ENAE788M Assignment 4b - Avoid the wall and find the bridge

Estefany Carrillo, Mohamed Khalid M, and Sharan Nayak

I. INTRODUCTION

This project involves using computer vision to perform two tasks. The first involves detecting a wall and avoiding it. The second involves finding a bridge and then flying over it.

II. BRIDGE DETECTION

We use the Gaussian Mixture Model (GMM) segmentation to determine the location of the bridge. Our detection makes the assumption that the bridge and the river can be seen when the quadrotor lifts off from the start position. The GMM is trained to segment the river from the background. We used images of river and bridge taken from different orientations as our training set. An example of segmentation of the river from background is shown in Fig. 1.

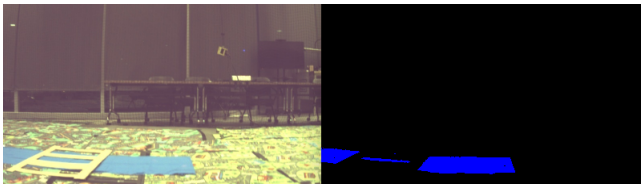

Fig. 1: Original (left) and GMM Segmented (right) image

We use dilation to connect nearby pixels together. We then use connected components and select three largest components in the image (Fig. 2). The bigger two corresponds to the river to the left and right side of the bridge and three connected component corresponds to the portion of the river between the tracks of the bridge.

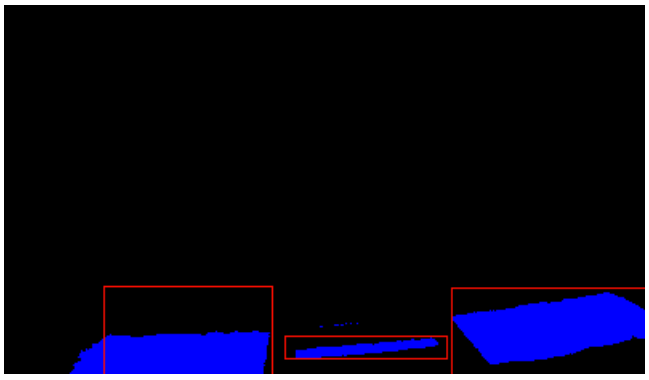

Fig. 2: Three connected components

We then select the third smallest component and calculate the centroid (Fig. 3). It should correspond to the center of the bridge. It is possible that in some cases, the third component is not found. If we get into this scenario, we calculate the centroid of the two bigger components and then calculate a weighted average. The bigger component is given less weightage so that centroid is not biased towards the bigger component.

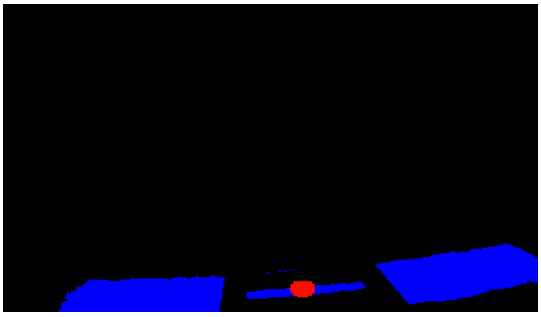

Fig. 3: Centroid (red blob) of bridge

We do alignment in the horizontal direction to get centroid at the central vertical column of the image. When the centroid falls within a threshold of 5 pixels, we give the command to the quadrotor to move forward a set distance and cross the bridge.

III. WALL DETECTION AND AVOIDANCE

To detect the wall in front of the quadrotor, we first apply GMM segmentation on the white color of the board to remove the floor from the images (Fig. 6). We then apply feature matching and compute optical flow to identify objects closer to the quadrotor. In order to compute the sparse optical flow between features detected in consecutive frames, we use the Kanade-Lucas-Tomasi Tracker (KLT). First, corner points are detected using the function from OpenCV, *goodFeaturesToTrack*. This function is an implementation of the Shi-Tomasi corner detection, in which we set the maximum number of corners that should be detected to 1000, the quality level of detection to 0.3, the minimum distance between detected corners to 10, and the block size to 20. Once we have the corner points, we then pass them to the OpenCV function *calcOpticalFlowPyrLK* along with the frames to obtain the positions of the tracked features. We then measured the disparity of the tracked features in consecutive frames using the function *linalg.norm*.

In order to identify objects closer to the quadrotor, we cluster the obtained disparities of the tracked features given that features in the carpet, features in the wall, and features in the objects behind the wall and its surroundings will have different optical flow values. Thus, closer objects like the carpet and wall will result in larger optical flow values for the tracked features. We perform clustering by applying the

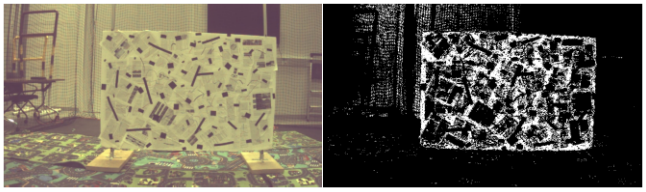

Fig. 4: Original (left) and GMM Segmented (right) image.

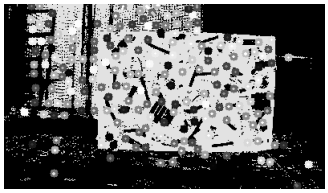

Fig. 5: Features obtained from segmented image.

function *KMeans* from the sklearn.cluster package. We set the number of clusters to 5. Fig. 6 shows the output from our K-means clustering approach.

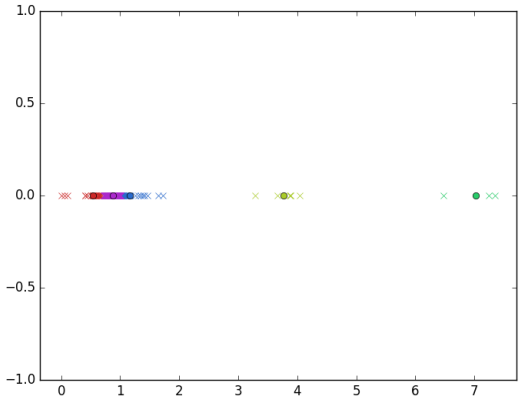

Fig. 6: Result from K-means clustering on the norm of disparity of features tracked between consecutive frames.

In the final step, we can identify the wall by selecting the cluster with the maximum number of data points and taking the average value of the pixels corresponding to this cluster. This average value is set as the centroid of the wall. In order to reduce variation in the wall centroid, we take a weighted average of the previous centroid and the current centroid obtained. Once, the wall centroid is computed, the control strategy is to align the quadrotor to the x-coordinate of the image corresponding to the wall centroid Fig. 7 . Depending on whether the y-coordinate of the centroid is above or below the y-coordinate of the midpoint of the image, the quadrotor is set to go down or up, respectively, by a user-specified distance once it is aligned. Then, the quadrotor is commanded to move forward by a user-specified distance as well. In the final race, we need to replace this user-specified distance with a depth estimate, and in order to do that, we need to reduce the noise in optical flow values.

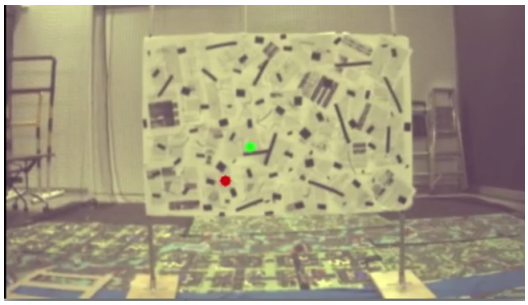

Fig. 7: Red dot represents current estimate of wall centroid and green centroid represents the moving average centroid calculated over time.

IV. RVIZ DISPLAY

We use Rviz to plot the real positions of the quadrotor wrto to the world frame. The wall (Fig. 8) is plotted using the visualization marker array message which is sent to the quadrotor every 0.1s. The actual position coordinates of the quadrotor were obtained from the odometry message of the quadrotor and were broadcasted from our program to Rviz every 0.1s.

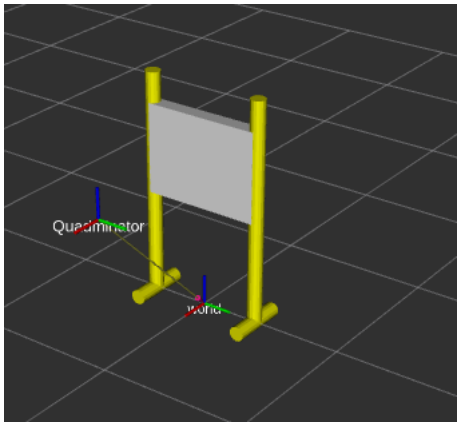

Fig. 8: Wall displayed in Rviz

V. CONCLUSION

In this project, we detected wall and avoided it. We also detected bridge and flew over it. The bridge detection is done using GMM segmentation. The wall detection is done using GMM segmentation and hough lines. We tried using optical flow to get relative depth between wall and background wall but the centroid detection was very noisy. Our future work will include removing noise from optical flow estimates.