A Learning-Based Approach for Word Segmentation in Text Document Images

Jean-Pierre Lomaliza, Hanhoon Park and Kwang-Seok Moon
Department of Electronic Engineering, Pukyong National University, Busan, South Korea
ksmoon@pknu.ac.kr

***Abstract*—In conventional document retrieval (DIR) systems based on locally likely arrangement hashing (LLAH), the word detection approach is sensitive to the distance between the camera and the text document, e.g. a single word may be detected as several words when the camera is too close. Thus, the systems work well only when the distance in which the text document was registered is similar to the one of the retrieval. Moreover, they were implemented in a desktop setup where it might not suffer from the distance problem since the camera is rigidly attached to the computer. In this paper, a new word segmentation approach is proposed to increase the robustness of LLAH-based DIR systems so that they may be implemented on a mobile platform where the distance between the camera and text document may be easily changeable. The proposed method uses a deep neural network to classify spaces between connected components as between-words space or intra-word space. From experiments results, the proposed method successfully could detect the same words in different camera distances and orientation as the neural networks offered classification accuracy as high as 92.5%. Moreover, it showed higher robustness than the state-of-the-art methods when implemented on a mobile platform.**

***Index Terms*—Word Segmentation; Deep Learning; Space Classification; Locally Likely Arrangement Hashing; Document Image Retrieval.**

I. INTRODUCTION

As there are numerous scanned images of old text documents or publications that existed before the era of digitalisation, there is a need to provide a fast and efficient technique to store and retrieve such document images in a large database. Several publications have proposed fast document image retrieval (DIR) systems [1-8]. According to the survey in [2], there are two groups of DIR systems: the recognition-based ones, where a document is recognized in its whole image and the similarity between documents is measured at symbolic level; and recognition-free ones, which rely on extracted features from the content of text documents.

The main idea of recognition-based DIR systems is to exploit some global information such as contrast or letters occurrences to recognise a stored text document. Using semantic properties of the text document through optical character recognition (OCR) has been a traditional approach in the recognition-based DIR systems. However, the OCR-based approach has disadvantages such as high computation time, sensitivity to the resolution of the image and language dependency as stated in [3].

Another recognition-based approach proposed to use global features [4, 5]. It considered texture, shape, size and the position of the text document as global features and used them for the storing or retrieval process. Among visual texture properties used as a texture, features are coarseness, contrast, directionality, line likeness, regularity, and roughness. In [6], the authors used global information such as the structural arrangement of each text document and the relationship within surrounding areas to form features of each text document. However, such global-information-dependent methods are vulnerable to occlusion.

As opposed to recognition-based approaches, the main idea of recognition-free systems is that, instead of storing the whole image of the document, which can be very expensive in the memory consumption, features of the document are stored altogether with corresponding hashing indexes for fast storing or retrieving process and cheap memory consumption. Some recognition-free methods proposed pixel-level descriptors for storing or recognising a text document image. Using as a descriptor, a histogram oriented gradient (HOG), that is a technique which counts occurrences of the gradient orientation in a local part of an image, authors in [7] retrieved stored documents images. Potential local characters have been detected with respective locations using HOG features extracted by sliding multi-scale window. Additionally, they used a support vector machine (SVM) to spot characters of a word in a document. Instead of generating pixel-level descriptors, other researches focused on representing a text document using local features such as words to be stored or retrieved. Especially the locally likely arrangement hashing (LLAH) based DIR system in [1] proposed to extract connected components as words, use local invariants from neighbour words to compute features that can describe a text document. However, it suffered from limitations concerning the distance between the camera and the text document. This limitation needs to be addressed to implement LLAH-based DIR systems on a mobile device, which is the main concern of this paper.

The remainder of this paper is organised as follows. First, after a state-of-the-art word segmentation method in LLAH-based DIR systems is introduced in Section II, the proposed method is elucidated in Section III. Then, the experimental results and limitations of the proposed method and future studies are discussed in Section IV and Section V. Finally, the conclusion is drawn in Section VI.

II. STATE-OF-THE-ART METHOD

Authors in [1] proposed an LLAH-DIR system, where a large number of text documents could be stored in the database and retrieved in a fast, discriminative and accurate way. They proposed to use local features to describe a text document. To do so, they proposed to use words of the text document. It is important here to specify that they did not use semantic way (such as OCR) to detect words. Instead, they

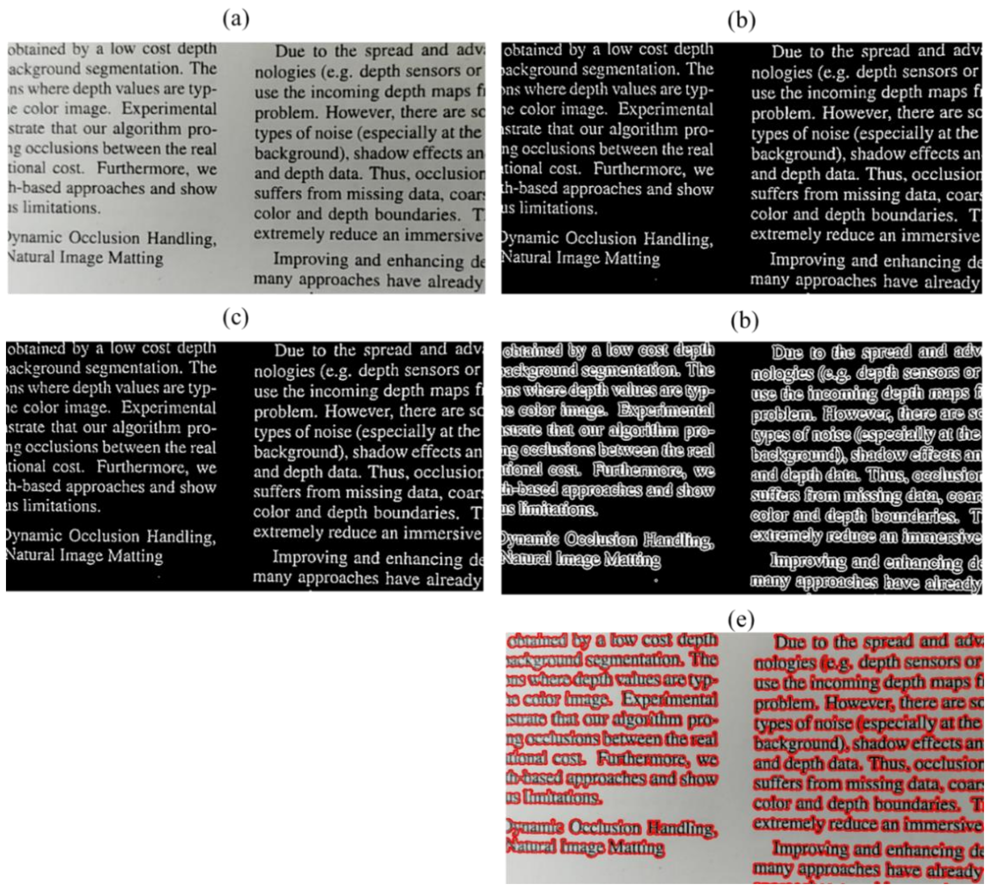

Figure 1: Connected component detection steps proposed in [1]. (a) Original image, (b) binary image after adaptive thresholding, (c) binary image after Gaussian blurring, (d) binary image after second thresholding, (d) detected connected components using image of (d) drawn on top of image (a).

simply detected connected components and considered them as words of the text document. To do so accurately, they proposed to first adaptively threshold the input image in order to get a binary image version of the input image as shown in Figure 1(b). As the binary image may contain some salt-and-pepper noises, they proposed then to apply a Gaussian blurring as shown in Figure 1(c). And finally, another thresholding was applied to strengthen edges that were weakened by the blurring operation. After those three operations, connected components were detected using the image shown in Figure 1(d). Those connected components were then considered as words, and their centroids were computed to be used later for features computation.

To compute local invariants, for each detected word, the seven closest neighbour words were found. Out of those seven detected words, six different combinations were performed, and each of those combinations was used to compute the features descriptor of the word. The idea of using different combinations is due to the fact that, a single word may have different closest neighbours from an image to another because of the perspective rotation of the camera. Moreover, they stated that, out of seven neighbour words, at least six might be similar in the different perspective angle of capture. Hence they used different combinations of those neighbours to compute several different features of the same word.

Each six-word combination was then ordered angle-wise. After ordering, for each six-word combination, all the arrangements of four words (e.g. ABCD where each of those letters represents the centroid of each word) were computed. Each four-word arrangement was used to compute invariant features by computing the ratio between the area of two bisectional triangles (e.g., ACD and ABC for ABCD). This was based on the fact that the triangle ratio is a robust feature that is invariant to camera rotations and distances. Finally, each six-word combination was represented as a sequence of those triangle ratios. Those sequences were then used to compute hash table indexes to store or retrieve each word. It appears clear here that, each word will have several feature representation and several hash table entries. Their system appeared to be very accurate and discriminative in document retrieval process due to the complexity of representative features. Moreover, since features were computed locally, their method showed very good robustness in case of occlusion. Nevertheless, their method was fast enough to be used in an augmented reality (AR) application for text annotation purpose proposed in [8]. However, their method was proposed to run on a desktop platform where the camera distance to the text document is mostly constant. Thus, their words detection step was mostly accurate to detect the same connected components even in case of different orientations. As our purpose is to provide such a system in a mobile platform, it is very important to consider the variation of the distance between camera and text document. Figures 2(a) and 2(c) show that a word detected as one in a camera distance of approximately 15 cm is detected as several words in a distance of approximately 8 cm. In such situations, the LLAH algorithm would more likely fail to identify several words and decrease its accuracy. Hence, in this paper, we propose to address such problem by proposing an algorithm that would detect similar connected components (words) in different camera distance by analysing spaces between those connected components within each line.

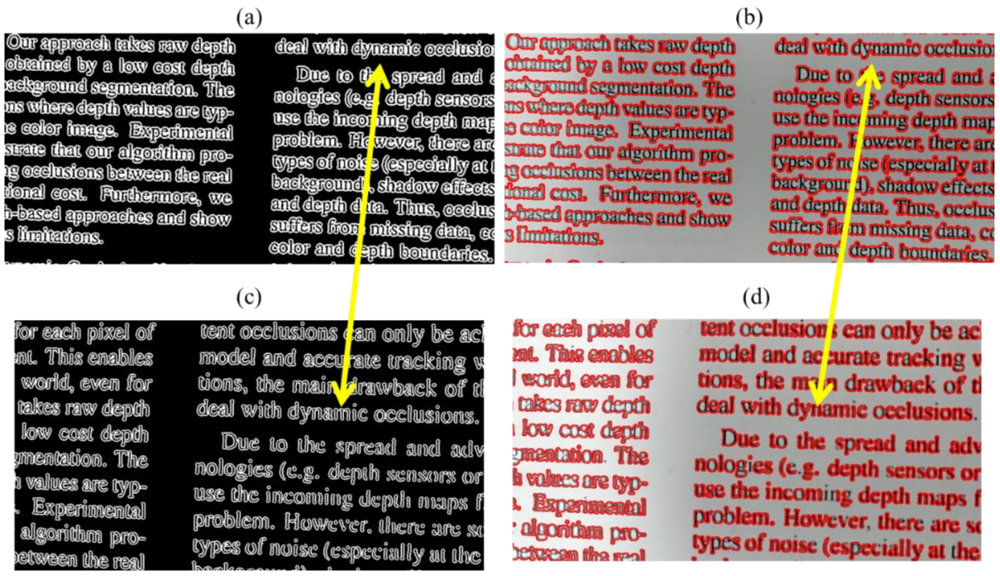

Figure 2: Connected component detection results of [1] at different camera-document distances. (a) Binary image of document captured at a distance of 15 cm, (b) detected connected components (in red contours) from (a), (c) binary image of document captured at a distance of 9 cm, (d) detected connected components (in red contours) from (c). In (a) and (b) the word "dynamic" is connected at a connected component while in (c) and (d) it is detected as six different connected components.

III. Proposed Method

The proposed method focuses on detecting same words (connected components) in different distances between the camera and the text document in order to make the LLAH-based DIR system robust to the distance change and so that it would be implanted on a mobile platform. To do so, the proposed method first detects connected components as done in [1] and then tries to classify spaces between those components as being intra-word space or between-words space. Thus, it appears important here to detect each text document line so that spaces between those connected components may be detected and classified. Hence, the proposed method includes the following steps: First, the text document's orientation is estimated. Second, the text document's lines are segmented. Third, word spaces are detected. And finally, the spaces are classified using a deep neural network (DNN).

A. Text Document Orientation Estimation

To perform text document line segmentation that will introduce in the next section, it is important here to first estimate the text document's orientation. First, the three operations proposed in [1] are done here to detect connected components. As explained before those operations consist of adaptive thresholding, Gaussian blurring, and another thresholding. To improve the computation time of this process, the obtained binary image is then downscaled in a factor of 4. Each connected component in the downscaled image is used then to estimate the text document orientation.

For each connected component, all non-black pixels are considered as points with respective coordinates in the source image. And then for each set of points (i.e. each connected component), a principal component analysis (PCA) is performed to get the first and second eigenvectors that will define orientations in which the connected component varies the most. Finally, the first eigenvector of each connected component is then used to estimate the whole text document's orientation by considering the peak of the smoothed histogram of orientations of connected components.

Different results of text document's orientation estimation are shown in Figure 3. It is important to specify that, as the mobile environment, such as Android provides a multi-threading capacity where operations may be parallelised, orientation estimation of connected components are done in parallel processing to boost the computation time.

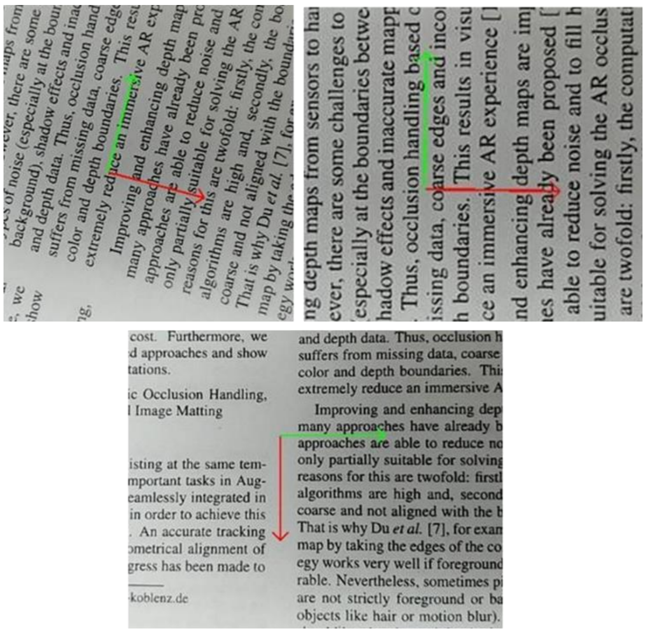

Figure 3: Estimated orientations (coloured arrows). Green arrows are representations of eigenvectors associated to largest eigenvalues, and red arrows are representation of those associated to the second largest eigenvalues.

B. Text Document Line Segmentation

To segment the text document image into lines, the estimated orientation angle computed in the previous section is used. First, a black image of the size of the input image is created. Then rotated rectangles are drawn using text document's rotation angle, the size of each connected component and also its centroid. For each connected component of width W and height H, a rotated rectangle with

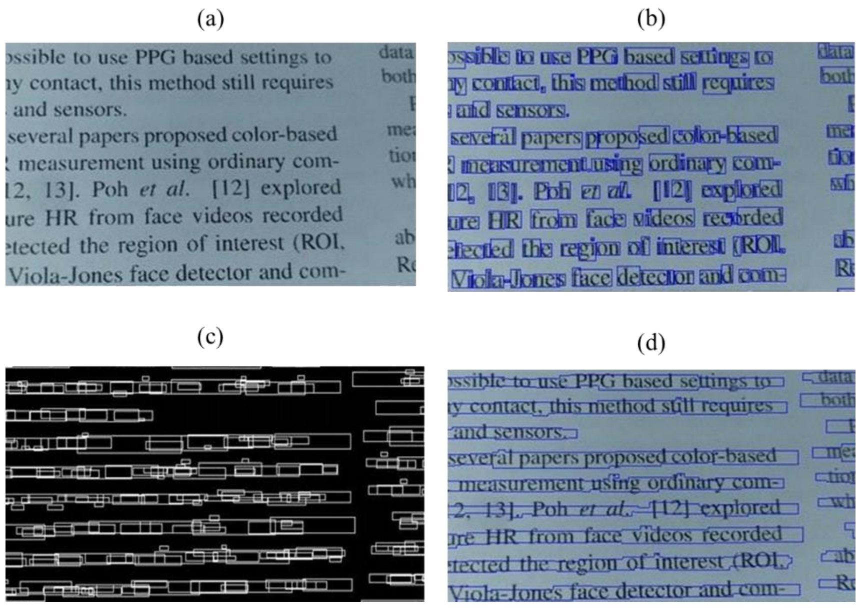

Figure 4: Text line segmentation process. (a) Original image, (b) detected connected components, (c) drawn scaled and rotated rectangles in a dark image, (d) detected line boundaries.

the width of 1.5 × W and height of 0.7 × H is drawn in the dark image as shown in Figure 4(c). The reason for increasing width size is to make rotated rectangles of each line touch each other. The reason for decreasing the height size is to avoid rotated rectangles of different lines to touch each other. As shown in Figure 4(c), all rotated rectangles of same lines touch each other. After the drawing process, a contour detection operation is performed again by considering only outer contours. This detection step allows detecting each line-possible boundary. Finally, boundaries detected from the black image and shown in Figure 4(d) are used to construct each line. A boundary (line) contains a word if the word's centroid is inside it. In this way, all words are grouped inside respective lines. Here also the word grouping process may be done in parallel processing to minimise the computation time.

C. Between Word Space Detection

This step aims to determine spaces between words in each line. Those spaces are then used later for the classification process. First, in each line, the middle line is estimated. The middle line is found with the least square fitting algorithm for centroids of all words of the same line.

After finding each middle line, a histogram is computed in each line by counting the points (non-black pixels) orthogonally projected onto each middle line. Then, each histogram is smoothened. Finally, space areas are detected as those having 1 or 0 projections. Refer to Figure 6.

D. Space Classification Using a Deep Neural Network

This step is the most important part for the contribution of the proposed method as the accuracy depends on the accuracy of the classification of spaces between connected components. To achieve this task, a DNN shown in Figure 5 is used. First, 1000 documents images were captured from different types of Android smartphones (Samsung Galaxy SII, LG G5, and Pantech Vega Racer 3), different orientations and different distances between the camera and the text document with a maximum distance of 30 cm.

As in each text document image 150 connected components are detected on average with a similar amount of potential spaces, our dataset to feed into the DNN has 151,302 samples. A data set collector program was then written in Java to process those training images and label each of detected spaces as intra-word space or between-words space.

As shown in the Figure 5 and Figure 6, the proposed DNN has an input layer with seven features describing the space width (f_1), the average of word widths within the line (f_2), the average of word heights within the line (f_3), the angle of text document (f_4), the width of the smallest word adjacent to the current space (f_5), the average space sizes within the line (f_6), and finally the ratio between line width and height (f_7). The DNN has four hidden layers of eighteen nodes each and an output layer of two nodes that define whether space is intra-word space or between words space through a probability distribution computed from a Softmax activation function. As one may guess, all these features information are collected during the training dataset collecting process.

It is important here to specify that the library used for the machine learning part is the Google's Tensorflow library [9] that comes with a flexible way to implement deep learning algorithms through the Python programming language. The training process is done in desktop PC, and the generated model is then used in the android application to classify spaces between connected components.

For the mobile application part, the Jetbrains's android studio [10] was used as development platform together with OpenCV [11] as image processing library and Tensorflow's trained model to classify detected spaces.

IV. Experimental Results

As mentioned in Section III.D, we first implemented a data collector and training programs on desktop PC. The used development platform was IntelliJ Idea [12]. The OpenCV library was used for image processing parts such as thresholding, blurring, PCA computation and contour detection.

For the experiment, 1000 document images were captured using the LG G5 device, the Samsung Galaxy SII device, and the Pantech Vega Racer 3 with the standard camera resolution of 640 × 480. Those images were captured in different ambient lighting situations, different camera orientations and

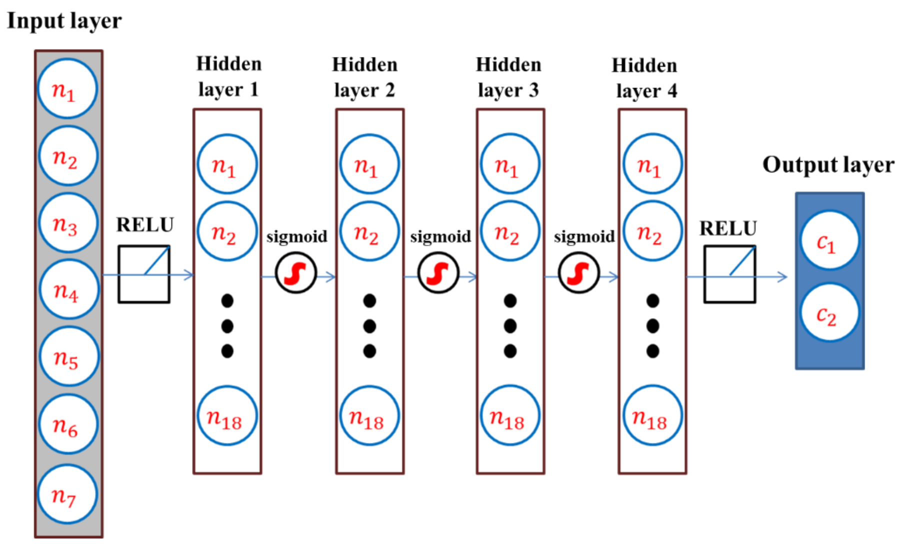

Figure 5: The proposed DNN for space pattern classification. One input layer with seven features as input nodes, four hidden layers with eighteen nodes each and one output layer with two nodes as two possible classes having each a probability to be the estimated class.

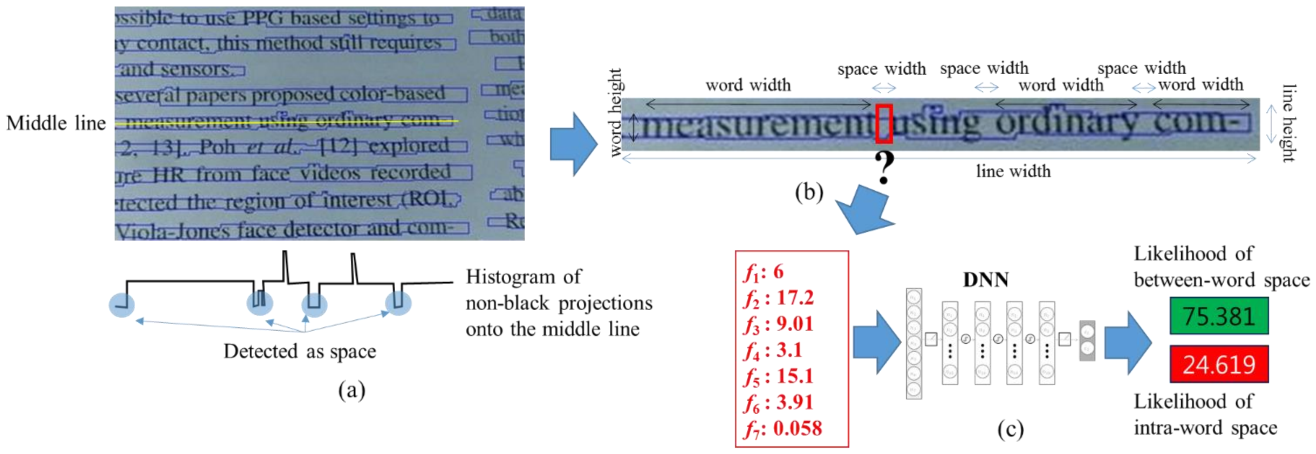

Figure 6: Space detection and classification. (a) histogram computation and space detection, (b) feature extraction for a space (marked by the red rectangle), (c) space classification by feeding in the features into DNN. The marked space is identified as between-word space.

different distances between the camera and the text documents. The capture distance varied between 5 cm to 30 cm. The choice of 5 cm as the minimum distance is because of technical limitations in a practical situation where the user usually holds the smartphone from the back, and such hand depth size might be estimated to 5 cm. In the other hand, the limitation of 30 cm as the maximum distance is also discussed in the next section where it is explained that over such distance several words are detected as a single word, and such problem will be subject for future studies.

The collector program was used to label each space and save feature information in a CSF format document that was then used later in the training program to generate the learned model through Tensorflow. The Pycharm IDE [13] was used for the training process together with Tensorflow. And finally, the trained model was imported into the Android application to classify spaces.

In the training process, the dataset was split into three subsets: 70% of the data for training, 15% for cross-validation, and 15% for testing. After training, the model had a 92.5% of accuracy in the testing samples. Figure 7 shows a part of the space identification results which were obtained through the processes depicted in Figure 3, Figure 4, and Figure 6. With the proposed method, words with different scales could be accurately detected unlike failed in Figure 2.

To measure the accuracy of the proposed method compared to the conventional LLAH-based one proposed in [1], we first prepared a set of 200 documents where words were manually marked. Then those marked words were compared to words that were detected by both methods. Each marked document was captured in three different distance (1-3 cm, 3-7 cm, and 7-15 cm) from the camera and three different random orientations. Table 1 shows average accuracies of both methods and shows how the proposed method improves the accuracy of the conventional one when facing dramatic changes of camera distance between the storing process and the retrieving process (the last line of Table 1). In Table 1, where the average number of stored words per document was 127.5, the first column expresses the distance (camera to the text document) difference between the storing process and the retrieving process. The second and third columns represent respectively the average number of words recovered (recognised) by the method proposed in [1] and the method proposed in this paper. As observed in Table 1, the method from [1] had much low accuracy as the distance change was getting large. Although the proposed method was better, it also detected much fewer features when the distance difference was large. However, as all features are stored altogether with their respective coordinates in the 2D image, it would require only at least three features to be recovered in

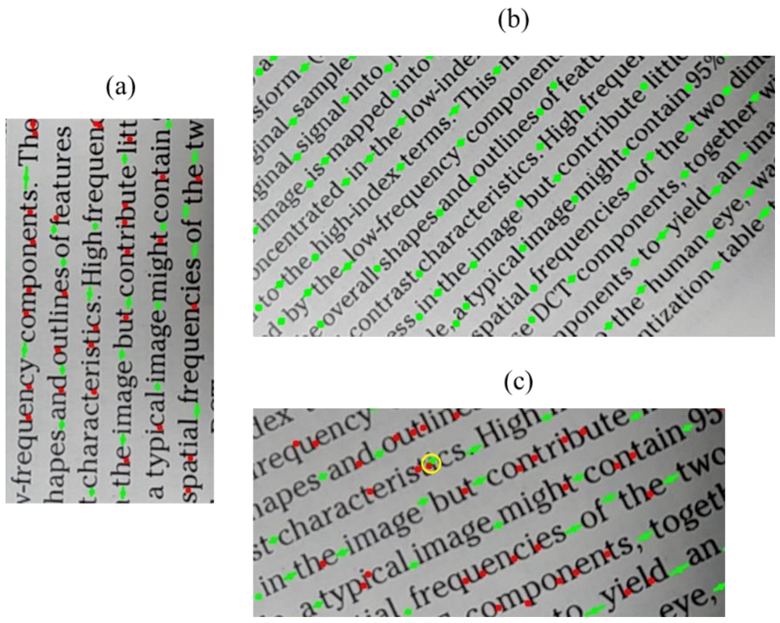

Figure 7: Space classification using the model generated with our trained DNN. Green lines (or dots) are spaces that are classified as between-words spaces and red lines (or dots) are those classified as intra-word spaces. (a) Image captured at approximately 8 cm of distance, (b) image captured at approximately 28 cm of distance, (c) image captured at 15 cm of distance with one miss-classified sample circled in yellow.

order to compute transforms that may help to estimate locations of undetected features in an AR application situation. Hence, the proposed method maintains solid usability in such situations.

Table 1
Comparison of the Results Using the Proposed Word Detection Method and That Proposed in [1]

Distance difference in cm	Average number of words detected by [1]	Average number of words detected by the proposed method
1 – 3	91.32	85.50
3 – 7	48.10	61.75
7 – 15	1.56	9.73

V. Limitations and Future Studies

This paper has addressed the limitation in which the state-of-the-art LLAH-based DIR system could not retrieve well the same text documents when storing distance and retrieving distance were greatly different. However, it limited the maximum distance between the camera and the text document to 30 cm. The maximum distance was set due to the fact that, over such distance, several words were detected as one word and even in some cases, a whole sentence was detected as a single connected component. This effect is mostly caused by the Gaussian blurring steps during the pre-processing of the image. Thus our future study will aim to increase such limitation on the capturing distance.

Besides the distance limitation, the proposed method also relies on the fact that all lines on a text document would have a similar orientation to compute the whole document’s orientation. However, this assumption may not be correct for some text documents having different text orientations. Thus, future studies will also focus on detecting text document orientation in the local level.

Moreover, even though several steps of the proposed method were implemented to run in parallel processing, the average computation time was 351.9 milliseconds as compared to one of the state-of-the-art systems that is at 85.31 milliseconds in our implementation. This makes the proposed method only usable for text document retrieval systems, but not practical in a real-time system such as AR applications. Thus, our future studies will try to reduce the computation time so that the proposed method may run in such AR systems.

VI. Conclusion

In this paper, we proposed a learning-based word segmentation method. The proposed method addressed the distance problem that weakened the conventional LLAH-based DIR systems by first detecting spaces between words in a text document and classifying through a DNN those spaces based on surrounding information such as space size, average space size in the current line, and average word width and height in the current line. Experimental results showed that the proposed method has better ability to recognise similar words even when distances during storing and retrieving processes were significantly different. However, the proposed method had some limitations such as maximum distance and processing time. Those limitations will be subjects of future studies.

Acknowledgement

This work was supported by a Research Grant of Pukyong National University (2017). Corresponding author: Kwang-Seok Moon.

References

[1] K. Takeda, K. Kise, and M. Iwamura, “Real-time document image retrieval for a 10 million pages database with a memory efficient and stability improved LLAH,” *Proc. of ICDAR*, pp. 1054-1058, 2011.

[2] F. Alaei, A. Alaei, M. Blumenstein, and U. Pal, “A brief review of document image retrieval methods: recent advances,” *Proc. of IJCNN*, 2016.

[3] A. Gordo, J. Gibert, E. Valveny, and M. Rusinol, “A kernel-based approach to document retrieval,” *Proc. of DAS*, 2010.

[4] A. Gordo, F. Perronnin, and E. Valveny, “Large-scale document image retrieval and classification with runlength histograms and binary embeddings,” *Pattern Recognition*, vol. 46, no. 7, pp. 1898-1905, 2013.

[5] M. Shirdhonkar, and M.B. Kokare, “Handwritten document image retrieval,” *Proc. of ICCMS*, pp. VI-506–VI-510, 2011.

[6] R.M. Haralick, K. Shanmugam, and I.H. Dinstein, “Textural features for image classification,” *IEEE Trans. on Systems, Man and Cybernetics*, vol. 3, no. 6, pp. 610-621, 1973.

[7] A. Mishra, K. Alahari, and C.V. Jawahar, “Image retrieval using textual cues,” *Proc. of ICCV*, 2013.

[8] H. Uchiyama, J. Pilet, and H. Saito, “On-line document registering and retrieving system for AR annotation overlay,” *Proc. of AH*, no. 23, pp. 1-5, 2010.

[9] Google Tensorflow is a machine learning library for python, https://www.tensorflow.org [Online; accessed 16-Jan-2018]

[10] Android Studio, http://www.developer.android.com/studio/index.html [Online; accessed 16-Jan-2018]

[11] OpenCV, http://www.opencv.org [Online; accessed 16-Jan-2018]

[12] IntelliJ idea, http://www.jetbrains.com/idea [Online; accessed 16-Jan-2018]

[13] Pycharm development tool for python, https://www.jetbrains.com/pycharm [Online; accessed 16-Jan-2018]