EE608 - Digital Image Processing

Decomposing Documents into HTML using Text Classification and Layout Contouring

FINAL REPORT

submitted by

Aditya Sarkar
IIT Mandi
b19003@students.iitmandi.ac.in

Gajraj Singh Chauhan
IIT Mandi
b19130@students.iitmandi.ac.in

under the supervision of

Dr. Renu M Rameshan

INDIAN INSTITUTE OF TECHNOLOGY, MANDI

November 2021

Abstract

In this project, we present a pipeline for generating HTML from a scanned image of a document. The pipeline firsts preprocesses the image by removing salt-and-pepper noise, deskewing the image and applying threshold for easy handling. Then it tackles two problems that are layout detection and text classification to form HTML. Our code can be found at this **Github repo** : https://github.com/aditya-sarkar441/DIP_final_project.git

Contents

Chapter 1

Introduction

Working with Digital formats is beneficial as one can not only compress huge information to a small file size, but can also share it over to other users. However users tend to work more with images, especially in cases when you have to send a scanned image of a document to someone. Taking printouts of scanned images are usually dull and hard to read through. Detecting text in images has been a tough task in image processing. With the development of Computer Vision, Deep learning and Natural Language Processing, one can get better results as compared to classic image processing algorithms. However this increases the computation significantly that one needs to use GPU to get the output in a reasonable amount of time. In this project, we have implemented the detection of text from scanned images of a document and recognized different characters like bold, underline, comma etc. and performed layout analysis in which we made contours across the paragraphs and columns.

1.1 Previous Work

This section deals with the previous work done in the field of text detection and layout analysis. Layout analysis actually consists of two parts that are - geometric layout analysis and logical layout analysis. Geometric analysis takes care of contouring paragraphs and columns, while logical analysis works on captioning. In this report, we have worked only on geometric layout analysis, and will be using the term ”layout analysis” for the same.

1.1.1 Layout Contouring

Most of the recent works can be classified broadly into bottom-up and top-down approach.
In **bottom-up approach**, we divide the image into black and white regions, and then agglomerate them to form words, then text lines and finally text blocks. These methods usually have high time complexity and requires a user defined termination condition at which the iterative algorithm will stop.
In **top-down approach**, we divide the document into columns and paragraphs using the white spaces and geometric information. These are usually fast and does not require any user-defined termination. However, it requires a threshold using which it will divide the document's image.
The most famous bottom-up algorithm which uses the classic image processing algorithms was developed by O. Gorman [1] in 1990. It gets small components in form of letters and uses the between letters width to form words, between words width to form lines, and so on to segment the layout into paragraphs and other lines.

1.1.2 Text Classification

In text classification, one normally uses a classifier that can predict the letter. One simple approach is correlations method in which we will have to compute the Euclidean distance between the test image and all the training images (they have labels). The label of the training image with the lowest distance is assigned to the test image. In this case, the time complexity is *O(MN)* where M is the number of training images and N is the number of testing images. This method is effective in our case as our training dataset is small.

Another method is template matching [2]. In template matching, user gives a template image and tries to find its occurences in the main image. This method wasn't working good in our case. More about template matching is described in the Experiments chapter.

Better classification can be achieved by using Deep Learning frameworks. Widely used method is Tesseract [3] [4] which relies on Convolutional Neural Networks and Long Short

Term Memory Networks. Tesseract is the best model as of now, and is widely used for text classification in images. In our project, we have used Tesseract for text classification.

1.2 Why is this problem hard ?

Text detection and layout contouring is a hard problem in image processing because -

- Time complexity of a straight forward method i.e. using hierarchical agglomeration on the extracted words is very high. It is $O(n^3 logn)$ where n is number of words identified in the document. Our main target in this project was to bring down the time complexity of our proposed algorithm. Our algorithm for layout analysis has a time complexity of $O(nlogn)$.

- Different documents have different levels of space in between paragraphs and lines. Some have more white space between the paragraphs while some have less space. Developing an ideal algorithm which can handle both the cases is tough.

- Variation of font sizes and a variety of layouts available has also made the problem difficult.

- Additionally the layout can contain any number of non text elements like images, and to add to the complexity the size and shape of layout can also be uncertain.

Chapter 2

Preprocessing

2.1 Dataset Collection

For **layout analysis**, we have manually collected images from Google Images. We want to evaluate our model on various types of images, hence we focused on variety rather than quantity. We selected scanned resume images, one column document image, two column document image and business letter image. Results of our algorithm on these documents can be found on Github. Link given in abstract.

For **text classification**, we have manually prepared the data by cropping the letter images from a document. These data includes 26 cropped images of small alphabets, capital alphabets, numbers and punctuation such as comma, full stop, forward and backward slash, hyphen etc.

2.2 Noise Removal using Non-local means

For removing noise, we have used non-local means. Non-local means method was developed by Buades *et. al.* [5]. In this method, we, instead of taking mean of neighbourhood pixels, take the weighted mean of all the pixels in the image, where the weights are dependent on the similarity of these pixels to the target one. We have used this as it preserves the edges as compared to Averaging or Gaussian filter. Also shown in figure 2.1.

2.3 Otsu Binarization

Binarization is an important step in our pipeline, as it mitigates the effect of lighting conditions in which the document was scanned. Lighting conditions degrades the quality of our pipeline. Otsu Binarization [6] is a classic method of threholding images. Binary images are beneficial in our problem as it tackles any noise present, which helps in improving the efficiency of our algorithm.

Fig. 2.1: Otsu thresholded document

2.4 Skew Removal using Hough Transform

Most of documents available are usually skewed. It means that the scanned image of the document is rotated by some angle. Tesseract has an inherent problem that it is not rotation invariant. It has very bad performance on skewed images. Hence deskewing is an important part of our pipeline.

In the deskewing process, we have assumed that the document is skewed by an angle less than 30 degree. We have deskewed the image by using Hough transform [7] [8], specifically Hough lines. Hough Transform is applied on the whole image. We get the angles of the lines. We take the median of these angles, say Θ and rotate the document with Θ-90. This angle can be easily calculated from geometry. Below is the result image.

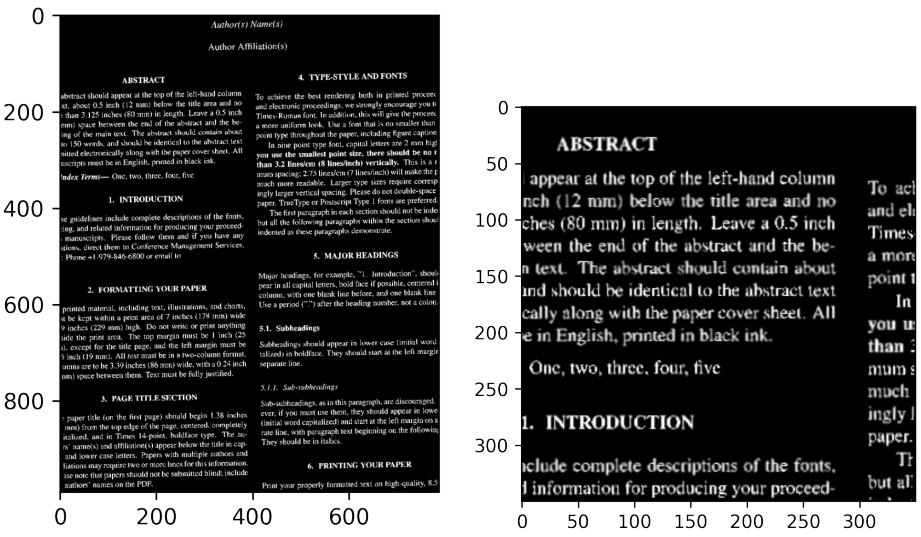

Fig. 2.2: Skewed document and deskewed tile

Chapter 3

Layout Contouring

3.1 Algorithm

We started by first segmenting the letters in the document using Connected Components Labelling. CCL considers the image of a scanned document as a non-directed graph G(V,E) where vertices (V) are the letters and edges (E) are the distance between the letters. Next we have used DBSCAN to cluster the letters to form text blocks.

3.1.1 Connected Components Labelling

Connected Components Labelling [9] is a graph based algorithm, which find the different components (i.e. disjoint sets) in the graph and then labels them using numbers. `connectedComponentsWithStats` method use for implementation was taken from the OPENCV library [10]. The idea behind using this method was to segment the letters as different components. Additionally some of the hyper parameters used by this algorithm is

1. connectivity - how many neighbors adjacent to a pixel must be looked at in the algorithm, the value can either be 4 to 8 as in 4 directions or 8 directions in a 2D coordinate system respectively.

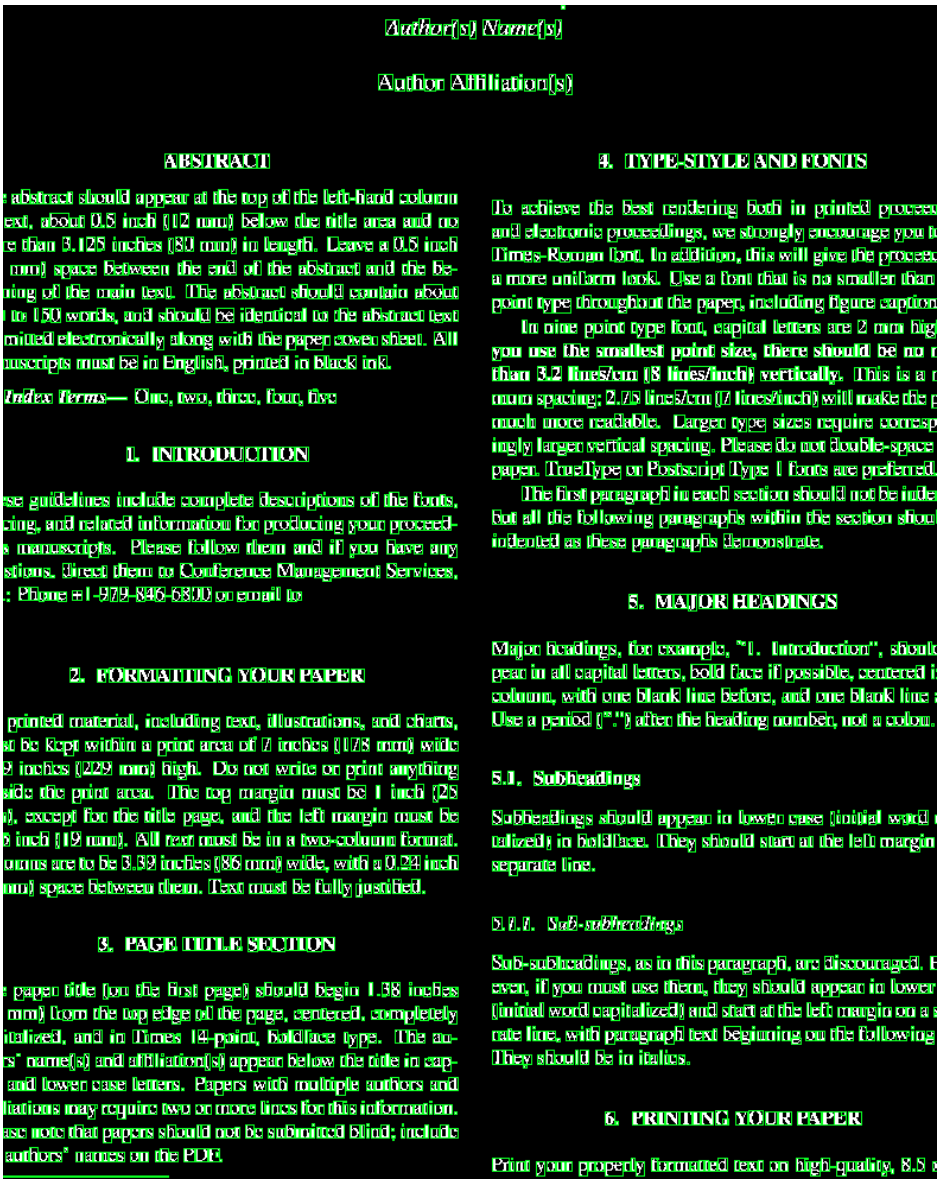

Fig. 3.1: Connected Components shown within green bounding box.

3.1.2 DBSCAN

Finally, the algorithm used for detecting layout was DBSCAN [11]. In order for segmenting the layout as paragraphs, title the centroids for the components found is used. DBSCAN requires us to specify two parameters for clustering. It is a density based clustering method, and expects to two parameters to classify the point are part of a cluster or an outlier. In in this project, the outlier points have not been considered as part of layout.

1. eps - maximum distance between points to be considered as part of cluster.

2. min_sample - minimum number of points in the neighborhood to be consider it as a core point (part of cluster).

Fig. 3.2: DBSCAN on connected components.

3.1.3 Contours Formation

For forming the contours, we take the coordinates of the cluster clustered by DBSCAN. We find the maximum and the minimum x and y coordinate. Using this, we can form the contour

accross the cluster.

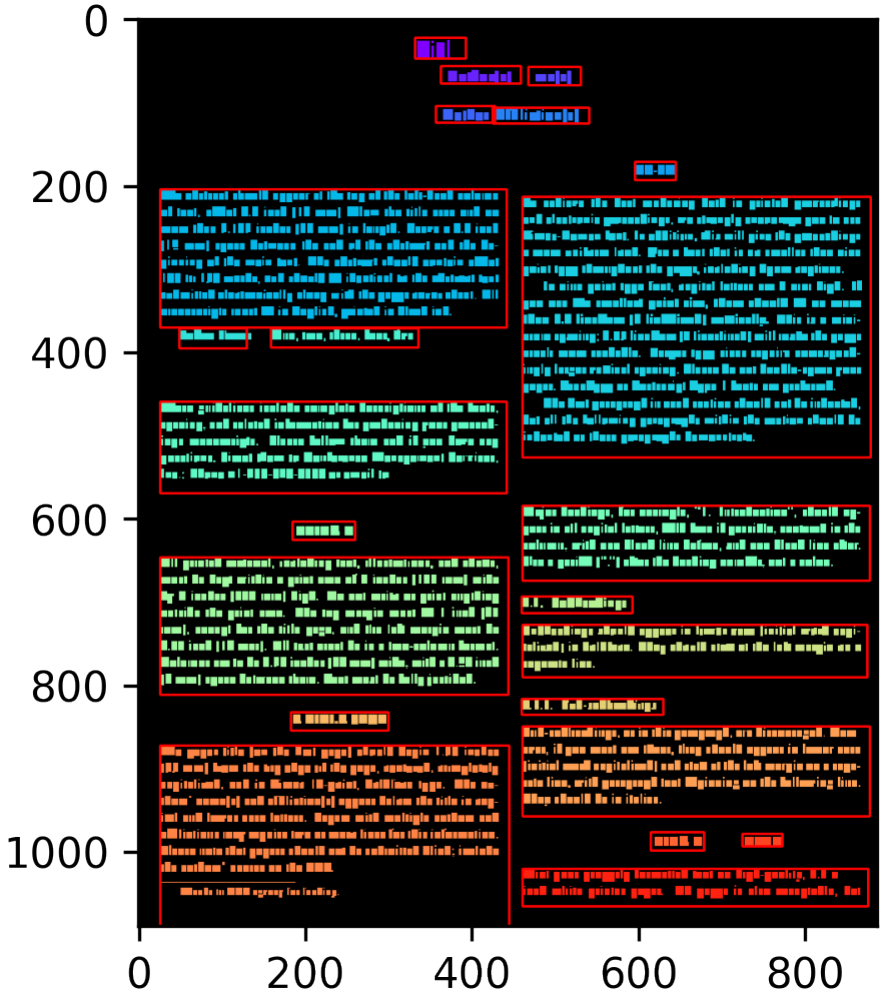

Fig. 3.3: Contour formation

3.2 Problems with the algorithm

This algorithm, just like existing algorithm, has some disadvantages. These can be concluded as follows :

1. CCL is not able to segment the letters if it is too much compact. Figures 3.2 show some examples.

2. We are having a bad performance of CCL when there are lot of letters.

3. We might additionally face problems by segmenting unintended components.

4. DB-Scan is dependent on the components and thus is error prone.

3.3 Quality of our approach

In our literature search, we found that most of the existing algorithms start by using some graph based approach to extract the letters from the scanned image. Then it targets the white spaces between them and forms the contours. Some old algorithms have used agglomeration which is having high time complexity and gives poor results.

In our approach, we have considered the letters as vector points in 2D space. These vector points can be easily clustered by some clustering algorithm. We were interested in a clustering algorithm which gives good performance at low time complexity. And DB-Scan satisfies both our conditions. DB-Scan has a time complexity of *O(nlgn)*. Before going for DB-Scan, we have also applied K-Means, BIRCH, Agglomeration, Affinity propagation and Spectral Clustering. We had to reject these algorithms, mainly because of two reasons - first, the poor clustering performance and second, they need a parameter of number of clusters. For a user who is having multiple scanned documents, it is difficult for them to specify this parameter.

DB-Scan also has a parameter called epsilon which clusters based on the distance between the points. This seems to be troublesome, but actually it is easy to predict, even for a naive user, because one only has to see the compactness of the document. If the compactness of the document is more, then epsilon will be less, otherwise it will be more.

Chapter 4

Text classification and HTML synthesis

Text classification is an important as well a difficult part in our pipeline. We tried implementing various algorithms such LDA, template matching, morphological operators. The accuracy of these algorithms was very poor, and hence we had to use Tesseract for making the prediction. However, these approaches are described in the Experiments section, along with their results. For text classification, we have assumed that the text is horizontal.

4.1 Algorithm

Tesseract is a widely used open-sourced OCR engine developed by HP labs at Bristol [3] in 1995. It has evolved over the years and currently uses Long Short Term Memory Networks (LSTM) and Convolutional Neural Networks (CNN) for making predictions. Its popularity is because of its ability to read not only computer-typed text, but also handwritten text. It accepts an image and *returns only the text present* in it. We extracted the contoured images from the previous step and passed it though the tool to get the required text. We will synthesize the HTML using `Dominate module` [12]. The parameter it takes is the position and the text that is to be put in the document. The position of the text is given by the centre coordinate, height and width of the contour formed and text is given by Tesseract OCR. The HTML can be downloaded and converted to other formats easily to be used for further use.

4.2 Quality of our approach

We evaluated the Tesseract tool by using different images. We found that it is having some inherent problems which can be concluded as follows :

1. It is highly correlated with the rotation of the document.
2. Noise affects the performance of Tesseract.
3. It assumes that the image is binary. Non-binary images affect the performance of the tool.
4. It only outputs the text. The position of the text is not provided, therefore making it bad at handling text which is present in different columns. Also for synthesizing the HTML, one needs the position of the text.
5. It cannot handle Page numbers. It considers them as a part of text.
6. It neglects the footnotes present in the document.
7. Tesseract was bad at predicting letters.

We rectified these problems with the tool using pre-processing steps and changed our **global approach to local approach** for extracting text. In terms of pre-processing, Otsu Binarization can handle case 2 and 3. Deskewing can handle case 1. To handle points 4, 5, 6 and 7, we have adopted a local text extraction approach. What we will do is, instead of giving the whole document to the Tesseract engine, we will give the a cropped version of the image. The cropped version will be decided by the contours we made in Chapter 3. The position of the contours will be the position of the text in the HTML as discussed above. Contouring can also handle the page numbers and the footnotes in the final HTML.

Thus our text classification algorithm is dependent on the above layout contouring process. Any error in the contouring method will get amplified in this process.

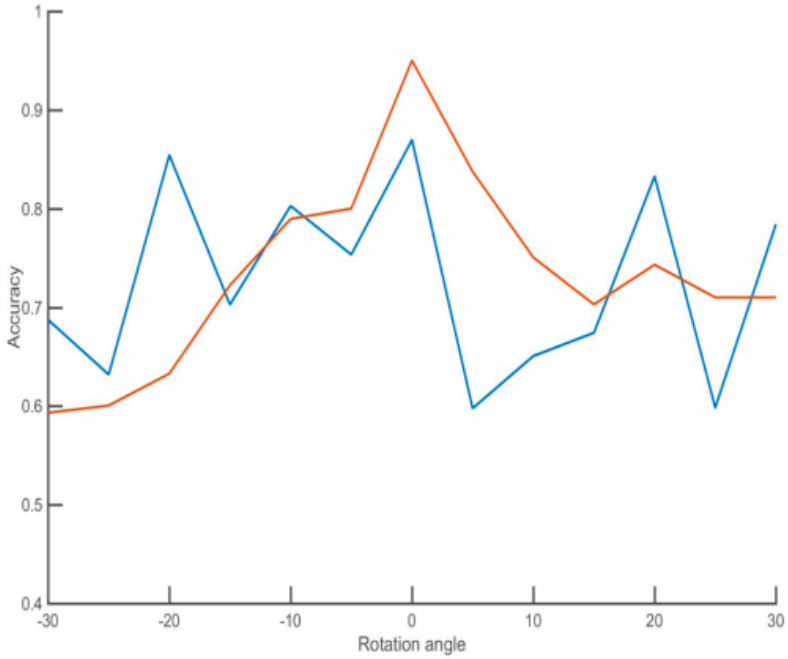

Fig. 4.1: Technical report by Yang and Chen [13] (Stanford University). The plot shows the accuracy of Tesseract (red) and Matlab OCR (blue) for various rotation angles. We can find that there is poor accuracy by Tesseract (red).

Chapter 5

Discussion and Experiments

We have performed various experiments before finalising our pipeline. All the experiments are described in this section.

5.1 Approach 1: Contouring using Bar Plots

White spaces play an important role in separating the texts. Our eyes perceive that two texts or letters are separated if there is some space between it. Initially we planned to work on exploiting the spaces to get our contours formed. After applying CCL, we decided to calculate the spaces between the *consecutive letters*. However, CCL does not give consecutive letters, albeit it gives them in a random fashion. So our important target was to arrange the centroids of the letters. Sorting was also required for predicting the text, as earlier, we were planning to use LDA for letter prediction and then joining them in the sequence to form the word. We sorted them in such a way that y coordinate is in ascending order and for each y coordinate, x coordinate is also in ascending order. We used Merge Sort to sort them in O(nlgn) time. This is also having a problem. Consider the word `gill` and its letters `l` and `g`. They lie in the same line, but not in terms of pixel positions. The centroid of `g` was actually slightly below the centroid of `l`. Because of this, `l` is coming before `g`, even though in english word, it is after `g`.

So we decided to vertically project on Hough lines. Making Hough lines has a time com-

plexity of O($n^3 lgn$), which is high. So we decided to use horizontal lines which are having a vertical distance of 5 pixels. And then project the points on these lines. Thus our complexity will be O(n) as we will be iterating over the points only once. Once they are projected, they were easy to sort using Merge Sort. Next we iterated over the letters, calculated the consecutive distance of the letters and plotted a bar plot. X axis was iteration number and Y axis was space distance. Then threshold was decided based on the text spaces. Text spaces were calculated in both horizontal and vertical directions. Thus we will have two thresholds. Using those thresholds, we can predict `where the column separations are` and `where there is vertical space between the text lines.`

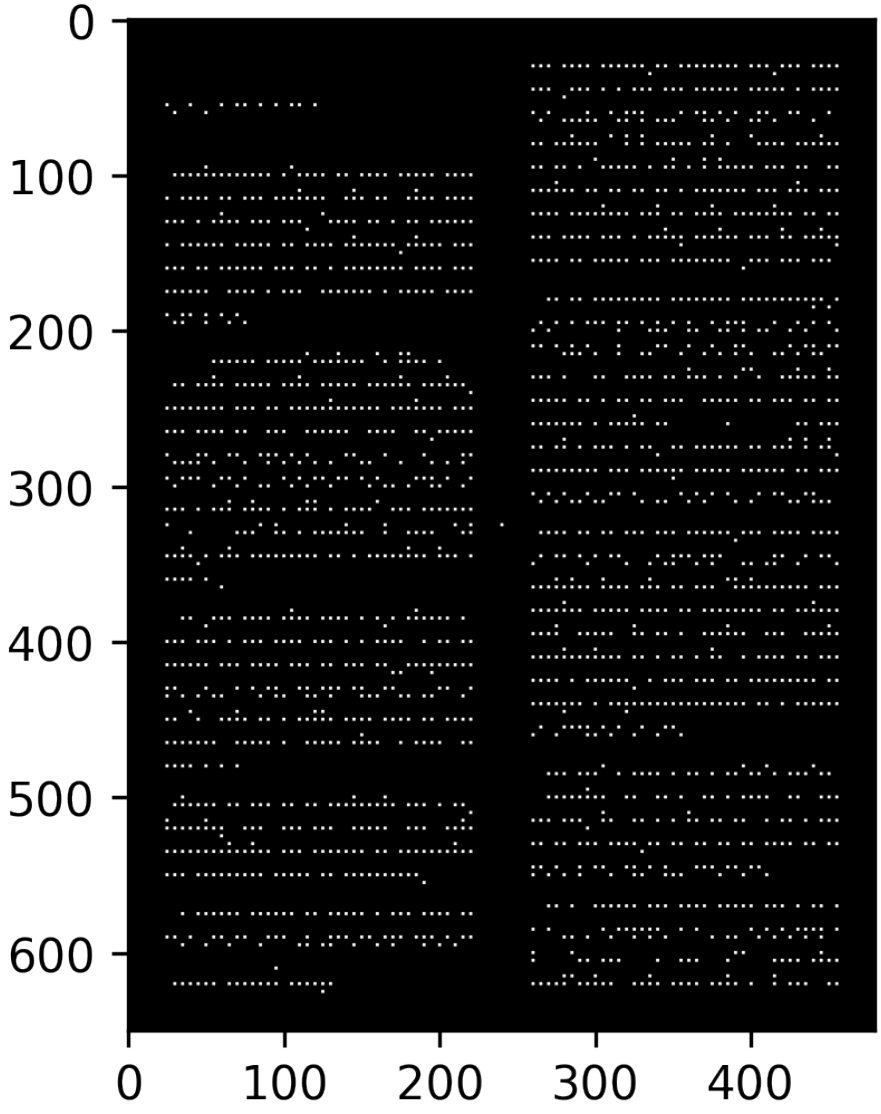

Fig. 5.1: Centroids after projection. You can see that they are placed at a distance of 5 pixels.

Column Detection : The horizontal spaces were calculated. Suppose we select threshold as T. Then all the spaces have values more than T are our consideration. We can find the iteration number using the bar plot. Suppose we got that at iteration I, we have that space value. Then the space value is the result of `x[I+1] - x[I]`. Thus we can say that there is a column between x[I] and x[I+1]. We will store these values for forming contours.

Paragraph Detection : The vertical spaces were calculated. If space values are more than some threshold T, then they are in our consideration. Let the iteration be I, then there is space between y[I+1] and y[I]. We store these values for forming the contours later.

Before forming the contours, we checked if the columns and paragraphs are properly separated. We found that it didn't work well. Hence we decided to go for clustering methods.

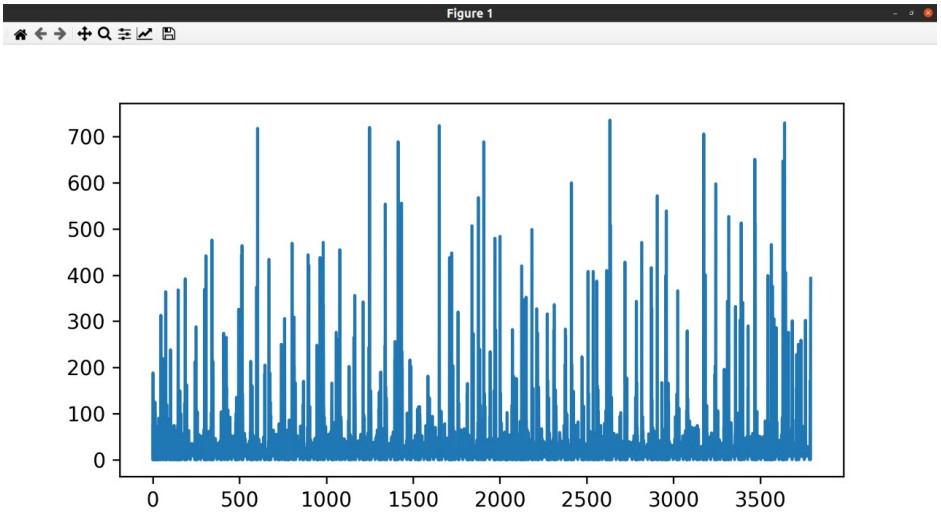

Fig. 5.2: Bar plot for vertical spaces. X is iteration number and Y axis is space in pixels

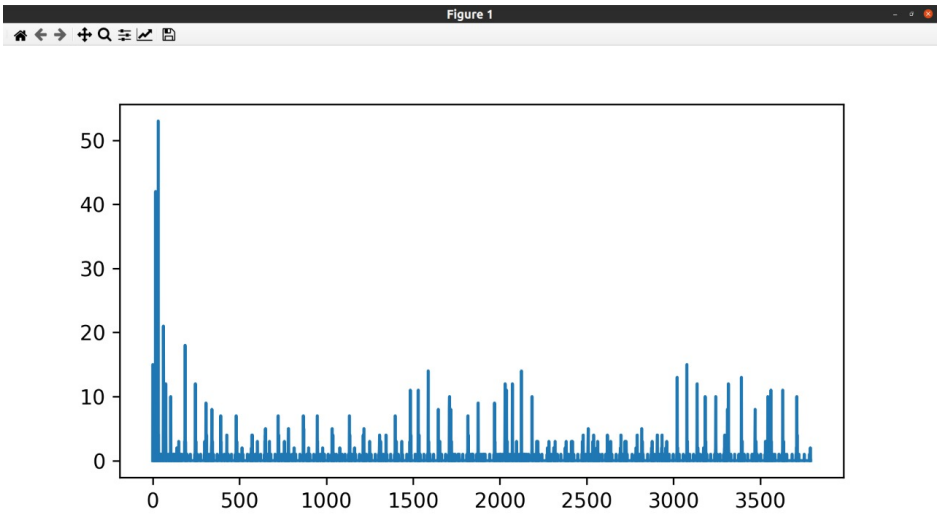

Fig. 5.3: Bar plot for horizontal spaces. X is iteration number and Y axis is space in pixels

5.2 Approach 2: Otsu Binarization and Clustering based approach

Otsu Binarization was not only important for running Tesseract, but also for CCL. The figure below shows the output of CCL without binarization. We found that the performance of CCL was bad without it and hence we used it for our further work.

As mentioned in earlier sections, our algorithm's uniqueness lies in the clustering method. There are many methods for clustering and selecting a particular method was tough. We carried out multiple experiments before concluding to a method. While performing the experiments, we had two needs - one is performance and other is time. We have to make a trade-off between the two. The table below shows the performance and the time. Please note that we rejected Spectral clustering as it was taking more than 120 secs.

We found that DB-Scan was giving the best performance. Hence we finalised it.

Algorithm	Time (in secs)
DBSCAN	11
Birch	16
Agglomeration	16
Affinity propagation	70
Spectral Clustering	>120
KMeans	13

Fig. 5.4: Time taken by various algorithms

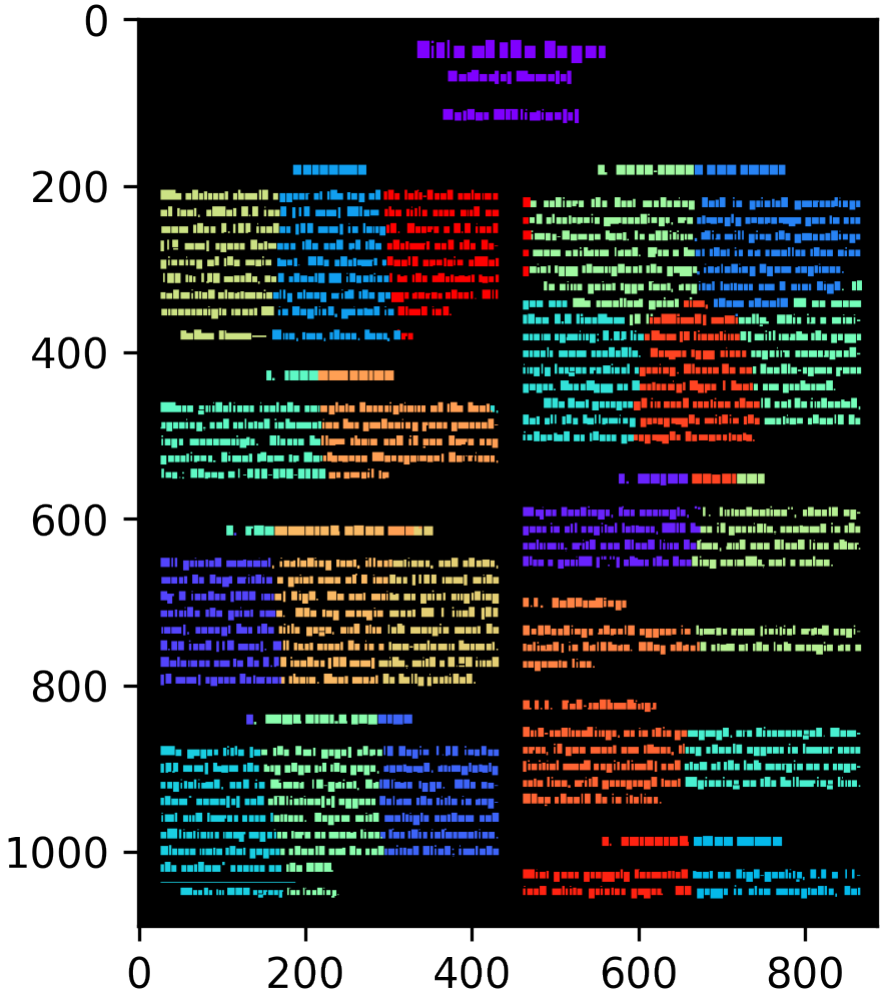

Fig. 5.5: Clustering by Kmeans

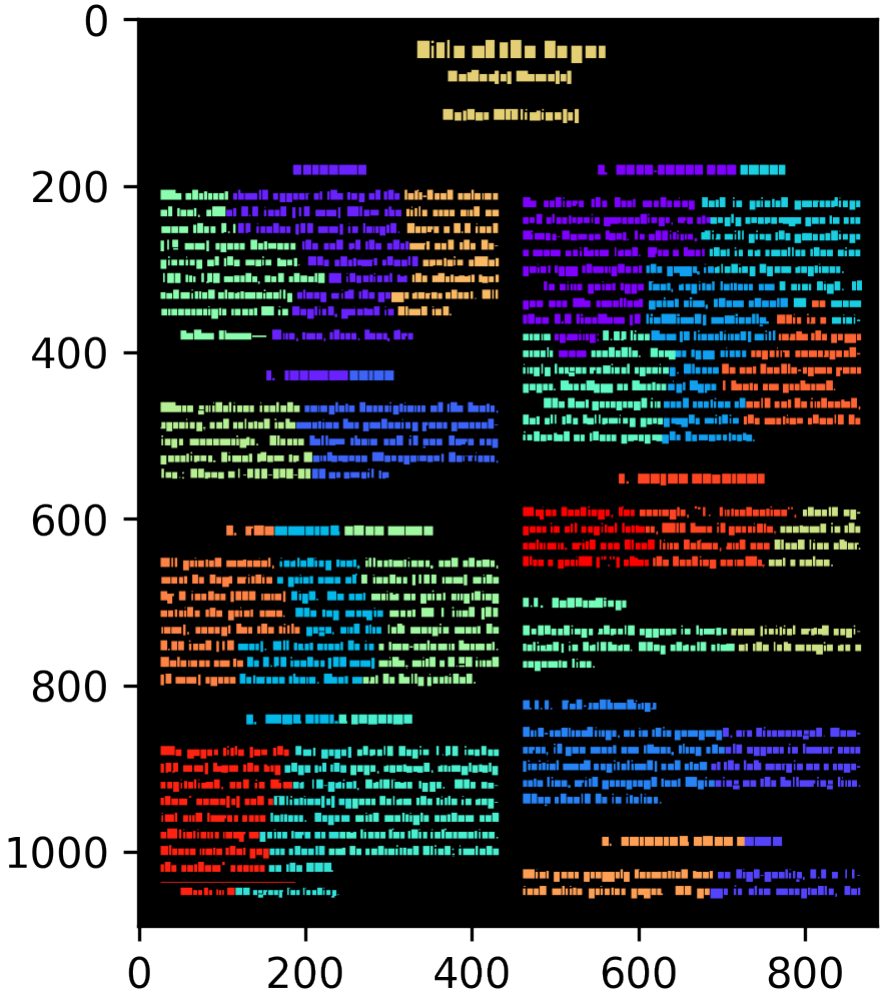

Fig. 5.6: Clustering by Birch

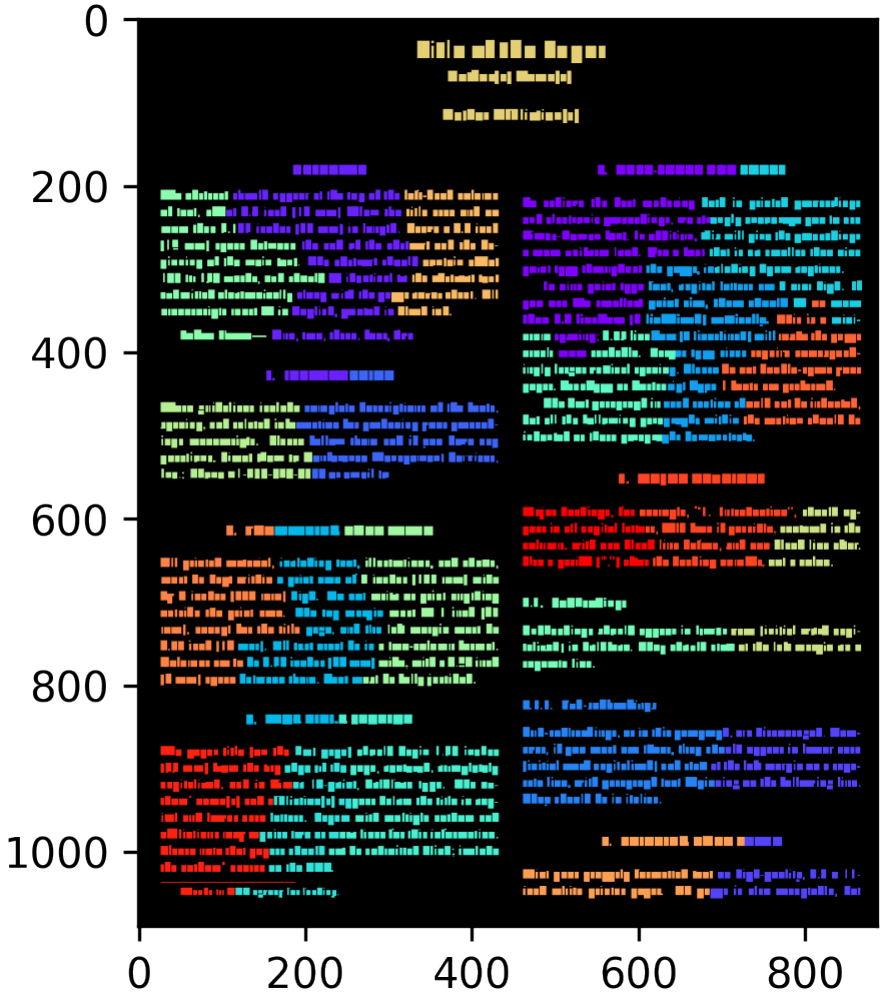

Fig. 5.7: Clustering by Agglomeration

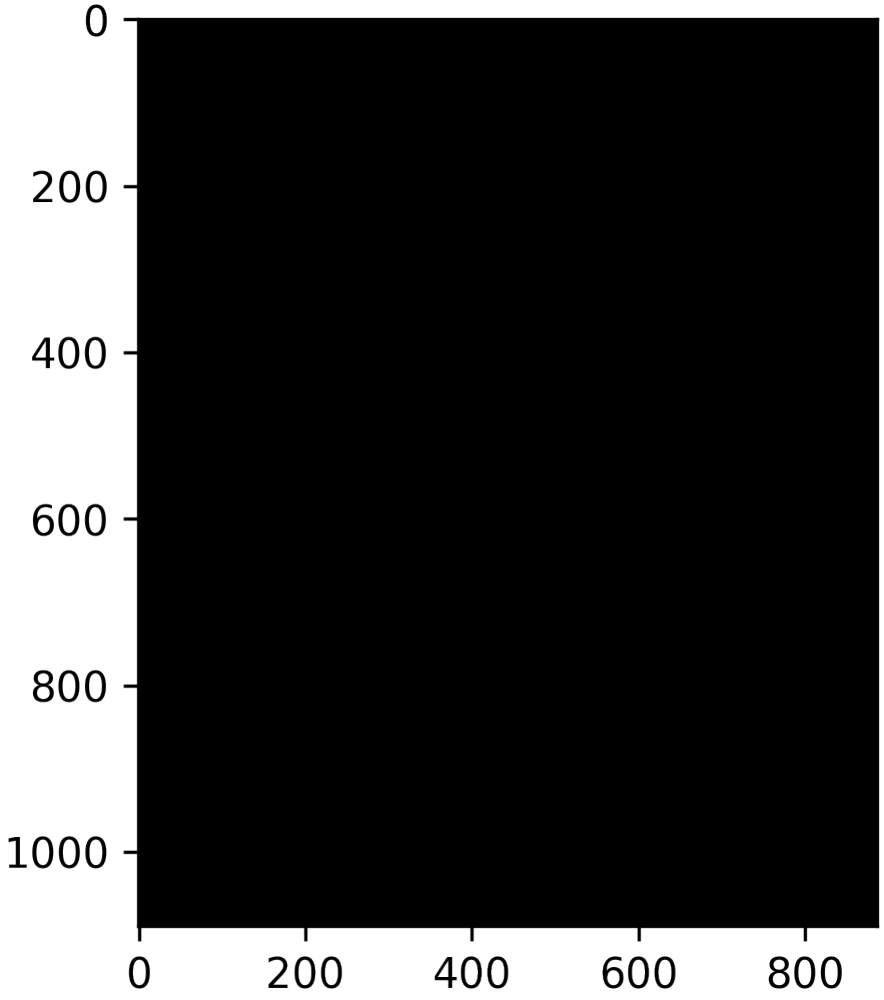

Fig. 5.8: Clustering by Affinity Propagation

5.3 Approach 3: Layout detection using window approach

After Otsu thresholding, we divided the image into windows of size 10x10. We calculated the mean of each window and selected those which were having high mean pixel values. High mean pixel values indicate that there is some text. For each of the window, we applied CCL and drew hough lines between a minimum and maximum value. This ensures that the Hough lines we get are all nearly horizontal and vertical, and not at any other position. On each horizontal Hough line, we found those bounding boxes which intersected with the Hough line. These actually means that they lie on the same line. Similarly, we did this for vertical Hough lines; this gives us that they lie on the same vertical line. Using this we can easily predict the position of column gap. However, it was having a very high time complexity with poor results. Hence we discarded it for layout detection. However, we have used a similar concept for skew detection.

5.4 Approach 4: Text classification algorithms

We implemented various text classification algorithms such as PCA-LDA, template matching and morphological operators.

Template matching - We started with the template matching. For template matching, we have used cropped letters' images as a template. And the main image was a binarized image. However, we got incorrect position of letters. Also all the letter's occurences weren't detected.

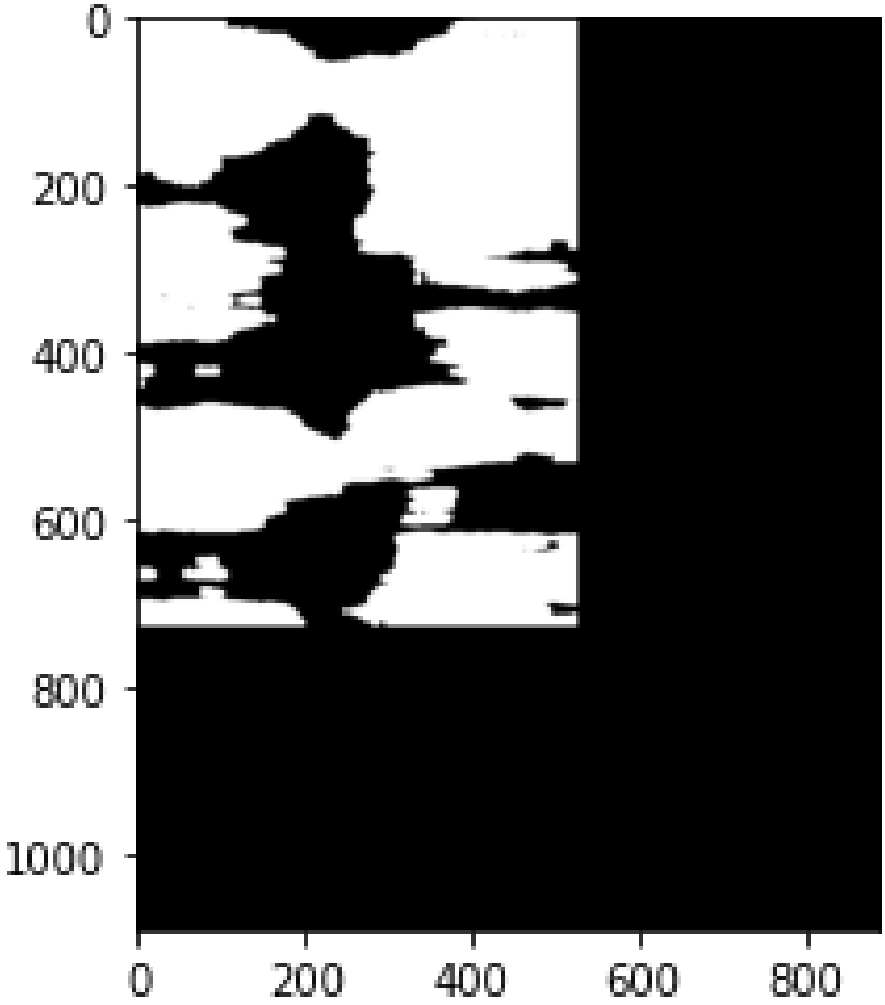

Fig. 5.9: TM ouput image. You can see it's a very bad prediction

LDA method - We manually prepared a dataset of typed letters (both in small and capital), punctuation and numbers. We segmented the letters from the image and applied LDA on them. Out of 3357 letters, we got only 5 correct letters, thereby making its accuracy of only 0.14%. The reason of such poor accuracy is that letters can be broadly divided into three classes - first consists of those which are having curved lines eg - `a,e,c,s` etc., second consists of those which are having straight lines eg - `A,H,L,I` etc. and third consists of those letters which have both curved and straight lines eg - `B,J,P` etc. LDA was not able to classify letters which belonged to only one of these classes. For example - c and s were being classified as c only. Letter e which is having the highest frequency in the text was misclassified as c.

Morphological operators - We worked on erosion and dilation. Suppose we want to classify c, then we selected the template as the cropped image of 'c', and iterated over all the letters applying erosion. We took the mean of the cropped 'c' image, and found the mean of other eroded letters. In an ideal case, we thought that we will get a very low value, almost equal to 0, for 'c'. However we ended up having even lower values for letters 'e' and 'r'. Same thing happened with dilation. We expected to get the highest mean value for 'c' but got higher mean values for 's'.

5.5 Approach 5: Probabilistic Hough Lines for contouring

Probabilistic Hough lines are hough lines which does not expand over the whole document. These are a softer version of hough lines. They can be useful for detecting columns. We implemented them on the whole document. However, just like the window approach, its performance didn't give much benefits over high time complexity. Hence we discarded it.

5.6 Discussion

Apart from the experiments mentioned above, we had some ideas which we weren't able to implement because of time constraints and our expectation of getting a poor result. Existing

works on layout analysis are brimmed with graph based approaches. Before adopting the graph approach, we were initially thinking of applying image segmentation using Otsu or Mean shift. This reduces our work space. So the output image will be white portion having black text, on a black background, similar to the image present in Dr Renu Rameshan's November 4^{th} lecture slides. Next we can draw the Hough lines on the white portion and using an appropriate height we can start extracting the letters. However, we were not able to quantify this height, and hence didn't implement it.

The failed experiments mentioned above guided us to the main solution, which we have provided earlier. Contour forming along with the pre-processing steps was important for our algorithm to work in a better way. They enhance the prediction of the Tesseract.

In case of predicting letters, our ideas have failed miserably, which led us to finally use Tesseract, a DL based tool. The major reason for the failure is because of poor quality dataset. Another could be because of the linear decision boundary made by LDA. We may have to work with classifiers which produce non-linear decision boundaries.

Chapter 6

Conclusion and Future Work

6.1 What we have achieved ?

Our proposal in the Google forms shared by Dr Renu Rameshan was :
Detecting text in images has been a tough task in image processing, it also has many applications like reducing workload in converting captions, its helpful for users to understand the image. The project will implement the detection of text from image both typed and ***hand written****, recognizing different format likes* ***bold, underline,*** *comma etc. and other layouts like paragraphs, column.*

In this project, we have achieved our goal. We were able to make a HTML which contains the text and is fast as it's time complexity is *O(nlgn)* However, there are some shortcomings which we found it hard to achieve.

First is recognising text which are in bold, italics or strike-through. Our algorithm fails to handle these font styles. We aspire to fix these things in future.

Second is detecting underlines. Our algorithm cannot identify underlines in the images. CCL bounds it to other texts and then Tesseract ignores them. One good approach is putting a threshold on ratio of width/height, which we haven't implemented.

Third is reading handwritten texts. Predicting handwritten text is a difficult task. Tesseract has a relatively low accuracy in predicting these handwritten texts. This can be further improved by using better text classification models like BERT and GPT4. Computer vision coupled with Transformers have achieved a recognition in recognising texts and are better than classic Tesseract.

Fourth are the errors caused by our algorithm. CCL in some cases make mistakes in segmenting the letters in the image. It stitches the letters. This does not affect the accuracy much as at the Tesseract handles it. In this project, we have restricted the dataset to images which are having a white background and black text. It is actually interesting to read text from images of, say, bookcovers, film posters, license plate, visiting card, Railways display screens etc. These are not having a white backgrounds. Detecting text from these is tough and requires a proper algorithm to handle these types of images.

Fifth is the comparison between our algorithm with the Deep Learning algorithms in making contours. DL algorithms are having better prediction and accuracy as compared. In the beginning of the document, we mentioned that they are heavy and requires GPU to run them. But with time, processors are also getting developed. Currently IPhone 13's A5 bionic chips can process at a similar speed of GPU. So they can run these DL algorithms without any problem.

6.2 Future Work

Our future work lies in the collaboration of these classic image processing techniques which can make the processing not only fast but also effective.

One improvement is to make the user independent of thinking any parameter while running any algorithm. So we can improve upon the DB-Scan algorithm. In our current algorithm, we have to specify the epsilon and it considers it constant throughout its clustering process. However, it would be nice if we can make it adapt to a epsilon automatically depending on

the type of the text. Suppose the algorithm is at the heading, it can increase its epsilon to cluster the headings properly. When it is in the body, it can reduce its epsilon to cluster the text given in the image properly. This way we can reduce the burden on the user.

We can think of various other improvements. Our concept of OCR can be used to create music by interpreting the letters in music scripts. Music scripts have various characters which can recognised using morphological techniques and template matching. We can then synthesize music from it, given some template music of flute or violin. There is an active research work in this field and we wish to work on it more. We plan to work on improving our algorithm more by making it more robust to noise, background etc. and publish it in some good conference by mid 2022.

References

[1] L. O'Gorman, "The document spectrum for page layout analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1162–1173, 1993.

[2] R. Brunelli, " Template Matching Techniques in Computer Vision: Theory and Practice," *Wiley*, 2009.

[3] T. N. S.V. Rice, F.R. Jenkins, "The Fourth Annual Test of OCR Accuracy, Technical Report 95-03," 1995.

[4] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, 2007, pp. 629–633.

[5] A. Buades, "A non-local algorithm for image denoising," *CVPR*, 2005.

[6] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[7] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, p. 11–15, jan 1972. [Online]. Available: https://doi.org/10.1145/361237.361242

[8] R. Ahmad, S. Naz, and I. Razzak, "Efficient skew detection and correction in scanned document images through clustering of probabilistic hough transforms," *Pattern Recognition Letters*, vol. 152, pp. 93–99, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865521003408

[9] H. Samet and M. Tamminen, "Efficient component labeling of images of arbitrary dimension represented by linear bintrees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 4, pp. 579–586, 1988.

[10] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD’96. AAAI Press, 1996, p. 226–231.

[12] (2020) Dominate open source Library. [Online]. Available: https://github.com/Knio/dominate/

[13] Yang and Shen, “Book Cover Recognition,” *Stanford University*, 2016.