This paper has been accepted for publication at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, 2023. ©IEEE

Learning Deep Sensorimotor Policies for Vision-based Autonomous Drone Racing

Jiawei Fu, Yunlong Song, Yan Wu, Fisher Yu, and Davide Scaramuzza

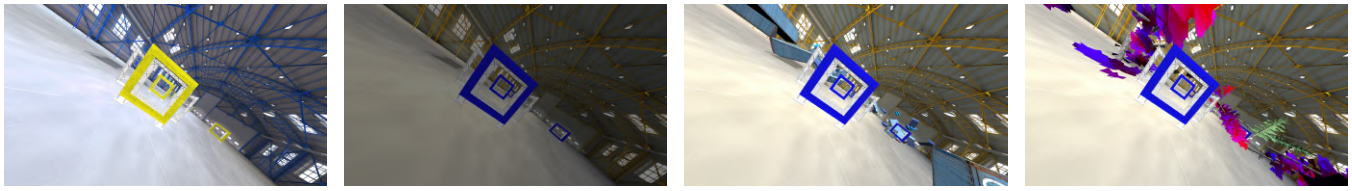


Fig. 1: This work explores learning deep sensorimotor policies for vision-based autonomous drone racing. The key is to leverage contrastive learning and data augmentation to obtain feature representations that are robust against unseen visual disturbances, including modifications in hue or brightness and the addition of blue boxes or random objects.

***Abstract*— The development of effective vision-based algorithms has been a significant challenge in achieving autonomous drones, which promise to offer immense potential for many real-world applications. This paper investigates learning deep sensorimotor policies for vision-based drone racing, which is a particularly demanding setting for testing the limits of an algorithm. Our method combines feature representation learning to extract task-relevant feature representations from high-dimensional image inputs with a learning-by-cheating framework to train a deep sensorimotor policy for vision-based drone racing. This approach eliminates the need for globally-consistent state estimation, trajectory planning, and handcrafted control design, allowing the policy to directly infer control commands from raw images, similar to human pilots. We conduct experiments using a realistic simulator and show that our vision-based policy can achieve state-of-the-art racing performance while being robust against unseen visual disturbances. Our study suggests that consistent feature embeddings are essential for achieving robust control performance in the presence of visual disturbances. The key to acquiring consistent feature embeddings is utilizing contrastive learning along with data augmentation.**

Video: `https://youtu.be/AX_fcnW9yqE`

I. INTRODUCTION

Vision-based autonomous drone racing is a challenging navigation task that demands the vehicle to operate at the edge of the vehicle limits. High speeds and quick rotations of the camera induce motion blur and rapid illumination changes, adding to the difficulty of the task. To avoid crashing, the system must tolerate very few mistakes, and any small error can be catastrophic. Consequently, even minor deviations from the desired path can significantly reduce task performance, making drone racing especially difficult.

Existing works on vision-based autonomous drone racing generally rely on globally consistent state estimation and trajectory planning [1]–[5]. Methods of this kind rely heavily on the quality of visual-inertial odometry, which is known to be very sensitive during high-speed flight [6]. In contrast, human pilots operate drones purely based on a video stream from the vehicle's onboard camera, that is, by directly mapping visual input to control commands without performing any explicit state estimation or trajectory planning. Human pilots can control the drone at its physical limit while being robust against environmental changes. This is primarily because they can select task-relevant visual information effectively [7]. Thus, we ask: how can we develop deep sensorimotor policies that emulate human pilots?

Several studies [8]–[10] have investigated the effectiveness of end-to-end sensorimotor policy learning for self-driving vehicles utilizing a convolutional neural network that directly outputs the steering angle via the front view image. However, the challenges in vision-based drone racing are distinct from those encountered in the aforementioned studies. Unlike autonomous driving, where the agent needs to adapt to different or unknown environments, the objective of drone racing is to achieve low lap time in fixed and known environments. Hence the major challenges in our task are (i) learning a consistent visual representation and (ii) training a high-performance control policy.

This work leverages contrastive learning and data augmentation to train a perception network that can extract useful feature representations from high-dimensional images. Using contrastive learning, the perception network learns to focus on useful visual features while ignoring irrelevant backgrounds. In the drone racing context, the gate is the most

J. Fu, Y. Song, and D. Scaramuzza are with the Robotics and Perception Group, Department of Informatics, University of Zurich, and Department of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland (`http://rpg.ifi.uzh.ch`). Y. Wu and F. Yu are with Visual Intelligence and Systems Group in the Computer Vision Lab at ETH Zurich. This work was supported by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics, the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 871479 (AERIAL-CORE), and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

relevant visual information for the task. Thus, in the feature space, representations are closely clustered when the robot is at the same place and widely dispersed when the robot visits different locations.

Given obtained feature representations, a vision-based control policy is trained using a privileged learning-by-cheating framework. Using reinforcement learning (RL), we first train a privileged state-based policy that can maximize its progress along the race track. Then, we use imitation learning to distill its knowledge into a vision-based student policy that does not rely on privileged state information, including the ground truth position of the vehicle and its target gate location.

Our experiments, conducted in a realistic simulator [11], show that our vision-based deep sensorimotor policy achieves the same level of racing performance as state-based policies while being resilient against unseen visual disturbances and distractors. We benchmark the performance of our vision-based policy against the time-optimal trajectory generation algorithm [12], which offers a theoretical minimum time. Our policy achieves lap time close to the time-optimal solution.

In conclusion, our results, albeit simulation only, suggest that end-to-end policies represent a promising approach for vision-based drone racing in challenging scenarios. Furthermore, to enhance the robustness of the vision-based policy, it is crucial to have consistent feature representations, which can be obtained by utilizing contrastive learning along with data augmentation. Contrastive learning enables the network to learn a representation that emphasizes the similarities between different image observations. Data augmentation further enhances this by generating additional training data with random visual disturbances.

II. RELATED WORK

Different approaches have been studied to tackle autonomous drone racing. State-based methods that rely on globally accurate position information have been used extensively. Foehn et al. [12] presented the time-optimal trajectory generation by jointly optimizing the time allocation and the trajectory. Combining with a model predictive controller, the algorithm enabled them to outperform human experts in drone racing. In [13]–[15], authors used RL to train a neural network as the policy. For example, Song et al. [14] utilized model-free RL to optimize a neural network policy that maps ground truth state to control command directly. Nagami et al. [15] initialized a network by mimicking a simplified controller and further trained it with RL. The hierarchy allowed the policy to outperform a trajectory planning policy. Although promising results can be achieved using state-based control, the assumption of accurate position information requires estimating the vehicle state and the gate pose, both are challenging during high-speed flight.

Prior work on vision-based drone racing decouples the perception, planning, and control modules. In the work of Foehn et al. [1], visual-inertial odometry (VIO) was fused with a CNN-based gate corner detection for robust state estimation. A receding horizon path planner generates a time-optimal trajectory using motion primitives based on a point-mass model of the drone platform. However, the point-mass assumption cannot represent the true actuation limits of the drone and may lead to dynamically infeasible trajectories. In [16]–[18], authors first use data-driven methods to train the neural networks that can predict the waypoint and the desired speed. Afterward, a minimum jerk trajectory is planned for passing through the waypoint and then tracked by a low-level controller. Muller et al. [19] propose to train a neural network for local trajectory planning, in which a downstream control policy is used to track the trajectory and generate low-level commands for vehicle control. The trajectory labeling requires additional engineering efforts and can result in ambiguity as each image can be labeled with different trajectories. The decoupling of the perception, planning, and control modules inevitably involves simplified assumptions and compounding errors, leading to sub-optimal control performance.

III. METHODOLOGY

An overview of our method is given in Fig. 2. Our approach consists of two key components: feature representation learning and policy training.

A. Task Definition

The drone racing problem can be formulated as an optimization problem, in which the optimization objective is to minimize the time for passing through a sequence of gates [14]. The drone perceives the environment via a single camera mounted on the nose of the drone and relies on an inertial measurement unit (IMU) for measuring the vehicle's orientation, velocity, and acceleration.

Due to the limited field of view of the camera, the environment is only partially observable. Our goal is to develop an autonomous system that can push the vehicle to its maximum performance while purely relying on the onboard camera and IMU.

B. Feature Representation Learning

In vision-based drone racing, the algorithm relies solely on visual feedback from an RGB camera attached to the drone to perceive the environment. The image captured by the camera provides crucial task information, including the target gate pose and the background information, such as obstacles and landmarks. We use YOLO(v5) [20] as the backbone to detect the gate since it offers state-of-the-art performance in object detection and can achieve very fast inference speed. An average pooling layer is appended to the YOLO's detection head to extract low-dimensional features. To enhance the robustness of the feature extractor and counteract visual disturbances, we use contrastive learning and data augmentation.

We use the architecture introduced in [21] for feature representation learning (Fig. 2). Given an image observation $\boldsymbol{i}$, two different augmentations are used to produce two augmented images $\boldsymbol{i}^{\text{Aug}}$ and $\boldsymbol{i}^{\text{Aug}'}$. Then, both augmented

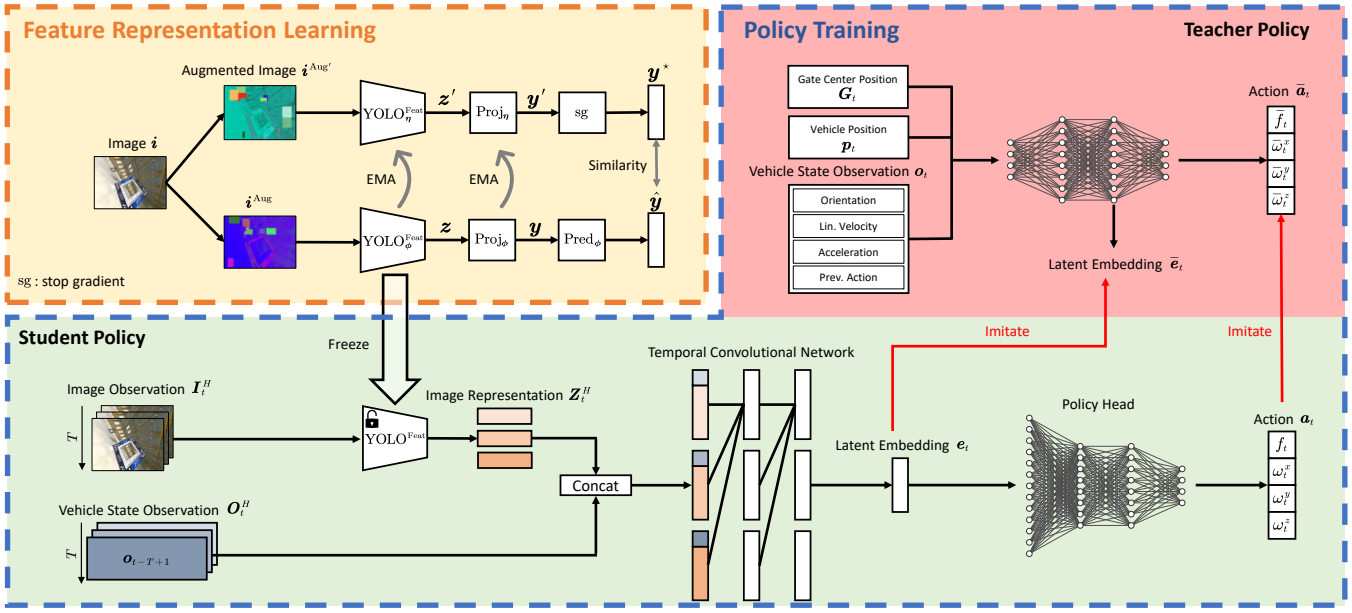


Fig. 2: Our approach consists of two key components: feature representation learning via contrastive learning and policy training via learning-by-cheating.

images (i^{Aug} and $i^{\text{Aug}'}$) are processed by two YOLO feature extractors (YOLO$_\phi^{\text{Feat}}$ and YOLO$_\eta^{\text{Feat}}$) to obtain low-dimensional features (z and z'), which are further processed via projections Proj$_\phi$ and Proj$_\eta$ to get $\boldsymbol{y}$ and $\boldsymbol{y'}$ separately. We then output a prediction of $\hat{\boldsymbol{y}}$ for image i^{Aug} via the predictor Pred$_\phi$. The representation $\boldsymbol{y}^\star$ is the direct output of the Proj$_\eta$. Here, sg means stop gradient. A cosine similarity loss is used to align the representations

$$\mathcal{L}^{cos}(\phi) = -\frac{< \hat{\boldsymbol{y}}, \boldsymbol{y}^\star >}{||\hat{\boldsymbol{y}}||_2 \cdot ||\boldsymbol{y}^\star||_2} \tag{1}$$

where $< \cdot, \cdot >$ denotes dot product between vectors.

Minimizing the loss $\mathcal{L}^{cos}(\phi)$ is equivalent to maximizing the mutual information retained between $\boldsymbol{y}$ and $\boldsymbol{y'}$ [22]. Thus, YOLO$_\phi^{\text{Feat}}$ learns to extract the representation of gates that are shared across augmented images and ignore unrelated information, e.g., distractors. The target network YOLO$_\eta^{\text{Feat}}$ is updated via an exponential moving average (EMA) of YOLO$_\phi^{\text{Feat}}$. After training, YOLO$_\phi^{\text{Feat}}$ is freezed and used as the feature extractor YOLO$^{\text{Feat}}$ in the student policy.

C. Policy Training

Teacher Policy Training: The first step is to obtain a state-based teacher policy that can push the vehicle to its maximum racing performance. We use RL to train a multi-layer perceptron (MLP) policy π_{teacher} for passing through a sequence of gates in minimum time [14]. The teacher policy has access to all ground truth information, including the gate state $\boldsymbol{G}_t$, the vehicle position $\boldsymbol{p}_t$, and other vehicle states $\boldsymbol{o}_t$ which includes orientation, linear velocity, acceleration, and previous action. Here, the gate state $\boldsymbol{G}_t$ contains the next two gate center positions $\boldsymbol{g}_t^1$ and $\boldsymbol{g}_t^2$. We separate position $\boldsymbol{p}_t$ from other states $\boldsymbol{o}_t$ for the brevity of notation; because the student policy introduced in the following section does not have the position $\boldsymbol{p}_t$ information. The policy maps the ground truth states directly to control command in the form of mass-normalized collective thrust and angular velocity: $\pi_{\text{teacher}}(\boldsymbol{G}_t, \boldsymbol{p}_t, \boldsymbol{o}_t) = \bar{\boldsymbol{a}}_t$, where $\bar{\boldsymbol{a}}_t = (\bar{f}_t, \bar{\boldsymbol{\omega}}_t)$, $\bar{f}_t$ is the mass normalized thrust, and $\bar{\boldsymbol{\omega}}_t = (\bar{\omega}_t^x, \bar{\omega}_t^y, \bar{\omega}_t^z)$ is the angular velocity.

The main objective in drone racing is to minimize lap time, which is equivalent to maximizing the path progress toward the target gate center. Similar to [14], at each simulation time step t, the agent receives a progress reward defined as $r_t^{\text{prog}} = \|\boldsymbol{g}_t^1 - \boldsymbol{p}_{t-1}\|_2 - \|\boldsymbol{g}_t^1 - \boldsymbol{p}_t\|_2 - \lambda\|\boldsymbol{\omega}_t\|_2$, where $\boldsymbol{p}_t$ and $\boldsymbol{p}_{t-1}$ are the vehicle positions at the current and previous time steps, respectively. Here, $\lambda\|\boldsymbol{\omega}_t\|_2$ is a penalty on the bodyrate multiplied by a coefficient λ. In addition, we maximize a perception-aware reward $r_t^{\text{percep}} = \exp\left(-(u^2 + v^2 + \dot{u}^2 + \dot{v}^2)\right)$ to maximize the visibility of the next gate, where (u, v) and $(\dot{u}, \dot{v})$ are the pixel position and pixel velocity of the target gate center $\boldsymbol{g}_t$ in the image plane of the camera. The perception-aware reward incentivizes the policy to face the camera toward the target gate and hence reduce motion blur [23], which is crucial for vision-based flight. The full reward is $r_t = r_t^{\text{prog}} + a \cdot r_t^{\text{percep}}$, where a is a coefficient that trades off between progress maximization and perception-awareness.

Student Policy Training: We use imitation learning to distill the teacher's knowledge $\pi_{\text{teacher}}(\boldsymbol{G}_t, \boldsymbol{p}_t, \boldsymbol{o}_t)$ into a student policy $\pi_{\text{student}}(\boldsymbol{I}_t^H, \boldsymbol{O}_t^H | \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$, where $\boldsymbol{I}_t^H = (\boldsymbol{i}_{t-H+1}, \cdots, \boldsymbol{i}_t)$ indicates history images and $\boldsymbol{O}_t^H = (\boldsymbol{o}_{t-H+1}, \cdots, \boldsymbol{o}_t)$ represents history vehicle states. The teacher policy provides strong supervision on the action that the student policy should output in each state it visits, hence reducing the sample complexity. While the teacher policy uses privileged information about vehicle position and

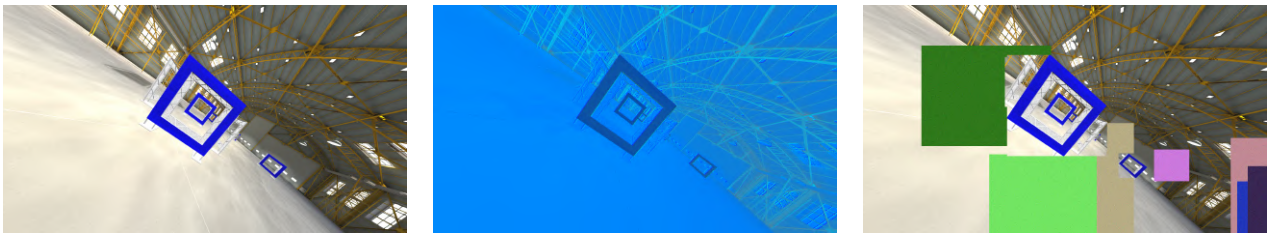


Fig. 3: Data augmentations used for training. Left: none. Middle: random convolution. Right: random cutout-color.

gate position, the student policy can only observe the drone state $\boldsymbol{o}_t$ and needs to infer other information, such as gates, from history image and state observations: $(\boldsymbol{I}_t^H, \boldsymbol{O}_t^H) \rightarrow (\boldsymbol{G}_t, \boldsymbol{p}_t)$. The feature extractor YOLO$^{\text{Feat}}$ extracts an image representation $\boldsymbol{z}_t$ from the image observation $\boldsymbol{i}_t$.

When using a single camera, the environment becomes only partially observable. Hence, we use a temporal convolutional network (TCN) [24] for the policy representation. The image representations $\boldsymbol{Z}_t^H = (\boldsymbol{z}_{t-H+1}, \cdots, \boldsymbol{z}_t)$ are concatenated with the state observations $\boldsymbol{O}_t^H$. The sequence of concatenated embeddings is then fed into the TCN to extract temporal information from history observations. Finally, we use a MLP as the policy head to regress the control command. The MLP takes the output of the TCN as input and produces the student policy's action $\boldsymbol{a}_t$. The imitation learning loss $\mathcal{L}_A$ is defined as the mean square error between the outputs of the teacher policy and the student policy:

$$\mathcal{L}_A(\boldsymbol{\theta}) = ||\pi_{\text{student}}(\boldsymbol{I}_t^H, \boldsymbol{O}_t^H|\boldsymbol{\theta}) - \pi_{\text{teacher}}(\boldsymbol{G}_t, \boldsymbol{p}_t, \boldsymbol{o}_t)||_2 \quad (2)$$

In addition, to achieve faster convergence [25], we include a latent loss $\mathcal{L}_E$ to supervise the output of the TCN $\boldsymbol{e}_t$ with the intermediate representation of the teacher $\bar{\boldsymbol{e}}_t$, written as $\mathcal{L}_E(\boldsymbol{\theta}) = ||\boldsymbol{e}_t - \bar{\boldsymbol{e}}_t||_2$. Therefore, we minimize the total loss $\mathcal{L} = \mathcal{L}_A(\boldsymbol{\theta}) + \lambda_E \mathcal{L}_E(\boldsymbol{\theta})$ for the imitation learning, where λ_E is a coefficient to weight the latent loss.

IV. EXPERIMENTS

A. Experimental Setup

Simulator Environment: We conduct experiments using the Flightmare [11] simulator, a realistic quadrotor simulator with various racing tracks and realistic racing environments. We set up three different race tracks (Circle, Figure8, and SplitS) in a warehouse environment (see the visualization in Fig. 3 left). For training the teacher policy, we use a customized implementation of the proximal policy optimization algorithm (PPO) [26] based on the code from [27]. We set the reward coefficients $\lambda = 0.01$ and $a = 0.15$. For training the student policy, we implement an imitation learning pipeline. We use the history length $H = 8$ and the coefficient $\lambda_E = 0.1$. For learning robust feature representations from raw images, we use data augmentation with random convolution and random cutout-color (see Fig. 3 middle and right).

Evaluation: To evaluate our policy, we rollout for 10 episodes for each setting, with quadrotor starting from different starting positions, which are sampled from a uniform distribution between -0.1m and 0.1m in x, y, z-axis of each. We evaluate the performance of our policy using two different metrics: Lap Time and Success Rate. The lap time indicates the racing performance of our policy, while the success rate shows the robustness of the policy. We report the lap time by computing the time required by the policy to finish one complete track and calculate the success rate by calculating the ratio that the quadrotor can successfully finish one full lap without crashing among the ten rollouts. The evaluation is done on a Ubuntu 20.04 machine with i7-10750H and RTX-2060 Mobile where a forward pass of our student policy takes 10ms.

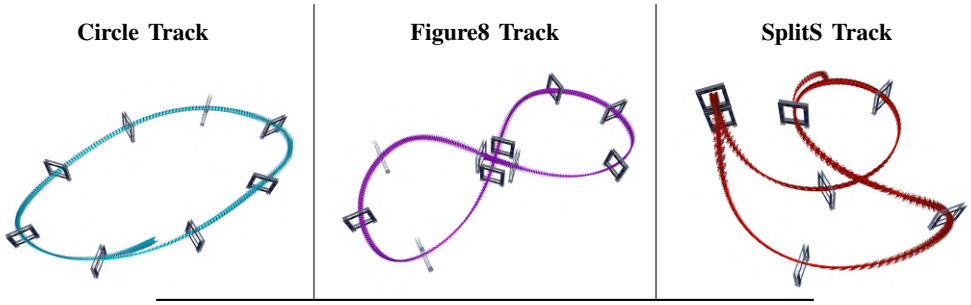
B. Baseline Comparisons

To evaluate the performance of our vision-based policy, we compare our method against two baselines: a state-based policy obtained via RL [14] and a time-optimal trajectory [12]. The state-based policy controls the drone using ground truth information about its state, including position, velocity, orientation, and acceleration, as well as the pose of the next two gates. The time-optimal trajectory serves as the theoretical minimum bound for our platform. On the other hand, our student policy uses only the camera and IMU to perceive the environment and control the drone. As a result, it can only observe the environment partially, similar to how human pilots control the drone using the first-person-view camera. Despite this limitation, our policy achieves strong performance on three different race tracks with high success rates, as shown in the results in Table I. Note that the reason for the student policy to achieve even lower lap time than the teacher policy is due to action errors, leading to risky behaviors such as cutting corners and smaller distance margins to the gate center. Nevertheless, our vision-based policies achieve state-of-the-art racing performance. Visualization of the trajectories of vision-based policies on three race tracks is shown in Table I

C. Handling Unseen Visual Disturbances

We test the robustness of our vision-based system in various unseen scenarios to investigate how it performs given unseen visual disturbances, such as color changes and brightness changes. Specifically, we darken the environment by lowering the brightness value from 1 to 0.8 and 0.5, and we also change environment colors by tuning the image hue value from 0 to both 0.5 and -0.5. Fig. 1 left and middle-left provide examples of environments with brightness values of 0.5 and hue values of 0.5, respectively. In addition, we also add visual distractors randomly in the environment.

TABLE I: Racing performance of different algorithms, including the time-optimal trajectory, state-based teacher policy, and vision-based student policy on three different race tracks.



	Lap Time [s]		
	Circle	Figure8	SplitS
Time-optimal Trajectory [12]	4.68	6.26	7.93
State-based Policy [14]	4.97±0.01	6.84±0.05	8.74±0.01
Vision-based Policy (ours)	4.95±0.01	6.76±0.01	8.58±0.01

For example, in Fig. 1 middle-right, we place multiple blue boxes randomly in the environment; and in Fig. 1 right, we instead add random objects that have irregular shapes in the environment.

Table II and Table III present the results of our comprehensive robustness study. Table II demonstrates the effectiveness of our system against visual disturbances. Our system maintains high success rates and fast lap times across all three racing tracks. However, we observed a slight decline in success rate on the challenging SplitS track when the number of boxes exceeded 60, dropping from 100% to 90%. Additionally, when we introduced 60 different random objects, our policy experienced failure in the Figure8 race track. Nonetheless, our policy demonstrated robustness against disturbances that are not experienced during training. In comparison, when using naive data augmentation, the success rate drops significantly on all three tracks as shown in Table III.

The key to having robust racing performance in the face of visual disturbances is to align the image representations, namely, image features extracted from the camera observations should be consistent across different scenarios. In Fig. 4, we present the qualitative results of image embeddings with contrastive learning and naive data augmentation. For each race track, we collect a trajectory of images by rolling out the teacher policy.

For visualization purposes, we extract the image representations with $\text{YOLO}_{\phi}^{\text{Feat}}$ and reduce the representations to 2-dimension with t-distributed stochastic neighbor representation (t-SNE) [28]. We can observe that the image features are well-aligned between training and evaluation when using contrastive learning to learn feature representations. During training, we use contrastive learning in which the similarity loss ensures that the feature extractor $\text{YOLO}_{\phi}^{\text{Feat}}$ learns the invariance between the two augmented images. As a result, the image representations among randomly augmented images are aligned in the image representation space. Thus, the image representations with visual disturbances are consistent with the representations of augmented images. It ensures that the TCN receives consistent image features from the feature extractor $\text{YOLO}^{\text{Feat}}$ when visual disturbances are present. Thus, our policy can maintain a high success rate under different disturbances (Table II). In comparison, there are significant deviations between image features during training and evaluation if we use naive data augmentation in feature representation learning. In this case, TCN gets significantly different image representations when visual disturbances appear. The policy fails to adapt to the disturbances and achieves 0 success rate.

D. Handling State Noise

For the observability of the system, our policy must rely on part of the drone state, including the orientation, linear velocity, and acceleration. The state information can be estimated using measurements from onboard sensors, such as IMUs, which are usually noisy. Hence, we further investigate the robustness of our policy against the state noise by adding Gaussian noise $\mathcal{N}(0, \sigma_n)$ individually to each component of the states, where σ_n is the standard deviation. Table IV shows the result. We can observe that our policy maintains a high success rate when the noise is small. The success rate on the most challenging SplitS track starts decreasing as we increase the standard deviation.

E. Understanding Network Attention

A fundamental question in representation learning is which features does the perception network learn? We visualize the attention map of the feature extractor $\text{YOLO}^{\text{Feat}}$ with Eigen-CAM [29] to investigate the perception module of our policy. Eigen-CAM provides interpretability of $\text{YOLO}^{\text{Feat}}$ by computing and visualizing the principle components of the learned image features from the convolutional layers. Our study suggests that different data augmentation allows $\text{YOLO}^{\text{Feat}}$ to learn different network attentions.

TABLE II: Success rate and lap time of our vision-based policy when facing different visual disturbances.

		Lap Time [s]			Success Rate		
		Circle	Figure8	SplitS	Circle	Figure8	SplitS
Brightness Change	0.5	4.88±0.01	6.71±0.01	8.60±0.01	1.0	0.8	1.0
	0.8	4.91±0.02	6.68±0.01	8.65±0.01	1.0	1.0	1.0
Hue Change	-0.5	4.91±0.01	6.72±0.03	8.66±0.01	1.0	1.0	1.0
	0.5	4.92±0.02	6.71±0.01	8.65±0.01	1.0	1.0	1.0
Blue Boxes	10	4.94±0.01	6.87±0.01	8.64±0.01	1.0	1.0	1.0
	60	4.90±0.02	6.77±0.01	8.73±0.01	1.0	1.0	0.9
Random Objects	10	4.94±0.02	6.75±0.02	8.67±0.01	1.0	1.0	1.0
	60	4.91±0.02	6.81±0.03	8.67±0.01	1.0	0.6	1.0

TABLE III: Success rate comparison between using contrastive learning and naive data augmentation for feature representation learning.

		Contrastive Learning			Naive Data Augmentation		
		Cricle	Figure8	SplitS	Cricle	Figure8	SplitS
Brightness Change	0.5	**1.0**	**0.8**	**1.0**	0.0	0.0	0.0
Hue Change	0.5	**1.0**	**1.0**	**1.0**	0.0	0.0	0.0
Blue Boxes	60	**1.0**	**1.0**	**0.9**	0.0	0.0	0.0
Random Objects	60	**1.0**	**0.6**	**1.0**	0.0	0.0	0.0

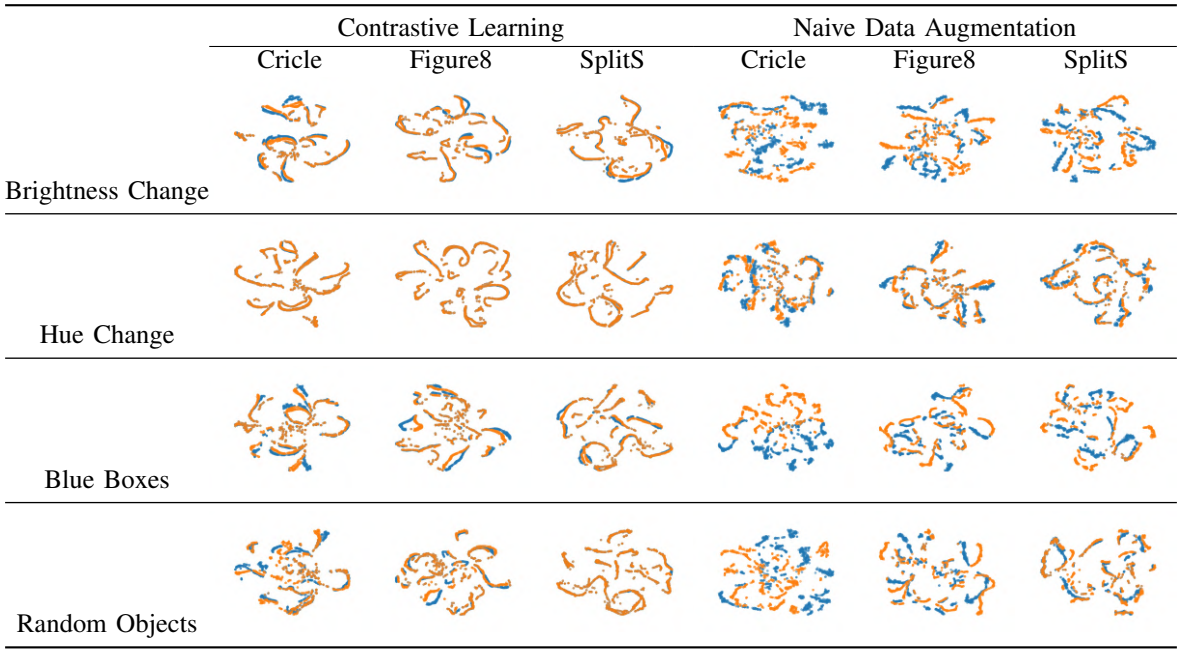


Fig. 4: A comparison of time-lapse t-SNE visualization of image representations by using contrastive learning and naive data augmentation. The blue and orange trajectories refer to the image representations of training augmentations and test disturbances, respectively.

Fig. 5 shows the applied random convolution during training and the attention of YOLO$^{\text{Feat}}$ during evaluation. Applying random convolution on the entire raw image allows changing the visual appearance of the image while retaining the shape information, e.g., edges. As a result, YOLO$^{\text{Feat}}$ learns to focus on both the gate and the background, which explains why the policy is robust against the change of the hue value and brightness.

Fig. 6 shows the applied random cutout-color during training and the attention of YOLO$^{\text{Feat}}$ during evaluation. During training, we randomly mask a large portion of the environment background, except the gates. As a result, YOLO$^{\text{Feat}}$ learns to focus attention mainly on the gates while ignoring the background information. Hence, the policy is robust against the change in the background, such as adding distractors.

TABLE IV: Success rates of the student policy when adding Gaussian noises to the drone states.

State Noise σ_n	Success Rate		
	Circle	Figure8	SplitS
0.04	1.0	1.0	1.0
0.12	1.0	1.0	0.7
0.20	1.0	1.0	0.4
0.28	0.3	0.5	0.0

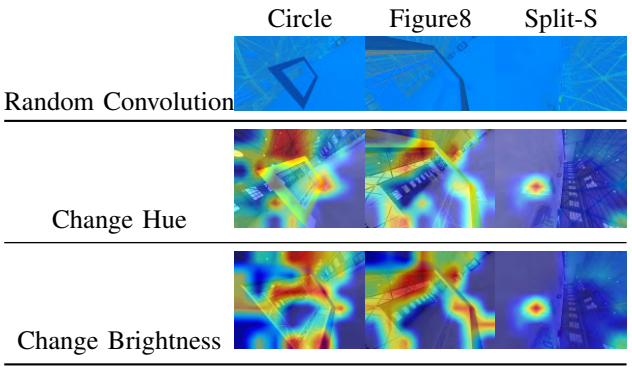


Fig. 5: Visualization of the attention map of the feature extractor YOLO$^{\text{Feat}}$, which is trained with contrastive learning and random convolution. Areas with yellow to red color indicate medium to high attention while areas in blue suggest low attention.

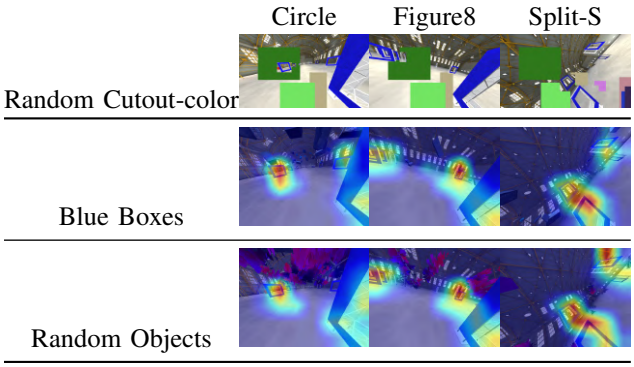


Fig. 6: Visualization of the attention map of the feature extractor YOLO$^{\text{Feat}}$, which is trained with contrastive learning and random cutout-color. Areas with yellow to red color indicate medium to high attention while areas in blue suggest low attention.

V. CONCLUSION

This work presented a method to learn deep sensorimotor policies for vision-based autonomous drone racing. We showed that a vision-based control policy allows predicting control commands with information extracted from images without explicitly estimating position information, trajectory planning, and tracking. The vision-based policy can achieve the same level of racing performance as the state-of-the-art state-based policy while being robust against different visual disturbances and distractors. The key to achieving robust sensorimotor control is to learn well-aligned image embeddings using contrastive learning and data augmentation.

However, there are some limitations to be mentioned and to be improved in the future. First, the deep sensorimotor policy still relies on accurate state measurements, rather than raw IMU measurements which are more noisy. A potential solution is to train a neural estimator that estimates the states from the sequence of image inputs and raw IMU measurements. Besides, the presented sensorimotor policy is both trained and evaluated with simulation data. The policy relies on well-aligned features, which can be sensitive to large distribution shifts, such as changing environments. Future work can focus on bridging the simulation-to-reality gap. A promising approach is to apply test-time adaptation [30] that adapts the vision-based policy to unknown environments. Another solution is to incorporate foundation models [31] into the feature extractor to increase its robustness to unknown environments.

REFERENCES

[1] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Autonomous Robots*, 2021.

[2] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, *et al.*, "Challenges and implemented technologies used in autonomous drone racing," *Intelligent Service Robotics*, 2019.

[3] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor, "Airsim drone racing lab," in *NeurIPS 2019 Competition and Demonstration Track*. PMLR, 2020.

[4] L. O. Rojas-Perez and J. Martínez-Carranza, "On-board processing for autonomous drone racing: an overview," *Integration*, vol. 80, pp. 46–59, 2021.

[5] C. De Wagter, F. Paredes-Valles, N. Sheth, and G. de Croon, "The sensing, state-estimation, and control behind the winning entry to the 2019 artificial intelligence robotic racing competition," *Field Robotics*, 2022.

[6] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

[7] C. Pfeiffer and D. Scaramuzza, "Human-piloted drone racing: Visual processing and control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3467–3474, 2021.

[8] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. Cun, "Off-road obstacle avoidance through end-to-end learning," *Advances in neural information processing systems*, vol. 18, 2005.

[9] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[10] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," *arXiv preprint arXiv:1709.07174*, 2017.

[11] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*, 2020.

[12] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, 2021.

[13] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[14] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2021.

[15] K. Nagami and M. Schwager, "Hjb-rl: Initializing reinforcement learning with optimal control policies applied to autonomous drone racing." in *Robotics: science and systems*, 2021.

[16] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Conference on Robot Learning*. PMLR, 2018, pp. 133–145.

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

[18] T. Wang and D. E. Chang, "Robust navigation for racing drones based on imitation learning and modularization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 724–13 730.

[19] M. Muller, G. Li, V. Casser, N. Smith, D. L. Michels, and B. Ghanem, "Learning a controller fusion network by online trajectory filtering for vision-based uav racing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[20] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, A. V, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, and M. T. Minh, "ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference," Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6222936

[21] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Ghesh-laghi Azar, *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[22] N. Hansen and X. Wang, "Generalization in reinforcement learning by soft data augmentation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 611–13 617.

[23] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. IEEE, 2018, pp. 1–8.

[24] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[25] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, pp. 1789–1819, 2021.

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

[27] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[28] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[29] M. B. Muhammad and M. Yeasin, "Eigen-cam: Class activation map using principal components," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[30] J. Liang, R. He, and T. Tan, "A comprehensive survey on test-time adaptation under distribution shifts," *arXiv preprint arXiv:2303.15361*, 2023.

[31] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.