Learning to Control Camera Exposure via Reinforcement Learning

Supplementary Material

1. Appendix

In this supplementary material, we provide

- Implementation and training details
- Further discussion of RL component design
- Discussion and future works

2. Implementation Details

Network Architecture. We adopt simple Multi-Layer Perceptron (MLP) layers as our DRL agent architectures (*i.e.*, actor and critic). Tab. 1 shows the architecture detail from the input to the output layers. Both actor and critic networks consist of one input layer, two intermediate layers, and one output layer. The dimensions of each network's input and output layers are determined by the dimensions of state and action vectors. Given the state vector, the actor network estimates next step actions (*i.e.*, exposure time and gain difference). The critic network receives a state and action as input and estimates a q-value. This q-value is then utilized in the soft actor-critic training process [2].

Training Settings. We use a machine equipped with a Ryzen 5950x CPU and NVidia 3080Ti GPU for the agent training. We use Adam optimizer [5] with an initial learning rate of $3 \cdot 10^{-4}$. The agent is trained with a batch size of 256 for 500k timesteps in a light-controlled darkroom environment. We set a maximum exposure value of 100 ms and a maximum gain of 40 dB for the control bound of the machine vision camera. In the darkroom environment with controlled LED lighting, the agent stores various exposure transition sets in the replay buffer for each episode. The actor and critic networks are optimized using the transition batches sampled from the buffer. The maximum episode length is set to 200 steps. It usually takes about 20 seconds per episode, including training and image acquisition time. Also, the agent is validated every 2000 time steps. The total training time usually takes 18 hours in these training conditions. For the other hyperparameters, we follow the default setting described in [2] and summarize them in Tab. 2.

Table 1. **Network architectures details.**

Layer Type	Actor	Critic
Linear	(state_dim, 256)	(state_dim + action_dim , 256)
Activation	ReLU	ReLU
Linear	(256, 256)	(256, 256)
Activation	ReLU	ReLU
Linear	(256, 256)	(256, 256)
Activation	ReLU	ReLU
Linear	(256, 2·action_dim)	(256, 1)

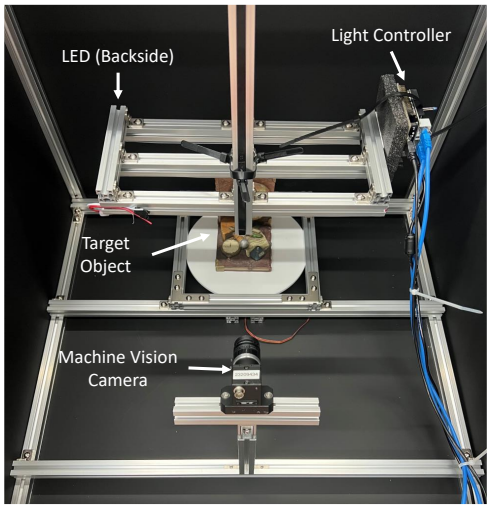


Figure 1. **Light-controlled darkroom.**

Light-controlled Darkroom. We build the darkroom environment to freely control the lighting. Our aim is to provide a various range of lighting conditions to the RL agent, within a short training time compared to the real sunlight condition. The darkroom environment is made with aluminum profiles and black acrylic plates of 5 mm thickness. Fig. 1 and Fig. 2 shows the constructed environment and each component's specification. We use a global shutter machine vision camera from Teledyne FLIR, which has a 3.2MP Sony IMX 265. For the light controller unit, we built a program based on the STM32F446RE Necleo board, which has a 180MHz ARM Cortex M4 CPU and a flash memory of 512 kbytes. Our LED bar is based on the WS2812B LEDs, which have 144 LEDs per meter. We use two LED bars for the darkroom environment. The light controller communicates with the RL gym environment located on the RL server through serial communication.

Table 2. **Hyperparamters for agent training.**

Parameter	Value
optimizer	Adam [5]
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.99
replay buffer size	10^6
target smoothing coefficient (τ)	0.05
batch size	256
initial random steps	10000

Machine Vision Camera

- Flir Blackfly S, BFS-U3-31S4C-C
- 2048 x 1536, 55fps
- 3.2MP, Sony IMX 265
- Global Shutter

Light Controller

- STM32F446RE Nucleo
- ARM Cortex M4, 180MHz CPU
- 512 kbytes Flash memory
- Direct Memory Access (DMA)

Controllable LED

- WS2812B Fully addressable LED
- 144LEDs/m (288 LEDS installed)
- 5V, 3A (15W for 144 LEDs)
- 800kbps data transfer

Figure 2. **Hardware specification used in the darkroom.**

3. Design Philosophy for RL Component

In this section, we describe the underlying philosophy of designing the RL components for the exposure control task.

3.1. State Design

CNN Feature. Perhaps, a naïve state designing is utilizing the CNN model to extract a feature map from the image and use the feature map as a state. However, CNN-based state design has three disadvantages. First, the extracted feature map has no clear relationship with the camera exposure level. Usually, CNN backbones (*e.g.*, ImageNet, VGG, ResNet) are trained in a brightness-agnostic manner via their data augmentation strategy. Therefore, the extracted feature usually includes semantic information rather than brightness information. Second, CNN brings additional domain gap problems in real-world inference. Third, the state extraction with the CNN model is computationally heavy, introduces additional learnable parameters, and requires lots of system memory to store the image states in the replay buffer. As a result, CNN state brings undesirable properties for deep reinforcement learning, such as reducing replay buffer size, limiting sample diversity, increasing training time, generalization problems, and unclear representation of brightness.

Intensity Value. Therefore, instead of using the CNN model, we utilize the averaged intensity values along the x-axis. The primary purpose of auto-exposure control is to ensure a proper image brightness level by adjusting camera exposure settings. Therefore, image intensity is the primary cue and a straightforward and effective representation for auto-exposure control. Also, we reduce the dimension by averaging intensity value along the x-axis rather than utilizing entire images to ensure high-level memory efficiency with minimum computation burden. Lastly, we stack 1-dimensional averaged intensity values of 3 frames and define the stacked intensity values as a *state*. We empirically found that frame stacking has a significant impact on making better decisions. By stacking consecutive frames, the agent is able to implicitly observe the lighting condition change and camera exposure change over a short period.

3.2. Action Design

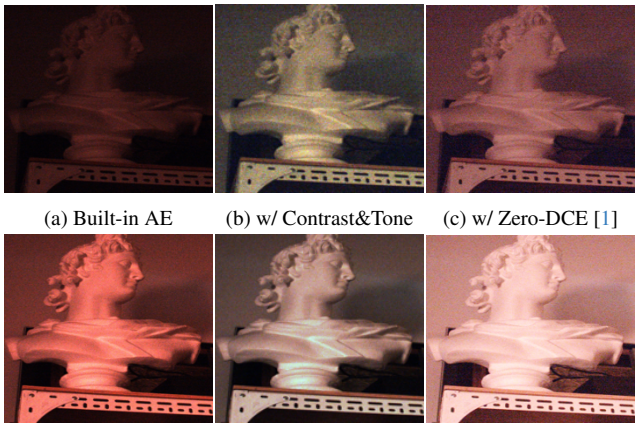
Discrete *vs.* Continuous Action Space. We initially considered designing the system to obtain quantized exposure and gain values using a discrete action space. However, due to the sensitive nature of the parameter optimization, which often causes oscillation, it was difficult to fully account for the changes even with a higher level of quantization. Therefore, we formulate the auto exposure control problem as a continuous action control task.

Absolute *vs.* Relative Action Range. We can consider absolute and relative actions as output values. The former indicates the agent estimate desired absolute action values (*i.e.*, absolute values of exposure time and gain, such as 10 ms, 4dB). The latter means the output is a relative difference in action values (*i.e.*, $\pm 10ms$, $\pm 2dB$) Relative control is more stable but has the disadvantage of slower convergence. On the other hand, absolute control can reach the desired value in one step but is more likely to be unstable. In practice, we found that absolute control faced difficulties in learning good policies. Therefore, we make the agent estimate relative action for the exposure control task.

3.3. Reward Design

Designing reward functions is crucial in deep reinforcement learning, as it determines the desired objective of the learning process. In this context, we will briefly mention the rationale behind the three reward functions presented in the main text. A common desired goal for auto exposure control is to acquire a high-quality image that has moderate brightness, low-level noise, and sharp edge information. Also, the convergence of the control process should be fast but stable.

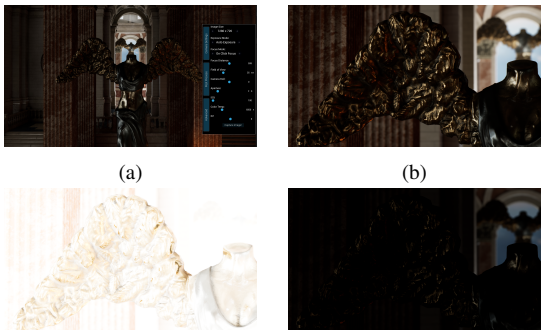
Therefore, the proposed reward functions are designed for these desired objectives. First, the mean reward term $\mathcal{R}_{mean}$ helps to ensure that the image has a median brightness. Instead of designing it linearly, we made it decrease more steeply around the median by adding non-linearity with p_m to maintain the center. The second flickering term $\mathcal{R}_{flk}$ suppresses the image flickering effect caused by the action's vibration and ensures smooth exposure transition while preserving image attributes. Lastly, the noise term $\mathcal{R}_{noise}$ reduces the overall image noise caused by excessively high gain, encouraging a balanced control between the exposure and gain parameters.



(a) Built-in AE (b) w/ Contrast&Tone (c) w/ Zero-DCE [1]

(d) Ours (e) w/ Contrast&Tone (f) w/ Zero-DCE [1]

Figure 3. **Impact of AE control on post-processing methods.**



(a) (b) (c) (d)

Figure 4. **Example image from Unreal-based camera simulator [6].** (a) Interface overview, (b) Auto-Exposure, (c) Over-exposed, (d) Under-exposed.

4. Discussion and Future Work

Camera AE vs. ISP. There are two main processes for camera image processing: Automatic Exposure (AE) control and Image Signal Processing (ISP). AE control includes automatic gain, exposure, and aperture control as well. The roles of AE control and ISP are quite different. The former aims to get high-quality images while rapidly adjusting exposure levels within hardware limitations. After that, the latter enhances the quality of the acquired image for its purpose by using various ISP tools, such as demosaicking, deblurring, denoising, color space correction, gamma correction, tone-mapping, HDR, and more. Therefore, the AE control (hardware level) and ISP (software level) are complementary rather than competitive relations. As the former stage obtains a better quality image, the quality in the later stage improves. We evaluate our method by combining conventional contrast enhancement & tone mapping method (*i.e.*, photoshop) (e) and Zero-DCE [1] (f). As shown in Fig. 3, the exposure control results highly affect its final outputs ((b)vs(e), (c)vs(f)).

Sim2Real via Camera Simulator. From the perspective of object, motion, and lighting diversity, a simulated camera model could be beneficial to learning camera exposure control with deep reinforcement learning. Modern photorealistic simulations, such as Unreal, Unity, and Blender, provide similar quality images to the real world and support partial functionality for auto-exposure control.

However, introducing simulation causes two domain gap issues: the domain gap 1) between actual and simulated environments and 2) between simulated and real camera acquisition models. The former issue leads to large performance differences between the models trained on simulated data and real data, as we can see in domain adaptation literature. For the latter issue, the simulated camera model provides an *incomplete* image acquisition model. As shown in Fig. 4-(c), when the camera captures the image with high gain or ISO, the image must contain severe noise within the image. However, the simulation doesn't support this functionality. Therefore, due to these issues, we decided to utilize the darkroom environment to investigate the possibility of deep reinforcement learning for automatic exposure control.

Future Work: DRL-AE in Simulation. Although the simulation has some disadvantages, it has many advantages, such as faster interaction speed, easy parallelization, and diverse controllable parameters. Therefore, in future work, we also plan to study Sim2Real based camera exposure control and compare the Sim2Real model with the real-world model trained with this paper's method.

Future Work: Motion-aware AE Control. Considering the motion blur is another future direction. As the proposed darkroom environment has only a fixed target object, it is difficult to consider motion blur that frequently happens in the real world. In future work, we plan to extend the current darkroom environment to make object motion, thus allowing the agent to consider a motion blur for their exposure parameter control.

Future Work: Various Reward Functions. In this paper, the proposed reward design might be a primitive and basic form for camera exposure control. However, it can be easily extendable by incorporating modern image assessment metrics [3, 4, 7]. Also, we can utilize human preference, network inference results (*e.g.* detection confidence), and the number of detected features as a reward function.

Future Work: Aperture Control. The machine vision camera used in the experiment has a fixed aperture size, which is not controllable with software. However, the aperture is also one parameter that affects camera exposure level and depth of field. Therefore, we plan to control aperture size by using a mechanic aperture control module.

References

[1] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1780–1789, 2020. 3

[2] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 1

[3] Bin Han, Yicheng Lin, Yan Dong, Hao Wang, Tao Zhang, and Chengyuan Liang. Camera attributes control for visual odometry with motion blur awareness. *IEEE/ASME Transactions on Mechatronics*, 2023. 3

[4] Joowan Kim, Younggun Cho, and Ayoung Kim. Proactive camera attribute control using bayesian optimization for illumination-resilient visual navigation. *IEEE Transactions on Robotics*, 36(4):1256–1271, 2020. 3

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[6] Unreal Cast Ltd. Arch camera system. https://www.unrealengine.com/marketplace/en-US/product/arch-cam-system-lite, 2023. [Online]. 3

[7] Ukcheol Shin, Jinsun Park, Gyumin Shim, Francois Rameau, and In So Kweon. Camera exposure control for robust robot vision with noise-aware image quality assessment. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1165–1172. IEEE, 2019. 3