Control de Interfaz de Usuario Mediante Detección de Parpadeo Ocular

Autores:
Adiane Cueto Portuondo, Universidad de las Ciencias Informáticas, adianecp@uci.cu
Abel Flores Torres, Centro de Inmunología Molecular, a2489261@gmail.com

Resumen

Este proyecto aborda la necesidad de métodos alternativos e intuitivos para la interacción con computadoras, particularmente relevante para usuarios con movilidad reducida o en entornos que demandan una interfaz sin contacto. La motivación principal fue desarrollar una solución accesible que utilizara hardware de consumo masivo para el control del ratón. La metodología empleada se centra en la visión por computadora, utilizando la biblioteca OpenCV para el procesamiento de video y la biblioteca Dlib para la detección precisa de puntos de referencia faciales en tiempo real. A partir de estos puntos, se calcula la Relación de Aspecto del Ojo (EAR) para monitorear el estado de apertura y cierre de cada ojo. Se implementó una lógica basada en umbrales y contadores para diferenciar entre parpadeos del ojo izquierdo, derecho y ambos ojos simultáneamente. Cada tipo de parpadeo se mapea a una acción específica del ratón (clic izquierdo, clic derecho y doble clic, respectivamente) mediante la librería pyautogui. Los resultados demuestran una implementación funcional capaz de ofrecer un control básico del ratón a través de parpadeos, validando la viabilidad de esta interfaz sin contacto y de bajo costo.

Introducción

En la era digital actual, la interacción con las computadoras es fundamental. Sin embargo, los métodos tradicionales de entrada, como el teclado y el ratón, pueden ser inaccesibles para individuos con ciertas discapacidades motoras o imprácticos en situaciones que requieren higiene estricta o manos libres. El problema central que aborda este proyecto es la limitación de las interfaces convencionales para proporcionar un control de usuario versátil y accesible en diversos contextos. La necesidad de interfaces hombre-máquina (HMI) que sean intuitivas, de bajo costo y no invasivas es creciente, abriendo la puerta a soluciones basadas en la visión por computadora.

Nuestra motivación para desarrollar este proyecto surge del interés en explorar las capacidades de la visión por computadora para mejorar la accesibilidad y la interacción del usuario. Buscamos ofrecer una alternativa práctica para el control básico del ratón,

aprovechando la disponibilidad general de las cámaras web y las librerías de procesamiento de imágenes de código abierto. Este enfoque permite la simulación de clics del ratón de una manera discreta y natural, utilizando una acción tan común como el parpadeo de los ojos.

El input a nuestro algoritmo es una secuencia de fotogramas de video en tiempo real, capturados desde una cámara web estándar. Cada fotograma es preprocesado para facilitar el análisis. De cada fotograma, el sistema extrae las coordenadas de 68 puntos de referencia faciales, con un enfoque específico en los 6 puntos que definen la forma de cada ojo. Esta información es crucial para el cálculo de la Relación de Aspecto del Ojo (EAR) [1]. Nuestro algoritmo utiliza el EAR para determinar el estado de apertura o cierre de cada ojo. Basado en la detección de parpadeos específicos (ojo izquierdo, ojo derecho o ambos ojos simultáneamente), el sistema está diseñado para output un evento simulado de ratón: un clic izquierdo, un clic derecho o un doble clic, respectivamente, controlando así el ratón del sistema operativo.

Trabajo Relacionado

El campo de las interfaces humano-computadora sin contacto ha visto una considerable investigación, especialmente en el control ocular [6]. Los enfoques previos se pueden categorizar principalmente en sistemas basados en seguimiento de la mirada (gaze tracking) y sistemas basados en la detección de parpadeos.

Los sistemas de seguimiento de la mirada, a menudo considerados el "estado del arte" para el control ocular completo, utilizan cámaras de alta velocidad y algoritmos complejos para determinar con precisión el punto exacto al que el usuario está mirando en la pantalla. Esto permite no solo la detección de clics sino también el movimiento continuo del cursor [6]. Ejemplos notables incluyen dispositivos comerciales y soluciones de investigación que emplean técnicas como la iridología o el seguimiento del reflejo corneal. Sin embargo, estos sistemas suelen ser costosos, requieren hardware especializado y a menudo son sensibles a las condiciones de iluminación o a movimientos de cabeza del usuario, lo que puede limitar su accesibilidad y usabilidad general.

Por otro lado, los sistemas basados en la detección de parpadeos ofrecen una alternativa más accesible y de bajo costo. Estos enfoques se centran en identificar el acto de parpadear como un comando específico, prescindiendo de la necesidad de un seguimiento preciso de la mirada. La tarea de detectar parpadeos ha sido abordada con diversas técnicas, desde el procesamiento de imágenes básicas hasta el aprendizaje automático. Una contribución pionera en este ámbito es el trabajo de Rosebrock (2017) [1] en PyImageSearch, que popularizó el uso de la Relación de Aspecto del Ojo (EAR) calculada a partir de los puntos de referencia faciales de Dlib [4] para una detección de parpadeo robusta y eficiente. Este método es particularmente inteligente por su simplicidad y efectividad, utilizando relaciones geométricas en lugar de algoritmos de aprendizaje complejo, lo que lo hace fácil de implementar con recursos mínimos.

Otros enfoques han explorado el uso de sensores electroencefalográficos (EEG) o electromiográficos (EMG) para detectar las señales asociadas al parpadeo, ofreciendo una

alta precisión pero a expensas de la invasividad y el costo del hardware [7]. Nuestro trabajo se diferencia de estos por su naturaleza no invasiva y su dependencia exclusiva de una cámara web estándar, buscando maximizar la accesibilidad. Se asemeja al enfoque de Rosebrock en la utilización del EAR, pero lo extiende para diferenciar los tipos de clics (izquierdo, derecho, doble) en función del parpadeo de ojos individuales o combinados. Aunque muchas personas aún realizan la tarea de control del ratón manualmente, proyectos como el nuestro buscan democratizar el acceso a interfaces alternativas.

Nuestro trabajo se basa en las metodologías presentadas por Rosebrock (2017) en su artículo sobre detección de parpadeo con OpenCV, Python y Dlib [1], que proporciona una base sólida para el cálculo del EAR y la lógica de detección. Adicionalmente, se consultaron recursos generales de visión por computadora para Python, como los proporcionados por GeeksforGeeks [2], para afianzar conceptos fundamentales.

Conjunto de Datos y Características

En este proyecto, el "conjunto de datos" es el flujo de video en tiempo real capturado directamente desde la cámara web del usuario. A diferencia de los modelos de aprendizaje automático que operan sobre conjuntos de datos estáticos de entrenamiento, validación y prueba, este sistema opera de forma continua y dinámica sobre un flujo de datos en vivo. No hay ejemplos de entrenamiento, validación o prueba en el sentido tradicional, ya que la detección se basa en umbrales predefinidos y no en un modelo entrenado.

Los fotogramas de video capturados son sometidos a un preprocesamiento mínimo pero esencial. Inicialmente, cada fotograma se redimensiona a un ancho fijo de 600 píxeles para estandarizar el tamaño de entrada y optimizar el rendimiento del procesamiento. Posteriormente, los fotogramas se convierten a escala de grises, lo que reduce la complejidad computacional y mejora la robustez de la detección facial, ya que los algoritmos de detección de puntos de referencia a menudo operan de manera más eficiente en imágenes en blanco y negro. No se aplicaron técnicas de normalización o aumento de datos adicionales, dado el enfoque en la detección en tiempo real basada en umbrales.

Las características principales utilizadas en este proyecto se derivan de los puntos de referencia faciales. Utilizando el predictor de forma facial shape_predictor_68_face_landmarks.dat de Dlib [4], el sistema detecta 68 puntos clave en el rostro humano en cada fotograma. De estos, nos centramos específicamente en los 12 puntos (6 por cada ojo) que definen la forma y el contorno de los ojos. Estos puntos son cruciales para el cálculo de nuestra característica principal: la Relación de Aspecto del Ojo (EAR) [1].

La EAR se calcula como una medida de la relación entre la altura y el ancho del ojo. Se utilizan las distancias euclidianas entre los puntos de referencia verticales y horizontales del ojo. La fórmula para el EAR se define como:

$$EAR = \frac{|P_2 - P_6| + |P_3 - P_5|}{2 \cdot |P_1 - P_4|}$$

Donde :

P_1-P_6 son los puntos de referencia 2D de un ojo específico, numerados en orden de reloj desde el punto más a la izquierda. Un valor de EAR alto indica un ojo abierto, mientras que un valor bajo (cercano a cero) indica un ojo cerrado. Esta característica geométrica es altamente efectiva y robusta para la detección de parpadeos [1].

Métodos

Este proyecto se basa en un enfoque de visión por computadora para la detección de parpadeos y la simulación de clics del ratón. Los algoritmos y la lógica propuestos se estructuran en varias etapas: detección facial, extracción de puntos de referencia, cálculo de la Relación de Aspecto del Ojo (EAR) y una lógica de decisión basada en umbrales para los clics.

Detección Facial y Puntos de Referencia: El primer paso en el procesamiento de cada fotograma es la detección del rostro humano. Para ello, se emplea el detector de rostros pre-entrenado de la librería Dlib [4]. Una vez que un rostro es localizado en el fotograma en escala de grises, se utiliza un predictor de forma facial (el modelo shape_predictor_68_face_landmarks.dat) para identificar 68 puntos de referencia faciales clave en el rostro detectado. Estos puntos incluyen la boca, las cejas, la nariz y, crucialmente, los contornos de los ojos.

Cálculo de la Relación de Aspecto del Ojo (EAR): A partir de los 6 puntos de referencia correspondientes a cada ojo (el ojo izquierdo y el ojo derecho), se calcula la Relación de Aspecto del Ojo (EAR) para cada ojo. El EAR es una métrica simple pero efectiva para cuantificar la apertura del ojo [1]. Como se mencionó en la sección anterior, la fórmula es:

$$EAR = \frac{|P_2 - P_6| + |P_3 - P_5|}{2 \cdot |P_1 - P_4|}$$

Donde P_i representa la coordenada (x,y) del i-ésimo punto de referencia del ojo. Un valor de EAR alto indica que el ojo está abierto, mientras que un valor bajo (generalmente por debajo de un umbral) indica que el ojo está cerrado.

Lógica de Detección de Parpadeo: La detección de un parpadeo se determina cuando el valor de EAR de un ojo cae por debajo de un umbral predefinido (EYE_AR_THRESH) durante un número consecutivo de fotogramas (EYE_AR_CONSEC_FRAMES) [1]. Para cada ojo, se mantiene un contador (left_COUNTER y right_COUNTER). Si el EAR de un ojo está por debajo del umbral, su contador aumenta; de lo contrario, se reinicia a cero.

- EYE_AR_THRESH: Este umbral (0.24) define el nivel de cierre del ojo necesario para considerarlo "cerrado".

- EYE_AR_CONSEC_FRAMES: Este valor (3) especifica cuántos fotogramas consecutivos el ojo debe permanecer por debajo de EYE_AR_THRESH para que se registre un parpadeo, ayudando a filtrar el ruido y los parpadeos accidentales.

Lógica de Diferenciación y Cooldown de Clics: Para evitar múltiples activaciones por un único parpadeo prolongado o por fluctuaciones del EAR, se implementa un mecanismo de "cooldown" (CLICK_COOLDOWN_TIME). Después de que se registra un clic, el sistema entra en un estado de enfriamiento durante 1.5 segundos, durante el cual no se procesan nuevos eventos de clic.

La lógica para diferenciar los tipos de clic es la siguiente, priorizando el doble clic:

- Doble Clic: Se detecta si left_COUNTER y right_COUNTER son ambos mayores o iguales a EYE_AR_CONSEC_FRAMES simultáneamente. Esto implica que ambos ojos han estado cerrados durante el número requerido de fotogramas.
- Clic Izquierdo: Si no se detectó un doble clic, se verifica si left_COUNTER es mayor o igual a EYE_AR_CONSEC_FRAMES. Esto indica un parpadeo solo del ojo izquierdo.
- Clic Derecho: Si no se detectó un doble clic ni un clic izquierdo, se verifica si right_COUNTER es mayor o igual a EYE_AR_CONSEC_FRAMES. Esto indica un parpadeo solo del ojo derecho.

Tras cada detección exitosa de clic, el last_click_time se actualiza y los contadores (left_COUNTER, right_COUNTER) se reinician.

Simulación de Clics: Una vez que se identifica el tipo de clic (izquierda, derecha o doble), se utiliza la librería pyautogui [5] para simular el evento de ratón correspondiente en el sistema operativo. pyautogui.leftClick(), pyautogui.rightClick(), y pyautogui.doubleClick(interval=0.1) se utilizan para estas acciones, enviando el clic a la posición actual del cursor del ratón.

Resultados

Los experimentos se realizaron empíricamente para validar la funcionalidad del sistema y optimizar los parámetros clave. El entorno de desarrollo fue un sistema operativo Ubuntu.

Parámetros Seleccionados y Justificación: Los hiperparámetros críticos que se ajustaron fueron EYE_AR_THRESH y EYE_AR_CONSEC_FRAMES.

- EYE_AR_THRESH = 0.24: Este umbral se determinó mediante observación en tiempo real de los valores del EAR de los ojos del usuario en estados de apertura y cierre. Un valor de 0.24 fue seleccionado como un equilibrio óptimo que permite la detección fiable de parpadeos naturales sin ser excesivamente sensible a pequeños movimientos oculares o fluctuaciones de la cámara, minimizando así los falsos positivos.

- EYE_AR_CONSEC_FRAMES = 3: Este valor especifica que el ojo debe permanecer "cerrado" (EAR por debajo del umbral) durante al menos 3 fotogramas consecutivos para ser considerado un parpadeo. Se eligió este valor para asegurar que solo los parpadeos intencionales y sostenidos fueran registrados, proporcionando una mayor robustez contra el ruido de la cámara o parpadeos involuntarios muy breves. Un valor más bajo incrementaba los falsos positivos, mientras que un valor más alto hacía que la detección fuera demasiado lenta o requiriera un esfuerzo excesivo para mantener el ojo cerrado.
- CLICK_COOLDOWN_TIME = 1.5 segundos: Este tiempo de enfriamiento fue crucial para la usabilidad. Se estableció en 1.5 segundos para evitar la detección de múltiples clics por un único parpadeo prolongado o por la rápida reapertura del ojo, lo que mejoró significativamente la experiencia del usuario.

El sistema demostró ser funcional para el control básico del ratón mediante parpadeos. La diferenciación entre clic izquierdo, derecho y doble clic fue generalmente intuitiva una vez que el usuario se acostumbró a la cadencia de parpadeo requerida. La retroalimentación visual en pantalla, que muestra el tipo de clic detectado, fue esencial para el proceso de aprendizaje del usuario y la depuración del sistema.

El algoritmo mostró una alta fiabilidad en condiciones de iluminación uniforme y cuando el usuario mantenía una posición relativamente estable frente a la cámara. La detección de parpadeos distintos para clics individuales y dobles funcionó consistentemente una vez que los umbrales fueron ajustados al usuario. La implementación de pyautogui [5] fue efectiva para simular los eventos de clic en el sistema operativo.

Fallos y Limitaciones:

- Sensibilidad a la Iluminación: El rendimiento disminuyó en condiciones de poca luz o con iluminación inconsistente, lo que podía afectar la precisión de la detección de puntos de referencia faciales y, por ende, el cálculo del EAR.
- Movimiento de la Cabeza: Movimientos bruscos o frecuentes de la cabeza podían desorientar el detector de rostros, resultando en una pérdida temporal de la detección de ojos y, por lo tanto, en la incapacidad de registrar parpadeos.
- Foco de la Ventana: Un hallazgo importante durante las pruebas fue que los clics simulados por pyautogui [5] siempre se dirigían a la ventana que tenía el foco activo. Si la ventana de la cámara de OpenCV (Control de Clic por Parpadeo) estaba activa, los clics se registraban dentro de ella, impidiendo el control de otras aplicaciones. Este comportamiento requirió que el usuario hiciera clic manualmente fuera de la ventana de la cámara para transferir el foco antes de intentar controlar el escritorio u otras aplicaciones.
- Sin Movimiento del Cursor: Como se mencionó, el proyecto se limita a la simulación de clics en la posición actual del cursor, no proporciona un control de movimiento del

cursor. Esto significa que el usuario debe mover el cursor manualmente (con un ratón físico o trackpad) a la ubicación deseada antes de poder hacer clic con los ojos.

No se aplicaron técnicas para mitigar el sobreajuste (overfitting), ya que el sistema es un clasificador basado en umbrales y no un modelo de aprendizaje automático que se entrena con datos. Su rendimiento depende de la precisión de la detección de características y la sintonización de los umbrales.

Conclusiones

Este proyecto ha demostrado exitosamente la viabilidad de implementar un sistema de control de ratón basado en la detección de parpadeo ocular utilizando librerías de visión por computadora de código abierto. La combinación de Dlib para la detección de puntos de referencia faciales, OpenCV para el procesamiento de imágenes y una lógica basada en el EAR y contadores demostró ser efectiva para diferenciar los tipos de clics (izquierdo, derecho, doble) de manera intuitiva. La capacidad de pyautogui para simular acciones del ratón permitió la integración con el sistema operativo. La facilidad de uso y la mínima dependencia de hardware especializado son fortalezas clave de esta implementación.

A pesar del éxito en la demostración del concepto, existen varias áreas para el trabajo futuro:

- Integración de Movimiento del Cursor: La extensión más significativa sería la incorporación de un sistema de seguimiento de la mirada (gaze tracking) o de seguimiento de la cabeza para permitir el movimiento del cursor. Esto transformaría el sistema en una interfaz de control ocular completa, eliminando la necesidad de interacción manual para posicionar el cursor.
- Mejora de la Robustez: Investigar técnicas para mejorar la robustez del sistema frente a variaciones en la iluminación, movimientos de cabeza del usuario y diversas características faciales. Esto podría incluir la adaptación dinámica de los umbrales de EAR o el uso de algoritmos más avanzados de seguimiento.
- Calibración Personalizada: Desarrollar una rutina de calibración inicial para que el sistema se adapte automáticamente a los umbrales de parpadeo individuales de cada usuario, mejorando la precisión y la comodidad.
- Interfaz de Usuario Avanzada: Crear una interfaz de usuario más sofisticada y amigable que proporcione retroalimentación visual mejorada, quizás con un teclado en pantalla controlable por mirada o parpadeo.
- Portabilidad: Explorar la optimización y empaquetado del proyecto para facilitar su despliegue y uso en diferentes sistemas operativos y configuraciones de hardware.

Referencias

[1] Rosebrock, A. (2017). Eye blink detection with OpenCV, Python, and dlib. PyImageSearch Blog. Recuperado de https://pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/

[2] GeeksforGeeks. (n.d.). Python Eye Blink Detection Project. Recuperado de https://www.geeksforgeeks.org/machine-learning/python-eye-blink-detection-project/

[3] Howse, J., & Minichino, J. (2020). Learning OpenCV 4 Computer Vision with Python 3 (3rd ed.). Packt Publishing.

[4] King, D. (2009). Dlib-ML: A machine learning toolkit. Journal of Machine Learning Research, 10, 1755-1758.

[5] Sweigart, A. (2015). Automate the Boring Stuff with Python: Practical Programming for Total Beginners. No Starch Press.

[6] Duchowski, A. T. (2017). Eye Tracking Methodology: Theory and Practice (3rd ed.). Springer.

[7] Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., & Vaughan, T. M. (2002). Brain-computer interfaces for communication and control. Electroencephalography and Clinical Neurophysiology, 113(6), 767-791.