Trajectory Optimization for a Simple Model of an RC Car

Shane Rozen-Levy

April 30, 2018

1 Introduction

This report describes the application of trajectory optimization for a simple model of an RC car. The car featured Ackerman Steering controlled by a servo, and a DC motor for the thrust. We solved for the weighted sum of minimum jerk and minimum time trajectory for the RC car to turn around on a narrow road and for the car to parallel park. The trajectories were solved using trapezoidal direct collocation in MATLAB with fmincon. This project featured a dynamics model not used in class, path constraints not used in class, and an objective function not used in class.

2 The Problem

2.1 Dynamics

The dynamics for this system are a modified version of dynamics presented Eric Monet's thesis [1]. I simplified the model of the motor so the force is set by the control and I modeled the servo as a first order system. I added a chain integrator to both F_D, the force from the motor, and $\phi_{desired}$, the desired steering angle, so that the control input can be zero at the start and end of the trajectories. This allows for a smooth transition to steady state for the trajectories. Equation 1 contains the definition for the state vector. Equation 2 contains a general definition of the dynamics. Figure 1 contains the general coordinates of the RC car.

$$\vec{z} = [x, y, \theta, v_u, F_d, \phi, \phi_{desired}] \tag{1}$$

$$\dot{\vec{z}} = f(t, \vec{z}, \vec{u}) \tag{2}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v_u} \\ \dot{F_d} \\ \dot{\phi} \\ \dot{\phi}_{desired} \end{bmatrix} = \begin{bmatrix} [cos(\theta) - \frac{b*tan(\phi)}{l} sin(\theta)]v_u \\ [sin(\theta) + \frac{b*tan(\phi)}{l} cos(\theta)]v_u \\ \frac{tan(\phi)}{l} v_u \\ \frac{v_u(b^2m+J)tan\phi}{(cos\phi)^2[l^2m+(b^2m+J)(tan\phi)^2]}\dot{\phi} + \frac{l^2(cos\phi)^2}{(cos\phi)^2[l^2m+(b^2m+J)(tan\phi)^2]}F_D \\ U_1 \\ \frac{-\phi}{\tau} + \frac{\phi_{desired}}{\tau} \\ U_2 \end{bmatrix}$$

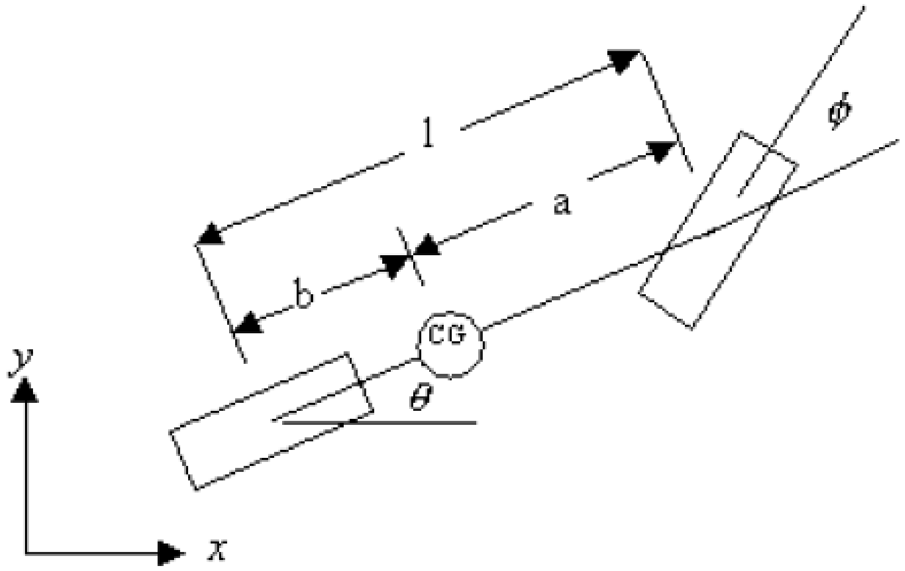

Figure 1: The general coordinates for the dynamics model [1].

2.2 Objective Function

The objective function is a combination of minimum jerk and minimum time. Since U_1 is the rate of change of the force, and the force is proportional to velocity. Then minimizing U_1 us roughly equivalent to minimizing jerk. β is the multiplier for steering angle. This helps to keep the steering angle smooth. α is the weighting factor for time. These terms combined result in a smooth trajectory that minimizes time and jerk for the RC car.

$$J = \int_0^T g(t, \vec{z}, \vec{u})dt$$
$$g(t, \vec{z}, \vec{u}) = U_1^2 + \beta * U_2^2 + \alpha$$

2.3 Constraints

The system has a few different constraints. The first set of constraints are limits on steering angle, desired steering angle, velocity and duration. The first three are limited between plus and minus a value. The duration is limited between 0 and ∞.

$$\vec{0} > \vec{L_1}(\vec{z})$$
$$\vec{L_1}(\vec{z}) = [\vec{z} - \vec{z_{max}}\,;\, \vec{z_{min}} - \vec{z}]$$

The next set of constraints are on the initial and final state. Both states have 0 steering angle, velocity, force, and desired steering angle. The x,y position and angle of the car are set based on the problem.

$$\vec{0} = \vec{h}(\vec{z}_0,\, \vec{z}_N)$$
$$\vec{h}(\vec{z}_0,\, \vec{z}_N) = [\vec{z}_0 - \vec{z_{start}}\,;\, \vec{z}_N - \vec{z_{end}}]$$

The final set of constraints are path constraints. The path constraints serve to keep some points on the car inside a smooth function. The edges of the road were modeled as a straight line. $LeftEdge < x < RightEdge$ and $BottomEdge < y < TopEdge$. For the more complicated problems you can define a boundary as $y > m(x)$ or $x > n(y)$. The first equation ensures all the points (x, y) are above the curve m. The second equation ensures all the points (x, y) are to the right of curve $n(y)$. These path constraints along with the limits on $\vec{z}$ form $\vec{L}(\vec{z})$.

2.4 Problem Definition

Problem Setup:

$$\begin{aligned}
&\text{minimize: } J = \int_0^T g(t, \vec{z}, \vec{u}) dt \\
&\text{subject to: } \vec{0} = \vec{h}(\vec{z}_0,\, \vec{z}_N) \\
&\text{and: } \vec{0} > \vec{L}(\vec{z}) \qquad\qquad \forall t \in [0, T] \\
&\text{dynamics: } \dot{\vec{z}} = f(t, \vec{z}, \vec{u})
\end{aligned}$$

3 Method

Both problems were turned into a nonlinear program using the trapezoidal collocation defined below. The resulting nonlinear program was solved using fmincon in MATLAB.

$$h = \frac{T}{N} \qquad\qquad t_k = k \cdot h$$

Decision Variables: $T, \vec{z}_0, \vec{z}_1, \ldots, \vec{z}_N, \quad \vec{u}_0, \vec{u}_1, \ldots, \vec{u}_N$

Objective Function: $J = \frac{h}{2} \cdot \sum_{k=0}^{N-1} \Big(g(t_k,\, \vec{z}_k,\, \vec{u}_k) \;+\; g(t_{k+1},\, \vec{z}_{k+1},\, \vec{u}_{k+1}) \Big)$

Boundary Constraints: $\vec{0} = \vec{h}(\vec{z}_0,\, \vec{z}_N)$

Path Constraints: $\vec{0} > \vec{L}(\vec{z}_k) \qquad\qquad k \in 0 \ldots (N)$

System Dynamics Constraints:

$$\vec{z}_{k+1} = \vec{z}_k + \frac{h}{2} \cdot \Big(\vec{f}(t_k,\, \vec{z}_k,\, \vec{u}_k) \;+\; \vec{f}(t_{k+1}, \vec{z_{k+1}},\, \vec{u}_{k+1}) \Big) \qquad\qquad k \in 0 \ldots (N-1)$$

For turning around Appendix A contains the parameters for the trajectory optimization. For parallel parking Appendix B contains the parameters for the trajectory optimization. Both problems involved feeding the output of one trajectory optimization as the initial guess for the next. The actual initial guess for both problems were set up the same. The control input was set to zero at all times. A spline was fit between the initial state, a middle state, and the final state. The middle state starts as the average of the initial state and the final state. θ is set to the angle of the initial position to the final position. ϕ is set to the change in θ from the middle state to the final state. The middle state helps to overcome the non-holonomic nature of the car by presenting an initial guess where for part of it, the car is driving in the correct direction.

For turning around we started off the optimization with time as a decision variable, and the car modeled as 10 points with some very large road dimensions. We then ran the optimization three times, slowly shrinking the road dimensions until we got to the desired road dimensions. We used the solution of one trajectory optimization as the initial guess for the next.

For parallel parking we started off the optimization with time as a decision variable, and not caring about the car intersecting with any objects. Then we ran the optimization with the car as a dot caring about the borders of the road and a spline modeling the parked cars. Next we ran the optimization where we checked 10 points along the border of the car for intersection with the borders of the road or the parked cars. Finally we ran the optimization with a smaller parking space. We ran the optimization with a smaller parking space because the optimization was tunneling through parts of the parked car. Once again we used the solution of one trajectory optimization as the initial guess for the next.

4 Results

We were able to come up with an optimal trajectory for both trajectory optimization problems. The trajectories were smooth and there was no tunneling in either solution. Appendix C contains the optimal trajectory for turning around. Appendix D contains the optimal trajectory for parallel parking. The video of the trajectories `https://youtu.be/jHWol7nHp68`.

References

[1] Moret, E. N., 2003. "Dynamic Modeling and Control of a Car-Like Robot". Thesis, Virginia Tech, Feb.

A Problem Parameters for Turning Around

Initial Position	
x (m)	0
y (m)	0
θ (radians)	0
Final Position	
x (m)	0
y (m)	0.085
θ (radians)	π
Road Dim	
Right Edge (m)	0.15
Left Edge (m)	-0.15
Bottom Edge (m)	-0.043
Top Edge (m)	0.125
Objective Function Parameters	
α (m)	20
β (m)	0.2
Limits	
ϕ(degrees)	40 deg
$\phi_{desired}$(degrees)	90 deg
v_u(m/s)	∞
Car Parameters	
Width (m)	0.05
Length (m)	0.1
a (m)	0.05
b (m)	0.05
Mass (kg)	0.2
Moment of Inertia (kg m^2)	$3.33 * 10^{-4}$
τ (1/s)	1/3

B Problem Parameters for Parallel Parking

Initial Position	
x (m)	0
y (m)	0
θ (radians)	0
Final Position	
x (m)	0.12
y (m)	0
θ (radians)	0
Road Dim	
Right Edge (m)	0.3
Left Edge (m)	-0.06
Bottom Edge (m)	-0.095
Top Edge (m)	0.05
Parked Car Geometry	
Spot Length (m)	0.15
Spot Depth (m)	0.05
Objective Function Parameters	
α (m)	10
β (m)	2
Limits	
ϕ(degrees)	40 deg
$\phi_{desired}$(degrees)	90 deg
v_u(m/s)	0.3
Car Parameters	
Width (m)	0.05
Length (m)	0.1
a (m)	0.05
b (m)	0.05
Mass (kg)	0.2
Moment of Inertia (kg m^2)	$3.33 * 10^{-4}$
τ (1/s)	1/3

C Trajectory for Turning Around

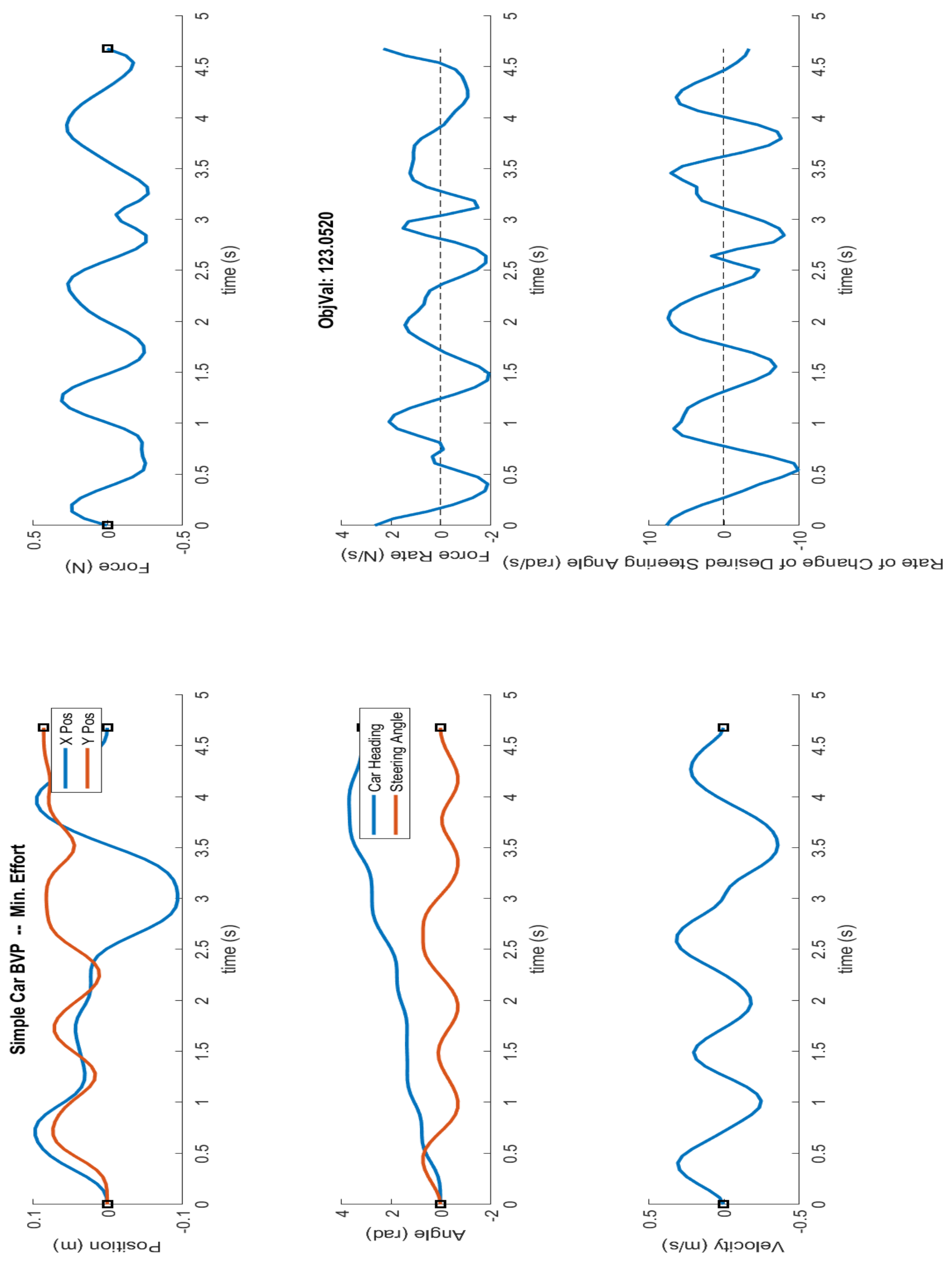

Figure 2: The optimal trajectory for a car turning around.

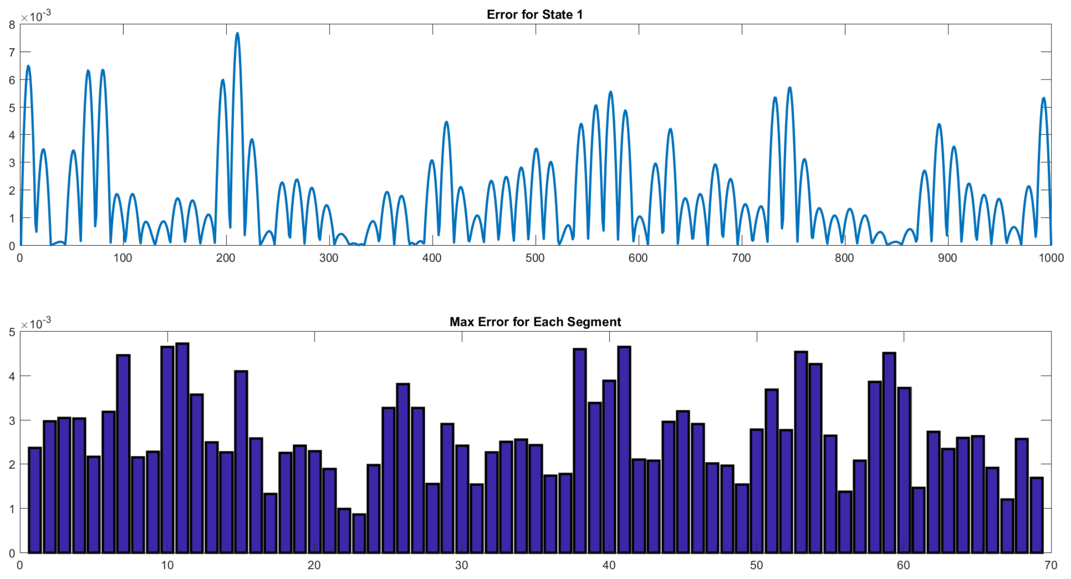

Figure 3: The error analysis of the optimal trajectory for a car turning around.

D Trajectory for Parallel Parking

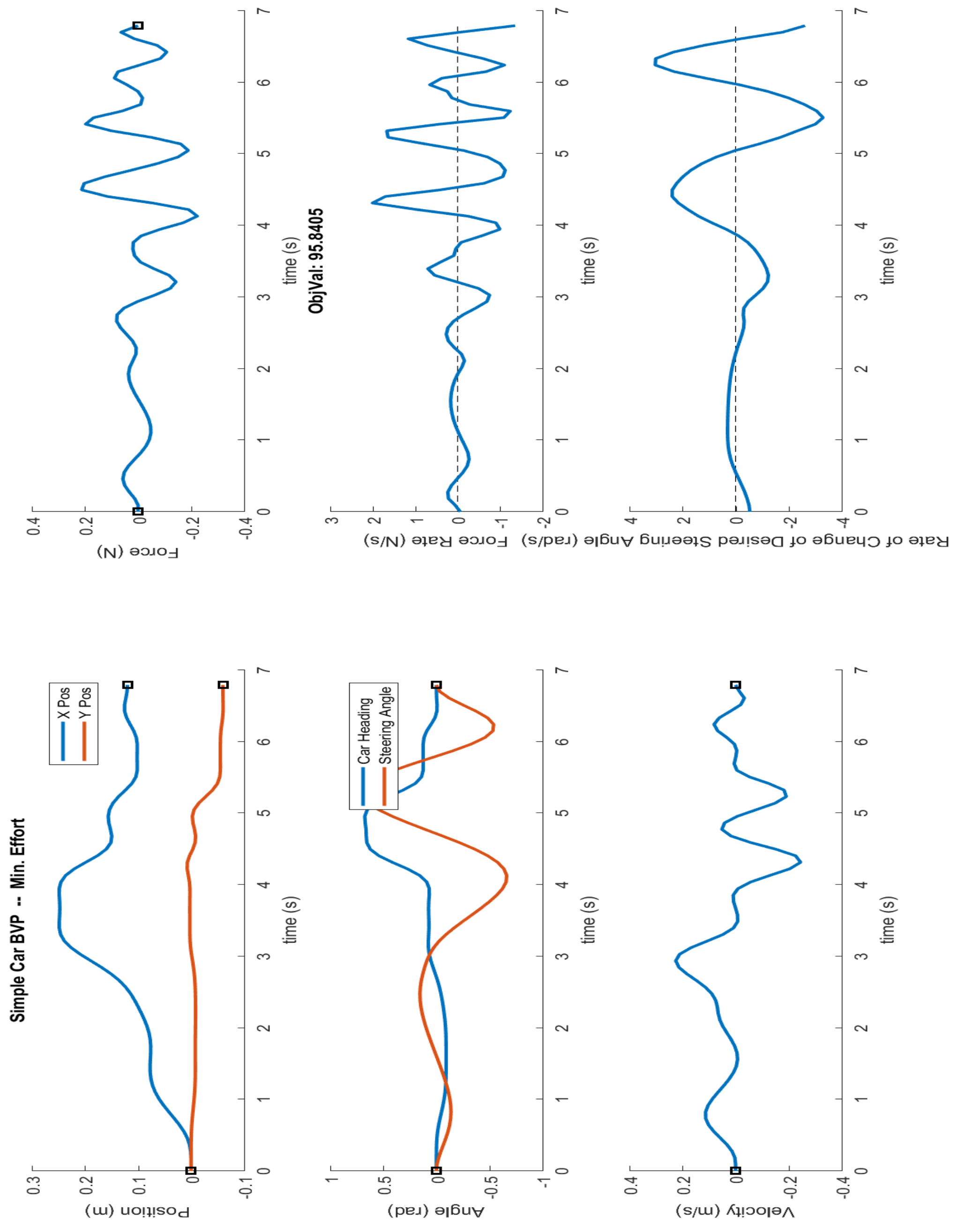

Figure 4: The optimal trajectory for a car parallel parking.

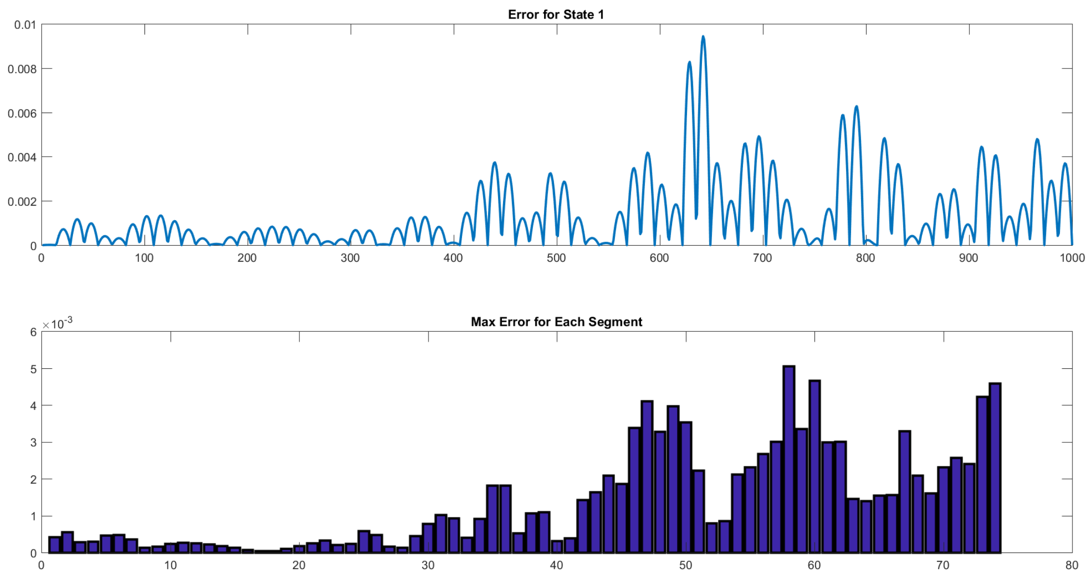

Figure 5: The error analysis of the optimal trajectory for a car parallel parking.