

# PUMA 560 Manipulator Project

Path Planning and Trajectory Planning , algorithm Comparison  
and Analysis

Saran Konala      Aryan Mehta  
Pakki Sai Kaushik      Panshul Jindal

Indian Institute of Technology, Hyderabad

November 18th, 2025

# Outline

- 1 Introduction
- 2 Path Planning Comparison
  - Linear Path + Bang-Bang
  - Arc Path + Bang-Bang
  - Parabolic Path + Bang-Bang
  - RRT Path + Bang-Bang
- 3 Trajectory Planning Comparison
  - Arc + Cubic Spline
  - Arc + Quintic Polynomial
  - Arc + LSPB
  - Arc + Bang-Bang
- 4 Conclusion

# Project Overview

- **Objective:** Compare path planning and trajectory planning algorithms for PUMA 560 manipulator
- **Path Planning:** Determines waypoints in Cartesian space (where to go)
- **Trajectory Planning:** Generates smooth joint-space motion (how to get there)
- **Approach:**
  - Fix one algorithm type, vary the other
  - Analyze performance differences

# Path Planning Comparison Setup

## Fixed Component

**Trajectory Planning:** Bang-Bang Acceleration Control

## Variable Component

### Path Planning Algorithms:

- ① Linear Path
- ② Arc Path
- ③ Parabolic Path
- ④ RRT (Rapidly-Exploring Random Tree)

# Linear Path + Bang-Bang Trajectory

## Path Planning: Linear

Require: Start  $S$ , End  $E$ , Points  $N$

```
1: for  $i = 0$  to  $N - 1$  do
2:    $t \leftarrow \frac{i}{N-1}$ 
3:    $P[i] \leftarrow (1 - t)S + tE$ 
4: end for
5: return  $P$ 
```

## Trajectory: Bang-Bang

Require:  $q_0, q_1, T$

```
1: if  $\tau < 0.5$  then
2:    $q = q_0 + 2(q_1 - q_0)\tau^2$ 
3: else
4:    $q = q_1 - 2(q_1 - q_0)(1 - \tau)^2$ 
5: end if
```

# Arc Path + Bang-Bang Trajectory

## Path Planning: Arc

Require: Start  $S$ , End  $E$ , Points  $N$

```
1:  $h \leftarrow 0.2$                                 ▷ Height
2: for  $i = 0$  to  $N - 1$  do
3:    $t \leftarrow \frac{i}{N-1}$ 
4:    $P[i] \leftarrow (1 - t)S + tE$ 
5:    $P[i]_z \leftarrow P[i]_z + 4ht(1 - t)$ 
6: end for
7: return  $P$ 
```

## Trajectory: Bang-Bang

Require:  $q_0, q_1, T$

```
1: if  $\tau < 0.5$  then
2:    $q = q_0 + 2(q_1 - q_0)\tau^2$ 
3: else
4:    $q = q_1 - 2(q_1 - q_0)(1 - \tau)^2$ 
5: end if
```

# Parabolic Path + Bang-Bang Trajectory

## Path Planning: Parabolic

**Require:** Start  $S$ , End  $E$ , Points  $N$

```
1: for  $i = 0$  to  $N - 1$  do
2:    $t \leftarrow \frac{i}{N-1}$ 
3:    $P[i] \leftarrow (1 - t)S + tE$ 
4:    $P[i]_y \leftarrow P[i]_y + 0.2 \sin(\pi t)$ 
5:    $P[i]_z \leftarrow P[i]_z + 0.2 \cdot 4t(1 - t)$ 
6: end for
7: return  $P$ 
```

## Trajectory: Bang-Bang

**Require:**  $q_0, q_1, T$

```
1: if  $\tau < 0.5$  then
2:    $q = q_0 + 2(q_1 - q_0)\tau^2$ 
3: else
4:    $q = q_1 - 2(q_1 - q_0)(1 - \tau)^2$ 
5: end if
```

# RRT Path + Bang-Bang Trajectory

## Path Planning: RRT

Require: Start  $S$ , Goal  $G$ , Max iter  $M$

```
1:  $T \leftarrow [S]$ 
2: for  $k = 1$  to  $M$  do
3:   Sample  $X_{rand}$ 
4:   Find nearest  $X_{near}$  in  $T$ 
5:    $X_{new} \leftarrow X_{near} + \epsilon \cdot \frac{d}{\|d\|}$ 
6:   if collision-free then
7:     Add  $X_{new}$  to  $T$ 
8:     if  $\|X_{new} - G\| < \epsilon$  then
9:       return path
10:    end if
11:   end if
12: end for
```

## Trajectory: Bang-Bang

```
1: if  $\tau < 0.5$  then
2:    $q = q_0 + 2(q_1 - q_0)\tau^2$ 
3: else
4:    $q = q_1 - 2(q_1 - q_0)(1 - \tau)^2$ 
5: end if
```

# Path Planning Comparison Summary

Algorithm	Smoothness	Obstacle Avoidance	Computation
Linear	Low	No	Very Fast
Arc	Medium	Partial	Fast
Parabolic	High	Good	Fast
RRT	Variable	Excellent	Slow

- **Linear:** Simplest, fastest, but no obstacle avoidance
- **Arc/Parabolic:** Good balance of smoothness and efficiency
- **RRT:** Best for complex environments with obstacles

# Trajectory Planning Comparison Setup

## Fixed Component

**Path Planning:** Arc Path

## Variable Component

**Trajectory Planning Algorithms:**

- ① Cubic Spline Interpolation
- ② Quintic Polynomial
- ③ LSPB (Linear Segment with Parabolic Blends)
- ④ Bang-Bang Acceleration Control

# Arc Path + Cubic Spline Trajectory

## Path Planning: Arc (Fixed)

```
1: for  $i = 0$  to  $N - 1$  do
2:    $t \leftarrow \frac{i}{N-1}$ 
3:    $P[i] \leftarrow (1 - t)S + tE$ 
4:    $P[i]_z \leftarrow P[i]_z + 4ht(1 - t)$ 
5: end for
```

## Trajectory: Cubic Spline

Require: Times  $t_{coarse}$ , positions  $q_{coarse}$

```
1: for each joint  $j$  do
2:   Create CubicSpline with clamped BC
3:    $q_{fine}[:, j] \leftarrow \text{Spline}(t_{fine})$ 
4: end for
5: Apply phase unwrapping
6: return  $q_{fine}$ 
```

**Properties:**  $C^2$  continuous,  
smooth

# Arc Path + Quintic Polynomial Trajectory

## Path Planning: Arc (Fixed)

```
1: for  $i = 0$  to  $N - 1$  do
2:    $t \leftarrow \frac{i}{N-1}$ 
3:    $P[i] \leftarrow (1 - t)S + tE$ 
4:    $P[i]_z \leftarrow P[i]_z + 4ht(1 - t)$ 
5: end for
```

## Trajectory: Quintic Polynomial

Require:  $q_0, q_1, T$

```
1:  $\tau \leftarrow \frac{t}{T}$ 
2:  $q(\tau) = a_0 + a_1\tau + \dots + a_5\tau^5$ 
3:  $a_0 = q_0, a_1 = a_2 = 0$ 
4:  $a_3 = 10(q_1 - q_0)$ 
5:  $a_4 = -15(q_1 - q_0)$ 
6:  $a_5 = 6(q_1 - q_0)$ 
```

Properties:  $C^3$  continuous, very smooth

# Arc Path + LSPB Trajectory

## Path Planning: Arc (Fixed)

```
1: for  $i = 0$  to  $N - 1$  do
2:    $t \leftarrow \frac{i}{N-1}$ 
3:    $P[i] \leftarrow (1 - t)S + tE$ 
4:    $P[i]_z \leftarrow P[i]_z + 4ht(1 - t)$ 
5: end for
```

## Trajectory: LSPB

Require:  $q_0, q_1, T$ , blend time  $t_b = 0.3T$

```
1: if  $t < t_b$  then
2:    $q = q_0 + \frac{1}{2}at^2$            ▷ Accel
3: else if  $t_b \leq t \leq T - t_b$  then
4:    $q = q_0 + v(t - t_b/2)$        ▷ Const vel
5: else
6:    $q = q_1 - \frac{1}{2}a(T - t)^2$     ▷ Decel
7: end if
```

Properties:  $C^1$  continuous,  
efficient

# Arc Path + Bang-Bang Trajectory

## Path Planning: Arc (Fixed)

```
1: for  $i = 0$  to  $N - 1$  do
2:    $t \leftarrow \frac{i}{N-1}$ 
3:    $P[i] \leftarrow (1 - t)S + tE$ 
4:    $P[i]_z \leftarrow P[i]_z + 4ht(1 - t)$ 
5: end for
```

## Trajectory: Bang-Bang

Require:  $q_0, q_1, T$

```
1:  $\tau \leftarrow \frac{t}{T}$ 
2: if  $\tau < 0.5$  then
3:    $q = q_0 + 2(q_1 - q_0)\tau^2$       ▷ Accel
4: else
5:    $q = q_1 - 2(q_1 - q_0)(1 - \tau)^2$   ▷ Decel
6: end if
```

Properties:  $C^0$  continuous, fast  
but jerky

# Trajectory Planning Comparison Summary

Algorithm	Smoothness	Obstacle Avoidance	Computation
Linear	Low	No	Very Fast
Arc	Medium	Partial	Fast
Parabolic	High	Good	Fast
RRT	Variable	Excellent	Slow

Table: Comparison of Path Planning Algorithms

- **Quintic:** Best for precision tasks requiring smooth motion
- **Cubic:** Good balance of smoothness and computation
- **LSPB:** Efficient for simple point-to-point motion
- **Bang-Bang:** Fastest execution time but causes jerk

# Key Findings

## Path Planning

- Choice depends on workspace complexity and obstacle presence
- RRT excels in cluttered environments but is computationally expensive
- Simple paths (linear, arc) sufficient for open workspaces

## Trajectory Planning

- Higher-order polynomials provide smoother motion
- Trade-off between smoothness and computational efficiency
- Bang-Bang suitable only when high jerk is acceptable

# Thank You!