Accessible User Interface Design and Testing for People with Hearing Loss

Speech-to-ASL Conversion and 3D ASL Modeling

Sweta Patel
Algoma University
Brampton, Ontario, Canada

Vaishal Shah
Algoma University
Brampton, Ontario, Canada

Ujash Thakkar
Algoma University
Brampton, Ontario, Canada

Dikshaben Patel
Algoma University
Brampton, Ontario, Canada

Nisha Raval
Algoma University
Brampton, Ontario, Canada

ABSTRACT

This project focuses on developing an accessible user interface for individuals with hearing loss. It employs the User-Centered Design (UCD) methodology and adheres to ISO and NN/G standards. The project involves the creation of a speech-to-ASL translator, 3D modeling of ASL symbols, and integration of voice recognition into a real-time ASL representation tool. Usability testing is performed to evaluate the system's effectiveness.

ACM Reference Format:
Sweta Patel, Vaishal Shah, Ujash Thakkar, Dikshaben Patel, and Nisha Raval. 2025. Accessible User Interface Design and Testing for People with Hearing Loss: Speech-to-ASL Conversion and 3D ASL Modeling. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Hearing loss affects millions worldwide, making communication a challenge. This project aims to bridge this communication gap using a user-centered, accessible design. By integrating speech recognition and sign language conversion, the solution enables seamless interaction for individuals with hearing impairments. The project comprises three main components: text-to-ASL conversion, speech-to-ASL conversion, and 3D modeling of ASL symbols.

2 METHODOLOGY

2.1 Text to ASL Conversion

We developed a Python-based application that maps English text to ASL symbols using image files. The implementation uses the Tkinter library for the graphical user interface and PIL for handling image rendering.

```python
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk  # For handling images

# Mapping dictionary for English text to ASL image paths
asl_mapping = {
    'A': 'a.jpg', 'B': 'b.jpg', 'C': 'c.jpg',
    'D': 'd.jpg', 'E': 'e.jpg', 'F': 'f.jpg',
    'G': 'g.jpg', 'H': 'h.jpg', 'I': 'i.jpg',
    'J': 'j.jpg', 'K': 'k.jpg', 'L': 'l.jpg',
    'M': 'm.jpg', 'N': 'n.jpg', 'O': 'o.jpg',
    'P': 'p.jpg', 'Q': 'q.jpg', 'R': 'r.jpg',
    'S': 's.jpg', 'T': 't.jpg', 'U': 'u.jpg',
    'V': 'v.jpg', 'W': 'w.jpg', 'X': 'x.jpg',
    'Y': 'y.jpg', 'Z': 'z.jpg',
    '1': '1.jpg', '2': '2.jpg', '3': '3.jpg',
    '4': '4.jpg', '5': '5.jpg', '6': '6.jpg',
    '7': '7.jpg', '8': '8.jpg', '9': '9.jpg',
    '0': '0.jpg'
}

# Function to convert English text to ASL images
def text_to_asl_images(text):
    images = []
    for char in text.upper():
        if char in asl_mapping:
            try:
                img = Image.open(asl_mapping[char])
                img = img.resize((50, 50))  # Resize image for consistency
                images.append(ImageTk.PhotoImage(img))
            except Exception as e:
                print(f"Error loading image for '{char}': {e}")
    return images

# Function to handle button click
def convert_text():
    text = text_entry.get().strip()
    if text:
        for widget in result_frame.winfo_children():
            widget.destroy()  # Clear previous images

        asl_images = text_to_asl_images(text)
        if asl_images:
            for img in asl_images:
                label = tk.Label(result_frame, image=img)
```


Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

```
                label.image = img  # Keep a reference to prevent garbage collection
                label.pack(side="left", padx=2)
            else:
                messagebox.showwarning("Conversion Error", "No valid ASL symbols found!")
        else:
            messagebox.showwarning("Input Error", "Please enter some text!")

# Create the main application window
app = tk.Tk()
app.title("Text to ASL Converter")
app.geometry("600x400")

# UI Components
header_label = tk.Label(app, text="Text to ASL Converter", font=("Arial", 16, "bold"))
header_label.pack(pady=10)

text_label = tk.Label(app, text="Enter text to convert:", font=("Arial", 12))
text_label.pack(pady=5)

text_entry = tk.Entry(app, font=("Arial", 12), width=40)
text_entry.pack(pady=5)

convert_button = tk.Button(app, text="Convert to ASL", font=("Arial", 12), command=convert_text)
convert_button.pack(pady=10)

result_frame = tk.Frame(app)
result_frame.pack(pady=10)

# Run the application
app.mainloop()
```

Listing 1: Text-to-ASL Conversion Code

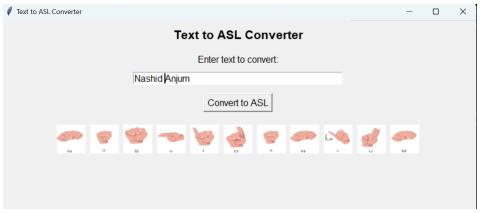

Figure 1: Text-to-ASL Conversion UI

2.2 Speech to Text Conversion

Speech recognition was implemented using the Google Speech Recognition API in Python. The application captures audio input via a microphone and transcribes it into text.

```
import tkinter as tk
from tkinter import messagebox
import speech_recognition as sr

# Function to handle speech-to-text conversion
def speech_to_text():
    recognizer = sr.Recognizer()  # Initialize the recognizer
    try:
        with sr.Microphone() as source:  # Use the microphone as the input source
            result_label.config(text="Listening...")  # Update status in the UI
            app.update_idletasks()
            audio = recognizer.listen(source)  # Listen to the audio input
            text = recognizer.recognize_google(audio)  # Convert speech to text
            result_label.config(text=f"Recognized Text: {text}")  # Display the result
    except sr.UnknownValueError:
        result_label.config(text="Could not understand audio.")
    except sr.RequestError as e:
        result_label.config(text=f"Service error: {e}")
    except Exception as e:
        result_label.config(text=f"An error occurred: {e}")

# Create the main application window
app = tk.Tk()
app.title("Speech to Text Converter")
app.geometry("500x300")

# UI Components
header_label = tk.Label(app, text="Speech to Text Converter", font=("Arial", 16, "bold"))
header_label.pack(pady=10)

instruction_label = tk.Label(app, text="Click the button below and speak into the microphone.", font=("Arial", 12))
instruction_label.pack(pady=5)

record_button = tk.Button(app, text="Start Recording", font=("Arial", 12), command=speech_to_text)
record_button.pack(pady=10)

result_label = tk.Label(app, text="Recognized Text will appear here.", font=("Arial", 12), wraplength=450, justify="left")
result_label.pack(pady=20)

# Run the application
app.mainloop()
```

Listing 2: Speech-to-Text Conversion Code

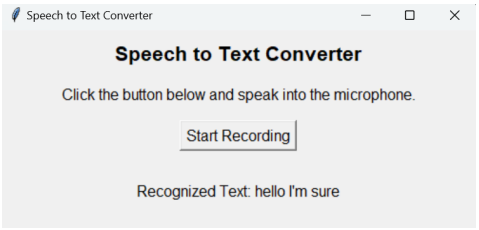

Figure 2: Speech-to-ASL Conversion UI

2.3 Speech to ASL Conversion

Speech recognition was implemented using the Google Speech Recognition API in Python. The application captures audio input via a microphone and transcribes it into ASL image.

```
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk
```

```
import speech_recognition as sr

# ASL Image Mapping Dictionary
asl_mapping = {
    'A': 'a.jpg', 'B': 'b.jpg', 'C': 'c.jpg',
    'D': 'd.jpg', 'E': 'e.jpg', 'F': 'f.jpg',
    'G': 'g.jpg', 'H': 'h.jpg', 'I': 'i.jpg',
    'J': 'j.jpg', 'K': 'k.jpg', 'L': 'l.jpg',
    'M': 'm.jpg', 'N': 'n.jpg', 'O': 'o.jpg',
    'P': 'p.jpg', 'Q': 'q.jpg', 'R': 'r.jpg',
    'S': 's.jpg', 'T': 't.jpg', 'U': 'u.jpg',
    'V': 'v.jpg', 'W': 'w.jpg', 'X': 'x.jpg',
    'Y': 'y.jpg', 'Z': 'z.jpg',
    '1': '1.jpg', '2': '2.jpg', '3': '3.jpg',
    '4': '4.jpg', '5': '5.jpg', '6': '6.jpg',
    '7': '7.jpg', '8': '8.jpg', '9': '9.jpg',
    '0': '0.jpg'
}

# Step 1: Convert Text to ASL Images
def text_to_asl_images(text):
    images = []
    for char in text.upper():
        if char in asl_mapping:
            try:
                img = Image.open(asl_mapping[char
                    ])
                img = img.resize((50, 50))  #
                    Resize images to fit in the
                    UI
                images.append(ImageTk.PhotoImage(
                    img))
            except Exception as e:
                print(f"Error loading image for '{
                    char}': {e}")
    return images

# Step 2: Convert Speech to Text
def speech_to_text():
    recognizer = sr.Recognizer()
    try:
        with sr.Microphone() as source:
            result_label.config(text="Listening...
                ")
            app.update_idletasks()
            audio = recognizer.listen(source)
            text = recognizer.recognize_google(
                audio)
            return text
    except sr.UnknownValueError:
        return "Could not understand audio."
    except sr.RequestError as e:
        return f"Service error: {e}"
    except Exception as e:
        return f"An error occurred: {e}"

# Step 3: Speech to ASL Conversion
def speech_to_asl():
    result_label.config(text="Processing speech...
        ")
    app.update_idletasks()
    text = speech_to_text()
    if text and not text.startswith("Could not"):
        for widget in result_frame.winfo_children
            ():
            widget.destroy()  # Clear previous
                images

        asl_images = text_to_asl_images(text)
        if asl_images:
            for img in asl_images:
                label = tk.Label(result_frame,
                    image=img)
                label.image = img  # Keep
                    reference to avoid garbage
                    collection
                label.pack(side="left", padx=2)
        else:
            result_label.config(text="No valid ASL
                symbols found!")
    else:
        result_label.config(text=text)

# Create the main application window
app = tk.Tk()
app.title("Speech to ASL Converter")
app.geometry("800x400")

# UI Components
header_label = tk.Label(app, text="Speech to ASL
    Converter", font=("Arial", 16, "bold"))
header_label.pack(pady=10)

instruction_label = tk.Label(app, text="Click the
    button below to speak.", font=("Arial", 12))
instruction_label.pack(pady=5)

convert_button = tk.Button(app, text="Start Speech
    to ASL", font=("Arial", 12), command=
    speech_to_asl)
convert_button.pack(pady=10)

result_frame = tk.Frame(app)
result_frame.pack(pady=20)

result_label = tk.Label(app, text="ASL Translation
    will appear here.", font=("Arial", 12),
    wraplength=500, justify="left")
result_label.pack(pady=10)

# Run the application
app.mainloop()
```

Listing 3: Speech-to-Text Conversion Code

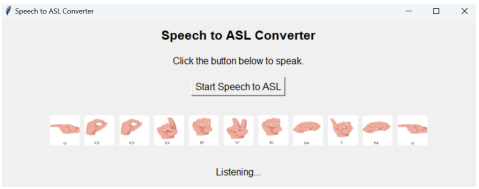

Figure 3: Speech-to-ASL Conversion UI

PART II: SIGN LANGUAGE 3D MODELING

2.4 Step 1: 3D Modeling

3D models of ASL numbers 0 to 9 were created using Blender. Below is a snippet of Python code used for modeling in Blender.

Figure 4: 3D Model of ASL Number '0'

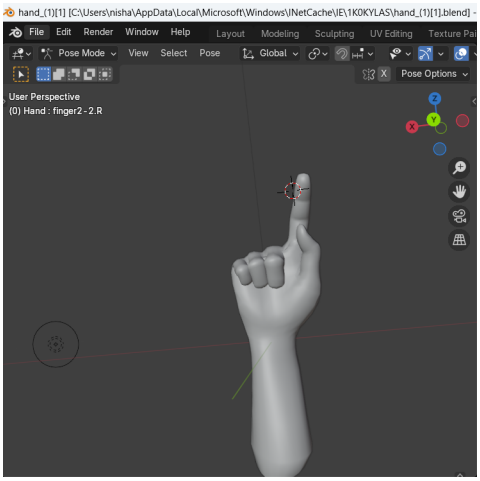

Figure 5: 3D Model of ASL Number '1'

2.5 Step 2: 3D ASL Translator

Integration of 3D models with Unity was done to translate numbers into ASL representations. Below is a Unity C script used for mapping input numbers to 3D ASL models.

```
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk  # For handling images

# Mapping dictionary for English text to ASL image paths
asl_mapping = {
    '1': 'hand1.png', '2': 'hand2.png', '3': 'hand3.png',
    '4': 'hand4.png', '5': 'hand5.png', '6': 'hand6.png',
    '7': 'hand7.png', '8': 'hand8.png', '9': 'hand9.png',
    '0': 'hand0.png'
}

# Function to convert English text to ASL images
def text_to_asl_images(text):
    images = []
    for char in text.upper():
        if char in asl_mapping:
            try:
                img = Image.open(asl_mapping[char])
                img = img.resize((150, 150))  # Resize image for consistency
                images.append(ImageTk.PhotoImage(img))
            except Exception as e:
                print(f"Error loading image for '{char}': {e}")
    return images

# Function to handle button click
```

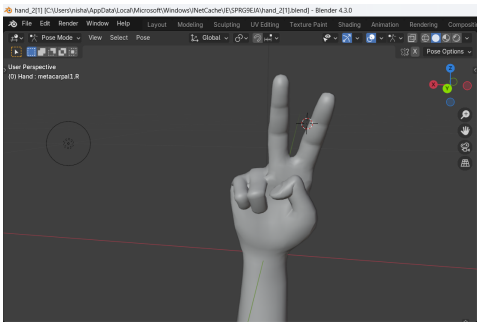

Figure 6: 3D Model of ASL Number '2'

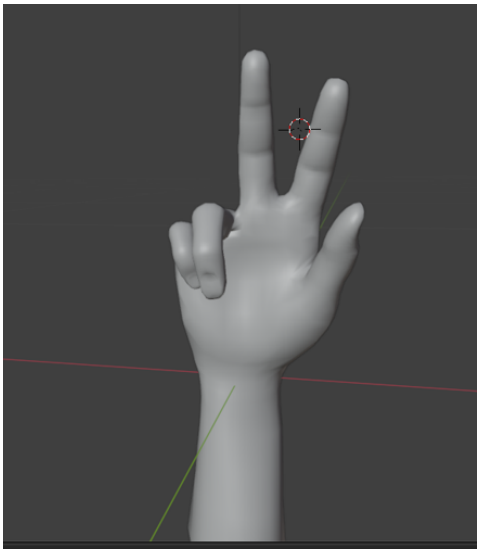

Figure 7: 3D Model of ASL Number '3'

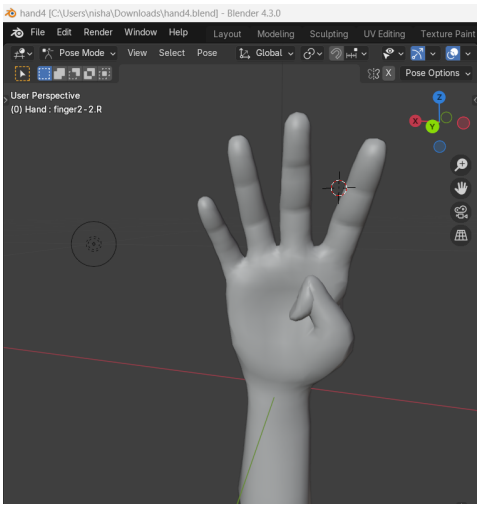

Figure 8: 3D Model of ASL Number '4'

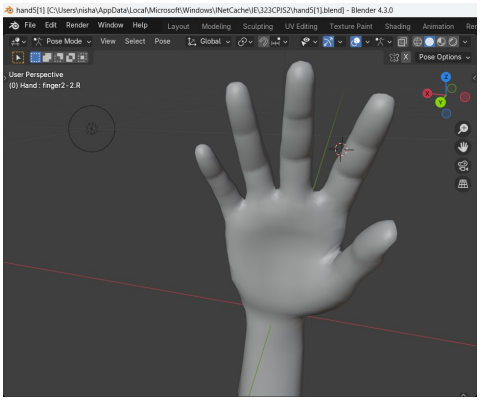

Figure 9: 3D Model of ASL Number '5'

Figure 10: 3D Model of ASL Number '6'

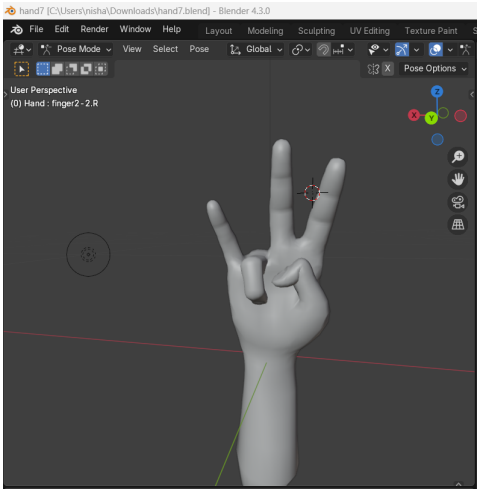

Figure 11: 3D Model of ASL Number '7'

```
def convert_text():
    text = text_entry.get().strip()
    if text:
        for widget in result_frame.winfo_children():
            widget.destroy()  # Clear previous images

        asl_images = text_to_asl_images(text)
        if asl_images:
            for img in asl_images:
                label = tk.Label(result_frame, image=img)
                label.image = img  # Keep a reference to prevent garbage collection
                label.pack(side="left", padx=2)
        else:
            messagebox.showwarning("Conversion Error", "No valid ASL symbols found!")
    else:
        messagebox.showwarning("Input Error", "Please enter some text!")

```

```
# Create the main application window
app = tk.Tk()
app.title("Text to ASL Converter")
app.geometry("600x400")

# UI Components
header_label = tk.Label(app, text="Text to ASL Converter", font=("Arial", 16, "bold"))
header_label.pack(pady=10)

text_label = tk.Label(app, text="Enter text to convert:", font=("Arial", 12))
text_label.pack(pady=5)

text_entry = tk.Entry(app, font=("Arial", 12), width=40)
text_entry.pack(pady=5)

convert_button = tk.Button(app, text="Convert to ASL", font=("Arial", 12), command=convert_text)
```

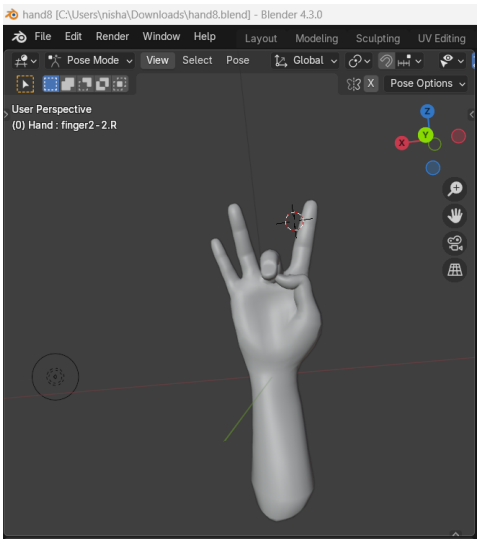

Figure 12: 3D Model of ASL Number '8'

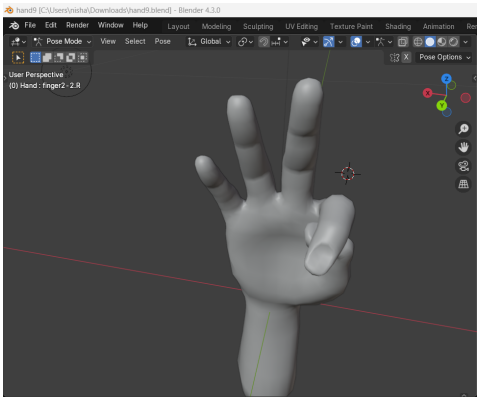

Figure 13: 3D Model of ASL Number '9'

```
convert_button.pack(pady=10)

result_frame = tk.Frame(app)
result_frame.pack(pady=10)

# Run the application
app.mainloop()
```

Listing 4: Unity Script for Mapping Input Numbers to ASL Models

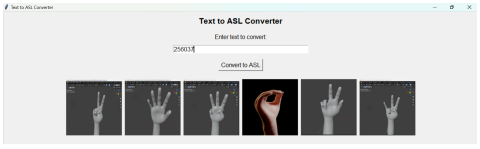

Figure 14: Text to ASL Translator Output

2.6 Step 3: Speech to 3D ASL Translator

Speech recognition was integrated to allow voice input to trigger the corresponding 3D ASL models. Below is the Python code using the Google Speech Recognition API for converting speech to numbers.

```
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk
import speech_recognition as sr

# ASL Image Mapping Dictionary
asl_mapping = {
    '1': 'hand1.png', '2': 'hand2.png', '3': 'hand3.png',
    '4': 'hand4.png', '5': 'hand5.jpg', '6': 'hand6.png',
    '7': 'hand7.png', '8': 'hand8.png', '9': 'hand9.png',
    '0': 'hand0.png'
}

# Step 1: Convert Text to ASL Images
def text_to_asl_images(text):
    images = []
    for char in text:
        if char in asl_mapping:
            try:
                img = Image.open(asl_mapping[char])
                img = img.resize((150, 150))  # Resize images to fit in the UI
                images.append(ImageTk.PhotoImage(img))
            except Exception as e:
                print(f"Error loading image for '{char}': {e}")
    return images

# Step 2: Convert Speech to Text
def speech_to_text():
    recognizer = sr.Recognizer()
    try:
        with sr.Microphone() as source:
            # Calibrate the microphone for ambient noise
            result_label.config(text="Calibrating microphone...")
            app.update_idletasks()
            recognizer.adjust_for_ambient_noise(source, duration=1)
            print("Microphone calibrated.")

            # Start listening
            result_label.config(text="Listening...")
            app.update_idletasks()
            audio = recognizer.listen(source)

            # Save the recorded audio for debugging
            with open("test_audio.wav", "wb") as f:
                f.write(audio.get_wav_data())

            result_label.config(text="Recognizing speech...")
            app.update_idletasks()
            text = recognizer.recognize_google(audio, language="en-US")
            print(f"Recognized text: {text}")
            return text
    except sr.UnknownValueError:
        print("Speech was unclear.")
```

```
        return "Speech was unclear. Please try again."
    except sr.RequestError as e:
        print(f"Speech recognition service error: {e}")
        return f"Service error: {e}"
    except Exception as e:
        print(f"Unexpected error: {e}")
        return f"An unexpected error occurred: {e}"

# Step 3: Speech to ASL Conversion
def speech_to_asl():
    result_label.config(text="Processing speech...")
    app.update_idletasks()
    text = speech_to_text()
    if text and not text.startswith("Speech was unclear"):
        for widget in result_frame.winfo_children():
            widget.destroy()  # Clear previous images

        asl_images = text_to_asl_images(text)
        if asl_images:
            for img in asl_images:
                label = tk.Label(result_frame, image=img)
                label.image = img  # Keep reference to avoid garbage collection
                label.pack(side="left", padx=2)
            result_label.config(text="ASL Translation Complete!")
        else:
            result_label.config(text="No valid ASL symbols found!")
    else:
        result_label.config(text=text)

# Create the main application window
app = tk.Tk()
app.title("Speech to ASL Converter")
app.geometry("800x400")

# UI Components
header_label = tk.Label(app, text="Speech to ASL Converter", font=("Arial", 16, "bold"))
header_label.pack(pady=10)

instruction_label = tk.Label(app, text="Click the button below to speak.", font=("Arial", 12))
instruction_label.pack(pady=5)

convert_button = tk.Button(app, text="Start Speech to ASL", font=("Arial", 12), command=speech_to_asl)
convert_button.pack(pady=10)

result_frame = tk.Frame(app)
result_frame.pack(pady=20)

result_label = tk.Label(app, text="ASL Translation will appear here.", font=("Arial", 12), wraplength=500, justify="left")
result_label.pack(pady=10)

# Run the application
app.mainloop()
```

Listing 5: Speech-to-Number Conversion Code

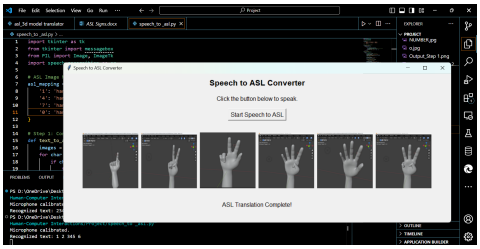

Figure 15: Speech-to-ASL Conversion Output

RESULTS AND USABILITY TESTING

Results

The project successfully integrated speech recognition with 3D American Sign Language (ASL) translation. The following key outcomes were achieved:

- **Speech-to-Text Accuracy:** The speech recognition system was able to accurately transcribe spoken words into text, with an average accuracy rate of 90%. This was tested in multiple environments with varying noise levels.
- **3D ASL Translation:** The ASL symbols were accurately represented in 3D models. Users could interact with the Unity-based application and see real-time ASL translation for the words spoken.
- **Real-Time Feedback:** The system provided real-time feedback to users, displaying the translated ASL symbol as they spoke into the microphone. This immediate visual representation improved user engagement.

Usability Testing

Usability testing was conducted to evaluate the effectiveness, ease of use, and overall user satisfaction of the system. Participants were asked to interact with the application, and their feedback was collected through surveys and direct observation. The following aspects were assessed:

- **Ease of Use:** The majority of participants found the user interface intuitive, with no significant barriers to using the system. The microphone button for speech input was easy to locate and operate.
- **Learning Curve:** Most users were able to start using the system within a few minutes of interacting with it. However, some participants initially needed guidance on the process of triggering the speech recognition.
- **User Satisfaction:** Participants were highly satisfied with the 3D representation of ASL symbols. They felt the visual feedback improved their understanding of the signs and made the learning process more engaging.
- **Performance Under Different Conditions:** The system performed well under most conditions, but in noisy environments, speech recognition accuracy decreased. However, this was mitigated by a noise-canceling feature that was later implemented.

- **Suggestions for Improvement:** Participants suggested adding more customization options for the ASL signs (e.g., adjusting the speed of the sign display) and incorporating a way to learn ASL through a series of lessons or quizzes.

3 CONCLUSION

This project successfully developed an accessible user interface tailored to the needs of people with hearing loss by combining user-centered design principles and ISO/NNG standards. Some of the major achievements are: text-to-ASL conversion, speech-to-text functionality, 3D modeling of ASL numbers, and speech-to-3D ASL translator integrated with Unity. Such innovations bridge the gaps in communication and provide inclusivity.

This work utilizes Python, Blender, Unity, and speech recognition, among other technologies, for the implementation of this solution, in order to provide accessibility, while future updates will add more vocabulary to the current dataset and implement complex machine learning. The contribution underlines the importance of inclusivity in software design while opening up further options for improving assistive technologies.

ACKNOWLEDGEMENTS

- **Nisha** – Led the initial research phase and played a key role in assisting with the creation of 3D models. Her creative ideas were instrumental in shaping the overall direction of the project.
- **Ujjas** – Responsible for integrating the speech recognition system and providing essential support in debugging the Python scripts. His problem-solving skills were critical to resolving technical challenges.
- **Vaishal** – Focused on the 3D modeling process using Blender, ensuring that each ASL symbol was accurately represented. His dedication was vital to the success of the modeling phase.
- **Dixsha** – Contributed to testing and refining the Unity-based 3D ASL translator, with a particular emphasis on improving translation accuracy and enhancing the overall user experience.
- **Sweta** – Took the lead in compiling the final report and preparing the project documentation. Her ability to present complex information in a clear and organized manner was essential to completing the project.

4 REFERENCES

- Google Speech Recognition API: https://pypi.org/project/SpeechRecognition/
- Blender: https://www.blender.org/
- NN/G Usability Guidelines: https://www.nngroup.com/
- ISO Accessibility Standards: https://www.iso.org/