UNIVERSIDAD VERACRUZANA

Facultad de Negocios y Tecnologías

Campus Ixtaczoquitlán

REPORTE

TECLADO VIRTUAL

Programa Educativo:

Tecnologías de la Información en las Organizaciones

Experiencia educativa:

Inteligencia Artificial

Académico:

Jesús Leonardo López Hernández

Estudiante:

Estefanía Oaxaca Rosas

Semestre:

Sexto

Contenido

TABLA DE ILUSTRACIONES

PROBLEMÁTICA

Muchas personas con movilidad reducida, ya sea a causa de discapacidades motrices parciales, enfermedades neuromusculares como la esclerosis lateral amiotrófica (ELA), parálisis cerebral, lesiones medulares, accidentes cerebrovasculares o incluso condiciones temporales como fracturas o cirugías, enfrentan serias dificultades para interactuar con dispositivos electrónicos mediante interfaces tradicionales como teclados físicos, ratones o pantallas táctiles. Esta limitación no solo restringe su acceso a la tecnología, sino que compromete su derecho a la comunicación, la educación, la vida laboral y la inclusión digital.

Aunque en el mercado existen soluciones de asistencia como teclados adaptados, pulsadores, sistemas de seguimiento ocular o control por voz, muchos de estos dispositivos son costosos, requieren hardware especializado o condiciones específicas del entorno para funcionar correctamente. Por ejemplo, el control por voz depende del reconocimiento claro del habla y un entorno silencioso, lo cual no siempre es posible en hospitales, fábricas o espacios compartidos. Asimismo, el seguimiento ocular puede ser poco preciso si no se cuenta con sensores dedicados de alta gama, lo que dificulta su uso masivo y accesible.

El desafío tecnológico consiste en desarrollar un sistema capaz de interpretar correctamente los gestos del dedo en tiempo real, sin errores, y que permita escribir texto de forma fluida y eficiente, marcando una diferencia significativa en la vida de muchas personas, mejorando su autonomía, autoestima y participación en la sociedad digital contemporánea.

JUSTIFICACIÓN

El desarrollo de un teclado virtual basado en la detección del dedo índice mediante visión por computadora responde a la necesidad de ofrecer una herramienta de comunicación más accesible para personas con movilidad limitada. Al utilizar una cámara común, como la de una laptop, y software que interpreta el movimiento del dedo, se puede generar una solución económica, sin contacto físico, fácil de usar y de implementar.

Este sistema tiene un fuerte potencial en el ámbito de la inclusión para personas con discapacidad motriz, particularmente aquellas afectadas por enfermedades neurovasculares como accidentes cerebrovasculares, isquemias o parálisis parcial, quienes a menudo conservan movilidad en una mano o en un dedo, pero presentan dificultades para usar dispositivos convencionales. En estos casos, el teclado virtual puede representar una vía efectiva de comunicación, especialmente cuando también existe afectación del habla (afasia), sin comprometer la capacidad cognitiva del usuario.

Además, la propuesta resulta útil en contextos donde el uso de teclados físicos no es viable o higiénicamente recomendable, como laboratorios, hospitales o industrias alimentarias. Al eliminar la necesidad de contacto físico, se reducen riesgos de contaminación y se mejora la ergonomía en tareas repetitivas.

De esta manera, el proyecto no solo promueve la autonomía personal y la equidad tecnológica, sino que también abre el camino a nuevas formas de interacción humano-computadora más seguras, inclusivas y universales, adaptables tanto a necesidades especiales como a condiciones profesionales exigentes.

OBJETIVO GENERAL:

Desarrollar un prototipo accesible, preciso y funcional de un teclado virtual que permita la entrada de texto mediante una interfaz visual controlada por el movimiento del dedo índice, utilizando técnicas de visión por computadora.

OBJETIVOS ESPECÍFICOS:

1. Aplicar los fundamentos de la visión por computadora para detectar y rastrear en tiempo real el movimiento del dedo índice del usuario.
2. Diseñar una interfaz de teclado virtual interactiva que permita la selección de caracteres a través del movimiento del dedo índice.
3. Incorporar un modelo de aprendizaje automático basado en redes neuronales (LSTM, RNN o GRU) para la predicción y autocompletado de palabras, mejorando la eficiencia en la escritura.

ARQUITECTURA DEL SISTEMA

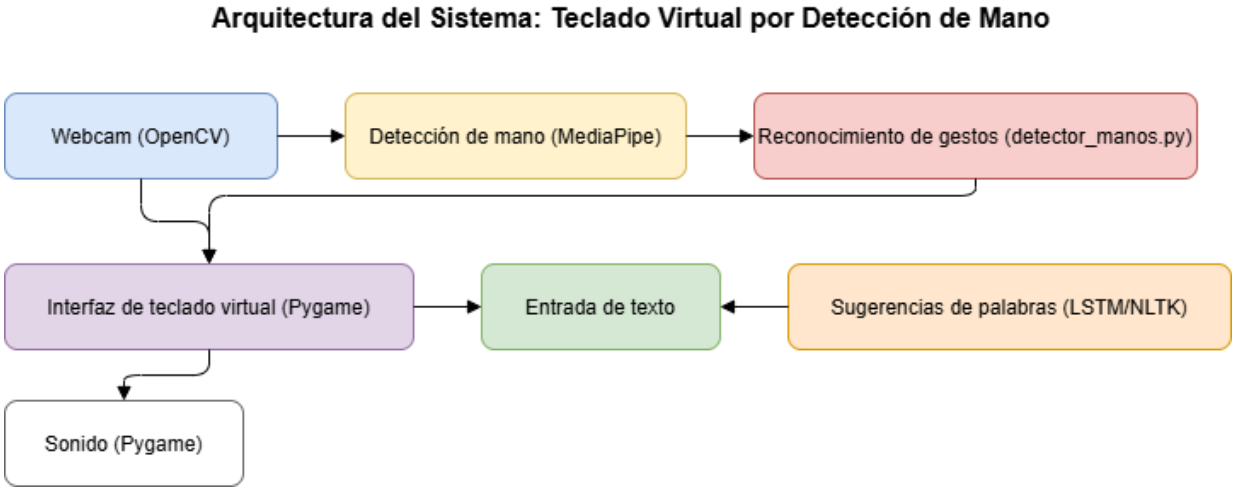

Ilustración 1 Arquitectura del sistema

1. Entrada de Video (OpenCV)

- Captura la imagen de la cámara en tiempo real.
- Proporciona los frames necesarios para el análisis de movimiento.

2. Detección de Manos (MediaPipe)

- Procesa los frames de video para detectar la mano y extraer los puntos clave (landmarks).
- Detecta gestos específicos (puño) usando lógica en el archivo llamado detector_manos.py

3. Lógica del Teclado Virtual (Pygame)

- Genera visualmente un teclado virtual en pantalla mediante la biblioteca Pygame.
- Interpreta la posición del dedo índice (landmark específico) para simular la pulsación de teclas sobre el teclado.
- Administra la entrada de caracteres, la visualización del texto y la interacción general del usuario con la interfaz.

4. Sugerencia de Palabras (LSTM/NLTK)

- Usa un modelo LSTM entrenado (Keras/TensorFlow) para autocompletar palabras.
- Si el modelo no está disponible, se ofrecen sugerencias simples basadas en frecuencia de palabras con ayuda de la biblioteca NLTK.
- El modelo se entrena con un dataset personalizable (palabras.txt) lo que permite adaptarlo a distintos contextos o lenguajes.

5. Sonido (Pygame)

- Reproduce sonidos al presionar teclas.

TECNOLOGÍAS IMPLEMENTADAS

OpenCV (Open Source Computer Vision Library):

Se utiliza para capturar y procesar video en tiempo real desde la cámara web. OpenCV es una biblioteca poderosa y eficiente para tareas de visión por computadora, permitiendo extraer frames del video y enviarlos a otros módulos del sistema. Es esencial en este proyecto porque actúa como la puerta de entrada de datos visuales que permiten rastrear el movimiento de la mano del usuario.

MediaPipe (Google):

Esta biblioteca permite detectar y rastrear manos con alta precisión mediante el uso de modelos de aprendizaje automático preentrenados. Se eligió MediaPipe por su capacidad de trabajar en tiempo real, con bajo consumo de recursos, y ofrecer coordenadas precisas de 21 puntos clave de la mano. Esto facilita el reconocimiento de gestos y el seguimiento del dedo índice, fundamentales para controlar el teclado virtual sin contacto físico.

Pygame:

Pygame proporciona una solución sencilla para crear interfaces gráficas e interactivas. En este proyecto, se utiliza para dibujar el teclado virtual, mostrar los caracteres

seleccionados y reproducir sonidos cuando se presionan teclas. Es ideal para proyectos en Python por su simplicidad, portabilidad y facilidad de integración con otros módulos.

TensorFlow/Keras:

Bibliotecas utilizadas para implementar y entrenar una red neuronal LSTM (Long Short-Term Memory) que se encarga del autocompletado de palabras. Este tipo de red es ideal para procesar secuencias de texto, ya que mantiene en memoria los caracteres previamente ingresados y puede predecir los siguientes con precisión. Se entrena con un conjunto de palabras personalizable para adaptarse al contexto del usuario.

NLTK:

En casos donde no se utilice el modelo LSTM, se emplea NLTK para sugerir palabras basadas en su frecuencia de uso en un corpus de texto. Es una herramienta muy útil para procesamiento de lenguaje natural básico, y actúa como una alternativa ligera y eficaz para la predicción de palabras.

Python:

Todo el sistema está desarrollado en Python debido a su versatilidad, sintaxis clara y amplia disponibilidad de bibliotecas especializadas. Python facilita la integración de visión por computadora, procesamiento de lenguaje, interfaces gráficas y lógica general del sistema, lo que lo convierte en el lenguaje ideal para un proyecto como este.

DESARROLLO Y PROTOTIPO

El desarrollo del prototipo se llevó a cabo en varias etapas, integrando diversas tecnologías para lograr un sistema funcional e interactivo. A continuación, se describen los principales pasos realizados:

1. **Creación de la estructura del proyecto:** Para mantener una organización adecuada, se creó una carpeta principal denominada **PROYECTO-IA**, en la que se integraron todos los archivos y módulos desarrollados. Además, se configuró un entorno virtual de Python para gestionar las dependencias y asegurar la correcta ejecución del sistema en diferentes entornos.

2. **Definición de la arquitectura del sistema:** Se diseñó la estructura general del sistema, seleccionando **Python** como lenguaje de programación principal y utilizando **OpenCV**, **MediaPipe** y **Pygame** para la captura de video, el reconocimiento de gestos y el desarrollo de la interfaz gráfica, respectivamente.

3. **Captura y procesamiento de video:** Se implementó la captura de video en tiempo real mediante **OpenCV**, permitiendo obtener los frames necesarios para analizar el movimiento de la mano del usuario.

4. **Reconocimiento de la mano y gestos:** Se integró la biblioteca **MediaPipe** para detectar la mano y extraer los **landmarks** (puntos clave) de los dedos en cada frame. Además, se desarrolló una lógica específica para reconocer el gesto de **puño cerrado**, el cual activa la función de **borrado rápido** de la última palabra escrita, facilitando así la edición del texto sin necesidad de eliminar letra por letra.

5. **Desarrollo del teclado virtual:** Se diseñó una interfaz gráfica interactiva utilizando **Pygame**, la cual muestra:

    - Las letras del abecedario.
    - La tecla “**Mayus**” para alternar entre mayúsculas y minúsculas.
    - La tecla “**Espacio**” para separar palabras.
    - La tecla “**Borrar**” para eliminar caracteres individualmente.
    - Tres columnas verticales con **números** y **signos de interrogación**.

La interacción se realiza mediante el seguimiento del dedo índice, y se añadió un **sonido de clic** al presionar teclas para mejorar la experiencia del usuario. El archivo de sonido fue descargado desde Pixabay en formato MP3 y posteriormente convertido a formato WAV, siendo ubicado en una subcarpeta dentro del proyecto para su uso dentro del sistema.

6. **Sistema de sugerencias de palabras:** Se implementó un sistema de **autocompletado** mediante una red neuronal **LSTM**, entrenada con un archivo de texto personalizable. En caso de no contar con el modelo entrenado, se utiliza un sistema alternativo de sugerencias por frecuencia de palabras mediante la librería **NLTK**, que trabaja con vocabulario en español.
7. **Integración y pruebas del sistema:** Finalmente, se integraron todos los módulos y se llevaron a cabo **pruebas exhaustivas** para validar la robustez del reconocimiento de gestos, la precisión del sistema de predicción de palabras y la fluidez de la interacción con el teclado virtual.

IMPLEMENTACIÓN

El sistema desarrollado está compuesto por varios módulos en Python, cada uno encargado de una parte fundamental del funcionamiento del teclado virtual controlado por gestos. A continuación, se describen las funciones y clases más relevantes:

1. detector_manos.py

Este módulo contiene la clase HandTracker, responsable de la detección y análisis de la mano en tiempo real utilizando MediaPipe y OpenCV.

- **Clase HandTracker**
  - **__init__:** Inicializa los objetos de MediaPipe necesarios para la detección de manos y el dibujo de los landmarks sobre la imagen de la cámara.
  - **get_finger_position(frame):**
    - Convierte el frame de la cámara a formato RGB y lo procesa con MediaPipe para detectar la mano.

- Si se detecta una mano, obtiene la posición de la punta del dedo índice (landmark 8) y la dibuja en la imagen.
- Devuelve las coordenadas (x, y) del dedo índice, que se utilizan para determinar sobre qué tecla virtual está el usuario.

- **is_fist(frame):**
  - Procesa el frame para detectar si la mano está en posición de puño.
  - Analiza los landmarks de los dedos (excepto el pulgar) y determina si todos están doblados.
  - Si detecta el puño cerrado, devuelve True, lo que activa la función de borrado de texto en el teclado virtual.

2. logica_teclado.py

Este módulo gestiona la lógica principal del teclado virtual, incluyendo la entrada de texto y la sugerencia de palabras.

- **get_suggestions(texto_actual)**:
  - Genera sugerencias de palabras basadas en el texto que el usuario ha escrito.
  - Si existe un modelo LSTM entrenado, lo utiliza para predecir las siguientes palabras posibles.
  - Si no hay modelo, utiliza el método de frecuencia de palabras con NLTK como respaldo.
  - Optimiza el rendimiento actualizando las sugerencias solo cuando el texto cambia.
- **procesar_tecla(tecla):**
  - Gestiona la lógica de entrada de texto, insertando caracteres, espacios o eliminando texto según la tecla seleccionada o el gesto detectado.
  - Se integra con la función de borrado por gesto (puño cerrado).

3. interfaz_teclado.py

Este módulo se encarga de la visualización y la interacción gráfica del teclado virtual usando Pygame.

- **dibujar_teclado()**:
    - Dibuja el teclado virtual en la pantalla, mostrando todas las teclas, el área de texto y las sugerencias de palabras.
    - Resalta la tecla sobre la que se encuentra el dedo índice.
- **dibujar_sugerencias(sugerencias)**:
    - Muestra las palabras sugeridas en la interfaz, permitiendo al usuario seleccionar una para autocompletar el texto.

4. config.py

Este archivo contiene la configuración global del sistema, permitiendo modificar fácilmente parámetros visuales y de funcionamiento sin alterar el resto del código.

- Define los tamaños del teclado, área de texto y sugerencias.
- Especifica los colores de fondo, teclas, resaltados y sugerencias, para elegir una paleta de colores específica se puede tomar de la página de colorhunt.
- Configura las fuentes y tamaños de texto utilizados en la interfaz, en mi caso utilicé una fuente instalada en mi sistema llamada “Georgia Belle”.
- Incluye los parámetros de la cámara (índice, ancho y alto de los frames).

5. main.py

Este archivo contiene el ciclo principal del programa, coordinando la captura de video, la detección de gestos y la actualización de la interfaz.

- **Ciclo principal**:
    - Captura los frames de la cámara en tiempo real.
    - Utiliza la clase HandTracker para detectar la posición del dedo índice y el gesto de puño cerrado.

- Actualiza la interfaz gráfica del teclado y gestiona la entrada de texto y sugerencias.
- Permite la interacción fluida entre el usuario y el sistema, respondiendo tanto a gestos como a la selección de teclas virtuales.

6. sugeridor_palabras.py

Este módulo contiene la lógica para entrenar y utilizar el modelo LSTM encargado de las sugerencias inteligentes de palabras.

- **Entrenamiento y uso del modelo LSTM**:
  - Permite entrenar el modelo con un archivo de palabras personalizado (palabras.txt) el cual contiene una cantidad significativa de palabras escritas de forma ordenada alfabéticamente.
  - Guarda el modelo entrenado en un archivo (lstm_suggester.pkl) para su uso posterior en el sistema.
  - Facilita la actualización del sistema de sugerencias simplemente cambiando el dataset y reentrenando el modelo.

Resumen de la interacción entre módulos:

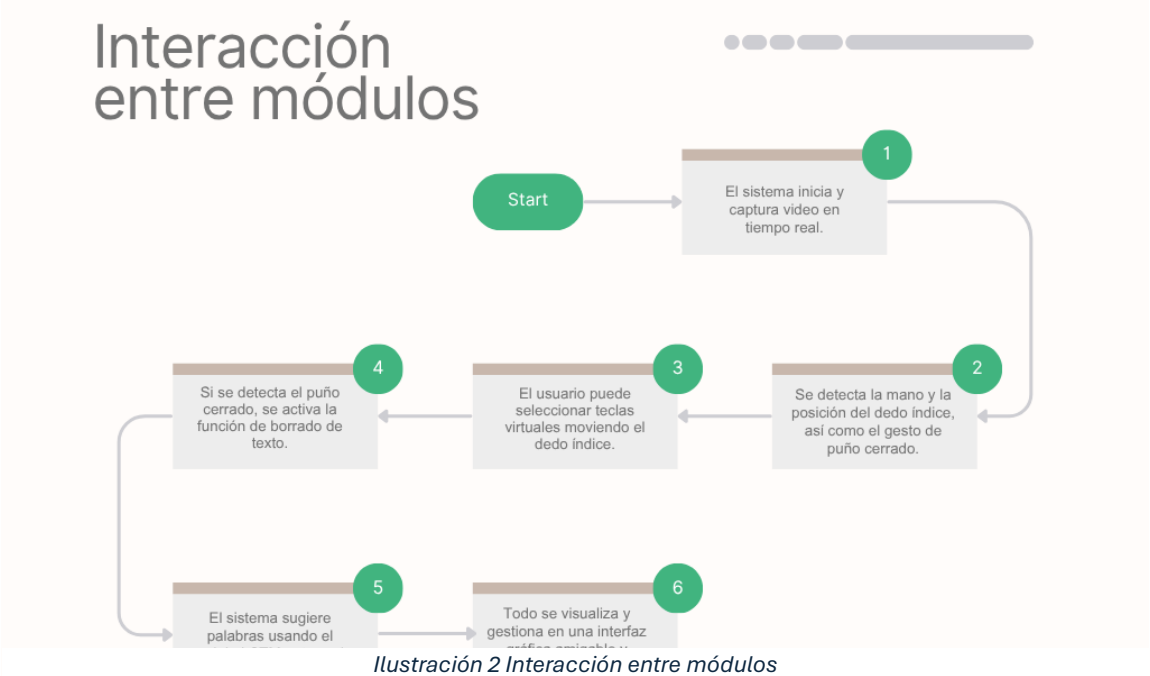

Ilustración 2 Interacción entre módulos

PRUEBAS

A continuación, se describen algunos casos de prueba realizados para validar el funcionamiento del teclado virtual por gestos y su sistema de autocompletado:

1. Ejecución del sistema:

Caso: Ejecutar el archivo principal con el comando: ***python main.py***

Resultado esperado: Se abre la ventana del teclado virtual y se muestra la imagen de la cámara en tiempo real, permitiendo la interacción por gestos.

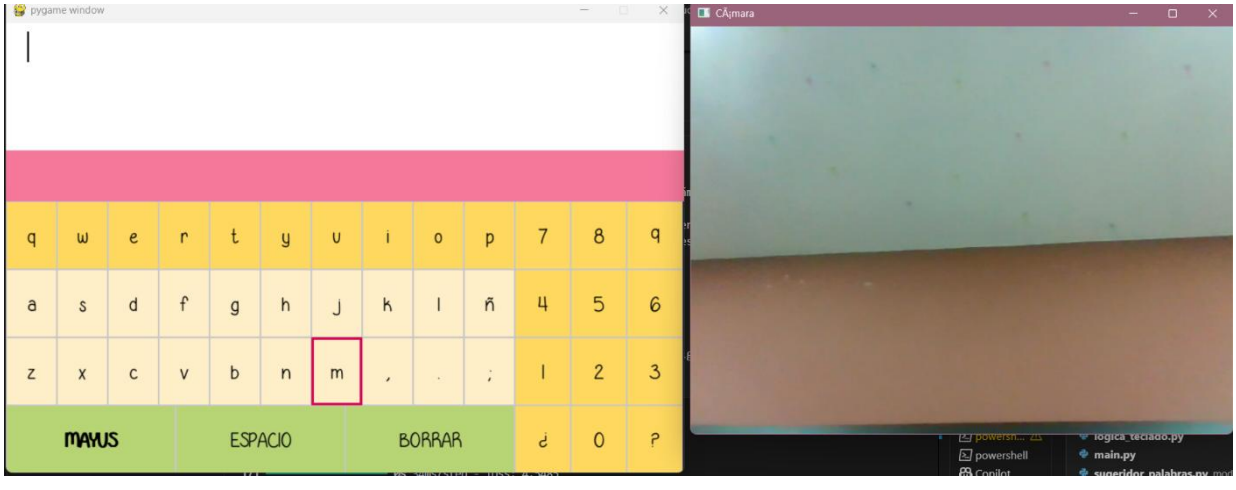

Ilustración 3 Teclado virtual y cámara

2. Selección de caracteres:

Caso: Mover el dedo índice frente a la cámara y posicionar sobre diferentes teclas del teclado virtual.

Resultado esperado: La tecla correspondiente se resalta en la interfaz y, tras mantener el dedo sobre la tecla durante 2 segundos, el carácter se inserta en el área de texto y se reproduce un sonido de clic.

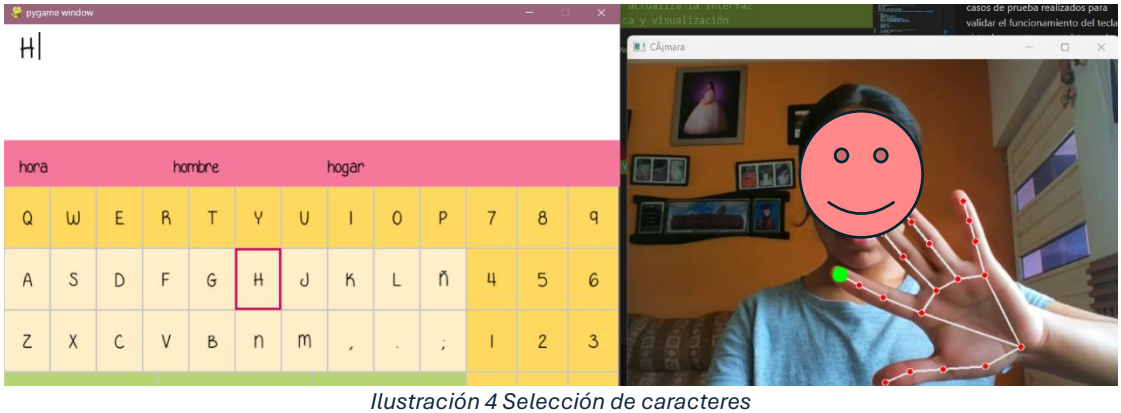

Ilustración 4 Selección de caracteres

3. Borrado de palabras por gesto:

Caso: Realizar el gesto de puño cerrado frente a la cámara.

Resultado esperado: Si han pasado al menos 1.2 segundos desde el último borrado, se elimina la última palabra escrita en el área de texto.

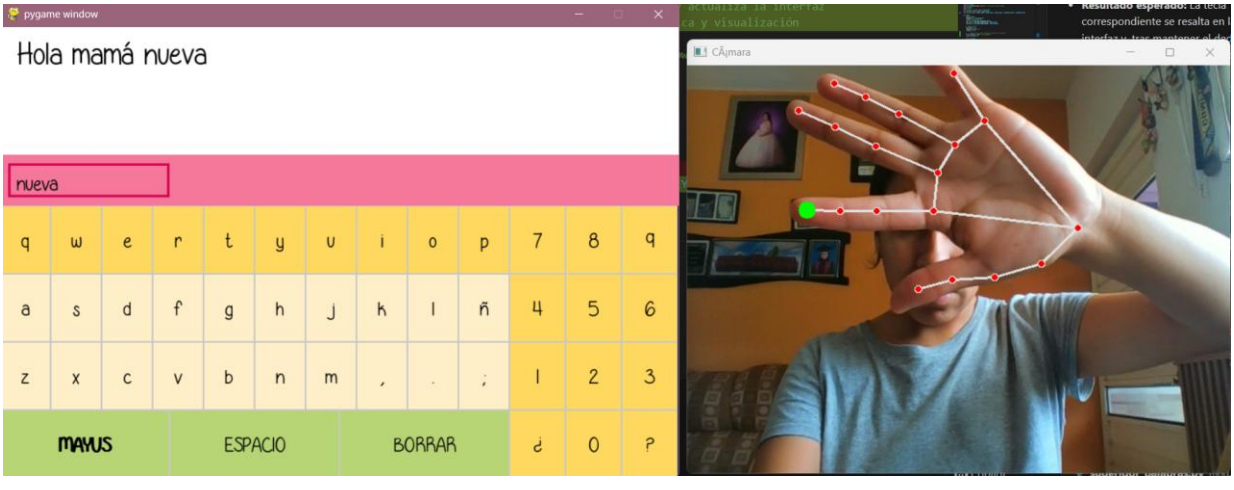

Ilustración 5 Texto completo

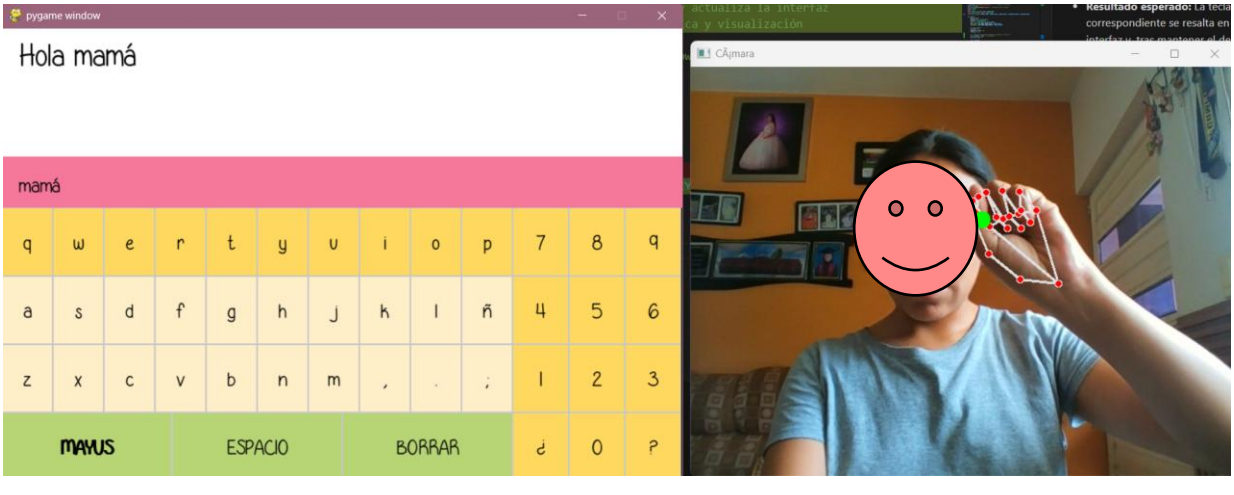

Ilustración 6 Palabra borrada con el gesto de puño

4. Sugerencias de palabras:

Caso: Se comienza a escribir una letra y se muestran sugerencias de autocompletado debajo del área de texto.

Resultado esperado: Las sugerencias corresponden a palabras que comienzan con el prefijo escrito, utilizando el modelo LSTM entrenado o el respaldo de NLTK.

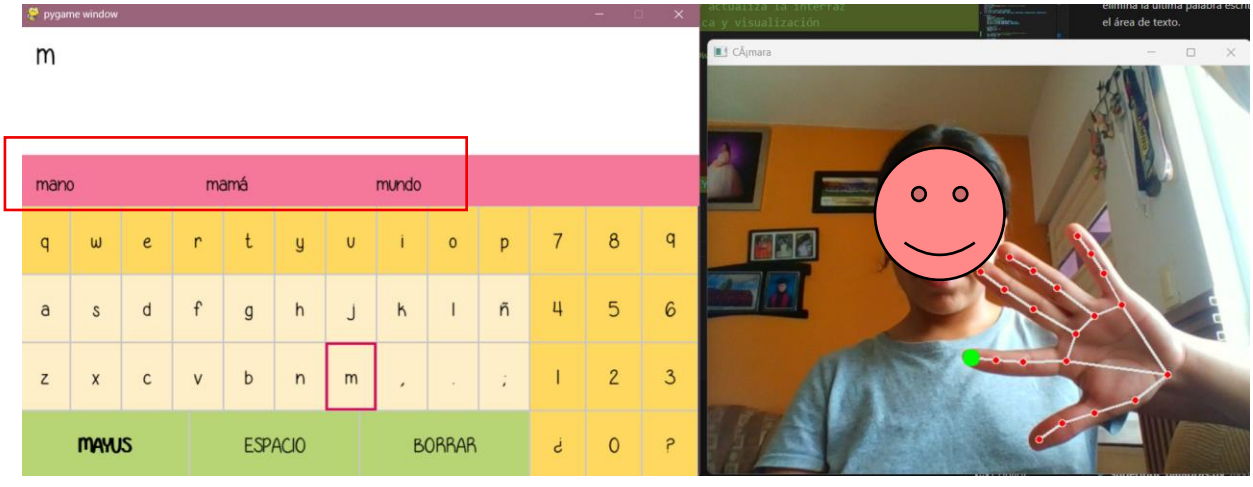

Ilustración 7 Sugerencias de palabras

5. Selección de sugerencias:

Caso: Se posiciona el dedo índice sobre una sugerencia y lo mantiene durante 1.6 segundos.

Resultado esperado: La palabra sugerida se inserta automáticamente en el área de texto, reemplazando la palabra incompleta.

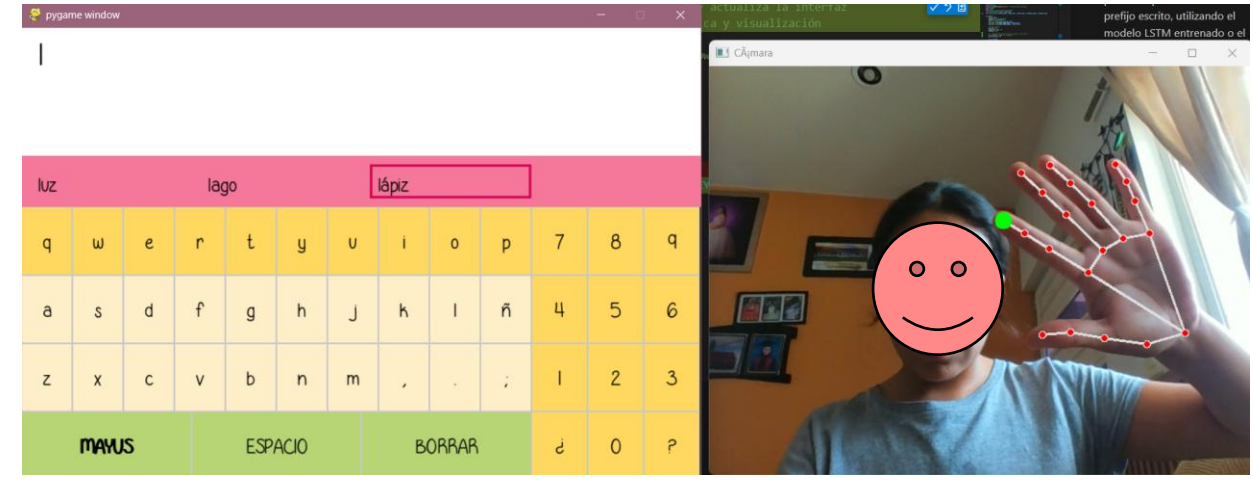

Ilustración 8 Seleccionar sugerencia

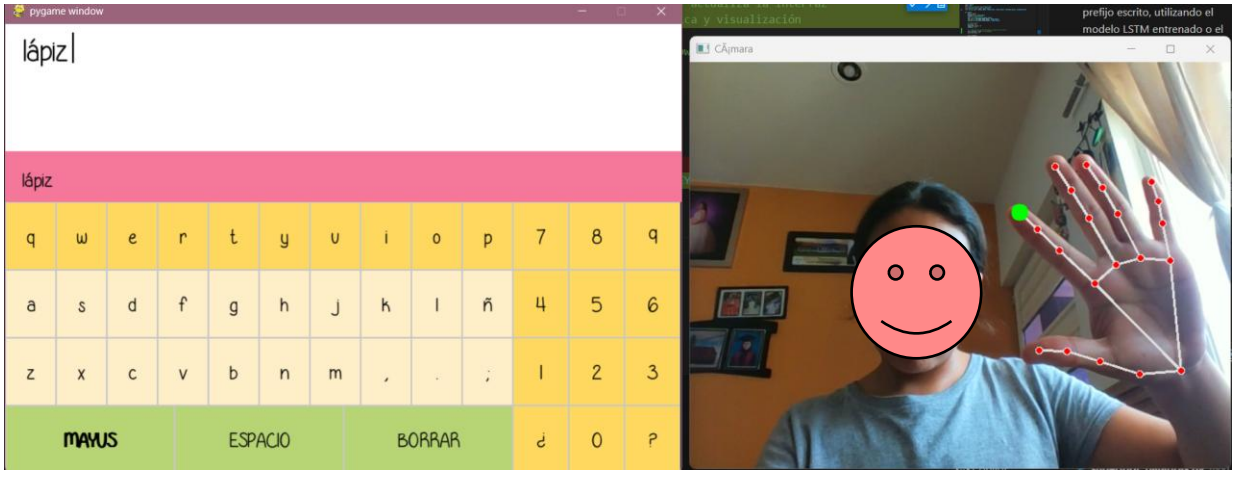

Ilustración 9 Autocompletado de palabra

CONCLUSIONES: LIMITACIONES Y MEJORAS FUTURAS

El desarrollo de este teclado virtual basado en visión por computadora ha demostrado ser una solución funcional y prometedora para mejorar la accesibilidad en la escritura digital mediante gestos con el dedo índice. Sin embargo, como todo prototipo en etapa inicial, presenta ciertas limitaciones que deben ser consideradas para su evolución futura.

En primer lugar, la precisión en la detección de gestos, aunque adecuada en condiciones controladas, puede verse comprometida ante variaciones en la iluminación, fondos visualmente complejos o posiciones atípicas de la mano. Este aspecto representa un desafío importante para su implementación en entornos reales y variados. Por otro lado, el sistema de sugerencia de palabras, basado en un modelo LSTM, depende fuertemente de la calidad y diversidad del dataset utilizado para el entrenamiento. Un conjunto de datos limitado puede llevar a sugerencias poco útiles o variadas, afectando la eficiencia del usuario al escribir.

En cuanto al rendimiento, el uso simultáneo de librerías como OpenCV, MediaPipe, Pygame y TensorFlow requiere una computadora con recursos moderados, lo que puede limitar su accesibilidad en dispositivos de bajo rendimiento. Cabe destacar también que el sistema está actualmente optimizado únicamente para el idioma español, lo cual restringe su adopción por parte de usuarios de otras lenguas.

Existen múltiples líneas de mejora para potenciar este sistema. Una de las principales sería la optimización de la detección de gestos, incorporando técnicas de preprocesamiento de imagen o modelos más avanzados para garantizar mayor robustez en distintas condiciones ambientales. Asimismo, se propone la ampliación del dataset de entrenamiento, incluyendo una mayor cantidad de palabras y múltiples idiomas, con el fin de enriquecer las sugerencias de autocompletado.

Otra mejora significativa sería el rediseño de la interfaz gráfica, orientada a una experiencia más intuitiva, accesible y visualmente amigable, incorporando animaciones y retroalimentación visual. También se plantea la posibilidad de hacer el sistema multiplataforma, adaptándolo para funcionar en distintos sistemas operativos e incluso dispositivos móviles.

En términos de personalización, una propuesta interesante sería permitir al usuario configurar gestos personalizados para ejecutar diferentes acciones según sus necesidades. Finalmente, integrar reconocimiento de voz como complemento para la entrada de texto puede brindar una alternativa híbrida útil en contextos específicos.

En resumen, este prototipo representa un avance importante en el desarrollo de interfaces accesibles y sin contacto, con amplio potencial de mejora y aplicación en diversos contextos sociales, educativos y médicos.