Autofocus by Deep Reinforcement Learning

Chin-Cheng Chan and Homer H. Chen; National Taiwan University; Taipei, Taiwan

Abstract

In recent years, smartphones have become the primary device for day-to-day photography. Therefore, it is critical for mobile imaging to capture sharp images automatically without human intervention. In this paper, we formulate autofocus as a decision-making process, in which the travel distance of a lens is determined from the phase data obtained from the phase sensors of a smartphone, and the decision-making policy is based on reinforcement learning, a popular technique in the field of deep learning. We propose to use a noise-tolerant reward function to combat the noise of the phase data. In addition, instead of using only the current phase data, each lens movement is determined using the phase data acquired along the journey of an autofocus process. As a result, the proposed machine-learning approach is able to expedite the autofocus process as well. Experimental results show that the method indeed improves the autofocus speed.

1. Introduction

Phase detection autofocus (PDAF) involves special sensors normally classified as left and right pixels [2], [3]. When these pixels are assembled together, all the left pixels form the left image, and all the right pixels form the right image. The phase shift between the left and right images can be calculated by phase correlation [4], [5], as suggested by Chan et al. [6]. The sign of the phase shift reflects the position of the object plane with respect to the focal plane, and the magnitude of the phase shift reflects the distance between the two planes. Ideally, there is a one-to-one correspondence between phase shift and the lens travel distance. However, due to some practical factors like image noise and sparsity of phase sensors, the measured phase shift is usually noisy. Fig. 1 shows the distribution of 80 phase shift profiles (each expressed as a function of lens position). We can see that the phase shift is noisier as the distance of the lens to the in-focus position increases and it almost reaches saturation when the distance of the lens to the in-focus position is larger than 300.

Chan et al. [1] proposed a statistical approach to determine lens movement from noisy phase shift. They modeled the distribution of phase shift given the distance of the lens to the in-focus position as a Gaussian distribution and employed the Bayes' theorem to determine the lens movement. Even though this statistical approach improves the reliability of lens movement, it uses the same strategy for making the lens movement in each step, assuming that autofocus is a stationary process. However, this assumption is not always true. In each process, the lens may start from any position. However, after a few lens movements, the lens gets closer to the in-focus position. Since the distance of the lens to the in-focus position changes, different strategies for determining the lens movement should be adopted in different steps of the autofocus process. This motivates us to explore the use of machine learning for autofocus.

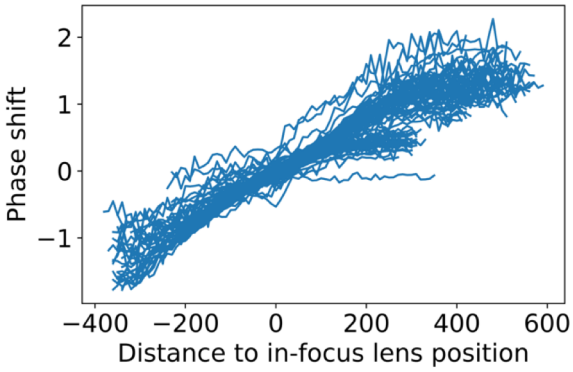


Fig. 1. The phase shift profiles corresponding to 80 different scenes with various in-focus lens positions.

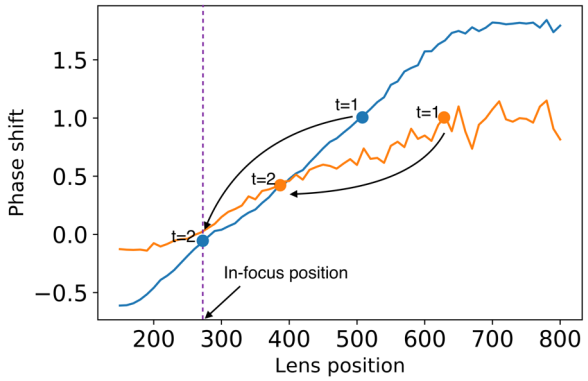


Fig. 2. Example lens movements for two different scenes. The blue and yellow curves represent the phase shift profiles. Solid dots indicate the lens positions, indexed by the temporal order.

Reinforcement learning (RL) is a general framework for learning a decision-making process, so it is possible to model autofocus as an RL problem if the determination of lens movement is considered a decision-making process of an agent. Specifically, the phase shift and the lens travel distance can be modeled as the observation and the action, respectively, and the mapping from phase shift to travel distance (called the policy) can be learned by RL. However, the design of the reward, which serves as a feedback signal for the learning algorithm to update the policy during the training time, is a challenging task in this approach. Unlike many other problems studied in the reinforcement learning field [7], [8], [9], no reward function for autofocus has been defined in the literature. The reward function has to work for the whole autofocus process. However, due to the fact that phase shift is noisy, erroneous estimate of lens movement is inevitable. Consider the example shown in Fig. 2. The two phase shift profiles have the

same in-focus position but different shapes. Suppose that, for the two scenes, we start with two different initial lens positions that have the same phase shift. The agent takes the same lens movement for both cases because the phase shift is the same, but it ends up in two different states after the lens movement. In one case, the lens reaches the in-focus position, whereas the lens is still far from the in-focus position in the other case. However, the agent is not responsible for making a mistake in the latter case because the same lens movement does lead to a correct result in the other case. Therefore, the reward function needs to tolerate the noise of the phase data.

In this paper, we use a noise-tolerant reward to combat the noise of phase data and use an recurrent neural network (RNN) [20] agent to learn the whole autofocus process. Even though deep learning has been used to assess the focus quality of microscopic images [10], to the best of our knowledge, it has not been used to learn a policy for autofocus, especially when the data are noisy. In addition, another advantage of the proposed method is that it can learn the autofocus process directly from phase data, and thus avoids human-designed rules for determining lens movements. This paper describes our current efforts toward applying deep learning to autofocus.

2. Reinforcement Learning

Reinforcement learning, usually modeled as a Markov decision process (MDP) [11], involves a series of interactions between an environment and an agent. Specifically, the agent first observes an initial state s_0 and makes an action a_0 according to a policy π. The policy can be stochastic or deterministic. In the stochastic case, the policy of the agent defines the distribution from which the action is drawn given the state. That is, for an arbitrary time step t, a stochastic policy can be described by:

$$\pi(a_t|s_t) = \mathrm{P}[A_t = a_t|S_t = s_t], \tag{1}$$

where A_t and S_t are random variables representing the action and the state, respectively, at time step t. In the deterministic case, the policy defines the mapping from state to action:

$$\pi: s_t \rightarrow a_t. \tag{2}$$

Then the environment gives the agent a reward r_0 representing how good the action is. This procedure is repeated until the agent reaches the final state s_T (here we only consider episodic environments, in which the final state always exists). The interactions between the environment and the agent from the initial state to the final state are called an episode. The goal is to learn a policy that maximizes the total reward received in an episode.

The learning of the policy involves two alternating phases: the simulation phase and the update phase. In the simulation phase, the agent uses the current policy to interact with an environment for several episodes. The agent also records the states it has observed, the actions it has taken, and the rewards it has received during the simulation process (all the recorded states, actions, and rewards are collectively called the experiences). Then in the update phase, the policy is updated using the experiences collected in the simulation phase using algorithms like policy gradient [12]. Specifically, in policy gradient, the policy is approximated by a function parameterized by θ, and for an arbitrary time step t in an episode, θ is updated by:

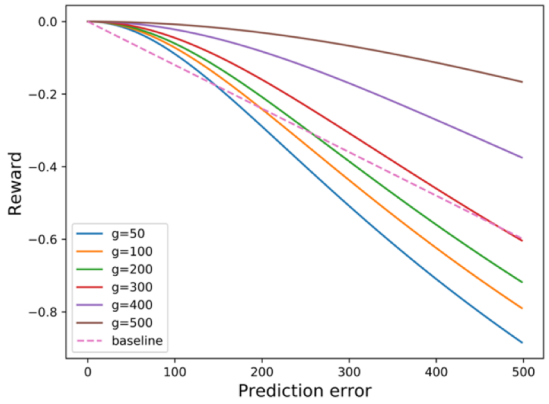


Fig. 3. The proposed reward function. Solid curves represent the proposed reward function for different distances to the in-focus position. The dashed line is the baseline reward function, which has no tolerance for prediction error.

$$\Delta\theta = \nabla_\theta \log \pi(a_t|s_t)\, v_t\,, \tag{3}$$

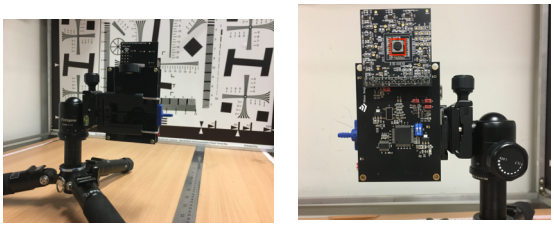
where $v_t = \mathrm{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots |S_t = s_t]$ is the cumulative future reward given the current state, R_{t+1} is a random variable representing the reward at time step $t+1$, and $\gamma \in [0,1]$ is a discount factor. The cumulative future reward can be estimated directly from the experiences as in REINFORCE [13].

In recent years, with the breakthrough in deep learning, the combination of reinforcement learning and deep neural networks led to great performance improvements [7], [14, 15]. Lillicrap et al. [16] proposed the deep deterministic policy gradient algorithm, which is based on the actor-critic algorithm [17]. In this approach, the actor is a neural network that takes the state as input and outputs the action. That is, the actor represents the policy of the agent. The critic is a feed-forward neural network that takes the state and an action as inputs and outputs the cumulative future reward. That is, the cumulative future reward is not estimated from experiences as in REINFORCE [13]; it is estimated by the critic. The critic can be used to update the actor so as to maximize the cumulative future reward during the training time by:

$$\Delta\theta^\pi = \frac{1}{N}\sum_{i=1}^{N} \nabla_a Q(s_i, a|\theta^Q) \cdot \nabla_{\theta^\pi} \pi(s_i|\theta^\pi)\Big|_{a=\pi(s_i|\theta^\pi)}, \tag{4}$$

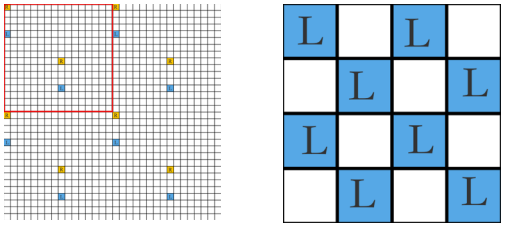
where $\pi(s_i|\theta^\pi)$ represents the actor parameterized by θ^π, $Q(s_i, a|\theta^Q)$ represents the critic parameterized by θ^Q, and N represents the total number of time steps in the experiences. The critic can be trained by minimizing the temporal-difference error [18].

All the previously mentioned methods assume that the environment state can be directly observed by the agent, but this assumption does not hold in many real-world applications due to practical factors like sensory noise. In such case, a single observation o_t of the environment cannot fully reflect the environment state s_t; more than one observation are needed. Heess et al. [19] solved the problem by replacing the feed-forward neural networks with RNNs. Since there are memory units in RNNs,



(a) (b)

Fig. 4. (a) The experimental setup. (b) The PDAF development platform. The red box indicates the camera module.



(a) (b)

Fig. 5. (a) A portion of the image sensor. The blue and the yellow boxes represent the locations of the left and right pixels, respectively. The red rectangle represents the basic pixel pattern that repeats over the sensor. (b) An illustration of an assembled left image.

information from previous observations may be preserved and utilized in a future step. Let T denote the length of each episode, and N the number of episodes in the experiences. The parameters of the RNN actor can be updated by:

$$\Delta\theta^{\pi} = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\nabla_a Q\left(h_t^i, a\middle|\theta^Q\right)\nabla_{\theta^{\pi}}\pi\left(h_t^i\middle|\theta^{\pi}\right)\Big|_{a=\pi\left(h_t^i|\theta^{\pi}\right)}, \quad (5)$$

where $h_t^i = (o_0^i, o_1^i \ldots, o_{t-1}^i, o_t^i)$ represents the series of observations up to time step t in the i^{th}episode. For RNN critic, the parameters are learned by minimizing the temporal-difference error [18]. Even though this approach achieves better performance than previous algorithms, it neglects the conflict between rewards when observations are noisy.

3. Proposed Method

We formulate autofocus as a decision-making process that involves the interactions between an agent and an environment. At time step t, the agent receives an observation o_t which consists of the current phase shift and the travel distance at the last time step. Then the agent uses a deterministic policy to make a lens movement a_t and then receives a reward r_t from the environment. This procedure is repeated until the lens reaches the in-focus position or the number of lens movements exceeds a pre-defined number T. In our experiment, we set T to 5.

As mentioned previously, conflict between rewards occurs when the phase data is noisy. We resolve the conflict by designing a reward function that tolerates prediction errors. That is, the agent receives almost the same reward for predictions with errors lower than a pre-defined value. Let g denote the ground-truth distance of the lens to the in-focus position, and e the absolute difference between the ground truth distance and the predicted distance (called the prediction error). The proposed reward function $R(e, g)$ is expressed by

$$R(e,g) = \frac{-1}{2}\ln\left(1 + \frac{e^2}{10\cdot b(g)^2}\right), \quad (6)$$

where $b(\cdot)$ determines the tolerance of the reward function to the prediction error: the higher it is, the higher the tolerance. Note that b is a function of the ground truth distance of the lens to the in-focus position instead of a constant because we need different degrees of tolerance for different distances of the lens to the in-focus position. When the distance is larger, the phase data is usually noisier as shown in Fig. 1, so we need more tolerance in this case. In contrast, we want less tolerance for prediction error when the lens is close to the in-focus position, where the phase data are less noisy and a subtle lens movement results in a huge change in the image sharpness. To determine $b(g)$, we note that the phase shift almost saturates when the distance of the lens to the in-focus position is larger than 300 as shown in Fig. 1. Therefore, when the distances of the lens to the in-focus position are 500 and 400, estimation errors around 200 and 100, respectively, are acceptable. In addition, it is common that the phase shift observed at the in-focus position is not 0, as shown in Fig. 1, so we need to tolerate a small amount of prediction error even if the lens is at the in-focus position. Empirically, we set the tolerance of prediction error for g = 0, 150, 300, 400, 500 to be 40, 55, 70, 150, 250, respectively and use these five points to fit a third degree polynomial for $b(g)$,

$$b(g) = 3\times10^{-6}|g|^3 - 8\times10^{-4}|g|^2 + 0.12|g| + 41. \quad (7)$$

The reward function used in our experiment is plotted in Fig. 3.

The autofocus agent consists of two neural networks: the actor network and the critic network. The actor network determines the lens movement from phase shift, and the critic network evaluates the cumulative future reward given the current state. To learn the whole autofocus process, we use RNN actor and critic to utilize information obtained from previous steps. Specifically, the actor is an RNN that takes the history and the current observation as input and outputs a lens movement. The critic is an RNN that takes the history, the current observation, and an action as inputs and outputs the estimated cumulative future reward resulted from taking the action under the current state. We train the agent using the recurrent deterministic policy gradient algorithm proposed by Heess et al. [19]

4. Experimental Results

The experimental setup and the PDAF development platform used in our experiments are shown in Fig. 4. We uniformly divided the range of lens positions into 600 units. The sensor size of camera is 3280×2464 pixels, and both the left and right images are of size 410×154 pixels. A portion of the image sensor and the distribution of the left and right pixels are shown in Fig. 5(a). Note that left pixels on the odd-numbered columns of the assembled left image shown in Fig. 5(b) are not vertically aligned with those on the even-numbered columns, the same for the assembled right image.

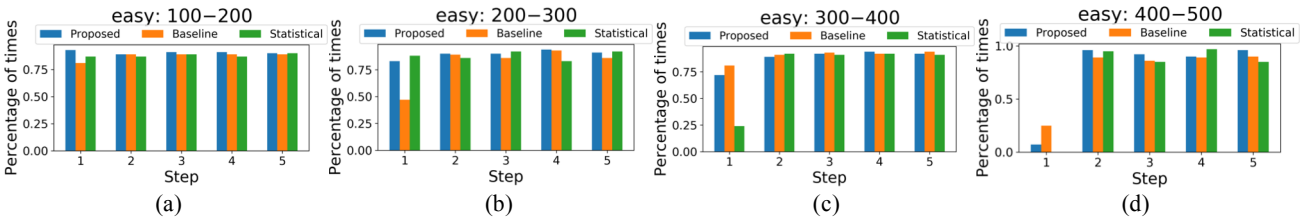


Fig. 6. Performance of RL with noise-tolerant reward (blue), RL with baseline reward (orange), and [1] (green) in easy cases. (a)–(d) Results when the initial distance of the lens to the in-focus position is in the range 100–200, 200–300, 300–400, and 400–500, respectively.

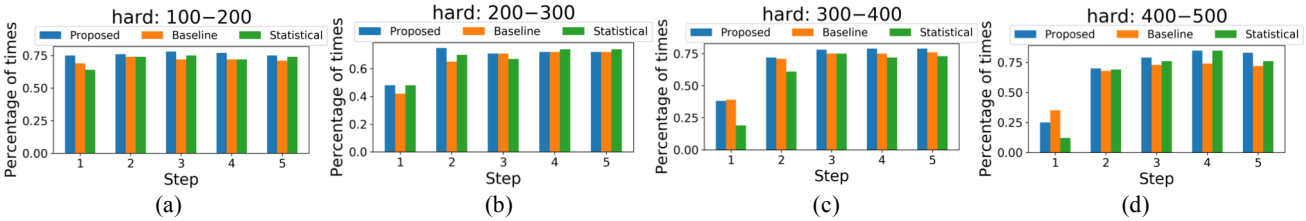


Fig. 7. Performance of RL with noise-tolerant reward (blue), RL with baseline reward (orange), and [1] (green) in hard cases. (a)–(d) Results when the initial distance of the lens to the in-focus position is in the range 100–200, 200–300, 300–400, and 400–500, respectively.

Therefore, we further divided the left (right) image into two sub-images—one consists left (right) pixels on odd-numbered columns and the other of left (right) pixels on even-numbered columns—and calculated the phase shift for both types of sub-images. Then we took the average of phase shifts calculated from the sub-images as the phase shift between the left and right images.

We trained the autofocus agent with 70 scenes and tested it on 50 scenes, all with various in-focus lens positions. We set the discount factor γ to 0.6 since this setting yields the best performance according to our experiences. The RNN architectures for both the actor and critic are long short-term memory cells [20] with two hidden layers: the first layer has size 400 and the second has size 300. We used the Adam optimizer with learning rate 10^{-3} for the critic and 10^{-4} for the actor. We set the batch size to 128 and trained our model for 15,000 iterations.

In order to evaluate the performance of the proposed autofocus agent in different autofocus scenarios, we divided our testing set into two categories (one easy and the other hard) based on the height and the fluctuation of the phase shift profile. We defined the height of a phase shift profile as the difference between the average phase shift of the last five images in the focal stack and that of the first five images. To define the fluctuation of a phase shift profile, we noted that the whole phase shift profile is not linear, but a local window of the profile may be treated as linear. Thus, we fit a linear function for each local window of the profile and chose the maximum of all the fitting lines' residual as the fluctuation of the whole profile. Then we empirically chose thresholds for both criteria. In our experiments, a phase shift profile was classified as the easy case if the fluctuation is less than 0.2 and the height is larger than 1.0. Otherwise, it falls into the hard case.

We evaluate the performance of an autofocus algorithm by counting the percentage of times the algorithm moves the lens to the hillsides of the focus profile as proposed in [1]. (A focus profile shows the image contrast of a scene at each lens position and usually has a hill-like shape with the peak at the in-focus position.) We compare the performance of three methods: 1. the proposed RL approach with noise-tolerance reward function, 2. the proposed RL approach with baseline reward function, and 3. our previous approach based on Bayesian statistics [1]. The performance of the three methods in both easy and hard cases are compared in Figs. 6 and 7, respectively.

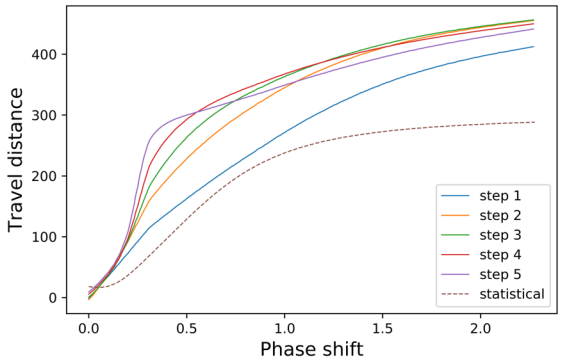


Fig. 8. Policies of the proposed method (solid curves) and the statistical approach (dashed curve). Since the proposed method uses an RNN actor, different lens movements may be taken for the same phase shift at different lens movements.

We can see that, both RL-based methods have better performance than the statistical approach in both easy and hard cases. Especially when the distance of the lens to the in-focus position is very large, 300–400 for example, RL-based approaches are much more likely to bring the lens to hillsides of the focus profile in the first movement. Thus, RL-based methods are faster than the statistical approach. Comparing the performance of the agents trained with noise-tolerant reward function and the baseline reward function, we can see that the latter is a little faster than the

former in the first step, but the former has a higher chance to bring the lens to the hillsides at the end of the fifth movement. In general, noise-tolerant reward function results in a better autofocus agent than the baseline reward function since success rate is usually more important than speed for autofocus.

Finally, the policy of the proposed method is visualized by calculating the average lens movement the agent makes for different phase shifts at different steps of an autofocus process. Fig. 8 shows the policies of the proposed method and the statistical approach [1]. We can see that the proposed method is more conservative in the first movement than in the subsequent movements. This behavior is reasonable because we only have a single observation of phase shift, from which the distance of the lens to the in-focus position may not be accurately determined. Therefore, the agent takes a conservative step. On the other hand, if there is a large phase shift after the first movement, it is very likely that the initial distance of the lens to the in-focus position is very large, so the agent makes a more aggressive movement. Comparing the policy of the proposed method and that of the statistical approach, we can see that the proposed method is more aggressive than the statistical approach for large phase shifts. This is the reason why the proposed method is faster than the statistical approach when the distance of the lens to the in-focus position is large.

5. Conclusion

In this paper, we have described a reinforcement learning approach to phase detection autofocus. It combats the noise of phase data by building noise tolerance into the reward function. Experimental results show that the agent trained with the noise-tolerant reward function has a higher chance to bring the lens to the hillsides of the focus profile than the one trained with a regular reward function when the phase data is noisy. Compared with our previous statistical approach [1], which only uses the current phase shift to determine the lens movement, the proposed method is able to exploit the phase data collected from the past. The policy learned by our method leads to fast autofocus and also provides some insights into autofocus policy design.

6. References

[1] C.-C. Chan and H. H. Chen, "Improving the Reliability of Phase Detection Autofocus," Electronic Imaging, vol. 2018, no. 5, pp. 1-5, 2018.

[2] M. Hamada, "Imaging device including phase detection pixels arranged to perform capturing and to detect phase difference," US20130088621, 2013.

[3] M. Levoy. (2010). Autofocus: phase detection. Available: http://courses.cs.washington.edu/courses/cse131/13sp/applets/autofocusPD.html

[4] H. Foroosh, J. B. Zerubia, and M. Berthod, "Extension of phase correlation to subpixel registration," IEEE transactions on image processing, vol. 11, no. 3, pp. 188-200, 2002.

[5] Q. Tian and M. N. Huhns, "Algorithms for subpixel registration," Computer Vision, Graphics, and Image Processing, vol. 35, no. 2, pp. 220-233, 1986.

[6] C.-C. Chan, S.-K. Huang, and H. H. Chen, "Enhancement of phase detection for autofocus," in Image Processing (ICIP), 2017 IEEE International Conference on, 2017, pp. 41-45: IEEE.

[7] V. Mnih et al., "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[8] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," Journal of Artificial Intelligence Research, vol. 47, pp. 253-279, 2013.

[9] G. Brockman et al., "Openai gym," arXiv preprint arXiv:1606.01540, 2016.

[10] S. J. Yang et al., "Assessing microscope image focus quality with deep learning," BMC bioinformatics, vol. 19, no. 1, p. 77, 2018.

[11] M. L. Puterman, Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

[12] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in Advances in neural information processing systems, 2000, pp. 1057-1063.

[13] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Machine learning, vol. 8, no. 3-4, pp. 229-256, 1992.

[14] D. Silver et al., "Mastering the game of go without human knowledge," Nature, vol. 550, no. 7676, p. 354, 2017.

[15] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," nature, vol. 529, no. 7587, p. 484, 2016.

[16] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," arXiv preprint arXiv:1509.02971, 2015.

[17] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in Advances in neural information processing systems, 2000, pp. 1008-1014.

[18] R. S. Sutton, "Learning to predict by the methods of temporal differences," Machine learning, vol. 3, no. 1, pp. 9-44, 1988.

[19] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," arXiv preprint arXiv:1512.04455, 2015.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

7. Author Biography

Chin-Cheng Chan received his bachelor's degree in electrical engineering from National Taiwan University in 2018. His research interests are in image processing with a focus on computational imaging, and his research topics include robust principal component analysis, autofocus for smartphone cameras, and deep learning for medical image analysis. He was also a summer intern at MediaTek, Taiwan in 2016.

Homer H. Chen is an IEEE Fellow. His professional career has spanned industry and academia. Since August 2003, he has been with the College of Electrical Engineering and Computer Science, National Taiwan University, where he is Distinguished Professor. Prior to that, he held various R&D management and engineering positions with U.S. companies, including AT&T Bell Labs and Rockwell Science Center, over a period of 17 years.