Mini Drone Race – The Planning And Control Saga!

Dushyant Patil
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, United States of America
dpatil1@wpi.edu

Keshubh Sharma
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, United States of America
kssharma@wpi.edu

***Abstract*—This project presents an approach to build a planning and control stack for DJI Tello drone for flying through objects of known shape and size. We successfully used the deep learning model trained in previous assignment for semantic segmentation to detect a window, calculate localized pose of drone and traverse through the window gap for multiple windows in the given environment.**

I. Problem Statement

The aim of this project is to estimate pose of a window of known shape and size. For this we will be using the semantic segmentation followed by a set of post-processing algorithm to get an estimate of window pose in drone frame. This estimated pose is used to align the drone with the window center and the estimated depth is used to successfully travel through the gap.

II. World map

The data provided to us was the text format which consists of information about boundaries of 'world' and approximate pose of windows.

1) **boundary**: This parameter is defined as $\begin{bmatrix} x_{min} & y_{min} & z_{min} & x_{max} & y_{max} & z_{max} \end{bmatrix}$ where, $x_{min}, y_{min}, z_{min}$ defines the lower left point & $x_{max}, y_{max}, z_{max}$ defines the upper right point of the rectangular environment.
2) **window**: This parameter is defined as:
$$\begin{bmatrix} x & y & z & x_{delta} & y_{delta} & z_{delta} & \\ qw & qx & qy & qz & X_{delta} & Y_{delta} & Z_{delta} \end{bmatrix}$$
where, x y z represents the approximate center of the window in meters. qw qx qy qz represents the approximate orientation of the window as a quaternion. While the remaining parameters are variances in the pose in a Gaussian distribution.

We load this data in blender as cube objects using `primitive_cube_add` function of bpy module. The Washburn laboratory has exactly same setup which replicated the map file given.

III. Planning and Control stack

A. Initial Positioning

Since we're provided with the rough locations of all windows in the world frame we decided to fly our drone to roughly 140cm high which is the center height of each window and roughly 180cm in front of each window which works as a rough setup point for fine tuning by the use of perception stack as explained ahead.

B. Semantic Segmentation using DL

For this project, we choose to use Unet.An encoder and a decoder make up the network. By using convolutional layers to extract information from the input image, the encoder reduces the spatial dimensions. Pixel-wise segmentation masks are then produced by the decoder once these features have been upscaled and improved. The use of skip connections, which link corresponding levels in the encoder and decoder, is unique to UNet. This improves segmentation accuracy by maintaining minute information during the upsampling process.
The latest frame captured by DJI Tello is inferred using the pre-trained model which gives us segmentation mask for individual windows present in the frame. Fig 1 shows an example frame captured by the drone and Fig 2 shows the segmentation mask.

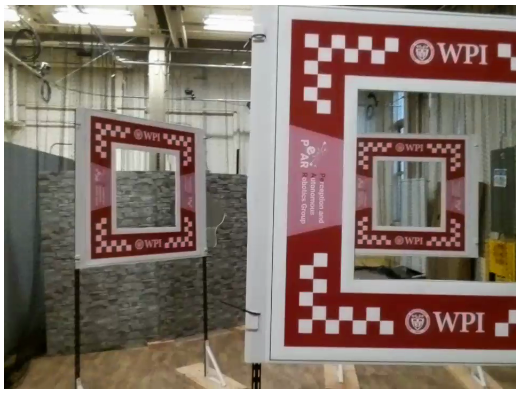
Fig. 1: Captured frame from DJI Tello

C. Closest window estimation

We used largest area criteria to estimate closest window among the predicted masks. We used contour detection with the help of opencv function cv2.findContours. We sorted these contours in the ascending order of area occupied by contours

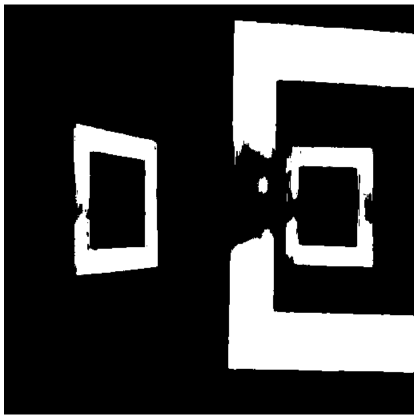

Fig. 2: Inference from trained Unet model

in pixels. We use the last contour as our closest window. As these contours are not perfect rectangles or squares, we apply further post processing to estimate corners of windows. We tried two different approaches which gave us certain accuracy at certain estimation speed. Below image show the inference and largest area detection using this method:

Fig. 3: RGB Images with applied homography and different backgrounds

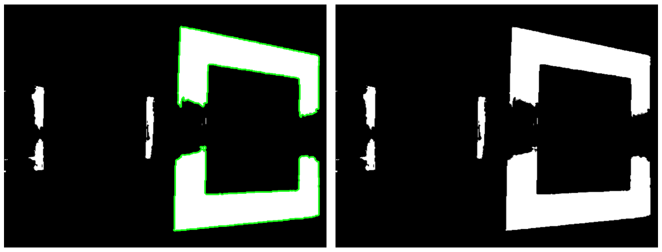

Fig. 4: Inference with closest window estimation

D. Delaunay Triangulation Based Estimation

To approximate the exact corners of the closest window, we apply erosion-dilation on the predicte mask followed by Canny edge detector. On the Canny edges, we apply probabilistic Hough transforms to get set of lines. We find intersection of all lines making an angle of 45 degrees or more with each other. Through all the intersection points, we apply Delaunay triangulation. On these Delaunay triangles, we find edges which are part of only one triangle. These edges represent outer edges of the estimated window shape. Using the points on these outer edges, we approximately fit a quadrilateral and the corners of these corners give us the corners of window which are pretty accurate. But this method takes a lot of time (around 1 s per frame). Thus we tried an alternative which would give similar accuracy in less time. Below images 5 show this algorithm which shows all the intermediate result of image processing:

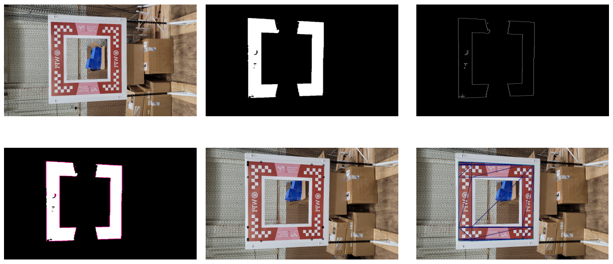

Fig. 5: Delaunay Triangulation pipeline

E. Rectangle Fitting

On the closest estimated window, we apply dilation-erosion. On this image, we approximate a convex hull. On this convex hull, we find the maximum and minimum x and y coordinates. Using these coordinates, we fir a rectangle covering enclosing the predicted mask. On the edges of the rectangle we try to find the points which are closest to the vertices but lie withing the white region of our predicted mask. This gives a good estimation of corners of the window. This method gives us real time corner detection with good accuracy. Below images **??** show the rectangle fitting pipeline to get bounding box of closest window:

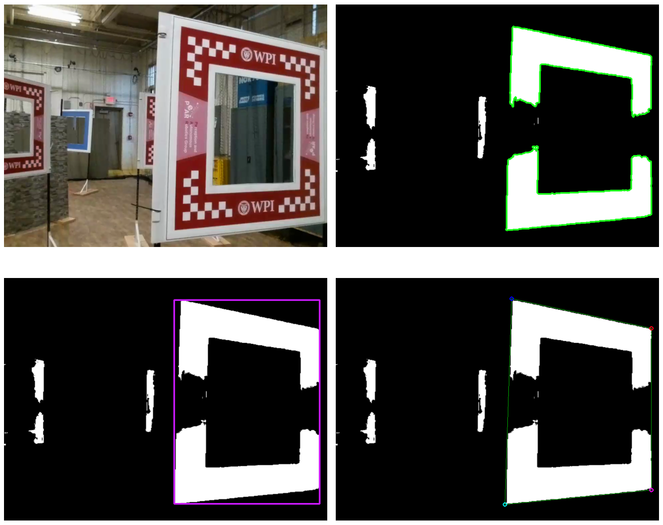

Fig. 6: Rectangle Fitting Corner Estimation

F. Pose Estimation

We use Perspective N-Points method to estimate the pose of the window. We use cv2.solvepnp function to estimate the

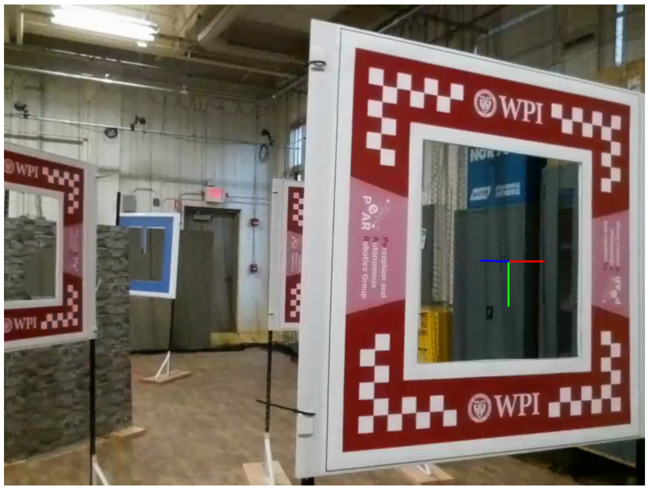

Fig. 7: "Reprojecting the Window center in drone camera frame

pose of closest window in camera frame. We reproject the coordinate frame on the image as shown in image 7 below:

G. *Control Stack*

Now that our drone is roughly in front of the window and have a calculated pose with respect to it we then proceed traverse through it.

1) We first align the position of drone with the calculated window center from PnP by moving the drone in Y-axis & Z-axis of the drone frame.
2) After the position alignment is complete we adjust the yaw of the drone to compensate for the rotation of window with respect to drone to avoid collision with corners.
3) Finally we use the estimated depth from PnP and move the drone in X-axis of the drone frame with an extension of about 40cm to make sure the drone always clears the window regardless of the unreliable in-built odometry.

IV. Conclusion

Through this project we were able to build a perception, planning and control stack for DJI Tello which uses a combination of deep learning and traditional computer vision to traverse through a series of window openings in a known setting. The perception stack is light enough to run on the Jetson Orin Nano and still capable of traversing the complicated path.

V. Challanges Faced

1) The DJI Tello drifts with uncertainity during takeoff when using the $takeoff$ command from the SDK. We were able to mitigate a lot of that issue by cheking for damage on the blade guards and clibrating the on-board IMU using the android app.
2) The perception stack is weird in the sense that it produces random errors in segmentation mask which needs to be processed and that creates issues. We believe a more trained model for segmentation masks will be able to solve this problem but this can't be verified at the moment.
3) The DJI Tello also cannot handle multiple UDP calls simultaneously and gives false positive response to commands in an unrealiable manner. To solve this issue we need to give long delays before every motion action but it reduces the overall speed of the task.
4) Also since we were using the roughly known position of the window and already flying close to the center we were not able to perform really small adjustments in position and rotation due to the SDK's limitations.

References

[1] A Quaternion-based Unscented Kalman Filter for Orientation Tracking
[2] Class Notes by Prof. Nitin Sanket
[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," 2015. Available: https://arxiv.org/pdf/1505.04597.pdf
[4] "Unet from Scratch —— Unet Tutorial —— Developers hutt," www.youtube.com.
[5] "Implementing original U-Net from scratch using PyTorch," www.youtube.com.
[6] "PyTorch Image Segmentation Tutorial with U-NET: everything from scratch baby," www.youtube.com.
[7] Tello SDK 2.0 User Guide.