Eye-Tracking and Pitch Modulation for Speech Therapy

Michael Clayton, Navonte Riggin, Max Dixon, Patrick Bobbie

Kennesaw State Computer Science Department

Abstract

Aphasia is a language disorder resulting from trauma to the brain. There are different diagnoses in each case of Aphasia that it can affect a person, one of which is causing difficulties in speaking even if the afflicted understands the meaning of the words. Speech therapy has been shown to help patients recover from even from digital means. We have created an app targeting Android platforms that allows users to enter text and have it be read aloud through by using eye tracking and text-to-speech.

Keywords

Eye tracking, deep learning, convolutional neural network, pitch modulation, speech therapy

Table of Contents

Contents

1. Introduction

Aphasia is a language disorder resulting from trauma to the brain. One of the symptoms is a loss of phonology in speech functions although it varies by each case. In some cases of Aphasia, the patients do not have difficulty comprehending the meaning of words, but they do struggle with pronunciation of those same words [1]. For these patients, speech therapy is a treatment used to improve their speed and effectiveness of their recovery. Speech therapy can be used through digital means by using text-to-speech functions to properly pronounce syllables for a patient to repeat [9]. This method is cheaper than requiring trained professionals to guide the process and allows for more convenience. Aphasia patients are notably capable of still singing words that they cannot say while talking normally. Melodic Intonation Therapy (MIT) is a form of speech therapy that uses this concept to introduce more musical such as pitch modulation to assist the process [2].

Eye tracking technologies can be divided up into two main categories: model-based methods and appearance-based methods. Model-based methods try to estimate the gaze of the user by building accurate 3D models of the eyes, nose, lips, and other facial features [6]. These methods are older and more resource intensive making it less viable for running real time on mobile devices. Appearance-based methods input image data taken from the user's eyes and learn a mapping function to estimate the user's gaze by using datasets [7]. While training an appearance-based method is resource intensive, the outcome is a mapping function that is easier for mobile devices to run in real time.

2. Related Work

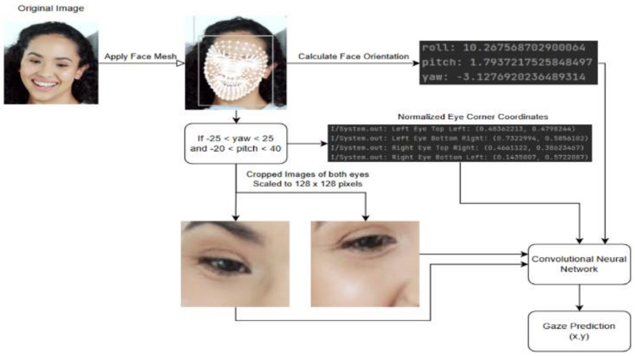
The GazeCapture dataset was designed to be used by appearance-based methods of gaze estimation which relies on the eyes of the user for direct input. The data was collected through crowdsourced means totaling 1474 different subjects and 2,445,504 frames of data. Each frame has the subject looking at a fixed point on their screen in various environments for a comprehensive set of data [3]. The introduction of this dataset allowed for appearance-based methods to develop further with no increase in complexity. A convolutional neural network (CNN) named iTracker was developed alongside the GazeCapture dataset using the eyes, face, and a face grid for inputs. The model was successful, and it achieved errors of 1.53cm on mobile phones and 2.38cm on tablets that was further increased through calibration to 1.34cm on mobile phones and 2.12 on tablets.

Using the GazeCapture dataset, EyeMU was created for the purpose of allowing users to interact with their device using solely their gaze and gestures [4]. The EyeMU implementation is an appearance-based method that tracks the eyes using a CNN to estimate the user’s gaze by inputting crops of both eyes. Along with the yaw, pitch, and roll of the user’s head which is collected using Face Mesh [5] they achieve an accuracy of 1.7cm.

3. Design

Our implementation was designed to be used on Android, specifically Android versions 10 and 11. It was developed using the Flutter library which allows it to be easily ported to iOS, Windows, and other applications in the future. It consists of two main segments: the eye tracking portion and text-to-speech portion. Both parts work together to read text aloud that is contained within the app on the main screen based on the where the user’s gaze is. Additionally, the user is given the option to change volume, text size, the speed of the text-to-speech functionality, and the color of the app to better suit readability. The eye tracking system can also be turned off to prevent unwanted reading of words.

3.1 Eye Tracking


Original Image
Apply Face Mesh
Calculate Face Orientation
If -25 < yaw < 25
and -20 < pitch < 40
Normalized Eye Corner Coordinates
Cropped Images of both eyes
Scaled to 128 x 128 pixels
Convolutional Neural Network
Gaze Prediction
(x,y)

```
roll: 10.267568702900064
pitch: 1.7937217525848497
yaw: -3.1276920236489314
```

```
I/System.out: Left Eye Top Left: (0.48362213, 0.4798244)
I/System.out: Left Eye Bottom Right: (0.7322994, 0.5856182)
I/System.out: Right Eye Top Right: (0.4661122, 0.38623467)
I/System.out: Right Eye Bottom Left: (0.1435807, 0.5722087)
```

Figure 1: Logic flow of eye tracking software

The eye tracking portion of our application uses Google ML Kit and TensorFlow to help with providing the gaze estimation capabilities. Google ML Kit allows us to apply a face mesh over a user's face, which allows us to extract features from the face that will be fed into a convolutional neural network. A face mesh is a coordinate mapping of a face that contains 468 different points on the face. The convolutional neural network is trained on the Gaze capture data set, which is a dataset that is commonly used for training models for gaze estimation purposes. A limitation of Google ML Kit is that the face mesh API is in beta, so it's only available for Android devices. So, while the app is a Flutter application, all the main functionality is on the Android side.

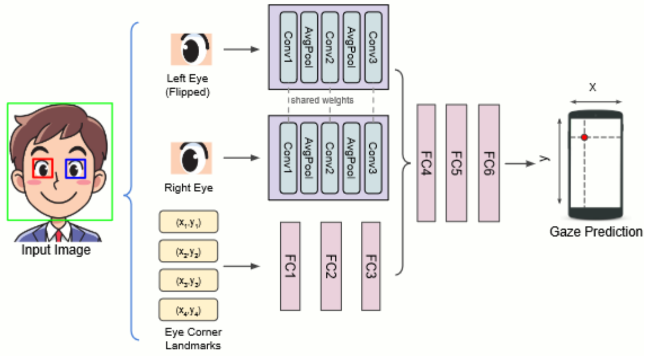


Figure 2: Architecture of model for eye-tracking based on Valliappan *et al.* [8]

When the eye tracking is activated, the camera will record frames, and if a face is detected in the frame, then Google ML Kit will apply a face mesh to the detected face. Then using this face mesh, we can calculate the orientation of the face by taking two different points on the face, and using a little bit of trigonometry, we determine the angles of the roll, pitch, and Yaw. If the Yaw and the pitch are within a certain range, the Yaw being from -25 to 25 degrees and the pitch being from -20 to 40 degrees, the app assumes that the user is looking at the screen and it will crop images of the eyes using the face mesh coordinates. After the eyes are cropped out, it will flip the image of the left eye horizontally so that the convolutional neural network can use the same set of weights for both eyes. Then it extracts the coordinates for the eye corners and normalizes them by dividing the x and y values by the width and height of the recorded image. All of these are fed into a tensor flow convolutional neural network using the ReLU activation function which will output a gaze prediction.

3.2 Text-to-Speech

The text-to-speech and syllable detection part of the project uses three main flutter packages: flutter_tts, audioplayers, and wav. Once the text-to-speech function gets called by the eye tracking, the flutter_tts package creates a waveform file of the word that gets passed through. Using the wav package, the audio can be analyzed to find the syllables and pair them up with music notes. Looking at the audio wave, the amplitude can be observed to determine when the text-to-speech is speaking the word and when it pauses, with the pauses representing breaks between syllables. Every 50 milliseconds, the function gets an average of the absolute value of the amplitude for that time range. Using this information, the peaks and valleys of the audio can be found, and the timeframes for the syllables are recorded. Once this information is gathered, the audioplayers package can play the text-to-speech sound along with different piano notes behind each syllable.

Looking at the amplitudes in Figure 3, it is clear that the text-to-speech is talking from roughly the start to about 0.45 seconds and from around 0.60 seconds to 1 second. The low amplitudes in the middle show the pause between the two syllables.

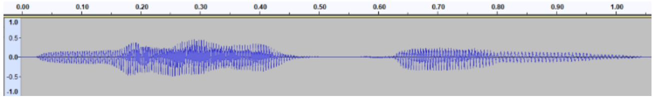


Figure 3: An example of an audio waveform using the word "welcome."

4. Results

While our eye-tracking model does run properly, it is wildly inaccurate. We believe that the training process is the source of this error and hope to re-train the model in the future. After this, we hope to achieve accuracy that was seen in the iTracker and EyeMU models. The system does accurately keep track of whether the user is facing the device. In the meantime, we have been using tapping to mark words to be read by the text-to-speech.

The biggest issue with the text-to-speech system is the inaccuracy of the syllable detection that happens with some of the words. While the word "welcome" from Figure 3 works very well, some other words do not. Instead of having a clear wave shape, they look more like a block with little to no pauses between the syllables. This messes with the system and causes one singular note to play behind multiple syllables. Outside of these cases, the text-to-speech works as intended.

In addition, the text-size, volume, and the speed of words spoken by the text-to-speech are all easily adjusted within the app allowing for the user to fine tune their experience to what fits them best.

5. Conclusion

Our work created a strong foundation that can be further developed. We have developed an app that allows the user to tap words in order to read them using the text-to-speech at various sizes of text, various speed of the text-to-speech, and two contrasting color modes. The eye-tracking system, while not giving the intended results, is already built and only requires tuning to give valid results. The text-to-speech system highlights difficulty in detecting the syllables of words that have constant waveform amplitudes but works as intended on words with strong syllables. Overall, there were issues, but there has been potential shown for this technology that we hope to take further in the future.

References

[1] Coppens, Patrick. *Aphasia and related neurogenic communication disorders*. Jones & Bartlett Publishers, 2016.

[2] Norton A, Zipse L, Marchina S and Schlaug G. Melodic intonation therapy:

shared insights on how it is done and why it might help. Ann N Y Acad Sci. 2009;1169:431-6. doi: 10.1111/j.1749-6632.2009.04859.x.

[3] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchi Bhandarkar, Wojciech Matusik and Antonio Torralba. “Eye Tracking for Everyone”. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[4] Andy Kong, Karan Ahuja, Mayank Goel, Chris Harrison. EyeMU Interactions: Gaze + IMU Gestures on Mobile Devices. In *Proceedings of the 2021 International Conference on Multimodal Interactions (ICMI ’21)*. Association for Computing Machinery, New York, NY, USA, 577-585. DOI:https://doi.org/10.1145/3462244.3479938

[5] Google. 2021. MediaPipe Face Mesh. https://google.github.io/mediapipe/solutions/face_mesh.html.

[6] R. Stiefelhagen, Jie Yang and A. Waibel, "A model-based gaze tracking system," *Proceedings IEEE International Joint Symposia on Intelligence and Systems*, Rockville, MD, USA, 1996, pp. 304-310, doi: 10.1109/IJSIS.1996.565083.

[7] L. Jigang, B. S. L. Francis and D. Rajan, "Free-Head Appearance-Based Eye Gaze Estimation on Mobile Devices," *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Okinawa, Japan, 2019, pp. 232-237, doi: 10.1109/ICAIIC.2019.8669057.

[8] Valliappan, N., Dai, N., Steinberg, E. *et al.* Accelerating eye movement research via accurate and affordable smartphone eye tracking. *Nat Commun* **11**, 4553 (2020). https://doi.org/10.1038/s41467-020-18360-5

[9] Oche A. Egaji, Ikram Asghar, Mark Griffiths, and William Warren. 2019. Digital Speech Therapy for the Aphasia Patients: Challenges, Opportunities and Solutions. In Proceedings of the 9th International Conference on Information Communication and Management (ICICM 2019). Association for Computing Machinery, New York, NY, USA, 85–88. https://doi.org/10.1145/3357419.3357449