2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Reinforcement Learning for Collision-free Flight Exploiting Deep Collision Encoding

Mihir Kulkarni and Kostas Alexis

***Abstract*— This work contributes a novel deep navigation policy that enables collision-free flight of aerial robots based on a modular approach exploiting deep collision encoding and reinforcement learning. The proposed solution builds upon a deep collision encoder that is trained on both simulated and real depth images using supervised learning such that it compresses the high-dimensional depth data to a low-dimensional latent space encoding collision information while accounting for the robot size. This compressed encoding is combined with an estimate of the robot's odometry and the desired target location to train a deep reinforcement learning navigation policy that offers low-latency computation and robust sim2real performance. A set of simulation and experimental studies in diverse environments are conducted and demonstrate the efficiency of the emerged behavior and its resilience in real-life deployments.**

I. Introduction

Aerial robots are tasked to undertake ever more complex missions in demanding environments, including high-risk applications in GPS-denied, cluttered environments such as subterranean exploration [1–3], forest under canopy mapping [4], industrial inspection in confined facilities [5, 6], and search and rescue missions [7]. Key to enabling resilient autonomy is identifying core functionalities that experience significant impediments in their performance and designing novel approaches to overcome such limitations. A prevalent issue especially for small flying robots is the typically-employed separation of collision-free motion planning and control, with the first relying on (online) maps enabling collision checking and the second merely following the commanded paths. As maps can present errors or not capture certain obstacles, this approach is prone to failures as discussed in the framework of challenging deployments like the DARPA Subterranean Challenge [1, 8]. Furthermore, collision-free flight that requires the online reconstruction of maps is bound to involve high-latency operations reducing the maximum possible update rate.

Motivated by the above, in this work we develop a novel navigation policy that learns to enable collision-free flight in confined environments, while solely relying on a single real-time depth observation and an estimate of the robot's odometry. Focusing on robust sim2real transfer, a modularized deep learning solution is proposed. Specifically, the method first exploits a Deep Neural Network (DNN) that serves as a Deep Collision Encoder (DCE) using the high-dimensional real-time depth image data as input and compressing them to a very low-dimensional latent space that retains information for collision building upon the principles of Variational Autoencoders (VAEs). In particular, depth images with resolution equal to 640×480 pixels are compressed to a latent vector with only 64 dimensions ($4800\times$ compression). Given the inference of this latent space in real-time, a Deep Reinforcement Learning (DRL) navigation policy is trained that exploits the latent space and the estimate of the robot's state, alongside information for the target location to enable an aerial robot to fly safely within cluttered environments by commanding reference velocities and yaw rate to the system's low-level autopilot.

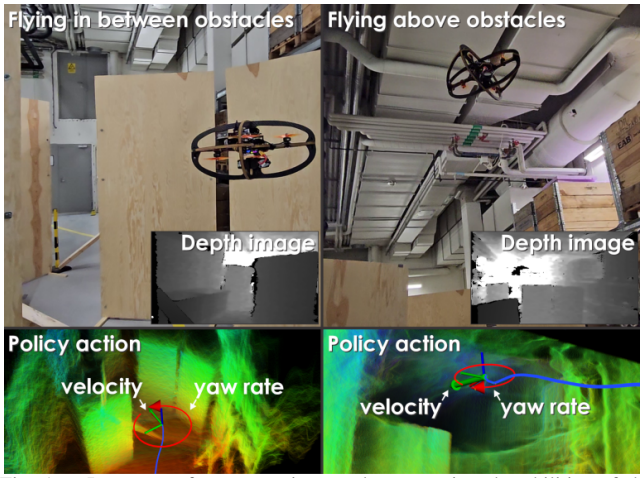

Fig. 1. Instances of two experiments demonstrating the abilities of the navigation policy trained using deep collision encoding and trained with reinforcement learning. If allowed and space is available, the intuitive behavior of flying above all obstacles is selected (right), while when the robot is constrained regarding its altitude it is capable of maneuvering through highly cluttered settings (left).

The proposed deep learning navigation policy boasts a set of contributions. First, through its modular architecture and the DCE it allows to reduce the sim2real gap as on one hand the validity of its low-dimensional latent space, trained with supervised learning, can be verified separately and on the other it allows to assimilate real-life depth image data for training. Second, it represents a DRL policy for aerial robots to fly through diverse cluttered environments without being limited to a particular task and associated environment assumptions or knowledge, such as for example in the recent pioneering work in drone racing [9–11]. Third, it represents a method verified not only in simulation but also on challenging experimental studies such as those in Figure 1, while further ensuring that low-latency inference is achieved onboard. We discuss the emerged behaviors, such as increasing or decreasing speed as necessary to maneuver

This work was supported by the AFOSR Award No. FA8655-21-1-7033 and the Horizon Europe Grant Agreement No. 101070405.

The authors are with the Autonomous Robots Lab, Norwegian University of Science and Technology (NTNU), O. S. Bragstads Plass 2D, 7034, Trondheim, Norway mihir.kulkarni@ntnu.no

around obstacles or flying above them all when possible.

In the remaining paper, Section II presents related work, followed by the problem statement in III. The proposed approach is detailed in Section IV, with evaluation studies in Section V and conclusions in Section VI.

II. Related Work

Deep learning has recently evolved as a focal point of research in collision-free navigation. A subset of this research aims to solve the global planning problem, where a complete map of the environment is available beforehand. These approaches may use top-down images or point clouds of the whole environment to plan paths [12–14]. Contrary to this line of work, here we focus on local navigation exploiting solely onboard observations without access to a global map.

In the domain of local navigation, a set of studies use imitation learning techniques to generate collision-free trajectories, which are then followed by model-based controllers [15, 16]. Methods that learn over different action spaces (velocity/steering angle, acceleration, or angular velocity/thrust commands) have also been explored, for example using a) reinforcement learning [17, 18], b) supervised learning where ground-truth commands are readily available in a driving dataset [19], provided by human operators [20] or demonstrated by an expert [21], as well as c) self-supervised learning [22–24]. In [18], the authors employ an end-to-end approach for training a DRL navigation policy that exploits depth data but train without considering the dynamics of micro aerial vehicles and test in sparse environments.

Further research in local autonomous navigation employs deep learning to create interpretable maps. These maps are then used by classical planners to navigate without collisions [25–28]. Alternatively, some works bypass traditional map representations and directly encode raw sensor data into latent vectors. These vectors are then used to infer control actions, enabling low-latency navigation [12–15, 29–32]. Previous work of the authors has explored the use of supervised deep collision prediction allowing to classify which among a set of motion primitives collide or not with the environment [33, 34]. Focusing on a particular application niche, namely drone racing, the authors in [9] build upon a host of investigations [10, 35–38] and demonstrate beyond-human performance using deep reinforcement learning that exploits the particular structure of the problem (e.g., flying through known types of gates).

III. Problem Formulation

The problem considered in this work is that of autonomous collision-free aerial robot navigation assuming no access to the maps of the environment, neither from offline data nor online reconstruction, and with access only to a) an estimate of the robot's pose, alongside b) the immediate depth observation using a frustum- and range-constrained sensor. Let $\mathcal{I}$, $\mathcal{B}, \mathcal{V}$ be the inertial, body- and vehicle ($\mathcal{I}$ rotated to have the same yaw as the robot) frames respectively, $\mathbf{x}_t$ the current depth image from an onboard sensor, and $\mathbf{s}_t = [\mathbf{p}_t, \mathbf{v}_t, \mathbf{q}_t, \boldsymbol{\omega}_t]$ the estimated robot state consisting of a) its 3D location in $\mathcal{I}$ ($\mathbf{p}_t = [p_{t,x}, p_{t,y}, p_{t,z}]$), b) the 3D velocity in $\mathcal{B}$ ($\mathbf{v}_t = [v_{t,x}, v_{t,y}, v_{t,z}] \in \mathbb{R}^{3\times 1}$), c) the attitude here represented in quaternion form $\mathbf{q}$, and d) the angular velocity in $\mathcal{B}$ ($\boldsymbol{\omega}_t$). Given a 3D goal location $\mathbf{p}^r$ expressed in $\mathcal{I}$, the problem is that of finding an optimized action vector $\mathbf{u}_t = [v^r_{t,x}, v^r_{t,y}, v^r_{t,z}, \omega^r_{t,z}]^T$ (in compact form $\mathbf{u}_t = [\mathbf{v}^r_t, \omega^r_{t,z}]^T$) involving the commanded robot velocities expressed on $\mathcal{B}$ and the yaw rate $\omega^r_{t,z}$ that can enable safe flight avoiding collisions despite the presence of a set of obstacles $\mathcal{S}_O$. This action vector $\mathbf{u}_t$ is then provided to be tracked by a low-level controller of the flying vehicle.

IV. Proposed Approach

The proposed approach on a deep learned modular collision-free navigation policy exploiting deep collision encoding and reinforcement learning is presented.

A. Task-driven Compression for Collision Encoding

At the core of the architecture of the proposed solution is the modularization of the two demanding learning tasks, namely a) processing high-dimensional depth image data to enable collision avoidance, and b) deriving an optimized policy for collision free navigation. In the first step, task-driven depth image compression for collision encoding takes place. In particular, building upon our work in [39], a DNN and specifically the architecture of a VAE is employed with the goal of encoding a high-resolution depth image to a very low dimensional latent space trained to retain the information necessary for collision prediction. This is a key distinction compared to methods that will attempt to train a method end-to-end which does not allow to assimilate real image data in the training in a practical way [33] and only allows to verify a navigation method regarding its ultimate performance with limited ability to ensure that separately verify how the high-dimensional image data are processed in a manner that ensures that collision information is retained.

In further detail, the proposed DCE considers depth images $\mathbf{x}$ that are remapped to derive a "collision image" $\mathbf{x}_{\text{coll}}$ that accounts for the fact that any 3D location captured within the sensor's frustum represents a collision-free point only if the robot with its non-zero size (modeled as a box with dimensions $D_R \times W_R \times H_R$) can fit in the respective 3D location. Through the DCE's remapping-encoding step, followed by its decoding step, depth images are compressed to a low-dimensional latent space $\mathbf{z}$ that gives rise to the reconstructed $\mathbf{x}^{\text{coll}}_{\text{recon}}$ that closely resembles $\mathbf{x}_{\text{coll}}$ as detailed in [39] and summarized in Figure 2. Of particular importance for the DRL navigation policy, the latent dimensions can be of very low dimension as $\mathbf{x}_{\text{coll}}$ is typically less complex than $\mathbf{x}$ owing to the "inflation" by the robot size.

To learn to both compress and remap the depth image $\mathbf{x}$ such that the difference between the reconstructed $\mathbf{x}^{\text{coll}}_{\text{recon}}$ and the collision image $\mathbf{x}_{\text{coll}}$ is minimized, the loss function guiding the supervised training of the DCE takes the form:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \beta_{\text{norm}} \mathcal{L}_{KL}, \tag{1}$$

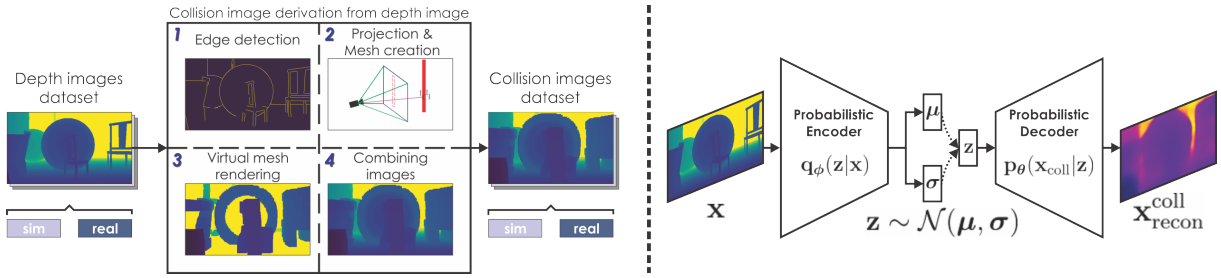

Fig. 2. Overview of the Deep Collision Encoder used to derive a low-dimensional latent space that retains collision information from depth images. The DCE is trained using supervised learning that exploits a dataset involving both synthetic and real depth images. The depth images are transformed to collision images that account for the size of the robot. The involved DNN exploits an architecture motivated by variational autoencoders, while the "encoder" and "decoder" elements are in fact also functioning to encode the depth image to the collision image and its reconstruction from the latent space.

where

$$\mathcal{L}_{\text{recon}}(\mathbf{x}_{\text{coll}}, \mathbf{x}^{\text{coll}}_{\text{recon}}) = \text{MSE}(\mathbf{x}_{\text{coll}}, \mathbf{x}^{\text{coll}}_{\text{recon}}),$$

$$\mathcal{L}_{KL}(\boldsymbol{\mu}, \boldsymbol{\sigma}) = -\frac{1}{2}\sum_{j=1}^{J}\left(1 + \log(\boldsymbol{\sigma}_j^2) - \boldsymbol{\mu}_j^2 - \boldsymbol{\sigma}_j^2\right).$$

Here, $\mathcal{L}$ denotes the overall loss consisting of the reconstruction loss $\mathcal{L}_{\text{recon}}$ and KL-divergence loss $\mathcal{L}_{KL}$, scaled by a constant β_{norm} [40]. These terms are inspired by autoencoder literature [41]. MSE denotes Mean-Square Error loss terms.

It is noted that focusing on sim2real transfer, in this work the method is extended in its ability to assimilate real data from a depth sensor despite the presence of invalid pixels driven by common stereo disparity challenges such as gaps, noise and quantization [42–44]. To enable this goal, the contribution of invalid pixels is removed from the loss terms presented above. Furthermore, Bernoulli sampling is used to determine invalid pixels and noisy depth values are obtained by sampling from a normal distribution.

B. Navigation Policy Learning

We employ a DRL framework to train neural network policies to navigate cluttered environments. We formulate this problem as a partially observable Markov decision process (POMDP) with the true state of the environment, robot and the previous action denoted by $\mathfrak{s}_t$ at a discrete time t. An action $\mathbf{a}_t \in \mathbf{A}$ can be applied to the environment to obtain a new state at the next time step $t+1$ with the transition probability $\mathbf{Pr}(\mathfrak{s}_{t+1}|\mathfrak{s}_t, \mathbf{a}_t)$. At each step, the agent observes the environment with the probability $\mathbf{O}(\mathbf{o}_{t+1}|\mathfrak{s}_{t+1})$. We utilize the robot state $\mathbf{s}_t$, goal position $\mathbf{p}^r$, and the depth image $\mathbf{x}_t$ to compute the observations that the agent gathers from the environment. The observations for the agent are defined as $\mathbf{o}_t = [\hat{\mathbf{n}}^g_t, ||\mathbf{n}^g_t||_2, \mathbf{v}_t, \phi_t, \theta_t, \boldsymbol{\omega}_t, \mathbf{a}_{t-1}, \mathbf{z}_t]$, where $\hat{\mathbf{n}}^g_t$ denotes the unit vector from the robot position $\mathbf{p}_t$ to the goal position $\mathbf{p}^r$ expressed in the vehicle frame $\mathcal{V}$, $||\mathbf{n}^g_t||_2$ is the Euclidean distance between the robot and the goal, $\mathbf{v}_t$ and $\boldsymbol{\omega}_t$ are the linear and angular velocities respectively expressed in the body frame $\mathcal{B}$. ϕ_t and θ_t correspond to the current roll and pitch angles of the robot and $\mathbf{a}_{t-1}$ is the vector of actions obtained from the network at time $t-1$. Finally, $\mathbf{z}_t$ denotes the compressed latent representation of image $\mathbf{x}_t$ obtained from the DCE.

The agent also maintains a belief over the states where $\mathbf{B}_t(\mathfrak{s}_t)$ denotes the probability that the environment/robot is in state $\mathfrak{s}_t$. The belief is updated based on the current observation, the current action and the previous belief state as $\mathbf{B}_{t+1} = \boldsymbol{\tau}(\mathbf{B}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$. For each state transition a reward is provided to the agent in the form of $\mathbf{R}(\mathfrak{s}_{t+1}|\mathfrak{s}_t, \mathbf{a}_t)$. A policy that yields actions given observations $\mathbf{a}_t = \boldsymbol{\pi}(\mathbf{o}_t)$ is learned by the agent such that the sum of rewards over an episode is maximized. The reward function for the state transitions is defined as:

$$\mathbf{R}(\mathfrak{s}_{t+1}|\mathfrak{s}_t, \mathbf{a}_t) = \sum_{i=1}^{4}\lambda_i r_i + \sum_{j=1}^{2}\eta_j p_j + p_{\text{crash}}, \qquad (2)$$

where r_i indicates a positive reward term and p_i indicates a negative reward term, defined as:

$$\begin{aligned}
r_1 &= r(||\mathbf{n}^g_t||_2, \nu_1),\\
r_2 &= r(||\mathbf{n}^g_t||_2, \nu_2),\\
r_3 &= \frac{|\nu_3 - ||\mathbf{n}^g_t||_2|}{\nu_3},\\
r_4 &= \nu_4(||\mathbf{n}^g_t||_2 - ||\mathbf{n}^g_{t-1}||_2),\\
p_1 &= \sum_k \nu_{5,k}(r(g(\mathbf{a}_t)_k, \nu_{6,k}) - 1),\\
p_2 &= \sum_k \nu_{7,k}(r(g(\mathbf{a}_t)_k - g(\mathbf{a}_{t-1})_k, \nu_{8,k}) - 1),\\
p_{\text{crash}} &= -\nu_9.
\end{aligned}$$

The function r is defined as $r(x, \nu) = e^{-\frac{x^2}{\nu}}$. The values $\lambda_i, \eta_j > 0$ and $\nu_i, \nu_{m,k} > 0$ represent tuning parameters. $\nu_1 \dots \nu_4$ and ν_9 are scalars while $\boldsymbol{\nu}_5 \dots \boldsymbol{\nu}_8$ are vectors with the same number of dimensions as the output of function g as defined in the next subsection. These values may vary during training as discussed later in this Section. With this environment definition, we utilize the Asynchronous Proximal Policy Optimization (APPO) algorithm from Sample Factory [45] to train a deep neural network policy to navigate a robot to the goal location in a collision-free manner.

C. Implementation Details

We define a neural network architecture containing 3 fully-connected layers consisting of 512, 256 and 64 neurons each with an ELU activation layer, followed by a GRU with a hidden layer size of 64. Given an observation vector $\mathbf{o}_t$, the policy outputs a 3-dimensional action command $\mathbf{a}_t = [a_{t,1}, a_{t,2}, a_{t,3}]$ with values in [-1, 1]. These action values are

mapped to the speed, inclination of the commanded velocity with the x-axis of the robot and yaw-rate. A function g is defined such that $\mathbf{u}_t = g(\mathbf{a_t})$, to convert these values to the input command for the velocity controller $\mathbf{u}_t = [v^r_{t,x}, v^r_{t,y}, v^r_{t,z}, \omega^r_{t,z}]^T$. The action outputs are converted to velocity and yaw-rate command as:

$$
\begin{aligned}
v^r_{t,x} &= s_{\max}(\frac{a_{t,1}+1}{2}\cos(i_{\max}a_{t,2})),\\
v^r_{t,y} &= 0.0,\\
v^r_{t,z} &= s_{\max}(\frac{a_{t,1}+1}{2}\sin(i_{\max}a_{t,2})),\\
\omega^r_{t,z} &= \omega_{\max}a_{t,3},
\end{aligned}
$$

where $s_{\max}$ is the maximum speed and $i_{\max}$ is the maximum angle of the commanded velocity with the x-axis of the robot, while $\omega_{\max}$ is the maximum commanded yaw-rate. This parameterization of the controller commands and s_{max}, i_{max}, ω_{max} are chosen to ensure that commanded velocity vector lies within the field-of-view of the depth sensors and prevent a sideways collision with the environment. The neural network is trained with an adaptive learning rate initialized at $l_r = 10^{-4}$. The discount factor is set to $\gamma = 0.98$. The neural network is trained with 1024 environments simulated in parallel with an average time step of 0.1s and rollout buffer size set to 32. We train this policy for approximately 26×10^6 environment steps aggregated over all agents.

The DCE is pre-trained and frozen during the training of the RL policy. For training the DCE network, collision images are generated for $10,000$ real images and $21,000$ simulated images. The network architecture and training methodology followed is similar to [39], with $\beta_{\text{norm}} = 3.0$. Furthermore, in this work, we utilize the sampled latent representation $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\sigma})$ for training the RL policy.

D. Training Environment

The Aerial Gym Simulator [46] provides the environment and the interfaces to train the DRL policy to navigate cluttered environments. The DCE and the learning framework are interfaced with the simulator as shown in Figure 3. The simulator provides capabilities for massively parallelized simulation of aerial robots that are equipped with depth cameras. The provided velocity controller is utilized for this work. We generate cluttered environments within the simulator consisting of a room-like environment bounded by walls containing static obstacles that are positioned and oriented in a randomized manner by uniformly sampling the position and Euler angles for each obstacles. These obstacles are kept floating in the environment (i.e., the effect of gravity is disabled on these objects) to allow more randomization in the training environments. While the length of an episode is fixed, each environment can run episodes asynchronously allowing different environments to provide the learning agent experience from different stages of the task simultaneously.

A curriculum learning framework is set up that gradually complexifies the environment by adding more objects to it as the agent learns to successfully navigate the robot in

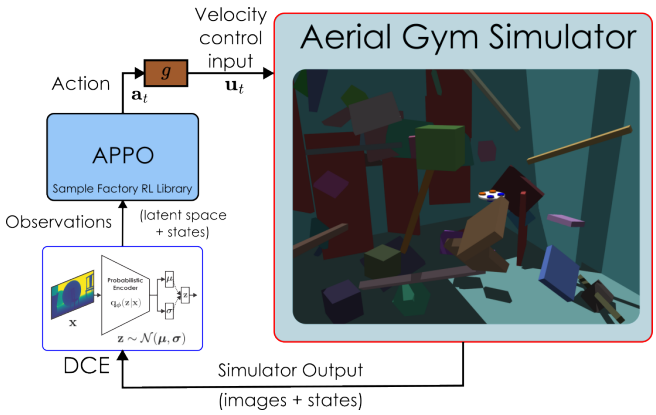

Fig. 3. Overview of the interface between the framework for RL agent training, the Aerial Gym Simulator and the DCE.

easier environments (with success rate greater than 70%), while making the environment easier if robots are crashing at a rate above a given threshold (here 30%). We set up a curriculum that involves logging the runs of the robots to measure successes, crashes and episode timeouts. Successful runs are said to happen when the robot is within a specified distance of the goal location (here 1 m) at the end of an episode. The start and goal locations are randomly sampled at opposite ends of the environment for each episode. Crashes are defined as instances where a robot is in contact with another object in the environment, this environment is then immediately reset. A timeout is said to happen when the robot has remained collision-free till the end of an episode but has not reached the goal location. The parameters of the reward function are varied linearly as the curriculum level increases such that $\lambda^n_i = \kappa_i n$, and $\eta^n_i = \xi_i n$, where λ^n_i and η^n_i are the values of λ_i and η_i at curriculum level n, while κ_i and ξ_i are positive constants.

To make the network robust against real-world uncertainty, random forces and torques are applied to the simulated robot at discrete time instances sampled from a Bernoulli distribution. Additionally, the position and the orientation of the camera are perturbed by small values (between ± 5 cm and ± 3 deg). The observations are also perturbed by small values to approximately simulate the uncertainty from real-world sensor measurements. To enable fast learning, we limit the image capture rate to approximately 10 Hz, while the physics simulation occurs at 100 Hz in simulated time. To simulate real-world sensor latency, we vary the number of timesteps simulated by the physics engine between two sensor measurements. We add Gaussian noise to the simulated depth images by sampling from a distribution with the standard deviation linearly dependent on the depth value of the pixel. Importantly, in Aerial Gym the dynamics of the simulated agent is matched with those of the real multirotor vehicle. Moreover, the parameters of the velocity controller are randomized to vary the step-response time constant of the robot by $\pm 10\%$.

V. Evaluation Studies

The proposed approach is extensively evaluated both in simulation and experimental studies.

A. Simulation Studies

Two sets of simulation studies are conducted to evaluate the method prior to experimental deployment. First, an extensive set of results are derived using the Isaac Gym-powered Aerial Gym simulator. Second, to test the method in a different simulation environment and further deploy it in virtual environments that are particularly different from those experienced in training some indicative results are recorded using Flightmare [47].

1) Aerial Gym-based Evaluation: The method was first evaluated in Aerial Gym. Specifically, considering 30 different curriculum levels, representing different levels of complexity in terms of obstacle clutter, the policy was tested for 500 runs per curriculum level. Each of the tests involved a box-shaped environment with dimensions $L \times W \times H$ within the set $[8, 12] \times [5, 8], \times [4, 6]$. The curriculum level n implies the number of obstacles in the environment (i.e., curriculum level 0 implies 0 obstacles, and curriculum level 30 implies 30 obstacles). From curriculum level 0 to level 5 the obstacles are large panels, while after the 5-th level increasing the level complexity is done through small obstacles employing primitive shapes such as boxes. Figure 4 presents indicative environments from different curriculum levels, while Table I summarizes the performance the policy is achieving across complexity levels. The maximum speed and yaw-rate were set to $s_{\max} = 1.5$m/s and $\omega_{\max} = 60$ deg/s. For environments up to a certain amount of obstacles, the performance is high but it naturally drops in settings with higher clutter.

TABLE I

EVALUATION OF THE TRAINED POLICY AGAINST ENVIRONMENTS OF DIFFERENT COMPLEXITY.

Level	Success %	Timeout %	Crash %
0	99.4	0.5	0.0
5	92.8	2.1	5.0
10	90.6	2.7	6.6
15	86.3	3.7	9.9
20	78.5	4.4	17.0
25	72.7	8.1	19.1
30	70.8	7.2	21.8

Fig. 4. Indicative simulation studies using the Aerial Gym simulator to evaluate the trained policy against increasingly complex environments.

2) Case Studies in Flightmare: Two case studies were ran in Flightmare and presented in Figure 5. Specifically, the method is deployed using the Hummingbird aerial vehicle model ferrying a depth sensor as described in [15] and deployed inside a) a forest, with density governed by a Poisson disc radius of 4m and b) an environment with cylinder and cube objects distributed according to a poisson radius of 6m. It is highlighted that neither type of objects, and especially trees were experienced during the training of the policy or of the DCE. For both tests the robot is flying is a maximum speed of $s_{\max} = 1.5$m/s and $\omega_{\max} = 60$ deg/s.

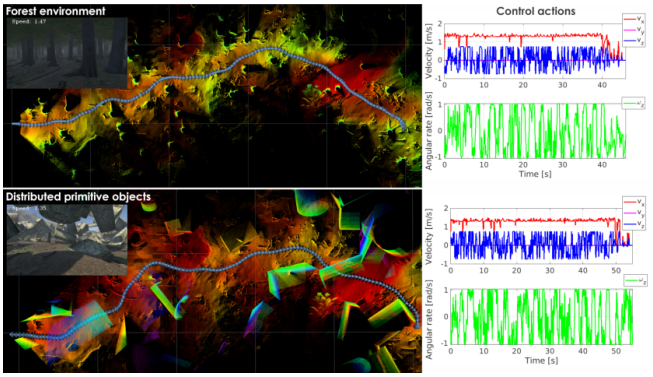

Fig. 5. Simulation studies using the Flightmare simulator with the goal of evaluating the method given environment diversity –compared to training data– especially in the case of the forest. On the right the commanded velocities (blue v_z, magenta v_y which is zero and v_x and yaw rate ω_z (green) are shown.

B. Experimental Studies

The proposed approach was evaluated in experiments involving navigation through cluttered environments with the goal of assessing the overall performance and especially how the sim2real gap is handled. Of special interest was to assess the emerging behaviors including the robot opting to fly above all obstacles when allowed and possible in the environment, alongside its ability to maneuver through clutter when necessary.

1) System Overview: To evaluate the method, we utilize a quadrotor called Learning-based Micro Flyer (LMF), evolved out of the works in [48] and [33, 34]. The robot has a diameter of 0.43m and weighs 1.2kg. It features a Realsense D455 for depth and RGB data at 640×480 resolution and 15 FPS, a PixRacer Ardupilot-based autopilot for velocity and yaw-rate control, and a Realsense T265 fused with the autopilot's Inertial Measurement Unit (IMU) for acquiring the robot's odometry state estimate. The system integrates an NVIDIA Orin NX in which the proposed method is executed exploiting its GPU. The platform is depicted in Figure 6.

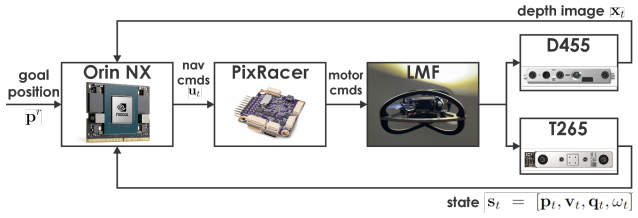

Fig. 6. Block diagram of the LMF robot. The proposed method is implemented onboard the Orin NX processor, while low-level commands are tracked by the embedded autopilot. A Realsense D455 sensors is providing the depth images $\mathbf{o}_t$, while a T265 camera offers odometry estimates $\mathbf{s}_t$.

2) Flying through or above cluttered settings: First, the system is deployed in a cluttered environment, depicted in Figure 7, and commanded to fly to a waypoint that is 15m forward compared to the starting location, 3m to the left and at a height of 1m. Two variations of the experiment are ran, namely a) one in which the vertical velocities of

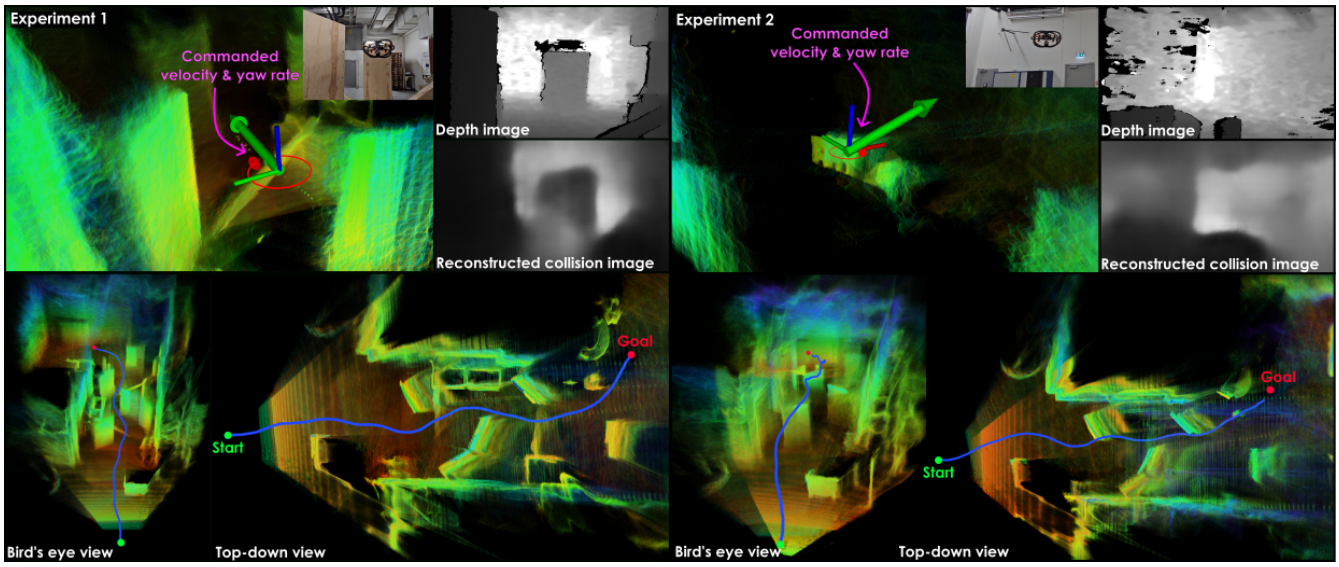

Fig. 7. Experiments in a cluttered corridor where the vertical velocity of the robot is constrained (in Experiment 1) and matched with training value (in Experiment 2) show that the robot is able to negotiate cluttered environments both passing safely through them when the vertical velocity is constrained, and going over them when possible along the shorter path to the goal location, exploiting the free space above the obstacles. The reconstructions from the DCE show that the encoder network is robust to noise and imperfect depth data from real-world sensors.

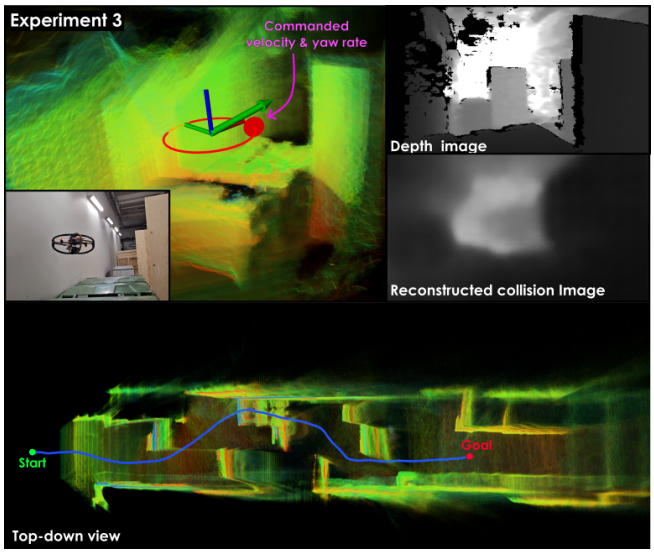

Fig. 8. Experimental evaluation in a longer cluttered corridor shows that the policy guides the robot safely to the goal location. The robot takes a more intuitive (and shorter) path to the destination choosing to travel through a more narrow opening. The method is robust to the imperfect depth reconstructions owing to the light sources in the environment as seen in the reconstructed collision image.

the robot are constrained (with $i_{\max} = 7.5$ deg) to prevent its ability to fly above the obstacles, and b) one in which the vertical velocities constraint is removed (with $i_{\max} = 30$ deg). As shown in Figure 7 the robot is able to negotiate the environment in both cases. When allowed to exploit its vertical navigation capabilities it acts in the intuitive manner of flying above all obstacles, while when it can only fly by maneuvering around all obstacles it accelerates and decelerates as necessary to achieve so. Importantly, this environment is denser than those encountered in training, so the maximum allowed speed and yaw-rate are set to $s_{\max} = 1.2$m/s and $\omega_{\max} = 40$ deg/s for both experiments.

3) Maneuvering in a cluttered corridor: Finally, we deploy the system in long corridor cluttered with obstacles depicted in Figure 8. The robot is commanded to reach a location that is 20m forward and at a height of 1m. Similar to the previous experiment, we limit the velocity of the robot to 1.2m/s, the maximum yaw-rate to $\omega_{\max} = 40$ deg/s and $i_{\max} = 7.5$ deg. The robot navigates the clutter and reaches the goal location.

The depth images during these experiments contained sensor noise and imperfect depth estimation resulting in invalid pixels. However, the reconstructions show that the DCE is robust to such noise and the encoded latent representation does not affect the performance of the RL policy that has been trained only in simulation using simulated depth images. Furthermore, the obstacles are novel - compared to the training set - which is another testament for its robust sim2real transfer primarily attributed to the modularized architecture and the low-dimensional encoding of collision information by the DCE. Finally, the proposed method has an inference time of 15 ms on the NVIDIA Orin NX board without any optimizations for accelerating inference.

VI. Conclusions

This paper presented a DRL-based navigation policy trained completely in simulation, that exploits highly compressed latent representation of depth images obtained from a deep collision encoder network that is trained on both simulated and real data. Extensive evaluation studies conducted in simulation against different obstacle densities and also in different environments show that the method is robust to unseen environments and variations in robot dynamics. Furthermore, real-world experiments show that the method is robust to sensor noise and can navigate the robot safely in cluttered environments.

REFERENCES

[1] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, and K. Alexis, "Cerberus in the darpa subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[2] M. Tranzatto, M. Dharmadhikari, L. Bernreiter, M. Camurri, S. Khattak, F. Mascarich, P. Pfreundschuh, D. Wisth, S. Zimmermann, M. Kulkarni *et al.*, "Team cerberus wins the darpa subterranean challenge: Technical overview and lessons learned," *arXiv preprint arXiv:2207.04914*, 2022.

[3] M. Tranzatto, F. Mascarich, L. Bernreiter, C. Godinho, M. Camurri, S. M. K. Khattak, T. Dang, V. Reijgwart, J. Loeje, D. Wisth *et al.*, "Cerberus: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the darpa subterranean challenge," *Field Robotics*, 2021.

[4] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu *et al.*, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022.

[5] 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.

[6] 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.

[7] 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.

[8] K. Ebadi, L. Bernreiter, H. Biggie, G. Catt, Y. Chang, A. Chatterjee, C. Denniston, S.-P. Deschênes, K. Harlow, S. Khattak, L. Nogueira, M. Palieri, P. Petracek, M. Petrlik, A. Reinke, V. Kratky, S. Zhao, A. akbar Agha-mohammadi, K. Alexis, C. Heckman, K. Khosoussi, N. Kottege, B. Morrell, M. Hutter, F. Pauling, F. Pomerleau, M. Saska, S. A. Scherer, R. Y. Siegwart, J. L. Williams, and L. Carlone, "Present and future of slam in extreme underground environments," *arXiv preprint arXiv:2208.01787*, 2022.

[9] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[10] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1205–1212.

[11] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning | Science Robotics," *Science Robotics*, vol. 8, no. 82, p. adg1462, 2023.

[12] 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.

[13] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, "Universal planning networks: Learning generalizable representations for visuomotor control," in *35th International Conference on Machine Learning (ICML)*, vol. 80, 2018, pp. 4732–4741.

[14] 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, 2021.

[15] 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.

[16] 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.

[17] A. Francis, A. Faust, H.-T. L. Chiang, J. Hsu, J. C. Kew, M. Fiser, and T.-W. E. Lee, "Long-range indoor navigation with prm-rl," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1115–1134, 2020.

[18] 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.

[19] 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.

[20] D. Shah and S. Levine, "ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints," in *Robotics: Science and Systems (RSS)*, 2022.

[21] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," in *Robotics: Science and Systems (RSS)*, 2020.

[22] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3948–3955.

[23] G. Kahn, P. Abbeel, and S. Levine, "Badgr: An autonomous self-supervised learning-based navigation system," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312–1319, 2021.

[24] G. Kahn, P. Abbeel, and S. Levine, "Land: Learning to navigate from disengagements," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1872–1879, 2021.

[25] L. Wang, H. Ye, Q. Wang, Y. Gao, C. Xu, and F. Gao, "Learning-based 3d occupancy prediction for autonomous navigation in occluded environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4509–4516.

[26] J. Frey, D. Hoeller, S. Khattak, and M. Hutter, "Locomotion policy guided traversability learning using volumetric representations of complex environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 5722–5729.

[27] M. G. Castro, S. Triest, W. Wang, J. M. Gregory, F. Sanchez, J. G. Rogers, and S. Scherer, "How does it feel? self-supervised costmap learning for off-road vehicle traversability," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 931–938.

[28] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8652–8661.

[29] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter, "Learning a state representation and navigation in cluttered and dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5081–5088, 2021.

[30] D. Dugas, J. Nieto, R. Siegwart, and J. J. Chung, "Navrep: Unsupervised representations for reinforcement learning of robot navigation in dynamic human environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, p. 7829–7835.

[31] R. B. Grando, J. C. de Jesus, and P. L. Drews-Jr, "Deep reinforcement learning for mapless navigation of unmanned aerial vehicles," in *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*. IEEE, 2020, pp. 1–6.

[32] R. B. Grando, J. C. de Jesus, V. A. Kich, A. H. Kolling, and P. L. J. Drews-Jr, "Double critic deep reinforcement learning for mapless 3d navigation of unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 104, no. 2, p. 29, 2022.

[33] H. Nguyen, S. H. Fyhn, P. De Petris, and K. Alexis, "Motion primitives-based navigation planning using deep collision prediction," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 9660–9667.

[34] M. Kulkarni, H. Nguyen, and K. Alexis, "Semantically-enhanced deep collision prediction for autonomous navigation using aerial robots," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 3056–3063.

[35] C. D. Wagter, F. Paredes-Vallés, N. Sheth, and G. de Croon, "Artificial intelligence behind the winning entry to the 2019 AI Robotic Racing Competition," *arXiv preprint arXiv:2109.14985*, 2021.

[36] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: autonomous drone racing," *Autonomous Robots*, pp. 307–320, 2022.

[37] 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.

[38] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Beauty and the beast: Optimal methods meet learning for drone racing," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 690–696.

[39] M. Kulkarni and K. Alexis, "Task-driven compression for collision encoding based on depth images," in *Advances in Visual Computing*. Cham: Springer Nature Switzerland, 2023, pp. 259–273.

[40] 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. [Online]. Available: https://openreview.net/forum?id=Sy2fzU9gl

[41] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[42] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *2012 second international conference on 3D imaging, modeling, processing, visualization & transmission*. IEEE, 2012, pp. 524–530.

[43] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart, "Kinect v2 for mobile robot navigation: Evaluation and modeling," in *2015 international conference on advanced robotics (ICAR)*. IEEE, 2015, pp. 388–394.

[44] O. Korkalo and T. Takala, "Measurement noise model for depth camera-based people tracking," *Sensors*, vol. 21, no. 13, p. 4488, 2021.

[45] A. Petrenko, Z. Huang, T. Kumar, G. Sukhatme, and V. Koltun, "Sample factory: Egocentric 3d control from pixels at 100000 fps with asynchronous reinforcement learning," 2020.

[46] M. Kulkarni, T. J. L. Forgaard, and K. Alexis, "Aerial gym – isaac gym simulator for aerial robots," 2023.

[47] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *4th Conference on Robot Learning (CoRL)*, 2020, pp. 1147–1157.

[48] P. De Petris, H. Nguyen, T. Dang, F. Mascarich, and K. Alexis, "Collision-tolerant autonomous navigation through manhole-sized confined environments," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2020, pp. 84–89.