2022
THE JOURNAL ON TECHNOLOGY AND PERSONS WITH DISABILITIES

Teleconference Sign Language Detection

Shane Angel, Allison Tate, Christian Vogler and Raja Kushalnagar
shane.angel@gallaudet.edu, allison.tate@gallaudet.edu,
christian.vogler@gallaudet.edu, raja.kushalnagar@gallaudet.edu
Gallaudet University

Abstract

Teleconferences spotlight the active speaker, based on audio. For deaf signer accessibility, we report user evaluations of a sign-detection algorithm to spotlight the active signer.

Journal on Technology and Persons with Disabilities
Santiago, J. (Eds): CSUN Assistive Technology Conference

Introduction

The pandemic led to the increased use of technologies like Zoom, which is helpful for large group meetings without additional costs, and educational institutions and companies have been using Zoom during the pandemic. (Iqbal, 2021). However, Zoom is not deaf-friendly since its spotlight feature only detects voices and doesn't have the technology to detect signers, which leads to professors having to spotlight every signer, a time-consuming task. Sign language detection [3] is defined as the binary-classification task for any given frame of a video if a person is using sign-language or not. Unlike sign language recognition, where the task is to recognize (classify) and interpret continuous signs in a video, or sign language identification, where the task is to identify which sign language is used, the task of sign language detection is to detect when something is being signed, which is easier to solve than recognizing discrete or continuous sign language detection. By reducing this to a binary classification, we can automatically spotlight any person who is signing, lessening the burden on the host, who otherwise must manually spotlight every DHH person when it is their turn to speak using American Sign Language (ASL). Apple already had sign detection technology used in FaceTime to spotlight any signers automatically, but they did not share their coding or technology with anyone else, so that leads to another current sign detection technology which Google invented. The Google app aimed to detect sign language while improving the accuracy of detecting sign language from non-sign language. We plan to use the Google app to minimize the number of errors and impact of errors by identifying which algorithms and UI can be improved or changed.

Background

A group of Google developers created a real-time sign detection application called Real-Time Sign Language Detection using Human Pose Estimation. (Moryossef et al., 2020) The goal

was to create a feature that accurately monitors the user's movement by using human pose estimation. They identified four monitoring methods and tested which method was the most successful at detecting sign language. The four monitoring methods were BBOX, Pose-Hands, Pose-Body, and Pose-All. (Moryossef et al., 2020) They used techniques previously developed by Texas A&M University to detect sign language within pixels. In contrast, the background movement and non-sign language movement were filtered out by separating the foreground and background of the video input. In the end, they decided that Pose-Body was the best method for detecting sign language in their working demo.

TensorFlow.js (TFJS) is a high-level Application Programming Interface (API) that implements both machine learning and deep learning to create an end-to-end platform that makes it possible for ASL detection applications to be built and deployed. ("Real-Time Human Pose," n.d.) As a result, TensorFlow significantly improved the development of ASL detection applications because it eliminates a large portion of the development process.

Methods

The two key goals for our testing were to determine results for accuracy and latency/lag. Looking at examples of sign videos, we hypothesize that the most challenging part of this task is to identify when a person starts signing, because a signer might initiate hand movement for other purposes, for example, to touch their face. Distinguishing this type of ambient motion from actual linguistic sign movement is not always straightforward. Although not explicitly studied on signers, people in different cultures exhibit different face-touching patterns, including frequency, area, and hand preference.

We compared our results to the Google group's results for accuracy and then compared results for latency to the other table and speech detection results. In addition, we will be using

the Google sign language detection application to determine the impact of errors and delays. Our research question is what algorithms or UI can be improved or changed to minimize errors and the impact of errors in current Sign Language Detection applications? To complete the testing, we set up a VM on Google's Cloud platform and the sign detection code.

Results

We recruited 18 deaf signers for a 30-minute evaluation, and participants were compensated a $15 Amazon gift card. We recruited participants using word of mouth, contacting students from our peers, a few from alumni at university through email or social media, and family and friends through word of mouth. Our research question asks: What algorithms or UI can be improved or changed to minimize errors and the impact of errors in current Sign Language Detection applications?

Data Analysis

After the conclusion of our app testing, we had our participants complete a Google Form that asked them questions about their perception of our sign detection application, followed by a demographics section to gain general information about the participants that could pertain to our testing of the sign detection app. When asked to agree on whether the sign detection app was useful? As shown in Figure 1, 11% strongly agreed, 50% agreed, 27% were neutral, and only 11% disagreed.

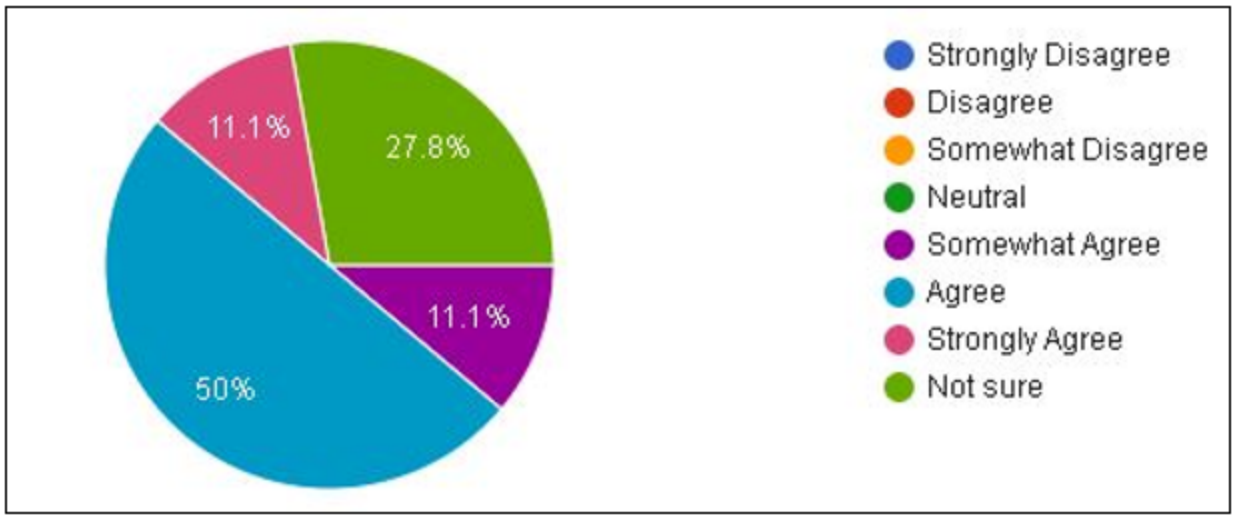

Fig. 1. How Useful is the Sign Detection App?

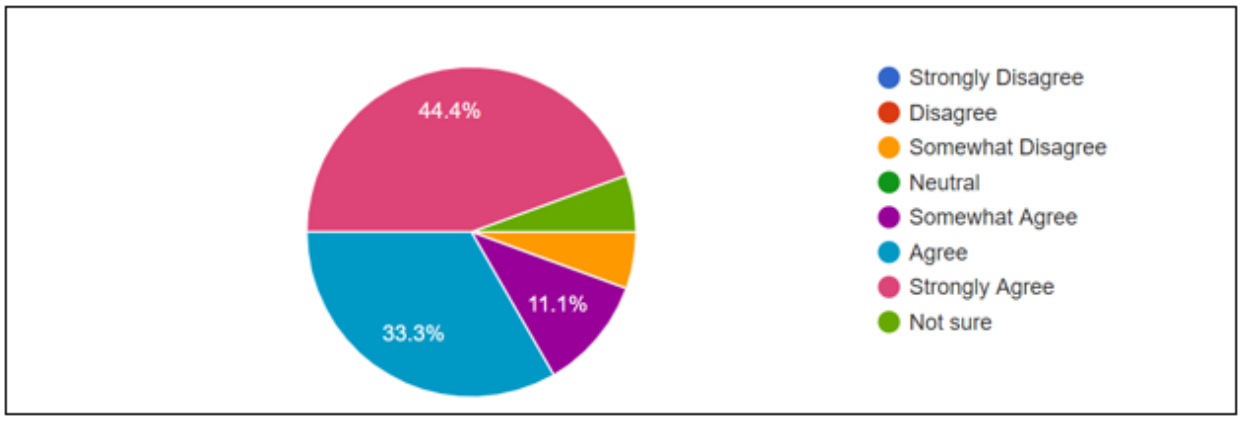

Fig. 2. How Much Do You Agree that The Gestures Will Occur Naturally?

The next question in our Google Form that had notable results was our question asking our participants if they agreed that the gestures, we had them complete would occur naturally when using a video conferencing platform. As shown in Figure 2, almost 90% of the participants either strongly agreed, agreed, or somewhat agreed that these gestures would occur naturally in a video conference platform. Based on the results, we conclude that the gestures we had our participants complete were proper and confirmed our testing procedures.

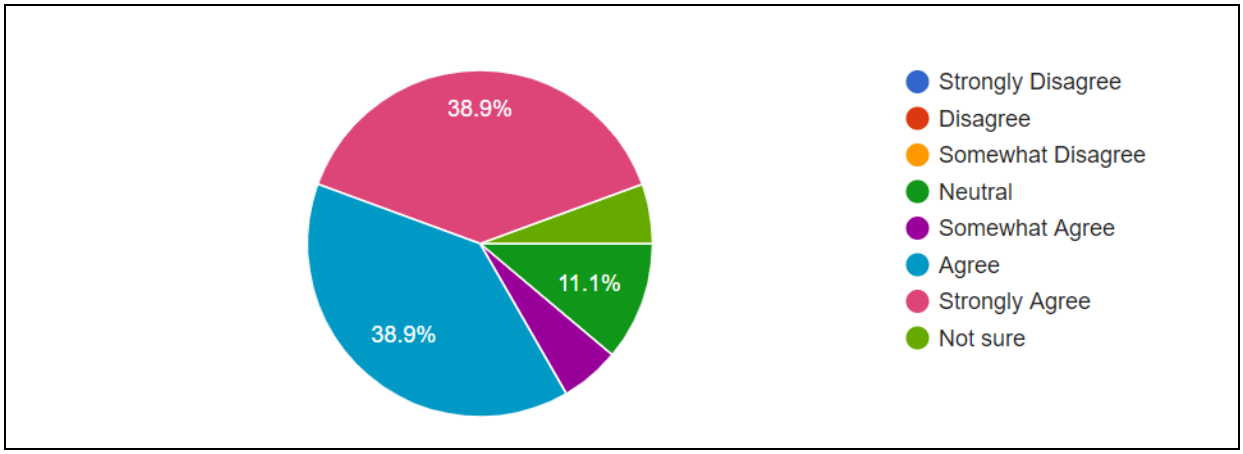

Fig. 3. Do You Agree the Lag Was Noticeable or Not?

Another question that produced significant results was the question asking if the participants agreed or disagreed that they did not experience any lag or latency while using the app demo. As indicated in Figure 3, 83.4% of our participants did not have a noticeable issue with lag or latency.

Due to FPS being different for each participant, we decided to find the averages for each group of FPS ranges. FPS was separated into the following four groups: 1-10 FPS, 11-20 FPS, 21-27, and 28+. Figure 4 below graphs the average for each group for each sign phrase/gesture.

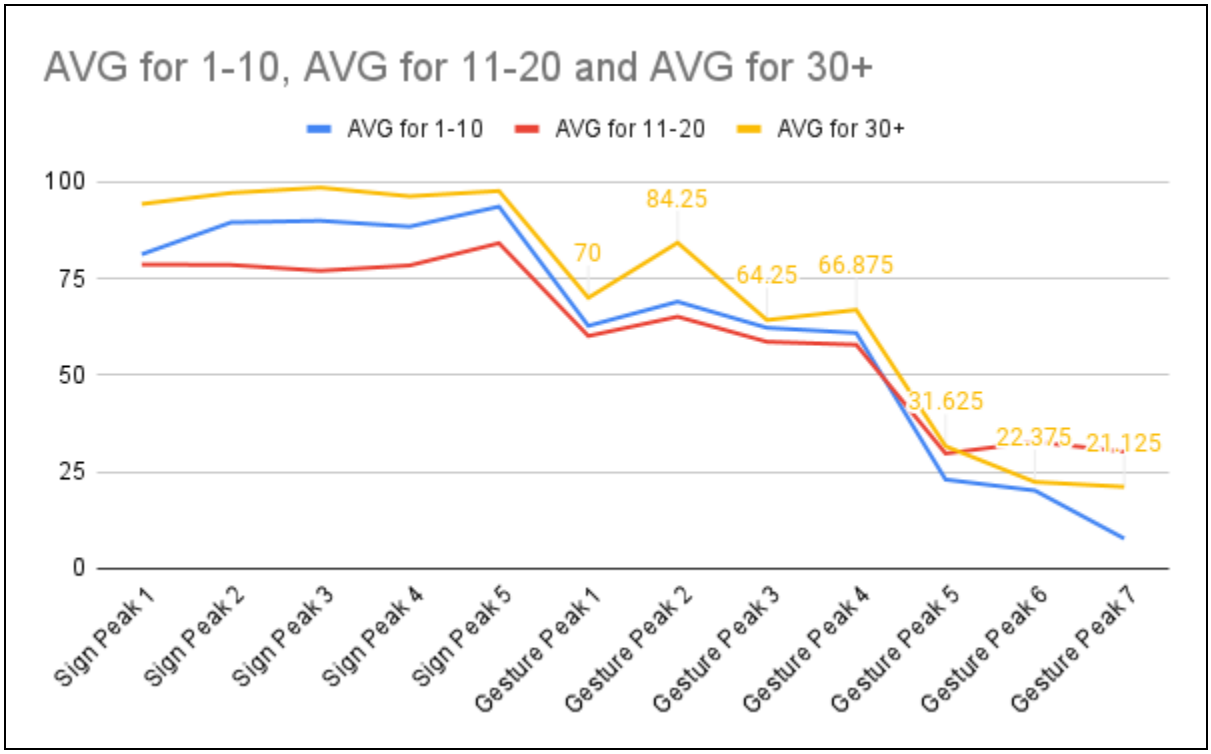

Fig. 4. A Noticeable Drop in Accuracy Below 28 FPS.

From Figure 4 above, it is evident that FPS can make a difference in the output for the users that use it. For example, 28+ FPS will typically result in higher sign detection compared to lower FPS. The main cause for higher FPS outputting a higher peak detection value is that the app detects signing by subtracting movement between frames, so by having more frames processed per second, it will cause a higher output. Likewise, as the FPS decreases, the peak sign detection will be lower compared to higher FPS computations.

Discussion

Due to being online and being unable to control what computers our participants used, cameras capture video at different FPS or at various resolutions. Given that the algorithm requires sufficient computing power to run the pose estimation system in real-time on any user's device. As we found, this proves to be challenging, as noted below that the minimum FPS requirement was not met on several participants' computers. Furthermore, as the algorithm only

looks at the input's optical flow norm, it might not be able to pick up on times when a person is just gesturing rather than signing. The authors noted that since this approach is targeted directly at sign language users rather than, the general non-signing public, erring on the side of caution and detecting any meaningful movements is preferred.

If a camera captures at an FPS below 28 frames per second, then the app will only be able to analyze the number of frames provided. This can make an impact in future sign detection applications because it will cause any improvements to account for the number of frames captured by the camera. Additionally, higher-quality cameras capture video at a higher resolution. This factors into the effectiveness of the app because with higher resolution, the app can more effectively complete body pose estimation and, in theory, would generate more accurate results.

Future Work

We have identified four areas of our app that need to be improved to potentially decrease the number of errors that occur and to improve usability for users. The app should allow for optimization throughout the processing on the client's side of the system. This change will directly benefit users because it will enable the app to process more frames per second and result in fewer errors in detection. Besides optimization, another change in the algorithms would be to recreate the machine language part of the app but with only specific sign languages and not an array of them. Lastly, we noticed that different lighting and background would seem to cause unwanted errors during testing. Therefore, we propose the addition of algorithms to minimize the impact that different lighting and background have on the application.

Conclusion

After completing our data analysis and combining our research, we were able to determine four areas of improvement for current. The proposed changes included changes in our selected app's UI and processing. Our proposed UI change would make our app's output clearer to the user. Our proposed processing changes include optimizing the body pose estimation software, training the app's machine learning process only to use American Sign Language, and adding a type of AI for better performance for different lightings and backgrounds.

Works Cited

Iqbal, M. (2021, March 10). Zoom revenue and usage Statistics (2020). https://www.businessofapps.com/data/zoom-statistics/.

Karappa, V., Monteiro, C. D. D., Shipman, F. M., & Gutierrez-Osuna, R. (2014). Detection of sign-language content in the video through polar motion profiles. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1290–1294. https://doi.org/10.1109/ICASSP.2014.6853805

Monteiro, C. D. D., Mathew, C. M., Gutierrez-Osuna, R., & Shipman, F. (2016). Detecting and Identifying Sign Languages through Visual Features. 2016 IEEE International Symposium on Multimedia (ISM), 287–290. https://doi.org/10.1109/ISM.2016.0063

Monteiro, C. D. D., Shipman, F. M., Duggina, S., & Gutierrez-Osuna, R. (2019). Tradeoffs in the Efficient Detection of Sign Language Content in Video Sharing Sites. ACM Transactions on Accessible Computing, 12(2), 1–16. https://doi.org/10.1145/3325863

Moryossef, A., Tsochantaridis, I., Aharoni, R. Y., Ebling, S., & Narayanan, S. (2020). Real-Time Sign Language Detection using Human Pose Estimation.

Real-time Human Pose Estimation in the Browser with TensorFlow.js. (n.d.). Retrieved May 27, 2021, from https://blog.tensorflow.org/2018/05/real-time-human-pose-estimation-in.html