Optimization-based Receding Horizon Trajectory Planner using Bernstein Polynomials

Research by Calvin Kielas-Jensen and Rachit Singhvi

November 5, 2020
Motivation

- Trajectory Planning
- Optimizing nominal trajectories
- Limited sensing range
- Safety and obstacle avoidance
- Performance limits of vehicle
- Guaranteed solutions
Optimization Approach

- Feasible nominal trajectory for “warm starting”
- Horizon constitutes small segment of trajectory
- \( \uparrow \) Horizon size \( \uparrow \) Computational expense
- Iterative optimization follows feasibility constraints
- Off-the-shelf optimizer such as fmincon, SciPy
Receding Horizon Planner

- $\tau$ $\rightarrow$ Timing parameter on terminal manifold
- $t$ $\rightarrow$ Timing parameter for current horizon

1st Iteration

Trigger time $\delta = 0.5$ s

Optimal v/s Nominal Trajectory
Key Definitions

- **Cost Function**: Function to be minimized in order to obtain an optimal solution
- **Constraints**: Conditions that ensure safety and feasibility of the solution
- **Feasibility**: If the solution adheres to all constraints, it is feasible
- **Optimality**: If the cost of a particular solution is less than all other solutions, it is optimal
Problem Formulation

**OCP:** Find $x : [t_k, t_k + T] \rightarrow \mathbb{R}^{n_x}, t \in [t_k, t_k + T], \tau_k \in [t_k + T, t_f]$ that solves

$$
\min_{x_k(t), \tau_k} J_{tot}(x_{cur}, \tau_k) = \Psi_k(\tau_k) + \int_{t_k}^{t_k + T} l(x_k(t)) dt
$$

subject to

- **Dynamic constraint:**
  $$\dot{x}_k(t) = f(x_k(t)) \quad \forall t \in [t_k, t_k + t]$$

- **Equality constraints:**
  $$e(x_k(t_k), x_k(t_k + T), t_k + T) = 0$$

- **Inequality constraints:**
  $$h(x_k(t)) \leq 0 \quad \forall t \in [t_k, t_k + t]$$

**Cost Function**

**Constraints**

Terminal manifold timing parameter

Cost-to-go

Running cost

State
Simulation Results

2-Trailer Truck Parking:
• Trigger time $\delta = 0.5$ s
• Horizon length $T = 60$ s

<table>
<thead>
<tr>
<th>$T$ [s]</th>
<th>20</th>
<th>40</th>
<th>60 *</th>
<th>80 *</th>
<th>120</th>
<th>FH</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta J_{tot}$ [%]</td>
<td>-12.5</td>
<td>-14.4</td>
<td>-23.0</td>
<td>-24.1</td>
<td>-26.3</td>
<td>-26.5</td>
</tr>
<tr>
<td>$\bar{t}_{RHP}$ [s]</td>
<td>0.05</td>
<td>0.14</td>
<td>0.32</td>
<td>0.64</td>
<td>3.6</td>
<td>14.0</td>
</tr>
<tr>
<td>$\Delta \bar{t}_{lat}$</td>
<td>0.34</td>
<td>0.96</td>
<td>1.8</td>
<td>3.1</td>
<td>7.3</td>
<td>14.0</td>
</tr>
<tr>
<td>$\Delta \bar{t}_{tot}$</td>
<td>-16.6</td>
<td>23.5</td>
<td>-30.2</td>
<td>-30.3</td>
<td>-26.2</td>
<td>-22.8</td>
</tr>
</tbody>
</table>

Fig 1. Reverse parking scenario
Optimization using BeBOT

- Impossible to solve the optimal trajectory generation problem
- Transcribe the problem into a standard optimization problem to be solved using off-the-shelf solvers
- Bernstein polynomials offer several useful properties
- Toolkit publicly available on GitHub

https://github.com/caslabuiowa/BeBOT
Properties of Bernstein polynomials

→ Bernstein Coefficients
  • Contained within the convex hull
  • End points

→ de Casteljau Algorithm
  • Split into two Bernstein polynomials
  • Efficient and robust

→ Degree Elevation
  • Order can be elevated (elevation matrix computed offline)
  • Coefficients converge to the polynomial as $N \to \infty$
Applications

- Non-agile vehicles such as truck-trailer system
- Smooth maneuvering at high speeds e.g. F1 cars
- Dynamic environments and moving obstacles
- Precision agriculture
- Space travel
IOWA
Algorithm

Initialization

BeBOT for Optimization

Feasibility Check

Optimality Check

Update Trajectory

Algorithm 1 Receding horizon planning

1: Input: \( x_0, x_f, T, \delta, \mathcal{R}_{\text{free}} \)
2: \((\bar{x}_0, u_{\text{init}}, \bar{T}_f) \leftarrow \text{Motion planner}(x_0, x_f, \mathcal{R}_{\text{free}})\)
3: \( \tau_0 \leftarrow \tau_0 + T \), \( T_{\text{init}} \leftarrow \tau_0 - \tau_0 \)
4: \((x_{\text{init}}, u_{\text{init}}) \leftarrow \text{resample}(u_{\text{init}}, x_{\text{init}} - \delta)\)
5: while \( \tau_k \neq \bar{\tau}_k \) do
6: Set \( \bar{x}_{\text{can}} = \bar{x}_{\text{can}} - (\bar{k}) \) in (19)
7: \((u_k^*, T_k^*) \leftarrow \text{Solve (19) using } u_{\text{init}}^*, x_k^*, T_k^*, \text{ and } \tau_k\)
8: if \( J(x_{\text{can}}, u_k^*, T_k^*) < \infty \) then
9: \( \Delta \tau_k \leftarrow (\tau_k + T_k) - (\tau_k + T_k) \)
10: \((u_{\text{can}}, x_{\text{can}}) \leftarrow \text{get cand}(u_k^*, x_k^*, \Delta \tau_k)\)
11: if \( J_{\text{total}}(x_0, u_{\text{can}}) < J_{\text{total}}(x_0, u) \) then
12: Update solution:
13: \((u_k, x_k) \leftarrow (u_{\text{can}}, x_{\text{can}})\)
14: \( \bar{T}_f \leftarrow \bar{T}_f - \Delta \tau_k \)
15: else
16: \((u_k, x_k, \bar{T}_f) \leftarrow (u_{k-1}, x_{k-1}, \bar{T}_f - 1)\)
17: end if
18: else
19: \((u_k, x_k, \bar{T}_f) \leftarrow (u_{k-1}, x_{k-1}, \bar{T}_f - 1)\)
20: end if
21: Send nominal trajectory to controller:
22: \( \text{send reference}(u_k, x_k)\)
23: Update receding horizon terminal constraint:
24: \( \bar{T}_{k+1} \leftarrow \text{update timing}(\bar{T}_{k+1}, \bar{T}_f)\)
25: Initialization for next iteration:
26: \( T_{\text{init}} \leftarrow \bar{T}_{k+1} - \bar{T}_{k} \)
27: \((x_{k+1}^*, u_{k+1}^*) \leftarrow \text{resample}(u_k, x_k, T_{\text{init}}^*/N)\)
28: Set \( k \rightarrow k + 1 \)
29: end while
2-Trailer Truck System

\[
x = [q^T \alpha \omega v_1 a_1]^T
\]

\[
q = [x_3 y_3 \theta_3 \beta_3 \beta_2]^T
\]

\[
\dot{q} = v_1 f(q, \alpha)
\]

\[
l(x, u) = 1 + \frac{1}{2} (\alpha^2 + 10\omega^2 + a_1^2)
\]