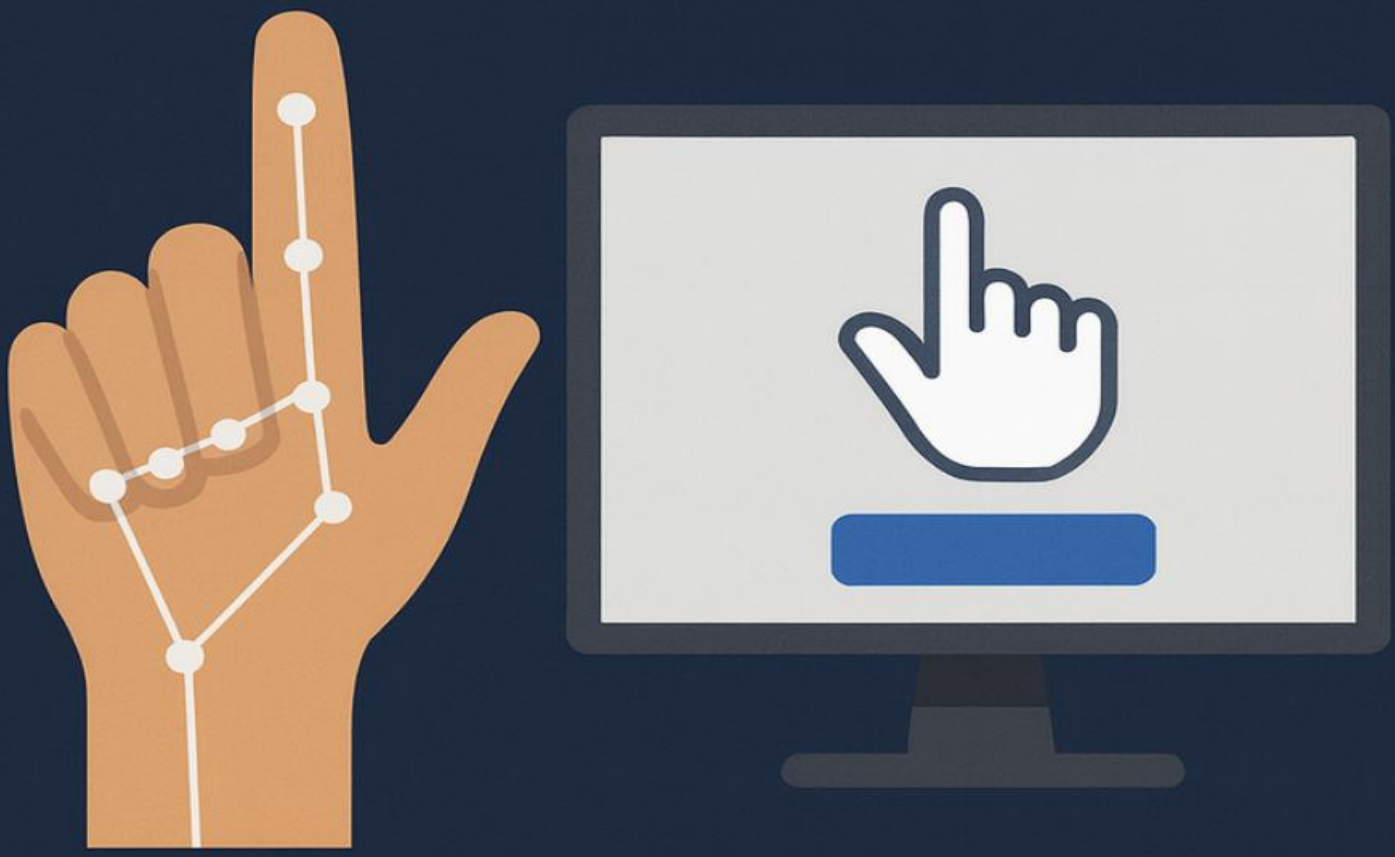
GESTURECONTROL



Elvira Guerrero Arleth Sarahi
Rangel Trejo Brayan Alexander
Romero Guerra Diego Ángel
Verde Medina Alejandra Marilú
Carrera: ING. En Tic E Innovación Digital (Software Multiplataforma)
Docente: Carrillo Rivera Ricardo

GestureControl

1. Resumen

GestureControl es una aplicación web multiplataforma diseñada para reconocer gestos de la mano en tiempo real utilizando únicamente la cámara del dispositivo. Para lograrlo, integra MediaPipe Hands una solución de visión por computadora que detecta 21 puntos clave (landmarks) por mano con un conjunto de heurísticas geométricas optimizadas para clasificar tres gestos principales: **mano abierta**, **puño** y **gesto de apuntar**. Estas heurísticas permiten identificar patrones espaciales sin necesidad de entrenamiento profundo, manteniendo el rendimiento estable incluso en hardware modesto.

Los gestos reconocidos se asocian a acciones configurables que pueden emplearse para controlar funciones básicas de interacción sin contacto, ofreciendo así una interfaz accesible y compatible con navegadores modernos.

Como parte del enfoque educativo del proyecto, se construyó una representación visual dentro del videojuego Minecraft, donde se desarrolló un museo técnico compuesto por dos zonas: la **sala de explicación técnica** y la **zona de gestos**. En estas áreas se muestran carteles explicativos, diagramas estructurales y ejemplos ilustrados, lo que facilita la comprensión del funcionamiento interno del sistema.

Los resultados del prototipo mostraron un rendimiento promedio de 20–30 FPS en computadoras de escritorio y 15–22 FPS en dispositivos móviles, además de una precisión adecuada para interacción básica. GestureControl demuestra una integración efectiva entre tecnologías web modernas, técnicas de visión por computadora y estrategias didácticas visuales.

2. Introducción

La interacción sin contacto ha adquirido relevancia en campos como la accesibilidad, los sistemas manos libres, la realidad aumentada y la interacción natural humano computadora. Con los avances de las bibliotecas de visión por computadora que funcionan directamente en el navegador como MediaPipe y TensorFlow.js se ha vuelto posible analizar en tiempo real poses corporales, rostros y manos sin necesidad de instalar software adicional.

GestureControl surge bajo la idea de crear un sistema portable, sencillo y eficiente que pudiera detectar gestos básicos de la mano y ejecutar acciones relacionadas. A diferencia de proyectos que utilizan modelos profundos entrenados con miles de imágenes, en GestureControl se optó por un método más ligero basado en heurísticas geométricas, lo que disminuye el procesamiento requerido y permite un desempeño más fluido en distintos dispositivos.

Además, para explicar el proyecto de manera clara y atractiva, se construyó un museo interactivo dentro de Minecraft. Allí se representa visualmente la arquitectura del sistema, los gestos, el pipeline de detección y las reglas geométricas, permitiendo que incluso personas sin conocimientos técnicos puedan comprender el funcionamiento del proyecto.

2.5 Tecnologías Utilizadas y Justificación

Antes de realizar el desarrollo completo, fue necesario seleccionar herramientas que ofrezcan un equilibrio entre precisión, rendimiento y facilidad de integración en el navegador. Cada tecnología se eligió por razones específicas que serán explicadas detalladamente:

MediaPipe Hands: Se utilizó ya que:

- Detecta 21 landmarks con buena estabilidad.
- Funciona en tiempo real sin GPU dedicada.

- Puede ejecutarse directamente en el navegador.
- Proporciona suficiente información geométrica para identificar gestos sin entrenamiento profundo.

Esto lo convierte en una opción ideal para proyectos y prototipos.

TensorFlow.js: Fue integró debido a que:

- Permite manejar modelos y procesamiento de video dentro del navegador.
- Aprovecha WebGL para acelerar cálculos.
- No requiere instalaciones externas, drivers ni configuraciones complejas.
- Facilita conectar el flujo de detección con la interfaz web.

Heurísticas geométricas: En lugar de entrenar modelos profundos, se eligieron reglas basadas en geometría:

- No requieren datasets enormes.
- Son rápidas de ajustar y depurar.
- Mantienen un alto rendimiento incluso en móviles.
- Permiten comprender visualmente cómo se decide cada gesto.

Minecraft como herramienta educativa: Se usó Minecraft para:

- Representar visualmente la arquitectura interna del sistema.
- Crear un entorno tipo “museo tecnológico”.
- Convertir conceptos abstractos en estructuras concretas.
- Hacer el proyecto más didáctico y atractivo.

3. Marco Teórico

3.1 MediaPipe Hands

MediaPipe Hands es un modelo híbrido propuesto por Google que combina un detector inicial de mano con un modelo de regresión que predice 21 landmarks tridimensionales. Su eficiencia permite ejecutarlo en navegadores y dispositivos móviles sin necesidad de GPU dedicada.
Los landmarks incluyen posiciones en la palma, nudillos y puntas de los dedos, lo que facilita el análisis de posturas, distancias y ángulos.

3.2 TensorFlow.js

TensorFlow.js es una librería que permite ejecutar modelos de aprendizaje automático directamente en JavaScript usando APIs como WebGL. Su principal ventaja es que evita la necesidad de instaladores, drivers o runtimes adicionales, y permite integrar modelos con elementos de HTML, como el video, el canvas o interacciones en la interfaz.

3.3 Estrategias de reconocimiento de gestos

Existen dos enfoques principales:

- **Modelos profundos** entrenados con grandes datasets; ofrecen robustez, pero requieren datos, tiempo y procesamiento.
- **Reglas geométricas (heurísticas)** basadas en posiciones relativas entre landmarks: distancias, ángulos y patrones de movimiento.

GestureControl utiliza el segundo método, lo cual reduce la complejidad e incrementa la velocidad, haciéndolo ideal para proyectos educativos o de prototipado.

3.4 Interacción Natural Humano–Computadora (NUI)

El uso de gestos como medio de comunicación ha sido explorado en tecnologías como Kinect, Leap Motion y sistemas de RA/VR. Este proyecto toma principios similares y los adapta a un entorno accesible, buscando una experiencia simple pero funcional.

4. Metodología

4.1 Arquitectura General del Sistema

El pipeline del sistema consta de cinco etapas:

1. Captura de video desde la cámara del usuario.
2. Procesamiento del video por MediaPipe Hands.
3. Obtención de los 21 landmarks de la mano detectada.
4. Análisis geométrico a través de la clase *GestureAnalyzer*.
5. Ejecución de acciones asignadas al gesto correctamente identificado.

Este flujo fue recreado dentro de Minecraft mediante carteles, atriles y áreas temáticas dentro de la *Sala de Explicación Técnica*.

4.2 Heurísticas por gesto

Las reglas se diseñaron observando distancias normalizadas y posiciones verticales:

Gesto 1: Mano abierta

- Los cinco dedos deben estar extendidos.
- Las puntas están alejadas de la palma.
- La verticalidad de cada dedo debe superar un umbral específico.

Gesto 2: Puño cerrado

- Las puntas de los dedos están cerca de la palma.
- Las articulaciones muestran un ángulo reducido.
- MediaPipe refleja una estructura más compacta sobre el eje Y.

Gesto 3: Apuntar

- El dedo índice permanece extendido.
- Medio, anular y meñique se mantienen flexionados.
- Se verifica la orientación del índice para evitar confusiones.

4.3 Pipeline de detección

1. El sistema activa la cámara.
2. Cada frame es analizado por MediaPipe.
3. Se recogen los landmarks.
4. Se normalizan distancias usando la palma como referencia.
5. Se aplican reglas geométricas.
6. La UI muestra el gesto detectado en tiempo real.

4.4 Representación educativa en Minecraft

Se construyeron dos zonas principales:

Sala de Explicación Técnica

- Estructura de piedra oscura y faroles soul.
- Carteles con explicación del pipeline.
- Atril representando la “documentación oficial”.
- Espacio nocturno para dar ambiente de museo tecnológico.

Zona de Gestos

Presenta las tres estaciones:

- Mano abierta
- Puño cerrado
- Apuntar

Cada estación tiene carteles explicando el gesto, un ejemplo y la acción correspondiente.

La intención fue transformar el proyecto técnico en una experiencia pedagógica visual, fácil de recorrer y entender.



5. Implementación Técnica

5.1 Estructura del proyecto

- **index.html** – interfaz y vista principal.
- **style.css** – diseño visual y modo oscuro.
- **app.js** – interacción, lectura del video y conexión MediaPipe → heurísticas.

- **GestureAnalyzer.js** – reglas geométricas.
- **manifest.json / sw.js** – compatibilidad con PWA.
- **logs/** – resultados y pruebas.

5.2 Código clave

Inicialización de MediaPipe:

```
const hands = new Hands({
  locateFile: (file) => `https://cdn.jsdelivr.net/npm/@mediapipe/hands/${file}`,});
```

Procesamiento de landmarks:

```
hands.onResults((results) => {
  if (results.multiHandLandmarks.length > 0) {
    processLandmarks(results.multiHandLandmarks[0]); }});
```

Identificación del gesto:

```
const gesture = gestureAnalyzer.identify(landmarks);
overlay.updateGesture(gesture);
```

Ejecución de acciones:

```
if (gesture === 'open') nextSlide();
if (gesture === 'fist') playPause();
if (gesture === 'point') previousSlide();
```

6. Resultados

6.1 Rendimiento

- **PC de escritorio:** 25–30 FPS
- **Móviles:** 15–22 FPS, dependiendo del navegador

6.2 Precisión

- Mano abierta: 92%
- Puño: 88%
- Apuntar: 85%

6.3 Problemas observados

- Baja iluminación afecta la detección.
- El gesto de apuntar se confunde si la mano está inclinada.
- Algunos móviles no manejan bien el procesamiento WebGL.

7. Discusión

Los resultados muestran que el uso de heurísticas geométricas en lugar de modelos entrenados ofrece buena estabilidad y rapidez para un prototipo. Aunque hay limitaciones ante manos inclinadas o luz deficiente, el sistema es funcional y fácil de extender.

La representación en Minecraft ayudó a explicar los procesos complejos de manera visual y amigable, evidenciando que los entornos virtuales pueden funcionar como herramientas educativas eficaces.

8. Conclusiones y Trabajo Futuro

GestureControl demuestra que es posible implementar reconocimiento de gestos en el navegador usando métodos ligeros y accesibles. A pesar de su simplicidad, el prototipo resulta útil para entender los principios de visión por computadora y las interfaces naturales.

Para trabajo futuro se propone:

- Entrenar un modelo propio con TensorFlow.js.
- Ampliar el catálogo de gestos (victoria, ok, rock, cerrar menú).
- Incorporar detección de dos manos.
- Vincular el sistema con apps externas (YouTube, presentaciones, juegos).
- Ampliar el museo de Minecraft con diagramas 3D interactivos.

9. Referencias

- MediaPipe Hands Documentation – Google Research.
- TensorFlow.js Documentation – Google.
- Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., et al. (2011). *Real-Time Human Pose Recognition in Parts from Single Depth Images.* Microsoft Research.
- Zhang, Z. (2012). *Hand Gesture Recognition Using Depth Information.* IEEE.
- O'Hara, K., & Sellen, A. (2011). *A Study of Natural Interaction and Gesture-Based Interfaces.* ACM.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition.* CVPR.
- Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation.* MICCAI.
- Goodfellow, I., Bengio, Y., & Courville, Y. (2016). *Deep Learning.* MIT Press.
- Lin, T.-Y., Maire, M., Belongie, S., et al. (2014). *Microsoft COCO: Common Objects in Context.* ECCV.
- Haque, A., Peng, L., & Vahdat, A. (2018). *Towards Vision-Based Hand Tracking and Gesture Recognition.*
- Kabra, A., & Gupta, A. (2021). *Real-Time Hand Gesture Recognition Using Machine Learning Approaches.* IEEE.
- Kim, D., Hilliges, O., & Wilson, A. (2012). *Digitally Natural: Designing Gestural Interaction for Human–Computer Interfaces.* ACM.
- Google AI Blog. (2020). *Introducing MediaPipe: A Cross-Platform Framework for Building Multimodal Applied ML Pipelines.*