Mini Drone Race - Planning and Control

Ankush Singh Bhardwaj
abhardwaj@wpi.edu

Sri Lakshmi Hasitha Bachimanchi
sbachimanchi@wpi.edu

Anuj Pradeep Pai Raikar
apairaikar@wpi.edu

***Abstract*—The project presents the planning and control stack and its integration with the designed perception stack for the autonomous navigation of a DJI Tello Edu drone through an approximately known environment. The project is inspired from Lockheed Martin's AlphaPilot competition where, the on board localization and perception stack works fast in real time for the drone to navigate through multiple windows in the order of their appearance. Racing across the environment from the start to the goal, while successfully identifying and passing through the windows collision free is of utmost importance. To this end, we planned paths for the DJI Tello drone using PID control and navigation using knowledge of the map environment given prior.**

I. Environment

The test track consists of 3 windows, which shall now be referred to, more appropriately, as, "Gates". They are placed at different locations and orientations, 3D pose of all of which are known approximately in the track map co-ordinate frame. The window is given to be a rectangular board with peculiar features on it including the WPI and PeAR group logos and the checkerboard pattern on the corners. Additionally, a sample format of the window is given in a map format with appropriate center location and orientation and its possible variation from the center location and orientation. The window is given as shown in Fig. 1. This map is used to store the window coordinates for navigation of drone through the window without colliding with the external window boundaries.

II. Implementation

Our approach is to navigate from waypoint to waypoint as safely and as fast as possible. The waypoints are the center locations of the gates in terms of height and we aim to stop at a distance before the gates, so that our drone can take an image successfully. A successful capture would contain the window. for detection of the gates and fly through them using our perception, navigation and control stack. The following subsections will provide more detail.

A. *Gate Detection*

We used the U-Net model from training our dataset on WPI's Turing GPU clusters to obtain the binary masks of the gates in real time. We transferred this model checkpoint on to NVIDIA Jetson Orin Nano and ran the inference during each time the drone took a shot before the window.

Fig. 1. Test Track with 3 Windows

B. *Camera Calibration*

The camera calibration matrix with focal lengths, principal point and distortion parameters. The camera was calibrated with the images from the drone app. For our race, we are using the video stream. The resolution of the photos are different from those taken on the app. We once again calibrated the camera by using the Calib.io calibration board from PeAR group's laboratory. Again, we utilized Matlab's Calibration toolbox. With the toolbox, the corner points of the checkboard are estimated on a set of checkerboard images captured with DJI Tello with 3D coordinates of the calibration corner pattern in the world frame. With the help of the toolbox, the estimated parameters are used to back project the world points onto the images and are compared with the observed image points for validation. As expected the intrinsic parameters of the camera changed with this change, and were incorporated in our pose estimation code. The intrinsic and extrinsic parameters of the DJI Tello Edu after calibration are as follows.

C. *3D Pose Estimation*

The 3D pose of the window was calculated using the *cv2.solvepnp* function. The camera Calibration had provided us with the K matrix for the camera, The world coordinate frame is assumed to be at the bottom left part of the window.Since the width and height of the window are known, and the thickness is negligible the world coordinates of the

Parameter	Value
Focal Length	$\begin{bmatrix}1.8229 \times 10^3 & 1.8210 \times 10^3\end{bmatrix}$
Principal Point	$\begin{bmatrix}1.2936 \times 10^3 & 968.5153\end{bmatrix}$
Image Size	$\begin{bmatrix}1936 & 2592\end{bmatrix}$
Radial Distortion	$\begin{bmatrix}0.0290 & 0.0641\end{bmatrix}$
Tangential Distortion	$\begin{bmatrix}0 & 0\end{bmatrix}$
Skew	0
Matrix K	$\begin{bmatrix}1.8229 \times 10^3 & 0 & 1.2936 \times 10^3 \\ 0 & 1.8210 \times 10^3 & 968.5153 \\ 0 & 0 & 1\end{bmatrix}$

window corner point can be obtained. The pixel coordinates of the corners are known by applying classical cv approaches as described above. PnP requires 3 points to solve and minimum of 4 to get a unique solution.

$$\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}f_x & 0 & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix} \begin{bmatrix}r_{11} & r_{12} & r_{13} & t_1\\r_{21} & r_{22} & r_{23} & t_2\\r_{31} & r_{32} & r_{33} & t_3\end{bmatrix} \begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

The above formula solves for the pose of the window with respect to the camera frame. In the formula (u, v) indicate the pixel coordinates, (f_x, f_y) and (c_x, c_y) are the focal length and principal point respectively, $(\mathbf{R}_{3\times3}|\mathbf{t}_{1\times3})$ represent the translation and rotation of the camera (which we are calculating from solvePnP), and (X, Y, Z) are the world coordinates of the window frame.

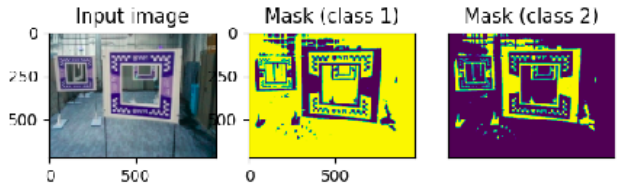

Fig. 2. Inference1

This gives us the pose of the window with respect to the drone.

D. Navigation and Control Stack

Initially for the planner, RRT Star algorithm was used to find the path between the points. Since, the test track has no other obstacles other than the windows, it was computationally economical to find intermediate points between the start location $(0, 0, 0)$ of DJI Tello and the approximate window locations. From the given map with window locations, the approximate center coordinates (X, Y, Z) are stored as goal coordinates and as the DJI Tello navigates to the goal location, it's current location was updated. Additionally, intermediate points are considered when the distance between the points is greater than 100 in X or Y or Z direction.

For the DJI Tello to move towards the window locations, it is instructed to move towards the window with slight adjustments with offset of 200 in the direction of y and 15 in x direction to capture the images of the window. With the built perception stack, the 3D pose of the window is calculated with respect to the drone. From the results, the centroid was calculated using the average of the obtained corner points. The calculated points were in camera frame and are translated from the camera frame to the drone frame. Since the DJI Tello takes incremental coordinates, the centroid points are directly fed and are given with an offset of +10 in the Y direction for it to pass through the point.

Additionally, a simple code for using odometry in our navigation stack is also written for obtaining the position of the drone in the map coordinates. This helps to navigate to the approximate coordinates of the next window and pass through it.

III. Testing

The navigation stack integrated with the built perception stack was tested on the real environment consisting of three windows as in Fig. 6 for the drone to fly through the windows without any collision. The real-time detections by DJI Tello while navigating through the windows are shown in the figures.

Fig. 3. Drone before passing through gate

Fig. 4. Drone after passing through gate

IV. References

1 Principles of Robot Motion: Theory, Algorithms, and Implementations" by Howie Choset, Kevin M. Lynch, et al.
2 https://docs.px4.io/main/en/flight_stack/controller_diagrams.html
3 https://github.com/milesial/Pytorch-UNet
4 https://github.com/damiafuentes/DJITelloPy/tree/master/djitellopy
5 https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello SDK 2.0 User Guide.pdf
6 https://www.deeplearningbook.org/