A Design of Assistive Technology Application for Visually Disabled Elderly Citizens

John-Tyler Sprankle
Ashland University
Ashland, OH

Selvanayaki Kolandapalayam Shanmugam
Assistant Professor/CS
Ashland, OH

ABSTRACT
In today's world, the growth in technology diminishes or eliminates the barriers faced by people with disabilities, especially visually and hearing-impaired citizens. This research study aims to design a device/application that could convert text to voice and vice versa and impart any societal benefit problems. The objective of this work is to create a program using Python that utilizes available Python libraries to convert a visual image of text into spoken words. In this case, a visual image of text could be a newspaper article or other text that one may encounter in daily life. One advantage of Python is that it contains an extensive set of libraries that any user can make use of. By utilizing libraries such as OpenCV, PyTesseract, and Google-Text-to-Speech this task was completed. This research work design is handled in three major steps: In the first step, the input image is taken and preprocessed using image preprocessing techniques; The second step includes the detection and extraction of text from the preprocessed image; The third and final step handles the process of converting the extracted text to its related voice using supporting libraries and techniques. The limitations could still apply to the hardware used in this project. A possible enhancement could be extending this system to have the features of speech-to-text.

General Terms
Pattern Classification, Data Preprocessing, Text Extraction.

Keywords
Data Preprocessing, Image Preprocessing, Text Extraction, Text-To-Speech, TTS, Speech-to-Text, STT, PyTesseract, OpenCV.

1. INTRODUCTION
There are many people throughout the world who are visually impaired or hearing-impaired. One group of these people includes the elderly who may experience problems with their vision or hearing as they age. The objective of this project was to create an application that would assist those who are visually impaired by providing them with a text-to-speech (TTS) converter. Through the utilization of Python's various libraries, an application was created that satisfied this objective.

The program currently assists those with visual impairments by helping them to read words that they may encounter in day-to-day life that they may otherwise be unable to read. This is done by letting the user take a picture of the text in question and the program will automatically convert the words in the picture into audio that they can instead listen to. This is accomplished using data preprocessing. Data preprocessing is the process of taking data gathered in a raw form and turning it into a specific format that can be utilized by a machine. In this case, the data that is being preprocessed is the image that the user takes. Though they may take a picture of text, say for example a newspaper article, there may be things in the background of this image that are distracting. The goal of data preprocessing is to remove these unwanted distractions or noise from the image so that the program can easily extract the text from the image.

This is a general-use application, but it could be specialized to fulfill certain needs. Currently, it is optimized to read dark text off a white sheet of paper, but it could be changed to read text from different sources as well as perform various operations with this information. One example could be the implementation of the scanning of pill bottle labels. The program could read the text from the labels and store useful data such as instructions on how to take the drug, as well as the specified refill date on the packaging. This data could then be read back to the user who may otherwise have trouble reading the label.

2. LITERATURE REVIEW
The first step in creating this application was learning the architecture behind TTS and speech-to-text (STT) applications as well as how to go about starting one. Existing systems were reviewed for both TTS and STT conversion and a standard set of steps for developing a speech recognition system was developed (Lawrence et. al). The most important aspect of TTS and STT conversion is to assure that any input speech/text that is gathered should first be preprocessed to eliminate unwanted background noise (Ayushi et. al). Only after preprocessing is finished can the speech/text then be transferred to a pattern classification algorithm where the input is recognized, and a correct output can be created.

It was important to create an application that was user-friendly to those who are visually disabled. Without the proper accommodations, the application could be useless to users who may need it. It was found that many people who are visually disabled still use a standard keyboard while some use braille keycaps or large print keycaps (Burgstahler). Some users will have trouble viewing material on the screen, so it is important to give information as well as instructions in an audio format.

Each Python library has its own official documentation that informs the user how the library works as well as the methods it contains and how to implement it into a program. For instance, the Google-Text-to-Speech documentation (GTTS, 2023) goes in-depth on how to fully utilize the library. It contains information such as the parameters that need to be given to the TTS method including the text itself, the language, and how fast the text should be read. Similar information for each Python library can be found in its own documentation. A review of such documentation is necessary when creating such a project as it is important to fully understand how a library works and whether it is suited for the application.

3. METHODOLOGY
3.1 gTTS
Google-Text-to-Speech or gTTS is a Python library that utilizes

Google Translates text-to-speech Application Programming Interface (API). An API is a way for two or more computer programs to communicate with each other. In this application, this library is used to turn text within the program into an mp3 file which in turn can be played to hear the original text as speech. This library has many different languages that it can recognize such as English, French, and Mandarin. The library also allows the user to manipulate how the text is spoken such as reading the text aloud slowly (GTTS, 2023).

3.2 OpenCV

OpenCV is a large library of functions that is focused mainly on real-time computer vision. In this specific project, it serves two major purposes. The first is to capture video by capturing individual frames and displaying them one after another in quick succession. This is done by creating a loop in the program where each iteration will show a single frame from a camera. This loop continues to run until the user manually stops it, and the last frame is then saved for use in the program (Getting Started with Videos, 2023).

The second purpose of OpenCV is to preprocess an image by removing the background through the process of masking. Masking essentially removes unwanted parts of a specific image. OpenCV performs masking by converting a specified image into Hue Saturation Value (HSV) format. This format allows individual colors within an image to be detected and altered. After the image has been converted the user can specify a range of HSV values that they wish to preserve. By running OpenCV's inRange() function with the parameters of the HSV formatted image, and the upper and lower range, all other HSV values will be removed from the image. The preserved colors in this case are displayed as white while the removed colors are all displayed as black (How to Use Background Subtraction Methods, 2023).

3.3 PyTesseract

PyTesseract is an optical character recognition Python library. When PyTesseract is given an image, it can recognize text that exists within that image. It reads this text much like how a person would read a book, from left to right and from top to bottom (PyTesseract, 2023). Once this text is extracted from an image it can either be saved as a text file, or as an object within the program itself.

3.4 OS

OS is a library in Python that includes miscellaneous operating system interfaces. In this application, the OS library has only one purpose, to play the finalized mp3 file. This is performed by simply running the OS library's system command to open the specified mp3 file, this will then cause the mp3 file to automatically begin playing (OS, 2023).

4. PROPOSED SYSTEM

The proposed system follows a process of gathering an input image, preprocessing that image, extracting text from the image, and finally converting the text into an audio format that can be played to the user (see Figure 1).

The first step of the TTS application is to get an image from the user that contains some form of text. In this case, OpenCV is used to open a camera that will be used to capture an image. When the camera is opened audio instructions are given to the user, and they read out "Press F to save an image of the text." A loop is then started within the program. This loop captures a single frame from the camera and then displays it to the user. This loop will continue to run indefinitely until the user stops the loop with the press of a specified button, in this case, the F key. Once the loop has been stopped the last frame is saved for the next part of the program.

The next step of the program is to preprocess this image. The only part of the image that is needed is the text, so everything else within that image is considered noise that needs to be dealt with, or else it may stop the program from running correctly. Preprocessing the image is done with the help of OpenCV and some of its various functions.

First, the image is converted into HSV form, which allows operations utilizing HSV colors to be performed. Once this conversion is completed the masking operation will take place. First, a lower and upper bound is set. These bounds will determine what range of colors will be allowed to remain through the masking process and other colors will be filtered out. In this case, the bounds were set such that dark blackish colors, such as black ink on a piece of paper, would appear, while lighter colors would be removed. These darker colors are turned into a single white color, while the lighter unwanted colors are turned into black (see Figure 2). The colors that are masked do not need to follow this exact pattern and can be changed depending on the text that needs to be read.

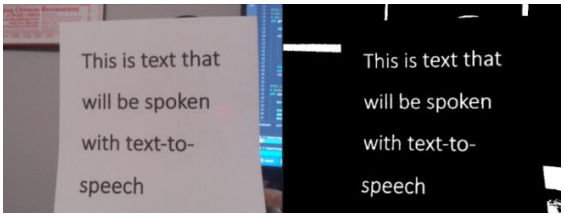

Figure 2: Side-by-side comparison of original and preprocessed image

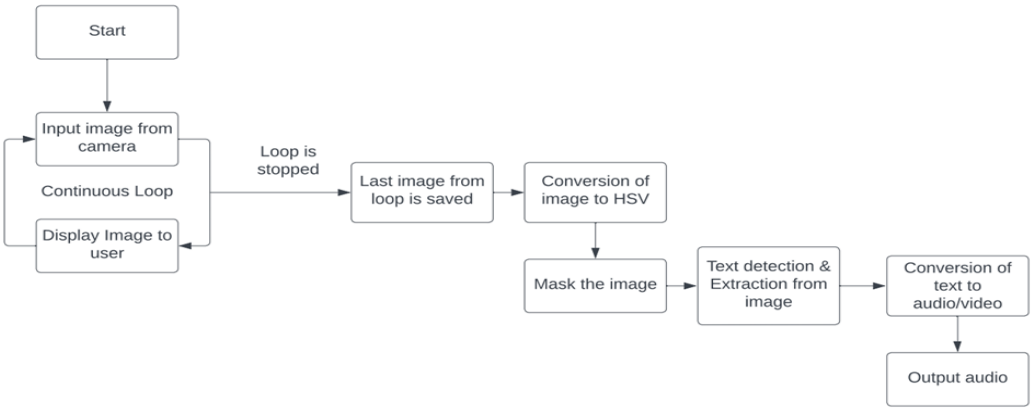

Figure 1: Flow diagram for the text-to-speech application

With all the background noise removed the image can easily be processed with the image-to-string function in PyTesseract. First, a language must be specified, in this case, English was selected. Then the image is simply passed as a parameter along with the language into the image-to-string function and the text that is found is saved as an object. Referring to Figure 2, PyTesseract extracts the text "This is text that will be spoken with text-to-speech" and saves it as an object within the code. The string object can then be used by the gTTS library to create a new object that essentially contains the text in a spoken format. Other parameters can be passed to gTTS depending on the situation. For instance, if the user wanted the text to be read to them slowly the parameter 'Slow = True' could be passed which would cause the mp3 file to be read aloud at a slower pace. This gTTS object can then be saved as an mp3 file using the save method. With this mp3 file saved to the machine, the OS library can be used to play the mp3 file. In this case, the words "This is text that will be spoken with text-to-speech" will be read aloud for the user to listen to.

5. RESULTS

5.1 General Results

Throughout the testing of the application, it was found to be successful if it receives quality input. This is important as the quality of the output completely relies on the quality of the input. If the input image contains dark text and that text is clearly displayed on a light background, the application will translate the written text from the image into spoken audio. Multiple different types of input were tested throughout the development. For instance, handwritten text can be translated if the handwriting is written neatly and if the writing utensil such as a pen or pencil is dark enough for the application to properly mask it.

5.2 Another Possible Preprocessing Method

There were multiple different data preprocessing methods that were reviewed throughout the development of this project. One such method that was implemented and tested was blurring the background of images using PixelLib. PixelLib separates an image into separate objects and performs methods on these objects based on the user's needs (PixelLib's Official Documentation, 2023). This library worked consistently when blurring out the background in images but the model that the library was trained on only recognized certain things as qualified objects. In this case, only things such as people, vehicles, bikes, and some animals such as cats were recognized. This made it impossible to not blur text unless a person was holding an object with text in their hands, which is not always the case. PixelLib would have been a potential option if there was an existing model for recognizing text, as creating a new model would not have been possible for the timeframe of this project. Because of this, OpenCV was selected as the preprocessing method using its masking process.

5.3 Problems with Image Masking

While image masking is the best choice when it comes to image preprocessing for this application it does have its downsides. As discussed previously, the bounds of the masking were set such that dark blackish colors would appear, while lighter colors would be removed. Because of this if the text on a page of something like a magazine is too light, the program will filter it out and it will not be able to be translated. See Figure 3 which contains a side-by-side comparison of a non-preprocessed image and a preprocessed image where the text on the page is white with a yellow background. Because of the preset bounds, the masking process masks out the white text as it is searching for dark text. While this can easily be fixed by changing the bounds to accommodate a light color text, it could be difficult to do this in a real-world scenario as the bounds would need to be changed in real-time. Currently these bounds are preset within the program.

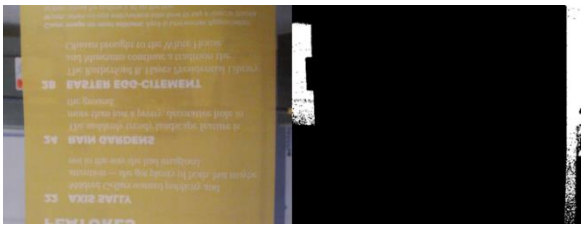

Figure 3: Side by side comparison of original and preprocessed image with lighter colored text

5.4 Proper Hardware

A current limitation resides with the quality of the output as it completely relies on the quality of the input. In the early stages of the application's development, a 1280x720 pixel resolution camera was used to capture all the images. While this was satisfactory for larger text, it had some trouble reading smaller text as it often became blurry and therefore the program was unable to translate it. The camera was replaced with a 1920x1080 pixel camera which eliminated this issue. To avoid similar problems, users must make sure that the equipment that is used is high quality to ensure proper output.

6. CONCLUSION AND FUTURE ENHANCEMENTS

Using various Python libraries, the goal of phase one of this project was accomplished. An application was created that when given the input of an image, it will be able to convert the text from that image into an audio output. By utilizing the concept of data preprocessing the background noise was able to be removed from the input and the program in turn became more accurate in its data reading. Though limitations such as the camera quality play a factor in the overall performance of the application, this can easily be improved with the proper equipment.

Currently, this application design assists those with visual impairments, but the goal is to assist those who also have hearing impairments. Currently, the program takes a visual input of text and converts it into audio that the user can hear. In the future, the program will be enhanced to also allow the user to input audio such as spoken words. This audio will then be converted into text that the user will be able to read. The application could also be improved by implementing a specialization such as a pill bottle scanner as mentioned in the introduction. This would turn this specific application into an application that serves a greater purpose.

7. REFERENCES

[1] Ayushi Trivedi, Navya Pant, Pinal Shah, Simran Sonik, Supriya Agarwal, Speech to Text and Text to Speech Recognition Systems- A review, IOSR Journal of Computer Engineering, (2278-8727), Volume 20 No.2, Mar-Apr 2018.

[2] Burgstahler Sheryl, Working Together: People with Disabilities and Computer Technology. University of Washington. Retrieved March 26, 2023, from https://www.washington.edu/doit/sites/default/files/atoms/files/Working_Together_People_with_Disabilities_and_Computers_a11y.pdf.

[3] "Getting Started with Videos." OpenCV. Retrieved March 26, 2023, from https://docs.opencv.org/4.x/dd/d43/tutorial_py_video_display.html.

[4] "GTTS." GTTS. Retrieved March 26, 2023, from https://gtts.readthedocs.io/en/latest/.

[5] "How to Use Background Subtraction Methods." OpenCV. Retrieved March 26, 2023, from https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html.

[6] Lawrence Rabiner, Biing-Hwang Juang, B.Yegnanarayana, Fundamentals of Speech Recognition. Retrieved March 26, 2023, from https://dl.acm.org/doi/10.5555/153687.

[7] "OS." Python Software Foundation. Retrieved March 26, 2023, from https://docs.python.org/3/library/os.html.

[8] "PixelLib's Official Documentation." PixelLib's Official Documentation - PixelLib 0.4.0 Documentation. Retrieved March 26, 2023, from https://PixelLib.readthedocs.io/en/latest/.

[9] "PyTesseract." PyPI. Retrieved March 26, 2023, from https://pypi.org/project/pytesseract/.