Trajectory Optimization with Dynamic Obstacles Avoidance

Philippe Weingertner and Minnie Ho
pweinger@stanford.edu minnieho@stanford.edu

Abstract

We study the problem of Trajectory Optimization for Autonomous Driving where we derive a motion plan on a pre-defined path. The challenge is to avoid ten vehicles crossing our path, optimizing efficiency and comfort, while making decisions in real-time. As part of a Model Predictive Control (MPC) setup, we propose a Collision Avoidance model based on an Elastic Model handling disjunctive constraints. We investigate different optimization algorithms: Interior Point methods and adaptation of a simplex algorithm to a problem initially defined over a quadratic cost function. We provide reference implementations of the MPC setup, Collision Avoidance Model and a fully custom solver. Finally we benchmark and demonstrate the efficiency of our collision avoidance model, both in terms of accuracy and real-time performances, on a serie of 100 randomly generated tests.

1 Introduction

This project investigates trajectory optimization [6] in the presence of obstacles [19, 3, 9, 20]. One such application for this class of problems is that of autonomous driving, where we have an ego vehicle and dynamic obstacles (vehicles, pedestrians) which may intersect our desired trajectory and which we wish to avoid using motion planning and control.

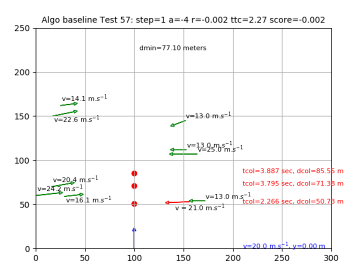


Figure 1: Collision Avoidance Tests Setup

Trajectory optimization minimize a cost function which takes into account start and terminal states, as well as cost along the trajectory path. The design space is subject to constraints on the states and control input at sampled time points.

2 Related Work

A detailed survey of Motion Planning techniques for Autonomous Driving is provided in [16, 9]. In order to enable real-time applicability, the problem is typically decomposed in a two-stage process.

First a path is planned and then a Motion Plan is derived over this pre-defined path (or over a set of pre-defined paths). This strategy has been in use since the DARPA challenge [19, 4, 18, 7]. Path planning typically relies on search algorithms (A*, D*, RRT* ...) [8, 15] while the derivation of a Motion Plan over a Path typically relies on either pre-computed velocity profiles or solving online an Optimal Control problem. While MPC is well established for pure trajectory tracking [12], assuming there exists a collision-free trajectory we want to follow, it can also be used to derive a collision-free trajectory. In the first case, a more complex and non linear vehicle dynamics model is used, while in the second case, a linear approximation of the dynamics is used, to quickly come up with a collision-free trajectory proposal. This collision-free trajectory is then further refined in terms of comfort and vehicle dynamics feasibility.

We focus on the Motion Planning problem where a path is pre-defined and we want to derive a collision-free trajectory considering a linear approximation of the vehicle dynamics. There are several publications on this subject [13, 3, 17], investigating the use of MPC to derive a collision-free velocity profile. But they assume that a higher level planner has already decided whether we have to proceed or yield the way w.r.t. any other vehicle crossing our path. While this may be a reasonable strategy when dealing with a single or two objects crossing our path, we would like to scale to multiple crossing objects. We would like to decide individually and optimally w.r.t. every crossing object, wether we should proceed or yield the way. This should be part of the optimization problem.

3 Problem Formulation

We define a MPC problem over 20 time steps of 250 ms each with a Quadratic Cost function with $x \in \mathbb{R}^{60}$ and 160 constraints. We have 120 linear and nonlinear ($\|x_{\text{ego}} - x_{\text{obj}}\| \geq d_{\text{saf}}$) inequality constraints and 40 linear equality constraints (Dynamics Model).

$$\text{subject to} \begin{cases} x_{k,\text{min}} \leq x_k \leq x_{k,\max} \\ u_{k,\text{min}} \leq u_k \leq u_{k,\max} \\ x_{k+1} = A_d x_k + B_d u_k \\ x_0 = x_{\text{init}} \\ \forall \left(t_{\text{col}}, s_{\text{col}}\right)_{i \in [1,10]} \qquad x_{t_{\text{col}}^{(i)}}\left[1\right] < s_{\text{col}}^{(i)} - \Delta_{\text{safety}} \text{ or } x_{t_{\text{col}}^{(i)}}\left[1\right] > s_{\text{col}}^{(i)} + \Delta_{\text{safety}} \end{cases}$$

We wan't to avoid up to 10 vehicles crossing our path. Spatio-temporal collision points $(t_{\text{col}}, s_{\text{col}})$ are defined with an associated uncertainty area $t_{\text{col}} \pm 250\ ms,\ s_{\text{col}} \pm \Delta_{\text{safety}}$

We use linear dynamics model: with Constant Acceleration in between 2 time steps

$$\begin{bmatrix} s \\ \dot{s} \end{bmatrix}_{k+1} = A_d \begin{bmatrix} s \\ \dot{s} \end{bmatrix}_k + B_d \left[\ddot{s}\right]_k \text{ with } A_d = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, B_d = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}$$

4 Methods

4.1 Collision Avoidance Model

We reformulate the collision avoidance model and later on demonstrate the improvements it provides in a set of benchmarks. An implementation of this model is available in mpc_mip.jl.

4.1.1 Disjunctive Constraints

In general the collision avoidance constraint is defined as $\left\|\text{pos}_{\text{ego}} - \text{pos}_{\text{obj}}\right\|_2 \geq \text{d}_{\text{safety}}$

When considering the evolution of an ego vehicle along a path denoted by $s\left(t\right)$ and a crossing-point for some other vehicle, at $(t_{\text{cross}}, s_{\text{cross}})$, the collision avoidance constraint is reformulated as: $|s\left(t_{\text{cross}}\right) - s_{\text{cross}}| \geq \text{d}_{\text{safety}}$. Which is equivalent to a disjunctive constraint:

$$s\left(t_{\text{cross}}\right) \leq s_{\text{cross}} - d_{\text{safety}} \quad \vee \quad s\left(t_{\text{cross}}\right) \geq s_{\text{cross}} + d_{\text{safety}}$$

In practice the fundamental question we should anwser is wether we should proceed or yield the way w.r.t. this other vehicle. To handle this disjunctive constraint, we introduce a binary slack variable such that the OR constraint is replaced by an AND constraint

$$s\left(t_{\text{cross}}\right) \leq s_{\text{cross}} - d_{\text{safety}} + My \quad \wedge \quad s_{\text{cross}} + d_{\text{safety}} \leq s\left(t_{\text{cross}}\right) + M\left(1-y\right)$$

with $y \in \{0,1\}$ and $M \in \mathbb{R}^{+}$ some large value s.t. when y=1 the constraint is always true

This way, even if we have defined two constraints via a AND, which is required to apply optimization algorithms like Interior Point Methods or Simplex, only one or the other constraint will be active: the other one being always true. By using a binary slack variable, we have to use a Mixed Integer Programming solver.

This problem reformulation corresponds to the Big-M reformulation of disjunctive constraints.

4.1.2 Elastic Model

We would like to have a convex formulation of the problem such that we can find as quickly as possible a guaranteed global minimum. The problem is that when defining a problem with such a collision avoidance constraint

$$\min_{x} \quad Q_{\text{uadratic}}\left(x\right)$$
$$\text{s.t. } a^T x \leq b \text{ (safety distance constraint)}$$

This might be causing infeasibility. In practice there may be no dynamically feasible motion plan to maintain a pre-defined safety distance. But we want to reveal by how much the constraint needs to be relaxed in order to become dynamically feasible. We are looking for a Motion Plan that is dynamically feasible and which violates at minimum our desired safety distance. In order to reveal this value, we introduce another slack variable, per collision avoidance constraint , such that the problem becomes:

$$\min_{x} \quad Q_{\text{uadratic}}\left(x\right) + y$$
$$\text{s.t. } a^T x \leq b\text{+y (safety distance constraint)}$$
$$\text{elastic slack variable: } y \in \mathbb{R}$$

If we do not use such an elastic slack variable, a convex solver would return an infeasibility verdict and Interior Point Methods would fail.

4.2 Optimization Algorithms

We use the following generic optimization formulation and notations:

$$\begin{array}{cc} \min\limits_{\mathbf{x}\in\mathbb{R}^n} & f\left(\mathbf{x}\right) \in \mathbb{R} \\ \text{subject to} & \mathbf{g}\left(\mathbf{x}\right) \leq \mathbf{0} \in \mathbb{R}^k \\ & \mathbf{h}\left(\mathbf{x}\right) = \mathbf{0} \in \mathbb{R}^m \end{array}$$

4.2.1 Penalty Methods

We first consider Penalty and Augmented Lagrangian methods [10] which transform the constrained problem into an unconstrained one via the use of the following penalty methods:

$$p_{\text{quadratic}}\left(x\right) = \sum_i \max\left(g_i\left(x\right), 0\right)^2 + \sum_j h_j\left(x\right) \quad p_{\text{Lagrange}}\left(x\right) = \frac{1}{2}\rho \sum_i h_i\left(x\right)^2 - \sum_i \lambda_i h_i\left(x\right)$$

The issue with these methods is that even if they may end up approaching a minimum, it may be from an infeasible region. This is why Interior Point Methods are usually preferred.

4.2.2 Interior Point Method with Inequality and Equality Constraints

We build upon the work that was done for AA222 Project 2. An Interior Point Method based on a Quasi-Newton method, BFGS, with backtracking Line search was implemented. But here we deal with equality constraints, of the form $Ax = b$ (corresponding mainly to our Vehicle Dynamics Model), in addition to inequality constraints. We modify the way we compute the search direction. At every step, we do a second order approximation of our minimization function. We express the Lagrangian $\mathcal{L}(x, \lambda)$ and solve for $\nabla_x \mathcal{L} = 0$. The solution of the resulting system of equations provides the new search direction: $d = \Delta x_{\text{newton\_step}}$, everything else being unchanged.

$$\underset{\text{subject to}}{\min} \begin{cases} \hat{f}(x+v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v & \text{Second order Taylor approximation} \\ A(x+v) = b \end{cases}$$

Via optimality conditions on $\mathcal{L}(x, \lambda)$ we get: $\begin{bmatrix} \Delta x_{\text{newton\_step}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\nabla f(x) \\ -(Ax - b) \end{bmatrix}$

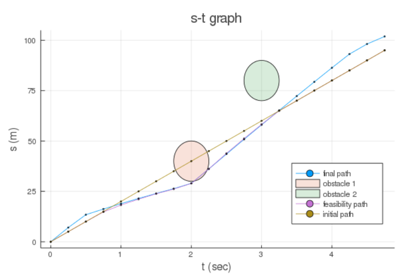
The details of the derivatoin can be found in [2] chapter 10 and our implementation in optimize.jl

4.2.3 Simplex Algorithm

Simplex is fast. We investigate how to bootstrap the feasibility search phase of an Interior Point method with a simplex algorithm.

4.3 Optimization under Uncertainty

We handle uncertainty as per the set-based minimax approach described in [10] chapter 17. We are dealing with imperfect observations of the surrounding vehicles and with even more uncertain driving models. As a consequence, the predicted crossing point $(t_{\text{cross}}, s_{\text{cross}})$ is uncertain. This uncertainty is represented by a random variable z and the crossing vehicle can be at any location within an uncertainty area represented as a circle in the (s, t) domain; the shadow area in the ST graph (longitudinal position S along the path vs Time).



We try to avoid the whole uncertainty area, complying to the minimax approach $\underset{x \in \mathcal{X}}{\min} \; \underset{z \in \mathcal{Z}}{\max} \; f(x, z)$. But if we can not avoid the full uncertainty area, we will remain as far as possible from its center, via the Elastic Collision Avoidance Model described previously. This Elastic Model strictly enforces dynamics constraints but relaxes the safety distance as little as possible.

5 Experiments

The gihub repo is AA222-project.

5.1 Solvers Benchmarks

We benchmark different solvers including our own implementation in optimize.jl. We check:

- Constraints compliance: we have defined a set of vehicle dynamics constraints and collision avoidance constraints.

- Objective value: the lower this value, the more optimal we are in terms of efficiency (maintaining a desired speed) and comfort (minimizing jerk)
- Runtime: as we are dealing with planning steps of 250 ms, we need to have a runtime below 250 ms for real time applicability as a bare minimum. But in reality we may have to react in less than 40 ms in case of emergency.

We first check our own implementation: Interior Point method vs Penalty method. As expected the Interior Point methods outperforms the penalty methods by ensuring feasibility.

We get with the Interior Point method:

```
OPTIMIZE.jl INTERIOR POINT METHOD
max   equality constraint violation:(1.256239e-11, 4) out of 40
max inequality constraint violation:(-0.000311, 120) out of 122
Pass: optimize returns a feasible solution on 2/2 random seeds.
```

While with the Penalty method we get:

```
OPTIMIZE.jl PENALTY METHOD
max   equality constraint violation:(0.06176, 17) out of 40
max inequality constraint violation:(3.2979, 120) out of 122
Fail: optimize returns a feasible solution on 0/2 random seeds.
```

We then benchmark a set of open source solvers like ECOS and SCS with a set of commercial solvers MOSEK and CPLEX and our own Interior Point implementation optimize.jl (with maxcount set to 20K). They all provide feasible solutions which are very close, and derive same values for the slack variables. The main difference is the runtime: ECOS and MOSEK are the fastest (we also tested Gurobi but out trial license expired before writeup. But it performs similar to CPLEX). As MOSEK and CPLEX are the only solvers handling both Quadratic Cost functions and Mixed Integer Programming, we will use MOSEK for our final set of benchmarks instead of ECOS: to test our Collision Avoidance model in a serie of 100 tests.

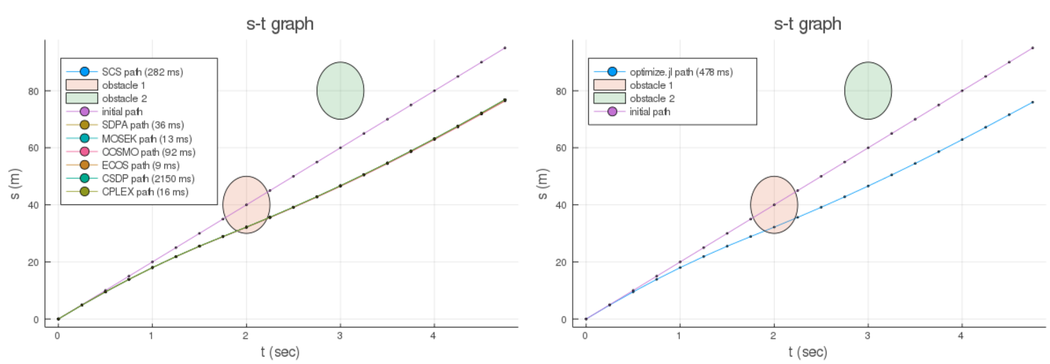


Figure 2: Solvers Runtime Benchmark

We then analyze further the runtime difference between our own implementation and MOSEK. We are using 160 BFGS iterations (40 during the feasibility search phase and 120 during interior point phase) while MOSEK just requires 20 iterations. We believe the difference is due to the fact that MOSEK implements a Primal-Dual Interior Point method instead of the raw Barrier Interior Point Method. It exhibits better than linear convergence and requires less iterations.

```
OPTIMIZE.jl INTERIOR POINT METHOD runtime=256 ms (maxcount set to 10K)
Iter 124 BFGS=1.62 ms: INV=0.63 ms, BT=0.38 ms, GRAD=0.40 ms
```

Also our BFGS iterations takes on average around 1.62 ms with:

- 1.62 ms spent doing a $100x100$ matrix inversion of our approximated Hessian. Which could be further improved with a direct Hessian inverse approximation.
- 0.38 ms in Backtracking line search.

- 0.40 ms in approximating Gradient via Finite Differences. By using a Complex Step approach we could probably also further improve this step.

As a conclusion, we will use MOSEK for our next set of experiments.

5.2 Anti Collision Tests Benchmarks

We use five metrics to evaluate the performance of our different approaches. (1) The main success metric is the percentage of cases where we reach a target state without collision. (2) The second metric is the agent runtime. (3) The third metric is a comfort metric: the number of hard braking decisions. (4) The fourth metric relates to efficiency: how fast we reach a target while complying to some speed limitation. (5) The last metric is a safety metric: for some of our randomly generated test cases, a collision is unavoidable. In these cases, we aim for a lower speed at collision.

Mean	Success	Runtime	HardBrakes	Steps	Collision Speed
Baseline (Constant Speed)	22%	$< 1\mu s$	0.0	40.0	20.0 m.s^{-1}
MPC	79%	19 ms	3.09	39.5	18.5 m.s^{-1}
MPC_MIP	96%	22 ms	0.19	53.8	18.7 m.s^{-1}
Oracle (Dynamic Programming)	96%	22.3 sec	0.48	33.1	15.5 m.s^{-1}

Table 1: Results over 100 anti-collision tests

6 Conclusion

We came up with a detailed analysis of a Trajectory Optimization problem. We tackle the issue of Collision Avoidance of dynamic obstacles where real-time decisions are required. We propose a Collision Avoidance model based on an Elastic Model handling disjunctive constraints. This solution is based on Mixed Integer Programming with core optimization algorithms relying on primal-dual interior point methods. We also investigate how the Simplex algorithm could be adapted to solve a problem initially defined over a quadratic cost function. Finally we demonstrate the efficiency of our proposed model, operating over a continuous state space, with a runtime of 22 ms. While being 1000 x faster than our Oracle, a Dynamic Programming implementation performing an exhaustive search over a discretized solution space, it achieves the same collision avoidance success rate.

References

[1] John T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Cambridge University Press, USA, 2nd edition, 2009.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004. 4.2.2

[3] Haoyang Fan, Fan Zhu, Changchun Liu, Liangliang Zhang, Li Zhuang, Dong Li, Weicheng Zhu, Jiangtao Hu, Hongye Li, and Qi Kong. Baidu apollo em motion planner. *ArXiv*, abs/1807.08048, 2018. 1, 2

[4] Dave Ferguson, Thomas M. Howard, and Maxim Likhachev. Motion planning in urban environments. In *The DARPA Urban Challenge*, 2009. 2

[5] T. Gu, J. M. Dolan, and J. Lee. Runtime-bounded tunable motion planning for autonomous driving. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 1301–1306, 2016.

[6] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. In *Astrodynamics 1985*, pages 3–12, August 1986. 1

[7] Thomas Howard, Colin Green, Alonzo Kelly, and Dave Ferguson. State space sampling of feasible motions for high performance mobile robot navigation in complex environments. *J. Field Robotics*, 25:325–345, 06 2008. 2

[8] Sertac Karaman, Matthew Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt*. pages 1478–1483, 06 2011. 2

[9] Christos Katrakazas, Mohammed A. Quddus, Wen hua Chen, and Lipika Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. 2015. 1, 2

[10] Mykel J. Kochenderfer and Tim A. Wheeler. *Algorithms for Optimization*. The MIT Press, 2019. 4.2.1, 4.3

[11] X. Li, Z. Sun, Z. He, Q. Zhu, and D. Liu. A practical trajectory planning framework for autonomous ground vehicles driving in urban environments. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1160–1166, 2015.

[12] Fen Lin, Yuke Chen, You Zhao, and Shaobo Wang. Path tracking of autonomous vehicle based on adaptive model predictive control. *International Journal of Advanced Robotic Systems*, 16:172988141988008, 09 2019. 2

[13] C. Liu, W. Zhan, and M. Tomizuka. Speed profile planning in dynamic environments via temporal optimization. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 154–159, 2017. 2

[14] Changliu Liu, Chung-Yen Lin, and Masayoshi Tomizuka. The convex feasible set algorithm for real time optimization in motion planning. *SIAM J. Control and Optimization*, 56:2712–2733, 2017.

[15] M. McNaughton, C. Urmson, J. M. Dolan, and J. Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *2011 IEEE International Conference on Robotics and Automation*, pages 4889–4895, May 2011. 2

[16] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, March 2016. 2

[17] Tianyu Gu, J. Atwood, Chiyu Dong, J. M. Dolan, and Jin-Woo Lee. Tunable and stable real-time trajectory planning for urban autonomous driving. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 250–256, 2015. 2

[18] Chris Urmson, Christopher Baker, John Dolan, Paul Rybski, Bryan Salesky, William Whittaker, Dave Ferguson, and Michael Darms. Autonomous driving in traffic: Boss and the urban challenge. *AI Magazine*, 30:17–28, 06 2009. 2

[19] Andreas Wachter and Lorenz Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 03 2006. 1, 2

[20] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Optimization-based collision avoidance. *ArXiv*, abs/1711.03449, 2020. 1