ALGORITMO DE SEGUIMIENTO OCULAR PARA GENERAR MOVIMIENTO

Tseng Martínez, Fabio Jhony; Arévalos, María Elena
Tutores: Arrúa Ginés, Jorge Luis; Ayala Diaz, Katia Andrea

fabiotseng@gmail.com; helema134@gmail.com;
Facultad Politécnica.
Universidad Nacional del Este.

RESUMEN

En este trabajo se desarrolló un algoritmo de interpretación del movimiento ocular mediante técnicas de visión artificial, como alternativa para la interacción humano-computadora, de manera no invasiva, es decir, sin el contacto con dispositivos físicos. La problemática motivadora del mismo es la de personas con discapacidades motrices en las extremidades del cuerpo tales como la tetraplejía, o esclerosis lateral amiotrófica (ELA).

El método de trabajo implicó la utilización de herramientas tales como la librería OpenCV, Dlib, el lenguaje de programación Python y el sistema operativo de distribución gratuita, GNU/Linux Debian 8.

Para la demostración del seguimiento ocular, se desarrolló una interfaz, con un entorno web utilizando lenguajes de programación HTML5, PHP, JAVASCRIPT y la biblioteca JQuery. El proyecto se centró técnicas de la visión artificial, que se basan en el procesamiento digital de imágenes para la detección del rostro y la pupila, hasta obtener la región de interés que fue detectada para realizar su seguimiento.

Durante las pruebas efectuadas, el algoritmo desarrollado arrojó como resultado un alto porcentaje de efectividad, tanto en la detección de rostros como de ojos, pudiendo de esa manera ser aplicable en sistemas de visión artificial o relacionados al seguimiento ocular.

Palabras clave: Pupila, detección facial, visión artificial, seguimiento ocular.

1. INTRODUCCIÓN

En una sociedad donde cada vez más la palabra accesibilidad es un compromiso, es necesario desarrollar o adecuar objetos que hagan posible el uso para todos. Esta premisa, y la problemática que aqueja a personas con discapacidades en las extremidades del cuerpo tales como la tetraplejía, o esclerosis lateral amiotrófica (ELA), motivan el desarrollo del presente trabajo.

Se plantea el desarrollo de un algoritmo capaz de detectar y realizar el seguimiento de la pupila de una persona, sin la necesidad de utilizar cualquier dispositivo invasivo.

Teniendo en cuenta lo expuesto, se utiliza métodos de procesamiento de imágenes a fin de proponer una solución de bajo poder computacional y confiable en su detección del movimiento deseado. Luego el resultado del trabajo realizado: seguimiento del movimiento de la pupila, puede ser aplicado a diversos escenarios donde se requiera proveer capacidad motriz a gente que la precise, por ejemplo para el movimiento de una silla de ruedas. Así, éste algoritmo en un futuro puede llegar a suministrar autonomía al usuario discapacitado de realizar ciertos movimientos, pudiendo prescindir de la ayuda de otra persona. Todo esto gracias al uso de algoritmos evolutivos para visión artificial.

Se pretende que el usuario sea capaz de controlar la dirección (derecha / izquierda) de una manera sencilla y sin esfuerzo en las extremidades del cuerpo, mediante movimientos oculares, a través de un reconocimiento de la pupila, aplicando técnicas de visión artificial.

2. OBJETIVOS

Objetivo General

- Desarrollar un algoritmo capaz de interpretar el movimiento ocular en el eje horizontal utilizando técnicas de procesamiento digital de imágenes.

Objetivos Específicos

- Analizar los distintos algoritmos de detección de objetos.
- Desarrollar el código para detección del rostro y ojos.
- Obtener la región de interés (pupila) mediante técnicas de procesamiento de imágenes más cálculo de matrices.
- Realizar seguimiento e interpretación de posición de la pupila.
- Reflejar los datos obtenidos en la fase de interpretación de posición, mediante una interfaz de simulación.

3. MATERIALES Y MÉTODOS

A continuación, se presentan las metodologías de técnicas de visión artificial

para detectar el rostro utilizando las plantillas de OpenCV, el clasificador de cascada HAAR, método de detección de objetos efectivo propuesto por Paul Viola y Michael Jones:

- *Descriptores tipo HAAR:* se utilizan los filtros HAAR, para calcular las imágenes a escalas, y de esa manera obtener como resultado el descriptor global de toda la imagen.

- *Imagen Integral:* cuando se acelera el cálculo de filtros HAAR, se suele transformar la imagen original en una imagen integral. Esta da como resultado una imagen nueva del mismo tamaño en el que el valor de cada píxel va a ser la suma de los píxeles en la imagen original que están situados a la izquierda y arriba.

- *Adaboost:* las características son clasificadas por medio de clasificadores especiales. El algoritmo se basa en la mejor forma de aplicar esos clasificadores para detectar favorablemente un evento. Además, posibilita tratar de forma eficiente un número elevado de características durante el proceso de aprendizaje, para luego obtener un clasificador robusto, que utilice únicamente un subconjunto de estas características. El algoritmo aprende y combina todos los clasificadores para obtener como resultado final un clasificador global que minimizará el error de clasificación. Al proceso para aprender cada uno de estos clasificadores simples es otorgando un peso relativo, el cual difiere para cada uno, dependiendo del resultado de los pasos de clasificación que durante el proceso fueron aprendidos.

- *Clasificador en cascada*: En este último paso se suele observar como resultado de aplicar el método Adaboost, una inmensa mayoría de ventanas abiertas en una imagen que no siempre corresponden a una cara. Este método posibilita alcanzar el objetivo deseado, que es la de encontrar una sola ventana que corresponda a una cara, mediante una combinación secuencial de clasificadores. Una imagen, solo es reconocida como cara si todos los clasificadores la aceptan, de esa manera el primer clasificador recibe todas las posibles ventanas de una imagen, todas aquellas que rechace este primer clasificador quedarán descartadas. Y así sucesivamente hasta que sólo se detecte una ventana como cara.

4. RESULTADOS Y DISCUSIÓN

En este apartado se describe cada uno de los resultados del proceso metodológico que se han presentado anteriormente.

Conversión de una imagen a escala de grises.

La librería OpenCv cuenta con formato predeterminado de color y se refiere a menudo como RGB; para un correcto funcionamiento de los algoritmos de detección de rostros usados en OpenCV, se convierte la imagen a escala de grises, para ello se utiliza el módulo cv2.cvtColor () (Fig. 1).

Figura 1: Conversión de una imagen a escala de grises.

Función para marcación.

Es necesario resaltar el área donde se encuentra el objeto detectado, hay funciones para realizar una marcación, ejemplos de ellas son

cv2.rectangle(), y cv2.circle().

cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)

cv2.circle(img, center, 1, (0, 255, 0), 2)

La primera función precisa parámetros que son, la imagen sobre la que se dibuja, la ubicación de los vértices superior izquierdo e inferior derecho, el color de la línea en formato RGB, y la anchura.

La segunda función requiere parámetros que son la imagen sobre la que se dibuja, la ubicación del centro, la distancia del radio, el color, y por último la anchura de la línea.

Detección de ojos en tiempo real.

Se utiliza OpenCV para detectar el rostro en una imagen para luego proporcionar ese resultado a la librería Dlib que es utilizada para la detección de ojos, la misma facilita dicho proceso ya que posee funciones capaces de detectar los rasgos o partes de un rostro, básicamente arroja como resultado una secuencia de puntos indexados que posibilitan identificar cada zona.

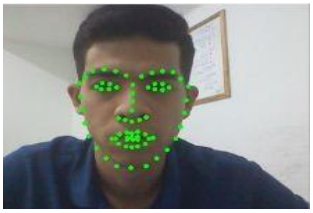

Figura 2: Detección de ojos en tiempo real.

Detección de la pupila mediante funciones de binarización.

Una vez extraída la zona del ojo mediante la librería Dlib, queda como siguiente paso la detección de la pupila, lo cual es posible aplicando la función cv2.threshold de la librería OpenCV. Básicamente, es una función de binarización consistente en la trasformación de una imagen escalada en grises, a una imagen en negro y blanco la cual

es determinada de acuerdo a los parámetros cargados en dicha función que establecen la frontera para la conversión. En resumen, si el valor del píxel es mayor que el valor umbral o frontera, se le asigna un valor (blanco), de lo contrario se le asigna el otro valor (negro).

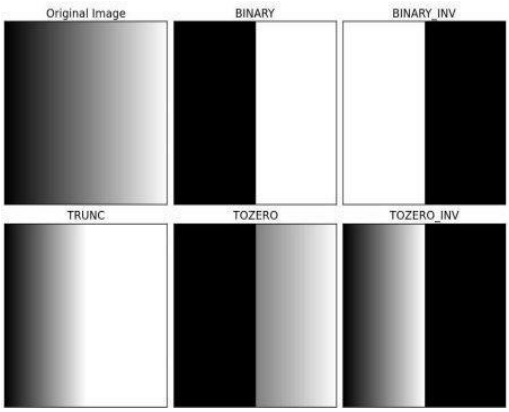

Figura 3: Funciones de binarización.

Código para binarización.

ret, imgbin = cv2.threshold(imgOjo, valorThresh, 255, cv2.THRESH_ TOZERO_ INV)

donde el primer argumento es la imagen fuente, que debería ser una imagen en escala de grises. El segundo argumento es el valor umbral que se utiliza para clasificar los valores de píxeles. El tercer argumento es el valor máximo que representa el valor a ser dado si el valor del píxel es más que el valor umbral. Al aplicar este proceso se logra obtener la parte más oscura, dependiendo del rango establecido en la función, el resultado es una pequeña porción de puntos negros que corresponde al iris del ojo.

Figura 4: Detección de la pupila.

Interpretación del movimiento ocular.

Para interpretar el movimiento ocular son necesarios dos datos: la ubicación del iris y la del ojo en la imagen de entrada, mediante los procesos de detección hechos anteriormente estos datos ya están disponibles. Se toma como referencia el perímetro del ojo y se calcula la diferencia de distancias existente entre los lados y el centro del iris. Con esto es posible detectar qué dirección va tomando durante la adquisición de imagen.

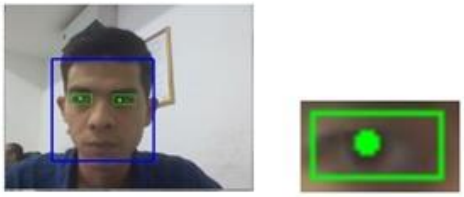

Figura 5: Interpretación del movimiento ocular.

Desarrollo de una interfaz.

Diseñada en un ambiente web, básicamente funciona mediante la interacción cliente–servidor.

El *software* de visión artificial envía los datos de procesamiento de imagen al servidor, el

cual refleja los datos obtenidos en una interfaz web, la herramienta websocket posibilita la conexión entre ambos facilitando el envío y recepción de datos en ambos sentidos.

Código websocket lado servidor.

.

socket = socket_ create(AF_ INET, SOCK_ STREAM, SOL_ TCP

Código websocket, lado cliente.

websocket.onmessage = function(ev)

arumsg = msg.message;

Figura 6: Pantalla principal.

Pruebas realizadas.

Las pruebas del sistema fueron realizadas con el objetivo de verificar la funcionalidad y eficacia en el proceso de detección tanto de rostros como de ojos, para ello se analizaron las posibles variantes y circunstancias a las que podría someterse el uso del mismo, por ende, fueron considerados dos factores: tipos de luces y el entorno, utilizando para ello una cámara web y un luxómetro (app de un Smartphone Android) a fin de referenciar la intensidad de luz.

La muestra utilizada fue de 11 personas, teniendo en cuenta, la fisionomía de cada rostro, la utilización de accesorios como anteojos, los diferentes tipos de ojos y la distancia entre el usuario y la cámara web. Para las mediciones de luminosidad del entorno los registros fueron utilizados a una altura promedio de 70 cm con respecto al suelo.

Detección del rostro.

La prueba consistió en verificar la efectividad de la detección de distintos rostros, fue utilizada una muestra de 7 personas, bajo un ambiente controlado, manteniéndose constantes las posiciones tanto del receptor, como del entorno.

Nro.	Imagen	Resultado	Observación
1		Detección Efectiva	Ninguna
2		Detección Efectiva	Ninguna
3		Detección Efectiva	Ninguna
4		Detección Efectiva	Ninguna
5		Detección Efectiva	Ninguna
6		Detección Efectiva	Ninguna
7		Detección Efectiva	Ninguna

Se obtuvo un 80% de efectividad (detección eficaz y constante), cabe destacar en cuanto al 20% de error, que fue debido a casos tales como, demasiada distancia entre el usuario y la cámara, baja luminosidad dentro del recinto, entre otros factores.

Detección de pupilas.

Esta prueba consistió en verificar la efectividad de la detección de ojos, fue utilizada una muestra de 7 personas, bajo un ambiente controlado, manteniéndose constante la posición tanto del receptor, como del entorno.

Nro.	Imagen.	Resultado.	Observación.
1		Detección Efectiva	Ninguna
2		Detección Efectiva	Ninguna
3		Detección Efectiva	Ninguna
4		Detección Efectiva	Ninguna
5		Detección Intermitente	Zona del ojo Izquierdo con poca iluminación
6		Detección Efectiva	Ninguna
7		Detección Efectiva	Ninguna

Se obtuvo un 77% de efectividad (detección eficaz y constante), cabe destacar en cuanto al 23% de error, que fue debido a los mismos casos que ocurrieron en la detección de rostro, tales como demasiada distancia entre el usuario y la cámara, baja luminosidad dentro del recinto, y, además, el uso de accesorios (lentes) que en algunas pruebas imposibilitaba la detección de las mismas.

Detección variable ángulo de incidencia.

Para esta prueba se utilizaron luces LED e iluminado (frontal, trasero, lateral izquierdo y lateral derecho).

Además, fue utilizado el aplicativo para la medición de la luminosidad, tanto para el registro de la intensidad de la luz del entorno, como la luz incidida por el objetivo directamente.

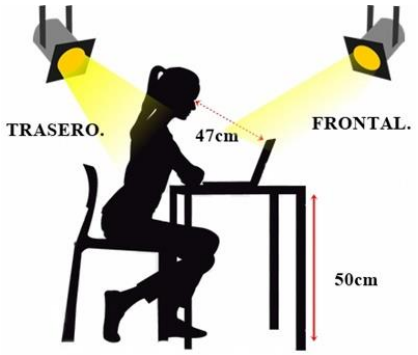

Figura 7: Iluminación lado frontal y lado trasero.

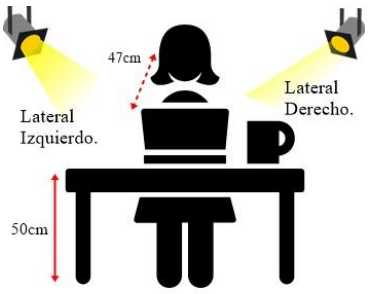

Figura 8: Iluminación lateral izquierdo y lateral derecho.

Figura 9: Utilización de la app para medir la luminosidad.

En la prueba solamente fueron encontrados falsos positivos intermitentes con la ubicación del punto luminoso en la parte posterior del objetivo. En las demás direcciones los resultados fueron satisfactorios.

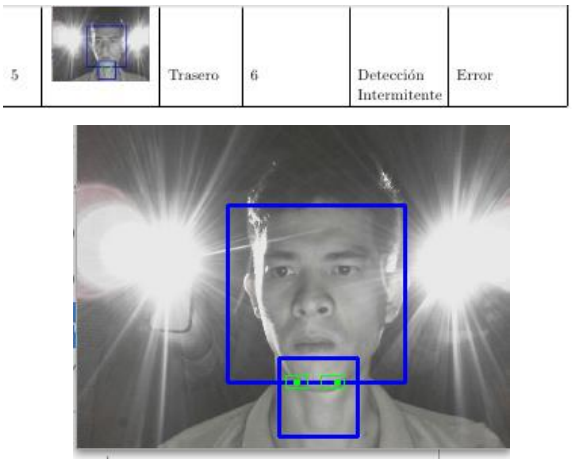

Figura 10: Falsos positivos en la imagen.

5. CONCLUSIONES

En base al primer objetivo, de analizar los algoritmos de detección de objetos, se encontraron varios algoritmos relevantes para cualquier tipo de detección, por ejemplo, los clasificadores HAAR, que vienen integrados dentro de la librería OpenCV, que posibilitaron realizar el propio algoritmo para detectar el rostro, y poder visualizarlo en las pruebas realizadas.

En total se obtuvieron 70 pruebas, considerando varios tipos de variables, de las cuales 55 pruebas resultaron exitosas (Detección eficaz) y 15 pruebas arrojaron algún tipo de error (Detección intermitente o falso positivo), demostrando un 78,5% de efectividad.

- Detección de rostros: se obtuvo un 80% de efectividad (detección eficaz y constante), cabe destacar en cuanto al 20% de error que se produjo, que fue debido a casos tales como, demasiada distancia entre el usuario y la cámara, baja luminosidad dentro del recinto, entre otros factores.
- Detección de pupila: se obtuvo un 77% de efectividad (detección eficaz y constante), cabe destacar en cuanto al 23% de error que se produjo, que fue debido a los mismos casos que ocurrieron en la detección de rostro, tales como demasiada distancia entre el usuario y la cámara, y baja luminosidad dentro del recinto.
  El uso de accesorios (lentes) también obstaculizó en las pruebas, la detección de las mismas.

Solución del problema de investigación.

Caso 1: Si existía más de un usuario frente a la cámara el software detectaba ambos rostros por ende el mismo no podía discernir a cuál de los rostros hacer el seguimiento ocular. Una posible solución a esto es dar prioridad al rostro más cercano de modo que pueda descartar a las personas que se ubican por detrás del usuario principal.

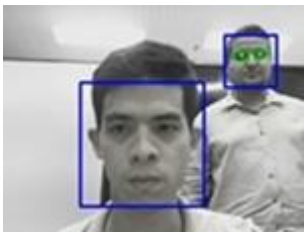

Figura 11: Detección Errónea.

Caso 2: En ambientes con baja iluminación se producían falsos positivos, es decir, se detectaban rostros no verdaderos, en sistemas de visión por computador también es posible el reconocimiento facial que consiste en distinguir si el rostro detectado pertenece o no a un usuario, con esto se podría solucionar este tipo de inconveniente

a nivel software, o bien a nivel hardware, incluir luces infrarrojas, el cual es captada por cámaras digitales pero no por los ojos humanos, de este modo la iluminación seria constante aun en condiciones oscuras.

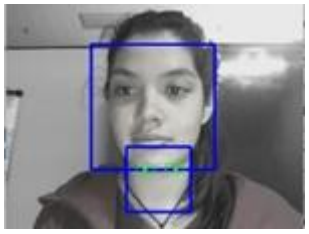

Figura 12: Falsos positivos en la imagen.

Caso 3: En las pruebas de detección y seguimiento ocular existían escenarios donde los usuarios utilizaban gafas, en ciertos movimientos el marco del mismo se interponía entre la cámara y los ojos, esto imposibilitaba la correcta detección. Una posible solución sería ubicar la cámara por debajo de la pantalla de este modo se obtendría un ángulo más preciso de la imagen.

Caso 4: El software hecho está limitado para detección del movimiento ocular en dirección horizontal, izquierda - derecha, esto es a consecuencia de la baja calidad de imagen con la que se ha trabajado, una posible solución sería la utilización de imágenes de alta resolución de modo que se pueda captar la distancia o la ubicación de los ojos a nivel vertical.

TRABAJOS FUTUROS

- Seguimiento del movimiento ocular en varios ejes, no solo horizontal, podría ser vertical y en diagonales, cuyo requisito es la utilización de imágenes de mayor resolución y uso de librería de procesamiento de digital de imagen por aceleración de hardware. Diseñar un sistema SCADA multiplataforma, que pueda ser controlado desde una página web, lo que evitaría la dependencia de una sola plataforma.
- Detectar la dirección con que el rostro mira hacia la cámara, con el fin de aumentar la robustez del seguimiento ocular mediante la utilización de la libreríaDlib y la plantilla ladmark.
- Desarrollar una app móvil, con el uso de OpenCV.
- Implementar sistemas de reconocimiento faciales como herramienta para la seguridad del hogar.
- Desarrollo de juegos como el bubble-shooter, Arkanoit utilizando este algoritmo de seguimiento ocular, con la finalidad de reemplazar las teclas de izquierda, derecha por la utilización de los ojos.

BIBLIOGRAFÍA

[1] Biblioteca de Ingeniería. Capítulo II - El o .Universidad de Sevilla. [Documento WWW], recuperado: url{http://bibing.us.es/proyectos/abreproy 12018/fichero/Memoria\%252F5+-

+El+ojo+humano.pdf}

[2] Departamento de ingeniería eléctric Introducción a la visión Artificial. España. [e línea]
url{<http://www.elai.upm.es/webantigua/s ain/Asignaturas/MIP_VisionArtificial/Apunt sVA/cap1IntroVA.pdf}.

[3] SABIA. Visión Artificial .España.
url{http://sabia.tic.udc.es/gc/Contenidos%. 0adicionales/trabajos/3D/VisionArtificial/i dex.html}.

[4] Ing. en Automatización y Control Industri .Aspectos de un sistema de visión artificia Universidad Nacional de Quilmes, [en línea]
url{http://iaci.unq.edu.ar/materias/vision/ rchivos/apuntes/Aspectos%20de%20un%2 0Proyecto%20de%20Visi%C3%B3n%20Ar ficial.pdf}. [Publicado: Octubre de 2005]

[5] Elipse Software. "Elipse Knowledgebase. KB-33868: Comunicando com S7-200 via PPI". 2011.

[6] D. Pérez, "El Android Libre," [En línea]. Disponible en: http://www.elandroidelibre.com/2015/01 /google-play-supera-la-appstore-en-cantidad-de-aplicaciones-y-desarrolladores.html. [Último acceso: 2015].