Universidad Veracruzana

Proyecto TecladoDigital

Facultad de negocios y Tecnologías-Campus Ixtac
Región Orizaba-Córdoba
Tecnologías de Información en las Organizaciones
Inteligencia Artificial
Alumno: León González Josué
Docente: Dr. López Hernández, Jesus Leonardo

Introducción: Problemática

Hoy en día, las interfaces hombre-máquina (HMI) se enfrentan al reto de ser más accesibles, intuitivas y adaptables a distintos contextos, como la discapacidad motriz o la falta de hardware físico. Este proyecto surge de la necesidad de desarrollar una alternativa inteligente a los teclados tradicionales, capaz de ser operado sin contacto físico, mediante el seguimiento de la mano y reconocimiento de gestos usando visión artificial. Además, se incorpora un sistema de predicción de palabras mediante aprendizaje de lenguaje, lo cual mejora la velocidad de escritura y la experiencia del usuario.

Justificación del Proyecto

El desarrollo de interfaces de entrada alternativas se ha vuelto esencial en un mundo donde la accesibilidad, la interacción sin contacto y la personalización de sistemas cobran cada vez más relevancia. Este proyecto responde a la necesidad de ofrecer una solución innovadora que permita la escritura sin necesidad de dispositivos físicos tradicionales como teclados o pantallas táctiles.

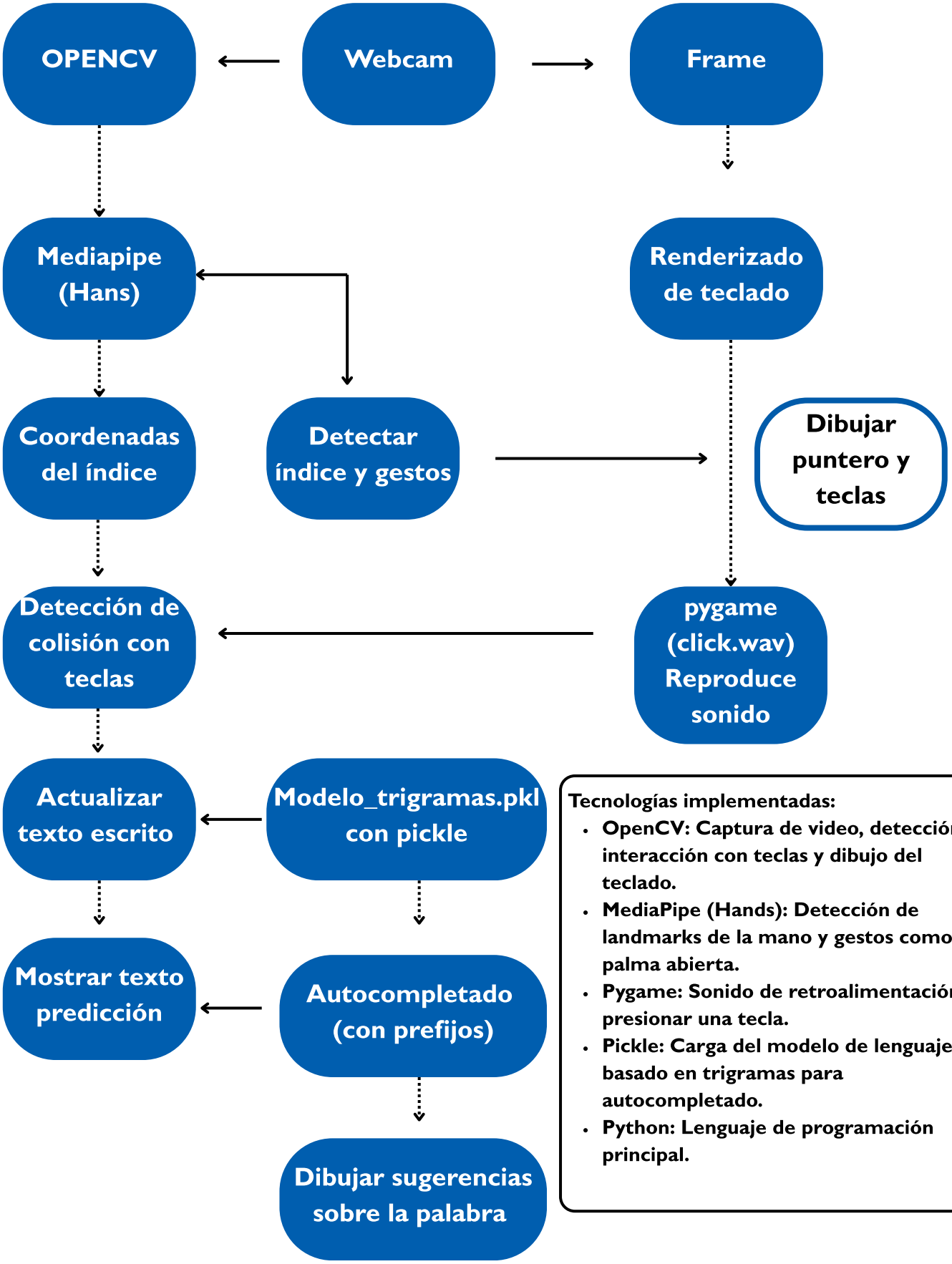
En particular, el teclado virtual controlado por gestos de la mano se justifica por su potencial para ser utilizado en diversas situaciones: desde entornos médicos o industriales donde no se pueden utilizar interfaces físicas, hasta casos de personas con movilidad reducida que necesitan métodos de interacción más inclusivos y accesibles.

Además, la incorporación de funciones como el autocompletado de palabras mediante modelos de lenguaje mejora significativamente la velocidad de escritura y reduce el esfuerzo cognitivo del usuario. Esto lo convierte en una herramienta práctica tanto para uso cotidiano como para la investigación en tecnologías de asistencia.

El uso de tecnologías ligeras como MediaPipe para la detección de gestos y modelos de predicción de texto basados en trigramas garantiza un buen rendimiento sin necesidad de grandes recursos computacionales, lo que lo hace viable incluso en equipos con hardware limitado.

En conjunto, este proyecto no solo explora nuevas formas de interacción hombre-máquina, sino que también promueve la inclusión digital, la eficiencia y la innovación tecnológica a través de una solución práctica, económica y adaptable.

Arquitectura: Diagrama del Sistema



Tecnologías implementadas:

- OpenCV: Captura de video, detección de interacción con teclas y dibujo del teclado.
- MediaPipe (Hands): Detección de landmarks de la mano y gestos como la palma abierta.
- Pygame: Sonido de retroalimentación al presionar una tecla.
- Pickle: Carga del modelo de lenguaje basado en trigramas para autocompletado.
- Python: Lenguaje de programación principal.

```
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=1, min_detection_confidence=0.7)
```

```
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.flip(frame, 1)
    height, width, _ = frame.shape
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    resultado = hands.process(rgb)
```

```
import pygame
import pickle
from collections import defaultdict

# Inicializar pygame para sonido
pygame.init()
click_sound = pygame.mixer.Sound("click.wav")
```

```
# Cargar modelo de predicción de trigramas
with open("modelo_trigramas.pkl", "rb") as f:
    modelo_trigramas = pickle.load(f)
```

Desarrollo y Prototipo

El prototipo consiste en un teclado virtual mostrado por pantalla, el cual puede ser controlado mediante el movimiento de la punta del dedo índice. Se detecta también la palma abierta como gesto de "borrar todo". Además, se incluye un sistema de predicción de palabras basado en trigramas entrenado a partir de un corpus propio.

- El usuario apunta con su dedo índice a una tecla y debe mantenerlo durante 1 segundo para seleccionarla.
- Las palabras sugeridas aparecen justo debajo de la palabra escrita, y pueden ser seleccionadas de forma similar.
- Al detectar la palma de la mano completamente abierta, el texto completo es borrado.

Implementación: Explicación del Código

El sistema desarrollado utiliza Python como lenguaje principal y emplea diversas bibliotecas como OpenCV, MediaPipe, Pygame y Pickle. Cada una cumple una función específica dentro del flujo de trabajo del teclado virtual. El código está estructurado para capturar la imagen desde la cámara web, identificar la mano del usuario, detectar la punta del dedo índice y determinar si se encuentra dentro de los límites de una tecla virtual.

Una de las funciones clave es la de detección de la punta del dedo índice, la cual permite señalar con precisión una tecla en la pantalla. Esta función trabaja junto con una lógica de temporización que permite seleccionar la tecla solo después de mantener el dedo en ella durante al menos un segundo, evitando pulsaciones accidentales.

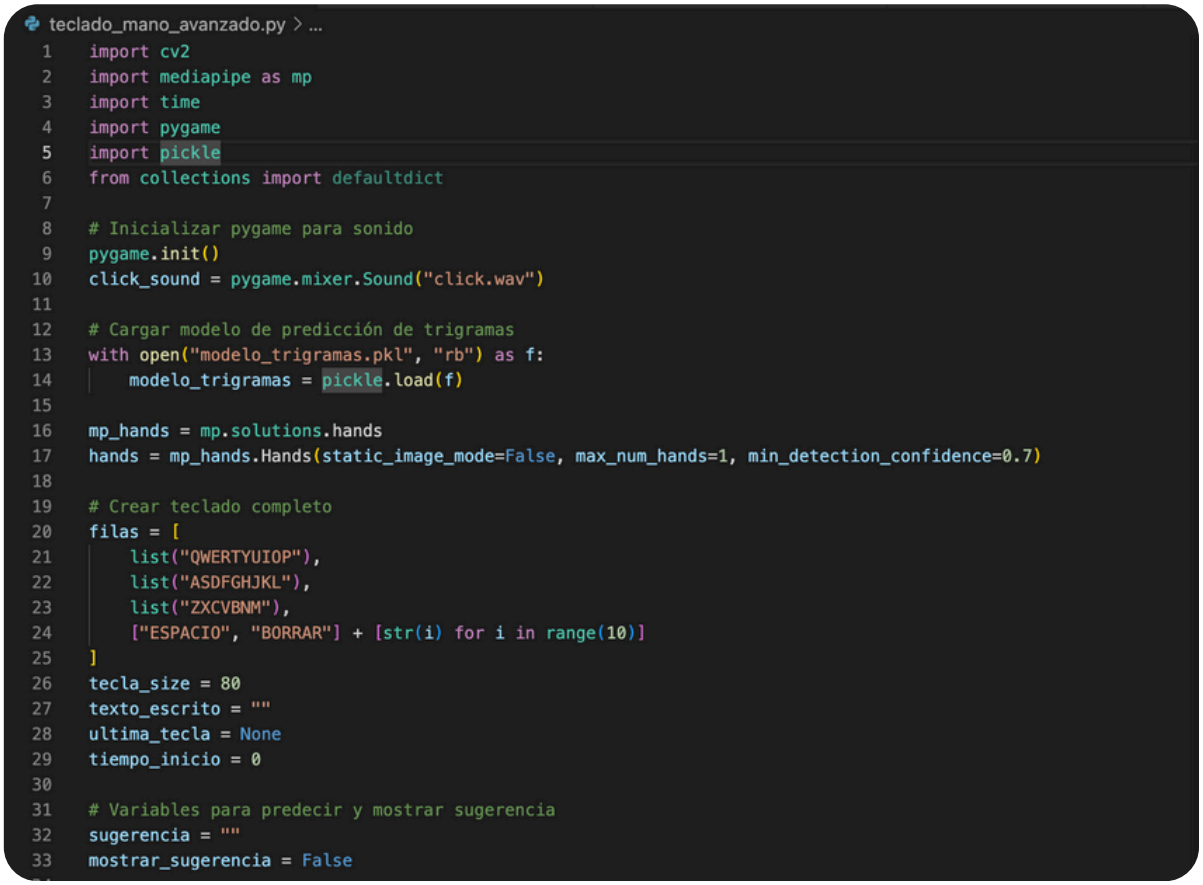
Otra función central es la detección de gestos de palma abierta, usada para borrar todo el texto escrito como un método rápido de limpieza. También se incorporó una función de reproducción de sonido usando Pygame para simular el clic de teclado al seleccionar una tecla.

En cuanto a la predicción de texto, se cargó un modelo entrenado previamente con frases y prefijos en español usando trigramas. Cuando el usuario comienza a escribir una palabra, el sistema analiza las últimas letras y despliega sugerencias de autocompletado en una pequeña ventana justo debajo del texto. Estas sugerencias también son interactuables y se pueden seleccionar con el dedo, igual que una tecla normal.

Pruebas del Sistema

Para validar el correcto funcionamiento del teclado, se realizaron diversos casos de prueba. Primero, se verificó la detección de caracteres individuales con precisión y sin errores de selección. Posteriormente, se probó la selección de la tecla "BORRAR" con el gesto de la mano abierta, la cual funcionó correctamente y de forma intuitiva.

El sistema también fue probado con frases comunes para evaluar la efectividad del autocompletado. Por ejemplo, al escribir ‘‘me gu’’, el sistema sugería ‘‘me gusta’’, y si el usuario colocaba su dedo sobre la sugerencia, esta se completaba automáticamente. La predicción funciona tanto a partir de letras individuales como a partir de combinaciones más avanzadas, demostrando un modelo de lenguaje ligero pero eficiente.

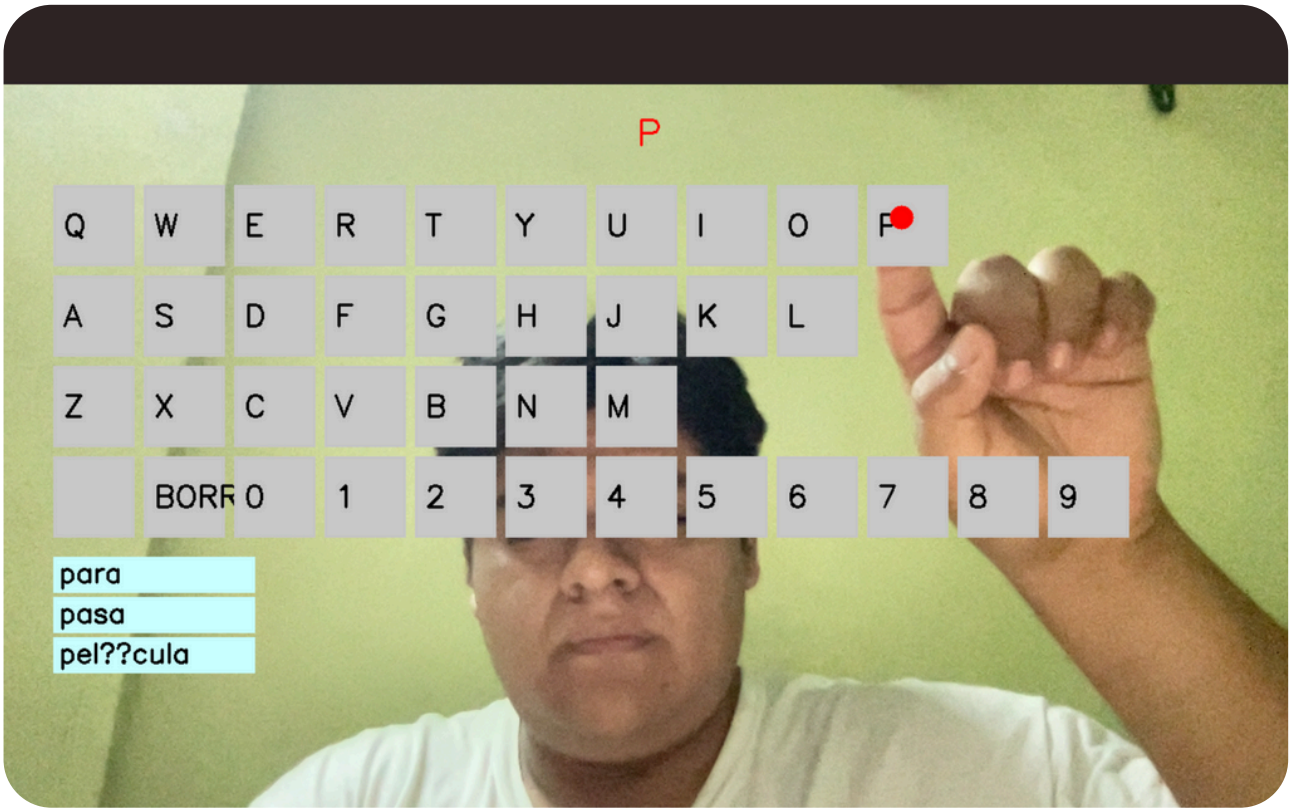


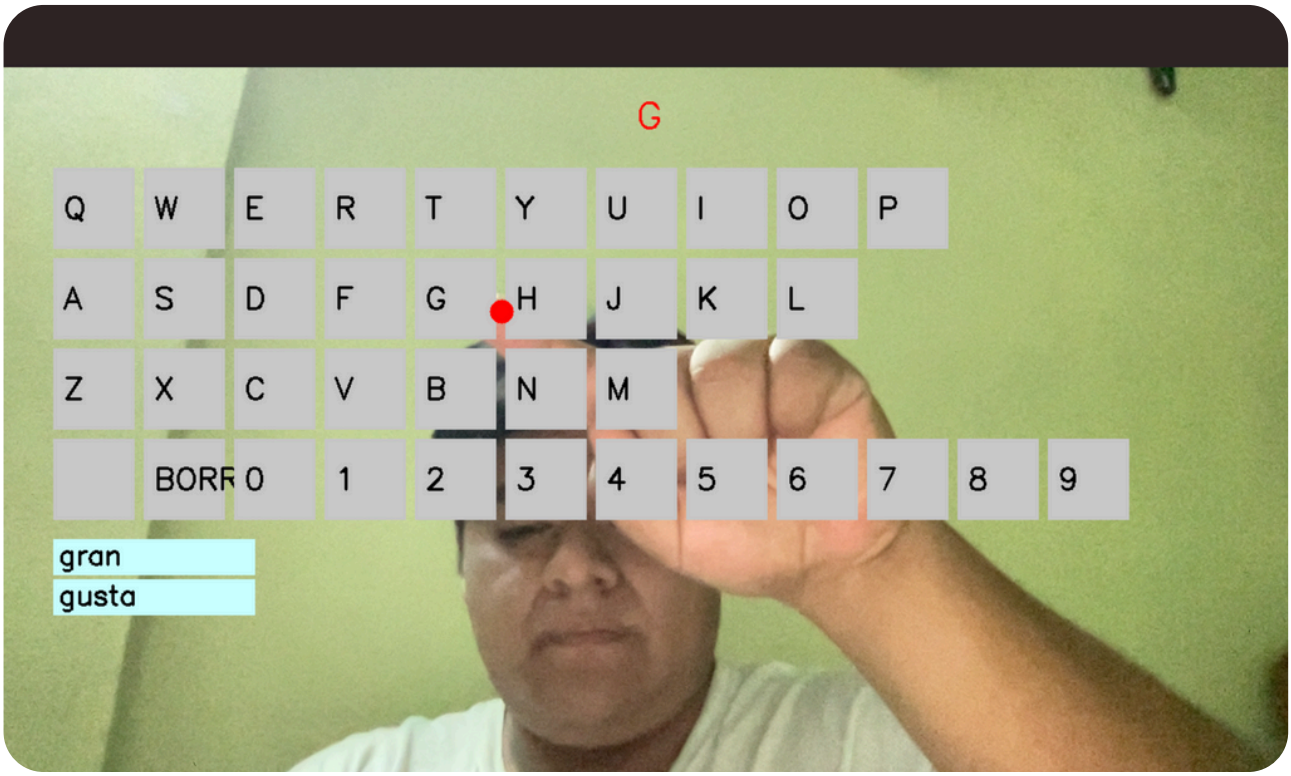
```python
import cv2
import mediapipe as mp
import time
import pygame
import pickle
from collections import defaultdict

# Inicializar pygame para sonido
pygame.init()
click_sound = pygame.mixer.Sound("click.wav")

# Cargar modelo de predicción de trigramas
with open("modelo_trigramas.pkl", "rb") as f:
    modelo_trigramas = pickle.load(f)

mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=1, min_detection_confidence=0.7)

# Crear teclado completo
filas = [
    list("QWERTYUIOP"),
    list("ASDFGHJKL"),
    list("ZXCVBNM"),
    ["ESPACIO", "BORRAR"] + [str(i) for i in range(10)]
]
tecla_size = 80
texto_escrito = ""
ultima_tecla = None
tiempo_inicio = 0

# Variables para predecir y mostrar sugerencia
sugerencia = ""
mostrar_sugerencia = False
```

```python
def detectar_gesto_palma_abierta(landmarks):
    dedos_abiertos = [
        landmarks.landmark[8].y < landmarks.landmark[6].y,
        landmarks.landmark[12].y < landmarks.landmark[10].y,
        landmarks.landmark[16].y < landmarks.landmark[14].y,
        landmarks.landmark[20].y < landmarks.landmark[18].y
    ]
    pulgar_izq = landmarks.landmark[4].x < landmarks.landmark[3].x
    pulgar_der = landmarks.landmark[4].x > landmarks.landmark[3].x
    return all(dedos_abiertos) and (pulgar_izq or pulgar_der)

def detectar_punta_indice(hand_landmarks, width, height):
    x = int(hand_landmarks.landmark[8].x * width)
    y = int(hand_landmarks.landmark[8].y * height)
    return x, y

def predecir_palabra(texto, modelo):
    palabras = texto.strip().split()
    if not palabras:
        return ""
    prefijo = palabras[-1].lower()
    opciones = set()
    for clave, siguientes in modelo.items():
        for palabra, _ in siguientes.items():
            if palabra.startswith(prefijo):
                opciones.add(palabra)
    return sorted(opciones)[:3]  # Máximo 3 sugerencias

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.flip(frame, 1)
    height, width, _ = frame.shape
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    resultado = hands.process(rgb)
```


```python
def detectar_gesto_palma_abierta(landmarks):
    dedos_abiertos = [
        landmarks.landmark[8].y < landmarks.landmark[6].y,
        landmarks.landmark[12].y < landmarks.landmark[10].y,
        landmarks.landmark[16].y < landmarks.landmark[14].y,
        landmarks.landmark[20].y < landmarks.landmark[18].y
    ]
    pulgar_izq = landmarks.landmark[4].x < landmarks.landmark[3].x
    pulgar_der = landmarks.landmark[4].x > landmarks.landmark[3].x
    return all(dedos_abiertos) and (pulgar_izq or pulgar_der)

def detectar_punta_indice(hand_landmarks, width, height):
    x = int(hand_landmarks.landmark[8].x * width)
    y = int(hand_landmarks.landmark[8].y * height)
    return x, y

def predecir_palabra(texto, modelo):
    palabras = texto.strip().split()
    if not palabras:
        return ""
    prefijo = palabras[-1].lower()
    opciones = set()
    for clave, siguientes in modelo.items():
        for palabra, _ in siguientes.items():
            if palabra.startswith(prefijo):
                opciones.add(palabra)
    return sorted(opciones)[:3]  # Máximo 3 sugerencias

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.flip(frame, 1)
    height, width, _ = frame.shape
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    resultado = hands.process(rgb)
```

```python
def detectar_gesto_palma_abierta(landmarks):
    dedos_abiertos = [
        landmarks.landmark[8].y < landmarks.landmark[6].y,
        landmarks.landmark[12].y < landmarks.landmark[10].y,
        landmarks.landmark[16].y < landmarks.landmark[14].y,
        landmarks.landmark[20].y < landmarks.landmark[18].y
    ]
    pulgar_izq = landmarks.landmark[4].x < landmarks.landmark[3].x
    pulgar_der = landmarks.landmark[4].x > landmarks.landmark[3].x
    return all(dedos_abiertos) and (pulgar_izq or pulgar_der)

def detectar_punta_indice(hand_landmarks, width, height):
    x = int(hand_landmarks.landmark[8].x * width)
    y = int(hand_landmarks.landmark[8].y * height)
    return x, y

def predecir_palabra(texto, modelo):
    palabras = texto.strip().split()
    if not palabras:
        return ""
    prefijo = palabras[-1].lower()
    opciones = set()
    for clave, siguientes in modelo.items():
        for palabra, _ in siguientes.items():
            if palabra.startswith(prefijo):
                opciones.add(palabra)
    return sorted(opciones)[:3]  # Máximo 3 sugerencias

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.flip(frame, 1)
    height, width, _ = frame.shape
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    resultado = hands.process(rgb)
```


P
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
BORR 0 1 2 3 4 5 6 7 8 9
para
pasa
pel??cula


G
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
BORR 0 1 2 3 4 5 6 7 8 9
gran
gusta

Conclusiones y Mejoras Futuras

El prototipo del teclado virtual controlado por la mano demostró ser funcional y práctico, integrando tanto accesibilidad como capacidades inteligentes de predicción. No obstante, existen algunas limitaciones a considerar. La precisión del modelo depende mucho de la iluminación y de la estabilidad del gesto del usuario, por lo que condiciones desfavorables pueden generar fallos en la detección.

Además, aunque el modelo de trigramas es ligero y eficiente, no tiene la capacidad de aprender nuevas palabras en tiempo real, ni adaptarse al contexto como lo haría un modelo más avanzado como GPT.

Como mejoras futuras, se propone:

- Entrenar el modelo con un corpus más amplio y variado.
- Añadir aprendizaje continuo para adaptarse al vocabulario del usuario.
- Permitir configuraciones personalizadas del teclado (idioma, tamaño, tiempo de selección).
- Incluir más gestos para realizar funciones como copiar, pegar o enviar texto.
- Optimizar el rendimiento para reducir aún más el retardo entre la selección y la escritura.

Con estas mejoras, el teclado podría utilizarse incluso en dispositivos móviles o para accesibilidad en personas con discapacidades motrices.