ME 5250 – Project Report

Name: Kamalnath Bathirappan

Key Challenges and Solutions

- **Hardware Implementation:**
  Initially, implementing the virtual environment simulation using MATLAB's default IK solver was straightforward, and we achieved this in a short period. However, transitioning from the virtual environment to physical hardware introduced unexpected complexities. These included calibration issues and inaccuracies in the joint angles derived from the mathematical IK solution. Developing a closed-form solution using analytical methods required significant effort, as we had to fine-tune equations to ensure smooth and accurate robot movement. Debugging these issues, especially in real-world testing, took considerable time and perseverance.
- **Obstacle Avoidance:**
  Implementing obstacle avoidance was particularly challenging. While the theoretical aspects seemed manageable, converting mathematical ideas into executable code proved to be complex. Developing a region-based approach for defining boundaries around obstacles required iterative testing and adjustments. These boundaries, defined using the position of the obstacle via April tags, helped ensure the robot could dynamically adjust its trajectory. The entire process required repeated simulations, testing on hardware, and debugging to achieve reliable performance.

Main Contributions

- **Programming and Debugging:**
  One of my primary contributions was converting the analytical IK equations into working MATLAB code. This involved meticulous debugging to address discrepancies between expected and actual results. For instance, I identified and resolved an issue where the solver returned invalid configurations, causing trajectory irregularities.
- **Integration of Mathematical Solutions:**
  I worked extensively on transforming mathematical models into functional code, especially for the inverse kinematics. Ensuring smooth transitions between waypoints required adding intermediate trajectories to avoid abrupt movements. My focus was on bridging the gap between theoretical formulations and practical implementation, which greatly improved the robot's overall performance.

Team Synergy

- My teammate and I divided tasks based on our strengths, which allowed us to work efficiently. I concentrated on programming, debugging, and translating mathematical equations into executable code, while my teammate focused on solving the inverse kinematics problem using closed-form equations and implementing the obstacle avoidance strategy. We consistently supported each other by sharing insights and debugging collaboratively.
- For instance, during the obstacle avoidance implementation, I assisted in refining the region-based logic, while my partner provided valuable inputs to optimize the robot's trajectory. This collaborative approach ensured we completed both the basic and bonus tasks on time.

Learning Outcomes

- This project significantly enhanced my MATLAB programming skills, particularly in robotics applications. Before this, I primarily used MATLAB for plotting and simulations, but now I can confidently use it for advanced tasks such as robot control.
- Working on both the simulation and hardware implementation provided valuable insights into bridging virtual environments with real-world applications.
- I gained a deeper understanding of analytical methods for solving inverse kinematics, which is crucial for mobile and industrial robotics.
- This experience has boosted my confidence in tackling complex robotics problems, and I feel well-prepared to apply these skills in future academic and professional projects.

[BONUS] Obstacle Avoidance:

For the obstacle avoidance, we used a region growing technique on the obstacles to repel the robot when it gets close to its boundary. This boundary is created by using the given dimensions of the obstacle with respect to the April tag position.

Adding on to it for the robot to get across the obstacle we simply used an if-else logic of reachability of the robot. So, while the obstacle is withing the reach of the robot, the robot moves around the obstacle away from its base frame, but if the obstacle is at the other side of its workspace, the robot goes behind the obstacle staying closer to its base frame. This the reachability is calculated based on the length of the end effector frame from its base and the boundary of the obstacle.

Algorithm:

1. Check if the obstacle position is not equal to the origin [0, 0, 0].
2. If there is an obstacle:

   a. Extract the x and y coordinates of the obstacle position.
   b. Determine the buffer based on the obstacle position:

      - If obstacle is near [0.2, 0.2], subtract buffer.
      - If obstacle is near [0.1, 0.1], add buffer.
      - If obstacle is near [0, 0], add buffer.
      - For any other position, use default buffer.

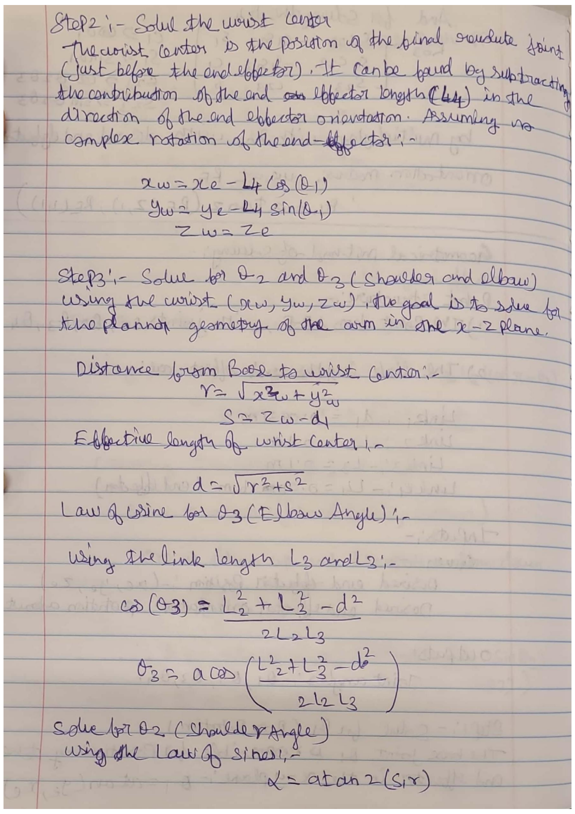
   c. Apply the buffer to the obstacle position.
   d. Generate inverse kinematics for the buffered obstacle position.
   e. Plan and execute trajectory from lift position to buffered obstacle position.
   f. Plan and execute trajectory from buffered obstacle position to drop approach position.

3. If there is no obstacle:

   a. Plan and execute trajectory directly from lift position to drop approach position.

[BONUS] Inverse Kinematic Solver:

Algorithm:

1. 

Step2 :- Solve the wrist center

The wrist center is the position of the final revolute joint (just before the end effector). It can be found by subtracting the contribution of the end ~~eff~~ effector length (L4) in the direction of the end effector orientation. Assuming no complex rotation of the end-effector :-

$$x_w = x_e - L_4 \cos(\theta_1)$$
$$y_w = y_e - L_4 \sin(\theta_1)$$
$$z_w = z_e$$

Step3 :- Solve for θ_2 and θ_3 (Shoulder and elbow) using the wrist (x_w, y_w, z_w), the goal is to solve for the planar geometry of the arm in the x-z plane.

Distance from Base to wrist center :-

$$r = \sqrt{x_w^2 + y_w^2}$$
$$s = z_w - d_1$$

Effective length of wrist center :-

$$d = \sqrt{r^2 + s^2}$$

Law of cosine for θ_3 (Elbow Angle) :-

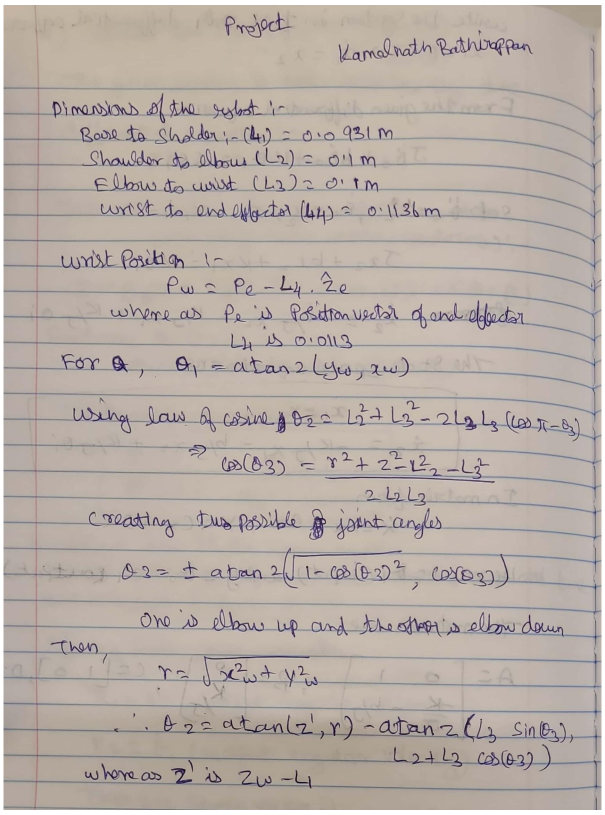
Using the link length L_3 and L_2 :-

$$\cos(\theta_3) = \frac{L_2^2 + L_3^2 - d^2}{2L_2L_3}$$

$$\theta_3 = a\cos\left(\frac{L_2^2 + L_3^2 - d^2}{2L_2L_3}\right)$$

Solve for θ_2 (Shoulder Angle) using the Law of sines :-

$$\alpha = atan2(s, r)$$

2. 

Project

Kamalnath Bathirappan

Dimensions of the robot :-

Base to Shoulder :- (L_1) = 0.0931 m

Shoulder to elbow (L_2) = 0.1 m

Elbow to wrist (L_3) = 0.1 m

wrist to end effector (L_4) = 0.1136 m

wrist Position :-

$$P_w = P_e - L_4 \cdot \hat{z}_e$$

where as P_e is Position vector of end effector

L_4 is 0.0113

For θ, $\theta_1 = atan2(y_w, x_w)$

Using law of cosine $\theta_2 = L_2^2 + L_3^2 - 2L_2L_3(\cos\pi - \theta_3)$

$$\Rightarrow \cos(\theta_3) = \frac{r^2 + z^2 - L_2^2 - L_3^2}{2L_2L_3}$$

Creating two possible joint angles

$$\theta_3 = \pm\, atan2\left(\sqrt{1 - \cos(\theta_3)^2}, \cos(\theta_3)\right)$$

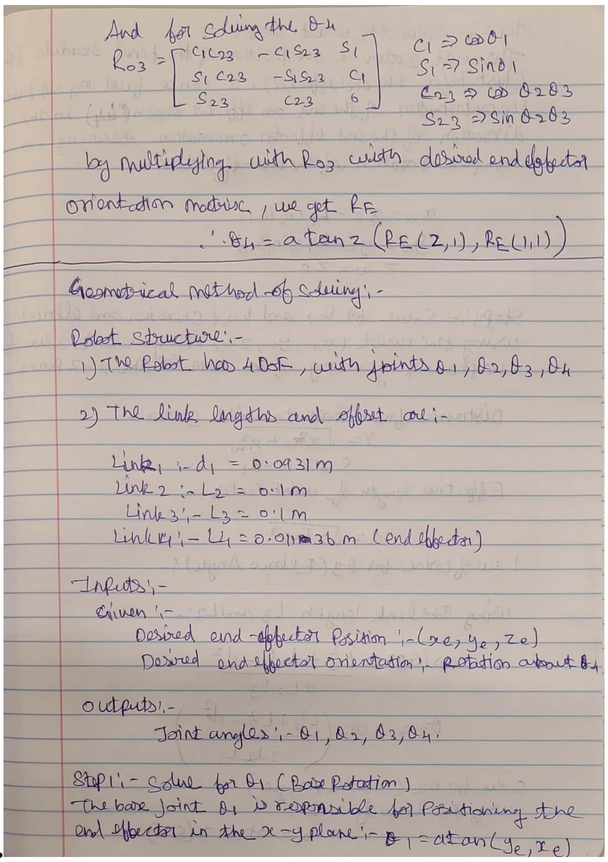
One is elbow up and the other is elbow down

Then,

$$r = \sqrt{x_w^2 + y_w^2}$$

$$\therefore \theta_2 = atan(z', r) - atan2(L_3 \sin(\theta_3), L_2 + L_3\cos(\theta_3))$$

where as z' is $z_w - L_1$

3. 

And for solving the θ_4

$$R_{03} = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 \\ s_1c_{23} & -s_1s_{23} & c_1 \\ s_{23} & c_{23} & 0 \end{bmatrix}$$

$c_1 \Rightarrow \cos\theta_1$

$s_1 \Rightarrow \sin\theta_1$

$c_{23} \Rightarrow \cos\theta_2\theta_3$

$s_{23} \Rightarrow \sin\theta_2\theta_3$

by multiplying with R_{03} with desired end effector orientation matrix, we get R_E

$$\therefore \theta_4 = atan2(R_E(2,1), R_E(1,1))$$

Geometrical method of solving :-

Robot structure :-

1) The Robot has 4 DoF, with joints $\theta_1, \theta_2, \theta_3, \theta_4$

2) The link lengths and offset are :-

Link1 :- d_1 = 0.0931 m

Link 2 :- L_2 = 0.1 m

Link 3 :- L_3 = 0.1 m

Link4 :- L_4 = 0.01136 m (end effector)

Inputs :-

Given :-

Desired end-effector Position :- (x_e, y_e, z_e)

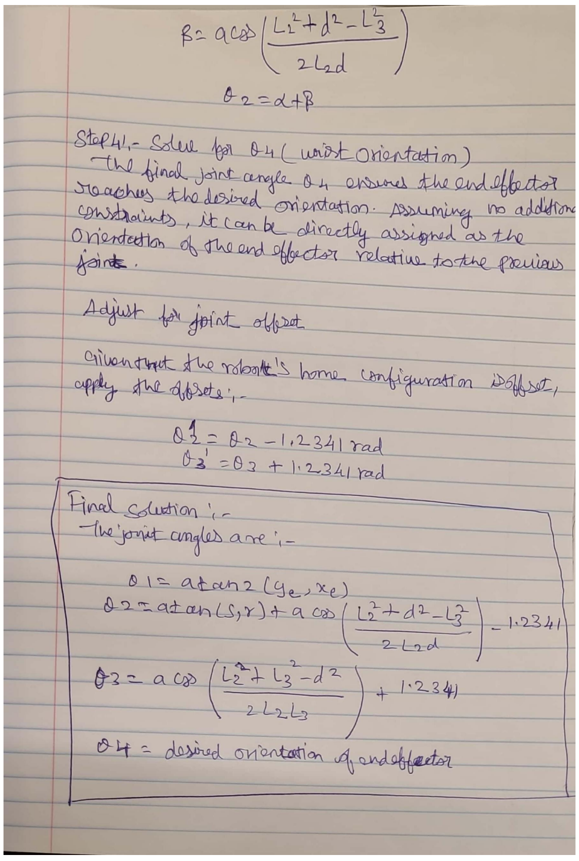
Desired end effector orientation :- Rotation about θ_4

Outputs :-

Joint angles :- $\theta_1, \theta_2, \theta_3, \theta_4$.

Step1 :- Solve for θ_1 (Base Rotation)

The base joint θ_1 is responsible for Positioning the end effector in the x-y plane :- $\theta_1 = atan(y_e, x_e)$

4. 

$$\beta = a\cos\left(\frac{L_2^2 + d^2 - L_3^2}{2L_2d}\right)$$

$$\theta_2 = \alpha + \beta$$

Step4 :- Solve for θ_4 (wrist Orientation)

The final joint angle θ_4 ensures the end effector reaches the desired orientation. Assuming no additional constraints, it can be directly assigned as the orientation of the end effector relative to the previous joint.

Adjust for joint offset

Given that the robot's home configuration is offset, apply the offsets :-

$$\theta_2' = \theta_2 - 1.2341 \text{ rad}$$
$$\theta_3' = \theta_3 + 1.2341 \text{ rad}$$

Final solution :-

The joint angles are :-

$$\theta_1 = atan2(y_e, x_e)$$

$$\theta_2 = atan(s, r) + a\cos\left(\frac{L_2^2 + d^2 - L_3^2}{2L_2d}\right) - 1.2341$$

$$\theta_3 = a\cos\left(\frac{L_2^2 + L_3^2 - d^2}{2L_2L_3}\right) + 1.2341$$

θ_4 = desired orientation of end effector

```
1. Initialize parameters:
    • Set link lengths L1, L2, L3, and L4
    • Set default elbow configuration (up = true)
    • Initialize joint configuration array q with zeros

2. Compute q1 (Waist joint):
    • Calculate q1 using atan2(T(2,4), T(1,4))

3. Perform wrist decoupling:
    • Calculate wrist position by subtracting L4 * T(1:3,3) from T(1:3,4)

4. Calculate intermediate values:
    • Compute r = sqrt(wrist_pos(1)^2 + wrist_pos(2)^2)
    • Compute z = wrist_pos(3) - L1
    • Compute D = sqrt(r^2 + z^2)

5. Compute q3 (Elbow joint):
    • Calculate cos_q3 = (D^2 - L2^2 - L3^2) / (2 * L2 * L3)
    • Clamp cos_q3 to [-1, 1]
    • If elbow up configuration:
        • q3 = atan2(sqrt(1 - cos_q3^2), cos_q3)
    • Else:
        • q3 = atan2(-sqrt(1 - cos_q3^2), cos_q3)

6. Compute q2 (Shoulder joint):
        • Calculate alpha = atan2(z, r)
        • Calculate beta = atan2(L3 * sin(q3), L2 + L3 * cos(q3))
        • Compute q2 = alpha - beta

7. Compute q4 (Wrist angle):
        • Calculate rotation matrix R03 using q1, q2, and q3
        • Calculate R36 = R03' * T(1:3, 1:3)
        • Compute q4 = atan2(R36(2,1), R36(1,1))
        • Wrap q4 to the range [-π, π]

8. Return the computed joint configuration q
```

Sequential Pick, Place, and Reloading

Concept:
The task involved performing a continuous pick-and-place operation, simulating an industrial setup. The robot picks an object from a feed track, places it into a clamp for laser engraving, and then transfers it to a conveyor belt for the next process. This sequence loops continuously as long as objects are fed into the system.

Implementation:

- **Object Detection and Positioning:**
  The robot uses predefined coordinates to pick objects sequentially from the feed track.
- **Trajectory Planning:**
  A trajectory was planned to lift the object, move it to the engraving clamp, and return to the feed track for the next object. This was accomplished using the IK solver and pre-defined waypoints for smooth movement.
- **Obstacle Avoidance Logic:**
  A dynamic boundary-based approach ensured that the robot avoided collisions with obstacles in its workspace. This involved calculating buffer zones around the April tag-detected obstacle positions and adjusting trajectories accordingly.

Algorithm Highlights:

- If an obstacle is detected, a buffer is applied based on its position relative to the robot.
- The robot adjusts its trajectory to move around the obstacle, either closer to or farther from its base, depending on reachability.
- If no obstacle is detected, the robot follows a direct path to complete the pick-and-place operation.

Key Takeaways:
This implementation demonstrated the integration of path planning, obstacle avoidance, and sequential operations in a loop, reflecting real-world automation scenarios in industrial robotics.