https://doi.org/10.2352/ISSN.2470-1173.2019.10.IQSP-301

Segmentation-Based Detection of Local Defects on Printed Pages[1]

Qiulin Chen [a], ***Renee Jessome*** [b], ***Eric Maggard*** [b], ***Jan P. Allebach*** [a]
[a] ***School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, U.S.A.***
[b] ***HP Inc., Boise, ID 83714, U.S.A.***

Abstract

Local defects are very common on printed pages. Automatic detection of such defects will help the product support personnel to diagnose the problem and fix it more efficiently. Among previous works on local defect detection on printed pages, most of them divide the printed page into small blocks and calculate the variation within each block. This method is time consuming and not robust in dealing with defects at different scales. In this paper, we propose a robust framework for detecting the local defects on scanned printed pages. To achieve the efficiency and robustness, our framework applies the Gaussian pyramids method and the selective search method. We also create manual features for classification to increase the detection accuracy. Finally, applying our method on printed pages demonstrates its efficacy.

1 Introduction

Printed image quality is always a top issue in the printing industry. No matter how you improve the hardware or software in the printer, printing defects always exist. It's important to detect and recognize them in a timely manner so the manufacturer can make adjustments to the printers accordingly.

Printing defects are mainly divided into three categories: streaks, banding and gray spots. In this paper, we will focus on the detection of gray spot defects. Comparing with other two kinds of defects, gray spot defects are caused by contamination, dirt or debris in the paper path while with inkjet printers, streak defects are mainly caused by multiple missing nozzles. The streak and banding defects according to their causes are more widely distributed and more easily spotted by human eyes. But gray spot defects are not easy to be detected directly by human eyes in a pure color test page due to their small size and sparse distribution. But such defects will cause problems when printing customer contents which are usually not pure color prints. This will cause severe flaws in the final printed image. It is important to find these defects and then, from the detection result, we can deduce the potential cause of them and get it fixed – either to clean the paper path or purge the printing nozzles. To achieve a more precise deduction, enough information from a batch of test prints is needed. Traditionally, when certain kinds of defects occur in the printed images, the manufacturer will send out technicians to investigate flaws by looking directly at the print under certain lighting condition. This is time consuming and lacks accuracy. So an automatic detection of the printing defects is important, and can greatly increase the work efficiency.

Many challenges come up in gray spot defects detection. Detecting the gray spot defects is similar to texture analysis of an image, trying to locate the region which has large variation from its neighborhood. This variation, in our case, is the non-uniformity of the texture neighborhood. But defect detection in scanned printed pages is a little different from analyzing textures of digital images. Firstly, the detection is conducted on the scanned printed pages. This means we will encounter the noise from high frequency halftone patterns. These halftone patterns cause a certain non-uniformity compared with the original digital image. If one zooms in the image to certain scale to search for the gray spots, they cannot be seen, since gray spots are overwhelmed by the halftone patterns. Secondly, gray spots have different scales even though they are very small comparing with the image scale. Thirdly, the gray spots are sparsely distributed. The size of our scanned images are very large (6535 $\times$ 5072 pixels). This requires the detection method to have a certain efficiency.

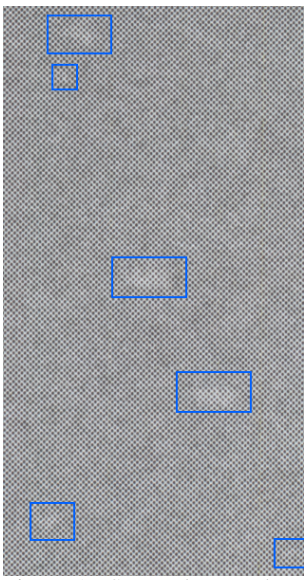


Figure 1: Scanned monochromatic printed pages.

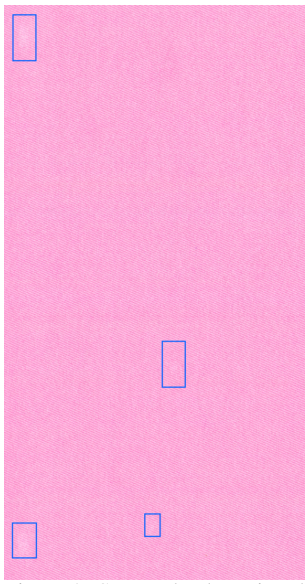


Figure 2: Scanned color printed pages.

Figure 1 shows an image clipped from a scanned monochromatic page. Figure 2 shows an example from a color printed page. Since they are scanned in high resolution, it is a little bit hard for human eyes to directly spotted the defects. The defects are marked out by blue bounding boxes. They are sparsely distributed and have variable sizes. It is much easier for human eyes to detect the defects directly by removing the high frequency halftone pattern first. We will give an comparison example about halftone

[1]Research supported by HP Inc., Boise, ID 83714

pattern removal later. Besides, by comparing these two examples, we can see that the monochromatic patch shows a higher contrast when searching for the gray spot defects. This implies that it is easier to do detection on grayscale images or a single L* channel instead of directly detecting defects on color images due to higher contrast.

In this paper, we propose a framework to automatically detect the gray spot defects on scanned printed images from an electrophotographic printer. We try to use an efficient segmentation-based method to detect the potential regions that may contain the gray spot defects and apply classification with manual features to increase the detection precision. Another superiority of our method is we do not need the reference image in the detection process, which makes the method more practical since there is often no reference image offered when inspecting the defects on customers' prints.

2 Related Work

A lot of work has been done on printed page defect detection. Most of them share similar procedures: image preprocessing , potential defect locations detection, feature extraction, defects classification and image postprocessing. Previously, researchers used a set of sliding windows with different sizes filtering through images to estimate the potential location of defects. This is time consuming and of low accuracy when we have defects with variable sizes. They designed all kinds of filters to detect the defects. Nowadays, as deep learning has become more commonplace, people are starting to use convolutional neural networks to detect defects.

For print defects detection, Wang et al. [1] designed a pipeline to automatically detect the local defects using reference images. But the detection algorithm is not very efficient and robust since they used fix-sized grids to locate the defects. Jing et al. [4] used the Structural Similarity (SSIM) [5] Index to score the quality of images containing all kinds of defects, such as streaks, bands, gray spots, dark spots, and fading defects. Their work is similar to Wang et al. [1] in that they used reference images for defect detection, which is not practical since usually we will not have the reference images for customer contents. Nguyen et al. [2] [3] designed a system to score the quality of a printed page based on detected defects using a support vector machine with hand-crafted features.

3 Methodology

The framework of our method, shown in Figure 3, can be divided into three parts: (1) color space transformation and descreening, (2) candidate region detection, (3) feature extraction, and (4) classification.

3.1 Preprocessing

As we discussed in the introduction, detecting defects in the L^* channel is much easier than in a color image. So we first transfer the original scanned images from the *sRGB* color space into the $L^*a^*b^*$ color space. We expect a higher contrast on gray spot defects.

Then, we try to remove the high frequency halftone pattern from the scanned print pages, since this will make the scanned print more smooth and will ease the later detection. Removing high frequency halftone textures is also call descreening. The descreening process actually simulates the human vision system which focuses on the low frequency textures. We apply the Nasanen filter, a low pass filter, to the scanned print page. Here, we show two images as an example proving the effectiveness of this filter in removing the halftone pattern.

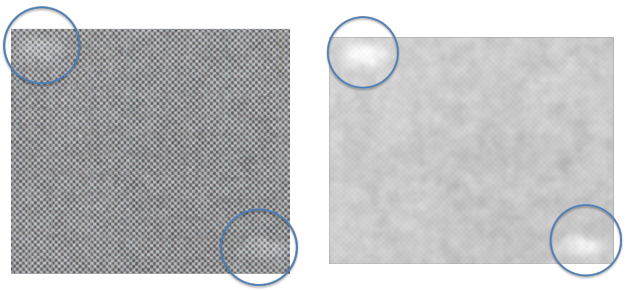


Figure 4: Patch on left hand side is an original patch while the other one is descreened to remove the high frequency halftone patterns by filtering through a low pass filter.

In Figure 4, we show the difference between the original patch and the preprocessed patch. The preprocessed patch is much smoother so it is easier to detect the local defect marked by the blue circles.

Local Defect Detection

In this step, we try to detect the potential regions of interest (ROI), which mark the possible defect locations. In order to deal with the local defects of different sizes, traditionally people use different sizes of sliding windows. But this method is not robust for defects with variable sizes; and it's time consuming. And the more variable the defect size is, the higher the cost is. So we design a new robust and efficient framework to make sure we can detect defects with variable size more precisely. To achieve this, we apply the Gaussian pyramid method [6] and the selective search method [7].

The Gaussian pyramid method has been widely used for a long time. To apply this method, we just need to use a Gaussian kernel to blur the image and downsample it. By doing this, we achieve the image in next level, which is small in size compared with original image. We keep doing this on the newly generated images to give us more images. If we stack them together, we will get a image pyramid. Detecting local defects in different pyramid levels makes sure that our result is more robust.

The selective search method is an object detection method. It is developed based on a graph-based segmentation method proposed by Felzenszwalb et al. [8]. We apply this method in different pyramid levels. This may give us multiple detection results for each defect in different pyramid levels. We can fuse the results for each defect in different levels. This will improve the detection precision, and give us more information about each defect, which will help us to discriminate the defect patches from non-defect patches in the classification step later. For example, we may miss the detection of some defect in one pyramid level but we may successfully detect it in other pyramid levels, which make these detection results in different pyramid levels complementary to each other. And some big local defects that are not detected in lower pyramid levels may be detected in higher pyramid levels.

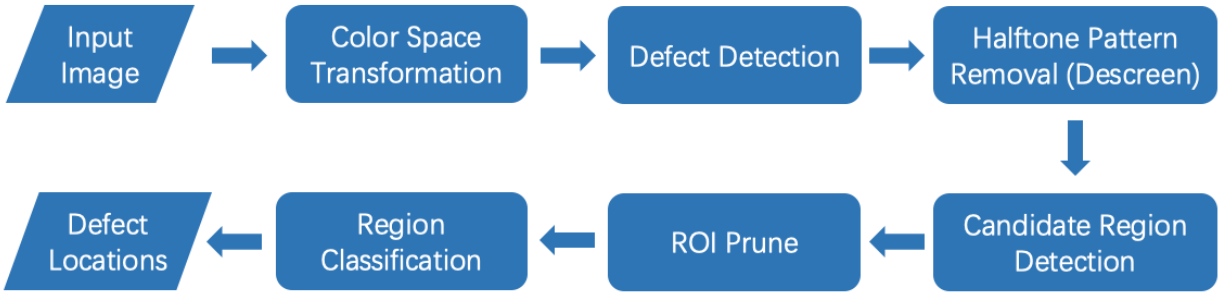


Figure 3: Gray spot defect detection framework.

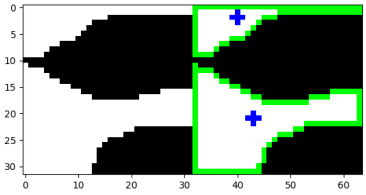


Figure 5: Segmentation results of a non-defect patch. Left image shows the original patch, while the right one shows the further segmentation result, with blue marks centers and green marks outlines of foregrounds.

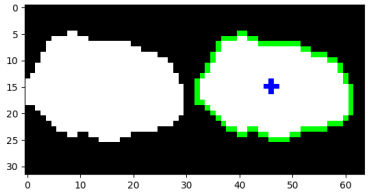


Figure 6: Segmentation results of a defect patch. Left image shows the original patch, while the right one shows the further segmentation result, with blue marks the only center and green marks outlines of the foreground.

Feature Extraction

After getting potential locations of gray spot defects, we need to do further classification on them. To achieve this, a naive way is to reshape patches extracted in the last step into vectors and feed them directly into our classifier. This method will achieve a certain precision, but is still not good enough since we will lose much spatial information contained in the image. And the image vectors will not focus on the features describing the patch containing defects. So we create some manual features from the image, which will focus on the texture distribution and average energy of the defect patches. From this, we can expect a higher detection precision. From the previous step, we achieve potential locations of defects, and overlay tight bounding boxes over them, represented by P. To derive finer locations of defects within the bounding boxes, we apply Otsu's method to further separate the defects from the background. And we can also decide the center and outline of the defects from the segmentation results. Figures 5 and 6 show a pair of examples to illustrate the difference between defects and non-defects segmentation results. The blue plus signs represent the cluster centers of the segmentation while the green lines represent the outline of the foreground. For defect patches, the foreground is usually a regular, approximately elliptical shape and with one single cluster and one center. But non-defect patches tend to have multiple centers and irregular shapes for the foreground. These observations are helpful for us in building manual features later.

After getting more precise locations and masks of our defects within each patch, we create manual features for each defects from the further binary segmentation. Let's represent each finely segmented defected region as $I \in P$ in each patch. To create the first feature, we convert the image patch from *sRGB* to *CIE* $L^*a^*b^*$ color space. Then for each patch P, we calculate the mean average values in three different channels.

$$L^*_{avg} = \frac{1}{|P|}\sum_{i\in P} L^*_i, \quad a^*_{avg} = \frac{1}{|P|}\sum_{i\in P} a^*_i, \quad b^*_{avg} = \frac{1}{|P|}\sum_{i\in P} b^*_i$$

Here, $|P|$ represents the cardinality of the set P. Based on this, we calculate the normalized ΔE value for each patch.

$$\Delta E_i = \sqrt{(L^*_i - L^*_{avg})^2 + (a^*_i - a^*_{avg})^2 + (b^*_i - b^*_{avg})^2} \quad i \in P$$

After this, we get the variance of ΔE values.

$$\sigma_l = \sqrt{\frac{1}{|P|-1}\sum_{i\in P}(\Delta E_i - \Delta E_{avg})^2}$$

$$\Delta E_{avg} = \frac{1}{|P|}\sum_{i\in P}\Delta E_i$$

We can still create other features such as area coverage ratio between foreground and background R_A, perimeter to area ratio R_PA, intensity difference I_{diff}, and Weber's contrast C_w. Let B represent the background in patch P, where $P = B \cup I$.

$$R_A = \frac{|I|}{|P|} \qquad I_{diff} = (\bar{I} - \bar{B})/255$$

$$R_{PA} = \frac{C(I)}{|I|} \qquad C_w = (\bar{I} - \bar{B})/\bar{B}$$

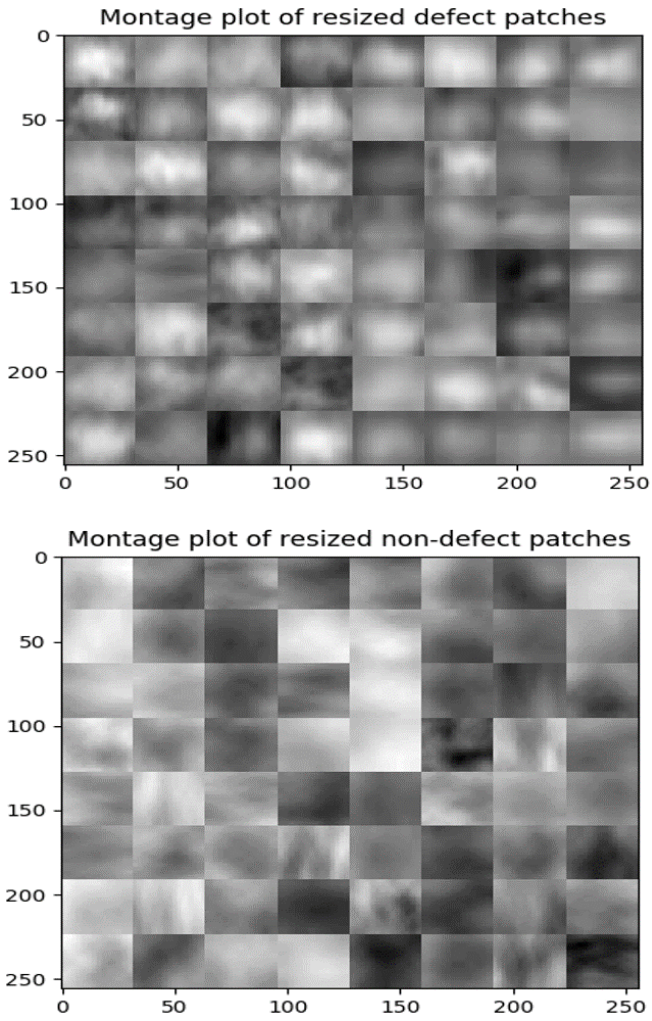


Figure 7: Patches example. The first patch shows us examples containing gray spot defects while the second patch shows examples which have no gray spot defects included. The textures and average luminance of the patches from the two classes are quite different. This shows us a promising way to separate them from each other.

In the above, $C(\cdot)$ represents the function to calculate the perimeter of the input. $\bar{I}$ and $\bar{B}$ represent the mean pixel value of defects and background, respectively.
As we mentioned previously, another naive way is to input the reshaped image patch into the classifier. We will compare the results of applying this naive way with the results by using our manual features in Section 3.

Classification

In this step, we use machine learning methods to classify the ROIs obtained from last step. We classify the data into two groups: defect and non-defect patches. We reshape the patches into size of 40×40. Below, we give an example of defect and non-defect patches for training.

From Figure 7, we can see some difference in the texture between defect and non-defect patches. Based on this observation, we use a machine learning method to classify the defect and non-defect patches to further improve our detection results.

In order to increase the precision of classification, we also create manual features for each patch, such as the ratio between foreground and background within each patch, the contrast of foreground compared to the background in each patch, and so on. In the current situation, we create more manual features to increase feature dimension, which will improve the classification result.

After we get the feature vectors for all the patches, we feed the features into a Support Vector Machine (SVM) classifier [9] to do the training. Then we apply our trained model on test data, which proves the goodness of our framework.

3 Experimental Results

In this section, we show some qualitative detection results. Below is our detection result on an example page in 3 different pyramid levels. The blue boxes represent the defects we detect while the red ones represent the non-defects.

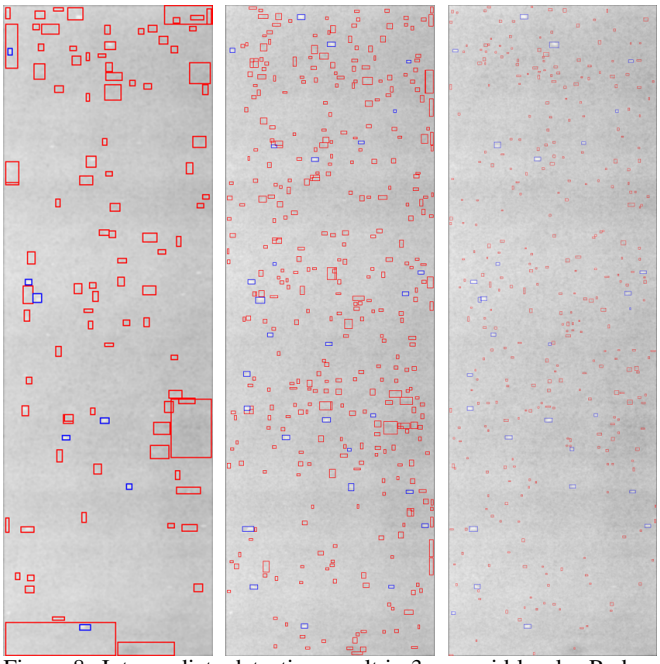


Figure 8: Intermediate detection result in 3 pyramid levels. Red boxes represent regions containing no defects (false alarms). Blue boxes represent regions containing defects. From these examples we can see the defects are very sparse, and hard to be directly detected by human eyes.

We used 7374 patches for training and 1844 patches for testing in total. We controlled the data amount ratio between defect and non-defect patches around $1:7$. Defects are manually annotated by ourselves. And the final detection accuracy through using our manual features is 92.0%, while the accuracy through directly feeding patches is 90.9%. From this, we can see our manual features do make an improvement in detection accuracy. Below, we use a table to show the cross validation accuracy during training.

Table 1: 5-folder cross validation accuracy.

	1st	2nd	3rd	4th	5th
Accuracy(%) patch	90.92	90.92	90.92	90.90	90.91
Accuracy(%) Manual Feature	92.05	92.05	92.11	92.11	92.10

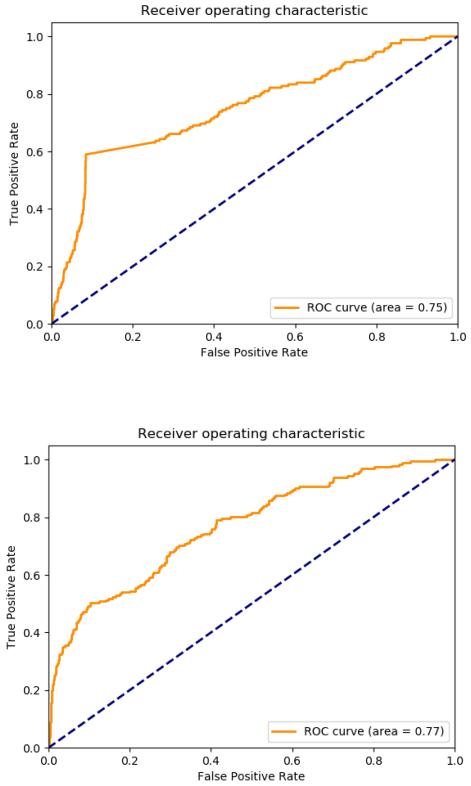


Figure 9: The plot on the top is the ROC curve for training using patches directly with $AUC = 0.75$. The other one on the bottom is ROC the curve for training using manual features with $AUC = 0.77$.

Here, we use receiver operating characteristics (ROC) curve [10] to show the improvement after using the manual features for training. During the classification, Each patch will be assigned with possibility of being classified as a defect patch. We vary the threshold between 0 and 1 as a decision boundary to labeling the patches as defect or non-defect, from which we get the ROC curve with False Positive Rate (FPR) varying from 0 to 1.

Area under curve (AUC) is a measure of the capability of the classifier to distinguish true objects from false alarms. When $AUC = 0.5$, it means the system has no separation capability. And the higher the AUC value, the better the model predicts defects as defects and non-defects as non-defects. We display the two ROC curves below. The orange lines show the performance of our classification system and the blue lines are the reference line for $AUC = 0.5$. We also calculate the area under curve (AUC) values separately. Our model trained with manual features achieves $AUC = 0.77$ while model trained with patches achieves $AUC = 0.75$, which implies that our model trained with manual features is better.

We also zoom in on the ROCs in Figure 10 for a False Positive Rate between 0.00 and 0.05, since this is the range that is more likely to be of interest in real applications. The red line shows results of our manual feature inputs with $AUC = 0.0072$, while blue line shows results of vectorized patch inputs with $AUC = 0.0066$. This shows us the manual feature do make an improvement in recognizing defects.

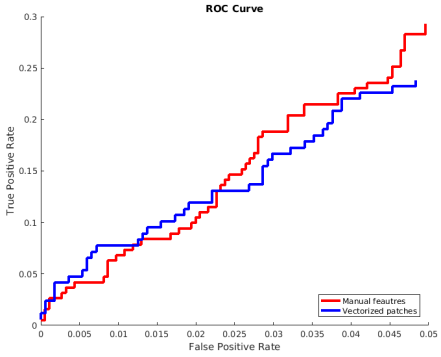


Figure 10: ROC plots zoom in range from 0.00 to 0.05. Red line, showing results of our manual feature inputs, has bigger $AUC = 0.0072$ which is bigger than that of blue line, showing results of vectorized patch inputs, with $AUC = 0.0066$.

4 Conclusion

In this paper, we propose a new segmentation-based framework for local print defects detection. It is more robust in detection of gray spot defects with variable sizes. We also apply a pyramid method to decrease the miss detection rate. To improve the detection accuracy, we create manual features to describe the characteristics of our patches. From the test results, we can see that our framework works really well in detecting gray spot defects on scanned printed pages.

References

[1] Jianyu Wang, Terry Nelson, Renee Jessome, Steve Astling, Eric Maggard, Mark Shaw, and Jan P. Allebach. "Local defect detection and print quality assessment." Electronic Imaging 2016, no. 13, 2016, pp. 1-7.

[2] Minh Q. Nguyen, Renee Jessome, Steve Astling, Eric Maggard, Terry Nelson, Mark Shaw, and Jan P. Allebach, Perceptual metrics and visualization tools for evaluation of page uniformity, In Image Quality and System Performance XI, vol. 9016, p. 901608. International Society for Optics and Photonics, 2014.

[3] Minh Q. Nguyen and Jan P. Allebach, Controlling misses and false alarms in a machine learning framework for predicting uniformity of printed pages, In Image Quality and System Performance XII, vol. 9396, p. 93960I. International Society for Optics and Photonics, 2015.

[4] Xiaochen Jing, Steve Astling, Renee Jessome, Eric Maggard, Terry Nelson, Mark Shaw, and Jan P Allebach, A general approach for assessment of print quality, in IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics, 2013, pp. 86530L 86530L

[5] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli, 2004. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing 13, no. 4, 2004, pp. 600-612.
[6] Edward H. Adelson, Charles H. Anderson, James R. Bergen, Peter J. Burt, and Joan M. Ogden. "Pyramid methods in image processing." RCA engineer 29, no. 6, 1984, pp. 33-41.
[7] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. "Selective search for object recognition." International journal of computer vision 104, no. 2, 2013, pp. 154-171.
[8] Pedro F. Felzenszwalb, and Daniel P. Huttenlocher. "Efficient graph-based image segmentation." International journal of computer vision 59, no. 2, 2004, pp. 167-181.
[9] Corinna Cortes, and Vladimir Vapnik. "Support-vector networks." Machine learning 20, no. 3, 1995, pp. 273-297.
[10] Tom Fawcett. "An introduction to ROC analysis." Pattern recognition letters 27, no. 8, 2006, pp. 861-874.

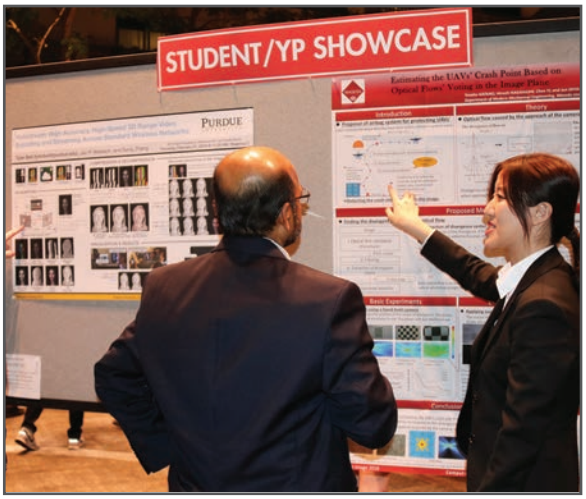
JOIN US AT THE NEXT EI!

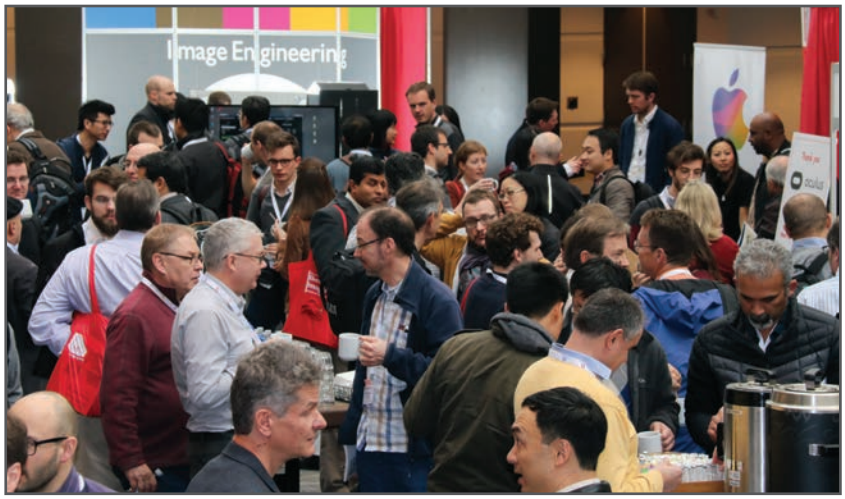
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

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





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

www.electronicimaging.org

IS&T
imaging.org