10/19/2021

Milestone #

Robotics (Edpt1009) W’21 course project

Robotics (EDPT1009)

Project Milestone: MS2

Team Number: 8

**Industrial Robotics**

Seif Sabra, Dina Mamdouh, Yusuf Swelam, Anthony Usama, Faris Ebiede

This milestone focused on implementing the forward kinematics of a 5-DOF robotic arm using the Denavit-Hartenberg (DH) convention within the CoppeliaSim simulation environment. The analysis involved defining the DH parameters for each joint, developing a Python script to compute the transformation matrices, and validating the calculated end-effector position against the simulated position in CoppeliaSim. The software design utilized Python and the CoppeliaSim remote API for joint control and data acquisition. Visualization was achieved directly within the CoppeliaSim environment, observing the robot's motion and comparing the calculated and simulated end-effector positions. The results demonstrated accurate forward kinematics calculations, with the calculated end-effector position closely matching the simulated position after the robot reached its target joint configuration.

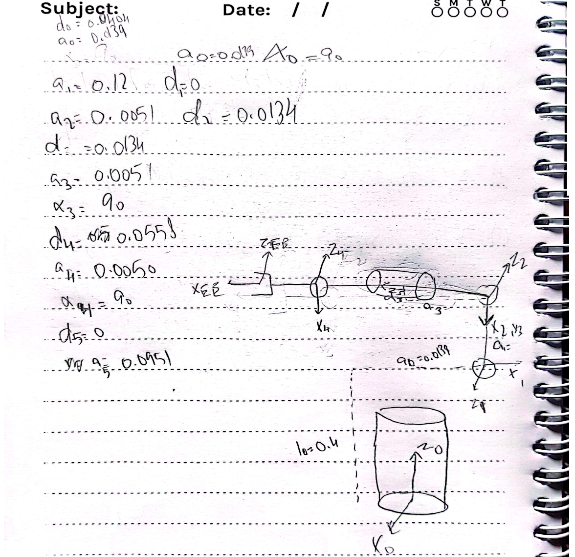
1. INTRODUCTION

The robotic manipulator is designed to seamlessly integrate into existing automated production lines, streamlining the handling of automotive components. This can dramatically increase production efficiency and accuracy, reducing manual labor and potential human error.

Versatile Handling of Automotive Products: The manipulator's ability to handle both circular and angular objects makes it highly adaptable for a wide range of automotive components, such as engine parts, body panels, and interior

Improved Efficiency and Safety: By automating handling tasks, the manipulator contributes to increased production efficiency and a safer working environment. It eliminates the need for workers to perform potentially hazardous or repetitive tasks, reducing the risk of injuries and worker fatigue.

1. TOPIC 02 (EX. ROBOT’S FRAME ASSIGNMENT)



1. TOPIC 03 (EX. DH CONVENTION)

Table 2: DH- Parameters Table

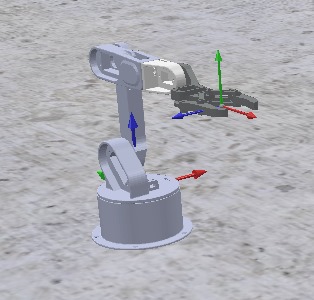
|  |  |  |  |
| --- | --- | --- | --- |
| Theta | d | a | alpha |
| q1 | 0.040 | 0.014 | 90 |
| q2 | 0 | 0.12 | 0 |
| q3 | 0.013 | 0.005 | 90.0 |
| q4 | 0 | 0.095 | 0 |

1. SIMULATION RESULTS

A close-up of a number

Description automatically generated

Initial position:

Final position:

Environment:



1. CONCLUSINS AND FUTURE RECOMMENDATIONS

This milestone successfully implemented the forward kinematics of the 5-DOF robotic arm in CoppeliaSim. The Python script accurately computed the transformation matrices and the end-effector position based on the provided DH parameters. The close agreement between the calculated and simulated end-effector positions validates the correctness of the implemented DH model and the forward kinematics calculations.

However, the current implementation only addresses forward kinematics. A more complete and functional robotic system requires the implementation of inverse kinematics to enable the robot to reach desired end-effector positions. Additionally, trajectory planning algorithms should be incorporated to generate smooth and efficient motions between different configurations. Finally, integrating a control strategy, potentially using Fuzzy Logic Control (FLC) or other advanced control techniques, would allow for precise and robust control of the robot arm in dynamic environments

**Future Recommendations:**

* **Implement Inverse Kinematics:** Develop an inverse kinematics solver to determine the joint angles required to achieve desired end-effector positions and orientations.
* **Trajectory Planning:** Integrate trajectory planning algorithms to generate smooth and time-optimal paths for the robot arm.
* **Control System Design:** Implement a control strategy, such as Fuzzy Logic Control (FLC) or PID control, to accurately track desired trajectories and compensate for external disturbances.
* **Collision Detection and Avoidance:** Incorporate collision detection and avoidance algorithms to ensure safe operation of the robot arm in complex environments.
* **Dynamic Simulation:** Extend the simulation to include dynamic effects such as gravity, inertia, and friction for a more realistic representation of the robot's behavior.