VuBot - Multimodal User Interfaces

Sophie Caroni[1], Aria David Darmanger[2], and Olivia Lecomte[1]

[1] *University of Fribourg*
[2] *University of Bern*
Swiss Joint Master of Science in Computer Science

31.05.2024

Table of contents

1 Introduction **2**
1.1 Overview . . . 2

2 Architecture and Code **3**
2.1 Architecture . . . 3
2.2 Merging the Modalities . . . 3
2.3 Detailed Features . . . 3
2.3.1 Gesture Recognition . . . 3
2.3.2 Detecting Singular Objects . . . 4
2.3.3 Color Recognition . . . 4
2.3.4 Detecting All Objects . . . 4
2.4 Interaction and Error Management . . . 4
2.5 Justification of Design Choices . . . 5

3 Evaluation **6**
3.1 Experiment . . . 6
3.2 User Preference . . . 6
3.3 Time . . . 6
3.3.1 Task Time . . . 6
3.3.2 Computation Time . . . 7
3.4 Accuracy . . . 7
3.4.1 Task Versions Compared . . . 7
3.4.2 Recognition Model Accuracy . . . 8
3.5 Interpretation of Results . . . 8

4 Discussion **9**
4.1 Limitations . . . 9
4.2 Future Works . . . 9

5 Conclusion **11**

Bibliography **12**

1. Introduction

Recent technological advancements have made way for the development of multimodal systems that assist individuals with sensory and cognitive deficits in navigating their environments. Inspired by the recent research paper "GazePointAR: A Context-Aware Multimodal Voice Assistant for Pronoun Disambiguation in Wearable Augmented Reality" (4), we developed VuBot. VuBot is a visual assistant capable of recognizing objects and colors in response to multimodal interaction. With VuBot, individuals experiencing recognition impairments due to deficits such as associative or color agnosia, would be granted more independence in their daily lives. Additional features, such as querying for object usage, could be developed to aid a wider variety of individuals based on their specific needs.

Our code and technical documentation are publicly available on GitHub[1]. The repository includes all necessary details in the README file, such as a guide on installing the app, use instructions, and a detailed project structure. A demo video and presentation are also included and can be found in the document folder.

1.1 Overview

Interacting with VuBot is easy (Table 1.1). To find the name or color of a single object, simply point to the object within view of the webcam and ask for either the name of the object or the color (Figure 1.1). Once selected, the name of the last queried object or color remains displayed on the screen until the next query is made. To detect all objects in a scene, simply change the hand gesture to a closed fist and ask for the objects. The objects are then bound by boxes with the name of the object displayed above the box. To enhance user confidence while using VuBot, the name of the identified gesture is displayed in the upper left corner, allowing users to modify their position if needed.

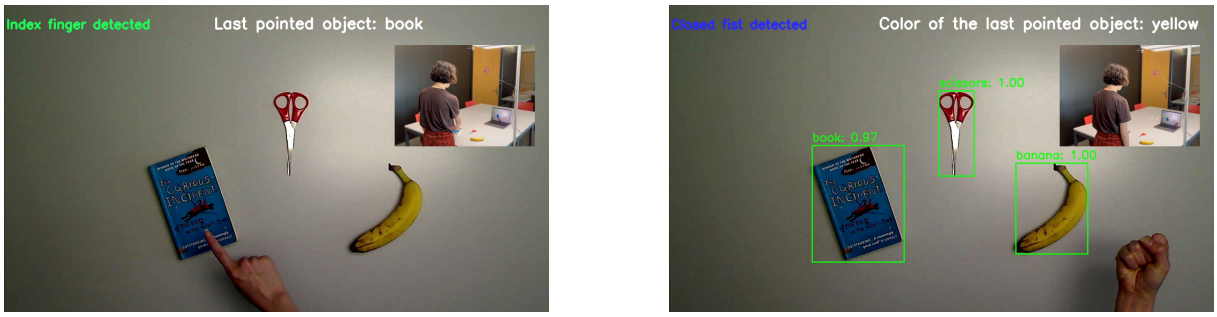

Figure 1.1: Example Gestures

Gesture	Trigger Word	Output
Pointing	'object'	Return object's name
Pointing	'color'	Return object's color
Closed Fist	'every item'	Highlight all detected objects

Table 1.1: Possible Multimodal Interactions

[1] https://github.com/darmangerd/vubot

2. Architecture and Code

2.1 Architecture

VuBot leverages several key technologies and libraries to deliver optimal performance. For gesture recognition, VuBot employs MediaPipe's model (version 0.10.9). This model offers high accuracy and speed, essential for real-time applications.

Live video capture and image processing are completed by OpenCV (version 4.10.0), a library renowned for its powerful functions and performance in handling video and images.

Speech commands are captured and processed by the OpenAI whisper model (small.en, version 20231117). This model is well-known for its proficiency in converting spoken words into text, ensuring reliable speech recognition.

For object detection, VuBot utilizes the facebook/detr-resnet-50 model from Hugging Face (version 4.41.3). This model identifies objects from a predefined list and returns the prediction certainty.

2.2 Merging the Modalities

VuBot employs parallel processing to handle simultaneous gesture and speech recognition managed by Python's threading module. The application operates using two main threads, each handling a different aspect of the system.

The first thread handles video capture and gesture recognition. Each video frame is captured by a webcam using OpenCV and is processed to detect gestures using MediaPipe. When a pointing gesture is detected, it identifies the coordinates of the index finger. This step is unnecessary when performing the closed fist gesture. The second thread is responsible for the speech recognition. It runs the `recognize_speech` method, which continuously listens for trigger words using the Whisper model. The simultaneous detection of a specific gesture and speech command triggers various actions such as object detection or color recognition depending on the speech query. Threading ensures that VuBot remains responsive and efficient while simultaneously processing video frames and speech commands (Figure 2.1).

2.3 Detailed Features

2.3.1 Gesture Recognition

MediaPipe provides machine-learning models to detect and track 21 hand landmarks from video frames captured by the webcam. The `gesture_callback` method processes each frame and checks the positions

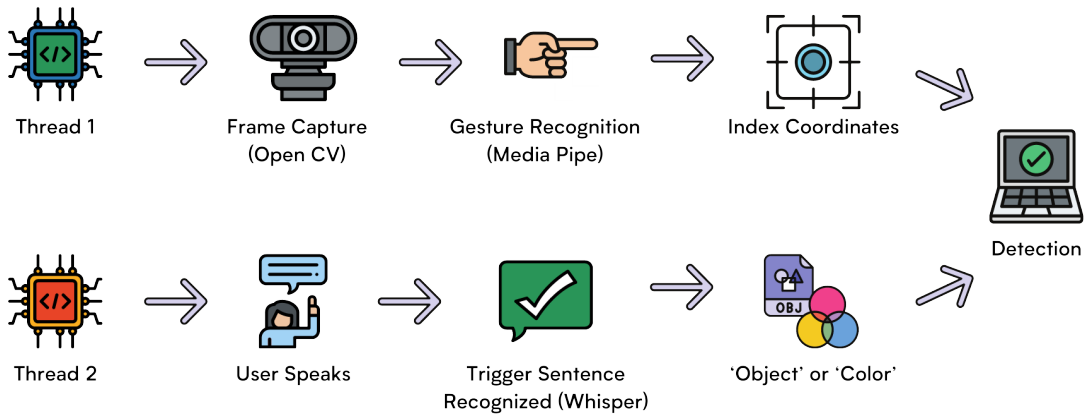

Figure 2.1: Fusion of Modalities

of these landmarks to identify specific gestures. For example, the 'Pointing Up' gesture is detected when the index fingertip is extended while other fingers are balled in a fist. Using the gestural information outputted by MediaPipe, we programmed the app to update its state and trigger corresponding actions in real-time.

2.3.2 Detecting Singular Objects

The pointing gesture is used to determine which object is being selected. When both the pointing gesture and the trigger word are recognized, the application simultaneously captures the coordinates of the index fingertip and scans the video frame for objects. This is achieved using the object detection algorithm that processes the frame to identify and classify all visible objects. The application then verifies if the index fingertip's coordinates are mapped within the bounding box of an identified object. If a match is found, the name of the selected object is returned.

2.3.3 Color Recognition

The color recognition function utilizes the bounding boxes of detected objects. When the user points at an object and queries for the color, the pixel values within the bounding box are averaged returning a single RGB value. This method determines the predominant color of the object, even if the object has multiple shades. To account for differing lighting conditions, the RGB value is converted to HSL. This value is then matched to the closest color from a predefined list of color names.

2.3.4 Detecting All Objects

The "detect all objects" function is triggered by the closed fist gesture. Following the simultaneous detection of this gesture and the trigger word, the algorithm outputs bounding boxes around each detected object along with its label. Bounding boxes appear sequentially at 3-second intervals until all detected objects are bound by boxes.

2.4 Interaction and Error Management

Both speech recognition and hand landmark detection are required to trigger any process. VuBot's reliance on the simultaneous and combined presentation of gesture and speech inputs demonstrates the principle of complementarity within the CARE model and aligns with the synergistic CASE model (Figure 2.2).

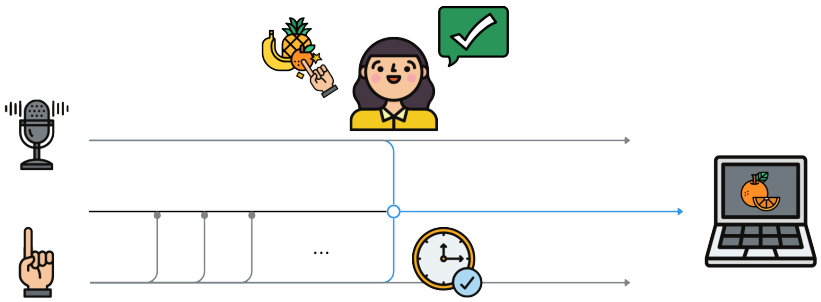

Figure 2.2: Care Case Model

If no hand landmarks are detected, the app continues capturing frames to stay responsive. For speech recognition, VuBot notifies the user when a trigger sentence is heard and keeps listening if no trigger words are detected, ensuring seamless interaction. If the webcam fails to capture a frame, the app exits the main loop and releases resources properly to avoid crashes or resource leaks.

To enhance user confidence, VuBot provides live on-screen visual feedback. This feedback confirms that the user's hand gestures are correctly detected. When the speech recognition model hears a trigger word, such as 'object' or 'color', users receive a message via the console. A second message is printed when the gesture and trigger word are detected simultaneously thus triggering the object or color detection. Finally, the detected output is displayed in the top right corner of the screen until the next query is made.

We have also implemented a debug mode that outputs more information in the terminal. In addition to providing the user with status updates pertaining to the object and color recognition, the debug mode also prints the live audio transcription and saves images for later analysis.

2.5 Justification of Design Choices

Given that our development team included one computer science and two neuroscience students, with varying degrees of programming expertise, we decided to code our project in Python. Python's simple and readable syntax combined with its extensive libraries made it easier for all team members to contribute and understand the code.

That being said, while VuBot is functional and shows the capabilities of gesture and speech recognition, the code could be improved. For instance, the many long if statements could be better handled with switch-case structures or shorter conditional statements. Likewise, the addition of functions to regroup repeated parts of the code would improve readability and maintenance. In its current state, the application lacks a clear separation of functions which could be improved by organizing the features (gesture recognition, speech recognition, and color detection) into separate classes. This modular approach would also help with debugging and future improvements.

The decision to develop the prototype using a webcam instead of a smartphone stemmed from a desire to simplify the development process, allowing everyone on the team to participate effectively. Developing for a webcam is less complex than for a mobile device, which would have required extra knowledge of mobile development frameworks and deployment processes.

In summary, the choice of technology and coding practices, while not perfect, were made to suit the team's varied skill levels and ensure everyone could actively contribute. By using Python and focusing on a webcam-based prototype, we simplified the development process and created a working application that effectively demonstrates the core functionalities.

3. Evaluation

3.1 Experiment

To evaluate the multimodal interaction with VuBot we designed an experiment in which healthy participants were tasked with finding four objects using the model. The objects, including several decoys, were arranged on a desk with a fixed webcam pointing down at the table. To simulate associative and color agnosia, the object and color names were manipulated ensuring that participants would rely on the model to complete the task (Table 3.1). The experiment was conducted twice to vary the input modality between speech or keyboard presses resulting in a within-group analysis.

Object & Color	Manipulated Speech	Manipulated Keys
Red Apple	Pink Ornament	Orange Ball*
Red Scissors	Pink Knife*	Orange Tongs
Black Card Reader	Black Cell Phone	White Calculator
Cyan Book	Green Box	Indigo Tablet*
Grey Book	White Box	White Tablet*
Orange Fork	Red Knife*	Green Pen
Orange Orange	Pink Candle*	Green Soap
Yellow Apple	Orange Ornament	Green Ball*
Yellow Banana	Red Candle*	Green Pencil Case

*task target objects and colors

Table 3.1: Manipulated Objects and Colors

For the speech input version, users were instructed to say 'object' or 'color' to query for the object or color name respectively. For the key press version, they were asked to press 'o' for object and 'c' for color. These spoken commands were combined with a pointing gesture to trigger the requested recognition. The order of the tasks was counterbalanced between the eight participants to control for order effects.

We hypothesized the following:

1. Users will prefer interacting with VuBot using the speech over the keyboard presses because speech is more natural and intuitive.
2. Users will complete the task faster while interacting with the keyboard presses than speech due to keyboard interactions' deterministic nature.
3. The time to compute the object detection will be similar to the time to compute the color detection.
4. The speech version of the task will result in more errors and a lower accuracy than the keyboard version because the speech recognition model is not optimized for live transcription.
5. The gesture, speech, and object recognition models will perform with high accuracy.

3.2 User Preference

The performance of VuBot was assessed both qualitatively and quantitatively. After the experiment, participants were asked which input method they preferred. 8 out of 9 participants indicated that they preferred the keyboard input method.

3.3 Time

3.3.1 Task Time

The task time was measured as the time between the first query and the final output. The task was considered completed when the model identified all four requested objects. Occasionally, the model failed

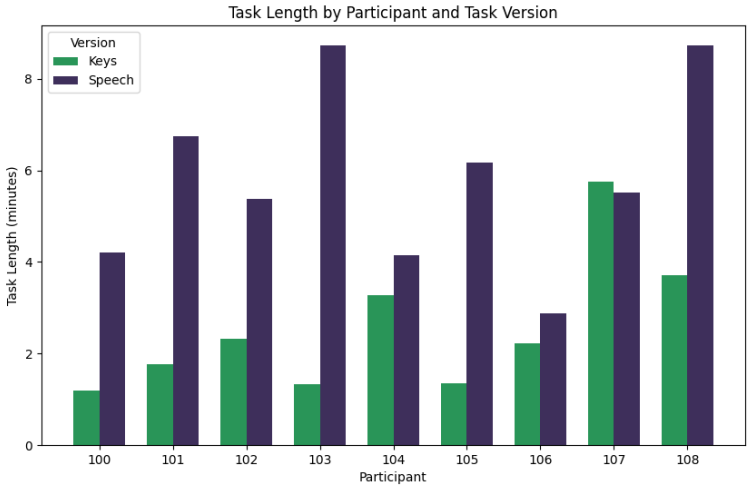

Figure 3.1: Difference in Task Duration Between Keys and Speech Input Modalities

to recognize an object or color. In such cases, the task finished when a sufficient attempt was made to query each object and its color.

Participants completed the task significantly faster when using the keyboard ($M = 2.55$ minutes, $SD = 1.49$) compared to the speech ($M = 5.83$ minutes, $SD = 2.01$) interaction ($t = -3.92, p < 0.001$) (Figure 3.1). Due to model errors, 1 out of 9 participants could not complete the task when querying with keys and 4 out of 9 when querying with speech.

3.3.2 Computation Time

The computational time for each query was measured from when the model detected the simultaneous presence of the gesture and speech or key press until the output was returned. A paired-sample t-test indicated that the time required to compute object detection ($M = 0.66s, SD = 0.09s$) was not significantly different than when computing color detection ($M = 0.68s, SD = 0.08s; t = -1.57, p > 0.05$).

3.4 Accuracy

3.4.1 Task Versions Compared

We hypothesized that querying for objects using speech would be less accurate than when using keyboard presses. To compare the overall accuracy of these two input methods, we evaluated the errors made by the system when responding to participants' queries. We identified six categories of errors: color (color was wrongly identified), object (no object or the incorrect object was identified), color-object (no color was identified because no object was identified), gesture (no gesture was detected despite the participant performing the correct gesture), keys (the recognition model was not triggered despite the participant performing a correct key press), speech (incorrect transcription of correct trigger words). The first four errors could occur in both versions of the task, while the latter were specific to the speech or keys version. We computed accuracy separately for each participant and task by determining the proportion of correctly identified inputs divided by the total number of queries. The accuracy in the speech commands task, when considering all types of errors, was lower ($M = 45.22\%$) and less variable between participants ($SD = 9.55\%$) compared to the accuracy in the keys presses task ($M = 76.69\%, SD = 11.97\%$). The difference in accuracy between the two input methods, evaluated by a dependent sample t-test, was significant ($t = 5.04, p < 0.001$). The same pattern was observed when considering only errors specific to speech and key presses. The accuracy in the speech task was again significantly lower ($M = 75.29\%$) and more variable ($SD = 15.95\%$) than the accuracy in the keys task ($M = 95.43\%, SD = 4.70\%; t = 3.62, p < 0.05$) (Figure 3.2).

There were fewer errors in both the object and color recognition tasks when using the keyboard input, confirming our hypothesis. Key presses outperformed speech commands, likely due to the non-optimization

of the speech model for live transcription and the deterministic nature of the key presses.

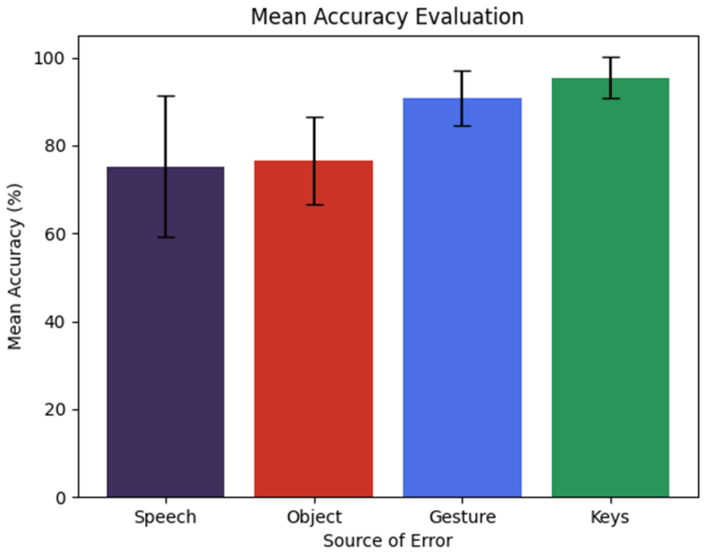

Figure 3.2: Mean Accuracy

3.4.2 Recognition Model Accuracy

We hypothesized that all recognition models would perform with high accuracy. To evaluate this, we filtered the error responses specific to each model separately. Indeed, the gesture recognition model achieved a high accuracy score ($M = 90.83\%, SD = 6.19\%$). We compared the average value to our ideal accuracy, arbitrarily defined at 95%, using a one-sample one-tailed t-test. The accuracy of the gesture recognition model was only marginally lower than our ideal accuracy level ($t = -2.02, p = 0.04$). Similarly, we tested the accuracy performance of the speech recognition model ($M = 75.29\%, SD = 15.95\%$) against our target accuracy of 95%. The speech recognition model's accuracy was significantly lower than our ideal level ($t = -3.71, p = 0.00$). Finally, we evaluated the mean accuracy of the object recognition model. Although we had wanted to set the model's detection threshold to 95%, it was clear that objects would not be reliably detected at this level. As a result, our model was programmed to return the names of objects with the default certainty of 80% or higher. The accuracy of the object recognition model ($M = 76.66\%, SD = 9.96\%$) was not significantly lower than the threshold we set (80%) ($t = -1.01, p > 0.05$), however, when compared to our target benchmark (95%), the mean accuracy differs significantly ($t = -5.53, p < 0.001$) (Figure 3.2).

These results indicate that all three models perform well while highlighting areas for improvement. We observed that both the object and gesture recognition models were significantly influenced by the lighting conditions and the angles of the objects and hands respectively. Specifically, the recognition of the pointing gesture was less likely when the wrist was near the edge of the frame. Additionally, since our participants were non-native English speakers, we noticed difficulties with the model's recognition of our trigger words. This contributed to the increased number of queries and errors leading to longer task times and likely resulting in the near-unanimous preference for the key press version of the task.

3.5 Interpretation of Results

Although the qualitative and quantitative evaluations suggest that the key press input modality is better than the speech modality, we argue that the speech modality remains the better option for further development. Speech is a more natural form of interaction than key presses and allows the user's hands to remain free to complete gestures. Additionally, speech provides a simpler way to scale our project to include more commands with greater complexity. We believe that by improving the model accuracy and reducing errors, the speech input modality will be more intuitive and pleasant to use.

4. Discussion

4.1 Limitations

At this stage, VuBot has several limitations, some of which are inherent to the chosen models.

One major limitation is the need to ignore the "*person*" outpul label during object detection. Since the object detection model often labels the user's hand as a person, frequent false positives arose due to overlapping bounding boxes. By excluding the "*person*" label, we can reduce these false positives. However, this stops VuBot from being able to detect people in the scene.

The color recognition process can also be improved. In the current iteration of this project, color recognition is only possible on objects that have been identified by the object recognition model. This is because the color detection process takes the average RGB value within the bounding boxes created by the object recognition model. This value is thus significantly influenced by the background color for objects that do not fill the bounding boxes due to a non-rectangular shape or simply because of the object's angle. Additionally, the user's hand, which is pointing at the object, must also be within the borders of the bounding box to select each object thus introducing another form of bias. Finally, since only one color is returned, the recognition of multicolored objects could be problematic.

The speech recognition component, while generally effective, is not the best choice for an application such as VuBot. As pointed out by the community on Hugging face (3) this model is not made for real-time transcriptions. A known issue, the Whisper model continuously attempts to transcribe audio even in the absence of speech. To mitigate this issue, we refined the thresholds and audio chunk sizes. In a completely silent room, this approach worked and only slightly impacted our accuracy measures. However, we recommend changing the model to one designed for live transcription to reduce misunderstandings and misinterpretations in future iterations of this project.

4.2 Future Works

There are several directions for future development of VuBot that could significantly enhance its capabilities and user experience.

Firstly, while the groundwork for the precise selection gesture is in place, it still requires some refinement. In future iterations of this project, we envision the possibility of selecting objects by averaging the coordinate point between two fingers. This would allow singular objects to be recognized by the model without requiring the index finger to overlap with the bounding box. We also hypothesize that this would reduce the selection errors of objects in crowded environments.

One major improvement would be developing a mobile version of VuBot. This would allow users to utilize the app on their smartphones, making it more accessible and convenient. A mobile version would require adapting the current functionalities to work with mobile hardware and optimizing the user interface for smaller screens.

Improving the audio speech handling is another crucial area for future work. We believe that adding filters, checking noise levels, and integrating a speaker-dependent system designed for live transcription could improve issues related to non-native English speakers' pronunciation while allowing the app to be used in noisier environments. These enhancements could increase the robustness of Vubot's speech recognition system and provide more accurate and reliable responses.

Expanding how users can interact with the system is also a key area for enrichment. For example, enabling interactions like "*Is this a/an [OBJECT]?*" with yes or no responses would make the app more versatile. This would involve adding more voice command recognition and response capabilities. Furthermore, integrating a Large Language Model (LLM) could significantly enhance VuBot. An LLM would enable more complex and natural interactions, allowing users to ask a wider range of questions. Combining this with a text-to-speech system would make VuBot more interactive and intuitive. This would transform VuBot into a comprehensive multimodal assistant, capable of understanding and responding to user queries conversationally. For example, when an object is recognized, a user could ask, "What can it be used for?".

This would have a significant impact on the lives of people of individuals living with agnosia and many other cognitive impairments.

Another advanced feature to consider is the ability to save specific objects in memory. Users could train VuBot to recognize personal items like their wallets or keys by providing pictures of these items from multiple angles. This would involve fine-tuning the object detection model to recognize these specific items accurately. Users could then ask VuBot to find these objects, making it a practical and personalized tool for everyday use.

5. Conclusion

VuBot is an early-stage visual assistant prototype with the potential to help clinical populations gain more independence in their day-to-day lives. Through the complimentary and synergistic fusion of gesture and speech modalities, it allows users to quickly and reliably recognize objects and their colors.

The evaluation of the prototype against a version which used key presses and pointing to query for objects and colors led to an unexpected result. Contrary to our hypothesis, most users preferred to interact with VuBot via key presses as opposed to speech. This preference likely stems from a high volume of errors related to misinterpretations made by the speech model, which is not designed for live transcription.

Despite the numerous limitations arising from the chosen models and the sub-optimal performance of the speech modality in our evaluation, the speech modality remains the best option for future development. The combination of speech and gesture modalities leads to naturalistic and intuitive interactions that scale with the growing complexity of VuBot. Additionally, these modalities allow users' hands to remain free for gestures, a critical feature for a potential mobile application. With improvements to model accuracy and error reduction, we anticipate that the user experience of the speech modality will increase, ultimately making it the preferred choice for users.

By addressing these challenges and implementing more ways to interact, VuBot has the potential to become an effective and user-friendly visual assistant, empowering users with greater independence in their daily lives.

Bibliography

[1] Google AI. Gesture recognition task guide. URL https://ai.google.dev/edge/mediapipe/solutions/vision/gesture_recognizer.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020. URL https://arxiv.org/abs/2005.12872.

[3] Hugging face's community, 2024. URL https://huggingface.co/openai/whisper-large-v3/discussions. [Accessed 14-05-2024].

[4] Jaewook Lee, Jun Wang, Elizabeth Brown, Liam Chu, Sebastian S. Rodriguez, and Jon E. Froehlich. Gazepointar: A context-aware multimodal voice assistant for pronoun disambiguation in wearable augmented reality, 2024.

[5] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022. URL https://arxiv.org/abs/2212.04356.