IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 14 June 2023, accepted 31 July 2023, date of publication 9 August 2023, date of current version 16 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3303844

RESEARCH ARTICLE

Detect to Focus: Latent-Space Autofocusing System With Decentralized Hierarchical Multi-Agent Reinforcement Learning

ANNA ANIKINA[1,2], OLEG Y. ROGOV[1], AND DMITRY V. DYLOV[1], (Member, IEEE)
[1]Skolkovo Institute of Science and Technology (Skoltech), 121205 Moscow, Russia
[2]Department of Computer Science, University of Copenhagen, 1172 Copenhagen, Denmark

Corresponding author: Dmitry V. Dylov (d.dylov@skoltech.ru)

This work was supported in part by the RFBR Fund under Grant 21-51-12012.

ABSTRACT State-of-the-art object detection models are frequently trained offline using available datasets, such as ImageNet: large and overly diverse data that are unbalanced and hard to cluster semantically. This kind of training drops the object detection performance should the change in illumination, in the environmental conditions (*e.g.*, rain or dust), or in the lens positioning (out-of-focus blur) occur. We propose a simple way to intelligently control the camera and the lens focusing settings in such scenarios using DASHA, a Decentralized Autofocusing System with Hierarchical Agents. Our agents learn to focus on scenes in challenging environments, significantly enhancing the pattern recognition capacity beyond the popular detection models (YOLO, Faster R-CNN, and Retina are considered). At the same time, the decentralized training allows preserving the equipment from overheating. The algorithm relies on the latent representation of the camera's stream and, thus, it is the first method to allow a completely no-reference imaging, where the system trains itself to auto-focus itself. The paper introduces a novel method for auto-tuning imaging equipment via hierarchical reinforcement learning. The technique involves the use of two interacting agents which independently manage the camera and lens settings, enabling optimal focus across different lighting situations. The unique aspect of this approach is its dependence on the latent feature vector of the real-time image scene for autofocusing, marking it as the first method of its kind to auto-tune a camera without necessitating reference or calibration data.

INDEX TERMS Artificial intelligence, neural networks, reinforcement learning, multi-agent systems, computer vision, photography, imaging, lenses.

I. INTRODUCTION

Modern computational photography is unimaginable without the image acquisition hardware and image analytics tandem [1]. The discipline absorbs the best of both worlds by embedding the state-of-the-art computer vision algorithms into the fastest image processing chips [2]. This tandem ultimately gave birth to the paradigm of ε-photography [3], where a stream of photos could be processed and computationally analyzed on the fly to overcome a wide range of camera limitations. Both limitations, optical (*e.g.*, the lens' limitations) and electronic (*e.g.*, the shot noise), can be eliminated by virtue of pre-trained artificial neural networks (ANN), embedded into the camera's hardware. That enabled a plethora of photo enhancement options for the end-user, unimaginable with a single-exposure camera some 10 years ago. Such algorithmic 'improvements' of the camera's hardware capacity include the higher dynamic range, the larger depth of focus, the broader color gamut, the wider/panoramic shooting, the night photography, and others.

The opposite direction of enrichment in this tandem has been inexplicably underestimated by the community until very recently [4]. In particular, the embedded algorithms rarely use the arsenal of image-improving hardware components within the camera to adjust/update *themselves*. The embedded models are typically pre-trained on offline datasets and, at best, use the recently proposed paradigm of online

The associate editor coordinating the review of this manuscript and approving it for publication was Paulo Mendes.

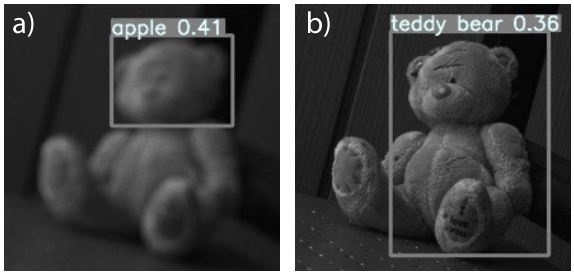


FIGURE 1. (Motivation) Detection models fail to recognize the object in a defocused image (a) . Our autonomous agents learn to adjust the camera and re-focus its lens until the image is good enough for detection models (b). The method requires nothing but the latent vector from the image (no-reference autofocusing).

learning [5] to update the pre-trained ANNs weights, yet, without venturing into the feedback dialogue with the camera hardware.

Recent rapid developments in imaging hardware have provided an opportunity for a new approach in lens autofocusing and exposure control in the image with the aid of feedback-based control loops. Practical applications of automated camera calibrations span from on-the-fly segmentation of road scenes for autonomous vehicles [6] and smartphones [7], to lithography [8], to biomedical imagining [9], [10], becoming indispensable in these areas.

In this study, we were motivated by the recent advances in reinforcement learning (RL), one versatile candidate for filling the hardware feedback niche [11], [12], [13] . We aspired to check if we could train *hardware* in the imaging system to perform image adjustments *live* in order to improve the performance of popular embedded scene analysis models, such as Yolo [14], Faster R-CNN [15], or RetinaNet [16]. In short, we present a perception-inspired automatic focusing system that needs no reference and is supported by hierarchical RL.

The contribution of this study is in the following:

- To the best of our knowledge, this is the first no-reference autofocusing system *based on the low-level representation* of a live scene in the latent space.
- A new paradigm for passive lens autofocusing by means of *hierarchical* reinforcement learning.
- The proposed approach is *decentralized*: both the camera and its lens are controlled by separate agents, preserving the hardware from overheating and guaranteeing functionality in adverse ambient conditions.

II. RELATED WORK

A. AUTO-FOCUSING STATE-OF-THE-ART

To automatically focus on an object, the exact distance from the focal plane to the object should be determined, which could be done by active and passive autofocus (AF).

Active methods include the presence of additional (auxiliary) elements, such as, an ultrasonic locator [17], infrared LEDs [18] or lasers [19].

Passive AF works through the analysis of the image information captured by the system. Passive AF, in turn, can be broadly divided into contrast-based and phase-based detection approaches. Contrast-based AF is performed by determining the image sharpness. To find the lens position, this approach requires capturing a sequence of images with different focal distance and then calculating each image's Focus Measure Value [20], [21], [22], [23], [24], [25], [26]. This sequence significantly affects the performance time. Phase detection methods represent the image by dividing it into the right and the left pixel parts on the camera sensor (the right phase image and the left phase image, respectively [27], [28]). After such splitting, the phase shift between the parts is computed, yielding the relative position between the object and the focal plane. Ultimately, the sign and the extent of the phase shift between the left and the right phase images is used to gauge the optimal lens position to align the focal and the object planes. Ideally, there is an exact correspondence between the phase shift and the travel distance of the lens. However, in practice, the phase shifts are very sensitive to the noise, making it challenging to find the optimal focal position [11], [29], [30].

Recent AF and autoexposure works have started leaning towards *perception* and *object detection* paradigms. For example, [31] features a system that synthetically renders a refocused video from a large-depth-of-focus video, using the upcoming video frames to deliver a context-aware autofocus in the current frame. Likewise, in [32], a center area of the image is partitioned, where the objects are most likely to be located, which is then followed by a battery-consuming frame stability estimation. The study in [33] reports an auto exposure control, optimized for a down-stream task, such as object detection, relying on the intersection-over-union loss. The authors of [34] optimized the shutter speed and the voltage gain on the sensor, relying on the offline dataset of 2.5k pictures in a simple scene with a limited number of known objects.

Despite the success of the classical methods, deep learning tools have been proactively engaged for performing the AF [35], including area-based depth maps [36], [37], [38], per-pixel depths from the multi-view stereo and the super-resolution [39] perspectives.

B. AUTOFOCUS WITH RL

Surprisingly few articles cover the reinforcement learning (RL) approach to the AF problem, with two functional implementations reported ([11] and [12]). These works use the reference value to evaluate the resulting image, subsequently imposing serious restrictions on hardware. Therefore, given a variable ambient environment the computations can be dramatically time-consuming and are unfavorable for the cutting-edge applications in computational photography. We describe these two papers in detail in the Supplement and use them for the comparison in Section V.

C. EXPOSURE ADJUSTMENT WITH RL

Several publications [40], [41] report the use of RL for controlling the exposure time of a camera, all of which do not use a direct learning on the hardware. We also describe these methods in the Supplement.

III. BACKGROUND: MULTI-AGENT RL & HIERARCHY

A. AGENTS AND OPTIMIZATION PROBLEM

Consider RL problem as a Markov Decision Process (MDP) formally represented in form of tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma,)$, where $\mathcal{S}$ is a finite set of states and $s_t \in \mathcal{S}$ denotes that agent at time t being in state s; $\mathcal{A}$ is a set of actions and agent interacts with the environment by choosing action $a_t \in \mathcal{A}$ at time t; $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the transition probability distribution describes the probability of arriving from s_{t-1} to s_t by choosing a_{t-1}; $\mathcal{R}$ is a scalar reward function that controls the behavior of agent: the environment gives reward to agent for each action; and $\gamma \in [0, 1]$ is the discount factor providing the importance of immediate reward versus future rewards.

Conventionally [42], [43], [44], two important concepts are introduced: the value and the state value functions. The state value function V^π is the cumulative sum of future rewards for the agent following the policy π from the state s:

$$V_t^\pi(s) = \mathbb{E}\left[\sum_{i=t}^{H-1} \gamma^{i-t} r_i | s_t = s\right]. \tag{1}$$

The state-action value function Q_t^π defines the expected discounted sum of rewards $\sum_{t=0}^{N} r_t$ from state s at time t to the H following the policy π:

$$Q_t^\pi(s, a) = \mathbb{E}\left[\sum_{i=t}^{H-1} \gamma^{i-t} r_i | s_t = s, a_t = a\right]. \tag{2}$$

The V^π provides the best value following policy π for each of next states, while Q^π shows the effectiveness of actions that the agent chooses by following the policy π for the next states. The corresponding objective function one wants to maximize over the policy parameters θ is:

$$L(\theta) = \hat{\mathbb{E}}_t\left[\log \pi_\theta(a_t|s_t)\hat{A}_t\right] \approx \hat{\mathbb{E}}_t\left[\hat{r}_t\hat{A}_t\right], \tag{3}$$

where $\hat{\mathbb{E}}_t$ denotes the empirical expectation over time steps, $\hat{A}_t := Q_t^\pi(s, a) - V_t^\pi(s)$ is the estimated advantage at time t, and $\hat{r}_t = \frac{\pi(a_t|s_t)}{\pi_{old}(a_t|s_t)}$ is the probability ratio under the new and old policies.

One of the most widespread optimization methods employed for RL is the Proximal Policy Optimization (PPO). To avoid an instability, one typically penalizes large policy changes to stay within a small interval $[1-\epsilon, 1+\epsilon]$, where ϵ is a hyperparameter, yielding the ultimate objective function needed for our work [43]:

$$L(\theta) = \hat{\mathbb{E}}_t\left[\min\left(\hat{r}_t(\theta)\hat{A}_t, clip\left(\hat{r}_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right], \tag{4}$$

where *clip* stands for the clipped objective restricting the values of $\hat{r}_t(\theta)$.

In our study, PPO was chosen as the main optimization method, because it provides a universally robust performance in different environments and on different imaging hardware [43]. The comparison with other RL optimization approaches is beyond the scope of this work.

B. HIERARCHICAL RL

Hierarchical Reinforcement Learning (HRL) is the case of multi-agent RL and has multiple hierarchy layers of policies compared to the conventional single-agent RL. In the standard two-tier HRL, high-level agent A^H and low-level agent A^L are defined with corresponding policies π^H and π^L. The higher-level policy sets goals for the lower-level policy, and the lower-level policy attempts to reach them. At the beginning of each episode, the higher-level policy receives the state (or observation) and forms a high-level action (or goal). Then, the low-level policy receives an observation and a goal and forms a low-level action that affects the environment. The low-level agent receives a reward for each step; at the end of the episode, the high-level agent receives the final reward, and then the process is repeated.

Our approach will differ from the classic HRL. The high-level agent will establish a 'favorable' RL-environment for the low-level agent to act in: until the optimal parameters of the internal environment are reached, the low-level agent does not take control. The high-level agent is responsible for the camera parameters (the exposure time t_{ex}), while the low-level agent works with the lens: the task is to get the selected object in focus.

C. OBJECT DETECTION

Object detection is an essential part of the proposed autofocusing system. In this work, we rely on popular object detection models, such as Yolo [14], RetinaNet [16], and Faster R-CNN [15] in their intact ('off-the-shelf') variants. Their description and implementation details are given in the Supplement.

IV. METHODS: DASHA

This section covers the method of controlling the camera's exposure and focusing, summarized in Fig. 2, using the decentralized hierarchical reinforcement learning framework with two agents: A^H (high-level agent, Algorithm 1) and A^L (low level agent, Algorithm 2). We begin by formally defining the state space, the action space, and the rewards for these agents.

A. STATE SPACE

We take advantage of deep auto-encoders, acquiring a low-dimensional feature space $\mathcal{S}$. The A^L observation space $s_t \in \mathcal{S}$ is a feature vector of size $[2048 \times 1]$ from the camera's image $\mathcal{I}$, obtained using a ResNet-152 Encoder $E(\cdot)$ pre-

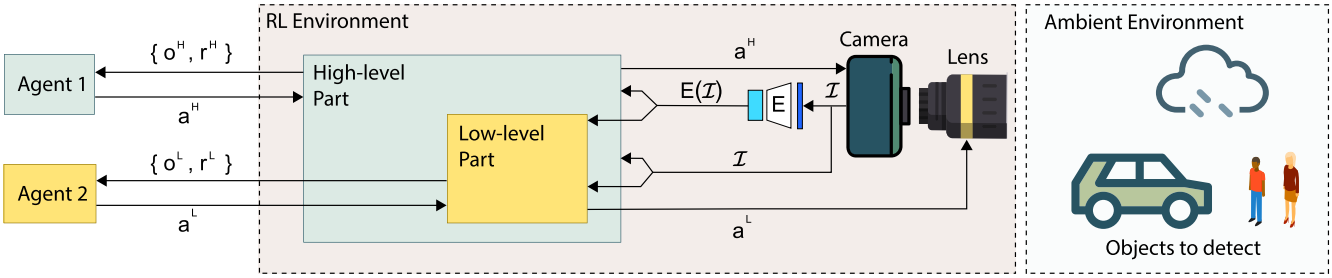


FIGURE 2. Decentralized autofocusing system with hierarchical RL agents (DASHA). There are two agents operating in the RL-Environment: a high-level agent (Agent 1: turquoise) and a low-level agent (Agent 2: yellow). o, r, a - observation, reward, and agent's action, respectively. The superscripts H and L indicate the high-level and the low-level agents, respectively; $\mathcal{I}$ is the image frame from the camera; E is a pre-trained ResNet-152 encoder, and E($\mathcal{I}$) is the latent space feature vector.

trained [45] with ImageNet [46]:

$$o^L = E(\mathcal{I}). \tag{5}$$

The observation space of A^H is a tuple of features and the histogram data (values of the histogram *val* [10 × 1] and bin edges *bin* [11 × 1]):

$$o^H = (E(\mathcal{I}), val, bin). \tag{6}$$

B. ACTION SPACE

Each policy has its own set of actions. The action space for the A^H is a discrete space of 146 values, where each value corresponds to its own exposure time t_{ex}. This size of the array of the possible exposure time values allows for the non-linear nature of the exposure scale and empirically covers the entire range of needed exposure time values. Otherwise, the action space would have consisted of 5 million values, significantly increasing the training time. The action space for A^L is a multi-discrete space of 2 values: the coarse and the fine, enabling to tune the Focus Measure Value. The minimum control signal that can be sent to the lens is 24.0 (in arbitrary voltage units of the focusing servomotor), and the maximum value is 70.0. In our calibration experiments, we determined that these values correspond to the physical distance between the camera's sensor and the lens of 36.4 and 24.0 mm, respectively.

Algorithm 1 High-Level Agent

1: A^H sends action a^H
2: Change t_{ex} in camera
3: Grab image $\mathcal{I}$
4: Evaluate a^H of A^H by computing `histogram`
5: Form observation o^H from $\mathcal{I}$ using encoder $E(\cdot)$ and `histogram`
6: Compute reward r^H
7: **if** (`histogram` < 10% *max* (too dark) or histogram > 90% *max* (too bright)) **then**
8: Episode completed with penalty for A^H
9: **else**
10: Pass action to A^L (Low-Level Agent) or successfully end episode (in case of Single Agent)
11: **end if**
12: **return** $(o^H, r^H, done)$

C. REWARD FUNCTION

The A^H gives the A^L an intrinsic reward for each step. The reward is defined by a piecewise function that takes into account different cases of agent behavior.

To properly construct the reward function, the following principles were followed:

- *Speed*. The faster the agents reach the desired state, the better. The more unsuccessful steps an agent needs, the greater is the penalty. We accelerate the training of the RL agents by the negative rewards, which force the end of the episode as soon as possible.
- *Encouragement for staying close to the focus*. When an action leads to a successful detection of a bounding box of an object, but the image quality assessment yields a metric value insufficient to pass the required threshold, the agent receives a reduced penalty, as it is generally moving in the right direction. The closer to the focus, the smaller the penalty.
- *The reward function convergence*. An upper bound is set for the reward that the agents can receive for their actions. This provides a clear graph of the agent's convergence (to visualize the correct actions) and keeps the maximum reward within 1.

Given these principles, the rewards can be formally determined as follows.
For the agent A^H,

$$r^H = \begin{cases} +1, & \text{if } P \in [50, 150], \\ -0.01 \cdot \mathcal{B}, & \text{if } P \in [25, 50) \cup (150, 175], \\ -1, & \text{otherwise.} \end{cases} \tag{7}$$

In Eq. (7), P is the peak histogram value and $\mathcal{B}$ is a no-reference image quality metric obtained on the low-level representation of the image by Eq. (5). Each step, A^H relies on the brightness histogram of the image, with the piece-wise function in Eq. (7) being guided by the following rationale. If the peak histogram value P falls within the interval

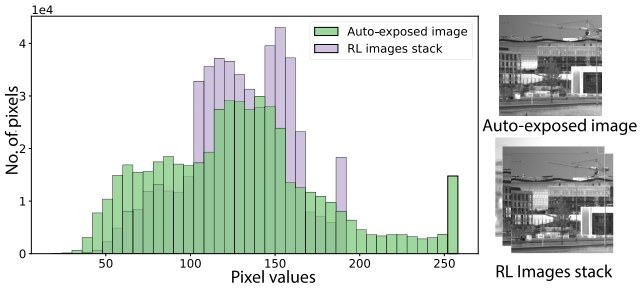


FIGURE 3. Histogram comparison for the RL-images stack (purple) and the image obtained by a built-in algorithm embedded into the camera (green). Taken to demonstrate consistency of our single RL agent that performs automatic exposure tuning.

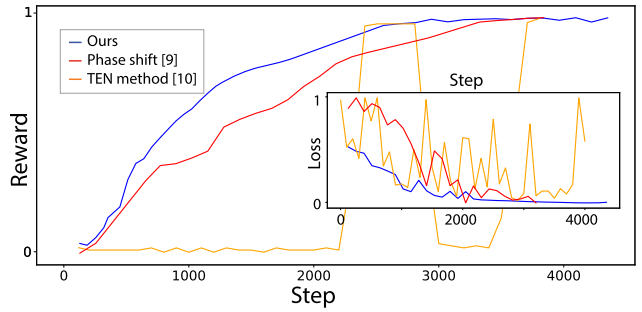


FIGURE 4. Learning curves of the competing autofocusing methods, demonstrating faster convergence of DASHA. Inset compares the three loss functions.

from 50 to 150 pixel values, the exposure of the image is considered satisfactory. Otherwise, the agent receives a penalty.

The image, however, is still discernible if P belongs to the interval from 25 to 50 or from 150 to 175 pixel values. In this case, the action is considered partially successful and a small negative reward is received. The Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) [47] (thereafter referred to as just $\mathcal{B}$) is calculated: the lower the value, the better the image. The detected image is further evaluated by the metric. The better the image, the lower the metric, the higher the reward for the agent. So, the agent receives information about which image is "bad" and which is "excellent". That is why we refer to such adjustments of the camera as "no-reference", implying that there is no ground truth image for the agents to learn from. Instead of the ground truth image, the agents rely on the blind metric $\mathcal{B}$ to gauge whether a given action improves or degrades the quality of the recorded scene. *For the agent* A^L,

$$r^L = \begin{cases} -1, & \text{if detection fails,} \\ -0.01 \cdot \mathcal{B}, & \text{otherwise.} \end{cases} \tag{8}$$

In Eq. (8), for each step when the object is not detected, the agent receives a penalty. In the case when the object detection is successful, the computation of $\mathcal{B}$ is still needed in order to further tune the sharpness of the image while the lens is being moved in the vicinity of that position where the first successful detection event occurred. Thus, the agent aims to reach the lowest possible value $\mathcal{B}$ while tending to converge to zero, which ultimately brings the lens to the optimal position.

D. HIGH-LEVEL AGENT A^H

This agent manages the built-in camera settings. The main role of this agent is to set up an 'optimal environment' for the A^L. At the beginning of each episode, the camera and the lens have the initial parameters set up with their 'factory' values. These parameters do not always match the ambient environment optimally, resulting in a poorly exposed scene. Hence, the correct configuration of these parameters is needed so that a visible image can be obtained. The A^H sends an action a^{H} to the camera, which changes the exposure time. The resulting image from the camera $\mathcal{I}$ is analyzed for content by constructing a histogram. If the image is good, the control of the RL environment is transferred to the A^L. If the image is unacceptable, the episode ends immediately with a penalty to the A^H. This routine is summarized in Algorithm 1.

E. LOW-LEVEL AGENT A^L

The A^L adjusts the lens position for an in-focus image acquisition. As soon as the action of the agent a^L is commanded to the lens, a new image is acquired from the camera. This image $\mathcal{I}$ is then sent to the object detection model that returns the binary result: the object is successfully *detected* (in this case, the coordinates of the object are extracted) or the object is *not detected*. All these actions are repeated until the moment of successful detection or until the end of the episode. In the case of successful object detection, the resulting image is evaluated for the quality using the metric $\mathcal{B}$ and the reward r^L is received, as outlined in Algorithm 2.

V. EXPERIMENTS

For the experiments, we used the 2/3" Basler acA2000-50gm camera (GigE, CMV2000 CMOS, 50 frames per second at a 2 megapixel resolution). The access to the camera is established via the official Basler library `pypylon`. We also employed the Corning Varioptic C-C-39N0-250 Lens controlled through the cp210x board (Silicon Labs Ltd.) and a DLL-file provided by Corning. To implement RL and to operate both the camera and the lens, we used the `rllib`

Algorithm 2 Low-Level Agent

A^L sends action a^L
Change lens position
Acquire image $\mathcal{I}$
if (Object is detected) **then**
    Successful completion of episode
else
    Send new action a^L until episode is `done`
end if
Form observation o^L from $\mathcal{I}$ using encoder $E(\cdot)$
Compute reward r^L
return $(o^L, r^L, done)$

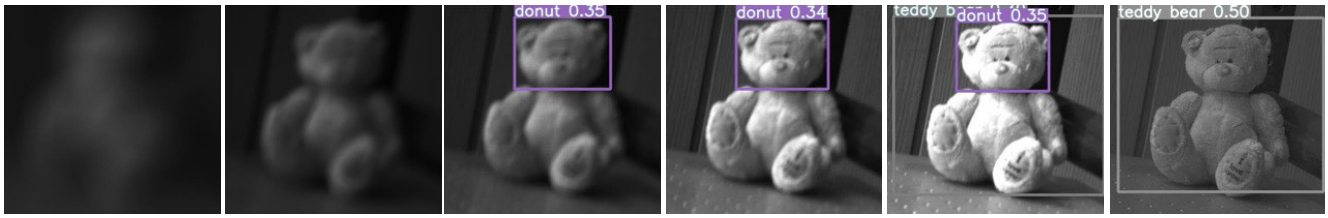


FIGURE 5. Consecutive iteration steps visualizing the effect of the exposure adjustment and the lens displacement on the detection of the object. Note how the detection certainty increases as the exposure and the position of the lens approach their optimal values. Intuitively, for this particular scene with a shallow depth-of-the-field (camera aperture of f/2), the position of the lens matters more than the proper exposure. Ambient conditions with the illuminance of 85 lx and the open camera aperture correspond to dark and blurry scene.

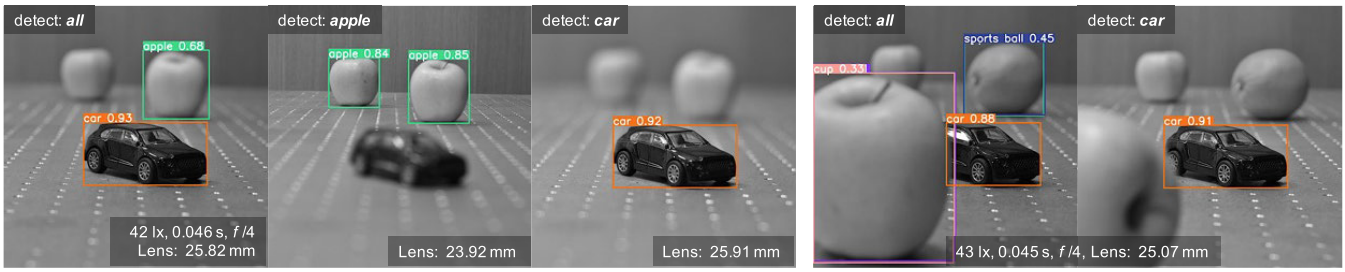


FIGURE 6. Multiple objects. Demonstration for 3-4 objects placed 0.5 m apart. Detection models have a prompt to search either *all* known objects or *some* specific ones. In detecting *all*, DASHA fine-tunes the focus for the object with the highest score. If one wants to detect, say, apples *only*, the agents adjust the camera to max the score for the apple(s) .

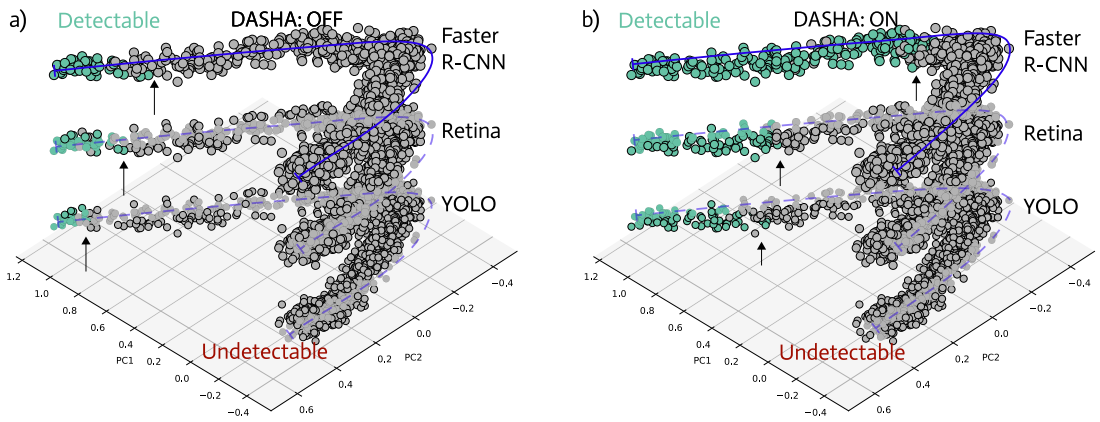


FIGURE 7. PCA plots demonstrating enhancement of object detection capacity of the popular models by DASHA. When DASHA is switched on, the detection boundary moves rightwards (highlighted by arrows). Each dot corresponds to an image taken at fixed illumination 37 lx and adjustable exposure time and focus. Blue lines are drawn to guide the eye.

library [48] on OS Windows 10 64 bit, 16 GB RAM, CPU Core i7 3.6 GHz.

For training either full DASHA model or the agents separately, the PPO was performed with the following parameters: the number of workers 1, the entropy coefficient 0.01, the rollout fragment length 128, the train batch size 128, the SGD minibatch size 128, the learning rate 25×10^{-5}. We use pre-trained models, like YOLOv5 (600 object categories), with no modifications or re-training ('off-the-shelf' or 'intact'). We detect 4K data points in 8 categories (cars, fruits, people, etc.), with at least 400 instances acquired in each category. Required staticity is set by AF speed (ours is the fastest in Table 1) in training RL; and by the lens response of the servo-motors in inference (nearly instantaneous). We use bounding boxes found by the pre-trained models with detection certainty (Fig. 1 in the Supplmement).

A. SINGLE AGENT TRAINING

Prior to training the multi-agent RL, we trained single agents separately to gauge their stand-alone capacity to autofocus and to control the exposure. Besides enabling us to assess the ablation effect, this study also allows for proper comparison

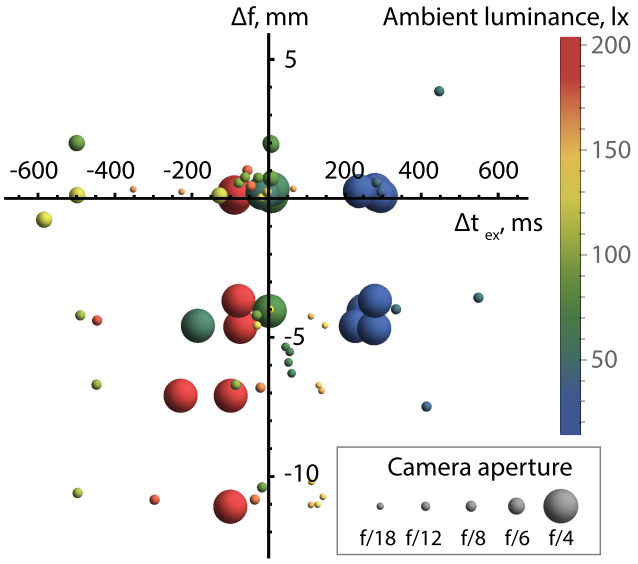


FIGURE 8. DASHA's simultaneous optimization of the displacement of the lens Δf (in mm) and of the exposure time Δt_{ex}(in ms).

with the prior RL art, where both tasks have never been dealt with simultaneously.

1) AUTOFOCUSING AGENT ALONE

The AF agent can be trained following the stand-alone Algorithm 2 of A^L, assuming a fixed ambient illumination and a properly exposed histogram. The learning routine for this experiment can consist of altering the object position along the optical axis of the camera, which we performed once every 30 minutes.

In order to demonstrate the advantage of the proposed agent system in the AF task, we report a comparison with [11] and [12] as with the most relevant studies. By replicating the reward functions and the observation space described by the authors, we could perform a consistent comparison, shown in Table 1. The image quality assessment (*IQA*) is the method implemented to assess the acquired image within the RL-environment. Opposite to our model, both benchmarks are reference-based (RB), requiring the ground-truth image to compute the output. The *AF time* shows the average time (an average over 10 experiments) required for finding the in-focus lens placement given a random position of the object along the optical axis. Our model is comparable to the modern Phase Detection AF (PDAF [7]) in terms of the AF time; but, unlike PDAF, it requires no extra phase components on the sensor [49].

In [12], the time required to find the proper Focus Measure Value is obtained in the following way. An image is acquired for every lens position with a respective calculation of the Focus Measure Value. The in-focus lens position, then, is the one with the highest Focus Measure Value. In this manner, the full cycle lasts 366 sec (an average over 10 experiments). In our experimentation with this benchmark, we witnessed the undesirable occasions, when it took more than 10 steps to get certain scenes in focus. Whereas, our model, once trained, required 2 steps on average to find the focus regardless of the arrangement of the objects (similarly averaged over 10 runs). *Virtual pre-training* is another feature in the benchmark [12] that we avoided altogether, because the *Deep Image Features (the latent space)* is a more efficient way to represent the distribution of data.

TABLE 1. Comparison of existing RL-based autofocusing methods: key practical measurements.

	Phase shift [11]	TEN [12]	**DASHA**
IQA	RB	RB	**NR**
AF time (sec)	58	366	10
AF steps	12.5	-	2
Virtual pre-train	✗	✓	✗
DF	✗	✗	✓

Note: *IQA* – Image Quality Assessment, *RB* – reference-based, *NR* – no-reference, *AF time* – average time to find focus, *AF steps* – the average number of steps to find focus, *Virtual pre-train* – a pre-trained agent before the online work with hardware, *DF* – deep image features (latent space). [1]

2) EXPOSURE-CONTROLLING AGENT ALONE

In this case, the agent only controls the t_{ex} parameter of the camera, interacting with the environment according to Algorithm 1. Here, a variety of illumination conditions is used for training, ranging from a completely dark room ($E_v =$ 13 lx) to a bright lighting ($E_v = 300$ lx). E_v is adjusted every 30 min with an increment of $\delta E_v = 10$ lx in a looped cycle, to make the model learn the varying ambient conditions. The outcome can be compared with the 'brute force' embedded AutoExposure method from the Basler camera. We confirmed that the RL agent functions similarly to the embedded software without any exposure metering or without the phase difference elements (Fig. 3).

B. MULTI-AGENT TRAINING

In the full DASHA configuration, both agents are allowed to 'learn' and to perform together. The high-level agent controls the exposure time t_{ex}, and the low-level agent attempts to place the lens so that the image is in focus. The training occurred with an alternating object position (140 cm – 200 cm away from the camera) and the ambient illumination (from $E_v = 13$ lx to $E_v = 300$ lx altered in a loop), with each new change introduced every 30 minutes. On average, the training process takes about 12 hours, with typical learning curves shown in Fig. 4. Here, we used Yolo to detect objects because of its faster inference. An example of a training scene, iteratively improving with the adjustment of the exposure and the position of the lens, is shown in Fig. 5; while Fig. 7 shows a summary of a large-scale study of DASHA's performance. The plots show 4×10^3 points in the PCA space, where each point corresponds to some image with a unique focusing distance.

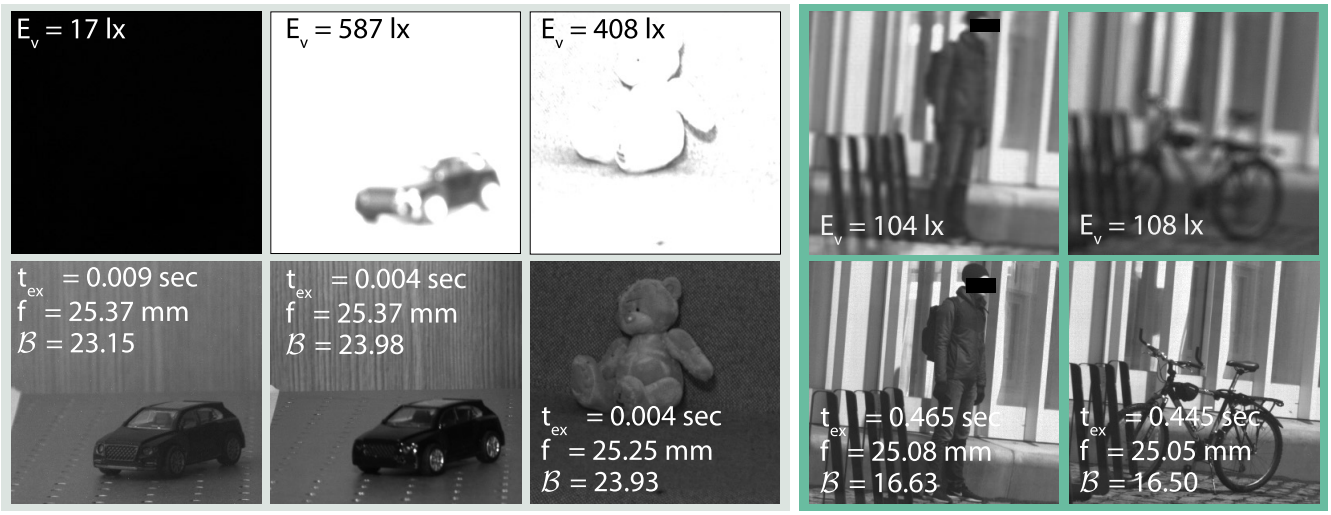


FIGURE 9. Automatic enhancement of poorly-lit, overexposed, and/or out-of-focus scenes by DASHA on-the-fly (inference), featuring objects located in the lab (left panel) and on the street (right panel). ***First row***: examples of initial frames. ***Second row***: RL-optimized results. DASHA finds optimal exposure time t_{ex} and focus $\mathtt{f}$ for each scene, optimizing nothing but the no-reference metric $\mathcal{B}$.

VI. DISCUSSION

Notice the enhancement of the object detection capacity of the popular models by DASHA in Fig. 7, with the detection boundary moving rightwards in the plot when our agents are active (the boundary is highlighted by arrows). The 'green' points when the model is enabled ('DASHA: ON'), corresponding to the images where the detection is successful, *move along the distribution* of the PCA data even for those images where the detection initially failed (compare to 'DASHA: OFF').

As such, this reports the aforementioned bi-directional enhancement of the capacity of the hardware-software tandem: the hardware 'learns' to adjust itself, using the outcome of the detection software; whereas, the detection models themselves begin to operate beyond their original limits (*i.e.*, detecting dark and blurry scenes becomes easier). At different t_{ex}, the SOTA methods detect objects differently, which explains the importance of introducing an agent that controls t_{ex} in the camera alongside the lens focusing agent.

Possible limitations: detection models can misclassify or miss objects, unless controlled by a targeted category search (Fig. 6). DASHA performs similarly to other passive AF methods in unusual and abstract scenes without detectable objects (like focusing on the sky or a blank wall; all passive methods would be bad at that).

Additionally, we performed a series of inference experiments for various ambient illuminations and various apertures of the camera (Fig. 8), confirming that the system is robust with regard to different depth of the field and ambient illumination conditions. Note how the system learned to compensate for the extra light entering the frame either by subtracting some ms of the exposure time or by proper re-focusing.

However, given that it relies on the latent representation, in some cases in Fig. 8, it decides to blur the image on purpose to make up for the missing light. These are the actions learned by DASHA which maximize the *no-reference* image quality. We speculate that the detection models at the core of our method are not that dependent on the image quality, being capable of functioning well even with a limited number of the crucial features pertinent to the detectable object. Remarkably, this trait is in no way an obstacle for the hardware to perform the autofocusing.

Visual examples of indoor and outdoor scenes, together with the parameters learnt by the camera and the lens, are presented in Fig. 9. The first row shows the initial images acquired by the camera at the beginning of the imaging process. The second row shows the same scenes marked as suitable according to the RL algorithm. These examples demonstrate how the over-/under-exposed and the blurry images are efficiently adjusted by DASHA, resulting in a successful object detection through the no-reference image quality assessment and the hardware tuning.

VII. CONCLUSION

In this work, we proposed a new approach to the problem of auto-tuning imaging equipment using hierarchical reinforcement learning. This methodology allows for correcting the improper focusing under various illumination conditions by employing two decentralized interacting agents that control the settings of the camera and the lens. The proposed way of autofocusing relies on the latent feature vector of the live image scene, being the first such method to auto-tune a camera without reference or calibration data.

The system proved efficient for detecting objects in the dark and the blurry initial states when three SOTA object detection methods originally failed to function. Being fast to learn (Fig. 4), our algorithm preserves the hardware from overheating, requires no active or phase-detection elements, and is functional for a range of camera apertures (*i.e.*, robust to the depth of the field variation). Decentralized HRL

reduces the load on the hardware during training, because it eliminates the need to engage the focusing servomotors until the electronic settings of the camera are adjusted to yield a well-exposed scene with sufficient object contrast.

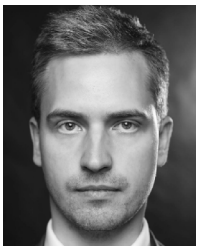
One possible limitation of the current study is that we considered focusing only on the natural scenes (those that the models like Yolo, Retina, *etc.* are acquainted with). Standard domain adaptation and transfer learning methods should mitigate the problems of applying our method to objects of a completely different nature (*e.g.*, cells in microscopy). Also, although our initial study partially explains the decisions made by DASHA (Fig. 8), an in-depth interpretability study is to be conducted, using such feature-based feedback solutions as Grad-CAM [50] or such targeted filtering techniques as GAFL [51]. We envision simple integration of our framework with a wide range of consumer cameras and motorized lenses, enabling a widespread improvement both to the imaging hardware and to the embedded object detection models.

REFERENCES

[1] R. Lukac, *Computational Photography: Methods and Applications*. Boca Raton, FL, USA: CRC Press, 2017.
[2] R. G. VidalMata et al., "Bridging the gap between computational photography and visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4272–4290, Dec. 2021.
[3] R. Raskar, "Computational photography: Epsilon to coded photography," in *Emerging Trends in Visual Computing*. Berlin, Germany: Springer, 2008, pp. 238–253.
[4] A. Ito, S. Tambe, K. Mitra, A. C. Sankaranarayanan, and A. Veeraraghavan, "Compressive epsilon photography for post-capture control in digital imaging," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–12, Jul. 2014.
[5] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *Proc. 24th Eur. Symp. Artif. Neural Netw.*, 2016.
[6] K. Zheng and H. A. H. Naji, "Road scene segmentation based on deep learning," *IEEE Access*, vol. 8, pp. 140964–140971, 2020.
[7] A. Abuolaim, A. Punnappurath, and M. S. Brown, "Revisiting autofocus for smartphone cameras," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 545–559.
[8] Z. Ren, Z. Xu, and E. Y. Lam, "Autofocusing in digital holography using deep learning," *Proc. SPIE*, vol. 10499, Feb. 2018, Art. no. 104991V.
[9] A. J. Hunter, B. W. Drinkwater, and P. D. Wilcox, "Autofocusing ultrasonic imagery for non-destructive testing and evaluation of specimens with complicated geometries," *NDT & E Int.*, vol. 43, no. 2, pp. 78–85, Mar. 2010.
[10] V. M. Leli, A. Rubashevskii, A. Sarachakov, O. Rogov, and D. V. Dylov, "Near-infrared-to-visible vein imaging via convolutional neural networks and reinforcement learning," in *Proc. 16th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Dec. 2020, pp. 434–441.
[11] C.-C. Chan and H. H. Chen, "Autofocus by deep reinforcement learning," in *Proc. IST Int. Symp. Electron. Imag.*, 2019, pp. 577-1–577-5.
[12] X. Yu, R. Yu, J. Yang, and X. Duan, "A robotic auto-focus system based on deep reinforcement learning," in *Proc. 15th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Nov. 2018, pp. 204–209.
[13] Z. Yao, X. Liang, G.-P. Jiang, and J. Yao, "Model-based reinforcement learning control of electrohydraulic position servo systems," *IEEE/ASME Trans. Mechatronics*, vol. 28, no. 3, pp. 1446–1455, 2023.
[14] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.
[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015, *arXiv:1506.01497*.
[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.
[17] H. Liang, K. Lu, X. Liu, and J. Xue, "The auto-focus method for scanning acoustic microscopy by sparse representation," *Sens. Imag.*, vol. 20, no. 1, p. 33, Dec. 2019.
[18] P. Robisson, J.-B. Jourdain, W. Hauser, C. Viard, and F. Guichard, "Autofocus measurement for imaging devices," *Electron. Imag.*, vol. 29, no. 12, pp. 209–218, Jan. 2017.
[19] X. Zhang, F. Fan, M. Gheisari, and G. Srivastava, "A novel auto-focus method for image processing using laser triangulation," *IEEE Access*, vol. 7, pp. 64837–64843, 2019.
[20] S. Pertuz, D. Puig, and M. A. Garcia, "Analysis of focus measure operators for shape-from-focus," *Pattern Recognit.*, vol. 46, no. 5, pp. 1415–1432, May 2013.
[21] G. Saini, R. O. Panicker, B. Soman, and J. Rajan, "A comparative study of different auto-focus methods for mycobacterium tuberculosis detection from brightfield microscopic images," in *Proc. IEEE Distrib. Comput., VLSI, Electr. Circuits Robot. (DISCOVER)*, Aug. 2016, pp. 95–100.
[22] C.-Y. Chen, R.-C. Hwang, and Y.-J. Chen, "A passive auto-focus camera control system," *Appl. Soft Comput.*, vol. 10, no. 1, pp. 296–303, Jan. 2010.
[23] X. Xu, Y. Wang, J. Tang, X. Zhang, and X. Liu, "Robust automatic focus algorithm for low contrast images using a new contrast measure," *Sensors*, vol. 11, no. 9, pp. 8281–8294, Aug. 2011.
[24] R. Chen and P. van Beek, "Improving the accuracy and low-light performance of contrast-based autofocus using supervised machine learning," *Pattern Recognit. Lett.*, vol. 56, pp. 30–37, Apr. 2015.
[25] X. Zhang, Z. Liu, M. Jiang, and M. Chang, "Fast and accurate autofocusing algorithm based on the combination of depth from focus and improved depth from defocus," *Opt. Exp.*, vol. 22, no. 25, p. 31237, 2014.
[26] X. Zhang, K. Matzen, V. Nguyen, D. Yao, Y. Zhang, and R. Ng, "Synthetic defocus and look-ahead autofocus for casual videography," in *Proc. SIGGRAPH*, 2019, pp. 1–16.
[27] M. G. Gluskin, R. M. Velarde, and J. Lee, "Phase detection autofocus using masked and unmasked photodiodes," U.S. Patent 9 804 357 B2, Oct. 31, 2017.
[28] W.-I. Hsu, D.-N. Yaung, F.-C. Hung, and K.-Y. Chou, "Phase detection autofocus techniques," U.S. Patent 9 905 605 B2, 2015.
[29] C.-C. Chan, S.-K. Huang, and H. H. Chen, "Enhancement of phase detection for autofocus," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 41–45.
[30] M. G. Gluskin, R. M. Velarde, and J. Lee, "Phase detection autofocus noise reduction," U.S. Patent 9 420 164 B1, Aug. 16, 2016.
[31] X. Zhang, K. Matzen, V. Nguyen, D. Yao, Y. Zhang, and R. Ng, "Synthetic defocus and look-ahead autofocus for casual videography," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–16, Aug. 2019.
[32] L. Peng, "Enhanced camera capturing using object-detection-based autofocus on smartphones," in *Proc. 4th Int. Conf. Appl. Comput. Inf. Technol./3rd Int. Conf. Comput. Sci./Intell. Appl. Inform./1st Int. Conf. Big Data, Cloud Comput., Data Sci. Eng. (ACIT-CSII-BCD)*, Dec. 2016, pp. 208–212.
[33] E. Onzon, F. Mannan, and F. Heide, "Neural auto-exposure for high-dynamic range object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7706–7716.
[34] Y. Wu and J. Tsotsos, "Active control of camera parameters for object detection algorithms," 2017, *arXiv:1705.05685*.
[35] C. Herrmann, R. S. Bowen, N. Wadhwa, R. Garg, Q. He, J. T. Barron, and R. Zabih, "Learning to autofocus," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2227–2236.
[36] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 2, 2014, pp. 2366–2374.
[37] M. Carvalho, B. L. Saux, P. Trou-Peloux, A. Almansa, and F. Champagnat, "Deep depth from defocus: How can defocus blur improve 3D estimation using dense neural networks?" in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 307–323.
[38] R. Garg, N. Wadhwa, S. Ansari, and J. Barron, "Learning single camera depth estimation using dual-pixels," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7627–7636.
[39] C. Wang, Q. Huang, M. Cheng, Z. Ma, and D. J. Brady, "Deep learning for camera autofocus," *IEEE Trans. Comput. Imag.*, vol. 7, pp. 258–271, 2021.
[40] H. Yang, B. Wang, N. Vesdapunt, M. Guo, and S. B. Kang, "Personalized exposure control using adaptive metering and reinforcement learning," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 10, pp. 2953–2968, Oct. 2019.
[41] R. Yu, W. Liu, Y. Zhang, Z. Qu, D. Zhao, and B. Zhang, "DeepExposure: Learning to expose photos with asynchronously reinforced adversarial learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 31, 2018, pp. 2153–2163.

[42] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, vol. 12, S. Solla, T. Leen, and K. Müller, Eds. Cambridge, MA, USA: MIT Press, 2000.
[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[44] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. 32nd Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, vol. 37, Lille, France, 2015, pp. 1889–1897.
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[47] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-reference image quality assessment in the spatial domain," *IEEE Trans. Image Process.*, vol. 21, no. 12, pp. 4695–4708, Dec. 2012.
[48] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," 2017, *arXiv:1712.09381*.
[49] L. F. Hovela, "Learning how to focus: Adaptive control of an autofocus camera," *Proc. SPIE*, vol. 1656, pp. 383–393, Aug. 1992.
[50] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.
[51] V. Shipitsin, I. Bespalov, and D. V. Dylov, "GAFL: Global adaptive filtering layer for computer vision," *Comput. Vis. Image Understand.*, vol. 223, Oct. 2022, Art. no. 103519.

ANNA ANIKINA received the bachelor's degree in applied physics and mathematics from the Moscow Institute of Physics and Technology, Dolgoprudny, Russia, in 2018, and the M.Sc. degree in information science and technology from the Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia, in 2020. She is currently pursuing the Ph.D. degree with the University of Copenhagen, Denmark. Her research interests include machine learning and medical applications.



OLEG Y. ROGOV received the M.Sc. degree in physics from Lomonosov Moscow State University, Moscow, Russia, and the Ph.D. degree in mathematical modeling and physics from the Russian Academy of Sciences, Moscow. From 2015 to 2019, he was a Research Engineer with the Keldysh Institute of Applied Mathematics, Moscow. He is currently a Research Scientist with the Computational Imaging Group, Skolkovo Institute of Science and Technology (Skoltech). His current research interests include among others, evolutionary algorithms and deep learning, information fusion and decision making, and data science (fundamentals of imaging system design and performance evaluation, signal processing, detection, and estimation).

DMITRY V. DYLOV (Member, IEEE) received the M.Sc. degree in applied physics and mathematics from the Moscow Institute of Physics and Technology, Moscow, Russia, in 2006, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 2010. He is currently an Associate Professor and the Head of the Computational Imaging Group, Skolkovo Institute of Science and Technology (Skoltech), Moscow. His research interests include computational imaging, computer/medical vision, and the fundamental aspects of image formation.