Navigation through Unstructured Environments

Ankush Singh Bhardwaj
abhardwaj@wpi.edu

Sri Lakshmi Hasitha Bachimanchi
sbachimanchi@wpi.edu

Anuj Pradeep Pai Raikar
apairaikar@wpi.edu

***Abstract*—Flying through irregular and unknown shaped windows becomes a necessity for search and rescue, exploration or reconnaissance operations. This project presents the perception and navigation stack and its integration with the designed perception stack for the autonomous navigation of a DJI Tello Edu drone through an approximately known environment. Optical flow can be calculated from the oscillation of the drone, to estimate appropriate servoing commands. To this end, we employed the SPyNet deep neural network for robust optical flow estimation for the scene, and classical computer vision techniques for effective detection of the gaps of unknown shape, size and location. Further, the results were used for servoing and navigation of the drone safely through the gap.**

I. Environment and Goal

The test track consists of a textured board that contains multiple irregular shaped gaps of different sizes. The window is given as shown in Fig. 1. The approximate location of the perforated board was provided. The objective is to correctly segment the background from the foreground and navigate through the gap without collision/contact with any of the boundaries.

Fig. 1. Real-life Scenarios

II. Implementation

Our methodology involves capturing pairs of images as the drone moves parallel to a wall containing gaps. These images are then processed using the SPyNet algorithm to infer dense optical flow, capturing the dynamic movement of pixels between consecutive frames. The resulting optical flow magnitudes are color-mapped to provide a visual representation of the motion field. Subsequent analysis includes contour generation, Canny edge detection, and the identification of the largest contour in the scene. The first moment of the pixels of the largest contour, is the centroid of the largest gap and serves as the calculated navigation point for the drone to pass through. Thus, showcasing a comprehensive solution for effective drone navigation in confined spaces. Each of the step is discussed in further sections.

A. *Perception Stack*

*1) **Navigation for Iterative Image Capture**:* We let the drone stabilise at a height after takeoff and then command it to navigate to a position approximately in front of the gaps. This ensures that our drone camera's field of view is majorly occupied by the wall and the irregular shapes. A successful image capture would contain all the gaps. Once the first image is captured by the drone at this position, we give a command to the drone to sharply move along the horizontal direction(positive X-axis in the environment). At the consequent new position, another image is captured by the drone. The drone is returned to the original position after the second image capture. The two above images are used as the input for our Deep Neural Network in the next step.

*2) **Estimating Optical Flow using Deep Learning Models**:* Optical flow, a fundamental concept in computer vision, serves as the cornerstone of our drone navigation approach. It involves the analysis of pixel-level motion patterns between consecutive frames, providing a dynamic representation of the environment's motion field. The optical flow information is crucial for understanding how the drone is moving relative to its surroundings.

It relies on the assumption of brightness constancy, wherein the intensity of a pixel is expected to remain constant between frames. This assumption allows us to track the apparent motion of visual features accurately. Our implementation focuses on dense optical flow, capturing motion vectors for every pixel in the image using SPyNet(Spatial Pyramid Network). SPyNet is one of the state-of-the-art deep learning approaches to estimate optical flow and is trained on a variety of large datasets. We found the "SintelFinal" pre-trained model to perform really well on images captured by Tello. It is a comprehensive approach provides detailed information about the entire scene, enabling the drone to navigate through intricate environments.

Since the network is light, gives us a quick inference, and, is also very robust. Therefore, it follows that our navigation will be faster.

*3) **Background Foreground Segmentation**:* The output is a ".flo" file of vectors that can be visualized as a mapping of magnitudes of flow. We exploit this to perform our background estimation. The minimum optical flow lies within the boundaries of the gaps. Whereas the other areas across the wall are seen to be having a higher magnitude of flow. This fact is used to discern the background and the wall although their appearance is similar. The image is then subjected to conversion to Grayscale image followed by thresholding. This will make it look like a Binary mask. We then use this mask for gap boundary detection. We then apply the findcontours function and apply a little bit of magic explained in section IV to obtain the centroid of our largest gap.

*4) **Safe Waypoint Detection**:* On the optical flow output, we employed contour generation and Canny edge detection. The grayscale optical flow magnitudes serve as input for contour extraction, allowing us to identify regions of interest. Canny edge detection enhances the accuracy of contour identification by highlighting boundaries, while Gaussian blur is applied to mitigate noise and improve the quality of boundary detected. Once we have identified the largest gap in the wall, we can use the cv2.moments to calculate the centroid. This is achieved by analyzing the generated contours, with emphasis on the largest contour as a potential passage for the drone. The center of this contour is then calculated and used as the optimal navigation waypoint into the "UNKNOWN" World.

B. Navigation through Gap

The centroid point, ensures precise and adaptive drone navigation through complex environments. The centroid obtained is then aligned with the image center using move commands on the tello. Considering the gap is large enough for the drone to pass, when the center of the drone and the gap centroid align; the drone is commanded to move forward in the direction of the gap. The Tello is asked to perform a safe landing after moving across the wall to end the trial.

III. Deployment on DJI Tello

After establishing the communication between NVIDIA Jetson Orin Nano with the Tello drone using the DJITelloPy library, the drone is initialized and its takeoff sequence is initiated. The drone is instructed to move to an approximate location near the gap and its movement is controlled by go_xyz_speed function. The drone's video stream is activated, and frames are captured for further analysis. Two images, namely one.png and two.png, are saved during the flight for subsequent processing by SPynet. The resultant output image with dense optical flow map is processed to analyze the flow. Contour analysis is conducted, and the centroid of the largest contour is calculated.

Fig. 2. Laboratory Testing Scenarios

The movement required for the drone is calculated based on the centroid of the largest contour is calculated. Depending on the calculated movement, flags (movexy, movex, movey) are set to indicate the desired direction of drone movement. Based on the calculated movement, the drone is instructed to move.

IV. Tips and Tricks used for better results in Gap detection

Real life deployment is not always direct. The optical flow calculated by your drone is not always accurate, might be the way two simultaneous photos were taken or the lighting conditions. The drone doesn't allow for it to move a distance less than 20 cm, hence to have good inputs for the neural network, we take advantage of the drift experienced by the drone. A command to move in left or right direction is given and then the drone is asked to return back to its original position; since the drone drifts, it never comes back to its original position it is always off by some distance. This helps us take good photos to calculate optical Flow. Fig.4. shows the inputs given to the neural network.

Using only *findContours()* will not yield a good result is calculating the gap as the boundaries many not be differentiated properly. This lead us to use Canny edge detection to find the gap boundaries and mark them with thick marker size so that it appears like a closed contour. On using *findContours()* on this image, it will definitely give us the largest gap to be the be biggest contour on the image. Using a 'gray' colour map while visualizing the flow also helped us compute the gap better.

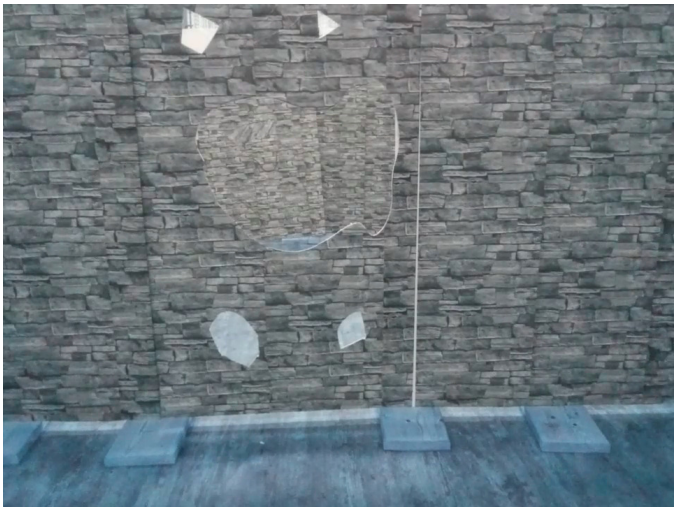

Fig. 3. First Photo Capture during Live Demo for optical flow calculation

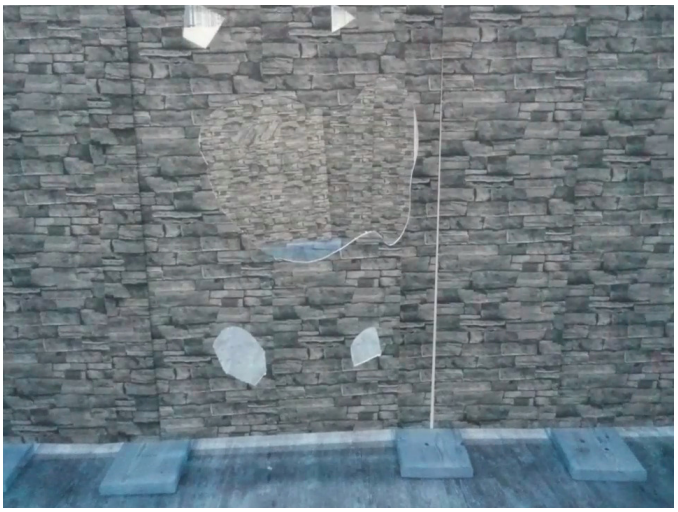

Fig. 4. Second Photo Capture during Live Demo for optical flow calculation

Keeping the height to the minimum after take off will avoid the drone from capturing the area and predicting it to be a possible gap for the drone to move through as seen in Fig.4.

V. Testing

The SpyNet network was first tested on the images obtained from simulation on the Blender. The Fig.8 to Fig.17 show the the gap detected in simulation and the IoU values obtained when compared with the Ground Truth. A script was written to convert the flow obtained to binary so as to compare it with the ground truth which was obtained by setting the pass value to 2 in blender and get the IoU values.

For the real life scenario , the navigation stack integrated with the built perception stack was tested on the real environment consisting of a irregular shaped window as in the figure for the drone to fly through without any collision. The real-time flow image, contour detection and centroid can be seen in the images.

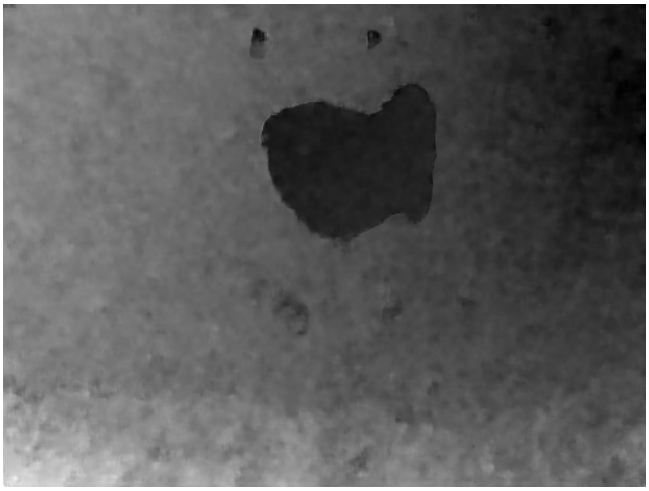

Fig. 5. Real-time Flow Image

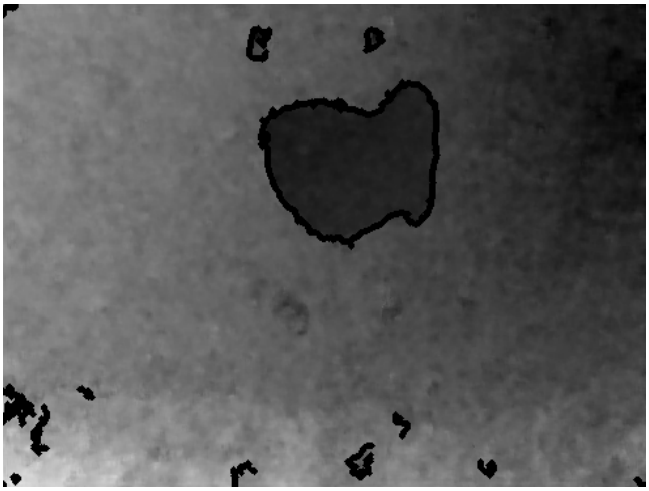

Fig. 6. Real-time Contour Detection

VI. 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/anuragranj/spynet
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/
7 https://learnopencv.com/

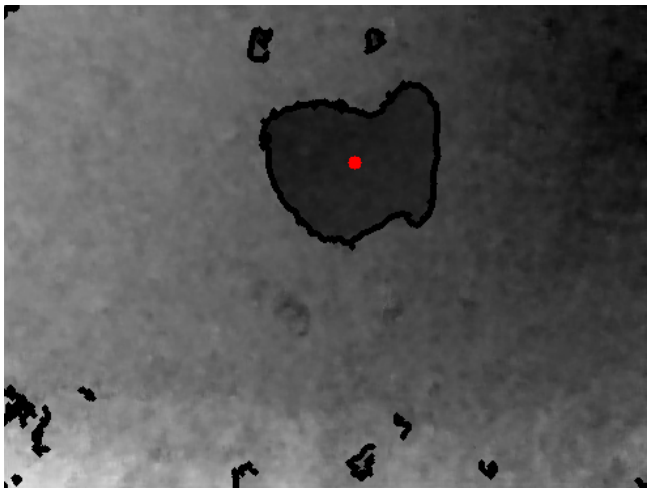

Fig. 7. Real-time Contour Detection with Centroid

Fig. 8. First Photo Capture for Arbitrary Shape 1

Fig. 9. Second Photo Capture for Arbitrary Shape 1

Fig. 10. Simulation Flow Image for Arbitrary Shape 1

Fig. 11. Simulation Contour Detection for Arbitrary Shape 1

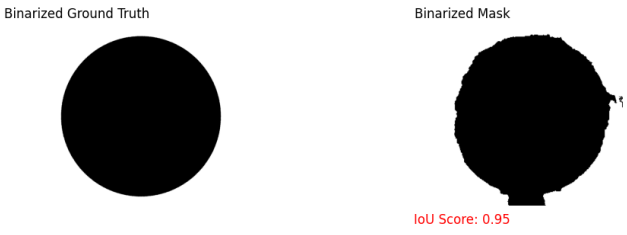

Fig. 12. Ground Truth Comparison Result: IOU 0.95

Fig. 13. First Photo Capture for Arbitrary Shape 2

Fig. 14. Second Photo Capture for Arbitrary Shape 2

Fig. 15. Simulation Flow Image for Arbitrary Shape 2

Fig. 16. Simulation Contour Detection for Arbitrary Shape 2

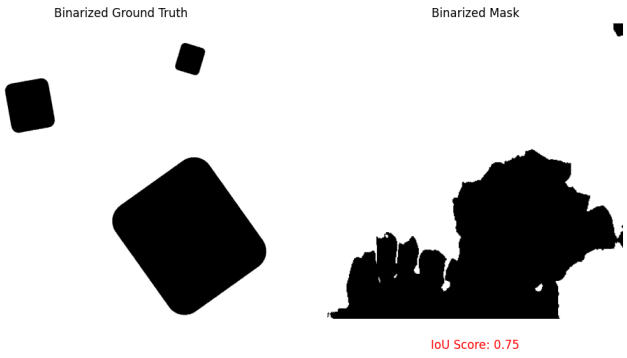

Fig. 17. Ground Truth Comparison Result: IOU 0.75