Action-driven Reinforcement Learning for Improving Localization of Brace Sleeve in Railway Catenary

Zhong, Junping; Liu, Zhigang; Wang, Hongrui; Liu, Wenqiang; Yang, Cheng; Nunez, Alfredo

DOI
10.1109/ICSMD50554.2020.9261697

Publication date
2020

Document Version
Final published version

Published in
International Conference on Sensing, Measurement and Data Analytics in the Era of Artificial Intelligence, ICSMD 2020 - Proceedings

Action-driven Reinforcement Learning for Improving Localization of Brace Sleeve in Railway Catenary

Junping Zhong
School of Electrical Engineering
Southwest Jiaotong University
Chengdu, China
zhongjunping@my.swjtu.edu.cn

Zhigang Liu*
School of Electrical Engineering
Southwest Jiaotong University
Chengdu, China
liuzg_cd@126.com

Hongrui Wang
Section of Railway Engineering
Delft University of Technology
Delft, Netherlands
H.Wang-8@tudelft.nl

Wenqiang Liu
School of Electrical Engineering
Southwest Jiaotong University
Chengdu, China
Liuwq_2009@126.com

Cheng Yang
School of Electrical Engineering
Southwest Jiaotong University
Chengdu, China
yangc@my.swjtu.edu.cn

Alfredo Núñez
Section of Railway Engineering
Delft University of Technology
Delft, Netherlands
a.a.nunezvicencio@tudelft.nl

***Abstract*—Brace Sleeve (BS) plays an essential role in connecting and fixing cantilevers of railway catenary systems. It needs to be monitored to ensure the safety of railway operations. In the literature, image processing techniques that can localize BSs from inspection images are proposed. However, the boxes produced by existing methods can contain incomplete and/or irrelevant information of the localized BS. This reduces the accuracy of BS condition diagnosis in further analyses. To address this issue, this paper proposes the use of an action-driven reinforcement learning method that adopts the coarse-localized box provided by existing methods, and finds the movements needed for the box to approach to the true BS position automatically and accurately. In contrast to the existing methods that predict one position of the box containing a BS, the proposed action-driven method sees the localization problem as a dynamic position searching process. The localization of BS is achieved by following a sequence of actions, which in this paper are position-moving (up, down, left or right), scale-changing (scale up or scale down) and shape-changing (fatter or taller). The policy of selecting dynamic actions is obtained by reinforcement learning. In the experiment, the proposed method is tested with real-life images taken from a high-speed line in China. The results show that our method can effectively improve the localization accuracy for 81.8% of the analyzed images. We also analyze cases where the method did not improve the localization and suggest further research lines.**

Keywords—railway catenary, localization, brace sleeve, reinforcement learning, action-driven learning.

I. Introduction

Catenary is an important component of the traction power supply system in high-speed railways. A key component in catenary is the brace sleeve (BS). BS plays an important role in connecting and fixing catenary cantilevers. Due to the physical/mechanical impact triggered by the high-speed vehicles and other location and environmental factors along the railway line, the BSs can develop defective states. Defective BSs increase the risk of disrupting the railway operation and compromising safety. To automatically monitor the catenary components, image processing methods have been developed to replace manual checking. Once a defective component is detected, the information updates the maintenance activities planning so that the component can be replaced. The first step of monitoring is the localization of BSs. Localization is an important issue because if it is not accurate, fault detection methodologies will not count with the correct information to perform diagnosis [1].

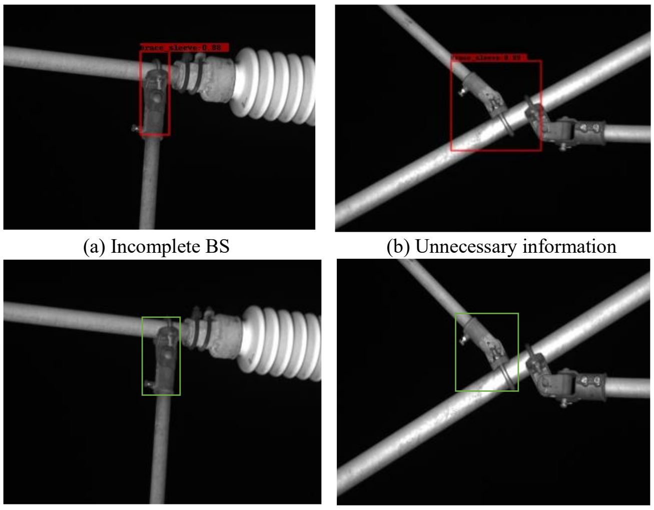

(a) Incomplete BS
(b) Unnecessary information
(c) Complete BS
(d) BS without unnecessary information

Fig. 1 Examples of detected BS using Faster R-CNN with a box containing (a) incomplete information, (b) unnecessary information. In (c) and (d) it is possible to see the optimal boxes for the previous cases.

In the past decade, two classes of localization algorithms have been widely used for railway component localization. The first class of algorithms are based on handcrafted features. Han *et al.* [2] used cascade support vector machines to classify a series of sliding window images, which are represented by a HOG to localize the catenary clevis. Zhong *et al.* [3] applied template matching on a standard catenary sleeve image and an original image to search the object position based on SIFT. Fan *et al.* [4] proposed a line LBP encoding method to represent a target object, which is used to localize fasteners on the rail track. The second class of algorithms are based on deep learning, which adopt supervised learning to train deep regression models that predict object position directly. In [5], a region-based convolutional neural network called Faster R-CNN is proposed to extract deep CNN features and localize general objects. Cai *et al.* [6] cascaded several regression modules in Faster R-CNN to further refine localization. Liu *et al.* [7, 8] and Kang *et al.* [9] applied improved Faster R-CNN to localize class-specific component, such as isoelectric line, brace sleeve screws and insulator. In works [10], deep learning architectures were developed to localize all catenary support components. Redmon *et al.* [11] introduced a strong deep CNN architecture called YOLO (You only look once) which allows to obtain a good trade-off between speed of detection and accuracy. Chen *et al.* [12] proposed an

*Zhigang Liu is the corresponding author. (e-mail:liuzgcd@126.com).

978-1-7281-9277-2/20/$31.00 ©2020 IEEE

improved YOLO for catenary components localization. Overall, handcrafted feature-based methods are simpler, but the performance of deep learning-based methods is by far superior for detecting catenary components. However, even state-of-the-art deep learning methods may provide incorrect localizations, either the localized BS is incomplete and/or with unnecessary information, as shown in Fig. 1. In this work, we propose a reinforcement learning (RL) method to address localization problems like the ones shown in Fig. 1.

The RL refers to a broad group of learning techniques. RL emulates the way living beings learn by trying actions and learning from successes and failures. As shown in Fig. 2, in RL, an agent is trained to make good decisions in a given environment by receiving rewards when the decisions are considered positive. The agent observes the state of a given environment, and takes actions that transform the environment to a new state according to its state-action policy, which is learned during training. A *Markov Decision Process* (MDP) is a formal mathematical representation of how the agent interacts with the environment to learn its policy. Recent works [13, 14, 15] in the RL field have proposed to combine deep neural networks with RL algorithms such as value function or policy function. By resorting of deep learning features, many difficult problems such as playing Atari games [17] or Go [13] can be successfully solved in a semi-supervised setting. For computer vision problems, various methods have been proposed in the literature. Caicedo *et al.* [18] proposed an active class-specific localization approach. Yun *et al.* [19] proposed an action decision method for object tracking by RL. In [20-21], RL was adopted to learn a policy of selecting a region from five fixed sub-regions, and realize object localization by only a few steps. So far, we are not aware of available literature applying RL to solve catenary component localization problems.

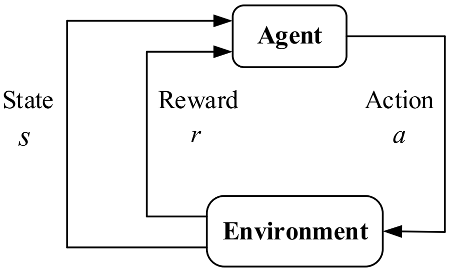


Fig. 2 A schematic flow of reinforcement learning system.

As shown in Fig.1(a) and Fig. 1(b), the boxes localized by existing region-based convolutional neural networks (R-CNN) methods do not enclose BS components tightly. In this paper, motivated by the reward-action in reinforcement learning and [19], we consider the localization improving problem as a control problem where a sequence of steps to refine the geometry of the localization box is to be obtained. Then, the localization refinement becomes a Markov Decision Process that can be trained with RL. We define the actions as position-moving, scale-changing and shape-changing. The reward is feedback about how well the current localization performed. Therefore, the action-decision policy can be learned according to the rewards being obtained. The agent is a deep CNN called ADNET (Action decision network) [19], which is presented in Section II. The application of reinforcement learning for BS localization is described in Section III. Experimental results and conclusions are given in Section IV and Section V, respectively.

The contributions of this paper are summarized as follows:

1. We investigate one possible method that employs RL to train an algorithm to generate a better bounding box for BS localization through a sequence of actions.
2. Different than the existing localization strategies for railway catenary systems, that localize objects following a single structured prediction model, the proposed method is a dynamic strategy that requires emphasis in the learning procedure and learning based on the time evolution of the performance (called history).
3. The preliminary results indicate that the proposed method is effective. The localization accuracy is improved while the time cost is low, which is beneficial for BS monitoring in railway.

II. Localization Improved by Reinforcement Learning

A. Method Overview

The overview of the action-driven method is shown in Fig. 3. The initial box image is an input for a deep CNN called ADNET (Action decision network). ADNET will select one of the actions, which are defined as transformations of the box by moving, scale changing, and shape-changing. Then, the initial geometry box is changed after taking the selected action and produces a new box image, which is sent to the ADNET to decide for the next action. Finally, the BS component is accurately localized by taking a sequence of actions. In this dynamic process, the applied control strategy of action selection is learned by reinforcement learning, which considers the performance feedbacks of all actions at each step. The learned action policy is aimed to automatically adjust the initial box to enclose the BS component automatically.

B. ADNET Structure

As shown in Fig. 3, the agent ADNET consists of three convolutional layers and four fully-connected layers. The parameters of conv1~conv3 and fc4 are similar to the widely used VGG [22] setting. The fc5 is concatenated by a base 1*1*512 and an action history vector. The fc6 is set as an action layer, whose output is a *m*-dimensions vector that represents probabilities of *m* actions. The fc7 is a 2-dimensions vector for classification (object/background). The input size is set to be 112*112*3. When an image patch is inputted into ADNET, it is firstly resized to 112*112*3, and then its deep CNN features are extracted from covn1 to fc6 and fc7 for action prediction and class prediction, respectively. The action and classification that have the max probabilities are selected, and the history *c* is also updated. The action selection strategy is trained by reinforcement learning.

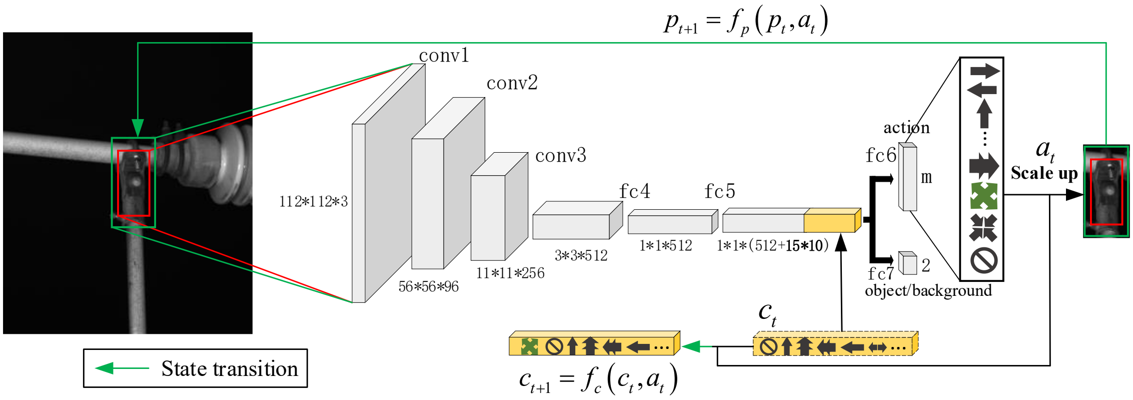


Fig. 3 The architecture of the proposed method.

III. Reinforcement Learning for BS

A. MDP Formulation

The proposed localization refinement strategy follows a Markov Decision Process (MDP). This setting provides a formal framework to model an agent that makes a sequence of decisions. The MDP is defined by states $s \in S$, actions $a \in A$, state transition function $s' = f(s, a)$, and the reward $r(s, a)$. Here, we take the ADNET as an agent to find accurate box regions for BS component by taking sequential actions. Through formulating the localization refinement as the MDP, the action policy of ADNET can be optimized by reinforcement learning. The action, state, state transition function and reward are formulated as follows.

Action: Make the initial box fit the position and shape of BS, transformations of moving {*left, right, up, down*}, scale changing {*scale up, scale down*}, and shape changing {*fatter, taller*} are defined as possible actions. Especially, when the agent finds the optimum location, or the current localized box is the same as the previous box, a stop action is needed to finalize the box searching. Specifically, we define the action space *A* as shown in Fig. 4. The space *A* consists of 15 actions, and provides sufficient transform options for box changing. Rotating options are not considered in this paper.

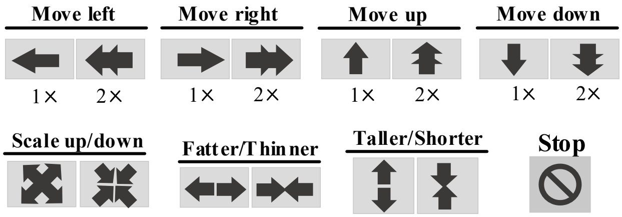


Fig. 4 The defined actions in our method.

State: As the localization refinement is a process of changing the geometry of box, using the information of what actions ADNET has done before is helping to predict better boxes [18, 19]. Thus, the image patch within box and the history actions are used to form the state *s*. For localization refinement in image *I* at step *t*, the state s_t is defined as a tuple (p_t, c_t), where $p_t \in \mathrm{R}^{112*112*3}$ is the image within the current box and the $c_t \in \mathrm{R}^{150}$ denotes the encoded vector of action history. The p_t can be formulated as,

$$p_t = \phi([\mathrm{x}_t, \mathrm{y}_t, \mathrm{w}_t, \mathrm{h}_t], \mathrm{I}) \qquad (1)$$

where (x_t, y_t) is the coordinate of center point of p_t in image *I*, w_t and h_t are the width and height of p_t respectively. The function ϕ crops p_t from image *I* and resizes it to the input size of ADNET. The c_t is a 150-dimensional vector, because we choose previous ten actions as history, and each action is encoded by 15 dimensions.

State transition function: When the ADNET selects an action a_t, the current s_t will transit to s_{t+1}. The state transition is performed by two functions $f_p(p_t, a_t)$ and $f_c(c_t, a_t)$, which are implemented on current image patch p_t and current action history c_t respectively. As for the f_p, the discrete amount of action transformations should be given. The discrete amounts of moving actions are given in (2). In (3), the discrete amounts of scale changing and shape changing actions are defined. As the initial box is not far away from the BS, we set factors α_1, α_2, α_3 and α_4 to 0.05 in our experiments.

$$\Delta x_t = \alpha_1 * w_t, \quad \Delta y_t = \alpha_2 * h_t \qquad (2)$$

$$\Delta w_t = \alpha_3 * w_t, \quad \Delta h_t = \alpha_4 * h_t \qquad (3)$$

As for the state transition f_c, it adds the current action a_t into action history c_t as the latest action, and removes the earliest action.

Reward: The reward can be regarded as feedback after taking an action. During the reinforcement learning training, if the selected action can make the state transition to a better state, then the agent will get a positive reward. Otherwise, a zero reward or negative reward will be returned. In this paper, the reward function $R_t(p_t, p_{t+1})$ is defined as follows.

$$R_t(p_t, p_{t+1}) = \begin{cases} +1, & if\ IoU(p_{t+1}, \mathrm{G}) > IoU(p_t, \mathrm{G}) \\ 0, & if\ IoU(p_{t+1}, \mathrm{G}) = IoU(p_t, \mathrm{G}) \\ -1, & if\ IoU(p_{t+1}, \mathrm{G}) = IoU(p_t, \mathrm{G}) \end{cases} \qquad (4)$$

where G is the ground-truth box of target BS, the *IoU*(p_t ,G) denotes overlap ratio of the current patch p_t and the ground truth G of the target BS with intersection-over-union criterion. In (4), only when the next patch gets closer to the ground-truth of BS than the case of current patch, the agent can obtain a positive reward.

B. Training Objective

The agent ADNET is trained by RL, whose goal is to learn a state-action policy that makes a sequence of action decisions. Before applying RL, we initialize the ADNET by utilizing the weight parameters trained by supervised learning

(SL), which has been proved useful for policy learning [13, 19].

In the SL stage, training samples p_i $\{i$=1, 2, …, $N\}$ are generated by imposing Gaussian noise on the BS ground-truth G_i $[x_i, y_i, w_i, h_i]$. The action label $o_i^{(act)}$ and class label $o_i^{(cls)}$ are defined by (5) and (6), respectively.

$$o_i^{(act)} = \underset{a}{\arg\max}\, IoU\left(f_p(p_i, a), \mathrm{G}_i\right), a \in A \tag{5}$$

$$o_i^{(cls)} = \begin{cases} 1, & \textit{if } IoU\left(p_i, G\right) > 0.6 \\ 0, & \textit{otherwise} \end{cases} \tag{6}$$

where A is the action space described in Section III, it has 15 actions. Then, the initial weight W_{SL}, $\{\mathrm{w}_1, \mathrm{w}_2, …, \mathrm{w}_7\}$, is learned by minimizing the loss L_{RL}.

$$L_{SL} = \frac{1}{N}\sum_{i=1}^{N}[L\left(o_i^{(act)}, \hat{o}_i^{(act)}\right) + L\left(o_i^{(cls)}, \hat{o}_i^{(cls)}\right)] \tag{7}$$

The loss L_{SL} includes action prediction loss and classification (object/background) loss. N is batch size of training samples and L denotes the cross-entropy loss. The $\hat{o}_i^{(act)}$ and $\hat{o}_i^{(cls)}$ are the predicted action and predicted class for sample i, respectively. Note that the history action vector is not used in this stage.

In the RL training stage, the W_{RL}, $\{\mathrm{w}_1, \mathrm{w}_2, …, \mathrm{w}_6\}$, is initialized by $W_{SL.}$ The fc7 is ignored because only action is concerned in RL. For an image frame l, an initial box will take T actions that are successively predicted by ADNET. In each step t, new features are drawn from the image, allowing the RL algorithm to adapt to new information. The action $a_{t,l}$ is selected by

$$a_{t,l} = \arg\max p\left(a \mid s_{t,l}; W_{RL}\right) \tag{8}$$

where $t = 1, 2, …, T\text{-}1, T$.

After action $a_{t,l}$ has been taken, the ADNET gets a reward $R_{t,l}$ according to (4). Meanwhile, a history of what actions ADNET has done before is also recorded. Then the history action vector c_t is updated by adding $a_{t,l}$ and removing the earliest action.

Finally, W_{RL} is updated using stochastic gradient ascent [23] to maximize the accumulated rewards of the training samples as follows.

$$\Delta W_{RL} \infty \sum_{l}^{L}\sum_{t}^{T_l}\frac{\partial \log p\left(a \mid s_{t,l}; W_{RL}\right)}{\partial W_{RL}} R_{t,l} \tag{9}$$

where L is the size of image patches that used at one iteration.

IV. Experimental Results

A. Dataset and Training Setting

The dataset in our system is collected from the Changsha-Zhuzhou high-speed rail line in China. It has 1596 catenary BS images that cropped from global catenary images. Each BS image is annotated with a tight box of ground-truth position. As BSs component have different sizes in global images. The width (or height) of each cropped BS image is 2.5 times of the truth width (or height) of the BS component, which makes the ADNET have proper regions for researching. We use 1020 images for training and 576 images for testing.

To train ADNET, we set the learning rate to 0.0001 for conv1-conv3 and 0.001 for fully-connected layers (fc4-fc7), momentum to 0.9, weight decay to 0.0005, and mini-batch size to 64. The epoch numbers of training iteration are set to be 200 for SL and 300 for RL, respectively. The experimental environment of reinforcement learning is as follows: Linux Ubuntu 14.04, MATLAB 2017a, CUDA 8.0 and NVIDIA GTX1080Ti GPU with 11 GB memory.

B. Experiment and Analysis

We evaluated our method on the built dataset. As the initial localized boxes can be distributed in any position around the BS component. Therefore, we apply a Gaussian function on the ground-truth BS position to produce the initial boxes. In the testing, the agent ADNET takes T consecutive actions (steps) in each image to refine the initial boxes. Here, T is set to be 20. Some selected localization results are shown in Figs.5-8. Performances of the proposed method over the entire dataset are summarized in Table 1. Detailed experiment analyses are as follows.

1) Visualization and analysis of RL-based refinement.

The dynamic processes of localizations by the agent are displayed in Figs.5-7, where the white box is the initial box and its color gets greener gradually after taking an action until reaching the final pink box. We divide these processes into three types, namely *Type A*, *Type B* and *Type C*, which are shown in Figs.5-7 respectively.

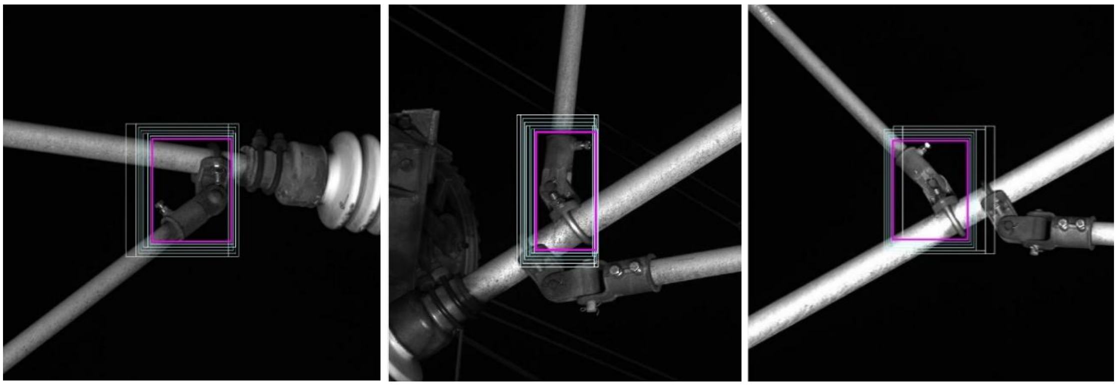


Fig. 5 Localization refined by RL agent for *Type A* cases. **Left**: case A1. **Middle:** case A2. **Right:** case A3.

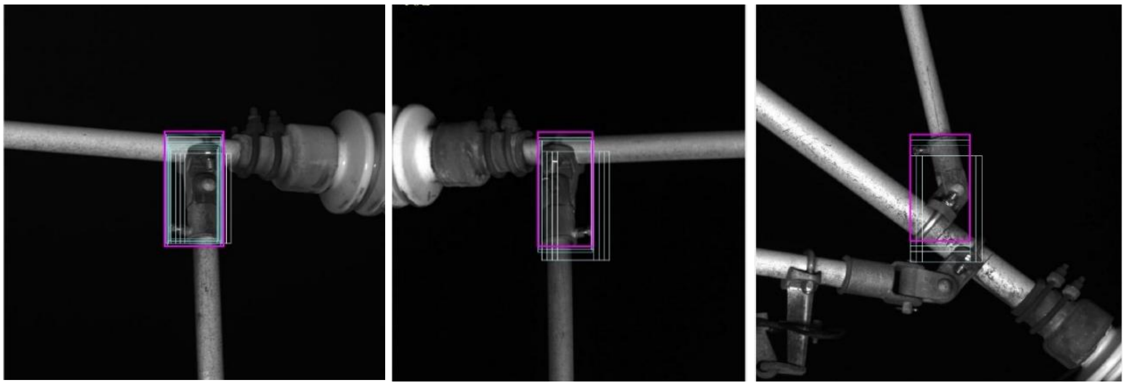


Fig. 6 Localization refined by RL agent for *Type B* cases. **Left**: Case B1. **Middle:** Case B2. **Right:** Case B3.

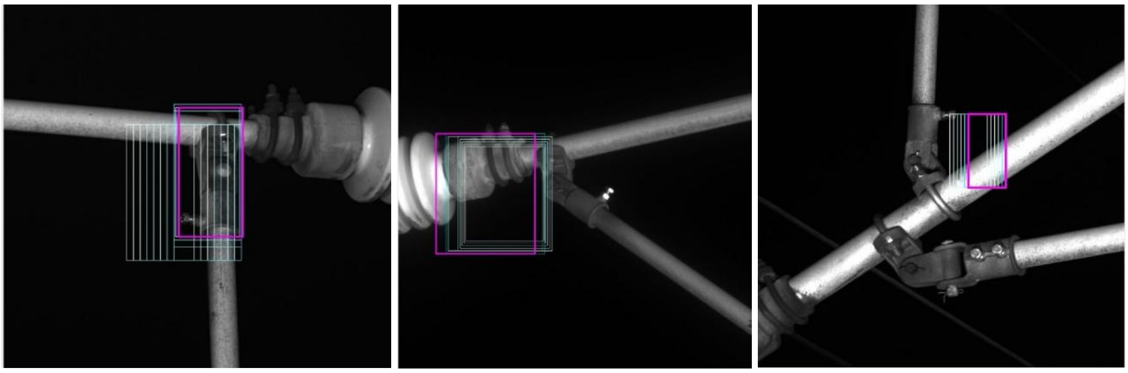
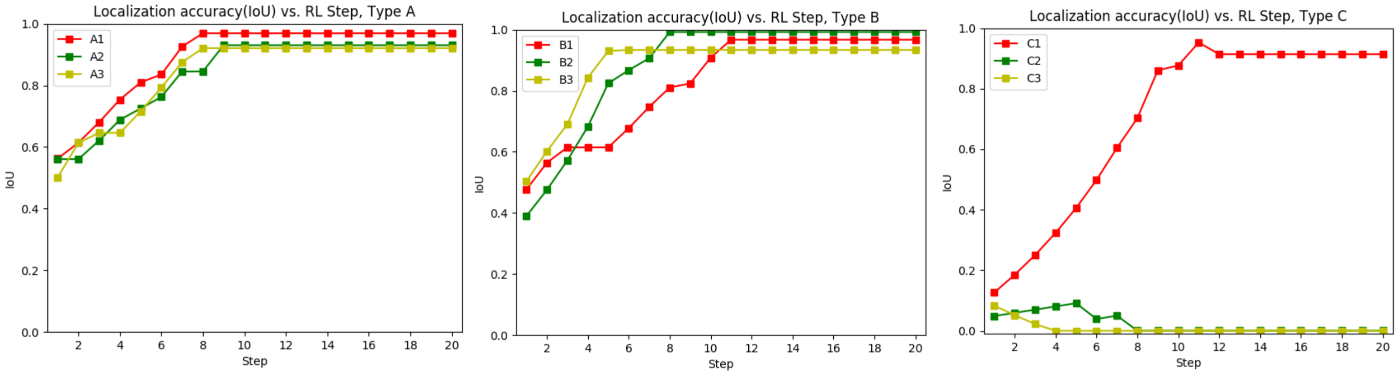


Fig. 7 Localization refined by RL agent for *Type C* cases. **Left**: case C1. **Middle:** case C2. **Right:** case C3.



(a) (b) (c)

Fig. 8 Localization accuracy (IoU) changes for three different types of cases in the testing. (a) IoU changes in *Type A* cases. (b) IoU changes in *Type B* cases. (c) IoU changes in *Type C* cases

In Fig. 5, when the *Type A* initial box is larger than the BS component, and it may contain unnecessary information from other components, such as case *A2* and case *A3*. The agent can adjust these initial boxes closer to BS components mainly by taking actions including *moving*, *scale down*, *shorter* and *thinner*. Fig. 8(a) shows the localization accuracy (IoU) changes of *Type A* cases when the step grows. The IoUs are getting larger even the initial boxes are with unnecessary information. In Fig. 6, the initial boxes of *Type B* cases are partly overlapped with the BSs. Particularly, the initial box of case *B3* contains not only the incomplete BS, but also unnecessary information from near components. However, the RL-learned agent can still move these initial boxes closer to the BSs. The accuracy changes of each case are shown in Fig. 8(b). Both *Type A* and *Type B* are cases that can be correctly refined by the proposed method. As for the speed of refinements, although we set the total steps for each image to 20, most of the successful cases take less than 12 steps to reach the final locations, as shown in Fig. 8 (a) and (b).

As we use the Gaussian function to randomly produce the initial boxes, some initial boxes have fewer overlaps with BS component, as the *Type C* cases showed in Fig. 7. The initial (first step) IoUs of *Type C* cases can be observed in Fig. 8(c). They are less than 0.15 at the beginning. In the experiment, many *Type C* cases led to failed BS localizations and the boxes are moved further away from the BS, as shown by the case *C2* and case *C3* in Fig. 7. However, there are still some cases like case *C1* can successfully adjust boxes closer to the BS component position. We conjecture that case *C2* and case *C3* are failed because very few features of BSs are contained within their initial boxes, while case *C1* still contains some useful features of BS, which can be observed from the IoUs at the first step in Fig. 8(c).

2) *Quantitative analysis of RL-based refinement result.*

The localization accuracy metric for the overall dataset is the widely used *Recall* [24]. The $\mathbf{Recall}_{T_{IoU}}$ **(IoU>0.5)** means the proportion of having a IoU larger than the threshold T_{IoU}. The speed metric adopted is FPS (frame/second). Quantitaive localization performances of the proposed method is shown in Table 1.

Table 1. Localization accuracy improvement by RL

Method	**Recall$_{0.5}$ (IoU>0.5)**	**Recall$_{0.8}$ (IoU>0.8)**	**Recall$_{0.9}$ (IoU>0.9)**	**Improved proportion**	**FPS**
Gaussian Initial	96.4%	19.9%	3.3%	--	--
Gaussian Initial with RL (ours)	92.0%	**79.2%**	**65.1%**	**81.8% (471/576)**	**6.13**

Table 1 shows that the Recall$_{0.5}$ is slightly decreased compared with the initial value, because few boxes are moving away from the BS positions. However, the Recall$_{0.8}$ is increased from 19.9% to 79.2%, and the Recall$_{0.9}$ is increased from 3.3% to 65.1%. Overall, among 576 test images, 471 images' IoUs become larger, which means 81.8% test images get better localizations. It indicates that the proposed RL method can adjust most of boxes closer to the BS positions and improve the localization accuracy. Besides, the RL agent takes only 0.163s (1/6.13) for each test image, which consumes very little time in applications.

V. Conclusion

This paper proposes a novel approach for improving the localization accuracy of BS (Brace Sleeve) in railway catenary systems. Differing from the existing localization strategies in railway that localize objects following a single structured prediction model, the proposed method adopts a dynamic searching strategy. We investigate one method that adopts RL to train an agent to generate an improved bounding box for BS localizations through a sequence of defined action transformations. Experimental results using real-life inspection images show that the proposed method can adequately and effectively refine the localization from a coarse-localized input. Nevertheless, there are still some further improvements to be conducted:

(1) As the case *C2* and case *C3* shown in Fig. 7 and the Recall$_{0.5}$ comparison shown in Table 1, there are some failed cases with the bounding boxes moving away from the BSs. This issue should be further researched and improved.

(2) When a BS is not localized by our agent, exploring and implementing a new searching strategy in the failed image will also reduce the number of failed cases.

(3) Except for the policy gradient-based reinforcement learning method [23] used in this paper. Elements from applications of reinforcement learning in other fields, such as robotics and control [25, 26], can also be considered for the dynamic monitoring of catenary systems.

References

[1] S. Gao, Z. Liu, L. Yu., "Detection and monitoring system of the pantograph-catenary in high-speed railway (6C)," in 7th *International Conference on PESA*, Hongkong, pp. 779-788, 2017.

[2] Y. Han, Z. Liu, X. Geng, and J. P. Zhong, "Fracture detection of ear pieces in catenary support devices of high-speed railway based on HOG eatures and two-dimensional Gabor transform," *J. China Railway Soc*., vol. 39, no. 2, pp. 52–57, 2017.

[3] J. Zhong, Z. Liu, G. Zhang, and Z. Han, "Condition detection of swivel clevis pins in overhead contact system of high-speed railway," *J. China Railway Soc*., vol. 39, no. 6, pp. 65–71, Jun. 2017.

[4] H. Fan, P. Cosman, Y. Hou, et al, "High-speed railway fastener detection based on a line local binary pattern," *IEEE Signal Processing Letters*., vol. 25, no. 5, pp. 788-792, 2018

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, Jun. 2015.

[6] Z. Cai and N. Vasconcelos, "Cascade R-CNN: delving into high quality object detection," *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 6154-6162, 2018.

[7] Z. Liu, L. Wang, C. Li, et al., "A high-precision loose strands diagnosis approach for isoelectric line in high-speed railway," *IEEE Transactions on Industrial Informatics,* 10.1109/TII.2017.2774242.

[8] Z. Liu, Y. Lyu, L. Wang, et al., "Detection approach based on an improved faster RCNN for brace sleeve screws in high-speed railways," *IEEE Transactions on Instrumentation & Measurement*, vol. 69, no. 7, pp. 4395-4403, 2020.

[9] G. Q. Kang, S. B. Gao, L. Yu, and D. Zhang, "Deep architecture for high-speed railway insulator defect detection: denoising autoencoder with multitask learning," *IEEE Transactions on Instrumentation and Measurement*, DOI: 10.1109/TIM.2018.2868490.

[10] Z. Liu, K. Liu, J. P. Zhong, Z. Han and W Zhang, "A high-precision positioning approach for catenary support components with multi-scale difference," *IEEE Transactions on Instrumentation & Measurement*, vol. 69, no. 3, pp. 700-711, 2020.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779-788.

[12] J. Chen, Z. Liu, H. Wang, et al., "Automatic defect detection of fasteners on the catenary support device using deep convolutional neural network," *IEEE Transactions on Instrumentation and Measurement*, 10.1109/TIM.2017.2775345.

[13] D. Silver, A. Huang, C. J. Maddison, et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, 529(7587): 484–489, 2016.

[14] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning,". *CoRR*, abs/1509.06461, 2015.

[15] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *In ICML*, 2014.

[16] F. Ruelens, B.J. Claessens, S. Quaiyum, et al., "Reinforcement learning applied to an electric water heater: From theory to practice," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3792-3800, 2018

[17] V. Mnih, K. Kavukcuoglu, D. Silver, et al., Playing atari with deep reinforcement learning. *arXiv preprint* arXiv:1312.5602, 2013.

[18] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," *IEEE International Conference on Computer Vision*, pp. 2488–2496, 2015.

[19] S. Yun, J. Choi, Y. Yoo, et al., "Action-Decision networks for visual tracking with deep reinforcement learning," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2711-2720, 2017.

[20] M. Bellver, X. Giro-I-Nieto, F. Marques, et al., "Hierarchical object detection with deep reinforcement learning," *arXiv preprint arXiv:1611.03718*, 2016.

[21] S. Liu, D. Huang and Y. Wang, "Pay attention to them: deep reinforcement learning-based cascade object detection," *IEEE Transactions on Neural Networks and Learning Systems*, PP (99):1-13. 2019.

[22] K. Simonyan, and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv: 1409.1556*, 2014.

[23] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, 8(3-4):229–256, 1992.

[24] B. Michael, and G. Fredric, "The relationship between recall and precision," *Journal of the American Society for Information Science*, vol. 45, no. 1, pp. 12-19, 1994.

[25] Y. Pane, S. Nageshrao, J. Kober, et al., "Reinforcement learning based compensation methods for robot manipulators," *Engineering Applications of Artificial Intelligence*, 78:236–247, 2019.

[26] T. de Bruin, J. Kober, K. Tuyls, et al., "Integrating state representation learning into deep reinforcement learning," *IEEE Robotics and Automation Letters*, 3(3):1394–1401, 2018.