

## RESEARCH ARTICLE

# An Adaptive Threshold for the Canny Edge With Actor-Critic Algorithm

KEONG-HUN CHOI<sup>1</sup> AND JONG-EUN HA<sup>2</sup><sup>1</sup>Graduate School of Automotive Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea<sup>2</sup>Department of Mechanical and Automotive Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea

Corresponding author: Jong-Eun Ha (jeha@seoultech.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2020R1A2C1013335).


**ABSTRACT** We propose a method to automatically select proper values of three thresholds in the Canny edge algorithm. Edge detection is widely used for object recognition, detection, and segmentation. Due to its good performance, the Canny edge algorithm is still widely used among many edge detection algorithms. But, it requires manually selecting three appropriate thresholds for the given image. Some approaches have been proposed for automatically setting thresholds in the Canny edge algorithm. But, they either deal with partial among three entries or only show their performance in a limited range of variation. In natural scenes, images are acquired under various illumination, pose, and weather conditions. This paper proposes a method that can operate in various environments. We formulate the given problem by adopting an actor-critic algorithm. We propose an actor and critic network to solve the problem with an actor-critic algorithm. Also, we suggest a reward configuration based on an edge evaluation network and measure to prevent the reversal between high and low thresholds. The edge evaluation network uses an original image and an edge image as input. We set a negative reward when reversing the high and low thresholds occur. The proposed algorithm can adapt to unseen environments using images without requiring ground truth labels. Experimental results using diverse datasets show the feasibility of the proposed algorithm.

**INDEX TERMS** Actor-critic algorithm, edge detection, deep reinforcement learning, deep learning.

## I. INTRODUCTION

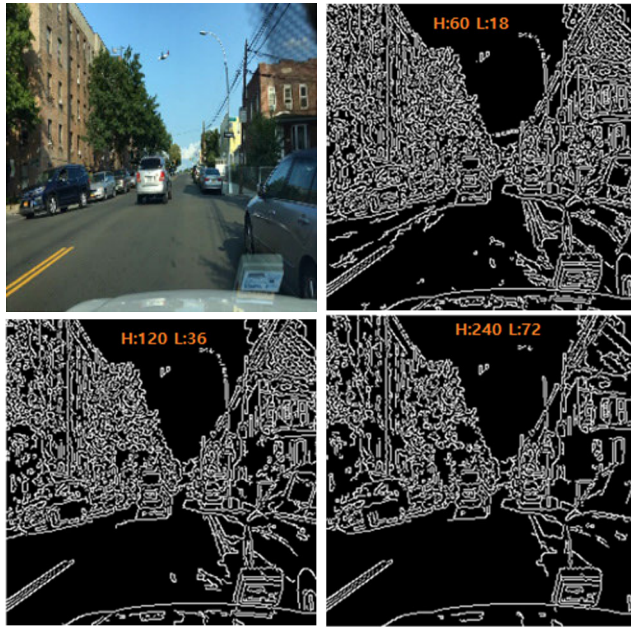
Edge information on images is helpful for object detection, image segmentation, motion analysis, and 3D reconstruction [1], [2], [3], [4]. Most traditional edge detection algorithms are filter-based and use statistical analysis. Recently deep learning has also been applied to detecting edges on images. The Canny algorithm proposed three decades ago is still widely used due to its good performance [5]. The user must set three parameters related to the smoothing window size and two thresholds in the hysteresis process. We obtain very different edge images according to values of three parameters, as shown in Figure 1.

In our previous work [6], we proposed an algorithm that automatically determines values of two thresholds in the

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

Canny algorithm using the Deep Q-Network (DQN) [7], [8]. We used a fixed value for a parameter related to the smoothing window size. In this paper, we propose an algorithm that can automatically determine the three parameters in the Canny algorithm. DQN is suitable for discrete types of actions. Extending our previous algorithm by adding additional parameters related to the smoothing window can cause a problem due to many actions. Therefore, this paper copes with this problem by adopting an actor-critic algorithm [9], [10], [11] with continuous action. Also, we modify the edge evaluation network used in our previous work [6] to use both an original image and a corresponding edge image as input for the improved evaluation of edge quality.

Also, the supervised learning-based method can be a candidate to solve the given problem. It might have a network structure in which an original image is used as input, and a corresponding optimal edge image is used as output.



**FIGURE 1.** The comparison of edge images according to different threshold values in the Canny algorithm.

Preparing the correct edge image per input image for training is necessary, which requires much time. In general, it is known that supervised learning works well in environments similar to the training, but it gives a poor performance in environments different from training. One way to adapt to a new environment is to retrain using an image of the corresponding domain. Also, this requires the preparation of ground truth labels. The proposed algorithm uses an edge evaluation network trained in a supervised manner. Still, we show that performance improvement is possible through additional training using only images from a new environment without updating the edge evaluation network.

The contributions of the proposed algorithm are as follows.

- (1) We present a method for automatically selecting all three threshold values that guarantee an excellent edge image quality with the Canny algorithm. We solve the problem caused by large combinations of three threshold values by estimating continuous actions based on an actor-critic algorithm.
- (2) We present a reward model for stable training of action and critic networks in the actor-critic algorithm. We configure the reward model by reflecting two terms. One is the output of the edge evaluation network. The other is a constraint for preventing the reversal of high and low thresholds. The edge evaluation network uses an original image and the edge image as inputs and produces a value for the quality of the edge.
- (3) In unseen environments, the proposed algorithm can improve performance by only using images without requiring ground truth labels.

The paper is organized as follows. Section II deals with related works. Section III shows the proposed method.

Experimental results are presented in section IV and finally conclusion is section V.

## II. RELATED WORKS

Edge detection algorithms can be classified into filter-based, learning-based, and recent deep learning-based algorithms.

### A. FILTER-BASED ALGORITHMS

Filter-based algorithms [3], [5] find edges by investigating dramatic changes in intensity, color, texture, etc. Learning-based algorithms find edges by leveraging hand-crafted features. Statistical Edges [12], Pb [13], and gPb [14] use features found by manual design using information theory. Early learning-based methods such as BEL [15], Multi-scale [16], Sketch Tokens [17], and Structured Edges [18] also heavily rely on manually designed features. Dollar et al. [18] detect structured edges by joint learning ground truth clustering and mapping image patches to clustered tokens. It showed state-of-the-art performance on the BSDS500 dataset until the advancement of deep learning-based algorithms. Learning-based algorithms can automatically generate edge images by using structured information on images. But, their applicability is shown using only a small number of images compared to the large number of images deployed in deep learning.

### B. DEEP LEARNING-BASED ALGORITHMS

Recent deep learning-based algorithms use features generated by convolutional neural networks (CNN) [19]. Bertasius et al. [20] use CNN to find features of candidate contour points. Xie et al. [21] propose holistically-nested edge detection (HED) that integrates the outputs from different intermediate layers with skip connections. Xu et al. [22] use a hierarchical model to find multi-scale features fused by a gated conditional random field. He et al. [23] propose a Bi-Directional Cascade Network (BDCN) structure to detect edges at different scales. They train the network using corresponding labeled edges for each scale. They introduced the Scale Enhancement Module (SEM), which utilizes dilated convolution to generate multi-scale features instead of using deeper CNNs or explicitly fusing multi-scale edge maps. Recent algorithms for edge detection focus on accurately detecting object boundaries, which can provide intrinsic cues for object detection, segmentation, and tracking.

Lu et al. [24] proposed an algorithm to automatically select thresholds for the Canny algorithm using a histogram of the gradient image. Fang et al. [25] proposed an algorithm to choose a high threshold for the Canny algorithm using the Otsu method [26]. Meanwhile, they cannot select a low threshold. Huo et al. [27] proposed an algorithm to determine high and low thresholds in the Canny algorithm. They choose a low threshold using a probability model. Lu et al. [28] adaptively select two thresholds in the Canny algorithm using minimal meaningful gradient and maximal meaningless gradient magnitude assumption. Yitzhaky and

Peli [29] proposed choosing the best edge parameters. First, they construct Estimated Ground Truth (EGT) with different detection results. Then, they determine the optimal parameter set using a Chi-square test. Medina-Carnicer et al. [30] proposed an algorithm for the unsupervised determination of hysteresis thresholds by fusing the advantages and disadvantages of two thresholding algorithms. They find the best hysteresis thresholds in a set of candidates. Medina-Carnicer et al. [31] proposed a method to automatically determine hysteresis thresholds of the Canny algorithm, which can be used as an unsupervised edge detector.

These traditional unsupervised methods have the advantage that they can be applied without a learning process. However, these methods only provide evaluation results for a few images. It is necessary to evaluate them using many data, such as deep learning, to assess their performance objectively.

Reinforcement learning [32] shows a good performance in temporal decision-making problems. In typical reinforcement learning, an agent aims to learn a policy that maximizes accumulated reward from an environment. Recently, integrating reinforcement learning and deep learning showed human-level control [33]. Deep Q-Networks (DQN) [7], [8] showed that human-level control is possible on Atari games. Deep reinforcement learning offers impressive successes on various tasks such as playing the board game GO [34], [35], [36], object localization [37], region proposal [38], and visual tracking [39].

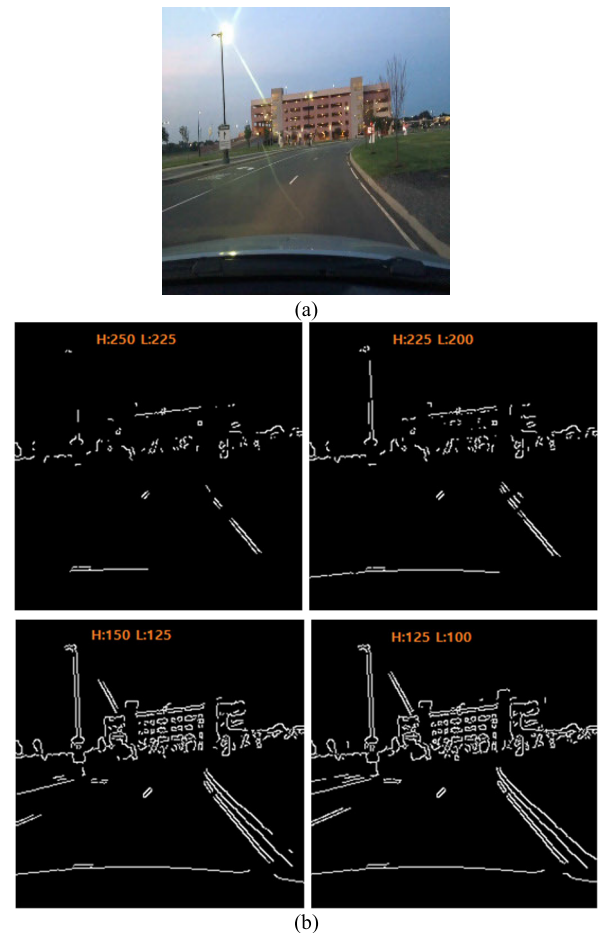
### III. PROPOSED METHOD

In this paper, we propose a method for automatically selecting suitable values of three thresholds in the Canny algorithm using the advantage actor-critic (A2C) method [9], [10], [11]. Figure 2 shows the resulting edge images by different threshold values in the Canny edge method. We notice that the difference in the resulting edge image is enormous according to the threshold values change. However, the resulting edge images are similar when threshold values are in close range. Therefore, we assume the resulting edge image would follow a normal distribution according to the threshold value.

We regard the threshold values in the Canny algorithm as actions of agents. Therefore, through deep reinforcement learning, we could automatically find appropriate threshold values that guarantee an excellent edge image by the Canny algorithm. Based on this assumption, we propose a method to automatically determine the three threshold values which ensure a superb edge image by the Canny algorithm adopting the A2C [9], [10], [11].

#### A. OVERVIEW OF ADVANTAGE ACTOR-CRITIC (A2C) ALGORITHM

In the standard reinforcement learning setting, an agent interacts with an environment  $\varepsilon$  over several discrete time steps. The agent selects an action  $a_t$  at a state  $s_t$  by policy  $\pi$ , where  $\pi$  is a mapping from states  $s_t$  to action  $a_t$ . An action  $a_t$  is chosen from some set of possible actions  $A$ . After the



**FIGURE 2.** The similarity in the resulting edge image according to the threshold values in the Canny algorithm (a) original image (b) resulting edge image.

action, an agent gets to the next state  $s_{t+1}$  and receives a scalar reward  $r_t$ . The process continues until the agent reaches a terminal state. The total accumulated rewards  $R_t$  from time step  $t$  is denoted return as follows.

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad (\gamma \in (0, 1]) \quad (1)$$

$\gamma$  is discounting factor. The agent wants to maximize the expected return from each state  $s_t$ . Policy-based model-free methods directly parameterize the policy  $\pi(a|s;\theta)$  and find the parameters  $\theta$  that maximize  $E[R_t]$ . The REINFORCE family of algorithms can be adopted [9]. Typical REINFORCE updates the policy parameters  $\theta$  in the direction  $\nabla_{\theta} \log \pi(a_t | s_t; \theta) R_t$ , which is an unbiased estimate of  $\nabla_{\theta} E[R_t]$ . It is possible to reduce the variance of this estimate while keeping it unbiased by subtracting a learned function of the state  $b_t(s_t)$ , known as a baseline [9], from return. The resulting gradient is as follows.

$$\nabla_{\theta} \log \pi(a_t | s_t; \theta) (R_t - b_t(s_t)) \quad (2)$$

A learned function of the value function  $V^{\pi}(s_t)$  is commonly used as the baseline, leading to a much lower

variance estimate of the policy gradient. The advantage of action  $a_t$  at state  $s_t$  is defined as follows.

$$A(a_t, s_t) = Q(a_t, s_t) - V(s_t) \quad (3)$$

$Q^\pi(a_t, s_t) = E(R_t | s_t = s, a_t = a)$  is an action value corresponding to the expected return for selecting action  $a$  in state  $s$  following policy  $\pi$ . Since  $R_t$  is an estimate of  $Q^\pi(a_t, s_t)$  and  $b_t$  is an estimate of  $V^\pi(s_t)$ ,  $A(a_t, s_t)$  can be used for  $R_t - b_t(s_t)$  in Eq. (2). This approach can be viewed as an actor-critic architecture where the policy  $\pi$  is the actor and the baseline  $b_t$  is the critic [10].

If we use a one-step return, Eq. (2) can be represented as follows.

$$\nabla_{\theta} \log \pi(a_t | s_t; \theta) (R_{t+1} + \gamma V(s_{t+1}; \theta_v) - V(s_t; \theta_v)) \quad (4)$$

In advantage actor-critic algorithm, two networks of  $\pi(a_t | s_t; \theta)$  and  $V(s_t; \theta_v)$  are used.

## B. STATE CONFIGURATION

First, we consider the original image as the state and threshold values as the action. If we regard an original image as a current state  $s_t$ , we may consider an edge image which corresponds to the result of actions as a next state  $s_{t+1}$ . If the resulting edge image is considered as a next state  $s_{t+1}$ , it may cause problems because an original image and an edge image has different properties. In Eq. (4), the term  $V(s_t; \theta_v)$  uses an original image as input while the term  $V(s_{t+1}; \theta_v)$  uses an edge image as input. Therefore, it isn't easy to train the value network if we consider the original image the current state and the resulting edge image the next state.

We regard a randomly selected image as the next state  $s_{t+1}$  not an edge image to solve this problem. If we train the model well, we can assume that appropriate actions will be selected for randomly chosen images. Therefore, the value evaluation for the randomly selected image would be similar to the value evaluation considering future actions. Throughout this, we can apply the A2C algorithm to choose thresholds in the Canny algorithm automatically.

## C. ACTOR AND CRITIC NETWORK CONFIGURATION

In the A2C, an actor network evaluates actions performed by an agent, and a critic network evaluates the accumulated return from the current state. There are two kinds of actor and critic networks configuration in the A2C algorithm. The first method uses the same backbone and has two branch outputs for the actor and critic networks. The second method uses two different networks for the actor and critic networks. Through experiments, we found that the first method that uses the same backbone gives inconsistent results, including divergence during training. Therefore, in this paper, the actor and critic networks use separate networks.

Compared to the DQN, the A2C algorithm has the advantage that it can deal with continuous actions. The policy will have a continuous output if it follows a normal

distribution.

$$\pi(a_t | s_t; \theta) = \frac{1}{\sigma(s_t; \theta) \sqrt{2\pi}} \exp\left(-\frac{(a_t - \mu(s_t; \theta))^2}{2\sigma(s_t; \theta)^2}\right) \quad (5)$$

$\mu(s_t; \theta)$  and  $\sigma(s_t; \theta)$  is the mean and standard deviation of the Gaussian distribution.

Figure 3 shows the structure of the proposed actor network. We use an original image as the input of the network. The actor network determines the mean and variance of the Gaussian distribution. The network output corresponds to the mean and variance of three actions. We select threshold values by random selection from the Gaussian distribution.

We extract features of the original image using a pre-trained CNN of the ResNet50 structure. Then they are used to yield the mean and standard deviation of the Gaussian distribution. Parameters related to the fully connected (FC) layers are trained after random initialization. We use tanh as the activation function at the last layer for the mean to have a value between  $-1$  and  $1$ . We use sigmoid as the activation function at the last layer for standard deviation to have a value between  $0$  and  $1$ . The output of the actor network is continuous real values, and it is necessary to convert them into the range used in the Canny algorithm. For the high and low thresholds, we convert the output of the actor network into an integer from  $0$  to  $500$ . For the filter size of the smoothing window, we convert the output of the actor network into an integer from  $3$  to  $9$ .

Figure 4 shows the structure of the proposed critic network. The input of the critic network is composed of an original image and action values, while a typical critic network uses only the current state as input. We randomly select action values from the normal distribution provided by the actor network and then use them as input for the actor network. The output of the critic network corresponds to  $V(s_t)$ . Since the critic network is a model that approximates the  $V(s_t)$  value according to each action, we use action values as the input of the critic network. Through this, we can guarantee that training would increase the probability of selecting proper actions from the normal distribution of the actor network.

We do not use an activation function at the output layer in the critic network. It is based on the consideration that the critical network evaluates the accumulated return from the current state. In addition, we concatenate three action values with features from the pre-trained model in the middle of the fully connected layer to prevent them from being reflected in a small proportion.

## D. REWARD COMPUTATION

In reinforcement learning for robot posture control, we can easily set rewards by checking whether the robot is straight or falling. However, in the case of the automatic selection of the three threshold values of the Canny edge to be solved in this paper, reward selection is complex. It is necessary to evaluate the resulting edge image by selected values of three thresholds. To this end, in this paper, the reward is computed using the result of the edge evaluation network.

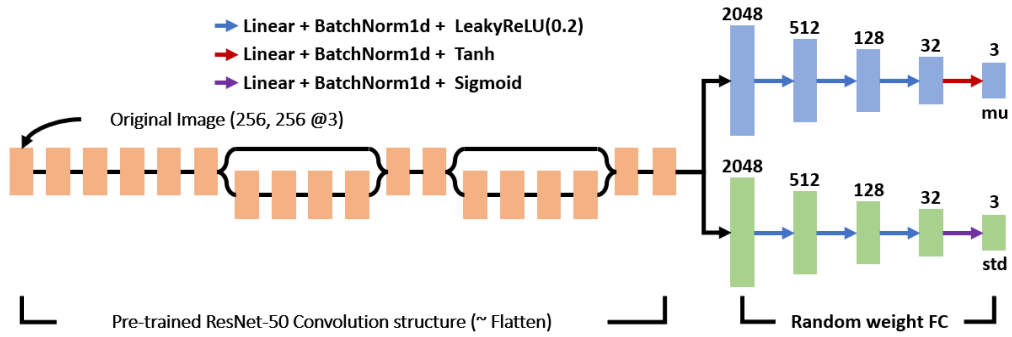


FIGURE 3. The structure of the proposed actor network.

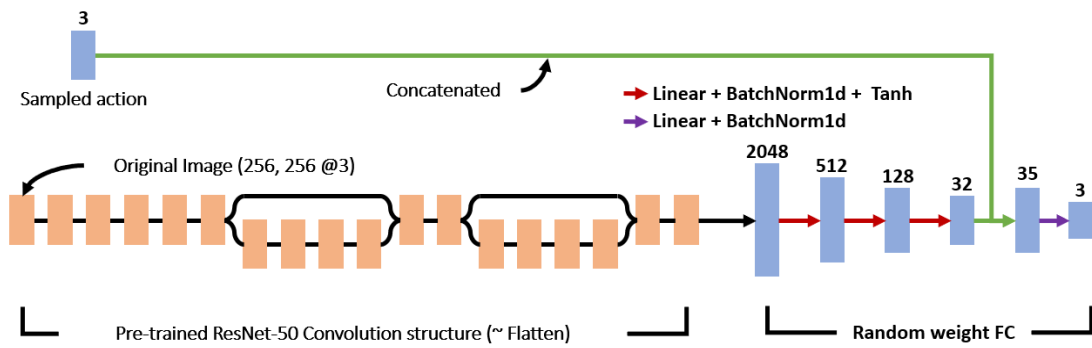


FIGURE 4. The structure of the proposed critic network.

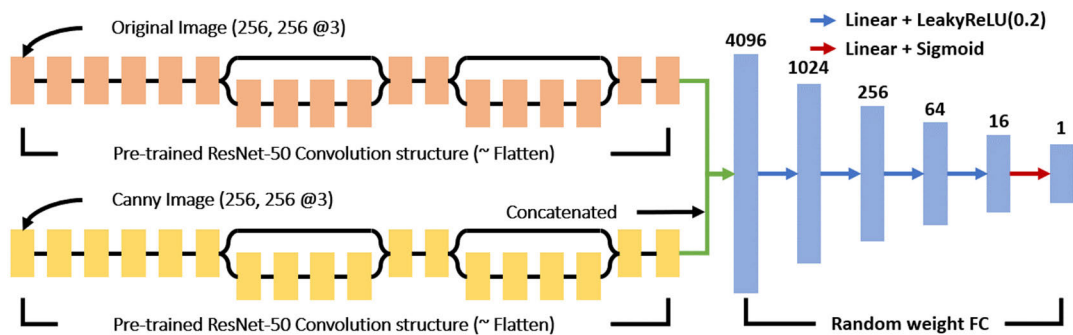


FIGURE 5. The structure of the proposed edge evaluation network.

We train it in an end-to-end manner. Therefore, it requires generating separate label data. Also, it has a disadvantage that generalization ability is poor in an environment that is not similar to the training.

Figure 5 shows the proposed edge evaluation network. It uses the original image and the edge image as input and outputs the goodness of the edge image. We use the pre-trained CNN structure of ResNet-50 for extracting features of the original image and the edge image. The extracted features are concatenated and go through a fully connected layer to yield the output. The last layer of the fully connected layer uses the sigmoid as an activation function. Through this, the output has a value between 0 and 1.

Our previous work [6] used an edge evaluation network that only used the original image as input. In experimental results, we show that the proposed edge evaluation network gives an improved outcome.

Since the proposed edge evaluation network has output values between 0 and 1, we always have positive output values if we use them directly as rewards. If only positive reward values are used, feedback on wrong actions is impossible in reinforcement learning [40]. In addition, it is necessary to take measures to suppress the occurrence of the reversal of high and low thresholds. The reverse of high and low thresholds is inevitable because we randomly choose actions from the Gaussian distribution.

**TABLE 1.** Reward value configuration by reflecting action result and edge evaluation result.

action	edge evaluation	reward value			
		case 1	case 2	case 3	case 4
high $\geq$ low	$\geq 0.999$	1.0	1.0	1.0	1.0
	$\geq 0.500$	0.5	0.5	0.5	0.5
	$< 0.500$	-0.5	-0.5	-0.5	-0.5
high < low	-	-1.0	-5.0	-10.0	-50.0

Table 1 shows the reward configuration after reflecting on these two considerations. We make reward to have positive and negative values according to the output of the edge evaluation network. When reversing the high and low thresholds occurs, we set the reward to a negative value. We experimented with four cases, when the reversal of high threshold and low threshold occurs, to investigate the effect of the magnitude of negative reward. For all four cases, reward values are negative but have different magnitudes.

#### IV. EXPERIMENTAL RESULTS

Experiments are done using NVIDIA 3090 and Intel i9-10900. Table 2 shows the values of hyperparameters used for training the actor, critic, and edge evaluation networks. All networks use images with sizes 256(H)X256(W)X3(C). We divide each pixel by 255, which results in a value between 0 and 1. The Adam optimizer was used for training, and the learning rate was 0.001.

##### A. RESULT OF EDGE EVALUATION NETWORK

We used 600 images from BDD100K [41] to train the edge evaluation network in Figure 5. We manually generated one negative and one positive edge image for each image. Then, we augmented them through geometric transformations. Finally, we used 20,000 images for training the edge evaluation network. We used the mean squared error (MSE) by the output of the network and the ground truth label as loss of the edge evaluation network. The accuracy is defined as follows.

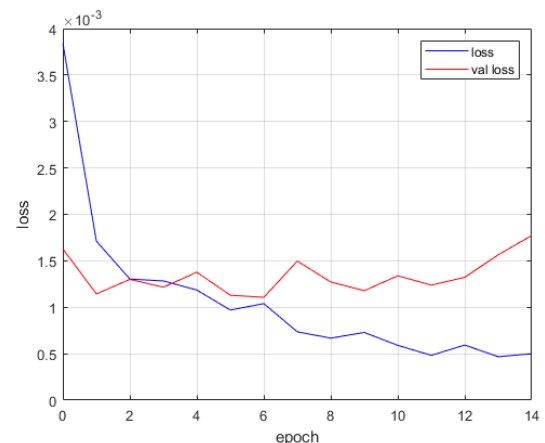
$$ACC = (1 - |O_g - O_p|) \times 100 \quad (6)$$

$O_g$  is a ground-truth value. A positive edge image has a value of 1, and negative edge images have a value of 0.  $O_p$  is an output of the edge evaluation network, and it has a value between 0 and 1.

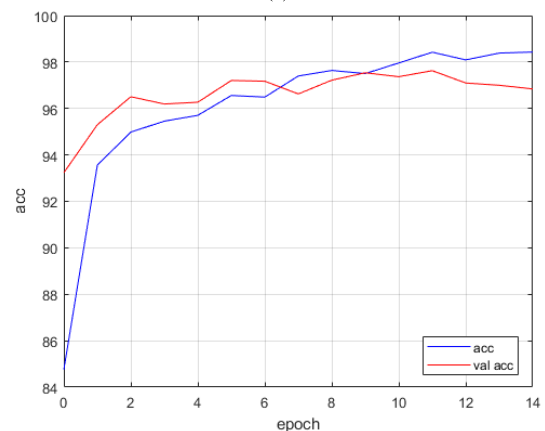
Figure 6 shows the variation of loss and accuracy during training of 14 epochs. We use the result of the 11th epoch, which offers the best outcome for the validation data. Figure 7 shows the comparison results of the proposed edge evaluation network that uses both the original image and edge image as input and our previous edge evaluation network [6], which only uses an edge image as input. The proposed edge evaluation network provides better results than our previous edge evaluation network [6].

**TABLE 2.** Hyperparameters used for training actor network, critic network, and edge evaluation network.

	actor network	critic network	edge evaluation network
input image	256(H)X256(W)X3(C)		original image: 256(H)X256(W)X3(C)
	next image: random		edge image: 256(H)X256(W)X3(C)
preprocessing	image resize $\rightarrow$ pixel range conversion ( $[0, 255] \rightarrow [0, 1]$ )		
optimizer	Adam Optimizer		
learning rate	0.001		
max epoch	35		20
train epoch	35		14
batch size	8	8	16
$\gamma$ (Eq. (4))	0.99		-



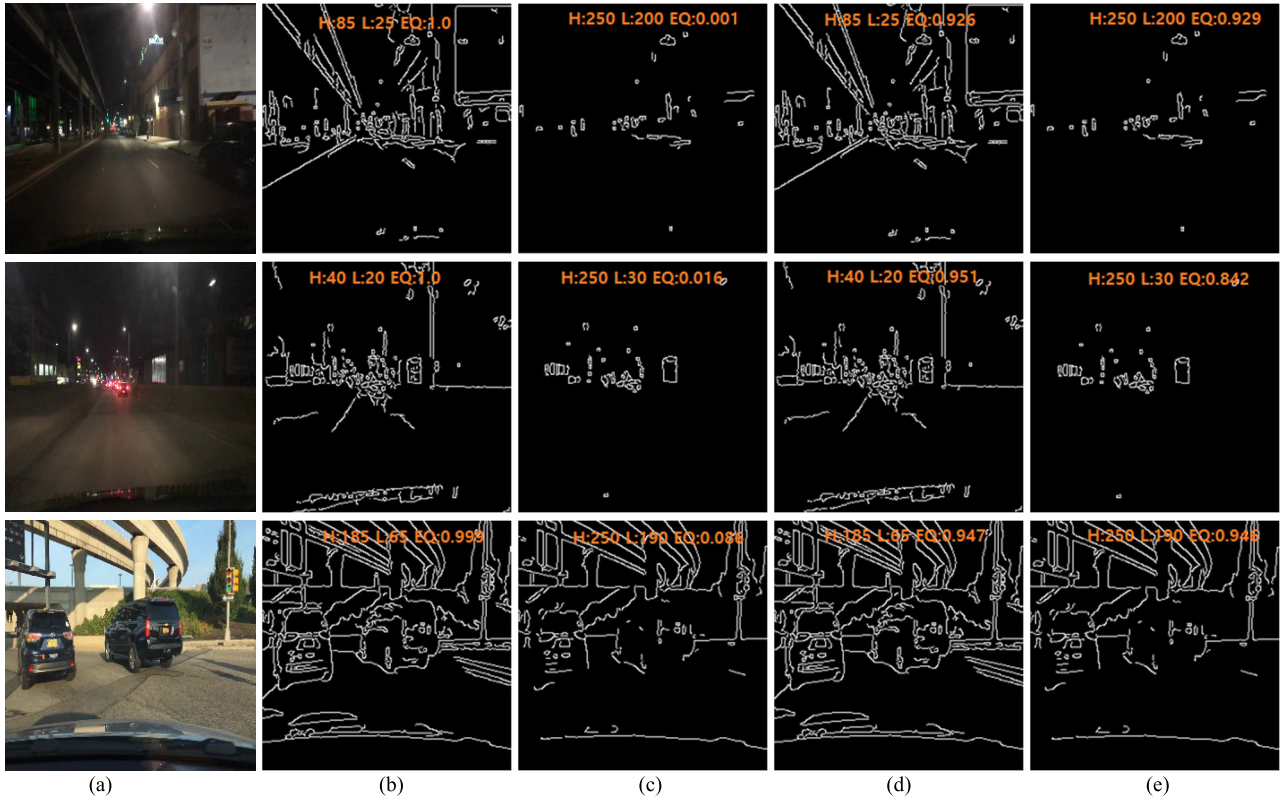
(a)



(b)

**FIGURE 6.** The variation of loss and accuracy during training (a) loss (b) accuracy.

Images acquired at nighttime have low contrast. Therefore, it is challenging to extract edges. The previous edge evaluation network used in [6] has difficulty in those images, as shown in Figure 7. It gives a wrong high evaluation for a low-quality edge image. The proposed edge evaluation network gives a correct assessment even in this case. Improved evaluation of edge image provides a proper reward, and at last, it helps train the proposed method. This fact is shown in experimental results.



**FIGURE 7.** The comparison results of the proposed edge evaluation model and evaluation model used in [6] (a) original image (b) proposed algorithm on a good edge image (c) proposed algorithm on a bad edge image (d) [6] on a good edge image (e) [6] on a bad edge image (EQ: stand for edge quality).

**B. RESULT OF THE PROPOSED METHOD**

We automatically determine the values of three thresholds for the Canny edge algorithm from the output of the actor network in Figure 3. The training of the actor network and the critic network was done using 50,000 images in BDD100K [41]. We use three items for the assessment of training results. First, we use the change of variance output in the actor network. Second, we use the ratio of reversal between high and low thresholds. Third, we check the average of the output of the edge evaluation network.

If training is done well, the actor network will yield appropriate threshold values in a narrow range. Therefore, when training is done in a proper direction, the magnitude of variance, one of the two outputs of the actor network in Figure 3, will decrease as training goes on.

Figure 8 shows the variance change of three actions according to four cases of reward configuration during training. In Figure 8, we notice that the magnitude of variance is continuously decreasing for all cases. It indicates that the training is proceeding in the correct direction. The variance of smoothing window size has a relatively large value, while the high and low thresholds variance falls below 0.3. We convert the smoothing window size value into a value between 3 and 9 and the high and low threshold values into a value between 0 and 500. The relatively high variance

of smoothing window size is acceptable, considering the conversion range difference. Among the four reward cases in Table 1, case 3 shows the smallest variance. Case 3 can be viewed as the most suitable reward model through this.

Figure 9 shows the variation of reward values during training by four reward models in Table 1. The average reward values by the last 80 samples for each trial are shown. We notice that the average reward value continuously increases to 1.6 million, which indicates that training is being appropriately performed. Finally, the average reward value has values larger than 0.5. This shows that the edge evaluation network obtained values larger than 0.5 in most images by referring to the reward value configuration in Table 1. When the actor network is appropriately trained, the edge evaluation network might have a high output value.

Figure 10 shows the edge evaluation network’s output value change during training. It shows a similar tendency to Figure 9. Also, we indirectly convince that the proposed method is well-trained in a direction that offers appropriate threshold values for the Canny algorithm.

Table 3 shows the comparison results of the proposed and the DQN [6] methods. The original DQN method [6] uses an original image as the input of the edge evaluation network. We also tested the DQN method using the proposed edge evaluation network, which uses an original image and an edge

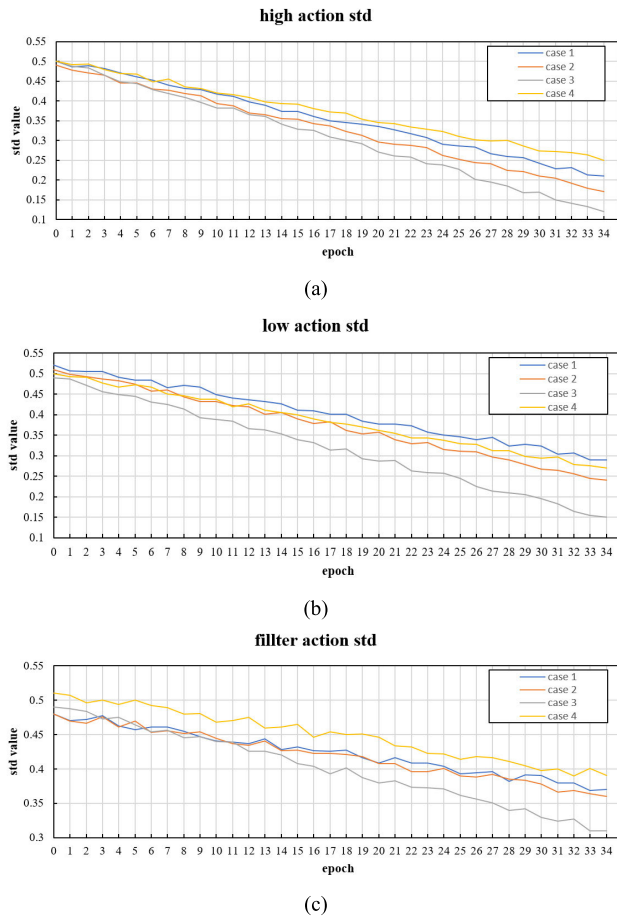


FIGURE 8. The change of variance of three actions according to the four reward cases during training (a) high threshold (b) low threshold (c) smoothing window size.

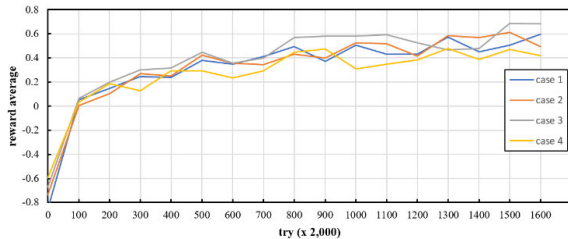


FIGURE 9. The variation of the mean reward of the last 80 samples during training.

image as input. Statistics in Table 3 are obtained using 20,000 images.

The proposed actor network’s output is the Gaussian distribution’s mean and standard deviation. Three Gaussian distributions corresponding to three thresholds are independently established. This fact makes reversing between the high and low threshold inevitable. We set a negative reward to prevent the reversal from occurring. Therefore, if the model is trained correctly, the high threshold should have a larger value than the low threshold. Checking the frequency of the high

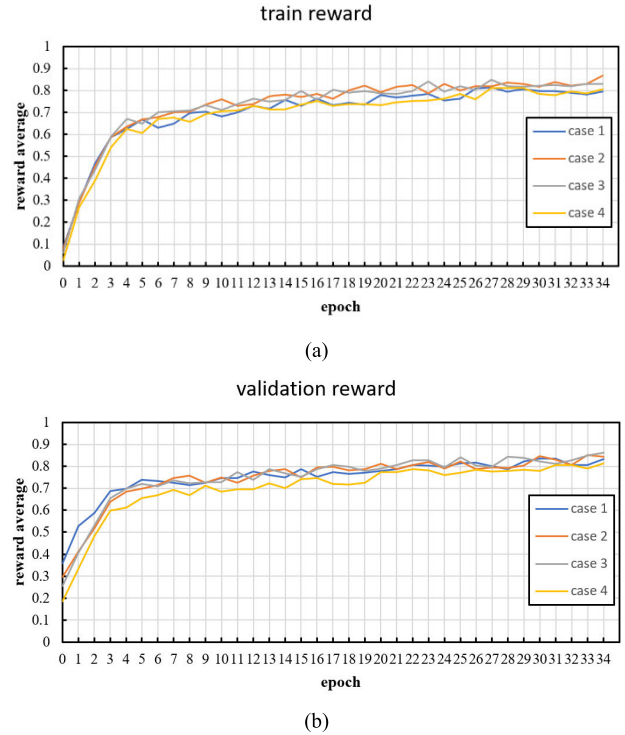


FIGURE 10. The variation of quality value of the edge evaluation model during training (a) training set (b) validation set.

TABLE 3. Comparison results of the proposed algorithm and DQN [6].

method	edge evaluation network input	reward type (Table 1)	reward average	no threshold reversal ratio(%)
DQN[6]	edge only	case 1	0.782	100.0
		case 2	0.781	100.0
		case 3	0.810	100.0
		case 4	0.804	100.0
proposed	original image and edge image	case 1	0.815	87.1
		case 2	0.832	94.9
		case 3	0.837	96.7
		case 4	0.794	97.4

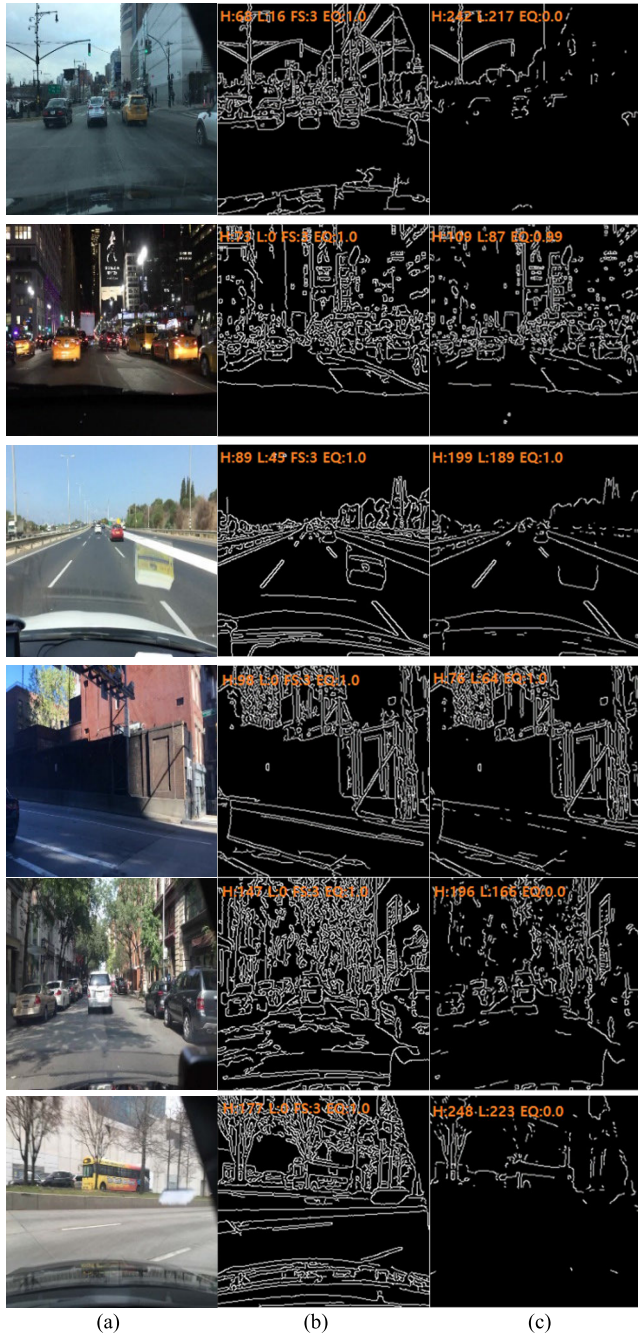
TABLE 4. Comparison result when applying to unseen images of YTVOS [42].

method	reward type (Table 1)	reward average	no threshold reversal ratio(%)
DQN[6]	case 1	0.658	100.0
	case 2	0.684	100.0
	case 3	0.721	100.0
	case 4	0.699	100.0
proposed	case 1	0.715	86.6
	case 2	0.723	94.7
	case 3	0.746	95.8
	case 4	0.698	88.6

and low threshold reversal is a good candidate for checking how well training goes.

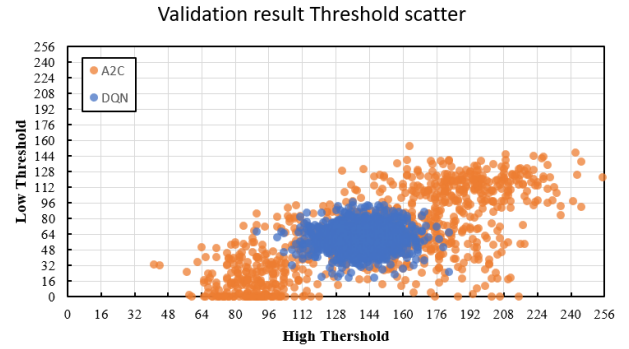
Table 3 shows that it is essential to give a negative reward when the reversal of the high and low thresholds



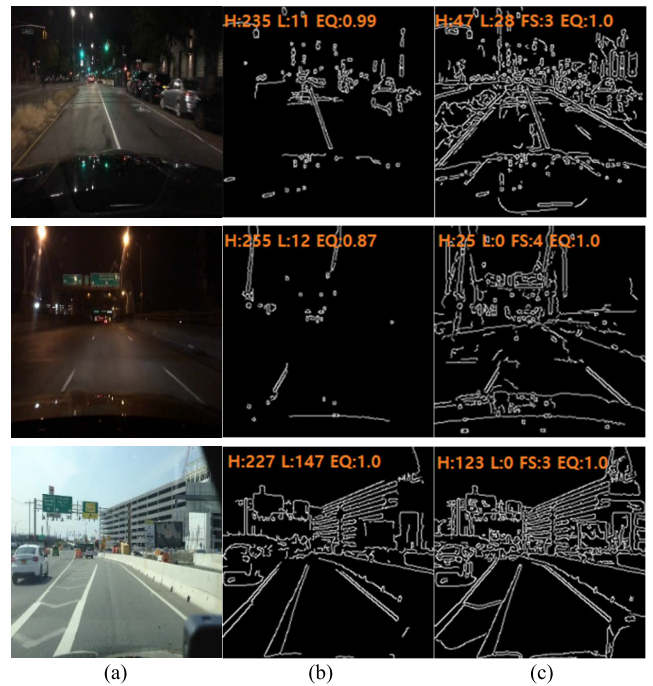


**FIGURE 11.** The comparison result of edge image by the proposed and DQN [6] method (a) original image (b) edge image by the proposed method (c) edge image by DQN [6] method (FS, EQ: stand for filter size and edge quality).

occurs. As we increase the magnitude of the negative reward, the occurrence of the reversal reduces. But it reduces the performance of the actor network. Therefore, choosing an appropriate negative reward value must consider the actor network’s performance simultaneously. The DQN method [6] has the advantage of no reversal between high and low thresholds. But, it gives a reward value smaller than the proposed method. In all four cases, the proposed algorithm offers superior results than the DQN method [6]. But 96.7%



**FIGURE 12.** The distribution of high and low thresholds by the proposed algorithm and DQN [6].



**FIGURE 13.** Result of the proposed method on images having false edge evaluation results (a) original image (b) false cases by edge evaluation network (c) result by the proposed method.

**TABLE 5.** Result of the proposed algorithm after retraining using YTVOS [42] images.

method	reward type (Table 1)	reward average	relative improvement(%)	no threshold reversal ratio(%)
DQN[6]	case 1	0.679	3.19	100.0
	case 2	0.671	-1.90	100.0
	case 3	0.752	4.30	100.0
	case 4	0.634	-9.30	100.0
proposed	case 1	0.735	2.80	89.0
	case 2	0.779	7.75	96.5
	case 3	0.801	7.37	97.2
	case 4	0.657	-5.87	86.2

of the 20,000 images were obtained without reversing high and low thresholds in the best case 3. In contrast, in the case of

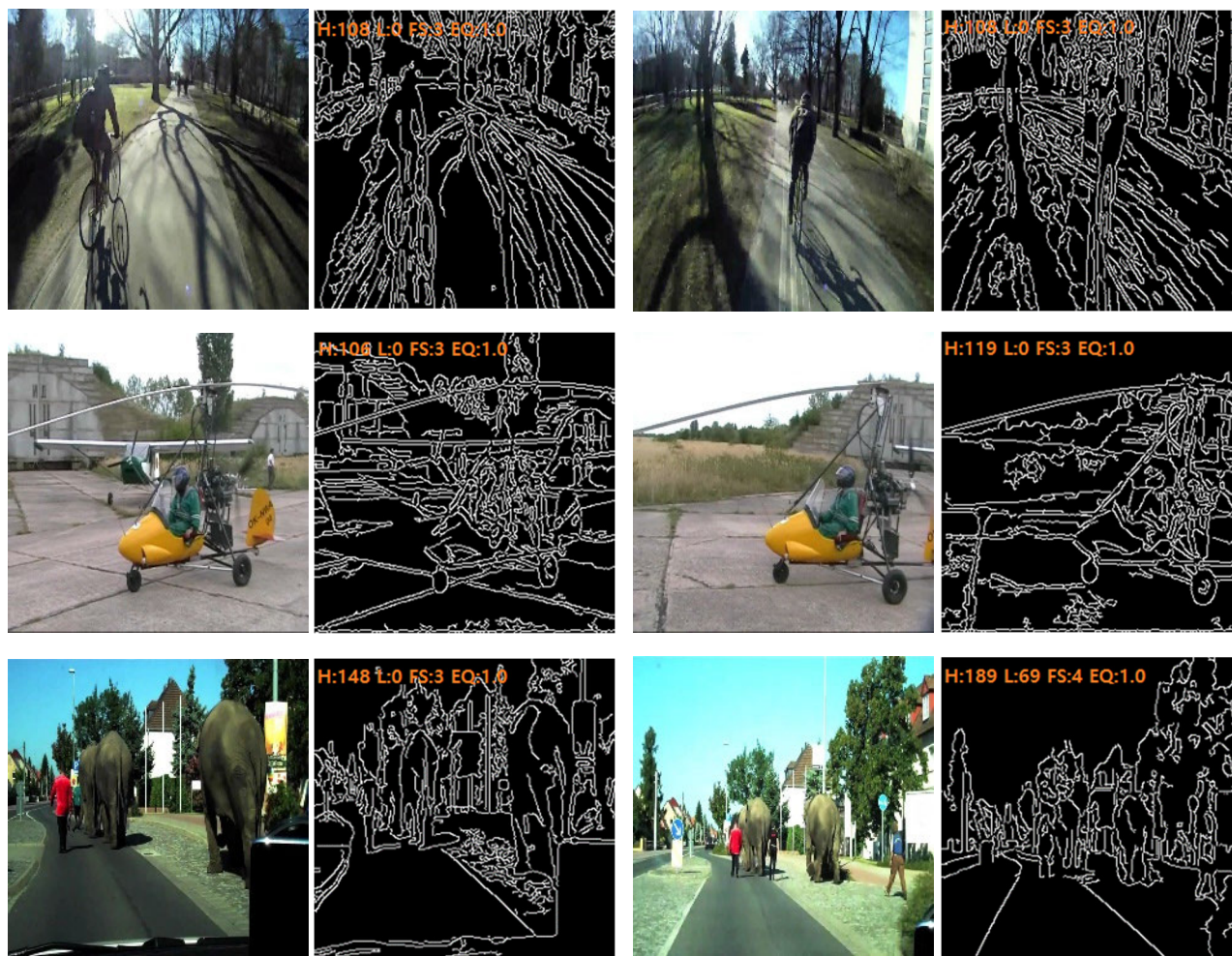


FIGURE 14. Results of the proposed method on unseen images in the YTVOS [42].

the DQN method, the threshold value reversal does not occur in all cases.

The reversal of high and low thresholds in the proposed algorithm is inherently connected to the procedure that randomly selects the mean and standard deviation from the output of the actor network. There is a radical possibility of reversing high and low threshold values during sampling even if the proposed network is well-trained. It requires further research to solve this problem. In contrast, in the DQN method, it is structurally possible to configure a high threshold always to have a larger value than the low threshold. Figure 11 shows the comparison results of edge images by the proposed and the DQN method [6]. In Figure 11, FS and EQ represent filter size and edge quality. EQ corresponds to the output of the edge evaluation network in Figure 5. The proposed method provides better results than the DQN method [6], particularly on difficult images with dark areas, small objects, and complex boundaries.

Figure 12 shows the distribution of high and low thresholds selected on 1,000 images by the proposed and DQN [6] methods. The proposed method gives a broader distribution

than the DQN method. From this fact, we can conclude that the proposed method gives more suitable threshold values than the DQN method.

The edge evaluation network in Figure 5 gives correct evaluation results for most images. However, it occasionally offers high scores even for the bad edge image. Figure 13 shows comparison results using the same image that the edge evaluation network produces a wrong result. Figure 13(a) is an original image. Figure 13(b) is an edge image that gives a bad score in that edge evaluation network. Figure 13(c) is the edge image by the proposed algorithm. Although the edge evaluation network has some drawbacks, the proposed algorithm improves results. It shows that training is possible even using an edge evaluation network with low accuracy, ultimately providing improved results.

### C. EXPERIMENTAL RESULT USING UNSEEN IMAGES

We test using unseen images to evaluate the generalization ability of the proposed method qualitatively. The proposed method uses three networks: actor, critic, and edge evaluation. All three networks are trained using images in

BDD100K [41]. We investigate the proposed method's generalization ability by applying a trained model with BDD100K to images in YTVOS [42]. Table 4 shows results applied to images in YTVOS [42], an environment different from training. We obtain results using 1,070 images in YTVOS. In the BDD100K, case 3, which provides the best result in the proposed method, showed an average reward value of 0.837. When applied to the YTVOS image, the average reward value decreased to 0.746. In the new environment, all four cases showed decreased reward value. In YTVOS, the proposed method still provides better results than the DQN method [6]. Among the four reward configurations in Table 1, case 3 offers the best result by the proposed and DQN [6] method as in the case of BDD100K. Figure 14 shows the results of the proposed method applied to images of YTVOS.

Next, we trained the actor and critic networks using only images in YTVOS. We use the same edge evaluation network without training using images in YTVOS. Therefore, the proposed method can do additional training without requiring ground truth labels of edge images. Table 5 shows results after additional training using 14,000 images in YTVOS. The statistics in Table 5 are obtained using the results of 1,070 images. The DQN method shows an improvement of 4.1% from 0.721 to 0.751, while the proposed method shows a gain of 7.4% from 0.746 to 0.801. Also, the proposed method gives a better result than the DQN [6] method. In Table 4 and Table 5, both the proposed and the DQN [6] methods used the model in Figure 5 as the edge evaluation network.

Though we cannot have results comparable to the case of BDD100K, the proposed algorithm can have improved results by training only using images in unseen environments. The edge evaluation network must be trained using images in unseen environments if we want a performance similar to the case of BDD100K.

## V. CONCLUSION

This paper proposes a method for automatically selecting appropriate values of three thresholds in the Canny edge algorithm. In the Canny edge algorithm, the resulting edge images have a similar tendency if their threshold values are similar. We adopt the A2C algorithm considering this fact. The proposed actor network extracts the feature of an image using pre-trained weights and then yields the mean and standard deviation of three thresholds through the fully connected layers. The proposed critic network also extracts features of an image using pre-trained weights then sampled actions are integrated into the fully connected layers. We propose a reward configuration using the edge evaluation network and considering the reversal of high threshold and low threshold. The proposed method can adapt to unseen images by training only using original images in unseen environments. The proposed method has shortcomings: it cannot completely cope with the reversal of high and low thresholds, and the edge evaluation network is trained in a supervised way. For further research, we are going to research to solve these problems. Also, we want to apply SAC [43] and

PPO [44], which give the best performance among many deep reinforcement learning algorithms, to our problem.

## REFERENCES

- [1] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 328–335.
- [2] L. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4545–4554.
- [3] M. Modava and G. Akbarizadeh, "Coastline extraction from SAR images using spatial fuzzy clustering and the active contour method," *Int. J. Remote Sens.*, vol. 38, no. 2, pp. 355–370, Jan. 2017.
- [4] T. Hermosilla, J. Palomar-Vázquez, Á. Balaguer-Beser, J. Balsa-Barreiro, and L. A. Ruiz, "Using street based metrics to characterize urban typologies," *Comput., Environ. Urban Syst.*, vol. 44, pp. 68–79, Mar. 2014.
- [5] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 34–43, Jun. 1986.
- [6] K. Choi and J. Ha, "An adaptive threshold for the Canny algorithm with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 156846–156856, 2021.
- [7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
- [8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
- [9] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.
- [10] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 2177–2182.
- [11] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous method for deep reinforcement learning," in *Proc. ICLR*, 2016, pp. 1928–1937.
- [12] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu, "Statistical edge detection: Learning and evaluating edge cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 1, pp. 57–74, Jan. 2003.
- [13] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 530–549, May 2004.
- [14] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.
- [15] P. Dollar, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2006, pp. 1964–1971.
- [16] X. Ren, "Multi-scale improves boundary detection in natural images," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 533–545.
- [17] J. J. Lim, C. L. Zitnick, and P. Dollár, "Sketch tokens: A learned mid-level representation for contour and object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3158–3165.
- [18] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 8, pp. 1558–1570, Aug. 2015.
- [19] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3982–3991.
- [20] G. Bertasius, J. Shi, and L. Torresani, "DeepEdge: A multi-scale bifurcated deep network for top-down contour detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4380–4389.
- [21] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1395–1403.
- [22] D. Xu, W. Ouyang, X. Alameda-Pineda, E. Ricci, X. Wang, and N. Sebe, "Learning deep structured multi-scale features using attention-gated CRFs for contour prediction," in *Proc. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3964–3973.

- [23] J. He, S. Zhang, M. Yang, Y. Shan, and T. Huang, "Bi-directional cascade network for perceptual edge detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3823–3832.
- [24] J.-W. Lu, J.-C. Ren, Y. Lu, X.-H. Yuan, and C.-G. Wang, "A modified Canny algorithm for detecting sky-sea line in infrared images," in *Proc. 6th Int. Conf. Intell. Syst. Design Appl.*, Oct. 2006, pp. 16–18.
- [25] M. Fang, G. X. Yue, and Q. C. Yu, "The study on an application of Otsu method in Canny operator," in *Proc. ISIP*, 2009, pp. 109–112.
- [26] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.
- [27] Y.-K. Huo, G. Wei, Y.-D. Zhang, and L.-N. Wu, "An adaptive threshold for the Canny operator of edge detection," in *Proc. Int. Conf. Image Anal. Signal Process.*, 2010, pp. 371–374.
- [28] X. Lu, J. Yao, K. Li, and L. Li, "CannyLines: A parameter-free line segment detector," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2015, pp. 507–511.
- [29] Y. Yitzhaky and E. Peli, "A method for objective edge detection evaluation and detector parameter selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 1027–1033, Aug. 2003.
- [30] R. Medina-Carnicer, A. Carmona-Poyato, R. Munoz-Salinas, and F. J. Madrid-Cuevas, "Determining hysteresis thresholds for edge detection by combining the advantages and disadvantages of thresholding methods," *IEEE Trans. Image Process.*, vol. 19, no. 1, pp. 165–173, Jan. 2010.
- [31] R. Medina-Carnicer, R. Muñoz-Salinas, E. Yeguas-Bolivar, and L. Diaz-Mas, "A novel method to look for the hysteresis thresholds for the Canny edge detector," *Pattern Recognit.*, vol. 44, no. 6, pp. 1201–1211, Jun. 2011.
- [32] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
- [33] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [34] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, and S. Dieleman, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [35] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, and Y. Chen, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
- [36] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.
- [37] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2488–2496.
- [38] Z. Jie, X. Liang, J. Feng, X. Jin, W. Lu, and S. Yan, "Tree-structured reinforcement learning for sequential object localization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 127–135.
- [39] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1349–1358.
- [40] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
- [41] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," 2018, *arXiv:1805.04687*.
- [42] N. Xu, L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. Price, S. Cohen, and T. Huang, "YouTube-VOS: Sequence-to-sequence video object segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 585–601.
- [43] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
- [44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.



**KEONG-HUN CHOI** received the B.S. and M.E. degrees in mechanical and automotive engineering from the Seoul National University of Science and Technology, Seoul, South Korea, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree with the Graduate School of Automotive Engineering. His current research interests include deep learning and deep reinforcement learning.



**JONG-EUN HA** received the B.S. and M.E. degrees in mechanical engineering from Seoul National University, Seoul, South Korea, in 1992 and 1994, respectively, and the Ph.D. degree in mechanical engineering from KAIST, Daejeon, South Korea, in 2000. From February 2000 to August 2002, he was with Samsung Corning, developing an algorithm for a machine vision system. From 2002 to 2005, he was in multimedia engineering with Tongmyong University. Since 2005, he has been a Professor with the Department of Mechanical and Automotive Engineering, Seoul National University of Science and Technology. His current research interests include deep learning, intelligent robots, and vehicles.

• • •