Quick Start Guide, Troubleshoot Guide, and High Level System Overview Version 1.0

Keegan Kelly

July 15, 2022

Contents

1 Quick Start Guide

Before starting ensure that the current camera has been calibrated (if you change camera or focal length re-calibrate), each robot has the correct code on it with the right id selected in RUMME.ino, and each ESP8266 has code uploaded with the right WiFi SSID and password. Also ensure the path given does not send any robot outside of the camera's view. It is also recomended to use a fairly large time step for the path (0.5-1s). Also make sure the origin is roughly in the center of the camera frame. This is not needed but will help with setting bounds on your path since it will be symetrical in +/- x and +/-y (not both since the aspect ratio is not 1:1).

1.1 Step 1: Start the Server

Navigate to the server folder and then AgentData-api-server subfolder. Open a terminal window within within this folder and type the following two commands in the order shown.

```
npm cache clean —force
npm start
```

1.2 Step 2: Starting the Tracking

Navigate to the hostSystem folder and open a terminal window within the folder. Then run the main.py

This script will then prompt you to type y/n for if each agent is being used. Wait a few seconds and a window should open with feed from the camera. Ensure that the origin marker is not covered, has been correctly observed (the corners line up), and that the axis drawn on it is appropriate (if the axis has proper on z orientation). If the origin marker has not been identified well, click into the video feed and hold the r key until you see the axis/marker outline move to a more appropriate observation.

1.3 Step 3: Turn on the Robots

Turn on the Robots and ensure that they are not in programming mode. If the robot moves to the first position in the path it is good. Once all agents being used are in the first position of the path they will send that to the server and they will get a signal to start traveling the desired path.

1.4 Step 4: Watch the Robots

While the robots are executing the desired path, the terminal where the python script was run will display the (x, y, θ) of each agent and the webcam feed will show the axis of each ArUco marker and the frame rate of the processed video. If any robot moves outside of the frame the localization will not work so make sure that all of the markers are in frame (it doesn't matter if the origin is covered).

1.5 Step 5: Shutdown

After the robots have traveled the path turn them all off. Then shutdown the Python script by clicking on the camera window and pressing q on the keyboard. Finally go in the terminal where the server is being ran and press "ctrl + c" on the keyboard then then y and enter. It is recommended that the server be shutdown and the cache cleaned after each operation. Having too many requests sent to the server or too many sent at the same time can cause the server to crash with a EPERM operation not permitted error.

2 Troubleshooting

2.1 All Robots Stop Moving

If the robots all stop moving after they move to a target position, it is likely that they are waiting for their localized position and that the server has crashed with a EPERM operation not permited error. Restart the server and try again. If the issue keeps happening look into how this error is caused and try to fix the server (I couldn't find anything but your millage may vary).

2.2 Inaccurate CameraTracking Position Estimate

If the relative location of the estimate is right, but the scale of all estimates is off (too small or too large) the size of the marker might be wrong or the markers might not be the same size. Measure each marker and ensure that the width agrees with the width given as an argument to the Tracker class constructor in main.py.

This may be caused by a bad position estimate of the origin. Hold r on the keyboard until the estimate of the origin marker changes on the webcam feed. If the axis and outline aren't changing the estimate is good. This issue also might be caused by poor calibration of the camera, re-calibrate and see if the issue persists. For more detailed instructions on calibrating the camera, check the Localization and Networking Documentation.

2.3 Marker Detection is Not Reliable or Position Estimate Has High Noise

If the marker corners are flickering back and forth or the axis of the markers are not stable the this will add noise to the position estimate. This has the same solution as markers not being detected reliably, increasing the size of the markers. Before printing and remounting all the markers, try bringing the markers a little closer to the camera and see if the noise or detection improves.

2.4 Robots Not Driving to Desired Location

If the robots are over steering, driving too far, or not far enough, the odometry is not accurate and should be re-calibrated. Check the control documentation for more detailed instructions on calibration.

2.5 Arduino Networking Problems

Connect to USB and watch the serial monitor to see if the requests are going through or connect another Arduino to the serial of the ESP8266 and watch the serial monitor to watch the requests on that end. If there are issues with the requests and address strings or the JSON is cut short, there is not enough memory allocated. To see how much memory is needed, replicate the JSON document on a test script and call the attribute .memoryUsage(). Then increase the memory a little beyond this result for a buffer.

2.6 Server Issues

If the server is being problematic and is crashing (in my testing crashes were very infrequent) consider building a new server if you have some experience with web development (all it needs is support for storing JSON's and processing HTTP requests), or trying a different prebuilt api server.

3   High Level System Overview

3.1   Server

The server is a JSON server that will facilitate HTTP requests. With this the Python script and the ESP8266's can send data to eachother with the server as the intermediary.

3.2   Python Script

The main.py script will use the webcam and openCV to look for the ArUco markers on top of the robots and on the floor. It will use the location of these markers in the frame, the known size of the markers, and the focal properties of the camera to calculate the location of each marker relative to the marker with id 10 which will be used as the origin. The origin will have its location in the frame stored between iterations so if it is covered, the location of the robots can continue to be tracked. It will then send the position estimates of the markers to the server. This script is also responsible for sending the path from the excel file to the server and checking if each robot is in ready position and then sending a signal to the server that the robots are ready to move.

3.3   Robots

Each robot is equipped with a ESP8266 WiFi module which will be communicated with over the Arduino's serial line to send HTTP requests to the server. The robot tracks its own position with odometry by counting the number of ticks on the wheel encoders and using physical dimensions and calibration parameters to convert that to linear and angular displacement.

The main script on the robot will get a path from the server, get a localized position, move to the first position in the path, tell the server that it is in a ready position, and then continually check the server to see if all of the robots are ready to go. When it gets the signal to go it will then drive the path driven and get a localized position after each segment of the path is completed. The path it gets is segmented into smaller chunks due to memory limitations. When it reaches the end of the path chunk, it will get the next chunk and repeat until it has completed the entire path.

3.4   ArUco Markers

The markers on the robots and the origin use the black and white squares to encode a binary number. Each agent and the origin are given a unique id on the marker. The origin has marker 10, agent 1 has marker 11, agent 2 has marker 12, and agent 3 has marker 13. If you need to print of new markers open the file from the ArUco Markers folder in Google Chrome and print from there (Windows' built in photos tool scales the marker poorly for printing resulting in a slightly blurry image). If you want to switch marker ids or generate new markers there is a script in the same folder to do so.