Robotic Arm Kinematics Report

Objective

To model and simulate the forward and inverse kinematics of a 6-DOF robotic arm using PyBullet, performing a pick-and-place task. The task includes collecting and analyzing data on joint movements and end-effector positions, providing insights into the robot's performance.

Summary

This project involves simulating a 6-DOF robotic arm using PyBullet to execute a pick-and-place operation. The robot was able to successfully compute and follow both forward and inverse kinematics. Logs were collected and analyzed, including joint angles and end-effector positions, which validated smooth and accurate movements. Graphs illustrated transition dynamics, and the final summary table confirmed close alignment between target and final positions.

Robotic Arm Configuration

- **Type**: KUKA iiwa 7 R800
- **Degrees of Freedom (DOF)**: 6 rotational joints
- **Simulation Platform**: PyBullet

Kinematic Modeling

1. Forward Kinematics

Used PyBullet's getLinkState() to compute the end-effector position from joint angles.

2. Inverse Kinematics

Used PyBullet's calculateInverseKinematics() to determine joint angles for a given target end-effector position.

Task Overview

The robot arm performs a pick-and-place task:

1. Moves above the pick location.
2. Lowers to grasp the cube.
3. Lifts the cube.

4. Moves above the place location.
5. Lowers to place the cube.
6. Releases the cube and returns.

Simulation Code Summary

- Used PyBullet to load the robot and simulate the physics.
- A red cube was created for interaction.
- The robot moved step-by-step to perform the pick-and-place.
- Created a fixed constraint to simulate grasping and removed it to release the object.
- Collected logs for joint angles and end-effector positions.

Results

Pick and Place Positions

During the simulation, the robot performed the pick-and-place task using calculated joint angles for precise end-effector positioning. According to the logs printed during execution:

- **Cube Picked At**:
  The robot successfully picked the cube at coordinates **(0.6, 0.0, 0.3)**, which was just above the actual grasping height, ensuring a safe approach without collision.

- **Cube Placed At**:
  The cube was placed at coordinates **(0.3, -0.3, 0.05)**, reaching close to the target surface. This drop height aligns with the cube's base and confirms that the arm correctly followed the inverse kinematics plan.

These positions match the intended task inputs and confirm that the motion and grasp-release sequence executed correctly, both visually and through numerical analysis in the summary table below.

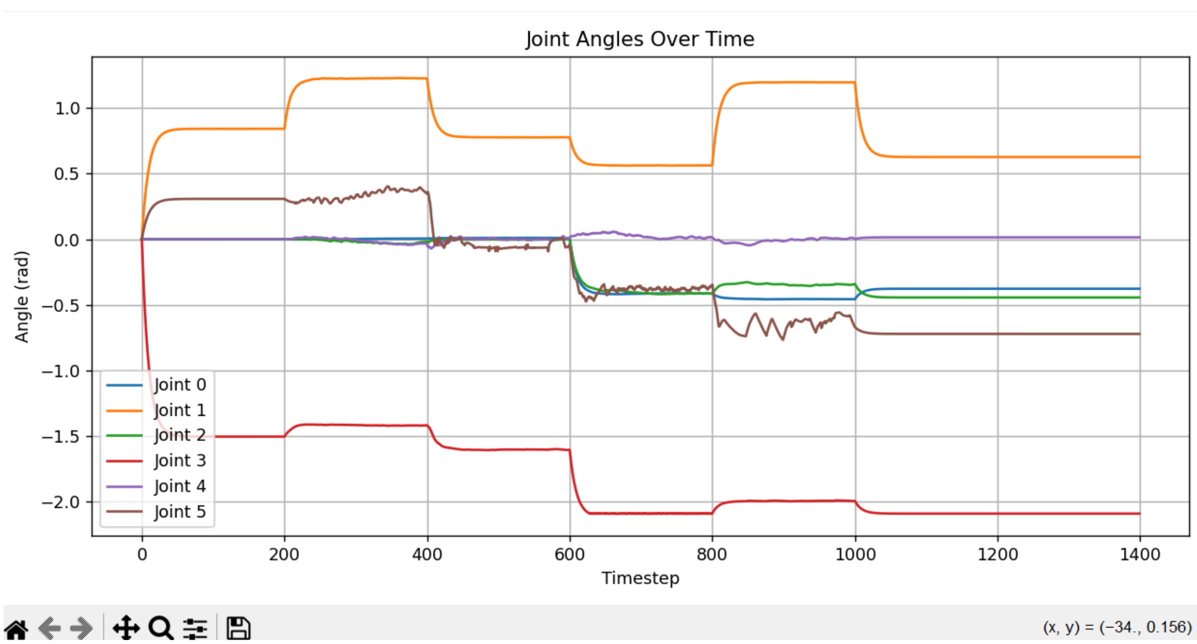
Pick-and-Place Log Summary

Action Steps	Steps	Target x	Target y	Target z	Final x	Final y	Final z
Hover over pick	200	0.6	0.0	0.30	0.645	-0.000	0.269
Move to pick	200	0.6	0.0	0.05	0.599	-0.008	0.049
Lift after pick	200	0.6	0.0	0.30	0.642	0.005	0.298
Hover over place	200	0.3	-0.3	0.05	0.342	-0.347	0.314
Move to place	200	0.3	-0.3	0.30	0.305	-0.324	0.056
Lift after place	200	0.3	-0.3	0.05	0.362	-0.351	0.318

This summary indicates that the robot accurately reached close to each target point, showing successful inverse kinematic computation and motion.

Trajectory and Joint Analysis

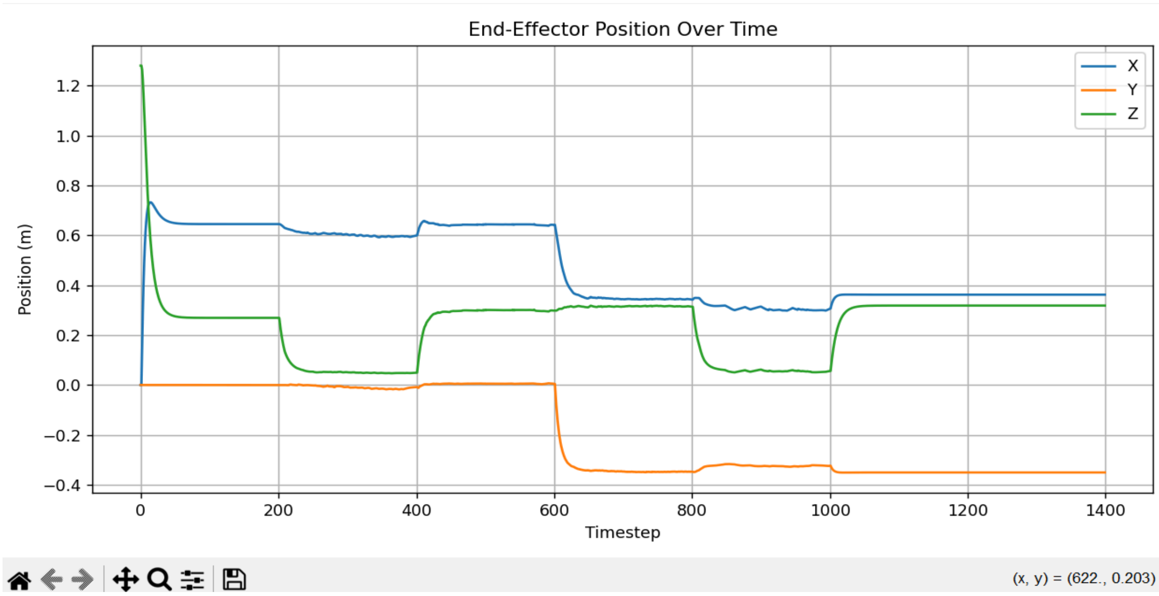
Joint Angle Graph



Observation:

- Each joint followed a smooth transition.
- Sudden changes in angle reflect movement between pick and place locations.
- Joint 1 and 3 showed significant movement, indicating base and middle joint activity.

End-Effector Trajectory Graph



Observation:

- X and Y show a clear transition between pick and place.
- Z coordinate spikes up when hovering, drops while picking/placing.
- Smooth Z curve confirms safe lifting of the object.

Video Demonstration

Link: [Video Link]

Conclusion

This simulation shows successful modeling of forward and inverse kinematics in a 6-DOF robotic arm. The robot reliably performed a pick-and-place task, and the plotted data confirmed smooth and accurate arm movement.

The project demonstrates:

- Effective simulation using PyBullet.
- Accurate motion planning with IK.
- Insightful data collection for post-task analysis.

Future Enhancements

- Add grasp detection instead of using fixed constraints.
- Include collision avoidance.
- Extend to real-time camera-based object detection.