Dynamic Control Barrier Function Regulation with Vision-Language Models for Safe, Adaptive, and Realtime Visual Navigation

Jeffrey Chen, Rohan Chandra

Supplementary Material, Videos at the following website: **saferobotnav.github.io/AlphaAdj**

***Abstract*— Robots operating in dynamic, unstructured environments must balance safety and efficiency under potentially limited sensing. While control barrier functions (CBFs) provide principled collision avoidance via safety filtering, their behavior is often governed by fixed parameters that can be overly conservative in benign scenes or overly permissive near hazards. We present AlphaAdj, a vision-to-control navigation framework that uses egocentric RGB input to adapt the conservativeness of a CBF safety filter in real time. A vision-language model (VLM) produces a bounded scalar risk estimate from the current camera view, which we map to dynamically update a CBF parameter that modulates how strongly safety constraints are enforced. To address asynchronous inference and non-trivial VLM latency in practice, we combine a geometric, speed-aware dynamic cap and a staleness-gated fusion policy with lightweight implementation choices that reduce end-to-end inference overhead. We evaluate AlphaAdj across multiple static and dynamic obstacle scenarios in a variety of environments, comparing against fixed-parameter and uncapped ablations. Results show that AlphaAdj maintains collision-free navigation while improving efficiency (in terms of path length and time to goal) by up to 18.5% relative to fixed settings and improving robustness and success rate relative to an uncapped baseline.**

I. Introduction

Autonomous robots increasingly operate in dynamic, crowded, and unstructured environments. Prior work [1], [2], [3] suggests that achieving human-like mobility [4] in cluttered environments often requires low-level controllers to rely on accurate state measurements of the surrounding scene, which is difficult to realize in practice, especially when the environment is dynamic. Most applications, however, would greatly benefit from navigation systems that produce human-like trajectories directly using input from onboard sensors such as cameras, without relying on expensive mapping and perception for exact state measurements [5]. In such scenarios, ensuring safety and responsiveness using only visual input, such as RGB, is a major challenge.

In this context, vision-language models (VLMs) offer promising advances in semantic perception, as these models can infer scene-level understanding directly from egocentric visual input [6]. Prior work has shown that vision can directly inform low-level navigation and safety, including CBF-based safety from point cloud geometry [7] and vision-based feedback control [8]. However, most existing pipelines still commit to a largely fixed safety behavior once deployed. This is limiting because the appropriate level of conservativeness around obstacles can change rapidly across scenes and over time. For example, a robot should be more assertive in an open hallway with no nearby hazards, maintaining higher speed and allowing tighter goal seeking, because the collision risk is slow. As the robot approaches a narrow doorway, the situation can change in an instant. A person may step into the doorway or cross the robot's path, increasing collision risk and requiring the robot to become more conservative by slowing down, increasing clearance, and prioritizing following of safe constraints. Once the passage clears, the robot should return to the more assertive settings to avoid unnecessary hesitation and achieve its goal more quickly. This risk-conditioned shift in required behavior motivates adaptive vision-based control.

Additionally, the massive sizes of VLMs and other foundation models mean that VLM-based visual navigation suffer from high latency for real-time robot navigation [9], [10]. This motivates adaptive, latency-aware vision-based control. In this work, we present an adaptive navigation algorithm that couples high-level visual semantic perception to adapt low-level safety behavior at runtime. Our framework extracts context from the egocentric RGB scene and outputs a collision-risk estimate using a VLM. The risk estimate is used to dynamically adjust a low level CBF control parameter which governs how conservatively (or assertively) the robot responds to potential hazards. This enables an adaptive and semantically aware navigation system that better matches the moment-to-moment demands of complex environments, enabling safer and more efficient navigation that fixed-policy vision-control pipelines.

A. Main Contributions

Our contributions are threefold:

- **Real-time adaptive mobility directly from RGB inputs:** We enable real-time adaptive navigation directly from egocentric RGB inputs by using a VLM-derived collision risk signal to modulate a low level control parameter that determines control policy conservativeness around obstacles. Our approach cautions safety when necessary near hazards while remaining agile in low-risk scenes, improving the safety–efficiency tradeoff without altering the underlying controller structure.
- **Latency-aware robustness for asynchronous VLM risk:** We show that asynchronous VLM inference latency can undermine safety when risk estimates are applied naively. To mitigate this, we introduce a latency-aware mechanism that detects stale or anomalous VLM updates and triggers fallback behavior. We also report practical implementation choices that reduced end-to-end VLM latency in our system.
- **Practical risk-to-α mapping and fusion policy:** We propose a geometric dynamic cap and a simple fusion rule that that (*i*) uses VLM-derived conservativeness when fresh, (*ii*) clamps overly permissive values using the cap, and (*iii*) defaults to conservative cap-driven behavior under staleness. This yields a deployable method for integrating semantic risk into safety filtering.

II. Related Work

A. Adaptive CBF Parameter Tuning

Control Barrier Functions (CBFs) are a powerful mathematical framework used to encode robot system safety in nonlinear systems [11], [12]. Adaptive CBF methods study how to adjust CBF parameters in order to further optimize robot performance while staying within these safe sets. Some formulations introduce adaptive CBFs that update safety certificates in response to uncertainty and changing operating conditions, improving practicality compared to static, hand-tuned choices [13]. However, rather than focus on overall system safety and performance, these adaptive CBF approaches focus more on addressing the uncertainty of the dynamic model [14], [15], [16]. Instead of adapting CBF parameters as such, we use egocentric RGB and a VLM to estimate a scalar risk and directly modulate a single parameter α that controls the CBF's conservativeness.

B. Vision-Based Navigation

There is increasing interest in using vision directly for control and safety. Frameworks like LivePoint [7] successfully integrate CBFs for decentralized multi-robot navigation using only LiDAR-derived point cloud inputs, while other RGB-geometry based approaches [8] and [17] couple 3D scene geometry with CBF-based filtering to activate safety constraints more proactively. Visual-servoing techniques use camera feedback directly to regulate robot motion [18], [19]. These approaches demonstrate that vision can inform low-level control, but typically rely on fixed safety behavior once deployed. There are further challenges that come with usage of large foundation models in mobile robots. Because GPUs cannot be equipped on such robots, the foundation models must be run remotely or on cloud-based systems [20]. Such reliance on external setups may degrade performance as communication latency between the robot and the remote system lead to delays [9]. This motivates latency-aware adaptive vision-based control, where visual understanding is used not only to set an initial safety policy, but to continuously adjust how the controller trades off safety and progress in real time.

III. Background

We formalize safety by discussing Control Barrier Functions (CBFs) [11], which are a mathematical framework used in control theory to ensure system safety while achieving desired control objectives. A CBF is a scalar function, $h(x)$ that is defined over the state space of a system. We define the safe set C as the set of all states for which $h(x) > 0$. Ensuring safety involves keeping the system state within C at all times; safety is guaranteed as long as the CBF is always non-negative. This is achieved by designing a control input, u, such that the following condition is satisfied:

$$\frac{d}{dt}h(x) \geq -\alpha(h(x)) \tag{1}$$

where $\frac{d}{dt}h(x)$ denotes the time derivative of $h(x)$ along system trajectories and α is an extended class $\mathcal{K}$ function, typically a linear or higher-order function that ensures the safety constraint is enforced with appropriate robustness. This inequality lower-bounds the rate of change of the barrier function $h(x)$ along system trajectories. When α is small, the right-hand side is less negative, which forces $\frac{d}{dt}h(x)$ to be larger (i.e., h must decrease more slowly or increase), leading to more conservative avoidance. When α is large, the bound becomes more negative, allowing $\frac{d}{dt}h(x)$ to take more negative values (i.e., permitting h to decrease faster), which corresponds to more aggressive motion that can approach obstacles more quickly before corrective action is required.

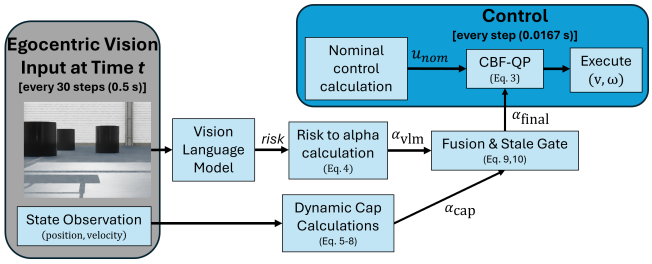


Fig. 1: System overview. Egocentric RGB frames are processed asynchronously by a VLM to produce a risk estimate r, mapped to α_{vlm}. A geometric, speed-aware cap α_{cap} and a staleness test gate the VLM output to produce α_{final}, which parameterizes the CBF safety filter that produces safe commands.

We seek safe yet efficient navigation by dynamically adjusting CBF behavior using risk information inferred from the VLM. In particular, we adapt the CBF α parameter, which controls the aggressiveness of the safety constraint enforcement: smaller α yields more conservative behavior near obstacles, while larger α permits more aggressive motion. We treat the CBF conservativeness $\alpha(t)$ as a time-varying scalar $\alpha(t) \in [\alpha_{\min}, \alpha_{\max}]$ calculated online based on the current RGB observation.

In our implementation, we use a per-obstacle CBF defined using a squared-distance safety margin. Let $p \in \mathbb{R}^2$ denote the robot position in the plane and let $c_j \in \mathbb{R}^2$ be the center of circular obstacle j with radius R_j. We model the robot with an effective radius R_R, a circular safety approximation of the robot body, and include an optional safety padding δ. Defining $R_{\text{sum},j} = R_R + R_j + \delta$, the CBF for obstacle j is thus:

$$h_j(x) = \|p - c_j\|^2 - R_{\text{sum},j}^2. \tag{2}$$

We first compute a nominal velocity u_{nom} that points the robot toward the goal, with its speed capped at the robot's maximum speed $v_{\max}$. We then compute a safe velocity u_{safe} by solving a small quadratic program that minimizes deviation from u_{nom} while enforcing the CBF constraint in Eq. 1. The QP is formulated as:

$$u_{\text{safe}} = \arg\min_{u \in \mathcal{U}} \frac{1}{2}\|u - u_{\text{nom}}\|^2 \tag{3}$$

$$\text{subject to CBF constraints holding and } \|u\| \leq v_{\max}.$$

This formulation preserves the goal-seeking behavior of u_{nom} as much as possible while guaranteeing collision avoidance through the CBF constraint.

IV. AlphaAdj: Approach

We present a latency-aware framework that uses egocentric RGB to adapt the conservativeness of a CBF safety filter in real time. At a high level, the robot runs a nominal goal-seeking controller, and a CBF filter enforces collision avoidance. A VLM periodically estimates collision risk from the current RGB frame, and this risk is mapped to the CBF parameter α. Because VLM inference is asynchronous and introduces non-trivial latency, we propose a dynamic cap and mitigation policy for stale requests that guarantees conservative behavior when the VLM output is delayed. The overall system overview can be seen in Figure 1.

A. Risk-to-α Mapping from VLM Output

We consider a mobile robot with egocentric RGB camera sensing operating in a cluttered environment with static or dynamic obstacles. The low-level controller executes every 0.0167 sconds (60 Hz). Every N = 30 control steps (0.5 seconds, 2 Hz), the controller captures an RGB frame and asynchronously queries our VLM risk service over HTTP. The VLM produces a scalar collision-risk estimate $r \in [0, 1]$ from the current egocentric RGB frame, where $r = 0$ indicates low risk and $r = 1$ indicates high risk. We interpret r as a hazard score for the robot's current motion context in relation to its environment (e.g., whether the next few seconds will require evasive action). We map risk to a bounded scalar α_{vlm} using the monotone decreasing function

$$\alpha_{\text{vlm}}(r) = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min})(1 - r)^{\gamma}, \tag{4}$$

where $\alpha_{\min} \le \alpha_{\text{vlm}} \le \alpha_{\max}$ and $\gamma > 0$ controls sensitivity. We devise the power-law mapping in Eq. 4 to satisfy four practical requirements for effective closed-loop control: (*i*) boundedness so that α_{vlm} stays within a stable range, (*ii*) monotonicity, to ensure higher predicted risk always yields lower α_{vlm} and hence more conservative behavior, (*iii*) smoothness, to avoid abrupt parameter changes under noisy outputs, (*iv*) and tunability using a single parameter γ.

a) VLM prompt construction.: Risk is obtained by querying a local HTTP endpoint with a PNG image and a text instruction. The instruction presented to the VLM is formed by concatenating (*i*) a fixed server-side formatting constraint that enforces a strict JSON response and (*ii*) a client-provided context string that is updated online with the robot's current state. The server prepends the following base instruction: "You are a risk estimator for mobile robot navigation. Given this scene image, output ONLY a single-line JSON of the form {"risk": <number between 0 and 1>. No prose, no markdown—just JSON.}" The controller then appends an additional context prompt containing lightweight runtime information: "You are a safety estimator for a mobile robot. Given the current RGB view from a camera mounted on the robot, estimate current collision risk as a number in [0,1]. Lower values meaning lower imminent collision risk, and higher values meaning imminent collision risk. Return strict JSON only with keys: risk (0..1 float). Context: $h_{\min}$ and current speed v m/s." where $h_{\min}$ is the minimum barrier value with respect to the closest obstacle and v is the current commanded speed estimate.

VLM inference is executed asynchronously to avoid blocking the control loop. Every 30 control steps, the current RGB frame is enqueued and sent to a separate VLM process over HTTP.

B. Dynamic α Cap from Geometric Margin

Because VLM outputs can be delayed and may occasionally be overly permissive, we impose a geometric dynamic cap on α so that the controller cannot behave too aggressively when the robot is close to obstacles. The cap is designed primarily as a safety envelope and fallback mechanism. At each time step, we define clearance margin m as the distance between our robot and the closest obstacle. We map clearance margin m to a cap by smoothly interpolating between a conservative near value α_{near} and a permissive far value α_{far}:

$$s = \min\left(1, \frac{m}{M_{\text{safe}}}\right), \tag{5}$$

$$\alpha_{\text{dist}} = \alpha_{\text{near}} + (\alpha_{\text{far}} - \alpha_{\text{near}})\, s^{\gamma_c}, \tag{6}$$

where $M_{\text{safe}} > 0$ is a clearance scale that defines when we consider the robot "safely far," and $\gamma_c = 2$ shapes the transition. Equation 5 defines a normalized clearance score s $s \in [0, 1]$ from raw margin m. We scale m by M_{safe} and clip at 1 so that (*i*) $s = 0$ at safety boundary, (*ii*) $s = 1$ once the robot is at least M_{safe} meters away from the closest obstacle, and larger clearances do not further increase the cap. Equation 6 then maps s to a distance-based cap α_{dist} by smoothly interpolating between a conservative near value α_{near} and a permissive far value α_{far}. Clearance alone is not sufficient: for the same clearance m, a faster robot should be more conservative. Let v denote the current forward speed estimate, and let $V_{\max} > 0$ be the maximum allowed forward speed. To α_{dist}, we apply a speed-dependent scaling factor

$$\eta(v) = \frac{1}{1 + G_v\left(\frac{\max(0,v)}{V_{\max}}\right)}, \tag{7}$$

where $G_v = 1$ controls how strongly speed reduces the cap. We use $\max(0,v)$ so that small or negative speeds do not increase conservativeness. The final dynamic cap is

$$\alpha_{\text{cap}} = \text{clip}(\alpha_{\text{dist}}\, \eta(v),\, \alpha_{\text{near}},\, \alpha_{\text{far}}), \tag{8}$$

where $\text{clip}(x, \ell, u) = \min\{u, \max\{\ell, x\}\}$ enforces bounds.

a) Fusion policy: We combine α_{vlm} with the cap using a simple fusion policy. We use the same cap function $\alpha_{\text{cap}}(m, v)$ throughout, but we instantiate it with two different near-obstacle floors to create two safety modes. When the VLM estimate is fresh, we use a soft cap that allows slightly larger α values near obstacles. This preserves responsiveness and avoids being unnecessarily conservative, while still preventing the VLM from choosing an overly permissive α. When the VLM estimate is stale, we use a more conservative hard cap that forces smaller α values near obstacles. This provides a guaranteed conservative fallback when semantic feedback is delayed or unavailable. Thus, when VLM estimate is fresh, we set

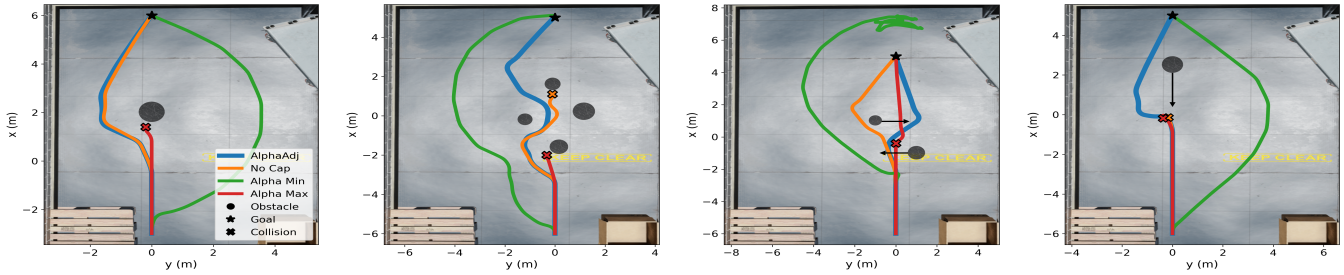
$$\alpha_{\text{final}} = \min\left(\alpha_{\text{vlm}},\, \alpha_{\text{cap}}^{\text{soft}}\right), \tag{9}$$

which ensures the VLM can only reduce conservativeness up to the safe envelope imposed by the soft cap. When the VLM estimate is stale, we ignore α_{vlm} and fall back to the hard cap,

$$\alpha_{\text{final}} = \alpha_{\text{cap}}^{\text{hard}}. \tag{10}$$

The overall control stack per timestep is:

1) If scheduled (once every 30 steps), query the VLM risk service asynchronously and map the returned risk to a candidate α_{vlm} (Section IV-A)
2) Read robot state x and compute geometric safety quantities such as clearance margin to closest obstacle and current speed estimate. (Section IV-B)
3) Compute geometric cap (α_{cap}), determine staleness, and fuse with the latest VLM-based α_{vlm} estimate to produce α_{final} (Eq. 8-10)
4) Compute a nominal goal-directed control u_{nom}, then apply a CBF safety filter, using α_{final}, to obtain a safe control u_{safe} (Eq. 3).
5) Convert u_{safe} into executable robot commands (v, ω) and step the simulator.



(a) Frontal obstacle. (b) Cluttered field. (c) Dynamic obstacle. (d) Dynamic frontal obstacle.

Fig. 2: Top-down trajectories in four warehouse scenarios.

	Frontal				Cluttered				Dynamic				Dynamic Frontal			
Metric	AlphaAdj	No cap	$\alpha_{\min}$	$\alpha_{\max}$	AlphaAdj	No cap	$\alpha_{\min}$	$\alpha_{\max}$	AlphaAdj	No cap	$\alpha_{\min}$	$\alpha_{\max}$	AlphaAdj	No cap	$\alpha_{\min}$	$\alpha_{\max}$
Success	Yes	Yes	Yes	No	Yes	No	Yes	No	Yes	Yes	No	Yes	Yes	No	Yes	No
Time-to-goal (s)	8.733	**8.583**	10.550	–	**11.717**	–	13.050	–	**10.267**	10.667	–	10.383	**10.850**	–	11.633	–
Min margin (m)	0.542	**0.522**	2.766	–	**0.246**	–	1.551	–	0.334	**0.225**	2.651	0.067	**0.114**	–	3.095	–
Path length (m)	9.967	**9.825**	12.192	–	**12.985**	–	14.982	–	11.689	12.165	24.180	**10.986**	**12.051**	–	13.651	–

TABLE I: Warehouse: Baseline comparisons across scenarios (AlphaAdj, no cap, fixed $\alpha_{\min}$, fixed $\alpha_{\max}$).

V. Experiments

We evaluate AlphaAdj in NVIDIA IsaacSim [21] using a Carter differential-drive mobile robot with an egocentric RGB camera. Reaching the goal constitutes a successful run, while a collision or simulation stop constitutes a failed run. We evaluate on four different scenarios, and consider two environments: a generated toy environment with a flat ground plane and a realistic warehouse environment. We conduct comparisons with the following:

1) AlphaAdj *with no dynamic cap:* Ablation baseline where we eliminate our dynamic cap approach.
2) AlphaAdj *with no alpha adjustment, minimum alpha used:* Conservative ablation baseline where we use a fixed minimum alpha value throughout.
3) AlphaAdj *with no alpha adjustment, maximum alpha used:* Aggressive ablation baseline where we use a fixed maximum alpha value throughout.

A. Discussion

We show qualitative trajectory plots for each scenario in the warehouse (Figure 2) environment, as well as report quantitative results for each scenario in the warehouse environment in Table I. Across all 8 environment–scenario pairs, AlphaAdj reaches the goal in 8/8 setups with 0 collisions. The only other setting that consistently avoids collisions is fixed $\alpha_{\min}$. However, this conservativeness drastically increases detours: relative to AlphaAdj, averaged over all eight setups, fixed $\alpha_{\min}$ increases path length by 3.21 meters (18.5%) and time-to-goal by 1.27 seconds (10.8%). Meanwhile, fixed $\alpha_{\max}$ is overly aggressive, producing 6 collisions and only 2/8 successful runs overall. Comparing AlphaAdj to the adaptive no-cap variant, removing the cap yields 2 collisions and 6/8 successes overall. Averaged across scenarios, AlphaAdj improves over no-cap by 0.53 seconds in time to goal (4.5%). In a small number of cases, no-cap is slightly faster when the raw VLM risk predictions are already well-calibrated and stable for that scene, so capping does not provide additional benefit. We view this as a favorable trade-off; over the entirety of the scenarios, the cap substantially improves robustness to VLM latency and occasional outlier risk scores, preventing collisions and improving reliability, while only incurring a negligible efficiency cost in the rare cases where uncapped VLM predictions are already accurate. Compared to no cap, AlphaAdj is more efficient when it comes to both path length and time to goal in the scenarios shown in Figures 2b, 2c, 2d.

Our success in every scenario is notable given our controller runs continuously, with one control step every 0.0167 seconds, without pausing for any reason, even when VLM updates are delayed. Because our VLM is queried asynchronously, the robot continues executing control updates at every timestep while waiting for VLM responses. In contrast, synchronous VLM-based approaches require the robot to wait, remaining stationary while visual input is captured, sent to the VLM, processed, returned, and then converted into a control action, which substantially increases loop time. In examining synchronous approach latency analysis in [22], the average VLM-side latency (image + instruction input → VLM → response) is 9.07 seconds for VLM-Nav [23], and 1.75 seconds for VLM-Social-Nav [10]. Synchronous Vision–LLM in [24] reports an overall decision–control loop time of approximately 6.2 seconds. These are far larger than our 0.0167 seconds control period, illustrating how synchronous designs are bounded by VLM response time, whereas our approach maintains a control rate suitable for real systems.

VI. Conclusion

In this work, we introduced AlphaAdj, a latency-aware vision-only to control navigation framework that uses only egocentric RGB and a VLM risk estimate to adapt the conservativeness of a CBF safety filter in real time. The results show that AlphaAdj successfully enables collision-free, realtime visual navigation while improving efficiency and robustness in a variety of complex environments.

References

[1] G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. Kim, and P. Agrawal, "Learning to jump from pixels," 2021. [Online]. Available: https://arxiv.org/abs/2110.15344

[2] S. Gouru, S. Lakkoju, and R. Chandra, "Livenet: Robust, minimally invasive multi-robot control for safe and live navigation in constrained environments," 2024. [Online]. Available: https://arxiv.org/abs/2412.04659

[3] V. Zinage, A. Jha, R. Chandra, and E. Bakolas, "Decentralized safe and scalable multi-agent control under limited actuation," 2024. [Online]. Available: https://arxiv.org/abs/2409.09573

[4] R. Chandra, V. Zinage, E. Bakolas, P. Stone, and J. Biswas, "Deadlock-free, safe, and decentralized multi-robot navigation in social mini-games via discrete-time control barrier functions," 2024.

[5] M. De Sa, P. Kotaru, and K. Sreenath, "Point cloud-based control barrier function regression for safe and efficient vision-based control," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, 2024, pp. 366–372.

[6] F. Bordes, R. Y. Pang, A. Ajay, A. C. Li, A. Bardes, S. Petryk, O. Mañas, Z. Lin, A. Mahmoud, B. Jayaraman, M. Ibrahim, M. Hall, Y. Xiong, J. Lebensold, C. Ross, S. Jayakumar, C. Guo, D. Bouchacourt, H. Al-Tahan, K. Padthe, V. Sharma, H. Xu, X. E. Tan, M. Richards, S. Lavoie, P. Astolfi, R. A. Hemmat, J. Chen, K. Tirumala, R. Assouel, M. Moayeri, A. Talattof, K. Chaudhuri, Z. Liu, X. Chen, Q. Garrido, K. Ullrich, A. Agrawal, K. Saenko, A. Celikyilmaz, and V. Chandra, "An introduction to vision-language modeling," 2024. [Online]. Available: https://arxiv.org/abs/2405.17247

[7] J. Chen and R. Chandra, "Livepoint: Fully decentralized, safe, deadlock-free multi-robot control in cluttered environments with high-dimensional inputs," 2025. [Online]. Available: https://arxiv.org/abs/2503.13098

[8] T. Chen, A. Swann, J. Yu, O. Shorinwa, R. Murai, M. K. III, and M. Schwager, "Safer-splat: A control barrier function for safe navigation with online gaussian splatting maps," 2025. [Online]. Available: https://arxiv.org/abs/2409.09868

[9] N. Hirose, C. Glossop, D. Shah, and S. Levine, "Asyncvla: An asynchronous vla for fast and robust navigation on the edge," 2026. [Online]. Available: https://arxiv.org/abs/2602.13476

[10] D. Song, J. Liang, A. Payandeh, A. H. Raj, X. Xiao, and D. Manocha, "Vlm-social-nav: Socially aware robot navigation through scoring using vision-language models," 2024. [Online]. Available: https://arxiv.org/abs/2404.00210

[11] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[12] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3420–3431.

[13] W. Xiao, C. Belta, and C. G. Cassandras, "Adaptive control barrier functions for safety-critical systems," 2020. [Online]. Available: https://arxiv.org/abs/2002.04577

[14] M. Black, E. Arabi, and D. Panagou, "A fixed-time stable adaptation law for safety-critical control under parametric uncertainty," in *2021 European Control Conference (ECC)*, 2021, pp. 1328–1333.

[15] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames, "Episodic learning with control lyapunov functions for uncertain robotic systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Nov. 2019, p. 6878–6884. [Online]. Available: http://dx.doi.org/10.1109/IROS40897.2019.8967820

[16] A. J. Taylor and A. D. Ames, "Adaptive safety with control barrier functions," in *2020 American Control Conference (ACC)*. IEEE, Jul. 2020, p. 1399–1405. [Online]. Available: http://dx.doi.org/10.23919/ACC45564.2020.9147463

[17] D. Tscholl, Y. Nakka, and B. Gunter, "Perception-integrated safety critical control via analytic collision cone barrier functions on 3d gaussian splatting," 2025. [Online]. Available: https://arxiv.org/abs/2509.14421

[18] A. Ahmadi, L. Nardi, N. Chebrolu, and C. Stachniss, "Visual servoing-based navigation for monitoring row-crop fields," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4920–4926.

[19] Y. Li and J. Košecka, "Learning view and target invariant visual servoing for navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 658–664.

[20] T. Niwa, S. Taguchi, and N. Hirose, "Spatio-temporal graph localization networks for image-based navigation," 2022. [Online]. Available: https://arxiv.org/abs/2204.13237

[21] NVIDIA, "Isaacsim," https://github.com/isaac-sim/IsaacSim, 2026, gitHub repository, accessed 2026-03-04.

[22] J. Chen, Y. Li, P. Jiang, J. Du, Z. Chen, C. Tie, J. Deng, and L. Shao, "Lisn: Language-instructed social navigation with vlm-based controller modulating," 2025. [Online]. Available: https://arxiv.org/abs/2512.09920

[23] D. Goetting, H. G. Singh, and A. Loquercio, "End-to-end navigation with vision language models: Transforming spatial reasoning into question-answering," 2024. [Online]. Available: https://arxiv.org/abs/2411.05755

[24] T.-Y. Kim and W.-S. Choi, "Autonomous vehicle maneuvering using vision–llm models for marine surface vehicles," *Journal of Marine Science and Engineering*, vol. 13, no. 8, 2025. [Online]. Available: https://www.mdpi.com/2077-1312/13/8/1553