Text Detection From Images Using CRAFT and EasyOCR

Mohammed Ali-Khan

CSE 6367

University of Texas at Arlington

Arlington, TX

Mohammed.ali-khan@mavs.uta.edu

Abstract – Text detection is a key task in computer vision, enabling applications in document digitization, assistive technology, automated content indexing and text reading. This project implements a complete Optical Character Recognition (OCR) Pipeline by integrating the Character Region Awareness for Text Detection (CRAFT) model for robust text area detection, and EasyOCR for multilingual text recognition. The system processes real-world images and image scans, identifies horizontal text regions, and accurately extracts the text content.

I. Introduction

Text Detection and Text Recognition has been a long-standing problem in computer vision. The ability to automatically extract textual information from natural scenes or documents plays an important role in enabling intelligent systems to interact with the real world. The applications for text detection range from assisting the visually impaired with added reading tools, automated license and security recognition, document and form digitalization, content-based image retrieval, and augmented reality systems that can translate text in real time [1].

Historically, Optical Character Recognition (OCR) Technology was always restricted to cleaned and structured documents, such as scanned text, printed forms, and typed pages. Early text detection systems, like Tesseract, were often rule-based, and highly dependent on strict input formats. These systems were prone to failure when recognizing text on natural images, or images with complex backgrounds, irregular layouts, various fonts and styles, and simple text distortions [2]. The evolution of deep learning allowed for OCR to be revolutionized by introducing data-driven approaches of learning different variations of text detection from large datasets. Making it possible to detect and recognize text in more challenging conditions.

In this project, two separate tasks are performed sequentially in order to detect text and then recognize text. Text Detection involves located regions of an image where text is likely to appear. Text Recognition is the process of identifying and interpreting letters and words in the recognized regions. Together, these two processes combine to create a more robust method of text detection.

II. Related Work

The major player used for text detection in this project is CRAFT (Character Region Awareness for Text Detection) [3]. CRAFT detects individual characters as letters and estimates the relation between letters to form words. This allows curved, rotated, and distorted text to be more effectively recognized, when compared to older text detection models. CRAFT then outputs a heat map indicating character presence on an image, and another heatmap estimating the connections between characters [4].

For text recognition, EasyOCR provides an easy-to-use solution to formulate the letters and words made out by CRAFT [5]. EasyOCR builds upon the idea of a Convolutional Recurring Neural Network (CRNN) architecture, which then recognizes such characters in a multitude of scripts and languages. This utilization of deep learning allows EasyOCR to do minimal preprocessing and provide text recognition for a number of image types.

The last element needed is a bridge between the two models used. A way to combine the outputs of CRAFT's text detection heatmaps to use EasyOCR's text recognition process to recognize the text displayed in the image. This is achieved by creating regional boxes between the individual letters estimated by CRAFT to form words. Furthermore, words occupying the same horizontal row can be merged to form a line of text, in order to make the preprocess procedure less overwhelming. The resulting regional boxes of predicted texts can be passed onto EasyOCR, which would then predict the final words in the image [6].

This project proposes a possible solution for text detection from an image by combining both the regional text detection model from CRAFT and text recognition model from EasyOCR. The goal is to develop a robust and flexible system capable of extracting text from natural images with minimal tuning. The system is tested on a dataset of real-world images, with various backgrounds, fonts, and text arrangements to evaluate its effectiveness [7].

III. Implementation

The system of this project consists of the following pipeline stages.

First CRAFT is initialized to detect character boxes from a given image. A model is loaded into Pytorch using pre-trained weights [8]. The EasyOCR Reader is also initialized and also loaded with specific language training [9]. In this project, only English text is to be detected. A folder is used for the input image files and passed on to CRAFT.

CRAFT then takes the image and estimates the location of potential characters, and the affinity between characters. The two heatmaps for character detection and affinity detection are produced [10].

Using the heatmaps, a box is formed around the supposed words. Words that occupy the same horizontal row are then merged into a singular box, in order to reduce the complexity and number of boxes being passed to the EasyOCR Reader [6].

The merged boxes are then passed to the EasyOCR Reader. The Reader detects the individual letters and words using its deep learning training models and its own built-in dictionary [5]. Each input box returns a prediction of the word or phrase recognized, along with a confidence level scored of the accuracy of the model prediction.

Finally, the EasyOCR prediction is given as an output both to the console for debugging purposes, as well as annotated overlay text on the original image with the detected box [9].

IV. Results

The following figures show the results for the project output.

Figure 1. Original Image

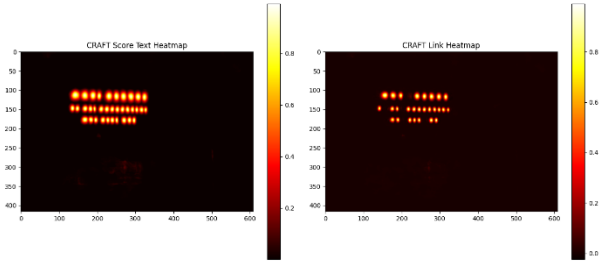

Figure 2. CRAFT Heatmaps

Figure 3. Detected Text Boxes

Figure 4. Modified Merged Boxes

Figure 5. EasyOCR Final Text Recognition

```
[PROCESSING] C:\Users\Abrar Khan\Downloads\test1.png
[DEBUG] Detected 8 text boxes.
[DEBUG] Merged into 3 horizontal boxes.
[TEXT BOX 0] "ROAD CLOSED" (confidence: 1.00)
[TEXT BOX 1] "TO ALL PEDESTRIAN" (confidence: 0.72)
[TEXT BOX 2] "AND BIKE USE" (confidence: 1.00)

Process finished with exit code 0
```

Figure 6. Final Output to Console

The Figures above show the process of the text detection system. Figure 1 is the input image, from which text will be extracted from. Figure 2 shows the output of the CRAFT process, which identifies where the individual letters are most likely to be, and the relationship between letters of the same word.

Figure 3 shows the system creating a box around the supposed text, with Figure 4 merging words that occupy the same line. In Figure 5, the merged boxes are passed to EasyOCR, which recognizes the text and annotates the final output image.

Finally, Figure 6 shows the console output of all recognized text. The output shows how many text words were detected, how many phrases it was condensed into, Each of the phrases that was recognized, and the confidence level of each detection.

V. Conclusion

This project successfully implements a robust and modular text detection and recognition pipeline, by leveraging the strengths and utilizations of CRAFT and EasyOCR. This in turn returns a higher accuracy in both detection and recognition in complex and cluttered image environments. Additionally, the pipeline integrates the two modules allowing for smoother and more accurate recognition.

However, there are further improvements that can be made. The current implementation of the project only detects horizontal bands of text. This means text that is vertically placed, will not be detected well [5]. Similarly, text that is slanted or tilted in either detection will not accurately be recognized [8]. Although the CRAFT detection method recognizes the letters, the box passed to EasyOCR does not account for the tilt, leading to improper text recognition. A possible solution to this problem is to

orient the boxes as horizontal regions before passing to EasyOCR [6].

Similarly, another issue with this implementation is text that contains jagged or staggered placement of letters or curved phrases [1]. This, again, leads to improper results from EasyOCR.

One solution to the problem would be to employ some type of natural language processing along with individual letter placements in order to better predict the complete word or phrase that has been jumbled around. Another possible solution is to shift the individual letters in order to form a more horizontally inline word or phrase, that can easily be detected by EasyOCR. These methods, however, are more complicated and resource consuming, as it becomes a necessary point in the preprocessing of an image. Note that these potential preprocesses are only truly needed for more complex images, as otherwise the images are processed enough for EasyOCR to recognize.

All in all, despite the minor limitations of this basic image detection system, the pipeline is robust and capable of detecting and recognizing text in complex image environments. This project highlights the importance of such implementations and advancements into the field of computer vision. With further development, this pipeline has the potential to be implemented in real time applications such as with assisted reading tools, text detection for the visually impaired, document digitalization and augmented reality for real time translation. These advancements underscore the growing significance of OCR technologies and pave the way for more intelligent, accessible, and interactive visual systems.

VI. References

[1] clumsy computer, “Coding OCR with machine learning from scratch in Python — no libraries or imports! (From Scratch #2),” YouTube, https://www.youtube.com/watch?v=vzabeKdW9tE (accessed Aug. 5, 2025).

[2] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, “Character region awareness for text detection,” arXiv.org, https://arxiv.org/abs/1904.01941 (accessed Aug. 5, 2025).

[3] Clovaai, “Clovaai/Craft-pytorch: Official implementation of character region awareness for text detection (CRAFT),” GitHub, https://github.com/clovaai/CRAFT-pytorch (accessed Aug. 5, 2025).

[4] Comment et al., “Text detection and extraction using opencv and OCR,” GeeksforGeeks, https://www.geeksforgeeks.org/python/text-detection-and-extraction-using-opencv-and-ocr/ (accessed Aug. 5, 2025).

[5] JaidedAI, “Jaidedai/EasyOCR: Ready-to-use OCR with 80+ supported languages and all popular writing scripts including Latin, Chinese, Arabic, Devanagari, cyrillic and etc..,” GitHub, https://github.com/JaidedAI/EasyOCR (accessed Aug. 5, 2025).

[6] N. Saxena, “Pytorch: Scene text detection and recognition by craft and a four-stage network,” Towards Data Science, https://towardsdatascience.com/pytorch-scene-text-detection-and-recognition-by-craft-and-a-four-stage-network-ec814d39db05/ (accessed Aug. 5, 2025).

[7] Ajeetverma, “Text detection with craft,” Kaggle, https://www.kaggle.com/code/ajeetverma/text-detection-with-craft (accessed Aug. 5, 2025).

[8] “Easyocr,” PyPI, https://pypi.org/project/easyocr/1.1.4/ (accessed Aug. 5, 2025).

[9] C. vision engineer, “Text detection with Python and Opencv | OCR using EasyOCR | Computer vision tutorial,” YouTube, https://www.youtube.com/watch?v=n-8oCPjpEvM (accessed Aug. 5, 2025).

[10] R. Mulla, “Textocr - text extraction from Images Dataset,” Kaggle, https://www.kaggle.com/datasets/robikscube/textocr-text-extraction-from-images-dataset/data (accessed Aug. 5, 2025).