ENAE788M Assignment 3b - Landing on Bullseye Marker

Estefany Carrillo, Mohamed Khalid M, and Sharan Nayak

I. INTRODUCTION

This project provides the ability to PRG Husky to land on the Bullseye Marker. We use image processing algorithms to detect the Bullseye marker and determine the location of the quadrotor with respect to the Bullseye marker. Finally, we use a proportional controller to move towards the marker.

II. TAG DETECTION

We detect circles in the tag by first performing color thresholding to extract white parts from the images. For the thresholding, we trained the Gaussian Mixture Model on a set of images taken under varying lighting conditions and camera poses. The training set is obtained by taking images of the tag and cropping regions of interest corresponding to white segments of the image. An image and the output of the model for this image are shown in Figs. 1 and 2, respectively.

Fig. 1: Image obtained with the stereo camera.

Fig. 2: GMM output from the image obtained with the stereo camera.

III. ELLIPSE FITTING

Once we obtain the segmented image, we apply the *HoughCircles* function from OpenCV with the option of HOUGH GRADIENT and setting the threshold for the Canny edge detection to 50 and the accumulator threshold for the circle centers to a value dependent on how far the quadrotor is from the centroid detected in the tag. If this distance is greater than 10, we set the accumulator threshold to 37, otherwise it is set to 60. These threshold values are set empirically based on the observation that when the quadrotor gets closer to the tag, circles can be more easily detected. Thus we can increase the threshold as the quadrotor gets closer to the tag. Since the stereo camera provides us with two images, we choose the one from the left camera as the detection of circles provides more consistent results. Fig. 3 shows the output of the ellipse fitting applied to the image shown in Fig. 1.

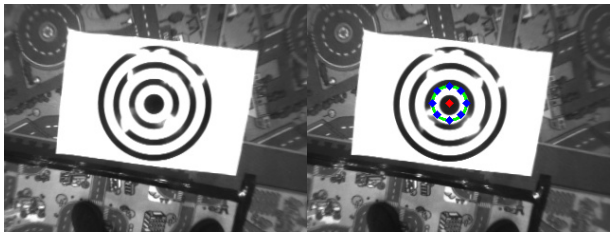

Fig. 3: Ellipse fitting output from the image obtained with Hough Circles. Note that the ellipse is detected on the image on the right.

IV. TAG POSE

To estimate the camera pose, we use the Odometry information that can be obtained from the Bebop. By subscribing to the Odometry topic, we can obtain position and orientation of the quadrotor in quaternion form. Finally, by applying the function *euler from quaternion*, we then extract yaw, pitch and roll values. Another option was to use PNP to estimate the camera pose, but this method resulted in noisy measurements.

V. RESULTS

Once the centroid in the tag is detected, the quadrotor is driven towards it by applying a proportional controller with saturation. The proportional gains for the x, y and z are set to 0.01. If the velocity inputs computed are above a threshold of 0.2, they are clamped at this value. Finally, a small velocity command is given once the quadrotor comes very close to the centroid of the tag in order to compensate for the small offset of the camera position relative to the quadrotor's center position.

VI. PLOTTING IN RVIZ

We use Rviz to plot the bullseye marker and real-time positions of the quadrotor. We use Blender program to convert the bullseye image to a dae 3D model file. We then publish the dae file as a mesh resource in a visualization marker message to Rviz. We set alpha-transparent channel to 0.9 to get the bullseye displayed in Rviz workspace. For plotting the position of the quadrotor, the ROS TF transform

containing the position coordinates is broadcasted to our program from Rviz every 0.1 sec. The position coordinates of the quadrotor were generated using PNP.

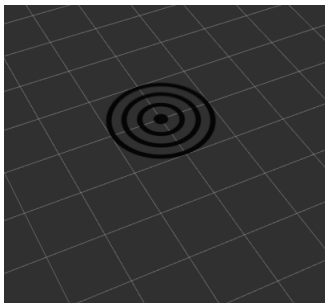

Fig. 4: Bullseye Marker displayed in Rviz