3D pose estimation of known windows

1st Venkateshkrishna
Masters in Robotics
Worcester Polytechnic Institute
Worcester, MA 01609
vparsuram@wpi.edu

2nd Athithya, Lalith
Masters in Robotics
Worcester Polytechnic Institute
Worcester, MA 01609
lnavaneethakrishnan@wpi.edu
(using 2 late days)

3rd Gampa, Varun
Masters in Robotics
Worcester Polytechnic Institute
Worcester, MA 01609
vgampa@wpi.edu

***Abstract*—This project focuses on estimating the free space in a window and flying the drone through them. The approximate locations are given which can be leveraged to use a start point to estimate the position of the windows more accurately. Then the drone estimates the location it needs to go to and then navigates to the desired location without crashing into the window. The flight ends after it passes through all the windows. This project builds on top of the previous project 3a, wherein the pipeline to detect the pose of the window with respect to the drone is given.**

I. Introduction

In this project, we describe the pipeline developed and used to detect the position of the window with respect to the drone camera. First we train a neural network to detect the corners of the window. After training is done, we calibrate the monocular camera of the drone to get its K matrix and distortion parameters. Finally, all of this is brought together, wherein the image is taken by the drone, and then the image is first rectified using the distortion parameters, the pixel coordinates of the corner points are estimated by the neural network, and finally, using the PnP function, which uses these image pixel coordinates, the K matrix and actual positions of the corner points with respect to window center, the position of the window center with respect to the camera is calculated. Now after this pipeline is it is integrated into the navigation pipeline. As per the approximate map of the environment we know the rough locations of the windows. We direct the drone to go to a location in front of the approximate location so that the drone view the window. Then it detects the position of window and directs it to pass through it's center and move some fixed distance away from the window through it's center. This is repeated for the three windows. In the subsequent section, the detection and navigation pipeline is described.

II. Using Efficient net-B0 for learning

Considering that we needed to evaluate a deep CNN on the Nvidia Jetson Orin Nano during runtime, we chose to select a relatively light network with a lesser number of parameters, but still complex enough to understand the data in various adverse conditions. For this reason, we chose to work with the efficient net-B0 model. Efficient net B0 has just about 4.5 million parameters compared to 11 million parameters of Resnet 18 but performs much better. Currently, it is considered the state-of-the-art CNN model, for classification, and it is used in the backbone of many architectures.

Hence we used the efficient net for generating 8 outputs which correspond to x and y coordinates of the 4 corner points. We used rmse loss function and starting with the pre-trained weights trained the entire network.

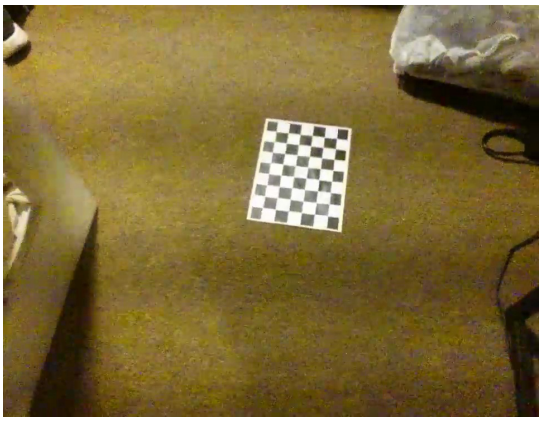

Fig. 1. Checker board pattern used for camera calibration

III. Camera Calibration

We used the checkerboard pattern printed on an A4 size paper. We used a checkerboard pattern, with a 30 mm square size to generate the K matrix and the distortion parameters. This was done using the vision toolbox in Matlab. The checkerboard pattern used for calibration is shown in figure 6. The K matrix obtained is:

$$\begin{pmatrix} 940.5992 & 0 & 470.5372 \\ 0 & 956.1483 & 359.2646 \\ 001.0000 & & \end{pmatrix}$$

During runtime of the drone, it appears that camera's output is a bit dim, but our detection algorithm appears to work well nontheless and hence is not refined.

IV. Pose Generation

To generate the pose of the window. Opencv's cv2.pnp is used. PnP or perspective n-point is a method to estimate the pose of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the

Fig. 2. Image in runtime

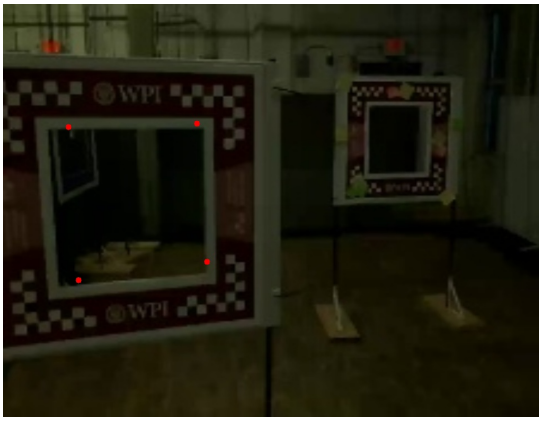

Fig. 4. window2 detection in runtime

Fig. 3. window1 detection in runtime

Fig. 5. window3 (occluded window) detection in runtime

image. The camera pose consists of 6 degrees of freedom (DOF) which are made up of the rotation (roll, pitch, and yaw) and 3D translation of the camera with respect to the world. For this to work efficiently 4 points are needed. In Opencv's implementation, we get the position of the world with respect to the camera frame.

V. Navigation logic

We first parse through the text file which gives the approximate locations of the windows. Then we move the drone to go 150 cm in front of the approximate location. It then captures the image, which has window in it. Using the method described, it calculates the position of the center of the window. Next the command is given to go 100 cm further along the line joining the position of the drone and the estimated center of the window. This process is repeated till all the windows are covered.

VI. Results

We see that the drone is able to navigate through the free space in the windows. Based on the estimated positions we recreated the map in blender and it agrees to a reasonable extent the real world. We have shown the images and their pose inference from the drone for all three images.

Poses of the drone, that is pose before viewing the window and pose after going through the window.

window 1

pose viewing W1	pose crossing W1
0.65	0.41
1.68	2.79
1.36	1.7

window 2

pose viewing W2	pose crossing W2
-0.55	-0.5
3.42	3.75
1.36	1.35

window 3

pose viewing W3	pose viewing W3
0.6	0.2
5.59	6.45
1.36	1.12

Reconstructed map based on estimates:

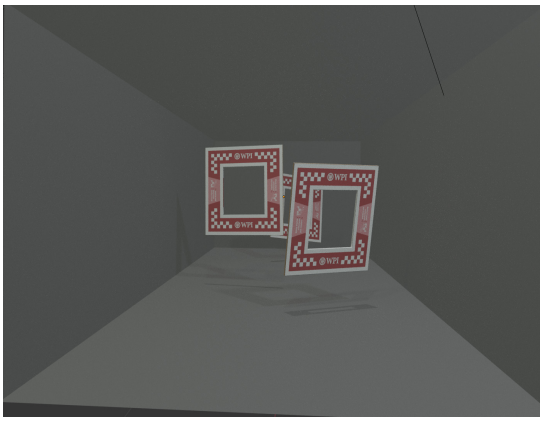

Fig. 6. front view of generated map

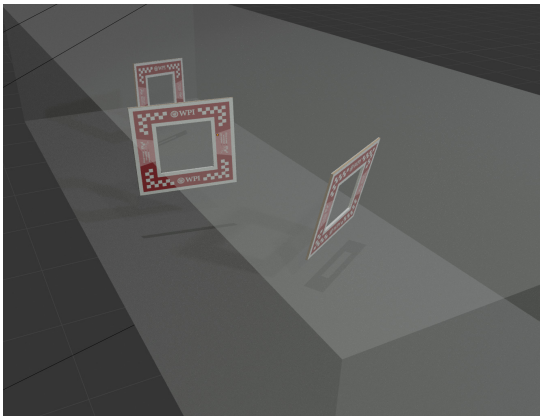

Fig. 7. isometric view of generated map

VII. Videos

A video of the footage of the drone navigating the obstacle course is shared in the folder under the name runVideo.mp4

VIII. Conclusion

In this project, we built the pipeline for navigating through free space of known windows. This is done using the video from DJI Tello's onboard camera which has a resolution of 720x960. The detection and navigation is done in real time.

References

[1] Open CV's PnP: link
[2] Efficientnet: link