TRAJECTORY PLANNING OF DELTA ROBOT - JACOBIAN METHOD

Arvin Mohammadi
student number: 810698303

December 3, 2022

Contents

Chapter 1

INTRODUCTION

This report is in continuation of the previous one, about trajectory planning of the Delta Parallel Robot(DPR), in which the point-to-point movement was solved. But the same problem remained for multi-point trajectory planning, Meaning if the End-Effector would have to go through a pre-defined path in 3D space, we failed to find an optimized way to solve this problem. Here we continue on one of the suggested algorithms in that report. The Jacobian method through a sequence of points.

1.1 Problem Definition

The main problem at hand is to move the EE through a path in 3D-space with bounded jerk. Specifically we want to move the EE through a path of circular shape with the center of x_0 and y_0 and radius of R. we assume the z-plane height is constant so the problem reduces from moving in 3D space to moving in 2D space. We study kinematics to understand this problem better. The result of studying kinematics is that we shouldn't worry about the actuated joints and only focus on finding the "Good" trajectory of the EE.

1.2 Kinematics Reminder

First of all we address the assumptions of this problem.

1. Theta is the actuated angles of motor (i):

$$\vec{\theta} = \theta_{1i} = \begin{bmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \end{bmatrix}$$

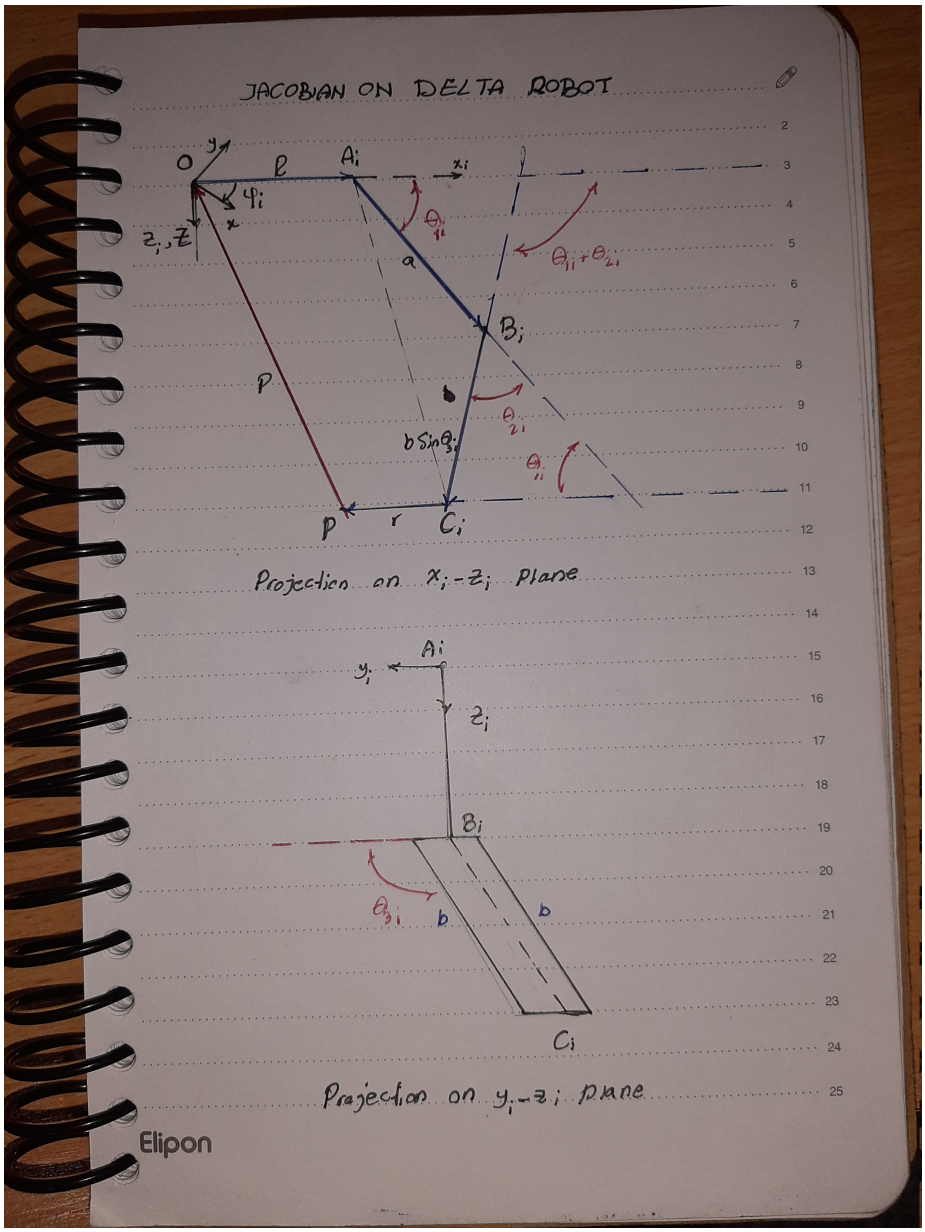
2. Global position of EE:

$$\vec{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

3. Jacobian matrix equation

$$J_\theta \dot{\vec{\theta}} = J_P \vec{v}$$

Figure 1.1: DPR Kinematics



Delta Parallel Robots kinematics are calculated through this method

Using the Jacobian matrix equation we can relate the velocity of EE to angular velocity of actuated joints for one point in space-time (x_0, y_0, z_0, t_0). The exact math can be found the previous report.

1.3 Suggested Algorithm

In this section we shortly mention the algorithm used to make the chosen EE trajectory work for the whole system.

- First Step: Generating the trajectory of EE as a function of time: Meaning we should define the path that we want EE to go through as a function of time and represent it using an equation in 3D space. (outputs trajectory equation)
- Second Step: Velocity profile calculation. This is easily doable by integrating the trajectory calculated in the previous step. (outputs velocity profile of EE)
- Third Step: IK of the multi-point trajectory. We already solved the problem of IK for DPR, so this step won't be a bother. (outputs angle of actuator joints through the time of movement)
- Fourth Step: Vector summation and θ_{ij} calculation. Meaning we'll find the actuated joint angular velocity profile with respect to time. (outputs the angular velocity profile of the actuator joints)

Chapter 2

JACOBIAN METHOD - DETERMINING FUNCTION OF EE MOTION

PREVIOUSLY we talked about the how to convert the trajectory of EE to angular velocity of the actuated joints. In this chapter we find the function that describes the trajectory of circular motion in EE.

Assuming that EE moves on the $z = 0$ plane, the circle formula will be:

$$(x - x_0)^2 + (y - y_0)^2 = R^2 \tag{2.1}$$

where R is the radius of the circle and x_0, y_0 are the central coordinates of the circle. Integrating the above equation

$$\dot{x}(x - x_0) + \dot{y}(y - y_0) = 0 \tag{2.2}$$

For the sake of simplicity we assume that $x_0 = y_0 = 0$. So we have:

$$y = (R^2 - x^2)^0.5$$
$$\dot{y} = -\dot{x}x/y$$

A "Good" trajectory will be defined such that:

- $\dot{x}(0) = \dot{x}(T) = 0$ where T is the time-period of motion.
- $\dot{x}$ is smooth through the path and doesn't have singularities
- $\dot{y}(0) = \dot{y}(T) = 0$ where T is the time-period of motion.
- $\dot{y}$ is smooth through the path and doesn't have singularities

2.1 Studying Different Functions of $x(t)$

To find a Good $x(t)$ there are a few obvious choices to pay attention to. such as $x = \cos(t)$ or $x = a.t^4 + b.t^3 + c.t^2 + d.t + e$ or trapezoidal.

2.1.1 Cosine(t)

if $x(t) = R\cos(t)$ then

$$\dot{x}(t) = -R\sin(t)$$
$$y(t) = R\sin(t)$$
$$\dot{y}(t) = R\cos(t)$$

This function has a problem, that is at the time instance zero ($t = 0$), the condition $\dot{y}(t) = 0$ is not met (since $\dot{y}(0) = R\cos(0) = 0$)

2.1.2 Polynomial

if $x(t) = at^4 + bt^3 + ct^2 + dt + e$ then we have:

$$\dot{x}(t) = 4at^3 + 3bt^2 + 2ct + d$$
$$y(t) = (R^2 - (at^4 + bt^3 + ct^2 + dt + e)^2)^0.5$$
$$\dot{y}(t) = -(4at^3 + 3bt^2 + 2ct + d)(at^4 + bt^3 + ct^2 + dt + e)/((R^2 - (at^4 + bt^3 + ct^2 + dt + e)^2)^0.5)$$

This also has the problem of

2.1.3 Trapezoidal

The trapezoidal function is:

$$\dot{x} = \begin{cases} at \\ v_{max} \\ a(T - t) \end{cases} \tag{2.3}$$

This function also has the problem of the two previous functions. Meaning $\dot{y}(0) \neq 0$

2.1.4 Deriving a more complex function

Implementing the first guess ($x = R\cos(t)$) that was the smoothest line we got. except that $\dot{y}(t = 0) \neq 0$. so we'll try to find another argument for x and y. since the problem was y at time instance zero, the best choice is to put an argument inside the sin that help with the value when derivation happens. For example $x = R\cos(t^2)$ is a safe bet. Then we have:

$$\dot{x}(t) = -2tR\sin(t^2)$$
$$y(t) = R\sin(t^2)$$
$$\dot{y}(t) = 2tR\cos(t^2)$$

It seems that the problem of $\dot{y}(t = 0) = 0$ has been solved. But as a reminder let's write down the conditions of a "Good" path again:

- $\dot{x}(0) = \dot{x}(T) = 0$ where T is the time-period of motion.
- $\dot{x}$ is smooth through the path and doesn't have singularities

- $\dot{y}(0) = \dot{y}(T) = 0$ where T is the time-period of motion.
- $\dot{y}$ is smooth through the path and doesn't have singularities

Seeing the items above, we notice that $\dot{y}(t = T) \neq 0$, which is not what we want. The solution to the new-found problem is easy, we just have to change the argument of the trigonometric function to $at^3 + bt^2 + ct$. Let's check why this is a good function:

$$\begin{aligned} x &= R\cos(at^3 + bt^2 + ct) \\ \dot{x} &= -(3at^2 + 2bt + c)R\sin(at^3 + bt^2 + ct) \\ y &= R\sin(at^3 + bt^2 + ct) \\ \dot{y} &= (3at^2 + 2bt + c)R\cos(at^3 + bt^2 + ct) \end{aligned} \tag{2.4}$$

If we assume that $a = 1$ and then apply the boundary conditions of $\dot{y}(t = 0) = \dot{y}(t = T) = 0$ we get:

$$\begin{aligned} a &= 1 \\ b &= -1.5T \\ c &= 0 \end{aligned} \tag{2.5}$$

So we can say that:

$$x = R\cos(t^3 - 1.5Tt^2) \tag{2.6}$$

This satisfies the boundary conditions, now we can continue onward with the IK calculation and python implementation

Chapter 3

JACOBIAN METHOD - INVERSE KINEMATICS AND ANGULAR VELOCITY PROFILE

After determining the path of EE movement as a function of time, we use IK and Jacobian to find the angular velocity profile of actuated joints. This will be done in 4 steps as previously mentioned, assuming that the path of EE in 3D-space is as followed:

$$\begin{aligned} x &= R\cos(t^3 - 1.5Tt^2) \\ y &= R\sin(t^3 - 1.5Tt^2) \\ z &= z_0 \end{aligned} \tag{3.1}$$

Where T is the time period needed for the motion to be completed.

Even though we established that $f = t^3 - 1.5Tt^2$, there are still a couple of minor problems with this function. For instance if we try to find T that makes f go from 0 to 2π, we reach: $T = -\sqrt[3]{4\pi}$. which is wrong since time cannot be smaller than zero. So $f = 1.5Tt^2 - t^3$.

This is also wrong because if we put $f(T) = 2\pi$ then it outputs $T = \sqrt[3]{4\pi}$, For every radius of circle imaginable, which is something we don't want. Because if radius is big then the jerk will be high up and if the radius is small then jerk will be small. For bounding the high difference in velocity profile, we should control the overall speed with an additional parameter such as a. So finally $f = a(1.5Tt^2 - t^3)$ and we have:

$$\begin{aligned} x &= R\cos(a(1.5Tt^2 - t^3)) \\ y &= R\sin(a(1.5Tt^2 - t^3)) \\ z &= z_0 \end{aligned} \tag{3.2}$$

3.1 Step 1: EE Velocity profile

From the determined path, the velocity profile of EE is desired.

3.1.1 Theory

In theory the velocity profile is as easy as getting the derivative of the path. Meaning with deriving the Eqn. (3.2) we reach:

$$\begin{aligned} v_x &= \dot{x} = -Ra(3Tt - 3t^2)\sin(a(1.5Tt^2 - t^3)) \\ v_y &= \dot{y} = +Ra(3Tt - 3t^2)\cos(a(1.5Tt^2 - t^3)) \\ v_z &= \dot{z} = 0 \end{aligned} \tag{3.3}$$

3.1.2 T Calculation

As for in the Python implementation, we have some more things to attend to. One of which is calculating T. Or to be more exact both T and a need to be calculated. First we have $f(t = T) = 2\pi$ and from this we get:

$$f(t = T) = a(1.5T^3 - T^3) = 0.5aT^3 = 2\pi$$

As an immediate result of the above equation, we have:

$$T = \sqrt[3]{\frac{4\pi}{a}} \tag{3.4}$$

Another equation is needed to calculate both T and a. The other equation comes from how fast we want our EE to go. we put this number as v_{max}. we know that $\dot{x}, \dot{y}, \dot{z} \leq v_{max}$ at all times. The extremums of a function can be found from its derivative. So $\dot{v_x} = 0$.

$$\begin{aligned} a_x &= \dot{v_x} = \ddot{x} \\ &= -Ra(3T - 6t)\sin(a(1.5Tt^2 - t^3)) - Ra^2(3Tt - 3t^2)^2\cos(a(1.5Tt^2 - t^3)) \\ &= 0 \end{aligned} \tag{3.5}$$

3.2 Step 2: IK of the trajectory

Taking samples of the $t \in [0, T]$ with steps of μ we get $N = \mu^{-1}$ instants of time. (e.g. $N = 101$), we get a time-instance-vector of:

$$\vec{t} = [0, \frac{1}{100}T, ...\frac{99}{100}T, T] \tag{3.6}$$

and a vector of position such as:

$$\vec{p} = [p_0, p_1, ...p_101] \tag{3.7}$$

Where

$$p_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} R\cos((\frac{i}{100}T)^3 - 1.5T(\frac{i}{100}T)^2) \\ R\sin((\frac{i}{100}T)^3 - 1.5T(\frac{i}{100}T)^2) \\ c_0 \end{bmatrix}$$

Using the IK that is done in the previous report we change x, y, z of EE into $\theta_{11}, \theta_{12}, \theta_{13}$, So the final output of this stage is:

$$\vec{\theta}_{\text{actuated joints}} = [\theta_0, \ldots, \theta_{101}] \tag{3.8}$$

Where

$$\theta_i = \begin{bmatrix} \theta_{11i} \\ \theta_{12i} \\ \theta_{13i} \end{bmatrix}$$

3.3 Step 4: Angular velocity profile

Given θ_{ij} from previous step, we can calculate j_{ix}, j_{iy}, j_{iz} and from that we can calculate the angular velocity profile such as"

$$\dot{\vec{\theta}} = J_{\theta}^{-1} J_P \vec{v} \tag{3.9}$$

Chapter 4

JACOBIAN METHOD - PYTHON IMPLEMENTATION AND RESULTS