A Framework for Word Segmentation in Images using Density-based Clustering

Hui Guo [1] and Qin Ding [2]
[1] North Carolina State University, USA
[2] East Carolina University, USA
hguo5@ncsu.edu, dingq@ecu.edu

Abstract

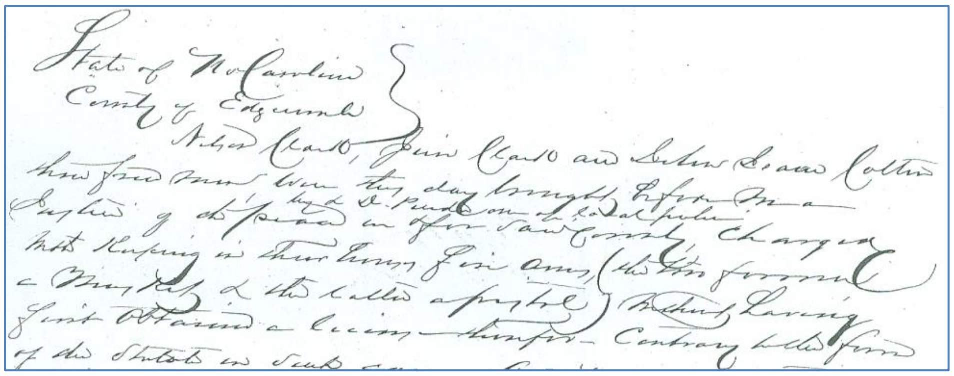
Word recognition is to identify words in images of printed or handwritten documents. It is especially challenging to recognize words from cursive handwriting documents. In this paper, we present a framework of using density-based clustering for word segmentation in printed or handwritten documents, including cursive handwriting. First, we performed various strategies for data preprocessing, including converting images to B/W images, adjusting the tilted images, and removing the background noises. K-means clustering and/or neighborhood density are used in finding parameters for the preprocessing steps. The preprocessing has shown to be very effective. For the word segmentation, we proposed density-based clustering to segment the words using multiple steps, including blurring, plotting, and clustering. We also developed a system for the framework, including preprocessing and clustering functionalities. Our approach works very well for printed documents. It works reasonably well for handwriting documents if words are relatively far from each other. The performance on handwriting documents can be further improved by using line segmentation.

1 Introduction

Word recognition is to identify words in images of printed or handwritten documents. There are abundant handwritten documents available, which carry a lot of potentially useful information. Handwriting recognition, also called handwritten text recognition, is a challenging task, especially cursive handwriting recognition [1, 2]. Extensive research has been done on handwriting recognition [3, 4], especially deep learning and neural network methods [5, 6, 7], but most of them are based on letter recognition. In this paper, we propose to perform word recognition instead of letter recognition, because for cursive handwritings, the latter is too difficult and unreliable. The first step of this process will be image segmentation in which the entire documents are segmented into words.

A sample image of cursive handwritten document is given in Figure 1. As can be seen from this sample, the data has the following characteristics:

- The useful information is in black and white, but the image has a gray scale and there are noise colors on the background.
- The handwriting is tilted, but the lines are fairly straight.
- There are noise pixels on this image, which could be ink, dust or dirt.
- Words overlap, especially the ones from different lines. Sometimes the distances between lines are smaller than the distances between words on the same line.
- Words have arbitrary sizes and shapes.



State of No Carolina
County of Edgecombe
Nelson Clark, [illegible] Clark and [illegible] Colton
three free men were this day brought before me a
by [illegible] on oath [illegible]
Justice of the peace in & for said County, charged
with keeping in their [illegible] fire arms (the former
a musket & the latter a pistol) without having
first obtained a license therefor contrary to the form
of the Statute in such [illegible]

Figure 1. A sample cursive handwriting image

The images of handwritten documents are usually scanned from paper documents. In order to get a good quality, the resulting image has to have a large number of pixels. All the characteristics makes the process challenging to segment cursively handwritten documents into separated images of words. Previous work has been done on this kind of segmentation using neural network [8]. In this paper, we will tackle this problem using data mining approach, in particular, density-based clustering approach.

This image segmentation problem differs from others in that for handwriting documents, words are in lines and they are not in random locations. Words on different lines should be separated, even though they may be connected physically. Therefore, the first step should be to find the lines. And line information can also help adjust tilted images, when it is assumed that all lines are horizontal.

From a data mining point of view, word segmentation resembles clustering of the dark pixels. In addition, since the clusters have arbitrary shapes, density-based clustering seems promising for an acceptable result. In this paper, we propose a framework to perform word segmentation on printed or handwriting documents through various strategies of data preprocessing and density-based clustering.

2 Data Preprocessing

To better prepare the document images for word segmentation, we perform multiple steps to preprocess the images, including converting to black/white images, adjusting the tilted images, and removing the background noises.

2.1 Converting to B/W images

In handwritten documents, information lies in the words themselves, instead of the sizes of the words or the colors and darkness of the pixels. Therefore, our first step is to change the images into black and white ones, or other kinds of two-color pictures.

A simple way to do this is thresholding. Choose a “darkness” threshold, and if the pixel is dark enough, it will be considered as a valid pixel, which can be replaced by a black pixel. If it is bright enough, then it is regarded as a background pixel, which can be replaced by a white pixel.

Since the original pictures have simple colors, as long as we choose a reasonable threshold, the result will be good enough. We performed a 2-means clustering on the brightness of all the pixels in a given image. The cutoff point of the two result clusters is used as a threshold. The result of this method is acceptable, as shown in Figure 2, but it also relies on the quality of the raw documents. In this image, the brightness of the word pixels is quite different from the background pixels.

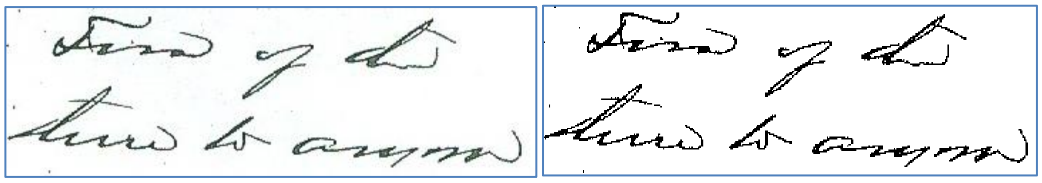

Original Picture Black and white image

Figure 2. Converting to B/W image

2.2 Angle Adjustment

Because the original images are often tilted, we need to find the best angle to rotate the image. To do this, line information can be helpful. If we count the number of black pixels in each line of the image, the lines with words will have very different numbers from the lines in the gap. And when the image is rotated in the best angle, this difference will be the biggest.

Again, for this step, we performed 2-means clustering of these numbers of pixels, and used the distance between the two resulting centers as the difference between lines and gaps. We rotated the image to find the best angle, in which this distance is the largest.

The result of this process is quite satisfying. This works fine when the original images are not tilted much (by just a couple of degrees). Figure 3 shows the performance of this process. The left one is the result from the previous step, and the right one is the result after angle adjustment.

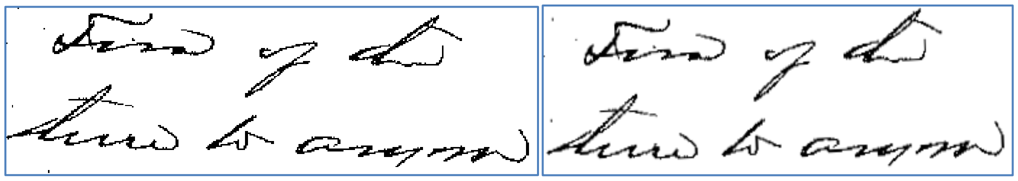

Black and white image Image after angle adjustment

Figure 3. Result of angle adjustment

2.3 Noise Removing

There can be ink, dust, or dirt on the original documents. They left random black dots in the image. Auto contrast cannot remove them, because they can be as dark as the valid pixels. They must be removed in data preprocessing, because they can affect the results of clustering later.

We didn't distinguish noise from outliers. For example, in handwriting, the dot in letter "i" can be considered an outlier in clustering, but it should not be removed, and it's not a noise. In this step, we simply count the number of black pixels in the neighborhood of each black pixel. Pixels without enough neighbors will be considered as noise and removed.

For this to work, we have to specify the size of the neighborhood and minimal number of points for each image. The result varies with the parameters. If the parameters are good enough, the result is acceptable. Figure 4 shows the performance of this process.

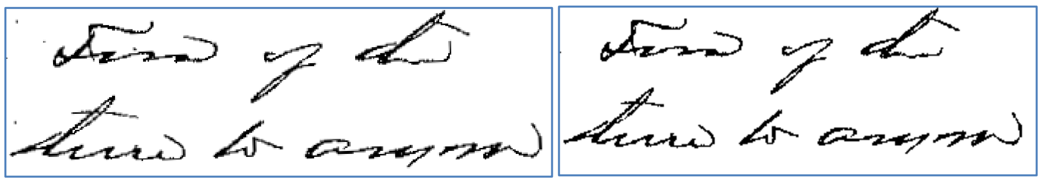


Image after angle adjustment Image after noise removal

Figure 4. Result of noise removal

3 Density-based Clustering for Word Segmentation

Intuitively, we can just perform clustering on black pixels to get words. Since the words are in arbitrary shape, a density-based clustering algorithm such as DBScan [9] will do the job, with specifically chosen parameters. Basically, the clustering algorithm will find "connected" pixels and group them into one cluster. However, a straightforward clustering won't work well for this problem, the reasons for which are listed below.

First, it is hard to choose the parameters. The strokes of the handwriting have different thickness everywhere. It's hard to find the perfect minimal number of points. Secondly, there are too many pixels. If we use an EPS distance of 1, the result will not be good, because pixels in one word don't necessarily connect to each other. If we use any larger EPS distance, the neighborhood will have too many pixels. The algorithm will never stop, or will take too much time to be practical.

Because of the above reasons, we developed our own method. The intuition behind this is that we don't have to care about every single pixel. We can draw dots to denote the area of each word and apply clustering on these dots, which have a much smaller count. The step-by-step process is given below.

3.1 Blurring

Blurring marks the area of a word. With a good parameter, blurring will fill the gaps within one word, and leave the blanks among words out. With blurring, we can easily see the area of each word.

In this step, we used a weighted sum method. For each pixel in the image, we get its neighborhood and the black pixels in it. For every black pixel in this neighborhood, we calculated its distance to the center. We add the reciprocals of all the distances together. This way we get the importance of every pixel in this image. This importance factor will be used to re-draw this image. This step is necessary. Without this step, the parameters in the following steps will be very hard to specify.

The result varies from the size of the neighborhood that we define. The larger this size is, the blurrier the image gets. If we use size of 0, the resulting image is the same as the original image. Figure 5 shows the performance of this process.

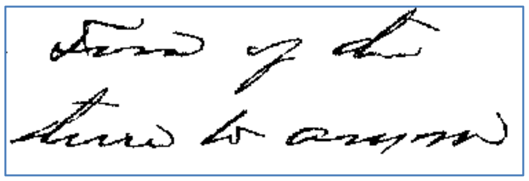


Image after preprocessing

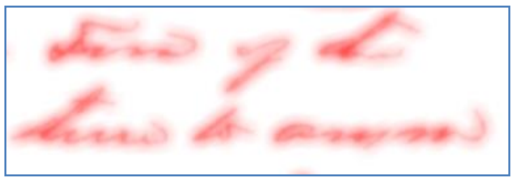
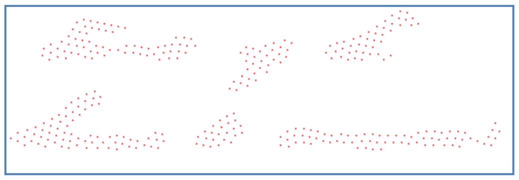

Blurred image. Neighborhood Radius = 10

Figure 5. Result of blurring

3.2 Plotting

We then plotted the blurred image with uniformly distributed dots. We used a threshold at this step to avoid the entire image being dotted, since the blurred words may cover the whole canvas. We can also specify the distance between dots. Figure 6 shows the results.


Plotted image with distance 5

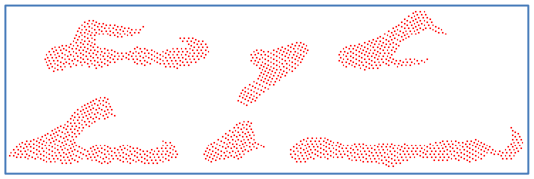

Plotted image with distance 2

Figure 6. Result of plotting

3.3 Clustering

With the dots left on the image, the clustering process is very straightforward. The algorithm will find the clusters with the given parameters. Using this clustering information, we can cluster the pixels in the preprocessed data, because we can simply put each pixel to the cluster which its closest clustered neighbor is in. Figure 7 shows the clustering result.

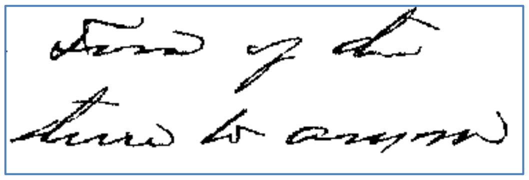


Image after preprocessing

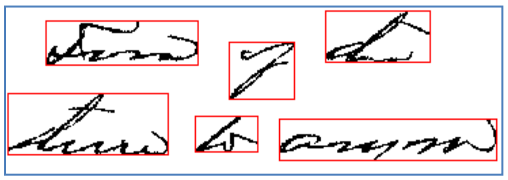
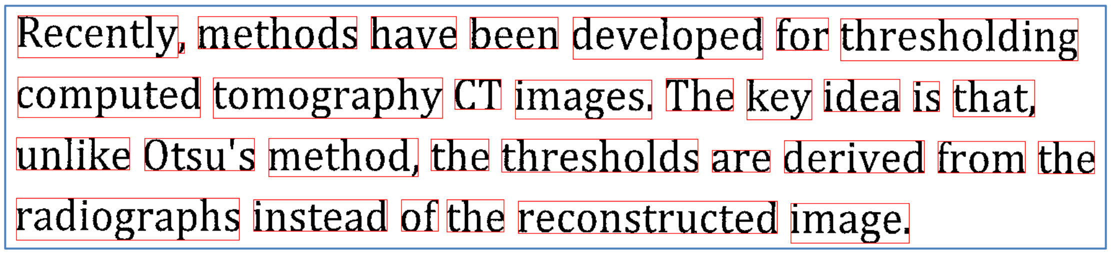

Result

Figure 7. Result of clustering

To this point, we have the result, which is the preprocessed image with cluster information. Words are separated. For printed images, this process works perfectly. An example is shown in Figure 8.



Recently, methods have been developed for thresholding computed tomography CT images. The key idea is that, unlike Otsu's method, the thresholds are derived from the radiographs instead of the reconstructed image.

Figure 8. Clustering result for printed words

3.4 Line Segmentation

In the steps above, line information is not used. If words in different lines are connected physically, it will be difficult to separate them using the process above. This is because if we don't define lines, connected letters or strokes are supposed to be in the same word, which is not always true. An example is given in Figure 9.

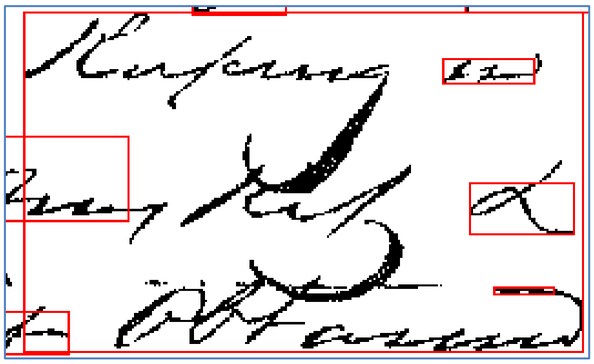


Figure 9. An example with connected letters or strokes

We already have the number of pixels in each line of the image. This number has to be large enough to be an actual line. Each actual line can result in multiple image lines with high pixel counts. So we tried to find where this number becomes larger than a threshold and where this number drops below the threshold. This way we can find the duration of large numbers, which indicates existence of an actual line.

This method worked for the first image we tested. However, when the image gets larger, it's hard to find a good universal threshold. We have to find a way to detect local maximums and keep in mind that they don't necessarily in turn indicate actual lines.

One threshold doesn't work well. So we took a closer look at the numbers (as shown in Figure 10). X-axis is line number from the top of the image to the bottom while Y-axis is number of black pixels in that line. We can see that one high threshold, say 300, will work for most part of this image. But it will also miss the first two lines. A smaller threshold, say 100, will find these two lines, but other peaks, which are not actual lines, will also be marked.

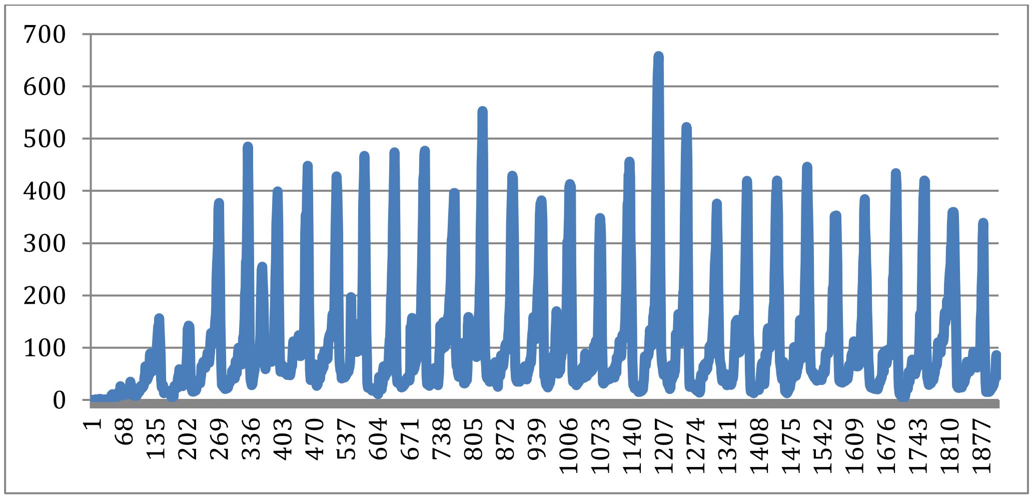


Figure 10. pixel distribution of lines (X-axis is line number from the top of the image to the bottom while Y-axis is number of black pixels in that line)

Currently we use two parameters to find the lines. But this method is not perfect. To determine a line, the number has to go higher than the threshold and drop back below it. And in between, the number has to be higher than a much higher threshold in order to be a peak. The result is shown in Figure 11. Blue lines show positions of lines. We can use their middle lines as gaps (red).

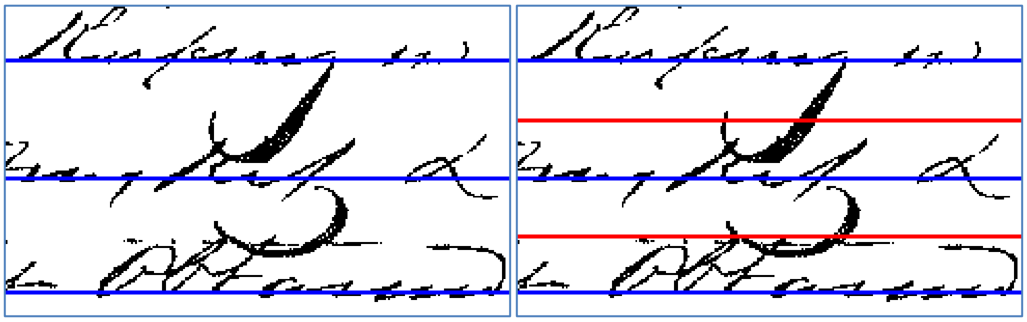


Figure 11. Determining the position of lines

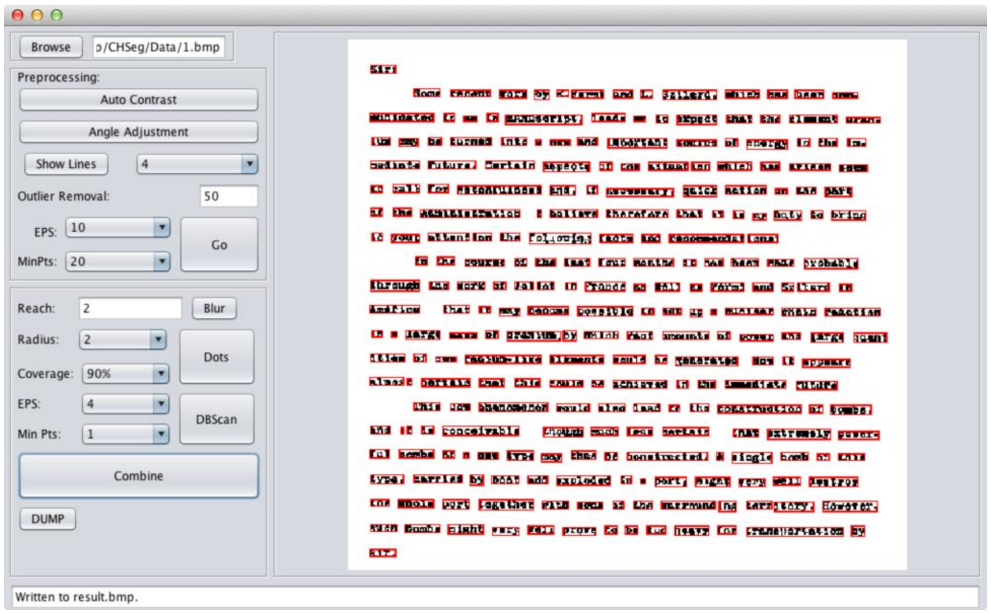
We can see that with this information guiding, the original cluster can be further separated into three clusters, i.e. three words.

4 Implementation

We implemented a system to perform preprocessing and clustering process discussed above. Figure 12 is a screenshot of the interface as well as the output.

For the preprocessing functionalities, *Auto Contrast* can make the picture black and white using 2-means algorithm. *Angle Adjustment* will find the best angle according to details about number of black pixels in each line. *Show Lines* will show lines of words according to the parameters. The lower threshold is specified in the Text Box. The higher threshold is the lower threshold multiplied by the number in the Drop Down List. Pressing this button will also generate a BMP file named “lined.bmp”. *Outlier Removal* will delete the pixels that have less than MinPts points in its neighborhood with radius of EPS. The image in the memory will be updated accordingly.

For the clustering functionalities, *Blur* button blurs the image according to the size of the neighborhood. *Dots* button plots the blurred image according to the distance between dots and a coverage threshold. Blurring has to be done before this step. *DBScan* clusters the remaining dots. Plotting must precede this step. In this step, the image won’t be updated. A user has to click “*Combine*” to get the result and a BMP file named “result.bmp” will be saved on disk. If a user wants to do the clustering again, he/she can start from Blur, without the need to reload the picture or preprocess it again. After any step above, clicking the *DUMP* button will get a copy of the showing picture. The saved file will be named “dump.bmp”.



o/CHSeg/Data/1.bmp
Show Lines: 4
Outlier Removal: 50
EPS: 10
MinPts: 20
Reach: 2
Radius: 2
Coverage: 90%
EPS: 4
Min Pts: 1
Written to result.bmp.

Figure 12. Screenshot of the interface and the output

5 Discussions

With good parameters, our approach can segment printed images perfectly. For handwriting images, our approach can find the words correctly if the words are relatively far from each other.

For preprocessing, to convert to B/W images, we used 2-means clustering on brightness of all the pixels to get a threshold. This works for the documents at hand. But if the images were in different colors, this may not work well, because it will only divide the pixels into two clusters. For example, if an image contains words in both black ink and red ink, red ink could get discarded, because it is possible that the red color is not dark enough according to the 2-means result. Similarly, angle adjustment will also have different results on different documents, since it’s also based on the 2-means clustering algorithm. This method works well on average. In our testing, it rotates only one testing image by 0.5 degrees when it shouldn’t have. Outlier removal yields satisfying results. However, it’s very sensitive to parameters. We have to find the appropriate EPS and MinPts values to make the result optimal. Also, it uses two universal parameters, instead of local ones. As we can see from the sample, one image may have different parts that require different parameters.

For clustering, Blurring and Plotting make clustering more efficient. However, blurring may cause the plotting to put dots where there weren't any black pixels originally. Blurring spreads the ink. The original intent is to fill the gaps within each word. But blurring not only fills the gaps within a word, but also spreads ink outside the word. This leads to inter-word connection. It actually makes the clustering less reliable. To choose a good blurring parameter, we have to find the radius large enough to fill the blanks within a word, but not large enough to connect two words. Plotting, on the other hand, can break a word. Usually a smaller distance works better.

Density-based clustering approach has some limitations [10], as discussed below.

- Sensitive to parameters

Even for printed images, parameters have to be chosen carefully to get a perfect segmentation. Each image is different. However, with a batch of similar documents, the parameters do not have to change much.

- Universal parameters

When choosing a good pair of parameters, we cannot separate certain words without breaking other words. And we cannot connect letters/stokes in one word without connecting words that aren't supposed to. This is because density-based clustering uses universal parameters, while the image has different characteristics on different parts. One possible solution is to use different local parameters. The simplest variation will be to separate vertical and horizontal parameters, e.g. to use a vertical EPS distance and a horizontal EPS distance. This way we can separate words on different lines without breaking a horizontally long word.

- Connectedness

Density-based clustering only considers pixels that are connected to each other as being in the same cluster. This is not exactly the case for cursive handwriting segmentation. Words can be connected to each other, or separated within themselves, which goes against the nature of clustering. Number of black pixels in each line gives quite sufficient information on the line segmentation. However, the method we used for line segmentation is very sensitive to parameters.

Here are some possible solutions. To detect the lines, an algorithm should be able to detect peaks. Local maximum is not enough. The peak point has to be much higher than the neighboring point. One actual line may have multiple peaks, and we should use the dominating one. Also, universal thresholds may not work well. Note that there may be different numbers of words in different lines. For example, in the beginning of a letter, the name of the recipient takes one line, but there is only one word in it. The number of pixels is relatively small, but is still much higher than the neighboring gaps. Wavelet analysis could be a reliable method for this task. Our current system didn't combine line information with word clustering, but it is considered for our future work. Here are some ideas about applications of line information in word clustering. One possible method is to apply line information before the clustering. We can put in white pixels in the line gaps, which will separate words in different lines. Or we can simply perform clustering for each line. Putting white pixels in gaps may erase valid pixels. This could potentially delete useful information, especially if the line segmentation is not perfect. Performing clustering for each line may seem plausible. And it gives us a chance to use different parameters for each line. However, we should keep in mind that lines are not strict. In handwriting, it's hard to keep all words in a straight line. There will be overlaps. Each line may contain small parts of words from other lines. They will end up with separate clusters.

We may also want to pay attention to the clusters that cross the lines after clustering. After the word clustering, line information is only useful to those clusters that cross the line. We can use line information to break them. However, there is no way to know how many clusters to break one into. This problem can be as hard as adding line information before clustering.

6 Conclusions

In this paper, we present a framework of using density-based clustering for word segmentation in printed or handwritten documents. The framework also includes multiple steps of preprocessing using 2-means clustering or neighborhood density. Preprocessing converts an image into black/white image, adjusts its angle, and then removes the noise. The preprocessing of the images has provided very good results. After preprocessing, the images will go through multiple steps, including blurring, plotting, and density-based clustering. We also developed a system for the framework, including preprocessing and clustering functionalities. Word segmentation using density-based clustering has achieved very good results on printed images. The results on handwritten documents vary depending on whether the words are relatively far from each other. For our future work, we plan to use line segmentation to further improve the result.

References

[1] H. Lee and B. Verma, "Binary segmentation algorithm for English cursive handwriting recognition", Patten Recognition, Vol. 45, Issue 4, 2012, pp. 1306-1317.

[2] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A Novel Connectionist System for Unconstrained Handwriting Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 31, Issue, 5, 2009.

[3] A. Senior and A. Robinson, "An off-line cursive handwriting recognition system", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, Issue 3, 1998, pp. 309-321.

[4] R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, Issue 1, 2000, pp. 63-84.

[5] R. Vaidya, D. Trivedi, S. Satra, and M. Pimpale, "Handwritten character recogniztion using deep-learning", International Conference on Inventive Communication and Computational Technologies, 2018.

[6] N. Shun, G. O. Hiroshi, T. Ogata, and J. Tani, "Handwriting Prediction Based Character Recognition using Recurrent Neural Network", IEEE International Conference on Systems Man and Cybernetics, 2011.

[7] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks", Conference on Neural Information Processing Systems, 2009.

[8] M. Liwicki, A. Graves, and H. Bunke, "Neural Networks for Handwriting Recognition", Studies in Computer Intelligence, 2012, pp. 5-24.

[9] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise", Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD), 1996, pp. 226-231.

[10] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd edition, Morgan Kaufmann, 2011.