RBE595: Hands-On Autonomous Aerial Robotics P4: Navigating The Unknown!

Shreyas Devdatta Khobragade
Department of Robotics Engineering
Worcester Polytechnic Institute
Email: skhobragade@wpi.edu
USING 4 LATE DAY

Brijan Vaghasiya
Department of Robotics Engineering
Worcester Polytechnic Institute
Email: bcvaghasiya@wpi.edu
USING 4 LATE DAY

***Abstract*—Safe flight of quadrotors through narrow openings such as windows requires robust perception of gaps under strong background–foreground blending and perspective distortions. Appearance-based segmentation often fails when the window frame and far background share similar color and texture. This work presents a complete perception-and-control pipeline that exploits parallax and dense optical flow to detect a window gap and visually servo a quadrotor through it in simulation. We employ RAFT optical flow between successive views generated in a SplatRenderer-based "VizFlyt" quadrotor simulator, convert the flow visualization to an edge map via Canny filtering, and extract the dominant closed contour corresponding to the gap. The centroid of this contour is mapped to incremental motion commands in the North–East–Down (NED) frame, which are tracked by a dynamic quadrotor model and PID-based controller along a trapezoidal velocity trajectory. Temporal consistency checks reject spurious detections, frames without a valid gap are skipped and the vehicle simply moves forward to generate a new parallax baseline. We validate the approach on synthetic Blender scenes and in the VizFlyt simulator. Results show that the proposed pipeline can reliably center the vehicle with respect to the window and drive it through the opening, despite challenging background blending and imperfect edges. We also discuss limitations and directions toward real-world deployment.**

***Index Terms*—Quadrotor navigation, visual servoing, RAFT optical flow, parallax, gap detection, SplatRenderer, Blender, simulation.**

I. Introduction

Quadrotor flight through constrained openings such as windows is a core benchmark for agile autonomous navigation. Robustly detecting the opening is difficult when the window contains glass or when the background visible through the window has similar color and texture to the surrounding wall. In such cases, simple color thresholding or even appearance-based semantic segmentation may fail due to foreground–background blending. Parallax provides an alternative cue: when the camera undergoes a lateral or forward motion, nearby structures (wall and frame) produce larger image displacements than distant background structures seen through the opening. Dense optical flow therefore encodes strong gradients along gap boundaries. Modern networks such as RAFT estimate dense optical flow with high accuracy even under large displacements and complex textures by building all-pairs correlation volumes and iteratively refining the flow field [2]. In parallel, neural scene representations such as 3D Gaussian Splatting enable photorealistic, real-time rendering of scenes from arbitrary viewpoints by representing the environment as a set of anisotropic 3D Gaussians and splatting them into the image plane [3]. This capability allows us to test perception algorithms with realistic lighting, textures, and geometry without resorting to expensive real-world experiments. In this work we combine these tools with an image-based visual servoing framework. Using VizFlyt, a quadrotor simulator coupled to a SplatRenderer, we generate a sequence of views as the quadrotor approaches a window. Between successive views we compute RAFT flow, infer an approximate window mask from the flow visualization, and then compute a gap centroid. A simple proportional mapping converts the centroid's deviation from image center into small NED displacements. These displacements are executed by a dynamic quadrotor model using a trajectory-tracking controller. The process repeats until the window appears centered and sufficiently large, at which point the vehicle is commanded to fly through the gap. The remainder of this paper details the algorithm and experimental evaluation.

II. Methodology

A. Coordinate Frames and Notation

We adopt a NED world frame $\mathcal{F}_{\mathrm{N}}$ with state variables

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3, \qquad \boldsymbol{\eta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in \mathbb{R}^3, \tag{1}$$

where x is forward, y is right, and z is down, ϕ, θ, ψ denote roll, pitch, and yaw, respectively. Images follow the OpenCV convention with pixel coordinates (u, v) and dimensions $W \times H$, u increases to the right, v downward. The quadrotor state for dynamics integration is

$$X = \begin{bmatrix} \mathbf{p}^\top & \mathbf{v}^\top & \mathbf{q}^\top & \boldsymbol{\omega}^\top \end{bmatrix}^\top, \tag{2}$$

with linear velocity $\mathbf{v}$, unit quaternion $\mathbf{q}$, and body angular velocity $\boldsymbol{\omega}$.

B. RAFT Optical Flow and Parallax

Given two consecutive RGB images I_{k-1} and I_k, a pre-trained RAFT network Φ_θ computes a dense optical flow field

$$\mathbf{F}_k(x, y) = \Phi_\theta(I_{k-1}, I_k)\,(x, y) = \begin{bmatrix} u_k(x, y) \\ v_k(x, y) \end{bmatrix}, \tag{3}$$

where u_k, v_k are horizontal and vertical displacements defined for every pixel [2]. For visualization and downstream processing, the flow is converted into a color image $\hat{F}_k$ via a standard flow-to-color mapping $\mathcal{C} : \mathbb{R}^2 \to \mathbb{R}^3$,

$$\hat{F}_k(x, y) = \mathcal{C}\big(\mathbf{F}_k(x, y)\big), \tag{4}$$

which encodes flow direction as hue and magnitude as saturation. With the quadrotor executing small forward and lateral motions, the wall and window frame (near structures) produce larger $\|\mathbf{F}_k\|$ than the distant background visible through the window. This relative motion yields high-contrast edges along the gap boundary in $\hat{F}_k$.

C. Gap Mask Extraction

Gap detection is formulated as the extraction of a single dominant closed contour from the flow visualization. Let $\hat{F}_k$ be the BGR image at step k.

1) Grayscale and smoothing: We form a smoothed grayscale image

$$G_k(x, y) = \big[\mathrm{Gray} * \mathcal{N}(0, \sigma^2)\big](\hat{F}_k)(x, y), \tag{5}$$

where $\mathrm{Gray}(\cdot)$ converts BGR to grayscale and $\mathcal{N}(0, \sigma^2)$ is a Gaussian kernel with σ corresponding to a 5×5 discrete window.

2) Canny edge detection: The Canny operator computes gradient magnitude $M(x, y)$ and direction $\theta(x, y)$ from G_k. An edge map E_k is formed by hysteresis thresholding using lower and upper thresholds τ_{low} and τ_{high}:

$$E_k(x, y) = \begin{cases} 1, & M(x, y) \text{ linked to a pixel with } M \geq \tau_{\text{high}}, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

with $\tau_{\text{low}} = 10$ and $\tau_{\text{high}} = 40$ (in 8-bit intensity units).

3) Binary closing: The binary edge map is thickened and small gaps are closed using morphological closing with a 3×3 structuring element K:

$$C_k = (E_k \oplus K) \ominus K, \tag{7}$$

applied iteratively three times, where $\oplus$ and $\ominus$ denote binary dilation and erosion.

4) Contour selection by area: Let $\{\mathcal{C}_i\}$ be the set of external contours extracted from C_k. For each contour, the area A_i is computed, and we define lower and upper area bounds

$$A_{\min} = \alpha_{\min} HW, \qquad A_{\max} = \alpha_{\max} HW, \tag{8}$$

with $\alpha_{\min} = 0.001$ and $\alpha_{\max} = 0.9$. A contour is valid if

$$A_{\min} \leq A_i \leq A_{\max}. \tag{9}$$

If no contour satisfies this constraint, the frame is labeled as having "no gap" and skipped.

5) Largest gap and centroid: Among all valid contours, the gap boundary is modeled as the contour with maximum area

$$\mathcal{C}^\star = \arg\max_{\mathcal{C}_i} A_i. \tag{10}$$

The centroid $\mathbf{c}_k = (c_{x,k}, c_{y,k})$ is computed from image moments:

$$m_{pq} = \sum_{(x,y) \in \mathcal{C}^\star} x^p y^q, \tag{11}$$

$$c_{x,k} = \frac{m_{10}}{m_{00}}, \qquad c_{y,k} = \frac{m_{01}}{m_{00}}. \tag{12}$$

The binary gap mask $M_k(x, y)$ is then defined as

$$M_k(x, y) = \begin{cases} 1, & (x, y) \text{ is inside } \mathcal{C}^\star, \\ 0, & \text{otherwise}. \end{cases} \tag{13}$$

D. Temporal Consistency and Frame Skipping

To avoid reacting to spurious detections, we enforce temporal smoothness on the gap centroid. Let $\mathbf{c}_k$ be the centroid at step k, and $\tilde{\mathbf{c}}_k$ the value actually used for control. We define the decision rule

$$\tilde{\mathbf{c}}_k = \begin{cases} \mathbf{c}_k, & \|\mathbf{c}_k - \tilde{\mathbf{c}}_{k-1}\|_2 \leq \tau_c, \\ \tilde{\mathbf{c}}_{k-1}, & \mathbf{c}_k \text{ exists and } \|\mathbf{c}_k - \tilde{\mathbf{c}}_{k-1}\|_2 > \tau_c, \\ \tilde{\mathbf{c}}_{k-1}, & \mathbf{c}_k \text{ undefined and } \tilde{\mathbf{c}}_{k-1} \text{ exists}, \\ \text{undefined}, & \text{otherwise}, \end{cases} \tag{14}$$

with center-jump threshold $\tau_c = 80$ pixels. If $\tilde{\mathbf{c}}_k$ is undefined, no visual alignment is applied and the quadrotor simply moves forward, generating new parallax until a stable centroid can be recovered.

E. Image-Plane Error to NED Motion

Given a valid centroid $\tilde{\mathbf{c}}_k = (u_k, v_k)$, we define normalized image-plane errors

$$e_u = \frac{u_k - W/2}{W/2}, \qquad e_v = \frac{v_k - H/2}{H/2}, \tag{15}$$

so that $e_u = e_v = 0$ when the gap is centered. The per-step NED displacement $\Delta\mathbf{p}_k$ is

$$\Delta\mathbf{p}_k = \begin{bmatrix} \Delta x_k \\ \Delta y_k \\ \Delta z_k \end{bmatrix} = \begin{bmatrix} \delta_f \\ -\kappa_y e_u \\ \kappa_z e_v \end{bmatrix}, \tag{16}$$

where $\delta_f > 0$ is a fixed forward increment, and $\kappa_y, \kappa_z > 0$ are proportional gains (typically $\kappa_y = \kappa_z = 0.30$). Lateral and vertical displacements are clamped to safety limits

$$\Delta y_k \in [-\Delta y_{\max}, \Delta y_{\max}], \qquad \Delta z_k \in [-\Delta z_{\max}, \Delta z_{\max}], \tag{17}$$

with $\Delta y_{\max} = \Delta z_{\max} = 0.06\,\text{m}$ during early steps and up to $0.12\,\text{m}$ when the quadrotor is close to the window. If no centroid is available, we set $\Delta\mathbf{p}_k = [\delta_f, 0, 0]^\top$. The waypoint for step k is then

$$\mathbf{p}_k^\star = \mathbf{p}_{k-1} + \Delta\mathbf{p}_k. \tag{18}$$

This defines an image-based visual servoing law in which the task function is the image-plane error between the gap center and the image center, the control objective is to drive this error to zero over time [4].

F. Final Traversal Through the Gap

After a fixed number of servo iterations (e.g., 16) the quadrotor is assumed to be well-centered and sufficiently close to the window, as indicated by a large gap mask area. The system then executes a short open-loop sequence of N_f forward increments of size $\delta_f = 0.05\,\mathrm{m}$:

$$\mathbf{p}_{\mathrm{final},i+1} = \mathbf{p}_{\mathrm{final},i} + \begin{bmatrix} 0.05 \\ \delta_y^* \\ \delta_z^* \end{bmatrix}, \quad i = 0, \ldots, N_f - 1, \quad (19)$$

with small bias corrections δ_y^*, δ_z^* chosen from the last servo step to remain approximately centered. During this phase, roll and pitch are constrained to zero to avoid aggressive attitudes near the gap.

III. Results

We summarize only the two main experimental setups:

- **Blender results:** synthetic window scenes were rendered in Blender with deliberately low contrast between wall, frame, and background. The proposed RAFT–parallax–edge pipeline consistently produced gap masks that closely matched manually annotated ground-truth windows. In particular, the intersection-over-union (IoU) between the predicted mask and the ground-truth window region was found to be **high, accurate, and robust across multiple Blender viewpoints**. This confirms that the flow-driven segmentation captures the true window geometry even when raw RGB appearance is ambiguous.

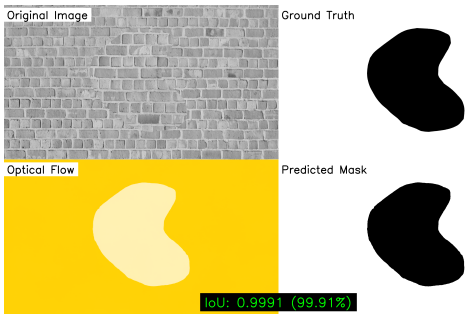

Fig. 1: Segmentation of Window 1 with Brick Texture

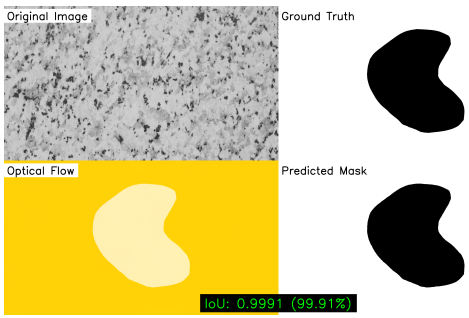

Fig. 2: Segmentation of Window 1 with Granite Texture

- **VizFlyt results:** closed-loop tests in the VizFlyt simulator using SplatRenderer, assessing whether the visual servoing pipeline can center the quadrotor and command a successful traversal through the simulated window.

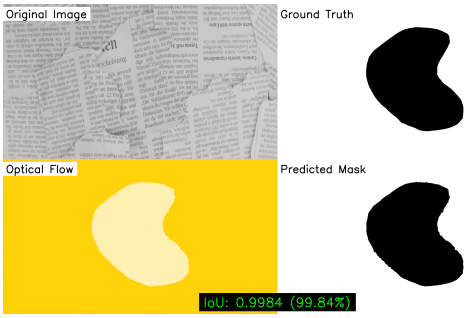

Fig. 3: Segmentation of Window 1 with Newspaper Texture

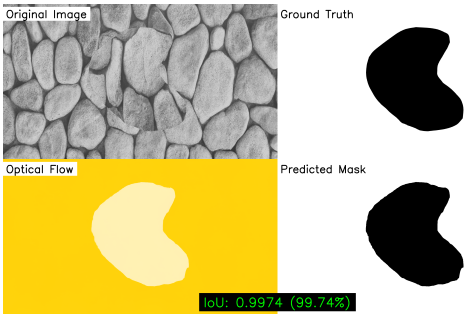

Fig. 4: Segmentation of Window 1 with Stone Texture

Fig. 5: Segmentation of Window 2 with Brick Texture

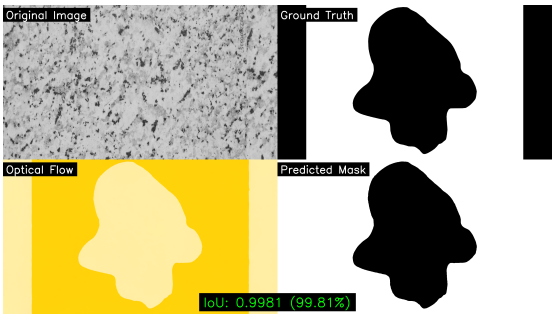

Fig. 6: Segmentation of Window 2 with Granite Texture

IV. Limitations

The proposed approach has several concise limitations:

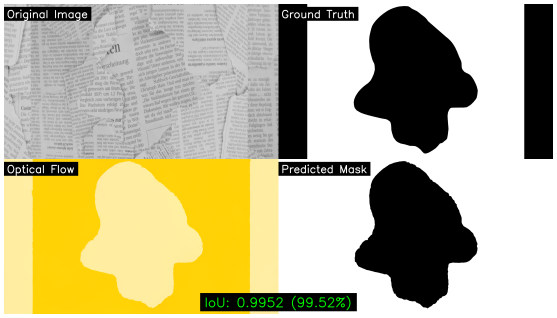

Fig. 7: Segmentation of Window 2 with Newspaper Texture

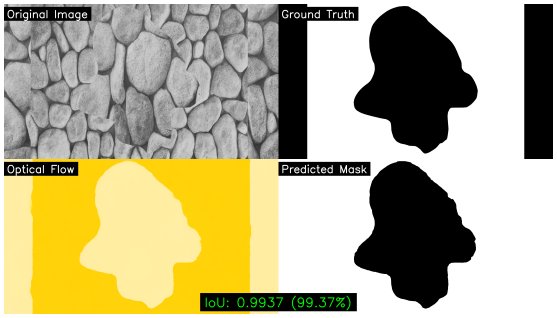

Fig. 8: Segmentation of Window 2 with Stone Texture

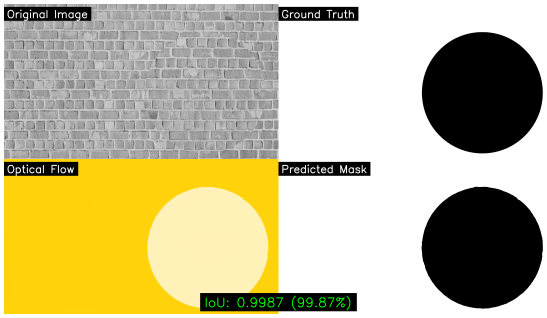

Fig. 9: Segmentation of Window 3 with Brick Texture

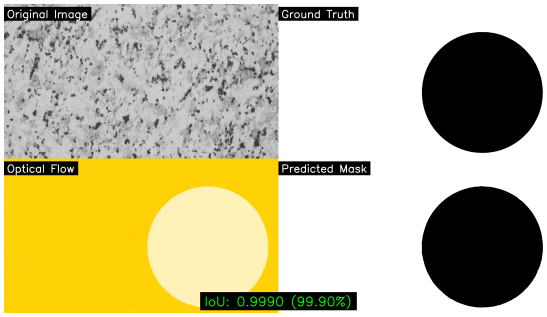

Fig. 10: Segmentation of Window 3 with Granite Texture

- It relies on RAFT quality and speed, dense flow may be too heavy for embedded hardware and degrades under blur or low light.
- Gap selection is heuristic (largest contour within area bounds) and may fail in cluttered scenes or with multiple openings.

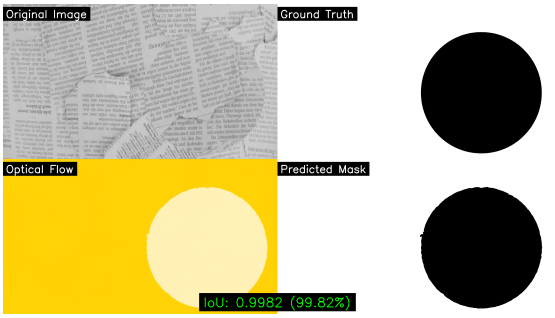

Fig. 11: Segmentation of Window 3 with Newspaper Texture

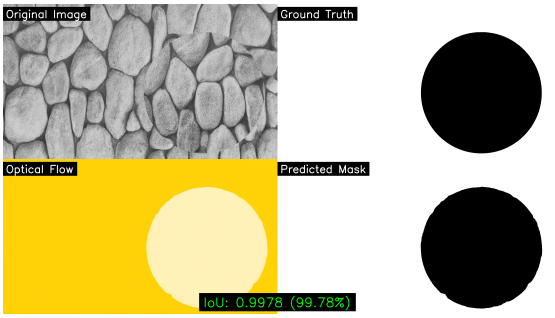

Fig. 12: Segmentation of Window 3 with Stone Texture

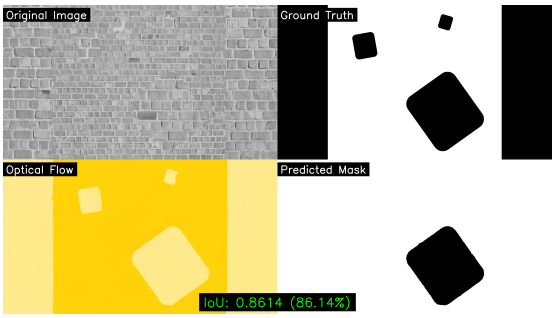

Fig. 13: Segmentation of Window 4 with Brick Texture

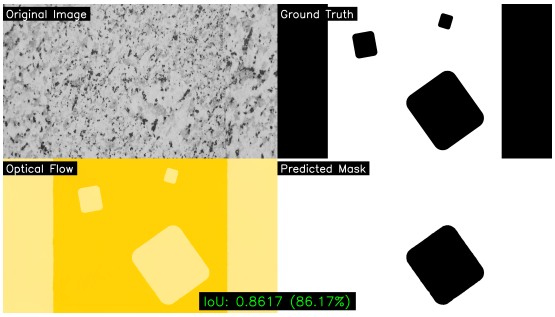

Fig. 14: Segmentation of Window 4 with Granite Texture

- The image-to-NED mapping is purely proportional, without explicit use of depth or camera intrinsics, and has only been validated in simulation.

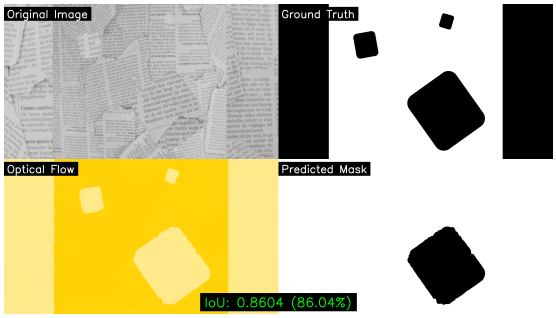

Fig. 15: Segmentation of Window 4 with Newspaper Texture

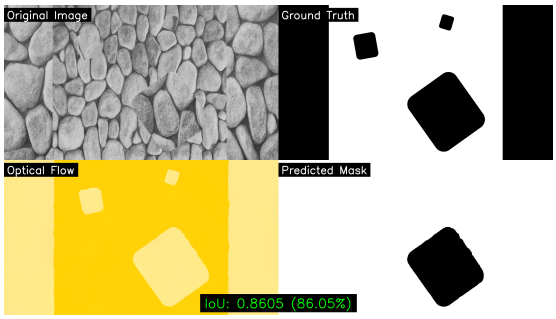

Fig. 16: Segmentation of Window 4 with Stone Texture

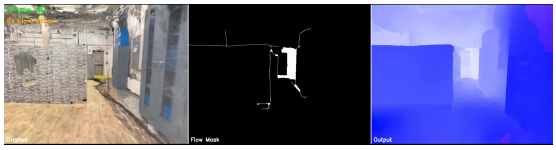

Fig. 17: No Gap Detected

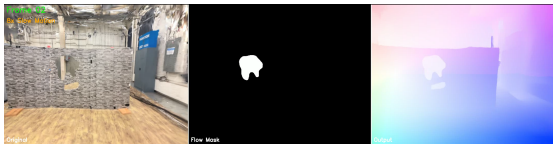

Fig. 18: Gap Detection 1

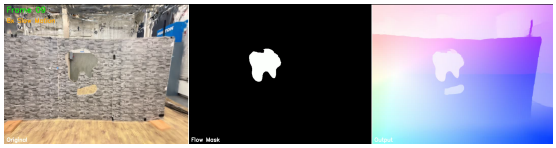

Fig. 19: Gap Detection 2

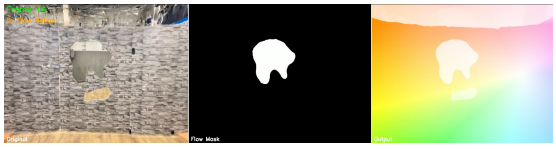

Fig. 20: Gap Detection 3

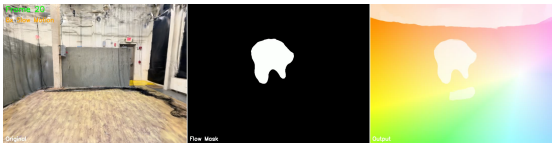

Fig. 21: Final Gap Detection 4

V. Conclusion

This paper presented a parallax-driven method for gap detection and visual servoing of a quadrotor through a window using RAFT optical flow and a Gaussian–splat-based simulator. By converting RAFT flow visualizations into gap masks through edge detection, morphological closing, and contour selection, we obtain a reliable estimate of the window centroid even when foreground and background appearances are similar. A simple image-based visual servoing law maps centroid deviations to incremental NED motions, which are executed by a dynamic trajectory-tracking controller. Experiments in Blender and VizFlyt show that the method can robustly center the quadrotor relative to the window and drive it through the opening under a range of conditions. The approach highlights the utility of dense optical flow and neural rendering for designing and testing perception-driven navigation strategies prior to real-world deployment.

VI. Team Contributions

The work was carried out collaboratively by two team members:

- **Brijan** designed the overall perception pipeline, integrated RAFT into the simulation framework, implemented the Canny morphology contour-based gap extraction.
- **Shreyas** integrated the perception module with the VizFlyt simulator and SplatRenderer and executed the closed-loop visual servoing experiments in simulation,and conducted the Blender experiments and IOU analysis for the synthetic window images.

Acknowledgment

The authors thank the instructor, teaching assistant and dataset providers.

References

[1] RBE595 Project 4, "Documentation," *RBE595 Course Website*, Fall 2025. [Online]. Available: https://rbe549.github.io/rbe595/fall2025/proj/p4/

[2] Z. Teed and J. Deng, "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow," in *Proc. Eur. Conf. Computer Vision (ECCV)*, 2020, pp. 402–419.

[3] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 1–14, Jul. 2023.

[4] F. Chaumette and S. Hutchinson, "Visual Servo Control. I. Basic Approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, Dec. 2006.