Wall Avoidance and Bridge Detection - USING 1 LATE DAY

Team `sudo rm -rf *`

Abhishek Shastry
Department of Aerospace Engineering
University of Maryland
College Park 20742
Email: shastry@umd.edu

Animesh Shastry
Department of Aerospace Engineering
University of Maryland
College Park 20742
Email: animeshs@umd.edu

Nicholas Rehm
Department of Aerospace Engineering
University of Maryland
College Park 20742
Email: nrehm@umd.edu

***Abstract*—In this report, an algorithm for wall avoidance and bridge detection/navigation with the PRG Husky is presented for ENAE788M: Hands on Autonomous Aerial Robotics. For wall avoidance, position information from the bebop's odometry is used in conjunction with temporally matched features from the forward facing video feed. This enables depth of matched features to be estimated and the features to be clustered to determine the position and height of the wall. For bridge detection, the downward facing gray scale video feed is used to directly detect the bridge by generating a feature mask to isolate featureless areas, and then thresholding for the brightness of the bridge. Position and orientation can then be found which is used to generate a path across the bridge in the correct direction. Results show that precautions must be taken to ensure good feature detection for the wall avoidance algorithm to be successful and that the method of bridge detection is very robust.**

I. Introduction

Quadrotors navigating in an unknown environment require a method of identifying obstacles in front of them to avoid collision. They may also need to use cues on the ground in order to generate a flight path, but may be limited in their downward facing sensor package. Wall detection can be achieved through a combination of onboard odometry and a forward facing camera. Features can be temporally matched from the forward facing camera image to determine disparity due to the known (estimated from odometry) change in quadrotor pose. Features detected in the foreground can be grouped to estimate the distance to and size of a planar surface such as a wall. In some cases, downward facing sensors may be limited to a grayscale camera which cannot detect colored features such as a river and bridge. In this case, alternative technique must be employed. Texture can easily be extracted from a grayscale image and thus the image can be segmented into featured and featureless areas. This method can be exploited and used in conjunction with brightness thresholding to extract particular features, such as a relatively brightly colored bridge with little texture.

Link to the result videos: Click Here

II. High Resolution Bridge Detection and Navigation

Bridge detection is achieved through the generation of an isolated bridge mask. Position and orientation of the bridge relative to the body frame of the quadrotor is estimated using the known geometry of the bridge and pre-defined mission altitude. Our method is extremely robust in rejecting any surface/texture that is not the bridge.

A. Bridge Mask

A binary mask of the bridge is generated from the grayscale downward facing duo camera using the following method. Figure 1 shows an example image (extreme case) in which our method of image processing is implemented.

Fig. 1. Example of grayscale image from duo camera.

1) Featureless areas: Featureless areas of the image are found by aggressive canny edge detection (Figure 2). The edge mask is dilated to fill the featured areas and inverted to produce a mask of the featureless areas.

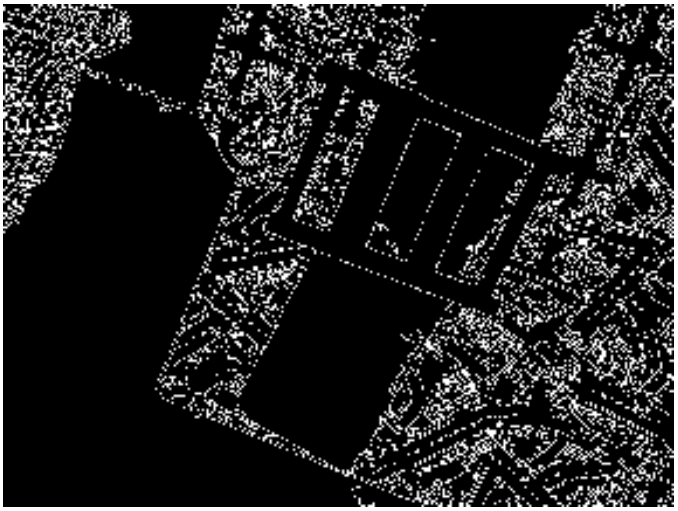

Fig. 2. Aggressive canny edge detection on test image tuned such that "featureless" areas have few detected edges.

2) Bridge from featureless areas: Of the featureless areas (Figure 3), the bridge can be isolated by means of a simple brightness threshold (Figure 4).

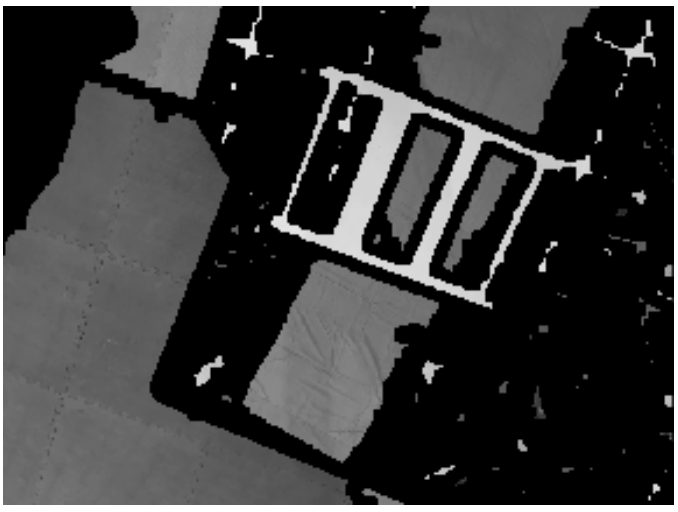

Fig. 3. Featureless areas of the original image. Note brightness of bridge compared to the river and black foam mats.

Fig. 4. Mask of bridge after brightness thresholding of featureless mask.

B. Bridge Position and Orientation Estimate

Using the bridge mask in Figure 4, the bridge position and orientation is easily found with the following method.

1) Polygon fitting: A polygon is fit to the largest contour in the bridge mask image and the four corners are found.

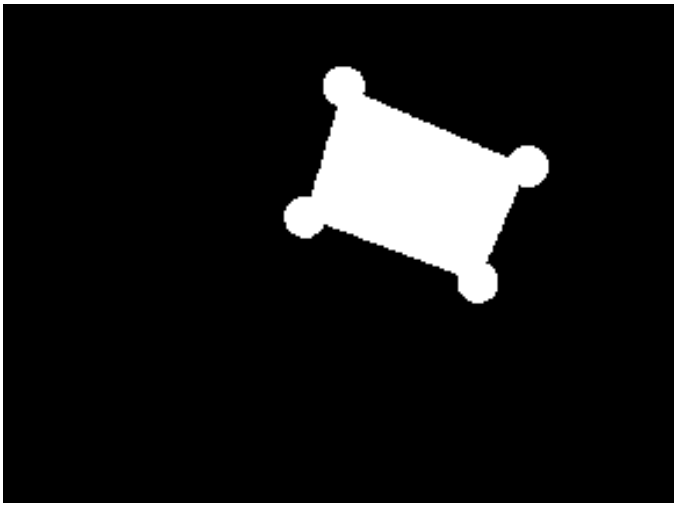

Fig. 5. Polygon fitted to largest contour of bridge mask with four detected corners.

2) Center detection: The center of the bridge is found by taking the average of the four corner coordinates that were found in the previous step. These are then converted from normalized pixel coordinates to units in the quadrotor body frame using the focal length of the duo camera and prescribed mission altitude as the known depth.

3) Orientation detection: The lengths of each of the fitted polygon sides are computed. The slope of the two longer sides are averaged to find the orientation across the bridge. Depending on the location of the bridge in the frame and the angle of the river (assumed normal to the angle of the bridge), the proper orientation across the bridge from the current position is computed. If only half of the bridge is

detected on the edge of the frame, the pixels of the bridge mask on the edge of the image are used to compute orientation as the previous method would incorrectly compute the orientation normal to the true orientation due to the proportions of the cut-off detected bridge. This method of orientation detection can accurately compute the correct path across the bridge regardless of it's relative angle or position in the image frame as shown with the difficult example in Figure 6.

Fig. 6. Final detected bridge center and correct orientation to traverse across it.

C. Mission Planner

The position and orientation across the bridge in the body frame of the quadrotor is filtered with a simple low pass filter to reduce noise and the effect of outliers in the estimates. The quadrotor follows a search path from left to right along the river until the bridge is detected. Using the filtered position and orientation estimate, a waypoint .5 meters before the bridge is computed in the inertial frame and published to the outer loop controller. After converging on this waypoint, a second waypoint is published 1 meter across the bridge. The quadrotor converges to this position and the mission is complete.

III. Low Resolution River and Bridge Detection

Based on features the image can be segmented into two parts - relatively highly featured areas and areas with relatively very low features. To do this the image is divided into blocks/patches of a certain pixel size n (which depends on the camera's height) and a score is calculated for each block/patch. The score is supposed to represent the "amount" of intensity variation in the image patch. The score for each of the patches is then scaled to create a very low resolution image containing the relative pixel intensity variation. A simple thresholding on this image will produce a binary image with areas that have relatively low intensity variation, denoting the river. Note, that the bridge has edges and hence, it is not classified as river. Now, the bigger blobs represent the river and their centers can be calculated by contour moment calculation. The smaller blobs represent the bridged area, and the mean of their centers will give the bridge's center. The orientation is found by calculating a vector that is perpendicular to the slope of the detected river, defined by the line joining their centers. The advantage of using this algorithm is that it is invariant to lighting conditions, but the disadvantage is that the bridge's location estimate is less accurate due to the creation of image patches, resulting in lower resolution.

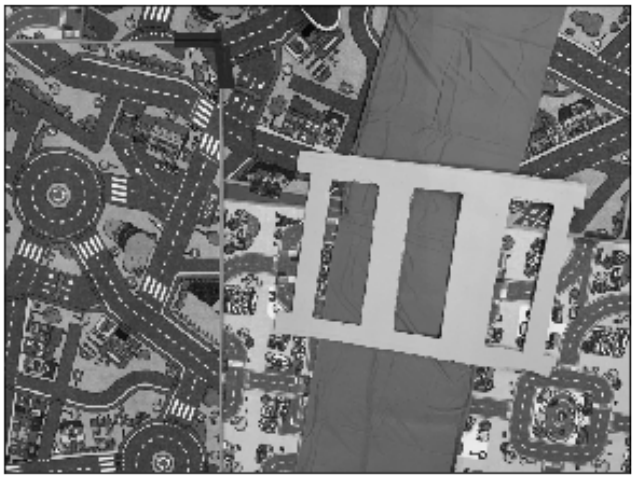

Fig. 7. Original Grayscale Image

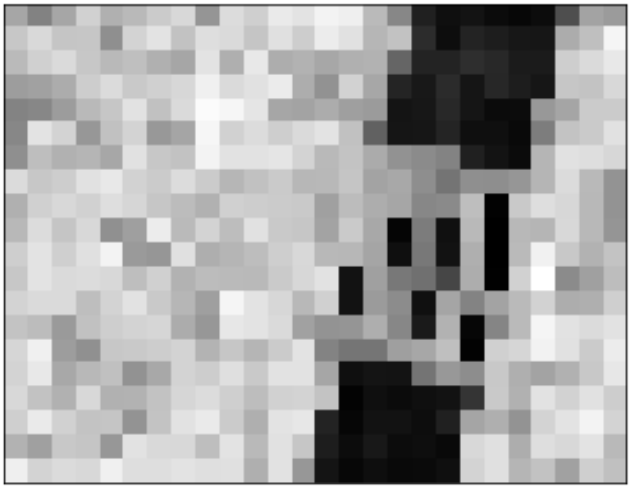

Fig. 8. Batch wise scoring with score $= k\,(max(I) - min(I))$

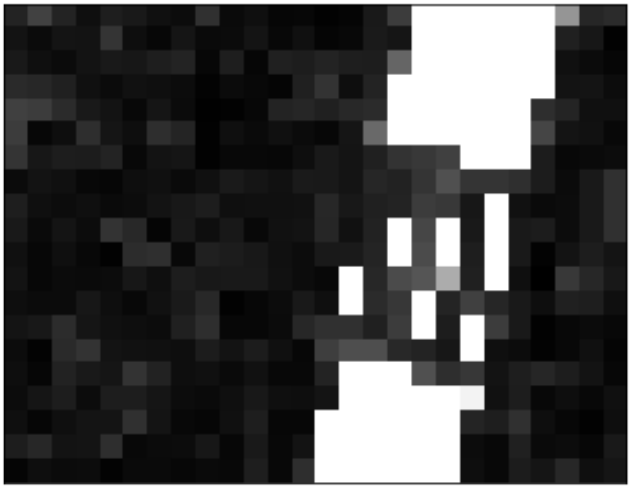

Fig. 9. Batch wise scoring with score $= k/\left(max(I) - min(I)\right)$

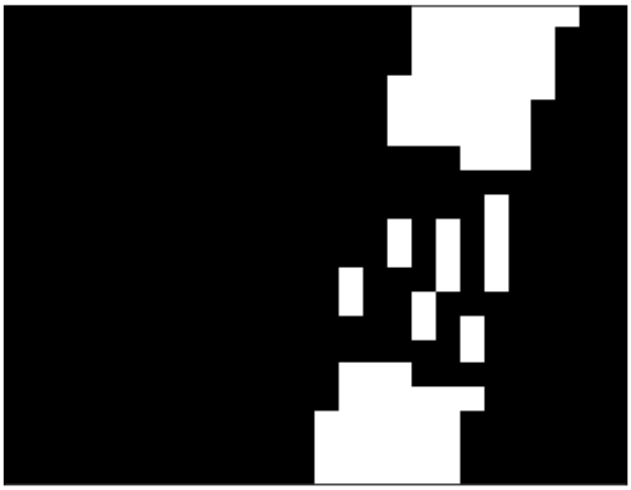

Fig. 10. Binary Image showing areas with less features, i.e, river as white.

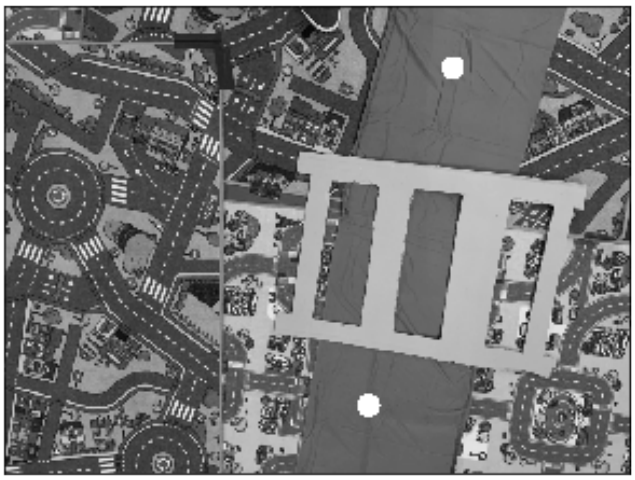

Fig. 11. The center of the rivers overlaid on the original image.

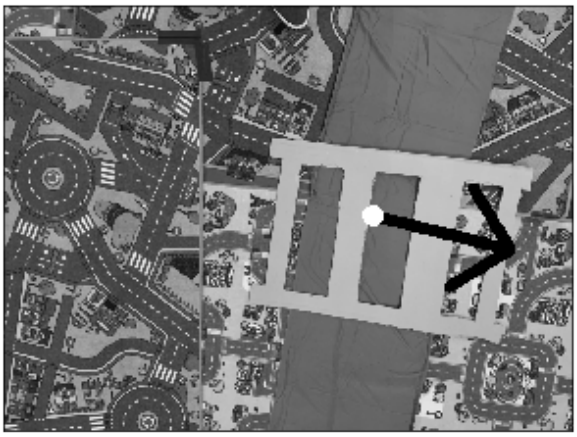

Fig. 12. Estimated bridge's position based on the river's discontinuity and orientation based on the perpendicular vector to the river's direction.

IV. Wall Detection and Avoidance

A. Feature Detection and Matching

From the front monocular camera, two consecutive images separated by a non-zero time and spatial distance are obtained on which features are detected and matched to get pixel displacements in normalized image coordinates. The feature matching is achieved by generation of ORB features in the two images and their brute-force matching. This implemented in OpenCV by creating the objects `orb = cv2.ORB_create()` and `bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck = True)` along with using the functions `orb.detectAndCompute` and `bf.match`.

B. Depth Estimate

Two methods have been implemented to estimate the depth and subsequently triangulate the matched features.

1) The velocity information from the quarotor's odometry can be used as a real-world measure of the spatial motion of the two consecutive monocular frames. The depth of each i^{th} feature is estimated by solving for Z_i in the following optical flow equation.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}_i = \begin{bmatrix} -\frac{1}{Z_i} & 0 & \frac{x_i}{Z_i} & x_i y_i & -1-x_i^2 & y_i \\ 0 & -\frac{1}{Z_i} & \frac{y_i}{Z_i} & 1+y_i^2 & -x_i y_i & -x_i \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \\ \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} \tag{1}$$

To simplify the above equation the quadrotor can be constrained to move linearly and hence the angular velocity terms can be dropped. The optical flow equation now assumes the form given below.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}_i = \begin{bmatrix} -\frac{1}{Z_i} & 0 & \frac{x_i}{Z_i} \\ 0 & -\frac{1}{Z_i} & \frac{y_i}{Z_i} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \tag{2}$$

It is preferred that the quadrotor moves parallel to the wall to reduce the risk of the quadrotor hitting the wall. The z-velocity term can now be dropped to simplify the equation even more.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}_i = \begin{bmatrix}-\frac{1}{Z_i} & 0\\ 0 & -\frac{1}{Z_i}\end{bmatrix}\begin{bmatrix}V_x\\ V_y\end{bmatrix} \tag{3}$$

The Z_i estimate can now be calculated by taking the ratio of the norm of the optical flow vector and the norm of the Quadrotor's velocity vector.

$$Z_i = \sqrt{\frac{\dot{x}^2 + \dot{y}^2}{V_x^2 + V_y^2}} \tag{4}$$

A sample point cloud generated by this method and thresholded by depth and magnitude of optical flow, is given in Figure 13

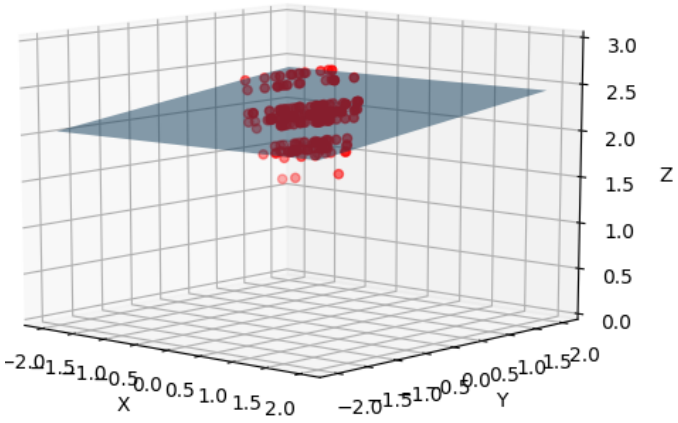

Fig. 13. Wall Point cloud generated by velocity and optical flow matching.

2) Two camera images separated by a known distance, called the baseline, can form a stereo pair. The quadrotor can be made to move in space sinusoidally, and images can be taken from the front monocular camera at the extreme ends of the sinusoidal motion. Features are then extracted and matched to get the pixel displacement on the normalized image coordinates. The depth is computed by taking the ratio of the norm of the pixel displacement to quadrotor's displacement.

$$Z_i = \sqrt{\frac{\Delta x^2 + \Delta y^2}{\Delta X^2 + \Delta Y^2}} \tag{5}$$

By using this method, we get much better estimate of depth as the baseline of the constructed stereo camera pair can be as large as possible. Additionally, the filtering of the bad feature matches and features that are not on the wall is made more robust by the use of this method. A sample sparse depth map image, color coded based on depth is shown in Figures 14.

Fig. 14. Sparse depth map. The red/blue points represents either the feature is too close/far or the feature matching is bad. The green points have acceptable depth values and are considered as a part of the wall.

A sample point cloud generated by this method and thresholded by depth and magnitude of pixel displacement, is given in Figure 15

C. Wall Position Estimate

The wall position is estimated by taking the mean of the extreme feature's normalized pixel coordinates and triangulating

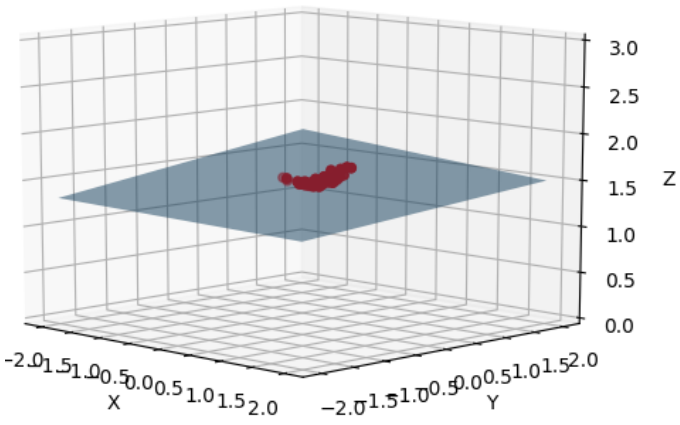

Fig. 15. Wall Point cloud generated by stereo construction with a baseline of 0.4 m.

it by using the mean of all the feature's depth value.

$$Z = \frac{1}{n}\sum_{i=0}^{n} Z_i$$
$$X = \frac{Z}{f_x}\left[\frac{1}{2}\left\{\min(x_i) + \max(x_i)\right\} - c_x\right]$$
$$Y = \frac{Z}{f_y}\left[\frac{1}{2}\left\{\min(y_i) + \max(y_i)\right\} - c_y\right]$$

D. Mission Planner

A typical conditional threshold on the wall's lower edge, $\max(y_i)$ has been used to make the decision of going over or under the wall. In both the cases for generating the desired height of the waypoint, an offset of 0.5 m from the Upper Edge or Lower edge based on the decision is provided. After estimating the wall's location the X and Y coordinates of the waypoints as well as the desired heading are generated by the following transformation.

$$\psi_d = \arctan(Y/X)$$
$$X_d = X \pm 0.5\cos(\psi_d)$$
$$Y_d = Y \pm 0.5\sin(\psi_d)$$

V. Results and Conclusion

The methods for bridge navigation and wall avoidance implemented here were successful in repeatedly carrying out their respective tasks. The bridge detection algorithm provided no false detections (*under the correct lighting conditions) allowing the vehicle to immediately cross the bridge after detection every trial. The only case in which a false detection occurred is when the lighting conditions were not set correctly according to the tuning of the algorithm. Robustness to scene brightness could be improved by including a pre-flight auto-exposure procedure. It was also found that a simple low pass filter is effective in rejecting noisy detections of the bridge, though EKF still provides protection in the case of a lone outlier. The wall detection algorithm was proven to repeatedly correctly identify the wall height allowing the vehicle to navigate over or under it. Our method of depth detection by defining our own baseline through quadrotor position worked well but did not provide a large number of data to refine the position estimate of the wall. The accuracy of this method with such a large baseline negated the need for more data points since it was overall more accurate. Pre-processing the image to give a sharper view of the scene may help with feature matching while the vehicle is moving to provide true temporal feature matching.