A Monitoring and Forewarning System for Rice Planthoppers

Wan-Ru Lin, Kuei-Tso Lee, and Sheng-Jyh Wang
Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan
Email: fuj30089@gmail.com

Ming-Hsin Lai
Taiwan Agricultural Research Institute, COA

Po-Hsun Chen
Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan

***Abstract*—We develop a monitoring and forewarning system to detect planthoppers in paddy fields. Our detection algorithm consists of two stages. At the first stage, we extract the main paddy in the middle of an image by some traditional image processing techniques. At the second stage, we use a convolutional neural network to detect planthoppers within the extracted region. Our detection model is revised from the Single Shot MultiBox Detector (SSD). The original SSD model usually misrecognizes reflected light as planthoppers since a lot of background information has been discarded in the max pooling layers of the SSD model. To solve this misrecognition problem, we propose a new kind of pooling-- Local Difference Pooling. This proposed method greatly improves the performance of planthopper detection to achieve 89.38% precision and 91.93% recall.**

***Index Terms*—planthopper, detection, SSD, local difference pooling**

I. Introduction

Planthoppers are a kind of rice pests that cause huge damage to the crops in several Asia countries. These pests may devastate crops in a short time and even carry some disease, causing the damage of paddy both morphologically and physiologically. Planthoppers are so small that it is difficult to observe them. Moreover, they reproduce extremely fast and spread quickly to a large area. If we do not take appropriate actions in time, they may cause damage in an explosive way. Although spraying pesticides can kill most planthoppers, it will also pollute the land, and the polluted rice will be harmful for human health. Therefore, it is urgent to develop a monitoring and forewarning system that is capable of detecting planthoppers in the paddy fields in time. With this system, farmers only need to spray pesticides when the number of planthoppers exceeds a pre-selected threshold. The reduced usage of pesticides would be helpful to human health.

Manuscript received April 15, 2019; revised August 22, 2019.

Recently, many methods have been proposed to recognize rice planthoppers in paddy fields. Zou and Ding [1] proposed a recognition algorithm to achieve real-time identification, consisting of single-threshold segmentation and wavelet-based pest edge extraction. YAO Qing *et al.* [2] proposed a three-layer model to detect whiteback planthoppers. This model consists of Harr feature based AdaBoost classifier, HOG feature based SVM classifier, and threshold-based judgment. This method achieves an 85.2% detection rate and a 9.6% false detection rate. Sarin Watcharabutsarakham *et al.* [3] proposed a method to monitor brown planthoppers which combines an image classifier with a mobile application. The classifier is modeled as a support vector machine based on color and local texture features. This method achieves an 83% accuracy rate. Tsai *et al.* [4] proposed a simple model with the following steps: (1) find the region of interest, (2) do color analysis, and (3) recognize rice planthoppers with the decision-tree algorithm.

However, those aforementioned methods have difficulty handling particular conditions, such as occlusion, lighting variations, and deformation. To deal with these problems, we will use a Deep Convolutional Neural Network (DCNN), which can extract high level features from the image to detect planthoppers.

Recently, there are two main streams using CNN models for object detection. The first approach uses Region Proposal Network (RPN) (Fast R-CNN [5]; Faster R-CNN [6]) to find ROIs (Region Of Interest) and feed in region proposals one by one to the classification network. The second approach discards ROI generation and makes prediction on class and location at the same time, such as SSD [7] and YOLO [8]. The second approach is typically faster than the first approach since there is no need to wait for the preparation of ROIs.

Comparing Faster R-CNN, SSD and YOLO with the same base network, the last two are ten times faster than Faster R-CNN. Besides, in terms of accuracy, SSD is better than the other two in Pascal VOC competition. Hence, we select SSD as our detection model and revise it properly to meet our requirements. For the base network, we use VGG-16 [9].

doi: 10.18178/joaat.6.3.180-186

The remainder of this paper is organized as follows. Section 2 introduces the source of our data and the methods we use in the experiments. Section 3 shows the results of our methods with different architectures and discusses some particular situations of our model. The last section draws the conclusion of this paper.

II. Materials and Methods

A. Image Data

This project is cooperated with Industrial Technology Research Institute (ITRI) and Taiwan Agricultural Research Institute. ITRI is responsible for setting up cameras in the paddy fields and collecting image data. Our dataset has 1152 training images and 50 testing images. Fig. 1 shows an example of the image data.

Figure 1. Example of image data. There are three different resolutions of images in the dataset: 3456x4608, 3672x4896, and 2592x4608 pixels (height x width).

In the image data, the planthoppers' height ranges from 23 to 214 pixels, while its width ranges from 12 to 152 pixels. The range is quite large due to the varying distance between the camera and the plant. Since planthoppers are relatively small in the image, we crop the image into smaller image patches before feeding them to the detection network. Moreover, our detection model is based on the SSD network, whose input size is fixed to be 300x300 pixels. Hence, we match the size of the image patches to this 300x300 size (as shown in Fig. 2). We also perform data augmentation to significantly increase the size of our dataset (as listed in Table I). The details about the preprocessing for training and testing images will be introduced in Section II.C and II.D.

Figure 2. Examples of patches cropped from the original image. Size: 300 x 300 pixels.

TABLE I. The Size of Our Dataset

Dataset	Number
Training images	1152
Testing images	50
Cropped image patches (training)	33346
Cropped image patches (testing)	32928

B. Model

The SSD network adds six extra layers to the end of the VGG-16 network. These layers decrease in size and the layers with smaller sizes are responsible for detecting larger objects in the original image. There are anchor boxes of different shapes and sizes in order to detect objects of different shapes. Choosing appropriate aspect ratios of anchor boxes may improve the accuracy of the model. To meet our requirements, we revise the original SSD model, which will be introduced in the following paragraph. Fig. 3 shows the architecture of the revised SSD model.

The original SSD model can classify 21 classes. However, our dataset only includes two different classes (planthopper/non-planthopper). It is not necessary to use such a complicated model for binary classification. Hence, we reduce the number of filters to 1/4 of the original ones. Moreover, since planthoppers are typically small in the image, we remove the last two layers which are designed to detect larger areas in the image.

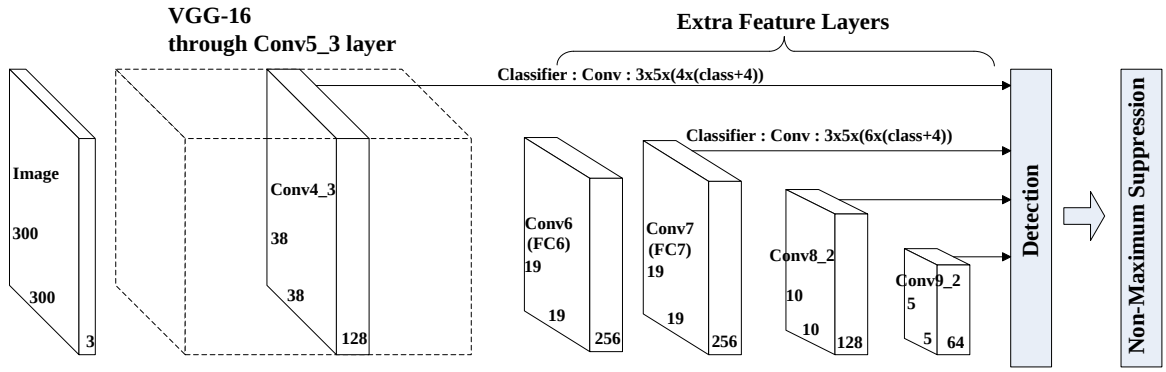


Figure 3. Revised SSD model.

To decide the aspect ratios of anchor boxes, we examine the shape of planthoppers in the whole dataset. Most planthoppers are thin and tall and their height-width ratio is around 2, as shown in Fig. 4. We use k-means clustering (with 3 clusters) to analyze the distribution of the planthopper's height-width ratio and obtain the three cluster centers with the aspect ratio being 1.3, 2 and 2.75, respectively. In our model, the aspect ratios are set to these three values.

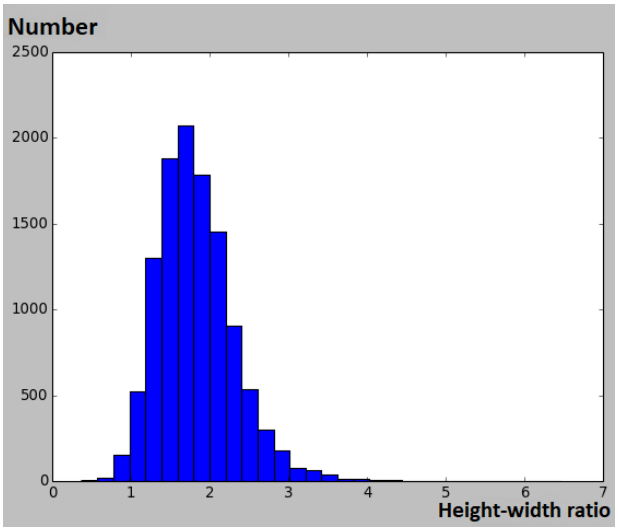


Figure 4. Histogram of the height-width ratio of planthoppers. Most of the planthoppers are thin and tall and their height-width ratio is around 2.

Finally, we observe that the receptive fields of the predictive convolutional filters are actually smaller than their corresponding anchor sizes. Hence, we enlarge the filter size from 3x3 to 5x3. With the new size, the receptive field can contain more information within the anchor box.

(a) Light reflection (b) Blurred planthoppers

Figure 5. Light reflection and blurred planthoppers.

C. *Local Difference Pooling*

With SSD, the major challenge is that our model may sometimes misrecognize light reflection as planthoppers since blurred planthoppers and reflected light have similar features and may cause confusion in detection. On the contrary, humans can identify their differences by observing the surrounding environment of the objects. If an object locates on a green leaf or stem, it is more likely to be a planthopper; otherwise it may be a false alarm, as shown in Fig. 5. Typically the color difference between a planthopper and its background is larger than the color difference between a false alarm and its background. Based on this observation, the inclusion of background information could be a useful way to distinguish blurred planthoppers from light reflection.

In the revised SSD model, we propose the Local Difference Pooling which calculates the local difference between the average value and the max value, as expressed in Equation (1) where α is set to 1/4.

$$\text{Local Difference Pooling}=\text{Average Pooling}-\alpha\times\text{Max pooling} \quad (1)$$

In order to reduce feature map size, Max Pooling is commonly used in CNN models. However, since max pooling only keeps the largest value and discards the rest, background information is lost after Max Pooling. On the other hand, even though Average Pooling can retain the background information, it does not show the color variation in a small region. To meet our requirements, we propose the Local Difference Pooling which calculates the local difference between the average value and the max value.

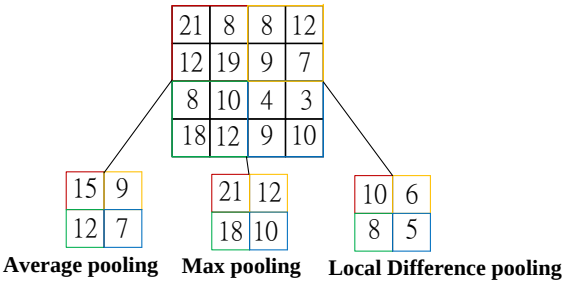


Figure 6. Illustration of average pooling, max pooling, and local difference pooling.

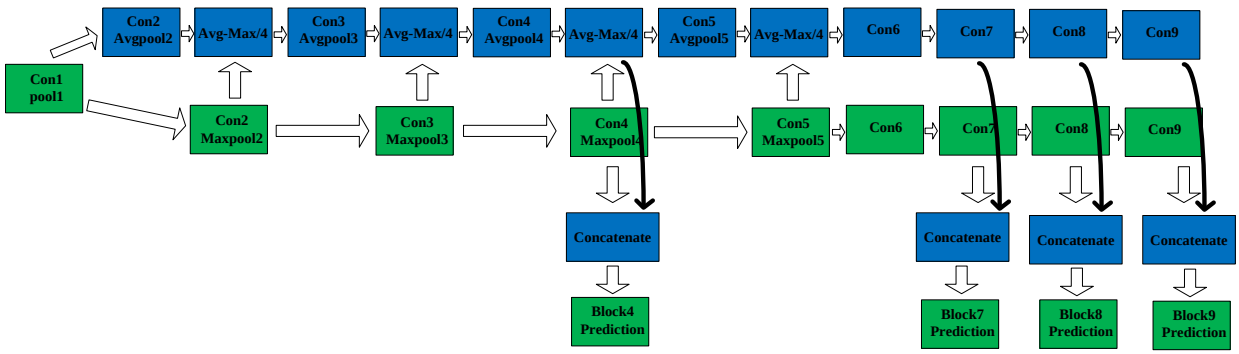
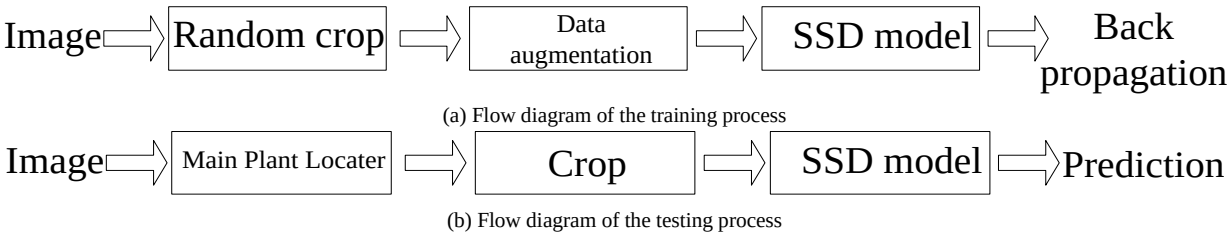


Figure 7. Improved SSD structure with local difference pooling. Green blocks are the original SSD layers. Blue blocks are new layers utilizing Local Difference pooling. Feature maps are combined to make prediction. In the improved structure, both foreground and background information can affect the predictive results.

Fig. 6 shows an example of different pooling approaches. The value of Local Difference Pooling is high if the small region contains prominent color contrast. This method effectively retains the background information and enables our model to distinguish blurred planthoppers from light reflection. Fig. 7 shows the Improved SSD structure with Local Difference Pooling which utilizes both foreground and background information.



(a) Flow diagram of the training process

(b) Flow diagram of the testing process

Figure 8. Flow diagram.

D. Training

As mentioned in Section 2.1, the image data should be cropped into small patches (300 x 300 pixels) to match the input size of the SSD network. The preprocessing procedure will be explained in details in the following paragraphs.

Fig. 8(a) shows the flow diagram of the training process. First, the image data are cropped into the size of 350 x 350 or 512 x 512 pixels. These patches are cropped in an overlapping way with a small margin (Fig. 9). Hence, planthoppers that locate on a border line of one patch may appear in another patch. In our setting, a planthopper in an image patch will be treated as a positive sample if more than 70% of its body appears in the patch. This setting will be helpful for detecting planthoppers that are partially occluded by leaves or stems. Finally, the patches with no planthoppers are discarded because they largely increase the number of negative samples.

Figure 9. Image with overlapped patches. The red lines and black lines denote different patches, which are overlapped with a small margin. We only draw some of the patches for illustration.

In the random crop module, patches are randomly cropped for data augmentation and then resized to 300 x 300 pixels. For random cropping, there are some constraints: (1) the area of a planthopper should not be smaller than two thirds of its original size; (2) the height-width ratio of a cropped box should range from 0.9 to 1.1; and (3) the cropped area should be larger than a half of the original area. In our system, we randomly generate a box, which satisfies the constraints (2) and (3), and randomly crop the patch. If the cropped patch violates the constraint (1), the patch is discarded and the system will do croping again.

In the data augmentation module, we do some color transformation and horizontal flip to create more variations of the training samples. The color transformation process contains random changes of brightness, saturation, contrast, and hue.

After the preprocessing process, these image patches are sent to the SSD network for training. The SSD network is trained by the error back-propagation algorithm with mini-batch of size 32. We choose Adam optimizer and set the learning rate to 0.001 initially. Learning rate is decreased every 2000 iterations with the decay rate 0.85. After about 30k iterations, the loss stops decaying and we add negative samples back to learn the features of non-planthopper. Finally, we finish the training at about 50k iterations.

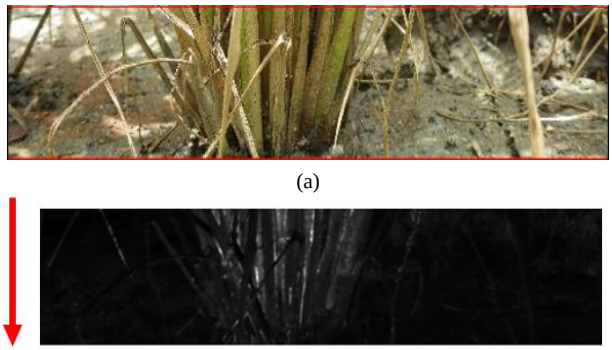
E. Main Plant Locator

For testing, we only concern about planthoppers on the main plant. Hence, we identify the location of the main plant first and discard the remaining area. The main plant is cropped into patches of 300x300 pixels with slight overlapping and these patches are sent to the SSD network for testing. Fig. 8(b) shows the flow diagram of the testing process. In the following paragraphs, we introduce the identification of the main plant region.

Figure 10. The area for stem width estimation. The region is set to range from the 0.2 x height to 0.4 x height of the image.

Observing our testing data, the difference between a main plant and the background is quite obvious in the lower portion of the image, as colored in red in Fig. 10. Within the red region, we first estimate the stem width of the main plant.

By examining each of the RGB channel in the red region, the green component is apparently higher than the blue component on the plant. On the contrary, the green component may be lower than the blue component in the background region. Hence, we calculate the division of the green component to the blue component of each pixel over the red region, as shown in Fig. 11(b).

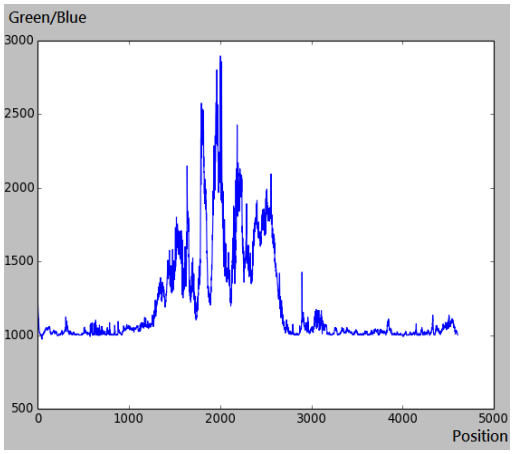


(a)

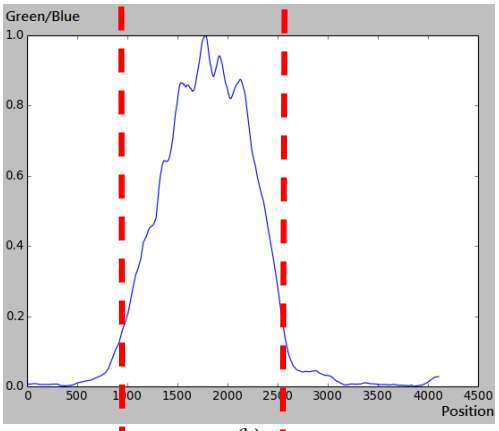
(b)

Figure 11. (a) RGB image (b) The division of the green component to the blue component of each pixel.

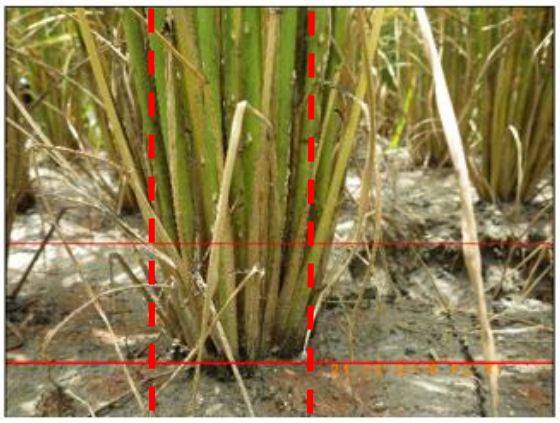
Next, we calculate the summation along the y axis of the G/B image, as shown in Fig. 12(a). Then moving average is applied to smooth the summation result. The smoothed curve is further normalized to range from 0 to 1, as shown in Fig. 12(b). The value is high when there is a plant and is low for the background. By setting a threshold, we can approximate the width of the stem. In our observation, it is appropriate to select 0.2 as the threshold. Next, the plant has a larger width on the upper portion of the plant. Hence, we approximate the width of the plant to be 1.5 times the width of the stem and Fig. 13 demonstrates the final prediction of the main plant area, which is indicated by the dotted red line.



(a)



(b)



(c)

Figure 12. Predict the width of the stem. (a) The result of summation along the y axis of Fig. 11(b). (b) After moving-average smoothing and normalization. (c) We set the threshold at 0.2 to approximate the width of the stem.

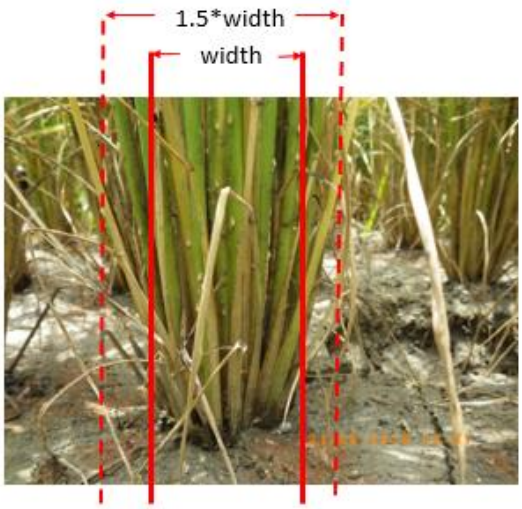


Figure 13. Illustration of the estimated main plant region.

III. RESULTS

After testing, it is considered a true positive if the prediction and the ground truth have their IOU value bigger than 0.5. It is a false alarm if the prediction is

wrong. It is a miss if a planthopper is not detected. The number of mistakes equal to the number of false alarms plus the number of miss. Here we use precision and recall to evaluate the performance of our algorithm. Precision is the number of true positives over the number of predictions. Recall is the number of true positives over the number of planthoppers in the image.

In our test dataset, there are 785 planthoppers in total. After testing, the total number of hit (true positives), false alarms, miss, and mistakes are listed in Table II. We evaluate precision and recall of every single image, and obtain the average value respectively based on the entire test dataset.

We compare different architectures in Table II, in which each architecture is revised from the previous one except the last one. In the baseline model, we detect planthoppers using the original SSD model without any modification. The false alarm is extremely high, as shown in Fig. 14. After analyzing the shape and size of planthoppers, we discard the last two layers of the SSD model (Model 2) and choose suitable aspect ratios of the anchors based on the k-means clustering result (Model 3). The false alarm rate and missing rate are reduced but the effect is not significant. We reduce the number of filters to 1/4 of the original number (Model 4) to save the training time. It is reasonable that the false alarm rate reduces while the missing rate increases.

On the other hand, it makes a dramatic improvement as we utilize the Local Difference Pooling (Model 5). The number of false alarms reduces from 177 to 72. Besides, as we enlarge the filter size from 3x3 to 5x3 (Model 6) to obtain our final model, our system can achieve the precision value of 89.38% and the recall value of 91.93%. Comparing Fig. 14 (the result of the baseline model) with Fig. 15 (the result of the final model), it can be observed that the final model can recognize almost every planthopper with only one false alarm. However, it may be doubted that the missing rate of the baseline model is lower than that of the final model. Actually, for the baseline model, it learns to detect not only planthoppers but also everything similar to planthoppers. Hence, it can certainly detect planthoppers with lower missing rate, but at the same time with an excessive false alarm rate. In comparison, our model has achieved almost 90% precision and recall with only a small number of mistakes.

The bottom row of Table II shows the performance of our system if we train our model without data augmentation. In this case, the precision value drops significantly for the testing data.

We have also evaluated our model without the use of the Main Plant Locater. As shown in Fig. 16, massive false alarms occur in the background. Since the purpose of our system is to estimate the number of planthoppers on the main plant, the use of the Main Plant Locater is preferred.

TABLE II. PERFORMANCE OF DIFFERENT ARCHITECTURES. EACH ARCHITECTURE IS REVISED FROM THE PREVIOUS ONE, EXCEPT MODEL 7. THE BASELINE MODEL DENOTES THE SSD MODEL WITHOUT ANY MODIFICATION. "TIME" DENOTES THE AVERAGE TIME TO TEST ONE IMAGE

Model	Model Description	Hit	False alarm	Miss	Mistake	Precision	Recall	Time(s)
1	Baseline	730	212	55	267	76.18	93.55	5.43
2	1 + Discard the last two layers	735	204	50	254	77.22	94.63	5.15
3	2 + Choose suitable aspect ratios	739	191	46	237	78.29	94.18	5.07
4	3 + Reduce the number of filters	718	177	67	244	79.29	92.01	4.57
5	4 + Use Local Difference Pooling	686	72	99	171	89.97	87.00	5.26
6	5 + Enlarge the filter size from 3x3 to 5x3	723	79	62	141	89.38	91.93	7.94
7	Baseline Without data augmentation	694	1088	91	1179	37.22	87.91	5.43

Figure 14. The test result of the baseline model. Red boxes are the prediction results, and blue boxes denote the ground truth. A total of 9 planthoppers are detected. There are 2 misses and 8 false alarms.

Figure 15. The test result of Model 6. Red boxes are the prediction results, and blue boxes denote the ground truth. A total of 10 planthoppers are detected. There are only 1 miss and 1 false alarm.

Figure 16. The test result of the baseline model but without the Main Plant Locater. Red boxes are the prediction results, and blue boxes denote the ground truth. Several false alarms occur in the background.

IV. Conclusion

In this paper, we propose a monitoring system to detect planthoppers in paddy fields. Our system consists of two main stages. At the first stage, we detect the location of the main plant and discard the background. At the second stage, we revise the original SSD model by proposing the use of Local Difference Pooling to properly retain the color contrast information between the foreground objects and the background. The simulation results demonstrate that the use of Local Difference Pooling does enable our model to better distinguish blurred planthoppers from light reflection. The new model dramatically improves our detection model to achieve the precision value of 89.38% and the recall value of 91.93%.

References

[1] X. G. Zou and W. M. Ding, "Design of processing system for agricultural pests with digital signal processor," *Journal of Information & Computational Science*, vol. 15, pp. 4575-4582, 2012.

[2] Y. Qing, D. X. Xian, Q. J. Liu, B. J. Yang, G. Q. Diao, and J. Tang, "Automated counting of rice planthoppers in paddy fields based on image processing," *Journal of Integrative Agriculture*, vol. 13, no. 8, pp. 1736-1745, August 2014.

[3] S. Watcharabutsarakham, I. Methasate, N. Watcharapinchai, W. Sinthupinyo, and W. Sriratanasak, "An approach for density monitoring of brown planthopper population in simulated paddy fields," *IEEE Computer Science and Software Engineering*, July 2016.

[4] T. Tsai, T. Lee, and P. Chen, "The ROI of rice planthopper by image processing," in *Proc. IEEE International Conference on Applied System Innovation*, 2017.

[5] R. Girshick, "Fast r-cnn," in *Proc. IEEE International Conference on Computer Vision*, 2015.

[6] S. Ren, *et al.*, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, 2015.

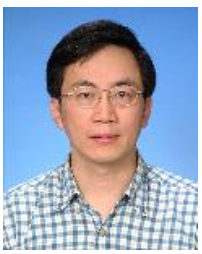
[7] W. Liu, *et al.*, "Ssd: Single shot multibox detector," in *Proc. European Conference on Computer Vision*, 2016.

[8] J. Redmon, *et al.*, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

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

Wan-Ru Lin received the B.S. degree in eletrical engineering and computer science from National Chiao Tung University, Taiwan, in 2015, and the M.S. degree in electrical engineering from National Chiao Tung University, Taiwan, in 2017. Her research interests include image recognition and machine learning.

Kuei-Tso Lee received B.S. degree in electrical engineering in 2014, and M.S. degree in communication engineering in 2017 from National Chiao Tung University, Taiwan. He is currently pursuing the Ph.D. degree in electronics engineering in National Chiao Tung University. His research interests are image processing and machine learning.



Sheng-Jyh Wang received the B.S. degree in electronics engineering from National Chiao Tung University, Taiwan, in 1984, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, USA, in 1990 and 1994, respectively. He is currently a Professor in the Department of Electronics Engineering, National Chiao Tung University, Taiwan. His research interests are in the areas of image processing, video processing, video surveillance, and image analysis.