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

VDS-Nav: Volumetric Depth-Based Safe Navigation for Aerial Robots–Bridging the Sim-to-Real Gap

Van Huyen Dang[1], Adrian Redder[1], Huy Xuan Pham[2], Andriy Sarabakha[2], and Erdal Kayacan[1]

***Abstract*—End-to-end navigation via deep reinforcement learning has become a key approach for vision-based tasks. However, the sim-to-real gap remains a challenge, especially for aerial robots, where policies trained in simulation often fail in real-world environments. In this work, we propose a novel navigation paradigm – volumetric depth-based safe navigation (VDS-Nav), which trains a policy to infer linear velocities and yaw rate directly from a sequence of depth images, bypassing the need for a pre-trained latent space encoder. We enhance safety with a depth-based reward design, enabling the seamless incorporation of system constraints via logarithmic barrier function methods. Most importantly, using explicit sensor information in our reward design leads to seamless sim-to-real transfer by strengthening the correlation between state-action pairs and received rewards. To evaluate the effectiveness of VDS-Nav, we compare it to a baseline that first trains a variational autoencoder to encode depth images into a latent space for policy training. The simulation results show that VDS-Nav outperforms the baseline in terms of success rate. Furthermore, real-world experiments validate the policy, with real-time performance closely matching simulation results, suggesting an effective sim-to-real transfer.**

***Index Terms*—Aerial robots, Vision-based navigation, Reinforcement learning, Reward engineering, Sim-to-Real transfer.**

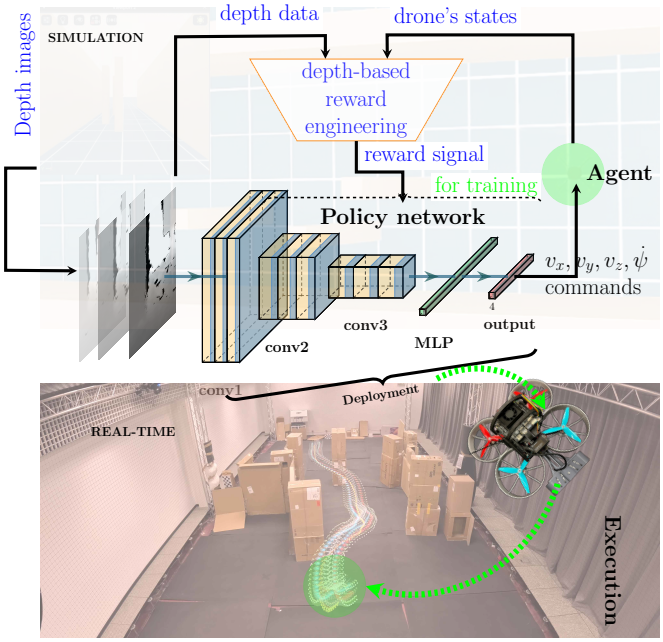


Fig. 1. Key elements of the VDS-Nav training paradigm. The top part shows how a VDS-Nav agent perceives the world during simulation and its policy architecture, which outputs linear velocities and yaw rates from a sequence of depth images. The bottom part highlights the zero-shot deployment of the trained architecture and successful navigation with a noisy depth camera.

I. Introduction

VISION-ONLY navigation methods have dominated recent applications of unmanned aerial vehicles (UAVs) due to their lightweight design and energy efficiency [1], [2], [3]. However, a major challenge of vision-only navigation frameworks is to develop robust and safe navigation algorithms with limited resources, such as relying only on a depth camera, to operate in unstructured and cluttered environments (Fig. 1). Early research on vision-based navigation has employed a modular approach, decomposing the navigation system into perception, planning, and control. Therein, the perception module estimates the states of robots by combining data from various sensors, such as cameras and inertial measurement units. Estimated states are then used as input to the planner and controller [4], [5]. Although this approach offers transparency into the interaction between these distinct modules, it often leads to suboptimal performance, as it introduces latency in processing and transferring information between modules [6].

Recent studies aim to unify navigation modules into a single, end-to-end module to reduce latency. A perception-control modularization is introduced in [7] in which perception is trained separately to transform visual input into waypoints or a latent representation vector. The waypoints are then processed with a model-based trajectory tracking control [8]. The latent vector is used in conjunction with privileged data, such as estimated states, for training navigation policies using reinforcement learning [9], teacher-student learning [10], or contrastive reinforcement learning [11]. This approach has demonstrated effectiveness in transferring policies trained in simulation environments to the real world, yet it poses the risk of information loss due to image compression, which may result in degraded performance. Alternatively, policies can instead be learned end-to-end, combining perception and planning, with deep reinforcement learning (DRL) to translate directly an image into motion primitives [12], collision-free trajectories [13], or velocity commands [14], [15]. Despite its

Manuscript received: April, 17, 2025; Revised July, 17, 2025; Accepted September, 2, 2025.

This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Horizon Europe under Grant 101119774, and Independent Research Fund Denmark, DFF-Research Project 1, with case number: 2035-00052B.

[1]Van Huyen Dang, Adrian Redder, and Erdal Kayacan are with the Automatic Control Group (RAT), Department of Electrical Engineering and Information Technology, Paderborn University, 33098 Paderborn, Germany {van.huyen.dang, adrian.redder, erdal.kayacan}@upb.de

[2]Huy Xuan Pham and Andriy Sarabakha are with the Artificial Intelligence in Robotics Laboratory (AiR Lab), Department of Electrical and Computer Engineering, Aarhus University, 8000 Aarhus C, Denmark {huy.pham, andriy}@ece.au.dk

advantages, the sim-to-real gap issue remains a challenge for end-to-end approaches [14], [16], [17], [18], [19].

To address this challenge while retaining the benefits of such approaches, we propose VDS-Nav, a method for training a navigation policy that infers control actions directly from a sequence of depth images without relying on a latent space encoder. Our primary contributions are:

- A novel depth-based reward design to improve generalization of safe vision-based navigation by strengthening the correlation between vision-action pairs and rewards through a depth-based reward design, utilizing raw sensor inputs, as well as privileged state information. This enables effective sim-to-real policy transfer.
- Enhanced learning effectiveness of navigation policies using volumetric vision inputs (i.e., a sequence of depth images to provide spatial information about obstacle locations and temporal cues about motion during training) verified in ablation studies.
- Detailed simulation studies and real-time experiments in comparison to baselines that use a latent space encoding and traditional penalty-based reward design [9], which show that VDS-Nav achieves state-of-the-art navigation performance for static and dynamic obstacles.

The remaining parts of this manuscript are organized as follows. Related works on learning-based methods for vision-based navigation and reward engineering methods are summarized in Section II. We briefly describe the problem formulation in Section III. In Section IV, we explain our proposed method. Section V describes our experimental setups, simulation results, and sim-to-real transfer. Finally, the conclusions are provided in Section VI.

II. Related work

Navigation problems are either solved with map-based or mapless solutions. A map-based approach relies on a given map to plan collision-free paths [20], [21]. A mapless approach exploits onboard sensing data to select navigation actions. For the latter, perception-control modularization [7], [8], [9], [22] and end-to-end methods [14], [12], [13] are commonly used to train navigation policies that can choose collision-free trajectories, velocity, acceleration, or thrust commands, using imitation learning [6], [14], [23], DRL [9], [12], [13], or combined reinforcement and imitation learning [10], [24]. In addition, for mapless navigation, it is common for the system/agent to perceive the environment using onboard sensors, such as a camera, so that navigation policies must infer actions from RGB images [14] or depth images [9].

Perception-control modularization starts training a feature extractor that either generates collision-free trajectories using model-based trajectory tracking control [8] or encodes a depth image into a latent representation. The feature extractor is then used in conjunction with privileged information to train collision-free navigation policies through reinforcement learning [9]. This approach helps to reduce the sim-to-real gap because i) policies rely on latent representation, which contains the essential, task-relevant information while disregarding low-level visual details, and ii) randomization techniques [16] are often applied to generate synthetic simulated RGB images [8], [14] for training feature extractors, or simulated depth image data [9] for training variational autoencoders. However, compressing a single-depth image into a lower-dimensional representation can result in information loss, which often leads to performance degradation.

In contrast, end-to-end approaches unify perception, planning, and control modules into one architecture that is usually trained using reinforcement learning algorithms such as deep Q-Networks (DQN) [25] or proximal policy optimization (PPO) [26] to maps raw depth images and relative positions to motion primitives [12], or a collision-free path [13]. Although end-to-end navigation policies are robust [8], they struggle with high sample complexity and poor generalization to circumstances not encountered during training. Additionally, the sim-to-real gap remains a challenge for learning-based policies trained in simulation due to the mismatch between simulation and the real world. To overcome high sample complexity, [10], [24] utilize a privileged learning-by-cheating framework, in which DRL is used to train the teacher policy with privileged information, which is then distilled through imitation learning to obtain the student policy. To improve generalization and reduce the sim-to-real gap, [14], [16] employ domain randomization techniques to diversify training environments, while [27] develops a simulation environment that resembles the real-world environment with realistic depth sensor noises.

Despite numerous contributions in vision-based policy network design and learning algorithms for both perception-control modularization and end-to-end approaches, there is a lack of structured reward design based on the actual sensor perception of a system equipped with a depth camera. Typical reward functions are designed based on global ground-truth information or binary feedback, excluding the explicit use of depth information. Consequently, such rewards can have a significantly more complex (often more delayed) relation to a vision-based state space, which can lead to a low correlation between state-action and short-term rewards. For instance, [12] designs a navigation reward function involving the distance to a goal, sparse binary penalties for collisions, and penalties for excessive deviations from a preplanned path to the goal. [9] proposes a reward function composed of four terms that reward the agent if it approaches the goal and stays there. Additionally, eight terms penalize the agent for ensuring that the commanded velocity vector lies within the field of view of the depth sensors, thereby preventing a sideways collision with the environment. The final term is a binary penalty if the agent collides with an obstacle. Similarly, [24] introduces a reward to reinforce the current yaw angle of the camera to align with the next flight direction.

In contrast, our volume depth-based navigation policy is trained end-to-end using a novel reward design, which includes a term that uses the dot product between the system's velocity vector and the coordinate vector of every depth camera pixel. With this, safe agent behavior can be reinforced continuously and immediately, without the need to encode individual obstacles. Most importantly, this leads to immediate strong correlation between state-action pairs and rewards, and in our experiments, to highly effective sim-to-real transfer.

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

III. Problem Formulation

In this work, we tackle vision-based UAV navigation problems that require autonomous exploration, such as search and rescue operations or inspections in confined facilities. Thus, we assume a mapless navigation problem, where the system perceives its environment using an onboard depth camera. It must navigate towards the open end of confined environments using a policy in the presence of static and/or dynamic obstacles. In other words, there is no explicit goal position. The overall architecture of the policy network is described in Fig. 2. We formalize the navigation problem as a discounted cost Markov decision process (MDP) [28], represented by a 5-tuple $(\mathbf{S}, \mathbf{A}, \mathbf{R}, p, \gamma)$. In this formulation, $\mathbf{S} = \{\mathbf{O}, \mathbf{P}\}$ denotes the state space that contains a set of depth images ($\mathbf{O}$) and privileged information ($\mathbf{P}$), e.g., the agent's states and collision binary signals. $\mathbf{A}$ represents the action space for which a control vector $\mathbf{a} \in \mathbf{A}$ including linear velocities v_x, v_y, v_z along x-, y-, and z-axes, respectively, and yaw rate $\dot{\psi}$ in the agent's body coordinate system. This control vector is then handled by geometric tracking control [29]. The reward function $\mathbf{R} : \mathbf{S} \times \mathbf{A} \longrightarrow \mathbb{R}$ assigns a reward $r(\mathbf{o}, \mathbf{p}, \mathbf{a})$ to the agent based on the action $\mathbf{a} \in \mathbf{A}$ taken while observing state $\mathbf{o} \in \mathbf{O}$ and privileged information $\mathbf{p} \in \mathbf{P}$. Additionally, p is the conditional probability kernel that describes the probability of state transitions given the current state and the action executed. Finally, the discount factor $\gamma \in (0, 1)$ is applied to evaluate the long-term rewards of the decision-making process.

To solve this MDP, we employ DRL to train a policy $\pi_\theta : \mathbf{O} \rightarrow p(\mathbf{A})$ that maps states without privileged information to a probability distribution over actions. Herein, the policy is represented by a deep neural network parameterized by θ, and $\pi_\theta(\mathbf{a}|\mathbf{o})$ represents the probability of selecting the action $\mathbf{a}$ in state $\mathbf{o}$. At each step of interaction with the MDP, the agent samples and executes an action from this distribution. Afterward, it receives a reward, and the next sensor state in the environment. The policy π_θ is updated sequentially using finite trajectories from the interaction data to maximize the discounted cumulative reward:

$$J_{\pi_\theta} := \mathbb{E}_{\tau \sim \pi_\theta, \mathbf{s}_0} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \right]. \tag{1}$$

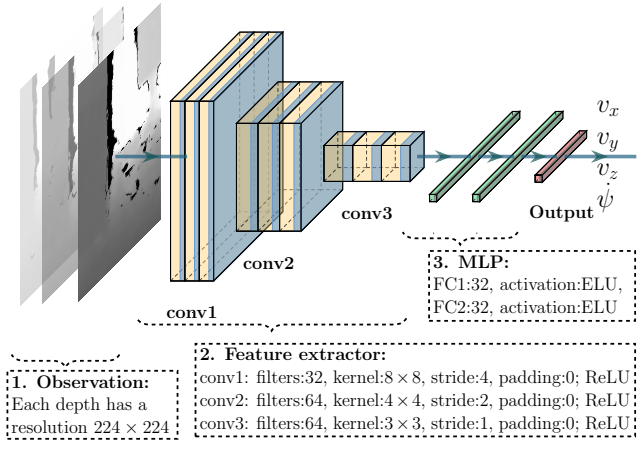


Fig. 2. The policy network takes only a sequence of depth images as input, and it includes a feature extractor, an MLP, and an output layer.

IV. Methodology

This work focuses on solving depth-based navigation problems in a mapless manner using a model-free DRL method. Various model-free algorithms for continuous action spaces exist – such as PPO [26], deep deterministic policy gradient [30], or soft actor-critic [31]. In this work, we employ PPO due to its balance of practicality, ease of implementation, and deployment efficiency. We propose two major contributions for our custom PPO implementation: i) the use of a sequence of depth images in the state space and ii) a reward design that uses explicit depth camera data to strengthen the correlation between state-action pairs and rewards, and in turn to enable effective sim-to-real transfer.

A. Volumetric depth-based policy network

This section details the policy network architecture when using VDS-Nav, in which a sequence of several consecutive depth images is used instead of a single depth image in the state space. Each depth image has a resolution of 224×224. The update frequency of the depth camera is set at 50Hz. We use a memory queue to store a sequence of the latest consecutive depth images, and this queue is renewed at each training reset. The sequence length is empirically chosen, and the impact of its variation will be discussed in Section V.

Unlike [9], in which a variational autoencoder architecture [32] has been employed to compress a high-dimensional depth image into a very low-dimensional latent space vector that is propagated to the policy network, including three fully connected layers, 512, 256, 64 neurons, followed by an ELU activation function, [8] has modified the DroNet architecture [33] to obtain a lightweight convolutional neural network (CNN) that allows on-board execution. [14] has built both Q-function and perception-based policy based on the VGG16 [34].

In this work, we simplify the VGG16 to obtain a lightweight CNN for feature extraction. The feature extractor contains three 2-dimensional convolution layers: conv1: filters 32, kernel 8×8, strike 4, and padding 0; conv2: filters 64, kernel 4×4, strike 2, and padding 0; conv3: filters 64, kernel 3×3, strike 1, and padding 0. Each convolution layer is followed by a ReLU non-linearity function. We then add an Multilayer perceptron (MLP) containing two fully connected layers, where each layer comprises 32 hidden neurons, with the ELU activation function, and a linear output layer.

B. Depth-based reward design

Before going into details, it is worth noting that a piecewise reward function is often established to train navigation policies using DRL. The reward function typically comprises several reward and penalty terms that vary according to specific conditions. For most problems, one can easily write a sparse reward function based on the task description, then apply reward shaping techniques [35], [36] to overcome the temporal credit assignment problem by utilizing privileged information, such as ground truth agent's states, predefined waypoints, and goal position [9], [12], [24]. The privileged information can

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

be easily obtained within the simulation environment, but may be noisy or not be available in the real world. This discrepancy may contribute to the sim-to-real gap, which can be alleviated by applying domain randomization techniques [14], [16], [27]. Additionally, if the privileged information used in reward design is not given to the policy network, it will usually lower the correlation between state-action pairs and rewards; consequently, it may lead to learning inefficiency. Naturally, using minimal privileged information while utilizing observation states in reward design is intended to reduce the sim-to-real gap, which in turn helps to perform the sim-to-real transfer more effectively.

To achieve this, we propose VDS-Nav, a depth-based navigation policy that maps a sequence of raw depth images to control actions, which is trained end-to-end using a general continuous reward design that uses raw depth images to enhance the correlation between state-action pairs and rewards for effective navigation and collision avoidance. In addition to enhancing correlation, designing such reward functions also necessitates a principled and interpretable approach, because despite numerous successful works in reward design for navigation tasks, there is a lack of intuition on how to construct them strategically. To address this, we draw on a constrained optimization problem, also known as nonlinear programming problem (NLP), which has long been developed and widely used in control theory and system optimization, providing a principled framework that parallels reward design: the reward corresponds to the objective, while task requirements can be expressed as constraints. Inspired by this perspective, we consider reward design as a task to define an NLP, allowing us to adopt standard optimization terminologies and handle task requirements through established methods, e.g., barrier function methods [37], resulting in a clearer and more interpretable formulation. This reward design is used to train a policy that infers actions from raw pixels, guiding the agent equipped only with a depth camera to fly ahead toward the open end of confined environments while avoiding obstacles.

We denote the total operational space as $\mathcal{C}$, which includes the free space $\mathcal{C}_{\text{free}}$, the space occupied by static and dynamic obstacles $\mathcal{C}_{\text{obs}}$, and the unknown space which is considered in a conservative planning manner. The objective is to keep the agent flying forward ($v_x^B > 0$) within a certain height range ($z_{\min} < z < z_{\max}$), while avoiding collision with obstacles ($\mathbf{x} \in \mathcal{C} \setminus \mathcal{C}_{\text{obs}}$). Hence, a constrained NLP can be written as

$$\textbf{NLP:} \quad \min_{\mathbf{a}} \left(\left| \dot{\psi}^B \right| \right) \quad \text{s.t.} \quad \begin{array}{l} \text{Moving forwards:} \quad v_x^B > 0 \\ \text{Height limits:} \quad z_{\min} < z < z_{\max} \\ \text{Collision avoidance:} \quad \mathbf{x} \in \mathcal{C} \setminus \mathcal{C}_{\text{obs}} \end{array}, \tag{2}$$

in which $\mathbf{a} = \left[v_x^B, v_y^B, v_z^B, \dot{\psi}^B\right]^T$ is a vector of control actions including linear velocity and yaw-rate commands in the agent's body coordinate, $\mathbf{x} = \left[x, y, z\right]^T$ is a vector of translational position of the agent along x-, y-, and z-axes in the inertial frame, $z_{\min}$ and $z_{\max}$ are the lower and upper limits of the agent's altitude. Since the agent is encouraged to pass through the open end of the confined environment, it is necessary to minimize the absolute value of the yaw rate. Thanks to the nature of the confined environment, e.g., open-ended hallway, it discourages the agent with a low yaw rate from turning sharply. We also add a small red cube at the end of the hallway to indicate the finish.

The collision-avoidance constraint can be seen as ensuring that the minimum Euclidean distance $\min(d_{\text{obs},i})$ from the agent to the nearest point on the facets of bounding volumes that approximate the geometry of each obstacle is greater than a safe threshold d_{safe}:

$$\min(d_{\text{obs},i}) > d_{\text{safe}} \quad \forall i = 1, 2, \dots N. \tag{3}$$

Generally, we can obtain information about obstacles, such as their states, shapes, and sizes, when training a DRL agent in simulation. However, we often randomize scenes to attain generalization by varying these attributes. Consequently, gathering information on these entities results in longer training times. Instead, we usually get a binary signal (true or false) that indicates whether a collision happens between the agent and obstacles. The lack of detailed information makes it infeasible to define the constraint described in (3). Additionally, once the agent has passed an obstacle, its information is no longer needed. Thus, we use depth data from the front-facing camera to establish a depth-based constraint that facilitates effective collision avoidance. Intuitively, the agent can comprehend collision avoidance through its movement in relation to its surroundings. As the agent moves straight toward an obstacle, the angles between its velocity (also the commanded velocity) and the unit vectors from the camera to the obstacle's pixels in the depth image stay small. Thus, we can use an inner product between the commanded velocity and the unit vectors pointing to obstacle pixels to indicate the level of safety or risk of collision, illustrated in Fig. 3. The inner product between the commanded velocity and the unit vector is defined as

$$\mathbf{I}_{i,j} = \mathbf{v} \cdot \mathbf{u}_{i,j} \quad i = 1, 2, \dots, H; \quad j = 1, 2, \dots, W, \tag{4}$$

where $H \times W$ represents the resolution of the depth image, $\mathbf{I}_{i,j}$ is the inner product between the agent's velocity $\mathbf{v}$ in its body coordinates, and $\mathbf{u}_{i,j}$ is a unit vector that points from the camera to the $< i, j >^{\text{th}}$ pixel. The unit vector $\mathbf{u}_{i,j}$ is determined based on intrinsic camera parameters,

$$\mathbf{u}_{i,j} = \frac{\left[\frac{i \cdot f}{W}, \frac{j \cdot f}{H}, f\right]}{\left\| \left[\frac{i \cdot f}{W}, \frac{j \cdot f}{H}, f\right] \right\|}, \tag{5}$$

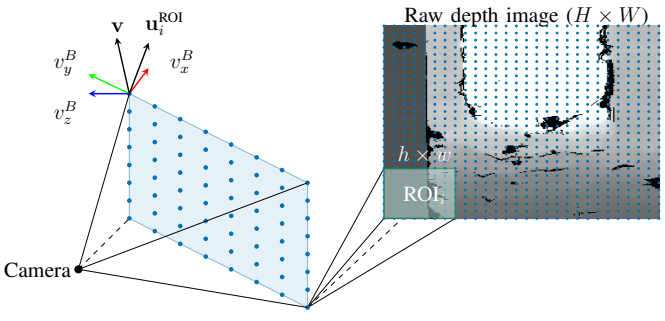


Fig. 3. The inner product ($\mathbf{v} \cdot \mathbf{u}_i^{\text{ROI}}$) indicates whether the agent is likely to move toward an obstacle or away from it.

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

in which the camera is described with a pinhole model, and f is the focal length. To account for the computational complexity that grows with the image size, we split the image into M regions of interest (region of interests (ROIs)) to concentrate on these key areas instead of calculating unit vectors for every pixel. Each ROI gets an average value of the surrounding pixels defined by

$$d_i^{\text{ROI}} = \frac{1}{h \times w} \sum_{j=1}^{h \times w} d_j \quad i = 1, 2, \ldots, M, \tag{6}$$

where d_j represents the relative distance between the camera and obstacles on the j^{th} pixel. Herein, we focus on the ROI that represents the shortest distance from obstacles when the agent is moving towards them, ensuring it falls within an effective distance range $\left(d_{\min}, d_{\max}\right)$.

Hence, the NLP in (2) can be reformulated as

$$\begin{aligned} \min_{\mathbf{a}} \left(\left|\dot{\psi}^B\right|\right) \\ \text{s.t.} \quad \begin{aligned} & v_x^B > 0 \\ & z_{\min} < z < z_{\max} \\ & d_{\min} < d_{\min}^{\text{ROI}} < d_{\max} \end{aligned} \end{aligned}, \tag{7}$$

in which the shortest distance, $d_{\min}^{\text{ROI}}$, is computed as

$$d_{\min}^{\text{ROI}} = \min\left(d_i^{\text{ROI}} \times \max(\mathbf{I}_i^{\text{ROI}}, \xi)\right), \tag{8}$$

where $\mathbf{I}_i^{\text{ROI}} = (\mathbf{v} \cdot \mathbf{u}_i^{\text{ROI}}); \forall i = 1, 2, \ldots, M$. Here, $\mathbf{I}_i^{\text{ROI}}$ is the inner product between the agent's velocity and the unit vector with respect to i^{th} ROI, denoted by $\mathbf{u}_i^{\text{ROI}}$. We maximize the magnitude of the inner product and a small positive number ξ to ensure that if the agent moves away from obstacles, it follows speed and height restrictions instead of being penalized by collision constraints. We then use the logarithmic barrier function method [37] to merge constraints into the cost function, resulting in an unconstrained NLP, whose negation can be used as a reward function expressed as

$$\begin{aligned} r(s_t, a_t) = & -\alpha_{\dot{\psi}} \left|\dot{\psi}^B\right| \\ & + \alpha_z \left[\log\left(\mathrm{z}_{\text{norm}} + \epsilon\right) + \log\left(1 - \mathrm{z}_{\text{norm}} + \epsilon\right)\right] \\ & + \alpha_d \left[\log\left(\mathrm{d}_{\text{norm}} + \epsilon\right) + \log\left(1 - \mathrm{d}_{\text{norm}} + \epsilon\right)\right] \\ & + \alpha_v \log\left(v_x^B + \epsilon\right), \end{aligned} \tag{9}$$

where the parameters $\alpha_{\dot{\psi}}$, α_z, α_v, and α_d are empirically selected weight factors that balance the contributions of the respective terms in the reward function. We normalize the height (z_{norm}) and the minimum relative distance of the ROI (d_{norm}) and introduce $\epsilon = 10^{-6}$ to ensure the validity of the logarithmic function:

$$\begin{aligned} \mathrm{z}_{\text{norm}} &= \frac{z - z_{\min}}{z_{\max} - z_{\min}}, \\ \mathrm{d}_{\text{norm}} &= \frac{d_{\min}^{\text{ROI}} - d_{\min}}{d_{\max} - d_{\min}} \end{aligned} \tag{10}$$

This reward function encourages the DRL agent to learn a policy that maximizes the cumulative reward while satisfying safety and performance constraints. For training purposes, we utilize only the global position in the x direction to terminate the training episode when the agent reaches the open-ended hallway. Additionally, we use the binary contact force signals to terminate the training if a collision occurs.

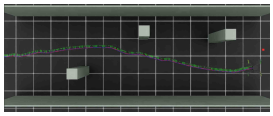
V. Experiments

A. Training and evaluation setups

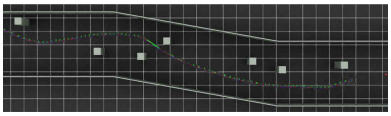
We design a training scenario with three cube obstacles placed in an open-ended narrow hallway, as shown in Fig. 4a. The training environment is $6\text{m} \times 2\text{m} \times 2.5\text{m}$ in terms of length, width, and height. During training, obstacle locations are randomized in each episode of reset. We establish 16 parallel training environments using NVIDIA IsaacLab [38] with a GPU Nvidia RTX A6000 Ada Generation and utilize a high-performance framework for reinforcement learning [39] to train our vision-based policy with PPO [26]. We train each policy for 2000 episodes, using a minibatch size of 1024. Consequently, the policy is trained for 2048000 steps with a wall-clock time of 4.12 hours. For evaluation, we generate 100 scenes for each of the following settings: i) A medium-scale scene ($18\text{m} \times 3\text{m} \times 2.5\text{m}$) with 7 or 11 static obstacles (see Fig. 4b). ii) A large-scale scene ($20\text{m} \times 10\text{m} \times 5\text{m}$) with 30 or 50 static obstacles (see Fig. 4c). iii) Setting i) and ii) with dynamic obstacles.

B. Baselines

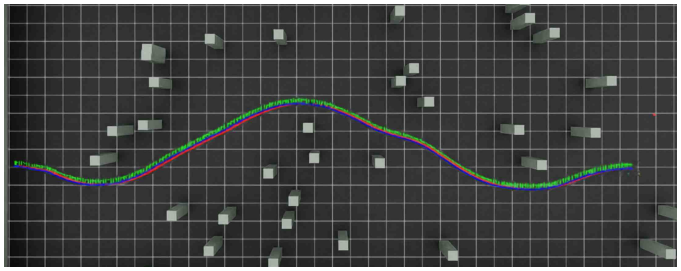
To evaluate the performance of vision-based policies trained with our suggested method, described in Section IV, we compare them against the state-of-the-art baseline: collision-free navigation policies using a pre-trained depth collision encoder (DCE) [9], in which a DCE is trained to encode depth images into a latent space vector. This comparison achieves our goal of comparing our end-to-end navigation policies learned directly from depth pixels with those derived from a perception-control modularization approach. To illustrate the effectiveness of VDS-Nav, we include three ablation studies on: i) volumetric depth state space effect, ii) scalability, and iii) depth-based reward design effect. We use metrics such as success rate (SR), collision rate (CR), timeout rate (TO), and average lap time (ALT) to evaluate the performance of all



(a) Training scenario



(b) Medium-scale evaluation scenario



(c) Large-scale evaluation scenario

Fig. 4. Training and evaluating scenarios used in the simulation studies to validate the scalability of our proposed method.

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

policies in simulation, and we measure additionally average mean absolute jerk (AMAJ) and average maximum jerk (AMJ) to evaluate the smoothness and responsiveness of the policies.

C. Simulation results

Policies trained with VDS-Nav exhibit better learning curves, and show superior training stability, presented in Fig. 5, compared to the baselines using DCE in terms of episodic cumulative reward mean under variation of the number of depth images used in the observation space after approximately 2 million training steps.

1) Volumetric effect: We compare navigation policies trained with VDS-Nav and DCE with variation in the number of depth images and using the same depth-based reward function, discussed in Section IV-B. The results, shown in Fig. 6, indicate that increasing the number of depth images in the observation space improves the success rate for DCE- and VDS-Nav-based policies under the presence of static and dynamic obstacles. Especially, navigation policies trained with VDS-Nav outperform the baselines. For a small-scale evaluation scene with three static obstacles, policies trained with our proposed method acquire success rates of 85%, 92%, and 93% corresponding to observation spaces including 1, 3, and 5 depth images, respectively. We observe that the collision rate decreases from 8% with a single depth to 4% for three depth images, and no collision with five depth images. These success rates decrease to 65%, 76%, and 83% when dynamic obstacles evolve, as dynamic movements of obstacles form movement-infeasible gaps and create sideways collisions reflected by an increase in timeout and collision rates. The average time taken to navigate the hallway successfully for policies trained with VDS-Nav is approximately 5.707 ± 0.03 seconds, whereas it is about 6.13 ± 0.22 seconds for policies trained with DCE. The average time taken for the baselines is higher, which can be attributed to the compression from a high-dimensional input to a very low-dimensional latent vector, causing information loss; therefore, the agent cannot find a feasible path. When dynamic obstacles are present, the average lap time taken is approximately 5.74 ± 0.056 and 6.15 ± 0.095 seconds for policies trained with VDS-Nav and DCE, respectively.

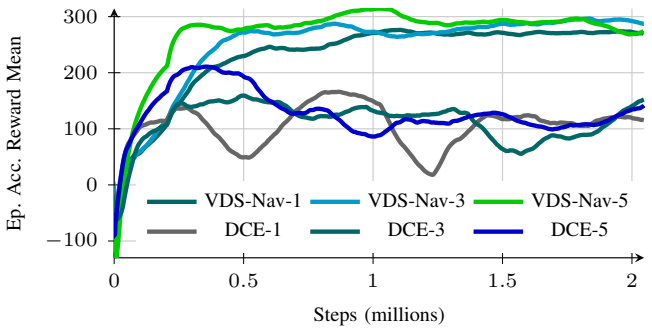


Fig. 5. Episodic cumulative reward mean according to policies trained with VDS-Nav and DCE under 1, 3, and 5 depth images used in the observation.

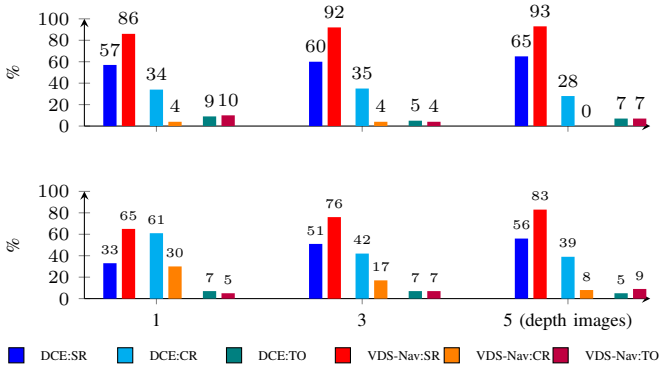


Fig. 6. Policies trained with VDS-Nav outperform the baselines using DCE across different numbers of depth images in terms of SR, CR, TO for static obstacles (top) and dynamic obstacles (bottom).

2) Scalability: We validate the scalability of our vision-based navigation policies in a medium-scale hallway ($18\text{m} \times 3\text{m} \times 2.5\text{m}$) with two levels of obstacle density: 7 and 11 obstacles. We maintain a fixed height for the drone when testing in the large-scale environment since there is no hallway/corridor effect. The results presented in Table I demonstrate that policies trained with VDS-Nav scale well in larger environments, achieving high success rates with static and dynamic obstacles. Notably, using depth images of 3 and 5 layers, the success rates for navigating around static obstacles are as follows: 88% and 91% with 7 obstacles, and 66% and 82% with 11 obstacles. When dynamic obstacles are introduced, the success rates decrease to 81% and 86% with 7 obstacles, and further drop to 54% and 60% with 11 obstacles. Additionally, we validate the policies trained with VDS-Nav in a larger environment measuring, where the number of obstacles is significantly increased to 30 and 50. The statistics details for these tasks are given in Table I.

3) Effect of depth-based reward design: To demonstrate the effectiveness of our proposed depth-based reward design, we use the penalty-based reward function from [9] as a baseline and simplify its goal-oriented term to better align with our defined task. In our scenario, the agent does not have a specific destination; instead, it needs to reach the open end of the hallway. We focus on training policies with a single depth and then test them in a training-like environment. The results, shown in Table II, indicate that the policy trained with the depth-based reward achieves an 86% success rate and a lower 10% timeout rate. In contrast, the policy trained using the penalty-based method only achieves a 56% success rate and a 20% timeout rate. This suggests that our proposed depth-based reward function more effectively guides the learning process than the baseline approach. Additionally, the average lap time for a policy trained with our depth-based reward design, along with the barrier function, is faster at 5.14 seconds, compared to 6.83 seconds for the penalty-based method. We also conducted real-time tests for both policies. We observed that the success rate for the policy trained with our proposed depth-based reward function is 80%, which is a slight decrease compared to earlier results. However, the policy trained with the penalty-based reward, using privileged information, achieved success only 20% (1 out of 5 trials). We do not consider the average lap time taken for this test due to only one successful flight.

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

TABLE I

SCALABILITY EVALUATION OF VDS-NAV POLICIES IN MEDIUM-SCALE ($18\text{m} \times 3\text{m} \times 2.5\text{m}$) AND LARGE-SCALE ($20\text{m} \times 10\text{m} \times 5\text{m}$) ENVIRONMENTS SUBJECT TO VARIATION OF THE DENSITY OF STATIC AND DYNAMIC OBSTACLES. METRICS MARKED WITH ↑ INDICATE THAT HIGHER VALUES ARE BETTER, WHILE METRICS MARKED WITH ↓ INDICATE THAT LOWER VALUES ARE BETTER.

	1 depth image				**3 depth images**				**5 depth images**			
# obstacles	**SR** [%] ↑	**CR** [%] ↓	**TO** [%] ↓	**ALT** [s] ↓	**SR** [%] ↑	**CR** [%] ↓	**TO** [%] ↓	**ALT** [s] ↓	**SR** [%] ↑	**CR** [%] ↓	**TO** [%] ↓	**ALT** [s] ↓
7 Static (medium-scale)	49	42	9	17.01	88	4	8	16.03	91	8	1	16.55
11 Static (medium-scale)	48	46	6	16.93	66	25	9	16.94	82	11	8	16.98
7 Dynamic (medium-scale)	37	58	5	16.69	81	13	6	17.11	86	1	13	16.81
11 Dynamic (medium-scale)	24	71	5	16.9	54	40	6	16.35	60	33	7	16.3
30 Static (large-scale)	87	12	1	17.62	89	10	1	17.16	91	7	2	17.13
50 Static (large-scale)	70	28	2	18.06	79	17	4	19.56	82	15	3	17.78
30 Dynamic (large-scale)	82	16	2	18.89	87	12	1	19.16	89	9	2	19.42
50 Dynamic (large-scale)	67	31	2	19.57	70	27	3	19.31	78	20	2	18.77

TABLE II

THE PERFORMANCE OF POLICIES TRAINED WITH THE PENALTY-BASED AND THE LOGARITHMIC BARRIER-BASED REWARD FUNCTIONS.

Methods	**SR** [%] ↑	**CR** [%] ↓	**TO** [%] ↓	**ALT** [s] ↓	**SR** [%] (Real) ↑	**ALT** [s] (Real) ↓
Penalty-based [9]	56	24	20	6.83	20	-
Barrier-based (ours)	86	4	10	5.14	80	10.1

TABLE III

EVALUATION OF POLICIES TRAINED WITH OUR PROPOSED VDS-NAV IN REAL-TIME WITH A VARYING NUMBER OF DEPTH IMAGES.

# images	**Static obstacles**				**Dynamic obstacles**			
	SR [%] ↑	ALT [s] ↓	AMAJ ↓	AMJ ↓	SR [%] ↑	ALT [s] ↓	AMAJ	AMJ
01	80	10.1	3.75	14.14	60	12.23	3.87	14.68
03	100	8.5	3.67	13.0	-	-	-	-
05	100	7.3	3.3	12.18	100	8.2	3.98	15.83

D. Sim2real transfer

We deployed policies trained with the VDS-Nav on a drone equipped with a Realsense D455 camera and an onboard computer (NVIDIA Orin NX). The Realsense D455 provides depth images as the observation space, and all computations are executed on the GPU of the onboard computer to output velocity and yaw rate commands in the drone's body coordinate. These commands are handled by using PX4 Autopilot and its flight stack. We validate our trained policies within settings that include static and dynamic obstacles, as shown in Fig. 7. We adjusted the number of depth images (1, 3, and 5) used in the observation space to investigate how effective a sequence of depth images is, as shown in Figs. 7a–7c. A person approaches and prevents the autonomous agent from flying along a path, showcasing the trained policy's ability to avoid dynamic obstacles, as depicted in Fig. 7d.

We conducted five trials per setting, and the results in Table III show that policies trained with VDS-Nav achieve higher SR, shorter ALT, and smoother trajectories than the policy with one depth image, as reflected in the lower AMAJ and AMJ values. In particular, the policies with 3 and 5 images achieve 100% success, completing the track in 8.5 and 7.3 seconds, respectively, while the one with 1 image achieves 80% success rate in 10.1 seconds. Given the comparable performance of policies with 3 and 5 images in static scenarios, dynamic tests were conducted for those with a single image and a sequence of five images. The policy with five images outperforms the policy with one image, achieving a 100% success rate with a shorter average lap time of 8.2 seconds, compared to 60% success and 12.23 seconds. Slightly higher AMAJ and AMJ values ($3.98\ \text{m/s}^3$ & $15.83\ \text{m/s}^3$) compared to ($3.86\ \text{m/s}^3$ & $14.68\ \text{m/s}^3$) for the policy with 5 images, can be understood as it is more responsive to a sudden obstacle in dynamic settings.

(a) 01 depth image (b) 03 depth images (c) 05 depth images

(d) Dynamic scene (e) Depth (f) Trajectory

Fig. 7. We established static and dynamic scenes, illustrated in 7a–7c, to evaluate VDS-Nav with 1, 3, and 5 depth images. We challenge the drone by involving a human (see Fig. 7d) where the depth image is shown in Fig. 7e and the complete trajectory is depicted in Fig. 7f.

VI. CONCLUSION AND FUTURE WORK

This manuscript presents a method for training vision-based navigation policies in an end-to-end fashion applied to UAVs with a depth camera. The trained policy infers velocity and yaw rate commands directly from a series of depth images without relying on any intermediary representation. Throughout simulation analysis and real-time experiments, we show that a single-depth image may not adequately reflect how fast the agent approaches obstacles. Hence, using volumetric depth data enhances the policy's performance. Additionally, using depth images to define constraints, seamlessly integrated into the cost function using logarithmic barrier functions, improves safety and minimizes the sim-to-real gap, as it strengthens the correlation between sensor state-action pairs and rewards.

Since VDS-Nav relies on depth-based observations, it can be generalized beyond indoor tasks to outdoor navigation, such as forest exploration or critical infrastructure inspection. Future work will evaluate the VDS-Nav in diverse outdoor scenarios to validate its efficiency, robustness, and effectiveness.

ACKNOWLEDGMENT

The authors would like to thank Marcus Hund, Guilherme Daudt, and Hürkan Şahin for helping with the real-time tests.

REFERENCES

[1] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. Ijspeert, D. Floreano *et al.*, "The current state and future outlook of rescue robotics," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, 2019.

[2] T. Özaslan, G. Loianno, J. Keller, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, "Autonomous navigation and mapping for inspection of penstocks and tunnels with mavs," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1740–1747, 2017.

[3] D. Thakur, G. Loianno, W. Liu, and V. Kumar, "Nuclear environments inspection with micro aerial vehicles: Algorithms and experiments," in *Proceedings of the 2018 International Symposium on Experimental Robotics*. Springer, 2020, pp. 191–200.

[4] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2016.

[5] Y. Wang, J. Ji, Q. Wang, C. Xu, and F. Gao, "Autonomous flights in dynamic environments with onboard vision," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1966–1973.

[6] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[7] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," *arXiv preprint arXiv:1804.09364*, 2018.

[8] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2019.

[9] M. Kulkarni and K. Alexis, "Reinforcement learning for collision-free flight exploiting deep collision encoding," *arXiv preprint arXiv:2402.03947*, 2024.

[10] J. Fu, Y. Song, Y. Wu, F. Yu, and D. Scaramuzza, "Learning deep sensorimotor policies for vision-based autonomous drone racing," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 5243–5250.

[11] J. Xing, L. Bauersfeld, Y. Song, C. Xing, and D. Scaramuzza, "Contrastive learning for enhancing robust scene transfer in vision-based agile flight," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5330–5337.

[12] E. Camci, D. Campolo, and E. Kayacan, "Deep reinforcement learning for motion planning of quadrotors using raw depth images," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[13] H. I. Ugurlu, X. H. Pham, and E. Kayacan, "Sim-to-real deep reinforcement learning for safe end-to-end planning of aerial robots," *Robotics*, vol. 11, no. 5, p. 109, 2022.

[14] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[15] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza, "Demonstrating agile flight from pixels without state estimation," *arXiv preprint arXiv:2406.12505*, 2024.

[16] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[17] E. Valassakis, Z. Ding, and E. Johns, "Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5372–5379.

[18] Y. Jang, J. Baek, S. Jeon, and S. Han, "Bridging the simulation-to-real gap of depth images for deep reinforcement learning," *Expert Systems with Applications*, p. 124310, 2024.

[19] H. X. Pham, A. Sarabakha, M. Odnoshyvkin, and E. Kayacan, "Pencilnet: Zero-shot sim-to-real transfer learning for robust gate perception in autonomous drone racing," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 847–11 854, 2022.

[20] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.

[21] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2020.

[22] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2169–2176.

[23] V. Tolani, S. Bansal, A. Faust, and C. Tomlin, "Visual navigation among humans with optimal control as a supervisor," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2288–2295, 2021.

[24] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza, "Learning perception-aware agile flight in cluttered environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1989–1995.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[27] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, O. Lehmann, T. Chen, A. Hutter, S. Zakharov, H. Kosch *et al.*, "Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition," in *2017 International conference on 3d vision (3DV)*. IEEE, 2017, pp. 1–10.

[28] S. Meyn, *Control systems and reinforcement learning*. Cambridge University Press, 2022.

[29] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. Pmlr, 2018, pp. 1861–1870.

[32] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *International conference on learning representations*, 2017.

[33] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[35] A. D. Laud, *Theory and application of reward shaping in reinforcement learning*. University of Illinois at Urbana-Champaign, 2004.

[36] S. Ibrahim, M. Mostafa, A. Jnadi, H. Salloum, and P. Osinenko, "Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications," *IEEE Access*, 2024.

[37] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[38] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[39] D. Makoviichuk and V. Makoviychuk, "rl-games: A high-performance framework for reinforcement learning," https://github.com/Denys88/rl_games, May 2021.