https://doi.org/10.2352/ISSN.2470-1173.2020.9.IQSP-323

Document Image Quality Assessment with Relaying Reference to Determine Minimum Readable Resolution for Compression*

Litao Hu[a], Zhenhua Hu[a], Peter Bauer[b], Todd J. Harris[b], Jan P. Allebach[a]
[a]Purdue University; West Lafayette, IN 47906, U.S.A.
[b]HP Inc.; Boise, ID 84714, U.S.A.

Abstract

Image quality assessment has been a very active research area in the field of image processing, and there have been numerous methods proposed. However, most of the existing methods focus on digital images that only or mainly contain pictures or photos taken by digital cameras. Traditional approaches evaluate an input image as a whole and try to estimate a quality score for the image, in order to give viewers an idea of how "good" the image looks. In this paper, we mainly focus on the quality evaluation of contents of symbols like texts, bar-codes, QR-codes, lines, and hand-writings in target images. Estimating a quality score for this kind of information can be based on whether or not it is readable by a human, or recognizable by a decoder. Moreover, we mainly study the viewing quality of the scanned document of a printed image. For this purpose, we propose a novel image quality assessment algorithm that is able to determine the readability of a scanned document or regions in a scanned document. Experimental results on some testing images demonstrate the effectiveness of our method.

Introduction

Nowadays, scanners are very commonly used both in the office and at home to digitize printed documents, drawings or handwritten documents, for convenient distribution. In most cases, these digitized documents will eventually be viewed on screens by human beings or fed into other software or algorithms for recognition. With a typical scanner, the user generally needs to select a scanning resolution when scanning the document. With high resolution such as 600 dpi or 300 dpi, the user may end up with a file that is excessively large for distribution. With low resolution, such as 100 dpi or 75 dpi, the quality degradation can be very severe, although the file is small. Therefore, choosing an appropriate resolution can sometimes be a very tricky task, since it depends on the purpose of the scanned document and the content to be scanned. In this paper, we seek the minimum resolution possible that ensures reading quality, since it allows for the smallest file size without losing too much information.

As pointed out in [1], for pictures and photos that will ultimately be viewed by human beings, their visual quality can only be correctly quantified by subjective evaluation. However, unlike pictures or photos, our interest here is in structural contents like texts, lines, bar-codes, QR-codes, and hand-writing. For such content, we take readability to be the only and necessary factor that determines the viewing quality. Therefore, a method that determines readability would be a better measure for viewing quality of such information in images.

In this paper, we propose a document image quality assessment algorithm that can be used to determine the readability of document images at different scanning resolutions and a compression system that segments a document image into different regions and then determine the minimum readable resolutions for text regions, before outputting a compressed digital file according the minimum readable resolution. The system diagram is shown in Figure 1.

In the following sections, we will introduce related work on several popular image quality assessment methods. Then, we will describe in detail our image readability assessment system, along with its key building blocks. Finally, experimental results will be presented, followed by a brief conclusion.

Related Work

Page Segmentation

Image segmentation has been a very active field of research, and there have been a great number of methods proposed. Page segmentation is a problem that is similar to image segmentation with a more specific setting; and there are many published works in this area. Page segmentation aims to segment a document page, usually obtained by scanning a printed document, into regions containing different types of content. The segmentation is usually realized by classifying pixels, groups of pixels, or sub-images into different categories such as vector, raster, symbol, or background.

In recent years, deep learning techniques and models have enabled a series of powerful data-driven methods. [2] introduced a method to segment document pages using a Convolutional Neural Network (CNN) with an encoder-decoder structure. In [3], a hierarchical approach combining a cross-correlation method, the Kolmogorov complexity measure, and a neural network classifier is proposed.

Although deep learning based methods can be very powerful, the amount of computational power required is not always available in a practical application. Conventional methods [4, 5] usually involve connected component analysis in combination with a sliding window on the binary image of a scanned page. However, connected component analysis normally has a high computation complexity, which may not be acceptable when the document page is scanned at high resolution. In [6], a memory-efficient strip-based approach is introduced, which greatly reduces the time and space complexity. But given the setting of our problem, we can actually sacrifice some accuracy for an even simpler and faster approach.

*Research supported by HP Inc., Boise, ID 83714

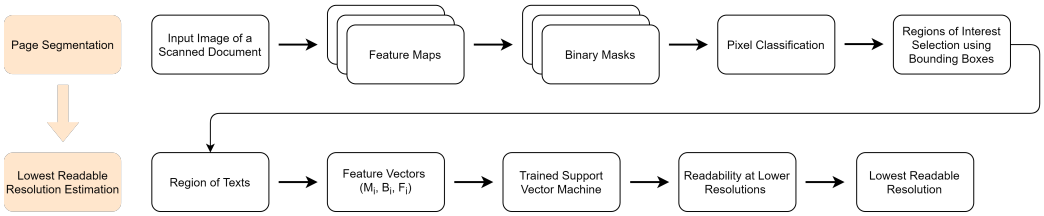


Figure 1: System Diagram.

Image Quality Assessment

There are numerous research papers and proposed methods for image quality assessment. Based on whether a reference image is used when assessing a target image, the methods can be divided into three groups, namely full-reference (FR) image quality assessment, reduced-reference (RR) image quality assessment, and no-reference (NR) image quality assessment. Full-reference image quality assessment, or FR-IQA, estimates the quality score by comparing the target image with a reference image that, in most cases, has high quality. Representative works on FR-IQA include [1, 7, 8]. At the opposite extreme, no-reference image quality assessment, or NR-IQA, tries to estimate the quality score of the target image in the absent of a reference image. Based on the purposes, NR-IQA can be further divided in to two categories, distortion-specific NR-IQA (DS-NR) and general-purpose NR-IQA (GP-NR). Representative works on DS-NR include [9, 10, 11]; and representative works on GP-NR include [12, 13, 14]. Betweem FR-IQA and NR-IQA, RR-IQA tries to use less information from the reference image, or only uses part of the reference image, to achieve high accuracy in quality score estimation. Representative works include [15, 16]. In this paper, we propose a full-reference image quality assessment method designed specifically for low resolution degradation. To the best of our knowledge, no similar resolution-specific FR-IQA like ours has been proposed before.

Proposed Method

As shown in Figure 1, the system consists of two parts. In the first part, the image of a scanned document is segmented and multiple rectangular regions will be located, which will then be classified as symbol region, raster region, or vector region. In the second part, feature vectors will be calculated from each symbol region. With the calculated feature vectors, a trained support vector machine will then be used to classify these features as either readable or not readable, from which the minimum readable resolution for the target region will eventually be derived.

Part 1. Page Segmentation and Region of Interest

In this section, we present a novel approach to segment a scanned document and form regions of interest. The process can be illustrated with the example in Figure 2. As the first step, the image of the document will be resized to a smaller size to reduce computation. In our implementation, we resized the source image from 600 dpi to 100 dpi with size 1100×850 for page segmentation. Then, features will be calculated for every pixel, forming three feature maps, each corresponding to one particular feature. In these feature maps, every pixel represents the feature value for the 5×5 window centered at the corresponding location in the input image. With the three feature maps, we then apply Otsu's method to convert them to binary masks. Note that for the average saturation feature map, we only apply Otsu's method to images with obvious color regions. If the maximum value in the feature map of average saturation is less than a preset threshold, for example 15, we will consider the input image to be a gray-scale image; and we will create an empty binary mask for this feature map instead.

Combining information from the three binary masks, we can classify pixels of the input image into three classes, namely symbol, raster, or vector. After that, we separate symbol pixels and raster pixels into two different masks, and apply morphological dilation to the two masks, respectively. Finally, we draw bounding boxes to bound all the connected component satisfying certain conditions in the two dilated masks, with red boxes indicating symbol regions and blue boxes indicating raster regions. Later in the pipeline, symbol regions and raster region will be processed separately, and treated differently.

Note that the proposed page segmentation algorithm is not the only method that can be used to get bounding boxes here. In fact, it can be replaced by other algorithms if they produce the same output.

Features, Feature Maps and Binary Masks

In raster order, we calculate three feature values for a 5×5 window centered at every pixel in the binary input image, gray-scale input image, or RGB color input image. The calculated features then form three feature maps that have the same size as the input image. The calculated features are as follows:

1. Averaged Gray-scale Image: Gaussian weighted average of all pixel values in a 5×5 window in the gray-scale image, which is equivalent to applying a 5×5 Gaussian filter to the gray-scale image obtained from the input RGB color image. We use $\sigma = 1.1$ for the 5×5 Gaussian kernel in our implementation.
2. Average Gradient: Gaussian weighted average of the absolute value of the difference between every pair of neighboring pixels in a 5×5 window of the gray-scale image. We use $\sigma = 1.1$ for the 5×5 Gaussian kernel in our implementation.
3. Average Saturation: For every pixel in a 5×5 window, calculate their saturation values in the HSV color system. We use the Gaussian weighted average of all saturation values in the window as the feature value for the center pixel. We use $\sigma = 1.1$ for the 5×5 Gaussian kernel in our implementation.

To understand how the pixel classification works, we need to first understand what these features mean. Below is how we can interpret the three features.

Averaged Gray-scale Image. Assuming a white or light-colored background, which is much more common than dark-

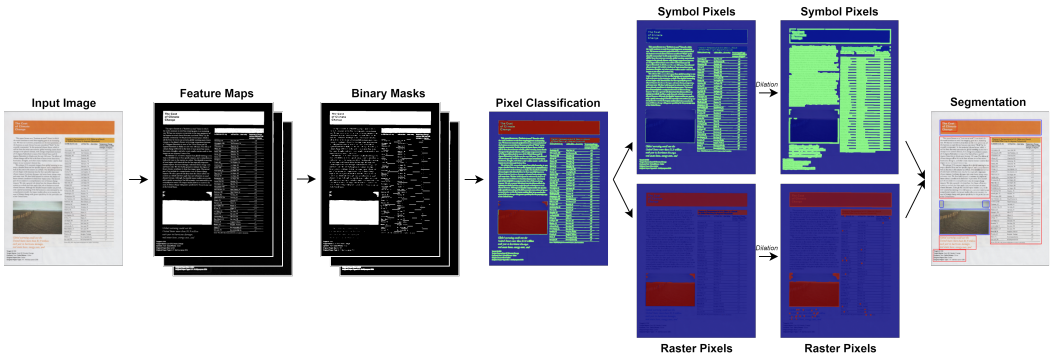
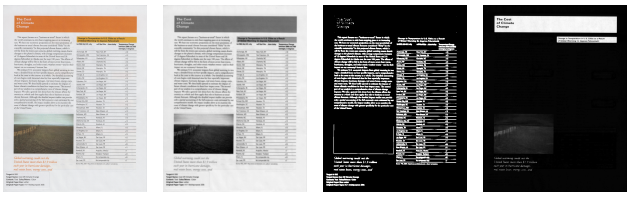
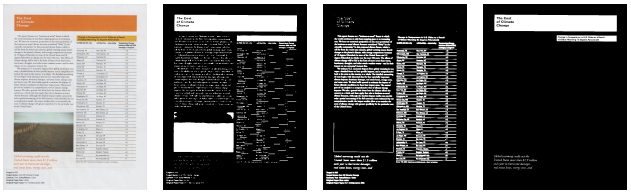


Figure 2: Page Segmentation and Regions of Interest Selection.



(a) (b) (c) (d)

Figure 3: Feature Maps: (a) Input Color Image; (b) Averaged Gray-scale Image; (c) Average Gradient; (d) Average Saturation.



(a) (b) (c) (d)

Figure 4: Binary Masks: (a) Input Color Image; (b) Binary Image (BI); (c) Average Gradient (AG); (d) Average Saturation (AS).

colored background, then foreground pixels will normally have dark colors. Therefore, the gray-scale image itself can be an effective feature map for a scanned document, and those dark-color pixels can be a good indication for main contents or non-background pixels. Since a raw scanned image may suffer from scattered noise pixels, we apply a 5×5 Gaussian filter to the gray-scale image in order to remove such noise.

Average Gradient. In a window in the gray-scale image, the "gradient" for a pixel is calculated as the average absolute value of the differences between that pixel and all its four direct neighbors. Average gradient then means the Gaussian weighted average of all the gradients inside a window. The average gradient will have greater value in regions where pixel values are alternating, which is often the case in high-frequency regions. Therefore, greater feature values will be seen in windows full of symbols, while smaller feature values will be seen in windows with vector content. In other words, this feature behaves similarly to an edge detector; and a higher average gradient can be a good indication for symbol pixels or other high-frequency content in the image.

Average Saturation. Achromatic colors will have small saturation values. As a result, any gray pixels will have a small saturation (ideally 0). On the other hand, for a chromatic color pixel, the saturation value will be greater. Therefore, assuming that symbol pixels are usually black, a greater saturation can be a good indication for raster pixels.

Figure 3 shows an example input color image, along with the three corresponding feature maps for the input image, and Figure 4 shows the example color image with the corresponding binary masks from the three feature maps. We can see that they coincide with our interpretation of the features. With the three binary masks, we can then proceed to pixel classification.

Pixel Classification and Bounding Boxes

As we have discussed previously, the average gradient can be a good indicator for symbol pixels, the average saturation can be a good indicator for raster pixels, and foreground pixels can be a good indicator for pixels related to the major content. Therefore, we adopt the decision tree shown in Figure 5 to classify pixels into one of the three categories. We use *BI*, *AG*, and *AS* to represent the three binary masks, respectively. As for the class indices, we use 2 for raster pixels, 1 for symbol pixels, and 0 for vector pixels.

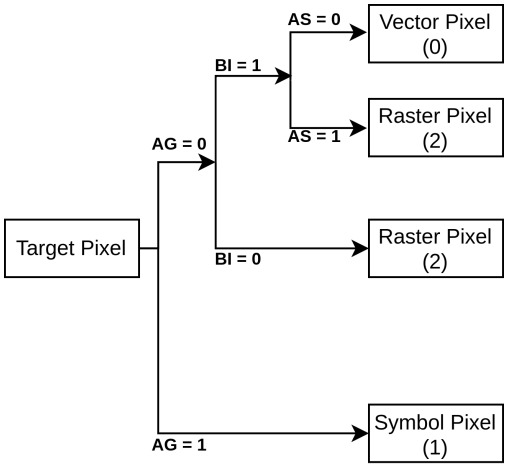


Figure 5: Pixel Classification Decision Tree.

Now that we have classified every pixel in the image into one of the three classes, namely, symbol, raster, and vector regions, we then draw bounding boxes to get symbol regions and raster regions. To find bounding boxes that fit our purpose, we first separate symbol pixels and raster pixels into two distinct masks, and then perform dilation separately on the two masks. To combine nearby texts or other symbol components, we apply one iteration of dilation with a 7×7 square kernel to the text mask. After that, we use bounding boxes to bound all connected components in both masks, with red for symbol regions and blue for raster re-

gions, respectively. After weeding out ineligible box candidates by constraining their heights and widths, we keep the rest for the following procedures.

Part 2. Image Quality Assessment

In this section, we present an approach to assess the quality of symbol regions in a scanned image at various resolutions and determine the minimum readable resolution for the image.

Because finding an exact resolution is not necessary for us, we simplify the output for minimum readable resolution to 4 tiers:

1. Tier 0: Minimum readable resolution is the base resolution (e.g. 600 dpi);
2. Tier 1: Minimum readable resolution is the base resolution divided by 2;
3. Tier 2: Minimum readable resolution is the base resolution divided by 4;
4. Tier 3: Minimum readable resolution is the base resolution divided by 8.

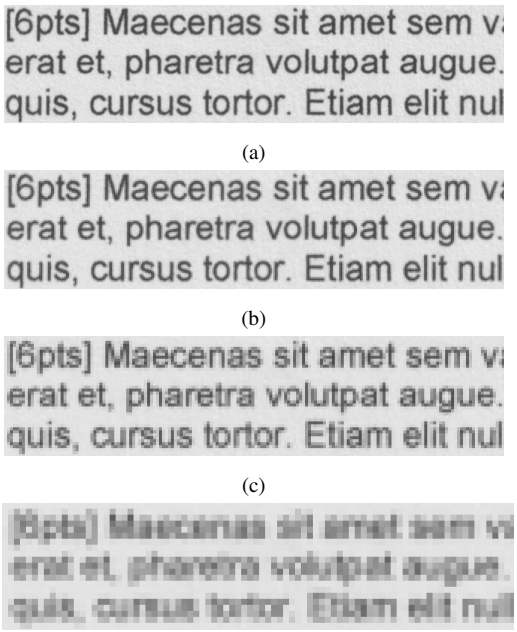
In our implementation, we choose 600 dpi as our base resolution, so the following tiers are 300 dpi, 150 dpi, and 75 dpi, respectively. In this section, we will use 600 dpi to represent the base resolution without further mention.

The process for our symbol-oriented quality assessment is presented in the second part of Figure 1. For an input image or sub-image containing symbols at 600 dpi, we first obtain its low-resolution versions at 300 dpi, 150 dpi, and 75 dpi. Then, we convert them to binary images using Otsu's method. After that, we calculate a set of three feature vectors from the four binary images by analyzing binary images at every pair of neighboring resolutions. Since we have identified four tiers for our output, we only need to consider images at four different resolutions, i.e. 600 dpi, 300 dpi, 150 dpi, and 75 dpi. Thus, a pair of neighboring resolutions means the two resolutions that are next to each other, for example, 600 dpi and 300 dpi, where the higher resolution (600 dpi) will serve as the reference and the lower resolution (300 dpi) will serve as the target. Then, we feed the feature vectors to a trained support vector machine (SVM) one by one, from higher resolutions to lower resolutions, until the SVM returns a positive output, indicating that the corresponding resolution is not readable for the input image. In this way, we are then able to determine the minimum readable resolution for the input image.

Merging Pixels, Breaking Pixels, and Hole-filling Pixels

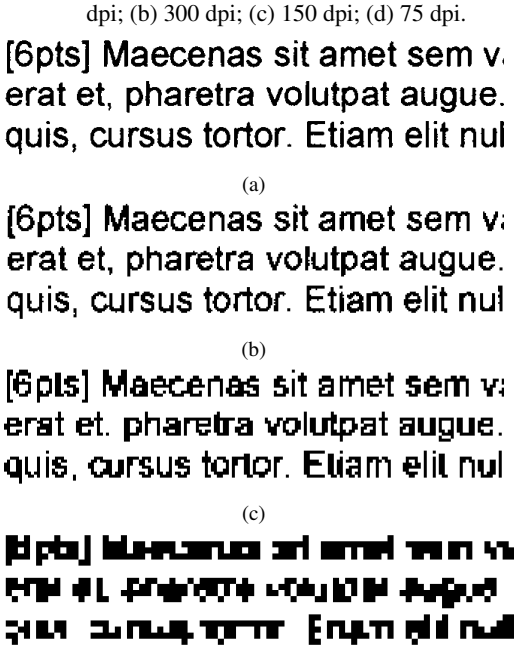
In our method, the determination of readability of symbols is based on the interactions among character glyphs exhibited in the binary images at various resolutions. To be specific, as the resolution decreases, we want to find some characteristic pixels that reflect the changing readability of an image or sub-image containing symbols.

Our observations on the degradation of text have shown that as we decrease the resolution, segments of the characters' glyphs start to merge or break, and holes and gaps between these segments will start to fill in in the binary image. Similar degradations are observed in other text-like content. Such degradation can be seen from the examples in Figure 7, which are the binary images obtained by applying Otsu's method to the samples shown in Figure 6.



(a) (b) (c) (d)

Figure 6: Image Crop at Different Scanning Resolutions: (a) 600 dpi; (b) 300 dpi; (c) 150 dpi; (d) 75 dpi.



(a) (b) (c) (d)

Figure 7: Binary Image Crop at Different Scanning Resolutions: (a) 600 dpi; (b) 300 dpi; (c) 150 dpi; (d) 75 dpi.

As the first step towards our goal, we defined three types of characteristic pixels that play important roles in the degradation:

1. Merging Pixel: In a binary image, pixels in different components that become connected or overlapped with each other as resolution decreases;
2. Breaking Pixel: In a binary image, pixels in the same com-

ponent that disappear and result in a breaking stroke as resolution decreases;
3. Hole-filling Pixel: In the binary image, pixels that emerge and fill holes (background surrounded by foreground pixels) as resolution decreases.

Examples of the three types of pixels are shown in Figure 8, where yellow represents merging pixels, cyan represents breaking pixels, and magenta represents hole-filling pixels. The three binary images are at 600 dpi, 300 dpi, and 150 dpi respectively. We investigated the impact of the number of merging, breaking, and hole-filling pixels on the severity of the degradation, and found a strong correlation. Such correlation can be observed from the number of yellow, cyan, and magenta pixels in the binary images across the different resolution tiers shown in Figure 8. In this example, we see that the quality seriously degrads at 75 dpi.

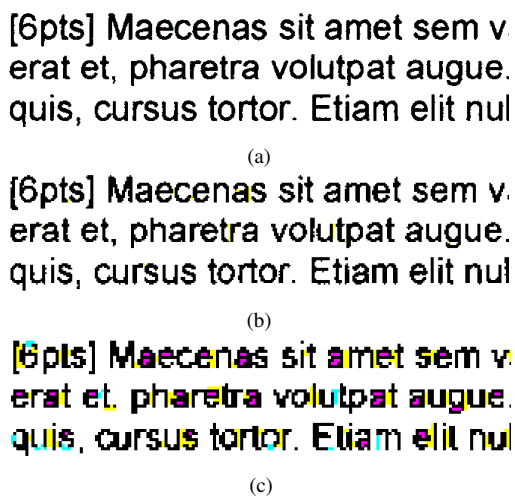


Figure 8: Merging (yellow), Breaking (cyan), and Hole-filling (magenta) Pixels: (a) Between 600 dpi and 300 dpi; (b) Between 300 dpi and 150 dpi; (c) Between 150 dpi and 75 dpi.

Feature Vectors and Minimum Readable Resolution

With the definitions of the three types of characteristic pixels, we want to count the numbers of these pixels between binary images at a pair of neighboring resolutions. We use m_j, b_j, and f_j to represent the numbers for the three types of pixels, respectively, with $j \in \{0,1,2\}$ representing the index of the image pairs.

To find m_j, b_j, and f_j, we proceed in the following way. We use two sliding windows of different sizes and different strides to scan through the reference binary image and the target binary image. The window sizes and strides for different resolutions are summarized in Table 1. Note that each stride is chosen to be half of its corresponding window size, so that we have overlapping windows. Using the window sizes and strides for the binary images at the corresponding resolutions, we will be able to get image patches matching the same contents in the two binary images. We can then find the three values m_j, b_j, and f_j, by comparing the two binary image patches.

With the two binary image patches, we then scale up the one at the lower resolution by pixel replication, so that the two image patches have the same size. For convenience, we name the scaled-up binary patch the "target patch", and the higher resolution binary patch the "reference patch". Overlaying the target patch and the reference patch, we will have three types of pixels, based on their existence in the two image patches. We name these pixels vanishing pixels, emerging pixels, and anchor pixels, as defined below:

Resolution	Window Size	Stride
600 dpi	48	24
300 dpi	24	12
150 dpi	12	6
75 dpi	6	3

Table 1: Window Sizes and Strides for Different Resolutions.

1. Vanishing Pixels: Foreground pixels in the reference patch that become background pixels in the target patch;
2. Emerging Pixels: Background pixels in the reference patch that become foreground pixels in the target patch;
3. Anchor Pixels: Foreground pixels in the reference patch that remain foreground pixels in the target patch.

Intuitively, emerging pixels are very likely merging pixels or hole-filling pixels, depending on whether they emerge on gaps between foreground lines or fill out isolated background blocks. In contrast, vanishing pixels are very likely breaking pixels, since they can cause breakages of lines or character strokes. Below, we describe how we find merging, breaking, and hole-filling pixels in a moving window.

Merging Pixels. We begin by finding all connected components in the reference patch and assigning each component a distinct label. After that, we copy the labels of all anchor pixels, leaving other pixels unlabeled, and form a new labeled map named LM_1. We then group the emerging pixels by connected components. For each emerging component, we look for labels of anchor pixels in LM_1 that are direct neighbors to the component, and collect their labels, forming a set S_1. If there are at least two labels in S_1, meaning that this component is linking two different strokes or lines in the reference patch, then pixels in this component are potential merging pixels.

Breaking Pixels. We begin by finding all connected components in the target patch and assign each component a distinct label. Similar to LM_1, we copy the labels of all anchor pixels, leaving other pixels unlabeled, and form a new labeled map named LM_2. We then group the vanishing pixels by connected components. For each vanishing component, we look for labels of anchor pixels that are direct neighbors to the components in LM_1, and collect their labels, forming a set S_1. Similarly, we also look for labels of anchor pixels that are direct neighbors to the components in LM_2, and collect their labels, forming a set S_2. If there is at most one label in S_1 and at least two labels in S_2, meaning that this component is breaking a stroke or a line from the reference patch into two or more parts in the target patch, then pixels in this component are potential breaking pixels.

Hole-filling Pixels. Hole-filling pixels can be found with merging pixels in a similar way. After we group the merging pixels by connected component, for each emerging component, we look for labels of anchor pixels in LM_1 that are direct neighbors to the component, and collect their labels, forming a set S_1. If there is at most one label in S_1 and there is no background pixel that is a direct neighbor to the component, meaning that this component is

filling a hole in the reference patch, then pixels in this component are potential hole-filling pixels.

Once we locate the potential merging, breaking, and hole-filling pixels in an image patch, we add 0.25 to all corresponding locations of these pixels in the corresponding accumulation map. We use the value 0.25 since every pixel will appear in four image patches using our choice of window sizes and strides. We will have three accumulation maps, each for one type of pixels among merging, breaking, and hole-filling pixels; and they will be initialized with zeros. After we iterate through all the image patches with the sliding window, we threshold the three accumulation maps and only keep those with values equal to 1 as valid merging, breaking, or hole-filling pixels. Finally, we count the total number of merging pixels, breaking pixels, and hole-filling pixels, respectively, and thus obtain values for m_j, b_j, and f_j.

Based on our calculation of m_j, b_j, and f_j, we designed three features as follows:

1. Merging Pixel Percentage M_i: The cumulative total of merging pixels between the base resolution and the target resolution, over the total number of foreground pixels at the base resolution;
2. Breaking Pixel Percentage B_i: The cumulative total of breaking pixels between the base resolution and the target resolution over the total number of foreground pixels at the base resolution;
3. Filling Pixel Percentage F_i: The cumulative total of between the base resolution and the target resolution hole-filling pixels over the total number of foreground pixels at the base resolution.

In the above definition, i is the index for feature vectors or target resolutions. Since we have three tiers of resolutions for every input image (300 dpi, 150 dpi, and 75 dpi), every input image will have three feature vectors. We use $i \in \{0,1,2\}$ to represent the three tiers of resolutions accordingly, ordered from high to low. The expressions for computing M_i, B_i and F_i are listed in Equation 1. In these expressions, m_j, b_j, and f_j represent the number of merging, breaking, and hole-filling pixels detected in the j^{th} pair of binary images at neighboring resolutions. N represents the total number of foreground pixels in the base resolution binary image, and 4^j represents a weight given to the number of pixels at different resolutions to account for the fact that the number of pixels at lower resolutions is less than that at higher resolutions for the same image.

$$\begin{cases} M_i = \frac{\sum_{j=0}^{i} 4^j m_j}{N} \\ B_i = \frac{\sum_{j=0}^{i} 4^j b_j}{N} \\ F_i = \frac{\sum_{j=0}^{i} 4^j f_j}{N} \end{cases}, \; for\; i \in \{0,1,2\} \tag{1}$$

With these three feature vectors, we then use a trained SVM to classify each feature vector, from the feature vector of the highest target resolution to the feature vector of the lowest target resolution. The SVM will take a feature vector as input, and return either 1 or 0, with 1 indicating unreadable and 0 the opposite. Therefore, getting a 0 from the SVM means that the current target resolution is a readable resolution for the input image or sub-image. When we get a 1, we stop the process, and claim that the previous readable resolution is the minimum we can get. For example, if the SVM returns 1 for the first feature vector, i.e. (M_0, B_0, F_0), then we say that the minimum readable resolution tier is Tier 0, or 600 dpi in our implementation. If we get 0 for all 3 feature vectors, the minimum readable resolution tier is then Tier 3, or 75 dpi in our implementation. These correspondences are summarized in Table 2, where P_i represents output for each feature vector, $i \in \{0,1,2\}$, from the SVM. In the table, "x" means "don't care".

Output (P_0, P_1, P_2)	Minimum Readable Resolution Tier
(1,x,x)	Tier 0
(0,1,x)	Tier 1
(0,0,1)	Tier 2
(0,0,0)	Tier 3

Table 2: From SVM Predictions for Feature Vectors to Minimum Readable Resolution.

Training the Support Vector Machine

In our method, a support vector machine (SVM) is used to discriminate feature vectors of images at readable resolution from feature vectors of images at unreadable resolution. Note that "readable" and "unreadable" here are subjective opinions ideally given by language experts or by a group of study subjects. For this paper, the opinions were rendered by the first co-author. The SVM in our system takes in a feature vector each time and return either 0 or 1 as output, with 0 indicating "readable" and 1 indicating "unreadable".

To gather training samples, we first generated 100 pages of MS Word documents containing random horizontal English texts in different typefaces and sizes. After that, we printed the documents and scanned them at 600 dpi without any post-processing. We than sliced the scanned documents horizontally into smaller pieces, each containing several lines of text. This gave us approximately 1000 image slices. Using the method we introduced previously, we computed a set of three feature vectors for each image slice. As a result, we had 3000 feature vectors as our training samples. To get ground truth labels for these feature vectors, we scanned the printed documents again, but at all four tiers of resolutions this time and with post-processing like denoising, sharpening, and JPEG compression. We then examined the processed scanned documents at various resolutions one by one and made a judgement of the minimum readable resolution for every image slice based on its actual viewing quality on a screen. Based on our judgements, we then assigned ground-truth labels 0 or 1 to every feature vector accordingly, according to Table 2. Finally, we fitted an SVM model using the training feature vectors with their ground truth labels.

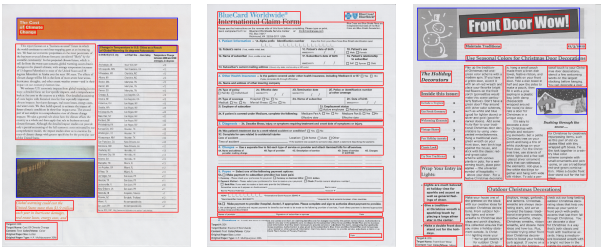
Experimental Results

Page Segmentation

To qualitatively demonstrate the effectiveness of our page segmentation algorithm, we show some examples of segmented images from our testing set in Figure 9. For visualization, we use red bounding boxes for symbol regions and blue bounding boxes for raster regions.

To quantitatively evaluate the performance, we built a page segmentation dataset containing 100 pages scanned at 100 dpi. The ground-truth bounding boxes were obtained by manually drawing on the images with a bounding box drawing software

tool. The resulting average intersection over union (IoU) is 0.81 for symbol regions and 0.60 for raster regions.



(a) (b) (c)

Figure 9: Page Segmentation Results.

Minimum Readable Resolution Determination

To test the effectiveness of our designed features, we built two separate dataset that contains images of English text and Chinese text. From each dataset, we extracted 3000 feature vectors, each with an associated ground-truth value of either 0 or 1 based on our judgements of the minimum readable resolutions of images in the dataset.

From each dataset, among the 3000 feature vectors and ground truth we generated, we selected one fifth of them as testing data, and the rest as training data for 5 times. The 5-fold validation accuracy are 95.1% for English text dataset and 91.6% for Chinese text dataset.

Conclusion

In this paper, we proposed a novel document image quality assessment method to determine the minimum readable resolution of scanned documents or regions in scanned documents. Our experiments successfully demonstrated the effectiveness of our proposed methods. For now, our system is only tested with English and Chinese texts. But the same idea can be easily expanded and applied to text in other languages, as well as other symbol contents that share similar characteristics with texts.

References

[1] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[2] C. Wick and F. Puppe, "Fully convolutional neural networks for page segmentation of historical document images," *CoRR*, vol. abs/1711.07695, 2017. [Online]. Available: http://arxiv.org/abs/1711.07695

[3] S. K. Yip and Z. Chi, "Page segmentation and content classification for automatic document image processing," in *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing. ISIMP 2001 (IEEE Cat. No.01EX489)*, May 2001, pp. 279–282.

[4] D. Drivas and A. Amin, "Page segmentation and classification utilising a bottom-up approach," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 2, Aug 1995, pp. 610–614.

[5] T. Pavlidis and J. Zhou, "Page segmentation and classification," *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 6, pp. 484 – 496, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/1049965292900689

[6] Z. Xiao, M. Gao, L. Wang, B. M. Bradburn, and J. P. Allebach, "Digital image segmentation for object-oriented halftoning," in *Color Imaging XXI: Displaying, Processing, Hardcopy, and Applications (Part of IS&T Electronic Imaging)*, 2016.

[7] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, vol. 2, Nov 2003, pp. 1398–1402 Vol.2.

[8] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "FSIM: A feature similarity index for image quality assessment," *IEEE Transactions on Image Processing*, vol. 20, no. 8, pp. 2378–2386, Aug 2011.

[9] Zhou Wang, H. R. Sheikh, and A. C. Bovik, "No-reference perceptual quality assessment of JPEG compressed images," in *Proceedings. International Conference on Image Processing*, vol. 1, Sep. 2002.

[10] T. Brando and M. P. Queluz, "No-reference image quality assessment based on DCT domain statistics," *Signal Processing*, vol. 88, no. 4, pp. 822 – 833, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168407003337

[11] H. R. Sheikh, A. C. Bovik, and L. Cormack, "No-reference quality assessment using natural scene statistics: JPEG2000," *IEEE Transactions on Image Processing*, vol. 14, no. 11, pp. 1918–1927, Nov 2005.

[12] A. K. Moorthy and A. C. Bovik, "A two-step framework for constructing blind image quality indices," *IEEE Signal Processing Letters*, vol. 17, no. 5, pp. 513–516, May 2010.

[13] M. A. Saad, A. C. Bovik, and C. Charrier, "A DCT statistics-based blind image quality index," *IEEE Signal Processing Letters*, vol. 17, no. 6, pp. 583–586, June 2010.

[14] W. Lu, K. Zeng, D. Tao, Y. Yuan, and X. Gao, "No-reference image quality assessment in contourlet domain," *Neurocomputing*, vol. 73, no. 4, pp. 784 – 794, 2010, Bayesian Networks / Design and Application of Neural Networks and Intelligent Learning Systems (KES 2008 / Bio-inspired Computing: Theories and Applications (BIC-TA 2007). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231209003890

[15] X. Gao, W. Lu, D. Tao, and X. Li, "Image quality assessment based on multiscale geometric analysis," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1409–1423, July 2009.

[16] D. Tao, X. Li, W. Lu, and X. Gao, "Reduced-reference IQA in contourlet domain," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1623–1627, Dec 2009.

Author Biography

Litao Hu received his BS in Electronic Engineering from the Hong Kong University of Science and Technology (2017) and is currently a PhD candidate in Electrical and Computer Engineering at Purdue University. As a research assistant in the Electronic Imaging System Laboratory, his research interests include image processing, machine learning, and deep learning.

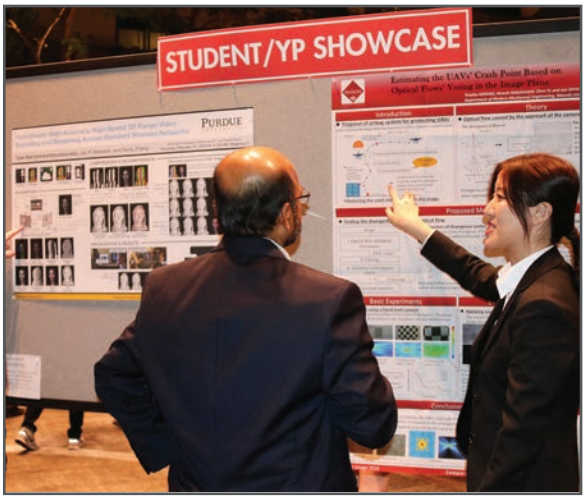
JOIN US AT THE NEXT EI!

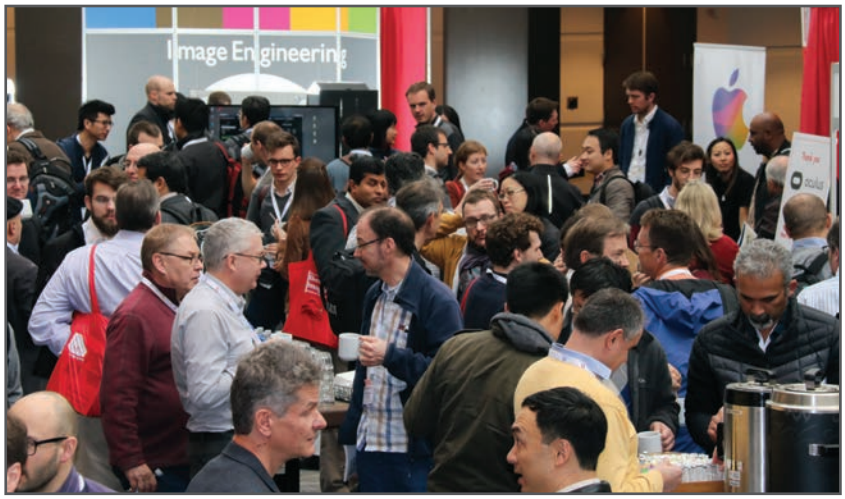
IS&T International Symposium on

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