IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

ASMA: An Adaptive Safety Margin Algorithm for Vision-Language Drone Navigation via Scene-Aware Control Barrier Functions

Sourav Sanyal, *Graduate Student Member, IEEE*, and Kaushik Roy, *Fellow, IEEE*

***Abstract*—In the rapidly evolving field of vision–language navigation (VLN), ensuring safety for physical agents remains an open challenge. For a human-in-the-loop language-operated drone to navigate safely, it must understand natural language commands, perceive the environment, and simultaneously avoid hazards in real time. Control Barrier Functions (CBFs) are formal methods that enforce safe operating conditions. Model Predictive Control (MPC) is an optimization framework that plans a sequence of future actions over a prediction horizon, ensuring smooth trajectory tracking while obeying constraints. In this work, we consider a VLN-operated drone platform and enhance its safety by formulating a novel scene-aware CBF that leverages ego-centric observations from a camera which has both Red-Green-Blue as well as Depth (RGB-D) channels. A CBF-less baseline system uses a Vision–Language Encoder with cross–modal attention to convert commands into an ordered sequence of landmarks. An object detection model identifies and verifies these landmarks in the captured images to generate a planned path. To further enhance safety, an Adaptive Safety Margin Algorithm (ASMA) is proposed. ASMA tracks moving objects and performs scene-aware CBF evaluation on-the-fly, which serves as an additional constraint within the MPC framework. By continuously identifying potentially risky observations, the system performs prediction in real time about unsafe conditions and proactively adjusts its control actions to maintain safe navigation throughout the trajectory. Deployed on a Parrot Bebop2 quadrotor in the Gazebo environment using the Robot Operating System (ROS), ASMA achieves 64%–67% increase in success rates with only a slight increase (1.4%–5.8%) in trajectory lengths compared to the baseline CBF-less VLN.**

***Index Terms*—Foundational models, vision-language navigation, scene-understanding, control barrier functions, safety-aware control.**

I. Introduction

FOUNDATIONAL models pretrained on exa-scale internet data have made significant strides in vision and language processing tasks with little to no fine-tuning, as exemplified by a new family of AI models such as BERT [1], GPT-3 [2], GPT-4 [3], CLIP [4], DALL-E [5], and PALM-E [6]. The fusion of vision and language models [4], [6] has enabled machines to interact with operating environments in increasingly intuitive ways. As VLN models become more widespread, the once sci-fi dream of robots understanding and interacting in complex environments through natural language commands is now a reality. This has been enabled by the emerging field of vision-language navigation (VLN) [7], [8], [9], [10], [11], [12]. Autonomous drones, pivotal in smart agriculture, search and rescue, and firefighting [13], are set to contribute up to $54.6 billion to the global economy by 2030 [14]. Imagine a scenario where VLN powered drones translate human-specified contextual instructions into actions. However, for VLNs, navigating dynamic environments using robot vision remains an open research problem. To that effect, Control Barrier Functions (CBFs) [15], [16] provide a formal mathematical framework for enforcing safe operating conditions in dynamical systems, making them useful for real-time applications where safety is crucial. On the other hand, Model Predictive Control (MPC) is an optimization framework that plans a sequence of future actions over a prediction horizon to ensure smooth trajectory tracking while obeying constraints.

We address the challenge of safe VLN for drone navigation by introducing ASMA—an Adaptive Safety Margin Algorithm that combines high-level vision-language reasoning with low-level safety control. Starting from a baseline system with a cross-modal Vision–Language Encoder [17] and YOLOv5-based object detection [18], ASMA translates language commands into ordered landmarks. To handle dynamic hazards, we propose a scene-aware CBF that uses real-time RGB–D observations for safety-aware control. ASMA adjusts drone control by tracking moving objects, predicting safety risks, and applying scene-aware CBFs within an MPC framework. It proactively responds to unsafe conditions in real time to ensure safe navigation throughout the trajectory. The proposed integration of VLNs with CBFs provides a formal safety layer which enhances VLN reliability of physical agents (in this work a drone).

Our main contributions are:

- A vision-language encoder with attention and detection for instruction-driven planning (Section III-A).

Received 10 March 2025; accepted 8 July 2025. Date of publication 23 July 2025; date of current version 6 August 2025. This article was recommended for publication by Associate Editor G. Shi and Editor G. Loianno upon evaluation of the reviewers' comments. This work was supported by the Center for the Co-Design of Cognitive Systems (CoCoSys), a center in JUMP 2.0, an SRC program sponsored by DARPA. *(Corresponding author: Sourav Sanyal.)*

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: sanyals@purdue.edu; kaushik@purdue.edu).

Demo code available at https://github.com/souravsanyal06/ASMA.

Digital Object Identifier 10.1109/LRA.2025.3592138

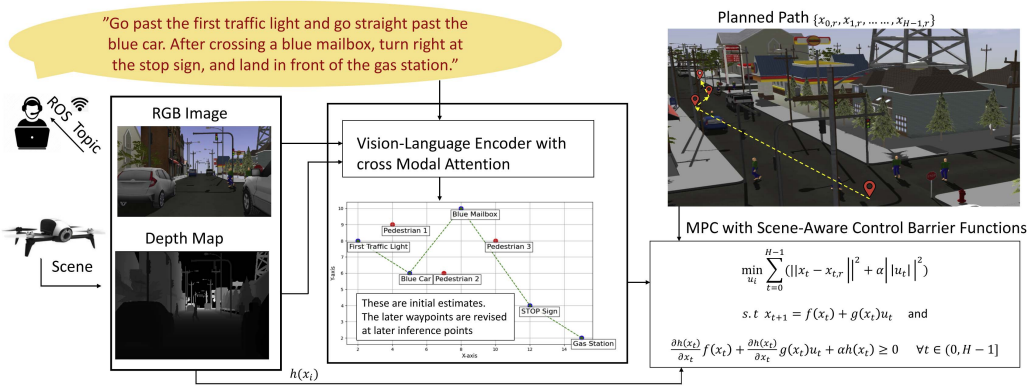

Fig. 1. Overview of the proposed ASMA framework. The system takes natural language instructions and processes RGB and depth data to create a 2D map with language-grounded landmarks and dynamic obstacles. A planned path is generated, and MPC with Scene-Aware Control Barrier Functions (CBFs) ensures safe navigation along the trajectory.

- A scene-aware CBF for safe navigation in dynamic environments (Section III-B).
- A full-stack modular system integrated in ROS for real-time navigation (Section III-C).
- Extensive evaluation showing improved safety and accuracy over a CBF-less VLN baseline in ROS-Gazebo on a Parrot Bebop2 quadrotor (Section IV).

In simulation, ASMA achieves 64%–67% increase in success rates with only a slight (1.4%–5.8%) increase in trajectory lengths compared to the baseline CBF-less VLN.

II. Related Work

Vision-Language Models for Robot Navigation: In Vision-Language Navigation (VLN), agents interpret language commands to navigate through environments using visual cues [7], [8], [9], [10], [11], [12]. Previous works, such as [8], [9], have expanded VLN into continuous environments (VLN-CE). Works in [10], [11], [12] have explored VLN focusing on interpreting visually-grounded instructions and developing models like VLN BERT to improve navigation performance through entity-landmark pre-training techniques. Safe-VLN [19] employs 2D LiDAR for safer waypoint prediction in VLN-CE, however the work only considers static objects along the navigation path. The works in [20], [21] integrate pretrained visual-language features with navigation maps. The work in [22] utilizes action prompts for improved spatial navigation precision. Room2Room [23] enables teleoperated communication using augmented reality, and [24] introduces the 'Tryout' method to prevent collision-related navigational stalls. However, these approaches do not address the physical dynamics of robots, crucial for verifying safety. Our work focuses on a teleoperated drone similar to [25] with VLN capabilities, utilizing an RGB-D sensor and aims to enhance its safety and reliability in presence of dynamic objects (moving arbitrarily) and complex structures.

Control Barrier Functions for Safety: Control barrier functions (CBFs) are essential tools from robust control theory, ensuring safety constraints are maintained in dynamic systems [15], [16]. By formally defining safe boundaries, CBFs dynamically adjust control actions to prevent safety violations. Vision-based control barrier functions (V-CBFs) [26] extend these protocols using conditional generative adversarial networks (C-GANs) to infer geometric safety constraints directly from visual observations. Neural formulations such as BarrierNet [27] integrate differentiable control barrier functions into end-to-end learning pipelines while maintaining formal safety guarantees. Other related works include [28], which develops a low-cost method for synthesizing quadratic CBFs over point cloud data, and [29], which addresses landing safety for aerial robots. However, none of these methods incorporate language-conditioned goals or semantic scene understanding. In contrast, our method dynamically synthesizes scene-aware CBFs grounded in both semantic visual understanding and language-driven intent, enabling reactive and context-aware safety enforcement in the presence of dynamic obstacles and complex structures. This allows the robot to proactively adapt its trajectory even without operator's explicit intervention, based on high-level task understanding and visual semantics, which existing visual or neural CBF methods do not support.

Although there exists several datasets and benchmarks for VLN, they do not take into account environmental changes which require online re-planning. Moreover, these datasets do not evaluate the interaction between language-driven intent and dynamic safety requirements, which our work explicitly addresses. Furthermore, the main aim in this work is not to compete with them, but to investigate the safety aspect of VLNs by synthesizing scene-aware CBFs on the fly. To that effect, this work considers two environments built from scratch using Gazebo and is tested on a Parrot Bebop2 quadrotor by considering the necessary robot-environment dynamics required for simulating and testing the safety and reliability challenges for agentic VLNs in a physical world.

III. Proposed Approach

We deploy a Parrot Bebop 2 quadrotor in a ROS-Gazebo environment with an RGB-D sensor, following [30], [31], [32]. We introduce ASMA—an Adaptive Safety Margin Algorithm for drone VLN (Fig. 1). The system processes language instructions and RGB-D input via a Vision-Language Encoder with cross-modal attention to build a language-grounded 2D map of landmarks and obstacles. A planned path is computed, and

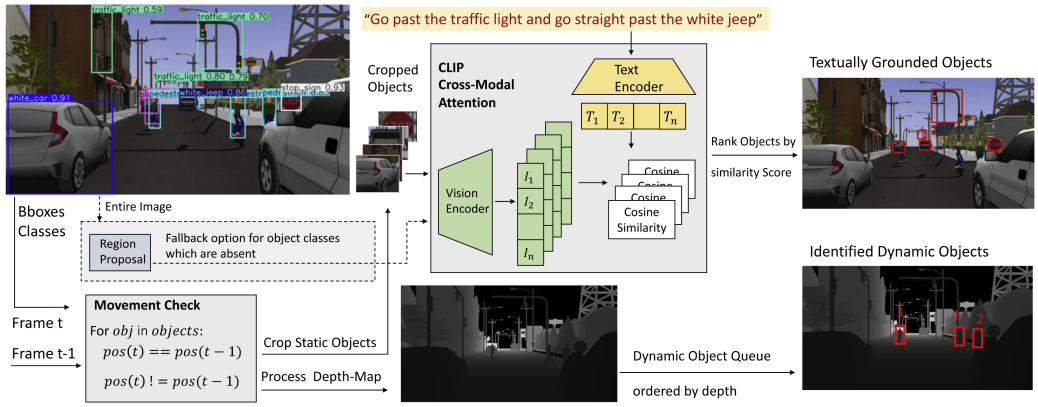

Fig. 2. Illustration of the multi-modal cross-attention pipeline for text-conditioned navigation. The system first detects objects using YOLOv5, followed by a movement check to classify static and dynamic objects across frames. Static objects are cropped and fed into a CLIP-based cross-modal attention module, which ranks objects based on textual relevance. For dynamic objects, a depth map is used to sort them by distance, enabling downstream obstacle-aware planning. A fallback mechanism is included to handle objects not recognized by YOLO, using region proposals and CLIP-based zero-shot recognition.

a Scene-Aware CBF-based MPC policy ensures safe, adaptive navigation.

A. Vision-Language Encoder With Cross-Modal Attention

Fig. 2 shows our pipeline for grounding language in visual objects. YOLOv5 detects objects, and CLIP matches cropped regions to instruction-referenced landmarks via a shared embedding space. Before processing, we distinguish dynamic from static objects by comparing bounding box centers over time. Let $\text{pos}_i(t)$ be the center of object i at time t. If

$$\text{pos}_i(t) = \text{pos}_i(t-1), \tag{1}$$

the object is static; otherwise,

$$\text{pos}_i(t) \neq \text{pos}_i(t-1), \tag{2}$$

it is dynamic and prioritized for collision avoidance. Static objects are matched via CLIP, while dynamic ones are handled through depth maps. The depth map also acts as instance segmentation, aiding avoidance of large structures like walls and trees for broader scene awareness.

Given a natural language instruction (e.g., *"Go past the first traffic light and go straight past the blue car."*), we extract an ordered list of landmark tokens such as `traffic light` and `blue car`. The onboard RGB image $\mathbf{I}$ is passed to YOLOv5, which outputs bounding boxes:

$$\mathbf{bbox}_i = \left(x_1^{(i)}, y_1^{(i)}, x_2^{(i)}, y_2^{(i)}, l_i\right), \quad i = 1, \ldots, N. \tag{3}$$

We crop each object from the image:

$$\mathbf{I}_i^{\text{crop}} = \text{CropImage}\left(\mathbf{I}, \mathbf{bbox}_i\right). \tag{4}$$

Each landmark phrase $\mathbf{T}_\ell$ is encoded by CLIP's text encoder:

$$\mathbf{z}_{\text{text}}^{\ell} = \Phi_{\text{text}}(\mathbf{T}_\ell), \tag{5}$$

and each object crop is encoded via CLIP's image encoder:

$$\mathbf{z}_{\text{obj}}^{i} = \Phi_{\text{img}}(\mathbf{I}_i^{\text{crop}}). \tag{6}$$

Cosine similarity is computed as:

$$S_{\ell,i} = \frac{\mathbf{z}_{\text{obj}}^{i} \cdot \mathbf{z}_{\text{text}}^{\ell}}{\|\mathbf{z}_{\text{obj}}^{i}\| \, \|\mathbf{z}_{\text{text}}^{\ell}\|}. \tag{7}$$

The bounding box $\mathbf{bbox}_i$ with the highest score is selected if $S_{\ell,i} > \theta$, where θ is a confidence threshold. If no match exceeds θ, we fall back to a lightweight region proposal strategy that samples a small number $(10 - 15)$ of image patches for CLIP-based matching. This fallback is rarely triggered in practice, as modern detectors trained on open-vocabulary datasets cover a broad range of objects. If no match is found, the landmark is flagged as unresolved (see Section V).

Path Generation with Landmark Order: Bounding-box centers $\mathbf{p}_\ell$ for each landmark ℓ are arranged in the sequence specified by the user:

$$\mathbf{p}_1 \rightarrow \mathbf{p}_2 \rightarrow \ldots \rightarrow \mathbf{p}_L.$$

A path planner connects these waypoints into a trajectory. As the drone moves, the list is dynamically updated as new landmarks become visible. By preserving instruction order and using CLIP similarity, the system ensures sequential, scene-aware navigation with safety guarantees.

B. Enhancing VLN With Formal Safety Methods

Control Barrier Functions (CBFs) are essential tools in safety-critical control systems that enforce safety constraints through mathematical functions. In this work, we integrate CBFs within a Model Predictive Control (MPC) framework, ensuring that safety constraints are proactively enforced while the drone follows its planned trajectory.

1) Preliminaries: We consider a quadrotor described by the following non-linear control affine dynamics with an ego-centric depth-map:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}, \quad \boldsymbol{\xi} = \Psi(\boldsymbol{x}, \boldsymbol{d}) \tag{8}$$

Here, $\dot{\boldsymbol{x}}$ denotes the time derivative of the state vector $\boldsymbol{x} \in \mathbb{R}^{12}$, covering the drone's positions, orientations, and velocities. The control input $\boldsymbol{u} \in \mathbb{R}^3$ represents the commanded linear velocity

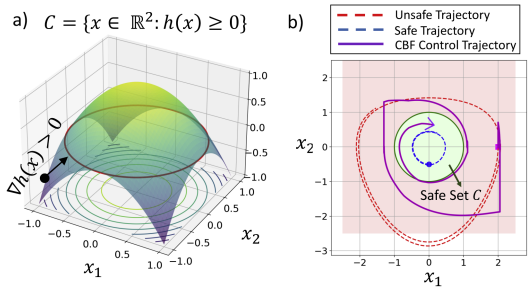

Fig. 3. Toy Illustration of Control Barrier Functions. (a) 3D view of the safe set $\mathcal{C}$ where $h(\boldsymbol{x}) \geq 0$ (above red ring). (b) Comparison of trajectories: unsafe (red dashed), safe (blue dashed), and CBF-controlled (magenta solid). The CBF-controlled trajectory starts outside the safe set and is driven into the safe region by the adaptive controller, demonstrating the forward-invariance property once inside.

of the quadrotor. This velocity-level abstraction assumes that the onboard flight controller is responsible for low-level stabilization using thrust and attitude control. This allows us to define first-order CBFs (later in Section III-B2) with respect to the velocity-controlled dynamics. Functions $f(\boldsymbol{x})$ form the autonomous dynamics and $g(\boldsymbol{x})$ signifies the dynamics that can be controlled in an affine manner. Additionally, the depth-map $\boldsymbol{d}$ provides obstacle distances, with $\boldsymbol{\xi} = \Psi(\boldsymbol{x}, \boldsymbol{d})$ transforming these distances from pixel to physical space, incorporating the drone's current position. Let $\mathcal{C} \subset \mathbb{R}^{12}$ represent a safety set defined through a continuously differentiable function $h(\boldsymbol{x})$ such that:

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^{12} : h(\boldsymbol{x}) \geq 0\} \tag{9}$$

Fig. 3(a) illustrates a toy safety set $\mathcal{C}$, marked by the boundary where $h(\boldsymbol{x}) = 0$ and the region where $h(\boldsymbol{x}) > 0$, indicating safe operational zones. The function $h(\boldsymbol{x})$ is characterized by its lie derivatives:

$$L_f h(\boldsymbol{x}) = \nabla h(\boldsymbol{x}) \cdot f(\boldsymbol{x}), \tag{10a}$$

$$L_g h(\boldsymbol{x}) = \nabla h(\boldsymbol{x}) \cdot g(\boldsymbol{x}), \tag{10b}$$

which are critical for monitoring the system's safety relative to state changes and control action changes.

Theorem (Safety Verification): For safety verification, it is required that:

$$L_f h(\boldsymbol{x}) + L_g h(\boldsymbol{x})\boldsymbol{u} + \alpha(h(\boldsymbol{x})) \geq 0, \tag{11}$$

where α is a class $\mathcal{K}$ function (meaning $\alpha(0) = 0$ and $\alpha(\mathcal{K}x_2) > \alpha(\mathcal{K}x_1)\, \forall\, x_2 > x_1$ and $\mathcal{K} > 0$). $\nabla h(\boldsymbol{x}) > 0$ in the unsafe region ($h(\boldsymbol{x}) < 0$) will drive $h(\boldsymbol{x})$ to become positive again.

Corollary (CBF-MPC Integration): If a function $h(\boldsymbol{x})$ can be designed such that adjustments in $\boldsymbol{u}$ satisfy the constraint in (11) at every time step, then embedding these constraints inside an MPC framework ensures that the system guarantees safety while optimizing trajectory tracking.

This forms the core of our Scene-Aware CBF-MPC framework (Algorithm 1), where the safety constraint is included as part of the trajectory optimization problem, rather than being enforced in a purely reactive manner.

$$\min_{\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{H-1}} J = \sum_{t=0}^{H-1} \Big\{ \|\boldsymbol{x}_t - \boldsymbol{x}_{r,t}\|^2 + \alpha \|\boldsymbol{u}_t\|^2 \Big\}, \tag{12}$$

Algorithm 1: Scene-Aware CBF-MPC.

function SA-CBF-MPC$\mathbf{p}_{\mathbf{curr}}, \mathbf{X}_{\text{plan}}, h(\boldsymbol{x})$, Obstacles
  Formulate MPC optimization problem over horizon H
  for each time step **do** $t = 0$ **to** $H - 1$
    Extract reference state $\mathbf{x}_{r,t}$ from $\mathbf{X}_{\text{plan}}$
    for each obstacle in Obstacles **do**
      Compute $h(\boldsymbol{x})$ using Equation (21)
      if $h(\boldsymbol{x}) < 0$ **then**
        Apply CBF constraints to enforce safety
      end if
    end for
  end for
  Solve MPC optimization incorporating CBF constraints
  returnoptimal control sequence $\boldsymbol{U} = [\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{H-1}]$
end Function

subject to:

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + g(\boldsymbol{x}_t)\boldsymbol{u}_t, \quad \forall t \in (0, H-1] \tag{13}$$

$$L_f h(\boldsymbol{x}_t) + L_g h(\boldsymbol{x}_t)\boldsymbol{u}_t + \alpha h(\boldsymbol{x}_t) \geq 0, \quad \forall t \in (0, H-1] \tag{14}$$

$$\implies \underbrace{\frac{\partial h(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} f(\boldsymbol{x}_t) + \frac{\partial h(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} g(\boldsymbol{x}_t)\boldsymbol{u}_t}_{\text{Expanded form of Lie derivatives}}$$

$$+\alpha h(\boldsymbol{x}_t) \geq 0, \quad \forall t \in (0, H-1] \tag{15}$$

where $\boldsymbol{x}_{r,t}$ denotes the reference trajectory computed from waypoints extracted by the vision-language encoder.

To integrate vision-language and depth outputs into the control loop, we convert pixel-level detections to 3D global positions. For each pixel (i, j) in the cropped depth map with depth d, we use intrinsic parameters—focal length f and camera center (c_x, c_y)—to compute 3D coordinates in the camera frame:

$$X_c = (i - c_x) \cdot \frac{d}{f}, \quad Y_c = (j - c_y) \cdot \frac{d}{f}, \quad Z_c = d. \tag{16}$$

These coordinates are then transformed into the global frame:

$$\begin{bmatrix} X_g \\ Y_g \\ Z_g \end{bmatrix} = \mathbf{R} \cdot \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} + \vec{\mathbf{p}}_{\text{curr}}, \tag{17}$$

Here, $\mathbf{R}$ is the rotation matrix for the drone's heading, and $\vec{\mathbf{p}}_{\text{curr}}$ is its estimated global position. Both are updated in real time using an EKF [33] to reduce sensor drift. The vision-language module detects targets and obstacles via bounding boxes; for each box center (i, j), we extract depth d and compute global coordinates $\vec{\mathbf{p}}_{\text{target}}$ or $\vec{\mathbf{p}}_{\text{obs}}$. Let

$$\vec{\mathbf{d}}_{\text{target}} = \vec{\mathbf{p}}_{\text{target}} - \vec{\mathbf{p}}_{\text{curr}}, \quad \vec{\mathbf{d}}_{\text{obs}} = \vec{\mathbf{p}}_{\text{obs}} - \vec{\mathbf{p}}_{\text{curr}}. \tag{18}$$

Obstacles are prioritized by

$$\text{Priority} = \frac{1}{\|\vec{\mathbf{d}}_{\text{obs}}\|}, \tag{19}$$

so nearer obstacles are processed first. If a high-priority obstacle lies along the path, we compute

$$\sigma_d = \text{sign}\left(\left(\vec{\mathbf{d}}_{\text{target}} \times \vec{\mathbf{d}}_{\text{obs}}\right)_z\right), \tag{20}$$

which determines whether the obstacle lies to the left ($\sigma_d < 0$) or right ($\sigma_d > 0$) of the target direction. This is used in (23a) to steer appropriately and avoid the obstacle.

2) Scene-Aware CBF: Since we operate at the velocity-control level, where control inputs correspond to desired velocities (Twist messages), we formulate first-order CBFs with respect to this abstraction. The proposed Scene-Aware CBF which is used within the MPC framework is defined as:

$$h(\boldsymbol{x}) = \begin{bmatrix} \vec{\mathbf{d}}_{\text{obs}} - \mathbf{d}_{\text{safe}} \\ \theta - \theta_{\text{safe}} \end{bmatrix} \tag{21}$$

$$\theta = \cos^{-1}\left(\frac{\vec{\mathbf{d}}_{\text{target}} \cdot \vec{\mathbf{d}}_{\text{obs}}}{\|\vec{\mathbf{d}}_{\text{target}}\| \, \|\vec{\mathbf{d}}_{\text{obs}}\|}\right), \tag{22}$$

where θ is the angle between $\vec{\mathbf{d}}_{\text{obs}}$ and $\vec{\mathbf{d}}_{\text{target}}$. The gradient for each component of $h(\boldsymbol{x})$is given by:

$$\frac{\partial h_1(\boldsymbol{x})}{\partial \boldsymbol{x}} = \sigma_d \frac{\partial \vec{\mathbf{d}}_{\text{obs}}}{\partial \boldsymbol{x}}, \tag{23a}$$

$$\frac{\partial h_2(\boldsymbol{x})}{\partial \boldsymbol{x}} = -\csc\theta \frac{\partial \cos\theta}{\partial \boldsymbol{x}}. \tag{23b}$$

These gradients feed into the Lie derivatives in (11), allowing the optimizer to adjust control inputs for safety. When multiple obstacles are detected, a priority queue processes them by increasing $\|\vec{\mathbf{d}}_{\text{obs}}\|$, ensuring the closest ones are handled first.

C. Adaptive Safety Margin Algorithm (ASMA)

Algorithm 2 gives a walkthrough of the entire system. Each iteration begins by tokenizing the VLN instruction (line 3). Thread A locks the image buffer (line 5), detects objects (line 6), and crops RGB and depth data (lines 7–8). Meanwhile, Thread B computes similarity scores between the cropped objects and instruction tokens (lines 12–15). A planning module then generates a trajectory using these detections (line 17), invoking a fallback if landmarks are missing (lines 18–20). Finally, the planned trajectory is passed to the SA-CBF-MPC module (line 21), which enforces safety and publishes the control command (line 22). This process repeats continuously.

Algorithm 2: Adaptive Safety Margin Algorithm (ASMA).

Input: RGB-D image $\mathbf{I}$, VLN instruction $\mathbf{cmd}$
Output: Drone command vector $\mathbf{u}$ with safety constraints

1: Initialize $\mathbf{global\_image}$, $\mathbf{global\_depth}$ from RGB-D
2: **while** not rospy.is_shutdown() **do**
3: $\mathbf{L} \leftarrow \mathbf{ExtractLandmarks}(\mathbf{cmd})$ ▷ Tokenize instruction
4: **Start Thread A**:
5: Acquire $\mathbf{image\_lock}$ ▷ Wait for $\mathbf{I}$
6: $\mathbf{bbox} \leftarrow \mathbf{DetectRelevantObjects}(\mathbf{global\_img}, \mathbf{L})$
7: $\mathbf{I_{crop}} \leftarrow \mathbf{CropObjects}(\mathbf{global\_img}, \mathbf{bbox})$
8: $\mathbf{D_{crop}} \leftarrow \mathbf{CropDepth}(\mathbf{global\_depth}, \mathbf{bbox})$
9: Release $\mathbf{image\_lock}$
10: **Start Thread B**:
11: Acquire $\mathbf{image\_lock}$ ▷ Wait for $\mathbf{I}$ and cropped objects
12: $\mathbf{E}_{\text{objs}} \leftarrow \mathbf{\Phi_{img}}(\mathbf{I_{crop}})$
13: $\mathbf{E}_{\text{text}} \leftarrow \mathbf{\Phi_{text}}(\mathbf{L})$
14: $\mathbf{S} \leftarrow \mathbf{ComputeSimilarity}(\mathbf{E}_{\text{objs}}, \mathbf{E}_{\text{text}})$
15: Release $\mathbf{image\_lock}$
16: $\mathbf{X}_{\text{plan}} \leftarrow \mathbf{GeneratePath}(\mathbf{L}, \mathbf{bbox}, \mathbf{D_{crop}})$
17: **if** any $\mathbf{L}$ not in detected object classes **then**
18: $\mathbf{bbox}_{\text{fallback}} \leftarrow \mathbf{RegionProposal}(\mathbf{global\_img})$
19: $\mathbf{X}_{\text{plan}} \leftarrow \mathbf{UpdatePath}(\mathbf{X}_{\text{plan}}, \mathbf{bbox}_{\text{fallback}})$
20: **end if**
21: $\boldsymbol{u} \leftarrow \mathbf{SA-CBF-MPC}(\vec{\mathbf{p}}_{\text{curr}}, \mathbf{X}_{\text{plan}}, h(\boldsymbol{x}), \{\vec{\mathbf{p}}_{\text{obs}}\})$
22: PUBLISH_CONTROL($\boldsymbol{u}$)
23: **end while**

IV. Results

A. Methodology

We implemented the ASMA framework in ROS on a parrot bebop2 quadrotor within the Gazebo environment. The pretrained CLIP model was obtained from OpenAI's repository [17]. Object detection was performed using YOLOv5 [18] (∼21 million parameters), with training data annotated via LabelImg [34]. The pretrained CLIP consisted of ∼149 million parameters. Because of the large model sizes, we performed thread synchronization (Algorithm 2) to toggle between the two inference modes. The threshold θ in Vision-Language Encoder was set to 0.2. Scene-Aware CBF with MPC optimizations sampled at 5Hz (to give sufficient time to the Vision-Language Encoder) were conducted using cvxopt [35], and the RotorS simulator [36] was used to integrate lower level control. $\vec{\mathbf{d}}_{\mathbf{safe}}$ was set to 2 meters and $\theta_{\mathbf{safe}}$ to 30 degrees. RGB-D focal length f was 10 meters.

B. Comparative Schemes

- *CBF-less VLN:* A baseline vision-language navigation (VLN) method that uses CLIP-YOLO for grounding instructions and a basic flight controller to execute trajectories, but does not enforce safety using control barrier functions (CBFs). This captures how standard VLN systems operate without dynamic scene-aware safety.
- *ASMA-Reactive:* Implements scene-aware CBFs based on the constraint in (11), enforcing safety reactively. The nominal control here is a basic PID. Forward invariance is restored upon violation by driving the system back into the safe set.
- *ASMA-MPC:* Integrates CBF constraints within an MPC framework for pro-active safety enforcement. Forward invariance is proactively maintained by forecasting safety

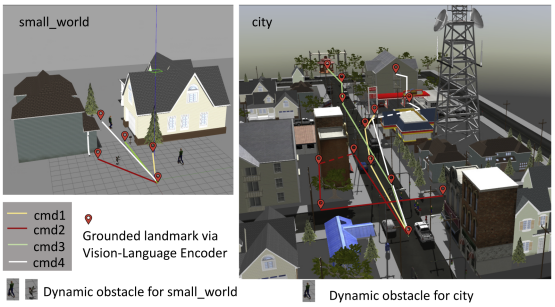

Fig. 4. Gazebo environments – **left:***small_world*, **right:** *city*.

TABLE I
VISION–LANGUAGE ENCODER PERFORMANCE

Environment	Average Similarity Score	Grounding Accuracy (%)
small_world	0.36	98
city	0.31	92

violations over the prediction horizon and adjusting the trajectory accordingly.

- *CBF-Only Navigation:* A language-agnostic baseline that uses CBFs for collision avoidance, assuming perfect knowledge of obstacle poses from simulation. It navigates toward a fixed target without any vision-language grounding, serving as a safety upper bound without perception or instruction-following.

We test our method in two simulated environments (Fig. 4): the *small world* for the CBF-based MPC formulation and the more complex and cluttered *city* environment for planning (Table II). In *city*, commands include flying through a narrow alley (**cmd2**), entering a gazebo and landing (**cmd3**), and ascending to a third-floor balcony to land (**cmd4**), all requiring spatial reasoning. Some instructions also contain logical conditionals—for example, **cmd2** includes "if a white truck is visible, land in front of it," which is grounded using vision-language attention. While the current examples do not include explicit temporal instructions (e.g., "wait for 5 seconds"), the framework supports such extensions via the same cross-modal grounding mechanism. See the additional media file for simulations. Performance is evaluated using Trajectory Length (TL), Success Rate (SR; trials reaching within 1 m of the target), and Navigation Error (NE; the final Euclidean distance from the target).

C. Vision–Language Encoder Performance

We evaluate our vision–language encoder by computing cosine similarity between CLIP-encoded text queries and cropped object embeddings. This includes both direct detections and fallback region proposals. Grounding accuracy is defined as the percentage of correctly matched landmarks. Table I reports results for the *small_world* and *city* environments. All similarity scores are obtained in a zero-shot setting, without fine-tuning, as the encoder combines outputs from a vision-language model and an object detector. While fine-tuning could improve scores, our goal is robust landmark extraction—fine-tuning may reduce generalization. This confirms that the encoder effectively grounds language to visual landmarks for downstream safety-aware control.

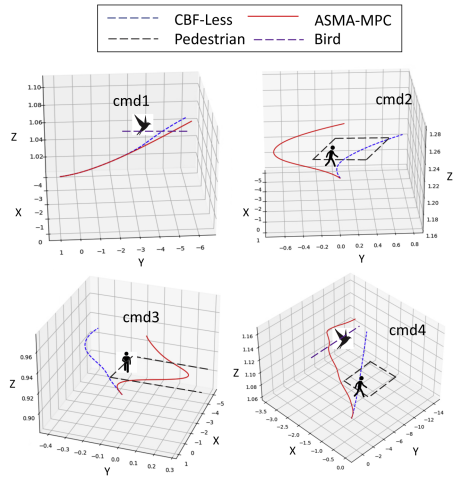

Fig. 5. Comparison of navigation trajectories for four VLN instructions in the **small world** environment. The blue dashed line represents the **CBF-less** method, while the red solid line corresponds to **ASMA-MPC**. Gray and purple dashed lines denote the trajectories of dynamic pedestrians and birds, respectively. Ghost icons indicate obstacle positions at the moment of closest encounter. ASMA-MPC demonstrates obstacle-aware local trajectory modulation in response to dynamic threats.

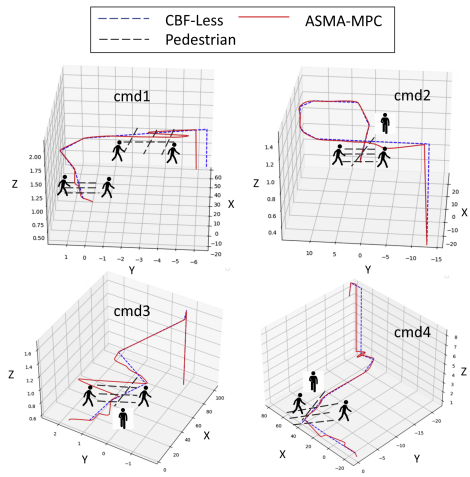

Fig. 6. Comparison of navigation trajectories for four VLN instructions in the **city** environment. The blue dashed line represents the **CBF-less** method, while the red solid line corresponds to **ASMA-MPC**. Black dashed curves represent pedestrian trajectories. Ghost pedestrian icons highlight positions at key avoidance moments. ASMA-MPC demonstrates smoother and safer trajectory adaptations in response to dense, dynamic pedestrian flows.

D. Vision-Language Navigation Performance

Table II compares four methods in *small_world* (top) and *city* (bottom). In *small_world*, CBF-less has low Success Rates (SR), while ASMA variants improve SR significantly (up to 96.02%). ASMA-Reactive sometimes increases Trajectory Length (TL), whereas ASMA-MPC achieves a better SR–TL tradeoff. CBF-Only improves over baseline but lacks language grounding. In the more complex *city*, ASMA-MPC anticipates hazards, reaching 95.42% SR and low Navigation Error (NE). Figs. 5 and 6 show that CBF-less (dashed blue) fails near obstacles, while

TABLE II
VLN PERFORMANCE IN TWO ENVIRONMENTS. ↓ INDICATES LOWER IS BETTER. ↑ IMPLIES HIGHER IS BETTER

Environment	VLN Instruction	Metric	CBF-Less	ASMA-Reactive	ASMA-MPC	CBF-Only
small_world	**cmd1**: Go to the house on the left.	TL ↓	8.54	8.60	8.40	8.52
		SR ↑	61.01	92.19	95.32	91.82
		NE ↓	2.80	3.11	2.40	2.40
	cmd2: Go to the tree on the right.	TL ↓	4.64	5.41	5.10	4.85
		SR ↑	55.59	90.08	93.80	92.49
		NE ↓	1.21	1.23	1.00	0.96
	cmd3: Find the mailbox, and fly towards it.	TL ↓	5.24	6.17	5.98	6.27
		SR ↑	55.32	91.18	94.21	91.47
		NE ↓	2.01	2.01	1.83	1.83
	cmd4: Fly between the two houses, look for a tree, and fly towards it.	TL ↓	14.51	14.70	13.90	18.82
		SR ↑	50.81	91.41	96.02	93.03
		NE ↓	0.51	0.50	0.35	0.36
city	**cmd1**: Go past the first traffic light and go straight past the blue car. After crossing a blue mailbox, turn right at the stop sign, and land in front of the gas station.	TL ↓	86.37	89.50	89.30	87.12
		SR ↑	58.32	88.90	94.45	90.10
		NE ↓	3.20	3.40	2.90	3.10
	cmd2: Follow the road. After the crossing, fly through the alley before the blue mailbox. Turn left, pass between buildings, turn left at the oak tree, and if a white truck is visible, land in front of it.	TL ↓	118.60	129.19	112.83	126.15
		SR ↑	52.25	87.30	93.21	89.99
		NE ↓	1.60	1.50	1.25	1.30
	cmd3: Head past the second traffic light. If an ambulance is on the left, fly past it and the stop sign. Enter the gazebo and land inside.	TL ↓	169.80	179.70	172.29	174.80
		SR ↑	54.10	89.50	94.12	90.55
		NE ↓	2.40	2.10	1.80	1.95
	cmd4: Fly past the first traffic light, then turn right before the gas station. Before the white truck, turn left. At the apartment with stairs, ascend to the third floor and land inside the hallway.	TL ↓	139.10	143.70	140.02	147.39
		SR ↑	48.92	89.60	95.42	91.80
		NE ↓	0.70	0.60	0.50	0.55

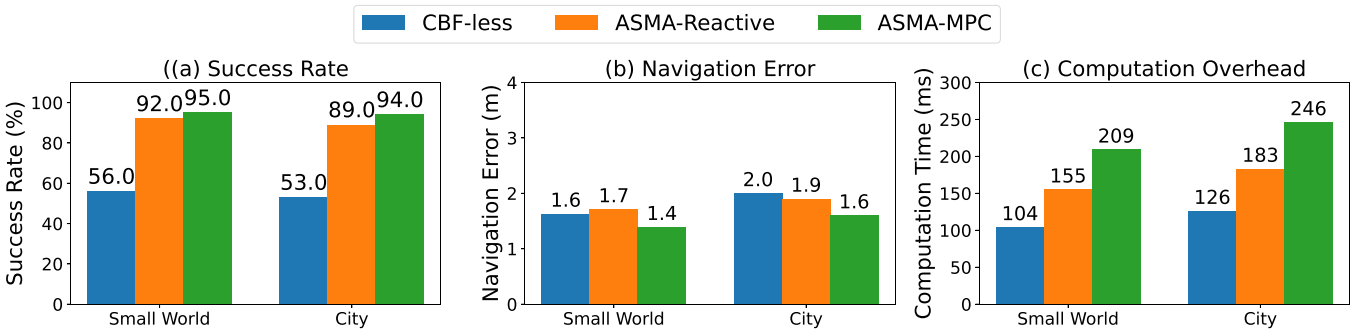

Fig. 7. Ablation study results comparing the three methods (CBF-less, ASMA-Reactive, and ASMA-MPC) across two environments (Small World and City). Bars indicate average Success Rate (SR), Navigation Error (NE), and Computation Time for 1 inference cycle.

ASMA-MPC (solid red) maintains safe, instruction-following paths. ASMA-Reactive and ASMA-MPC boost SR by 64.1% and 67.5%, with only 1.4%–5.8% TL increases from safety detours.

E. Ablation Study

Our ablation study compares three variants—CBF-less, ASMA-Reactive, and ASMA-MPC—across both *small world* and *city* environments. As shown in Fig. 7, CBF-less yields lower success rates, while ASMA variants significantly improve performance. ASMA-MPC achieves the highest success rate and lowest navigation error by proactively enforcing safety via model predictive planning, at the cost of increased computation. To support latency-sensitive deployments, ASMA-Reactive offers a lower-latency alternative that enforces safety reactively. As seen in Fig. 7(c), ASMA-MPC incurs higher latency but delivers better safety and task performance. Dynamic/static object classification is performed via lightweight bounding box comparisons and has negligible runtime cost. ASMA-MPC increases processing time by 30–40%, which is justified by its gains. All variants meet real-time constraints on our evaluation setup (Intel Core i9, RTX 3090, 2 GB). Further optimization is needed for embedded deployment. This highlights a practical tradeoff: ASMA-MPC for higher compute budgets, ASMA-Reactive for time-critical settings.

V. DISCUSSION

The current framework advances state-of-the-art agentic navigation by integrating scene-aware Control Barrier Functions (CBFs) with CLIP-YOLO for language-conditioned visual navigation, enabling robust safety enforcement even in the presence of arbitrarily moving obstacles and complex structures, while effectively handling edge cases like occlusion or out-of-view objects. If a referred object is initially occluded or out of view, it is not detected immediately. However, the navigation policy proceeds using available spatial cues, and detection resumes once the object becomes visible. This enables implicit handling of partial observability, though the system does not explicitly model occlusion or object permanence. While not active in the current version, basic user feedback (e.g., "The object does not exist") can be incorporated using a lightweight decoder trained on detection confidence and instruction progress. Mechanisms

like visual memory, semantic mapping, or uncertainty-aware exploration may further improve handling of occlusions and ambiguity. ASMA supports extension to other perception modalities (e.g., segmentation, optical flow), making it a general framework for scene-aware CBFs in VLN safety. Real-world deployment with dynamic obstacles such as pedestrians or birds remains challenging, as aerial platforms raise safety and regulatory concerns which are better addressed in simulation.

VI. Summary

We discussed contemporary VLN methods do not address dynamic obstacles in evolving environments. We introduced ASMA (Adaptive Safety Margin Algorithm) to enhance VLN safety for drones using a novel scene-aware CBF formulation. By continuously identifying potentially risky observations, the system performs prediction in real time about unsafe conditions. ASMA dynamically adjusts control actions based on real-time depth data, ensuring safe navigation in complex environments. Compared to a baseline CBF-less VLN model, two variants of ASMA achieves 64%–67% increase in success rates with only a slight (1.4%–5.8%) increase in trajectory lengths compared to the baseline CBF-less VLN.

References

[1] J. Devlin, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[2] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.

[3] J. Achiam et al., "GPT-4 technical report," 2023, *arXiv:2303.08774*.

[4] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.

[5] A. Ramesh et al., "Zero-shot text-to-image generation," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 8821–8831.

[6] D. Driess et al., "PaLM-E: An embodied multimodal language model," 2023, *arXiv:2303.03378*.

[7] D. Shah et al., "LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Proc. 6th Conf. Robot Learn.* 2023, pp. 492–504.

[8] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee, "Beyond the NAV-Graph: Vision-and-language navigation in continuous environments," in *Proc. 16th Eur. Conf. Compu. Vis.*, Glasgow, U.K., 2020, pp. 104–120.

[9] Y. Hong, Z. Wang, Q. Wu, and S. Gould, "Bridging the gap between learning in discrete and continuous environments for vision-and-language navigation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 15439–15449.

[10] P. Anderson et al., "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3674–3683.

[11] Y. Hong, Q. Wu, Y. Qi, C. Rodriguez-Opazo, and S. Gould, "VLN BERT: A recurrent vision-and-language bert for navigation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1643–1653.

[12] Y. Cui, L. Xie, Y. Zhang, M. Zhang, Y. Yan, and E. Yin, "Grounded entity-landmark adaptive pre-training for vision-and-language navigation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 12043–12053.

[13] F. Giones and A. Brem, "From toys to tools: The co-evolution of technological and entrepreneurial developments in the drone industry," *Bus. Horiz.*, vol. 60, no. 6, pp. 875–884, 2017.

[14] DroneII, "Drone market report," 2024. [Online]. Available: https://droneii.com/product/drone-market-report?srsltid=AfmBOorqVivIWByTvlyTvfvt_ZNWT1ZJJ7N52-IK0j_4QoQ0cImOEU1

[15] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 3420–3431.

[16] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[17] OpenAI, "Clip: Connecting text and images," 2021. [Online]. Available: https://github.com/openai/CLIP

[18] Ultralytics, "YOLOV5: Object detection at 640x640," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[19] L. Yue, D. Zhou, L. Xie, F. Zhang, Y. Yan, and E. Yin, "Safe-VLN: Collision avoidance for vision-and-language navigation of autonomous robots operating in continuous environments," *IEEE Robot. Automat. Lett.*, vol. 9, no. 6, pp. 4918–4925, Jun. 2024.

[20] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 10608–10615.

[21] Z. Wang, X. Li, J. Yang, Y. Liu, and S. Jiang, "GridMM: Grid memory map for vision-and-language navigation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 15625–15636.

[22] B. Lin, Y. Zhu, Z. Chen, X. Liang, J. Liu, and X. Liang, "ADAPT: Vision-language navigation with modality-aligned action prompts," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 15396–15406.

[23] T. Pejsa, J. Kantor, H. Benko, E. Ofek, and A. Wilson, "Room2Room: Enabling life-size telepresence in a projected augmented reality environment," in *Proc. 19th ACM Conf. Comput.-Supported Cooperative Work Social Comput.*, 2016, pp. 1716–1725.

[24] D. An et al., "ETPNav: Evolving topological planning for vision-language navigation in continuous environments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 7, pp. 5130–5145, Jul. 2025.

[25] R. Ibrahimov, E. Tsykunov, V. Shirokun, A. Somov, and D. Tsetserukou, "DronePick: Object picking and delivery teleoperation with the drone controlled by a wearable tactile display," in *Proc. 28th IEEE Int. Conf. Robot Hum. Interactive Commun.*, 2019, pp. 1–6.

[26] H. Abdi, G. Raja, and R. Ghabcheloo, "Safe control using vision-based control barrier function (V-CBF)," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 782–788.

[27] W. Xiao et al., "BarrierNet: Differentiable control barrier functions for learning of safe robot control," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 2289–2307, Jun. 2023.

[28] M. De Sa, P. Kotaru, and K. Sreenath, "Point cloud-based control barrier function regression for safe and efficient vision-based control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 366–372.

[29] V. N. Sankaranarayanan et al., "A CBF-adaptive control architecture for visual navigation for UAV in the presence of uncertainties," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024.

[30] S. Sanyal and K. Roy, "RAMP-Net: A robust adaptive MPC for quadrotors via physics-informed neural network," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1019–1025.

[31] S. Sanyal, R. K. Manna, and K. Roy, "EV-planner: Energy-efficient robot navigation via event-based physics-guided neuromorphic planner," *IEEE Robot. Automat. Lett.*, vol. 9, no. 3, pp. 2080–2087, Mar. 2024.

[32] A. Joshi, S. Sanyal, and K. Roy, "Real-time neuromorphic navigation: Integrating event-based vision and physics-driven planning on a parrot Bebop2 quadrotor," 2024, *arXiv:2407.00931*.

[33] M. R. Fernandes, G. M. Magalhães, Y. R. C. Zúñiga, and J. B. R. do Val, "GNSS/MEMS-INS integration for drone navigation using EKF on lie groups," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 59, no. 6, pp. 7395–7408, Dec. 2023.

[34] Tzutalin, "Labelimg," 2020. [Online]. Available: https://github.com/tzutalin/labelImg

[35] M. S. Andersen, J. Dahl, and L. Vandenberghe, "CVXOPT: Python software for convex optimization," 2021. [Online]. Available: https://cvxopt.org

[36] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS—A modular Gazebo MAV simulator framework," in *Robot Operating System (ROS): The Complete Reference*, vol. 1. Cham, Switzerland: Springer, 2016, pp. 595–625.