Industrial Robotics TP2

University of Aveiro

Bruno Silva 98374

Github Repository

December 19, 2023

Contents

1.1 Introduction

This program was developed with the purpose of studying and consolidating knowledge about robotic kinematics, including direct, inverse and differential kinematics, three foundational pillars of any robotic system.

The general goal was to create an 9 degree of freedom(DOF) robot similar to a construction crane, made of both rotational and prismatic joints, capable of following a pre-determined path. The conceptual goal was to simulate the robot that would be in a hypothetical real life scenario assembling big Christmas trees.

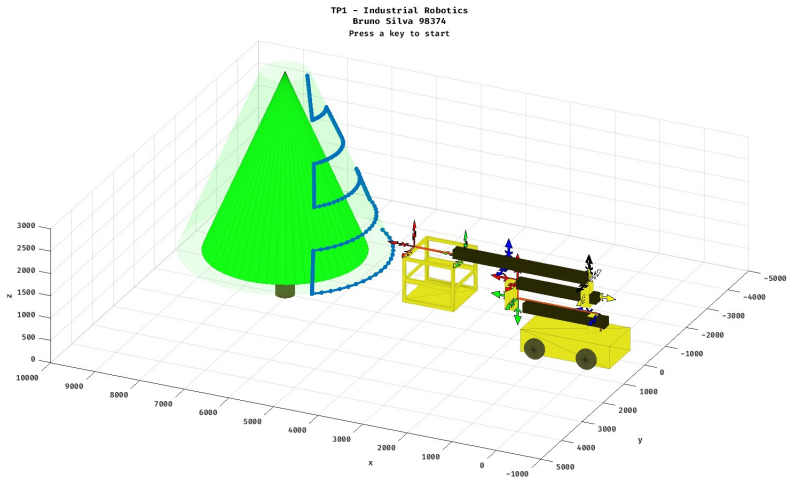

Figure 1.1: Robot and Path to follow.

1.2 Script Analysis

This section is aimed at briefly explaining the steps the script takes in order to accomplish the desired objective as stated in the previous section.

1.2.1 Tree Initialization

The tree models were modeled in *SolidWorks* and exported as *STL's*. Afterwards, a script found ind *Matlab's File Exchange* was used to import the *STL's* into *Matlab* as patch objects. Being patch objects, any geometric transformation can be applied to them. This was done to position the tree at the correct position and modify the offset from the robot.

1.2.2 Computing path and ψ

If the tree had been drawn by computing the equations "by hand", this step would've been easier. The procedure was the following:

1. Find the tip of the tree by visually inspecting the patch object drawn;
2. Compute the end points of each segment based on the tree radius and height;
3. Do linear interpolation for the linear segments
4. Use the **arc3** to compute the point along the circular segments.[1]

To compute the angle of the tool end point (TCP) of the crane, a simple dot product of the vectors made by the points represented in the figure.

The figure represents a cross-section of the tree for a given radius. Based on P_c,P_0,P_i, the angle ψ is computed for each point. Afterwards, to determine the angle signal, the cross-product of the same vectors is computed.

[1] Unlike the first project, here the third point required by the script was quite easy to compute considering all circular segments are half full circles and were always on the *Oxy* plane

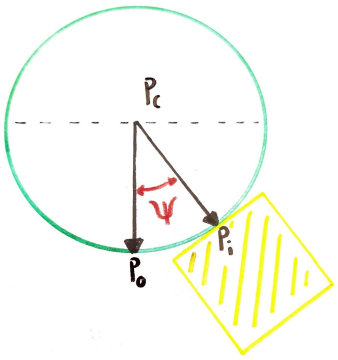

Figure 1.2: ψ computing based on a cross-section of the tree
2

1.2.3 Robot Initialization

The robot dimensions are loaded from a text file that should be present in the main script's path and named *"tp2.txt"*. The joint lengths should be in the format $La = int$. A status message should be displayed to verify if the parameters established by the user are correctly loaded.

In order to establish the ***Denavit-Hartenberg*** parameters, a schematic of the robot was made using a diagram drawing software, *LucidChart*. The reference frames and the resulting **DH table** can be seen below:

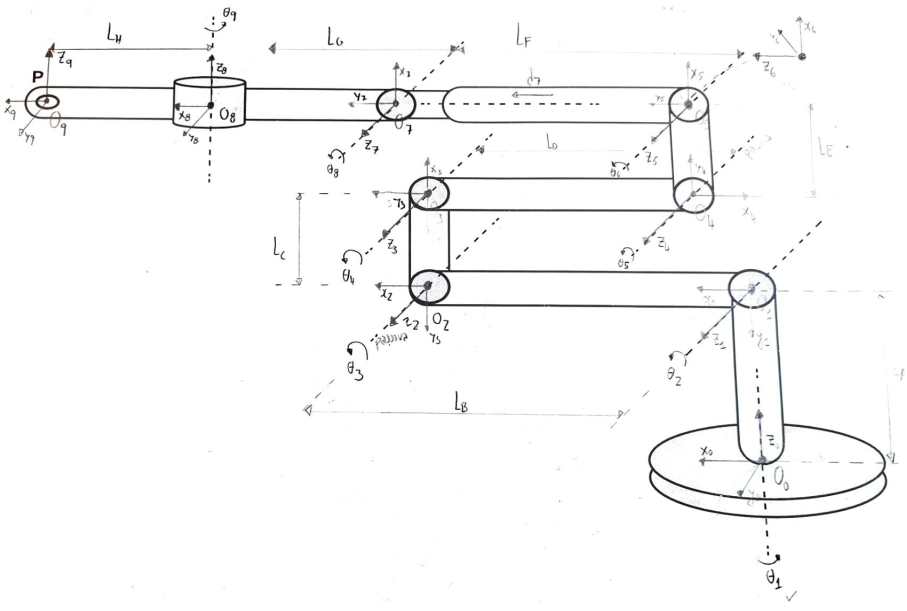

Figure 1.3: Robot schematic and reference frames

Joint	θ_i	l_i	d_i	α_i
1	θ_1	0	La	$-\frac{\pi}{2}$
2	$-\theta_2$	Lb	0	0
3	$\theta_3 - \frac{\pi}{2}$	Lc	0	0
4	$\theta_4 - \frac{\pi}{2}$	Ld	0	0
5	$\theta_5 + \frac{\pi}{2}$	Le	0	0
6	θ_6	0	0	$-\frac{\pi}{2}$
7	0	0	$Lf_{\min} + d7$	$\frac{\pi}{2}$
8	$\frac{\pi}{2} + \theta_8$	Lg	0	$\frac{\pi}{2}$
9	θ_9	Lh	0	0

Table 1.1: DH table

To draw the joints, the same procedure used to load the trees was used. To align them and move them with the robot reference frames, in the *animateRobot* function, the cumulative AA transformation was computed and was applied to each body corresponding to the joint it was representing.

1.2.4 Inverse Kinematics

To compute the inverse kinematics(IK) the approach was the following:

1. Compute the IK of the joints 6 to 9;
2. Compute the θ_1 and update θ_9 based on the ψ angle;
3. Compute the IK of the joints 2 to 6;
4. Stitch the segments together.

Matlab's symbolic calculus capabilities were heavily used to solve the equations showed ahead.
The first segment, from joint 1 to 9, assumed that:

- $\theta_8 = -\theta_6$

With this simplification, the position elements of the AAA matrix from joint 6 to 9 yields the following system of equations :

$$\begin{cases} x_{69} = -\sin(\theta_6)(Lf_{\min} + d7) \\ y_{69} = Lg + Lf_{\min}\cos(\theta_6) + Lh\cos(\theta_9) + d7\cos(\theta_6) \\ z_{69} = Lh\sin(\theta_9) \end{cases} \tag{1.1}$$

Solving this system using Matlab's built-in solver gave the following results :

$$\begin{cases} q_9 = arcsin(\frac{z_{69}}{L_h}) \\ k = L_g + L_h * cos(q_9) - y_{69} \\ q_6 = artan(\frac{x_{69}}{k}) \\ d_7 = \frac{sin(arctan(\frac{x_{69}}{k}))}{-L_{fMin} * sin(arctan(\frac{x}{k})) + x_{69}} \end{cases} \tag{1.2}$$

With this, the robot could go to the desired points, however 2 things were still wrong:

- No orientation relative to the tree
- The prismatic joint limit was being surpassed

This can be seen in the figure below[3].

[3]This is a old commit of the project, a lot of features were still missing and the path for example was being wrongly displayed

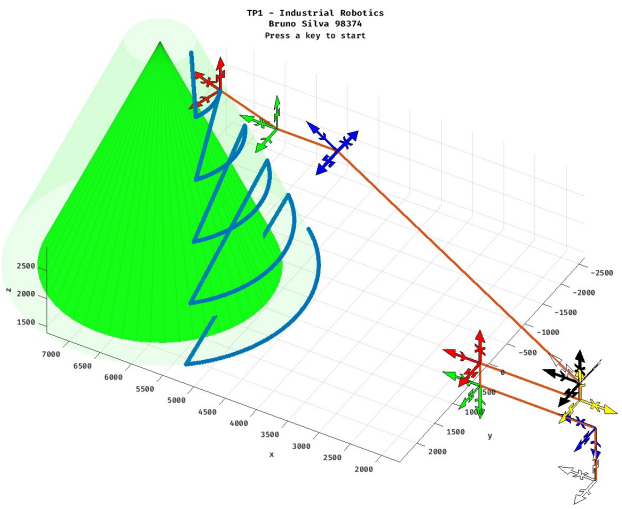

Figure 1.4: KI just with joints 6 to 9 unlocked

After, the next step was to unlock the base joint, to allow for redundancy and establish the desired orientation. The figure below shows a top view of the robot's joints for a given θ_6 and d_7. From the figure using sin's law the equation for $theta_1$ and θ_9 can be deduced. Here, the initial values for θ_6 and d_7 were arbitrated to be the output from the previously computed KI from joint 6 to 9.

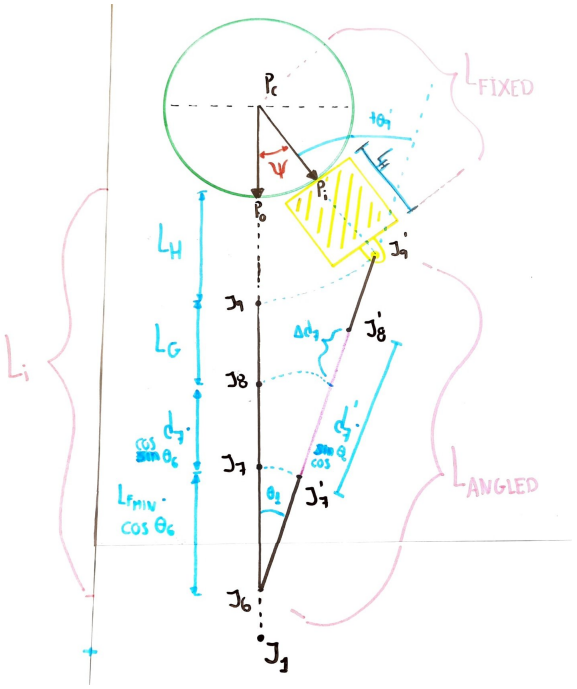

Figure 1.5: Top view of the robot with orientation to the tree

Using the terminology from the figure 1.5, a few things are worth highlighting:

- L_{fixed} is always the same known lengths;
- L_{angled} and L_i change according to the values of the joints 6 and 7;
- The following equations are valid only for the case where the joint 1 and 6 are vertically aligned. This will be further explained after.

Directly from sin's law comes :

$$\frac{sin(\psi)}{L_{angled}} = \frac{sin(\theta_1)}{L_{fixed}} = \frac{sin(\pi - \theta_9)}{L_i} \tag{1.3}$$

From here, L_{angled} is still unknown, as the segment is longer and requires a new value of either θ_6 or d_7. L_{angled} was determined using the cosine's law :

$$L_{angled} = \sqrt{L_{fixed}^2 + L_i^2 - 2 \cdot L_{fixed} \cdot L_i \cdot \cos(\psi)} \tag{1.4}$$

And the updated d_7 was computed like so:

$$d_7 = \frac{L_{angled} - L_{f_{min}} \cdot \cos(|\theta_6|) - L_g}{\cos(|\theta_6|)} \tag{1.5}$$

Solving these equations in order to θ_1 and θ_9 yields the new values, aligning the robot with the tree.

1.2.5 Differential Kinematics

1.2.6 Prismatic Joint Limit Analysis

1.3 Additional notes

- Video links
  - Demo Video
- Github Repository link