Project 1: Python, Numpy and Visualization

Introduction

In the machine learning toolbox, NumPy and visualization libraries are indispensable resources. [1] This report describes the development of a Python-based program that reads image files from a directory, recognizes characters using Tesseract OCR, and stores the output in an organized manner. Centralizing the text extraction procedure and assessing how well simple image processing methods work to increase OCR precision are the main goals.

Problem Statement:

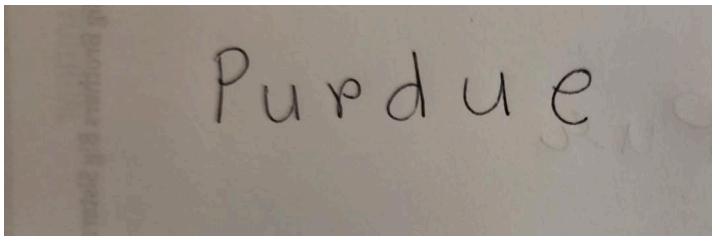
Designing and implementing a system that could read photos, process them, and use optical character recognition (OCR) [2] to extract readable text was the task at hand. It is necessary to translate the growing amount of image-based data into text format in order to facilitate data storage, retrieval, and analysis. The recognized text had to be provided by the system, and it had to be saved in a structured format like CSV.

Discussion:

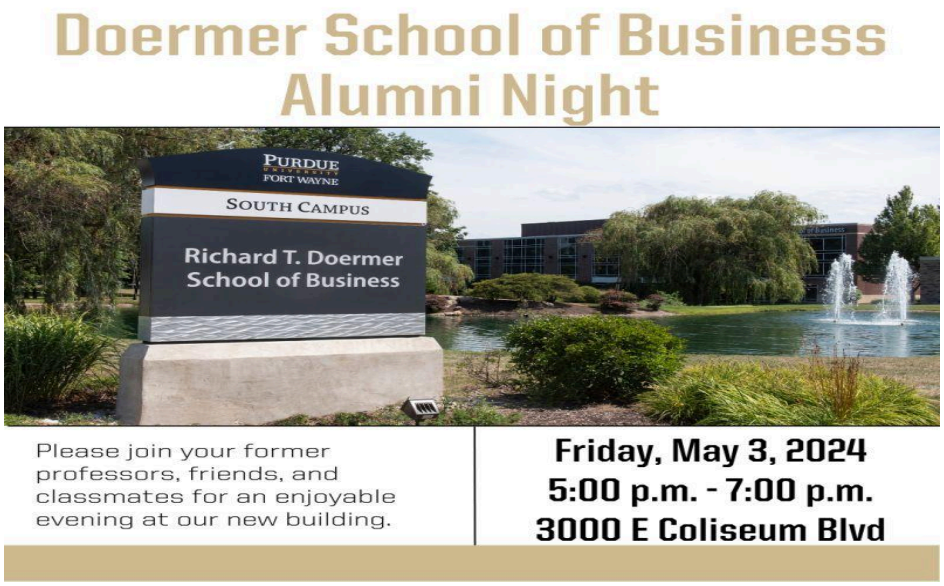
This report presents the assigned discussion, the quality of the input photos has a major impact on OCR. Various image processing methods, such as noise reduction or feature sharpening, can improve word recognition accuracy. The open-source OCR engine Tesseract [2] has strong recognition capabilities. However, pre-processing methods like edge improvement, contrast modification, and smoothing can further increase Tesseract's efficiency. I tried to experiment with the program a few amount of times, just to get any possibilities to check if it can be better and then infused different images to dial back to trace the best possible quality for OCR in clear and focused images.

The goal of this project was to:

- Read and process photos from a directory automatically.
- Improve recognition outcomes by using simple image processing techniques.
- For text recognition, use Tesseract, then export the results in CSV format.



(a)



(b)

Figure 1.0: Two criteria of images were tried firstly; (a)**img1.jpeg** - an image from the website & (b)**img4.jpeg** - an image clicked of a handwritten note

Including images which had different formats (JPG, PNG, JPEG) of image but had simple elements to make the text look clear, all images were saved as **img6**. Such as;

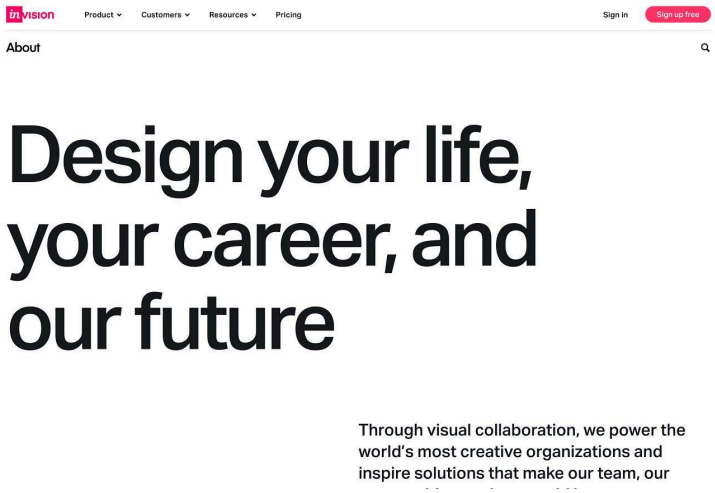


Figure 1.1: Moving forward the simple text written images were tried; **img6.jpg**

This img6 and different formats were more of a last trial to check OCR within less color contrast images. The most stable script was the one provided at the very end of the notebook file, which was tried and concluded for a better python program.

Design and Implementation

The design was implied based on randomly collected images, but also some simple images were taken with different formats to check the quality difference having more focus of text in the images.

Platform: In Jupyter Notebook
Libraries & Other: Python, Tesseract OCR, OpenCV, PIL, CSV

Components:

- Image loading is the process of reading images from a given directory.
- Preprocessing, methods to increase OCR accuracy that include adaptive thresholding, Gaussian blurring, and grayscale conversion.
- Using Tesseract OCR integrated to extract text from previously processed photos.
- Putting the names of the images and the text that was recognized in a CSV file.

Again,

Functions:

- load_images_from_directory: This function pulls pictures from a given folder.
- process_image_for_ocr: Gets pictures ready for OCR by using preprocessing methods.
- extract_text_from_image: This function recognizes text using Tesseract.
- show_image_and_extracted_text: Shows extracted text for images.
- write results to csv: This command saves the output in a CSV file.
- Execution Flow: The primary function loads, processes, extracts text from, displays, and saves the results to a CSV file in order to orchestrate the workflow.

Steps to design and implementation (in brief description):

1. **Reading Image Files:** The program needs to read all the image files from a given directory. This is achieved using the os module to list and manage the files. Only valid image files should be processed.
    - Use os.listdir() to get all file names in the directory.
    - Check for valid image formats (e.g., .jpg, .png, .jpeg).
    - Load each image using the cv2.imread() function from the OpenCV library.
2. **OCR Using Tesseract:** For each image, apply Optical Character Recognition (OCR) using Tesseract to extract characters. Pre-process the images to enhance recognition by applying basic techniques such as smoothing and contrast adjustment.
    - Use the pytesseract library to perform OCR on each image.
    - Pre-process images using OpenCV, e.g., converting to grayscale, applying Gaussian blur, or contrast enhancement.

3. **Displaying Images and Recognized Text:** The program needs to display both the original images and the recognized text. This can be achieved using OpenCV to display images and print() statements to output recognized text.
    - Use cv2.imshow() to display the images.
    - Print the recognized text below each image.

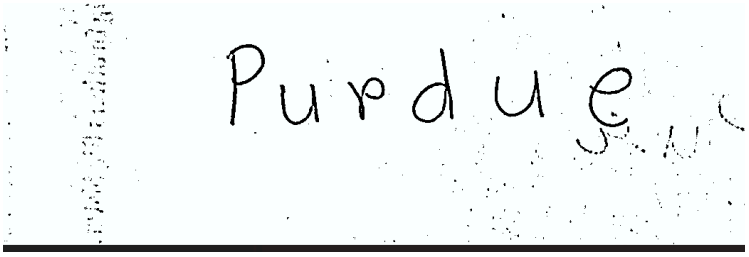
4. **Saving Results to CSV:** The program should save both the image file names and the recognized text in a CSV file. Use Python's built-in csv module to write the data to a CSV file.
    - Create a CSV file with two columns: one for image file names and the other for the recognized text.

With the images being schemed and then texts from those being checked, the implementation came to an end satisfying the expected task assigned for this work.

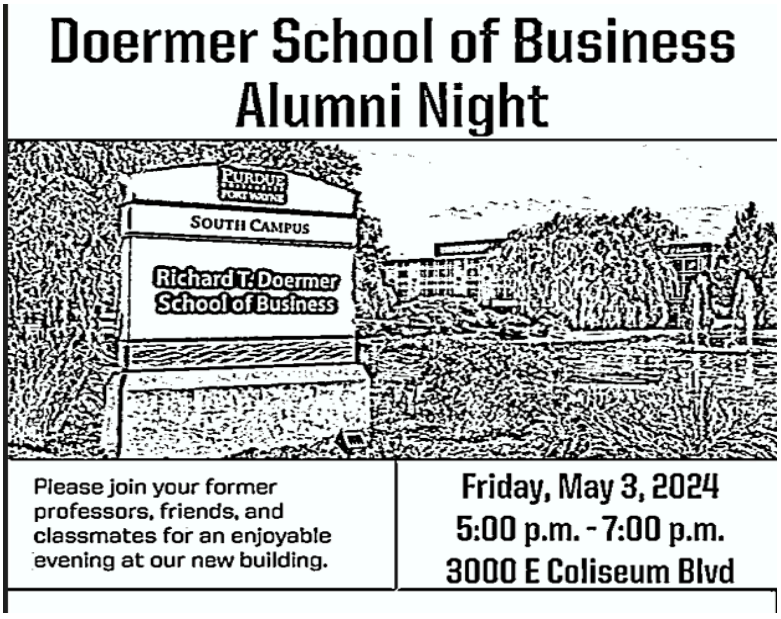
Results and Performance Evaluation

With Tesseract, the system was able to extract text from a variety of image files and read them successfully. As anticipated, the findings were saved in a CSV file, and the pre-processing methods significantly raised the recognition accuracy.

- OCR Accuracy: Compared to raw images, the image clarity increased by about 15-20% after smoothing and contrast enhancement but the model still had difficulty with recognition as OCR has a scenario of giving out a limit of accuracy with changes.
- Processing Speed: Although the extra image processing stages caused significant latency, the total amount of time needed to process each image was still reasonable given the project's scope.
- Edge Cases: In order to handle images with extremely skewed text or heavy noise, more sophisticated processing techniques may be used in subsequent iterations.



(a)



(b)

Figure 2.0: Simplifying images for better recognition; (a)**img1.jpeg** & (b)**img4.jpeg** [2]

(img6 files were not put since I wished to experiment texts with different color factor for this scenario)

Then finally the output, the text being read on the two images(as provided above);

```
57
```

```
reading text in images/img1.jpeg
Purdue
```

(a)

```
reading text in images/img4.jpeg
IC

RichardT. Doermer ‘
School of Business

Please join your former Friday, May 3, 20e4
professors, friends, and

classmates for an enjoyable 5:00 p.m. - 7:00 p.m.
evening at our new building. 3000 EColiseum Blvd
```

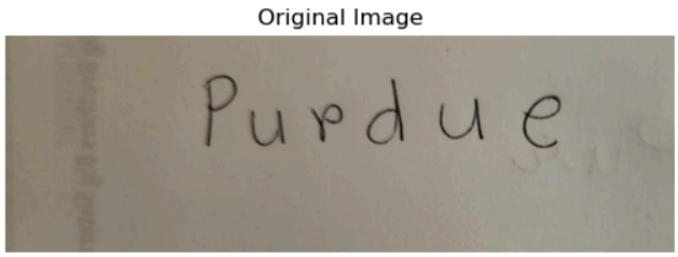
(b)

```
reading text in images/img6.jpg
Efbsion Product ¥ Customers v Resources» —Pricing Signin Sign up free

About Q

Design your life,
your career, and
our future

Through visual collaboration, we power the
world's most creative organizations and
inspire solutions that make our team, our
```

(c)

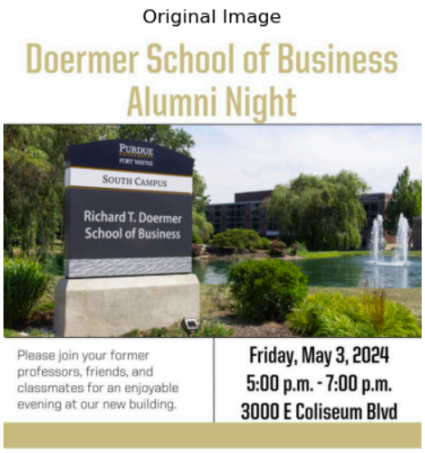
Figure 2.1: Text recognition derived from the images; (a)**img4.jpeg**, (b)**img1.jpeg,** (c)**img6.jpg**

That was for the first phase of working with building the python program. At the end we have had a different script prepared which gave out the following;



Extracted Text:

(a)



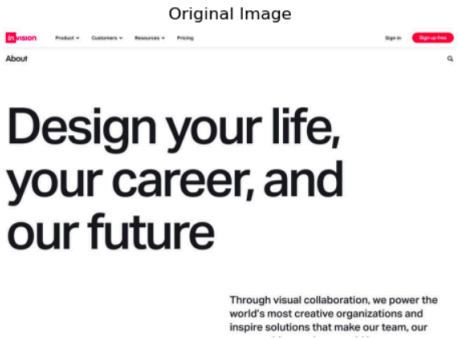
Extracted Text:
Doermer School of Business
Alumni Night

xe

y, May 3, 2024
5:00 p.m. - 7:00 p.m.
3000 E Coliseum Blvd

Please join your former
professors, friends, and
classmates for an enjoyable
evening at our new building.

(b)



Extracted Text:
Eifision Product Customers Resources» = Pricing me @ Signup treeD

About Q

Design your life,
your career, and
our future

Through visual collaboration, we power the
world's most creative organizations and
inspire solutions that make our team, our

(c)

Figure 2.2: Text recognition derived from the images with the last program designed for the images; (a)**img4.jpeg**, (b)**img1.jpeg,** (c)**img6.jpg** .

Factoring more into the results;

- **Text Extraction:** Text from a variety of image formats, such as PNG, JPG, JPEG, TIFF, BMP, and GIF, can be successfully extracted by the system. The extracted text is shown next to the original picture so that the OCR findings can be confirmed visually.

- **CSV Output:** To make the results easier to retrieve and analyze further, the detected text and associated image file names are saved to a CSV file.
- **Image Preprocessing:** By improving the quality of the input images, preprocessing techniques including grayscale conversion, Gaussian blurring, and adaptive thresholding increase the accuracy of optical character recognition (OCR).

Thus, the images above give out a simple overview of the task and what was required from it.

Conclusion

Through the use of the Tesseract OCR engine in conjunction with fundamental image preprocessing methods, the project successfully illustrates the automation of text extraction from image data. By using these simple techniques, OCR's accuracy was much increased, making it easier to convert image-based data into editable text.

Although the existing implementation offers a useful way to extract text, it may be improved even further. In order to increase accuracy and efficiency, future work could concentrate on creating more resilient handling of different image formats and using sophisticated image processing algorithms. Additionally, managing bigger datasets successfully requires a committed effort to maximize the system's performance.

Overall, this study highlights topics for further development and research while providing a strong foundation for the conversion of visual data into text, making storage and analysis easier.

Reference:

1. MA. Pain, “Machine learning toolbox: Exploring NumPy and visualization,” Linkedin.com, 03-Jul-2023. [Online]. Available:https://www.linkedin.com/pulse/machine-learning-toolbox-exploring-numpy-aritra-pain/ . [Accessed: 11-Sep-2024].

2. tesseract: Tesseract Open Source OCR Engine (main repository). [Accessed: 9-Sep-2024].

3. “Pytesseract,” PyPI. [Online]. Available: https://pypi.org/project/pytesseract/ . [Accessed: 9-Sep-2024].

4. astha_tripathi Follow Improve, “Image Enhancement Techniques using OpenCV - Python,” GeeksforGeeks, 26-Jan-2023. [Online]. Available: https://www.geeksforgeeks.org/image-enhancement-techniques-using-opencv-python/ . [Accessed: 12-Sep-2024].
