Minimum Jerk Polynomial Path Planning

(Preliminary Results)

Collected Notes and Ideas
David Kooi

Introduction

It can be proven that a scalar function p(t) has minimum jerk when the 5th derivative of that function is zero. This project demonstrates uses a minimum jerk function to solve the following path planning problems in conjunction:

Problem 1: Given two points A and B, what is a trajectory between the two points minimizes the jerk?
Problem 2: Given two points A and B, what is a trajectory that avoids all obstacles and with minimum path length?

Problem 1 will be first addressed by using a 5th order polynomial. Problem 2 will then be addressed by discretizing the state space into a state-lattice and using the D* algorithm for obstacle avoidance. Both problems are solved in conjunction, resulting in a minimum jerk, obstacle avoiding, polynomial path between two points.

Problem 1: Given two points A and B, what is a trajectory between the two points minimizes the jerk?

Minimum Jerk Trajectory Derivation

TBD

Problem 1: Given two points A and B, what is a trajectory between the two points minimizes the jerk?

Sufficient Conditions for Minimum Jerk

A trajectory p will have minimum jerk if p satisfies the condition:

$$p^{(5)}(t) = 0$$

Accordingly, a 5th order polynomial provides minimum jerk trajectory solution:

$$p(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0$$

Problem 1: Given two points A and B, what is a trajectory between the two points minimizes the jerk?

Solving for Coefficients

A minimum jerk trajectory can be solved by finding the polynomial coefficients that satisfy the initial conditions of point A and the final conditions of point B.

The solution involves solving for the polynomial coefficients.

$$p(t) = a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0$$

The following coefficients are defined by the initial conditions:

$$a_0 = p(0)$$
$$a_1 = p'(0)$$
$$2 * a_2 = p''(0)$$

The following coefficients can be defined by the final conditions at time T.:

$$p(T) = a_5T^5 + a_4T^4 + a_3T^3 + a_2T^2 + a_1T + a_0$$

$$p(T)' = 5a_5T^4 + 4a_4T^3 + 3a_3T^2 + 2a_2T + a_1$$

$$p(T)'' = 20a_5T^3 + 12a_4T^2 + 6a_3T + 2a_2$$

Problem 1: Given two points A and B, what is a trajectory between the two points minimizes the jerk?

Solving for Coefficients

$$p(t) = a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0$$

Given the initial and final conditions we can solve for a0...a5

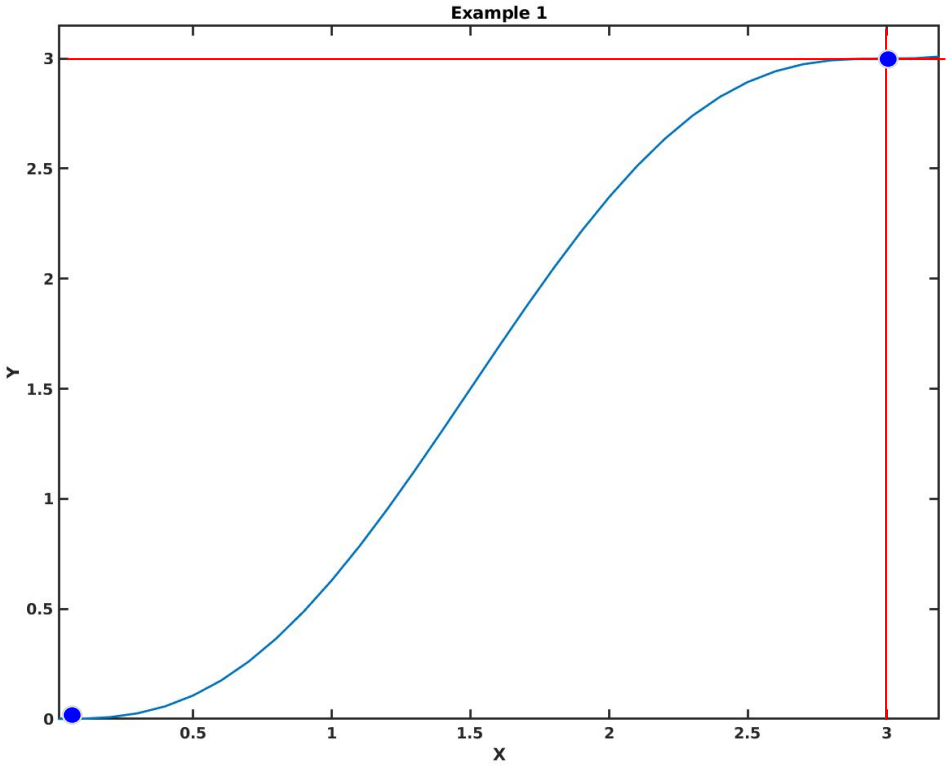
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 5*T^4 & 4*T^3 & 3*T^2 & 2*T & 1 & 0 \\ 20*T^3 & 12*T^2 & 6*T & 2 & 0 & 0 \end{bmatrix} + \begin{bmatrix} a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} p(0) \\ p'(0) \\ p''(0) \\ p(T) \\ p'(T) \\ p''(T) \end{bmatrix}$$

Problem 1: Given two points A and B, what is a trajectory between the two points minimizes the jerk?

Examples: Simple 2D

Initial Conditions:
x(0), y(0) = (0,0)
x'(0),y'(0) = (1,0)
x''(0), y''(0) = (0,0)

T = 4
Final Conditions:
x(T), y(T) = (3,3)
x'(T),y'(T) = (1,0)
x''(T), y''(T) = (0,0)

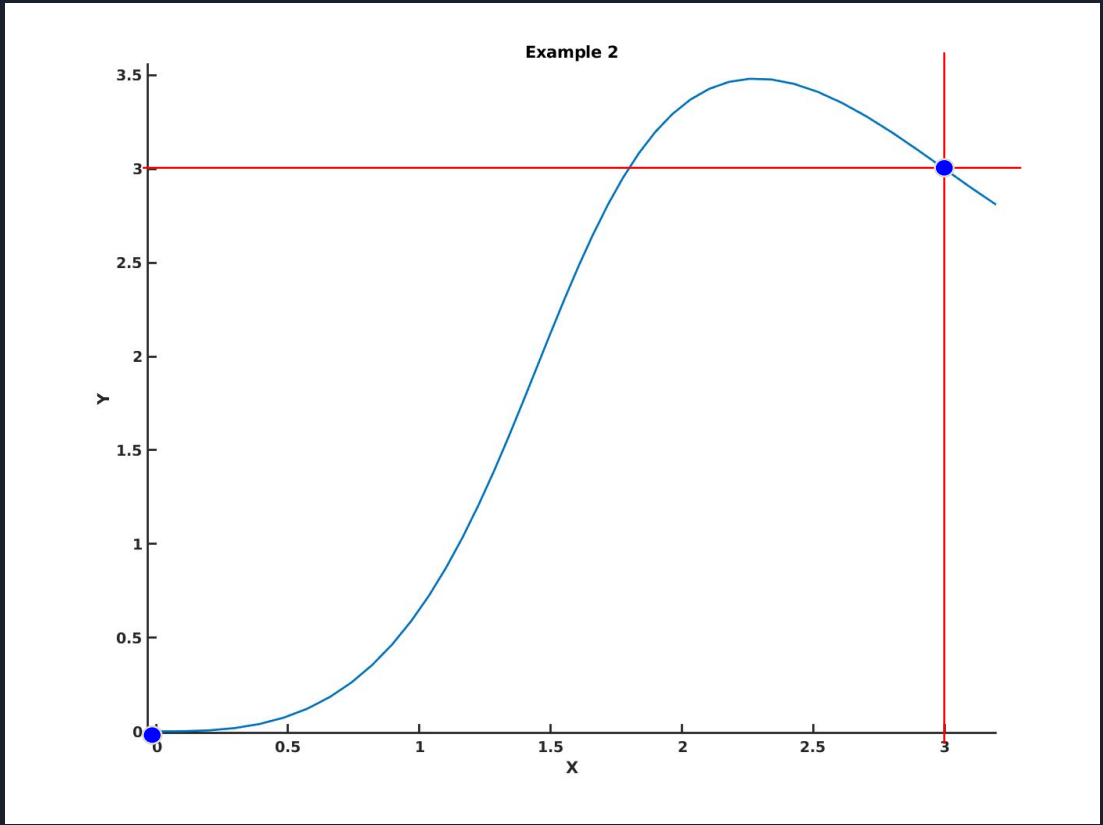


Problem 1: Given two points A and B, what is a trajectory between the two points minimizes the jerk?

Examples: Simple 2D

Initial Conditions:
x(0), y(0) = (0,0)
x'(0),y'(0) = (1,0)
x''(0), y''(0) = (0,0)

T = 4
Final Conditions:
x(T), y(T) = (3,3)
x'(T),y'(T) = (1,-1)
x''(T), y''(T) = (0,0)



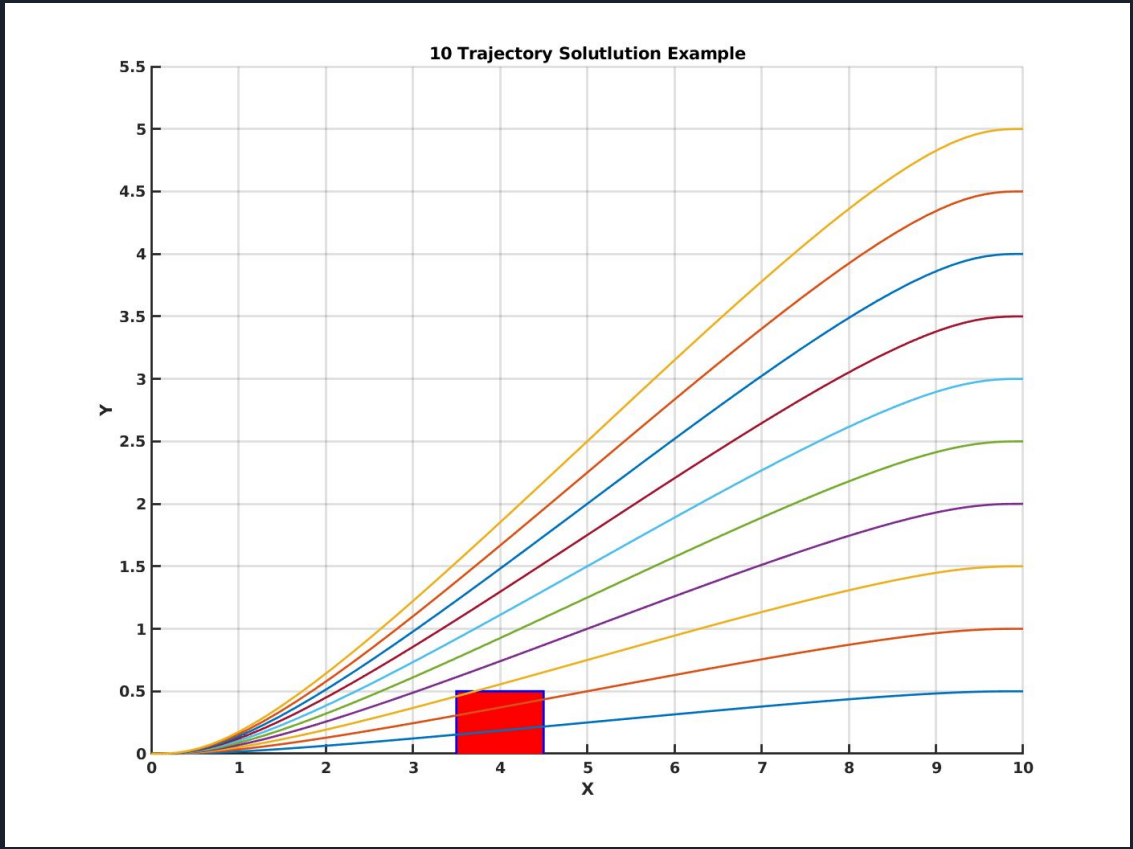
Problem 1: Given two points A and B, what is a trajectory between the two points minimizes the jerk?

Examples: Multiple Goal Trajectories

Initial Conditions:
x(0), y(0) = (0,0)
x’(0),y’(0) = (1,0)
x’’(0), y’’(0) = (0,0)

T = 5
Final Conditions:
x(T), y(T) = (10, 0.5*i)
For i=1:10

x’(T),y’(T) = (x’(0), y’(0)
x’’(T), y’’(T) = (x’’(0), y’’(0))



Problem 2 Introduction

Problem 2 is to travel from point A to point B while avoiding obstacles and with minimum path length. The state space of the trajectory will be discretized into a state-lattice in order to support the necessary points states of problem 1. Then the D* algorithm is used to provide obstacle avoidance with a cost function of path length to ensure the shortest path is found.

D* Algorithm

The D* algorithm is an informed incremental search algorithm. The algorithm has two main parts:

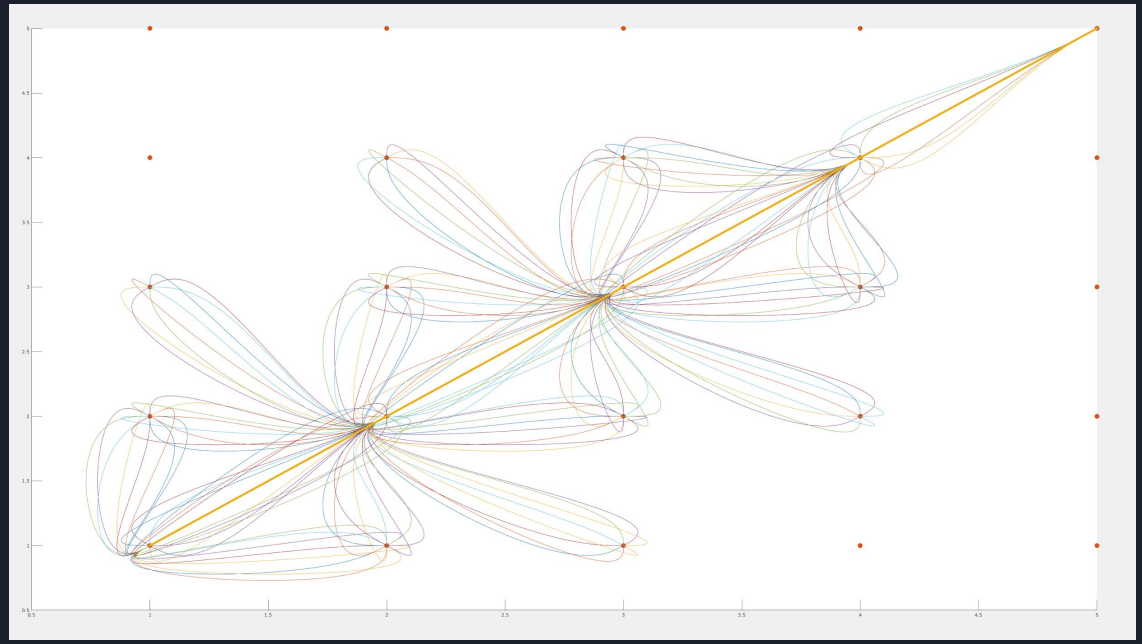
1. Initial path planning: Given initial information of the environment, create a path from point A to point B guaranteeing obstacle avoidance.

2. Replanning: Given updated information of the environment, replan the trajectory guaranteeing obstacle avoidance

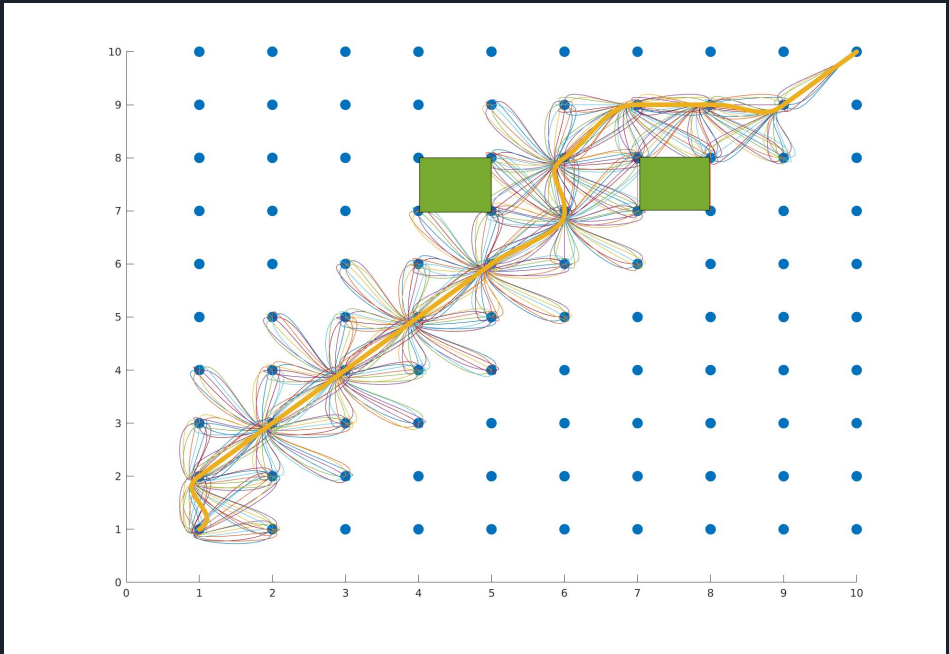
The preliminary results of this project use only part 1 of D*. Part 1 by itsel is identical to Backwards A*.

Results

Shortest Path Confirmation

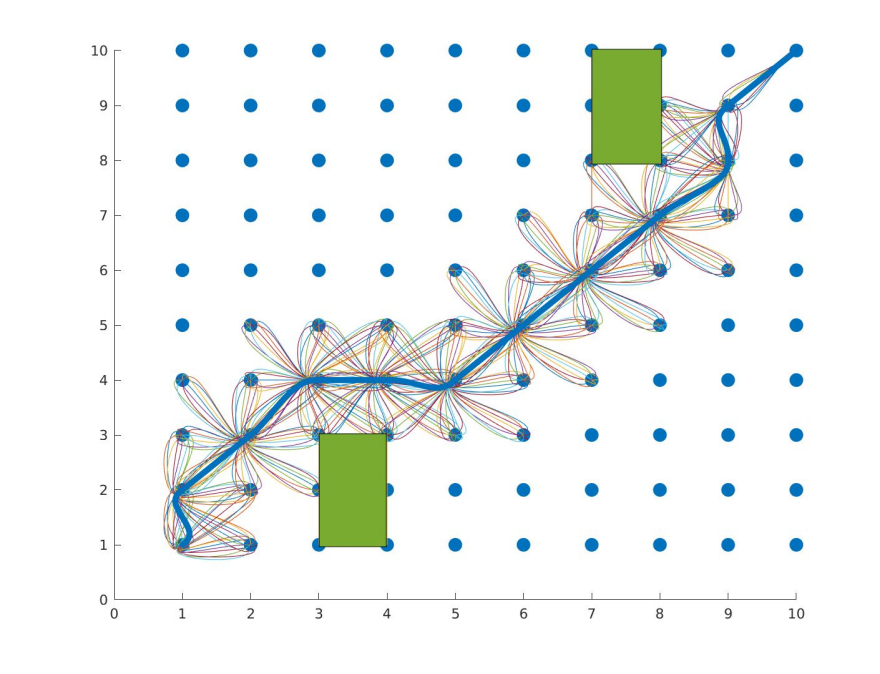


Results



10x10 Lattice
Start: (10,10) with xv=-1, yv=-1
End: (1,1) with xv=1, yv=1
Solution Time: 2.46s

Results



10x10 Lattice
Start: (10,10) with xv=-1, yv=-1
End: (1,1) with xv=1, yv=1
Solution Time: 2.61s

Appendix

Matlab Computation Time

One dimension coefficient solution: 0.0026 s (2.6 ms)
Two dimension coefficient solution: 0.0025s (2.5ms)

10 Two dimensional solutions: 0.0139s (13ms)

Computer Specs

Memory: 15.6 GiB
Processor: Intel i7 @ 2.8GHz (8 core, 64 bit)

Algorithm Overview

procedure CalculateKey(s)
{01'} return $[\min(g(s), rhs(s)) + h(s_{start}, s) + k_m ; \min(g(s), rhs(s))]$;

procedure Initialize()
{02'} $U = \emptyset$;
{03'} $k_m = 0$;
{04'} for all $s \in S$ $rhs(s) = g(s) = \infty$;
{05'} $rhs(s_{goal}) = 0$;
{06'} U.Insert(s_{goal}, CalculateKey(s_{goal}));

procedure UpdateVertex(u)
{07'} if $(u \neq s_{goal})$ $rhs(u) = \min_{s' \in Succ(u)}(c(u, s') + g(s'))$;
{08'} if $(u \in U)$ U.Remove(u);
{09'} if $(g(u) \neq rhs(u))$ U.Insert(u, CalculateKey(u));

procedure ComputeShortestPath()
{10'} while (U.TopKey() $\dot{<}$ CalculateKey(s_{start}) OR $rhs(s_{start}) \neq g(s_{start})$)
{11'} $k_{old} =$ U.TopKey();
{12'} $u =$ U.Pop();
{13'} if ($k_{old} \dot{<}$ CalculateKey(u))
{14'} U.Insert(u, CalculateKey(u));
{15'} else if $(g(u) > rhs(u))$
{16'} $g(u) = rhs(u)$;
{17'} for all $s \in Pred(u)$ UpdateVertex(s);
{18'} else
{19'} $g(u) = \infty$;
{20'} for all $s \in Pred(u) \cup \{u\}$ UpdateVertex(s);

Excerpt of D* Lite Algorithm of Koening and Likhachev
http://idm-lab.org/bib/abstracts/papers/aaai02b.pdf

This project uses a MATLAB translation of the psudo-code shown.

procedure Initialize()
{02'} U is the priority queue holding states s and prioritized by key(s) = CaculateKey(s)
{03'} km is used in the replanning part of D*
{04'} S is a set of states within the state-lattice. All states within S are initialized. T

Algorithm Overview

procedure CalculateKey(s)
{01'} return $[\min(g(s), rhs(s)) + h(s_{start}, s) + k_m ; \min(g(s), rhs(s))]$;

procedure Initialize()
{02'} $U = \emptyset$;
{03'} $k_m = 0$;
{04'} for all $s \in S$ $rhs(s) = g(s) = \infty$;
{05'} $rhs(s_{goal}) = 0$;
{06'} U.Insert(s_{goal}, CalculateKey(s_{goal}));

procedure UpdateVertex(u)
{07'} if $(u \neq s_{goal})$ $rhs(u) = \min_{s' \in Succ(u)}(c(u, s') + g(s'))$;
{08'} if $(u \in U)$ U.Remove(u);
{09'} if $(g(u) \neq rhs(u))$ U.Insert(u, CalculateKey(u));

procedure ComputeShortestPath()
{10'} while (U.TopKey() $\dot{<}$ CalculateKey(s_{start}) OR $rhs(s_{start}) \neq g(s_{start})$)
{11'} $k_{old} =$ U.TopKey();
{12'} $u =$ U.Pop();
{13'} if ($k_{old} \dot{<}$ CalculateKey(u))
{14'} U.Insert(u, CalculateKey(u));
{15'} else if $(g(u) > rhs(u))$
{16'} $g(u) = rhs(u)$;
{17'} for all $s \in Pred(u)$ UpdateVertex(s);
{18'} else
{19'} $g(u) = \infty$;
{20'} for all $s \in Pred(u) \cup \{u\}$ UpdateVertex(s);

Excerpt of D* Lite Algorithm of Koening and Likhachev
http://idm-lab.org/bib/abstracts/papers/aaai02b.pdf

This project uses a MATLAB translation of the psudo-code shown.

procedure Initialize()
Line 02 initializes the priority queue U as empty.
03 initializes variable Km used in replanning
04 sets all states within state-set S

Literature

Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments

Charles Richter, Adam Bry, and Nicholas Roy

5-24-2012

Minimum Jerk Trajectory Planning for Trajectory Constrained Redundant Robots

Philip Freeman
Washington University in St. Louis

Thoughts

By itself, trivial polynomial planning does not guarantee collision free trajectories, accommodate, holonomic constraints, nor solve the path planning problem.

Methods like RRT* solve the path planning problem through incremental search.
RRT* uses a steering function to extend it's graph. I.e RRT* samples the space ahead of the vehicle and chooses the path that is least cost and is collision free.

Can the computational effort be reduced using polynomial planning? If an incremental search can provide "sparse" points within the free space, could it be guaranteed that a polynomial curve can result in a collision free trajectory that satisfies vehicle constraints?

Matlab Solving Coefficients

```
function coeffs = ppp(x0,v0,a0,xf,vf,af,tf)

B = [x0;v0;a0;xf;vf;af];
A = [0,       0,       0,      0,     0,     1;...
     0,       0,       0,      0,     1,     0;...
     0,       0,       0,      2,     0,     0;...
     tf^5,    tf^4,    tf^3,   tf^2,  tf,    1;...
     5*tf^4,  4*tf^3,  3*tf^2, 2*tf,  1,     0;...
     20*tf^3, 12*tf^2, 6*tf,   2,     0,     0];
coeffs = A\B
```