Path Planning and Trajectory Planning

Algorithm Comparison and Analysis

Saran, Aryan, Sai Kaushik, Panshul

November 18, 2025

Outline

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: $\quad t \leftarrow \frac{i}{N-1}$
3: $\quad P[i] \leftarrow (1 - t)S + tE$
4: **end for**
5: **return** P

Trajectory: Bang-Bang

Require: q_0, q_1, T

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

Arc Path + Bang-Bang Trajectory

Path Planning: Arc

Require: Start S, End E, Points N

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

Trajectory: Bang-Bang

Require: q_0, q_1, T

if $\tau < 0.5$ **then**
    $q = q_0 + 2(q_1 - q_0)\tau^2$
else
    $q = q_1 - 2(q_1 - q_0)(1 - \tau)^2$
end if

Parabolic Path + Bang-Bang Trajectory

Path Planning: Parabolic

Require: Start S, End E, Points N

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

Trajectory: Bang-Bang

Require: q_0, q_1, T

if $\tau < 0.5$ **then**
  $q = q_0 + 2(q_1 - q_0)\tau^2$
else
  $q = q_1 - 2(q_1 - q_0)(1 - \tau)^2$
end if

RRT Path + Bang-Bang Trajectory

Path Planning: RRT

Require: Start S, Goal G, Max iter M

$T \leftarrow [S]$
for $k = 1$ to M **do**
    Sample X_{rand}
    Find nearest X_{near} in T
    $X_{new} \leftarrow X_{near} + \epsilon \cdot \frac{d}{\|d\|}$
    if collision-free **then**
        Add X_{new} to T
        if $\|X_{new} - G\| < \epsilon$ **then**
            return path
        end if
    end if
end for

Trajectory: Bang-Bang

if $\tau < 0.5$ **then**
    $q = q_0 + 2(q_1 - q_0)\tau^2$
else
    $q = q_1 - 2(q_1 - q_0)(1 - \tau)^2$
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)

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

Trajectory: Cubic Spline

Require: Times t_{coarse}, positions q_{coarse}
for each joint j **do**
    Create CubicSpline with clamped BC
    $q_{fine}[:, j] \leftarrow \text{Spline}(t_{fine})$
end for
Apply phase unwrapping
return q_{fine}

Properties: C^2 continuous, smooth

Arc Path + Quintic Polynomial Trajectory

Path Planning: Arc (Fixed)

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

Trajectory: Quintic Polynomial

Require: q_0, q_1, T
$\tau \leftarrow \frac{t}{T}$
$q(\tau) = a_0 + a_1\tau + ... + a_5\tau^5$
$a_0 = q_0,\ a_1 = a_2 = 0$
$a_3 = 10(q_1 - q_0)$
$a_4 = -15(q_1 - q_0)$
$a_5 = 6(q_1 - q_0)$

Properties: C^3 continuous, very smooth

Arc Path + LSPB Trajectory

Path Planning: Arc (Fixed)

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

Trajectory: LSPB

Require: q_0, q_1, T, blend time $t_b = 0.3T$
if $t < t_b$ **then**
  $q = q_0 + \frac{1}{2}at^2$ ▷ Accel
else if $t_b \leq t \leq T - t_b$ **then**
  $q = q_0 + v(t - t_b/2)$ ▷ Const vel
else
  $q = q_1 - \frac{1}{2}a(T - t)^2$ ▷ Decel
end if

Properties: C^1 continuous, efficient

Arc Path + Bang-Bang Trajectory

Path Planning: Arc (Fixed)

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

Trajectory: Bang-Bang

Require: q_0, q_1, T
$\tau \leftarrow \frac{t}{T}$
if $\tau < 0.5$ **then**
    $q = q_0 + 2(q_1 - q_0)\tau^2$ ▷ Accel
else
    $q = q_1 - 2(q_1 - q_0)(1 - \tau)^2$ ▷ Decel
end if

Properties: C^0 continuous, fast but jerky

Trajectory Planning Comparison Summary

Algorithm	Continuity	Smoothness	Control	Speed
Cubic Spline	C^2	High	Partial	Medium
Quintic Poly	C^3	Very High	Full	High
LSPB	C^1	Medium	Partial	Low
Bang-Bang	C^0	Low	None	Very Low

- **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!