Navigating The Unknown!
Team Nimbus Navigators
Two Late days used

Chaitanya Sriram Gaddipati
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetss 01609
Email: cgaddipati@wpi.edu

Ankit Talele
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetss 01609
Email: amtalele@wpi.edu

Shiva Surya Lolla
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetss 01609
Email: slolla@wpi.edu

***Abstract*—This study focuses on designing a perception stack for the DJI Tello EDU quadcopter, enabling it to autonomously navigate through irregular, unknown-shaped windows. The primary goal is to identify and fly through the largest gap in a wall. The process begins by maneuvering the quadcopter to a position where the full gap is visible, followed by detecting the optical flow of the window. Subsequently, this flow data is post-processed to outline the largest gap's contour and pinpoint its center. With the center identified we employ visual servoing to guide the quadcopter to align its image center with the gap's center, facilitating a successful flight through the gap.**

I. Introduction

In the rapidly evolving field of autonomous drone technology, the capability to navigate through complex environments remains a crucial challenge. This project introduces an innovative solution, focusing on the development of a specialized perception stack for the DJI Tello EDU quadcopter. Our aim is to enable the quadcopter to autonomously identify and navigate through irregularly shaped, unknown windows—a task that simulates complex real-world scenarios where drones might operate in urban environments or disaster zones.

The process is initiated by flying the drone to a position where the entire gap is visible. To detect the gap, we utilize the Recurrent All-Pairs Field Transforms (RAFT) network for optical flow detection. We exploit the drone's slightly unstable hovering to gather images for input to our RAFT network which provides a substantial optical flow for analysis.

The next phase involves processing this flow to delineate the contour of the gap and accurately locate its center. This is a critical step in our post-processing routine. Once we have identified the center of the gap, the challenge is to precisely position the drone such that its image center aligns with the gap's center in the image. Achieving this alignment ensures that the drone can move forward through the gap without colliding with the surrounding wall.

To accomplish this precise positioning, we employ visual servoing techniques. These techniques allow us to navigate the drone effectively, ensuring that the image center and the gap center are closely aligned. Once this alignment is achieved, we are ready to guide the quadcopter through the window, successfully completing our objective of autonomous navigation through irregularly shaped gaps.

II. Implementation

We realized that motion is important to detect unknown-shaped windows, especially in scenarios where the background and foreground have similar textures. Therefore, we explored methods for identifying the largest gap in the wall given we have two images taken from different quadcopter positions in front of the wall such that the gap to be detected is visible in both the images.

A. Watershed algorithm

In our project's initial stages, we utilized the watershed algorithm for gap detection. This method first used Otsu's method to distinguish foreground from background. We then employed the watershed algorithm, which effectively segments an image into different regions based on topographical features. Despite its potential, this method was ultimately discarded due to its ineffectiveness in environments where foreground and background had similar colors and textures, as it is based on color thresholding. Different scenarios with the algorithm are shown in fig 1 and fig 2.

B. Edge-Based Gap Detection Using Sobel Operators

In a subsequent phase of our project, we shifted to a method that was based on the premise that as the quadcopter moves, the background visible through the gap changes, creating a distinct intensity change at the gap's boundary compared to other areas.

This involved capturing two images at different positions, aligning them using the SIFT algorithm and optical flow calculations, and then applying Sobel operators to create gradient maps. By normalizing these gradient maps and then subtracting them from each other, we aimed to isolate the gap by exploiting the differences in edge intensity caused by the background change. Despite the theoretical promise of this method in highlighting the gap by detecting edge intensity variations, it proved impractical. The sensitivity of the Sobel

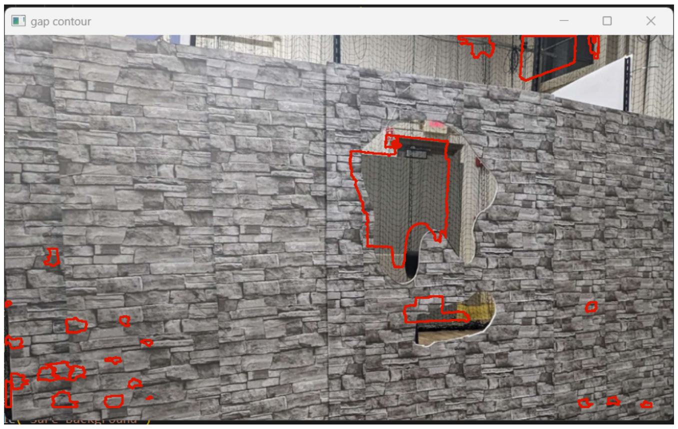

Fig. 1. Output of watershed algorithm

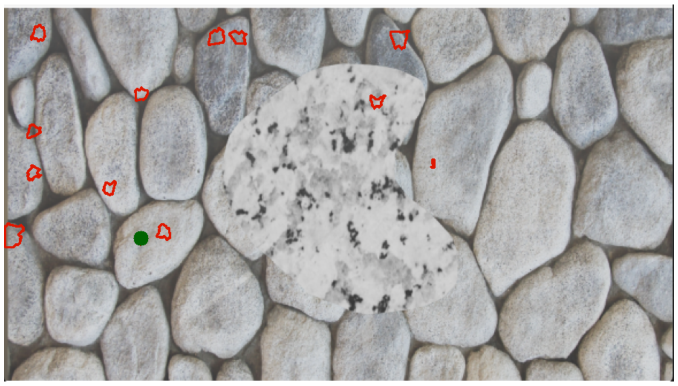

Fig. 2. Watershed algorithm failure case

operators led to the detection of excessive edges, making it challenging to isolate the largest gap. Consequently, this approach was deemed ineffective and abandoned in favor of more reliable techniques.

C. Optical Flow Estimation using RAFT

The fundamental assumption of optical flow is that the brightness of any object point in an image remains constant over time. Mathematically, this is represented by the brightness constancy constraint equation:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{1}$$

where $I(x, y, t)$ is the intensity of the pixel at position (x, y) at time t, and dx, dy are the displacements of the pixel in the x and y directions between times t and $t + dt$.
In the context of drone navigation, optical flow aids in understanding the movement of the drone relative to its surroundings. By analyzing the optical flow of frames captured during the drone's flight, it is possible to infer motion patterns and make navigational decisions. The procedure involves:

1) Capturing consecutive frames during the drone's movement.
2) Applying computer vision techniques to these frames to calculate optical flow, which represents the motion between them.

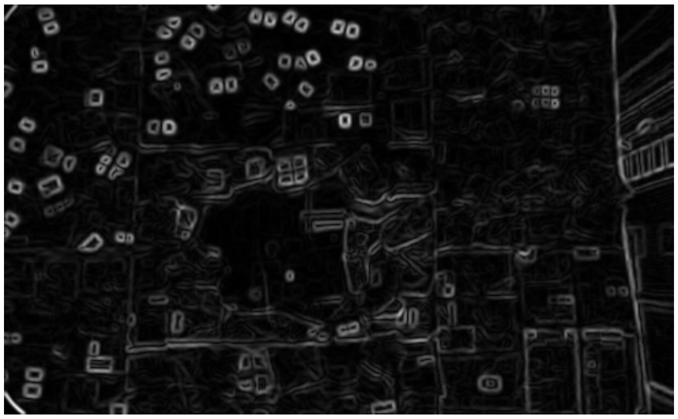

Fig. 3. Normalized gradient map at first position using Sobel operator

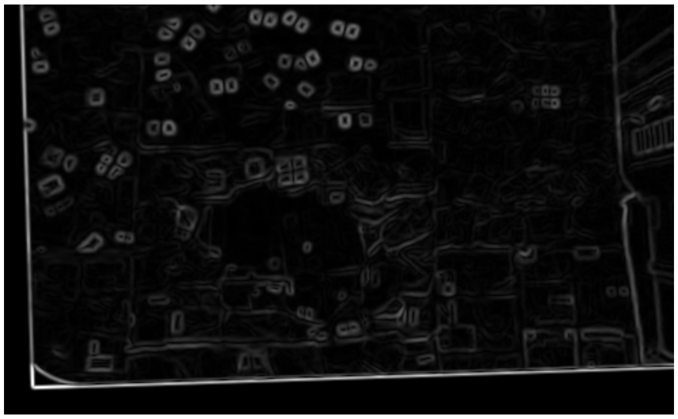

Fig. 4. Normalized gradient map at second position using Sobel operator

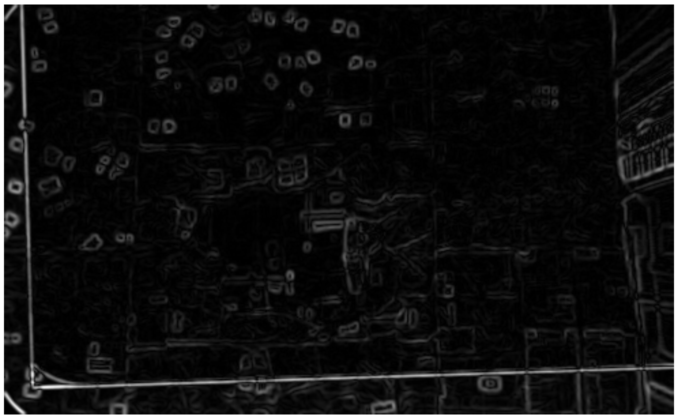

Fig. 5. Difference of the gradient maps

3) Post-processing the optical flow data to extract meaningful information like the movement direction and magnitude.

In this specific application, the drone captures two frames while performing a vertical movement (up and down). The relative motion induced by this movement enhances the quality of the optical flow output. This output is then processed to generate contours, which help in identifying and locating gaps

or openings (like windows) in the environment when fitted with convex hulls. The center of these gaps is calculated using the optical flow data, assisting the drone in aligning itself accurately for navigation or other tasks.
In summary, optical flow provides a dynamic and robust method for real-time analysis of the drone's environment, playing a pivotal role in autonomous navigation and obstacle avoidance.

D. Use of RAFT Pre-trained Optical Flow Network

For the optical flow image generation, we utilized the RAFT (Recurrent All-Pairs Field Transforms) pre-trained network, known for its accuracy and efficiency in calculating optical flow. One of the key factors influencing our choice was RAFT's performance when combined with CUDA processing, allowing it to generate optical flow outputs in under one second. This rapid processing capability is crucial for real-time applications, especially in scenarios requiring immediate response and adaptation, such as drone navigation. By integrating RAFT with CUDA, we ensured that the optical flow computations are not only accurate but also swift, enabling the drone to process visual data and make navigational decisions in real-time. This efficiency negates the need for the drone to hover or stall while waiting for inference results, thereby enhancing operational efficiency and responsiveness. Such real-time processing capabilities are vital for applications involving dynamic environments and time-critical tasks, making RAFT an ideal choice for our drone navigation system.
The RAFT model as mentioned in the paper [1] is trained on different datasets and we found the weights trained for Sintel dataset gave the best results for us. First the network is tested for different gap shapes and foreground and background textures in blender. These can be seen in figure 6. Further we tested on a scenario where both foreground and background textures are exactly same as shown in figure 7 and it worked here as well giving us the confidence to proceed on to real world usage. The intersection over union is calculated for each case and is shown in figure 8. It can be seen that the IOU is slightly less in the same texture scenario (second image from left) which is understandable.
The network and post-processing steps discussed in the next section are implemented on the drone and figure 9 shows all the flows detected during one of the successful runs (the video links to which are in conclusion section). Here the drone initially gets a flow and calculates the distance between the gap center and the image center and corrects its position. In row 2 we see the same calculations being done and as this is within tolerance we go through the gap.

III. Post-processing

In our post-processing, adaptive thresholding was first employed on the optical flow image obtained. Unlike Otsu's method, which is a global thresholding technique, adaptive thresholding adjusts to varying lighting conditions across different parts of the image. It transformed the image such that the key features, especially the edges of the gaps, were highlighted.
Then we performed edge detection using the Canny algorithm which is well-regarded for its effectiveness in identifying strong edge areas in an image. The detected edges were then dilated to connect discontinuous parts of objects, followed by a morphological 'closing' operation to fill in small holes within the edges. This step was crucial in ensuring a continuous and clear representation of potential gaps.
After processing the image, we moved to contour detection, a process of finding continuous lines or curves that bound or cover the full boundary of objects in the image. We identified and sorted these contours based on their area, focusing on the largest contour as the likely representation of the largest gap. This was a pivotal step as it transitioned our process from general edge detection to pinpointing a specific target.
For the largest contour, we calculated its centroid using image moments, a method that provides the average of the contour's coordinates, effectively identifying the center of the gap. This gap center is the key feature that will next be used for visual servoing of the quadopter for positioning itself effectively to fly through the gap.

IV. Visual Servoing

Our visual servoing process consists of the following steps:

1) **Calculate Distance between Centers:**
   - Find the center of the gap in the image.
   - Calculate the Euclidean distance between this center and the center of the image.
   - This distance helps determine how far the drone is from being directly aligned with the target in the 2D plane withput depth towards the gap.
2) **Check Distance Against Tolerance:**
   - If this distance is less than a tuned tolerance value, the drone is close enough to proceed through the window.
   - Otherwise, the drone needs to align its position with the gap center.
3) **Horizontal Adjustment:**
   - Calculate the horizontal difference by comparing the coordinates of the image center and the center of the gap.
   - Convert this difference into a horizontal movement command, scaling it by a conversion factor (that is tuned after multiple trials).
   - This command directs the drone to move left or right, aligning it horizontally with the gap center.
4) **Vertical Adjustment:**
   - Similarly, calculate the vertical difference using the coordinates of the image center and the gap center.
   - Convert this vertical difference into a vertical movement command, also scaled by the same conversion factor.
   - This command instructs the drone to move up or down, aligning it vertically with the target.
5) **Movement Execution:**

- The drone moves horizontally and vertically based on the calculated commands.
- The magnitude and direction of these commands dictate the speed and direction of the drone's movement.

In our experiments we observed that if the height of the drone initially is level with the gap, then we do not need the vertical adjustment mentioned above which reduces the additional tuning. Furthermore if you look at figure 9 (Row 2 Last image) before the drone went through the window, the circle radius measuring the tolerance is large but the drone still goes through the window, this is because we found that compared to the body frame the front camera of the tello drone is slightly tilted down increasing the tolerance limit.

V. Conclusion

In this project we were able to demonstrate that a robust deep learning neural net for optical flow estimation can be used to identify a gap of unknown shape and texture in real-time. A post-processing approach is developed to identify the gap and its center. A visual servoing approach is explored to aligh the drone with the gap center to go through it. In the future a better and more robust gap tracking algorithm can be explored.

Please use the following links to look at videos of successful run of the drone in different views: link1: drone POV, link2: Camera man view.

Some of the challenges faced across the project were discussed thoroughly in the report. Additionally since the odometry of the Tello drone is not perfect some of the runs are unsuccessful because the drone won't accurately go to the specified coordinates. This is a hardware limitation to which the solutions should further be explored to create more reliable trajectories. Overall the performance of the drone is good and it passes through the gap reliably.

References

[1] Teed, Zachary, and Jia Deng. "Raft: Recurrent all-pairs field transforms for optical flow." Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. Springer International Publishing, 2020.

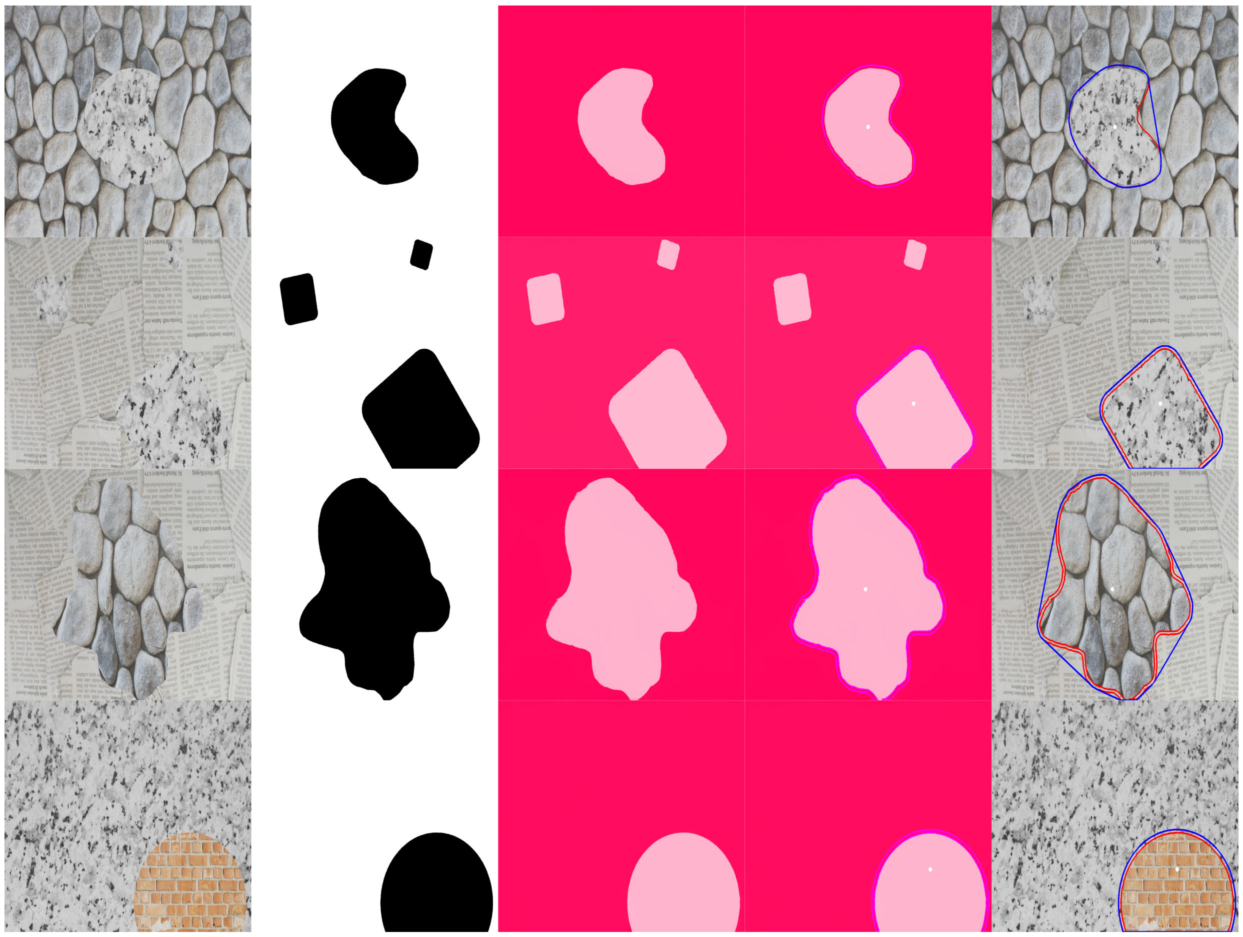

Fig. 6. Results of RAFT and post processing in different scenarios on blender. (Left to Right: Camera frame, Groundtruth blender masks, Flow detected, Gap contour identified on flow, Gap contour and Convex hull.)

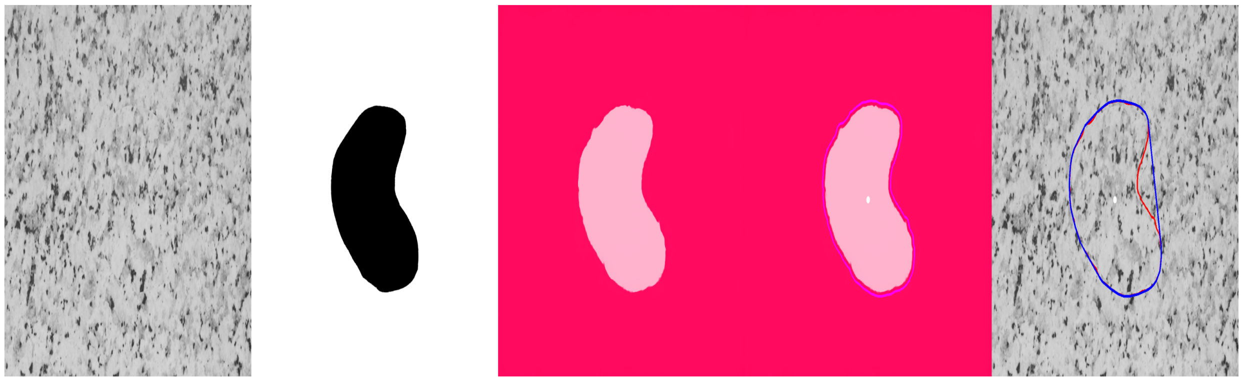

Fig. 7. Results of RAFT and post processing when both foreground and background textures are same on blender.(Trust me there is a gap there)

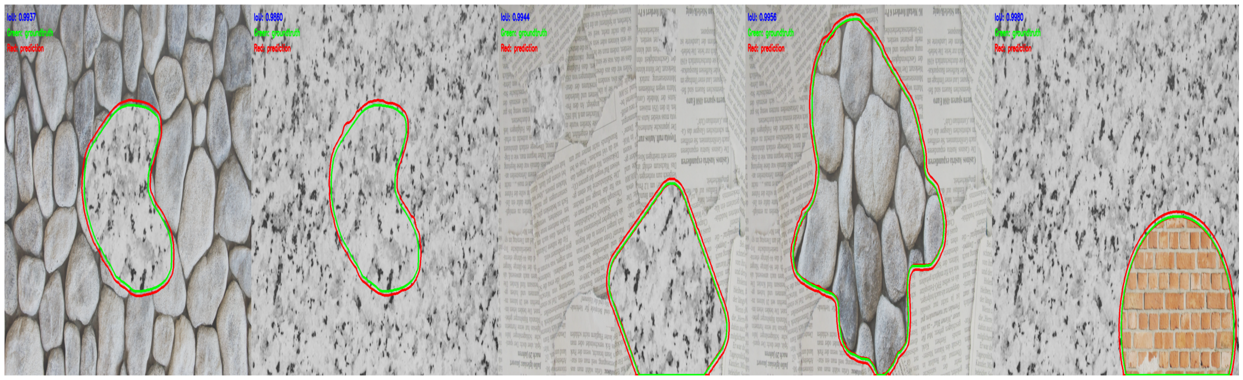

Fig. 8. Intersection over Union for blender simulation

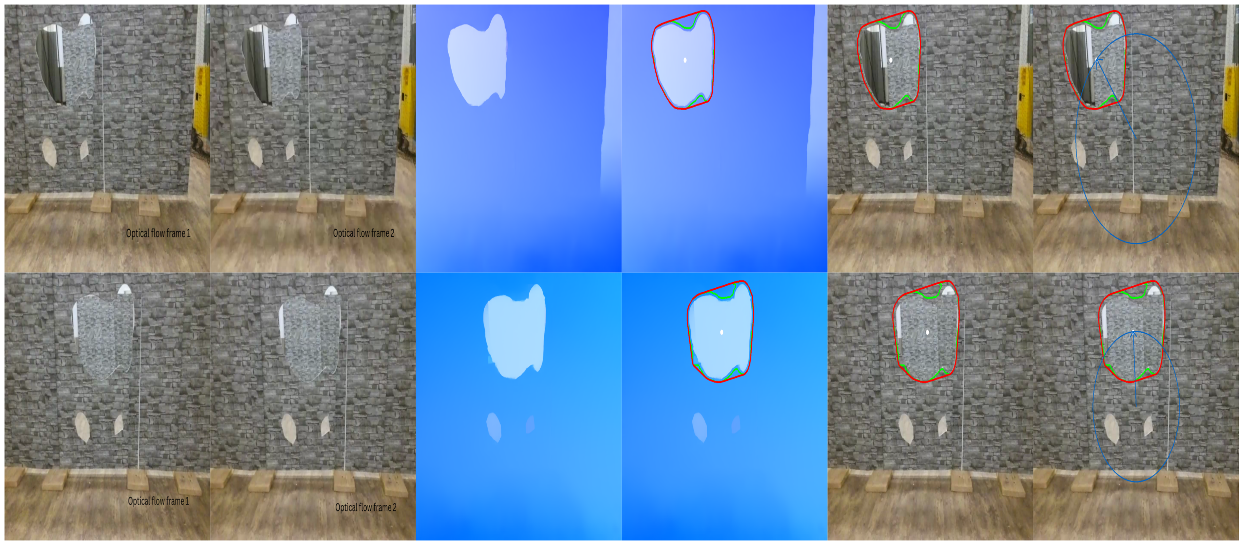

Fig. 9. Results from a real world run. Row 1: Flow estimation at the initial location, Row 2: Flow estimation after correction and just before going through gap. (Left to right: Frame 1, frame 2 used for flow, Optical flow, Gap contour and center identified on flow, Contour and center on frame, Distance between gap center and image center)