Neural Computing and Applications (2020) 32:4227–4238
https://doi.org/10.1007/s00521-019-04330-6

EMERGING TRENDS OF APPLIED NEURAL COMPUTATION - E_TRAINCO

# Continuous drone control using deep reinforcement learning for frontal view person shooting

Nikolaos Passalis[1] · Anastasios Tefas[1]

Received: 31 October 2018 / Accepted: 28 June 2019 / Published online: 8 July 2019


**Abstract**
Drones, also known as unmanned aerial vehicles, can be used to aid various aerial cinematography tasks. However, using drones for aerial cinematography requires the coordination of several people, increasing the cost and reducing the shooting flexibility, while also increasing the cognitive load of the drone operators. To overcome these limitations, we propose a deep reinforcement learning (RL) method for continuous fine-grained drone control, that allows for acquiring high-quality frontal view person shots. To this end, a head pose image dataset is combined with 3D models and face alignment/warping techniques to develop an RL environment that realistically simulates the effects of the drone control commands. An appropriate reward-shaping approach is also proposed to improve the stability of the employed continuous RL method. Apart from performing continuous control, it was demonstrated that the proposed method can be also effectively combined with simulation environments that support only discrete control commands, improving the control accuracy, even in this case. The effectiveness of the proposed technique is experimentally demonstrated using several quantitative and qualitative experiments.

**Keywords** Deep reinforcement learning · Continuous control · Aerial cinematography · Drone control

## 1 Introduction

The wide availability of drones, which are also known as unmanned aerial vehicles (UAVs), led to the development of versatile autonomous systems that can be used to aid several challenging tasks, ranging from promptly responding to medical emergencies [7], to detecting fires in forests [21], and protecting the wildlife [15]. The high versatility of drones and their ability to capture spectacular aerial shots make them especially suitable for performing aerial cinematography tasks [27]. However, flying a drone in this setting requires the coordination of several people. A pilot has to control each drone, while a camera operator has to control the shooting camera on each drone. At the same time, the director has to coordinate several pilots and camera operators, if multiple drones are used, and ensure the quality of the captured shots. This situation increases the cost of using drones for aerial cinematography and severely limits the shooting flexibility by putting a significant cognitive load on the director and drone/camera operators.

The aforementioned limitations led to the development of various techniques for assisting drone-based cinematography, ranging from techniques for automated planning for multiview drone shooting [27], and crowd avoidance for complying with drone legislation [45], to autonomous drone control systems [11], and various techniques for steering drone video shooting [32]. These techniques automate various parts of the shooting process, reducing the cost of aerial cinematography and the cognitive load of human operators. At the same time, the computationally complexity of the developed methods must be taken into account, since drones usually have limited computational resources (processing power and memory), leading to the development of various approaches to lower the computational requirements of the aforementioned methods [29, 35].

✉ Nikolaos Passalis
passalis@csd.auth.gr

Anastasios Tefas
tefas@csd.auth.gr

[1] Department of Informatics, Aristotle University of Thessaloniki, 541 24 Thessaloniki, Greece

Currently, we are still far from developing fully autonomous drones that would be able to automatically shot professional-grade footage according to a predefined plan, as designed by the director, while detecting salient events for performing opportunist shooting. Instead, this paper focuses on automating part of the shooting process. More specifically, we aim to appropriately control the height and the relative position of the drone with respect to the person of interest to acquire a clear frontal shot. To this end, a framework for developing deep reinforcement learning (RL) methods, that are able to learn accurate control policies for the *continuous fine-grained* control of a drone from raw pixel inputs, is presented. Note that even though deep RL has been applied for solving several challenging and delicate control tasks [8, 14, 50], that would usually require the difficult and laborious fine-tuning of traditional control methods [3], little work has been done for developing deep RL techniques for cinematography-oriented drone control.

The contributions of this paper are briefly summarized below. First, a realistic simulation environment is developed using the Head Pose Image Database (HPID) [13]. However, the HPID only contains a discrete number of poses for each person, supporting only discrete control commands. To overcome this limitation and effectively simulate the effect of continuous control commands, the simulation environment is combined with 3D models, along with the face alignment and warping technique proposed in [51]. This allows for training RL methods that will perform continuous fine-grained control. Then, an appropriate reward-shaping approach is proposed to improve the stability of the employed continuous RL method. Apart from performing continuous control, it is demonstrated that the method introduced in this paper can be also effectively combined with simulation environments that support only discrete control commands, improving the control accuracy. Finally, the proposed approach is compared both to an RL method that performs discrete control, and to a traditional controller that directly uses the output of a deep model that performs pose estimation. It is experimentally demonstrated that the proposed approach improves the control accuracy over these methods.

This paper is an extended version of our previous work [33], where a method capable of performing drone control in discrete steps for frontal view shooting was proposed. This paper significantly extends our previous work by proposing a method capable of performing continuous fine-grained drone control using deep RL. The proposed method is called "continuous drone control" (CDC) through the rest of this paper. First, the limitations of the simulation environment that was used in [33, 36] are lifted by using an advanced 3D modeling method that can simulate the effect of continuous control commands [51]. This allows for developing novel fine-grained continuous control methods for frontal view shooting using deep RL. Second, the proposed approach is appropriately extended and evaluated using a policy gradient method, instead of Q-learning, allowing for directly performing continuous control without the need of discretizing the action space. Finally, CDC can be also readily used in simulation environments that only support discrete action spaces. Indeed, it is experimentally demonstrated that CDC can improve the control accuracy over control methods that perform discrete control, even when environments that only support discrete control are used. To further boost RL research, that critically relies on the availability of simulation environments, we provide an open-source implementation of the developed simulation environments as well as of the developed techniques at https://github.com/passalis/continuous_drone_frontal_rl.

The rest of the paper is structured as follows: The related work is briefly discussed and compared to the proposed approach in Sect. 2. Then, CDC is introduced and described in detail in Sect. 3. The experimental evaluation is provided in Sect. 4. Finally, future work is discussed and conclusions are drawn in Sect. 5.

## 2 Related work

Several approaches can be used to appropriately control a drone/camera according to a specific cinematography-oriented objective. Perhaps the most widely used approach is to project a set of known 2D landmark points, e.g., nose, eyes, mouth, etc., into the 3D space and solve the corresponding Perspective-n-Point (PnP) problem to estimate the pose of the head of the person of interest [19, 30]. Then, a PID controller can be used to appropriately control the drone to acquire the desired frontal shot [1, 34]. However, this approach requires accurately detecting several 2D facial landmark points, which can be especially difficult if a low-resolution image input is used. Note that this is usually the case, since drones frequently use a low-resolution video feed for performing the on-board processing tasks, even when high-resolution cameras are available, due to their limited processing power and memory. Apart from these, careful calibration of the system is required and it is non-trivial to extend this approach for estimating the pose of other objects, since the landmark points have to be appropriately redefined and a corresponding detector must be developed.

Some of the aforementioned drawbacks can be addressed by using deep pose estimation algorithms that are directly trained to estimate the pose of various objects using a training set of data that contains images along with their pose annotations [12, 38, 43]. Even though such deep

learning approaches have been shown to increase the pose estimation accuracy and lead to more robust pose estimators, there is no guarantee that they will be *optimal* for performing control tasks, while, on the other hand, developing optimal control algorithms has a long history in various engineering fields [5, 20]. Quite recently, it was shown that combining the great learning capacity of deep learning models with reinforcement learning (RL) techniques can indeed lead to the development of robust control policies, that can work under stochastic and noisy environments, producing spectacular results, that often outperform humans on sophisticated control tasks [24, 26, 46]. This also led to the development of several recent deep RL approaches for various robotics applications [9, 10, 22, 33, 40]. However, using a deep RL approach for learning accurate control policies from raw pixel inputs is not straightforward, requiring the development of the appropriate simulation environments along with the careful encoding of the problem objective into a reward function. Apart from that, *reward shaping* can be also required to ensure that the method will indeed perform as it was intended and ensure the stable convergence of the learning process [28]. To the best of our knowledge, this is the first work where deep RL is used to perform accurate continuous fine-grained drone control from raw pixel input for frontal view shooting using a dedicated simulation environment that employs an advanced 3D modeling method that can simulate the effect of continuous control commands.

## 3 Proposed method

The proposed CDC method is presented in this Section. First, a brief introduction to RL is provided along with the used notation. Then, the developed simulation environments are presented. Finally, the proposed reward-shaping scheme, as well as the complete methodology for developing a deep RL agent for continuous drone control, are derived and discussed in detail. Note that the term *agent* in this paper refers to parameterized software agents that learn to individually perform various actions in order to maximize the reward obtained from the environment and should not be confused with multiagent systems [48].

### 3.1 Deep reinforcement learning

Given an environment for which the *Markov property* is satisfied (the future state depends only on the current state and the selected action), RL can be modeled using Markov decision processes (MDPs). A MDP can be defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}(\cdot), \mathcal{R}(\cdot), \gamma)$, where:

1. $\mathcal{S}$ is the set of possible states for the environment.
2. $\mathcal{A} = \{a_1, a_2, \ldots, a_{N_a}\}$ is the set of possible actions, where $N_a$ is the number of possible actions. Note that the action space can be continuous: $\mathcal{A} \subseteq \mathbb{R}^m$, where $m$ is the number of controls.
3. $\mathcal{P}(s_{t+1}|s_t, a_t)$ is the probability that the environment will transit from the state $s_t$ to the state $s_{t+1}$ given that the action $a_t$ has been performed.
4. $\mathcal{R}(s_t, a_{t_1}, s_{t+1}) = r_{t+1}$ is the *immediate* reward that the agent receives when performing the action $a_{t+1}$ and transitioning from state $s_t$ to state $s_{t+1}$.
5. $\gamma$ is a discount factor that defines the importance of immediate rewards versus future rewards, where typically $0 \leq \gamma < 1$. For $\gamma = 0$ the agent is *short-sighted*, while higher values increase the importance of future rewards.

An agent starts at state $s_0$ and selects the next action according to a *policy* $\pi(s)$ that defines the action that the agent will perform. Note that the policy $\pi(\cdot)$ can be stochastic and it is usually described as a probability distribution over the available actions: $\pi : \mathcal{S} \rightarrow \mathcal{P}(A)$. The agent's behavior defines a sequence of state–action–rewards $s_0, a_1, r_1, s_1, a_2, r_2, s_2, a_3, r_3, s_3, \ldots, s_T$ that describes the history of the agent for a given episode that ends after $T$ steps. The discounted accumulated return for an episode consisting of $T$ steps is defined as:

$$R = \sum_{t=1}^{T} \gamma^{t-1} r_t, \tag{1}$$

where each action is selected according to the policy $\pi(s)$. RL aims to learn the optimal policy $\pi^*$ that provides the maximum expected return:

$$\pi^* = \arg\max_{\pi} \mathbb{E}[R|\pi]. \tag{2}$$

Several approaches have been proposed for tackling this problem [39]. For example, Q-learning aims to learn the optimal action-value function $Q^*(s, a)$, which is used to express the expected reward of performing the action $a$ from the state $s$, given that an optimal policy is then followed. Then, the optimal policy can be directly derived as:

$$\pi^*(s) = \arg\max_{a} Q^*(s, a). \tag{3}$$

Even though simple look-up tables can be used to represent and store the optimal values ($Q$-values), this approach quickly becomes impractical as the size of the state set $\mathcal{S}$ increases. To overcome this limitation, deep neural networks can used to approximate the action-value function (deep Q-learning). The neural network is updated after each time step using stochastic gradient descent [17], to minimize an appropriately defined loss function $\mathcal{L}(\cdot)$ that

measures the error between the current estimation $Q(s_t, a_t, \mathbf{W}_t)$ and the updated estimation, where $\mathbf{W}_t$ denotes the parameters of the neural network after $t$ optimization steps. The difference between the current estimation and the updated estimation is defined as:

$$\delta = Q(s_t, a_t, \mathbf{W}_t) - (r + \gamma \max_a Q(s_{t+1}, a, \mathbf{W}_t)), \quad (4)$$

and can be minimized using any appropriate loss functions, e.g., the squared loss $\mathcal{L}(\delta) = 0.5\delta^2$.

Even though deep Q-learning can provide state-of-the-art solutions to several RL problems, it is not easy to directly adapt it for solving *continuous* control problems, while it does not directly optimize the policy of the agent. (Instead it indirectly learns the optimal policy by estimating the $Q$-values.) Such continuous control problems can be more efficiently solved using a different kind of RL algorithms, the policy gradients methods [24, 25], that directly learn the policy of the agent. A special category of these methods works by using two different estimators: the first one, called *actor*, is directly trained to estimate the optimal policy, while the second one, which is called *critic*, is used to estimate the *advantage* that a state/selected action provides and then used to appropriately train the actor. Again, deep neural networks can be used to implement the actor and critic models. It is worth noting that usually the critic model is trained similarly to the Q-learning approach by observing the rewards/punishment the agent obtains. Next, the actor is appropriately updated to increase the probability of selecting the actions that lead to maximizing the expected reward. A recently proposed deep RL method that can be used to train agents to perform continuous control, the Deep Deterministic Policy Gradient method [24], is employed for the experiments conducted in this paper. The interested reader is also referred to [2, 23, 39], for a more in-depth review of RL.

### 3.2 Proposed drone simulation environment

The Head Pose Image Database [13], abbreviated as "HPID" thereof, was used to develop the simulation environment. HPID contains 2790 face images of 15 subjects in various poses taken in a constrained environment. More specifically, for each person face images with various head tilts (vertical angle) and pans (horizontal angle) were taken. For the tilt, head photographs at $-90°$, $-60°$, $-30°$, $-15°$, $0°$, $15°$, $30°$, $60°$, and $90°$ were taken, while for the pan images that depict the head at $-90°$, $-75°$, $-60°$, $-45°$, $-30°$, $-15°$, $0°$, $15°$, $30°$, $45°$, $60°$, $75°$, and $90°$ were used. The full pan range exists only for the images with tilt angles between $-60°$ and $60°$. Therefore, we restrict the available tilt angles used in the simulator to the aforementioned range.

The developed environment for discrete control supports five different actions (assuming that the drone moves on the sphere):

1. *stay* do not perform any action (to be used when a clear frontal view has been obtained),
2. *left* move the drone left by $15°$ → pan decreases by $15°$,
3. *right* move the drone right by $15°$ → pan increases by $15°$,
4. *up* move the drone upward by $15°/30°$ (depending on the available annotations) → tilt decreases by $15°/30°$, and
5. *down* move the drone downward by $15°/30°$ (depending on the available annotations) → tilt increases by $15°/30°$.

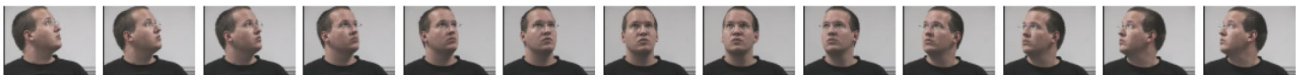
During these movements, we assume that the camera is appropriately controlled to keep the face centered, e.g., using a PID controller [34]. This is a quite realistic assumption, since deep face detection algorithms are able to successfully detect faces with high accuracy, while even simple PID-based control approaches work quite well for this task [34]. If an agent requests a control command that exceeds the limits of the simulator, e.g., an angle larger than $90°$, then the simulator remains at its last valid state. The developed simulator uses these images to simulate the movement of a drone in a part of a sphere defined by the center of the head of the subject. Some examples of face images of one person are shown in Fig. 1. This environment can be directly used to simulate the movement of a drone in discrete steps: $15°$ steps for the pan and in $15°/30°$ steps for the tilt.

Furthermore, this environment can be also used in *pseudo-continuous* mode by translating the continuous control commands into multiple discrete steps. In this work, we assume that a maximum of three consecutive discrete steps can be performed in one continuous control step. Therefore, the continuous control output of the agent is quantized into four regions: no action, one control step, two control steps (the same action is repeated twice), and three control steps (the same action is repeated three times). This allows to significantly reduce the number of controls that the agent has to handle, since instead of having an agent that can perform 49 (7 actions per control axis) different combinations of actions (when the tilt and pan are simultaneously controlled), only two continuous control outputs are needed (one for the tilt and one for the pan).

However, the approach described above cannot be used to perform fine-grained continuous control, since it is still limited by the available face poses. To overcome this limitation and provide a more realistic simulation environment, the face alignment and warping technique

**(a)** Face images at $-30°$ tilt (pan varies from $-90°$ to $90°$ in steps of $15°$)

**(b)** Face images at $0°$ tilt (pan varies from $-90°$ to $90°$ in steps of $15°$)



**(c)** Face images at $30°$ tilt (pan varies from $-90°$ to $90°$ in steps of $15°$)

**Fig. 1** Discrete simulation environment: the drone moves in a sphere (approximately) centered at the face of the subject. The control commands (left/right/up/down) are simulated using the appropriate images from HPID

proposed in [51] were used to obtain 3D generated images of each person for various pans/tilts. To this end, we directly used an open-source implementation of this technique, provided by the authors of [51], available at https://github.com/junyanz/FaceDemo. One frontal image from each person, which was randomly selected from the two frontal images provided by the HPID dataset, was used to generate the texture for the 3D model. In the developed continuous control environment, the pan ranges from $-45°$ to $45°$ in $1°$ steps (this can be assumed to be the control accuracy), while the tilt ranges from $-20°$ to $20°$ in $1°$ steps. One face at various pans, as provided by the developed simulation environment for continuous control, is depicted in Fig. 2. Note that only the cropped face is fed to the RL agent. This can be easily ensured in a real application using face detection methods [44, 47, 49] and then appropriately cropping the part of the image that contains the face. It is worth noting that an agent trained on the discrete/pseudo-continuous environment can be used to perform coarse-grained control and approximately center the face image, while the agent trained on the fine-grained environment can be then employed to fine-tune the drone position accordingly and make small corrections. All the developed environments (discrete, pseudo-continuous, and continuous) are *OpenAI Gym*-compatible [4] and provided at https://github.com/passalis/continuous_drone_frontal_rl.

### 3.3 Continuous drone control with deep reinforcement learning

The complete pipeline of the proposed deep RL technique for drone control is described in this subsection. The RL agent interacts with the developed environment and *observes* its state (acquired shot) as shown in Figs. 1 and 2 . To simplify the learning process, a face detector can employed to detect and crop the face image [44] (or face annotations can be used if they are available). After appropriately cropping the image, it was resized to $64 \times 64$ pixels. Therefore, the RL agent at the $t$-th time step observes a tensor $\mathbf{x}_t \in \mathbb{R}^{64\times64\times3}$ that corresponds to the cropped face image.

Defining a meaningful reward function is critical for the fast and stable convergence of RL algorithms. Even though RL can deal, to some extent, with sparse and time-delayed rewards, we experimentally found out that rewarding (or punishing) the agent after each action can significantly speed up the learning process. The reward of the agent is related to the current control error. The control error at the $t$-th step is defined as:

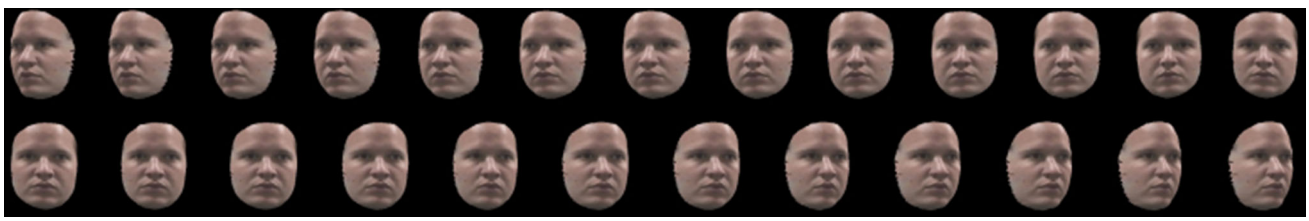


**Fig. 2** Continuous simulation environment: some of the 3D generated images for pan that ranges from $-30°$ to $-30°$ using steps of $3°$ (note that the simulator resolution is $1°$). Tilt was set to $0°$

$$e_t = \frac{1}{2}\left(\left(\frac{x_t}{x_t^m}\right)^2 + \left(\frac{x_p}{x_p^m}\right)^2\right), \tag{5}$$

where $x_t$ is the current tilt (in degrees) and $x_p$ is the current pan (in degrees), while $x_t^m$ and $x_p^m$ is the absolute maximum tilt/pan that can be observed. Note that it is trivial to modify this error function to acquire non-frontal shots around the desired tilt/pan range. Then, the reward function is defined as:

$$r_t^{(\text{raw})} = 1 - e_t. \tag{6}$$

However, this reward function provides a small positive rewards even for the wrong control movements. Therefore, a threshold $e_{\text{thres}}$ can be used for rewarding the agent, leading to a thresholded reward function:

$$r_t^{(\text{thres})} = \begin{cases} 0, & \text{if} \quad e_t > e_{\text{thres}} \\ 1 - e_t/e_{\text{thres}}, & \text{otherwise} \end{cases}, \tag{7}$$

If $e_{\text{thres}}$ is set to 1, then the agent is rewarded at every time step. Again, note that even though the reward is proportional to the control error $e_t$, this can slow down the learning process. In the conducted experiments, the agent was rewarded only when it was near to acquiring a correct frontal shot, i.e., $e_{\text{thres}}$ was set to 0.2. Then, to further boost the learning process, an extra small reward $a_r$ or punishment $a_p$ can be provided whenever the agent makes a correct or wrong movement:

$$r_t^{(\text{bonus})} = \begin{cases} -a_p, & \text{if} \quad e_t > e_{t-1} \\ a_r, & \text{if} \quad e_t < e_{t-1} \\ 0, & \text{otherwise} \end{cases}. \tag{8}$$

Using a higher penalty for wrong actions ensures the stability of the control process and discourages control oscillations. The optimal values for these parameters can be chosen by performing cross-validation. In the conducted experiments, it was established (by performing a simple line search) that the best control accuracy is obtained for $a_p = 0.4$ and $a_r = 0.2$. Finally, the reward function is defined as the sum between the raw reward $r_t^{(\text{thres})}$, which depends on the control error $e_t$, and the "bonus" reward $r_t^{(\text{bonus})}$, that encourages correct and stable control actions:

$$r_t^{(\text{shaped})} = r_t^{(\text{thes})} + r_t^{(\text{bonus})}. \tag{9}$$

Three different agents were trained in this work: (a) an agent capable of performing discrete control (baseline), (b) an agent capable of performing pseudo-continuous control, and (c) an agent capable of performing continuous control. The double Q-learning approach was employed for training the first agent (discrete control) [16], while the Deep Deterministic Policy Gradient (DDPG) method was used for training the other two agents [24]. The neural networks used for approximating the action-value function in Q-learning and the networks used for implementing the actor and critic in the DDPG method use the same architecture, which is shown in Table 1. Only the output layer of the networks is appropriately modified to accommodate the needs of each method, i.e., 5 output neurons are employed for the network used in the Q-learning method (each neuron corresponds to the $Q$-value of the corresponding action for the given state), 1 output neuron is used for the network that implements the critic, and 2 output neurons (equipped with a *tanh* activation function) are used for the actor network. (The output of each of the neurons is used to directly control the pan and tilt.) Batch normalization is used after each convolutional layer [18], while the *ReLU* activation function is used for all the convolutional and dense layers (except from the final one). Also, note that the critic receives the output of the actor in its penultimate fully connected layer and predicts the $Q$-value of the action selected by the actor for the given input state, while the actor and critic share the same layers up to the first layer after the last convolutional layer. The latter allows to share the knowledge encoded between the actor and critic, while speeding up the learning process.

*Learning hyper-parameters* The size of the experience replay buffer is set to $N_{\text{replay}} = 5000$ samples when the discrete/pseudo-continuous environment is used and to $N_{\text{replay}} = 10{,}000$ samples for the continuous environment. We used the smaller replay buffer size that allowed the models to converge smoothly. Using larger sizes is not expected to significantly affect the learning process. Batches of 128 samples were used for the optimization, while the target model was updated with a rate of 0.001 after each

**Table 1** Neural network architecture. The size of the final output layer depends on the RL method that is used

| Layer type | Activation function | Output shape |
|---|---|---|
| Input | – | $64 \times 64 \times 3$ |
| Convolutional ($5 \times 5$, stride 2) | – | $30 \times 30 \times 16$ |
| Batch normalization | ReLU | $30 \times 30 \times 16$ |
| Max pooling ($2 \times 2$) | – | $15 \times 15 \times 16$ |
| Convolutional ($3 \times 3$) | ReLU | $13 \times 13 \times 32$ |
| Batch normalization | – | $13 \times 13 \times 32$ |
| Max pooling ($2 \times 2$) | – | $6 \times 6 \times 32$ |
| Convolutional ($3 \times 3$) | ReLU | $4 \times 4 \times 64$ |
| Batch normalization | – | $4 \times 4 \times 64$ |
| Max pooling ($2 \times 2$) | – | $2 \times 2 \times 64$ |
| Dense | ReLU | 128 |
| Dense | ReLU | 64 |
| Dense | –/tanh/– | 1/2/5 |

iteration. The Huber loss function was employed for training the critic network, as well as the network used in Q-learning:

$$\mathcal{L}(\delta) = \begin{cases} 0.5\delta^2, & \text{if} \quad \delta < \delta_{\text{thres}} \\ |\delta| - 0.5\delta_{\text{thres}}, & \text{otherwise} \end{cases}, \tag{10}$$

where $\delta$ is the temporal difference error defined in Eq. (4). Huber loss is used instead of squared loss, since it is more robust to outliers, providing smoother gradients and stabilizing the learning process ($\delta_{\text{thres}}$ was set to 1). This is especially important when the total episode reward can accumulate to relatively large values, as in the conducted experiments. For updating the weights of the network, the RMSProp optimizer was used [42]. The learning rate was set to $\eta = 0.0001$ for the double Q-learning method, while the learning rate for the actor was set to $\eta_a = 0.00001$ and to $\eta_c = 0.01$ for the critic. Setting the learning rates to the appropriate values was of crucial importance in order to ensure the stable convergence of the training process. To explore the solution space, a linear exploration policy was used. For the Q-learning method, exploration starts with an initial rate of 1 and linearly decreases to 0.1 during the first 95,000 training steps. For the DDPG method, a Gaussian process with mean 0 was used. The standard deviation of the process was linearly decreased from 0.5 (0.2 for the continuous environment) to 0.01 during the initial 95,000 steps (45,000 steps for the continuous environment). The agents were trained for 100,000 training steps for the discrete/pseudo-continuous environments and for 50,000 training steps for the continuous environment. The number of steps for each episode was set to 50. For each episode, a random initial position was used for the drone. For training, head images from 10 persons of the HPID were used, while for the evaluation images from the rest 5 persons were used. The discount factor was set 0.95 for the discrete agents, but it was reduced to 0.5 for the continuous agents, since larger values led to stability issues during the training process. The keras-rl library was used to implement CDC [37].

## 4 Experimental evaluation

The experimental evaluation is provided in this section. For evaluating the method, 500 random episodes were used, where for each episode the agent was allowed to perform 20 control actions. The discrete control method is abbreviated as "D-RL," the pseudo-continuous drone control method as "P-CDC," and the continuous drone control method as "CDC". The learned agents were also compared to two other strategies: a) using a dummy agent that does not perform any control action (abbreviated as "Stay") and b) using a deep CNN to perform pose regression and then appropriately control the drone (abbreviated as "Pose regressor"). The Pose regressor network uses the same architecture as the CNN used for estimating the $Q$-values (Table 1), and it was trained to directly regress the tilt and pan of a face image using the same train/test setup, leading to a mean tilt error of 16.67° and a mean pan error of 13.71° (evaluation on the test set).

The evaluation results for the discrete and pseudo-continuous environments are reported in Table 2. Note even though these environments have different action spaces, they have identical states, since the same states are observed by the agents. Therefore, the results obtained with the D-RL and P-CDC are directly comparable, since the observations, rewards, and control limits are the same. Both the simple ("raw") and the appropriately shaped ("shaped") reward functions are compared. Several conclusions can be drawn from the results reported in Table 2. First, using RL to train the agents leads to significantly better results than the "Stay" baseline agent, which corresponds to a dummy agent that does not perform any control action. (The drone remains at its initial position.) The deep pose regressor also leads to higher error, demonstrating the importance of learning optimal controllers for the task at hand instead of relying on handcrafted methods to perform control. Furthermore, the proposed reward-shaping approach always significantly improves the control accuracy, regardless of the used RL method. The proposed pseudo-continuous control approach also leads to overall better results than the discrete control

**Table 2** Drone control evaluation for frontal view person shooting using the discrete and pseudo-continuous environments

| Method | Eval. type | Mean absolute tilt error (°) | Mean absolute pan error (°) |
|---|---|---|---|
| Stay | Train | 29.94 | 48.24 |
| Pose regressor | Train | 6.87 | 7.44 |
| D-RL (raw) | Train | 8.82 | 13.32 |
| D-RL (shaped) | Train | 1.47 | 2.58 |
| P-CDC (raw) | Train | 9.24 | 6.96 |
| P-CDC (shaped) | Train | **0.00** | **0.03** |
| Stay | Test | 29.37 | 48.93 |
| Pose regressor | Test | 14.07 | 12.18 |
| D-RL (raw) | Test | 25.26 | 19.08 |
| D-RL (hint) | Test | 9.09 | **3.36** |
| P-CDC (raw) | Test | 12.81 | 5.88 |
| P-CDC (hint) | Test | **6.12** | 4.11 |

Both the simple ("raw") and the appropriately shaped ("shaped") reward functions are compared. The absolute mean tilt and pan error (to obtain a perfectly frontal shot) are reported

Best results (lower is better) are in bold

**Fig. 3** Qualitative drone control evaluation: comparing the D-RL approach (first row) to the P-CDC approach (second row). The state after each of the first 8 control actions is depicted. The test set was used for the evaluation. D-RL achieves 30°/45° tilt/pan error after two control steps (best achieved in 7 control steps), while P-CDC achieves 15°/15° tilt/pan error after two control steps (best achieved in 4 control steps)

**Fig. 4** Qualitative drone control evaluation: comparing the D-RL approach (first row) to the P-CDC approach (second row). The state after each of the first 8 control actions is depicted. The test set was used for the evaluation. D-RL achieves 0°/75° tilt/pan error after two control steps (best achieved in 7 control steps), while P-CDC achieves 0°/0° tilt/pan error after two control steps (best achieved in 2 control steps)

**Fig. 5** Qualitative drone control evaluation: comparing the D-RL approach (first row) to the P-CDC approach (second row). The state after each of the first 8 control actions is depicted. The test set was used for the evaluation. D-RL achieves 30°/30° tilt/pan error after two control steps (best achieved in 5 control steps), while P-CDC achieves 15°/15° tilt/pan error after two control steps (best achieved in 3 control steps)

approach, reducing the mean control error from 6.23° to 5.12° (test evaluation). Finally, note that the agents are especially prone to overfitting. (The P-CDC method learns to almost perfectly control the drone during the training).

These improvements are also demonstrated in the qualitative evaluation shown in Figs. 3, 4, and 5. The first row corresponds to the D-RL method, while the second one to the P-CDC method. The P-CDC method manages to obtain a frontal view of the subjects significantly faster, since it is capable of adaptively and simultaneously controlling both axes (pan and tilt). For example, in Fig. 3 the P-CDC method obtains a frontal view of the subject in just

**Table 3** Drone control evaluation for frontal view person shooting using the continuous environment

| Method | Eval. type | Mean absolute tilt error (°) | Mean absolute pan error (°) |
|---|---|---|---|
| Stay | Train | 10.00 | 23.41 |
| CDC (raw) | Train | 2.62 | 6.86 |
| CDC (shaped) | Train | **0.18** | **0.72** |
| Stay | Test | 10.28 | 23.48 |
| CDC (raw) | Test | 3.25 | 5.75 |
| CDC (shaped) | Test | **0.51** | **2.00** |

Both the simple ("raw") and the appropriately shaped ("shaped") reward functions are compared. The absolute mean tilt and pan error (to obtain a perfectly frontal shot) are reported

Best results (lower is better) are in bold

two control steps, while the D-RL method needs 7 control steps and it is more prune to oscillations around the center position. Similar behavior can be also observed in the other two figures, confirming the superior behavior of CDC for pseudo-continuous control.

The results for the fully continuous environment are shown in Table 3. Again, the shaped reward function leads to significantly better control accuracy than training with the raw reward function, highlighting the importance of using correctly designed reward functions. Note that the agent learns to almost perfectly center the subjects using fine control commands, both during the training and during the testing. This behavior is also confirmed in the test control sequences depicted in Fig. 6. The agent correctly controls the drone to obtain a clear frontal view, while usually small or no oscillations around the center are observed.

*Learning stability*

The rewards and mean $Q$-values during the training process are plotted in Fig. 7. Both the mean $Q$-value and reward steadily increase during the training process. However, note that the rewards fluctuate during the training process. The stochasticity of the training process greatly

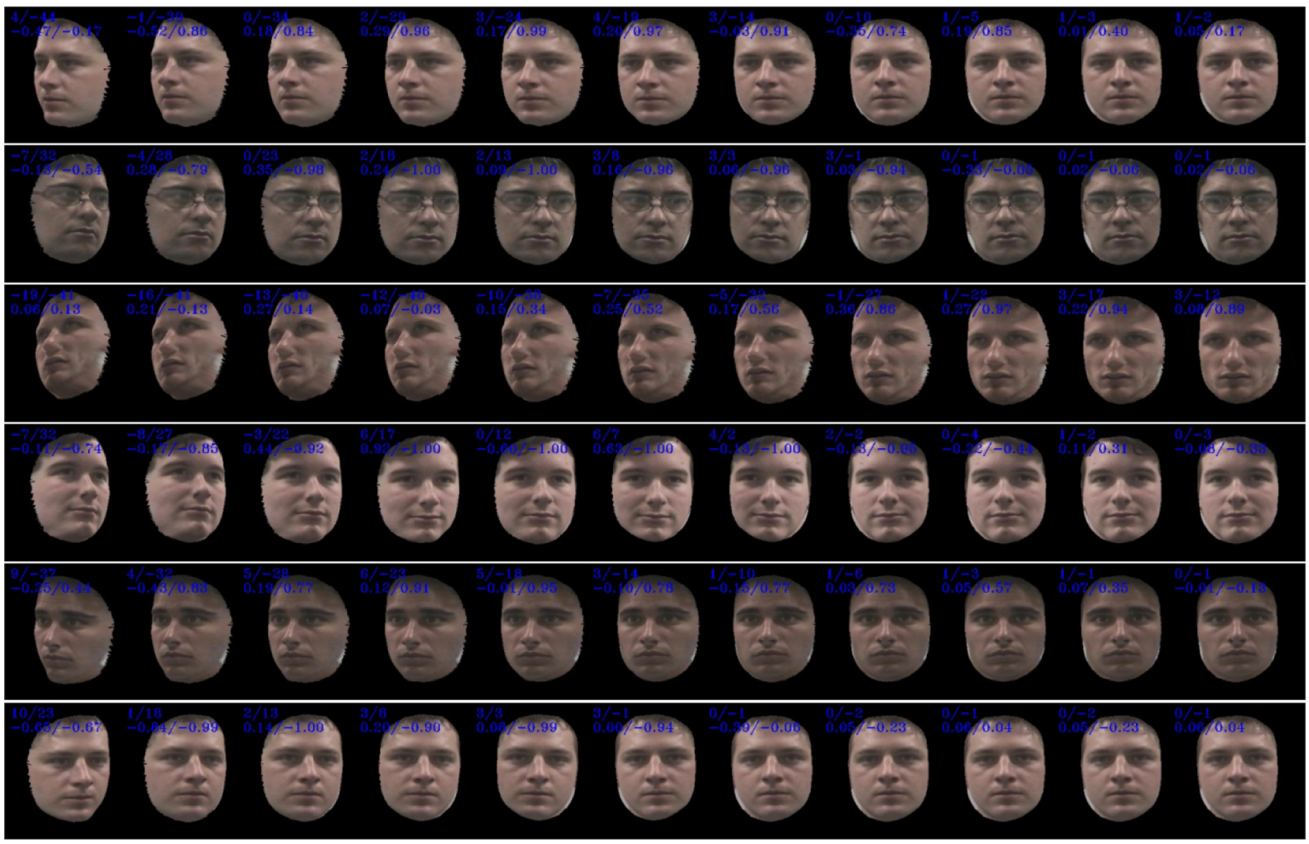


**Fig. 6** Qualitative drone control evaluation using the continuous environment for six different test settings. The state after each of the first 11 control actions is depicted

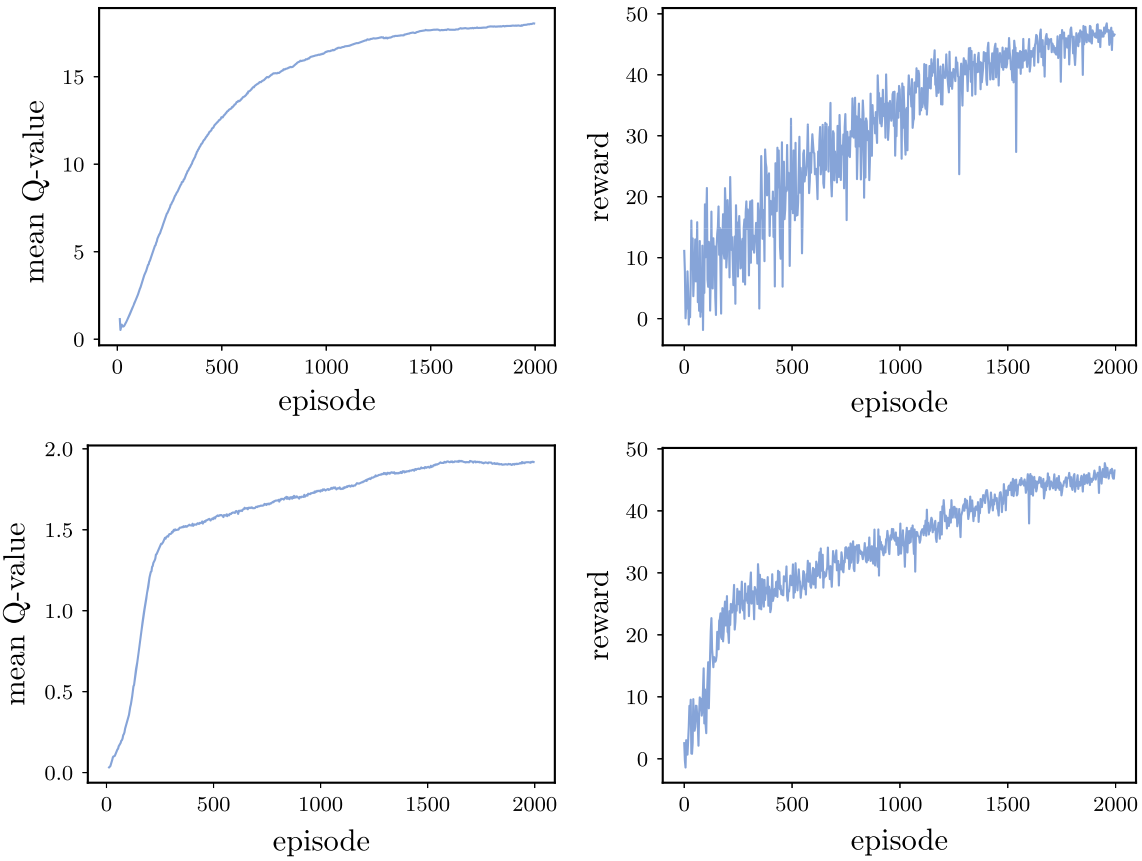



**Fig. 7** Convergence plots depicting the mean $Q$-value and the reward per training episode for the discrete control agent. The convergence of the D-RL agent using the shaped reward (first row) and the P-CDC agent using the shaped reward (second row) is depicted

contributes to this behavior, since the environment is randomly initialized before starting each training episode. This behavior can be also attributed to the quite unstable nature of deep RL algorithms, as also discussed in [24, 26], which often require several heuristics, such as using a replay buffer and target network updates, in order to ensure the smooth convergence of the training process. As a result, this observation led us to fully evaluate the learned agent every few training iterations (instead of just relying on the reward obtained on one specific episode). The evaluation results are shown in Fig. 8. Note that the behavior of the agent can vary significantly during the training process (especially when the DDPG method is used), even though a) the convergence seems to be smooth

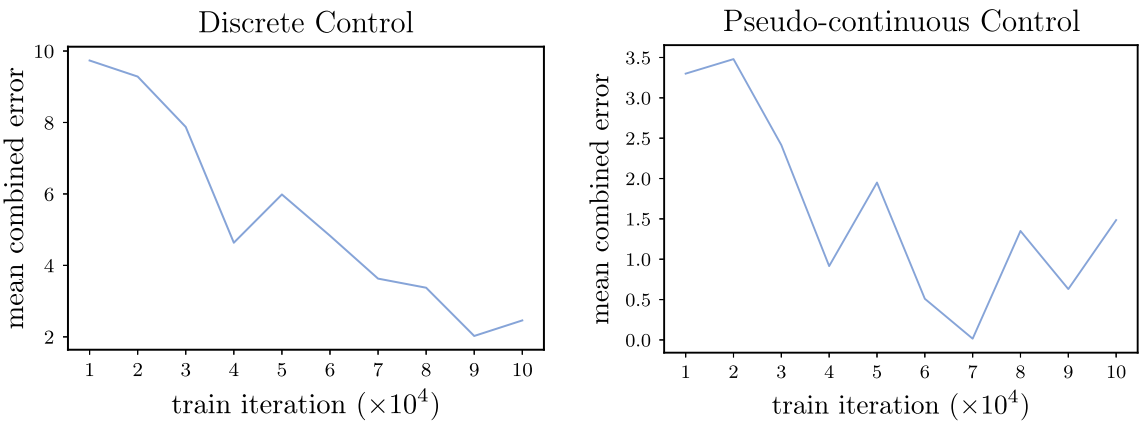



**Fig. 8** Validation error during the training for the D-RL (shaped reward) and P-CDC (shaped reward)

and the evaluation was performed using (multiple episodes of) the training set (on which the agent was optimized). Therefore, to avoid using a sub-optimal model, the agent was evaluated every 5000 training iterations and the best model (measured using the training set) was chosen as the final model. This process ensured that the best model that was found during the training process will be used. Indeed, it was experimentally confirmed that this process can increase the accuracy both for training and for test evaluation, even though the best model was selected using the training set.

## 5 Conclusions

A method for developing deep RL agents for continuous fine-grained drone control to acquire high-quality frontal view person shots was presented in this paper. First, a realistic simulation environment was developed using the HPID. This environment was extended using 3D models, along with a face alignment and warping technique, to allow for simulating the effects of continuous control commands, overcoming the restrictions arising from the limited number of facial poses contained in this dataset. An appropriate reward-shaping approach was also proposed to improve the stability of the employed continuous RL method. Apart from performing continuous control, it was demonstrated that CDC can be also effectively combined with simulation environments that support only discrete control commands, improving the control accuracy, even in this case. Finally, the proposed approach was compared both to an RL agent that performs discrete control, and to a traditional controller that directly uses the output of a deep model that performs pose estimation. It was experimentally demonstrated that the proposed approach improves the control accuracy over these methods, highlighting the importance of learning optimal controllers instead of relying on handcrafted control techniques.

Several interesting future research directions exist. First, the impressive performance of the proposed technique paves the way for several other deep RL-based control methods for autonomous drone-based cinematography. However, it is worth noting that we are currently still far away from being able to directly deploy the developed methods in real-world applications. In fact, even though in our work we employed datasets with real images, instead of computer-generated graphics, it is likely that much larger datasets should be created in order to ensure the successful deployment of these methods in real applications. Furthermore, CDC can be applied to more dynamic environments, e.g., moving targets can be used, and more sophisticated deep architectures, e.g., recurrent networks [6], can be used to accurately predict the movement of the target and control the drone. Furthermore, the use of end-to-end trainable systems, that simultaneously perform camera and drone control, can be developed and evaluated. Finally, the development of transfer learning techniques for RL [31, 41], that can combine the use of real datasets and simulators, which use computer-generated graphics, will allow for training RL techniques under a wider range of scenarios, while ensuring that they can be directly deployed in real-world applications.

**Acknowledgements** This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731667 (MULTIDRONE). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

### Compliance with ethical standards

**Conflict of interest** The authors declare that they have no conflicts of interest.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

## References

1. Ang KH, Chong G, Li Y (2005) PID control system analysis, design, and technology. IEEE Trans Control Syst Technol 13(4):559–576
2. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) Deep reinforcement learning: a brief survey. IEEE Signal Process Mag 34(6):26–38
3. Åström KJ, Hägglund T, Astrom KJ (2006) Advanced PID control, vol 461. ISA-The Instrumentation, Systems, and Automation Society, Research Triangle Park, NC
4. Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) Openai gym. Technical Report. arXiv:1606.01540
5. Bryson AE (1975) Applied optimal control: optimization, estimation and control. CRC Press, Boca Raton
6. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555
7. Claesson A, Fredman D, Svensson L, Ringh M, Hollenberg J, Nordberg P, Rosenqvist M, Djarv T, Österberg S, Lennartsson J et al (2016) Unmanned aerial vehicles (drones) in out-of-hospital-cardiac-arrest. Scand J Trauma Resusc Emerg Med 24(1):124
8. Duan Y, Chen X, Houthooft R, Schulman J, Abbeel P (2016) Benchmarking deep reinforcement learning for continuous control. In: Proceedings of the international conference on machine learning, pp 1329–1338
9. Finn C, Levine S (2017) Deep visual foresight for planning robot motion. In: Proceedings of the IEEE international conference on robotics and automation, pp 2786–2793
10. Finn C, Yu T, Fu J, Abbeel P, Levine S (2016) Generalizing skills with semi-supervised reinforcement learning. arXiv preprint arXiv:1612.00429
11. Galvane Q, Fleureau J, Tariolle FL, Guillotel P (2017) Automated cinematography with unmanned aerial vehicles. arXiv preprint arXiv:1712.04353

12. Garcia-Garcia A, Orts-Escolano S, Oprea S, Garcia-Rodriguez J, Azorin-Lopez J, Saval-Calvo M, Cazorla M (2017) Multi-sensor 3D object dataset for object recognition with full pose estimation. Neural Comput Appl 28(5):941–952
13. Gourier N, Hall D, Crowley JL (2004) Estimating face orientation from robust detection of salient facial features. In: ICPR international workshop on visual observation of deictic gestures. Citeseer
14. Gu S, Holly E, Lillicrap T, Levine S (2017) Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: Proceedings of the IEEE international conference on robotics and automation, pp 3389–3396
15. Gynnild A (2014) The robot eye witness: extending visual journalism through drone surveillance. Digit J 2(3):334–343
16. Hasselt HV, Guez A, Silver D (2016) Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI conference on artificial intelligence, AAAI'16, pp 2094–2100. AAAI Press. http://dl.acm.org/citation.cfm?id=3016100.3016191
17. Haykin S, Network N (2004) A comprehensive foundation. Neural Netw 2(2004):41
18. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the international conference on machine learning, pp 448–456
19. Kazemi V, Josephine S (2014) One millisecond face alignment with an ensemble of regression trees. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1867–1874
20. Kirk DE (2012) Optimal control theory: an introduction. Courier Corporation, North Chelmsford
21. Krüll W, Tobera R, Willms I, Essen H, von Wahl N (2012) Early forest fire detection and verification using optical smoke, gas and microwave sensors. Procedia Eng 45:584–594
22. Levine S, Pastor P, Krizhevsky A, Quillen D (2016) Learning hand–eye coordination for robotic grasping with large-scale data collection. In: Proceedings of the international symposium on experimental robotics, pp 173–184
23. Li Y (2017) Deep reinforcement learning: an overview. arXiv preprint arXiv:1701.07274
24. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971
25. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. In: Proceedings of the international conference on machine learning, pp 1928–1937
26. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529
27. Nägeli T, Meier L, Domahidi A, Alonso-Mora J, Hilliges O (2017) Real-time planning for automated multi-view drone cinematography. ACM Trans Graph 36(4):132
28. Ng AY, Harada, Russell S (1999) Policy invariance under reward transformations: theory and application to reward shaping. In: Proceedings of the international conference on machine learning
29. Nousi P, Patsiouras E, Tefas A, Pitas I (2018) Convolutional neural networks for visual information analysis with limited computing resources. In: Proceedings of the IEEE international conference on image processing, pp 321–325
30. Ohayon S, Rivlin E (2006) Robust 3D head tracking using camera pose estimation. Proc Int Confer Pattern Recognit 1:1063–1066
31. Pan SJ, Yang Q et al (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
32. Passalis N, Tefas A (2017) Concept detection and face pose estimation using lightweight convolutional neural networks for steering drone video shooting. In: Proceedings of the European signal processing conference, pp 71–75
33. Passalis N, Tefas A (2018) Deep reinforcement learning for frontal view person shooting using drones. In: Proceedings of the IEEE conference on evolving and adaptive intelligent systems, pp 1–8
34. Passalis N, Tefas A, Pitas I (2018) Efficient camera control using 2D visual information for unmanned aerial vehicle-based cinematography. In: Proceedings of the international symposium on circuits and systems (to appear)
35. Passalis N, Tefas A (2018) Training lightweight deep convolutional neural networks using bag-of-features pooling. IEEE Trans Neural Netw Learn Syst 30(6):1705–1715
36. Passalis N, Tefas A (2019) Deep reinforcement learning for controlling frontal person close-up shooting. Neurocomputing 335:37–47
37. Plappert M (2016) Keras-rl. https://github.com/matthiasplappert/keras-rl. Accessed 6 July 2019
38. Ranjan R, Patel VM, Chellappa R (2017) Hyperface: a deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. IEEE Trans Pattern Anal Mach Intell 41:121–135
39. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction, vol 1. MIT Press, Cambridge
40. Tang L, Liu YJ, Tong S (2014) Adaptive neural control using reinforcement learning for a class of robot manipulator. Neural Comput Appl 25(1):135–141
41. Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains: a survey. J Mach Learn Res 10:1633–1685
42. Tieleman T, Hinton G (2012) Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA Neural Netw Mach Learn 4(2):26–31
43. Toshev A, Szegedy C (2014) DeepPose: human pose estimation via deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1653–1660
44. Triantafyllidou D, Nousi P, Tefas A (2017) Fast deep convolutional face detection in the wild exploiting hard sample mining. Big Data Res 22:65
45. Tzelepi M, Tefas A (2017) Human crowd detection for drone flight safety using convolutional neural networks. In: Proceedings of the European signal processing conference, pp 743–747
46. Van Hasselt H, Guez A, Silver D (2016) Deep reinforcement learning with double Q-learning. Proc AAAI Confer Artif Intell 16:2094–2100
47. Vong CM, Tai KI, Pun CM, Wong PK (2015) Fast and accurate face detection by sparse Bayesian extreme learning machine. Neural Comput Appl 26(5):1149–1156
48. Wooldridge M (2009) An introduction to multiagent systems. Wiley, New York
49. Yang S, Luo P, Loy CC, Tang X (2016) Wider face: a face detection benchmark. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5525–5533
50. Zhang F, Leitner J, Milford M, Upcroft B, Corke P (2015) Towards vision-based deep reinforcement learning for robotic motion control. arXiv preprint arXiv:1511.03791
51. Zhu JY, Agarwala A, Efros AA, Shechtman E, Wang J (2014) Mirror mirror: crowdsourcing better portraits. ACM Trans Graph (SIGGRAPH Asia 2014) 33(6):11