Digitizing Lecture Notes

Can Kocabalkanli, Rafael de la Tijera Obert, Daniel Hsu
660.461: Computer Vision - Final Project

***Abstract*—A method to digitize lecture notes written on blackboards is proposed to allow students to more efficiently organize and learn from class material. Texts are segmented from the processed image and fed into an open source deep-learning model to recognize handwritten words. Graphs are segmented by locating the axes using the Hough Transform. The preprocessing stage works well in transforming the blackboard image and removing unwanted noise through the use of an Otsu threshold. The handwritten character recognition works well when lines are parallel and words are well separated, but fails when texts intersect each other. Plot segmentation also yields good results, but parameters still need to be optimized for each image in its current state. Future work can be done to combine modules using Deep Learning to automate the process and to improve efficiency.**

***Index Terms*—Handwritten character recognition, Otsu thresholding, Edge filter, Hough transform**

I. Background

Many students find it hard to take complete notes and pay attention in class, especially if the instructor is going fast. This motivated us to develop software to capture the instructor's inscription and drawings, as well as isolating the graphs. In short, our objective is to digitize lecture notes by automatically recognizing handwritten text and plots, then outputting them as digital text and figures.

II. Proposed Method

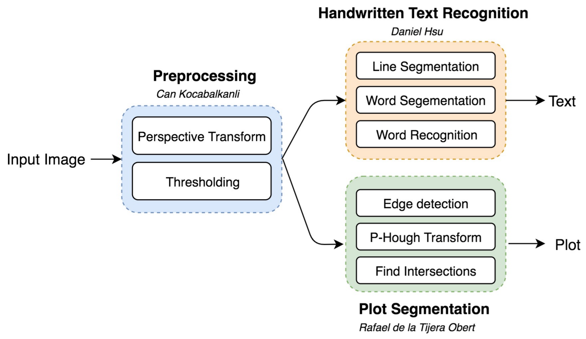

Fig. 1. Pipeline of proposed method with three modules

Figure 1 shows the three modules of our proposed method with their respective stages. A picture of the blackboard is fed as input to the preprocessing module which then outputs a transformed, binarized image. The HTR module performs a series of segmentation on the text on the binarized image until each word is separated and fed into a deep learning model pre-trained on the IAM dataset. The Plot Segmentation module uses a Probabilistic Hough Transform on the image to obtain axes and segment a graph to make it a separate element.

III. Preprocessing

A. Perspective Transform

During a lecture students may not have the luxury to wait after class and capture a well-positioned photo, and instead may have to take a photo from their seats at a skewed angle. The preprocessing stage takes care of this problem by applying perspective transformation [1]. To do so the user defines the corners of the area of interest, which are then ordered using the locations of the corner pixels. Then, the size of the new image array is calculated as the greatest horizontal and vertical distances between any two corners. The transformation matrix between the original and transformed images is then calculated with an OpenCV method, and is used to transform the original image into the new, rectangular one that looks more like lecture notes.

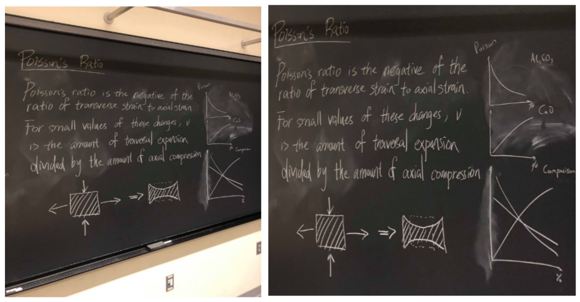

Fig. 2. Before and after picture of perspective transformation on an image of a blackboard

B. Smoothing and Thresholding

After reusing and erasing boards many times, the blackboard surface is no longer clean, and has has less contrast with what's written on it. To alleviate this, the image goes through thresholding to separate the background from the foreground pixels. Multiple thresholding methods such as global and adaptive thresholding were experimented with [2]. However, in the end Otsu Thresholding is used, so that the threshold value is not fixed but is adapted to each individual image, resulting in a smoother distinction between foreground characters and graphs and the background. Before implementing the threshold, a Gaussian Blur is applied to smooth and denoise the image.

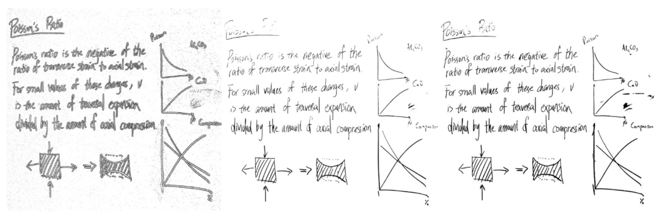

Fig. 3. Comparison of various thresholdig methods. *Left: Adaptive, Middle: Global Binary, Right: Otsu*

IV. Handwritten Text Recognition

As shown in Figure 1, the HTR problem can be treated in three steps: line segmentation, word segmentation and character recognition.

A. Line Segmentation

Line segmentation of handwritten characters differs from that of printed ones in that handwritten text are seldom parallel and characters from different rows can often intersect each other. To solve this problem, our software uses an implementation that can associate components with the correct row by modeling the lines as bi-variate Gaussian densities and looking at the probability of the component under each Gaussian or the probability obtained from a distance metric. The paper from which the implementation is based on demonstrated an accuracy of 98.81% on over 200 handwritten images [3], [4]. Note that Line 1 in Fig. 4 incorrectly segments the word ”of” due to line 3 intersecting line 4 and not being completely parallel with it.

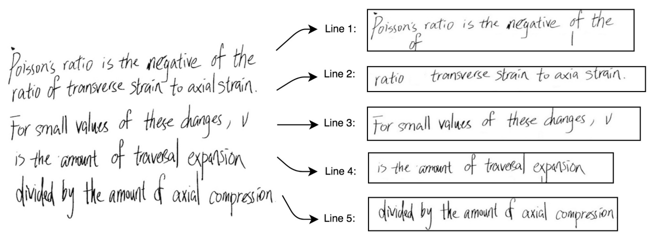

Fig. 4. Line segmentation of the text block

B. Word Segmentation

After the lines are segmented, they are fed into a word segmentation algorithm to produce individual words. The implementation is based on a scale-space blob technique which outlines the method of filtering an image to create blobs corresponding to each word and segmenting them using an connected component algorithm [4]. Figure 5 demonstrates the algorithms effectiveness at segmenting words. Note that the algorithm fails to segment ”to axia” due to insufficient white space between the words.

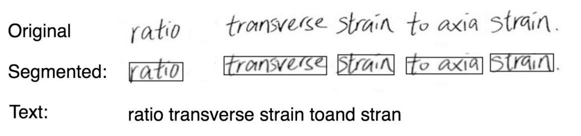

Fig. 5. Word segmentation and recognition performed on Line 2 in Fig 5.

C. Word Recognition

The segmented words are fed into a deep learning model pre-trained on the IAM dataset using Tensorflow consisting of 5 CNN layers, 2 RNN layers and a CTC decoder layer [5]. The implementation provided was able to achieve 68% accuracy on the IAM dataset [5]. Notice in Figure 5 that the model fails to recognize the last two words, partly due to faulty segmentation.

V. Plot Segmentation

After preprocessing, the working image passes through the Plot Segmentation module to isolate it. The image is first passed through vertical and horizontal Sobel filters independently, in order to improve axes detection. These filtered images then pass through a Probabilistic Hough Transform (P-HT) model built-in to the OpenCV library. In the current implementation, parameters such as the threshold and the minimum line length need to be manually optimized for each image. Next, the vertical and horizontal lines generated by the P-HT are checked for preliminary intersection with one another. Constraints are imposed on the preliminary elements to obtain the final set of intersections. Some examples of these constraints are minimum allowable distance from the edges of the image and allowable proximity of intersections to one another. Finally, the final set of intersections is used to mark one of the corners of the segmentation window. In the current implementation, the size of this window needs to be manually optimized for each example to obtain the best result.

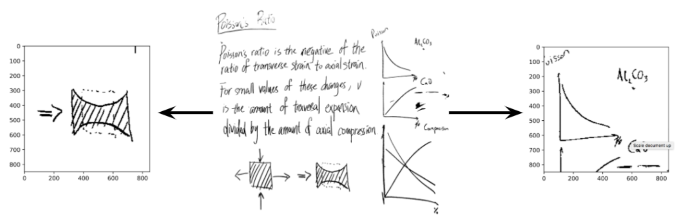

Fig. 6. An example of good plot segmentation (right) with but a non-plot element (left) segmented as well.

VI. Conclusion

Our software can accurately recognize text when the inter-word distance is large and text lines are parallel. Plot segmentation works well but requires manual optimization of parameters to yield the best result. Improvements could be made to more accurately segment diagrams and texts, and do so automatically, using a neural network instead of standard image processing techniques for better results. To improve user experience improvements and features can be implemented. For example, a GUI that allows users to organize texts and diagrams as they desire, automatic equation and simple function-in-plot recognition.

References

[1] Rosebrock A., *4 point OpenCV getPersective Transform Example*, https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/

[2] *Image Thresholding*, https://docs.opencv.org/3.0-beta/doc/py_tutorials /py_imgproc/py_thresholding/py_thresholding.html

[3] ngthanhtin, *Line Segmentation*, https://github.com/ngthanhtin/Line_Segmentation

[4] Arivazhagan, Manivannan, et al. ”A Statistical Approach to Line Segmentation in Handwritten Documents.” *Document Recognition and Retrieval XIV*, 2007.

[5] githubharald, *SimpleHTR*, https://github.com/githubharald/SimpleHTR

[6] githubharald, *Word Segmentation*, https://github.com/githubharald/Word Segmentation