RAS 545 Final Project:
Smart and Accurate Path Planning with the Mycobot Pro 600: Solving Mazes

ANUSHA CHATTERJEE
achatt53@asu.edu
1234397790

ABHIJIT SINHA
asinh117@asu.edu
1234397959

VAMSHIKRISHNA GADDE
vgadde5@asu.edu
1234697141

Article Info

Report Submitted as a part of the coursework requirements of RAS545: Robotic Systems 1

Instructor: *Dr. Mostafa Yourdkhani*
Lab Incharge: *Rajdeep Adak*

ABSTRACT

This report describes how the MyCobot Pro 600 collaborative robot was paired with a digital twin to achieve accurate motion planning and execution. The workflow began in Python with OpenCV, which solved a 4 × 4 maze and extracted the solution's waypoints. After linearly interpolating these waypoints into the robot's coordinate frame, they became the start and end targets for trajectory generation. The path was then segmented into intermediate points, and a MATLAB inverse-kinematics model calculated the required joint angles for each segment. These angles were sent to the robot over TCP, enabling smooth, real-time execution. Rigorous simulation of the digital twin preceded physical runs, ensuring both precision and reliability. Overall, the project showcases the powerful combination of digital-twin simulation, computer vision, and robotic control for practical automation tasks

1. INTRODUCTION

Elephant Robotics' myCobot Pro 600 is a compact, collaborative arm built for a wide range of commercial and industrial uses. Pairing the robot with a digital twin brings major advantages to automation projects: engineers can simulate, analyse, and fine-tune every motion in a virtual space before deployment, cutting errors and boosting accuracy. This project links the physical myCobot Pro 600 to its Simscape Multibody digital twin to plan, test, and validate motion tasks—specifically, guiding the arm through a 4 × 4 rectangular maze. An AI Kit camera captures the maze, while image-processing algorithms identify the solution path and convert it into a sequence of waypoints. Those way-points are delivered to the robot via socket programming, and the digital twin first verifies each move to ensure safe, precise navigation. By merging computer vision, kinematic simulation, and real-time control, the work demonstrates how digital-twin technology can make robotic systems more accurate, adaptable, and efficient when tackling structured challenges like maze navigation.

2. METHOD

- Kinematic Diagram:

We created a kinematic diagram for the MyCobot Pro 600, which visually represented the robot's structure and the relationships between its joints.

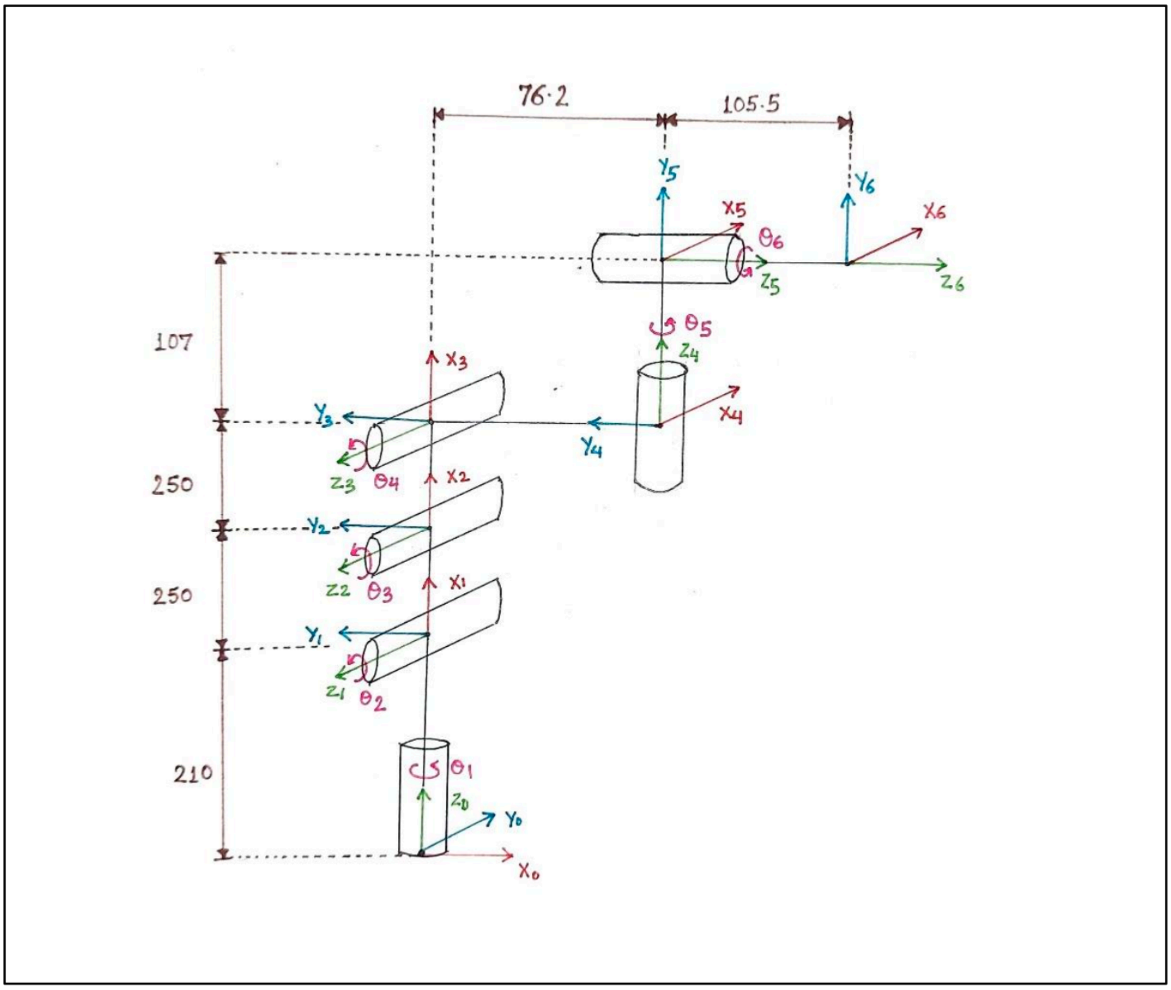

Fig 1 Kinematic Diagram of Mycobot Pro 600

• <u>Homogeneous Transformation:</u>

Homogeneous transformation matrices are derived from the configuration diagram of the robot model. Here, the homogeneous transformation matrices of the MyCobot 600 pro robot are generated manually from the configuration diagram shown in Figure 1, and the corresponding transformation matrices obtained are as follows:

We multiplied the individual HTMs in MATLAB to obtain:

$$^{0}H_6 = {}^{0}H_1 {}^{1}H_2 {}^{2}H_3 {}^{3}H_4 {}^{4}H_5 {}^{5}H_6$$

```
H01 = [cos(t1)      0     sin(t1)     0;
        sin(t1)      0    -cos(t1)     0;
        0            1       0         210;
        0            0       0         1];

H12 = [-cos(t2)   -sin(t2)    0        -250*cos(t2);
        -sin(t2)    cos(t2)    0        -250*sin(t2);
         0           0        -1         76.2;
         0           0         0         1];

H23 = [-cos(t3) -sin(t3) 0        250*cos(t3);
        -sin(t3)  cos(t3) 0        250*sin(t3);
        0         0       -1       76.2;
        0         0        0       1];
```

```
H34 = [-cos(t4)    0   sin(t4)     0;
        -sin(t4)    0  -cos(t4)     0;
        0          -1      0        76.2;
        0           0      0        1];

H45 = [cos(t5)    0    sin(t5)     0;
        sin(t5)    0   -cos(t5)     0;
        0          1      0         107;
        0          0      0         1];

H56 = [-cos(t6)    sin(t6) 0      0;
        -sin(t6)   -cos(t6) 0      0;
        0          0        1      109.5;
        0          0        0      1];
```

- <u>Robot Modeling in MATLAB</u>
- Robot Definition**:** We first define a robot model for the MyCobot Pro 600 using MATLAB's Robotics System Toolbox. This can be done by either importing a URDF file that specifies the robot's geometry and joint parameters or building a `rigidBodyTree` model programmatically.
- Forward Kinematics (FK)**:** Once the model is defined, the built-in FK utilities in the Robotics System Toolbox enable quick verification of the end-effector pose for any given set of joint angles. This step ensures the model is consistent with the real robot's configuration.

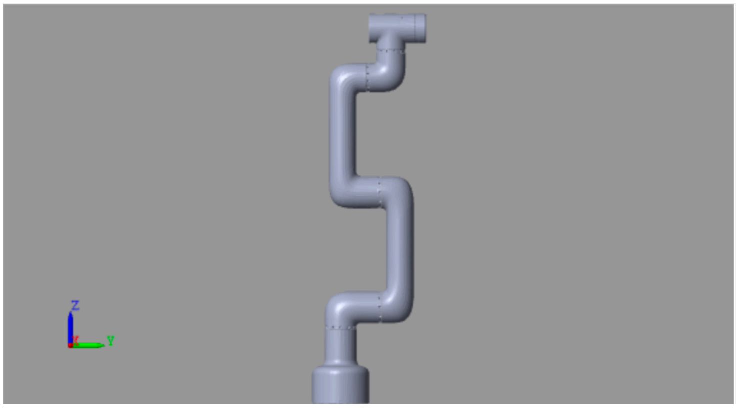

Fig 2 Digital Twin of mycobot Pro 600

- <u>Maze Acquisition</u>
- **Live feed** is opened with OpenCV (`VideoCapture`).
- Operator presses **c** to capture a distortion-free frame.
- Frame is auto-cropped: `rgb2gray → imbinarize → regionprops` finds the largest connected component (the maze board) which is then rotated to a canonical orientation.

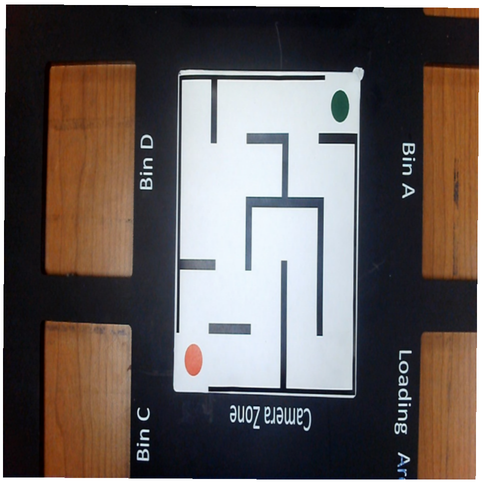

Fig 3 After camera setup

• <u>Maze Solution</u>

Acquired a maze image with the AI-Kit camera.

Developed an image-processing pipeline that:

- Extracts maze walls and corridors,
- Locates the entry and exit nodes(Identify red blob and green blob), and
- Solves the maze with a Dijkstra algorithm to obtain the optimal route.

Translated the resulting pixel-based path into real-world coordinates by referencing the calibrated camera workspace.

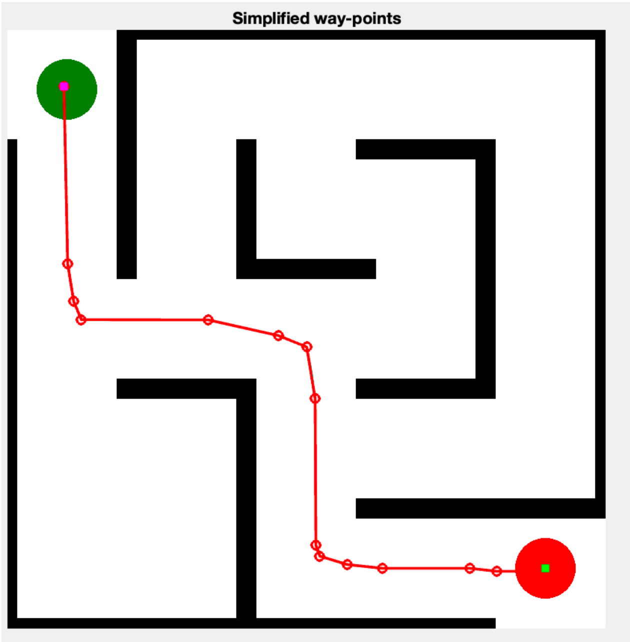

Figure 4: Finding the shortest path between two blobs

• <u>Path Planning & Validation</u>

- Extracted the solution path as straight-line way-points.
- Densified the route with intermediate targets to guarantee smooth, continuous motion.
- Fed each target into a MATLAB inverse-kinematics solver to obtain the required joint angles.
- Ran the full trajectory inside the digital-twin model, confirming both reachability and collision-free execution before deploying to the physical robot.

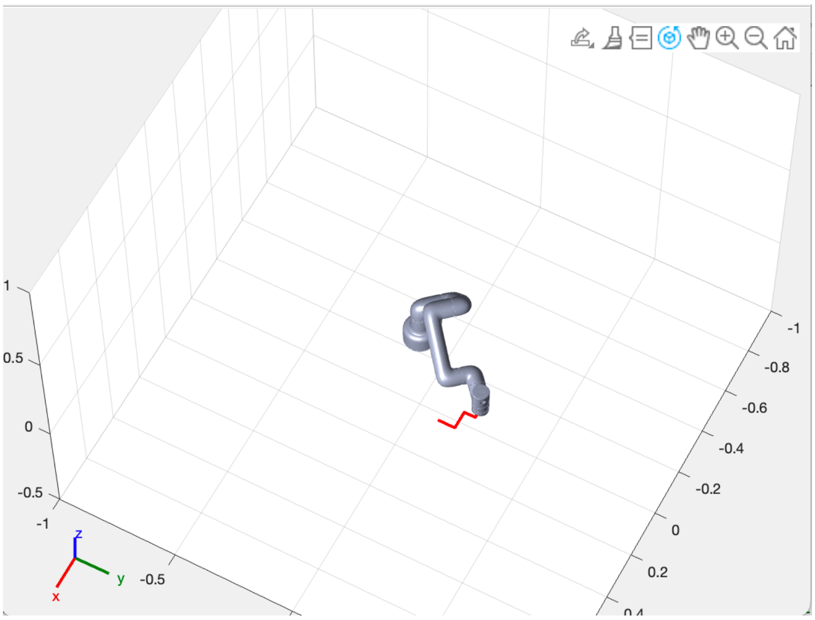

Fig 5 Digital Twin result

- <u>Motion Execution</u>

- The joint-angle sequence generated by the inverse-kinematics solver was streamed to the myCobot Pro 600 over a TCP link.
- A Python script handled the socket communication, allowing the robot to trace the straight-line trajectory between the two markers precisely as planned.

- <u>Comparative Analysis & Final Validation</u>

- Logged the end-effector coordinates during the live run and cross-checked them with the positions predicted by forward kinematics in the twin model.
- Investigated and corrected any deviations, ensuring the virtual simulation and the physical myCobot Pro 600 remained in close alignment.

Fig 6 Demonstration Picture

Here, I am attaching the detailed link to the Final Project with a video
https://drive.google.com/drive/folders/1z-Yw3Gj1HG1zU_HUEZpT56SMKt-uV438?usp=drive_link

3. CONCLUSION

This project successfully tackled the task of guiding a myCobot Pro 600 through a 4 × 4 rectangular maze. By fusing computer-vision techniques, kinematic modelling, and a digital-twin simulation, the system could autonomously plan and execute an accurate route. The digital twin was essential for validating motion plans, eliminating errors, and guaranteeing safe deployment on the physical robot.

Major milestones included:

- Crafting a reliable maze-solving algorithm.
- Precisely converting image-plane coordinates into the robot's workspace.
- Streaming the final joint-angle sequence over a lightweight socket interface for smooth, real-time execution.

These results showcase how the synergy between robotics and simulation can solve structured problems with both adaptability and precision. Beyond underscoring the versatility of the myCobot Pro 600, the project highlights the critical role of simulation-driven testing in modern robotic automation and lays a solid foundation for tackling larger, more complex scenarios that leverage digital twins in real-world applications.

4. REFERENCES

[1] Elephant Robotics, "MyCobot Pro 600 Documentation," Available: https://docs.elephantrobotics.com/docs/gitbook-en/2-serialproduct/2.3-myCobot_Pro_600/2.3-myCobot_Pro_600.html. Accessed: Sep. 14, 2024.
[2] Elephant Robotics, "Environment Building for ROS1," Available: https://docs.elephantrobotics.com/docs/pro600-en/12-ApplicationBaseROS/12.1-ROS1/12.1.2-EnvironmentBuilding.html. Accessed: Oct. 12, 2024.