Leveraging Artificial Intelligence to Augment the Visually Impaired

Harry J McCaffery

Northern Michigan University

CS 480

November 29, 2021

Abstract

While the initial intent of this project was to explore the various challenges presented when designing and building a device to interface with the visually impaired, it rapidly developed into an experiment in the many engineering challenges presented in stretching the capabilities of the hardware used, and the many workarounds in both hardware and software made in order to create a functioning device. These workarounds and techniques were mae requisite by the goals of the project, which included:

I. Glasses are wearable on the user's head and possess a sleek, presentable design.
II. Glasses can last five or more hours on battery power
III. Glasses operate reliably as intended
IV. Processes images and provides output in five seconds or less
V. Can be used discreetly in variable environments
VI. Can be interfaced with intuitively without the use of eyesight
VII. Device makes use of recent technological developments allowing for a product much cheaper than others on the market.

VizAssist, as it will be referred to throughout this paper, was built with the intent of upholding these rules to the furthest point capable, and exploring the limits of the technologies used. The primary use case of the device is by someone who possesses failing- but not entirely useless vision in situations that do not consider the visually impaired.

Introduction

I. Hardware

The VisAssist platform consists of two parts: the actual interface device, which provides sound to the user and possesses a camera, and uses a raspberry pi zero w as its brain, as well as a main processor device that acts as a main server as and handles image processing and other more intensive tasks.

Initially, the device was going to be self-contained to the glasses, but a two part system was designed because it was not possible to both have enough processing power for the desired features and still possess a small form factor. Even were it possible, the power draw would SEVERELY limit battery life. For example, a realistic battery capacity that could fit into this form factor would equal approximately 2000 mah @ 3.7v. Assuming perfect efficiency, which is impossible (a reasonable estimate might be close to 80%) this leaves a capacity of 7.4 watt-hours. The raspberry pi zero draws a maximum of 1.1 watts, so this device's battery, in practice, will last 5.3 hours minimum. For comparison, the main processor used has a maximum power draw of well over 20 watts, which would require a battery that could not practically be fit into a glasses sized device, but can be easily fit into a bag or pocket.

II. Software

Both the glasses and the processor run variants of Ubuntu, selected for its ease of use, compatibility, flexibility, and simplicity. The glasses run a specific version, designed to be

extra-lightweight, and for a pi, called DietPi. The software is primarily written in python, which is widely integrated with many different neural network and artificial intelligence libraries. Some of the libraries used include:

- Tensorflow: a widely-used library for machine learning and other artificial intelligence applications
- Pyttsx3: used for speech generation
- OpenCV: used for image processing
- Tesseract: an open-source library maintained by Microsoft for processing text. It is very useful, however can be finicky by default.
- Textblob: dictionary powered text processing library. Useful for telling gibberish from not, which AI can be bad at on its own.

Design:

Upon startup, the glasses search for and connect to the processor, which is running a server written in python, and broadcasting its own wifi signal at 2.4 ghz. This signal is reliable, and because the signal is broadcast by the server, it naturally does not require another network.

Numerous software techniques were required in order to achieve the final product. In the initial design, a live video feed was passed to the processor for image processing. This technique proved to be inferior to providing an instantaneous snapshot on button press, for the following reasons:

I. It creates a much greater burden on both CPUS, especially the processor, thus drastically reducing battery life.

II. The supported resolution is much lower. Only 1080x720 was achievable in the stream, however 2592 x 1944 is achievable per photograph, and even higher resolutions are achievable with slightly better hardware. High resolution is integral for neural networks.

III. Finally, due to latency and the nature of the interface, it is much easier to interact with the device this way.

Once the image is captured, it is passed to the server via a POST request, along with the length of the button press. If it is a short press, the text in the image will be processed with the designed code, and it will be passed back to the glasses. A long press passes the image to the captioning AI, and the returned description is passed on. The text processing code functions as follows.

Two approaches can be made to text recognition by a neural network: training the network to recognize and decipher texts of all fonts and formats, or preprocessing the image to make the text much more recognizable to the AI. Unlike a human, a neural network might recognize black on white text of a given font, but not red on blue if it hasn't learned this heuristic. Training the AI to recognize all colors and shapes and other nuances is unrealistic, unreliable, and even if it does work, slower, and for this reason the latter method was chosen. The image is passed through a filter to convert it to grayscale, a gaussian blur (to remove noise and effectively edge-detect the text, and erode and dilate the image, to effectively further this effect.) The intent

of these operations is to get the image as close to 1 bit color as possible, because this is what the Microsoft's Tesseract model is trained on, and this is the model we will be using to decipher the text.

After the preprocessing, the text must be located efficiently using a deep learning neural network. OpenCV allows us to apply the model to the image, and return the locations of the detected character strings. The EAST model, developed by Megvii Technology in Beijing, China, is used for its efficient and accurate text detection. It is worth noting that while this algorithm can efficiently detect the location of text, it is not capable of discerning the meaning of the text.

Next, the image is spliced into many smaller images where the text was detected, and these images are passed on to the tesseract algorithm. This technique is SIGNIFICANTLY more accurate than just splicing, or just preprocessing, as the tesseract algorithm performs best when passed images containing only text, as close to one-bit as possible. Once this algorithm is looped over (the most processor intensive part of the program, taking 0.5-2 seconds for a normal amount of text) an array of detected text is returned and is ready for post processing before being returned to the interface device.

While the words returned are generally very accurate, the EAST and tesseract networks are incapable of discerning between gibberish and human-readable words. This is why the Textblob library is used to separate readable text from not, and then only return adjectives and proper nouns, as these are most likely to be useful descriptors, unlike other parts of speech such

as articles and conjunctions. Finally, this text is returned within 5 seconds via http to the interface device.

In the case of a long press, a different, much more simply implemented yet equally useful feature is applied. The image is passed to a deep learning powered Neural network- trained on a dataset of over 8,000 images, (Hodosh, P. Young and J. Hockenmaier Volume 47, pages 853-899M) for the captioning of various random images and scenes. This model is applied to provide the user with a general description of a setting, giving them useful information such as the type of room/location they are in, subjects such as people or animals, etc. While this model is fast in actual use, and accurate, it is extremely resource intensive to generate, and somewhat stochastic as well. While an initial attempt was made to generate the model using tensorflow's CPU build, this quickly proved impractical in spite of the fact that the CPU being used was relatively modern. Upon moving to a CUDA based GPU powered build, the model was able to be generated much more quickly, (thanks to the RTX 3070) but it still took a fair bit of time (upwards of an hour), and over 20 GB of RAM. Also, the stochastic nature means that given x amount of time to generate a model from a given dataset d, n number of times, each model generated will have a different 'fitness' or ability to accurately describe a scene.

Once the model returns a description, this is passed back to the interface device which then speaks the text to the user via a custom built audio amplifier, specically designed for the pi zero, as it does not natively possess audio output capabilities. While this audio output does place a burden on the battery life of the device, it was designed in such a manner as to minimize physical footprint and power draw, and allow for flexible output devices.

Conclusion

While the device in current state meets the numerous goals laid in the outset of the project, there are a variety of ways that it could be improved. While many very low cost components are used, a slight upgrade in quality (and cost) of the components would have significant positive impact on the performance of the device, especially the camera. The processor could also be condensed, both in terms of cost and form factor, to a much smaller state. The artificial intelligence capabilities could be much more efficiently utilized (in terms of time and power consumption) if the processor were to possess a sufficient GPU, however this may not be practical in a real world scenario. Finally, a few more features such as color and facial recognition could make it directly competitive with other products on the market while still maintaining a much lower cost.

Citations

DATAmadness. *Tensorflow 2 - CPU vs GPU performance comparison*. (2019, October 27). Retrieved December 1, 2021, from https://datamadness.github.io/TensorFlow2-CPU-vs-GPU.

Hodosh, P. Young and J. Hockenmaier *Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics", Journal of Artificial Intelligence Research*, Volume 47, pages 853-899M (2013) . http://www.jair.org/papers/paper3994.html

Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017, July 10). EAST: An Efficient and Accurate Scene Text Detector. Retrieved December 1, 2021, from https://arxiv.org/pdf/1704.03155.pdf%C2%A0.

Balot, H., Chollet, F., Kalbermatter, S., Brownlee, J., Geitgey, A., Malisiewicz, D. T., Zdziarski, D. Z., Szalontay, Z., & IP, P. (2021, June 9). *You can master computer vision, deep learning, and opencv.* PyImageSearch. Retrieved December 1, 2021, from https://www.pyimagesearch.com/.

NVIDIA. (n.d.). *CUDNN Documentation*. NVIDIA Documentation Center. Retrieved December 1, 2021, from https://docs.nvidia.com/deeplearning/cudnn/developer-guide/index.html.