

Lesson 10:
Trajectory Generation

SEARCH

RESOURCES

CONCEPTS

1. From Behavior to Trajectory

2. Lesson Overview

3. The Motion Planning Problem

4. Properties of Motion Planning ...

5. Types of Motion Planning Algori...

6. A* Reminder

7. A* Reminder Solution

8. Hybrid A* Introduction

9. Hybrid A* Tradeoffs

10. Hybrid A* Tradeoffs Solution

11. Hybrid A* in Practice

12. Hybrid A* Heuristics

13. Hybrid A* Pseudocode

14. Implement Hybrid A* in C++

15. Implement Hybrid A* in C++ (s...

16. Environment Classification

17. Frenet Reminder

18. The Need for Time

19. s, d, and t

20. Trajectory Matching

Mentor Help

Ask a mentor on our Q&A platform

Peer Chat

Chat with peers and alumni

Implement Quintic Polynomial Solver C++

SEND FEEDBACK

Instructions

1. Implement the JMT(start, end, T) function in main.cpp

2. Hit Test Run and see if you're correct!

Tips

Remember, you are solving a system of equations: matrices will be helpful! The Eigen library used from Sensor Fusion is included.

The equations for position, velocity, and acceleration are given by:

$$s(t) = s_i + \dot{s}_i t + \frac{\ddot{s}_i}{2} t^2 + \alpha_3 t^3 + \alpha_4 t^4 + \alpha_5 t^5$$
$$\dot{s}(t) = \dot{s}_i + \ddot{s}_i t + 3\alpha_3 t^2 + 4\alpha_4 t^3 + 5\alpha_5 t^4$$
$$\ddot{s}(t) = \ddot{s}_i + 6\alpha_3 t + 12\alpha_4 t^2 + 20\alpha_5 t^3$$

and if you evaluate these at $t = 0$ you find the first three coeffecients of your JMT are:

$$[\alpha_0, \alpha_1, \alpha_2] = [s_i, \dot{s}_i, \frac{1}{2} \ddot{s}_i]$$

and you can get the last three coefficients by evaluating these equations at $t = T$. When you carry out the math and write the problem in matrix form you get the following:

$$\begin{bmatrix} T^3 & T^4 & T^5 \\ 3T^2 & 4T^3 & 5T^4 \\ 6T & 12T^2 & 20T^3 \end{bmatrix} \times \begin{bmatrix} \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} s_f - (s_i + \dot{s}_i T + \frac{1}{2} \ddot{s}_i T^2) \\ \dot{s}_f - (\dot{s}_i + \ddot{s}_i T) \\ \ddot{s}_f - \ddot{s}_i \end{bmatrix}$$

All these quantities are known except for $\alpha_3, \alpha_4, \alpha_5$

main.cpp

grader.h

```
1 #include <cmath>
2 #include <iostream>
3 #include <vector>
4
5 #include "Dense"
6 #include "grader.h"
7
8 using std::vector;
9 using Eigen::MatrixXd;
10 using Eigen::VectorXd;
11
12 /**
13  * TODO: complete this function
14  */
15 vector<double> JMT(vector<double> &start, vector<double> &end, double T) {
16     /**
17      * Calculate the Jerk Minimizing Trajectory that connects the initial state
18
19      * to the final state in time T.
20      * @param start - the vehicles start location given as a length three array
21      * corresponding to initial values of [s, s_dot, s_double_dot]
22      * @param end - the desired end state for vehicle. Like "start" this is a
23      * length three array.
24      * @param T - The duration, in seconds, over which this maneuver should occur.
25      *
26      * @output an array of length 6, each value corresponding to a coefficent in
27      * the polynomial:
28      * s(t) = a_0 + a_1 * t + a_2 * t**2 + a_3 * t**3 + a_4 * t**4 + a_5 * t**5
29      *
30      * EXAMPLE
31      * JMT([50, 10, 0], [100, 20, 0], 10) returns [0, 10, 0, 0, 0, 0]
```

RESET QUIZ

TEST RUN

SUBMIT ANSWER