Scene Text Detection & Data Extraction

Michael Edukonis
Daniel Ip
Ananay Mathur
Canruo Zou

CS445 Computation Photography
Dr. Derek Hoiem
University of Illinois Urbana-Champaign
December 4, 2023

Motivation/Impact

In today's digital era, textual content is a crucial part of our visual environment, offering essential context to various scenes, objects, or individuals. Despite its prevalence, efficiently digitizing text from diverse and intricate backgrounds like documents, road signs/vehicles, buildings, and more remains a formidable task. This project was selected to explore the convergence of computer vision and computational photography for the purpose of extracting textual information from images. Our objective is to gain a deeper understanding of the complexities involved in text detection across different scenarios and to enhance our proficiency in data extraction methods.

Approach

In this project, we employed the EAST (Efficient and Accurate Scene Text) detector to identify text within various images. Our code, written in Python, leverages several essential libraries to facilitate this process. Key imports include numpy for numerical operations, cv2 (OpenCV) for image processing, and matplotlib.pyplot for visualizing the results. The use of OpenCV is particularly crucial, as it provides the necessary functions to work with the EAST detector and process the images effectively.

Before applying the EAST detector, we conducted image preprocessing to enhance text detection accuracy. The preprocessing steps included converting images to grayscale, applying Gaussian blur to reduce noise, and using Otsu's thresholding for binarization. These steps help in accentuating the textual components against varied backgrounds. The EAST detector, loaded via OpenCV with cv2.dnn.readNet, is then used to detect text. It processes the image and outputs potential text regions as rectangular coordinates. We applied Non-Maximum Suppression (NMS) to these regions to avoid overlapping bounding boxes, ensuring that each detected text region is distinct and clearly defined.

Once the text regions were identified and bounded using the EAST detector, we utilized Pytesseract for Optical Character Recognition (OCR). Pytesseract is a powerful tool that converts the image-based text within the specified bounding boxes into string format, making it readable and usable for further processing. This step is pivotal in digitizing the text content from various scenes, as it allows for the conversion of visual text data into a manipulatable digital format. The combination of EAST for text detection and Pytesseract for text extraction exemplifies a streamlined approach to handle complex text detection and digitization tasks efficiently.
Overall, this methodology showcases an effective blend of image processing techniques and OCR, using Python as the programming language to implement a simpler yet robust pipeline for text detection and extraction.

We have successfully implemented the 3 step pipeline as outlined in “EAST: An Efficient and Accurate Scene Text Detector” (Zhou et al 2017). The images are passed through a deep learning fully convolutional network. Thresholding and non maximum suppression outputs the detection with highest confidence in various orientations.

Results

Three runs were conducted with the same dataset of 37 images of various natural scenes and documents. The first run did not utilize any pre-processing. Run 2 utilized preprocessing function written by the team to introduce Gaussian Blur to reduce noise. Run 3 utilized OpenCV Canny edge detection as a preprocessing alternative. The images were run through the EAST text detector function to locate areas of text and bound them within a box for later processing. No test code was written or available for assessing the success rate of the EAST text detector function on it's own. Despite this limitation, preliminary evaluations using a small dataset have yielded promising results. The team observed that the detector successfully identified text in approximately 90% or more of the cases. However, it's important to note that while the detection rate was high, the detector exhibited some inaccuracies in determining the precise dimensions of the bounding boxes. Additionally, a few instances of false positives were recorded, but these were relatively infrequent. This preliminary assessment suggests that while the EAST text detector demonstrates a high potential for accurate text detection, further refinements are necessary for precise bounding box determinations and minimizing false positives.

The bounded areas of detected text in the images were then put through the Tesseract OCR to extract the text. Test code was utilized at this stage to compare the extracted text vs ground truth text that the team put together after looking at all of the images manually.

1st Run – No Preprocessing
Average Token Match Ratio: 0.05
Average Word Match Ratio: 0.37
Average Line Match Ratio: 0.26

2nd Run
Average Token Match Ratio: 0.02
Average Word Match Ratio: 0.34
Average Line Match Ratio: 0.23

3rd Run
Average Token Match Ratio: 0.01
Average Word Match Ratio: 0.17
Average Line Match Ratio: 0.12

The first run with no pre-processing yielded the best results.

Implementation Details

The team's implementation utilizes several open source libraries. The EAST text detection function is a standard implementation (Mageshwaran 2021) using the pre-trained model "frozen_east_text_detection.pb" which is then passed to OpenCV's deep neural network function to make the prediction for text. An addition by the team was the non maximum suppression code (NMS). Early versions were returning a photo with many bounding boxes. NMS reduced the number of bounding boxes to one with the highest confidence. There are several variables that can be adjusted such as the confidence threshold (90% for our project) and pixels of padding which will add to the height and width of a bounding box around detected text. Experimental runs were completed using various padding and confidence levels however the best results were achieved with 90% confidence and 0 padding.

Pytesseract required some tweeking to improve OCR results. There are some function parameters which helped to improve results such as setting the language and page segmentation (Mageshwaran 2021)

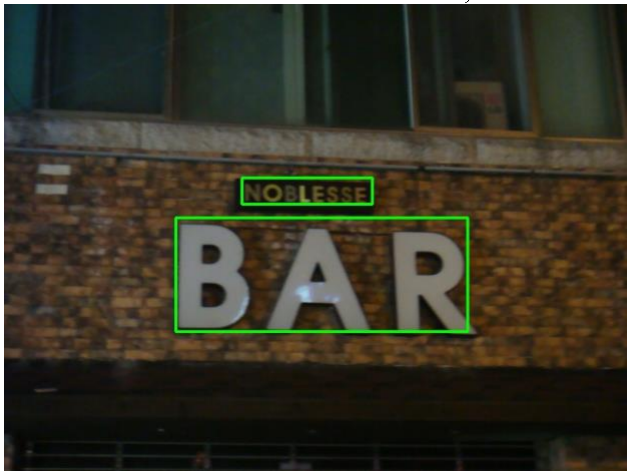
Challenge / Innovation
Some text areas are not recognized no matter what tools are used.
In the paper, specific details about the implementation, such as the software and hardware used, the programming language, and the libraries or frameworks, are omitted. These details are crucial for anyone trying to replicate the study or use the proposed method in practical applications. However, because the proposed pipeline only has three steps, we were able to replicate enough to achieve some success.

In reflecting upon the challenges we faced with the EAST text detector, it's clear that the difficulties may not solely lie within the technology itself, but rather in our approach to pre-processing and the lack of comprehensive implementation details provided in the paper. Certain text areas remained undetected, suggesting that our pre-processing techniques might require refinement to fully harness the detector's capabilities. This aspect is crucial, as effective pre-processing can significantly influence the accuracy and efficiency of text detection.

Furthermore, the paper's omission of specific details about the implementation, such as the precise software and hardware setups, the programming language, and the libraries or frameworks used, hinders the ability of others to replicate or build upon our work. This lack of transparency is a considerable impediment, especially for those looking to apply this method in practical applications or to further research. Despite these challenges, the simplicity of the paper's three-step pipeline allowed us to achieve a degree of success.

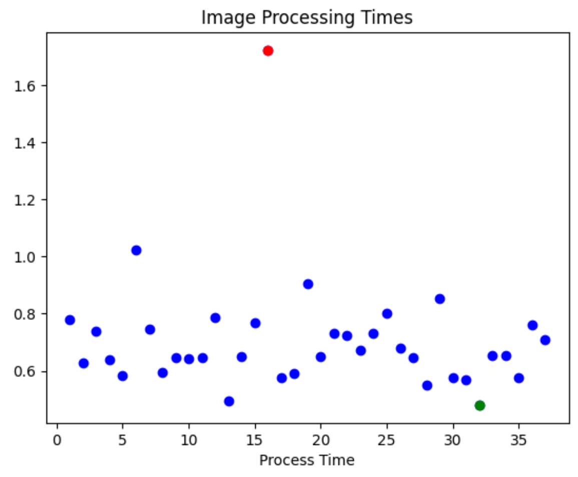
Highest word match, Run 1 photo 30 - 88%

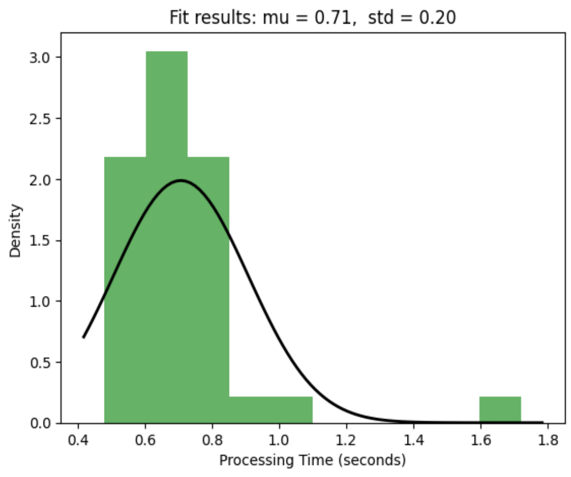
detected tokens: ['inoblesse,bar']
ground truth tokens: ['noblesse', 'bar']
Photo 30 - Token Match Ratio: 0.00, Word Match Ratio: 0.88, Line Match Ratio: 0.16



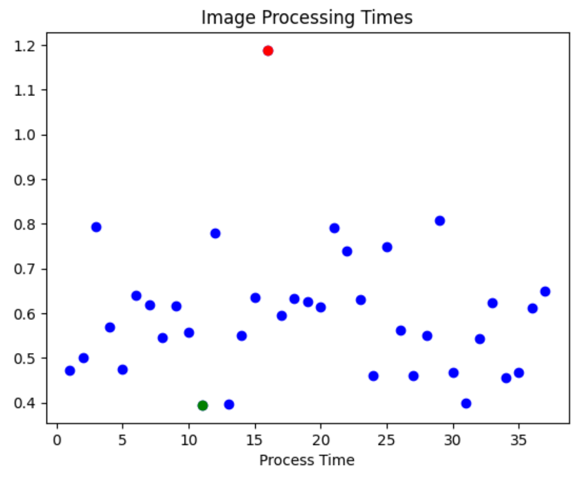
Processing times - seconds

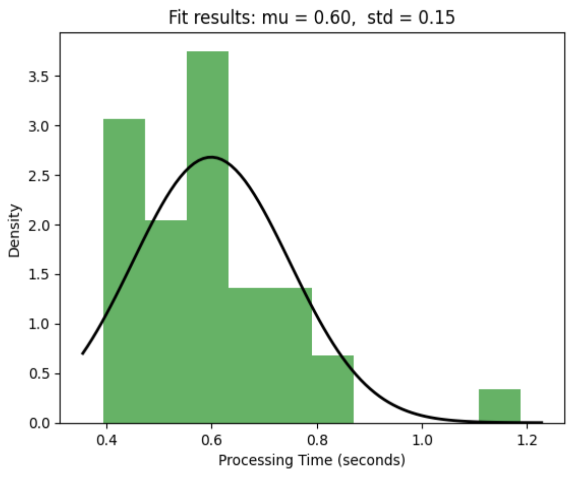
Run 1 – No preprocessing



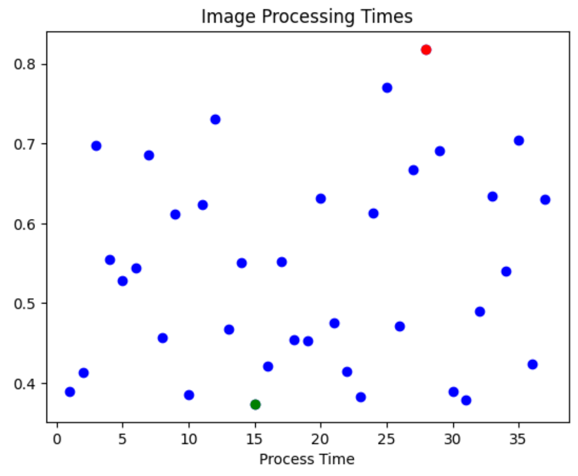


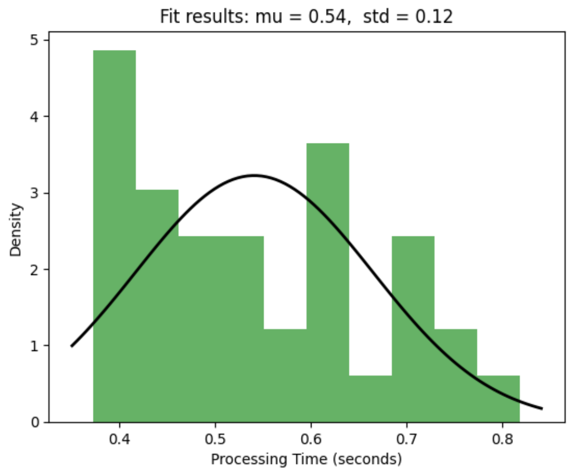
Run 2 – Gaussian Blur





Run 3 – Edge detection





Code repository: https://gitlab.engr.illinois.edu/meduk2/cs445_final_project

Acknowledgments / Attribution

Zhou, Xinyu, et al. "EAST: An Efficient and Accurate Scene Text Detector." *arXiv:1704.03155v2 [cs.CV]*, 10 Jul. 2017, Megvii Technology Inc., Beijing, China.

Mageshwaran, R. "Scene Text Detection In Python With EAST and CRAFT." *Technovators*, 29 Apr. 2021, https://medium.com/technovators/scene-text-detection-in-python-with-east-and-craft-cbe03dda35d5

"7 Steps of Image Pre-Processing to Improve OCR Using Python." *NextGen Invent*, https://nextgeninvent.com/blogs/7-steps-of-image-pre-processing-to-improve-ocr-using-python-2/

Mageshwaran, R. "Review of Best Open-Source OCR Tools for Scanned Documents, STR." *Technovators*, 11 Jul. 2021, https://medium.com/technovators/review-of-best-open-source-ocr-tools-fc839a20e61f

Photos for dataset
"IAPR-TC11." *KAIST Scene Text Database*, International Association for Pattern Recognition - Technical Committee 11, http://www.iapr-tc11.org/mediawiki/index.php?title=KAIST_Scene_Text_Database. Accessed November 20, 2023. Dataset: http://www.iapr-tc11.org/dataset/KAIST_SceneText/(E.S)C-outdoor2.zip.