TEAM BOLLYWOOD

CSCI 3302: Introduction to Robotics
Recitation 012
December 2021

Final Project Report

PROJECT NAME

Magnificent Mavic Mazerunner

TEAM MEMBERS (Team Bollywood)

Name	Email
Ethan Meyer	Ethan.Meyer@Colorado.edu
Etash Kalra	Etash.Kalra@colorado.edu
Ishika Patel	Ishika.Patel@colorado.edu

ABSTRACT

We have created a multi-robot simulation that leverages color detection, image analysis, shortest path Algorithms, and robot odometry to allow two robots to work together to complete a maze in uncharted territory.

The Mavic robot has the responsibility of flying over and mapping the built out maze as well as identifying both obstacles and target objects. The Mavic takes an image of our map, and then we use this pixel representation to perform pixel analysis to better understand the maze. We implemented color detection on an aerial image in order to transform our image into a workable matrix to perform path planning on. While the drone is flying, the ground robot (ePuck) is in a waiting state until prompted by the Mavik to begin navigating the maze. The ground robot then navigates along waypoints produced by our path finding algorithm.

Applications of this final project could include autonomous robot driving based on real life obstacle data such as in military applications. In unsafe areas we could minimize human interaction by allowing robots to communicate and traverse unknown landscapes. This project started with us pitching a completely different idea, but with some creativity we are proud to deliver this Drone/Ground-Robot system implementation!

EQUIPMENT

Webots Software with the following components:

- E-Puck Robot
- DJI Mavic 2 Pro: https://cyberbotics.com/doc/guide/mavic-2-pro

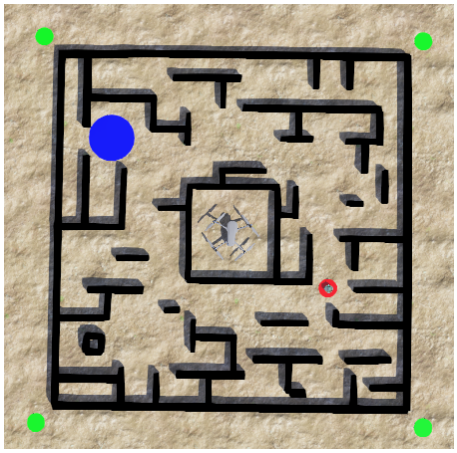
Python Libraries

- `import pdb`
- `import pickle`
- `import random`
- `import copy`
- `import cv2`
- `import numpy as np`
- `from simple_pid import PID`

DELIVERABLES & IMPLEMENTATION PLAN

Over this project and for the entirety of class, our team has been very lucky to have the ability to meet in person to program the labs. This allowed us the ability to engage in group programming and group debugging. Each of us had a hand in every part of the final project and what features to implement as well as how.

1. **Create Webots World** – Lead: Ishika
    a. Create Webots ground world as a complex maze that has marked targets for the four corners of the maze, a starting point marked and an ending point marked
    b. Test: Ensure the world is a maze that is able to be navigated by a ground robot (ie there is enough space between walls to allow the robot to move through them)
    c. **Implementation/Debugging:** Complete. We initially started off with a Webots scenic world with specified obstacles in order to control some of our environment for testing. As our idea evolved, this led to the implementation of a webots maze in order to test a different version of path planning and object detection. We continuously tweaked the world in order to better fit the parameters of our exploration. For example, we adjusted the world's JSON file in order to adjust the thickness of the maze walls for multiple wall objects as opposed to manually changing each setting.



***Green** = Maze Boundaries, **Red** = Start, **Blue** = Goal*

2. **Create Mavic Vision System** – Lead: Ethan
    a. Install OpenCV Python Package & Verify it works with simulated webcam
    b. Understand how to manipulate both motor and camera controls of DJI Mavic 2 Pro
    c. Attach camera add on to the Mavic robot
    d. Fly the robot to elevation to take a photo of the complete maze
    e. Test: Ensure the full environment is pictured in the camera range of the Mavic robot. Program should save an image of the world to the local file system.
    f. **Implementation/Debugging:** Complete. A massive learning curve that we all faced was understanding how to move the Mavic quadcopter since it utilizes space in all three dimensions. We all learned how to manually operate the robot and fly it; this came with some pretty hilarious failures of practically breaking the robot on the ground, changing its motor inputs so poorly that it would not lift off and often sending the robot into unrecoverable spirals. Thankfully, we were able to find Webots' own basic implementation which helped us determine appropriate input parameters. We also experience some difficulties in adjusting the drone's camera to get the best image. We had to think about shadows from light sources as well as image disfigurement due to the altitude and camera angle. We considered trying to perform matrix transformations to automatically adjust for angled pictures but we were not able to implement this feature. In the end, we were able to get the quadcopter flying and save the images it took.

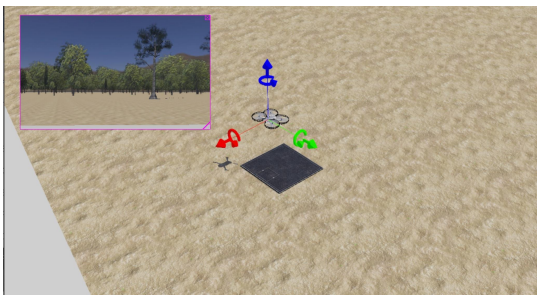
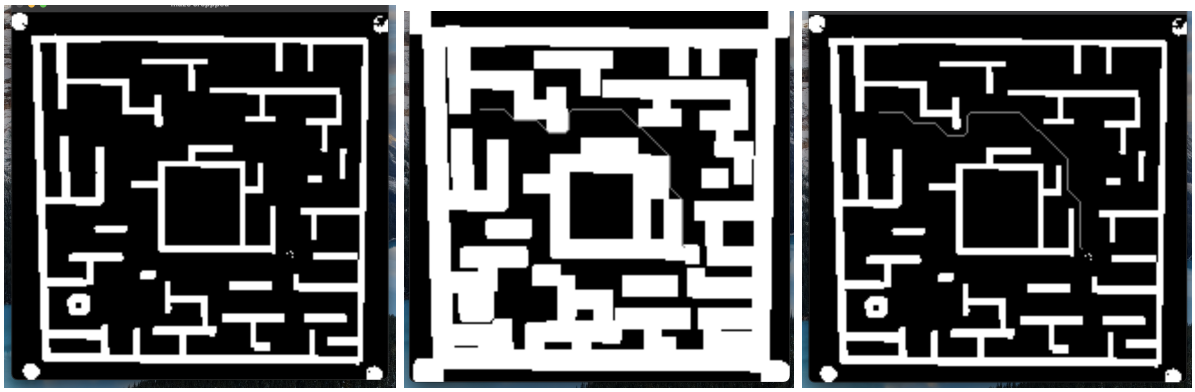


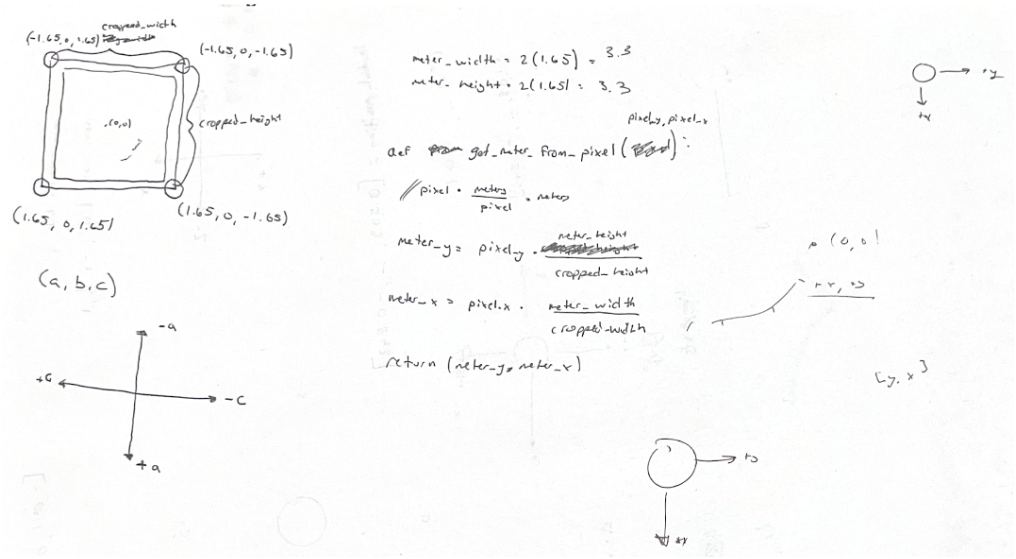
Image of Quadcopter Moving and Camera View

3. **Maze Detection Python** – Pair Programming: Etash and Ishika
    a. Amplify the implementation of Homework 3 that we used in order to leverage this technology for a robot to take an exact certain path without having seen the environment.
    b. Create different color masks in order to detect the position of start points, end points, corners of the world, and the maze boundaries themselves
    c. Image cropping feature to allow us to path plan more effectively given an image: a mask as input and output a cropped mask based on the min/max x/y values
    d. **Implementation/Debugging:** Complete; our mapping Python file outputs different images of computed masks. We used the HW 3 and Lab 5 templates as a starting point and extrapolated to incorporate different features. We used different image masks as in HW 3 to create positional images of each of the features. Using blob detection we were able to locate the pixel coordinates of the start and goal regions. We implemented a completely new feature to crop an image once we have computed the various image blobs into color masks. Debugging came in the form of running the python file and seeing which features were showing up on the masks and if the mapping of the image to a matrix was accurate. We sometimes had blunders in traversing different masks to create a final mask. With the maze crop feature, the camera needs to be ontop of the maze and in full view in order to get the correct image.



From Left to Right: Cropped Maze Mask, Padded Maze Mask, Maze with Path in White

4. **E-Puck Path Planning** – Ethan
    a. Design and implementation of A-star algorithm with euclidean distance heuristic to find path from start to goal in binary matrix produced via mazed detection
    b. Sample every 20 points along path to reduce jitters with feedback controller
    c. Develop pixel to meter function to convert path to list of waypoints for ePuck to visit along path to goal
    d. Implementation/Debugging: There were two primary issues we faced during this phase of the project. First, it was difficult to determine an appropriate mapping from pixels to meters as well as the right rate to sample points from. Second, it was very hard to align all the different perspectives of axis frames with each other - especially between ePuck's coordinate frame and that of the maze.
    e. Implementation Scratch Work:



5. **E-Puck Robot Navigation and Path Planning System** – Pair Programming: Etash, Ishika and Ethan
    a. Creation of E-puck Feedback Controller with help from principles of Lab 3 applied to new robot
    b. Test: See if the E-Puck is moving in the world.
    c. **Implementation/Debugging:** Up to this point in the lab, we are able to populate a numpy array with the pixel path to take. During this piece we manipulated the e-puck controller to move, and the Mavik controllers to send information. We had some major problems with getting the epuck to move because there were two controller files to work between. We had to debug some file I/O to allow for communication of the robots regardless of if they are in range of one another. The most challenging aspects were converting the pixels to meters for directions, and configuring the E-Puck to move.

GitHub Repository Link

https://github.com/Etash47/Robotics-2D-Mapping-DJI-Mavic-Drone