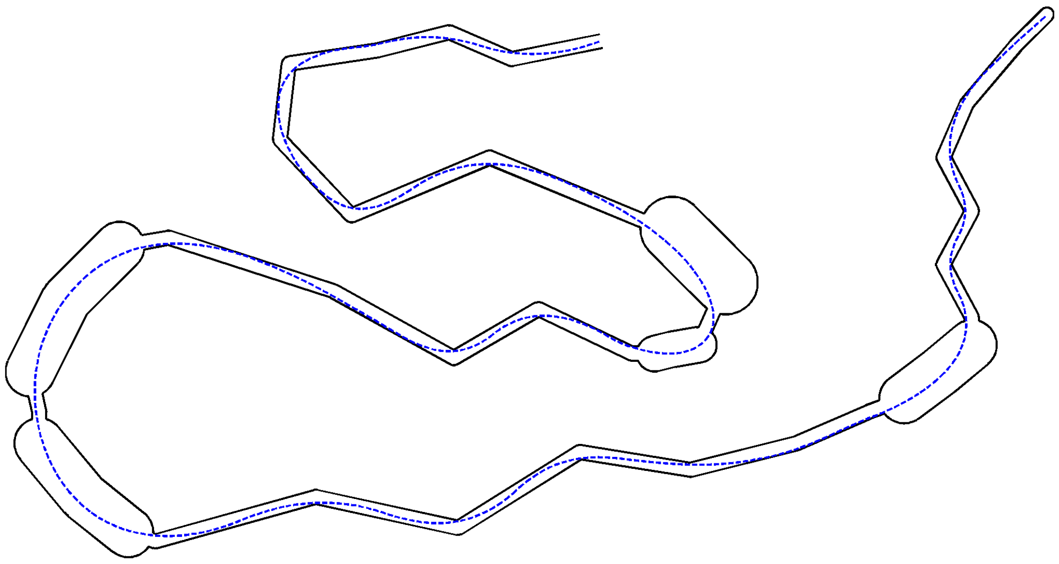


(b)

Figure 4.27: The results comparing the direct optimal control solution to the blended PSO and SQP solutions in both (a) computation and (b) final track times. It should be noted that the computation time for the direct optimal solution was too large to be shown in full.



(a)



(b)

Figure 4.28: The results of the a) blended path algorithm and b) direct optimal control algorithm, on Challenge Course 1. The course and gates were hand encoded based on the picture provided by Equipe Racing.

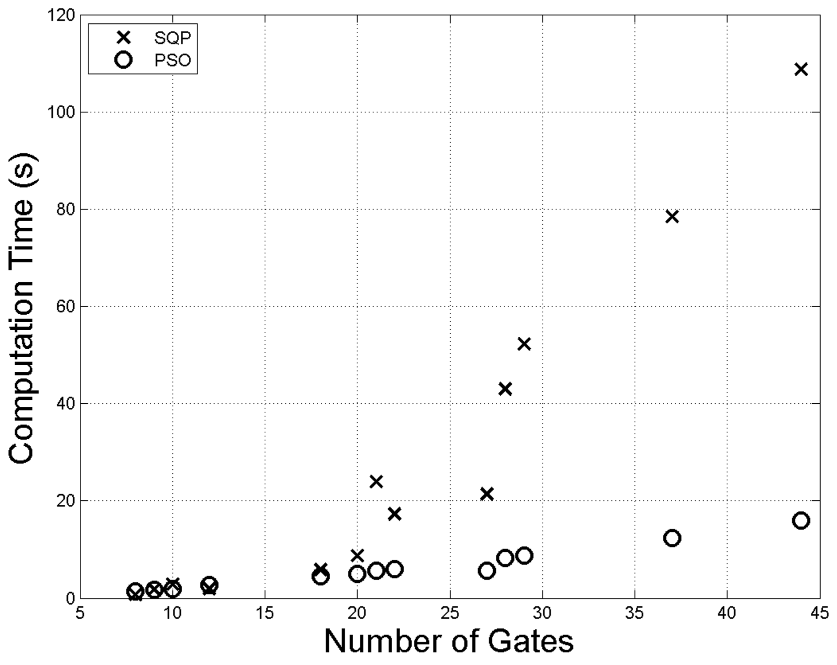


Figure 4.29: A direct comparison of the number of gates to the computation time of both the SQP and PSO algorithms.

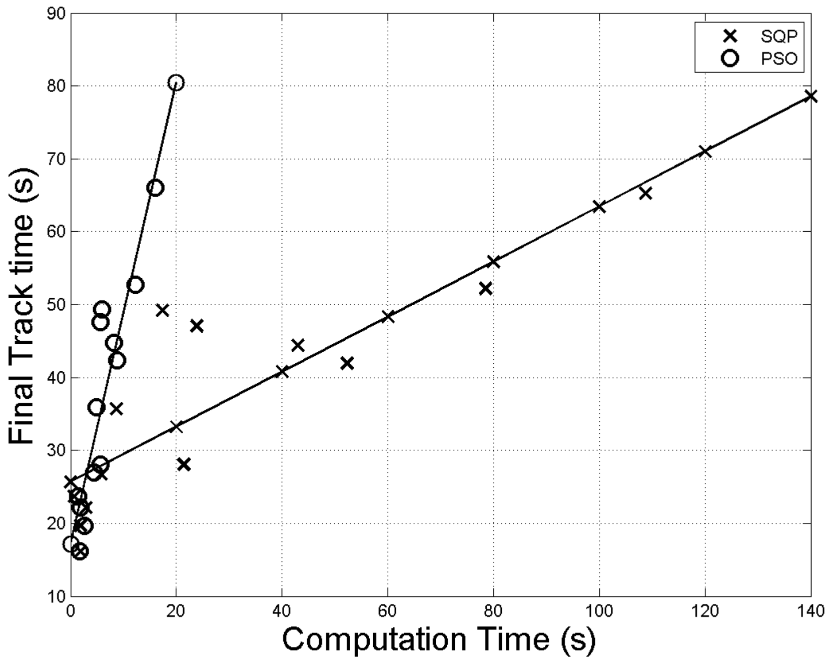


Figure 4.30: A comparison of the final track time to the computation time. A line through each of the different points has been added to show how well each of the algorithms will perform as time increases. Although for some of the courses PSO does not return the global minimum. The fact that it's computation time is substantially faster than SQP is a good sign.

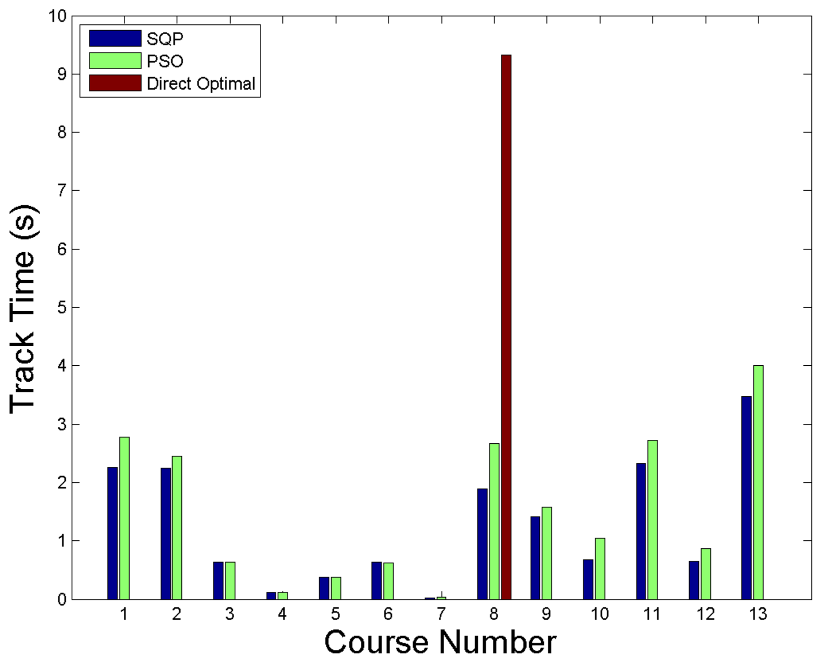


Figure 4.31: A bar graph of the direct optimal, PSO and SQP final track time decreases when compared to the MMC and shortest paths. This graph shows the absolute value of the final track time, therefore taller bars indicate a faster track time. Note that the direct optimal control path was only run for track 8, the Challenge Course 1.

Chapter 5

Summary

5.1 Conclusion

This thesis compared three path generating algorithms: a direct optimal control path, and a computationally efficient but sub-optimal path planning algorithm that blended two paths using either particle swarm optimization (PSO), or sequential quadratic programming (SQP). The paths generated by these algorithms were directly compared on several standard race manuvers and multiple autocross courses. Each of the algorithms were compared based on the computation time of the path, and the final track time of the generated path. It was expected that the direct optimal path would produce a shorter track time, at the expense of a large computation time. This direct optimal solution was used to modify and improve the computationally efficient algorithm by adding additional gates to enable a wider smoother path through turns.

An autocross course is generally defined by a set of cones on a large flat(parking lot) surface. Based on the cone placement a set of gates and corresponding corridors are chosen such that when a vehicle traverses each of the gates while staying within the given corridors in order it will have successfully completed the course. These gates are used to plan a final path through the race course. The direct optimal control problem utilizes these gates to initialize a multi-phase problem where the cost function (final track time) is minimized.

The computationally-efficient method initially plans two straight line primitive paths through the course. A computationally-efficient algorithm was presented that built upon the work of Choi and Feibich by generating two paths through a course: one path minimizes the maximum curvature (MMC), while the second path is the shortest distance path. The computationally-efficient algorithm blends the two paths to generate a sudo-optimal path through the course. Which is than smoothed using fourth order beizer curves.

Two different blending algorithm were compared using the the computationally-efficient blending method. The first method is SQP method which is an iterative method similar to Newton's method. The second method is a PSO method, which is a multiple shooting method. These two algorithms differ in the types of problems they are generally used to solve, and therefore demonstrated large difference in the solutions returned.

The SQP method generally returned a solution with a shorter track time when compared to the PSO method. However, the computation time when compared to PSO was exponential as the number of gates increased. On large courses (25+ gates) the algorithm took over a minute to converge on a solution. Compared to the SQP method which returned track times within 1-2% of the SQP method with a computation time that remained linear as the number of gates increased.

Comparisons between the direct optimal control path and blended path show comparable performance to the optimal course on short course manuevers. The final track times on the Chicane, Slalom, and hairpin turns are within 10% of the optimal path, after a choice of optimal straight line gate positions. The blended algorithm also had an improvement of up to 60% when compared to the origional non-blended solution.

In one specific example of an actual autocross course (Challenge Course 1), the two blended solutions were within 14% of the direct optimal solution (placing both within the top 10 of the human drivers on the same course). However the computation time for the blended solution (8-70 seconds) was far less than that of the optimal solution (7 hours). This increased computational efficiency allows for the blended algorithms to be used when the direct optimal solution is simply unfeasible because of the computation time required.

5.2 Future Work

The blended algorithm has proven to successfully navigate several different autocross courses, in track times that approach, though worse, are comparable to the optimal solution. Planned future work seeks to extend the algorithms capabilities by determining an analytical solution to a 4th order Bézier curve, improving the vehicle model, and modifying the alogrithm to enter and exit maneuvers with a large angle. At the same time the algorithm can be extended to assist course designers in developing fair/fast/fun autocross courses, and by extending the test platform of the vehicle.

1. Section 3.3.3 examined the differences between the 2nd order and 4th order Bézier curves used to smooth the final path. An analytical solution to the minimization of the curvature of a 2nd order Bézier curve was found in [8]. However a solution to the 4th order Bézier curve has not yet been calculated. Instead the 4th order Bézier control points are simply chosen based on the optimal 2nd order control points. Although was shown that the 2nd and 4th order Bézier curves matched closely. The optimum minimal curvature path has not yet been solved. By utilizing a true 4th order optimal path a faster course time can be achieved.

2. This research uses a point mass vehicle model to calculate the vehicles state and control variables. The point mass solution provides many inherit benefits

including faster computation time and simplicity. However [31] showed that the point mass model can cause loss of controllability as seen when the vehicle velocity approaches zero. Other vehicle models exist including a half car model, and a full seven degree of freedom model (see Section 1.3). These higher order models do not suffer from this instability at low speeds, as shown in [32]. By switching to a higher order model the vehicle constraints could be loosened. The benefits from this improvement are unknown and should be explored.

3. The comparison of the direct optimal path and the blended path showed that the blended path was unable to enter and exit maneuvers at wide angles. This leads to the blended solution having a smaller velocity when enter and exiting curves. Currently the method for handling straight segments is to drive in a straight line through them. It may be beneficial to traverse the straight cells along a wide curve such that the vehicle enters and exits the maneuver with a large velocity.

4. Autocross racing differs from many other styles of car racing such as Formula One or NASCAR, because the performance of the vehicle should be negligible. In the perfect autocross course the type of vehicle being driven should not matter when compared to the skill level of the driver. Many autocross course designers desire a method of quickly determining the final tracks time for multiple different vehicles. For example, a straight section that allowed some cars to travel upwards of 60 mph does not entirely reflect a drivers skill. While at same time course designers want to guarantee that their courses will be exciting, by limiting the number of turns that force the car to slow to less than 10 km/h. By changing the output of the path planning algorithm to detect points on a course that are unwanted the course designers can guarantee that their course is exciting and fast for the drivers.

5. Comparing the abilities of the blended path algorithm to that of other research institutes and companies can allow an opportunity to recognize previously unseen optimizations. For this purpose "The Open Racing Car Simulator"

(TORCS). TORCS provides an open platform to test path planning and driver models. Many researchers profile their path planning algorithms during one of the many events hosted by TORCS [25]. By linking the path planning algorithm to TORCS previously unseen opportunities for improvement may be gained. TORCS can be run by converting the code to C++ code and importing the TORCS libraries. The process of converting the code from Matlab to C++ has already been started with the PSO blending method.

Appendix A

Generating A Velocity Profile

A method for calculating the maximum velocity profile through a given path was proposed in [22]. The proposed method was able to quickly generate an achievable velocity at all points along a discretized path. The generated velocity profile does not violate the the acceleration limits of the vehicle and instead determines a velocity profile such that the vehicles acceleration are always along the edge of the acceleration constraint as seen in Fig. A.1. For completeness, the steps of that algorithm are outlined again here:

1. Discretize the path Γ into a finite number, K, of closely spaced points.

2. Calculate the curvature, κ, at each point, i, using Eq. A.2. Where the path Γ is composed of the points $x(i), y(i)$ where $i = 1, 2, \ldots, K$. Therefore:

$$\begin{aligned} \dot{x}(i) &= \frac{dx(i)}{dt} \\ \dot{y}(i) &= \frac{dy(i)}{dt} \\ \ddot{x}(i) &= \frac{d^2x(i)}{dt} \\ \ddot{y}(i) &= \frac{d^2y(i)}{dt} \end{aligned} \tag{A.1}$$

$$\kappa(i) = \frac{\dot{x}(i)\ddot{y}(i) - \dot{y}(i)\ddot{x}(i)}{(\dot{x}(i)^2 + \dot{y}(i)^2)^{\frac{3}{2}}} \qquad i = 1, 2, \ldots, K \tag{A.2}$$

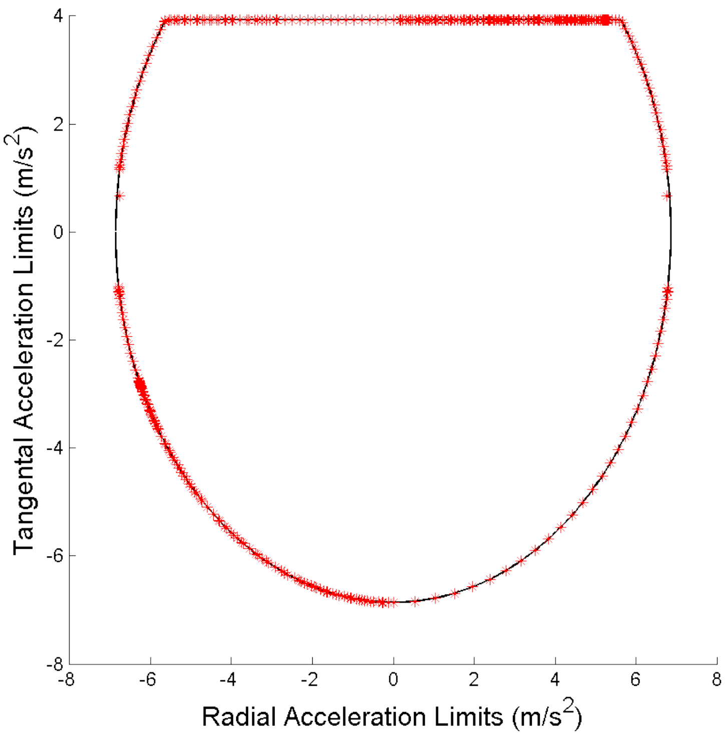


Figure A.1: The acceleration constraints of a vehicle generated as an ellipse with a cut off top. Along with the resultant acceleration point, (a_r, a_t) marked in red stars, based on a generated maximum velocity profile.

3. Based on the curvature, calculate the local extrema points along the curve. These are the points of maximum curvature along the path as can be seen in Fig. A.2. In these points the turning radius reaches a local minimum.

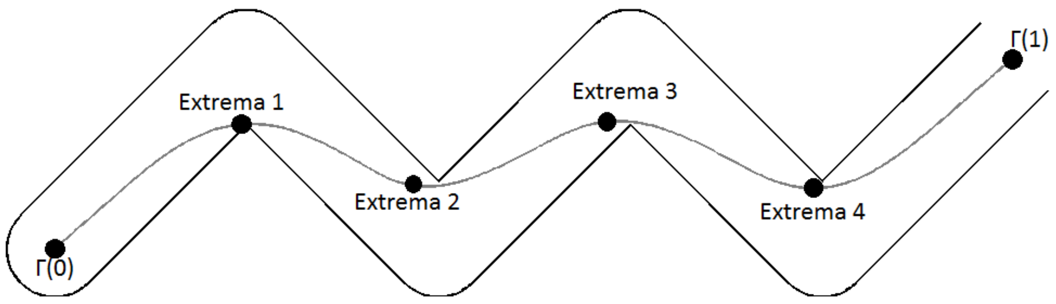


Figure A.2: An optimal path through a race course, Γ, stating at $\Gamma(0)$ and ending at $\Gamma(1)$. All the extrema points are marked along the path, locally these points indicate spaces where velocity is at a minimum. These places are where the velocity profile calculations start.

4. To achieve a maximum velocity path at the extrema points the tangential acceleration should be 0, and the radial acceleration should be at a maximum. This allows the car to achieve the highest velocity at the extrema point. The maximum acceleration at this point can be calculated by the maximum radial acceleration and the curvature at that point:

$$V_{max}(i) = \frac{a_{r_{max}}}{\kappa(i)} \tag{A.3}$$

5. Before and after a point of maximum curvature, the vehicle can move faster because the curvature at these adjacent points is smaller. In order to maintain the highest allowable overall velocity profile the adjacent velocities must be determined as maximally allowed by the acceleration constraints.

$$V_{i+1} = \min\left(V_{max}(i+1) \quad , \quad \frac{a_{t_i}\Delta S_i}{V_i} + V_i\right)$$

$$a_{r_{i+1}} = \kappa_i V_{i+1}^2$$

$$a_{t_{i+1}} = \min\left(a_{t_{max}} \quad , \quad a_{t_{min}}\sqrt{1 - \frac{a_{r_{i+1}}}{a_{r_{max}}}^2}\right)$$

   Where S_i is the distance between point i and $i+1$.

6. As the velocity profile is calculated both forwards and backwards from each point of maximum curvature, the velocity will cease to increase. This is because the velocity at those points is so high that the radial acceleration is at its limit. At this point the velocity profile for that point of maximum curvature can no longer be calculated, due to a neighboring point of maximum curvature, which in turn requires a lower speed at the original point so its lower limit must be observed.

7. A similar velocity profile is calculated at the start point for the given initial velocity.

8. As shown in Fig. A.3, the highest allowable velocity profile is the minimum of all possible velocity profiles.

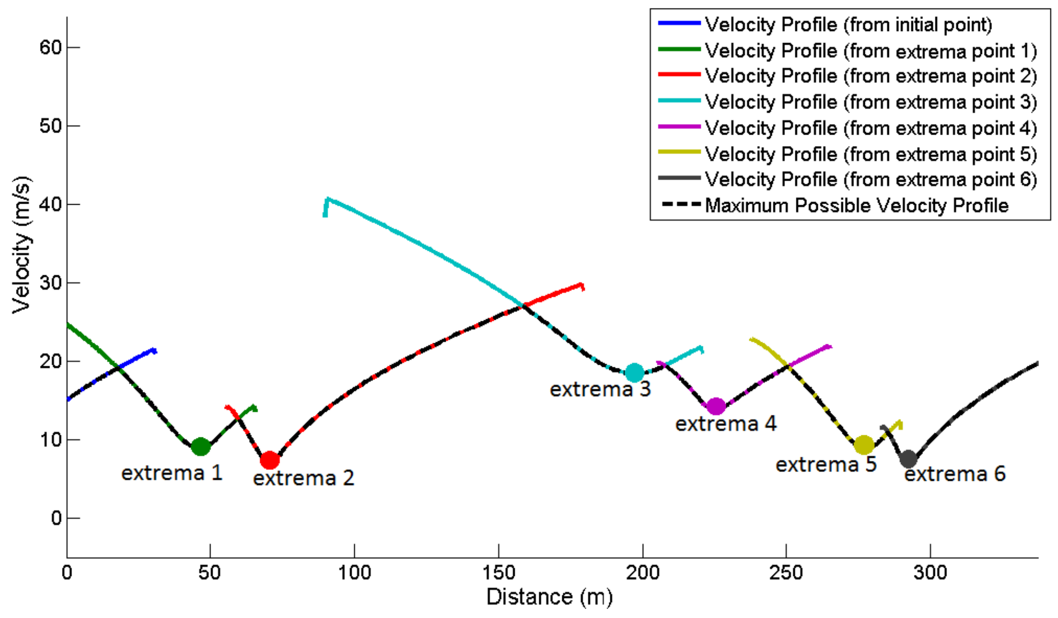


Figure A.3: A velocity profile from an example course containing multiple turns. The minimum of the combination of the velocity profiles from each course segment is shown as the dashed-black line.

9. For the given highest allowable velocity profile the cost function (time) as the total time over each segment, as calculated by the segment length divided by the velocity achieved during that segment:

$$t_f = \sum_{i=1}^{K} \frac{\Delta S_(i)}{V(i)} \tag{A.4}$$

Bibliography

[1] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[2] F Braghin, F Cheli, S Melzi, and E Sabbioni. Race driver model. *Computers & Structures*, 86(13):1503–1516, 2008.

[3] Luigi Cardamone, Daniele Loiacono, Pier Luca Lanzi, and Alessandro Pietro Bardelli. Searching for the optimal racing line using genetic algorithms. In *CIG*, pages 388–394, 2010.

[4] D. Casanova. *On Minimum Time Vehicle Manoeuvring: The Theoretical Optimal Lap*. PhD thesis, Cranfield University, November 2000.

[5] Raja Chatila and Jean-Paul Laumond. Position referencing and consistent world modeling for mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 138–145. IEEE, 1985.

[6] Ji-Wung Choi. *Real-Time Obstacle Avoiding Motion Planning For Autonmous Ground Vehicles*. PhD thesis, University of California Santa Cruz, December 2010.

[7] Ji-wung Choi, Renwick Curry, and Gabriel Elkaim. Path planning based on bézier curve for autonomous ground vehicles. In *World Congress on Engineering and Computer Science 2008, WCECS'08. Advances in Electrical and Electronics Engineering-IAENG Special Edition of the*, pages 158–166. IEEE, 2008.

[8] Ji-wung Choi, Renwick E Curry, and Gabriel H Elkaim. Minimizing the maximum curvature of quadratic bézier curves with a tetragonal concave polygonal boundary constraint. *Computer-Aided Design*, 44(4):311–319, 2012.

[9] JW Choi, Renwick E Curry, and Gabriel Hugh Elkaim. Continuous curvature path generation based on bezier curves for autonomous vehicles. *IAENG International Journal of Applied Mathematics*, 40(2):91–101, 2010.

[10] Andy Cost. Equipe rapide sports car club, 2014.

[11] Lester 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*, pages 497–516, 1957.

[12] Russ C Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[13] Rida T Farouki. The bernstein polynomial basis: a centennial retrospective. *Computer Aided Geometric Design*, 29(6):379–419, 2012.

[14] Reece Fiebich. Bézier-based trajectory generation for autonomous ground vehicles. Masters thesis, University of California Santa Cruz.

[15] Jeong hwan Jeon, Raghvendra V Cowlagi, Steven C Peters, Sertac Karaman, Emilio Frazzoli, Panagiotis Tsiotras, and Karl Iagnemma. Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. In *American Control Conference (ACC), 2013*, pages 188–193. IEEE, 2013.

[16] KG Jolly, R Sreerama Kumar, and R Vijayakumar. A bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robotics and Autonomous Systems*, 57(1):23–33, 2009.

[17] DP Kelly and RS Sharp. Time-optimal control of the race car: a numerical method to emulate the ideal driver. *Vehicle System Dynamics*, 48(12):1461–1474, 2010.

[18] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[19] R Krtolica and D Hrovat. Optimal active suspension control based on a half-car model. In *Decision and Control, 1990., Proceedings of the 29th IEEE Conference on*, pages 2238–2243. IEEE, 1990.

[20] Eckhard Kruse, Ralf Gutsche, and Friedrich M Wahl. Estimation of collision probabilities in dynamic environments for path planning with minimum collision probability. In *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 3, pages 1288–1295. IEEE, 1996.

[21] Jae-Won Lee, Wook Hyun Kwon, and Jinhoon Choi. On stability of constrained receding horizon control with finite terminal weighting matrix. *Automatica*, 34(12):1607–1612, 1998.

[22] Marko Lepetič, Gregor Klančar, Igor Škrjanc, Drago Matko, and Boštjan Potočnik. Time optimal path planning considering acceleration limits. *Robotics and Autonomous Systems*, 45(3):199–210, 2003.

[23] DJN Limebeer, G Perantoni, and AV Rao. Optimal control of formula one car energy recovery systems. *International Journal of Control*, 87(10):2065–2080, 2014.

[24] Thomas Lipp and Stephen Boyd. Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6):1297–1311, 2014.

[25] Daniele Loiacono, Julian Togelius, Pier Luca Lanzi, Leonard Kinnaird-Heether, Simon M Lucas, Matt Simmerson, Diego Perez, Robert G Reynolds, and Yago Saez. The wcci 2008 simulated car racing competition. In *Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On*, pages 119–126. IEEE, 2008.

[26] Toshiyuki Ohtsuka. A continuation/gmres method for fast computation of nonlinear receding horizon control. *Automatica*, 40(4):563–574, 2004.

[27] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[28] James B Rawlings and Kenneth R Muske. The stability of constrained receding horizon control. *Automatic Control, IEEE Transactions on*, 38(10):1512–1516, 1993.

[29] Sukn-Hwan Suh and Kang G Shin. A variational dynamic programming approach to robot-path planning with a distance-safety criterion. *Robotics and Automation, IEEE Journal of*, 4(3):334–349, 1988.

[30] Julian P Timings and David J Cole. Minimum maneuver time calculation using convex optimization. *Journal of Dynamic Systems, Measurement, and Control*, 135(3):031015, 2013.

[31] E Velenis and P Tsiotras. Optimal velocity profile generation for given acceleration limits; the half-car model case. In *2005 IEEE International Symposium on Industrial Electronics*, pages 355–360, 2005.

[32] Efstathios Velenis and Panagiotis Tsiotras. Optimal velocity profile generation for given acceleration limits: Theoretical analysis. *system*, 2:5, 2005.

[33] RB Wilson. *A simplicial method for convex programming.* PhD thesis, PhD thesis, Harvard University, 1963.