Visión por Computador

Gaze Controlled User Interface.

Using low resource consuming tools to control a user interface in a constrained environment with a laptop webcam.

Cristian Marrero Vega

Javier Gómez Falcón

Motivación

El presente proyecto surge de la necesidad de explorar y desarrollar un sistema de seguimiento ocular (gaze tracking) básico utilizando recursos limitados. El objetivo principal es determinar el máximo rendimiento posible que se puede alcanzar sin recurrir a modelos pre-entrenados o conjuntos de datos extensos. Para lograr esto, se centrará en un modelo de calibración simple y optimizado, diseñado específicamente para entornos con restricciones de hardware.

Este enfoque es especialmente relevante en contextos donde el acceso a grandes capacidades de procesamiento o almacenamiento es limitado. Se busca demostrar que es posible implementar un sistema de seguimiento ocular funcional y razonablemente preciso incluso en dispositivos con recursos restringidos.

El proyecto conlleva un seguimiento de las técnicas de seguimiento ocular existentes, con un enfoque particular en aquellas que son adecuadas para entornos con recursos limitados. Se explorarán diferentes algoritmos de detección de características faciales y oculares, así como métodos de calibración y seguimiento de la mirada.

Se desarrollará un prototipo de sistema de seguimiento ocular basado en los resultados de la investigación. Este prototipo se implementará y probará en un entorno con recursos limitados, como un dispositivo móvil o una computadora de baja potencia. Se evaluará el rendimiento del sistema en términos de precisión, velocidad y robustez.

Objetivo de la propuesta

El objetivo principal es la creación de un sistema de gaze tracking funcional que permita controlar el cursor en una pantalla utilizando la mirada. Se busca lograr una interacción básica con el ordenador, como la lectura de artículos, navegación web y cierre de ventanas, utilizando la información visual obtenida de la webcam a través de un proceso básico de calibración y un modelo de regresión lineal para inferir la información.

Descripción técnica del trabajo realizado

El sistema se compone de tres módulos principales: detección de ojos, calibración y predicción de la mirada.

Descripción de los módulos.

Detección de ojos: Se utiliza la librería MediaPipe para la detección de puntos de referencia faciales (landmarks). Se extraen las coordenadas de los landmarks relevantes para el iris y se normalizan con respecto a una caja delimitadora que abarca ambos ojos.

Calibración: Se realiza una calibración en 25 puntos distribuidos en la pantalla. Se muestra un punto rojo en cada posición de calibración durante un tiempo determinado, mientras se capturan frames de la webcam para registrar las características del ojo.

Predicción de la mirada: Se utiliza un modelo de regresión lineal para mapear las características del ojo a las coordenadas de la pantalla. Se optó por este modelo debido a su simplicidad y buen rendimiento en las pruebas realizadas.

Descripción de funciones principales.

`calibrate()` (calibration.py): Esta función realiza la calibración del sistema. Muestra puntos en la pantalla y captura la imagen del usuario mirando a esos puntos. Asocia las coordenadas de la pantalla con las características del ojo.

`get_eye_features()` (detection.py): Extrae las características del ojo (coordenadas de los landmarks) de un frame utilizando MediaPipe.

`train_model()` (model.py): Entrena un modelo de regresión lineal con los datos de calibración.

`predict_gaze()` (model.py): Predice la posición de la mirada en la pantalla a partir de las características del ojo.

`smooth_position()` (main.py): Suaviza la posición del cursor para evitar movimientos bruscos.

`scroll_based_on_cursor_position()` (cursor_functions.py): Implementa la función de scroll en función de la posición del cursor en la pantalla.

`detect_wink()` (detection.py): Detecta si el usuario ha guiñado un ojo, para cerrar la ventana activa.

Fuentes y tecnologías utilizadas

Entorno de desarrollo:

-Visual Studio Code

Lenguajes y librerías:

- **Python**
  - Mediapipe
  - OpenCV
  - Scikit-learn
  - NumPy
  - PyAutoGUI
  - PyGetWindow

Cómo usar la aplicación.

Instalación de librerías: Asegúrate de tener instaladas las librerías mencionadas en el punto anterior.

1. **Ejecución:** Ejecuta el archivo `main.py` para iniciar la aplicación.

2. **Calibración:** Aparecerán 25 puntos rojos en la pantalla. Mira fijamente a cada punto durante el tiempo indicado. La aplicación capturará la imagen de tus ojos al final de cada punto, por lo que es crucial mantener la mirada fija en el punto hasta que desaparezca.

3. **Control:** Una vez completada la calibración, podrás controlar el cursor con la mirada. La aplicación detectará el movimiento de tus ojos y moverá el cursor en

consecuencia.

Recuerda:

- Mantén tu rostro visible para la cámara web.
- La precisión del control depende de la calidad de la calibración.
- Si experimentas problemas, intenta recalibrar el sistema.

Procedimiento de Investigación.

El desarrollo del sistema Gaze Tracker implicó una fase exhaustiva de investigación y experimentación para determinar la mejor combinación de tecnologías y modelos que permitieran un seguimiento ocular preciso y eficiente, dadas las limitaciones de recursos, incluyendo el uso de una webcam de baja resolución (0.307 megapíxeles) que restringía la cantidad de información visual disponible.

Se exploraron diferentes alternativas para la detección de la mirada y la predicción de coordenadas, buscando un equilibrio entre precisión, eficiencia y complejidad. Se prefirió el desarrollo de un sistema propio en lugar de la utilización de modelos pre-entrenados, como el modelo L2CS-net proporcionado por Roboflow, que requería un servidor local para realizar peticiones HTTP, lo que añadía complejidad a la implementación.

Detección de la Mirada:

Se exploraron dos modelos principales para la detección de la mirada:

1. **Mediapipe:** Esta librería, de código abierto, ofrece un conjunto de herramientas para el análisis facial, incluyendo la detección de puntos de referencia (landmarks) en el rostro. Se evaluó su precisión y rendimiento en la detección de la posición de los ojos y la pupila.

2. **L2CS-net:** Este modelo, basado en redes neuronales convolucionales, se especializa en la detección de landmarks faciales con alta precisión. Se consideró su uso a través de la plataforma Roboflow para la inferencia, pero se descartó debido a la necesidad de un servidor local para realizar peticiones HTTP.

Modelo de Regresión:

Se probaron diferentes modelos de regresión para mapear las características del ojo a las coordenadas de la pantalla:

1. **Regresión Lineal:** Un modelo simple y eficiente, que se evaluó como punto de partida debido a su baja complejidad computacional.

2. **Gradient Boosting Regression:** Un modelo más complejo, que combina múltiples árboles de decisión para mejorar la precisión de la predicción.

3. **Random Forest Regressor:** Similar al Gradient Boosting, pero con una estructura de bosque aleatorio que reduce el overfitting.

4. **Red Neuronal Convolucional:** Se implementó una red neuronal convolucional simple con 3 capas para evaluar su capacidad de predicción de la mirada.

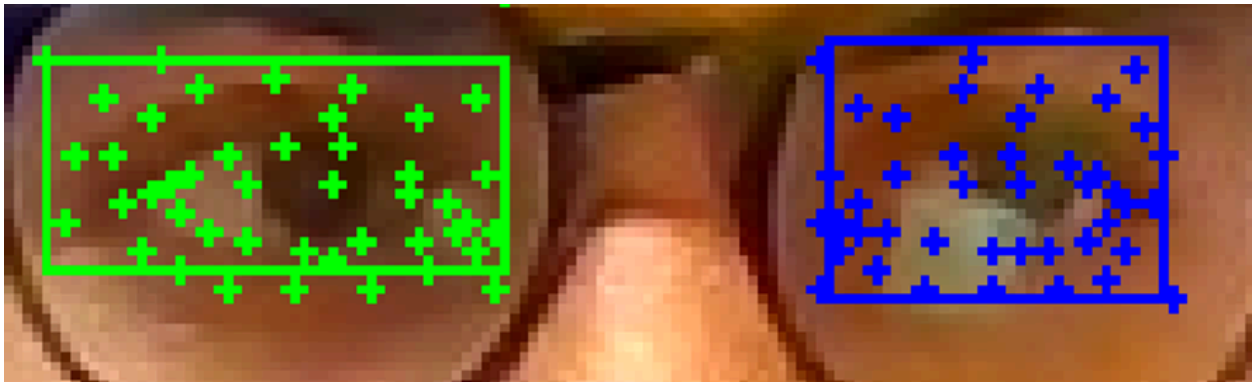
Pruebas y Evaluación:

Se realizaron pruebas exhaustivas con diferentes combinaciones de modelos de detección y regresión, variando también el número de landmarks utilizados y el método de normalización de coordenadas. Se evaluó el rendimiento de cada combinación en términos de precisión, estabilidad y robustez frente a variaciones en la posición de la cabeza e iluminación.

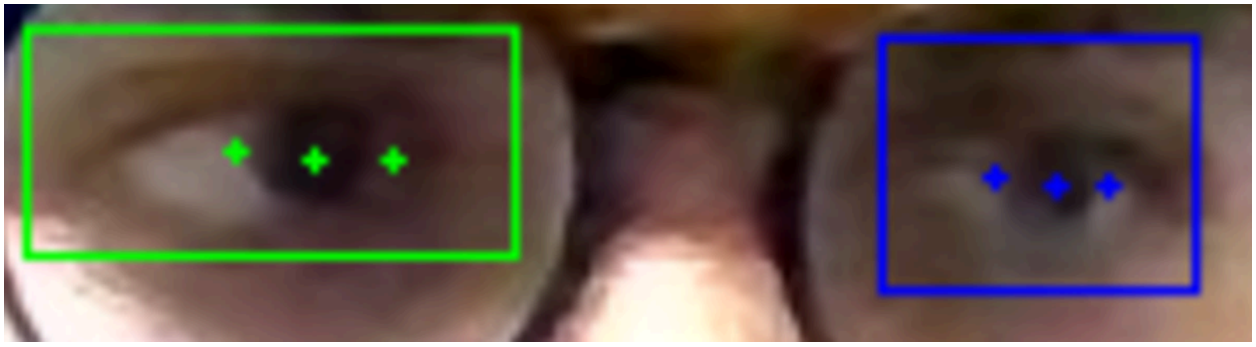
Resultados:

- **Mediapipe** se seleccionó como el modelo de detección de la mirada, debido a su buen equilibrio entre precisión y eficiencia.
- **Regresión lineal** resultó ser el modelo más efectivo para la predicción de la mirada, a pesar de su simplicidad. Los modelos más complejos, como Gradient Boosting, Random Forest y la red neuronal convolucional, no mostraron una mejora significativa en la precisión y en algunos casos presentaron overfitting o dificultades para generalizar con la cantidad limitada de datos de entrenamiento, especialmente considerando la baja resolución de la webcam.

- Se determinó que el uso de **3 landmarks** en cada ojo, normalizados con respecto a una **caja delimitadora** que abarca ambos ojos, ofrecía el mejor rendimiento en términos de precisión y estabilidad. Antes de determinar que este era el número ideal de landmarks, se realizaron las siguientes pruebas:



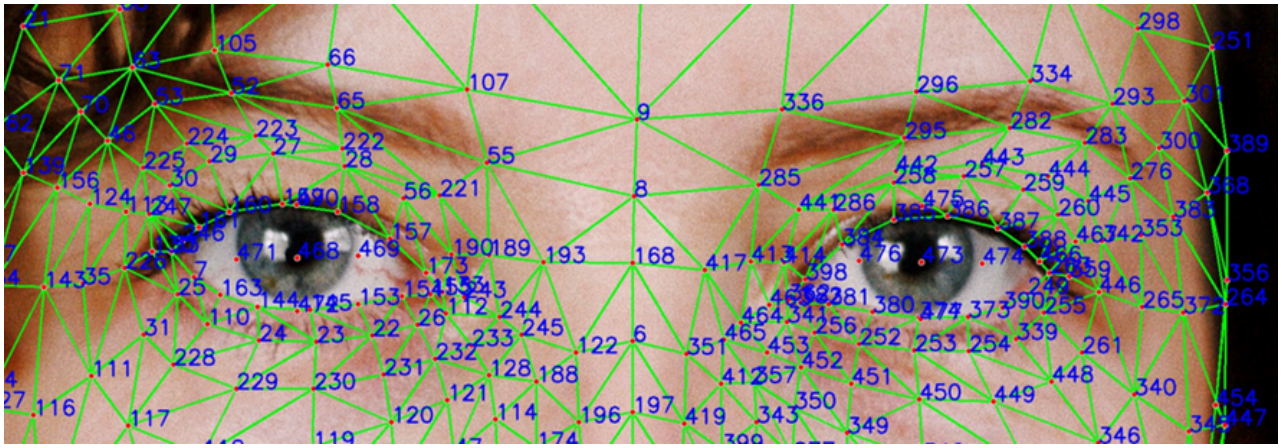
1. **Prueba** con **todos** los landmarks dentro de una Bounding Box que rodea cada ojo. Se usaba la posición relativa dentro de cada bounding box.



2. **Prueba** con **3** landmarks dentro de dos Bounding Box, una para cada ojo. Se sigue usando la posición relativa de cada landmark dentro de la Bounding Box para realizar la regresión.

3. **Prueba** con 7 landmarks en cada ojo, usando una posición respecto a todo el frame, no respecto a una Bounding Box. Probado también con una posición normalizada dentro de una Bounding Box.

4. **Prueba** con 3 landmarks en cada ojo, usando las coordenadas normalizadas dentro de solamente una caja delimitadora como la mostrada en la siguiente foto, usando los landmarks de Mediapipe **21 y 447**.



Zona delimitadora para normalizar las coordenadas de los 3 landmarks sacados de cada ojo. Este opción fue la que se decidió que era mejor frente a todas las pruebas anteriores y posteriores que se habían hecho.

Conclusión

A través de este proceso iterativo de investigación, prueba y error, se concluyó que la combinación de Mediapipe para la detección de la mirada y la regresión lineal para la predicción de coordenadas ofrecía el mejor rendimiento para el sistema Gaze Tracker, dadas las restricciones de recursos, la necesidad de una configuración simple y la limitada información proporcionada por la webcam de baja resolución.

El proceso de investigación fue algo frustrante, ya que a pesar de probar modelos complejos como redes neuronales y modelos de regresión ponderada, la simple conclusión fue que un modelo de regresión lineal era el más adecuado para desarrollar esta tarea de manera propia, sin usar muchos recursos ya existentes. Además, la inversión de tiempo en investigar qué modelo era mejor para la detección de la mirada fue mucho mayor de el que se estimó.

Se ha logrado desarrollar un sistema de gaze tracking básico que permite controlar el cursor en pantalla con la mirada. Se ha demostrado que es posible obtener resultados aceptables con un modelo de calibración simple y una cantidad reducida de datos de entrenamiento.