Mini Drone Race - Perception Saga

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 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. Neural Network based approaches work best to train the model to identify and segment the windows. To this end, custom datasets in a Blender simulated environment were curated and used to train our network to better predict the windows under various conditions of lighting and occlusions. The goal for this module is to detect robust window/corner segmentations in real life. The ultimate goal of the project is to navigate drone through these windows without any stray contact with any surfaces or boundaries in real life.**

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, 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

We decided our approach to be the application of classical computer vision methods over the segmentation masks of the gates obtained using U-Net-based Semantic Segmentation to approximate the pose and the centroid of the gate in 3D coordinates. The following subsections will provide more detail.

A. Semantic Segmentation

*1) **Neural Network**:* We decided to use a PyTorch based U-Net with custom dataloaders to obtain the binary masks of the gates. The feature chosen was the unique checkerboard pattern at the corners of the window. Since our task is highly specialized, we decided to train the network from scratch over the WPI Turing clusters in various hyperparameter configurations. Robust mask detection for the gates nulifies the need to filter multiple detections.

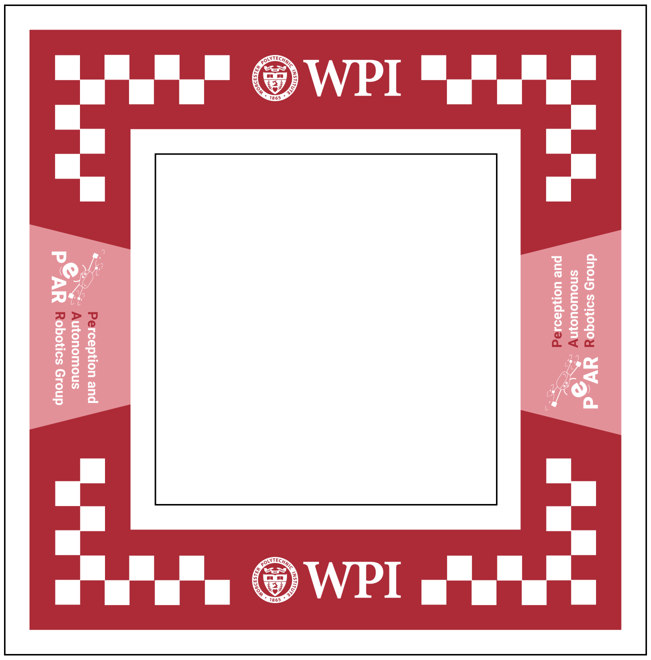
WPI
WPI
Perception and Autonomous Robotics Group
PeAR
Perception and Autonomous Robotics Group
PeAR

Fig. 1. Window

*2) **Images Dataset Generation**:* Data for training the network was generated in blender. The windows are spawned in the 3D blender environment. We considered a multitude of scenarios of varying camera angles, lighting conditions, occclusions and number of windows. Augmentations were performed by varying the position of the camera and we employed alpha matting to add as the backdrop; photos of the test environment in the laboratory to make the detections robust to any kind of background.

In our first try the dataset was created without the usage of any background. After running the inference over the image captured from the drone from the real environment, along with performing segmentation of the window in view, the model segmented background environment features such as the safety net and the floor tiles. Hence more datasets were created by composting the initially captured images with different background images with textures involving rectangles, flying net and some random backgrounds.

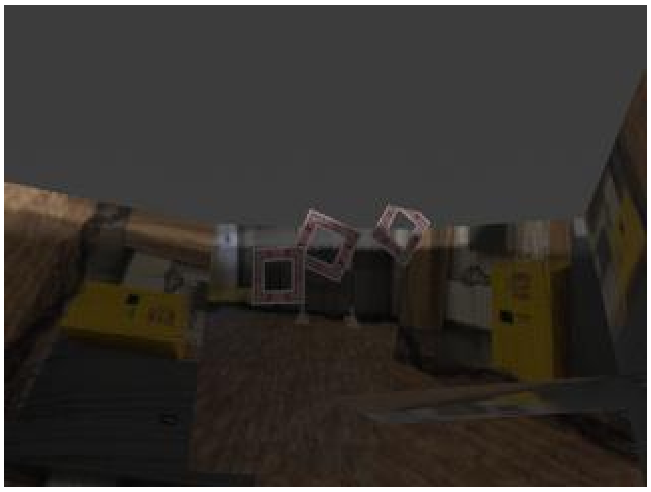

Fig. 2. Sample Image from the Dataset

However, when blending the custom background images with the first set of the plain blender generated window images, visibility of the window images was compromised. This gave poor inference over blended images. Hence, backgrounds were added as 3D objects to the existing scene of three windows and large datasets were created using various camera poses and lighting. The total size of the dataset is 6K images.

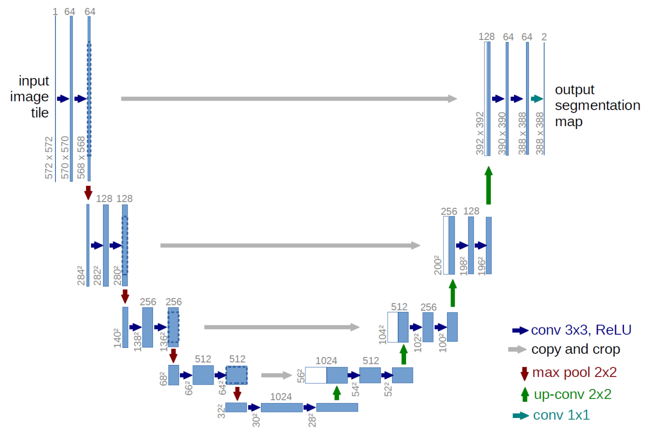

Fig. 3. U-Net Structure

*3) **Masks Dataset Creation**:* While capturing images of the scene from blender, a JSON file is created which stores the 2D pixel coordinates of the corners of the window by mapping from the 3D coordinates. The pixel coordinates from the JSON file are used for creating masks of the windows to be used for training the model. Based on the vertex coordinates, the mask is generated by filling the outer and inner polygons with white and black colors respectively. The generated binary masks are used as a ground truth data for training and evaluating the model. The masks provide the model, a labelled information about the window object to be segmented from the image. The raw images generated from Dataset Generation and these masks are used to better train the model to predict accurate segmentation.

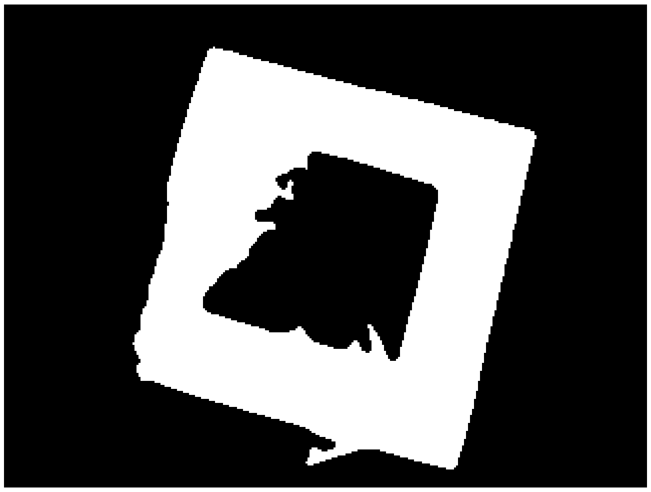

Fig. 4. Mask obtained on a random image after less training

B. *Camera Calibration*

The camera calibration matrix with focal lengths, principal point and distortion parameters. The camera of the DJI Tello Edu was calibrated by printing a checkerboard and utilizing 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. The intrinsic and extrinsic parameters of the DJI Tello Edu after calibration are as follows.

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	$\begin{bmatrix}1.8229 \times 10^3 & 0 & 1.2936 \times 10^3 \\ 0 & 1.8210 \times 10^3 & 968.5153 \\ 0 & 0 & 1\end{bmatrix}$

The validation showing the projected points and observed points is shown in figure Fig 5.

C. *Corner Detection*

After obtaining the masks of the windows on the track, classical CV approaches were used to determine the corners of the window. Since the neural network is trained to detect and segment windows, the *cv2.findcontour* is performed. The

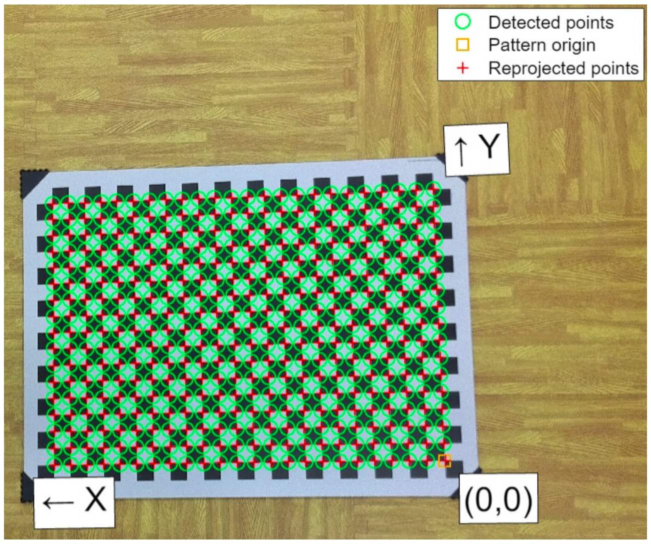

Fig. 5. Camera Calibration

largest counter is supposed to be that of the largest window. The contours may not be perfectly segmented and not provide proper window corners, to solve this we added bounding box to our largest contour. This provided us with better results. The pixel values of the four corners of our box provided us with the approximate four pixel coordinates in the picture.

D. 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 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.

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

III. Testing

The designed perception stack was tested on the real environment consisting of three windows as in Fig. 6. The individual images of window were captured with the camera of DJI Tello Edu and inference was run on these images. Images were captured varying the pose, lighting, color, occlusion and backgrounds. The real-time detections are as shown in figures.

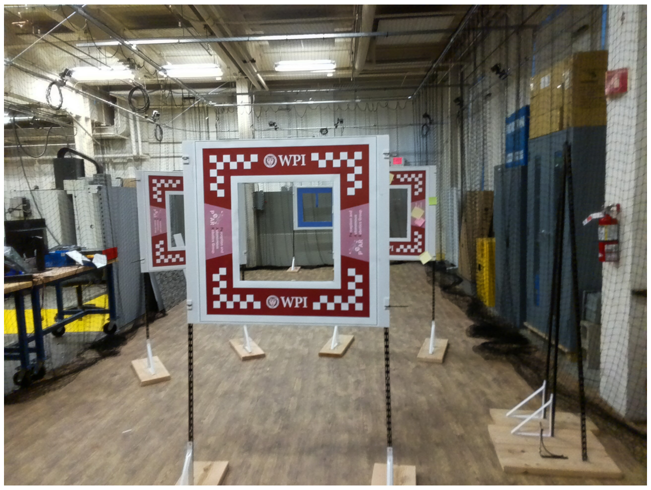

Fig. 6. Test Track with 3 Windows

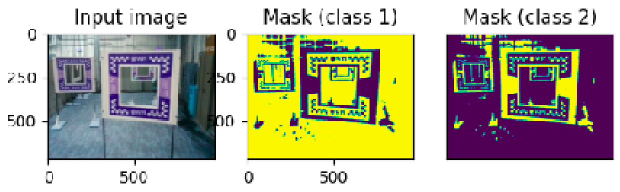

Fig. 7. Inference1

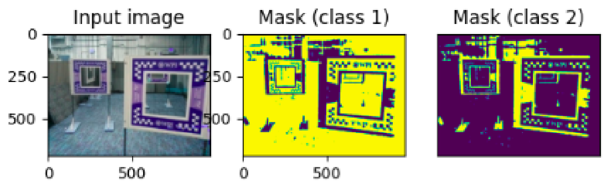

Fig. 8. Inference2

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

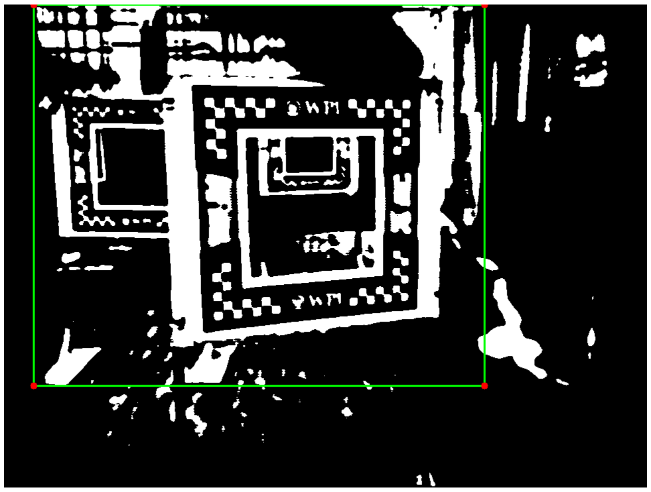

Fig. 9. Inference2

5 https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello SDK 2.0 User Guide.pdf

6 https://www.deeplearningbook.org/