CogNav: Human-Inspired Collision Avoidance Strategy

Mahsa Nikmard[1,*,†], Patrizio Pelliccione[1,†] and Gianlorenzo D'Angelo[1,†]

[1]Gran Sasso Science Institute, Department of Computer Science, L'Aquila, Italy

Abstract

We present CogNav, a human-inspired framework for real-time collision avoidance in unknown, unstructured environments using only monocular vision. Motivated by human map-free navigation, CogNav maximizes trajectory convergence to a potentially unsafe reference path under partial observability. The modular pipeline combines a lightweight, fine-tuned monocular depth model that outputs a compact directional distance vector with a reactive Behavior Tree that uses temporal filtering and certainty tracking for low-latency avoidance and steering. This design enables fair benchmarking and transfer across domains and robot dynamics. Preliminary results of a proof of concept indicate accurate depth-based safety cues and competitive navigation in novel scenes.

1. Introduction

Autonomous navigation in unstructured environments remains a fundamental challenge in robotics. Many modern systems combine heuristic planning with data-driven learning and high-fidelity sensing [1, 2], and typically assume either a pre-defined map or online mapping via SLAM [3]. While effective in many domains, these approaches often depend on expensive sensors (e.g., LiDAR, GPS, compasses, RGB-D), require substantial domain-specific priors, and lack modular, transferable architectures. Consider a mobile robot operating in an initially unknown, unstructured environment. Enabling autonomous, real-time decision-making typically requires either (i) hand-crafted rule-based navigation logic or (ii) a learned control policy (e.g., via machine learning or LLM-based methods). However, fully specifying robot behavior in dynamically changing, partially observable settings is not only an open problem, it may also be impractical or even impossible to achieve in full [4, 5].

Behavior Trees [6] provide a modular and interpretable way to encode navigation logic, but they are most effective in structured settings where conditions and action alternatives can be specified upfront. In unstructured environments, enumerating reliable perceptual predicates and behavioral branches quickly becomes intractable, forcing BT-based systems to rely on learned perception modules to produce BT-compatible decisions. These models typically require large datasets and may still fail to generalize to unseen scenes [7, 8], so the overall reactivity is ultimately limited by the robustness of the perception pipeline under uncertainty and partial observability.

Motivated by this perception bottleneck, recent advances in monocular depth estimation and scene understanding provide an alternative route to navigation in novel environments. In particular, robots can leverage learned vision-based models to infer the spatial geometry of navigable regions directly from single RGB images [9]. Nevertheless, these methods are subject to several limitations: (i) decision-making is often performed offline rather than in real time, (ii) generalization degrades significantly for states unseen during training, and (iii) end-to-end architectures couple perception and control in a manner that hinders modular analysis, ablation, and transfer.

Related Works. Vision–Language Models (VLMs) show strong potential for robotics, but their deployment is often constrained by safety requirements [10]. Recent work proposes general vision foundation models that transfer across robot platforms via fine-tuning [11], while NoMaD trains transformer-based

8th International Workshop on Robotics Software Engineering (RoSE'26).

*Corresponding author.

†These authors contributed equally.

mahsa.nikmard@gssi.it (M. Nikmard); patrizio.pelliccione@gssi.it (P. Pelliccione); gianlorenzo.dangelo@gssi.it (G. D'Angelo)

policies to generate waypoints, optionally aided by depth-derived local maps [12]. However, the effectiveness of such learning-based navigation remains limited by the diversity and scale of available training data.

These foundation-model navigation systems emphasize perception in unknown, map-free environments, but typically do not provide robust, explicit collision-avoidance guarantees. To address safety, safe-VLN augments VLN-CE by integrating a collision-avoidance module that uses LiDAR-based occupancy masks to filter waypoint candidates [13]. Other approaches combine RGB perception with external localization: for example, [14] estimates obstacle distances via monocular depth and projects detections into a global frame using RTK-GPS, making the method dependent on GPS for obstacle localization in addition to relative depth cues. CARE [15] provides a plug-and-play collision-avoidance framework that uses RGB images to enable navigation in unseen environments. It first estimates depth from monocular RGB, constructs a top-down obstacle map, and then applies an Artificial Potential Field (APF) to adjust the robot's trajectory. However, APF is prone to local-minima entrapment in narrow corridors and cluttered scenes, and its computational overhead can introduce latency that limits real-time applicability.

Paper Contribution. In this paper, we adopt an explicitly human-inspired design perspective. Rather than assuming access to global maps or rich state estimation, we draw inspiration from how humans navigate unfamiliar environments: they move toward a goal without GPS, a compass, or a complete understanding of the scene, relying instead on local perceptual cues to make steady progress while continuously avoiding hazards. Accordingly, we consider an agent that begins with no pre-compiled map, operates under substantial occlusion, and may encounter dead-ends or narrow passages that require reactive re-planning, yet humans still succeed by combining instantaneous visual cues with short-term spatial memory and learned heuristics about traversable geometry [16]. Building on this observation, we study *collision avoidance*, a core challenge in autonomous navigation, for mobile robots operating under the same constraints: no pre-built map, partial observability, and limited sensing. We propose a modular architecture that enables real-time decision-making using only a monocular camera, in direct analogy to human navigation. A modular design offers two key benefits. First, it supports fair comparison with advances in vision-based models by isolating individual components of the pipeline. Second, it enables targeted analysis of module-specific failure modes across different domains and robot dynamics, facilitating systematic progress toward robust real-time operation in novel environments. Concretely, our framework derives relative collision estimates from single image frames, eliminating the need for global localization and avoiding the latency of map-based planning.

Summarizing, our contributions are: (i) we formulate and motivate a human-inspired collision-avoidance problem that optimizes for maximal convergence to a reference path, (ii) we propose a modular, reactive decision-making system that transfers across domains and robot dynamics with minimal modification, and (iii) we define evaluation metrics and provide a preliminary assessment of our approach, showing that it is competitive in the considered scenarios[1].

2. Problem Statement

Given an unknown static environment $s \in \mathcal{S}$, including one robot r, multiple obstacles $\mathcal{O}\lfloor\int = \{o_1, \ldots, o_m\}$, and a reference trajectory, the robot plans to follow the reference trajectory while avoiding obstacles. The mobile robot is using an onboard RGB camera to see surroundings and no map or positioning information is provided for the robot. Accordingly, we assume the robot operates in the environment as a human would, relying exclusively on visual perception.

We formulate the problem as a *Partially Observable Markov Decision Process* (POMDP) $\mathcal{M} = \big(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{Z}, \mathcal{O}_z, \gamma, l\big)$, in which $\mathcal{S}$ and $\mathcal{A}$ are the set of states and actions, respectively. The transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ represents the probability of moving to state $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. $\mathcal{Z}$ is the observation space, where each observation $z \in \mathcal{Z}$ corresponds to an RGB image I captured by the robot's onboard camera. $\mathcal{O}_z : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{Z})$ is the observation

[1] https://github.com/MahsaNikmard/CogNav.git

function, giving the probability distribution over observations upon arriving in state s' after taking action a. Also, γ is a discount factor and l is a loss function defined below. At every timestep t, the robot's perception of the environment is captured by the observation $z^t \in \mathcal{Z}$ (i.e., the current camera image I^t), rather than the true state s^t, which remains hidden. Let $\mathcal{P}$ denote the set of all reference paths. In the absence of positioning information, we define a *random exploration* as a reference path $P \in \mathcal{P}$ given by an ordered sequence of n waypoints in the 2D plane, $P = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. To traverse consecutive waypoints $(x_{i-1}, y_{i-1}) \rightarrow (x_i, y_i)$, the robot executes a control primitive consisting of action $a_i \in \mathcal{A}$ applied for t_i timestep, where both a_i and t_i are randomly sampled. The *length* of a reference path P is defined as $L(P) = \sum_{i=1}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$. The *Trajectory Convergence* $\mathcal{C}_P$ is the length of the entire trajectory P covered by the robot from the initial position to the goal position.

Thus, the collision avoidance problem tries to solve the problem of following the reference path P, avoiding all obstacles in the path while maximizing the trajectory convergence $\mathcal{C}_P$ for all $P \in \mathcal{P}$. Obviously, $0 \leq \mathcal{C}_P \leq L(P)$. A value $\mathcal{C}_P = L(P)$ indicates that the robot completed the entire reference path without interruption. We define the loss function $l : \mathcal{P} \rightarrow \mathbb{R}^{\geq 0}$ as the mean shortfall between the reference path length and the length actually covered by the robot, i.e., $l(P) = \frac{L(P) - \mathcal{C}_P}{n}$, where n is the number of waypoint segments in P. This loss is non-negative by construction and equals zero only if the robot traverses the entire reference path successfully. Lower values indicate that the robot avoids collisions while minimizing oscillations and excessive steering. Given a reference path $P \in \mathcal{P}$, the collision-avoidance problem seeks a policy $\pi : \mathcal{Z} \rightarrow \mathcal{A}$ that commands the robot to track the waypoints of P while avoiding obstacles, with the objective of maximizing the trajectory convergence $\mathcal{C}_P$ (equivalently, minimizing $l(P)$) for all $P \in \mathcal{P}$. In other words, it can be seen as the problem of minimizing the loss function $l(P)$ for every $P \in \mathcal{P}$. Therefore, we can also say that we wish to learn a policy π^* such that $\pi^* = \underset{\forall \pi, \forall P \in \mathcal{P}}{\mathbf{Arg}\,\min}\, l(P)$. The reference path is provided to the robot beforehand and is not guaranteed to be collision-free. The robot must therefore follow each successive waypoint of the path while reactively avoiding obstacles, with the endpoint of P treated as the *mission goal*. We say the robot *successfully* completes the mission if it reaches the goal without any collisions.

3. CogNav

To address collision avoidance in a static environment, CogNav adopts a modular design with two main components: (i) online depth estimation, and (ii) control flow.

Online depth Estimation. At a high level, the robot must perceive its surroundings to make informed decisions. In the absence of a map, it estimates relative obstacle distances and selects an action $a^t \in \mathcal{A}$ at each timestep t. Actions are defined as motion directions in the world frame, drawn from a predefined *directional convention*. We define this convention as N evenly spaced directions on a circle, with angle 0 aligned with the robot's initial heading, effectively acting as a simulated compass, increasing N yields smoother, near-continuous trajectories.

To keep the pipeline lightweight, we avoid dense depth maps, top-down mapping, or object detection. Instead, at each inference step we run a fine-tuned monocular depth estimator (Depth Anything V2 [17]) and output a compact distance vector $v^t = v_1^t, \ldots, v_N^t$, where v_i^t denotes the estimated distance to the nearest obstacle along direction i. Because single-frame estimates can be noisy—due to perception errors, robot rotation, and actuation effects—we temporally aggregate v^t and compute per-direction *certainty* and *danger* scores. This filtering converts raw frame-level predictions into more stable indicators, reducing oscillations and enabling reliable collision-aware decisions. Each training sample consists of a rendered panorama and its corresponding segmentation mask, during training we set $N = 36$.

A key design choice is to supervise the model in this *distance-vector* space rather than in pixel space. This compact representation treats the robot as the origin, provides geometrically meaningful outputs for control, and avoids reliance on dense depth ground truth while still yielding informative gradients through the decoder. Moreover, training on Webots-rendered cylindrical views helps bridge the gap between the model's natural-image prior and both the synthetic rendering style and the cylindrical

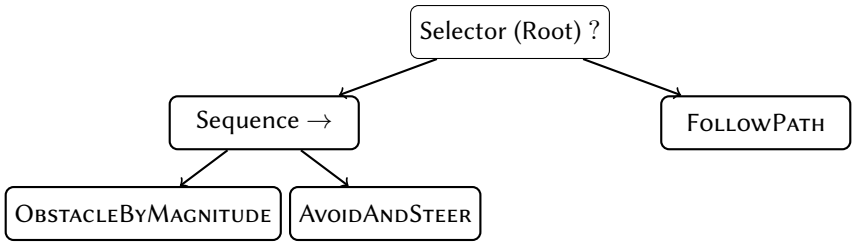
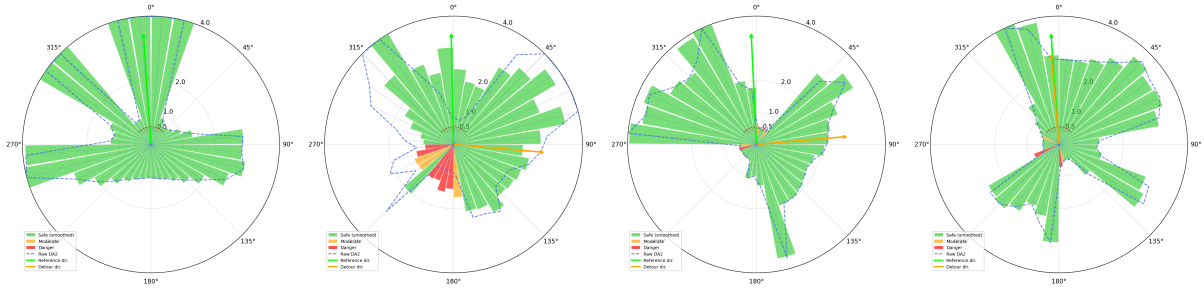


Figure 1: Control Flow Behavior Tree

projection geometry absent from the original Depth Anything V2 training distribution.

Control flow. The control-flow component represents robot behavior using a *Behavior Tree* (BT). The perception post-processing described above provides the necessary and sufficient conditions to trigger BT-based collision avoidance at run time. BTs offer a hierarchical and interpretable formalism for specifying agent behavior as a directed, rooted tree whose nodes represent tasks or control logic. Execution proceeds by propagating discrete *ticks* from the root through the tree according to the semantics of the control-flow nodes. Nodes are typically categorized as *composite*, *decorator*, or *leaf* nodes, and each node b returns a status $\theta(b) \in$ Success, Failure, Running on every tick. We implement a reactive collision-avoidance layer on top of the metric depth estimator using a BT architecture augmented with an *Exponential Moving Average* (EMA) certainty tracker. The resulting policy runs online at each inference step. In our design, the root is a *Selector* (OR) node with two children: (i) a *Sequence* (AND) subtree that handles avoidance and (ii) an action node FollowPath. At each tick, the Selector evaluates its children left-to-right and returns the status of the first child that does not return Failure. The Sequence first ticks ObstacleByMagnitude, which returns Success iff a danger condition is active. When this condition succeeds, the Sequence ticks *AvoidAndSteer* and the avoidance action is executed, so FollowPath is not ticked. If ObstacleByMagnitude fails, the Sequence returns Failure and the Selector falls back to FollowPath. This structure explicitly prioritizes *avoiding danger* over *tracking the reference path* when the condition node ObstacleByMagnitude returns Success. The resulting BT is shown in Fig. 1.

To avoid oscillations caused by frame-to-frame variations in directions that lie outside the robot's feasible motion, we filter out directions that do not influence navigation. We define a $\mathcal{C}(t) = \{i \mid i \in \{1, \ldots, n\}\}$ as the set of active directions that directly affect the robot's motion and decision-making. The forward cone is configurable based on the robot's dynamics. Forward active cone is configurable based on the robot's physics. The depth estimator outputs a vector of N raw relative distances in N directions. At every inference frequency, for every direction i, d_i^t is filtered with an EMA to cancel the frame-to-frame model noise. To further denoise the predicted relative distances at run time, we define a binary condition indicator $Q_i^t \in \{0, 1\}$ for each timestep t and direction $i \in \mathcal{C}^t$. The condition holds iff (i) the predicted distance in direction i decreases over two consecutive timestep while the robot moves forward, and (ii) the predicted distance lies within a predefined detection range. We set $Q_i^t = 1$ when the condition holds and $Q_i^t = 0$ otherwise, so Q_i^t indicates the presence (1) or absence (0) of a potential collision in direction i across two consecutive timestep. Then we intend to compute a *certainty value* $c_i^t \in [0, 1]$ using a second EMA. Certainty value specially rises when binary condition indicator holds for several consecutive timestep over time which is a strong assumption to collide an obstacle in the next few timestep in the active forward cone. When certainty value shows a blocked direction with a high assurance, we use the predicted distances to compute the *danger score* s_i^t which showcase a possible collision in that direction. When the obstacle is at the detection boundary the score is zero and when it is at zero range, $s_i = c_i$. The joint formulation ensures that a high-certainty but distant obstacle is weighted less than a lower-certainty but imminent one. In Fig. 2 the visualization plots of a scene in 4 different timestep is provided. At $t = 465$, the robot detects obstacles via post-processed depth estimates. However, since this is the first tracked frame, the certainty remains low, still allowing the robot to navigate through narrow corridors without premature avoidance. At $t = 505$, the accumulation



(a) Timestep 0 (b) Timestep 465 (c) Timestep 505 (d) Timestep 1000

Figure 2: Polar plot of depth estimator outputs at three key timestep: at $t = 0$, the raw model output (dashed blue) reflects stable depth estimates, at $t = 465$, post-processing yields a smoothed perception signal (green) that suppresses oscillation in the narrow passage, at $t = 505$, the certainty tracker converges toward the raw estimates, demonstrating the effectiveness of the proposed frame-to-frame filtering approach.

of frame-to-frame tracking increases the confidence level, while post-processing stabilizes the depth output following an immediate steering correction, ultimately triggering a decisive avoidance maneuver.

The remainder of this section details the semantics of the BT employed in the proposed framework.

• ObstacleByMagnitude. The condition node returns Success when the accumulated danger score exceeds a threshold, and triggers an immediate stop if any forward-cone slice is below d_{hard}. Danger is accumulated within a detection range and persists until the score drops below a clearance threshold and the minimum distance exceeds d_{hard}, reducing oscillations. At each tick, the node writes an n-dimensional binary *valley* vector to the blackboard indicating blocked (1) and free (0) directions.

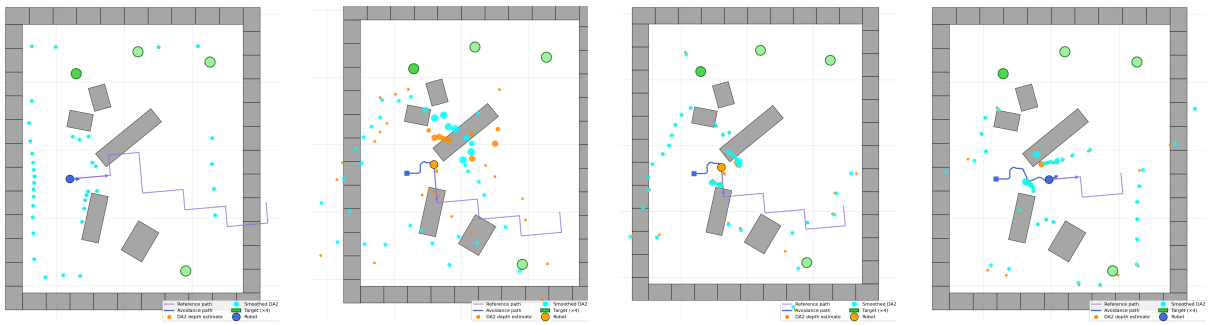
• AvoidAndSteer. When ObstacleByMagnitude returns Success, AvoidAndSteer applies VFH to the blocked-direction vector to select a free valley: it follows the reference direction if feasible, otherwise steers to a safe direction with EMA smoothing.

• FollowPath. When no danger is detected, the danger branch fails and the selector ticks FollowPath, which steers toward the current reference waypoint. The reference path is represented as a sequence of (slice-index steps) segments, advancing to the next segment after the specified number of steps.

4. Preliminary Results and Conclusion

In this section, we provide the metrics and basic results of CogNav. The metrics are defined in two categories: offline depth estimation metrics evaluated on the syntactic Webots-rendered dataset and online navigation metrics. The analysis is provided in Table 1.

The depth estimation model, will provide an estimation of the closest surface of obstacles in N slices all around the robots. We report the performance of the fine-tuned depth estimator as *Mean*



(a) Timestep 0 (b) Timestep 465 (c) Timestep 505 (d) Timestep 1000

Figure 3: Top-down scene visualizations at key timesteps, with the purple trajectory denoting random exploration and the corresponding polar plots depicting depth estimator outputs.

Absolute Error (MAE) over F samples, $\mathbf{MAE} = \frac{1}{F \cdot N} \sum_{f=1}^{F} \sum_{i=1}^{N} \left| \hat{d}_{i,f} - d_{i,f} \right|$ which is the average of the predicted $\hat{d}_{i,f}$ versus ground truth $d_{i,f}$ distance at each direction i in sample f. We formally define *Pearson Correlation* as $\mathbf{Pear_{cor}} = \frac{\text{Cov}(\hat{d},d)}{\sigma_{\hat{d}},\sigma_d}$ where $\text{Cov}(\hat{d}, d)$is the covariance of $\hat{d}$ and d, $\sigma_{\hat{d}}$ and σ_d are the standard deviation of $\hat{d}$ and d respectively. This value measures the relative ordering of distances is preserved. A value between $[-1, 0]$ reports inverted or random tracking.

Fig. 3 illustrates key timesteps of a navigation scene, where orange dots represent post-processed depth estimates and the purple line denotes the random reference path. At $t = 0$, random exploration reveals a potential collision, the robot's objective is to track the reference path while reactively avoiding obstacles. The reference path, defined as a (slice-index, steps) tuple, is updated upon each avoidance maneuver. At $t = 465$, proximity is detected (blue dots), however, since a single frame is insufficient to raise the certainty threshold, no heading adjustment is triggered, preventing premature oscillation. The corresponding polar plot (Fig. 2) shows the raw depth output (dashed blue) alongside the post-processed result (green), confirming that reactive steering effectively suppresses noisy depth estimates. At $t = 505$, the certainty tracker converges toward the raw estimates, triggering a decisive avoidance maneuver. At $t = 1000$, the robot has cleared two collision points, residual hazards outside the forward cone do not affect navigation, underscoring the role of forward-cone constraints in reducing spurious reactions. Table 1 reports average performance of CogNav across 10 random environments, where $\mathbf{D}$ and $\mathbf{K}$ denote path efficiency and collision count, respectively. Path efficiency confirms stable tracking with minimal oscillation, while collision metrics indicate that further refinement of the steering computation remains necessary.

Table 1: Depth estimation metrics

Metric	Value
$\mathbf{MAE}$	0.079
$\mathbf{Pear_{cor}}$	0.98%
$\mathbf{D}$	$89.3 \pm 21.9\%$
$\mathbf{K}$	70.0%

Overall, CogNav demonstrates that a lightweight, modular, monocular-vision pipeline can achieve reliable real-time collision avoidance in novel environments without maps or global localization.

Declaration on Generative AI

During the preparation of this work, the author(s) used **Grammarly** in order to **grammar and spelling check**. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] J. C. Bongard, Probabilistic robotics. sebastian thrun, wolfram burgard, and dieter fox. (2005, mit press.) 647 pages, Artificial Life 14 (2008) 227–229.

[2] S. M. LaValle, Planning Algorithms, Cambridge University Press, USA, 2006.

[3] H. Durrant-Whyte, T. Bailey, Simultaneous localization and mapping: part I, IEEE Robotics & Automation Magazine 13 (2006) 99–110.

[4] R. Y. Siegwart, I. R. Nourbakhsh, D. Scaramuzza, Introduction to autonomous mobile robots, second edition, in: Intelligent robotics and autonomous agents, 2011.

[5] R. Caldas, J. A. P. García, M. Schiopu, P. Pelliccione, G. Rodrigues, T. Berger, Runtime verification and field-based testing for ros-based robotic systems, IEEE Transactions on Software Engineering 50 (2024).

[6] M. Colledanchise, L. Natale, On the implementation of behavior trees in robotics, IEEE Robotics and Automation Letters 6 (2021) 5929–5936. doi:10.1109/LRA.2021.3087442.

[7] F. Fusaro, E. Lamon, E. D. Momi, A. Ajoudani, A human-aware method to plan complex cooperative and autonomous tasks using behavior trees, in: 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids), 2021.

[8] M. Behery, J. Deutsch, M. Trinh, D. Koetter, C. Brecher, G. Lakemeyer, Human robot collaborative assembly using behavior trees and dynamic tree dispatching, in: 56th International Symposium on Robotics, VDE, 2023.

[9] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, V. Koltun, Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2022) 1623–1637.

[10] J. Blumenkamp, S. Morad, J. Gielis, A. Prorok, Covis-net: A cooperative visual spatial foundation model for multi-robot applications, volume 270 of *Proceedings of Machine Learning Research*, PMLR, 2025.

[11] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, S. Levine, ViNT: A foundation model for visual navigation, in: 7th Annual Conference on Robot Learning, 2023.

[12] A. Sridhar, D. Shah, C. Glossop, S. Levine, Nomad: Goal masked diffusion policies for navigation and exploration, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024.
[13] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, S. Lee, Beyond the nav-graph: Vision-and-language navigation in continuous environments, in: Computer Vision – ECCV 2020, Springer-Verlag, 2020.
[14] V.-H.-A. Phan, C.-T. Nguyen, D.-T. Au, T.-D. Phan, M.-T. Duong, M. H. Le, Vision-based perception for autonomous vehicles in obstacle avoidance scenarios, 2025 17th International Conference on Human System Interaction (HSI) (2025).
[15] J. Kim, J. Sim, W. Kim, K. P. Sycara, C. Nam, Care: Enhancing safety of visual navigation through collision avoidance via repulsive estimation, volume 305 of *Proceedings of Machine Learning Research*, PMLR, 2025.
[16] W. H. Warren, Non-euclidean navigation, Journal of Experimental Biology 222 (2019).
[17] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, H. Zhao, Depth anything v2, in: Advances in Neural Information Processing Systems, volume 37, 2024.