RBE595 - Project 5 - Navigating through Windows and Gaps with Optical Flow

Pranay Katyal
Worcester Polytechnic Institute
Worcester, MA, USA
pkatyal@wpi.edu

Anirudh Ramanathan
Worcester Polytechnic Institute
Worcester, MA, USA
aramanathan@wpi.edu

Hudson Kortus
Worcester Polytechnic Institute
Worcester, MA, USA
hkortus@wpi.edu

***Abstract*—The goal of this project was to develop a perception stack capable of detecting generic gaps in a planar scene within the VizFlyt simulation environment, and enabling autonomous navigation through them using an existing navigation and control stack. With the use of a pre-trained RAFT model and the addition of the TS2P algorithm, the system was designed to monitor the optical flow of the overhead scene to detect and navigate an obstacle course forward and backward within a constrained time. An error-correction control loop with a manual servoing method was integrated to initially align the drone's sensor plane to locate the windows and gaps in the scene for accurate re-orientation and traversal to and back from the endpoint.**

I. Problem Statement

Project 3 involved developing a perception stack for a constrained navigation setting where rectangular window frames offered clear geometric cues that simplified detection and path planning. Project 4 addressed an unconstrained navigation scenario in which the system had to interpret complex visual conditions, extract the outline of an irregular opening within a textured wall, and navigate through it using only RGB imagery.

In the current project, these two approaches are merged into a unified perception stack designed to operate in a composite environment that features three consecutive rectangular windows followed by a textured wall containing an arbitrary hole near the end of the course. The system must detect and traverse each structure in sequence, then repeat the course in reverse, while remaining fully robust to the added challenge that the rear faces of the windows and wall contain no texture at all, requiring consistent shape detection and classification without reliance on appearance cues.

Unlike some of the previous projects, the VizFlyt quadrotor in this setup does not provide RGB-D depth data. As a result, all necessary spatial understanding had to be inferred solely from RGB imagery using image processing and mathematical modeling.

II. Methodology

The assignment simulated the motion of an aerial robot with the dynamics of a DJI Tello drone [1]. The drone dimensions were found from [2] and state the the drone's length, width, and height are 98, 95, and 41mm respectively. For the ease of simulation we simplified to drone to be a 100x100x200mm cuboid. We chose the slightly inflated x,y dimensions primary for convenience and extra safety margin. We chose to significantly increase the z height of the quadcopter because if an object came this close to the top or bottom of it in real life it would alter the propeller aerodynamics enough to cause a crash.

Fig. 1: Tello Drone

A. Window/Gap in Scene

The RGB camera onboard the simulated drone provides a continuous stream of frames that must be processed to detect navigable gaps under a wide range of visual conditions. Reliable detection requires removing environmental noise and isolating foreground structures even when foreground and background textures are identical. Classical thresholding or color-based extraction offers limited performance because these methods depend on assumptions about texture, lighting, and geometry that fail under arbitrary scene configurations. As a result, conventional image processing cannot consistently reveal gap boundaries, especially when visual contrast is absent. These limitations motivate the use of motion-based perceptual cues that can operate independently of appearance.

B. Optical Flow

To achieve consistent gap detection, the system relies on optical flow to estimate pixel motion between consecutive

frames. Optical flow is a powerful tool for identifying motion discontinuities that arise due to parallax, which naturally differentiates between foreground elements and background surfaces at different depths. This depth reasoning occurs even when textures are identical, making optical flow suitable for detecting gaps that would be indistinguishable in static imagery. By leveraging motion-induced contrast rather than appearance-based contrast, the system can recover reliable information about scene structure. This provides the foundation for the gap segmentation stage described later.

C. RAFT Neural Network

The pre-trained model used in this project was RAFT (Recurrent All-Pairs Field Transforms) [3], a state-of-the-art deep learning approach for dense optical flow estimation. Unlike traditional methods that rely on local pixel correspondences, RAFT computes correlations between all pairs of pixels in consecutive frames and iteratively refines the flow through a recurrent update mechanism. This enables the model to produce highly accurate and detailed flow estimates even in regions with large motion, low texture, or complex visual patterns. RAFT's robustness to variations in lighting, texture, and occlusions makes it particularly effective in realistic navigation scenarios. We mainly have 4 windows from the start, 3 Normal racing windows in Red Color checkered pattern, and 1 Irregularly shaped window on which PnP fails, so we navigate through it via pure tssp-optical flow method.

D. TS2P Extension to RAFT

The perception system also uses GapFlyt [4], which incorporates the TS2P (Temporally Stacked Spatial Parallax) algorithm as a temporal and spatial smoothing procedure applied on top of the original RAFT network. TS2P is based on the minimalist, structure-less philosophy described in GapFlyt, where the objective is not to reconstruct a full 3D model of the environment but instead to extract only the information required for the task of navigating through an unknown opening. The algorithm leverages active vision principles by exploiting controlled camera motion to amplify motion parallax, creating a stronger distinction between foreground and background depth layers across stacked frames. TS2P computes temporally averaged and spatially refined inverse optical flow fields, which emphasize depth discontinuities associated with actual openings regardless of the shape, texture, or appearance of the surrounding surfaces. This temporal stacking also increases robustness by reducing frame-level noise and suppressing dynamic artifacts, while the spatial component enhances edge fidelity around the gap boundary, producing a clearer and more stable representation of the opening than using RAFT alone.

By fusing RAFT's dense flow predictions with these temporally accumulated parallax cues, TS2P yields motion fields that are smoother, more coherent, and naturally aligned with the physical depth structure of the scene. This results in more efficient gap detection because the system no longer depends on any appearance or texture cues from the foreground or background, which allows continuous traversal of the entire course and a reliable return path even when the visual textures on the walls differ significantly or offer no helpful contrast at all.

E. Window/Gap Segmentation

Raw optical flow alone cannot directly identify navigable passages, so several post-processing steps are applied to extract gap locations. The magnitude of the horizontal and vertical flow components is computed to obtain a simpler scalar field that highlights motion differences between foreground structures and the background. Pixels with the largest flow magnitudes are labeled as foreground, while the remainder are treated as background, allowing isolated enclosed regions to be extracted. Only background blobs fully surrounded by foreground are considered valid holes, excluding open edges that lead out of the scene. From the remaining set of valid gaps, the largest regular/irregular blob is selected, its centroid is computed, and its displacement from the frame center is passed to the visual servoing controller.

An example of this window-segmentation pipeline is shown for both the rectangular window frames as well as the textured wall in Fig. 2 and Fig. 3. From left to right, the first image is the RGB VizFlyt image, the second is the optical flow magnitude, the third is the masked image of the window/gap, and the final image shows the detection of the window/gap.

This approach is reliable as long as the frame does not dominate the entire field of view. This assumption is reasonable: if the drone is already close enough for the frame to fill the image, it is almost always close enough to simply fly through the opening without further alignment.

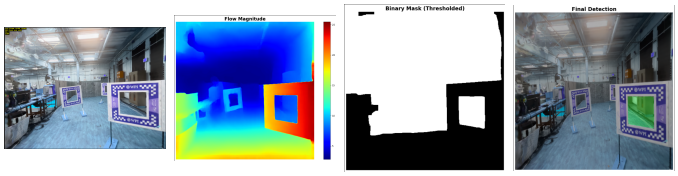

Fig. 2: Window Segmentation Pipeline

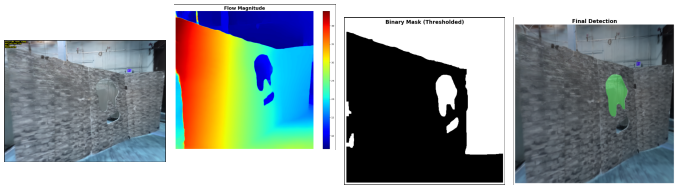

Fig. 3: Gap Segmentation Pipeline

Two additional examples, shown in Fig. 4 and Fig. 5 demonstrate that the GapFlyt implementation operates reliably regardless of the texture of the target.

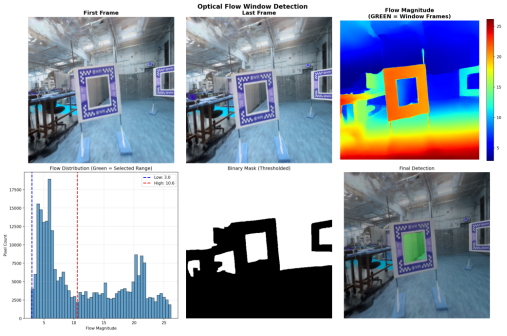

Fig. 4: Window Front Segmentation Pipeline

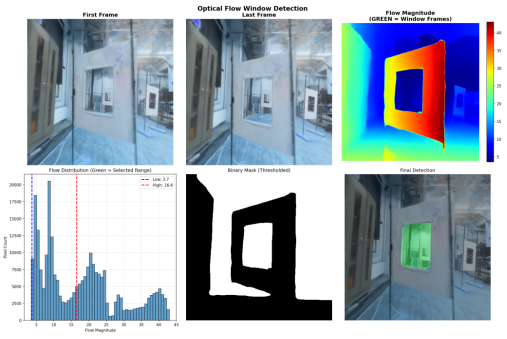

Fig. 5: Window Back Segmentation Pipeline

F. Visual Servoing

Camera intrinsics are a fundamental component of the visual servoing pipeline because they define the geometric relationship between image measurements and the physical motion of the drone. Visual servoing relies on image-space errors, such as the displacement between a detected target and the image center, and converts these errors into control commands that adjust the drone's position and orientation. To perform this conversion consistently, the system must know how pixel coordinates relate to rays in 3D space, which is determined entirely by the camera's intrinsic parameters. Without accurate intrinsics, pixel-level offsets cannot be interpreted correctly, leading to unstable or inaccurate alignment during flight.

In this project, the camera intrinsics are computed directly from the known horizontal field of view of the simulated camera using a standard pinhole camera model. The focal length in pixels along the horizontal axis is calculated as

$$f_x = \frac{W}{2\tan(\mathrm{FOV}_h/2)}, \tag{1}$$

where W is the image width in pixels and FOV_h is the horizontal field of view expressed in radians. Square pixels are assumed, so the vertical focal length is set equal to the horizontal focal length, $f_y = f_x$. The principal point is assumed to be located at the center of the image, with $c_x = W/2$ and $c_y = H/2$, and the model assumes zero skew and no pre-applied lens distortion. These values are assembled into the intrinsic camera matrix

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

This intrinsic matrix allows image-space measurements to be expressed in a normalized camera frame, enabling the visual servoing controller to compute meaningful alignment and motion commands from pixel-based observations during navigation.

Control inputs are computed by multiplying the pixel displacements by a scaling factor of 0.0008. While this approach is extremely crude, it is simple, fast, and far more robust than attempting to estimate the physical size of the gap to determine the required motion.

Once the robot has centered itself with respect to the gap, we simply command it to fly through and continue to the end of the course.

III. Results

A. Race Time

After tuning the controller, velocities, and visual servoing heuristics, our drone was able to reliably complete the course and render the frames in 264.60 seconds (4.41 minutes) of simulation time, with a total inference time of 0.729 seconds. Fig. 6 shows the overall simulation time from the saved logs.

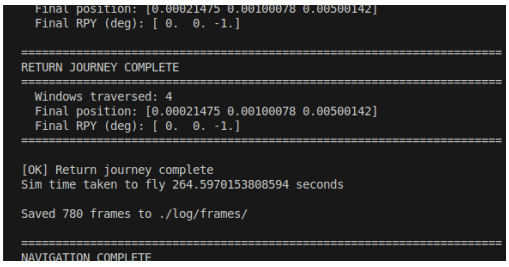

Fig. 6: Execution Results

IV. Challenges

Throughout this project, we encountered many challenges that hindered our progress.

One major issue arose from the collision checker, which, while capable of detecting collisions and halting the simulation, inherently prevented the drone from passing through windows or gaps. The checker proved overly sensitive, leaving no tolerance for traversal regardless of how extensively the forward distance and safety parameters were tuned. As a result, the collision checker was ultimately excluded from the navigation skills, and collision avoidance was instead handled through precise manual servoing, allowing the drone to achieve high-accuracy alignment and safely navigate through openings. Despite this adjustment, collisions with the back wall of the fourth window still occur and are treated as unavoidable failures within the current system.

Another challenge arose from the provided controller, which did not function as intended within our system. As a result, the controller was redefined and reimplemented from scratch to better align with the structure, specifications, and requirements of our codebase. Clarification was requested regarding debugging the original controller, but timely guidance was not available, so development proceeded independently. This approach ultimately resulted in a functional and more tightly integrated controller tailored to the needs of the project.

V. Conclusion

In this project, a perception system was implemented on a simulated drone, which enabled it to interpret the scene and detect both regular windows and arbitrarily shaped gaps. Using the RAFT optical flow model with the addition of TS2P, the system identified the rectangular window or largest gap, determined its center through segmentation and contour analysis, and flew through the identified area.

The drone starts at a location that was given to us, $[0.001, 0, 0]$ and with a given starting pose of $[0, 0, 10]$ roll, pitch, and yaw values.

The drone then employed a visual servoing strategy with a PID controller to align its camera center with the detected window/gap. Once alignment was achieved within a minimal error threshold, the drone navigated through the gap while maintaining its trajectory relative to the window center. This drone navigated a single gap after which it completed its mission.

This work demonstrates the effectiveness of utilizing RAFT-based optical flow with additional temporal and spatial enhancements contributed by TS2P for the perception and visual servoing for autonomous navigation in complex, visually challenging environments. It highlights how powerful optical flow is, and how one can navigate in an environment even without any depth information.

Key accomplishments include:

- Goal reaching accuracy within 10px.
- Successful implementation of GapFlyt optical flow estimation model to achieve general segmentation and detection.
- Successfully ran inferences in almost real time, in a Simulated environment made with VizFlyt.
- An inference time of 789 ms (1.26 Hz) was achieved.

A. Combined Video

The combined video for the P5 project can be found here.[5]. This video showcases a side-by-side view of the RGB render and Optical flow video. When the model is inferring, it briefly switches to the segmentation view, and when it's servoing its back to normal vision.

VI. Contribution

Hudson contributed 33 percent of the overall work. His efforts focused on adapting and migrating the RAFT-based perception pipeline from the previous assignment and implementing a substantial portion of the pose estimation system using a PnP formulation.

Anirudh contributed 33 percent of the overall work. He worked on combining the U-Net and RAFT approaches from earlier projects into a single unified perception module to enable detections using the higher accuracy model. He also contributed to part of the manual servoing method and performed parameter tuning and refinement to make sure the drone could align and fly through the window.

Pranay contributed 33 percent of the overall work. He integrated the GapFlyt framework into the perception system for this project and developed some of the manual servoing approach, ensuring that the drone could adjust, align, and successfully traverse the scene.

References

[1] "Tello," https://store.dji.com/product/tello, DJI (Ryze Tech), 2025, accessed: 2025-09-30.

[2] "Tello specs," https://www.ryzerobotics.com/tello/specs, Ryze Tech, 2025, accessed: 2025-09-30.

[3] Princeton Vision and Learning Group, "Raft: Recurrent all-pairs field transforms for optical flow," https://github.com/princeton-vl/RAFT, 2020.

[4] N. J. Sanket, C. D. Singh, K. Ganguly, C. Fermüller, and Y. Aloimonos, "Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight," https://doi.org/10.1109/LRA.2018.2843445, 2018.

[5] P. Katyal, A. Ramanathan, and H. Kortus, "Project files for rbe595 - project 5 - video," YouTube, 2025, accessed: 2025-12-12. [Online]. Available: https://youtu.be/f01C6C_SvAk