**Background**

Dyslexia affects 5-10% of the general population, with some estimates going up to 17%.

This has a profound effect on productivity of the labor force, as well as creating economic loss as

tens of thousands of dollars are spent by families to accommodate their dyslexic children.

However, even after their formative years, dyslexic people struggle to hold down a job due to

their disability. Furthermore, about 3.4 million people in the US alone are blind.

One of the best investments any dyslexic person can make for their children is a purchase

of the pen reader. It is a product for dyslexic, visually impaired, or otherwise handicapped people

that reads written text aloud upon scanning text. It utilizes an Optical Character Recognition

engine implemented in software to take raw input from the scanner at the front of the pen and

cleans it up until it recognizes a character of a certain font. The pen reader then compiles words

and sentences out of these characters and outputs and reads them aloud. It can also read aloud the

meanings of certain words so that the user can understand unfamiliar words. It can also translate

between languages. However, the pen reader's limitations include an inability to read colored text

on colored backgrounds. This can get in the way as a lot of jobs depend on reading text whether

or not it is of a different color, or that it is on a colored background.

Our main objective is to explore the use of different filters and image processing

techniques to improve the pen reader's performance in reading colored text on colored

backgrounds, or to design our own pen reader which would be able to read such text.

**Statement of the Problem**

Current pen readers are very advanced and optimized with their proprietary OCR engines, however, they cannot read text in color, or text on colored backgrounds. This is a problem because a lot of work is done with text that isn’t simply black text on white background. For example, a supermarket needs workers to sort out their inventory and determine which items are on sale. Items that are sold in these stores are almost never simply black text on white backgrounds, but may be colored or stylized. People with reading difficulties and visual difficulties are then effectively barred from working these jobs.

**Rationale**

The fundamental reasons behind the hardware are thus:

The Raspberry Pi 3B was chosen because it is a small computer with the most resources, processing power, and documentation available.

The speaker amplifier and speaker were chosen to make the text audible as well as keep the size small enough to have the ability to attach directly to the Raspberry Pi

The battery was chosen to match the size of the Raspberry Pi, as well as provide a large reservoir of energy to be able to use the color pen reader for a long time. The estimated battery life of the color pen reader is 5 hours at minimum (with constant scanning).

The Raspberry Pi Camera Module V2 was used to capture the images. It is the only camera module that is compatible with the Raspberry Pi by default. Other cameras can be used but due to time constraints we were not able to test different cameras with the color pen reader.

The Trinket M0 was primarily chosen for its small size and powerful chip. It is also phenomenally easy to program. A microcontroller in general was used because they are better at handling realtime inputs, outputs, and operations.

**Design**

The design of the color pen reader was chosen to be simple and minimalist to satisfy time constraints and to have as much time to refine each component as possible. The design is detailed in the block diagrams on the USB. First, the camera controller runs and takes visual input from the page being scanned and input from our feedback system. Upon encountering the signal that the user is scanning and the signal that a picture of the text should be captured, the camera controller sends the command to capture the current image and save it in the rawImages directory. It repeats this until the user stops scanning. The feedback system uses a button to tell when the user is scanning text and a rotary encoder to measure the distance that the color pen reader has traversed, and sends a voltage impulse whenever a certain distance (about a centimeter in our case). The set of images then goes into an image stitcher that combines the images into one single image. This then goes into the color filter stage, which detects the edges of the text image. Then, it scans each letter for its type and figures out whether the current character has no holes (like an L), one hole (like an O), or two holes (like a B). Every letter in the English language is one of these types. It then fills in the text with black color. Then the image goes into Tesseract to convert into a string of text and is read out by festival, a text-to-speech converter. Tesseract and festival were installed so that we didn’t have to implement our own OCR engine and text-to-speech converters (a much too difficult task for our available resources and time constraints).

**Development**

Development started with a meeting with AHRC and an actual user of current pen reader technology. We then determined the requirements of our project, which was originally to design a modular attachment for an existing pen reader and to use a robot arm to make it autonomously sort objects by label. We proceeded with research on how to filter out the colors from a text image. We tried various methods of edge detection like Sobel filtering, making a filter that utilizes the cosine similarity theorem, and a Canny edge detector. We determined that the Canny filter worked best of all. We then started to work on how to fill the text in with black color. At the same time we also contacted the company that manufactures the pen reader we were modifying and found that it was impossible to make any attachments. With this new information our team decided to design its own color pen reader from scratch using open-source components.

We first drew up the high level design of the color pen reader. Once we were satisfied with the design we proceeded to determine the requirements of each component that we planned out. After determining this we started implementing each component backwards; that is, from the final stage that gives the user the output to the input stages. We chose to do this because the color filter was already half finished and we successfully tested the Tesseract OCR engine together with a text-to-speech converter called festival. Not only that, but the inputs to the already-finished components were already well-defined. Working backwards thus made development faster.

After creating the color filter and implementing festival and Tesseract, we implemented the image stitcher quickly and started work on the feedback and the camera controller. The feedback was developed first because that determines the inputs to the camera controller. Just after finishing the code for the feedback, it was already a couple of weeks before the CREATE symposium. So, we paused the main development and instead focused on building the first prototype to show at the symposium. For this we also quickly designed a simple 3D printed scanner and attached all the hardware necessary to make it portable (the battery, amplifier, speaker, button, and camera). This first prototype could take a single image, filter out the color successfully, and speak what was on the image. After the symposium we finished the feedback and the camera controller and assembled the final prototype in order to enable the color pen reader to read lines of text in multiple images instead of just one at a time. A new 3D printed enclosure and wheel were also designed to house the feedback system as well. We assembled the final prototype, tested it, determined the issues that should be resolved afterwards, and presented it at our final presentation.

**Evaluation**

The color pen reader has limited functionality, with some major issues to be resolved. The main issues, as outlined in the demonstration video, include a latency between the feedback system sending the signal to capture an image of the text and the camera actually capturing the image. Not only that, but the latency almost seems to vary. This can be resolved by using different hardware or by using an algorithm that is faster at capturing images, perhaps using two threads running on two of the Raspberry Pi’s available four cores (one for capturing the image and saving it to a buffer and another thread for writing the buffer into an image file). This should remove any variation between the latencies, and once they are equal, the images taken can be cropped according to where an overlap between the images exists. This would clean up the input to the processes that process the input images into sound.

Another major issue is that the Tesseract OCR engine in its out-of-the-box form does not read text at any angle. If the text is not completely horizontal, Tesseract will think it’s looking at an empty page. This can potentially be resolved by training Tesseract with a database of images (since it is based on a neural network). This should resolve any reliability issues assuming the user doesn’t scan the text in a wavy motion.

The speaker that outputs the generated speech is quiet. This can be rectified by selecting a different sound amplifier to use with the color pen reader. The speakers may also be changed in case they are not as efficient as they can be.

There are issues with focus in capturing images. Due to the nature of using a camera to capture images instead of something like a linear CCD sensor, the camera needs to be at the correct distance away from the text in order for the image processing to take place correctly. We have placed the wheel of the scanner at approximately the correct distance away, but there are issues when the color pen reader is not vertical. This can potentially be corrected by either using a different sensor altogether to capture images of the text or by adjusting the parameters of the color filter (in the edge detection portion). There can also be an LED light that tells the user when the camera is in focus (by affixing a realtime edge detector to the camera controller and finding a maximum in the edge detector’s output, which controls the brightness of the LED).

These issues must be fixed by future developers if the color pen reader is to be reliable enough for practical use.

**Discussion**

The color pen reader is able to filter out any colored text on any colored background. Our design is such that in theory it should be able to even filter out multicolored text. However, there is still room for improvement.

The color pen reader can be made more efficient in software in the processing required. We have tried to minimize processing time but due to time constraints, there should be room for improvement. A more efficient algorithm means a less powerful computer is required to contain the color pen reader. This can be a Raspberry Pi Zero, and owing to its small size, it is less power-hungry. This means a smaller battery can be used and the color pen reader’s size can be optimized into a single handheld device.

The color pen reader is designed to be modular, so more stages can be added in between different stages, and therefore it can be expanded upon. There can be a dictionary or a translator function built in between the Tesseract stage and the text-to-speech stage, so that language barriers can be lifted along with reading difficulty barriers.

**References**

Siegel, Linda S. “Perspectives on Dyslexia.”, Paediatrics & Child Health 11.9 (2006):

581–587. Print.

Bailey R. N., Indian R. W., Zhang X., Geiss L. S., Duenas M.R., Saaddine J. B. “Visual

impairment and eye care among older adults—five states,” MMWR 2006; 55(49):1321–1325.