Facultad de Negocios y Tecnologías

Región Orizaba - Córdoba

PROYECTO FINAL

INTELIGENCIA ARTIFICIAL

Licenciatura en Tecnologías de Información en las organizaciones

Estudiante:

Angel Gabriel Torres Hernández

Docente:

Jesus Leonardo Lopez Hernández

Periodo: Febrero – junio 2025

1. Introducción: Problemática y Justificación

1.1. Problemática

En la era digital, la interacción fluida con dispositivos electrónicos es fundamental para la autonomía, la comunicación y la participación social. Sin embargo, las personas con diversas discapacidades motrices enfrentan barreras significativas al no poder utilizar interfaces convencionales como teclados y ratones. Esto genera una brecha de accesibilidad que limita su acceso a la información, la educación, el trabajo y la vida social. Las soluciones alternativas a menudo resultan costosas, complejas o insuficientemente adaptables, lo que acentúa la necesidad de innovaciones inclusivas y de bajo costo que la tecnología moderna puede ofrecer. El desafío reside en desarrollar un sistema de entrada de texto que no solo sea intuitivo y accesible, sino también eficiente, aprovechando hardware común como una simple cámara web.

1.2. Justificación y Solución Propuesta

La visión por computadora y las técnicas de Inteligencia Artificial ofrecen una oportunidad transformadora para superar estas barreras. En lugar de depender de métodos más complejos o invasivos, este proyecto se centra en el seguimiento de manos (Hand Tracking) como un método de control robusto, directo y natural. Al detectar y analizar la posición de la mano y gestos específicos como el "pellizco" (la unión del pulgar y el índice) podemos emular de manera intuitiva un puntero de ratón y la acción de "click".

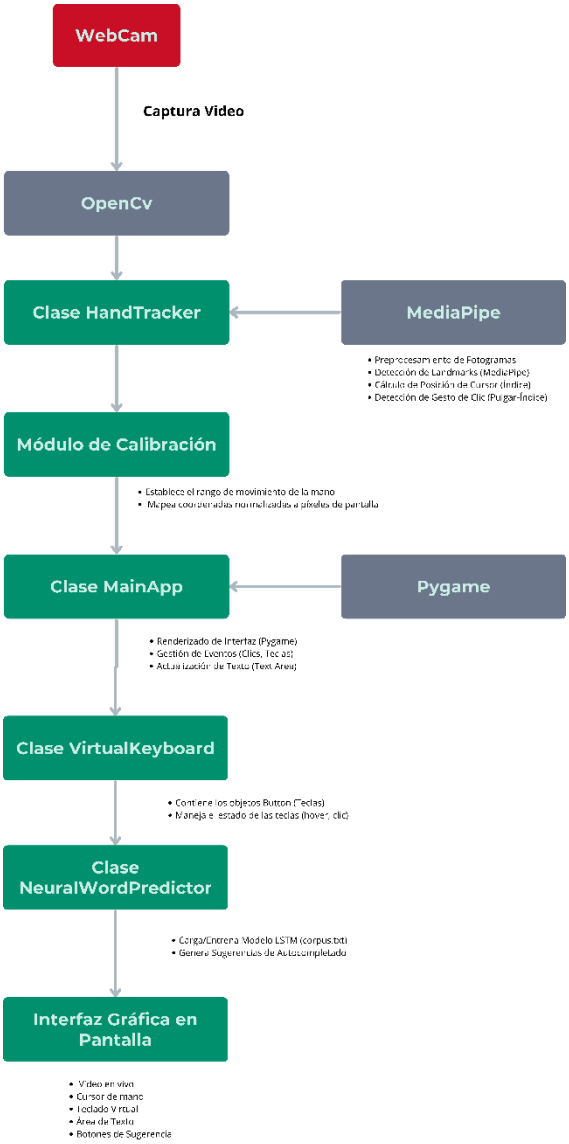
Para optimizar la velocidad y fluidez de la escritura, integramos un componente avanzado de Procesamiento de Lenguaje Natural (PLN). La solución propuesta es el desarrollo de un prototipo de teclado virtual controlado por gestos de la mano, que incorpora una Red Neuronal Recurrente (LSTM) para predecir y autocompletar palabras de forma inteligente. Esto no solo facilita la interacción, sino que también acelera significativamente el proceso de escritura.

Este enfoque innovador combina una interfaz humano-computadora intuitiva y sin contacto con un asistente de escritura inteligente, todo ello diseñado para funcionar eficientemente con una cámara web estándar, haciendo la tecnología más inclusiva y al alcance de todos.

2. Arquitectura: Diagrama del Sistema y Tecnologías

El sistema está diseñado con una arquitectura modular donde cada componente tiene una responsabilidad clara, desde la captura de la imagen hasta la predicción del texto.

2.1. Diagrama de Flujo del Sistema



2.2. Tecnologías Implementadas:

- **Python 3:** Lenguaje principal de desarrollo.
- **OpenCV:** Utilizada para la captura de video desde la cámara web y el procesamiento básico de imágenes.
- **MediaPipe (de Google):** Es el corazón del seguimiento de manos. Proporciona un modelo pre-entrenado que detecta la malla de la mano (hand landmarks) en tiempo real con alta precisión.
- **Pygame:** Es el motor de la aplicación. Se encarga de crear la ventana, renderizar toda la interfaz gráfica (teclado, área de texto, video), manejar el bucle principal y gestionar los eventos.
- **TensorFlow / Keras:** Frameworks de Machine Learning utilizados para construir, entrenar y ejecutar el modelo de red neuronal LSTM para la predicción de palabras.
- **NumPy:** Librería fundamental para todos los cálculos numéricos, especialmente para medir distancias entre los puntos de la mano y realizar el mapeo de coordenadas.

3. Desarrollo y Prototipo

El desarrollo se abordó en fases modulares, permitiendo construir y probar cada parte de forma independiente antes de su integración.

3.1. Seguimiento de la Mano (HandTracker):

- Se implementó una clase (HandTracker) para encapsular la lógica de MediaPipe.
- El sistema captura el video de la cámara, lo invierte horizontalmente (efecto espejo) y lo convierte a formato RGB.
- MediaPipe procesa el fotograma y devuelve los 21 landmarks (puntos clave) de la mano detectada.
- Se extrae la posición de la punta del dedo índice (landmark #8) como la coordenada del cursor.

- Se calcula la distancia euclidiana entre la punta del dedo índice (#8) y la punta del pulgar (#4). Si esta distancia es menor a un umbral (CLICK_THRESHOLD), se registra un gesto de "clic".

3.2. Calibración y Mapeo de Coordenadas:

- Para que el movimiento de la mano se traduzca de forma intuitiva a la pantalla, se implementó un sistema de calibración de 2 puntos.
- **Proceso:** El usuario apunta a una esquina superior izquierda y hace el gesto de clic, y luego lo repite en una esquina inferior derecha.
- El sistema almacena las coordenadas normalizadas de la mano en esos dos puntos y las utiliza para mapear linealmente todo el rango de movimiento de la mano al área completa de la pantalla. Se añade un pequeño padding para asegurar que se puedan alcanzar los bordes.

3.3. Interfaz Gráfica (VirtualKeyboard y MainApp):

- Utilizando Pygame, se diseñó un teclado virtual completo con filas para números, caracteres y teclas especiales (MAYÚS, BORRAR, ESPACIO).
- Cada tecla es un objeto Button con su propio estado (normal, hover) y valor.
- Se diseñó un área de texto donde se muestra lo que el usuario escribe y un área superior donde aparecen las sugerencias de palabras.
- La ventana también muestra el feed de la cámara en una esquina, para que el usuario siempre tenga referencia de la posición de su mano.

3.4. Predictor de Palabras con LSTM (NeuralWordPredictor):

- Esta es la mejora clave de IA. Se diseñó un modelo de Red Neuronal Recurrente de tipo LSTM (Long Short-Term Memory).
- **Entrenamiento (en tiempo de ejecución):** Al iniciar la aplicación, el modelo se entrena con un archivo de texto (corpus.txt). El texto se descompone en secuencias de caracteres (ej. "h", "ho", "hol", "hola").

- **Arquitectura del Modelo:** El modelo tiene una capa de Embedding (para convertir caracteres en vectores), una capa LSTM (para aprender el contexto secuencial) y una capa Dense con activación softmax que predice la probabilidad del siguiente carácter.
- **Inferencia:** Cuando el usuario escribe un prefijo (ej. "proy"), el modelo predice el siguiente carácter más probable. Este se añade al prefijo y el proceso se repite hasta que el modelo predice un espacio, completando así la palabra.

4. Implementación: Explicación del Código Clave

4.1. HandTracker.get_frame():

- Esta función es el motor del control por gestos. Lee un fotograma de la cámara, lo procesa con self.hands.process(rgb_frame) y, si detecta una mano, extrae los landmarks.
- Calcula la distancia entre los dedos índice y pulgar para determinar si se está realizando el gesto de "pellizco" (is_clicking).
- Devuelve el fotograma (para mostrar en pantalla), la posición normalizada del cursor y el estado del clic.

4.2. NeuralWordPredictor._train_model():

- Esta función prepara los datos del corpus para el entrenamiento. Lee las palabras, crea un vocabulario de caracteres únicos y los convierte en secuencias numéricas.
- Usa pad_sequences para que todas las secuencias tengan la misma longitud.
- Finalmente, define la arquitectura del modelo con Sequential de Keras y lo entrena con model.fit().
- **Nota:** En un producto final, este modelo se entrenaría offline y solo se cargarían los pesos al iniciar.

4.3. NeuralWordPredictor.predict(prefix):

- Recibe un prefijo (ej. "intel") y entra en un bucle.
- En cada iteración, convierte el prefijo actual a una secuencia numérica, la formatea con pad_sequences y la pasa al modelo (self.model.predict()).
- El modelo devuelve un vector de probabilidades para el siguiente carácter. Se elige el más probable (np.argmax()), se añade al prefijo, y el bucle continúa hasta que se predice un espacio o se alcanza una longitud máxima.

4.4. MainApp.handle_calibration() y _map_hand_to_screen():

- handle_calibration gestiona el estado del proceso de calibración, pidiendo al usuario que apunte a las esquinas y registrando las coordenadas de la mano (normalized_pos) cuando hace clic.
- _map_hand_to_screen aplica una regla de tres simple para convertir la posición normalizada de la mano (un valor entre 0.0 y 1.0) a coordenadas de píxeles en la pantalla, usando los límites (min_x, max_x, etc.) guardados durante la calibración.

5. Pruebas: Casos de Uso

Se definieron los siguientes casos de prueba para validar la funcionalidad del prototipo:

Caso de Prueba	Acción del Usuario	Resultado Esperado
Calibración Exitosa	El usuario inicia la app, apunta a la esquina superior izquierda y pellizca, luego repite en la esquina inferior derecha.	El sistema pasa al modo "TYPING". El cursor rojo en pantalla sigue el movimiento de la mano de forma precisa.

Escritura de Caracteres	El usuario mueve el cursor sobre las teclas 'H', 'O', 'L', 'A' y pellizca en cada una.	El texto "hola" aparece correctamente en el área de texto.
Uso de Teclas Especiales	1. Clic en 'MAYÚS'. 2. Clic en 'P'. 3. Clic en '←' (Borrar). 4. Clic en '␣' (Espacio).	1. El modo mayúsculas se activa. 2. El texto muestra "...P". 3. La 'P' se borra. 4. Se añade un espacio al texto.
Predicción y Autocompletado	El usuario escribe "inteli".	Aparece un botón de sugerencia con el texto "INTELIGENCIA". Al hacer clic en él, el texto del área principal se actualiza a "inteligencia ".
Robustez del Clic	El usuario mantiene el gesto de pellizco sobre una tecla.	La tecla se escribe una sola vez. No se repiten caracteres gracias al sistema de cooldown (CLICK_COOLDOWN_MS).

6. Conclusiones: Limitaciones y Mejoras Futuras

6.1. Conclusiones

El proyecto cumplió exitosamente con su objetivo general, desarrollando un prototipo funcional que permite la entrada de texto mediante gestos de la mano y lo acelera con un predictor neuronal. Se aplicaron con éxito técnicas de visión por computadora (MediaPipe) y aprendizaje automático (TensorFlow/Keras), integrándolas en una aplicación cohesiva y usable. El sistema demuestra ser una alternativa viable y de bajo costo a las interfaces tradicionales.

6.2. Limitaciones Actuales

- **Dependencia del Entorno:** La precisión del seguimiento de MediaPipe puede verse afectada por una mala iluminación, fondos complejos o si la mano está parcialmente oculta.
- **Modelo Predictivo Básico:** El modelo LSTM se entrena "al vuelo" con un corpus pequeño, lo que limita la calidad y variedad de sus predicciones. Además, el entrenamiento al inicio ralentiza el arranque de la aplicación.
- **Calibración Simplista:** La calibración de 2 puntos es rápida pero puede no ser perfectamente precisa si el usuario mueve la mano en un plano no del todo paralelo a la cámara.
- **Fatiga del Usuario:** Mantener la mano en el aire durante periodos prolongados puede ser cansado.

6.3. Mejoras Futuras

- **Cargar un Modelo Pre-entrenado:** En lugar de entrenar el LSTM cada vez, se podría entrenar un modelo mucho más grande y complejo de forma offline y simplemente cargar el archivo de pesos (.h5) al iniciar la app, mejorando drásticamente el tiempo de arranque y la calidad de la predicción.
- **Calibración Multipunto:** Implementar una calibración que use más puntos (ej. 9 puntos, como una cuadrícula) y un modelo de regresión para un mapeo más preciso y no lineal.
- **Sistema de Dwell Click (Clic por Permanencia):** Añadir una opción para que el "clic" se active automáticamente si el usuario mantiene el cursor sobre una tecla durante un tiempo determinado, reduciendo la necesidad de hacer el gesto de pellizco constantemente.
- **Gestos Adicionales:** Integrar más gestos para acciones complejas, como "mano abierta" para borrar la última palabra o "puño cerrado" para activar/desactivar el cursor.

7. Ventajas y Desventajas

Ventajas

- **Interacción sin contacto físico:** Permite el uso sin necesidad de manipulación manual.
- **Tecnología accesible:** Utiliza componentes de software y hardware comunes.
- **Feedback auditivo y visual:** Proporciona retroalimentación clara al usuario.
- **Personalizable:** Se adapta a las necesidades y rango de movimiento de cada usuario.
- **Bajo costo de implementación:** Basado en bibliotecas y hardware relativamente económicos.

Desventajas

- **Precisión del seguimiento:** Puede verse afectada por la iluminación o movimientos bruscos.
- **Curva de aprendizaje:** El usuario podría necesitar tiempo para acostumbrarse al control por gestos.
- **Fatiga del usuario:** El uso prolongado podría causar cansancio en la mano/brazo.
- **Dependencia del hardware:** Requiere una cámara web compatible para su funcionamiento.
- **Predicción limitada:** La calidad de las sugerencias de texto depende del corpus de entrenamiento.

8. Aprendizajes Adquiridos

Durante este proyecto, profundicé mi comprensión sobre:

- **Integración de Visión Computacional e IA:** Aprendí a combinar seguimiento de mano y procesamiento de lenguaje natural para una interfaz interactiva.
    - **Mi Código:** La base está en HandTracker (visión) y NeuralWordPredictor (IA/PNL).

- **Interfaces Sin Contacto Físico:** Logré desarrollar un entorno funcional donde las teclas se seleccionan solo con el seguimiento del dedo índice.
    - **Mi Código:** Esto se ve en cómo HandTracker mapea la posición a la pantalla y la clase Button detecta el hover y la lógica de "pellizco" en MainApp genera el clic.
- **Funcionalidades Inteligentes para Teclados:** Implementé autocompletado, borrado y sugerencias de texto.
    - **Mi Código:** La NeuralWordPredictor es clave para el autocompletado, y la lógica de handle_typing en MainApp maneja el borrado y el espacio.
- **Mejora de la Experiencia del Usuario (UX):** Comprendí la importancia de la retroalimentación visual, la tolerancia al error y la adaptabilidad.
    - **Mi Código:** Implementé retroalimentación visual (cambio de color en hover, cursor de mano), suavizado del cursor (SMOOTHING_FACTOR) para tolerancia al error, y una fase de calibración para la adaptabilidad.