Journal of Intelligent & Robotic Systems (2024) 110:55
https://doi.org/10.1007/s10846-024-02085-4

REGULAR PAPER

Trading-Off Safety with Agility Using Deep Pose Error Estimation and Reinforcement Learning for Perception-Driven UAV Motion Planning

Mehmetcan Kaymaz[1] · Recep Ayzit[1] · Onur Akgün[2] · Kamil Canberk Atik[1] · Mustafa Erdem[2] · Baris Yalcin[4] · Gürkan Cetin[4] · Nazım Kemal Ure[3]

Received: 4 April 2023 / Accepted: 28 February 2024

Abstract
Navigation and planning for unmanned aerial vehicles (UAVs) based on visual-inertial sensors has been a popular research area in recent years. However, most visual sensors are prone to high error rates when exposed to disturbances such as excessive brightness and blur, which can lead to catastrophic performance drops in perception and motion planning systems. This study proposes a novel framework to address the coupled perception-planning problem in high-risk environments. This achieved by developing algorithms that can automatically adjust the agility of the UAV maneuvers based on the predicted error rate of the pose estimation system. The fundamental idea behind our work is to demonstrate that highly agile maneuvers become infeasible to execute when visual measurements are noisy. Thus, agility should be traded-off with safety to enable efficient risk management. Our study focuses on navigating a quadcopter through a sequence of gates on an unknown map, and we rely on existing deep learning methods for visual gate-pose estimation. In addition, we develop an architecture for estimating the pose error under high disturbance visual inputs. We use the estimated pose errors to train a reinforcement learning agent to tune the parameters of the motion planning algorithm to safely navigate the environment while minimizing the track completion time. Simulation results demonstrate that our proposed approach yields significantly fewer crashes and higher track completion rates compared to approaches that do not utilize reinforcement learning.

Keywords Autonomous systems · Deep learning · Reinforcement learning

1 Introduction

Recent developments in artificial intelligence (AI) have led to a surge in popularity in adopting AI methodologies for the perception, planning, and control of autonomous vehicles [1]. In particular, unmanned aerial vehicles (UAVs) have greatly benefited from advances in deep learning [2], as highly accurate perception systems are necessary for solving challenging UAV navigation tasks [3]. There have been numerous powerful demonstrations of how deep learning and reinforcement learning can enhance UAV perception and planning performance in both simulated and real-life experiments [4, 5], and it can be asserted that deep learning-based perception systems [6] are becoming ubiquitous in this field. However, it should be noted that the current performance of most systems still fall short when compared to their human-controlled counterparts [7].

One of the critical shortcomings of modern deep learning-based perception systems is their decreased prediction accuracy under noisy visual inputs. While most networks are robust to small disturbances in input images, excessive brightness and blur, which are often present in real-world environments, can render the perception system useless and put the UAV in danger. For example, the well-studied DroNet architecture [8] yields excellent results under nominal environmental conditions and can be used to solve complex navigation tasks such as drone racing [9, 10]. However, as we demonstrate in the results section, the performance of such systems degrades significantly when high-noise visual inputs are introduced into the system, even for a short duration.

A simple but effective strategy to address such adversarial environments is to reduce the speed and agility of the UAV whenever excessive noise is detected. By doing so, the UAV has a lower risk of crashing when the perception system

✉ Mehmetcan Kaymaz
kaymazm16@itu.edu.tr

Extended author information available on the last page of the article

Published online: 27 March 2024

becomes ineffective. However, implementing this strategy in a heuristic manner, such as manually setting thresholds for commanding UAV speed based on perceived error levels, typically results in highly conservative solutions that further degrade the system's performance. We view this problem as a risk management issue since high-speed and high-agility maneuvers enable completing the navigation task in a shorter amount of time but also introduce a higher risk of crashing when the perception system becomes unreliable under noisy inputs. Conversely, low-speed maneuvers significantly decrease this risk but also increase the task completion time. The main focus of this work is to develop a machine learning-based solution to this risk management problem by combining ideas from deep learning-based error estimation for predicting errors of the perception system under high noise inputs and reinforcement learning for learning an effective trade-off strategy between the agility of the maneuvers and safety from collisions.

Contributions

Our proposed architecture for addressing the risk management problem in navigating under high noise conditions is presented in Fig. 1. Our focus is on the minimum-time navigation of a quadcopter through a sequence of gates in an unknown environment, a well-studied scenario that captures the essential challenges of many UAV navigation tasks, including collision avoidance, pose estimation, minimum-time trajectory generation, and agile maneuver control. We specifically consider scenarios where the perception system experiences infrequent disturbances from high brightness/blur inputs. Our contributions are as follows:

- On top of the established DroNet [8] architecture, we develop an error estimation network PE2Net (Pose Error Estimation Network), which is specifically tailored towards predicting the gate pose errors of DroNet under high disturbance conditions, such as excessive brightness. We validate the effectiveness of this approach in simulations.
- We have developed a reinforcement learning (RL) agent that tackles the risk management problem of selecting an appropriate velocity for the motion controller when faced with high noise conditions. The agent uses pose error estimations from PE2Net, pose estimation from DroNet, and UAV state measurements as inputs, and is trained using the Deep Q Network (DQN) algorithm. Our results show that the velocity profiles set by the RL agent strike a good balance between low-velocity maneuvers in high-noise conditions and high-velocity maneuvers in low-noise conditions. We have validated our approach across several different tracks with increasing complexity, and have found that our proposed RL approach yields significantly fewer crashes and higher track completion rates compared to alternative approaches that stick to low or high-velocity maneuvers, or approaches that set the velocity profile based on heuristics.

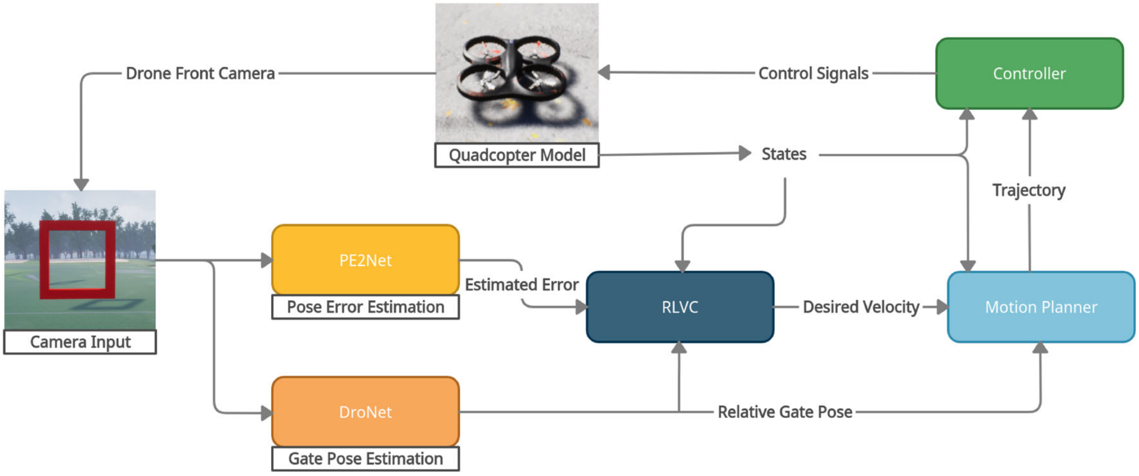


Fig. 1 The proposed architecture aims to solve the risk management problem when navigating in high-noise conditions. We utilize the DroNet architecture [8] to estimate the relative position and orientation of the gates with respect to the quadcopter, based on input from the front camera. Additionally, we introduce the PE2Net architecture, which estimates pose prediction errors based on the same camera input. The reinforcement learning velocity controller (RLVC) processes these estimations and quadcopter states to predict a target velocity for the motion planner, effectively setting a velocity appropriate for the estimated noise conditions. Subsequently, the motion planner generates a trajectory between the current location and the relative gate pose using the velocity set by the RLVC. Finally, a model predictive controller (MPC) computes the control signals required to follow the trajectory

1.1 Outline

The outline of the paper is as follows: we first review relevant previous work in UAV perception and planning in Section 2. Then, we describe the proposed perception methodology and pose error estimation network (PE2Net) in Section 3. Next, the motion planning and control algorithms are detailed in Section 4. In Section 5, we present the training and implementation of the RL agent. Finally, the simulation results are provided in Section 6.

2 Related Works

The following two subsections present the previous research on relevant perception and motion-planning algorithms.

2.1 Perception

Recent robotics applications gained considerable progress in sensing capabilities due to advances in deep learning. Despite these outstanding demonstrations, many perception algorithms do not work well in clustered and complex environments [11]. In particular, balancing safety and agility appears to be an unmet need in research on perception systems for autonomous vehicles. Previous work on gate identification and safe navigation via collision avoidance, which is the focus of our work, also has similar shortcomings. According to Foehn et al. [3], gate pose detection is a perception problem with uncertainties where the precise determination of relative gate distance and orientation is crucial for track completion. There are two approaches to this problem: one that requires an a priori map of the environment in order to estimate gate pose uncertainty [12], and another that does not require a map but does not provide uncertainty estimation, such as snake gate detection [13], end-to-end learnable visuomotor policies [9], and gap detection [14].

Additionally, high-precision gate detection is required for the drone to pass through the gates securely. Along with relaxing the dependence on GPS, image-based navigation opens up a large field for neural network-based solutions to this problem. However, Gal et al. [15] indicate that deep neural networks can distort data outside the training distribution in certain instances, producing erroneous predictions without providing a clear measure of certainty. Visual inputs are used for navigation [16, 17], which are less precise and robust but quicker and require only a camera.

Due to the low accuracy of the aforementioned system in high-noise conditions, it is vital to design risk-aware perception systems to provide safe navigation. The visual input quality is critical for building a high-performance perception system that enables safe and agile flying. Motion blur and signal noise are the two most prominent causes of image quality loss in digital imaging. Motion blur and noise may have a significant impact on image quality, especially in low-light circumstances [18].

Since these methods are less accurate, it is necessary to develop risk-aware perception systems for safe navigation. In [19], the authors combined both concepts of safety-aware learning and safety-critical control, which is a strong method for achieving safe behaviors on complex robotic systems in practice. Cassel et al. [20] proposed safety standards for advanced driver assistance systems (ADAS) that enable autonomous driving with proper perception systems. However, according to their safety argument, the human driver is always prepared to take control of the automobile in the case of a faulty sensor.

Kraus and Dietmyer [21] presented single-stage object identification using the Bayesian YOLO approach to evaluate the uncertainty of road users, such as pedestrians, automobiles, and bicycles, for risk-aware autonomous vehicles. Our work follows a similar mindset. By assessing the perception uncertainty using our proposed PE2Net, we develop a safe navigation strategy that can avoid potential collisions.

Richter et al. [22] proposed a strategy based on a deep neural network for a vision-guided robot that can improve its speed by 50% compared to a safe baseline policy of motion in similar environments and revert to a safe policy in a different environment. By contrast, PE2Net is a deep neural network that assesses the accuracy of DroNet predictions. In this study, we aim to demonstrate that our perception algorithm can succeed even when DroNet's predictions are inaccurate due to high ambient noise. As a result, the PE2Net structure estimates the accuracy of the DroNet gate pose predictions, allowing the drone to move more safely by utilizing PE2Net's pose error estimate information.

2.2 Motion Planning and Reinforcement Learning

Motion planning algorithms are a crucial part of agile UAV navigation tasks. According to a review by Gonzalez et al. [23], motion-planning algorithms fall into four groups: graph search, interpolating, numerical optimization, and sampling. In a study by Jin et al. [24], a graph search-based planning method is proposed to compute feasible smooth, minimum-time trajectories for a quadcopter. Zhou et al. [25] offer a decoupled strategy that includes processes for path discovery and trajectory optimization. Tordesillas et al. [26] suggest a strategy for resolving sluggish and conservative challenges by allowing the local planner to optimize in both known and unknown regions. Kayacan et al. [27] use a sampling-based motion-planning algorithm to solve the formation landing problem of a quadcopter, and Gebhardt et al. [28] demonstrate an optimization-based trajectory planning for quadcopters. In contrast, our study uses user-provided rough keyframes to generate feasible 3D trajectories via optimization-based methods. However, all of the algorithms outlined above share a common vulnerability to environmental noise.

There are notable results in the area of perception-aware motion planning. In [29], the authors address localization uncertainty by combining perception and motion planning. A recent end-to-end method [30] applies dual neural network structures. The first structure imitates the behaviors of the controller and motion planner, while the second structure is a reinforcement learning block to fine-tune the previously trained model. However, a disadvantage of preceding end-to-end approaches is that they require the usage of an expert algorithm. Another study maps neural network outputs to a motion planner, generating minimum-jerk trajectories to reach the desired goal [7]. In a recent study for high-speed quadcopter autonomous navigation, a robust and perception-aware re-planning strategy based on topological path-finding was proposed [31]. The suggested planner has been strengthened by the perception-aware method, which pays extra attention to places that may be hazardous to the quadcopter. Another alternative, entitled Model-based Reinforcement Learning (MBRL) with a latent space method, is used to decouple the environment from the dynamics [32]. It was used to teach a drone to fly to the desired location [33]. However, this approach is not efficient in terms of stability and safety. In [34], the navigation problem for the drone is mathematically defined as a Partially Observable Markov Decision Process (POMDP). Once the POMDP is solved mid-flight and in real-time utilizing augmented belief trees, a motion strategy is obtained. Completing the track safely in autonomous navigation is also a research area in autonomy where the problem is solved using curriculum learning [35].

In [36], reinforcement learning with a meta gradient is applied for safe navigation under disturbances for quadcopter motion planning. However, the proposed method does not involve visual disturbances. Also, there are new works on the usage of reinforcement learning for motion planning like double critic RL [37] and MPTD3 [38]. Yet, none of these works solves navigation under visual disturbances.

Recently, several studies on vision-based teaching-and-repeat systems [39, 40] have been published to overcome the challenge of motion planning within a given map. In [40], a vision-based drone is used to evaluate infrastructure on a recurring basis. During the training phase, the operator demonstrates the required trajectory, and some keyframes from the visual SLAM are captured as checkpoints. Then during the repetition phase, local trajectories connecting these checkpoints are constructed using minimum-snap polynomials [41]. However, these methods require smooth teaching trajectories and consistent configurations throughout the repetition phase [39].

The presented work on quadrotor control encompasses various methodologies aimed at enhancing the stability and tracking performance of UAVs. In [42], the authors explored a dual-loop single dimension fuzzy-based sliding mode control , highlighting robust tracking capabilities. Similarly, they proposed a fuzzy-based backstepping control for stabilization considering unmodeled dynamic factors [43] and colleagues evaluated the performance of different control methods, incorporating a position estimator and disturbance observer [44]. The methodologies discussed above have significantly contributed to the field of quadrotor control, emphasizing stability, tracking, and robustness. However, these existing methods may fall short in addressing the intricate trade-off between safety and agility, which may require a more sophisticated approach to balance safety considerations with the agility needed for dynamic UAV maneuvers.

In conclusion, while there have been significant advances in the fields of risk-aware and perception-aware planning, there is still a considerable research gap when it comes to establishing a direct connection between the performance and safety of autonomous systems in terms of agility and collision avoidance, respectively, based on the level of perception uncertainty.

3 Perception System and Deep Pose Error Estimation

As emphasized in the Introduction Section, we focus our attention on navigating a quadcopter through a sequence of gates, where the layout of the map and gates are not known beforehand. The central perception problem for this scenario is gate pose prediction [9, 13]. Figure 2 shows the images obtained from the drone camera in the simulation environment.

This subsection develops the architecture of the deep neural network-based system that predicts, both standard pose estimates (relative distance and orientation between the quadcopter and the gate), and the errors of each pose prediction. The error estimations are then utilized to evaluate the perception system's uncertainty, which in turn is used for online tuning of the motion controller described in Section 4 using the RL method developed in Section 5.

3.1 Perception System Architecture

Our proposed perception system consists of two architectures as presented in Fig. 3. First, we utilize DroNet [8], which is a convolutional neural network (CNN) based network with eight residual layers, for predicting the gate poses $\hat{y}_{B_G}$ from the individual input images. DroNet takes RGB images with 200 × 200 size as inputs, and predicts a four-dimensional

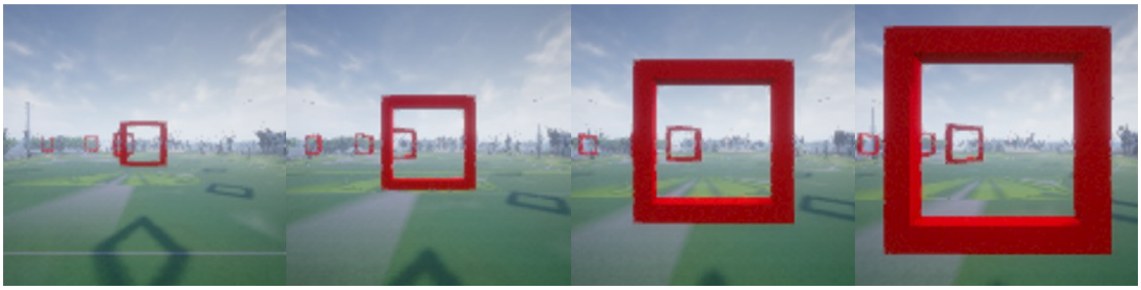


Fig. 2 Door views from various angles and orientations captured with the drone's camera. The perception system determines the distance between the drone and the closest gate

vector $\hat{y}_{B_G} = [\hat{r}, \hat{\phi}, \hat{\theta}, \hat{\psi}] \in \mathbb{R}^4 \in SO(3)$, which represents the predicted gate pose by DroNet. $\hat{r}$ denotes the distance between the drone body frame origin and gate frame origin whereas $\hat{\phi}$ and $\hat{\theta}$ are the relative orientation of the drone and the gate in spherical coordinates. $\hat{\psi}$ is the yaw angle difference between the drone body frame and the gate frame. After computing the difference between the gate and body frames in the Cartesian coordinate system, the result is converted to spherical coordinates using (see Eq. 1). These angles serve as reference points for the drone's trajectory-control system, which guides the drone to the target gate from its current location. DroNet is a pre-trained network, we only fine-tune the last layer of the network to match our simulation data distribution.

The second section of the perception system is where we present the Pose Estimation Error Network (PE2Net) as a novelty to the literature. PE2Net, like DroNet, is a CNN-based network that employs the same image input format. The PE2Net output, $\sigma^2_{DroNet} \in \mathbb{R}$, reflects the target gate's pose estimate error based on visual input. The DroNet can be trained with noisy data as well, but training with high noise is unsuccessful. We also want the DroNet system to be thought of as a closed box, that is, a perception system that is used as is. As a result, a new system is required to perceive uncertainty. The Pose Error Estimation system can also be trained to predict ambient noise, but the noise does not always indicate incorrect estimation.

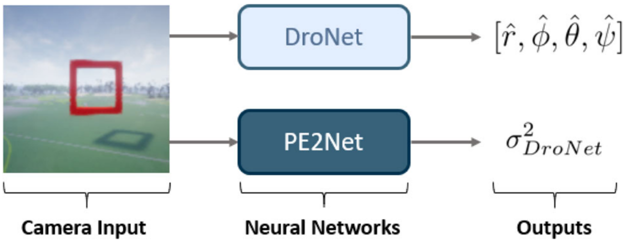


Fig. 3 Perception system architecture. DroNet and PE2Net are both CNN-based networks that take images as inputs. DroNet generates the target gate's relative pose and orientation, whereas PE2Net predicts the error of the DroNet predictions

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2}, \\ \phi &= \arctan \frac{y}{x}, \\ \theta &= \arccos \frac{z}{r}. \end{aligned} \tag{1}$$

3.2 Frame Convention

The inertial frame is represented by I, the body frame is represented by B and the gate frame is represented by G. P_{I_B} is the drone position with respect to the inertial frame, and P_{I_G} is the position of the gate with respect to the inertial frame. To calculate the gate located in the inertial frame, our proposed network takes measurements in P_{B_G}, which is the position of the gate in the drone body frame.

P_{I_B} and P_{I_G} are parameterized as Cartesian coordinate system, with variables (x, y, z). For the training, P_{I_B} data are provided as ground truth in the simulation and estimated with the flight control unit, whereas P_{I_G} data are computed by the onboard system based on perception system data and flight control data. P^s_{GB} is parameterized in spherical coordinates in the body frame: (r, θ, ϕ).

$$\begin{aligned} x &= r \sin\theta \cos\phi, \\ y &= r \sin\theta \sin\phi, \\ z &= r \cos\theta. \end{aligned} \tag{2}$$

3.3 Pose Error Estimation with PE2Net

PE2Net (Fig. 4) is a CNN-based network that makes use of the ResNet8 [45] architecture, which is similar to the residual layers used in DroNet. The 200x200-pixel input picture is fed into the network and encoded using convolutional and fully connected layers. PE2Net produces a one-dimensional vector that contains the total of the squared differences between the ground truth values and DroNet outputs. This design is motivated by the fact that anytime the camera is subjected to

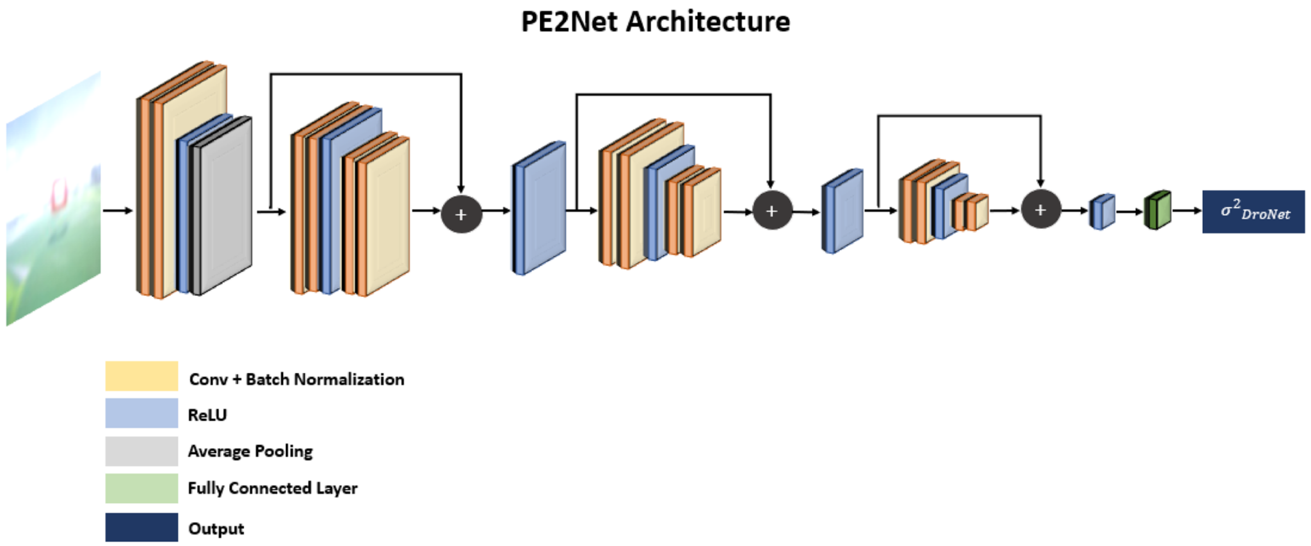


Fig. 4 PE2Net architecture for estimating the error of DroNet predictions. 200x200x3 input image passes through the 3x32 convolutional layer and the pooling layer whose kernel size is 2. The network continues with ResNet-8 architecture with three residual layers with a stride of 2 whose output channels are 32, 64, and 128 respectively, followed by the ReLu activation function. The image is encoded with these layers and the network outputs the error of the DroNet after a fully connected layer

high amounts of noise, such as excessive or inadequate light exposure, the output of the PE2Net σ^2_{DroNet}, which reflects the uncertainty of the predicted gate pose, should increase as expected. Thus, abnormalities observed by PE2Net in DroNet outputs can be utilized for quantifying the uncertainty in perception estimations.

3.4 Data Generation and Training Procedure for the Perception System

We use the AirSim [46] simulator to generate the training data for the DroNet and PE2Net networks. 175000 images were captured in AirSim utilizing a variety of gate and drone location setups to train and test DroNet (95% for the train and 5% for the test). The gates were located randomly in the area that can reside in the drone front camera's field of view.

First, we train the DroNet in isolation, using the loss function in Eq. 3, which is the mean squared error between the true and predicted gate pose values. To train the network, we utilized stochastic gradient descent with a learning rate of 10^{-4}, a batch size set of 16, a weight decay of 10^{-2}, and a training epoch length of 100.

$$L\left(\theta_{DroNet}\right) = \frac{1}{N}\sum_{n=0}^{N} ||y^s_{BG_n} - \hat{y}^s_{BG_n}||^2 \tag{3}$$

In the second training phase, we train PE2Net using a separate data generation procedure and outputs of the DroNet trained in the first phase. The input visuals for PE2Net were captured using the drone's onboard camera in the simulator, as seen in Fig. 5. Gaussian noise with a mean of 0 and standard deviation of 0.8 was used to modify the brightness of the picture in order to obtain noisy measurements. 25000 noisy images of the target gates at various distances and angles were captured and the ground truth position of each gate with respect to the drone was recorded in the simulation. As expected, DroNet's predictions diverge from the ground truth values under these noisy images.

The labels of PE2Net are the pose estimation errors of DroNet. The output of the DroNet with respect to drone is $\hat{y}_{BG_n} = [\hat{x}, \hat{y}, \hat{z}, \hat{\psi}]$ and the ground truth is $y_{BG_n} = [x, y, z, \psi]$. The pose estimation error is defined as,

$$e_n = (y_{BG_n} - \hat{y}_{BG_n})H(y_{BG_n} - \hat{y}_{BG_n})^T \tag{4}$$

H in Eq. 4 is the coefficient matrix to normalize each squared error due to the metrics of $[x, y, z]$ and the metric of ψ are not the same. Mean square error is used as a loss function to train PE2Net.

$$L\left(\theta_{PE2Net}\right) = \frac{1}{N}\sum_{n=0}^{N}(e_n - \hat{e}_n)^2 \tag{5}$$

The learning rate, batch size, and weight decay parameters for PE2Net were manually tuned at 10^{-3}, 16, and 10^{-3}, respectively, by comparing several combinations. In order to minimize overfitting, dropout and weight decay regularization approaches were also used. The prediction performance of the perception system is presented in Section 6.

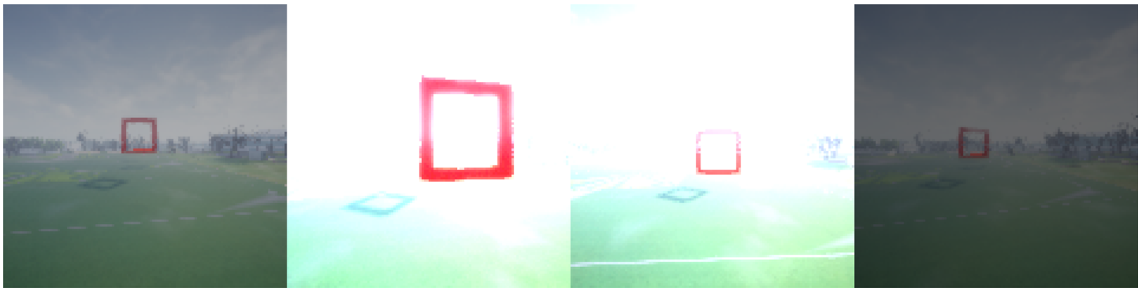


Fig. 5 Noisy images captured by the drone's onboard camera in the AirSim simulator. Noise impairs the accuracy of DroNet's outputs. PE2Net estimates the accuracy of DroNet's image-based predictions

4 Motion Planning and Control

In this section, we present the details of the motion planning/trajectory generation methodology, as well as the low-level control approach.

4.1 Motion Planning

Although computationally heavy optimization-based approaches can be used for the drone navigation problems [47, 48], we employ a simple polynomial trajectory generation algorithm [49] that does not require intense computations, while still providing agile trajectories that are sufficient for completing the navigation task. The location and orientation of the gate relative to the drone camera are the outputs of the DroNet module discussed in Section 3. The Cartesian coordinates as well as the yaw angle of the gate relative to the drone camera are obtained after converting the gate pose from the spherical frame to the Cartesian frame. The trajectory's time is determined by dividing the gap between the current and target positions by the desired velocity estimated via reinforcement learning.

4.2 Controller Design

Model Predictive Control (MPC) has grown in popularity as a paradigm for quadcopter control due to its ability to optimize actuation restrictions and performance objectives concurrently [50]. Model Predictive Control (MPC) is selected as the preferred control methodology over alternative approaches, owing to its heightened efficacy in managing intricate and nonlinear systems characterized by inherent uncertainties. Furthermore, MPC demonstrates superior predictive control capabilities. Notably, the proposed system is versatile, as it can seamlessly integrate with alternative controllers, provided they adhere to the designated trajectory. MPC's success is dependent on the availability of an accurate dynamics model of the underlying system since the approach involves online model-based online optimization.

The optimization problem for Nonlinear Model Predictive Control (NMPC) to control the quadcopter is defined as,

$$\underset{X,\,U}{\text{minimize}} \sum_{k=0}^{N-1} (x_{k+1}^{ref} - x_{k+1})^T Q_x (x_{k+1}^{ref} - x_{k+1}) + \Delta u_k^T R \Delta u_k$$

$$\begin{aligned} \text{subject to}\; & x_{\min} \le x_k \le x_{\max},\, x_k \in \mathbb{R}^{12} \\ & u_{\min} \le u_k \le u_{\max},\, u_k \in \mathbb{R}^4 \\ & x_{k+1} = f_{dyn}(x_k, u_k) \end{aligned}$$

Here N is the prediction horizon and x_k is the states, $x_k^{ref} \in \mathbb{R}^{12}$ is the reference trajectory, u_k is the control input, Δu_k is the change in u all at kth time step. $R \in \mathbb{R}^{4\times4}$ is a coefficient matrix that penalizes relative big changes in u and $Q_x \in \mathbb{R}^{12\times12}$ is the coefficient matrix that reflects the relative importance of x. $f_{dyn}()$ is the dynamic model of the quadcopter in which the model defined in [51] is used with RK4.

The solution of the nonlinear optimization problem gives the necessary control inputs for the quadcopter. The IPOPT [52] is used for solving the optimization problem for the quadcopter control.

5 Reinforcement Learning for Trading-Off Safety with Agility

The perception, motion planning, and control systems presented in Sections 3 and 4 usually provide sufficient performance under nominal conditions, where there are no high amplitude noises present in the environment. However, when noise levels become excessive, the integrated perception-planning system may fail catastrophically (see Section 6). As sensing errors can only be compensated up to a certain level by modifying the perception system, the only viable option is to modify the trajectories and control signals generated by the planning subsystem. For example, decreasing the reference velocity of the computed trajectory for high-noise events can enable UAVs to bypass these events with relative safety. In this section, we design an algorithm that continuously alters the reference velocities based on the error estimations of the perception system. To minimize the number of manually set parameters and remove domain expertise bias, we design this algorithm as a reinforcement learning (RL) agent and allow the algorithm to learn a good policy for mapping estimated noise to reference velocity from interactions with simulated events. The RL agent's primary objective is to change the required velocity of the quadcopter and ensure task completion in the presence of various disturbances on the camera.

5.1 State and Action Space of the RL Agent

During both training and testing, the agent has access to input data, which includes previously trained PE2Net error estimations y_{PE2Net}, DroNet gate estimations y_{DroNet}, and quadcopter states. These inputs are given in Eq. 6. The output/action of the agent is to either increase or decrease the velocity or maintain it constantly. We use the Deep Q Network (DQN) [53] method to train our agent because of its simple yet efficient structure. The complete architecture of the system is illustrated in Fig. 6.

$$\begin{aligned} y_{DroNet} &= [\hat{r}, \hat{\theta}, \hat{\phi}, \hat{\psi}], \\ y_{PE2Net} &= [\sigma^2_{DroNet}], \\ x &= [v_x, v_y, v_z, \phi, \theta, \psi]. \end{aligned} \tag{6}$$

The policy network of the agent is represented by four fully connected layers with ReLU activations in the middle layers and linear output at the last layer. The output of the policy is mapped to a discrete action $a \in [0, 1, 2]$. Here, $a = 0$ denotes decrease in the velocity ($V \downarrow$), $a = 1$ indicates that the velocity (V) should remain constant, while $a = 2$ indicates that the velocity should increase ($V \uparrow$) as shown in Eq. 7.

$$v_{t+1}^{des} = \begin{cases} v_t^{des} - \Delta v & \text{, if } a = 0 \\ v_t^{des} & \text{, if } a = 1 \\ v_t^{des} + \Delta v & \text{, if } a = 2 \end{cases} \tag{7}$$

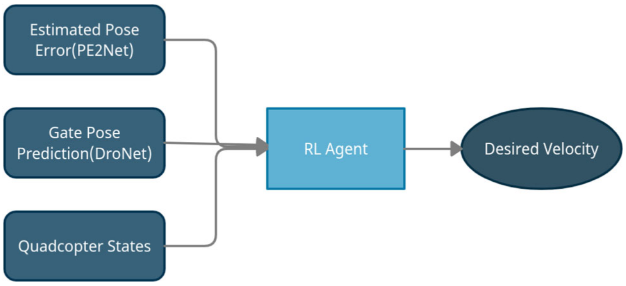


Fig. 6 Reinforcement Learning (RL) agent architecture. The network is a fully connected neural network that takes eleven inputs and outputs three values. There are three hidden layers, each of which has 64, 128, and 64 neurons

In Eq. 7, Δv is a constant whose value is dictated by the frequency of policy runs ($\approx 10Hz$) and the physical restrictions of our real-world quadcopter. As a result, whereas large-valued Δv is viable in simulation, it is not practical in physical applications. It is possible to consider a finer resolution action space by decreasing Δv and increasing the number of actions, however in our experiments, we were able to get the desired performance by just using three actions (see Section 6).

As is the case with many other learning-based algorithms, the definition of the reward function is critical to the agent's performance. The agent's ultimate objective is to maximize agility while maintaining safety measures. With this in mind, the reward function is set as shown in Eq. 8,

$$R = \begin{cases} -1000 & \text{, if drone crashes or leaves the track} \\ +1000 & \text{, if drone completes the track} \\ V & \text{, Otherwise} \end{cases} \tag{8}$$

When the agent crashes the quadcopter or leaves the track, the agent receives a large negative reward. On the other hand, when the agent completes the track, it receives a positive reward. In other circumstances, the agent receives current velocity as a positive reward during the flight. As a result, the agent receives the maximum cumulative reward when it

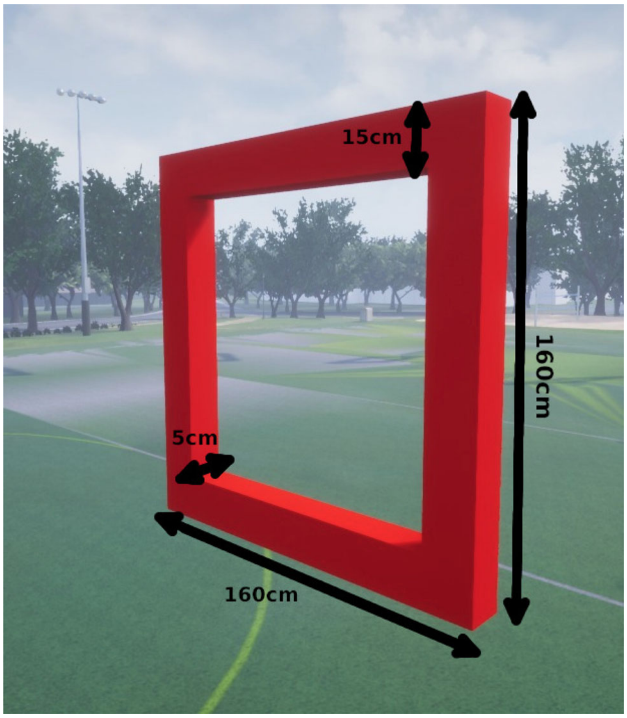


Fig. 7 Representation of a gate used in the simulation for vision-based UAV navigation. The dimensions of the gate are the same as the ones used in [9]

flies the track with high velocity while avoiding crashing. Hence the reward function motivates selecting actions that lead to safe and agile trajectories.

5.1.1 Terminal States

The terminal states are the states in which an episode comes to a conclusion if the agent's actions bring it to these specified states. There are two primary methods for concluding an episode in this scenario. The first is if the drone successfully completes the track, while the second is if it crashes. When the drone goes through all of the track's gates, an episode is completed. An accident can occur in one of two ways: the first is when the drone collides with a gate on the track, and the second is when the drone goes out of the track zone, which is the cubic area that includes all the gates.

5.2 RL Agent Training Procedure

Training of the RL agent is completely executed in the AirSim environment described in Section 3. The output of previously trained and frozen networks (DroNet, PE2Net), in conjunction with the interactions between the agent and the simulation, defines the agents' essential transitions (state, action, reward, next state, terminal). The track utilized for training purposes in the Airsim environment is displayed in Fig. 10 and consists of three gates (Fig. 7). The network parameters are updated using the following Eq. 9.

$$\begin{aligned}\widehat{y} &= r_t + \gamma \max_a \widehat{Q}(s_{t+1}, a, \widehat{\theta})(1 - d_t) \\ L(\theta) &= (\widehat{y} - Q(s_t, a_t, \theta))^2 \\ \theta &= \theta - \alpha \nabla_\theta L(\theta)\end{aligned} \tag{9}$$

Where $(s_t, a_t, r_t, s_{t+1}, d_t)$ denote state, action, reward, next state, and terminal respectively, Q indicates the value-action network, θ denotes weights of value-action network and $\widehat{Q}, \widehat{\theta}$ represent target network and its weights. $\widehat{y}$ is target value, γ is discount factor and α is learning rate. The entire training procedure is detailed in Algorithm 1.

6 Results

In this section, we present the performance results of the isolated perception system and the integrated perception-planning system.

Algorithm 1 Training DQN algorithm.

```
Input: randomly initialized action-value network $Q_\theta(s, a)$ and target action-value network $\widehat{Q}_\theta(s, a)$, replay buffer $D$, pre-trained DroNet, pre-trained PE2Net
Parameters: learning rate $\alpha = 0.003$, number of episodes $E$ =25000, $\epsilon_{initial} = 1$, $\epsilon_{final} = 0.05$, $\epsilon_{decay} = 0.999$, batch size is 256, discount factor $\gamma = 0.99$, $\epsilon = \epsilon_{initial}$.
for $episode = 1, E$ do
    Update $\epsilon$ according to $\epsilon = max(\epsilon_{final}, \epsilon \times \epsilon_{decay})$
    Initialize a random normal distribution $\mathcal{N}$
    Retrieve image $img_1$ and quadcopter state $x_1$ from the environment
    Add noise $n_1 \in \mathcal{N}$ to $img_1$
    Feed image $img_1$ into DroNet and PE2Net and obtain $y_{DroNet_1}, y_{PE2Net_1}$
    Construct current state $s_1 = [y_{DroNet_1}, y_{PE2Net_1}, x_1]$
    for $t = 1, T$ do
        With probability of $\epsilon$ select a random action $a_t$
        otherwise $a_t = argmax_a Q_\theta(s_t, a_t)$
        Execute action $a_t$ and observe image $img_{t+1}$ and quadcopter state $x_{t+1}$
        Add noise $n_{t+1} \in \mathcal{N}$ to $img_{t+1}$
        Feed new $img_{t+1}$ into DroNet and PE2Net and obtain $y_{DroNet_{t+1}}, y_{PE2Net_{t+1}}$
        Calculate reward $r_t$ with respect to Eq. 8
        Construct next state $s_{t+1} = [y_{DroNet_{t+1}}, y_{PE2Net_{t+1}}, x_{t+1}]$
        Check whether the episode ends w.r.t Section 5.1.1
        Store transition $(s_t, a_t, r_t, s_{t+1}, d_t)$ in buffer $D$
        Sample a random minibatch of transitions from buffer $D$
        Perform gradient descent update according to Eq. 9
        Every $K$ steps $\widehat{Q}_\theta(s, a) \leftarrow Q_\theta(s, a)$
    end for
end for
```

6.1 Isolated Performance of the PE2Net Percepton System

PE2Net is evaluated in the Airsim environment by introducing brightness and blur to the input picture. DroNet and PE2Net are provided with the image received from the Airsim test environment after it has been altered by adjusting the brightness of the image. Brightness noise is used to picture the one-third period of the whole test procedure, as seen in Fig. 8. The green region depicts the image's brightness noise, while the orange line reflects the mean squared error between the DroNet prediction and the ground truth values. The blue line depicts the anticipated error of the gate posture using PE2Net. The initial and final stages of the test method are noise-free, resulting in the lowest possible error values for both the ground truth and the PE2Net model. When Gaussian noise is applied to an image, DroNet is quite likely to begin to fail by producing more mistakes on target gate pose predictions. In this situation, the output of the PE2Net also increases, suggesting that prediction error has increased and that, in order to ensure a safe flight, the reinforcement learning strategy may favor a drop in velocity. It should be noted that the output of the PE2Net is mainly used for quantifying

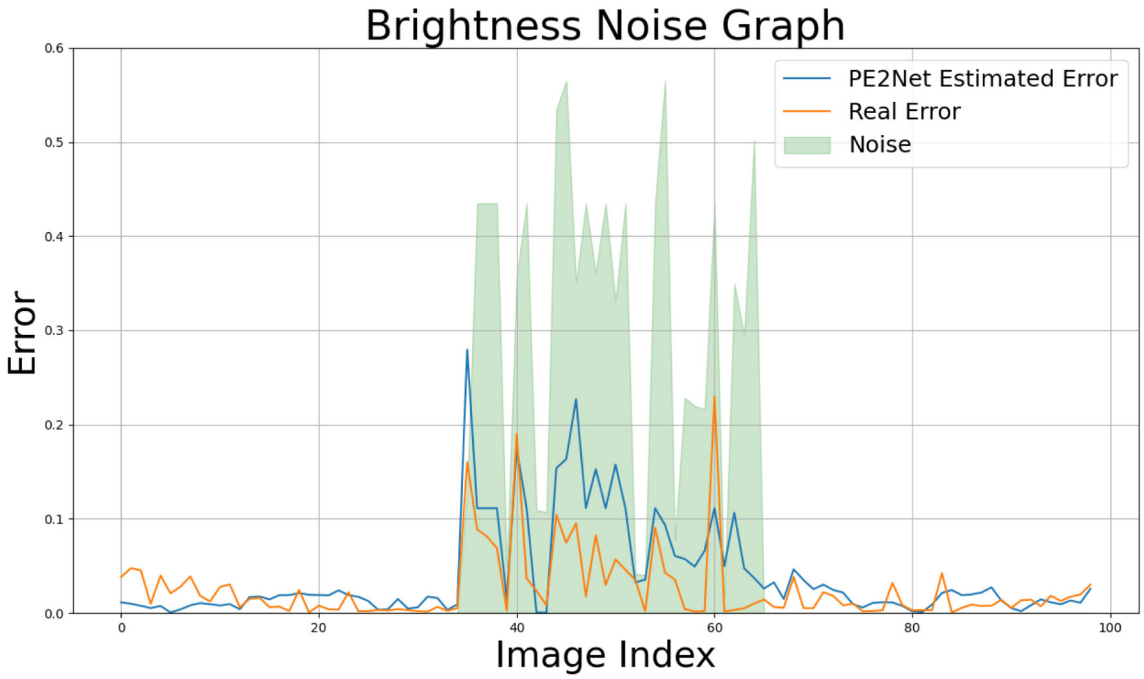


Fig. 8 PE2Net results under brightness noise. Plots of estimated errors produced by PE2Net (blue) and real errors of DroNet predictions (orange) where the input image is exposed to noise in the green region are shown in the figure. It can be seen that outputs of the PE2Net are correlated with noise level on images. Error in this plot is a unitless quantity, computed as the sum of normalized values in different units

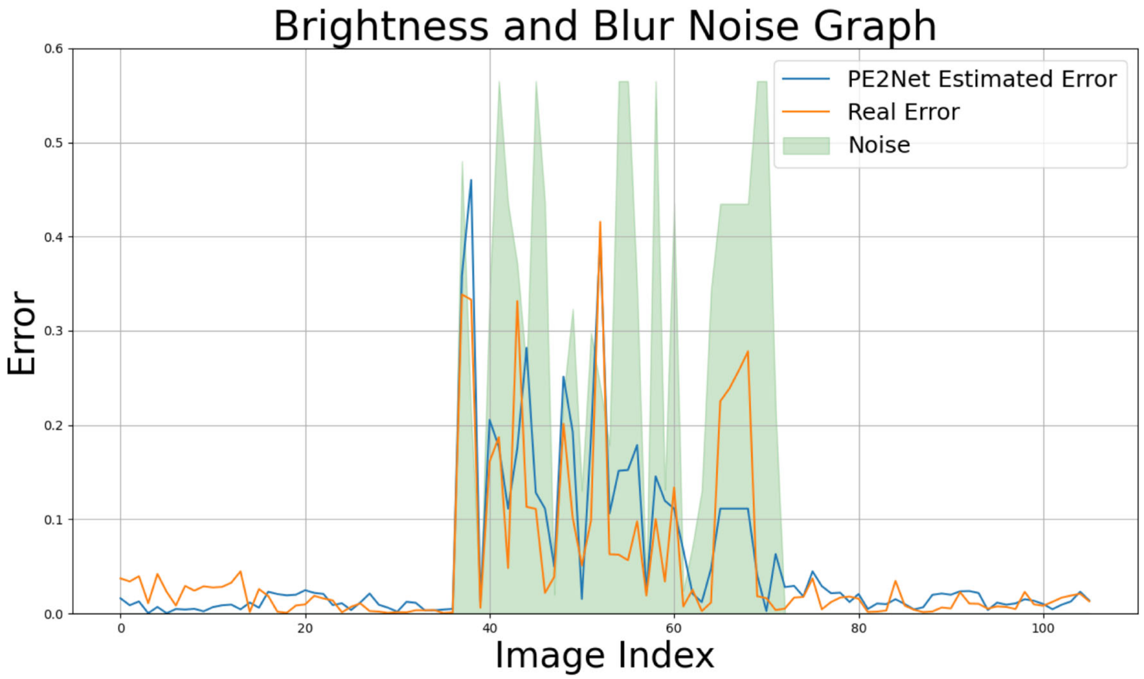
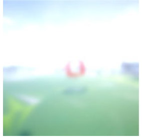
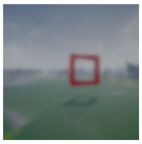


Fig. 9 PE2Net results for both bright and blurred noise. Plots of estimated errors produced by PE2Net (blue) and real errors of DroNet predictions (orange) where the input image is exposed to noise in the green region are shown in the figure. It can be seen that outputs of the PE2Net are correlated with noise level on images. Error in this plot is a unitless quantity, computed as the sum of normalized values in different units

Table 1 PE2Net prediction and pose estimation error (ground truth) are compared

Image Number	1	2	3	4
Image				
Pose Estimation Error	0.003440	0.04456	0.4156	0.02500
PE2Net Prediction	0.003217	0.04732	0.3978	0.02089

The gates are positioned and oriented differently. The predicted gate poses are subjected to a variety of noises, including changing lighting conditions and motion blurring. Error in this figure is a unitless quantity, computed as the sum of normalized values in different units

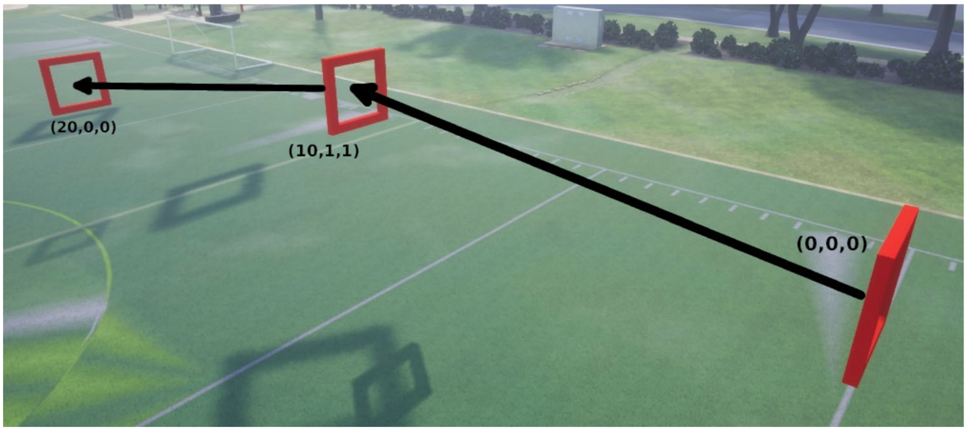


Fig. 10 Track 1 - On this track, our proposed RL velocity adjusting agent is trained. The relative gate location with respect to the first gate is given in the figure in (x,y,z) format in meters. The episode begins with the quadcopter in position (-10,0,0), with the objective of passing through all gates sequentially

Table 2 Heuristic velocity adjusting techniques

Strategy	Features
NVC_{fast}	Throughout the episode, the target velocity always increases until it reaches the maximum velocity of 10 m/s.
NVC_{mean}	The target velocity rises continually, but the maximum velocity is limited to 6 m/s.
NVC_{slow}	The target velocity increases constantly, but the maximum is 3 m/s.
AVC_{fast}	The target velocity is determined by the estimated error from PE2Net. $v_{t+1}^{des} = \begin{cases} v_t^{des} - \Delta v & \text{if } \sigma^2_{DroNet} > \tau_{AVC_{fast}} \\ v_t^{des} + \Delta v & \text{if } \sigma^2_{DroNet} < \tau_{AVC_{fast}} \end{cases}$ (10) The upper limit for the desired velocity is the same with NVC_{fast} and $\tau_{AVC_{fast}}$ is 0.1.
AVC_{mean}	Identical to AVC_{fast}, but with an extra threshold parameter. $v_{t+1}^{des} = \begin{cases} v_t^{des} - \Delta v & \text{if } \sigma^2_{DroNet} > \tau_{AVC_{fast}} \\ v_t^{des} & \text{if } \tau_{AVC_{slow}} < \sigma^2_{DroNet} < \tau_{NVC_{fast}} \\ v_t^{des} + \Delta v & \text{if } \sigma^2_{DroNet} < \tau_{AVC_{slow}} \end{cases}$ (11) The upper limit for the target velocity is the same as NVC_{mean}.
AVC_{slow}	Identical to AVC_{fast}, but with a lower *tau* threshold. $v_{t+1}^{des} = \begin{cases} v_t^{des} - \Delta v & \text{if } \sigma^2_{DroNet} > \tau_{AVC_{slow}} \\ v_t^{des} + \Delta v & \text{if } \sigma^2_{DroNet} < \tau_{AVC_{slow}} \end{cases}$ (12) The target velocity upper limit is the same for NVC_{slow} and $\tau_{AVC_{slow}}$ is 0.03.
$RLVC$	The target velocity in this approach is determined by the actions of the RL agent as described in Eq. 7.

These methods are crafted based on domain expertise and compared against our data-driven RL-based approach that does not require manually crafted thresholds

Table 3 Performance Evaluation Results on Track 1 across different velocity adjusting approaches

Track 1 (Training Track)			
Algorithm	Agility(%)	Success rate (With noise)	Success rate (No noise)
NVC_{fast}	$91 \pm 8\%$	11%	50%
NVC_{mean}	$58 \pm 8\%$	57%	80%
NVC_{slow}	$9 \pm 7\%$	91%	100%
AVC_{fast}	$75 \pm 8\%$	40%	60%
AVC_{mean}	$52 \pm 7\%$	58%	70%
AVC_{slow}	$8 \pm 7\%$	91%	100%
$RLVC$	$35 \pm 7\%$	90%	100%

This track is also used for training the proposed RL algorithm. The agility metric measures the mean speed of the quadcopter, while the success rate indicates the number of successfully completed tracks. It is observed that the proposed approach yields the highest track completion rates

the magnitude of the error, rather than getting a precise estimate to correct DroNet outputs, which might be infeasible due to the magnitude of noise levels. Similar test results are seen in Fig. 9 images distorted with both brightness and blur. The only distinction is the noise, which in this case is a combination of brightness and blur. In comparison to Fig. 8, DroNet predictions become less dependable as the noise level grows, whereas the error estimation provided by PE2Net increases in accordance.

Table 1 illustrates the PE2Net model's test results in a simulation setting on some sample pictures. The table compares both ground truth and PE2Net estimated error for noise-free and noisy pictures. The first image (index 1) is completely noise-free. The related PE2Net output and the actual error are on a 10^{-3} scale with a precision of 6.5 percent. When the second image is compared to the fourth image, it is evident that the error estimation is not just reliant on noise, but also on features such as the distance and orientation of the gate. Even when the camera is subjected to noise, DroNet can provide accurate findings, and the PE2Net model can provide a high degree of assurance regarding the error estimation. The third picture produces a result that is distinct from the rest. The third image was subjected to high-intensity noise, and both the genuine error value and the PE2Net error estimation revealed an uncertainty of around 0.4 (unitless). As a result, it can be said that the PE2Net model is capable of predicting the uncertainty associated with DroNet forecasts.

6.2 Comparative Performance Results of the RL-Based Integrated Perception-Planning System

In this section, we evaluate the performance of the main contribution of this work, an integrated perception-planning system that can adjust the reference velocity to mitigate the risk in high-noise events.

Three distinct tracks were used to evaluate the proposed structure. Only the first track was used to train the RL algorithm. The training track can be viewed in Fig. 10. We trained the algorithm on the training track for 25000 episodes.

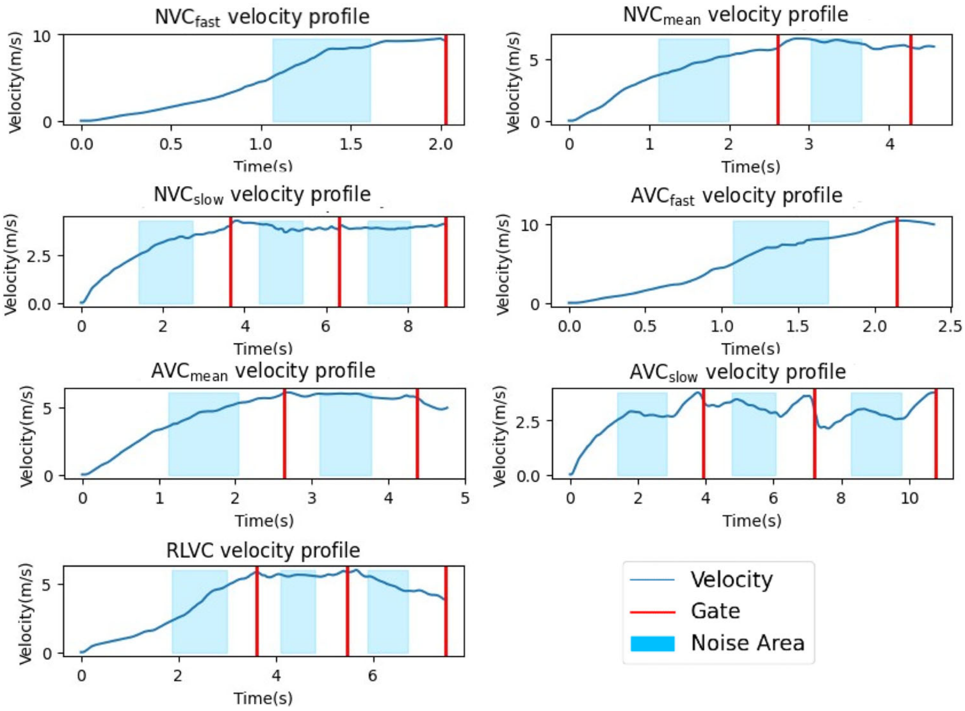
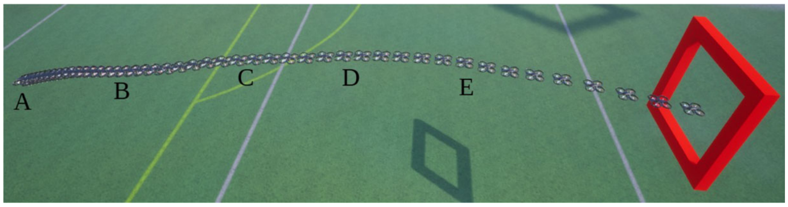


Fig. 11 The velocity profiles of the compared strategies are plotted under the same noise distribution. In this case, NVC_{fast} and AVC_{fast} crash to the first gate, whereas NVC_{mean} and AVC_{mean} crash immediately after the second gate. NVC_{slow}, AVC_{slow}, and $RLVC$ successfully finishes the track. The noise area is the time interval where the quadcopter is subject to noise

Table 4 PE2Net pose error estimations and the actions of the RL agent in track 1

Trajectory					
Locations	A	B	C	D	E
Image					
Estimated Error	.021	.023	.231	.175	.0149
DQN Action	V↑	V	V↓	V↑	V↑

V↑ indicates speed increase, V↓ indicates speed decrease, and V indicates steady speed

The track is designed to be not overtly challenging, however, with added perception noise, even the most cautious planner might struggle to complete the track successfully. Six distinct heuristic velocity-adjusting systems were tested and compared to our RL-based system. The whole list of these adjusters, together with their attributes, can be seen in Table 2. The objective here is to show that data-driven approaches such as RL, can show better performance compared to handcrafted heuristic solutions.

All methods were evaluated 100 times on each track to compute mean performance under uncertain navigation conditions. At every episode, the position and the yaw angle of the quadrotor are kept constant. However, during an episode when, how long and how big the noise will be is determined randomly. All results in this section are the mean performance metrics that are averaged over these 100 runs. To imitate changing lighting conditions, Gaussian noise is introduced into the newly collected image to imitate shifting illumination conditions. For all techniques, the initial conditions at the start of each episode (see Section 5.1.1) are identical. All solutions were compared on a basis of two primary performance indicators: mean agility and mean success rate. The mean agility term refers to the average velocity magnitude throughout all episodes. Each run from initialization until a crash occurs or the quadcopter completes the track is an episode, whereas the mean success rate term refers to the average of all episodic success rates expressed in percentage. The drone is considered successful if it passes through all gates and completes the track.

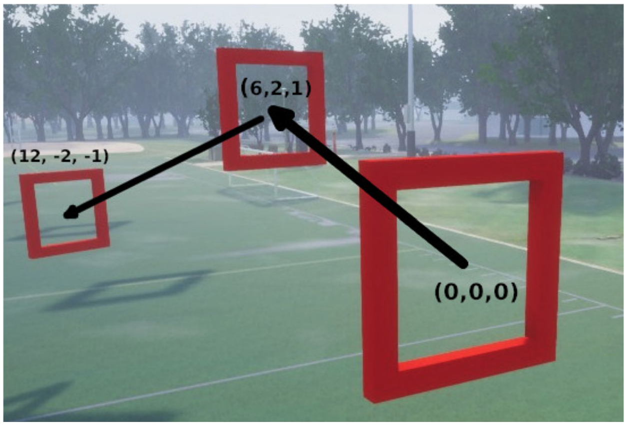


Fig. 12 Overview of the geometry of Track 2, one of the test tracks. This track consists of three gates that are close in proximity to one another on the x-axis but farther apart on the y and z-axes, making it more difficult than the training track in Fig. 10. The illustration depicts the relative gate location in relation to the first gate in (x,y,z) format, in meters

The evaluation results on Track 1 indicate that flying slowly in a noisy environment leads to a higher success rate. As seen in Table 3, the fastest velocity tuning approach NVC_{fast} is only 11 percent successful at completing the track. In comparison to NVC_{fast}, NVC_{mean} as a velocity adjuster achieves a higher success rate (57%), but not high enough to compete with NVC_{slow} (91%), and $RLVC$. (90%).

Table 5 Comparative performance evaluation results for Track 2

Track 2 (Test Track)			
Algorithm	Agility(%)	Success rate(With noise)	Success rate(No noise)
NVC_{fast}	$73 \pm 9\%$	0%	40%
NVC_{mean}	$36 \pm 9\%$	28%	60%
NVC_{slow}	$7 \pm 3\%$	72%	100%
AVC_{fast}	$72 \pm 4\%$	0%	50%
AVC_{fast}	$26 \pm 5\%$	34%	70%
AVC_{slow}	$7 \pm 3\%$	72%	100%
$RLVC$	16±3%	70%	100%

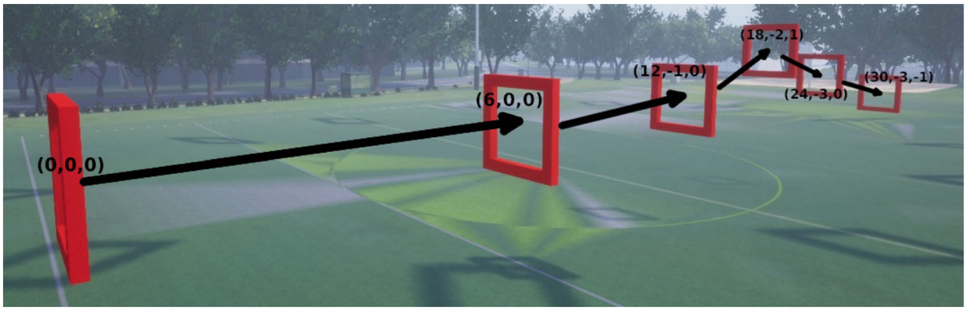


Fig. 13 Track 3, the final test track. This track has six gates and the same gate placement as Track 2. Due to the number of gates, this course is difficult to complete in a noisy environment. The graphic depicts the relative gate location in relation to the first gate in (x,y,z) format, with the metric meter

Given the presence of unexpected and extreme noise, even in a simulation environment, a 100 percent success rate is implausible. When comparing NVC techniques to their adaptive counterparts AVC, it becomes clear that utilizing a rule-based approach as the velocity adjuster is often safer but not more agile. The proposed solution $RLVC$ successfully completes the track at a rate of 90%, which is extremely similar to the rates of NVC_{slow} and AVC_{slow}. Our technique, on the other hand, is 25% more agile than its competitors on average. Hence for the training track, the proposed approach outperforms the other methods in terms of both agility and safety.

The velocity profiles for the compared methods are depicted in Fig. 11. NVC approaches to maintain a steady speed for the quadcopter regardless of whether there is noise in the visual input. The AVC algorithms alter the quadcopter's speed in response to estimated errors from PE2Net and predefined thresholds.

As seen in Fig. 11 and Table 4, our approach has a unique velocity profile. For example, despite the absence of noise in the B frame and the low predicted error, the agent elected to retain its pace. Similarly, the RL agent recommends accelerating despite the harsh illumination circumstances and the estimated high error. This is the anticipated behavior of our proposed method since it takes into account not just the estimated error but also the quadcopter's current states and DroNet predictions. The algorithm's architecture enables the quadcopter to be securely controlled in challenging circumstances, as demonstrated in Table 4.

The earlier findings presented in Table 3 demonstrate the algorithms' performance on the same track where the agent was trained. In order to demonstrate the generalization capability of the proposed approach, we evaluate the results on two test tracks, which were not available to the RL agent during the training phase. The proposed velocity adjusters are evaluated on another test track, Fig. 12, which presents a greater challenge for planners due to the placement of gate locations. Even the slowest but most cautious planner could only complete the track with a success percentage of 72% as depicted in Table 3. Planners with greater agility, such as NVC_{fast} and AVC_{fast}, were unable to complete the track at all. Our approach $RLVC$ has a success rate of 70%, which is comparable to NVC_{slow} and AVC_{slow}, however, the proposed method outperforms the other strategies in terms of agility. Table 5 provides a consolidated view of the performance of all algorithms.

Finally, we present another test track, displayed in Fig. 13. This track is even more challenging and it is designed to assess how the algorithms perform in a larger area and more demanding setting. Although this track is more difficult to complete, the algorithms' performances are relatively similar to the track in Fig. 12. As a consequence, it can be concluded that our algorithm performs much better than the other algorithms in terms of overall performance, as it is capable of balancing the trade-off between agility and safety by learning a data-driven risk-management strategy (Table 6).

7 Conclusion

As UAV navigation tasks become increasingly demanding and the number of autonomous operations in the real world increases, UAVs are exposed to more adversarial environmental conditions that can seriously degrade perception performance, leading to weak planning performance. In this work, we propose a novel reinforcement learning-based architecture that utilizes pose error estimation under high-noise conditions and learns to adjust the reference velocity for motion planners to manage the risk of crashing by trading-off agility with safety based on the perceived noise levels. Our

Table 6 Comparative performance evaluation results for Track 3

Track 3 (Test track)			
Algorithm	Agility(%)	Success rate(With noise)	Success rate(No noise)
NVC_{fast}	$74 \pm 7\%$	0%	30%
NVC_{mean}	$46 \pm 7\%$	30%	60%
NVC_{slow}	$8 \pm 6\%$	69%	100%
AVC_{fast}	$57 \pm 8\%$	0%	30%
AVC_{mean}	$29 \pm 8\%$	32%	60%
AVC_{slow}	$8 \pm 7\%$	68%	100%
$RLVC$	18±7%	68%	100%

results show that the pose error estimation system works well, and the simulation results demonstrate that the integrated system outperforms heuristic approaches that rely on manually crafted rule sets in terms of both agility and track completion rates. In comparison to the most secure heuristic methodologies, the algorithm under consideration demonstrates a noteworthy enhancement in agility, with an approximate mean increase of 15 percent. Concurrently, this augmentation in agility is accompanied by a marginal reduction in safety, amounting to only 1 percent. Therefore, it can be concluded that the reinforcement learning approach has the potential to discover unique risk-management strategies that cannot be easily obtained by using domain expertise and/or manual tuning.

For future work, we recommend conducting real-world experiments on the integrated system for further validation of the reinforcement learning methodology. Additionally, it is worth noting that although this work focused on UAV navigation, the framework is actually platform agnostic and can be easily adapted to any autonomous system that requires tight coupling between perception and planning modules.

Acknowledgements This work is supported by Havelsan and the Scientific Research Project Unit (BAP) of Istanbul Technical University, Grant NO: MOA-2019-42321.

Author Contributions **Mehmetcan Kaymaz** conceived the research, wrote the article, and contributed to reinforcement learning, motion planning, modeling, control, and simulation. **Recep Ayzit** conceived the research, wrote the article, and contributed to the perception system, and simulation. **Onur Akgün** wrote the article, surveyed the literature, and contributed simulation. **Kamil Canberk Atik** wrote the article, surveyed the literature, and contributed to the perception system. **Mustafa Erdem** wrote the article, surveyed the literature, and contributed simulation. **Baris Yalcin** supervised the research. **Gürkan Cetin** supervised the research. **Nazım Kemal Ure** wrote the article and supervised the research.

Funding This work is supported by Havelsan Grant NO: MOA-2019-42321. Mehmetcan Kaymaz has received research support.

Availability of Code and Data The codes and datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Declarations

Competing interests The authors have no relevant financial or nonfinancial interests to disclose.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Bojarski, M., del Testa, D.W., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K.: End to end learning for self-driving cars. arXiv:1604.07316 (2016)
2. Bengio, Y., Lecun, Y., Hinton, G.: Deep learning for ai. Commun. ACM **64**(7), 58–65 (2021)
3. Foehn, P., Brescianini, D., Kaufmann, E., Cieslewski, T., Gehrig, M., Muglikar, M., Scaramuzza, D.: Alphapilot: Autonomous drone racing. arXiv:2005.12813 (2020)
4. Bartolomei, L., Teixeira, L., Chli, M.: Semantic-aware active perception for uavs using deep reinforcement learning. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3101–3108 (2021). https://doi.org/10.1109/IROS51168.2021.9635893
5. Akbari, Y., Almaadeed, N., Al-maadeed, S., Elharrouss, O.: Applications, databases and open computer vision research from drone videos and images: a survey. Artif. Intell. Rev. **54**(5), 3887–3938 (2021)
6. Tai, L., Liu, M.: Deep-learning in mobile robotics-from perception to control systems: a survey on why and why not. arXiv:1612.07139 (2016)
7. Kaufmann, E., Loquercio, A., Ranftl, R., Dosovitskiy, A., Koltun, V., Scaramuzza, D.: Deep drone racing: learning agile flight in dynamic environments. arXiv:1806.08548 (2018)
8. Loquercio, A., Maqueda, A.I., Blanco, C.R.D., Scaramuzza, D.: Dronet: learning to fly by driving. IEEE Robot. Automation Lett. (2018). https://doi.org/10.1109/lra.2018.2795643
9. Bonatti, R., Madaan, R., Vineet, V., Scherer, S., Kapoor, A.: Learning controls using cross-modal representations: bridging simulation and reality for drone racing. arXiv:1909.06993 (2019)
10. Jung, S., Hwang, S., Shin, H., Shim, D.H.: Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. IEEE Robot. Automation Lett. **3**(3), 2539–2544 (2018). https://doi.org/10.1109/LRA.2018.2808368
11. Sharma, V.D., Toubeh, M., Zhou, L., Tokekar, P.: Risk-aware planning and assignment for ground vehicles using uncertain perception from aerial vehicles (2020). arXiv:2003.11675
12. Kaufmann, E., Gehrig, M., Foehn, P., Ranftl, R., Dosovitskiy, A., Koltun, V., Scaramuzza, D.: Beauty and the beast: optimal methods meet learning for drone racing. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 690–696 (2019). https://doi.org/10.1109/ICRA.2019.8793631
13. Li, S., Ozo, M.M., De Wagter, C., de Croon, G.C.: Autonomous drone race: a computationally efficient vision-based navigation and control strategy. Robot. Autonomous Syst. **133**, 103621 (2020)
14. Sanket, N.J., Singh, C.D., Ganguly, K., Fermüller, C., Aloimonos, Y.: Gapflyt: active vision based minimalist structure-less gap detection for quadrotor flight. IEEE Robot. Automation Lett. **3**(4), 2799–2806 (2018)
15. Gal, Y.: Uncertainty in deep learning (2016)

16. Li, R., Wang, S., Long, Z., Gu, D.: Undeepvo: monocular visual odometry through unsupervised deep learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 7286–7291 (2018)
17. Chakravarty, P., Narayanan, P., Roussel, T.: Gen-slam: generative modeling for monocular simultaneous localization and mapping. In: 2019 International Conference on Robotics and Automation (ICRA), IEEE, pp. 147–153 (2019)
18. Kurimo, E., Kunttu, L., Nikkanen, J., Grén, J., Kunttu, I., Laaksonen, J.: The effect of motion blur and signal noise on image quality in low light imaging, pp. 81–90 (2009). https://doi.org/10.1007/978-3-642-02230-2_9
19. Cosner, R.K., Tucker, M., Taylor, A.J., Li, K., Molnár, T.G., Ubellacker, W., Alan, A., Orosz, G., Yue, Y., Ames, A.D.: Safety-aware preference-based learning for safety-critical control. arXiv:2112.08516 (2021)
20. Cassel, A., Bergenhem, C., Christensen, O., Heyn, H.-M., Leadersson-Olsson, S., Majdandzic, M., Sun, P., Thorsén, A., Trygvesson, J.: Perception safety requirements and multi sensor systems for automated driving systems. (2020). https://doi.org/10.4271/2020-01-0101
21. Kraus, F., Dietmayer, K.: Uncertainty estimation in one-stage object detection. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE, ??? (2019). https://doi.org/10.1109/itsc.2019.8917494
22. Richter, C., Roy, N.: Safe visual navigation via deep learning and novelty detection. In: Robotics: Science and Systems (2017)
23. González, D., Pérez, J., Milanés, V., Nashashibi, F.: A review of motion planning techniques for automated vehicles. IEEE Trans. Intell. Transp. Syst. **17**(4), 1135–1145 (2015)
24. Liu, S., Atanasov, N., Mohta, K., Kumar, V.: Search-based motion planning for quadrotors using linear quadratic minimum time control. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2872–2879 (2017). https://doi.org/10.1109/IROS.2017.8206119
25. Zhou, B., Gao, F., Wang, L., Liu, C., Shen, S.: Robust and efficient quadrotor trajectory generation for fast autonomous flight. IEEE Robot. Automation Lett. **4**(4), 3529–3536 (2019)
26. Tordesillas, J., Lopez, B.T., How, J.P.: Faster: fast and safe trajectory planner for flights in unknown environments. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 1934–1940 (2019)
27. Dong, Y., Fu, C., Kayacan, E.: Rrt-based 3d path planning for formation landing of quadrotor uavs. In: 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), pp. 1–6 (2016). https://doi.org/10.1109/ICARCV.2016.7838567
28. Gebhardt, C., Hepp, B., Nägeli, T., Stevšić, S., Hilliges, O.: Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pp. 2508–2519 (2016)
29. Costante G., Forster C., Delmerico J., Valigi P., Scaramuzza D.: Perception-aware path planning (2016). arXiv:1605.04151
30. Lin J., Wang L., Gao F., Shen S., Zhang F.: Flying through a narrow gap using neural network: an end-to-end planning and control approach (2019). arXiv:1903.09088
31. Zhou, B., Pan, J., Gao, F., Shen, S.: Raptor: robust and perception-aware trajectory replanning for quadrotor fast flight. IEEE Trans. Robotics **37**(6), 1992–2009 (2021)
32. Richard, A., Aravecchia, S., Geist, M., Pradalier, C.: Learning behaviors through physics-driven latent imagination. In: Faust A., Hsu D., Neumann G. (eds.) Proceedings of the 5th Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 164, pp. 1190–1199. PMLR, ??? (2022). https://proceedings.mlr.press/v164/richard22a.html
33. Becker-Ehmck, P., Karl, M., Peters, J., van der Smagt, P.: Learning to fly via deep model-based reinforcement learning (2020). arXiv:2003.08876
34. Sandino, J., Maire, F., Caccetta, P., Sanderson, C., Gonzalez, F.: Drone-based autonomous motion planning system for outdoor environments under object detection uncertainty. Remote. Sens. **13**, 4481 (2021)
35. Ozturk, A., Burak Gunel, M., Dagdanov, R., Ekim Vural, M., Yurdakul, F., Dal, M., Kemal Ure, N.: Investigating value of curriculum reinforcement learning in autonomous driving under diverse road and weather conditions. In: 2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops), pp. 358–363 (2021). https://doi.org/10.1109/IVWorkshops54471.2021.9669203
36. Yu, Q., Luo, L., Liu, B., Hu, S.: Re-planning of quadrotors under disturbance based on meta reinforcement learning. J. Intell. & Robotic Syst. **107**(1), 13 (2023). https://doi.org/10.1007/s10846-022-01788-w
37. Grando, R.B., de Jesus, J.C., Kich, V.A., Kolling, A.H., Drews-Jr, P.L.J.: Double critic deep reinforcement learning for mapless 3d navigation of unmanned aerial vehicles. J. Intell. & Robotic Syst. **104**(2), 29 (2022). https://doi.org/10.1007/s10846-021-01568-y
38. Xu, G., Jiang, W., Wang, Z., Wang, Y.: Autonomous obstacle avoidance and target tracking of uav based on deep reinforcement learning. J. Intell. & Robotic Syst. **104**(4), 60 (2022). https://doi.org/10.1007/s10846-022-01601-8
39. Gao, F., Wang, L., Zhou, B., Zhou, X., Pan, J., Shen, S.: Teach-repeat-replan: a complete and robust system for aggressive flight in complex environments. IEEE Trans. Robotics **36**(5), 1526–1545 (2020)
40. Fehr, M., Schneider, T., Dymczyk, M., Sturm, J., Siegwart, R.: Visual-inertial teach and repeat for aerial inspection (2018). arXiv:1803.09650
41. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: 2011 IEEE International Conference on Robotics and Automation, IEEE, pp. 2520–2525 (2011)
42. Abro, G.E.M., Bin Mohd Zulkifli, S.A., Asirvadam, V.S.: Dual-loop single dimension fuzzy-based sliding mode control design for robust tracking of an underactuated quadrotor craft. Asian J. Control 25(1), 144–169 (2023) https://arxiv.org/abs/https://onlinelibrary.wiley.com/doi/pdf/10.1002/asjc.2753. https://doi.org/10.1002/asjc.2753
43. Abro, G.E.M., Zulkifli, S.A.B.M., Ali, Z.A., Asirvadam, V.S., Chowdhry, B.S.: Fuzzy based backstepping control design for stabilizing an underactuated quadrotor craft under unmodelled dynamic factors. Electronics **11**(7) (2022). https://doi.org/10.3390/electronics11070999
44. Mustafa Abro, E.G., Ali, Z., Zulkifli, S., Asirvadam, V.: Performance evaluation of different control methods for an underactuated quadrotor unmanned aerial vehicle (quav) with position estimator and disturbance observer. Math. Problems Eng. **2021**, 1–22 (2021). https://doi.org/10.1155/2021/8791620
45. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015). arXiv:1512.03385
46. Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: high-fidelity visual and physical simulation for autonomous vehicles. In: FSR (2017)
47. Foehn, P., Romero, A., Scaramuzza, D.: Time-optimal planning for quadrotor waypoint flight. Sci. Robotics (2021)
48. Foehn, P., Scaramuzza, D.: CPC: Complementary Progress Constraints for Time-Optimal Quadrotor Trajectories. (2020). https://doi.org/10.48550/ARXIV.2007.06255. arXiv:2007.06255
49. Wang, Y.-S., Sun, L., Zhou, L., Liu, J.-T.: Online minimum-acceleration trajectory planning with the kinematic constraints. Acta Automatica Sinica **40**(7), 1328–1338 (2014). https://doi.org/10.1016/S1874-1029(14)60014-8

50. Emami, S.A., Banazadeh, A.: Simultaneous trajectory tracking and aerial manipulation using a multi-stage model predictive control. Aerospace Sci. Technol. **112**, 106573 (2021)
51. Wang, P., Man, Z., Cao, Z., Zheng, J., Zhao, Y.: Dynamics modelling and linear control of quadcopter. In: 2016 International Conference on Advanced Mechatronic Systems (ICAMechS), pp. 498–503 (2016). https://doi.org/10.1109/ICAMechS.2016.7813499
52. Pólik, I., Terlaky, T.: In: Di Pillo G., Schoen F. (eds.) Interior Point Methods for Nonlinear Optimization, pp. 215–276. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11339-0_4
53. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Mehmetcan Kaymaz received B.Sc. and M.Sc. degrees from Istanbul Technical University, in 2021 and 2023, respectively. He is currently working as an Engineer at Roketsan. His main research interests include applications of deep learning and deep reinforcement learning for autonomous systems, and the development of high-performance guidance navigation and control algorithms.

Recep Ayzit received the B.S. degree in astronautical engineering from Istanbul Technical University, Istanbul, Türkiye, in 2023. He is currently a student of M.S. in aerospace engineering. His current research interests include flight performance, control and optimization.

Onur Akgün received his Master's degree in Control and Automation Engineering from Yildiz Technical University, Istanbul, Turkey, in 2018. He is currently pursuing his Ph.D. in Curriculum Learning in Robotics at Istanbul Technical University, Istanbul, Turkey. Alongside his doctoral studies, he serves as a Research Assistant at the Turkish-German University, contributing to both academic research and teaching. His research interests are centered on artificial intelligence for robotics, with a particular focus on developing sophisticated learning algorithms that enhance robotic autonomy and adaptability.

Kamil Canberk Atik received the B.S. degree in Astronautical Engineering from Middle East Technical University, Ankara, Türkiye, in 2020. He is currently a student of M.S. in aerospace engineering. His current research interests include vision based mapping and object detection.

Mustafa Erdem received the B.S. degree in 2014 and the M.S. in 2019 both in Mechatronics Engineering. He is currently a Ph.D. candidate and Research Assistant in Mechatronics Engineering at Istanbul Technical University, Turkey. His research interests include multi-agent reinforcement learning, intelligent control, and their applications in robotics.

Baris Yalcin received his B.Sc. degrees in Electrical and Electronic Engineering from Bogaziçi University in 2005, and his M.Sc. and Ph.D. degrees in Robotics and Control Systems from Keio University in 2007 and 2010, respectively. He is currently a Lead Engineer at Havelsan. His main research interests include robotics and autonomous systems.

Gürkan Cetin received the B.Sc. and M.Sc. degrees in Astronautical Engineering from Istanbul Technical University, in 2002 and 2006, respectively. He is currently a Leader Engineer at Havelsan. His main research interests include Robotics and Autonomous Systems.

Nazım Kemal Ure received the B.Sc. and M.Sc. degrees from Istanbul Technical University, in 2008 and 2010, respectively, and the Ph.D. degree in Department of Aeronautics and Astronautics from the Massachusetts Institute of Technology, in 2015. He is currently an Associate Professor at the Department of Artificial Intelligence and Data Engineering, Istanbul Technical University. His main research interests include applications of deep learning and deep reinforcement learning for autonomous systems, large scale optimization, and the development of high-performance guidance navigation and control algorithms.

Authors and Affiliations

Mehmetcan Kaymaz[1] · Recep Ayzit[1] · Onur Akgün[2] · Kamil Canberk Atik[1] · Mustafa Erdem[2] · Baris Yalcin[4] · Gürkan Cetin[4] · Nazım Kemal Ure[3]

Recep Ayzit
ayzit18@itu.edu.tr

Onur Akgün
akgun@tau.edu.tr

Kamil Canberk Atik
atik20@itu.edu.tr

Mustafa Erdem
erdemm@itu.edu.tr

Baris Yalcin
barisyalcin@havelsan.com.tr

Gürkan Cetin
gcetin@havelsan.com.tr

Nazım Kemal Ure
ure@itu.edu.tr

[1] Faculty of Aeronautics and Astronautics, Istanbul Technical University, Maslak, Istanbul 38000, Turkey

[2] Faculty of Engineering, Mechatronics Engineering, Turkish-German University, Beykoz, Istanbul 38000, Turkey

[3] Faculty of Computer and Informatics Engineering, Istanbul Technical University, Maslak, Istanbul 38000, Turkey

[4] Havelsan, Havelsan, Çankaya, Ankara 38000, Turkey