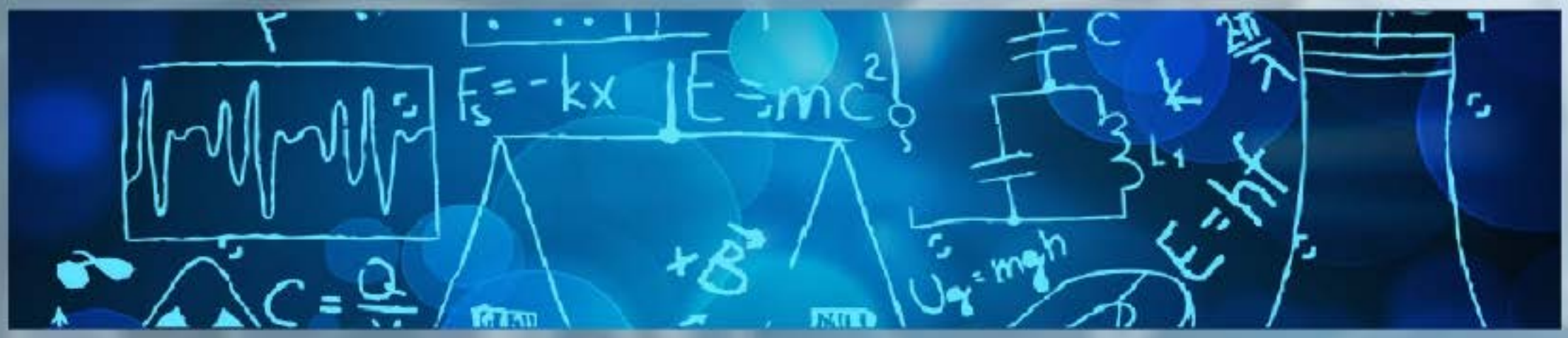
Coloquio de Investigación Multidisciplinaria

JOURNAL CIM : Revista Electrónica Arbitrada
(ISSN2007-8102)

latindex



Ingeniería Administrativa

Ingeniería Industrial

Ingeniería Química

Ingeniería Electrónica

Ingeniería Mecatrónica

Ingeniería Robótica

Investigación Educativa

Sistemas Computacionales

OCTUBRE 2016

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Tecnológico Nacional de Mexico®
Instituto Tecnológico de Orizaba

Journal CIM. Vol.4, Num. 1
Coloquio de Investigación Multidisciplinaria 2016
ISSN 2007-8102

Comparación de algoritmos para la detección ocular utilizando OpenCV.

E.A. Cuellar Cortes[1], C.E. Miranda Medina[1], A.P. Fuentes Garcia[2]*
[1]Instituto Tecnológico de Orizaba, Ingeniería Electrónica
[2]Instituto Tecnológico de Orizaba, Ciencias Económico-Administrativas
Oriente 9 No. 852, Emiliano Zapata, 94320 Orizaba, Ver., México
**kikeale80@gmail.com*
***Área de participación**: Ingeniería Electrónica*

Resumen

A continuación, se presentará la comparación entre dos algoritmos de detección ocular realizados con una librería open source para C++, denominada OpenCV. Ambos utilizan funciones de ésta para generar dos alternativas de seguimiento ocular. Se explicará cada uno por separado tomando en cuenta el método de adquisición de la imagen (la cual varía dependiendo del algoritmo, para facilitar el procesamiento). En la sección de resultados se detallarán las características a favor o en contra entre cada uno de los algoritmos.
***Palabras clave**: Procesamiento de Imágenes, OpenCV, Seguimiento Ocular*

Abstract

The present work shows the comparison between two eye detection algorithms performed with a C++ open source library, named OpenCV. Both algorithms apply library functions to generate two eye tracking options. The algorithms will be explained separately considering the image adquisition method (which varies depending on the algorithm, to simplify the execution). The advantages and disadvantages for every algorithm are explained in the results section.
***Key words**: Image processing, OpenCV, Eye Tracking*

Introducción

El seguimiento de ojos es una técnica mediante la cual la posición se utiliza para determinar dirección de la mirada de una persona en un momento dado [12]. Esto es llevado a cabo para analizar los movimientos oculares, mientras que una persona se encuentra en diferentes ocupaciones. Diferentes técnicas se desarrollaron a lo largo de los años, uno de los primeros estudios se le adjudican a Emlie Javal un oftalmólogo francés quien observó (en 1879) con ayuda de un espejo y determinó que los movimientos oculares no son contínuos a lo largo de una fase, sino compuestos de movimiento rápidos. Posteriormente Edmud Huey en 1908 construyó un dispositivo usando lentes de contacto con un orificio para la pupila. En 1901 Dodge investigó la velocidad de movimiento de los ojos muy precisa y no invasiva, basada en reflexión corneal, llamado foto cronógrafo. Cuatro años más tarde, en 1905, Charles H. Judd desarrolló un dispositivo que permitía registrar los movimientos en ambas direcciones. Paul Fitts, en 1947 mejora la seguridad aérea, relacionando el movimiento de los ojos y la actividad cognitiva. Se utiliza la cámara de vídeo para capturar y estudiar la actividad ocular de los pilotos de aviones durante los vuelos. Hartridge y Thompson inventaron el primer rastreador ocular montado en la cabeza, de este modo las limitaciones del movimiento de la cabeza fueron eliminados. En los años 70´s los esfuerzos se centraron en la investigación sobre el funcionamiento del ojo humano y lo que puede ser revelador de los procesos perceptivos y cognitivos [13]. Con la aparición de la computadora en los 80´s, los científicos tuvieron un instrumento muy importante para el procesamiento de datos con velocidad. Es aquí donde inicia la interacción entre ser humano y la computadora. Siendo la principal investigación la interacción con personas discapacitadas. También fue aquí cuando se estudió el seguimiento de los ojos en los anuncios en revistas, observando qué páginas se leen realmente y qué zonas son las preferidas. A principios de los 90´s, el seguimiento de los ojos fué utilizado por la NFL para determinar qué parte de la pantalla de tv era más vista, teniendo un gran éxito y así fue utilizado por una gran agencia de publicidad y comercialización en sitios web.

El dispositivo de medición para calcular los movimientos del ojo es comunmente conocido como "Seguidor Ocular". En general hay dos tipos de técnicas de monitoreo: uno que mide orientación del ojo con respecto a la cabeza y otro que mide la orientación del ojo en el espacio o el punto que mira [11]. Esta última se utiliza normalmente cuando la preocupación es la identificación de los elementos en una escena visual, por ejemplo, en aplicaciones interactivas. Hay cuatro categorías generales de las metodologías de medición de los movimientos oculares que implican el uso o la medición de: Electrooculografía (EOG), Foto-oculografía (POG) o Video-oculografía (VOG), y combinado basado en video y la reflexión corneal. Estos últimos los rastreadores basados en video, utilizan cámaras relativamente económicas y procesamiento de imágenes para calcular el punto que se mira. El presente trabajo utilizará este método de seguimiento ocular con dos versiones de algoritmos.

OpenCV es una biblioteca de visión por computadora de código abierto más popular, con más de 500 algoritmos optimizados para la imagen y el análisis de vídeo. Debido a esto se ahorra tiempo y energía. OpenCV se distribuye con una licencia BSD, lo que significa que se puede hacer una aplicación comercial sin revelar el código fuente. Sin embargo, hay algunos algoritmos proporcionados que están patentados. OpenCV puede ser programado en C ++, C, Python, Java y las interfaces son compatibles con Windows, Linux, Mac OS,
iOS y Android [1][2].

Metodología

ALGORITMO A.

La figura 1 muestra un diagrama a bloques del algoritmo A, simplificando en 4 etapas principales: la captura de la imagen, el pre-procesamiento de la imagen, donde se acondiciona la imagen capturada y desde luego el procesamiento principal que se encarga de la detección ocular, así como la última etapa que es la salida en pantalla como resultado del algoritmo.

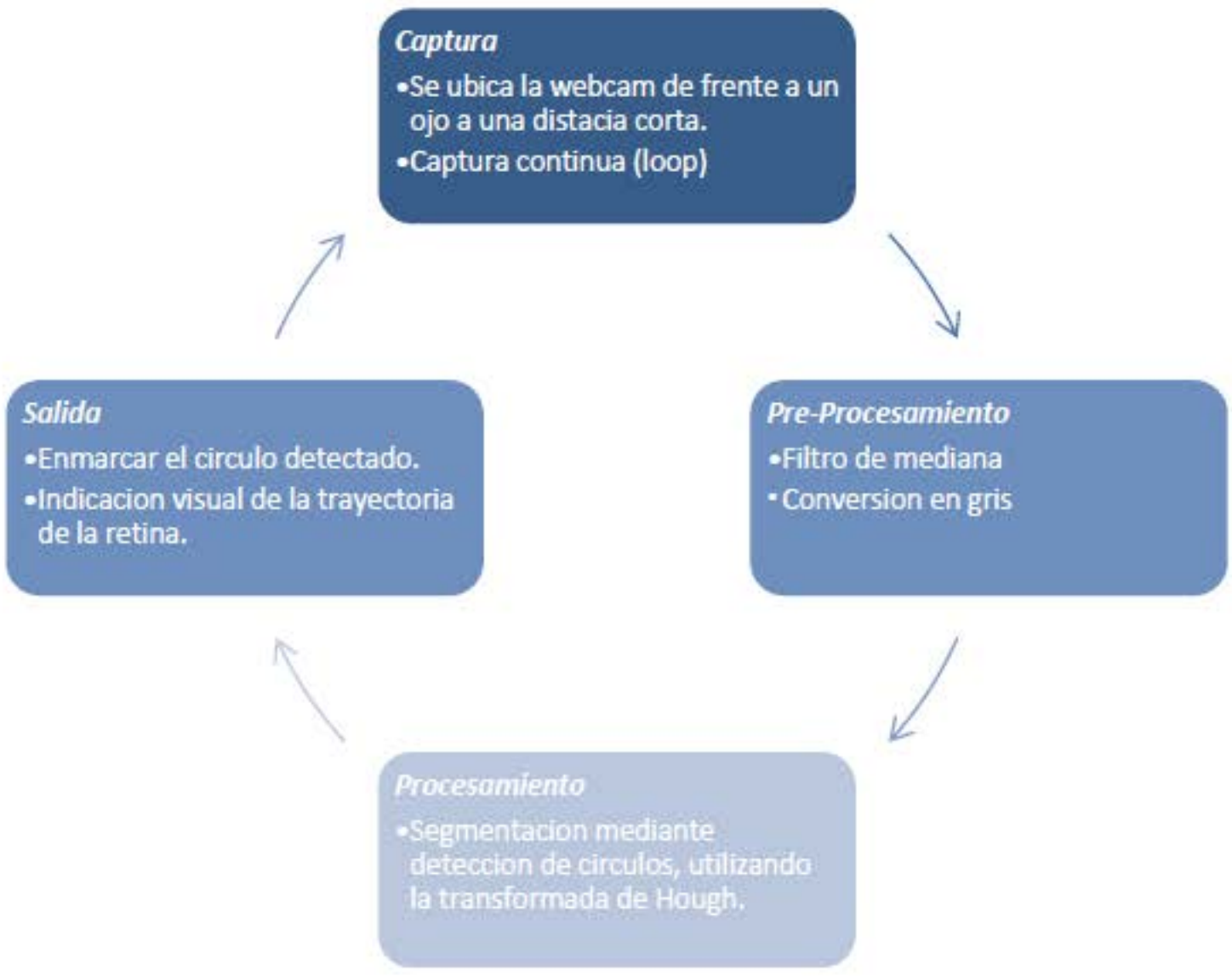


Figura 1. Diagrama a bloques del Algoritmo A.

ISSN 2007-8102

▫ Forma de captura en el algoritmo A.

La captura de la imagen en este algoritmo se hace con una cámara cerca del ojo (aproximadamente 10cm), en este caso con una cámara USB, utilizada para analizar fallos en tuberías con una resolución de 640x480, la cual se colocó en una diadema y mediante reflexión sobre un espejo se podía observar el ojo, ya que, si se pone de frente al ojo, simplemente estorbaría la visión del usuario, impidiendo así una detección correcta de su retina. En la figura 2 se puede observar con rojo lo que la cámara visualiza en este caso es una sección muy pequeña del rostro del usuario, pero el ojo se encuentra presente en la imagen. La captura es continua, se encuentra en un loop infinito realizando continuamente el pre-procesamiento y procesamiento, para así tener una salida en tiempo real.

Figura 2. A. Se observa la colocación de la cámara en el usuario, montada cerca del ojo del usuario. B. Cámara utilizada en este algoritmo.

▫ Descripción del algoritmo A.

Una vez realizada la captura de la imagen, entramos a primera etapa, utilizando la función (`medianBlur`) filtro de mediana el cual suavizar la imagen esto debe realizarse debido a la baja resolución de la cámara. La sintaxis es muy simple solo necesita una imagen de entrada, una de salida, y un kernel que por default es de 3 [3, 4, 5]. Posteriormente se convierte a escala de grises mediante (`cvtColor`). Esta función convierte un espacio de color a otro y por último la umbralización binaria (`adaptiveThreshold`), convierte la imagen a blanco y negro, esto permitirá realizar una mejor segmentación del ojo humano, ya que la esclerótica es blanca y el iris y la pupila de color obscuro existiendo así un claro contraste para identificar el centro de la pupila.

La parte principal del algoritmo A es la transformada de Hough, que se encarga de detectar los círculos de la pupila e iris para darle un centro determinado e indicar el seguimiento ocular. Suena fácil ya que existe la función en OpenCV (`HoughCircles`) no obstante configurar esta función dependerá de muchos factores como la iluminación, tamaño del circulo a detectar y la colocación de la cámara.

La trasformada Hough inicia con una imagen a la cual se le aplico la detección de bordes (Canny o Sobel), esta imagen provee puntos que les da sentido y trata de encontrar curvas, rectas, círculos. Para determinar círculos con un radio `R` y con el centro en `(a,b)`, así se describen las ecuaciones paramétricas `x = a + R cos(θ), y = b + R sin(θ)`, cuando el ángulo θ es barrido través de todo el rango de 360 grados los puntos (x, y) traza el perímetro de un círculo. [3, 4, 5]..

Sintaxis en OpenCV

```
HoughCircles(InputArray im,
             OutputArray cir,
             int method,
             double dp,
             double minDist,
             double param1=100,
             double param2=100,
             int minRadius=0,
             int maxRadius=0)¶
```

Argumentos

- *im*: Imagen de entrada en escala de grises
- *cir*: Vector que almacena 3 valores: x, y, r para cada círculo detectado.
- *method*: método CV_HOUGH_GRADIENT único por el momento.
- *dp*: La relación inversa de la resolución (por default 1).
- *minDist*: Distancia mínima entre centros detectados.
- *param1*: Umbral superior de Canny (detector de bordes).
- *param2*: Umbral para la detección del centro.
- *min_radiu*: Radio mínimo para detección.
- *max_radius*: Maximo radio detectado (por default 0).

ISSN 2007-8102

Como última parte de este algoritmo, se toma los datos del vector `cir`, donde se almacenaron las coordenadas (x,y) y r para que mediante las herramientas de dibujo de OpenCv se ponga un rectángulo azul en el centro de la pupila (como se muestra en la figura 3).

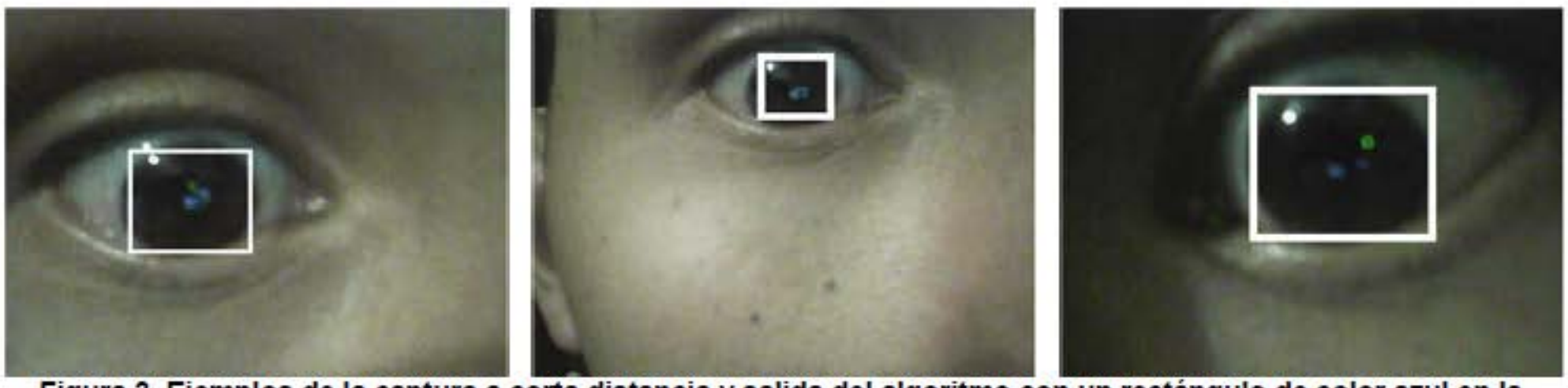


Figura 3. Ejemplos de la captura a corta distancia y salida del algoritmo con un rectángulo de color azul en la detección de la pupila e iris, tomando el centro de ese cuadrado como la deteccion ocular.

ALGORITMO B.

La figura 4 muestra un diagrama a bloques del algoritmo B, en cuatro etapas, la captura, pre-procesamiento, procesamiento principal y la salida.

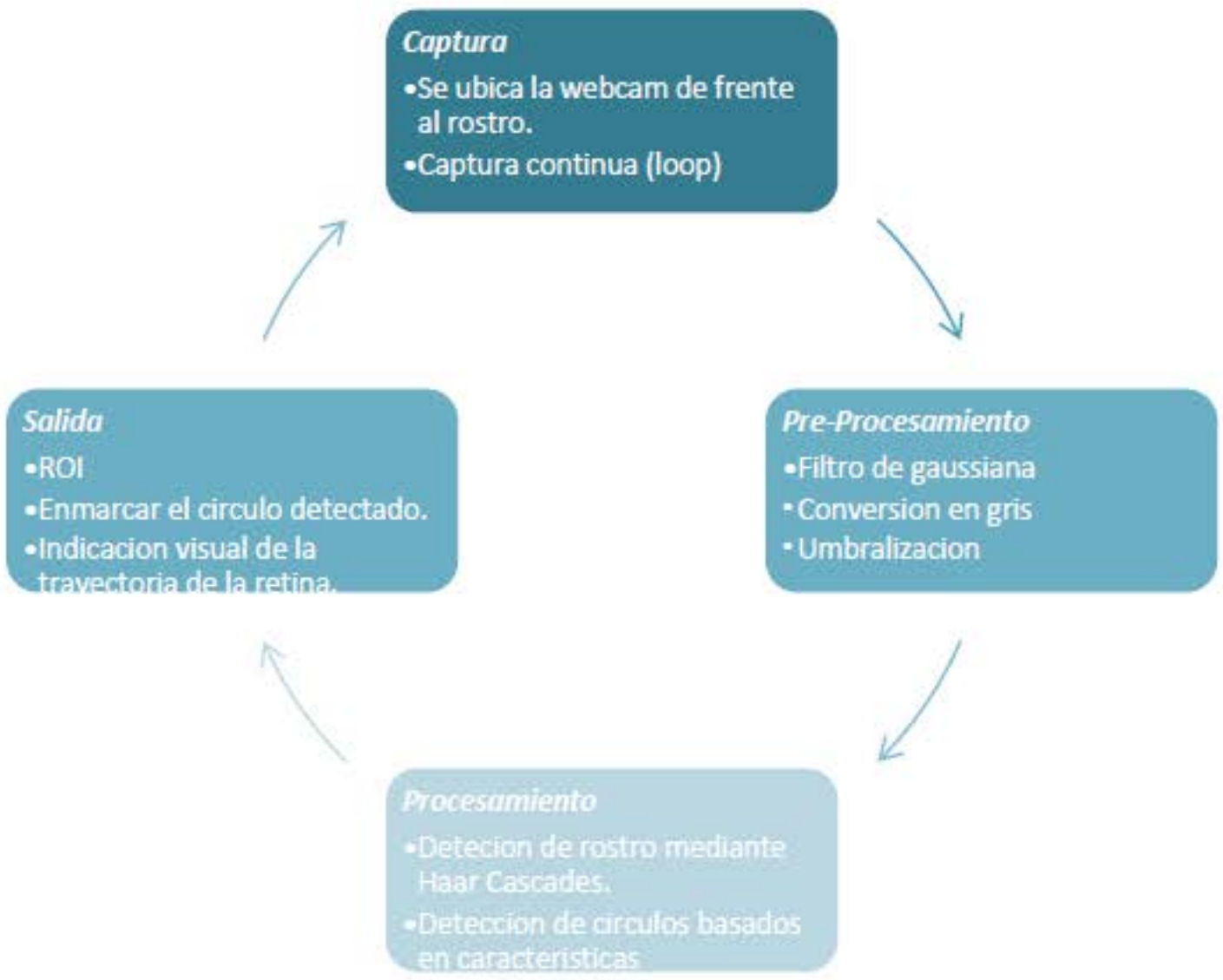


Figura 4. Diagrama a bloques del Algoritmo B.

▫ Forma de captura en el algoritmo B.

La captura de la imagen en este algoritmo se ubica de frente al rostro para facilitar la detección del mismo, en un inicio se pensó utilizar la webcam que traen en la pantalla las computadoras portátiles, pero la posición de ésta varía según el ángulo de inclinación de la pantalla, por lo cual se optó por una cámara Logitech C170 montada en un tripié. La captura es contínua, se encuentra en un loop infinito realizando continuamente el pre-procesamiento y procesamiento, para así tener una salida en tiempo real.

A B

Figura 5. A. Se observa la colocación de la cámara en el usuario, montada en un tripié de frente al rostro. B. Cámara utilizada en este algoritmo.

▫ Descripción del algoritmo B.

Con la cámara frente al rostro se toma una imagen la cual se pre-procesa con un filtro gaussiano (`GaussianBlur`). Este filtro es de los más utilizados, aunque computacionalmente hablando no es de los más rápidos. Se realiza mediante la convolución de cada punto de la matriz de entrada (imagen) con un kernel gaussiano y luego sumando todos ellos para producir la matriz de salida.

Lo primero que se detectará es el rostro de la persona mediante los clasificadores Harr Cascades, utilizando OpenCV. Haar Cascades en realidad es un detector de objetos propuesto por Paul Viola [9] y mejorado por Rainer Lienhart [10]. Primero se tiene un clasificador que es entrenado con cientos de ejemplos de un determinado objeto (el rostro, ojos, un coche, celular, etc) todas estas muestras están escaladas al mismo tamaño. Después que el clasificador está entrenado puede ser aplicado a una región de interés de una imagen de entrada buscando regiones probables que muestren el objeto dando una salida de 1 lógico o caso contrario 0. La palabra cascade en el nombre del clasificador significa que se compone de varios clasificadores sencillos (o etapas) que se aplican continuamente a la región de interés hasta que en un momento dado se rechaza o se acepta el objeto encontrado.

OpenCV ya cuenta con Haar Cascades entrenadas para la detección de rostro y ojos, este algoritmo primero localizará la forma de un rostro humano en toda la toma, una vez localizado segmentará el rostro que es la única área de interés, posteriormente utilizando nuevamente Haar Cascades sobre la imagen ya segmentada, se localizará la forma de los ojos enmarcando en un rectángulo las pupilas e iris que sería la parte más obscura del ojo, dando así el centro para la deteccion ocular [5]

Dentro de la carpeta `/opencv/data/haarcascades/` ya se cuenta con clasificadores los cuales en el código se cargan como variables globales:

```
String face_cascade_name = "haarcascade_frontalface_alt.xml";
String eyes_cascade_name = "haarcascade_eye_tree_eyeglasses.xml";
```

De manera simplificada el código para este algoritmo B se muestra continuación:

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
img = cv2.imread('sachin.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]
eyes = eye_cascade.detectMultiScale(roi_gray)
for (ex,ey,ew,eh) in eyes:
cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
```

Las dos primeras líneas como se observa son las variables globales que mandan a llamar a los clasificadores desde la carpeta indicada. La imagen a color (estática para este ejemplo) se carga en la variable `img`, posteriormente se realiza una conversión de RGB a GRIS, y se almacena en la variable `gray`.

La instrucción `detectMultiScale` es la que se encarga de detectar objetos de diferentes tamaños en la imagen de entrada. En este caso la variable gray es la imagen de entrada de `detectMultiScale` y "`faces`" es una matriz que almacena el resultado. Aclarando que no guarda los rostros detectados si no la información de donde estos rostros se encuentran en la imagen de entrada por tal motivo se utilizan ciclos anidados para la lectura, generando así también la salida gráfica como se muestra a continuación en la figura 6.

Figura 6. Ejemplo de tomas en diferentes tiempos de captura, donde se muestra al sujeto teniendo una identificación positiva de los ojos y su centro.

Resultados y discusión

Para mostrar una estadística del número de veces que ocurre la detección ocular, no se realizaró en tiempo real este experimento, fueron tomas al azar. Fotogramas tomados de una secuencia. Se tomaron 5 sujetos de prueba y dos iluminaciones (iluminación media e iluminación alta). Los cinco sujetos cuentan con características muy diferentes y tonos de piel distintos, como se muestra a continuación en la figura 7.

Figura 7. Se muestran las 5 personas con características diferentes y que fueron los sujetos de prueba.

Los algoritmos son ejecutados en un ciclo infinito tomándose 50 muestras aleatorias con los dos tipos de iluminación para cada algoritmo, de tal manera que se obtuvieron 250 muestras para el algoritmo A y 250 para el algoritmo B. A partir de esto, se tomó el porcentaje de detección ocular por cada sujeto. Los resultados se muestran en la tabla 1.

ISSN 2007-8102

Tabla 1. Datos estadísticos de la detección ocular.

Sujeto	1	2	3	4	5	Promedio
Algoritmo A						
Iluminación media	87%	83%	78%	52%	90%	78%
Iluminación alta	91%	88%	83%	70%	93%	85%
Algoritmo B						
Iluminación media	88%	79%	77%	63%	94%	80.2%
Iluminación alta	93%	75%	83%	76%	97%	84.8%
Promedio	89.75%	81.25%	80.25%	65.25%	93.5%	82%

De la tabla anterior se puede observar que los mejores resultados sin importar el algoritmo utilizado, se dan con una alta iluminación y se muestra una mejora de 7% en el algoritmo A de iluminación media-alta. Y en algoritmo B se observa una diferencia de 4.6% mejora no significativa.

Ambos algoritmos tienen un promedio de 82%, en la identificación y seguimiento ocular, una de las causas de tan bajo porcentaje es el fondo de las imágenes de prueba, en los sujetos 1 y 5 el fondo de la imagen se encuentra estable en color, y con los sujetos 2,3 y 4 se observan muchos objetos y diferentes colores, generando que ambos algoritmos den falsos negativos, esto lo podemos observar con el alto porcentaje de 93.5% del sujeto 5 donde el fondo es simple.

El sujeto 4, presentó el menor porcentaje de acierto para ambos algoritmos e iluminaciones, para el algoritmo A presento un 52% llegando a la conclusión que los ojos pequeños y el tono de piel y lunares fue la principal causa, ya que para el algoritmo A, con la cámara de cerca al ojo no era posible identificar su ojo. Y en el algoritmo B con la gran cantidad de pecas de tonalidad obscuro generaba identificaciones erróneas.

Una problemática identificada al principio con los sujetos 2 y 5 es la utilización de lentes, pues en ellos se reflejaba el monitor o el brillo del armazón generaba errores en la identificación, por lo que desde un inicio se estableció que las pruebas se realizarían sin lentes.

El algoritmo B es el que mayor porcentaje de identificación presenta con respecto al algoritmo A, esto debido a la identificación del rostro y posteriormente de los ojos. Una ventaja del algoritmo B es que la cámara se encuentra frente al usuario, mientras que la cámara del algoritmo A es invasiva al ser colocada en el rostro del usuario.

En al algoritmo A, una vez detectada la posición ocular y debido a la posición fija de la cámara es posible determinar la trayectoria ocular ya que el algoritmo almacena las coordenadas "X" y "Y". Con esta información podemos seguir el ojo en referencia a la imagen que siempre es fija con respecto a la cámara, el ojo nunca sale de la escena. Aunque el usuario mueva la cabeza, la cámara se encuentra fija y posicionada en su ojo. Con respecto al algoritmo B de igual manera el algoritmo almacena las coordenadas "X" y "Y", sin embargo, como la cámara se encuentra viendo una escena más amplia, y donde el usuario puede mover la cabeza, no se tiene punto referencia. El usuario podría estar moviéndose de derecha a izquierda y esto varia la posición de la detección, pero significa que sea el movimiento de los ojos, si no es el desplazamiento de la persona.

Trabajo a futuro

Estos algoritmos se pretende implementarlos en Python (sistema operativo Linux) en Raspberry 3 model B, con cámara CSI conectada al puerto de la tarjeta. Todo esto se desarrollará en un proyecto de investigación entre las áreas de ingeniería electrónica e ingeniería en sistemas del Instituto Tecnológico de Orizaba. Lo anterior será con la finalidad de diseñar un dispositivo capaz de prevenir accidentes por fatiga del conductor en vehículos de transporte de carga. El sistema monitoreara la mirada y parpadeo el conductor, que junto con un algoritmo deberá ser capaz de detectar que una persona se está durmiendo al volante, activando una alarma sonora y visual que dé aviso.

ISSN 2007-8102

Conclusiones

Dadas las circunstancias presentadas en los dos algoritmos mostrados, se concluye que el algoritmo A resulta ser óptimo para realizar la detección y seguimiento ocular ya que la cámara se encuentra fija y posicionada en el ojo, sin embargo resulta ser invasivo y sólo permite el seguimiento del ojo que está enfocando; el Algoritmo B presentó resultados con detección de rostro a través de Haar Cascades, presentando altos porcentajes de detección ocular sin ser invasivo ya que captura la posición de los dos ojos y no obstaculiza la visión del usuario, sin embargo no resulta tan preciso para el seguimiento ocular como el Algoritmo A. Sin duda la clave para el buen funcionamiento es una muy buena iluminación y un fondo controlado. Otro punto a destacar es que el algoritmo B tienda a encontrar falsos negativos con características faciales muy particulares. Como una primera aproximación al seguimiento ocular OpenCV brinda las herramientas necesarias y si se recomienda para una amplia gama de proyectos relacionados con la visión y el procesamiento de imágenes. Lo anterior se resume en la tabla 2, a continuación:

Tabla 2. Características encontradas en cada uno de los algoritmos.

ALGORITMO A	ALGORITMO B
Funciona para seguimiento y detección ocular	Funciona para detección ocular
Invasivo	No invasivo
Limitación a un solo ojo	Ambos ojos (rostro)
No influye el fondo	El fondo influye.
Funciona con alta iluminación	Funciona con alta iluminación.

Agradecimientos

Los autores agradecen por las facilidades y el apoyo brindado por la carrera de Ingeniería Electrónica del Instituto Tecnológico de Orizaba.

Referencias

1. Daniel Lélis Baggio, Shervin Emami, David Millán Escrivá, Khvedchenia Ievgen, Naureen Mahmood, Jason Saragih, Roy Shilkrot, Mastering OpenCV with Practical Computer Vision Projects, PACKT publishing, 2012.
2. Gary Bradski y Adrian Kaebler, Learning OpenCV Computer Vision with the OpenCV, O´Relly, primer edición, 2008.
3. Gary Bradski y Adrian Kaebler, Learning OpenCV Computer Vision with the OpenCV, O´Relly, segunda edición, 2010.
4. Jayneil Dalal y Shil Patel, Instant Opencv Starter, Pack Publishing, 2013.
5. Andrew Duchowski, Eye Tracking Methodology, Segunda edición, Springer, 2007.
6. Samarth Brahmbh, Practical OpenCV, Technology in action, 2013.
7. Robert Laganiere, OpenCV 2 Computer Vision Application Programming Cookbook, Pack Publishing, primera edición, 2011.
8. Ishan Goradia Á, Jheel DoshiÁ y Lakshmi Kurup, Articulo sobre Oculus Rift & Project Morpheus, International Journal of Current Engineering and Technology, INPRESSCO, 2014.
9. Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001. The paper is available online at http://research.microsoft.com/en-us/um/people/viola/Pubs/Detect/violaJones_CVPR2001.pdf
10. Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.
11. Laurence R. Young, David Sheena. Behavior Research Methods & Instrumentatio, Survey of eye movement recording methods, September 1975, Volume 7, Issue 5, pp 397–429
12. Poole A., Linden J. Ball, Eye Tracking in Human Computer Interaction and Usability Research: Current Status and Future Prospects. Encyclopedia of Human Computer Interaction, ISBN13: 9781591405627, ISBN10: 1591405629, EISBN13: 9781591407980, 2006.
13. Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises (Section Commentary), in The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research. Ed. by J. Hyona, R. Radach, and H. Deubel, 573−605, Amsterdam, Elsevier Science (2003).