https://doi.org/10.2352/ISSN.2470-1173.2019.9.IMSE-360

Hybrid Image-based Defect Detection for Railroad Maintenance

Gaurang Gavai, Hoda Eldardiry: Palo Alto Research Center, Palo Alto, California, USA
Wencheng Wu, Beilei Xu: Goergen Institute for Data Science, Rochester, NY, USA
Yoshihiro Komatsu: East Japan Railway Company, Tokyo, Japan
Shigeki Makino: Brierley+Partners Japan, Tokyo, Japan

Abstract

In this paper, we describe a novel method for image-based rail defect detections for railroad maintenance. While we developed the framework to handle a broad range of defect types, in this paper we illustrate the approach on the specific example of detecting cracks located on fishplates connecting rails in images. Our algorithm pipeline consists of three major components: a preprocessing and localization module, a classification module, and an on-line retraining module. The pipeline first performs preprocessing tasks such as intensity normalization or snow pixel modification to better prepare the images, and then localizes various candidate regions of interest (ROIs) where the defects of interest may reside. The resulting candidate ROIs are then analyzed by trained classifier(s) to determine whether the defect is present. The classifiers are trained off-line using labeled training samples. While the system is being used in the real-world, more samples can be gathered. This gives us opportunity to refine and improve the initial models. Experimental results show the effectiveness of our algorithm pipeline for detecting fishplate cracks as well as several other defects of interest.

Introduction

Infrastructure maintenance is a crucial part of ensuring high quality and safe transportation to customers of railway companies. Surveys have shown that various rail organizations spend up to $85000 per kilometer a year on such maintenance [1]. This is mainly due to railroad operators employing a variety of different solutions to ensure there are multiple fail-safes in place to capture defects in their infrastructure in an effective and timely fashion. Towards this end, the Palo Alto Research Center (PARC) has been collaborating with the East Japan Railway Company (JR) to develop condition-based maintenance technologies for their assets.

Figure 1 shows examples of some of these defects including cracks that develop on the fish plates that join two rails and broken bondwires that conduct electricity between them. Examples of these defects shown in Figure 1 are demarcated by red boxes. The aim of this effort was not only to ensure greater redundancy in detecting defects but also add an automated detection layer that could perhaps find defects tough to find with the human eye. To this end, we investigated the use of a blend of traditional computer vision and deep network based features to evince these hard to find defects.

This novel hybrid approach to detecting defects is what we wish to highlight in this paper, particularly in the case of detecting cracked fishplates. In the dataset section we outline some related algorithms and approaches to ours in literature; this section also discusses some characteristics and statistics of the images dataset we worked with and how it informed some of the choices we made. Thwe defect detection section covers the rail surface segmentation, fishplate localization, snow detection and crack detection algorithms that are all part of the automated pipeline, and the conclusion discusses our learnings from this work and our path forward.

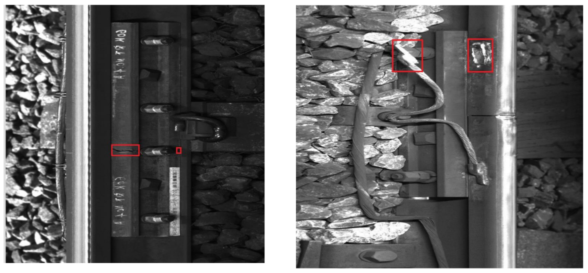


Figure 1. *Examples of cracked fishplate and broken bondwire.*

Related Work

Computer vision aided rail defect detection has been studied and researched in many different forms. Numerous traditional computer vision based approaches exist to detect particular defects such as cracks on railroad tracks [2] and marks on the railroad surface [3]. Several researchers have explored using other methods to detect defects such as hardware LED assemblies [4], geometrical analysis [5], and condition based-maintenance using detailed models of parts [6].

Deep networks have gained popularity as solutions to image classification and anomaly detection problems in the recent years. In cases where the number of defect examples is large enough, some groups have attempted to use convolutional neural networks for the detection of rail defects [7] [8].

Tangentially, hybrid approaches of traditional computer vision and deep networks such as ours have been used in different combinations to solve image processing challenges [9]. However our approach is novel not only in the make-up of algorithms we use in it but also in the application area of condition-based maintenance and defect detection.

Dataset

The dataset we used was provided by the East Japan Railway Company (JR). It consisted of 2 different types of images:

- Contiguous grayscale images of the track taken from 3 views - directly above, at an angle to the left and at an angle to the right [G2,G1,G3]
- Contiguous range images [10] taken from the right and the left [R1,R2]

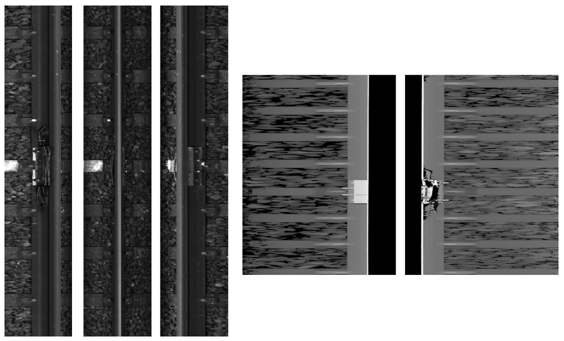


Figure 2. *Example set of [G1,G2,G3] (left) and [R1,R2] (right) of a cracked fishplate.*

Over 10000 image sets were acquired as described above, taken from 5 different lines operated by JR. These sets contained some rail images that had fishplates and some that did not. However, all of these sets did not have any images with *cracked* fishplates. A separate sample of 27 sets were provided that had cracked fishplates with labeled crack locations taken on 7 distinct days (except for one day with 26) for a total of 188 cracked fishplate image sets (188 of G1, G2, G3, R1, and R2) . Figure 2 shows one of these 27 cracked fishplate sets as an illustration. The unbalanced nature of the dataset in terms of positive and negative samples honestly reflects the anomalous nature of these defects. This imbalance motivated a lot of our algorithmic choices and decisions. At the end of the project we were provided with an independent test set of 304 cracked fishplates without crack location labels and 398 non-cracked fishplates to validate our models. The test set was unlabeled since crack detection is evaluated at a plate level granularity.

Defect Detection Pipeline

In this section, we describe our defect detection pipeline for railroad maintenance applications. Our method uses a combination of image processing and machine learning techniques to detect and sometimes quantify the rail defects. We discuss and illustrate this in relation to cracked fishplates; though similar methods have been tested and verified by us on a number of rail defects including black spots, head anomalies, broken bond-wires and many more. The algorithm pipeline flows as follows:

Rail Track Surface Segmentation

The first step of the pipeline is the rail surface segmentation algorithm. Its purpose is to localize the rail surface area within an image. This step would not only aid fishplate localization, but also simplify the task of rail surface defect detection and/or anomaly detection such as head-check, dark spots, wavy rail etc. Although we will only illustrate one of these examples in this paper, it is worthwhile to point out that our rail surface segmentation algorithm is used in multiple places in various rail defect detection tools. This algorithm involves three steps as shown in Fig. 3. The first step is all we need for horizontally locating fishplate. The remaining two steps are used to segment out rail surface in finer resolution for the purposes of detecting other rail surface defects (e.g., head-check, dark spots, and wavy rail). Hence we will focus on the first step. *Estimate rough location via 1-D peak detection*:

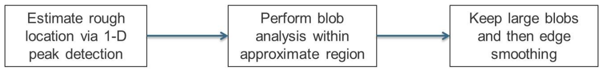

Estimate rough location via 1-D peak detection
Perform blob analysis within approximate region
Keep large blobs and then edge smoothing

Figure 3. *Algorithm flowchart for rail surface segmentation*

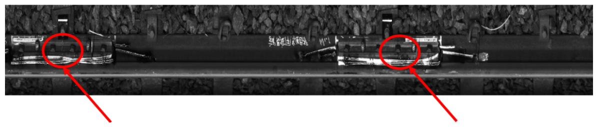


Figure 4. *Example image of cracked fishplate.*

A rough location of the rail surface is identified by peak detection on a one-dimensional (1-D) intensity profile of the image. The intuition is that the rail surface area would appear as a comparatively lighter streak running the length of the image.

Fishplate Localization

Once we've localized the rail, the next step is to use the rail as a bearing to localize the fishplate. Cracks are minuscule features that occupy a very small area in a captured image as shown in Figure 4. Exhaustively searching for a small crack on an entire rail image is not only time consuming but also will increases false positive detections due to the many confusing structures a detector would encounter across an image (e.g., ballast, clips, etc.) so we llocalize the fishplate(s) in an image. Figure 5 shows the multi-step localization process for fishplate detection. We describe this localization process in detail below:

Pre-localization through a threshold-based method

As shown in Fig. 5, a new image is first processed by the threshold-based localization method. It first horizontally localizes the fishplate and then vertically. For the horizontal localization, since the fishplate is placed next to rail tracks and the rail imaging system captures images while traveling along the rail; it is expected that the fishplate would be next to the track. With this in mind, we can now use the rail surface found by the rail segmentation algorithm with a heuristic rule to identify its horizontal location. We use the mean of the edge locations of the rail, derived from the above rail surface segmentation algorithm, as the reference positions. From there, we select an offset to the right or left (depending on whether the image is G3 or G1) of 300 pixels and identify the next 500 pixels as the horizontal location of the fishplate. As you can see in Figure 6, the method works very well since all rail images have a very similar layout. For the next vertical localization step, we developed an algorithm that runs on the thin slices of horizontally located fishplates. The vertical algorithm runs a pair of contiguous sliding 50 pixel windows across this slice and computes the Euclidean difference between the features of both patches. Here, HoG or Histogram of Oriented Gradient features [11] are used and in reality, we run the algorithm every 10 rows to speed up computation. Once the Euclidean distance is calculated, it is checked against a threshold that was set after experimentation to ensure fewer false positives (non-fishplate structures detected as fishplates). The intuition of this algorithm is that when the pair of windows straddles the edge of a fishplate the difference between the HoG features of the two patches would be maximal since there would be a large difference in the local object appearance of a fishplate versus an empty rail.

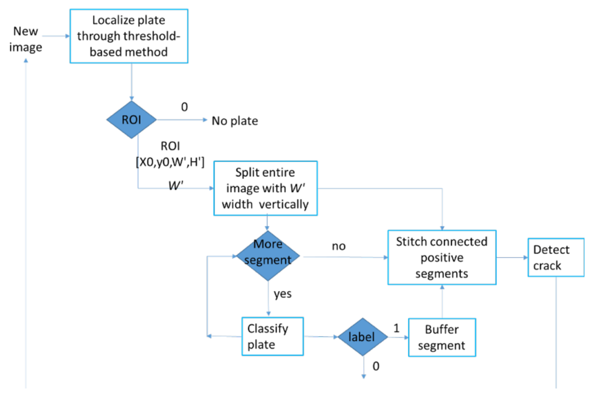

New image
Localize plate through threshold-based method
ROI
0
No plate
ROI [X0,y0,W',H']
W'
Split entire image with W' width vertically
More segment
no
yes
Stitch connected positive segments
Detect crack
Classify plate
label
1
0
Buffer segment

Figure 5. *Flowchart for fishplate localization*

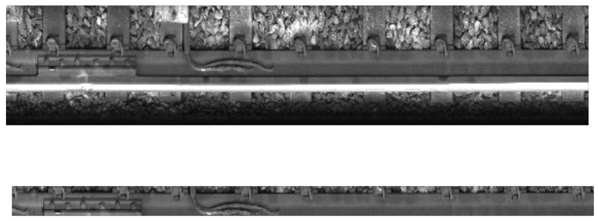


Figure 6. *Horizontal Pre-Localization of Rail image*

We used the G1 images containing fishplates from the 159 of approximately 10000 unique images sets described in Section along with 245 randomly selected non-fishplate G1 images to test our pre-localization algorithm. This is the dataset we used to test our fishplate localization algorithm. Although this threshold-based vertical pre-localization method was able to localize quite a few fishplates from the rail images (as seen in Figure 7), the accuracy was not satisfactory as shown in Table 1.

Performance of threshold-based pre-localization method $L_0 \sim L_3$, refer to different threshold values on the strength of the HoG features as discussed in section 1)

G1	L_0	L_1	L_2	L_3
TP	159	97	15	0
TN	6	240	245	245
FP	239	5	0	0
FN	0	62	144	159

In this table the labels shown at the top row (L_0, L_1, L_2 and L_3), refer to different threshold values applied to the strength of the HoG features as discussed previously. Here, L_0 is has the lowest threshold value and L_1, L_2, and L_3 are increasingly higher. In particular, it is noted that the false negative rate (the number of missed plates), at $L_1 \sim L_3$ levels were quite significant. In addition, comparing the columns 2-5 in Table 1 shows that the performance of this pre-localizer is very sensitive to where the threshold value is set, which means that the process can be sensitive to noise sources during image acquisition such as illumination variation and presence of different confusing objects (e.g., bond-wire, handwriting marks, etc.), if the threshold value is set for one set of images and applies to another. More importantly, the method is constrained to detect a single plate per image. For cases as shown

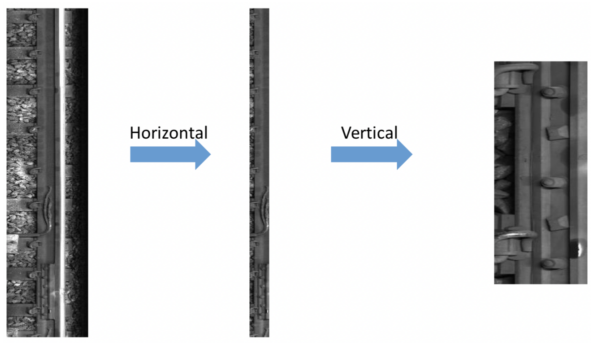


Figure 7. *Vertical Pre-Localization of Rail image*

in Figure 4 when multiple plates present in a single image, some plates would not be detected.

We can see from the right-most column in Table 1 that, at the L_0 level, though there were a lot of false positive detections, there was no false negatives either, i.e., no missed fishplates from the pre-localizer. This is a very desirable property from the pre-localization process because if a plate is not detected in this step, then it is impossible to recover the plate later in the process. On the other hand, if an area is falsely identified as a fishplate, it is possible to eliminate it through other means such as a fishplate classifier as it is demonstrated below. The current pre-localization step in the final pipeline is thus set at the L_0 level; and the identified ROIs are then passed to the refinement step to detect the final locations of the fishplates.

Refinement step using splitting-and-merging along with a foreground/background classifier

After a captured image is processed through the pre-localization step using the L_0 threshold, a refinement step utilizing foreground/background classifier (also referred as binary fishplate classifier or BFC) was introduced. The basic idea is to incorporate machine learning into the process so that a more discriminative foreground/background comparison can be achieved. This is what the pre-localizer lacks since without a weighting (e.g., through linear SVM) on the delta of the image features, the HoG features are not sufficient to locate the plate accurately.

One approach for the refinement is to focus on removing the false-positives in the pre-localizer step while leaving the spatial locations unaltered. [12] That is, for each detected fishplate ROI the classifier is used only to verify the presence of a fishplate. The benefit of such approach is that it only adds a small overhead at run-time once the binary fishplate classifier is trained. However, it would not be able to improve the localization of other true positives. In the field of object detection, such performance can be assessed via *IOU* or the Intersection Over Union between the true region of the fishplate and the localized region of the fishplate through algorithms.

Instead of using HoG as the image features like that in pre-localizer, we explored both HoG and BoW (Bag of visual Words) as the image features for the binary fishplate classifier. The results of using BFC as a post-selection for the candidate ROIs from pre-localizer are shown in Table 2. From the results, one can see the clear accuracy advantage of using BoW over HoG even though

HoG calculation is about 60 times faster than BoW calculation. We think the accuracy advantage comes from the fact that BoW features are less sensitive to errors due to shifts in the image from the pre-localizer. Comparing to the G1 column to Table 1 at the L_0 level, one can see that although introducing a binary fishplate classifier significantly reduced the false positive detections from the pre-localizer at the L_0 level, it is also noted that the false negative rate was increased and was still relatively high (44 plates for G1 as shown in Table 2). The reason of this increase of false-negatives may be due to poor localization in the pre-localizer. This leads us to introduce a more complex refinement step, which aims to reduce false-positive rate in the pre-localizer as well as fine-tuning the localized positions of the fishplate. Toward this end, we use the sliding window trick that is commonly used in converting object detection task into object classification task.

Comparison of fishplate classifier using BoW and HoG features (numbers in brackets))

BoW (HoG)	G1	G3
TP	115 (78)	149 (137)
TN	245 (234)	245 (234)
FP	0 (11)	0 (11)
FN	44 (81)	10 (22)

Our final refinement algorithm proceeds as follows. First, we train a patch BFC using positive patches (e.g., random portion of fishplate with $S_w = 300$ pixel length) and negative patches (e.g., random crop of non-fishplate regions with S_w pixel length). We use a fixed length patches in the training so that we only need to apply one sliding window size for computational efficiency. Note that this patch BFC is similar to the BFC above but trained with patches. Now, for each candidate ROI from pre-localizer, say with length × width = $H \times W'$, we segment it into N_w overlapped regions of size $S_w \times W'$ each. Here, $N_w = H/S_w + 1$ to ensure no gaps between the split patches while keeping the number of patches small to save computation. Each patch is input to the patch BFC to yield the $0/1$ label and its confidence of being a fishplate patch (e.g., using SVM score). If at least one patch within the candidate ROI has label 1 or positive, then we have found a fishplate in this candidate ROI; and the final refined positions of this found fish plate is found by merging connected positive patches starting from the patch with highest SVM score(s). Note that by nature of splitting and merging of connected positive patches into refined ROI for fishplate, it is possible that a given candidate ROI identified from pre-localizer may have zero (if no positive patches), one, or more final refined ROIs for fishplate. Since we use patch BFC, we can now refine and even split the candidate ROI. This greatly shifts the burden from pre-localizer to this refinement step. However, it is still worthwhile to have pre-localized step in practice. This is because (1) the computation is several order faster in pre-localizer (so it is not a larger overhead to add to) than split-and-merge with patch BFC and (2) large fraction of the acquired rail images does not have fishplate in it (the pre-localizer served to filter many images that do not need to get into split-merge step and fishplate defect detection step). The performance of this modified approach is shown in Table 3. With multiple ROI proposals per image, this splitting-and-merging process reduced the number of false negatives comparing to Table

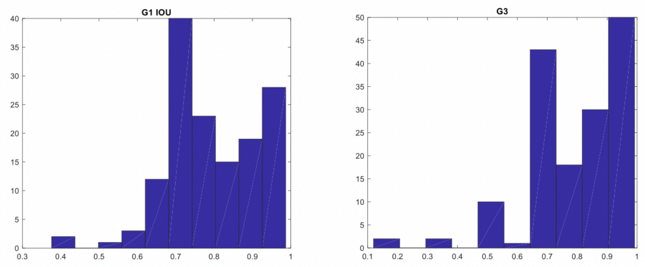


Figure 8. Improved IOU with the splitting-and-merging process.

2. As expected, this step not only improves the overall fishplate detection rate it also improve the intersect-of-union (IOU) over the pre-localizer with the classifier as demonstrated in Figure 8, where the left and right figures correspond to the result from G1 and G3 in y direction, respectively.

Performance of fishplate localization with sliding window and patch BFC

BoW	G1	G3
TP	143	156
TN	242	243
FP	3	2
FN	16	3

Snow Detection

A fair number of the images we came across in the rail image data set were taken on rail lines that experienced snowfall. This snow that is present in the image accumulated along the side of the track can sometimes interfere with the rail segmentation algorithm. Poor rail surface segmentation would lead to poor localization of fishplate and thereby the following process of crack detection. In order to mitigate the effect of this, we developed an algorithm to detect images with severe snow-on-track so that we can filter them out of the fishplate analysis, where the detection is not robust , or *removing* the snow via image matting techniques. We tested our algorithm on a set containing 87 G1 and G3 fishplates with snow images and 21 G1 and G3 non-snow fishplate images.

Figure 9 depicts our snow-on-track detection and image matting algorithm. The idea is to first assess the severity of accumulated snow or over-exposure of the acquired image via white pixel count. Based on this rough assessment, we apply simple image processing and machine learning methods to further separate between “severe” and “non-severe” conditions. For the “severe” condition, we flag the image and route it to manual inspection. For “non-severe” case, we perform simple image-matting method to replace white pixels in non-rail-surface regions with the mean of the images and then pass it on to the remaining process for defect localization and detection processing. The performance differences without vs. with our snow-on-track defect and image matting are summarized in Table 4. As shown in the table, we have gained 10% to 20% on the true-positives while have very little increase on false-positives. Note that it may appear that the FP has increase quite a bit (14%) but we only have 21 negative samples (no plate) in this test. Hence the high FPR increase should not be an issue here.

Performance of fishplate localization without and with snow-on-track detection and image-matting.

	Without image-matting		With image-matting	
	G1	G3	G1	G3
TP	74 (85%)	57 (66%)	83 (95%)	74 (85%)
TN	20	21	17	21
FP	1 (5%)	0 (0%)	4 (19%)	0 (0%)
FN	13	30	4	13

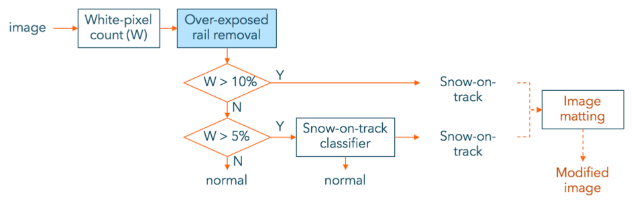


Figure 9. Snow-on-Track detection and image matting for removing its impact on rail surface segmentation.

Crack Detection Pipeline

Once a fishplate ROI is localized, the next step is to classifier whether the fishplate is cracked or not. To accomplish this, the fishplate ROI is first split into overlapping patches that can then be tested for the presence of cracks as seen in Figure 10.

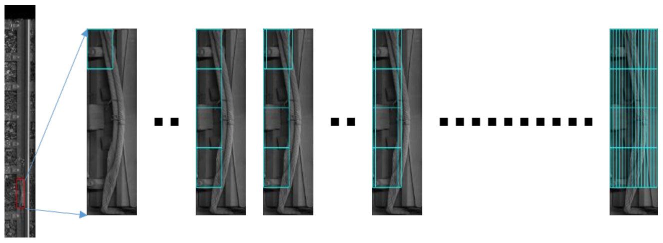


Figure 10. Splitting the fishplate image into patches.

A series of experiments were performed to select the set of parameters to optimize the classifier including the patch size, the feature type, and the kernel of the classifier. Experiment results show that a patch size of 128×128 pixels using deep-features (from the pre-trained VGG-16 model [13]) along with a linear SVM classifier gave the best classification performance. These patches performed better than the 64×64 or 32×32 size patches - our intuition here is that the larger patches gave more context to differentiate between cracks and features that looked like cracks but weren't. If the number of patches within a fishplate ROI classified as cracked exceed a pre-defined threshold η_s (eventually set at 2 after experimentation), the entire plate is declared as a cracked plate.

Besides tuning these parameters, the classifier performance using different numbers of training samples was also evaluated as shown in Table 5. We created a balanced training dataset (*Initial Set*) of 138 positive (cracked) *patches* and 157 negative (non-cracked) *patches* from the 27 unique crack location labeled plates we had in our set. Since the detection is at a patch level, cracked plates produce a large number of non-cracked training example patches too along with the cracked patches. We used the other independent crack location unlabeled testing set of 304 positive (cracked) *plates* and 398 negative (non-cracked) *plates*. Table 5 shows the experiment results, where the positive results and the negative results were evaluated using the testing data mentioned previously.

As can be seen in Table 5 the classifier achieved greater than 99% accuracy for the positive class at the plate level, but for the negative class the accuracy was below 5%. To improve the classification performance on the negative patches, the number of negative patches were increased. Since we did not have more unique positive patches, the number of positive patches remained the same throughout the experiment. The Additional Plates column in Table 5 shows the number of additional negative training plates from which negative training patches were extracted in subsequent datasets used in the experiment. The Name column indicates which line the additional negative plates were pulled from to extract the negative patches and the number signifies the number of plates from that line. The reason we talk in terms of plates and not patches is since the ultimate goal of the crack detection is at the plate level. As mentioned before we used a threshold of 2 cracked patches detected in a plate to classify it as a cracked plate.

Classification results using different number of negative training samples.

Name	Additional Plates	Evaluating Plates	
		Positive accuracy	Negative Accuracy
Initial set	none	99%	5%
CHUO	3 plates	87%	40%
	6 plates	84%	59%
	13 plates	80%	70%
ECHIGO	10 plates	77%	86%
	30 plates	75%	92%
	50 plates	73%	96%

As one can see from Table 5, the classification performance is significantly affected by the number of negative samples introduced to the training. With the current limitation of the number of training samples, it is impossible to reach a desired performance where both positive and negative classes achieved a comparable accuracy. Hence, to further improve the accuracy of the plate-crack detection, as mentioned earlier we toyed with the threshold η_s. For example, for the model using 6 plates from CHUO line (row #5 in Table 5), Table 6 shows the performance using different threshold of the number of cracked patches, η_s. The first row used one patch per image and the 2nd was two patches per image and so on. Over our experiments with multiple lines we found that the most optimal threshold for η_s was 2 patches per image, i.e., only if there are at least two patches are detected as cracked, the plate is classified as a cracked plate. This threshold can be adjusted for different lines in the future. This 6 plate CHUO model with a η_s threshold of 2 achieved on average a 76 % accuracy in finding cracks which is commendable for a defect that is often confused even by human engineers.

Comparison of number of minimum cracked patch for cracked plate detection.

η_s	Positive accuracy	Negative accuracy
1	84%	59%
2	80%	71%
3	77%	77%
4	71%	83%

Discussion and Conclusion

We constructed a pipeline that allows us to look for different types of defects on a railway image that functions with a high accuracy to detect cracked fishplates . This pipeline was trained on another simpler to find defect (broken bond-wires connecting railway tracks) and achieved 87% accuracy as well; thus demonstrating the generalizability of this approach. We are exploring collaborations with the JR engineers to augment the defect data using more model-based approaches as well using deep methodologies to extract more expressive features that may find defects in their most nascent stages so that early interventions can occur in a timely fashion.

References

[1] Tomas Liden. Railway infrastructure maintenance - a survey of planning problems and conducted research. *Transportation Research Procedia*, 10, 2015.

[2] Mohammad Farukh Hashmi and Avinash G. Keskar. Computer-vision based visual inspection and crack detection of railroad tracks. *Recent Advances in Electrical and Computer Engineering*, 2014.

[3] Limin Chen, Yin Liang, and Kaimin Wang. Inspection of rail surface defect based on machine vision system. *Information Science and Engineering (ICISE)*, 2010.

[4] E. Deutschl, C. Gasser, A. Niel, and J. Werschonig. Defect detection on rail surfaces by a vision based system. *Intelligent Vehicles Symposium, 2004 IEEE*, 2004.

[5] Lin Jie ; Luo Siwei ; Li Qingyong ; Zhang Hanqing ; Ren Shengwei. Real-time rail head surface defect detection: A geometrical approach. *IEEE International Symposium on Industrial Electronics*, 2009.

[6] Hao Feng ; Zhiguo Jiang ; Fengying Xie ; Ping Yang ; Jun Shi ; Long Chen. Automatic fastener classification and defect detection in vision-based railway inspection systems. *IEEE Transactions on Instrumentation and Measurement*, 63, 2013.

[7] Shahrzad Faghih-Roohi ; Siamak Hajizadeh ; Alfredo Nez ; Robert Babuska ; Bart De Schutter. Deep convolutional neural networks for detection of rail surface defects. *International Joint Conference on Neural Networks*, 2016.

[8] YoungJin Cha, Wooram Choi, and Oral Bykztrk. Deep learningbased crack damage detection using convolutional neural networks. *Computer Aided Civil and Infrastructure Engineering*, 2017.

[9] Sheng Guo; Weilin Huang; Limin Wang; Yu Qiao. Locally-supervised deep hybrid model for scene recognition. *IEEE Transactions on Image Processing*, 2017.

[10] Wikimedia Foundation. Range imaging.

[11] Robert K. McConnell. Method of and apparatus for pattern recognition, 07 1982.

[12] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

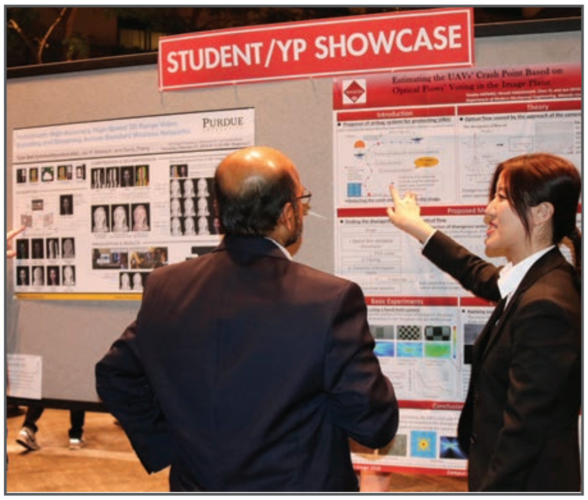
JOIN US AT THE NEXT EI!

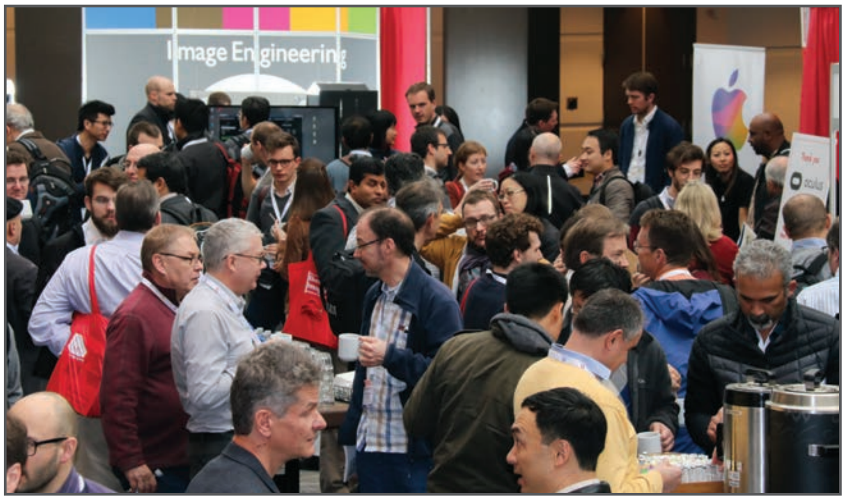
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





AR is a fundamental and transformative interface and platform.
20+ years from my initial working demo to today's AR displays
Optimistic that it won't take another 20 to reach ubiquitous consumer AR!
My first working AR prototype
Today's platforms
Ubiquitous consumer AR

• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org