Reed Rogers
CS 437
University of Illinois
Video: https://youtu.be/esBKtxF1N-Y
GitHub: https://github.com/reedwrogers/Pen-To-Pi/tree/main

Motivation

The purpose of this project, *Pen-To-Pi*, is to create an Internet of Things (IoT) system capable of translating handwritten notes into digitized formats. In a world where digital record-keeping is vital for productivity and organization, handwritten notes remain a cornerstone for creative brainstorming, quick thoughts, and meeting outlines. Personally, I find that handwritten notes are indispensable for my learning and professional workflows. Writing by hand allows me to capture ideas in free-form formats - whether through sketches, diagrams, or quick tables - that are often difficult to replicate with typed notes.

However, this reliance on physical notes creates inefficiencies when it comes to managing, storing, and searching through them. For instance, while digital notes can be indexed and searched for specific terms, physical notes are far more challenging to organize and retrieve. To bridge this gap, digitizers and stylus-based solutions exist, but they require external software that is often incompatible with my setup. Additionally, these solutions cannot fully replicate the tactile experience and ease of use that traditional pen-and-paper methods provide.

This project seeks to address these challenges by enabling handwritten notes to be seamlessly transformed into digital documents, improving their accessibility and utility. Beyond simple digitization, Pen-To-Pi aims to add features tailored to the unique nature of handwritten notes, such as recognizing titles, subtopics, and maintaining version control. These capabilities will make the system particularly beneficial for students, researchers, and professionals who rely on both digital and physical mediums for their work. On a personal level, I anticipate using this system daily to streamline my note-taking process and enhance my productivity. By combining

the best of both analog and digital worlds, Pen-To-Pi represents an innovative step toward more efficient personal information management.

Technical Approach

The Pen-To-Pi project utilizes a high-level, step-by-step pipeline to achieve seamless translation of handwritten notes into digitized formats. The core of the system operates on a Raspberry Pi, which serves as the central hub for image capture, processing, and data transmission. The overall workflow can be broken down into the following key stages:

1. Remote Access and Automation: The system is accessed via SSH to the Raspberry Pi, allowing for remote execution of scripts. This capability facilitates streamlined control of the system, enabling the automation of note digitization with minimal user intervention. Considering the fact that I will be using a different device other than the Raspberry Pi (my work computer) when this system needs to be utilized, ssh'ing in gives me the flexibility needed to run the processes that need to be run.
2. Image Capture and Preprocessing: A script on the Raspberry Pi initiates the capture of an image using a camera module mounted to the desk. The camera is carefully aimed to ensure the entire notebook page is in frame. This raw image is then preprocessed to enhance quality and improve the accuracy of subsequent text recognition. Preprocessing steps include cropping to locate the notebook page, adjusting brightness and contrast, and reducing noise. I managed to achieve very good results with my preprocessing steps and my text recognition model.
3. Text Recognition Using Google Cloud Vision API: Once the image is prepared, it is sent to the Google Cloud Vision API for text extraction. This API processes the image and returns the recognized text in a structured format. The top-left word of the page is used as the title of the note, while all other words are collected and stored in a text file. This ensures that the digitized notes are searchable and well-organized. I used the Google Cloud Vision API because while OCR is possible on the Raspberry Pi (despite limited compute), handwriting recognition is a notoriously difficult task, regardless of the

amount of power on the computing device. Therefore, I decided to outsource my image to Google Cloud Vision, an already established OCR handwriting model. The results were nothing short of amazing.

4. Data Storage and Synchronization with GitHub: The system integrates with GitHub using the GitHub API and an authentication key to upload the digitized content. Three types of files are stored in a designated repository:
    a. The raw image, preserving the original handwritten note.
    b. The processed image, reflecting any adjustments made during preprocessing.
    c. The text file, containing the recognized words for easy searchability and retrieval.

    Github was chosen because it is accessible from any device, and it has an API that is easy to work with. In this way, I will be able to see my notes from any device, and will be able to trigger my Pi using any device that I can ssh with.

This approach emphasizes modularity and scalability, ensuring that each component of the pipeline - from image capture to data storage - operates independently yet cohesively. By leveraging existing tools such as Google Cloud Vision and GitHub APIs, the system minimizes complexity while maximizing functionality. This structure also allows for future enhancements, such as improved image preprocessing techniques or additional features for organizing notes.

By keeping the technical approach focused on high-level stages, implementation details are reserved for later sections, ensuring clarity and readability. This structured process provides a solid foundation for achieving the goals of Pen-To-Pi.

Implementation Details

Physical Set-up of the Pi and Environment

The physical setup of the Pen-To-Pi system was designed with simplicity and precision in mind to ensure consistent and accurate image capture. Two key components of the setup are as follows:

1. **Green Masking Tape for Bounding Box Creation**
   To standardize the area captured by the Raspberry Pi camera and facilitate accurate cropping, I placed green masking tape on the floor of my office to outline the notebook's designated position. This tape serves as a visual marker in the captured image, allowing the scale.py script to identify the bounding box and crop the image effectively. By leveraging this simple yet reliable physical marker, the system minimizes the inclusion of extraneous background in the processed image, ensuring the OCR focuses only on the notebook content.
2. **Raspberry Pi Mounting**
   The Raspberry Pi and its camera module were securely mounted to my desk to maintain a fixed perspective and optimal alignment for image capture. By physically screwing the Raspberry Pi to the desk, I eliminated variability in the camera's position, ensuring consistent image quality and reducing the need for frequent adjustments. This stable mounting setup is crucial for achieving the repeatability required for the system's automated workflow. These physical design choices enhance the reliability of the Pen-To-Pi system and complement the software components by providing a controlled environment for accurate image capture and processing.

Code

The Pen-To-Pi project consists of several Python scripts, each responsible for a distinct part of the workflow. The modular design ensures clarity, maintainability, and extensibility of the system. Below is an overview of each script and its role in the implementation:

photo.py

This script handles the initial image capture using the Raspberry Pi's camera module. Leveraging the libcamera-still command via a Python subprocess call, the script captures a photo and saves it

locally in an images directory on the Raspberry Pi. Since the storage location of the captured image is predetermined, photo.py does not return any values, relying on other scripts to access the image for further processing.

scale.py

The purpose of this script is to preprocess and scale the captured image, focusing on isolating the notebook from the background. To achieve this:

- The script uses the skimage library to identify and crop the image to the notebook's bounding box, with the help of green masking tape placed around the notebook as a physical marker.
- After cropping, the script rotates the image to ensure it is horizontally aligned for optimal text recognition.
- The preprocessing is finalized by calling the preprocess_image_for_ocr method from process.py, which enhances the image for OCR (Optical Character Recognition).
- The scaled and rotated image is saved with a new suffix (_scaled_rotated) for clarity.

process.py

This script contains the logic for preprocessing the image to prepare it for OCR. The method follows a structured pipeline:

- Load the image and convert it to grayscale.
- Apply a Gaussian filter with a sigma of 0.2 to reduce noise, as higher sigma values were found ineffective for handwriting.
- Convert the image to an 8-bit unsigned integer format.
- Binarize the image using a threshold to separate text from the background.
- Invert the image so that text appears white on a black background, which improves OCR accuracy.
- These steps optimize the image for accurate text extraction by the Google Cloud Vision API.

words.py

This script handles interactions with the Google Cloud Vision API. It includes the following methods:

- CloudVisionTextExtractor: Communicates with the API to process the scaled and preprocessed image.
- getTextAndTitleFromVisionResponse: Extracts all recognized words and determines the note's title based on the word located at the top-left corner of the image. The bounding box of the annotation is checked to ensure it corresponds to the top-left position.
- get_words(): Combines the above methods to return both the note's title and a list of all recognized words.

main.py

This script orchestrates the entire process, integrating functionalities from the other scripts. The main steps are:

1. Retrieve the scaled and preprocessed image from scale.py.
2. Set the GitHub API key, repository name, and other credentials.
3. Extract the title and words from the image using the logic in words.py.
4. Define file paths for the raw image, processed image, and text file.
5. Upload these files to a dedicated folder in the GitHub repository using a customized commit message.

By modularizing the implementation into distinct scripts, the project ensures that each component focuses on a specific task. This approach simplifies debugging, enables easy integration of new features, and ensures the overall robustness of the system.

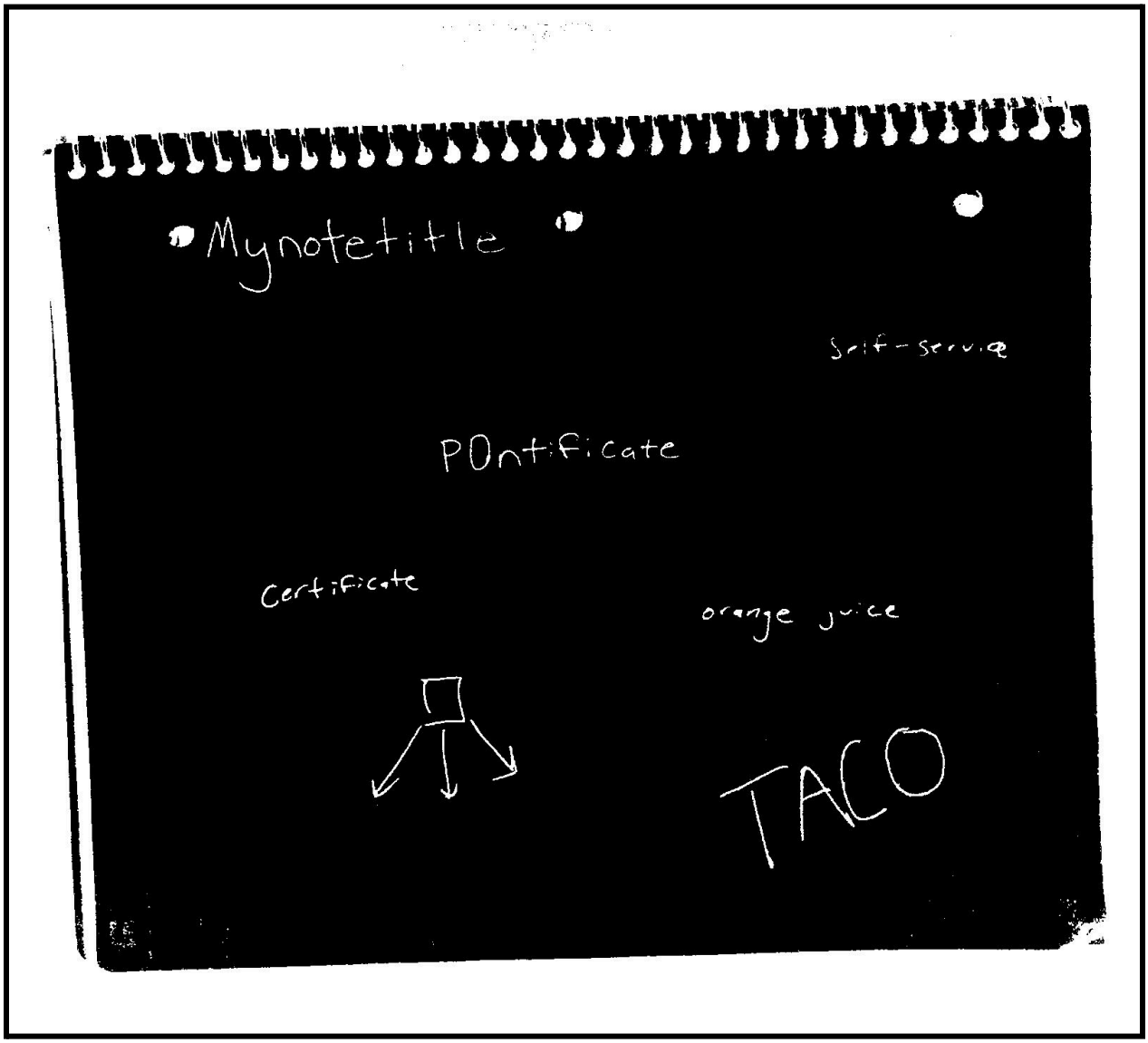
Results

As far as results are concerned I can give some details into what my implementation really looked like in this section. Let me start by showing the setup of my Pi on my desk:

Now, let me give an example of a typical image that is captured using my Raspberry Pi, including some same sample notes on the page.

- As you can see, this is the green masking tape that is on the floor of my office. It is used to help my processing function create the bounding box and crop the image.
- The lighting you are seeing may seem somewhat dark, but in actuality, I discovered that this is the perfect amount of light for processing the image. When I used more light, many times my camera was unable to pick up the differences in contrast between the page and the words.

After running my processing function, the resulting image was generated on the Pi (using Google Cloud Vision API) and sent to the GitHub repository:



- Not only is it quite easy to read the words that are on the page from a human perspective (which will be important, as most of the time I will be simply going into the repository to look at my notes, many of which will be more diagram intensive rather than word intensive), but Google Cloud Vision was able to read these words extremely well. You can see this in the other file that was generated and sent to the repository, which is a list

of all words in the image (as deduced by Cloud Vision) in a .txt document:

“

• Mynotetitle

Certificate

Pontificate

Self-service

orange ju

TACO•MynotetitleCertificatePontificateSelf-serviceorangejuTACO

“

As you can see, most of the words in the image were returned in this document. This allows for searching of the GitHub repository for key topics.

- I will also note that this image you see, and the processing function I created, took a lot of tinkering to get fully correct. I believe that this is due to the variety of images that can be preprocessed for OCR. In my case, I was working with a very white background, a thinner pen stroke, and a lower light environment, so parameters needed to be adjusted accordingly. Accuracy was overall extremely good, with my system almost always grabbing the title and text of the image correctly.
- On average, Google Cloud Vision was able to churn out these results in less than about 15 seconds. This differs greatly from some of the OCR models I tried running locally on my Pi. We can only assume that this is due to the difference in processing power of my Pi and whatever machines Google’s Cloud Vision is using, as well as the ability of the model they have.
- Lastly, I will say that these results are going to be extremely helpful for me in my day to day job. I can now take notes during meetings, or draw diagrams, and instantly convert those into digital assets that can be shared with team members, or just stored for my own personal use later. Not only this, but I now have a system to search what exactly I was writing notes about, and when I was writing these notes. I may experiment with retroactively scanning notes from previous dates, or just start fresh from now.

As far as other results are concerned, you can simply view my public GitHub repository as it is linked at the beginning of this report to see my code and other examples of images taken. You can even try searching for specific topics as GitHub repositories allow for a pretty useful search capability, preventing me from needing to create anything from scratch. Thank you for taking the time to look at my final project!