HEAD MOUSE

Camilo Andres Duarte 2420162021
Daniel Rodriguez 2420162008
Universidad de Ibagué
2020

***Resumen*—La presente investigación cuyo objetivo es el diseño, desarrollo y la implementación de un algoritmo con el que se pueda mover el cursor de la pantalla a través de la captura del movimiento facial, y simular así el movimiento del ratón físico. Además, ayudara a las personas con discapacidades motrices u otros problemas a manejar un computador sin necesidad de contacto físico con el dispositivo. La presente investigación se produce a través de técnicas de deteccion facial y/o sus implicaciones.**

***Index Terms*—Python, OpenCV, Mouse, Ratón, Movimiento facial, Discapacidad motriz.**

I. Introducción

En la actualidad existen técnicas de detección facial, detección ocular y sus derivados, las cuales permiten obtener información sobre la mirada debido al gran avance tecnológico ocurrido en los últimos años área activa para los investigadores de visión artificial. Esto se debe a que la localización y el seguimiento de los ojos pueden ser útiles para estimar la dirección de la mirada. Lo cual se usa en múltiples acciones, por ejemplo, en el seguimiento de la mirada ante una imagen para saber que aspectos nos interesan mas de la misma o en la interacción persona-ordenador. También es útil en otras áreas como en psicología, lingüística cognitiva y diseño de productos.[1]
Aunque, el enfoque del proyecto es diferente, es indispensable conocer las técnicas y avances relacionados. Por otro lado La detección de gestos realizados por usuarios se esta convirtiendo en uno de los mas importantes mecanismos para la interacción humano-computador, debido a la naturalidad e intuitividad que ofrece, además de ser un mecanismo llamativo, lo que motiva a las personas a usarlo. Sin embargo, para realizar una detección apropiada de gestos se esta utilizando hardware especializado, el cual puede ser de difícil acceso por costos e infraestructura.[2] Es por eso que nos vemos obligados a indagar alternativas que sean viables para el proyecto.

Usualmente en la universidad y contexto técnico hay una variedad de lenguajes de programación a utilizar. Para este proyecto se decidió utilizar el lenguaje de programación de Python el cual es un excelente lenguaje de ”dirección”para códigos científicos escritos en otros lenguajes. Sin embargo, con herramientas básicas adicionales, Python se transforma en un lenguaje de alto nivel adecuado para el código científico y de ingeniería que a menudo es lo suficientemente rápido como para ser inmediatamente útil, pero también lo suficientemente flexible como para ser acelerado con extensiones y/o librerías adicionales.[3]

Para el diseño del algoritmo es esencial contar con tal programa para utilizar los recursos disponibles que disponen los desarrolladores. Para esto, el uso de NumPy el cual es un paquete de Python que significa “Numerical Python”, es la librería principal para la informática científica, proporciona potentes estructuras de datos, implementando matrices y matrices multidimensionales.[4] También el uso de las librerías Open CV y dlib. Por su parte Open CV (Open Computer Vision Library) que como su nombre en ingles lo indica es una libreria de visión por computadora de código abierto. La biblioteca está escrita en C y C ++ y se ejecuta en Linux, Windows y Mac OS X. Hay un desarrollo activo en interfaces para Python, Ruby, Matlab y otros lenguajes.
OpenCV fue diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real. OpenCV está escrito en C optimizado y puede aprovechar los procesadores multinúcleo. Si desea una mayor optimización automática en las arquitecturas Intel [Intel], puede comprar las bibliotecas de Intel Integrated Performance Primitives (IPP), que consisten en rutinas optimizadas de bajo nivel en muchas áreas algorítmicas diferentes. OpenCV usa automáticamente la biblioteca IPP apropiada en tiempo de ejecución si esa biblioteca está instalada.
Uno de los objetivos de OpenCV es proporcionar una infraestructura de visión por computadora fácil de usar que ayude a las personas a desarrollar rápidamente aplicaciones de visión bastante sofisticadas. La biblioteca OpenCV contiene más de 500 funciones que abarcan muchas áreas de la visión, incluida la inspección de productos de fábrica, imágenes médicas, seguridad, interfaz de usuario, calibración de la cámara, visión estereoscópica y robótica. Debido a que la visión por computadora y el aprendizaje automático a menudo van de la mano, OpenCV también contiene una biblioteca de aprendizaje automático (MLL) completa y de uso general. Esta sublibreria se centra en el reconocimiento y agrupamiento de patrones estadísticos. El MLL es muy útil para las tareas de visión que están en el centro de la misión de OpenCV, pero es lo suficientemente general como para ser utilizado para cualquier problema de aprendizaje automático.[5]
La librería mencionada anteriormente es esencial para combinarse con dlib la cual puede ser usada desde C++ y Python, además, contiene muchos algoritmos de aprendizaje de máquinas, compresión, análisis de imágenes, entre otras, esta librería es de código abierto [6] y se debe compilar para

poder utilizarse junto con Open CV y así lograr detección de cada una de las regiones de la cara. Y Por ultimo, el uso de PyAutoGUI. PyAutoGUI es un módulo Python de automatización de GUI multiplataforma para seres humanos. Se usa para controlar mediante programación el ratón(mouse) y el teclado.[7]

II. Alternativas de solución

II-A. Analizar e interpretar la problemática

La realización del algoritmo exige estudiar diversos métodos con los cuales se podría realizar el proyecto, tal que, se escoja el mas viable enfocado al algoritmo para mover el cursor a través de movimientos faciales y/o gestos. Entre las distintas opciones, análisis y síntesis se permite procesar la información de la temática para llegar a generalizaciones. Siendo así, se revisan las distintas técnicas y métodos existentes en la actualidad. Entre los métodos consultados hay un tipo de interacción que abre las puertas para controlar sistemas sin la necesidad de manipular dispositivos físicos y el usuario podría navegar y controlar un sistema de manera natural. Muchos grupos de investigación se encuentran en la "carrera" por desarrollar el standard para reconocimiento de gestos. Existen alternativas de reconocimiento desde accesorios complejos, como trajes completos, a no invasivos como en el caso de cámaras de profundidad infrarrojas y el Kinect.[8]. Esta posible solución tienen un factor económico elevado, dado que los trajes, cámaras de profundidad infrarroja, y demás actualmente se encuentran a un alto costo. Siguiente a este, se encuentran 3 métodos. El primero es basado en la librería Overflow el cual tiene un enfoque hacia la programación de redes neuronales por medio de matrices o tensores (de 2 dimensiones) los cuales procesa fácilmente. Asi mismo, en la técnica de electro-oculografía (EOG) se usan electrodos que pueden ser invasivos. Aunque la técnica EOG es muy usada por su gran ventaja que es la obtención de buenos resultados. Este sistema puede resultar molesto a un usuario que por un largo periodo de tiempo permanezca realizando pruebas con el.[1] También, la técnica del Análisis de Componentes Independientes (ICA) frente al método tradicional del Análisis de Componentes Principales (PCA) muy utilizadas para entrenar redes neuronales con enfoques principales en robótica y seguridad social.[9]

Todas las anteriores tienen características similares pero en cuanto aplicaciones se obtienen diferentes enfoques y de dificultad avanzada.
Como se menciono anteriormente, un inconveniente importante de los enfoques anteriores es que generalmente imponen implícitamente requisitos demasiado fuertes en la configuración , en el sentido de la pose a la cámara de cara (orientación de la cabeza), resolución de imagen, iluminación, dinámica de movimiento, trajes especiales, cámara de alta resolución, etc.

Asi que, en definitiva se opta por hacer uso de Open CV ya que cuenta con las características mencionadas anteriormente ideales para el desarrollo algoritmo sin los requisitos de los enfoques anteriores y ya que conjunto con la familiarizada librería NumPy, dlib y el módulo de Python "PyAutoGUI", serian las librerías suficientes con las cuales se podrá controlar el movimiento del cursor.

Teniendo en cuenta la técnica llamada Eye-Aspect-Ratio traducida al español como relación de aspecto ocular (EAR) que propone explotar detectores de puntos de referencia faciales de última generación para localizar los ojos y el contorno de los párpados. De los puntos de referencia detectados en la imagen, derivamos la relación de aspecto del ojo (EAR) que se utiliza como una estimación del estado de apertura de los ojos. Dado que el EAR perframe no necesariamente reconoce el ojo cuando se parpadea correctamente, si no que un clasificador que toma en cuenta una ventana temporal más grande de un marco es entrenado.[10]

II-B. Diseño de solución

Una vez el problema ha sido analizado e interpretado, se diseña la solución. Como se menciono anteriormente en el apartado experimental, el método viable para la realización del proyecto y diseño del algoritmo se calculara con una técnica llamada Eye-Aspect-Ratio (EAR), presentada por Soukupová y Čech en su artículo Real-Time Eye Blink Detection using Facial Landmarksde en 2016.[10]
Antes de esto, ya se venia planteando la solución y avances en este proyecto, para esto se hizo uso de solamente de OpenCV el cual mediante una base de datos, analizaba y detectaba la imagen png o imagen en tiempo real en donde encerraba la zona facial con un cuadrado, pero ahora, necesitamos hacer mas funciones para no solo mover el mouse conforme se mueve la zona facial si no también hacer la función del clic derecho e izquierdo. En esta ultima parte, entra en juego la técnica EAR y la base de datos de la librería dlib, debido a que estas permite tener reconocer el rostro, detectar el contorno, hacer landmark o en otras palabras marcar puntos de referencia, en este caso son 68 puntos sobre varias regiones de la cara como por ejemplo la nariz, ojo izquierdo, ojo derecho, boca o cejas, tal cual como se observa en la figura 1 y figura 2.

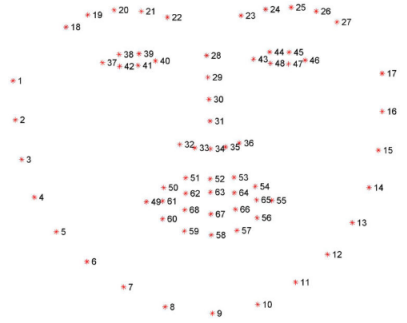


Figura 1. 68 puntos de referencia "Facial landmark"

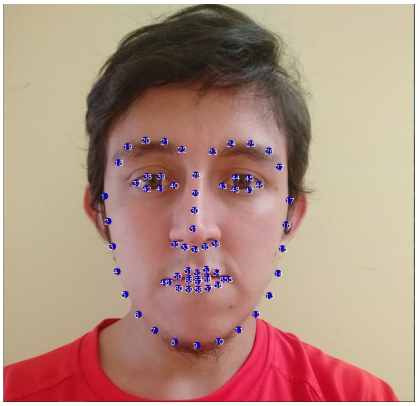
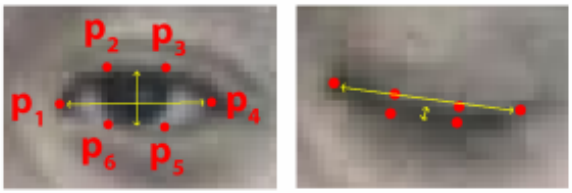


Figura 2. 68 puntos de referencia, Daniel Rodriguez

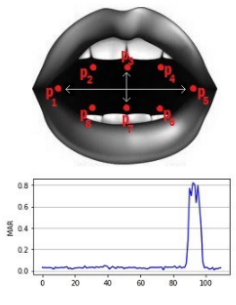
Luego de marcar estos 68 puntos sobre el rostro, mediante las ecuaciones normalizadas planteadas de Eye Aspect Ratio (EAR) tomar la referencia de una región facial para compararlas en tiempo real con el valor inicial normalizado, tal que al momento de hacer un cambio ya sea abriendo o cerrando el ojo, pues este lo interprete como un gesto, como se observa en la figura 3.



$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|},$$

Figura 3. Eye Aspect Ratio (EAR)

Siguiente a esto se utiliza una técnica basada en el mismo principio de relacion-aspecto-ocular llamada Mouth-Aspect-Ratio (MAR) para analizar si se produce un gesto con la boca, como se observa en la figura 4.



$$\text{MAR} = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{2\,\|p_1 - p_5\|}$$

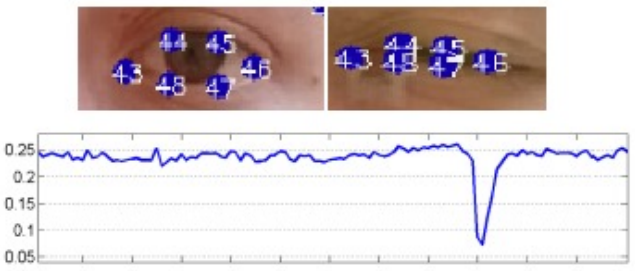
Figura 4. Mouth-Aspect-Ratio (MAR)

Finalmente a través de estas comparaciones, se asigna un propósito a cada gesto como se observa en la figura 5

Figura 5. Gestos para realizar funciones básicas con el cursor

III. Resultados

Teniendo en cuenta la solución se ejecuta el algoritmo el cual está comentado en los archivos anexados.
Este nos arroja los siguientes resultados
Reconocimiento y aplicación de EAR en el ojo



$$EAR = \frac{|p44 - p48| + |p45 - p47|}{2|p43 - p46|}$$

Figura 6. «Facial landmark y tecnica EAR» Ojo de Daniel Rodriguez

Tambien se obtienen los resultados de la figura 7, 8, 9, 10, 11 y 12.

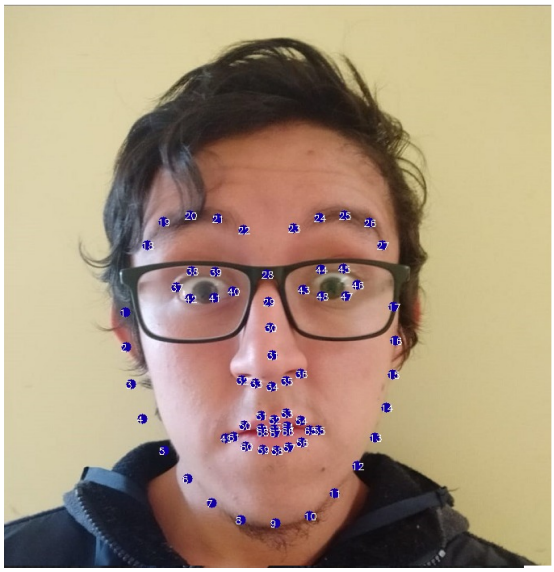


Figura 7. «Facial landmark» Clic Izquierdo

Figura 8. Comportamiento gráfico de ojos de asombro. Clic Izquierdo

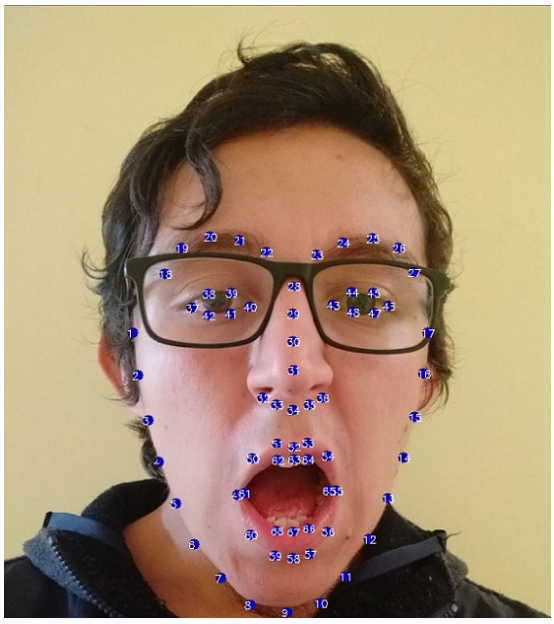


Figura 9. «Facial landmark» Clic Derecho

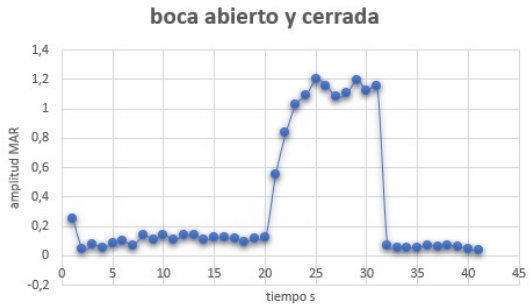


Figura 10. Comportamiento gráfico de boca abierta. Clic Derecho

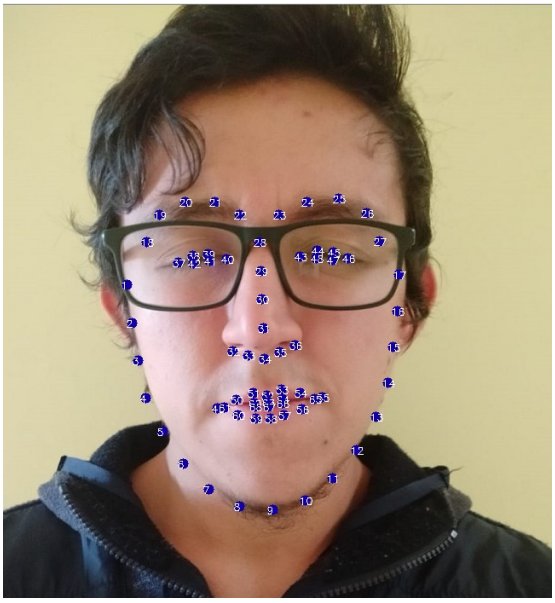


Figura 11. «Facial landmark» Doble Clic

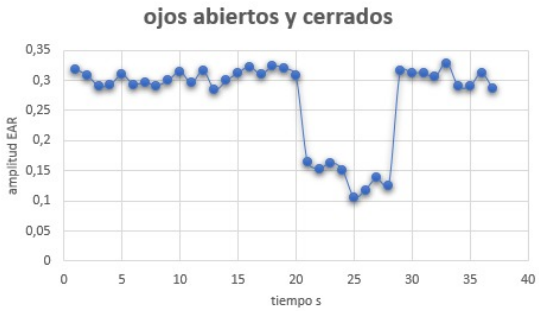


Figura 12. Comportamiento gráfico de ojos cerrados. Doble Clic

IV. Análisis y conclusiones

Se demostró que era posible hallar la posición de distintas partes faciales sin necesidad de una cámara de alta resolución, como lo es la del computador portátil.

Existe una restricción que consiste en no estar a una distancia mayor a 45 cm ni menor a 35 cm de la cámara,

esto por que es posible que la cámara pueda no reconocer un rostro debido a la diferencia de contraste y luz.
Como se esta trabajando a escala de grises, cuando la cara pierde contraste es decir baja la iluminación al sistema se le hace difícil detectar la posición exacta de la cara.

Se evidenció que cuando se ejecuta el algoritmo se necesita de altos o luz blancos sobre todo en las partes mas diminutas de la cara, específicamente en los ojos.
Al usar gafas y tener un contraste elevado, pues las gafas reflejan la luz lo que hace que el programa no detecta bien la posición de los ojos. Si no detecta los ojos, entonces, no detectara el gesto por lo tanto es posible que no efectué la función propuesta.

Entre mas pequeño el objeto a detectar, se evidencia mas ruido en las señal. Por ejemplo en los ojos se evidencian pocos puntos, así que los cambios en la señal son mas ruidosos. En contraste, la boca tiene mas puntos, por lo que los cambios en la señal se evidencian mas pequeños, y al hacer un gesto con la boca pues los picos se detectan fáciles. Cabe destacar, que aunque la señal de los ojos sea mas ruidosa, detectar los picos de esta no sera un problema.

Finalmente el algoritmo funciona de manera eficiente siempre y cuando se cumplan con las posiciones e intensidad de luz, cabe resaltar, que la velocidad del movimiento de la cara al manejar el cursor no deberá ser demasiado veloz.

REFERENCIAS

[1] S. M. Moraleda Moreno, «Analisis del comportamiento de la mirada frente a un ordenador,» Madrid, 2018.

[2] D. C. García Cortes, «Reconocimiento de Gestos de Manos como Mecanismo de interaccion Humano-Computador,» Bogota D.C, 2014.

[3] T. E. Oliphant, "Python for Scientific Computing,ïn Computing in Science Engineering, vol. 9, no. 3, pp. 10-20, May-June 2007, doi: 10.1109/MCSE.2007.58.

[4] L. Gonzales, «Introduccion a la librería NumPy de Python,» [En línea]. Available: https://ligdigonzalez.com/introduccion-a-numpy-python-1/.

[5] G. Bradski y A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, 2008.

[6] «Face Landmarks Detector con Dlib y OpenCV,» 09 11 2017. [En línea]. Available: http://acodigo.blogspot.com/2017/11/face-landmarks-detector-con-dlib-y.html.

[7] «PyPI,» [En línea]. Available: https://pypi.org/project/PyAutoGUI/.

[8] «Reconocimiento de gestos en tiempo real basado en heurísticas y reconocimiento de patrones,» Puebla, [En línea]. Available: http://catarina.udlap.mx/u_dl_a/tales/documentos/mcc/«lopez_r_o/capitulo1.pdf

[9] »V. M.Asuncion, C. Fernández, A. G. Gil y L. Payá, «Equivalencia entre ICA y PCA como métodos de extraccion de caracteristicas en reconocimiento visudal basado en apariencias,» Elche (Alicante).

[10] »T. Soukupova y J. Čech, «Real-Time Eye Blink Detection using Facial Landmarks,» Slovenia, 2016.

[11] »D. C. García Cortes, «Reconocimiento de Gestos de Manos como Mecanismo de interaccion Humano-Computador,» Bogota D.C, 2014.