# Deep Reinforcement Learning with semi-expert distillation for autonomous UAV cinematography

1st Andreas Sochopoulos
Department of Electical and Computer Engineering
Aristotle University of Thessaloniki
Thessaloniki, Greece
aasochop@ece.auth.gr

2nd Ioannis Mademlis
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
imademlis@csd.auth.gr

3rd Evangelos Charalampakis
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
charevan@csd.auth.gr

4th Sotirios Papadopoulos
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
papasot@csd.auth.gr

5th Ioannis Pitas
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
pitas@csd.auth.gr

***Abstract*—Unmanned Aerial Vehicles (UAVs, or drones) have revolutionized modern media production. Being rapidly deployable "flying cameras", they can easily capture aesthetically pleasing aerial footage of static or moving filming targets/subjects. Current approaches rely either on manual UAV/gimbal control by human experts, or on a combination of complex computer vision algorithms and hardware configurations for automating the flight+filming process. This paper explores an efficient Deep Reinforcement Learning (DRL) alternative, which implicitly merges the target detection and path planning steps into a single algorithm. To achieve this, a baseline DRL approach is augmented with a novel policy distillation component, which transfers knowledge from a suitable, semi-expert Model Predictive Control (MPC) controller into the DRL agent. Thus, the latter is able to autonomously execute a specific UAV cinematography task with purely visual input. Unlike the MPC controller, the proposed DRL agent does not need to know the 3D world position of the filming target during inference. Experiments conducted in a photorealistic simulator showcase superior performance and training speed compared to the baseline agent, while surpassing the MPC controller in terms of visual occlusion avoidance.**

***Index Terms*—autonomous drones, UAV cinematography, Deep Reinforcement Learning, policy distillation, Model Predictive Control, Deep Neural Networks**

## I. Introduction

The rapid popularization of commercial, battery-powered, camera-equipped Unmanned Aerial Vehicles (UAVs, or "drones") during the past decade, has deeply affected media production. UAVs have proven to be an affordable means for swiftly acquiring impressive aerial footage in diverse scenarios. They offer fast shot setup, the ability to hover above a point of interest, access to narrow spaces and novel aerial shot types not easily achievable otherwise, at a minimal cost [1], [2].

According to recent UAV shot type taxonomy proposals [3], [4], [5], [6], [7], [8], UAV shots consist in valid combinations of Camera Motion Types (CMTs) and Framing Shot Types (FSTs). The most interesting among them involve a still or moving target/subject. Professional UAV filming requires specialised personnel for flight and filming control, i.e., separate drone and camera/gimbal operators. That renders the possibility of fully autonomous drones highly attractive, since it would reduce the need for human operators and the overhead that comes with executing demanding CMTs, such as "Orbit".

A popular class of methods for achieving autonomy in complex unknown and stochastic environments is Deep Reinforcement Learning (DRL), which relies on Deep Neural Networks (DNNs). Classical control algorithms such as Model Predictive Control (MPC) become highly complex in large-scale problems and require sensor information that is typically not available, or very expensive to acquire. DRL agents on the other hand can learn to navigate in stochastic environments with limited sensor information [9], [10]. DRL and classic control methods are traditionally seen as alternatives, while *none has been exploited up to now for autonomous CMT execution in UAV cinematography*.

The goal of this paper is to try to leverage the benefits of both worlds, with as little overhead as possible. To achieve this, a novel DRL agent training algorithm is proposed: it exploits knowledge conveyed by a manually designed, task-specific MPC controller which requires 3D target position information in order to function. The training process is complemented by collision avoidance and occlusion avoidance objectives. After the DRL agent has been trained, its task is to constantly and autonomously adjust the UAV's trajectory so that a desired CMT is executed, without relying on detailed 3D maps or on knowledge of the 3D position of the target being filmed; only RGB video input from the UAV-mounted camera is utilized at each time step. The proposed vision-based MPC-distilled DRL agent exhibits superior performance in comparison to the baseline one, without relying on complex and time consuming trajectory optimisation, pose estimation,

This work has received funding from the European Union's Horizon 2020 programme under grant agreement No 951911 (AI4Media).

3D map reconstruction or state estimation methods.

Thus, the novel contributions of this paper are the following three items:

- An MPC controller designed for autonomous UAV CMT execution (i.e., for the "Orbit" CMT [3]). No such algorithm has been previously proposed.
- Exploiting policy distillation during training, by combining the semi-expert MPC controller with the Deep Deterministic Policy Gradient (DDPG) DRL algorithm, in contrast to the common trend of distilling policies from expert teachers.
- Introducing a task-specific DRL training reward for visual occlusion avoidance, so that the collected footage clearly depicts the target being filmed.

These novelties, taken together, allow the proposed MPC-distilled DDPG agent to tackle the complex problem of a UAV autonomously orbiting around a target being filmed in an obstacle-cluttered environment, using purely visual input data.

## II. Related Work

Research on UAV control using vision-based DRL typically utilizes video frames from a UAV-mounted front-facing camera, to enrich the observations of the environment with visual information. In [11], a hierarchy of independent DQNs and non-neural controllers is used to solve the UAV autonomous landing problem, by breaking it down to three simpler tasks. In [10], a variation of the RDPG [9] algorithm for UAV navigation missions achieves faster convergence than the baseline, by using the actor-critic architecture to break up the correlation between sequential observations. In [12], a DD-DQN [13] agent generates label maps for training a segmentation model, so as to achieve UAV path planning with simultaneous obstacle avoidance.

There have also been attempts to utilize MPC in DRL using either imitation learning [14], Guided Policy Search [15], or the Hamiltonian of the optimal control problem [16]. In contrast to these approaches, this paper combines a semi-expert MPC controller, which can provide only trajectory tracking and approximate obstacle avoidance, with Deep Deterministic Policy Gradient using policy distillation [17], [18].

Unlike DRL for generic robotic and UAV control, DRL specifically for UAV cinematography tasks has not been explored in depth. A few relevant algorithms have been presented for frontal view person shooting [19], for object tracking using multiple drones, or for combining trajectory optimisation and DRL in order to automate artistic choices [20].

## III. Technical Background

The proposed method includes a semi-expert MPC controller, able to autonomously orbit around a target and avoid obstacles, and a CNN-based actor trained by DDPG with policy distillation.

**Model Predictive Control**. MPC is a discrete time control technique that provides a solution to the finite horizon, constrained optimal control problem:

$$\begin{aligned} \min_{u(\cdot)} \quad & \Phi(\boldsymbol{x}(t_f)) + \int_0^{t_f} l(\boldsymbol{x},\boldsymbol{u},t)\ dt \\ \textit{subject to} \quad & \boldsymbol{x}' = f(\boldsymbol{x},\boldsymbol{u},t),\ \boldsymbol{x}(0)=\boldsymbol{x}_0, \\ & g(\boldsymbol{x},\boldsymbol{u},t) = 0, \\ & h(\boldsymbol{x},\boldsymbol{u},t) \geq 0, \end{aligned} \tag{1}$$

where $t_f$ is the time horizon, $\boldsymbol{x}$ is the system state, $\boldsymbol{u}$ are the inputs to the system, $\boldsymbol{x}_0$ a given initial state, $\Phi(\cdot)$ the final cost and $l(\cdot)$ the intermediate cost function. $f(\cdot), g(\cdot)$ and $h(\cdot)$ are time-dependent vector fields defining the system dynamics, the equality constraints, and the inequality constraints, respectively [15]. The problem's associated optimal value function $V$ (cost-to-go) is defined as:

$$V(t,\boldsymbol{x}) = \min_{\boldsymbol{u}} \quad \Phi(\boldsymbol{x}(t_f)) + \int_0^{t_f} l(\boldsymbol{x},\boldsymbol{u},t)\ dt. \tag{2}$$

**Deep Deterministic Policy Gradient**. DDPG is a model-free Deep Reinforcement Learning (DRL) algorithm for continuous action control, based on the actor-critic paradigm. It is a significant improvement in comparison to Deterministic Policy Gradient due to three novelties: the utilisation of Deep Neural Networks (DNNs) for the functional approximation of the actor and the critic, the use of target networks and the use of experience replay [21].

Actor-critic methods model the actor with a DNN and find the synaptic weights that encode the optimal policy $\pi^*$. This is achieved by updating its weights in the direction of the gradients of the expected future reward, using error back-propagation and a form of gradient descent. Because that gradient needs an unbiased estimator of the Q-function in order to guide the weights of the actor towards the optimal policy, the critic network is first updated towards the direction of the gradient of the temporal difference at each training iteration.

Target networks are networks of identical structure as the actor and critic, but their weights are updated in such a way to slowly follow the weights of the actual actor and critic. This way, stability in the learning process of the actor and the critic is improved. The concepts of target networks and the experience replay have been introduced in Deep Q-Learning [22] and have become common in DRL algorithms.

Actor-critic methods can either be on-policy or off-policy, meaning that the action taken at every step for exploring the environment is sampled from the actor network or from another policy, respectively. DDPG is off-policy, in the sense that the action used at every step is the sum of the actor policy at the current state of the environment plus an appropriate exploratory noise. In this paper, the Ornstein-Uhlenbeck process is used as the additive exploration noise.

Below, the critic $Q_{\boldsymbol{\theta}^Q}(\boldsymbol{o},\boldsymbol{a})$ and the actor $\mu_{\boldsymbol{\theta}^\mu}(\boldsymbol{o})$ are parameterized by $\boldsymbol{\theta}^Q$ and $\boldsymbol{\theta}^\mu$. Essentially, $\boldsymbol{\theta}^\mu$ encodes the policy being learned by the DDPG agent during its training. $\boldsymbol{s}_t$, $\boldsymbol{o}_t$ and $\boldsymbol{a}_t$ are the state description, observation and action of the $t$-th timestep, respectively.

**Policy Distillation**. Policy distillation is the transfer of knowledge from a teacher actor $T$ to a student actor $S$. This is traditionally achieved by generating a dataset $D_T = \{\boldsymbol{s}_i, \boldsymbol{q}_i\}_0^N$ [17] using the teacher and, subsequently, exploiting it as ground-truth to train the student Q-network through regression. One core novelty of the proposed method is to exploit a suitable MPC controller as teacher under a policy distillation framework, in order to augment the training process of a neural DRL agent. Thus, policy distillation is modified in this paper in order to account for the semi-expert nature of the employed teacher. The aim is to transfer knowledge to the student, while the latter becomes more robust and learns according to rewards that cannot be incorporated in the MPC setting. The processes of knowledge transfer and DRL training have to be simultaneous, leading to *on-line policy distillation*.

## IV. Proposed Method

The proposed method consists of a properly adapted form of a DDPG agent, trained using a novel, task-specific reward function for autonomous UAV cinematography, and the novel on-line policy distillation term from a MPC teacher. Autonomous execution of the "Orbit" CMT has been selected below as the desired cinematography task, due to its challenging nature (it is difficult to execute manually) and its high aesthetic appeal. However, the proposed method may alternatively be employed for executing *any* UAV cinematography CMT, provided that a proper MPC controller can be designed.

### A. MPC for Executing Orbit CMTs

Holding the assumption that the UAV is velocity-controlled based only on its kinematics, the motion equations are:

$$\begin{aligned} \dot{x} &= a_x \\ \dot{y} &= a_y \\ \dot{z} &= a_z, \end{aligned} \tag{3}$$

where $a_i$ is the velocity set-point along direction $i$ and the vector $\boldsymbol{a} = [a_x, a_y, a_z]^T$ represents the action. We define $\boldsymbol{q} = [x, y, z]^T$ as the actual 3D position of the UAV in the environment and $\boldsymbol{q}_d \in \mathbb{R}^3$ as the orbital position where, ideally, the UAV should currently be in if executing a proper "Orbit" CMT.

The MPC controller's task is to track the orbital trajectory in 3D space, which is computed according to the "Orbit" equation in [3]. This computation requires the 3D position of both the drone and the filmed target/subject to be known. Furthermore, obstacle avoidance is implemented by surrounding all obstacles by spheres and then using the distance of the UAV from the sphere as a constraint. Obviously, this also requires the 3D positions in space of all obstacles to be known. The inequality constraints that arise are the following ones:

$$d_i = ||\boldsymbol{q} - \boldsymbol{c}_i||^2 - r_i^2, \; for \; i = 1, 2, .., M, \tag{4}$$

where $M$ is the number of spheres and $c_i$/$r_i$ are the center/radius of the $i$-th sphere, respectively. Thus, the $M+1$ constraints are:

$$\begin{aligned} d_i &\geq 0, \; for \; i = 1, 2, ..., M \\ z &> 0. \end{aligned} \tag{5}$$

Function $l(\cdot)$ is constructed as the sum of the distance between the UAV and the ideal orbital position at that time instant, the square of the action vector and a term for creating smooth action signals:

$$l_t = ||\boldsymbol{q}_t - \boldsymbol{q}_{d_t}||^2 + ||\boldsymbol{a}_t||^2 + \boldsymbol{a}_{t-1}^T \boldsymbol{a}_t. \tag{6}$$

High input sampling frequencies result in more accurate control policies, but require faster inference, which is not always feasible on embedded/on-drone computational hardware. Thus, a sampling period of $T = 0.4$ seconds was chosen.

### B. MPC-distilled DDPG

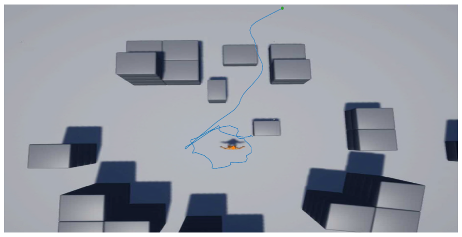
After the task-specific MPC controller has been designed, the proposed purely-vision based DRL agent can be trained using on-line policy distillation.

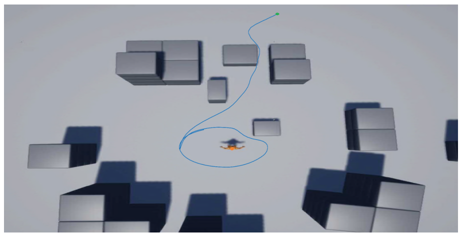
The main idea is to leverage the useful properties MPC carries in a DRL framework. Since the task of autonomous "Orbit" execution from visual input only is highly complex, solving it with a baseline, pure DRL method would result in a prohibitively high required training time and in a DNN size too large for embedded computers. Moreover, accurate reward shaping is crucial, but rather difficult in large-scale problems such as this one, since any defects in its design would likely lead to even higher training times, instabilities and sub-optimal solutions.

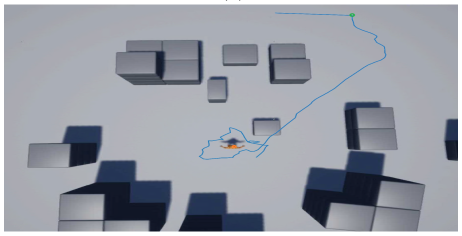
All these issues are significantly ameliorated by introducing a loss term that penalizes distance between the student's action and the MPC-teacher's action, at each step of the training process. This serves as a way to exploit the known UAV kinematics model while training the DRL agent, in order to improve its performance. The proposed distillation loss term is incorporated into the gradient of the cost $J$ [21], in order to guide the policy being learnt near to the MPC controller's policy. Notably, the MPC controller is not a perfect expert (as is typically the case in imitation learning) but simply an automated algorithm with access to additional information (i.e., the target's 3D position at each time step). The gradient that is used to update the weights of the actor DNN is:

$$\begin{aligned} &\nabla_{\boldsymbol{\theta}^\mu} J = \\ &\boldsymbol{E}_{\boldsymbol{s}_t \sim \rho^\beta} \left[ \nabla_{\boldsymbol{\theta}^\mu} Q(\boldsymbol{o}, \mu(\boldsymbol{o}) | \boldsymbol{\theta}^Q)|_{\boldsymbol{o}=\boldsymbol{o}_t} \nabla_{\boldsymbol{\theta}^\mu} \mu(\boldsymbol{o}|\boldsymbol{\theta}^\mu)|_{\boldsymbol{o}=\boldsymbol{o}_t} \right. \\ &\quad \left. + \; \nabla_{\boldsymbol{\theta}^\mu} l(\mu(\boldsymbol{o}|\boldsymbol{\theta}^\mu), \boldsymbol{a}_{MPC}(s))|_{\boldsymbol{s}=\boldsymbol{s}_t, \boldsymbol{o}=\boldsymbol{o}_t} \right] \end{aligned} \tag{7}$$

In the problem tackled, the state $\boldsymbol{o}_t \in \mathbb{R}^{84x84x3}$ is the RGB image that the drone camera captures in time $t$, $\boldsymbol{s}_t$ the vector containing the UAV, target and obstacle positions, $\boldsymbol{a}_t \in \mathbb{R}^3$ the action vector as defined in (3) and $\boldsymbol{a}_{MPC} \in \mathbb{R}^3$ the corresponding MPC action given UAV position $\boldsymbol{q}_t$. It must be noted that the proposed method only controls the UAV position, since complementary, cinematography-oriented, purely vision-based algorithms already exist for camera gimbal control [23] [19].


(a)


(b)


(c)

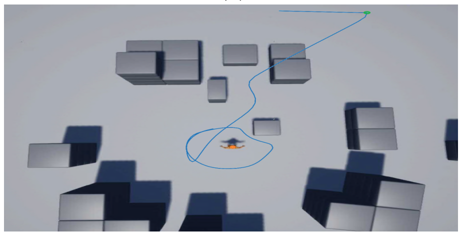

(d)

Fig. 1: Top-down view of the $x-y$ plane trajectories outputted by the proposed MPC-distilled DDPG agent ((a) and (c)) and the semi-expert MPC controller ((b) and (d)). In (a),(b) and (c),(d) the starting UAV position is $[0,0]$ and $[0,20]$, respectively.

**Algorithm 1** MPC-distilled DDPG

```
Initialize critic $Q_{\boldsymbol{\theta}^Q}(\boldsymbol{o},\boldsymbol{a})$ and actor $\mu_{\boldsymbol{\theta}^\mu}(\boldsymbol{o})$ with parameters $\boldsymbol{\theta}^Q$ and $\boldsymbol{\theta}^\mu$. Initialize target critic $Q_{\boldsymbol{\theta}^{Q'}}(\boldsymbol{o},\boldsymbol{a})$ and target actor $\mu_{\boldsymbol{\theta}^{\mu'}}(\boldsymbol{o})$ with weights $\boldsymbol{\theta}^{Q'} \leftarrow \boldsymbol{\theta}^Q$ and $\boldsymbol{\theta}^{\mu'} \leftarrow \boldsymbol{\theta}^\mu$.
Initialize Replay Buffer R with an extra slot for MPC actions.
for episodes=1,M do
    Initialize a stochastic process $N$ for action exploration.
    Receive an initial observation $\boldsymbol{o}_0$ and an initial state $\boldsymbol{s}_0$.
    for t=1,T do
        Obtain action $\boldsymbol{a}_t = \mu_{\boldsymbol{\theta}^\mu}(\boldsymbol{o}_t) + \boldsymbol{N}_t$
        Execute action $\boldsymbol{a}_t$, obtain reward $r_t$, observation $\boldsymbol{o}_t$ and state $\boldsymbol{s}_t$
        Calculate MPC action using the full state $\boldsymbol{s}_t$
        Store transition $(\boldsymbol{a}_t, \boldsymbol{a}_t^{MPC}, \boldsymbol{o}_t, r_t)$ into R
        Sample a minibatch of L transitions $\{(\boldsymbol{a}_i^{MPC}, \boldsymbol{a}_i, \boldsymbol{o}_i, r_i)\}_{i=1}^{L}$
        Set $y_i = r_i + \gamma Q_{\boldsymbol{\theta}^{Q'}}(\boldsymbol{o}_i, \mu_{\boldsymbol{\theta}^{\mu'}})$
        Compute critic update
            $\Delta\boldsymbol{\theta}^Q = \frac{1}{L}\sum_i (y_i - Q_{\boldsymbol{\theta}^Q}(\boldsymbol{o}_i,\boldsymbol{a}_i))\frac{\partial Q_{\boldsymbol{\theta}^Q}(\boldsymbol{o}_i,\boldsymbol{a}_i)}{\partial \boldsymbol{\theta}^Q}$
        Compute actor update
            $\Delta\boldsymbol{\theta}^\mu = \frac{1}{L}\sum_i \frac{\partial Q_{\boldsymbol{\theta}^Q}(\boldsymbol{o}_i,\mu_{\boldsymbol{\theta}^\mu}(\boldsymbol{o}_i))}{\partial \boldsymbol{a}}\frac{\partial \mu_{\boldsymbol{\theta}^\mu}(\boldsymbol{o}_i)}{\partial \boldsymbol{\theta}^\mu} + \nabla_{\boldsymbol{\theta}^\mu} l(\mu_{\boldsymbol{\theta}^\mu}(\boldsymbol{o}_i), \boldsymbol{a}_{MPC}(\boldsymbol{o}_i))$
        Update actor and critic using Adam
        Update target network of critic
            $\boldsymbol{\theta}^{Q'} \leftarrow \epsilon\boldsymbol{\theta}^Q + (1-\epsilon)\boldsymbol{\theta}^{Q'}$
        Update target network of actor
            $\boldsymbol{\theta}^{\mu'} \leftarrow \epsilon\boldsymbol{\theta}^\mu + (1-\epsilon)\boldsymbol{\theta}^{\mu'}$
    end for
end for
```

## *C. Reward shaping for the DDPG agent*

In order to train a suitable DDPG agent, 4 individual rewards were linearly combined to form the final complete reward function. This is computed at each timestep by the training environment and provided to the DDPG algorithm.

First, the UAV must learn how to orbit the target. To achieve this, the next orbital trajectory 3D waypoint $\boldsymbol{q}_d$ is dynamically computed on-the-fly at each timestep, using the relevant time-parameterized "Orbit" equations from [3]. Thus, the following error is defined:

$$e_d = ||\boldsymbol{q} - \boldsymbol{q}_d||^2, \tag{8}$$

where $\boldsymbol{q}$ is the actual 3D position of the UAV at the current timestep.

Next, an error threshold is specified, above which the orbital reward becomes zero. Eventually the reward for the orbital trajectory is:

$$r_d \;=\; \begin{cases} 0 & , \; if\ e \geq e_{th}, \\ 1 - \frac{e_d}{e_{th}} & , \; otherwise \end{cases} \tag{9}$$

Additionally, a reward that punishes low or high altitudes is defined. This is motivated by the fact that the footage needs to be captured from an altitude not much greater than that of the filmed target, thus avoiding too large a gimbal pitch angle which would corrupt the CMT. Two options are viable to achieve this: i) explicitly reward small pitch angles, or ii) specify a band of altitudes within which the reward is zero, but outside it is -0.5. Option ii) was chosen for this paper:

$$r_z \;=\; \begin{cases} -0.5 & , \; if\ z \geq z_{max}\ or\ z \leq z_{min}, \\ 0 & , \; otherwise \end{cases} \tag{10}$$

Moreover, obstacle avoidance is an essential factor for robotic navigation within complex environments. Although obstacle avoidance is implemented by the MPC controller and the student DDPG actor will distill that knowledge, an optional obstacle avoidance reward term was still defined for DRL training. This is significantly more accurate than the bounding sphere approximations internally computed by the MPC controller. An on-board LiDAR is used to sense obstacles located within a range of up to $d_{max}$ meters from the UAV, along a FoV of $360^o$ around it. Then, the minimum value $d_{lidar_{min}}$ of the LiDAR output is used to form the reward:

$$r_{obs} = -e^{-a\frac{d_{lidar_{min}}}{d_{max}}}. \tag{11}$$

| Accumulated rewards for the cases of Fig. 1 | | |
|---|---|---|
| Algorithm | Initial Position $[0, 0]$ | Initial Position $[0, 20]$ |
| MPC | 0.72595 | 0.69523 |
| MPC-distilled DDPG | 0.74524 | 0.50416 |
| Baseline DDPG | 0.10650 | 0.39576 |

TABLE I: Accumulated rewards for the cases shown in Fig. 1. Evidently, the MPC-distilled DDPG agent is able to reap higher/lower rewards when the target is visible/invisible at the very first steps. The baseline DDPG agent completely fails to achieve the task and crashes.

A visual occlusion avoidance reward term is defined, since the UAV is expected to capture clear images of the target being filmed while executing the "Orbit" CMT. This reward term was borrowed from [20], where a semantic segmentation map is used during training to find the on-frame surface of the bounding box surrounding the filmed target. The UAV is set to various positions inside the environment, where the target/subject is not occluded from the camera's FoV. Then, the various measured distances between the UAV and the bounding box of the target/subject are used as data pairs for curve fitting a polynomial $\boldsymbol{p}(d)$. The actual reward term is:

$$r_{occ} = e^{-\beta(S_{actual} - S_{measured})^2} \tag{12}$$

where $S_{actual}/S_{measured}$ is the estimated actual/measured bounding box surface, respectively.

Lastly, a reward aiming to smooth out the UAV trajectory is defined, using the angle between the previous action vector and the current one. This reward should be: i) negative when the angle is greater than 90 degrees, and ii) positive when lower than 90 degrees. The highest value of 1 should be at 0 degrees. This behaviour is accurately captured by the cosine similarity between $\boldsymbol{a}_{k-1}$ and $\boldsymbol{a}_k$:

$$r_{smooth} = \frac{\boldsymbol{a}_{k-1}^T \boldsymbol{a}_k}{||\boldsymbol{a}_{k-1}|| ||\boldsymbol{a}_k||} \tag{13}$$

All the individual reward terms described above are combined into the final reward function:

$$r = \sigma_d r_d + \sigma_z r_z + \sigma_{obs} r_{obs} + \sigma_{occ} r_{occ} + \sigma_{smooth} r_{smooth}, \tag{14}$$

where $\sigma_i$ are the reward coefficients.

## V. Experiments

We trained an MPC-distilled DDPG agent in a virtual environment from the AirSim simulator [24]. The environment consists of obstacles in the form of grey boxes and a still target around which an "Orbit" CMT must be performed. The target is orange in color, so that it is clearly distinguishable in the RGB images the UAV gets. The environment is visible in Fig. 1.

Optimal hyperparameter values were found via grid search: the action sampling time was set to $0.4$ seconds, the MPC horizon to $10$ time steps, the reward coefficients to $\sigma_d = 1$ , $\sigma_z = 1$ , $\sigma_{obs} = 0.7$ , $\sigma_{occ} = 0.6$ and the learning rates for the actor and the critic subnetwork to $0.0001$ and $0.0002$ respectively. Identical CNNs were used for the actor and critic subnetworks.

The average rewards for the MPC-distilled DDPG and the baseline DDPG, along with comparisons against the MPC controller, are presented in Fig. 3[1]. Evidently, while the baseline DDPG is unable to learn in $1000$ episodes a good policy, MPC distillation allows the proposed augmented DDPG agent to rapidly learn the desired task in just $200$ episodes. This is extremely fast for such a difficult task in such a complex environment.

Fig. 1 depicts a top-down view of the trajectories that the MPC-distilled DDPG agent and the MPC agent have followed.

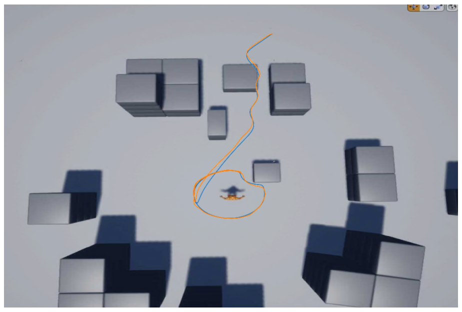


Fig. 2: Trajectory of MPC (in blue) and of MPC-distilled DDPG with smoothness reward (in orange).

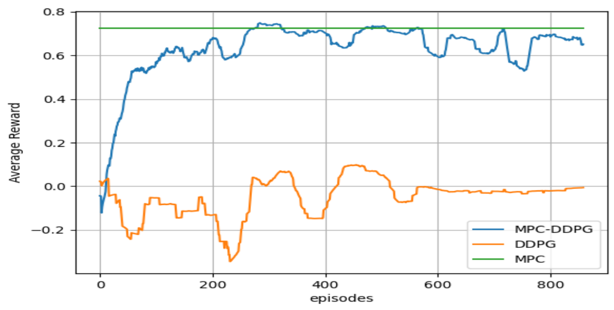


Fig. 3: Average reward for the proposed MPC-distilled DDPG and the baseline DDPG, with both of them trained using the proposed reward function. Evidently, the proposed distillation loss term significantly speeds up learning. The corresponding MPC controller's reward is shown in green, for comparison purposes.

[1]These accumulated rewards per episode do not include the smoothness term.

In (a), (b) the agent's starting position is $[0, 0]$ while in (c), (d) it is $[0, 20]$. Evidently, the two trajectories look similar when there is no occlusion, or when the distance from the obstacles is large enough, but with the proposed MPC-distilled DDPG agent's trajectory being a lot less smooth once the UAV reaches the target's vicinity. However, the proposed agent chose to follow from the start a trajectory that provides the minimum occlusion, while the MPC controller does not account at all for visual occlusions.

The smoothness of the orbital trajectory can be justified by the fact that the training took place for less than 1000 episodes. Since the starting position of the drone is always on a circle around the target, the UAV does not get a lot of opportunities during training to reach the target or complete a full orbital trajectory. Thus, the UAV has learned the optimal behaviour in positions close to the starting points, but the orbital trajectory lacks smoothness and seems uncertain.

In order to compensate for that effect, the smoothness reward of Eq. (13) was introduced along with the use of recurrent layers in the actor and critic network. The result is a much smoother trajectory, achieved with the same number of training episodes. Fig. 2 displays the achieved trajectory on top of the reference trajectory.

It must be clearly emphasized that the effect of the proposed MPC-distilled DDPG algorithm is to render *training* for autonomous UAV CMT execution achievable, by significantly speeding up policy acquisition. As it can be seen in Fig. 3, the baseline DDPG agent does not ever learn to execute the task at all within almost 1000 episodes.

## VI. Conclusions

The method proposed in this paper tackles a complex task from autonomous UAV cinematography, i.e., physically orbiting a target while filming it, using a novel Deep Reinforcement Learning setup. The proposed agent plans on-the-fly the UAV's 3D trajectory waypoints, while relying only on visual input during deployment. However, during training it leverages semi-expert MPC knowledge to reduce DNN size/complexity and significantly lower required training time, while allowing the agent to learn to behave according to the rewards designed for the DDPG algorithm. Thus, the proposed agent is able to choose trajectories satisfying requirements that the semi-expert policy does not account for (i.e., visual occlusion avoidance, improved obstacle avoidance). Although the final behaviour of the trained MPC-distilled DDPG agent is not as smooth as that of the MPC controller, the proposed agent has the advantage of not requiring knowledge of the target's 3D position at each time step. Future work may involve additional trajectory smoothing reward terms at the training stage, in a multitask setting where the camera gimbal is also controlled.

## References

[1] I. Mademlis, V. Mygdalis, N. Nikolaidis, and I. Pitas, "Challenges in autonomous UAV cinematography: An overview," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2018.

[2] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous unmanned aerial vehicles filming in dynamic unstructured outdoor environments," *IEEE Signal Processing Magazine*, vol. 36, pp. 147–153, 2018.

[3] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous UAV cinematography: A tutorial and a formalized shot-type taxonomy," *ACM Computing Surveys*, vol. 52, pp. 1–33, 09 2019.

[4] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, "High-level multiple-UAV cinematography tools for covering outdoor events," *IEEE Transactions on Broadcasting*, vol. 65, no. 3, pp. 627–635, 2019.

[5] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "UAV cinematography constraints imposed by visual target tracking," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2018.

[6] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "Shot type feasibility in autonomous UAV cinematography," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[7] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "Shot type constraints in UAV cinematography for autonomous target tracking," *Information Sciences*, vol. 506, pp. 273–294, 2020.

[8] I. Mademlis, C. Symeonidis, A. Tefas, and I. Pitas, "Vision-based drone control for autonomous UAV cinematography," *Multimedia Tools and Applications*, 2023, accepted.

[9] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," 2015.

[10] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. PP, pp. 1–1, 01 2019.

[11] R. Polvara, M. Patacchiola, S. Sharma, J. Wan, A. Manning, R. Sutton, and A. Cangelosi, "Toward end-to-end control for UAV autonomous landing via deep reinforcement learning," in *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018.

[12] S.-Y. Shin, Y.-W. Kang, and Y.-G. Kim, "Obstacle avoidance drone by deep reinforcement learning and its racing with human pilot," *Applied Sciences*, vol. 9, no. 24, 2019.

[13] Z. Wang, N.D. Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2016.

[14] K. Lee, K. Saigol, and E.A. Theodorou, "Safe end-to-end imitation learning for model predictive control," *arXiv preprint arXiv:1803.10231*, 2018.

[15] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[16] J. Carius, F. Farshidian, and M. Hutter, "MPC-Net: A first principles guided policy search," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2897–2904, 2020.

[17] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2016.

[18] W. M. Czarnecki, R. Pascanu, S. Osindero, S. M. Jayakumar, G. Swirszcz, and M. Jaderberg, "Distilling policy distillation," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*. 2019, PMLR.

[19] N. Passalis and A. Tefas, "Deep reinforcement learning for frontal view person shooting using drones," in *Proceedings of the IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 2018.

[20] R. Bonatti, W. Wang, C. Ho, A. Ahuja, M. Gschwindt, E. Camci, E. Kayacan, S. Choudhury, and S. Scherer, "Autonomous aerial cinematography in unstructured environments with learned artistic decision-making," *Journal of Field Robotics*, vol. 37, no. 4, pp. 606–641, 2020.

[21] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[23] N. Passalis, A. Tefas, and I. Pitas, "Efficient camera control using 2D visual information for Unmanned Aerial Vehicle-based cinematography," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018.

[24] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.