Journal of Intelligent & Robotic Systems (2022) 104: 47
https://doi.org/10.1007/s10846-022-01593-5

REGULAR PAPER

Vision-Based 2D Navigation of Unmanned Aerial Vehicles in Riverine Environments with Imitation Learning

Peng Wei[1] · **Ryan Liang**[1] · **Andrew Michelmore**[2] · **Zhaodan Kong**[1]

Received: 26 August 2021 / Accepted: 3 February 2022 / Published online: 4 March 2022

Abstract

There have been many researchers studying how to enable unmanned aerial vehicles (UAVs) to navigate in complex and natural environments autonomously. In this paper, we develop an imitation learning framework and use it to train navigation policies for the UAV flying inside complex and GPS-denied riverine environments. The UAV relies on a forward-pointing camera to perform reactive maneuvers and navigate itself in 2D space by adapting the heading. We compare the performance of a linear regression-based controller, an end-to-end neural network controller and a variational autoencoder (VAE)-based controller trained with data aggregation method in the simulation environments. The results show that the VAE-based controller outperforms the other two controllers in both training and testing environments and is able to navigate the UAV with a longer traveling distance and a lower intervention rate from the pilots.

Keywords Imitation learning · Vison-based control · Unmanned aerial vehicle · Riverine environments

1 Introduction

Multirotor unmanned aerial vehicles (UAV) have achieved considerable success in the past few years due to their high maneuverability and vertical take-off and landing capabilities. The modern UAVs have been deployed in a wide range of applications such as remote inspection, precision agriculture, search and rescue, aerial photography, site surveying and package delivery [19, 20, 35, 36, 54]. Despite its success, autonomous navigation of UAVs with obstacle avoidance capability in outdoor environments remains a challenge especially when the environment becomes complex and unknown (e.g., GPS-denied riverine environments involve heavy foilage/forest canopy, see Fig. 1). In this paper, we developed a navigation policy which allows the UAV to fly in riverine environments solely relying on the visual inputs and is trained with machine learning algorithm. Our drone is able to navigate itself autonomously while avoiding collision with nearby obstacles in the simulation environments.

✉ Zhaodan Kong
zdkong@ucdavis.edu

[1] Department of Mechanical and Aerospace Engineering, University of California, Davis, CA 95616, USA

[2] Advanced Farm Technologies, Davis, CA 95618, USA

Traditional approaches to navigating a robot in complex and GPS-denied environments usually integrate the visual-inertial odometry (VIO) or simultaneous localization and mapping (SLAM) techniques [10, 50] with trajectory planning. The procedure consists of localizing the agent with perception sensors (e.g., LiDAR, camera), building a map of global or local environment, and planning feasible trajectories within the map. The optimal trajectory is then mapped to the control commands of the robot in order to reach the goal points as well as avoid collisions. While these planning-based approaches have been widely adopted by many scholars [11, 57], the algorithm itself can be very computationally intensive and does not guarantee the performance when environment is non-static. The separation of perception and control may also cause unexpected behaviors as pointed out in [31]. In our work, we designed a reactive controller of which the control commands are directly calculated from the visual inputs. The proposed controller is able to provide feasible solutions to the UAV navigation inside complex riverine environments efficiently and effectively.

Learning-based approaches have achieved great success in solving sequential decision-making problems, for example, in the field of autonomous driving [5] and playing computer games [34]. Among them, reinforcement learning (RL) has gained many attentions due to its strong capability and compelling results [37, 51]. Although proven successful in many

Fig. 1 A drone flying in GPS-denied riverine environments requires a navigation policy

tasks, the RL approach is known to be sample inefficient and require a substantial amount of data in order to achieve good results, which makes it unsuitable for many safety-critical systems in the real world. On the other hand, the imitation learning (IL) [1, 4], also called learning from demonstrations (LfD), is another attractive approach for a robot to learn a safe control strategy by mimicking an expert's behavior based on the demonstrations. The IL approach overcomes many of the limitations of reinforcement learning, and thus is adopted to learn a navigation policy in our work. We want to utilize the human knowledge and let the autonomous agent learn a good behavior directly and more efficiently from human demonstrators instead of the expensive trail-and-error methods used in the reinforcement learning.

In this paper, we built a learning pipeline and proposed several vision-based policies which allows the UAV to fly inside GPS-denied riverine environments. The controller computes yaw rate commands directly from visual inputs of a front facing camera and navigates the UAV in two-dimensional space with a fixed altitude. The UAV should learn a good behavior through the training data given by the human pilot and demonstrate its performance in novel environments which the agent has never seen before. To overcome the issues of classical supervised learning, we adopted an intervention-based data aggregation (DAgger) algorithm and trained different control policies which command the UAV to perform reactive maneuvers in the simulation environments. We compared the performance of a VAE-based controller with a linear regression-based controller and an end-to-end neural network controller trained from 15 human subjects in the simulation. We evaluated the performance of different controllers by deploying them in novel environments which the agent has never seen during the training. The simulation results show that the VAE-based controller outperform the linear regression-based controller and end-to-end neural network controller with a lower intervention rate from the pilots and a longer distance traveled by itself. The VAE-based controller also generalizes well to the novel environments. In summary, the main contributions of this paper are:

1. We develop an imitation learning framework and use it to train navigation policies for the UAV flying inside GPS-denied riverine environments. The UAV only relies on a forward pointing camera to perform reactive maneuvers and navigate itself.
2. We compare the performance of different vision-based controllers trained from 15 human subjects in the simulation environments. Based on the results, we find out that the VAE-based controller outperforms a linear regression-based controller and an end-to-end neural network controller with a lower intervention rate from the pilots and a longer traveling distance.
3. We evaluate the performance of our trained control policies in novel environments which the agent has never seen during the training. The results show that the VAE-based controller generalizes well to the novel environments and can navigate the UAV autonomously.

It's noted that while in this paper, we focus on the problem of UAV navigation in riverine environments, the proposed method can be applied to other complex environments which require vision-based solutions and human demonstrations can be collected to benefit the training and therefore allows the agent to learn a good behavior. Section 2 introduces the related works. Section 3 introduces our proposed method and experiment setup in the simulation environments. We provide the experimental results in Section 4 and our discussions in Section 5. Section 6 concludes the paper.

2 Related Work

There has been a wide variety of work done on navigating a UAV in challenging environments while assuring collision avoidance in the literature. A motion capture system is generally used indoors to get the accurate state of the UAV and then trajectory optimization-based methods can be applied to achieve safe maneuvers in cluttered indoor environments [33, 41]. For outdoor environments, the GPS and other exteroceptive sensors are commonly used to estimate the position of the robot [27, 46]. However, in real-world scenarios when the UAV is flying in heavy vegetation, foilage and forest canopy, the GPS signals will be significantly degraded or fully absent and therefore advanced navigation policy is necessary.

Early researches have attempted to navigate a rotorcraft inside riverine environments but still required intermittent GPS signal [7, 46]. Recent techniques which have been evolved by researchers to allow UAV navigating inside GPS-denied environments involve lidar-based and vision-based approaches. Carrying a lidar on multirotor aircraft may heavily impact the flight time due to a limited payload and power source [47], therefore making vision-based approach a popular choice [3, 9, 11, 32, 38, 45]. Two main categories of vision-based solutions include planning-based approaches [15, 22] and reactive approaches [38, 44]. For planning-based approaches, visual-inertial odometry [10] or simultaneous localization and mapping [50] are commonly used to localize the agent and build a map of the unknown environment. Path plannings are performed after retaining a map of the environment and the optimal trajectory is selected to maneuver the robot [52, 58]. The drawback of this technique is that the state estimation may introduce bias into the system, and an incorrect map or localization result will significantly impact the planning performance and therefore leads to unsafe behaviors. Besides, planning-based approaches require a lot of processing power not only for the 3D mapping, but also calculating the optimal trajectory from the candidates [39, 53]. Visual-teach-and-repeat (VT&R) is another appealing method which allows the robot to operate in challenging environments replying on only visual inputs [12]. In the teaching pass, a human operator or a high-level mission planner provides a demonstrated path, and the map and keyframes from the visual odometry are recorded. In the repeating pass, the robot will localize and plan a path to track the pose of keyframes accordingly. The VT&R has been tested successful in indoor environments [56] and outdoor long-range rover autonomy by using stereo camera [12] and monocular vision [8], and can also deal with the seasonal changes of the environments [25]. However, the VT&R requires the robots to repeat a previously traversed route since it utilizes the maps built in the teaching phase. This method is useful under the scenario that the operation environments remain the same, such as an UAV emergency return-to-launch during a GPS failure [55]. However, the VT&R does not work in new environments in which no route has been demonstrated before, therefore cannot provide a generalizable performance.

On the other hand, a reactive approach directly generates motion commands based on sensor readings which can generally provide faster responses and require less computational resources. Existing researches have shown that a reactive controller is able to navigate a UAV in the forest [38, 44] and urban environments [31] successfully with visual inputs, and can generalize well to the new scenes different from the training environments. Traditional methods calculating optical flow [11, 29] or stereo vision [18, 32] provide depth estimation in the unknown environments. However, these techniques request powerful computing resource and add extra delay to the navigation tasks [38]. Since the success of deep neural network (DNN) in ImageNet image classification competition [26], increasing efforts have been spent on adopting DNN to learn low-dimensional control commands from high-dimensional image inputs with deep learning algorithms. Even though the training of deep learning methods takes a long time to converge due to its data-driven nature, the deep neural network structure allows it to generate control action directly from raw image data, and therefore, the total execution time is shorter than the classical perception, planning and control framework. The RL is one popular choice and a lot of work have been done in recent years. For example, Mnih et al. [34] used deep Q-network to train an agent playing Atari games based just on pixel image inputs and achieved super-human level performance. Levine et al. [30] used guided policy search with deep convolutional neural networks to train a robotic visuomotor policy that can perform a range of real-world manipulation tasks on unmolded objects. However, the RL algorithm is known to be sample inefficient and requires costly data collection procedure. This causes severe problems for training with a safety-critical system. In contrast, IL is another appealing approach which allows the agent to learn the expert's behaviors from demonstrations and has proven to be useful for many real-world problems. For example, Bojarski et al. [5] proposed an end-to-end learning approach for self-driving cars and tested on the road. Giusti et al. [13] used the imitation learning approach for UAVs to traverse forest trails based on monocular visual perception of trees and foilage. These advantages make IL a good alternative to learn safe behaviors while expert demonstrations are available.

The classical behavioral cloning solves the IL as a supervised learning problem which assumes that the data is independent and identically distributed. This assumption does not hold on real-world data, therefore even small error may compound over time and lead the agent to

a state that it has never seen during the training. This shifted distribution may cause catastrophic failures when the agent does not know how to recover [40]. To overcome these problems, Ross et al. [42] proposed a non-regret online learning approach called data aggregation (DAgger) which learns optimal policy through multiple iterations. The DAgger solves the distribution shift issue by rolling out the learned policy in environments and collect new data from the learner's own distribution. However, the DAgger has its own drawbacks, for example, the agent is allowed to execute the policy that is not fully trained while the expert does not have sufficient control authority. Human experts also lack the feedback from the system trained on which is likely to degrade the quality of the provided labels [43]. Considering these two facts, in this work, we deployed an intervention-based DAgger algorithm so that the human pilot can always take over the control when the UAV has reached an unsafe region and provide recovery actions. Relevant work [2, 14, 21] have shown that the intervention-based approach can learn a policy more effectively and achieve better performance.

3 Methodology

The proposed method is introduced in this section. We start with introducing the simulation environments designated for the training, then talk about different options for the vision-based control policies and explain our imitation learning framework. Finally, we discuss our human subject experiment.

3.1 Simulation Environments

The synthetic environments we used to train our navigation policies in were created in the Unreal Engine 4. The drone simulator is powered by the Microsoft AirSim [48] which provides useful tools for low-level UAV controls and computer vision and can be directly integrated into the Unreal Engine. High-fidelity riverine environments were designed in the Engine to simulate the real-world looks of rivers and foilage used for the training. These custom riverine environments or maps are divided into three difficulty levels (i.e., easy, medium, hard). In the easiest maps, rivers are wider and more straight while in the hardest maps, rivers are narrower and more curved. This variety avoids the issue of over-fitting to specificities of the simulation environments or maps. Example maps are displayed in Fig. 2.

3.2 Controller

Different from other researches which assume that the pilot has a global observability about the environment, we let the pilot share the same observability as the agent and only rely on the onboard camera to command the drone. For the vision-based navigation, we considered three different controller options in this paper: 1) a linear regression-based controller, 2) an end-to-end neural network controller, 3) a variational autoencoder-based controller and each of them is introduced in sequence.

3.2.1 Linear Regression-Based Controller

The design of our linear regression-based controller is inspired from the work in [11] and [44]. The raw image from the camera is first passed through a feature extraction script defined by human experts and tuned manually. The features extracted from the raw images are common visual features used in navigation (e.g., ground or aerial vehicles) and popular methods in the literature of computer vision. The reason why we want to investigate the performance of this controller is that, first it has proven to work on real UAV platforms [44], and second, unlike other complicated methods which use neural networks, this type of controller is more hand-crafted, easier to tune and also converges faster. The raw RGB image which has a resolution of 320×240 is first split into 6 vertical and 8 horizontal windows without overlap. The sliding windows not only help to accelerate the computation but also provide better results empirically since not all regions of the image have equal importance in making the decision for navigation. Then visual features are calculated for each small window. Four visual features are extracted from the image in this paper, which are:

1. *Hough Transform*: Hough transform is used to calculate the dominant lines in each window. A Hough transform is applied to a window across 15 angles. This yields a matrix who's columns represent the angles and who's rows represent pixel width of the image. The rows are then reduced down by averaging every five rows. Finally the largest two values for each angle are selected and arranged into a 30 element vector and used as the Hough transformed features for the window.
2. *Law's Masks*: Law's masks are used to get information about the textures in the image. The masks are selected based on their effectiveness in traditional machine vision applications which are L5E5, L5S5, L5R5, E5E5, E5S5, E5R5, S5S5, S5R5, R5R5. L represents for the level, E represents for the edge, R represents for the ripple and S represents for the spot. Each window is first converted to YCrCb colorspace and the Y channel is filtered with all the masks. The filtered windows are then averaged to get the 9-element feature vector.
3. *Structure Tensor*: Structure tensors represent the structures and shapes in a window. The coherency and orientation angle at each point in a window is first calculated

(a) (b) (c)

Fig. 2 A subset of the environments used for training with different difficulty levels: (a) easy; (b) medium; (c) hard

and then we accumulate the coherency value with a 15-bin histogram for the entire window based on the angles. This yields features in the form of a 15-element vector.

4. *Optical Flow*: Optical flow feature shows the motion in the current frame compared to the previous one and works as a depth estimation to the image. In this paper, dense optical flow is calculated and the maximum, minimum and averaged magnitude of the optical flow in each window are taken to form a 3-element vector as our flow features.

The collected feature vectors for each window are stacked and form a big vector X for the whole image. The feature vector is normalized to have zero means and unit variances to balance the contributions of different visual features. After that, a ridge regression is performed on the data we collected from the human demonstrations and calculate the weight of different feature element. The controller outputs the control command and send it to the drone. The structure of our linear regression-based controller can be visualized in Fig. 3.

3.2.2 End-to-End Neural Network Controller

The second controller we considered is a pure end-to-end controller composed of neural networks. The end-to-end training combines the perception and control so the control policy is trained all once. The control command is generated autonomously from the pixel values read from the camera. The design of our end-to-end network is similar to the one in [31]. We remove the prediction on the probability of collision in the final layer and take RGB images as the inputs instead of grayscale images in the original paper (see Fig. 5). The architecture of our network is a single convolutional neural network consists of a ResNet-8 with 3 residual blocks followed by dropout and ReLU activation functions. The output is then passed to a fully-connected layer to carry out yaw angular velocity prediction. The dimension of the input image is 64×64 and the output is the control command to the UAV.

The structure of our end-to-end neural network is displayed in Fig. 4. When we decided the network structure for the end-to-end controller, we also considered other existing models such as the ResNet-34, ResNet-50 [16] and VGG-16 [49]. To decide which model fit our purpose the best, we trained different models on the same dataset and found out that the neural network in Fig. 4 can achieve an equivalent performance while it has a much smaller network size as well as fewer parameters to tune compared with the other candidates, therefore the training time is significantly reduced. As a result, we use this simple, lightweight and powerful neural network as one of the three controllers (Fig. 5).

3.2.3 Variational Autoencoder (VAE)-Based Controller

The third controller we considered is based on the variational autoencoder. The VAE is known to have a good capability to compress the data from high-dimensional space into

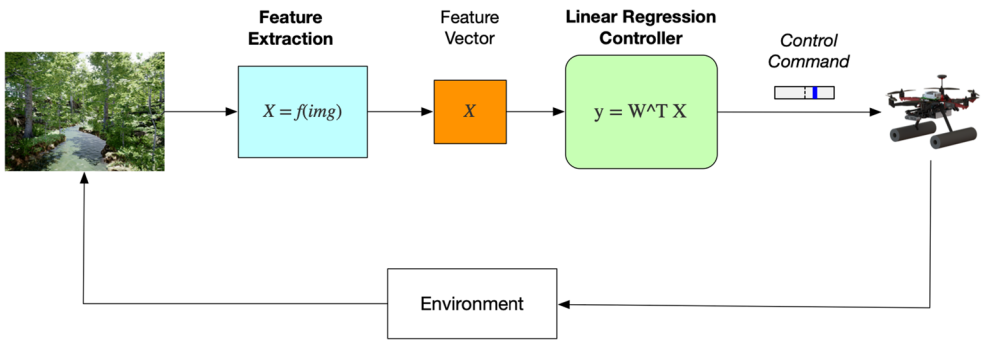


Fig. 3 System diagram of the linear regression-based controller

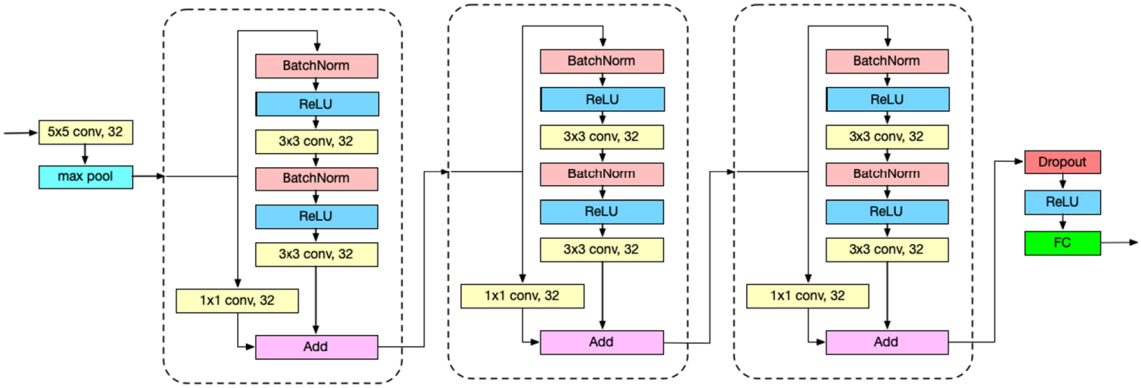


Fig. 4 The architecture of end-to-end neural network

low-dimensional space [24]. The extracted low-dimensional result, also called latent variables, normally represent for some dominant aspects of the original images (e.g., position, scale, rotation, lightning). The steps to training our VAE-based controller are twofold. First, we trained a VAE model based on the image data collected from the simulation environments. After the results of the VAE network is satisfying, we took the encoder network out from the VAE and used it to compute latent representations. The latent representations are regarded as the input to our controller network which is composed of neural network. Next, we trained our controller network by freezing the parameter values in pre-trained encoder network and only updated weights in the controller network based on the gradients.

$$\mathscr{L} = \mathbb{E}_{q(z|X)}[log(p(X|z))] - \beta\ D_{KL}[q(z|X)||p(z)] \quad (1)$$

Different types of VAE algorithms have been considered at the early stage, which include the vanilla-VAE [24], β-VAE_h [17], β-VAE_b [6], factor-VAE [23], and VAE-GAN [28]. Empirically, we found out the β-VAE_h works better than all the other structures in terms of the training stability and disentanglement ability, thus we adopted the β-VAE_h method to train our VAE model. The objective function can be seen in Eqn. Eq. 1, and β is selected to be 10. The architecture of our VAE network is presented in Fig. 6 which consists of 4 convolutional layers in encoder network and 4 transpose convolutional layers in decoder network. The reconstruction performance of our VAE is shown in Fig. 7. The reconstructed images look similar to the raw images which shows a good training result of the VAE. The structure of the VAE-based controller can be visualized in Fig. 8. The raw image is first resized to 64×64 and fed into the encoder network which was pre-trained on 200K images collected in the simulation environment, and then frozen during the training of controller network. The controller network is trained through the imitation learning. The latent representation has a dimension of 64 and the controller network is composed of fully-connected network which has two hidden layers. The output of the controller network is the control command to the UAV.

3.3 Learn from Demonstrations

The classical behavioral cloning does not perform well due to the distribution shift between the training and testing datasets and may cause catastrophic results. A more proper way to utilize the human knowledge is by taking the DAgger which the control policy is updated through multiple iterations. At each iteration, we execute the agent control policy and afterwards query human experts to provide correct commands. This method works well in many applications which typically has a slow dynamics and discrete action space.

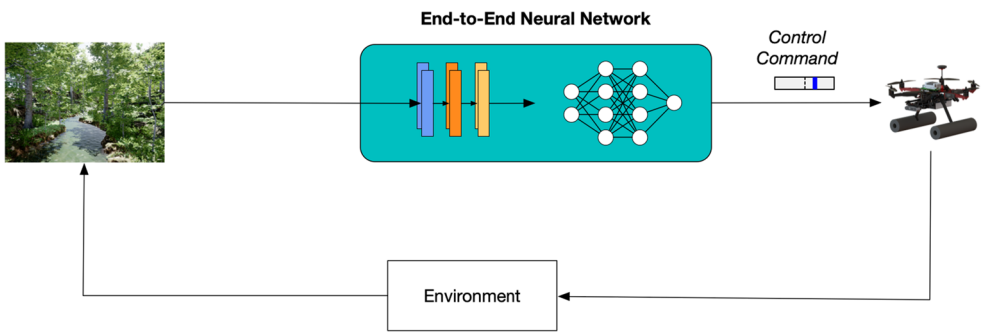


Fig. 5 System diagram of the end-to-end neural network controller

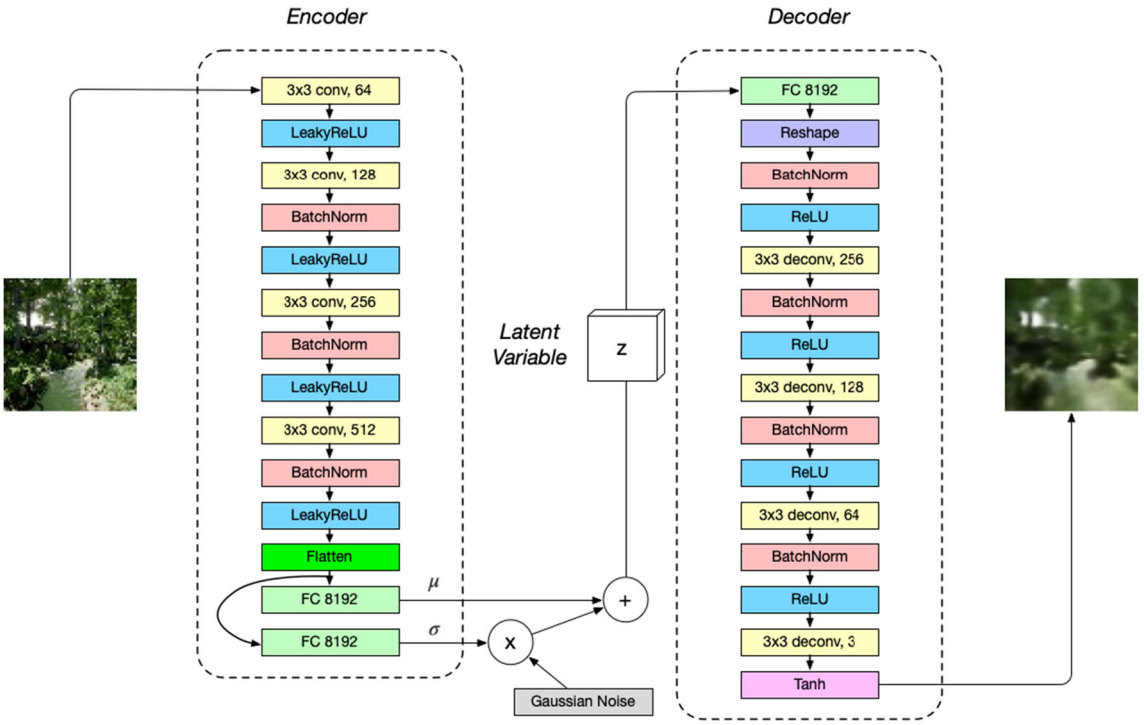
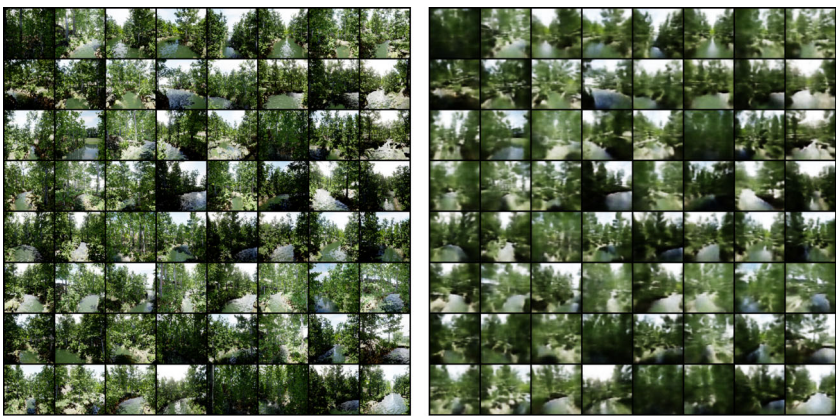


Fig. 6 The architecture of variational autoencoder network

However, there are several issues with the DAgger, for example, we have to let the agent control the robot at the early stage of the training which can be dangerous. Meanwhile, since the querying process is taken off-line, the experts lack the visual feedback from the environment which means they do not know how their corrective demonstrations will perform on the robots. Instead of querying an expert offline, the approach we take in this work is that, we query the expert in the real-time when the drone is flying in the simulation environment. The human expert or pilot has the authority to take over the control from the agent policy when he thinks the agent output may lead to a failure or unsafe situation. When the pilot takes over the control from the agent, the new demonstrations are appended to the training dataset. After a certain period of time when the pilot puts the drone back to the normal and safe condition, he can return the control back to the agent. The policy is retrained after each iteration as the normal DAgger is conducting. This method is also called intervention-based DAgger which the human expert can intervene the agent control online anytime. Another benefit with this method is that the expert has a continuous time to control the robot which is more natural in the real-life. This type of imitation learning approach has been adopted in the



(a) sample images (b) reconstructed images

Fig. 7 The reconstructing performance of our VAE, which (a) are the sample images and (b) are the reconstructed images trained after 150 epochs

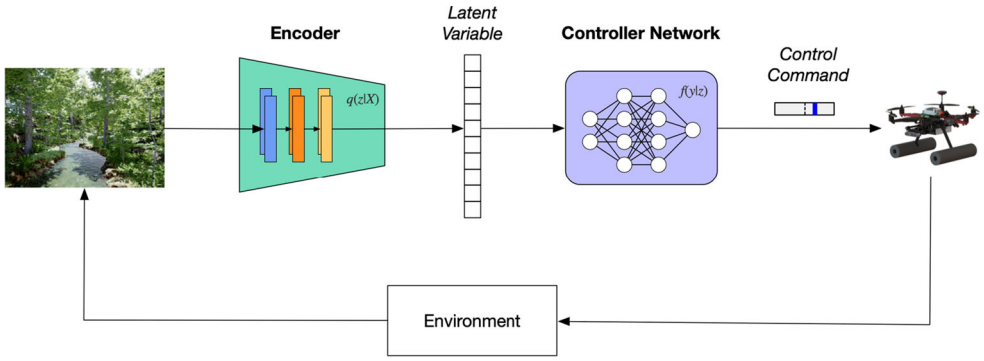


Fig. 8 System diagram of the VAE-based controller

literature [2, 21] and proven beneficial [2, 21]. The structure of our adopted intervention-based DAgger framework is displayed in Fig. 9.

3.4 Human Subject Experiment

We realized the fact that allowing the human pilots to control the height makes the imitation learning quite challenging due to the multi-model behaviors from the human pilot and causes the state space becoming too large. Also, sometimes the pilot preferred to staying at extreme locations (e.g., very close to the water or high above the river) to increase the success rate but it's lack of obstacles at these locations. Therefore, the learned policy is not useful under the normal situations. To address these issues and simplify the problem, we fixed the height of the UAV to 5 m and forward speed to 2 m/s which is achieved by a low-level controller. The yaw angular velocity was set to 0 by default. Therefore, without any input, the UAV will keep flying forward and not consider collisions along the way. We designed our vision-based controller to provide the UAV ability to avoiding obstacles along the river path based on the visual features and by generating the yaw rate control command. The human pilots were only allowed to adjust the yaw angular velocity during the demonstration. The maximum yaw angular rate was limited to 45 deg/s to avoid instability. The front facing camera data was gathered at 10 Hz and the control command was updated at the same rate.

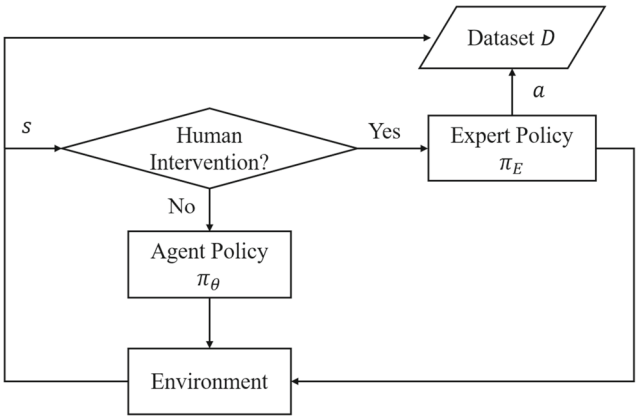


Fig. 9 A conceptual illustration of our intervention-based learning method

To compare the performance of different controllers, we recruited 15 human subjects to collect demonstrated data in the simulation environments. The 15 human subjects were divided into 3 groups, and each group contains 5 people. Group one was trained with the linear regression-based controller, group two was trained with the end-to-end neural network controller, and group three was trained with the VAE-based controller. To ensure the results are compatible, we assigned both new pilots and experienced pilots in each group to balance the knowledge level of the human subjects. The expertise levels of the human subjects are given in Table 2. Each human subject was required to fly the drone in six maps and provide corrective demonstrations using the intervention-based DAgger method to collect training data. The difficulty levels of the maps are displayed in Table 1.

With three different controller options described above, we have two hypotheses which we want to test in this work:

Hypothesis 1 *The agent trained with a VAE-based controller can achieve a better performance than an agent trained with a linear regression-based controller and an agent trained with an end-to-end neural network controller using the same DAgger method and number of iteration.*

Hypothesis 2 *The VAE-based controller generalizes well to the novel environments which the agent has never seen during the training compared to a linear regression-based controller and an end-to-end neural network controller.*

Due to the strong capability of deep neural network and the outstanding performance of VAE in the task of computer vision, in the first hypothesis, we hypothesize that the

Table 1 The level of difficulty for different maps

	Map 1	Map 2	Map 3	Map 4	Map 5	Map 6
Difficulty level	Easy	Easy	Medium	Medium	Hard	Hard

Table 2 The expertise level of different human subjects. The human subjects are split into three groups and each group consists of both new and expert pilots

Group One	Pilot 1	Pilot 2	Pilot 3	Pilot 4	Pilot 5
Expertise Level	Beginner	Beginner	Expert	Beginner	Expert
Group Two	Pilot 6	Pilot 7	Pilot 8	Pilot 9	Pilot 10
Expertise Level	Beginner	Expert	Expert	Beginner	Beginner
Group Three	Pilot 11	Pilot 12	Pilot 13	Pilot 14	Pilot 15
Expertise Level	Expert	Beginner	Beginner	Beginner	Expert

VAE-based controller will achieve the best performance among all three controller options if they are trained with the same imitation learning method and the level of training. Since the VAE compresses the high-dimensional visual inputs to low-dimensional latent representations, we believe that such compression will extract the key information from the raw data and makes the training more effective and efficient. In the second hypothesis, we want to test about the generalization capability of three controllers. Since the VAE is expected to extract the hidden dominant features in the latent space, we believe that such intermediate representations make the VAE learn a more general control policy and is less sensitive to the distribution shift between the training and testing datasets. Therefore, we have the hypothesis that the VAE-based controller will have a better performance compared to the other two controllers when deployed in the novel environment which the agent has never seen during the training. The evaluation metrics we used to compare the performance of different controller options and test our hypotheses are as follows:

- Intervention rate from the human pilots during the training.
- Maximum distance that an agent can travel without human interventions.
- Average distance flown by the agent before a failure.
- Processing time for each frame to generate the control command.

4 Results

In this section, we present the results of three controllers trained from 15 human subjects in the simulation environments. Section 4.1 shows the results in training environments and Section 4.2 describes the performance for the three controllers in testing environments. The results will be used to evaluate the performance of different vision-based control policies navigating a UAV in riverine environments and also test our two hypotheses. All the simulations and training were running on an AMD Ryzen 3900X CPU, 64 GB RAM, and RTX 2080Ti GPU computer in the lab.

4.1 Training Results

As we mentioned in the previous section, each human subject group was trained with a specific controller type using the intervention-based DAgger approach. More specifically, group one was trained with the linear regression-based controller, group two was trained with end-to-end neural network controller, and group three was trained with the VAE-based controller. For each group, all three difficulty levels of the maps were utilized and the pilots were requested to navigate the drone inside each map along the river path with random initial positions and avoid collisions. Sample trajectories of the UAV flying in some environments are provided in Fig. 10.

Ideally, the performance can keep improving with more and more data collected from the environments as an increasing number of iteration. However, empirically we found out that the pilots had a difficult time to judge whether the decision made by the agent would lead to a collision or not as the training iteration increases and the agent policy performance gets improved. As a result, the corresponding demonstrations collected from the human at those iterations may degrade the agent performance and the retrained agent policy will actually become worse. Some early training results show that the pilots are able to provide accurate and good demonstrations up to 3 iterations and therefore, our training are terminated after 3 iterations.

To test our hypotheses, first we calculate the percentage of intervention from human pilots during the training for each controller type. The results are averaged among all human subjects in the same group and the mean and variance values are provided in Fig. 11. Based on the results, we observe that the intervention rate from human pilots decreases as the training iteration increases which shows the effectiveness of our imitation learning method. Since the hard maps contain sharp curves and therefore more challenging for the UAV to navigate, the intervention rate in these maps are higher than the intervention rate in the easy and medium maps. In addition, we can see that the VAE-based controller has the lowest intervention rate across various maps with different difficulty levels while the end-to-end neural network controller requires the most human intervention compared to the other two controllers. This remains true over different training iterations. We also performed a T-test and calculated the t-score and p-value about the intervention rate between the linear regression-based and VAE-based controller, as well as the end-to-end neural network and VAE-based controller. The results are provided in Table 3. We choose the significance level $\alpha = 0.10$ and a p-value smaller than α means we can reject the null hypothesis. Based on the calculated T-test results, we find out that the VAE-based controller outperforms

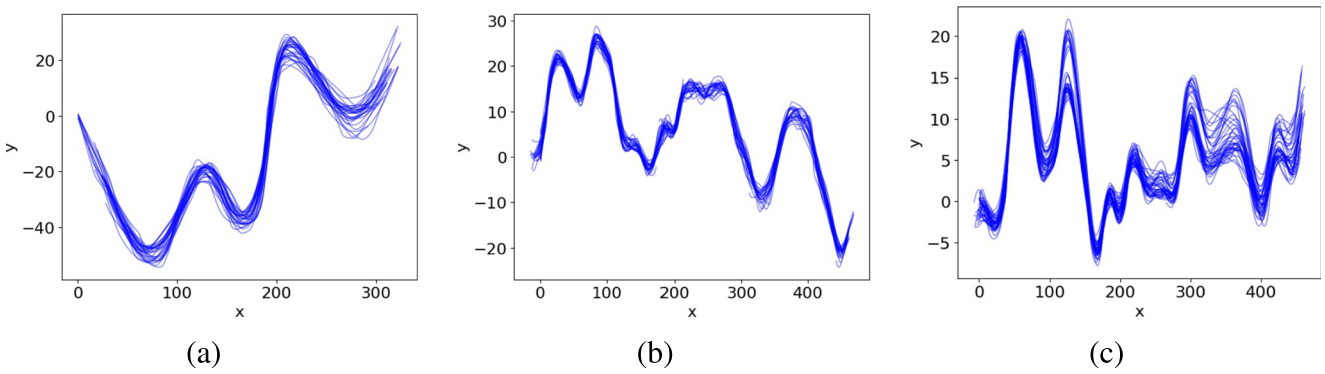


Fig. 10 Sample trajectories of the UAV in simulation environments (a) Map 1 (easy); (b) Map 3 (medium); (c) Map 5 (hard)

the linear regression-based controller and end-to-end neural network controller with significance.

4.2 Testing Results

Next, we deployed the trained controllers into testing environments. An agent that has been trained to perform well in the training environments does not inherently guarantee to perform well in a target or novel environment. Therefore, we executed our controllers trained after 3 iterations in the testing environments which the agent has never seen during the training time to evaluate their performances in novel environments.

The maximum distance that the drone can fly by itself before a failure is displayed in Fig. 12. The testing map has a total length of 225 meters and we let the agent start from the same location every time. Since all the agents start from the same initial location, the scenes they experienced remain the same and therefore the results are compatible. The distances they traveled are averaged across 5 trials and sorted by the mean value. Based on the results, we can see that the VAE-based controller can achieve the longest maximum traveling distance while the end-to-end neural network controller has the shortest maximum traveling distance before a failure. In fact, two VAE-based controllers can almost reach the end of the river without any intervention or failure. The reason why the end-to-end neural network controller achieves the worst behavior is probably because the end-to-end method highly relies on the distribution of datasets, and therefore, when the testing dataset is different from the training dataset, it cannot provide consistent performance. The linear regression-based controller achieves a better performance compared to the end-to-end neural network controller but still crashes earlier than a VAE-based controller. From this result, the VAE-based controller beats the other two controllers with significance and generalizes well to scenarios completely unseen at training time which support our two hypotheses.

The average distance traveled per failure for each controller type is shown in Fig. 13. An intervention from human pilot is performed and considered as a failure when the UAV is about to crash. The UAV started from random locations in the testing map and we counted the total distance it traveled and the number of failures it experienced. The average distances per failure are then calculated and sorted by the values. Similar to the results from the maximum distance, we can see from the figure that the VAE-based controller achieves a better performance compared to the linear regression-based controller and the end-to-end neural network controller with a higher average distance flown by itself.

Since the processing time for different maps varies slightly due to the rendering speed of the simulation, we only compared the processing time in the same map during testing. The processing time results are provided in Fig. 14 and the numeric values are shown in Table 4. Based on the results, we can see that the linear regression-based controller takes the longest time to process due to the feature extraction part while the two neural network-based controllers can predict the control command at much faster rates. The VAE-based controller is slightly faster than the end-to-end neural network controller due to a simpler structure but the difference is negligible.

We further investigated the performance of our controllers under different lighting conditions and want to see how robust each controller is to the changes of environment. To achieve that, we selected the best controller from each human subject group based on its average distance performance in the testing map. This gives us three different controllers and their performances under the normal lighting condition. We then changed the lighting of the environment to a dark condition and a bright condition and deployed the controllers in the same map. The changes of the camera view can be seen in Fig. 15. The comparison results of three controllers under different lighting conditions are displayed in Fig. 16. We found out that the VAE-based controller is less sensitive to the environment lighting changes and has a robust performance. The linear-regression-based controller performance does not change much in the bright light but

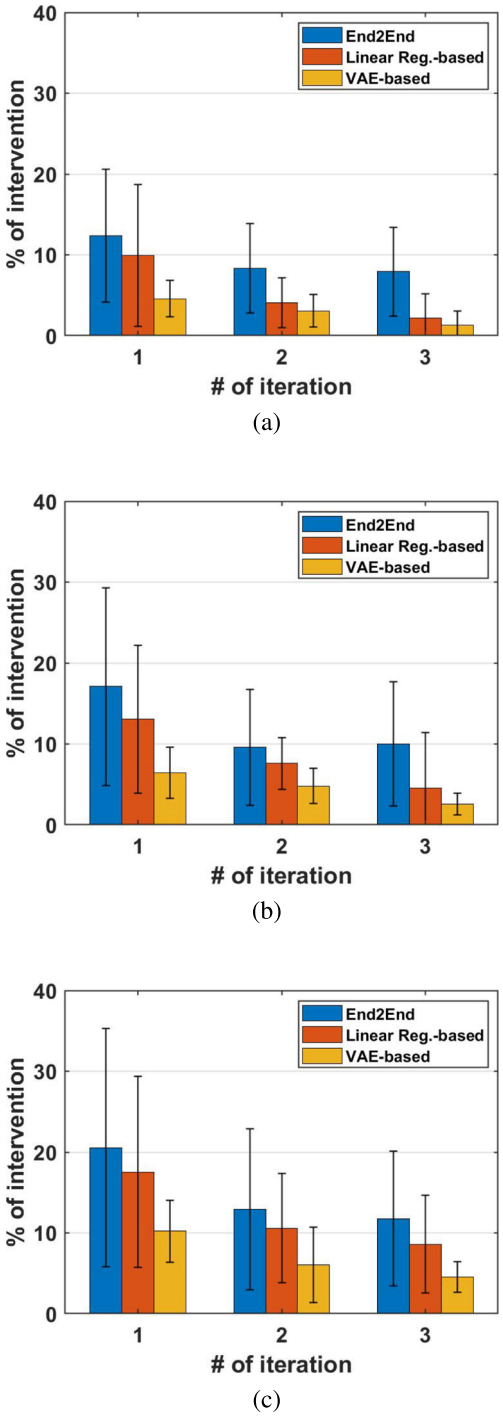


Fig. 11 Percentage of interventions from human pilots in (a) easy maps; (b) medium maps; and (c) hard maps during the training

suffers in the dark condition. The performance of end-to-end neural network controller degrades significantly for any lighting changes of the environment.

Another set of experiments were performed to investigate the effect of height errors on the controller performance. Since the UAV was commanded to fly at a fixed height during training, we want to see whether the controller is

Table 3 T-Test results with a significance level $\alpha = 0.10$

Maps	T-test	Linear reg.-based vs. VAE-based			End-to-end vs. VAE-based		
		iteration 1	iteration 2	iteration 3	iteration 1	iteration 2	iteration 3
Easy	t-score	1.8785	0.8205	0.7302	2.8860	2.8364	3.6180
	p-value	**0.08292**	0.42328	0.47583	**0.01275**	**0.01319**	**0.00280**
Medium	t-score	2.1600	2.2705	0.8890	2.6620	2.0282	2.9990
	p-value	**0.04859**	**0.03647**	0.39147	**0.01956**	**0.06201**	**0.01109**
Hard	t-score	1.8563	1.7503	2.0190	2.1440	1.9757	2.6537
	p-value	**0.08458**	**0.09809**	**0.06308**	**0.05153**	**0.06689**	**0.02103**

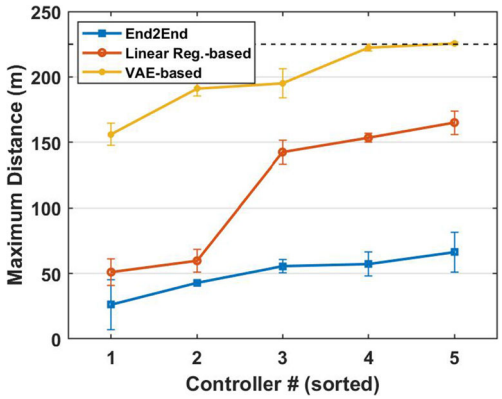


Fig. 12 Maximum distances traveled before a failure of three different controller types

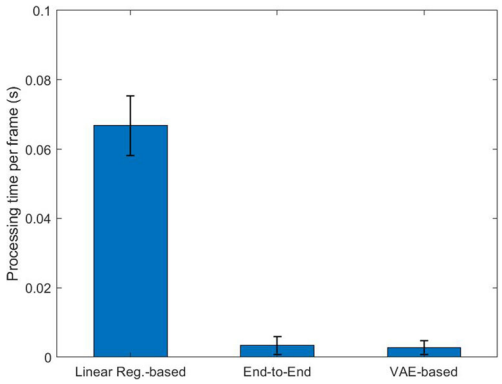


Fig. 14 Processing time of three different controller types

robust enough when height error is introduced. We used the same setting from previous experiments which consists of three controllers under nominal conditions. During the experiment, we set the target height to 3m, 5m, 7m, 9m, respectively, where 5m is the height used for training and we calculated the average distance traveled per failure under different height conditions. One thing we noticed is that since the environment is unstructured, the drone sometimes flew above the river bank to avoid obstacles at high altitudes while no collisions occurred at those locations. In order to make the comparison fair, we restricted the UAV to fly along the river path and considered it a failure if the drone attempts to pass the river bound. The performance is plotted in Fig. 17. Based on the results, we can see that the controllers are robust to small height errors. The VAE-based controller outperforms the other two controllers under various height conditions. An interesting finding is that the linear-regression-based controller experienced less collisions at lower altitude which may due to the fact that the images at low altitude contain more edge and texture information. The performances of all three controllers dropped significantly when the drone was flying at 9m. This

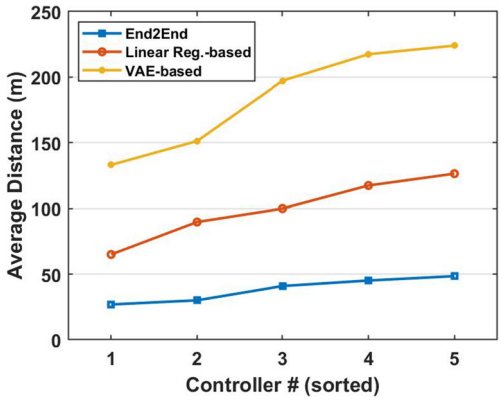


Fig. 13 Average distances traveled per failure of three different controller types

is due to the change of perspective and leads to a distribution shift issue which the images contain a large portion of sky and less visual features at high altitudes compared to the training data. One solution to it is to use a gimbal-stabilised camera to compensate for the perspective change similar to the one in [55] and collect more demonstration data at various altitudes. However, we didn't implement it in this paper and will leave it as our future work.

Based on these results, we successfully tested our two hypotheses that the VAE-based controller is better than the linear regression-based controller and the end-to-end neural network controller in the task of navigating UAV inside our simulated riverine environments with a lower intervention rate from the human pilot, a longer maximum traveling distance before a failure, a longer average traveling distance per failure, and a faster processing time. The VAE-based controller also generalizes well to the novel environments and is robust to the lighting changes of the environment. For supplementary video see: https://www.youtube.com/watch?v=aPOqHHGbZgs

We also compared the performance of one of our VAE-based controller with a visual-teach-and-repeat controller in [8, 12]. The training and testing maps are different in terms of the river shape, landscapes and lightning condition. We plot the trajectories for both environments as well as sample images along the river in Fig. 18. From the figure, we can see that the training and testing map have quite different trajectories. We also select three metrics to evaluate

Table 4 Numerical values for the processing time of three different controllers

	Processing time per frame (s)
Linear-Regression-Based Controller	0.0668 ± 0.0086
End-to-End Neural Network Controller	0.0034 ± 0.0026
VAE-Based Controller	0.0028 ± 0.0020

Fig. 15 Three different lighting conditions: (a) dark (b) normal and (c) bright from the same environment

the image similarity between two maps at a similar left-turn location (two sample images displayed in Fig. 18(a) and (b)). The metrics include the root mean squared error (RMSE), structural similarity index (SSIM), and perceptual hash (PHASH). The results of them are RMSE = 91.6 ([0,255], 0 means identical), SSIM = 0.112 ([0,1], 1 means identical), and PHASH = 0.507 ([0,1], 1 means identical), respectively, which shows that the visual inputs from the testing map can be considered novel compared to the training one. It's noted that no new trajectory information should be provided in the testing map because we want to evaluate the generalization capability of the controller in novel environments which the agent has never seen before. Therefore, the VT&R agent used the stored trajectory from the training map and visual inputs to navigate in the testing map while the VAE-based controller only relied on the visual inputs. The trajectory in Fig. 18 (b) from the human is only used for visualizing the river path and not used by either controller. Based on the experiment result, we see that in the training map (263m long in total) the VT&R controller tracked the demonstrated route very precisely and reached the final point without a failure. The VAE-based controller also managed to navigate the vehicle to the final point without a crash, but since it learned a reactive behavior, the VAE-based controller does not necessary need to follow the human trajectory. Then we tested the two controllers in the new environment (225m long in total) which both agents have never seen before. The VT&R cannot manage the task since the correct trajectory in this map was not provided. By contrast, our VAE-based controller was able to complete the mission. We didn't plot the trajectory of VT&R controller in the testing map since it always failed at the beginning. We show that even VT&R works well in the demonstrated map, this method cannot be used in our case since it requires the robot to operate in the same environment.

5 Discussion

We trained different reactive navigation policies based on human demonstrations that can reliably control a drone from a single forward-looking camera. Compared with other approaches, our proposed controller requires no planning and localization in the environment, thus save a lot processing power and computing time. We didn't perform any image pre-processing except the downsampling in order to increase the training speed. Results show that our controllers

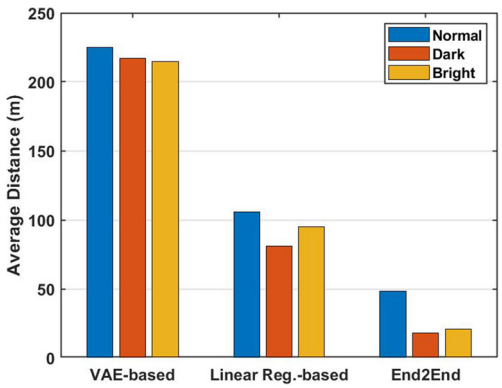


Fig. 16 Average distances traveled per failure of three controller types under different lighting conditions

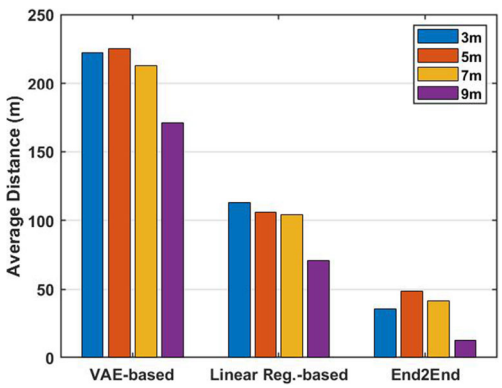


Fig. 17 Average distances traveled per failure of three controller types under different heights

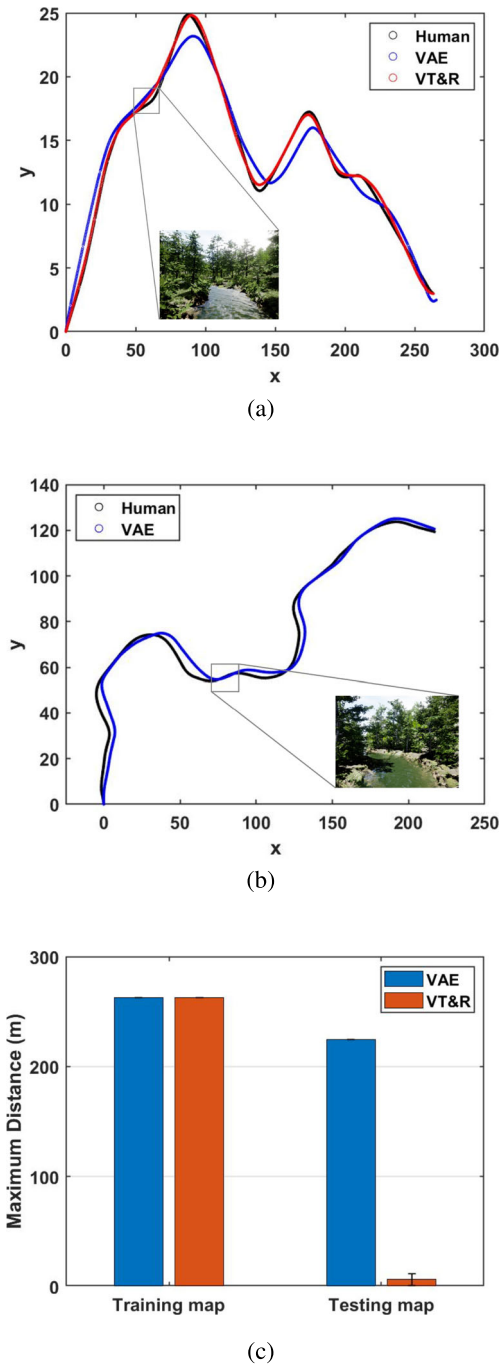


Fig. 18 Our trained VAE-based controller versus a visual-teach-and-repeat controller. (a) Trajectories of both controllers in the training (demonstrated) map, (b) Trajectories of the VAE-based controller in the testing map, and (c) Maximum distance that the agent traveled before a failure in the training and testing map

can handle the noise pretty well and learn a robust control policy. We also observed that the non-collision can be achieved without the information of the current velocity of the UAV.

Based on the training and testing results, we find out that the end-to-end neural network controller performs bad in both training and testing environments. In order to understand why it happened, we plot the histogram of the pilot demonstration data in Fig. 19. The pilots are divided into three groups evenly and each column in the figure represents for the data trained from a different controller type. Based on the histogram results, we notice that the end-to-end neural network controller tends to overfit to the existing data in general and therefore does not generalize well to the new state-action pairs if their distributions are different. With a limited amount of data, the end-to-end neural network controller will result in a worse performance compared to the other two controller types. To improve its performance, more data needs to be collected to cover the distribution of the entire space, however, this can be extremely labor intensive since the demonstrations in our work are provided from human experts. It's also observed that since the ridge regression is used to calculate the weight for the linear regression-based control, the shape of the predicted control command is more conservative and concentrates in the middle region compared to the other two neural network-based controllers. This causes the linear regression-based controller unable to generate aggressive enough commands when the drone needs a large yaw angular rate command in some maneuvers. The VAE-based controller, on the other hand, balances the distribution variance and the fitting performance, therefore compete the other two controllers.

We applied the method in [48] to investigate which part of the image is the most important for the neural network to make the decision. In Fig. 20, the VAE-based controller and end-to-end controller are supplied with the same image and we displayed the activation maps from both controllers. To make the comparison fair, we trained two controllers from the same human pilot, therefore the behavior is consistent. From the figures, we notice that the VAE-based controller concentrated on the vegetation on the right side, and it generated a left yaw command to avoid that region. In contrast, the end-to-end controller was less concentrated, attracted by different parts of the frame, and resulted in a near zero output that may lead the drone to crash. Based on this result, we believe that the VAE part benefits the controller to learn more important information from the high-dimensional visual inputs, therefore makes the training more efficient and the controller more effective. For supplementary video see: https://www.youtube.com/watch?v=xQW2wcYjHus

To further investigate when the controller may lose its performance, we plot the locations where the drone commonly crashed in the simulation environments in Fig. 21. We observe that all three controllers are likely to fail and lead to a crash when the drone is flying towards thin tree branches and leaves (see Fig. 21 (a)) which the visual algorithm may not be able to detect them. After a large control command when the UAV is off by the river path too much

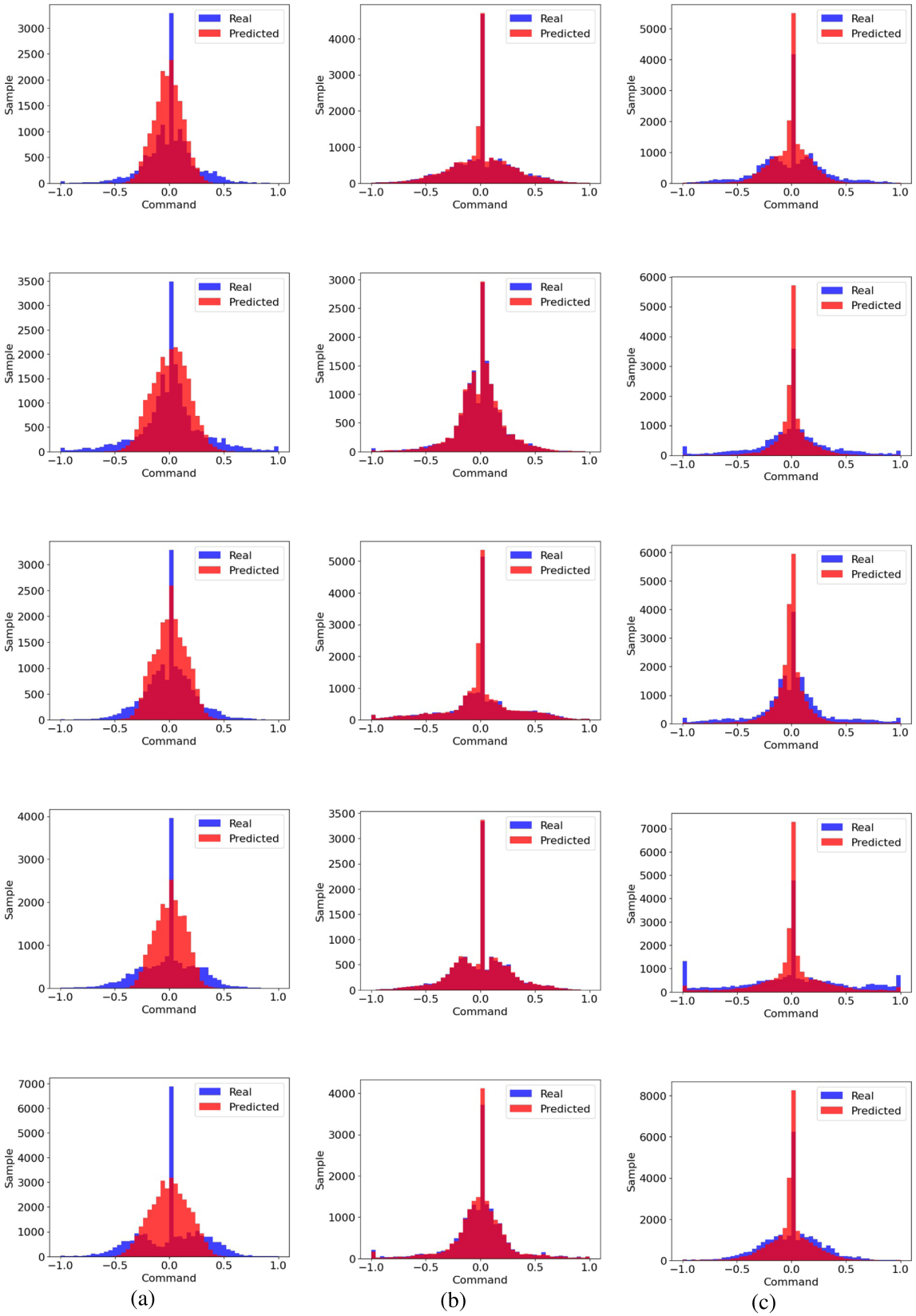


Fig. 19 Histogram of pilot demonstration data compared to the model prediction. The pilots are divided into three groups evenly, and each group contains five members. Each group is trained on a different controller type: (a) linear regression-based controller; (b) end-to-end neural network controller; and (c) VAE-based controller

(a) (b) (c)

Fig. 20 We visualize the neural network decision by showing: (a) the input image (b) the VAE-based controller activation map, and (c) the end-to-end neural network controller activation map on the same input image

and there is no river in the field of view, the UAV may also generate unexpected behaviors (see Fig. 21 (b)). The shadow existing in the environment from the trees also confuses the control policy and lead to extensive or incorrect control commands (see Fig. 21 (c)). In addition, since the drone is flying at a constant forward speed, if the obstacle is right in front of the drone and too close, it's hard to provide safe navigation even for human operators and usually lead to crashes. All of these common failure locations will be considered in the future work to improve our vision-based navigation policies.

6 Conclusion and Future Work

The paper presents an imitation learning framework and compares several vision-based control policies for the task of UAV autonomous navigation in complex outdoor environments. Training the agent in simulation allows us to demonstrate the capability of the developed framework and avoid unnecessary losses before deploying it in the real world. We introduce an intervention-based DAgger framework and use it to train a vision-based UAV capable of flying inside complex and GPS-denied riverine environments. We compare the performance of a VAE-based controller with a linear regression-based controller and an end-to-end neural network controller trained with human demonstrations in the simulation environments. The results show that the VAE-based controller outperforms the other two controller types in both training and testing processes and is able to navigate the UAV autonomously.

One drawback of our controller is that since we only train the high-level control command, the dynamic of the UAV is neglected thus the trained controller may not generalize well to other vehicle platforms with different sizes. Future works include incorporating low-level control and taking the UAV dynamics into considerations. The common failure locations in riverine environments will be further investigated and improved in the next generation of our vision-based control policies as a safety guarantee. In this work, we trained our controllers in static environments and did not consider dynamic objects. We'd like to see whether the controller can retain its performance under non-static environments by re-designing our simulation environments as a future work. The robustness of our controller can be further improved by adding a gimbal-stabilised camera and collecting more demonstration data at various height conditions. Another limitation of our work is that, even though our controllers perform well most of time, a human operator is still required to monitor and intervene if the agent displays unsafe behaviors. In the future, we want to remove the human assistance by designing a recovery plan and a riskiness evaluation metric, therefore the recovery plan will be triggered if the risk level

(a) (b) (c)

Fig. 21 The typical failure locations (a) thin branches and leaves (b) no river in the field of view, and (c) shadow from trees

is too high and it will put the agent back to a safe state automatically and continue the mission. Eventually, we will implement the proposed method in real riverine experiments to evaluate its performance and demonstrate the framework can be used for the real autonomous navigation tasks.

Acknowledgements The authors would like to thank all human subjects who helped with the data collection in the simulation environments. We would also like to thank Greesan Gurumurthy and Marie Cor Croz from the Cyber-Human-Physical Systems lab for their kind help with the project.

Author Contributions P.W., R.L., A.M., and Z.K. designed the study. R.L. created high-fidelity environments in the simulation. P.W. and A.M. developed different vision-based controllers and the imitation learning framework. P.W. and A.M. wrote the script. P.W. and Z.K. designed the experiment. P.W. coordinated the experiment and analyzed the data. All authors wrote the final manuscript.

Funding The work was supported by the Office of Naval Research (ONR) under the NEPTUNE 2.0 program (No. N00014-20-1-2268).

Declarations

Ethics Approval This study was approved by the Institutional Review Board (IRB) of the University of California Davis.

Consent to Participate Freely-given, informed consent to participate in the study has been obtained from all human subjects.

Consent for Publication All human subjects have consented to have their data published in a journal article.

Conflict of Interests The authors declare that they have no conflict of interest for this work.

References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the 21st International Conference on Machine learning, p. 1 (2004)
2. Ablett, T., Marić, F., Kelly, J.: Fighting failures with fire: Failure identification to reduce expert burden in intervention-based learning. arXiv:2007.00245 (2020)
3. Alvarez, H., Paz, L.M., Sturm, J., Cremers, D.: Collision Avoidance for Quadrotors with a Monocular Camera. In: Experimental Robotics, pp. 195–209. Springer (2016)
4. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst **57**(5), 469–483 (2009)
5. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv:1604.07316 (2016)
6. Burgess, C.P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., Lerchner, A.: Understanding disentangling in β-vae. arXiv:1804.03599 (2018)
7. Chambers, A., Achar, S., Nuske, S., Rehder, J., Kitt, B., Chamberlain, L., Haines, J., Scherer, S., Singh, S.: Perception for a River Mapping Robot. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 227–234. IEEE (2011)
8. Clement, L., Kelly, J., Barfoot, T.D.: Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery. J. Field Robot **34**(1), 74–97 (2017)
9. Daftry, S., Zeng, S., Khan, A., Dey, D., Melik-Barkhudarov, N., Bagnell, J.A., Hebert, M.: Robust monocular flight in cluttered outdoor environments. arXiv:1604.04779 (2016)
10. Delmerico, J., Scaramuzza, D.: A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 2502–2509. IEEE (2018)
11. Dey, D., Shankar, K.S., Zeng, S., Mehta, R., Agcayazi, M.T., Eriksen, C., Daftry, S., Hebert, M., Bagnell, J.A.: Vision and Learning for Deliberative Monocular Cluttered Flight. In: Field and Service Robotics, pp. 391–409. Springer (2016)
12. Furgale, P., Barfoot, T.D.: Visual teach and repeat for long-range rover autonomy. J. Field Robot **27**(5), 534–560 (2010)
13. Giusti, A., Guzzi, J., Cireşan, D.C., He, F.L., Rodríguez, J.P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., et al.: A machine learning approach to visual perception of forest trails for mobile robots. IEEE Robot. Autom. Lett **1**(2), 661–667 (2015)
14. Goecks, V.G., Gremillion, G.M., Lawhern, V.J., Valasek, J., Waytowich, N.R.: Efficiently combining human demonstrations and interventions for safe training of autonomous systems in real-time. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 2462–2470 (2019)
15. Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous uav guidance. J. Intell. Robot. Syst. **57**(1-4), 65 (2010)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
17. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: Beta-vae: Learning basic visual concepts with a constrained variational framework (2016)
18. Hrabar, S., Sukhatme, G.S., Corke, P., Usher, K., Roberts, J.: Combined Optic-Flow and Stereo-Based Navigation of Urban Canyons for a Uav. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3309–3316. IEEE (2005)
19. Iost Filho, F.H., Heldens, W.B., Kong, Z., de Lange, E.S.: Drones: Innovative technology for use in precision pest management. J. Econ. Entomol. **113**(1), 1–25 (2020)
20. Irizarry, J., Gheisari, M., Walker, B.N.: Usability assessment of drone technology as safety inspection tools. Journal of Information Technology in Construction (ITcon) **17**(12), 194–212 (2012)
21. Kelly, M., Sidrane, C., Driggs-Campbell, K., Kochenderfer, M.J.: Hg-Dagger: Interactive Imitation Learning with Human Experts. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 8077–8083. IEEE (2019)

22. Kendoul, F.: Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. J. Field. Robot **29**(2), 315–378 (2012)
23. Kim, H., Mnih, A.: Disentangling by Factorising. In: International Conference on Machine Learning, pp. 2649–2658. PMLR (2018)
24. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv:1312.6114 (2013)
25. Krajník, T., Cristóforis, P., Kusumam, K., Neubert, P., Duckett, T.: Image features for visual teach-and-repeat navigation in changing environments. Robot. Auton. Syst. **88**, 127–141 (2017)
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances Neural Inf. Process. Syst **25**, 1097–1105 (2012)
27. Kwak, J., Sung, Y.: Autonomous uav flight control for gps-based navigation. IEEE Access **6**, 37947–37955 (2018)
28. Larsen, A.B.L., Sønderby, S.K., Larochelle, H., Winther, O.: Autoencoding beyond Pixels Using a Learned Similarity Metric. In: International Conference on Machine Learning, pp. 1558–1566. PMLR (2016)
29. Lee, D.J., Merrell, P., Wei, Z., Nelson, B.E.: Two-frame structure from motion using optical flow probability distributions for unmanned air vehicle obstacle avoidance. Mach. Vis. Appl. **21**(3), 229–240 (2010)
30. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. J. Mach. Learn. Res **17**(1), 1334–1373 (2016)
31. Loquercio, A., Maqueda, A.I., Del-Blanco, C.R., Scaramuzza, D.: Dronet: Learning to fly by driving. IEEE Robot. Autom. Lett **3**(2), 1088–1095 (2018)
32. Matthies, L., Brockers, R., Kuwata, Y., Weiss, S.: Stereo Vision-Based Obstacle Avoidance for Micro Air Vehicles Using Disparity Space. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 3242–3249. IEEE (2014)
33. Mellinger, D., Kumar, V.: Minimum Snap Trajectory Generation and Control for Quadrotors. In: 2011 IEEE International Conference on Robotics and Automation, pp. 2520–2525. IEEE (2011)
34. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv:1312.5602 (2013)
35. Naidoo, Y., Stopforth, R., Bright, G.: Development of an Uav for Search & Rescue Applications. In: IEEE Africon'11, pp. 1–6. IEEE (2011)
36. Natalizio, E., Surace, R., Loscri, V., Guerriero, F., Melodia, T.: Filming sport events with mobile camera drones: Mathematical modeling and algorithms (2012)
37. Ng, A.Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., Liang, E.: Autonomous inverted helicopter flight via reinforcement learning. In: Experimental Robotics IX, pp. 363–372. Springer (2006)
38. Oleynikova, H., Honegger, D., Pollefeys, M.: Reactive Avoidance Using Embedded Stereo Vision for Mav Flight. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 50–56. IEEE (2015)
39. Oleynikova, H., Taylor, Z., Siegwart, R., Nieto, J.: Safe local exploration for replanning in cluttered unknown environments for microaerial vehicles. IEEE Robot. Autom. Lett **3**(3), 1474–1481 (2018)
40. Pomerleau, D.A.: Alvinn: An autonomous land vehicle in a neural network. In: Advances in Neural Information Processing Systems, pp. 305–313 (1989)
41. Richter, C., Bry, A., Roy, N.: Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In: Robotics Research, pp. 649–666. Springer (2016)
42. Ross, S., Bagnell, D.: Efficient reductions for imitation learning. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 661–668 (2010)
43. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 627–635 (2011)
44. Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., Hebert, M.: Learning monocular reactive Uav control in cluttered natural environments. In: 2013 IEEE International Conference on Robotics and Automation, pp. 1765–1772. IEEE (2013)
45. 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. Autom. Lett **3**(4), 2799–2806 (2018)
46. Scherer, S., Rehder, J., Achar, S., Cover, H., Chambers, A., Nuske, S., Singh, S.: River mapping from a flying robot: state estimation, river detection, and obstacle mapping. Auton. Robot. **33**(1-2), 189–214 (2012)
47. Scherer, S., Singh, S., Chamberlain, L., Saripalli, S.: Flying fast and low among obstacles. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, pp. 2023–2029. IEEE (2007)
48. Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In: Field and Service Robotics (2017)
49. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
50. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A Benchmark for the Evaluation of Rgb-D Slam Systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 573–580. IEEE (2012)
51. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT Press, Cambridge (2018)
52. Tordesillas, J., How, J.P.: PANTHER: Perception-aware trajectory planner in dynamic environments. arXiv:2103.06372 (2021)
53. Tordesillas, J., Lopez, B.T., Carter, J., Ware, J., How, J.P.: Real-time planning with multi-fidelity models for agile flights in unknown environments. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 725–731. IEEE (2019)
54. Valavanis, K.P., Vachtsevanos, G.J.: Handbook of unmanned aerial vehicles, vol. 1 Springer (2015)
55. Warren, M., Greeff, M., Patel, B., Collier, J., Schoellig, A.P., Barfoot, T.D.: There's no place like home: Visual teach and repeat for emergency return of multirotor uavs during gps failure. IEEE Robotics and Automation Letters **4**(1), 161–168 (2018)
56. Wen, H., Clark, R., Wang, S., Lu, X., Du, B., Hu, W., Trigoni, N.: Efficient indoor positioning with visual experiences via lifelong learning. IEEE Trans. Mob. Comput. **18**(4), 814–829 (2018)
57. Yang, J., Rao, D., Chung, S.J., Hutchinson, S.: Monocular vision based navigation in Gps-denied riverine environments. In: Infotech@ Aerospace 2011, p. 1403 (2011)
58. Zhou, B., Gao, F., Wang, L., Liu, C., Shen, S.: Robust and efficient quadrotor trajectory generation for fast autonomous flight. IEEE Robot. Autom. Lett **4**(4), 3529–3536 (2019)

Peng Wei received the B.S. in energy and power engineering from Huazhong University of Science and Technology, China, and the M.S. in mechanical engineering from Arizona State University, USA, in 2015 and 2016, respectively. He is currently working towards his Ph.D. degree in mechanical and aerospace engineering from University of California, Davis, USA. His research interests include UAV control, optimal control and machine learning.

Ryan Liang is a R&D Aeronautical Engineer at Sandia National Labs. He received his Bachelor's and Master's degree in Mechanical and Aerospace Engineering from University of California, Davis in 2019 and 2021 respectively. His main interests include advancing and delivering seeker technologies, hypersonic vehicle controls, and national defense capabilities.

Andrew Michelmore is a software engineer working at Advanced Farm Technologies. He received his Bachelor's and Master's degrees in electrical engineering from Santa Clara University. He is interested in quadcopter dynamics and controls and enjoys flying and testing control systems on this platform. His work focuses on motion control and state estimation in robotics.

Dr. Zhaodan Kong is an Associate Professor in Mechanical and Aerospace Engineering at University of California, Davis. He received his Bachelor's and Master's degrees in Astronautics and Mechanics from Harbin Institute of Technology, Harbin, China, in 2004 and 2006, respectively, and his Ph.D. degree in Aerospace Engineering with a minor in Cognitive Science from the University of Minnesota, Minneapolis, MN, USA, in 2011. Before joining UC Davis in 2015, he was a Postdoctoral researcher at Boston University. His research focuses on control theory, machine learning, formal methods, and their applications to aerial robotics, humanmachine systems, and neural engineering.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com