Neiva, 25 – 07 - 2018

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

Vanessa Giraldo Calderón, con C.C. No. 1075286714, autor(es) de la tesis y/o trabajo de grado o titulado Diseño de un sistema de seguimiento del ojo (pupila), usando técnicas de visión artificial presentado y aprobado en el año 2018 como requisito para optar al título de Ingeniera Electrónica ;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE:

Firma: Vanessa Giraldo Calderón

TÍTULO COMPLETO DEL TRABAJO: Diseño de un sistema de seguimiento del ojo (pupila), usando técnicas de visión artificial.

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
Giraldo Calderón	Vanessa

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Salgado Patrón	José de Jesús

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre

PARA OPTAR AL TÍTULO DE: Ingeniera Electrónica

FACULTAD: Ingeniería

PROGRAMA O POSGRADO: Ingeniería Electrónica

CIUDAD: Neiva **AÑO DE PRESENTACIÓN:** 2018 **NÚMERO DE PÁGINAS:** 73

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas_X__ Fotografías_X_ Grabaciones en discos___ Ilustraciones en general__X__ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros_X_

SOFTWARE requerido y/o especializado para la lectura del documento: No

MATERIAL ANEXO: Si

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Inglés		Español	Inglés
1.	Clasificador Haar	Haar Classifier	**6.**	Trasnformada de Hough	Hough Transform
2.	Segmentación	Segmentation	**7.**	Matlab	Matlab
3.	Filtro de Canny	Canny Filter	**8.**	Visión por Computador	Computer Vision
4.	Raspberry Pi	Raspberry Pi	**9.**	_____	_____
5.	Camara NoIr V2	NoIr Camera V2	**10.**	_____	_____

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

Este documento presenta un algoritmo para el seguimiento del ojo (pupila), usando técnicas de visión artificial con el objetivo de relacionar la posición de la pupila y la ubicación donde está mirando el sujeto, está principalmente dirigido a personas con discapacidad motora en sus miembros superiores e inferiores que conservan la movilidad voluntaria de los músculos de la cabeza. El dispositivo seleccionado para la adquisición de imágenes utilizadas durante el desarrollo de este proyecto es la Pi NoIr Camera V2, diseñada y fabricada por Raspberry Pi. Fue escogida principalmente debido a que no tiene ningún filtro de infrarrojos (NoIr) en la lente, lo que la hace ideal para la toma de fotografías en ambientes con poca luz y para resaltar la pupila en la imagen.

Las imágenes adquiridas inicialmente deben ser sometidas a una etapa de preprocesamiento, donde se aplica el Haar Cascade Classifier con el fin de detectar el área de los ojos, luego de esto se realiza un procesamiento que da como resultado el realce de las pupilas eliminando el ruido y objetos indeseados en la escena. Después se realiza la identificación y seguimiento de las pupilas utilizando la Transformada de Hough. Posteriormente se extraen las coordenadas de posición de las pupilas para identificar hacia donde está mirando la persona, teniendo un 90% de exactitud.

Dentro del diseño del sistema se tuvieron en cuenta las limitaciones por las cuales se puede afectar la confiabilidad de los datos obtenidos, las cuales se reflejan en la posible variación en la iluminación de la escena, la distancia existente entre los componentes del sistema, el ángulo de inclinación dentro de un radio semifijo, la posición estática de la cabeza para evitar errores, establecer la escena en un ambiente controlado, entre otras, con el fin de tener en cuenta los efectos que todo esto pudo presentar en el momento de la identificación.

ABSTRACT: (Máximo 250 palabras)

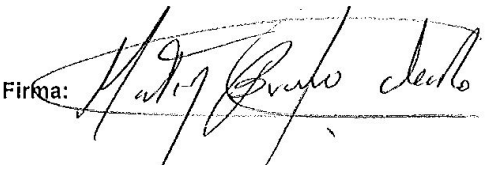
This document presents an algorithm for tracking the eye (pupil), using artificial vision techniques with the aim of relating the position of the pupil and the location where the subject is looking, is mainly aimed at people with motor disability in their upper limbs and inferiors that preserve the voluntary mobility of the muscles of the head. The device selected for the acquisition of images used during the development of this project is the Pi NoIr Camera V2, designed and manufactured by Raspberry Pi. It was chosen mainly because it has no infrared filter (NoIr) in the lens, which makes it ideal for taking photographs in low light environments and to highlight the pupil in the image.

The images acquired initially must be submitted to a preprocessing stage, where the Haar Cascade Classifier is applied to detect the eye area, after that a processing is carried out that results in the enhancement of the pupils eliminating the noise and unwanted objects in the scene. The identification and tracking of the pupils is then performed using the Hough Transform. Subsequently, the position coordinates of the pupils are extracted to identify where the person is looking, with a 90% of accuracy.

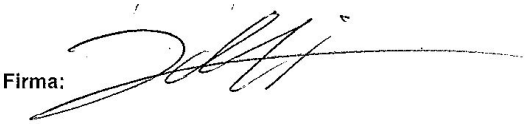
Within the design of the system, the limitations for which the reliability of the obtained data could be affected were taken into account, which are reflected in the possible variation in the illumination of the scene, the distance between the components of the system, the angle of inclination within a semi-fixed radius, the static position of the head to avoid errors, establish the scene in a controlled environment, among others, in order to take into account the effects that all this could have at the time of identification.

APROBACION DE LA TESIS

Nombre Jurado: Martin Diomedes Bravo Obando

Firma: 

Nombre Jurado: Julián Adolfo Ramírez Gutiérrez

Firma: 

DISEÑO DE UN SISTEMA DE SEGUIMIENTO DEL OJO (PUPILA), USANDO TÉCNICAS DE VISIÓN ARTIFICIAL

VANESSA GIRALDO CALDERÓN
Código:20121110998

UNIVERSIDAD SURCOLOMBIANA

FACULTAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
NEIVA-HUILA
2018

DISEÑO DE UN SISTEMA DE SEGUIMIENTO DEL OJO (PUPILA), USANDO TÉCNICAS DE VISIÓN ARTIFICIAL

VANESSA GIRALDO CALDERÓN
Código: 20121110998

Trabajo de grado presentado para optar al título de
Ingeniera electrónica.

Director
ING.JOSÉ DE JESÚS SALGADO PATRÓN

UNIVERSIDAD SURCOLOMBIANA

FACULTAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
NEIVA-HUILA
2018

Nota de aceptación:

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Surcolombiana para optar al título de Ingeniera Electrónica.

José de Jesús Salgado Patrón
Director

Martin Diomedes Bravo Obando
Jurado

Julián Adolfo Ramírez Gutierrez
Jurado

Neiva - Huila, Junio 20 de 2018

A Dios por haberme guiado con sabiduría y darme la fortaleza necesaria para culminar con éxito esta etapa de mi vida.

A mis padres German y Leidy, por ser el motor de mi vida, brindarme todo su amor, animo, confianza y apoyo incondicional para alcanzar mis metas.

A mi hermana Valentina por hacerme su ejemplo a seguir y alentarme a concluir este proyecto.

A mi novio Fernando por darme valor, impulsarme en momentos difíciles y no dejarme desfallecer.

AGRADECIMIENTOS

La autora expresa sus agradecimientos a:

Mis tías Lina y Tatiana, por su apoyo incondicional en cada etapa de mi vida, creer siempre en mí y darme el valor necesario para luchar por este sueño.

El Ingeniero José de Jesús Salgado Patrón por haberme guiado en el proceso siempre con paciencia y disponibilidad, por el tiempo que dedicó para explicarme y aclarar las dudas que surgieron en el camino para culminar con éxito este trabajo.

El Ingeniero Faiber Ignacio Robayo Betancourt por sus consejos, orientación, paciencia y ayuda incondicional desde el primer día de mi pregrado.

El Ingeniero Martin Diomedes Bravo Obando por todas las enseñanzas brindadas, el soporte, discusión critica y confianza que me hicieron dar lo mejor de mí.

Y a todas las personas que de una u otra manera aportaron un granito de arena para la culminación de este trabajo.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

RESUMEN

Este documento presenta un algoritmo para el seguimiento del ojo (pupila), usando técnicas de visión artificial con el objetivo de relacionar la posición de la pupila y la ubicación donde está mirando el sujeto, está principalmente dirigido a personas con discapacidad motora en sus miembros superiores e inferiores que conservan la movilidad voluntaria de los músculos de la cabeza. El dispositivo seleccionado para la adquisición de imágenes utilizadas durante el desarrollo de este proyecto es la Pi NoIr Camera V2, diseñada y fabricada por Raspberry Pi. Fue escogida principalmente debido a que no tiene ningún filtro de infrarrojos (NoIr) en la lente, lo que la hace ideal para la toma de fotografías en ambientes con poca luz y para resaltar la pupila en la imagen.

Las imágenes adquiridas inicialmente deben ser sometidas a una etapa de preprocesamiento, donde se aplica el Haar Cascade Classifier con el fin de detectar el área de los ojos, luego de esto se realiza un procesamiento que da como resultado el realce de las pupilas eliminando el ruido y objetos indeseados en la escena. Después se realiza la identificación y seguimiento de las pupilas utilizando la Transformada de Hough. Posteriormente se extraen las coordenadas de posición de las pupilas para identificar hacia donde está mirando la persona, teniendo un 90% de exactitud.

Dentro del diseño del sistema se tuvieron en cuenta las limitaciones por las cuales se puede afectar la confiabilidad de los datos obtenidos, las cuales se reflejan en la posible variación en la iluminación de la escena, la distancia existente entre los componentes del sistema, el ángulo de inclinación dentro de un radio semifijo, la posición estática de la cabeza para evitar errores, establecer la escena en un ambiente controlado, entre otras, con el fin de tener en cuenta los efectos que todo esto pudo presentar en el momento de la identificación.

Palabras clave: Haar Cascade Classifier, segmentación, filtro de Canny, Transformada de Hough, Visión por Computador.

ABSTRACT

This document presents an algorithm for tracking the eye (pupil), using artificial vision techniques with the aim of relating the position of the pupil and the location where the subject is looking, is mainly aimed at people with motor disability in their upper limbs and inferiors that preserve the voluntary mobility of the muscles of the head. The device selected for the acquisition of images used during the development of this project is the Pi NoIr Camera V2, designed and manufactured by Raspberry Pi. It was chosen mainly because it has no infrared filter (NoIr) in the lens, which makes it ideal for taking photographs in low light environments and to highlight the pupil in the image.

The images acquired initially must be submitted to a preprocessing stage, where the Haar Cascade Classifier is applied to detect the eye area, after that a processing is carried out that results in the enhancement of the pupils eliminating the noise and unwanted objects in the scene. The identification and tracking of the pupils is then performed using the Hough Transform. Subsequently, the position coordinates of the pupils are extracted to identify where the person is looking, with a 90% of accuracy.

Within the design of the system, the limitations for which the reliability of the obtained data could be affected were taken into account, which are reflected in the possible variation in the illumination of the scene, the distance between the components of the system, the angle of inclination within a semi-fixed radius, the static position of the head to avoid errors, establish the scene in a controlled environment, among others, in order to take into account the effects that all this could have at the time of identification.

Keywords: Haar Cascade Classifier, segmentation, Canny filter, Raspberry Pi, NoIr Camera V2, Hough Transform, Matlab, Computer Vision.

INTRODUCCIÓN

La biometría ha tomado gran importancia a través de los años debido a que con ella es posible lograr tareas como la autenticación de un individuo, el seguimiento del movimiento de los ojos, la detección de enfermedades, entre otras. En este sentido, las computadoras son parte fundamental de la tecnología informática, son sistemas digitales capaces de procesar y almacenar información a partir de un grupo de instrucciones llamado programa[1]. Sin embargo, el manejo de un computador tradicional requiere la capacidad física de movimiento de al menos una de las extremidades del ser humano, con lo cual se limita el acceso a personas con cierto tipo de lesiones medulares. De acuerdo con esto, la disposición de una herramienta que permita realizar el seguimiento de pupila y relacionar a donde está mirando una persona, es sin duda alguna, un gran avance de la tecnología biométrica, debido a que permite que cualquier persona con algún tipo de discapacidad motora en sus miembros superiores e inferiores, que además conservan la movilidad voluntaria de los músculos de la cabeza, pueda interactuar con un computador y con esto finalmente poder realizar algunas acciones cotidianas que le asegurarán un mejoramiento en su calidad de vida.

Este proyecto pretende establecer las bases necesarias para realizar futuros proyectos y aplicaciones, fortaleciendo la praxis de visión artificial y procesamiento de imágenes buscando contribuir a la solución de un problema social, siendo esta la primera etapa de un macroproyecto de un sistema asistencial que ayuda a la inclusión y mejoramiento de la calidad de vida de personas con algún tipo de discapacidad motora. En pro de mejorar y optimizar los resultados del proyecto, las muestras se tomaron a nivel personal bajo ambientes controlados, por lo que se trató al máximo de reducir factores como anteojos, maquillaje, pestañas abundantes, mucha iluminación en la escena, movimientos bruscos del rostro, y demás factores que pueden incurrir en adquisición de datos errados, pues no es objeto de estudio en esta investigación tomar muestras en situaciones reales. Lo que se obtuvo finalmente fue un algoritmo capaz de posicionar la dirección de la vista por medio del seguimiento de las pupilas. El algoritmo analiza las imágenes captadas por la cámara, que primero son sometidas a una etapa de preprocesamiento aplicando el Haar Cascade Classifier para detectar el área de los ojos, un procesamiento para eliminar factores indeseados y se continua con una transformada de Hough para identificar y realizar el seguimiento de las pupilas. Seguidamente se acumulan las coordenadas de los centros las pupilas que permiten identificar la posición de la vista por medio de un empalme con el entorno.

[1] E Turégano. EyeBoard: Un Periférico Alternativo Visual. 2006 [revisado febrero 2018]. Disponible en: http://robolab.unex.es/research/doc/libro.pdf

1. OBJETIVOS

1.1. OBJETIVO GENERAL

Diseñar un sistema de seguimiento de la trayectoria del ojo(pupila) usando técnicas de visión por computador.

1.2. OBJETIVOS ESPECIFICOS

- Implementar un sistema de adquisición de imágenes.
- Implementar algoritmos para la detección de la pupila por medio de Matlab.
- Implementar algoritmos para relacionar la posición de la pupila y la ubicación donde está mirando la persona.
- Realizar el código de programación en el software Matlab o Python.
- Diseñar una interfaz de usuario amigable para exponer los resultados.
- Validar y poner a punto los algoritmos.
- Elaborar un documento final exponiendo el trabajo realizado.

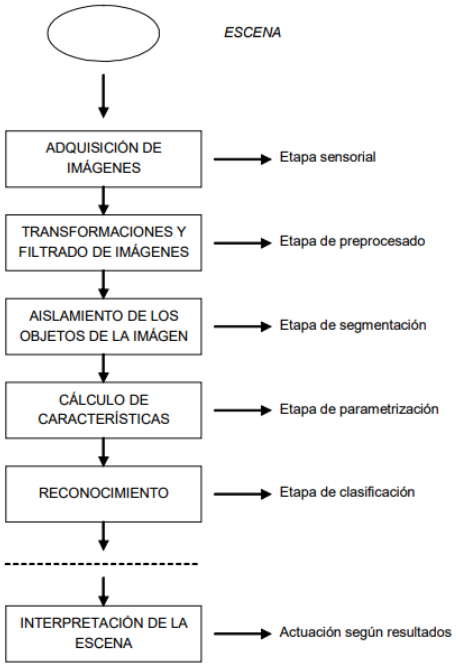
2. MARCO TEÓRICO

2.1. VISIÓN POR COMPUTADOR

La visión por computador o visión artificial es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un computador. Tal y como los humanos usamos nuestros ojos y cerebros para comprender el mundo que nos rodea, la visión por computador trata de producir el mismo efecto para que las computadoras puedan percibir y comprender una imagen o secuencia de imágenes y actuar según convenga en una determinada situación[2].

Las estructuras y propiedades del mundo tridimensional que se quieren deducir en la visión artificial incluyen no solo sus propiedades geométricas, sino también sus propiedades materiales. Ejemplos de propiedades geométricas son la forma, tamaño y localización de los objetos. Ejemplos de propiedades materiales son su color, iluminación, textura y composición[3]. En la Figura 1 se presentan los pasos fundamentales relacionados con hardware, software y desarrollos matemáticos que se aplican a la visión artificial.

Figura 1. Etapas en un proceso de visión artificial (Técnicas y algoritmos básicos de visión artificial, 2006).



[2] Dr. Reinhard Klette. Concise Computer Vision - An Introduction into Theory and Algorithms. (Vol. 1). Department of Computer ScienceThe University of Auckland, New Zealand: Springer, 2014.
[3] González, Marcos et al. Técnicas y algoritmos básicos de visión artificial. Universidad de la Rioja - Servicio de Publicaciones, 2006.

- La primera fase, consiste en la captura o adquisición de las imágenes digitales mediante algún tipo de sensor. Busca conseguir que la imagen sea lo más adecuada posible para que se pueda continuar con las siguientes etapas[4]. Una correcta adquisición de la imagen supone un paso muy importante para que el proceso de reconocimiento tenga éxito[3].

- La segunda etapa consiste en el tratamiento digital de las imágenes, con objeto de facilitar las etapas posteriores. En esta etapa de pre-procesado, mediante filtros y transformaciones geométricas, se eliminan partes indeseables de la imagen o se realzan partes interesantes de la misma. Los algoritmos de preprocesamiento permiten modificar la imagen para eliminar ruido, transformarla geométricamente, mejorar la intensidad o el contraste, etc[3].

- El siguiente paso se conoce como segmentación, y consiste en aislar los elementos que interesan de una escena para comprenderla, es decir, consiste en diferenciar los diversos objetos y donde se encuentran con respecto al fondo de la imagen. Al final de la etapa de segmentación se tienen que conocer perfectamente los objetos que hay, para extraer las características propias de cada uno de ellos[4].

- La parametrización, que recibe también el nombre de extracción de características o selección de rasgos, se dedica a extraer características que producen alguna información cuantitativa de interés o rasgos que son básicos para diferenciar una clase de objetos de la otra: textura, color, orientación, posición, etc[3].

- El reconocimiento es el proceso que asigna una etiqueta a un objeto basada en la información que proporcionan los rasgos diferenciadores (clasificación)[3].

- La interpretación lleva a asignar significado al conjunto de objetos reconocidos[3].

2.2. HAAR CASCADE CLASSIFIER

El clasificador en cascada tipo Haar es un algoritmo de detección de objetos altamente efectivo, propuesto por Paul Viola y Michael Jones y posteriormente mejorado por Rainer Linehart y Jochen Maydt. Se utiliza principalmente para localizar caras, peatones, objetos y expresiones faciales en una imagen.
Este método utiliza cuatro conceptos claves:

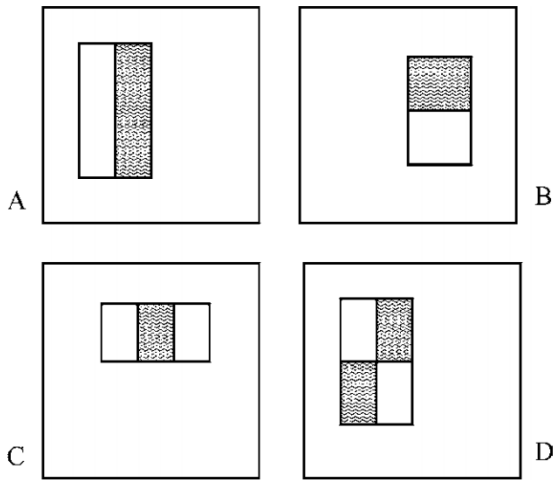
- Funciones rectangulares simples, llamadas funciones Haar.
- Varios números de imágenes positivas y negativas.

[4] Jose F Velez serrano. et al. Visión por computador. S.L. – DYKINSON. Madrid, 2003.

- Una imagen integral para una rápida detección de características
- El método de aprendizaje Adaboost.
- Un clasificador en cascada para combinar muchas funciones de manera eficiente.

Las funciones usadas por Viola y Jones están basadas en Wavelets de Haar. Estos clasificadores están conformados por simples ondas cuadradas en dos dimensiones, en el cual una onda cuadrada es un par de rectángulos del mismo tamaño, adyacentes horizontal o verticalmente (uno claro y otro oscuro)[5] como se muestra en la Figura 2.

Figura 2. Ejemplo. Características de los rectángulos utilizados en el Haar Cascade Classifier. Donde A y B son dos características de rectángulo, y C y D son tres y cuatro características rectángulo *(Viola & Jones, 2004)*.



La suma de los píxeles en los rectángulos blancos se resta de la suma de los píxeles en los rectángulos en gris. Por lo tanto, el resultado del filtro es la diferencia en la suma de los valores de la intensidad de los pixeles entre zonas de color gris y las zonas de color blanco. Lo anterior se puede presentar mediante la Ecuación 1.

$$F_{haar} = \sum I_{RG}(x,y) - \sum I_{RB}(x,y) \qquad \text{(Ecuación 1)}$$

Donde F_{haar} es el resultado del filtro aplicado a una imagen en dos dimensiones, $I_{RG}(x,y)$ es la intensidad de los pixeles en la región gris y $I_{RB}(x,y)$ es la intensidad de los pixeles en la región blanca. Posteriormente este resultado es normalizado debido al tamaño de la ventana del filtro con el fin de que los valores sean invariantes al tamaño de los objetos y comparables a escalas diferentes[6]. Todos los filtros se aplican por toda la imagen en todas las posibles posiciones, es decir, en

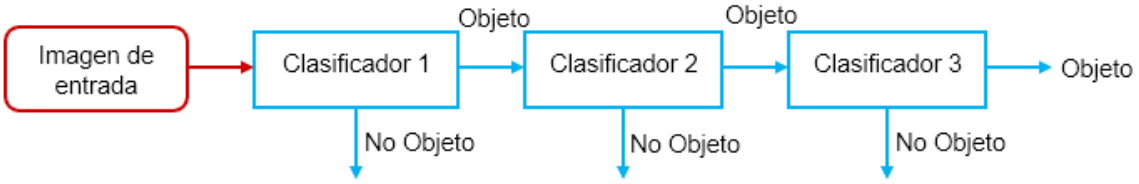
[5] Paul Viola y Michael J Jones. Robust Real-Time Face Detection International Journal of Computer Vision, 2004.

todas las posibles escalas; por ende, el resultado de aplicar cada filtro en cada posición y escala va a ser un componente del vector final de características.

Al aplicar cada uno de estos filtros, la clasificación resultante está compuesta por varias clasificaciones más sencillas combinadas secuencialmente. En este caso cada función Haar que caracteriza el objeto, representa un clasificador sencillo (estados), que son aplicadas subsecuentemente en una región de interés.

Una vez el clasificador ha sido entrenado puede ser aplicado en diferentes regiones de interés a una imagen de entrada, la imagen es aceptada si todos los clasificadores sencillos son admitidos, de lo contrario si uno de ellos no es aceptado, la imagen es rechazada. Los clasificadores de cada estado de la cascada están construidos en base a clasificadores básicos (funciones Haar), usando el método de Boosting, que consiste básicamente en combinar los resultados de varios clasificadores débiles para obtener un clasificador robusto[6].

Figura 3. Cascada de clasificadores. Solo se identifica como un objeto si todos los clasificadores lo aceptan *(Autor).*



Los filtros en cada nivel están capacitados para clasificar imágenes que han pasado todas las etapas anteriores. Durante su ejecución, si alguna región de la imagen no pasa alguno de estos filtros, esa región no califica como un objeto. Cuando la región de la imagen pasa por toda la cadena de filtros es considerada un objeto. Los filtros de mayor peso se aplican primero para eliminar las regiones de la imagen que no son consideradas como un objeto lo más rápido posible. La salida del clasificador será 1 si el área está dentro de los parámetros estipulados en el entrenamiento y 0 si no coincide[6].

2.3. TRANSFORMADA DE HOUGH

La transformada de Hough es una herramienta que permite detectar curvas en una imagen, por consiguiente, es posible encontrar todo tipo de figuras que puedan ser expresadas matemáticamente, tales como rectas, circunferencias o elipses[7].

[6] Antonio López Peña, Ernest Valveny y Maria Vanrell. Detector basado en Haar/Adaboost - Cascada de clasificadores. Coursera [en línea], mayo 2017 [revisado: febrero 2018]. Disponible en: https://es.coursera.org/learn/deteccion-objetos/lecture/pRnHu/l5-5-cascada-de-clasificadores

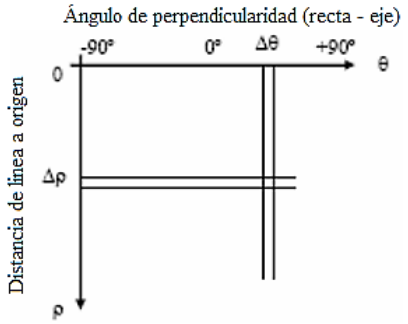
[7] Departamento de Ingeniería electrónica, Telecomunicación y Automática. Área de Ingeniería de Sistemas y Automática. Segmentación. Transformada de Hough. Universidad de Jaén, 2006.

La forma estándar de una línea recta es $y = a + b * x$, la cual requiere dos parámetros para su definición. Esta forma de parametrización es inadecuada para su uso en la detección de líneas en una nube de puntos, debido a que la pendiente de una línea vertical es infinita[8]. El objetivo de la transformada de Hough es encontrar puntos alineados que puedan existir en la imagen, es decir, puntos en la imagen que satisfagan la ecuación de la recta en forma polar, para distintos valores de ρ y θ[7], como se muestra en la Ecuación 2.

$$\rho = x \cos\theta + y \sin\theta \qquad \text{(Ecuación 2)}$$

Donde ρ es la distancia de la línea al origen y θ es el ángulo que forma la perpendicularidad a la recta con el eje. Así, se puede graficar cada punto (x_i, y_i) en la curva del plano (ρ, θ) definido por $\rho = x_i \cos\theta + y_i \sin\theta$[8]. Para aplicar la transformada de Hough es necesario discretizar el espacio de parámetros en una serie de celdas denominadas celdas de acumulación. Cada celda del acumulador representa una figura cuyos parámetros se pueden obtener a partir de la posición de la celda.

Figura 4. Transformada de Hough. Discretización del espacio en celdas de acumulación *(Segmentación. Transformada de Hough, 2006)*.



Esta discretización se realiza sobre los intervalos (ρ_{min}, ρ_{max}) y $(\theta_{min}, \theta_{max})$. El siguiente paso es evaluar la ecuación de la recta para cada punto de la imagen (x_k, y_k), si se cumple esta ecuación se incrementa en uno el número de votos de la celda. Un número de votos elevado indica que el punto pertenece a la recta[7].

2.4. HARDWARE DE ADQUISICIÓN DE IMÁGENES

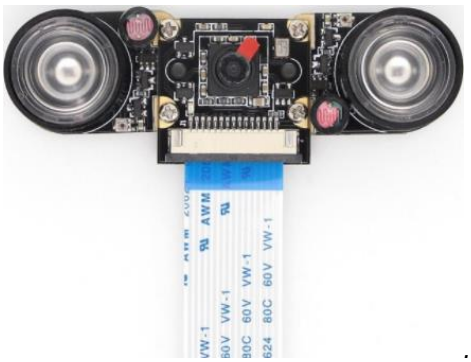
Para la adquisición de las imágenes se debía contar con una cámara y una tarjeta de adquisición que satisficiera los puntos expuestos en el numeral **3.1. Requerimientos.** Para la realización de este proyecto se optó por una Pi NoIr Camera V2 y una Raspberry Pi 3 Model B.

[8] Duda, R. O; Hart, P. E. Pattern Classification and Scene Analysis. 1973. John Wiley & Sons. New York.

2.4.1. Cámara

La Pi NoIr Camera V2 es un módulo de alta calidad de 8 megapíxeles con el sensor de imagen Sony IMX219 diseñado a medida para Raspberry Pi, con una lente de foco fijo y tamaño óptico de 1/4 “. Es capaz de tomar imágenes estáticas de 3280 x 2464 pixeles, y también es capaz de tomar vídeo de 1080p30, 720p60, y 640x480p90. Se conecta a la Raspberry Pi a través de uno de los pequeños conectores en la superficie superior de la placa y utiliza la interfaz dedicada CSi, diseñada especialmente para la interfaz con las cámaras. La Pi NoIr Camera V2 no tiene ningún filtro de infrarrojos (NoIr) en la lente, lo que la hace ideal para hacer fotografía infrarroja y toma de fotografías en ambientes con poca luz[9]. Además de esto, para mejorar la captura de datos, se acoplaron a la cámara dos focos led infrarrojos, como se muestra en la Figura 5. Estos focos eran controlados por dos células fotoeléctricas, donde a medida que la luz en el entorno disminuía la potencia del led aumentaba y por lo tanto la cámara tenía mayor alcance de visión.

Figura 5. Pi NoIr Camera V2 con dos focos led infrarrojos acoplados para aumentar el alcance de visión (Autor).



Los principales motivos para escoger esta cámara fueron, su tecnología NoIr que permitía trabajar en ambientes con poca luminosidad, haciendo que en las imágenes adquiridas se resaltaran las pupilas independientemente de cuál fuera el color de los ojos, su tasa de muestreo de 30 fotogramas por segundo garantizaba una buena fluidez de las imágenes sin comprometer la calidad del video. Adicionalmente la cámara permitía elegir entre múltiples resoluciones de video dando la ventaja de probar cuál de ellas se acoplaba especialmente a los requerimientos del sistema para brindar mejores resultados.

2.4.2. Tarjeta de adquisición

Para el funcionamiento y control de la Pi NoIr Camera V2, se contaba con una Raspberry Pi 3 Model B, la cual era ideal para el proyecto debido a que cuenta con conectividad Bluetooth Low Energy (BLE) y Wi-Fi integrados, lo cual permitía controlar el dispositivo remotamente desde diferentes aplicaciones y programas, sin

[9] Pi NoIr Camera V2. Raspberry Pi. Febrero 2017 [revisado febrero 2018]. Disponible en internet: https://www.raspberrypi.org/products/pi-noir-camera-v2/

importar en que sistema operativo se ejecutaran estos; como es el caso de Matlab, programa desde el cual se realizaba una conexión en tiempo real y se ejecutaban funciones de control del dispositivo y periféricos. El proceso de conexión de la Raspberry Pi con Matlab se expone en el **Anexo A.**

Adicional a esto, cabe resaltar que este modelo de la Raspberry cuenta con un potente chip Broadcom BCM2837, un ARM Cortex-A53 64bit Quad Core, la GPU proporciona Open GL ES 2.0, aceleración por hardware OpenVG y un perfil de decodificación alto de 1080p30 H.264 y es capaz de generar 1Gpixel/s, 1.5Gtexel/s o 24 GFLOPs de procesamiento de cómputo, con lo cual cumple con las propiedades necesarias expuestas en el numeral **3.1 REQUERIMIENTOS**. La administración de energía era por medio de una fuente de alimentación conmutada que alcanza hasta 2,5 amperios, para soportar dispositivos USB externos más potentes[10].

Figura 6. Raspberry Pi Model B - Computadora de placa única con LAN inalámbrica y conectividad Bluetooth *(Raspberry Pi, 2018)*.

[10] Raspberry Pi Model B. Raspberry Pi. Febrero 2017 [revisado febrero 2018]. Disponible en internet: https://www.raspberrypi.org/products/pi-noir-camera-v2/

3. CONSIDERACIONES

3.1. REQUERIMIENTOS DE AMBIENTE CONTROLADO

Una parte importante cuando se realizó el diseño del sistema, fueron los requerimientos de ambiente controlado que se tuvieron en cuenta para poder alcanzar el objetivo. Teniendo en cuenta lo anterior, se analizaron los requerimientos respecto a las necesidades que se debían cumplir para que el sistema tuviese una buena fiabilidad, autenticidad y robustez dentro de un ambiente controlado. A continuación, se definen las funciones que el sistema podía realizar y las características que pudieron limitarlo.

- El sistema debe establecer una conexión exitosa con la cámara para adquirir las imágenes de entrada, teniendo en cuenta factores externos que podían afectar la escena.

- Se requiere que la cámara se encuentre ubicada correctamente, sin obstáculos. La iluminación del lugar donde se hacen las pruebas debe presentar características iguales o similares siempre, es decir, un ambiente controlado.

- El rostro de la persona tiene que encontrarse frente a la cámara, con una posición definida y sin hacer ningún tipo de gesto, solo se debe realizar el movimiento de los ojos en diferentes direcciones.

- La cámara debe tener propiedades aceptables de resolución, velocidad y no debe contar con filtro infrarrojo con el fin de destacar las pupilas.

- Las distancias entre la cámara, los ojos y el tablero deben ser invariantes y equidistantes, con el fin de obtener un sistema más robusto.

- Es necesario que la Raspberry Pi y el computador donde se ejecuta el programa se encuentren dentro de la misma red Wi-Fi.

- La herramienta de programación debe tener herramientas de visualización integradas, disponibilidad de librerías y documentación, un entorno de desarrollo interactivo y un amplio soporte de hardware.

3.2. CALIBRACIÓN

Con el fin de garantizar el correcto funcionamiento del sistema es necesario realizar una calibración de la cámara con respecto a la posición en la que se encuentran los ojos al momento de realizar la medición. La correcta calibración de la cámara proporciona la seguridad de que los resultados entregados sean correctos y se mitiguen los riesgos de cometer errores.

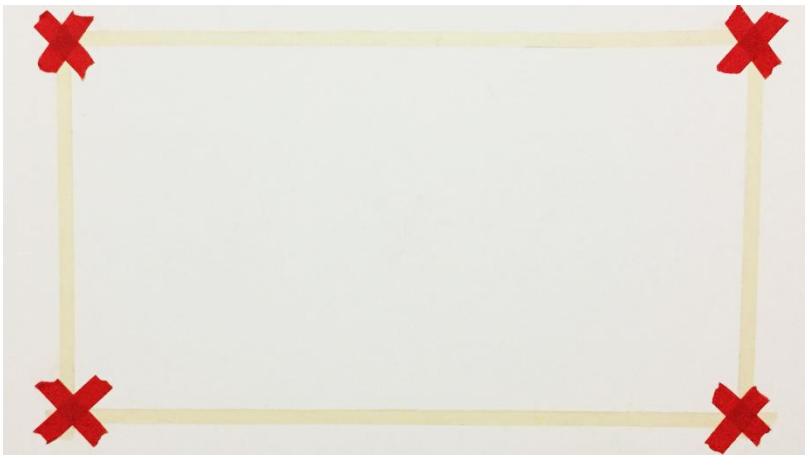
Inicialmente cuando se realiza la calibración es necesario ubicar algunos marcadores en el rostro de la persona y en el tablero de posicionamiento, de la siguiente manera:

- El marcador del rostro era un sticker en forma de cruz color negro que se debe colocar en la mitad de los dos ojos, como se muestra en la Figura 7:

Figura 7. Ubicación de marcador en el rostro para realizar la calibración *(Autor).*

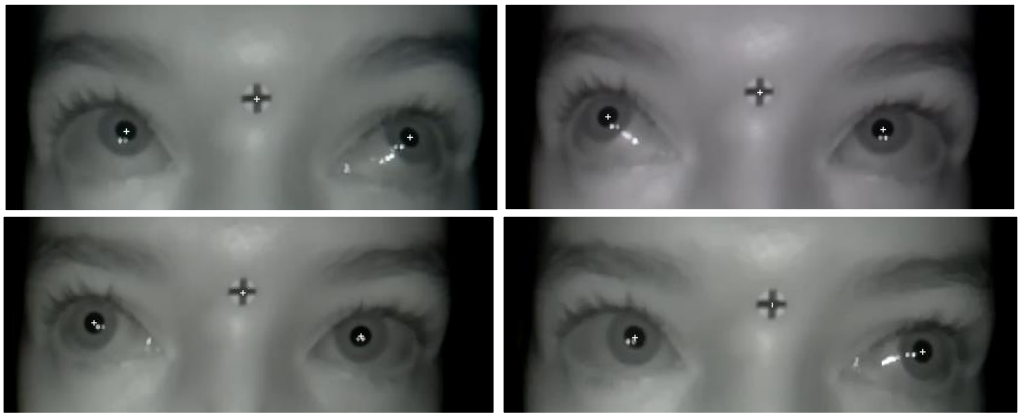
- En cada una las esquinas del tablero de posicionamiento se ubican cuatro equis (X) de color rojo para establecer los límites del recuadro que el ojo iba a mirar como se muestra en la Figura 8:

Figura 8. Ubicación de marcadores en el tablero de posicionamiento para realizar la calibración *(Autor).*



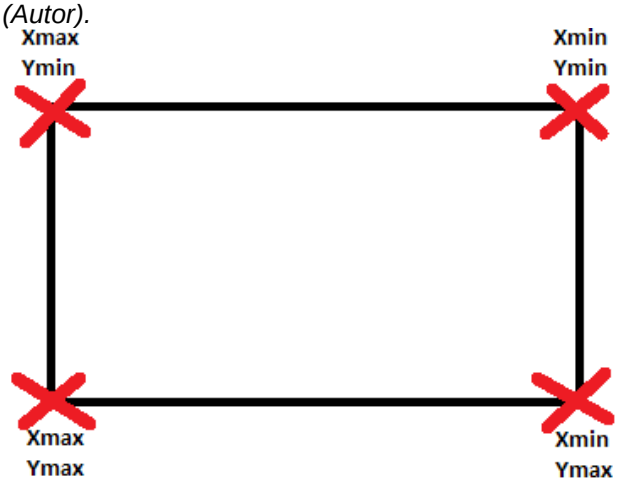
Una vez se encuentran ubicados los marcadores, la persona debe ubicarse frente a la cámara y mirar simultáneamente y por algunos segundos cada uno de los marcadores establecidos en el tablero. A medida que la persona observa fijamente cada una de las equis (X), el sistema almacena en dos variables los pixeles de posición de los ojos y el pixel del centro del marcador ubicado en el rostro como se muestra en la Figura 9.

Figura 9. Calibración de la cámara mirando los cinco puntos del tablero de posicionamiento e identificación de pixeles de posición de los ojos y del marcador del rostro, con una cruz color blanco *(Autor).*



Los pixeles almacenados en cada posición del tablero se utilizan para construir dos planos (un plano por cada ojo) de dos dimensiones de coordenadas (x,y) donde se establecen los valores máximos y mínimos que podían tener las posiciones de los ojos en determinado momento y de esta manera poder realizar el escalado de los pixeles al plano real o tablero de posicionamiento.

Figura 10. Plano de dos dimensiones de coordenadas (x, y), con ubicación de valores máximos y mínimos de los pixeles *(Autor).*



El marcador del rostro se utiliza como centro de referencia, es decir, un punto invariable en el espacio 2D de los fotogramas del rostro (equivalente a un punto invariable en el espacio 3D de movimientos de la cabeza), el cual corresponde a la posición neutra del rostro. Gracias a esto es posible identificar si existe un leve movimiento en la posición de la cara y por ende en la posición de los ojos en algún momento determinado. Al identificar la posición del pixel del marcador se puede establecer la distancia que este se desplaza y así corregir errores que alteran la medición del posicionamiento de los ojos.

Las medidas recomendadas para un óptimo funcionamiento del sistema son:

- La cámara debe encontrarse en un rango de distancia de 20cm a 25cm de los ojos con una inclinación aproximada de 0°.

- El tablero debe estar máximo a 75cm de distancia de la cámara y debe tener dimensiones mínimas de 55x30 cm.

- El tablero tiene que encontrarse en lo posible sobre la línea de vista de los ojos con un ángulo no mayor a 15°.

- La mesa debe tener entre 50cm y 70cm de alto, los ojos y la cámara deben ubicarse aproximadamente a 50cm y el tablero aproximadamente a 40cm sobre la base de la mesa.

3.3. SELECCIÓN DE TÉCNICAS DE PROCESAMIENTO DE IMÁGENES

Para el funcionamiento del sistema de seguimiento del ojo (pupila) es necesario incorporar técnicas preestablecidas de detección e identificación de características, las cuales se seleccionan al realizar un análisis y una comparación de propiedades de diferentes técnicas encontradas.

3.3.1. Detección de ojos

Para seleccionar la mejor técnica de detección de ojos se realizó una comparación de características de rapidez, precisión de detección y las principales aplicaciones de tres métodos de detección ampliamente utilizados que empleaban clasificadores en cascada: Haar, LBP (Patrones Binarios Locales) y HOG (Histograma Orientado a Gradientes). En la Tabla 1 se exponen las características de cada uno.

De acuerdo con la Tabla 1 se puede evidenciar que la mejor técnica de detección para este proyecto es la de Haar, aunque su velocidad de procesamiento es inferior con respecto a los otros métodos, tiene la ventaja que su principal aplicación es la detección de rostros y su morfología (ojos, nariz, boca, etc), presentando una precisión de aproximadamente el 95%. Además de esto, al realizar la consulta de cada una de las técnicas, la que presenta mayor número de fuentes de información y documentación es la de Haar, en efecto, se encontraron más trabajos, investigaciones y aplicaciones de esta técnica presentando un fácil acceso a la información.

Tabla 1. Cuadro comparativo de características de tres métodos de clasificación en cascada *(Mathworks, Train a Cascade Object Detector)*, *(Haar Cascades vs. LBP Cascades en detección de rostros, 2012)*.

Característica	Haar	LBP	HOG
Rapidez	• Velocidad de procesamiento 3 veces más lenta que LBP.	• Mayor velocidad de procesamiento que Haar y HOG.	• Velocidad de procesamiento menor que LBP.
Precisión detección	• La detección puede operar en el rango de precisión del 95%.	• La detección de tiene una precisión del 80% al 85%.	• La detección opera en el rango de 85% de precisión[11].
Aplicación	• Detección de características morfológicas (Rostros, Ojos, Boca, Personas, etc). • Representación de texturas en escala fina.	• Detección de personas, rostros y objetos. • Representación de texturas en escala fina.	• Detección de personas en general, animales y objetos como automóviles. • Captura la forma de los objetos de manera general[12].

3.3.2. Identificación de pupilas como círculos

Para la identificación de las pupilas en la imagen es necesario utilizar una técnica efectiva a la hora de encontrar círculos, debido a que es la figura geométrica que más se asemeja a una pupila. En la Tabla 2 se evidencian las características analizadas de dos técnicas de identificación de círculos.

Tabla 2. Cuadro comparativo de dos técnicas de identificación de círculos en una imagen *(Mathworks, Hough Transform)*, *(Visión Artificial Industrial, 2012)*.

Característica	Formula de circularidad	Transformada Hough
Precisión	• La precisión se encuentra en un rango del 85%.	• La identificación opera en un rango de precisión del 97%.
Procesamiento	• Se necesitan varias líneas de código que aumentan el tiempo de procesamiento.	• Solo una línea de código que no aumenta considerablemente el tiempo de procesamiento.
Aplicación	• Los círculos no deben presentar deformaciones.	• Reconoce círculos así presente deformaciones.

Como se expone en la Tabla 2 la mejor técnica de identificación de círculos en una imagen es la aplicación de la transformada de Hough debido a que presenta una precisión mayor al 90%, lo que hace que los resultados sean confiables, como también presenta la propiedad de reconocer círculos así no tengan una forma

[11] Haar Cascades vs. LBP Cascades en detección de rostros. Stack Overflow. 12 de enero de 2012 [Revisado: mayo 2018]. Disponible en internet: https://stackoverflow.com/questions/8791178/haar-cascades-vs-lbp-cascades-in-face-detection.

[12] Train a Cascade Object Detector. Mathworks. [Revisado: mayo 2018]. Disponible en internet: https://la.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html

circular perfecta. Otra ventaja que presenta esta técnica es que solo requiere de una línea de código como se expone en el numeral **4.2.10 Identificación de pupilas como círculos con Transformada de Hough**, lo cual no afecta considerablemente el tiempo de procesamiento.

4. DESARROLLO DEL ALGORITMO DE PROCESAMIENTO

4.1. ENTORNO DE PROGRAMACIÓN

Para implementar el código se escogió el software Matlab 9.0 (R2016a) (The Mathworks, Natick, Massachusetts, USA) debido a la facilidad de implementación de los algoritmos de procesamiento, disponibilidad de librerías, amplia integración de software, su plataforma de desarrollo de interfaz de usuario y demás ventajas que poseía en comparación con el software Python[13]. Las características de interés de cada software analizadas para este proyecto se evidencian en la Tabla 3.

Tabla 3. Cuadro comparativo de características de los softwares Matlab y Python *(Comparación entre MATLAB y Python: principales razones para elegir MATLAB)* **.**

Prestación	Matlab	Python
Sintaxis matemática	• Lenguaje orientado a matemáticas y matrices.	• Lenguaje de uso general. • Los arrays numéricos y los tipos de datos no forman parte del lenguaje básico.
Visualización de datos científicos	• Visualización integrada en la base del lenguaje y el escritorio de MATLAB. • Exploración y anotaciones interactivas.	• Herramienta de visualización fragmentada. • Sin interfaz común.
Toolboxes que funcionan	• Toolboxes perfeccionadas proporcionan cobertura para aplicaciones científicas y de ingeniería habituales. • Toolboxes diseñadas para colaborar entre ellas. • Toolboxes documentadas por profesionales en un lenguaje específico de dominio.	• La calidad, la exhaustividad y el mantenimiento de los paquetes varía ampliamente. • Documentación escrita por desarrolladores que suele dar por sentadas las habilidades de un desarrollador de software. Documentación no integrada entre librerías.
Rendimiento inicial	• Estadística, gráficos y álgebra lineal de alto rendimiento. • Llamadas a librerías optimizadas. • Multithreading implícito. • Compilación just-in-time para bucles y llamadas de función rápidos.	• El rendimiento mejorado requiere instalar, compilar, validar y adoptar complementos orientados a desarrolladores.

[13] Comparación entre MATLAB y Python: principales razones para elegir MATLAB. Mathworks. Revisado mayo 2018. Disponible en internet: https://la.mathworks.com/products/matlab/matlab-vs-python.html.

Tabla 3. *Continuación.*

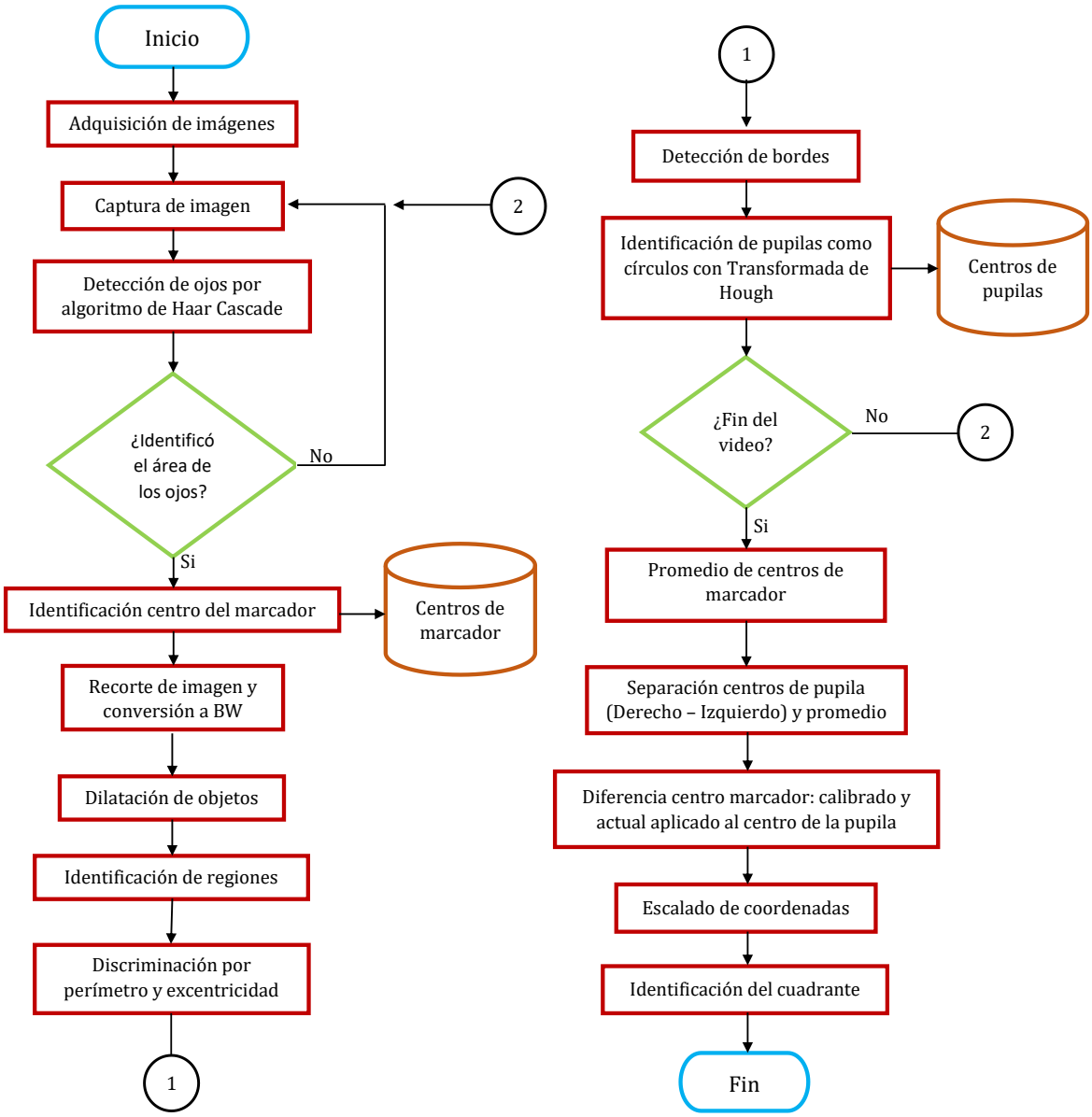
Prestación	Matlab	Python
Escritorio de cálculo científico	• Entorno de desarrollo perfeccionado para facilitar la exploración y descubrimiento. • Entorno de desarrollo, lenguaje y librerías integrados para que funcione en conjunto. • Las herramientas de exploración de datos mejoran los flujos de trabajo de análisis y depuración. • Un depurador, un analizador de rendimiento y un analizador de código integrados mejoran la calidad de los programas.	• La mayoría de los entornos de desarrollo integrados (IDEs) están diseñados para desarrolladores de software profesionales. • Niveles muy distintos de integración entre IDEs y librerías.
Documentación	• Referencias y documentación de usuario exhaustivas. • Visor de documentación integrado con flujos de trabajo. • Documentación redactada para ingenieros y científicos. • Miles de ejemplos.	• Documentación de usuario a menudo limitada en cuanto a profundidad y amplitud. • Documentación de distintas librerías dispersa en diversos sitios web. • La documentación suele dar por sentado un conocimiento de la programación avanzada.
Desarrollo y distribución de aplicaciones de escritorio	• Plataforma de desarrollo de IU interactiva. • Empaquetado de ejecutables independientes.	• Sin plataforma de desarrollo de IU interactiva. • Herramientas para desarrolladores de IU profesionales que requieren experiencia en programación avanzada.
Integración de hardware	• Amplio soporte de hardware. Instalación y configuración simples y automatizadas. • Interfaces de alto nivel coherentes y fáciles de usar.	• Soporte ad-hoc limitado para la integración con hardware. • Instalación y configuración normalmente difíciles y largas.

De acuerdo con lo expuesto en la Tabla 3, se puede evidenciar que el programa Matlab es el software óptimo para la ejecución del algoritmo debido a las ventajas que presenta sobre el programa Python, teniendo en cuenta lo expuesto en el numeral **3.1 Requerimientos.**

4.2. ALGORITMOS[14]

El software del sistema consta de los pasos que se muestran en el algoritmo del diagrama de flujo de la Figura 11. Con el fin de evidenciar el funcionamiento del software, a lo largo de este capítulo se explicará cada paso detalladamente.

Figura 11. Diagrama de flujo del algoritmo del sistema de seguimiento del ojo (pupila), usando técnicas de visión artificial (Autor).



[14] Mathworks. Soporte y documentación de funciones del programa Matlab. Mathworks. Revisado: mayo 2018. Disponible en internet: https://la.mathworks.com/support.html?s_tid=gn_supp

4.2.1. Adquisición de imágenes

La adquisición de las imágenes se realiza estableciendo una conexión con la Raspberry Pi y la Pi NoIr Camera V2 de la siguiente manera:

```
clear rpi
rpi = raspi();
cam = cameraboard(rpi, 'Resolution', '1024x768');
```

Inicialmente se elimina la conexión con la Raspberry Pi del workspace aplicando `clear rpi` luego de esto se volvía a realizar la conexión ejecutando `rpi = raspi()` donde `rpi` devolvía información para interactuar con el hardware de la siguiente manera:

```
           DeviceAddress: '192.168.1.7'
                 Puerto: 18725
             Nombre de la placa: 'Raspberry Pi Model B Rev 2'
          Leds disponibles: {'led0'}
PigsDigital disponibles: [4 7 8 9 10 11 14 15 17 18 22 23 24 25 27 30 31]
AvailableSPIChannels: {}
  AvailableI2CBuses: {'i2c-0' 'i2c-1'}
         I2CBusSpeed: 100000
```

Para finalizar, se inicia la conexión con la cámara ejecutando `cam = cameraboard(rpi, 'Resolution', '1024x768')` donde es necesario nombrar la conexión preestablecida con la Raspberry Pi (`rpi`) y se establece la resolución deseada para las imágenes (`'Resolution', '1024x768'`).

4.2.2. Captura de imagen

Después de adquirir las imágenes, se realiza la captura de la imagen a procesar de la siguiente manera:

```
captura_video = snapshot(cam);
```

Donde, por medio de la función `snapshot` se realiza el almacenamiento de una imagen para ser procesada, indicando el nombre de la cámara desde la cual se va a hacer la adquisición (`cam`), esta función devuelve una imagen en RGB en la resolución preestablecida, es decir, crea una matriz de *768 x 1024 x 3* pixeles.

4.2.3. Detección de ojos por algoritmo de Haar Cascade

Para la detección de los ojos se utiliza el clasificador pre entrenado Haar Cascade Classifier que utiliza el algoritmo de Viola & Jones disponible en la Computer Vision System Toolbox de Matlab de la siguiente manera:

```
detector_ojos = vision.CascadeObjectDetector('EyePairBig')
detector_ojos.MinSize=[80 150];
ojos=step(detector_ojos, captura_video);
```

Inicialmente se crea el objeto `vision.CascadeObjectDetector` donde se deben establecer sus propiedades de funcionamiento, en este caso se usa la propiedad de detección de ojos grandes (`'EyePairBig'`) la cual detecta los ojos que estén verticales y mirando hacia adelante; Este modelo usa las características de Haar para codificar los detalles de la región de los ojos. Además de esto, se ajustan los parámetros del tamaño mínimo que deben tener los ojos para ser detectados con la función `detector_ojos.MinSize`. Las pruebas realizadas para seleccionar este tamaño se evidencian el numeral **6.1 ADQUISICIÓN DE IMÁGENES Y DETECCIÓN DE OJOS.**

Después de parametrizar el clasificador, por medio de la función `step` se aplica a la imagen (`captura_video`) el detector de ojos (`detector_ojos`); esto entrega un vector de cuatro variables que contiene la información de los pixeles que forman un rectángulo que encierra el área de los ojos en un rectángulo como se muestra en la Tabla 4.

Tabla 4. Información de cada posición del vector al aplicar el detector de ojos a la imagen *(Autor)*.

Posición	**1**	**2**	**3**	**4**
Información	Coordenada en X del pixel superior izquierdo del rectángulo	Coordenada en Y del pixel superior izquierdo del rectángulo	Distancia en pixeles del rectángulo en X (Ancho)	Distancia en pixeles del rectángulo en Y (Largo)

Si no es posible detectar el área de los ojos en la imagen, se realiza una nueva captura de imagen, es decir, se regresa a lo expresado en el numeral **4.2.2 Captura de imagen.**

4.2.4. Identificación centro de marcador

Luego de detectar los ojos en la imagen, se procede a identificar el centro del marcador ubicado en el rostro como se muestra en el numeral **3.2 CALIBRACIÓN** aplicando la siguiente función:

```
centro_marcador=detectar_marcador(captura_video,0.4)
```

Para conocer el pixel del centro del marcador se introduce la imagen (`captura_video`) y el número del umbral de binarizacion (`umbral`) el cual se establece en un valor de 0.4, debido a que así se resalta de la mejor manera la cruz del marcador.

```
function centro_marcador = detectar_marcador(captura_video, umbral)
im_gray = rgb2gray(captura_video);
rec = imcrop(im_gray,[530,50,150,250]);
rec_bw = imbinarize(rec,umbral);
rec_bw = ~ rec_bw;
rp = regionprops(rec_bw);
```

```
centro_marcador = rp(1).Centroid;
centro_marcador(1) = centro_marcador(1) + 530;
centro_marcador(2) = centro_marcador(2) + 50;
```

Inicialmente a la imagen se le realiza un procesamiento que consiste en convertirla de RGB a escala de grises (`rgb2gray`), realizar un recorte de esta para que el único objeto en la imagen sea el marcador (`imcrop`) donde las dimensiones del recuadro (`[530,50,150,250]`) se establecen de la misma manera que en la Tabla 4. Luego de esto se aplica la umbralización mediante la función `imbinarize` la cual crea una imagen binaria remplazando todos los valores de la matriz por encima del umbral (0.4) con 1s y configurando todos los demás valores con 0s, además se invierten los pixeles de la matriz convirtiendo los 1s en 0s y viceversa (`~rec_bw`). Luego de esto, se utiliza la función `regionprops` para encontrar la cruz en la imagen por medio de la identificación de regiones como objetos y se utiliza la propiedad `Centroid` para encontrar el pixel de centro de la cruz. Para finalizar se aumenta la posición del pixel tanto en X (`centro_marcador(1)`) como en Y (`centro_marcador(2)`) para que sea proporcional a la imagen original. La posición del pixel de centro en cada imagen es almacenada de la siguiente manera:

```
g=g+1;
if g==1
   cruz=centro_marcador;
else
   cruz=[cruz;centro_marcador];
end
```

En la primera imagen procesada, cuando `g==1` se crea la variable `cruz` igualándola a `centro_marcador`; para las siguientes imágenes `cruz` se guarda como una matriz donde se agrega la coordenada del nuevo pixel de centro (`centro_marcador`) en la última posición.

4.2.5. Recorte de imagen y conversión a BW

El recorte y conversión a blanco y negro de la imagen se realiza ejecutando las siguientes funciones:

```
imrec=imcrop(captura_video,ojos);
imrecBW = imbinarize(rgb2gray(imrec),0.15)
imrecBW= ~ imrecBW;
```

Para realizar el recorte de la imagen en el área de los ojos se utiliza la función `imcrop` donde las dimensiones del recuadro de recorte se definen por el vector `ojos` como se explica en la Tabla 4. Luego de esto la imagen se pasa a escala de grises (`rgb2gray`) y posteriormente se aplica la binarizacion (`imbinarize`) con un umbral de 0.15 debido a que así se resaltan de la mejor manera las pupilas sin la presencia de mucho ruido; esta función devuelve una imagen binaria remplazando todos los valores de la matriz por encima del umbral con 1s y configurando todos los demás

valores con 0s, para finalizar se invierten los pixeles de la matriz convirtiendo los 1s en 0s y viceversa (`~imrecBW`).

4.2.6. Dilatación de objetos

La dilatación de los objetos en la imagen se realiza con el fin de expandir el tamaño de las pupilas en forma circular para que su identificación sea más efectiva, aplicando las siguientes funciones:

```
SE = strel('disk',3);
imrecBW = imdilate(imrecBW, SE);
```

Primero con la función `strel` se define el elemento estructurante (`SE`) en forma de disco (`'disk'`) y el tamaño de su radio de máximo 3, debido a que con este el ruido no afecta de manera considerable la forma de las pupilas. Cabe resaltar que el elemento estructurante es un vecindario con valores binarios, ya sea bidimensional o multidimensional, en el que los píxeles verdaderos se incluyen en el cálculo morfológico y los píxeles falsos no.

Luego de esto se usa la función `imdilate` para dilatar la imagen binaria (`imrecBW`) aplicando el elemento estructurante `SE` . Esta función toma cada píxel del objeto (con valor 1) y establecía al valor de 1 todos aquellos píxeles pertenecientes al fondo que tienen una conectividad circular con el píxel del objeto. En pocas palabras, pone en 1 los píxeles del fondo vecinos a los píxeles del objeto[15].

4.2.7. Identificación de regiones

La detección de objetos en la imagen se realiza por medio de la identificación de regiones de la siguiente manera:

```
region=regionprops(imrecBW,'BoundingBox','Perimeter','Eccentricity'
```

Aplicando la función `regionprops` se identifican las regiones existentes en la imagen, donde se debe especificar la imagen a tratar (`imrecBW`) y las características que se necesitaban conocer del objeto (`BoundingBox`,`Perimeter`,`Eccentricity`). Esta función devuelve una matriz donde se especifican los valores del cuadro delimitador, el perímetro y la excentricidad de cada objeto.

4.2.8. Discriminación por perímetro y excentricidad

La discriminación por perímetro y excentricidad se realiza identificando las regiones mayores y menores a un rango determinado de la siguiente manera:

```
smen=find([region.Perimeter]<100);
smay=find([region.Perimeter]>400);
exc=find([region.Eccentricity]>0.63);
```

[15] Cáceres Tello Jesús. La visión artificial y las operaciones morfológicas en imágenes binarias. 2015. Servicios Informáticos. Escuela Técnica Superior de Informática. Universidad de Alcalá. España.

```
for n=1:size(smen,2)
  d=round(region(smen(n)).BoundingBox);
  imrecBW(d(2):d(2)+d(4),d(1):d(1)+d(3))=0;
end
for m=1:size(smay,2)
   d=round(region(smay(m)).BoundingBox);
   imrecBW(d(2):d(2)+d(4),d(1):d(1)+d(3))=0;
end
for m=1:size(exc,2)
   d=round(region(exc(m)).BoundingBox);
   imrecBW(d(2):d(2)+d(4),d(1):d(1)+d(3))=0;
end
```

Con la función `find` se encuentran los perímetros menores a 100 y mayores a 400 y las excentricidades mayores a 0,63, para seleccionar este rango de valores se realizaron una serie de pruebas que se exponen en el numeral **6.2 PROCESAMIENTO**. Para discriminar los objetos con perímetro por fuera de este rango, con la función `round` se redondea cada elemento encontrado con los valores del cuadro delimitador (`BoundingBox`), para luego por medio de un barrido igualar a 0 los pixeles que se encuentran dentro de ese recuadro.

```
imrecBW(d(2):d(2)+d(4),d(1):d(1)+d(3))=0.
```

4.2.9. Detección de bordes

Luego de discriminar las regiones diferentes a las pupilas se realizaba la detección de bordes aplicando el filtro de Canny con la siguiente función:

```
contorno=edge(imrecBW,'canny');
```

Donde la función `edge` devuelve una imagen que contiene 1s donde la función encuentra bordes y 0s en otro lugar usando el algoritmo de Canny el cual realiza un adelgazamiento del ancho de los bordes a un pixel. El análisis de selección del método de Canny como el detector de bordes se evidencia en el numeral **6.2 PROCESAMIENTO.**

4.2.10. Identificación de pupilas como círculos con Transformada de Hough

La identificación de las pupilas como círculos se realizaba utilizando la función `imfindcircles` de la siguiente manera:

```
[centers, radii] = imfindcircles(contorno,[15 95]);
viscircles(centers, radii,'EdgeColor','b');
centers(:,1) = double(centers(:,1)) + double(ojos(1));
centers(:,2) = double(centers(:,2)) + double(ojos(2));
```

Esta función viene preestablecida para encontrar círculos en la imagen (`contorno`) usando la Transformada de Hough, cuyos radios son aproximadamente iguales a lo determinado (`[15 95]`). La selección de este rango se expone en el numeral **6.3**

IDENTIFICACIÓN DE POSICIÓN DE PUPILAS; esta función devuelve las coordenadas de los centros en una matriz y los radios estimados correspondientes a cada centro de círculo. Con la función `viscircles` se encierran las pupilas en círculos de color azul (`'EdgeColor','b'`) para su visualización utilizando los resultados de `centers` y `radii`.

Luego de esto se aumenta la posición de cada pixel tanto en X (`centers(:,1)+double(ojos(1))`) como en Y (`centers(:,2)+ double(ojos(2))`) para que sea proporcional a la imagen original. La posición de los pixeles de centro de las pupilas en cada imagen es almacenada de la siguiente manera:

```
c=c+1;
if c==1
   centros=centers;
else
   centros=[centros;centers];
end
```

En la primera imagen procesada, cuando `c==1` se crea la variable `centros` igualándola a `centers`; para las siguientes imágenes `centros` se guard como una matriz donde se agrega las coordenadas de los nuevos pixeles de centro de las pupilas (`centers`) en la última posición. Luego de esta etapa se regresa a lo expresado en el numeral **4.2.2 Captura de imagen** hasta aplicar el proceso a todas las imágenes adquiridas.

4.2.11. Promedio de centros de marcador

Al finalizar la etapa de procesamiento de todas las imágenes, se procedía a encontrar el promedio de las posiciones de los centros del marcador con el fin de determinar una única coordenada como punto de referencia de la siguiente manera:

```
cruzx=cruz(:,1);
cruzy=cruz(:,2);
cruzx=mean(cruzx);
cruzy=mean(cruzy);
```

Inicialmente se separaban las variables X (`cruz(:,1)`) de Y (`cruz(:,2)`) para posteriormente ser promediadas al ejecutar la función `mean`; esta función devuelve el valor promedio de cada vector: `cruzx` y `cruzy`.

4.2.12. Separación centros de pupila (Derecho - Izquierdo) y promedio

Después de realizar la acumulación de los pixeles de centro de las pupilas en una matriz y conocer el pixel promedio del centro del marcador, los centros de las pupilas deben clasificarse con el fin de identificar si pertenen al ojo derecho o izquierdo de la siguiente manera:

```
cx=centros(:,1);
cy=centros(:,2);
```

```
derecho=find(cx<cruzx);
derechox=cx(derecho);
derechoy=cy(derecho);

izquierdo=find(cx>cruzx);
izquierdox=cx(izquierdo);
izquierdoy=cy(izquierdo);

mx=mean(izquierdox);
my=mean(izquierdoy);
```

Inicialmente se separan las variables X (`centros(:,1)`) de Y (`centros(:,2)`) y posteriormente con la función `find` se encuentran las posiciones en el vector donde los pixeles en X eran menores a `cx<cruzx` para el caso del ojo derecho y mayores a `cx>cruzx` en el caso del ojo izquierdo. Luego de tener las posiciones que pertenen a cada ojo, se crea un vector por X (`derechox` e `izquierdox`) y uno por Y (`derechoy` e `izquierdoy`) para almacenar estos pixeles en correspondencia de sus posiciones. Cabe resaltar que se escogen los resultados de solo un ojo, en este caso el izquierdo, debido a que realizar el proceso faltante para los dos ojos resulta redundante en consecuencia de que con cualquiera el resultado es el mismo. Para finalizar con la función `mean` se halla el promedio de la posición del centro de la pupila del ojo izquierdo con el fin de determinar una única coordenada.

4.2.13. Diferencia marcador: calibrado y actual aplicado a centro de la pupila

Para identificar si hubo movimiento en la posición del rostro con respecto a la calibración, se realiza una diferencia entre las coordenadas del centro del marcador calibrado y el centro del marcador actual (`cruzx` y `cruzy`) así:

```
dif_cruzx=cruzx_calibrado-cruzx;
dif_cruzy=cruzy_calibrado-cruzy;

px=mx+dif_cruzx;
py=my+dif_cruzy;
```

Las coordenadas del centro del marcador calibrado (`cruzx_calibrado` y `cruzy_calibrado`) se establecen cuando se realiza la calibración de la cámara, la diferencia en cada coordenada (X y Y) es un número decimal que posteriormente se le suma a la posición de coordenada del ojo (`px py`) con el fin de realizar una equivalencia de la posición del ojo dentro de los parámetros calibrados.

4.2.14. Escalado de coordenadas

El escalado de coordenadas al tablero de posicionamiento se realiza por medio de la ecuación 5 expuesta numeral **5.5. ESCALADO DE COORDENADAS** y se aplica de la siguiente manera:

```
tx=((px-ixmin)/(ixmax-ixmin));
ty=((py-iymin)/(iymax-iymin));
```

Donde los valores `ixmin`, `ixmax`, `iymin` e `iymax` se establecen al realizar la calibración de la cámara, siendo los valores máximos y mínimos de las posiciones de los ojos con respecto al tablero de posicionamiento. Al aplicar esta ecuación se entrega un número decimal que sirve de referencia para encontrar el cuadrante del tablero al que se miraba.

4.2.15. Identificación del cuadrante

Al tener el resultado del escalado de coordenadas el valor de cada coordenada se evalúa dentro de un grupo de declaraciones de la siguiente manera:

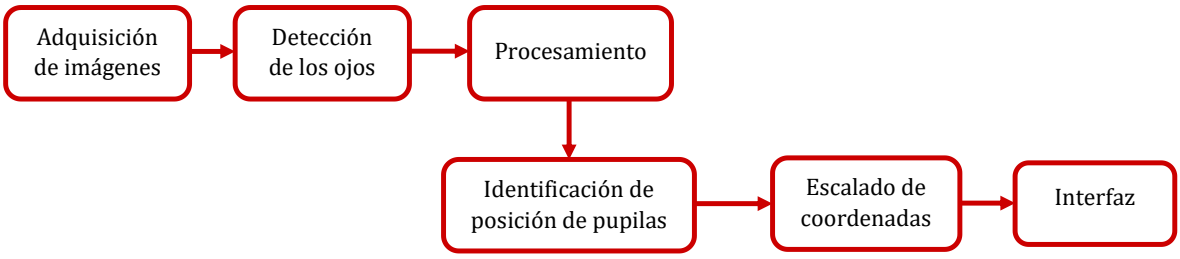
```
switch tx & ty
   case (0<=tx)&&(tx<=0.33) && (0<=ty)&&(ty<=0.5)
      mensaje='El ojo esta mirando la posicion 3';
   case (0.34<=tx)&&(tx<=0.66) && (0<=ty)&&(ty<=0.5)
      mensaje='El ojo esta mirando el cuadrante numero 2';
   case (0.67<=tx)&&(tx<=1) && (0<=ty)&&(ty<=0.5)
      mensaje='El ojo esta mirando la posicion 1';
   case (0<=tx)&&(tx<=0.33) && (0.51<=ty)&&(ty<=1)
      mensaje='El ojo esta mirando la posicion 6';
   case (0.34<=tx)&&(tx<=0.66) && (0.51<=ty)&&(ty<=1)
      mensaje='El ojo esta mirando el cuadrante numero 5';
   case (0.67<=tx)&&(tx<=1) && (0.51<=ty)&&(ty<=1)
      mensaje='El ojo esta mirando la posicion 4';
   otherwise
      mensaje='La camara se encuentra mal calibrada';
end
```

Dependiendo del valor de `tx` y de `ty`, si el número se encuentra entre 0 y 1, el bucle entrega un mensaje anunciando el cuadrante del ojo al que ha mirado la persona; si el número entregado no se encuentra entre 0 y 1 el bucle entrega la respuesta `'La cámara se encuentra mal calibrada'`. Los rangos para cada cuadrante se establecen de acuerdo con la Figura 20 y lo expuesto en el numeral **5.5. ESCALADO DE COORDENADAS.**

5. DESARROLLO DE LA APLICACIÓN

El desarrollo de la aplicación consta principalmente de las etapas plasmadas en la Figura 12, las cuales serán explicadas a lo largo de este capítulo.

Figura 12. Esquema general del sistema de seguimiento de ojo (pupila), usando técnicas de visión artificial *(Autor).*



5.1. ADQUISICIÓN DE IMÁGENES

La adquisición de las imágenes se realizó por medio de la Pi NoIr Camera V2, diseñada y fabricada por Raspberry Pi, debido a que cumplía a cabalidad con los requerimientos expuestos en el numeral **3.1. Requerimientos**. La cámara se conectó a una Raspberry Pi a través de la cual se almacenaban las imágenes para ser posteriormente procesadas.

Para la adquisición de las imágenes era necesario que la persona se ubicara frente a la cámara de tal manera que los ojos quedaran centrados y en posición directa hacia la cámara, observando fijamente el tablero de posicionamiento. Con el fin de evitar desplazamientos involuntarios en la ubicación del rostro que podían afectar las mediciones, se ubicó un soporte a la altura de la cámara sobre el cual se debía posicionar la cara, con esto también se aseguraba que los ojos se ubicaran generalmente en un mismo rango de posición lo que hacía más robusto el sistema. La iluminación en la escena debía ser mínima, era necesario solo un poco de luz para que la persona pudiera ver con claridad el tablero de posicionamiento.

Una vez se adquirían las imágenes mediante la Pi NoIr Camera V2 entre el programa Matlab y la Raspberry Pi se realizaba una conexión por medio de una red Wi-Fi, debido a que era el único módulo de conexión con la Raspberry Pi incluido en los paquetes de soporte de hardware de Matlab. En el programa Matlab 9.0 (R2016a) (The Mathworks, Natick, Massachusetts, USA) se ejecutaba el análisis y procesamiento de las imágenes y se utilizó su herramienta de interfaz gráfica (Guide) para exponer los resultados.

Para la adquisición de las imágenes se escogió una resolución de 1024x768 (RGB) debido a que era la máxima resolución aceptada por el programa Matlab para

procesar las imágenes sin que se afectara la velocidad de transmisión y procesamiento, además que una buena resolución permitía realizar un mejor tratamiento de las imágenes y por ende obtener mejores resultados. Por otra parte, cabe resaltar que toda imagen que se adquiere por medios ópticos, electro-ópticos o electrónicos sufre en una cierta medida los efectos de la degradación que se manifiestan en forma de ruido, pérdida de definición y fidelidad de la imagen.

5.2. DETECCIÓN DE LOS OJOS

El primer tratamiento que se le practicaba a la imagen adquirida era un pre-procesado, donde mediante el algoritmo Haar Cascade Classifier se realzaban partes de interés en la imagen, es decir, con este algoritmo se realizaba la identificación y detección del área de los ojos. Gracias a esto se eliminaban partes indeseables de la imagen que no eran de interés y que generaban lentitud en el sistema al incluirlas en el procesamiento y de esta manera se facilitaban las etapas posteriores.

Para ejecutar el algoritmo de Haar Cascade fue necesario utilizar la Computer Vision System Toolbox de Matlab que incluía el vision.CascadeObjectDetector, un Object System que viene con varios clasificadores pre-entrenados para detectar objetos cuya relación de aspecto no varía significativamente, como es el caso de los ojos.

El detector de objetos en cascada utilizaba el algoritmo Viola-Jones para detectar el área de los ojos en la imagen, sin embargo, fue necesario establecer ciertas propiedades condicionales que definían características especiales de los ojos (tamaño, distancia, detector, etc), como se expone en el numeral **6.1 ADQUISICIÓN DE IMÁGENES Y DETECCIÓN DE OJOS**, debido a que el clasificador en algunos casos cuando realizaba el barrido por toda la imagen detectaba falsos positivos que generaban errores en la medición, asimismo, al aplicar estas propiedades condicionales el clasificador se hacía más robusto.

Una vez se identificaba el área de los ojos, el detector entregaba las coordenadas de los pixeles que formaban un rectángulo que bordeaba la zona donde se encontraban los ojos, como se muestra en la Figura 13. Después, esta zona era extraída de la imagen para posteriormente continuar con el procesamiento y hacerlo más eficiente, debido a que el análisis del área extraída era mucho menor que si se analizaba la imagen completa.

5.3. PROCESAMIENTO

Justo después de que se extraía la imagen a la medida del área de los ojos, la matriz de coordenadas RGB se transformaba a escala de grises y se le realizaba una binarizacion junto con una umbralización, con el objetivo de resaltar áreas grandes

y filtrar áreas pequeñas que generaban ruido, es decir, se le ejecutaba un proceso de segmentación a la imagen, como se muestra en la Figura 14.

Figura 13. Utilización del el Haar Cascade Classifier para la detección de los ojos en la imagen, bordeados por un rectángulo *(Autor).*

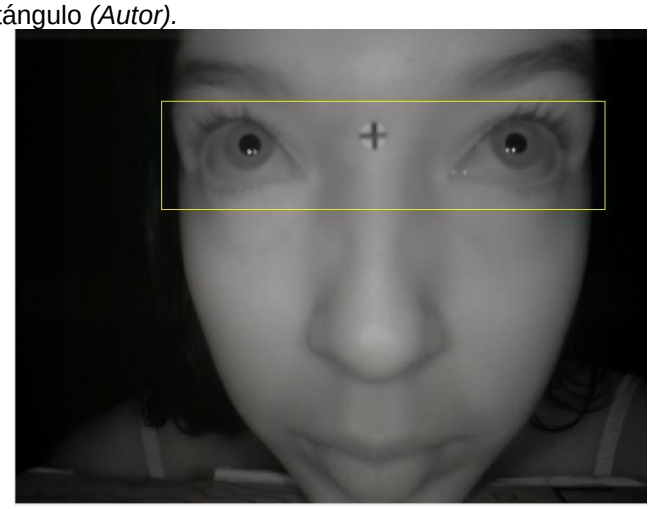
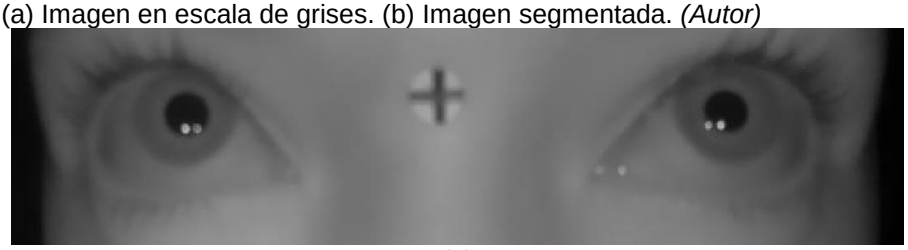


Figura 14. (a) Imagen en escala de grises. (b) Imagen segmentada. *(Autor)*



(a)

(b)

Luego de esto se aplicó una dilatación y se creó un filtro de disco para cerrar los bordes en la imagen, debido a que la forma de las pupilas se asemeja a un disco o circulo, de esta manera se diferenciaron correctamente los objetos en la imagen del fondo, como se muestra en la Figura 15.

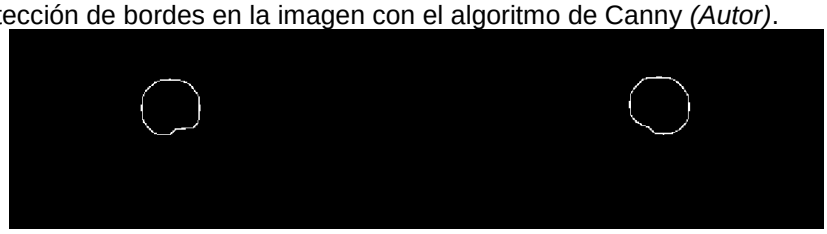
Figura 15. Dilatación de objetos en la imagen por medio de un filtro de disco *(Autor).*

Sin embargo, como se muestra en la Figura 15 junto con las pupilas también se resaltaban las pestañas y en algunos casos partes de las cejas, por este motivo se efectuó una discriminación por perímetro y por excentricidad para eliminar los objetos que no cumplieran con las características de las pupilas, como se muestra en la Figura 16. Para determinar los valores promedio que debían tener las pupilas en cuanto a perímetro y excentricidad, se hizo un análisis empírico con datos extraídos de varias pruebas que se realizaron con videos pregrabados, los cuales se evidencian en el numeral **6.2 PROCESAMIENTO**.

Figura 16. Discriminación de objetos no deseados en la imagen por perímetro y excentricidad *(Autor).*

Asimismo, cuando los únicos objetos existentes en la imagen eran las pupilas, sus bordes se detectaban por medio de la aplicación del filtro de Canny, dado que es uno de los detectores más precisos puesto que utiliza máscaras de convolución, es adaptable a varios ambientes, no disminuye su precisión ante la presencia de ruido, entre otros, como se muestra en la Figura 17.

Figura 17. Detección de bordes en la imagen con el algoritmo de Canny *(Autor)*.



5.4. IDENTIFICACIÓN DE POSICIÓN DE PUPILAS

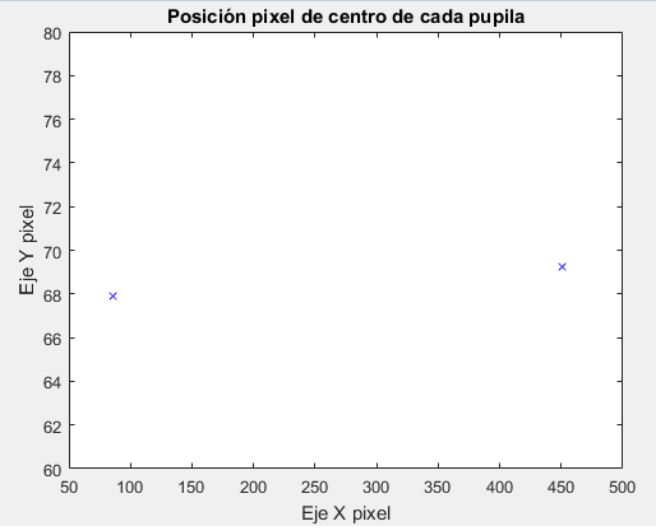
Después de obtener los bordes de los objetos en la imagen, se utilizó la transformada de Hough para identificar las pupilas como circunferencias, dado que un circulo es la forma geométrica que más se asemeja a una pupila, asimismo es mucho más sencillo tratar, analizar y extraer características de un objeto que tenga forma de figura geométrica. La transformada de Hough, entregaba como resultado las coordenadas de la ubicación del centro de la circunferencia y el tamaño de su radio. Con estos datos se dibujaban las circunferencias sobre la imagen, es decir, se encerraban las pupilas dentro de dos circunferencias con el fin de visualizar los resultados de una manera más dinámica, tal como se muestra en la Figura 18.

Figura 18. Aplicación transformada de Hough para encontrar círculos (pupilas) en la imagen *(Autor).*

Luego de esto, los centroides de las circunferencias se utilizaban para ubicar marcadores sobre la imagen en forma de cruz en el centro de cada pupila que servían para evidenciar su seguimiento y trayectoria durante la ejecución del programa. Además de esto, las coordenadas de los centroides se almacenaban en dos variables acumulativas (una por cada ojo) y se graficaban en un plano de dos dimensiones, como se muestra en la Figura |9. Cabe resaltar que estas dos variables acumulaban las posiciones (pixeles) de los centros de las pupilas de todos los frames (frames por minuto de la cámara) que se adquirían para ser procesados.

Figura 19. (a) Ubicación del centro de la pupila en la imagen procesada. (b) plano de dos dimensiones donde se muestra la ubicación de los centros de las pupilas. *(Autor)*

(a)



(b)

5.5. ESCALADO DE COORDENADAS

Además de identificar el centro de las pupilas, también se identificaba el centro del marcador ubicado en el rostro, el cual se usaba como referencia para identificar si hubo movimiento en la posición del rostro con respecto a la calibración. Para

identificar cuanto se desplazó el marcador, simplemente se calculó la diferencia entre la posición del marcador durante la calibración y durante el seguimiento de la pupila, como se muestra en la Ecuación 3.

$$Pixeles\ corridos_{(x,y)} = Pixel\ calibracion - Pixel\ seguimiento \qquad \text{(Ecuación 3)}$$

Donde $Pixeles\ corridos_{(x,y)}$ hace referencia a los pixeles que se desplazó el rostro y por ende los ojos con respecto a la calibración con relación a las posiciones en (x, y), $Pixel\ calibracion$ se refiere a la posición del pixel del centro del marcador durante la calibración y $Pixel\ seguimiento$ es la posición del pixel del centro del marcador durante el procesamiento de las imágenes. Al conocer los pixeles que se desplazaba el rostro, tanto en X como en Y, se procedía con la suma de estos a la posición del pixel del centro de cada pupila para poder identificar hacia donde estaba mirando, de acuerdo con los parámetros establecidos durante la calibración, como se muestra en la Ecuación 4.

$$Pixel\ real\ pupila_{(x,y)} = Pixel\ centro\ pupila + Pixeles\ corridos \qquad \text{(Ecuación 4)}$$

Donde $Pixel\ real\ pupila_{(x,y)}$ hace referencia a la posición de la pupila en ese instante con respecto a los parámetros calibrados, $Pixel\ centro\ pupila$ es el pixel del centro de la pupila (derecha o izquierda) y $Pixeles\ corridos$ son los pixeles que se desplazó el rostro, hallados anteriormente.

El escalado de coordenadas al plano real se realizaba haciendo una equivalencia entre el tablero de posicionamiento y el plano de coordenadas (X,Y), utilizando los valores máximos y mínimos de las posiciones de los ojos determinados durante la calibración, donde en el escalado al plano real la posición mínima era 0 y la posición máxima era 1. Esta equivalencia se realizaba por medio de la Ecuación 5.

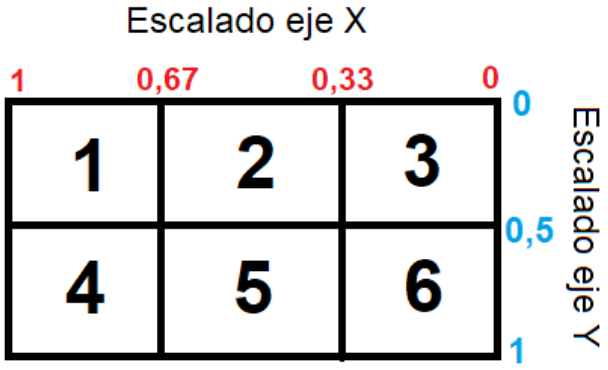
$$Posicion\ tablero\ (x, y) = \frac{Pixel\ real\ pupila - Pmin}{Pmax - Pmin} \qquad \text{(Ecuación 5)}$$

Donde $Posicion\ tablero$ equivalía a un número entre 0 y 1 con el cual se identificaba la posición de la pupila en el tablero de posicionamiento, haciendo el caculo en las dos coordenadas (x,y), $Pixel\ real\ pupila$ hace referencia a la posición de la pupila con respecto a la calibración y $Pmax\ y\ Pmin$ se refiere a los valores máximos y mínimos de las posiciones de los ojos determinados durante la calibración.

Como se muestra en la Figura 20 para poder identificar de una manera práctica hacia donde estaba mirando la persona, el tablero de posicionamiento se dividió en seis cuadrantes, además como el resultado del escalado de coordenadas era entre valores de 0 y 1, el tablero también se dividió de esa manera, es decir, que se podía identificar hacia donde estaba mirando la persona ubicando los valores del resultado de la ecuación anterior en las dimensiones del tablero de posicionamiento. Los

rangos para cada cuadrante se establecieron haciendo una equivalencia entre la distribución de los pixeles en la imagen, los cuales aumentaban de izquierda a derecha y de arriba abajo, y la identificación de los pixeles cuando se miraba cada cuadrante.

Figura 20. Tablero de posicionamiento dividido en seis cuadrantes y equivalencia del escalado de coordenadas a valores entre 0 y 1 *(Autor).*



5.6. INTERFAZ DE USUARIO

En el contexto del proceso de interacción persona-computadora, la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático[16]. Siguiendo con lo anterior, para evidenciar los resultados de manera práctica, por medio del entorno de programación visual de Matlab, Guide, se realizó la interfaz gráfica de usuario, debido a que permitía un control sencillo (con uso de ratón) de las aplicaciones de software, eliminando la necesidad de aprender un lenguaje y escribir comandos a fin de ejecutar una aplicación[17].

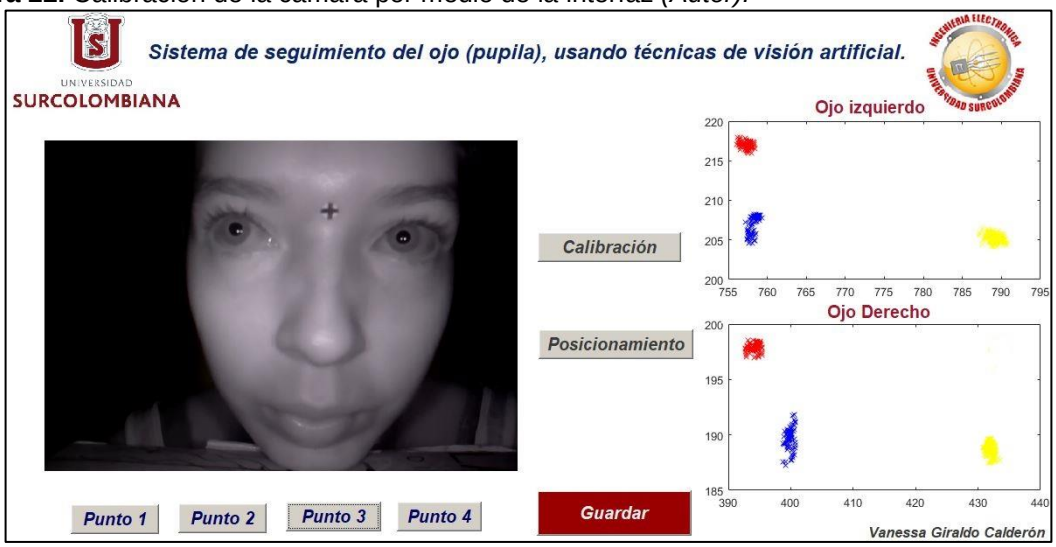
Principalmente la interfaz constaba de 3 páginas: portada, calibración y posicionamiento. A continuación, se explica su funcionamiento y como se exponen los resultados.

[16] Gloria Areitio y Ana Areitio. Información, Informática e Internet: del ordenador personal a la Empresa 2.0. 2009. Editorial Visión Libros. España. Páginas: 195-197.
[17] Mathworks. Creación de apps con interfaces gráficas de usuario en MATLAB. GUI de Matlab. Revisado: mayo 2018. Disponible en internet: https://la.mathworks.com/discovery/matlab-gui.html

Figura 21. Portada de interfaz del sistema de seguimiento del ojo (pupila), usando técnicas de visión artificial *(Autor).*

Como se muestra en la Figura 21 la portada de la interfaz constaba de imágenes alusivas al sistema y de dos botones: *Calibración* y *Posicionamiento*. Lo que se debía hacer para iniciar el sistema era presionar el primer botón con el objetivo de realizar la calibración de la cámara.

Figura 22. Calibración de la cámara por medio de la interfaz *(Autor).*



Como se muestra en la Figura 22 al presionar el botón de calibración, la interfaz modificaba sus recuadros y botones, de manera que:

- En el recuadro a la izquierda de la pantalla aparecía el video en vivo de la persona, donde se resaltaban con una cruz blanca los centros de las pupilas y el centro del marcador del rostro en forma de cruz.

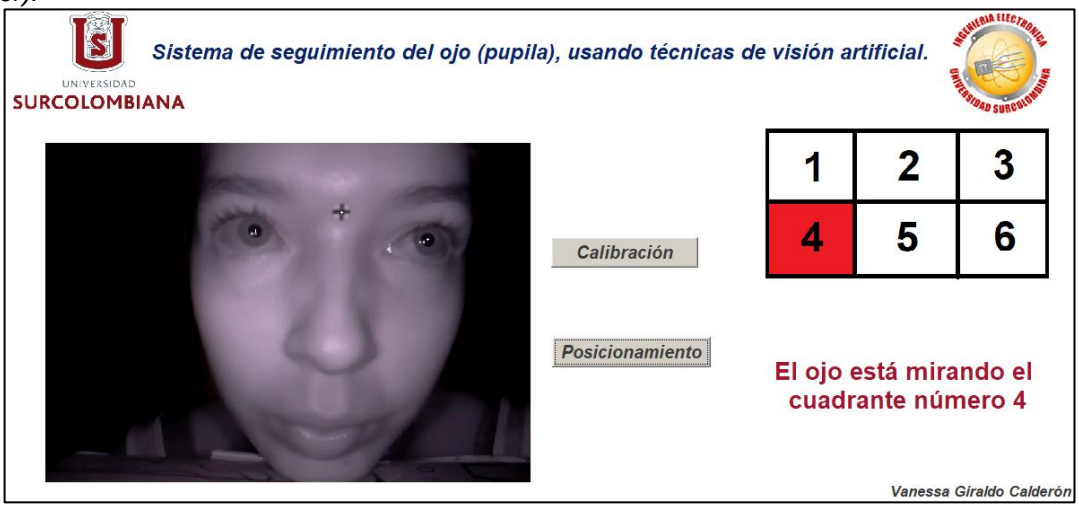
- Debajo de este recuadro aparecían cuatro botones que se usaban para realizar la calibración de los puntos máximos del tablero de posicionamiento, acumulando los centros de las pupilas. En otras palabras, lo que se debía hacer era presionar el botón *Punto 1* y mirar la primera equis (x) del tablero de posicionamiento durante cinco segundos y así repetir el proceso con cada botón.

- A medida que se iban calibrando los puntos, en la parte derecha de la pantalla se mostraba el recorrido de los pixeles de posicionamiento de las pupilas. En el plano superior se mostraba el ojo izquierdo y en el plano inferior el ojo derecho.

- Cuando se tenían los cuatro puntos máximos calibrados, se debía observar que en los planos de los ojos estos puntos formaran las esquinas de un rectángulo, de ser así se debía presionar el botón *Guardar* ubicado en la parte inferior de la pantalla para almacenar los resultados, si los puntos no formaban un rectángulo se debían volver a calibrar todos los puntos nuevamente.

Al tener todos los puntos máximos debidamente calibrados y almacenados se procedía a realizar el posicionamiento de los ojos. En esta etapa se debía presionar el botón *Posicionamiento* y mirar un cuadrante del tablero durante cinco segundos.

Figura 23. Identificación del cuadrante que miraba la persona en el tablero de posicionamiento *(Autor).*



Luego de lo anterior, como se muestra en la Figura 23, el programa mostraba el cuadrante identificado poniendo en rojo el fondo del cuadrante del tablero, ubicado en la parte superior derecha de la pantalla y con un mensaje que se mostraba en la parte inferior derecha de la pantalla. Cabe resaltar se podía repetir el procedimiento de posicionamiento de los ojos las veces que se quisiera, siempre y cuando no se hubieran producido movimientos en la posición del rostro, de ser así era necesario recalibrar la cámara para obtener resultados correctos.

6. ANÁLISIS DE RESULTADOS

6.1. ADQUISICIÓN DE IMÁGENES Y DETECCIÓN DE OJOS

Para el sistema de seguimiento del ojo (pupila) este primer paso era imprescindible debido a que sí las imágenes adquiridas no presentaban las condiciones recomendadas no se podía garantizar que los resultados entregados fueran los correctos. Con el fin de determinar los lineamientos que debía cumplir el sistema para garantizar su funcionamiento, se realizaron pruebas con la cámara a diferentes distancias y grados de inclinación en relación con los ojos identificando en cuál de estas se presentaban las mejores condiciones de detección del área de los ojos. Además de esto también se realizaron pruebas para encontrar el parámetro del tamaño mínimo que debían tener los ojos para que el Haar Cascade Classifier funcionara de una manera óptima. Las pruebas fueron realizadas a tres personas, analizando un total de 2691 frames en cada caso (distancia, inclinación y tamaño) de la siguiente manera:

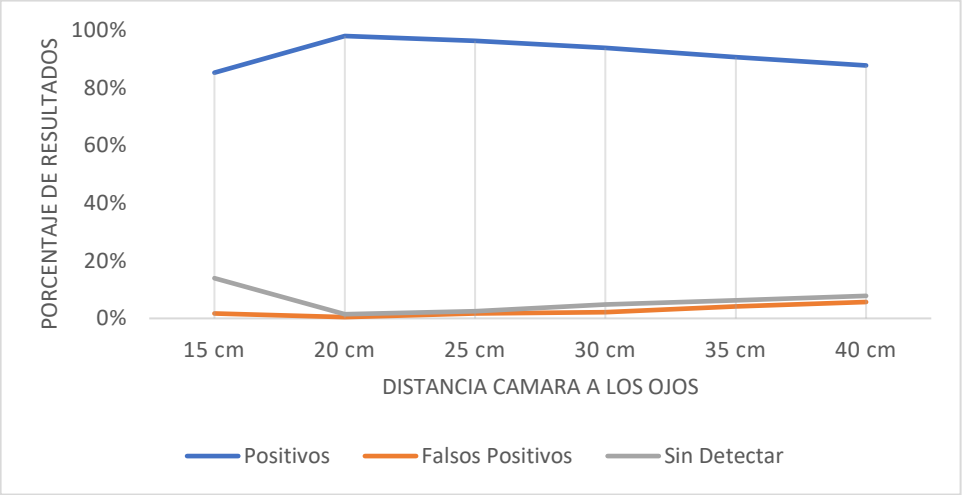
- Sí se lograba detectar correctamente el área de los ojos se hablaba de muestras positivas.

- Sí por el contrario detectaba otras áreas como si fuesen los ojos, se hablaba de falsos positivos.

- Sí no lograba identificar el área donde se encontraban los ojos, ni ninguna otra área, se hablaba de muestras sin detectar.

Los resultados de las pruebas se observan en las gráficas de las Figuras 24, 25 y 26.

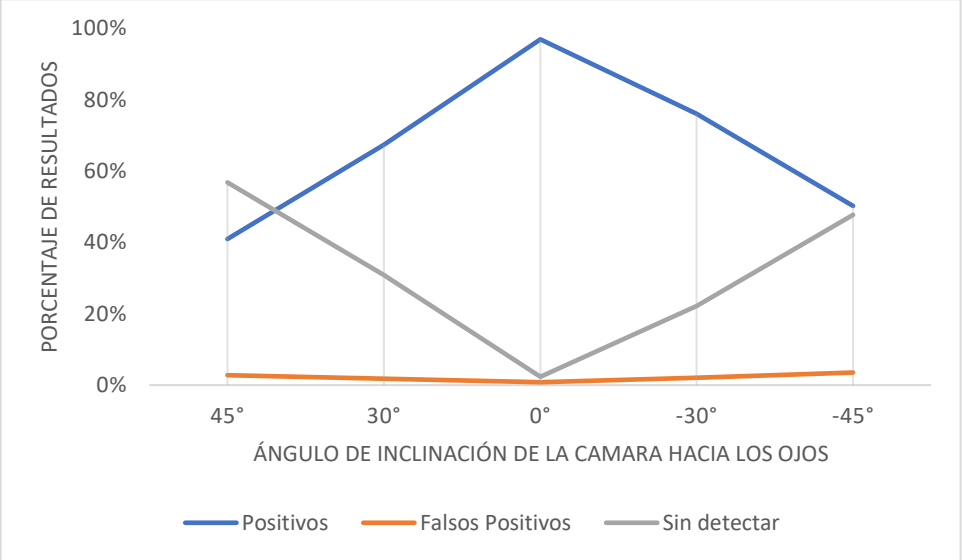
Figura 24. Pruebas de adquisición de imágenes con la cámara a diferentes distancias con respecto a la posición de los ojos *(Autor).*



De la Figura 24 se puede inferir que todas las distancias analizadas presentan resultados favorables debido a que los positivos se encuentran por encima del 80% de los frames analizados, sin embargo, los mejores resultados se presentan cuando la cámara se encuentra en un rango de 20cm a 25cm de la posición de los ojos, donde se presentan la mayor cantidad de positivos y la menor cantidad de falsos positivos y sin detectar. Por estos motivos se establece que la cámara siempre tendrá que estar a una distancia de 20cm a 25cm de la posición de los ojos con el fin de garantizar que los resultados obtenidos sean óptimos para continuar con las siguientes etapas.

Luego de esto, se procedió a realizar las pruebas de inclinación de la cámara a diferentes ángulos manteniendo una distancia fija de 20cm entre los ojos y la cámara, los resultados se muestran en la Figura 25.

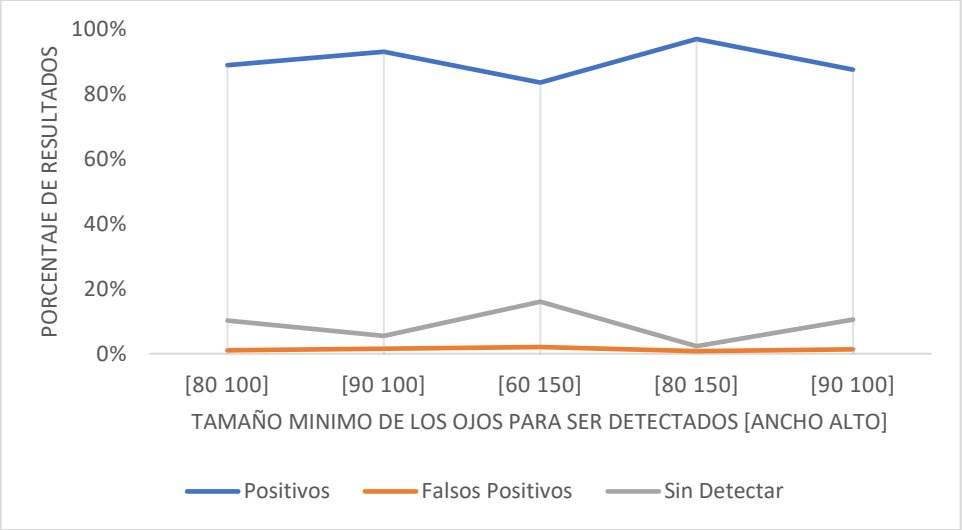
Figura 25. Pruebas de adquisición de imágenes con la cámara en diferentes ángulos de posición con respecto a los ojos *(Autor).*



En la Figura 25 se puede observar que los mejores resultados se daban cuando la cámara se ubicaba en un ángulo de 0°, es decir, cuando se ubicaba directamente frente a los ojos. En esta posición el porcentaje de error era del 4%, lo que significaba que las imágenes adquiridas eran confiables para continuar con las otras etapas del sistema, dicho en otras palabras, la etapa de detección funcionaba correctamente debido a que tenía un 96% de eficacia, puesto que detectaba los ojos de toda persona, aunque a veces se presentaran falsos positivos en la escena o no se lograra identificar el área de los ojos. Además de esto se puede observar en la Figura 25 que a medida que el ángulo de posición se aleja de 0° la detección va disminuyendo, hasta el punto en que los frames en que se detectan los ojos son proporcionales a los que no se detectan.

Después de haber establecido la distancia y el ángulo de inclinación de la cámara con respecto a la posición de los ojos, también fue necesario realizar una serie de pruebas con el fin de configurar el Haar Cascade Classifier en cuanto a los parámetros del tamaño mínimo que los ojos debían tener para ser reconocidos. Para realizar la detección de los ojos existen dos modelos de clasificación *EyePairSmall* y *EyePairBig,* sin embargo, al realizar pruebas con el primero no se obtuvieron resultados favorables en consecuencia de la cercanía de los ojos a la cámara, lo que hacia los ojos en la imagen demasiado grandes para ser detectados con los parámetros establecidos en este modelo. En el caso del modelo *EyePairBig* se analizaron diferentes tamaños detectables con pruebas de ensayo y error verificando el porcentaje de positivos encontrados en los frames analizados. En la Figura 26 se muestran los resultados de las pruebas donde los positivos se encontraban por encima del 80% y las muestras sin detectar por debajo del 20%.

Figura 26. Pruebas de detección de ojos aplicando el modelo de clasificación EyePairBig del Haar Cascade Classifier *(Autor).*



Como se muestra en la Figura 26 para el Haar Cascade Classifier el tamaño mínimo de los ojos se especificaba como un vector de dos elementos [ancho alto] expresado en píxeles. Además de esto se puede observar que únicamente en cinco tamaños las muestras positivas se encontraban por encima del 80% y las muestras sin detectar por debajo del 20%. No obstante, aunque todos los tamaños mostrados en la Figura 26 eran aceptables, se escogió el tamaño de [80 150] como el tamaño mínimo de los ojos debido a que con un 97% presentaba el mayor porcentaje de muestras positivas detectadas.

6.2. PROCESAMIENTO

Inicialmente para realizar la identificación de las pupilas se utilizó un método de detección de esquinas, en el cual se detectaban varios puntos dentro del área de los ojos y posteriormente se realizaba un filtrado donde quedaban únicamente los puntos de los centros las pupilas, sin embargo, el sistema era muy sensible ante los cambios y no era posible realizar una buena identificación y seguimiento. Este método se remplazó con uno donde se detectaban las regiones presentes en la imagen y por medio de una discriminación de perímetro y excentricidad se podían identificar las regiones que correspondían a las pupilas. Este método demostró ser mucho más robusto y confiable que el anterior.

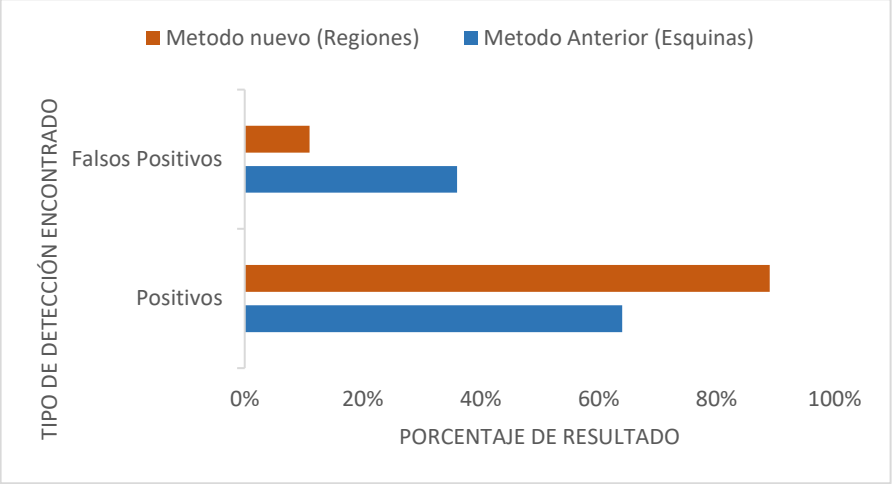
Para realizar esta demostración, se analizaron muestras de video de tres personas (2628 frames) donde se aplicaba cada método de detección y se identificaba la cantidad de positivos y falsos positivos en cada frame de la siguiente manera:

- Sí en el frame se detectaban correctamente las pupilas se hablaba de resultados positivos.

- Sí por el contrario en el frame se detectaban otras áreas como si fuesen las pupilas, se hablaba de falsos positivos.

Para este caso no se evidenciaron resultados de frames en los que no se presentaba detección debido a que en todos los frames siempre se identificó algún objeto independientemente del método utilizado. Los resultados se muestran en la Figura 27.

Figura 27. Comparativo de métodos de detección de las pupilas en la imagen *(Autor)*.



En la Figura 27 se puede observar que el método de detección de regiones evidenciaba mejores resultados en comparación con el método de detección de

esquinas, dado que este último presentaba un alto porcentaje de falsos positivos en las pruebas realizadas, por lo tanto, no era un método confiable y no garantizaba la efectividad del sistema. Cabe resaltar que en el método de detección de regiones también se presentan falsos positivos en los frames analizados, sin embargo, utilizando este método era mucho más fácil realizar las correcciones correspondientes para eliminar la mayoría de los falsos positivos.
Otro análisis que se realizó en esta etapa fue el análisis por medio del cual se determinaron los valores promedio del tamaño de los ojos en cuanto a las características de perímetro y excentricidad. Para esto, en un vector se extrajeron los valores de todos los perímetros y excentricidades de los objetos encontrados en cada frame de un video pregrabado, estos datos fueron analizados empíricamente, es decir, los resultados obtenidos se basaron en la observación de la tendencia de los datos experimentalmente. En efecto, inicialmente los datos se ordenaron de menor a mayor y se realizó una observación detallada los rangos de perímetros que más se repetían, luego estos datos se extrajeron y se observaron las excentricidades de cada uno, identificando que los perímetros que presentaban las menores excentricidades y leves variaciones correspondían a los ojos. Para verificar el procedimiento anterior y poder establecer el rango de valores del perímetro y la excentricidad de los ojos, se analizaron las muestras de video por frames de tres personas más, los resultados de cada uno se muestran en la Tabla 5.

Tabla 5. Resultados del análisis de los valores máximos y mínimos del perímetro y excentricidad de los ojos en cuatro personas *(Autor).*

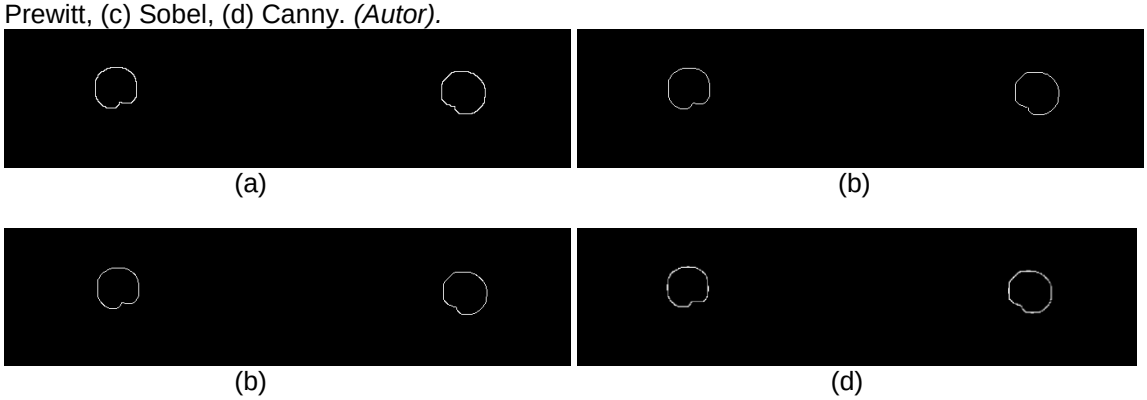
Persona	Perímetro		Excentricidad	
	Máximo	**Mínimo**	**Máximo**	**Mínimo**
1	343,8043	133,0593	0,6189	0,5489
2	295,5329	89,3854	0,5983	0,5733
3	377,9642	192,439	0,6073	0,5638
4	358,9964	93,7540	0,5624	0,5057

De acuerdo con los resultados plasmados en la Tabla 5 se puede observar que los valores de los perímetros se encuentran en un rango de valores entre 377,9642 y 89,3854 pixeles, por este motivo para el sistema se estableció que el valor máximo del perímetro era de 400 y el mínimo de 100 con el fin de asegurar la exactitud de la identificación, es decir, se aseguraba la medición si en algún caso el perímetro de los ojos se encontraba por encima o por debajo de los resultados de las pruebas.

Para el caso de la excentricidad los valores se encuentran entre 0,6189 y 0,5057, de acuerdo con esto únicamente se estableció que el valor máximo de excentricidad de los ojos es de 0,63 y no se estableció valor mínimo debido a que los ojos siempre presentaban la menor excentricidad de los objetos en la escena en consecuencia de que se interpretaban como circunferencias, los demás objetos presentaban excentricidades cercanas a 1 debido a que no se asemejaban a ninguna figura geométrica.

Además de todo lo anterior, también se realizó un análisis sobre el método de detección de bordes que debía utilizarse, para esto se tuvieron en cuenta los métodos de Roberts, Prewitt, Sobel y Canny, debido a que son los métodos de detección de bordes que trabajan con operadores de primera derivada y son los más utilizados en el procesamiento de imágenes. Para identificar el método a utilizar, se tomaron muestras de tres personas y se analizó la detección de bordes en cada frame aplicando cada uno de los métodos mencionados, el resultado en uno de los frames se evidencia en la Figura 28.

Figura 28. Aplicación de los cuatro métodos de detección de bordes a una imagen. (a) Roberts, (b) Prewitt, (c) Sobel, (d) Canny. *(Autor).*



(a) (b)

(b) (d)

De acuerdo con la Figura 28 se puede inferir que los cuatro métodos analizados evidencian resultados similares, por consiguiente, no existen diferencias significativas entre ellos que puedan asegurar cual método de detección de bordes es mejor para el sistema. Ante esto se decidió seleccionar el detector de Canny debido a sus características de precisión y estabilidad, aunque haya presencia de ruido en la escena, su adaptabilidad a varios ambientes y su propiedad de adelgazamiento del ancho de los bordes a un pixel, entre otras.

6.3. IDENTIFICACIÓN DE POSICIÓN DE PUPILAS

Antes de nada, en esta etapa se debía realizar la configuración de los parámetros para el funcionamiento de la transformada de Hough, para ello se realizaron pruebas con el fin de encontrar el rango del radio que debían tener las circunferencias (pupilas) para ser detectadas de manera óptima. Para realizar las pruebas se tomaron los resultados del análisis de los valores máximos y mínimos del perímetro de los ojos, mostrados en la Tabla 5. A cada uno de los perímetros de los ojos encontrados en cada frame se le aplico la Ecuación 6 para encontrar su radio, los resultados obtenidos de las pruebas se muestran en la Tabla 6.

$$Radio = \frac{Perimetro}{2\pi} \qquad \text{(Ecuación 6)}$$

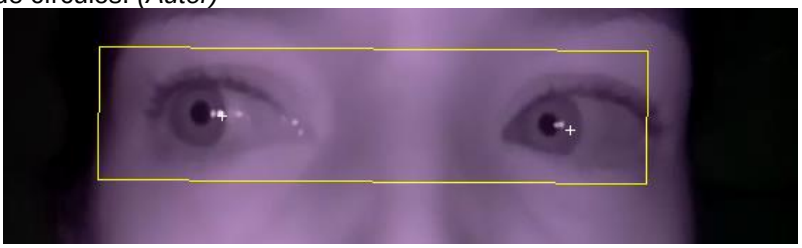
Tabla 6. Resultados de aplicar la ecuación 6 a los valores máximos y mínimos de los perímetros de las pupilas *(Autor).*

Persona	Radio	
	Máximo	Mínimo
1	86,5489	37,0925
2	68,3621	25,1073
3	91,9856	30,6275
4	88,9668	20,9567

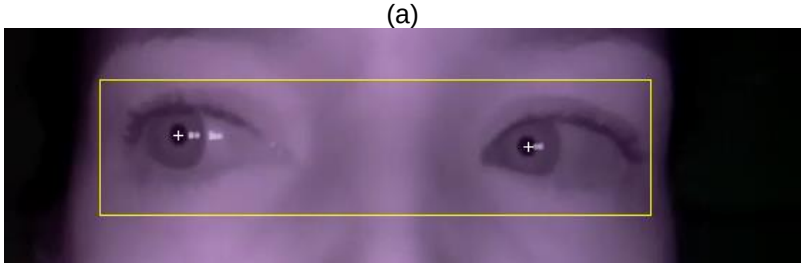
De la Tabla 6 se puede observar que los valores de los radios se encuentran en un rango de valores entre 20,9567 y 91,9856, por este motivo para el sistema se estableció que el valor máximo del radio de la circunferencia para la transformada de Hough era de 95 y el mínimo de 15 con el fin de asegurar la exactitud de la identificación, es decir, se aseguraba la medición si en algún caso el radio de los ojos se encontraba por encima o por debajo de los resultados de las pruebas.

En un principio para identificar la posición de las pupilas en cada frame se trabajó con la transformada geométrica, este método consistía en identificar los puntos de las pupilas en el primer frame y realizar un rastreo de esos puntos a lo largo del proceso, sin embargo, no era muy preciso en consecuencia de que en algunos casos los puntos de la pupila se movían varios pixeles y no se reajustaban a donde debían estar, por lo tanto, se generaba un error que afectaba totalmente las mediciones y los resultados, además de esto, se presentaban quiebres en el rectángulo de identificación del área de los ojos debido a que su relación de aspecto varia significativamente. Este método se remplazó con la transformada de Hough, donde se identificaban las pupilas como circunferencias en cada frame, lo cual mejoro considerablemente la medición y por ende los resultados.

Figura 29. (a) Transformada geométrica para seguimiento de puntos. (b) Transformada de Hough para detección de círculos. *(Autor)*



(a)



(b)

Cabe resaltar que al hacer la detección de las pupilas en cada frame la velocidad del sistema se veía comprometida debido a que el procesamiento era mayor, no obstante, el porcentaje de precisión del seguimiento de las pupilas aumento considerablemente y para evidenciarlo se realizaron una serie de pruebas con videos de tres personas donde miraban los seis cuadrantes del tablero simultáneamente, luego de esto se analizó la detección del centro de las pupilas en cada frame de la siguiente manera:

- Sí en el frame se detectaban correctamente los centros de las pupilas se hablaba de resultados positivos.

- Sí por el contrario en el frame se detectaban otras áreas como si fuesen los centros de las pupilas, se hablaba de falsos positivos.

- Sí en el frame no se lograba identificar el centro las dos o de alguna de las pupilas, se hablaba de resultados sin detectar.

Los resultados se muestran en la Tabla 7.

Tabla 7. Comparación de métodos de identificación y seguimiento de pupilas *(Autor).*

Método	Positivos	Falsos Positivos	Sin detectar
Transformada Geométrica	786	1527	485
Transformada Hough	2399	15	217
Total Frames Analizados: 2628			

De la Tabla 7 se puede deducir que el método de la transformada de Hough presenta significativamente mejores resultados que el método de la transformada geométrica, en consecuencia, de que los resultados positivos corresponden al 91% de los frames analizados, en cambio para el caso de la transformada geométrica los resultados positivos son mucho menores que los falsos positivos que se encuentran en el 58% de los frames, esto quiere decir que es un método nada confiable. Lo anterior evidencia que, al utilizar el método de la transformada de Hough, aunque la velocidad del sistema se veía comprometida los resultados de identificación del centro de la pupila eran satisfactorios y confiables para continuar con las siguientes etapas del sistema.

6.4. ESCALADO DE COORDENADAS

En esta etapa final el sistema debía ser capaz de identificar hacía que cuadrante del tablero de posicionamiento estaba mirando la persona, para ello era necesario realizar un escalado de coordenadas, es decir, utilizando los resultados de las coordenadas (pixeles) de los centros de las pupilas, aplicándolos a una formula y

relacionando los resultados en la Tabla 8, era posible identificar hacía que cuadrante se estaba mirando.

Por ejemplo, si la persona estaba mirando el cuadrante 2, el resultado del escalado de coordenadas en X debía ser un numero entre 0,33 y 0,67 y en Y debía ser un numero entre 0 y 0,5. Para establecer la confiabilidad del sistema se realizaron pruebas con tres personas, donde tenían que mirar durante un tiempo determinado cada uno de los seis cuadrantes del tablero; el procedimiento se repitió cuatro veces por persona, realizando 12 pruebas y analizando un total de 72 cuadrantes, los resultados se muestran en la Tabla 9.

Tabla 8. Escalado de coordenadas en pixeles a coordenadas en el tablero de posicionamiento *(Autor).*

No. Cuadrante	Xmin	Xmax	Ymin	Ymax
1	0,67	1	0	0,5
2	0,33	0,67	0	0,5
3	0	0,33	0	0,5
4	0,67	1	0,5	1
5	0,33	0,67	0,5	1
6	0	0,33	0,5	1

Tabla 9. Pruebas de posicionamiento de los ojos por cuadrantes correctos o incorrectos *(Autor).*

Prueba	Cuadrantes	
	Correctos	**Incorrectos**
1	5	1
2	6	0
3	4	2
4	6	0
5	6	0
6	6	0
7	4	2
8	5	1
9	6	0
10	6	0
11	5	1
12	6	0

De la Tabla 9 se puede deducir que la cantidad de identificaciones correctas de la posición de los ojos son mucho mayores que donde no se identificaba correctamente, en efecto, de los 72 cuadrantes analizados el 90,3% se identificó de manera acertada contra un 9,7% que presento fallas en la correlación de los datos.

Sin embargo, cabe resaltar que al analizar detalladamente los resultados se encontró que en las pruebas donde los resultados no fueron favorables era como consecuencia de que la persona presentaba pequeños movimientos involuntarios en la posición del rostro los cuales eran difíciles de controlar y por ende se veían afectadas las mediciones y los resultados. De manera que con una probabilidad de aceptación mayor al 90% se puede concluir que la posición de la pupila con respecto al tablero de posicionamiento es confiable y efectiva para el sistema.

6.5. VALIDACIÓN DEL ALGORITMO

La validación y puesta a punto del algoritmo se realizaba por medio de dos métodos: programación y análisis.

6.5.1. Método de implementación del algoritmo

El método de programación consistía en convertir el pseudocódigo en código, y a través de un ordenador o dispositivo programable, verificar que el algoritmo funcionaba, es decir es una verificación centrada en el funcionamiento[18].

La validación del algoritmo por este método se podía evidenciar en el numeral **4.2. SOFTWARE DE LA APLICACIÓN**, donde se relacionaba el pseudocódigo de la Figura 11 con su ejecución en el programa Matlab; con lo expuesto allí se podía comprobar que el algoritmo funcionaba correctamente explicando paso a paso cada una de sus etapas y las funciones utilizadas.

6.5.2. Método de análisis

El método de análisis consistía básicamente en realizar una verificación del funcionamiento del sistema por medio la observación de los resultados proporcionados por el algoritmo. En la Tabla 10 se muestra una recopilación de los resultados analizados para el sistema de seguimiento del ojo (Pupila).

Tabla 10. Resumen de los resultados de los análisis realizados para el funcionamiento del sistema *(Autor).*

Etapa	Premisa	Resultado
Adquisición de imágenes y detección de ojos	Cámara a diferentes distancias de los ojos	La cámara debe ubicarse en un rango de 20 a 25cm de distancia con respecto a los ojos. **Positivos:** Por encima del 95% **Falsos Positivos:** Aproximadamente del 2% **Sin Detectar:** Aproximadamente del 2%

[18] W. García. Introducción a la programación. 2015 [Revisado: mayo de 2018]. Disponible en internet: https://issuu.com/syhwh/docs/introducci__n_a_la_programaci__n

	Cámara a diferentes ángulos de inclinación de los ojos	La cámara debe ubicarse preferiblemente con un ángulo de inclinación de 0°. **Positivos:** Por encima del 95% **Falsos Positivos:** Aproximadamente del 1% **Sin Detectar:** Por encima del 2%

Tabla 10. *Continuación*

Etapa	Premisa	Resultado
Adquisición de imágenes y detección de ojos	Tamaño mínimo de los ojos para ser detectados aplicando el modelo de clasificación EyePairBig del Haar Cascade Classifier	El tamaño más favorable era 80 pixeles de ancho por 150 pixeles de largo.
Procesamiento	Comparación de métodos de detección de pupilas en la imagen (Detector de esquinas vs Detector de regiones)	La detección de regiones fue más efectiva para la detección de las pupilas que el método de detección de esquinas. **Positivos:** Por encima del 92% **Falsos Positivos:** Por debajo del 8% **Sin Detectar:** En todos los frames hubo detección
	Rango de valores para discriminación de objetos por perímetro y excentricidad	**Perímetro:** Valores entre 400 a 100 pixeles correspondían a las pupilas. **Excentricidad:** Valores entre 0 y 0,63 correspondían a las pupilas.
	Selección del método de detección de bordes realizando una comparación entre Roberts, Prewitt, Sobel y Canny	Se escogió el método de Canny debido a sus características de precisión y estabilidad, aunque todos los métodos presentaron resultados similares.
Identificación de posición de pupilas	Rango de valores para el radio de la circunferencia para identificar las pupilas.	Se estableció un rango de valores para el radio de la circunferencia entre 15 y 95 pixeles.
	Comparación de métodos de identificación y seguimiento de pupilas. (Transformada Geométrica y Transformada de Hough).	La identificación de las pupilas fue mucho más efectiva por medio de la Transformada de Hough que por la Transformada Geométrica.
Escalado de coordenadas	Identificación de posicionamiento de los ojos por cuadrantes (Correctos vs Incorrectos).	Las identificaciones correctas de la posición de los ojos son mucho mayores siendo un 90,3% del total contra un 9,7% que presento fallas en la correlación de los datos

De acuerdo con lo expuesto en la Tabla 10, al observar los resultados obtenidos se puede evidenciar que el sistema funcionaba correctamente, como consecuencia de que en las pruebas que se guiaban por la identificación de positivos, falsos positivos y muestras sin detectar los resultados positivos del método o característica seleccionada superaban el 90% en cada caso. Además de esto, de acuerdo con la

Tabla 10 se comprueba que el algoritmo se puso a punto debido a que para establecer los parámetros necesarios para la ejecución de diferentes funciones del código se realizaron las pruebas correspondientes que entregaban como resultado los parámetros más favorables para el funcionamiento del sistema.

7. CONCLUSIONES

Este trabajo de investigación se centró en realizar una detección y seguimiento a los ojos (pupilas) con el fin de identificar el posicionamiento de estos en un tiempo determinado, concluyendo lo siguiente:

- Se seleccionó una plataforma óptima de desarrollo, compuesta por una cámara Raspberry Pi NoIr, un microcomputador Raspberry Pi 3 y el programa Matlab 9.0 (R2016a), que se comunicaban a través de una red Wi-Fi.

- La adquisición de las imágenes para un sistema de seguimiento de pupilas es la parte más importante del proceso debido a que el rendimiento y efectividad de las demás etapas se ve afectado por la calidad de las imágenes que se dispongan para esta tarea.

- El Haar Cascade Classifier resulta ser un método de detección de ojos efectivo, permitiendo obtener múltiples características de la imagen para poder tratarla y así ahorrar tiempo de procesamiento, así como también posee la gran ventaja que se pueda usar en aplicaciones en tiempo real.

- La aplicación de técnicas de visión por computador permitió identificar de forma objetiva las pupilas en función de su seguimiento en tiempo real.

- Determinar el posicionamiento de las pupilas permite identificar de una manera práctica y efectiva hacia donde está mirando una persona con solo realizar un escalado de sus coordenadas al plano real.

- El uso apropiado de los métodos y técnicas mostrados permiten que el sistema de seguimiento del ojo (pupila), funcione de manera óptima.

- El resultado obtenido hace que este sistema de visión por computador sea prometedor para resolver el problema planteado, ya que es un inicio para la construcción de un sistema asistencial para personas con discapacidad motora.

8. RECOMENDACIONES

- Evitar cualquier tipo de objeto o característica que pueda perjudicar las mediciones, tales como maquillaje, pestañas postizas, anteojos, entre otros.

- Ejecutar la aplicación en lugares donde se pueda tener un ambiente controlado y con la menor cantidad de luz posible con el fin de evitar fallas en el funcionamiento del sistema.

- La fuente de iluminación no debe estar frente al rostro debido que se pueden presentar problemas de brillo en el área de los ojos.

- Antes de iniciar confirmar que la Raspberry Pi y el programa Matlab estén debidamente conectados.

- Asegurarse de tener un buen soporte para el rostro que no permita movimientos involuntarios que puedan afectar las mediciones.

Las medidas recomendadas para un óptimo funcionamiento del sistema son:

- La cámara debe encontrarse en un rango de distancia de 20cm a 25cm de los ojos con una inclinación aproximada de 0°.

- El tablero debe estar máximo a 75cm de distancia de la cámara y debe tener dimensiones mínimas de 55x30 cm.

- El tablero tiene que encontrarse en lo posible sobre la línea de vista de los ojos con un ángulo no mayor a 15°.

- La mesa debe tener entre 50cm y 70cm de alto, los ojos y la cámara deben ubicarse aproximadamente a 50cm y el tablero aproximadamente a 40cm sobre la base de la mesa.

9. TRABAJOS FUTUROS

Como trabajos futuros para mejorar aún más este sistema de seguimiento del ojo (pupila), usando técnicas de visión artificial, se puede realizar:

- Remplazar el soporte del rostro por uno que lo mantenga totalmente inmóvil, con el fin de prescindir del marcador del rostro como punto de referencia.

- Permitir posicionar el cursor en la pantalla de un computador de acuerdo con el seguimiento de los ojos (pupilas) de la persona.

- Generar la posibilidad de reconocimiento de gestos faciales para realizar acciones de selección como el clic derecho e izquierdo del mouse por medio del parpadeo de cada ojo, apagado del equipo manteniendo los ojos cerrados por un tiempo, ejecución del teclado por medio del tablero de posicionamiento, entre otras.

- Usar un mejor algoritmo de machine learning capaz de tener un mejor desempeño sin importar la iluminación del entorno y demás factores que puedan afectar la detección.

- Realizar un prototipo de un sistema asistencial capaz de funcionar en plataformas como Android e IOS.

BIBLIOGRAFÍA

Areitio, G., & Ana, A. (2009). Información, Informática e Internet: del ordenador personal a la Empresa 2.0. España: Visión Libros.

De la Fuente López , E., & Trespaderne, F. (2012). *Visión Artificial Industrial.* Valladolid: Secretariado de Publicaciones .

Departamento de Ingeniería electrónica, T. y. (2006). *Segmentación. Transformada de Hough.* Universidad de Jaén.

Duda, R., & Hart, P. (1973). *Pattern Classification and Scene Analysis.* New York: John Wiley & Sons.

González Marcos, A., Martínez de Pisón Ascacíbar, F., Pernía Espinoza, A., Alba Elías, F., Castejón Limas, M., Ordieres Meré, J., & Vergara González, E. (2006). *Técnicas y algoritmos básicos de visión artificial.* Universidad de la Rioja - Servicio de Publicaciones .

Haar Cascades vs. LBP Cascades en detección de rostros. (12 de 1 de 2012). Obtenido de Stack Overflow: https://stackoverflow.com/questions/8791178/haar-cascades-vs-lbp-cascades-in-face-detection

Ingeniería de Sistemas y Automática. (2010). Visión por computador. Universidad Miguel Hernández.

JOSE F. ET AL. VELEZ SERRANO. (2003). *Visión por computador.* Madrid: S.L. - DYKINSON.

Klette, D. (2014). *Concise Computer Vision - An Introduction into Theory and* Algorithms (Vol. 1). Department of Computer ScienceThe University of Auckland, New Zealand: Springer.

López Peña, A., Valveny, E., & Vanrell, M. (2017). *Detector basado en Haar/Adaboost - Cascada de clasificadores.* Obtenido de Coursera: https://es.coursera.org/learn/deteccion-objetos/lecture/pRnHu/l5-5-cascada-de-clasificadores

Lopez Perez, N., & Toro Agudelo, J. (2012). *Tecnicas de biometrias basadas en patrones faciales del ser humano.* Pereira, Colombia : Universidad Tecologica de Pereira, Ingenieria de Sistemas y Computación.

Mathworks. (s.f.). *Comparación entre MATLAB y Python: principales razones para elegir MATLAB.* Obtenido de Mathworks: https://la.mathworks.com/products/matlab/matlab-vs-python.html

Mathworks. (s.f.). *Creación de apps con interfaces gráficas de usuario en MATLAB. GUI de Matlab.* Obtenido de Mathworks: https://la.mathworks.com/discovery/matlab-gui.html

Mathworks. (s.f.). *Hough Transform.* Obtenido de Mathworks: https://la.mathworks.com/help/images/hough-transform.html?searchHighlight=hough&s_tid=doc_srchtitle

Mathworks. (s.f.). *Train a Cascade Object Detector.* Obtenido de Mathworks: https://la.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html

Raspberry Pi. (2018). Obtenido de Pi NoIr Camera V2: https://www.raspberrypi.org/products/pi-noir-camera-v2/

Sialat M., Khlifat N., Bremond F., & Hamrouni K. (2009). People detection in complex scene using a cascade of Boosted classifiers based on Haar-like-features. *Intelligent Vehicles Symposium*, 83-87.

Tolosa Borja, C., & Giz Bueno, Á. (2010). *Sistemas Biometricos.* Obtenido de https://www.dsi.uclm.es/personal/MiguelFGraciani/mikicurri/Docencia/Bioinformatica/

Turégano, E. (2006). EyeBoard: Un Periferico Alternativo Visual. http://robolab.unex.es/research/doc/libro.pdf.

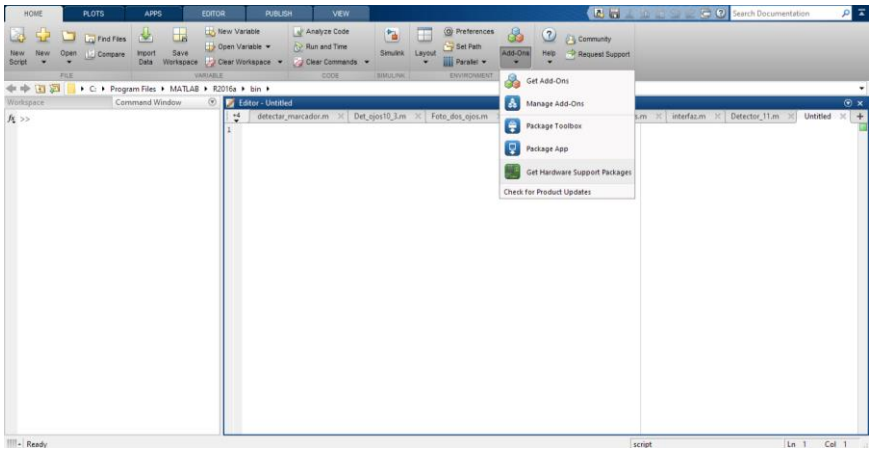
Viola, P., & Jones, M. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*.

ANEXOS

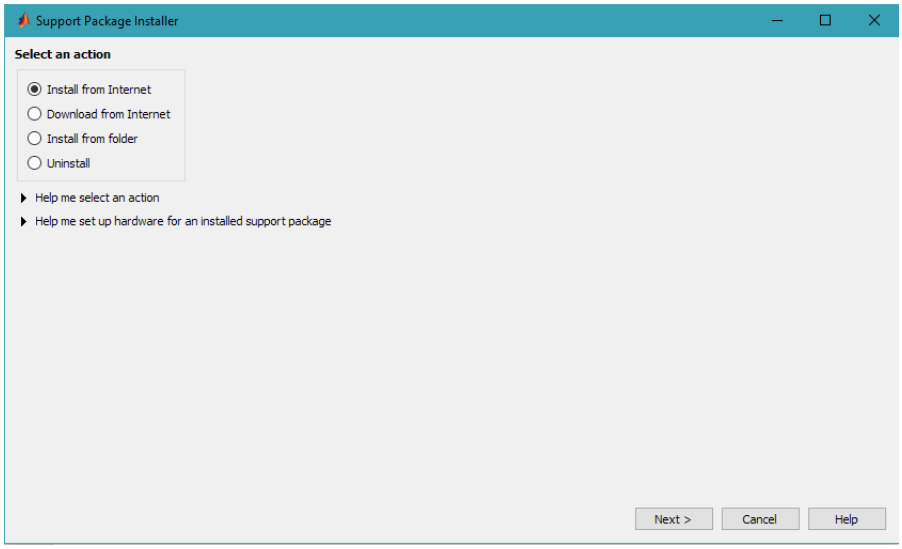
ANEXO A: CONEXIÓN ENTRE LA RASPBERRY PI Y MATLAB

Antes que nada debía asegurarse que el computador tuviese instalada la versión Matlab 9.0 (R2016a) y que se encontrara conectado dentro de la misma red Wi-Fi que la Raspberry Pi. Si se contaba con un computador de escritorio, en el mercado existen adaptadores de red inalámbrica con conexión USB. Además de esto se debía contar con una tarjeta MicroSD y un adaptador para tarjeta SD. Para realizar la conexión se debían seguir estos pasos:

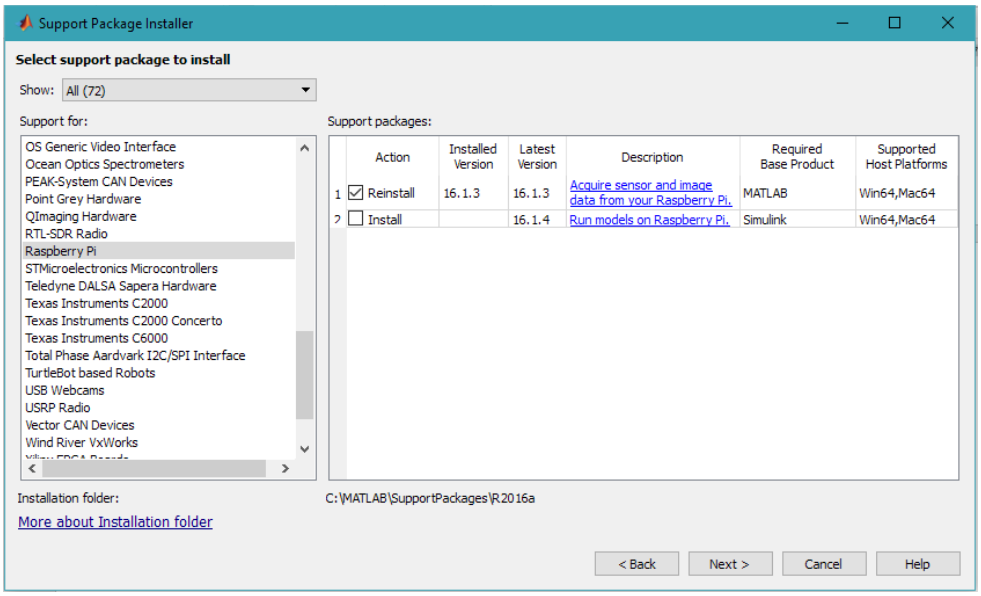
1. En la pestaña home hacer clic en Add Ons y luego en Get Hardware Support Packages.



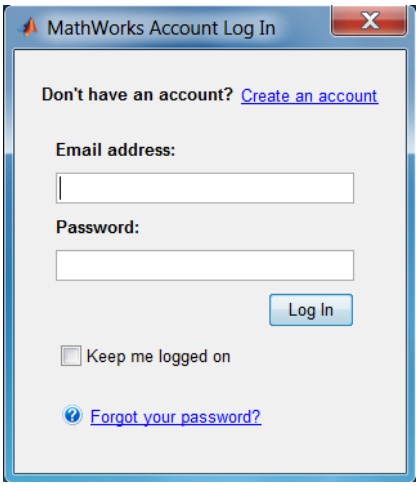
2. Luego seleccionar Install from Internet, seguido de el botón Next>.



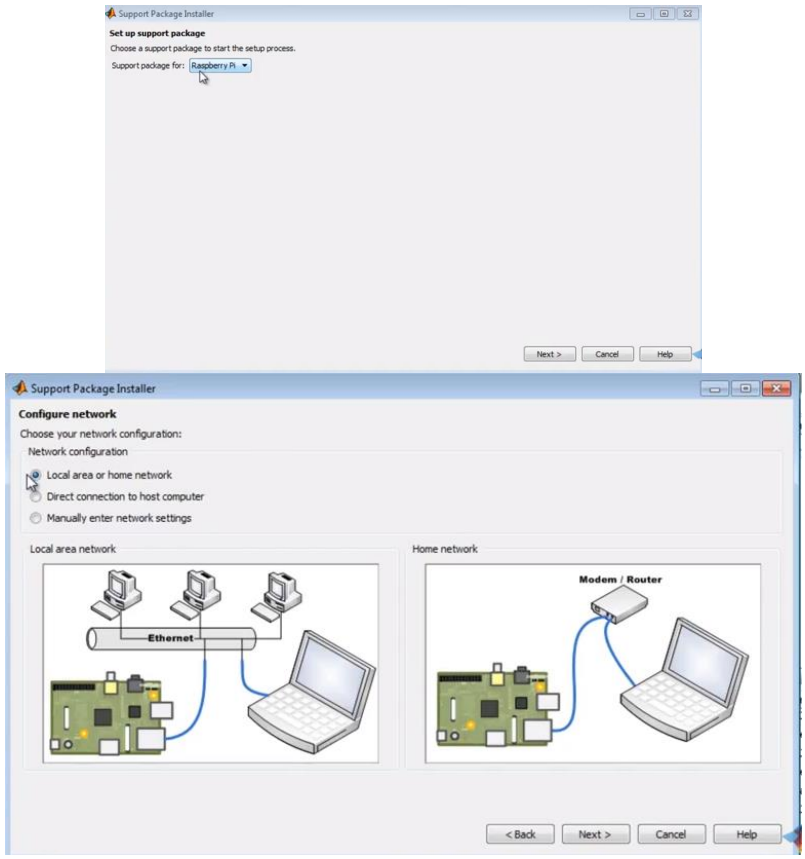
3. Después se busca entre las categorías de soporte la Raspberry Pi y se selecciona instalar (Install) la que contenga la descripción: Acquire sensor and image data from your Raspberry Pi y luego clic en Next>.



4. Luego de esto la aplicación pide iniciar sesión con la cuenta de correo y la contraseña preestablecida. Si no se tiene una cuenta de ingreso se selecciona la opción créate an account y se realiza su creación.

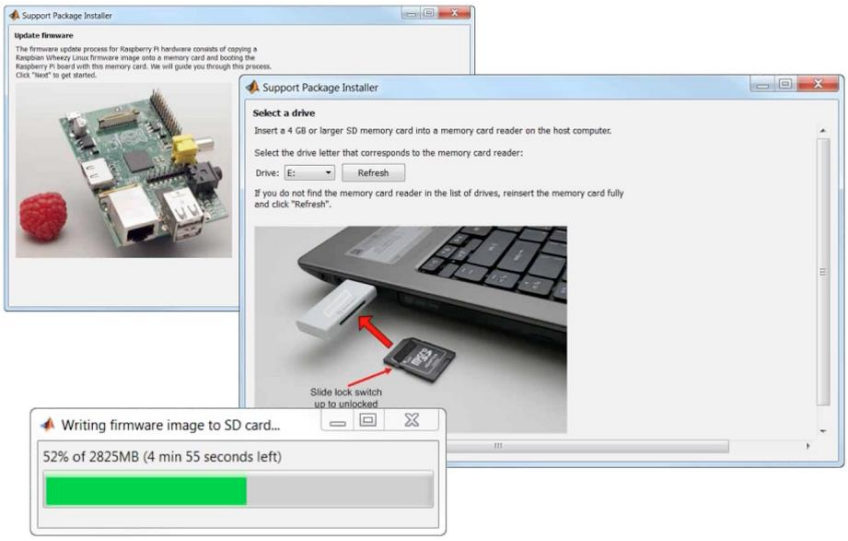


5. En la siguiente pantalla se selecciona Raspberry Pi seguido de Next>. En la siguiente ventana se selecciona la opción Local área or home network, debido a que se va a realizar una conexión por medio de la red Wi-Fi. Luego se hace clic en Next>.

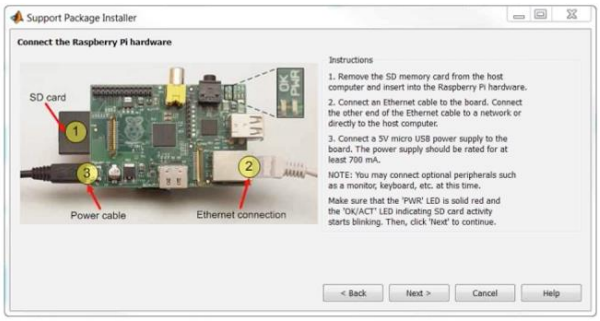


6. Luego de esto aparece una pantalla donde se descarga el firmware, el cual establece la lógica de más bajo nivel para controlar los circuitos electrónicos de la Raspberry Pi, en otras palabras, es el software que tiene directa interacción con el hardware siendo así el encargado de controlarlo para ejecutar correctamente las instrucciones externas. Se hace clic en Next>.

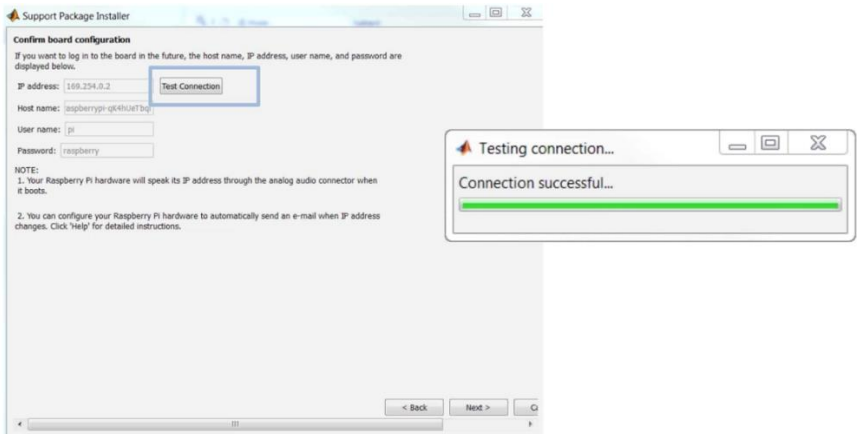
7. Después de esto se debía insertar la tarjeta MicroSD en el adaptador de tarjetas SD y conectarlo al computador por el puerto de tarjetas SD o por un adaptador de SD a USB. Una vez detectado el dispositivo por el programa, se hace clic en Next> y se espera el tiempo que dure la escritura de la imagen en la SD.



8. Luego se debe extraer la MicroSD e insertarla en la Raspberry Pi, después encenderla conectándola a una fuente de 5V. Después de esto se realiza la identificación con el computador haciendo clic en Next> y se espera que la Raspberry sea detectada.



9. Luego de esto se realiza una prueba de conexión entre los dos dispositivos haciendo clic en Test Connection, luego debe aparecer una notificación: Connection successful para indicar que la conexión se realizó correctamente.



10. Después se cierran las ventanas y se dirige a la Raspberry Pi con el fin de establecer la dirección ip dada en la conexión como una ip fija, debido a si la dirección cambia no se podría establecer conexión con el programa Matlab. Los comandos ejecutados son los siguientes:

    Usuario: pi
    Contraseña: raspberry

    sudo nano /etc/network/interfaces

    Auto lo
    iface lo inet loopback
    iface eth0 inet dhcp
    allow-hotplug wlan0
    auto wlan0

    iface wlan0 inet static
        address 192.168.1.7
        netmask 255.255.255.0
        gateway 192.168.1.1
        wpa-ssid “Movistar_8664957”
        wpa-psk “G9074653op”

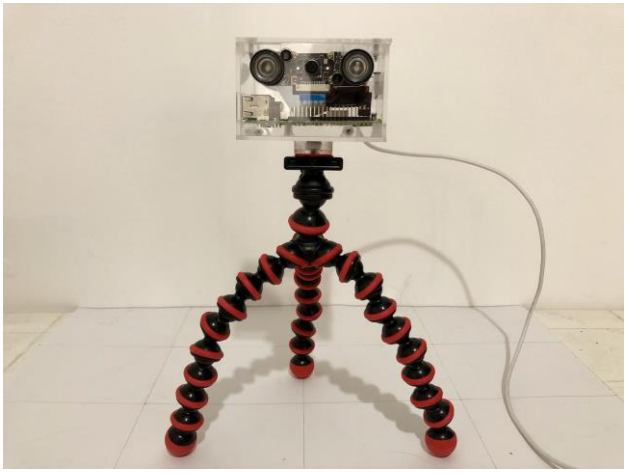
Donde cada instrucción se relaciona en la siguiente tabla:

Instrucción	Significado
address	Dirección Ip Estática
netmask	Mascara de red
gateway	Puerta de enlace
wpa-ssid	Nombre de la red Wi-Fi
wpa-psk	Contraseña de la red Wi-Fi

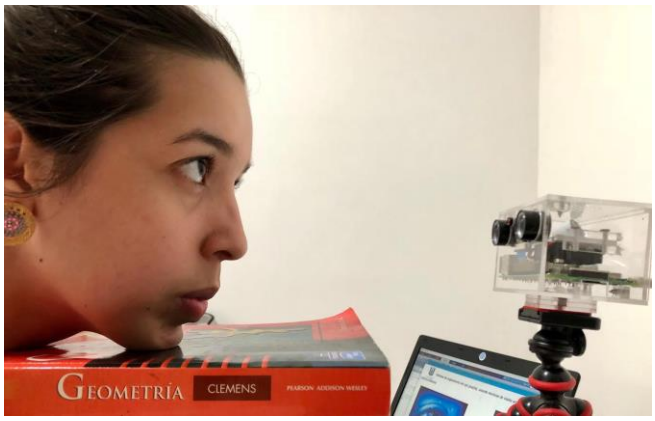
ANEXO B: MANUAL DE USUARIO DEL SISTEMA

Antes de iniciar debe asegurarse que la Raspberry Pi y el computador se encuentren debidamente conectados entre sí, como se explica en el **ANEXO A.**

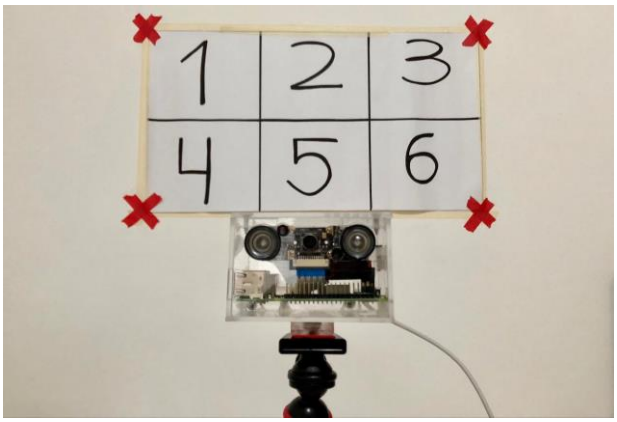
1. En la mesa (de 50cm a 70cm de altura) ubicar la Raspberry Pi (que se encuentra conectada a la cámara) en el soporte y conectarla a una fuente de alimentación micro USB de alta calidad de 2.5 A a 5V.



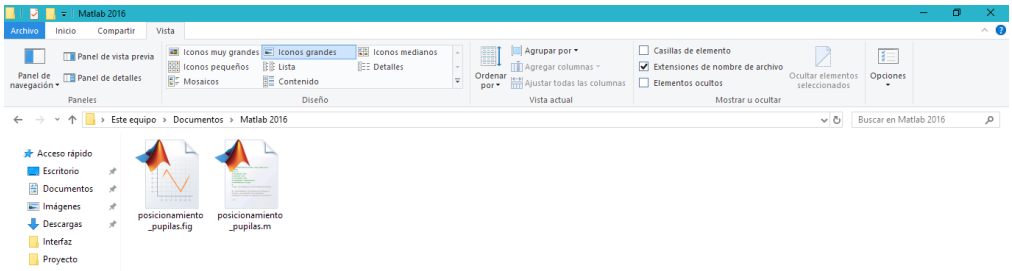
2. Situar el soporte del rostro de tal manera que los ojos queden en posición directa hacia la cámara con una distancia entre 20cm y 25cm y un ángulo aproximado de 0°.



3. Ubicar el tablero de posicionamiento (de 55x30cm) centrado a máximo 75 cm de distancia de la cámara y con una inclinación no mayor a 15° de la posición de los ojos, con una equis (X) roja en cada esquina.



4. Encender el computador e iniciar la aplicación *“posicionamiento_pupilas.fig”,* la cual se ejecuta en el programa Matlab.



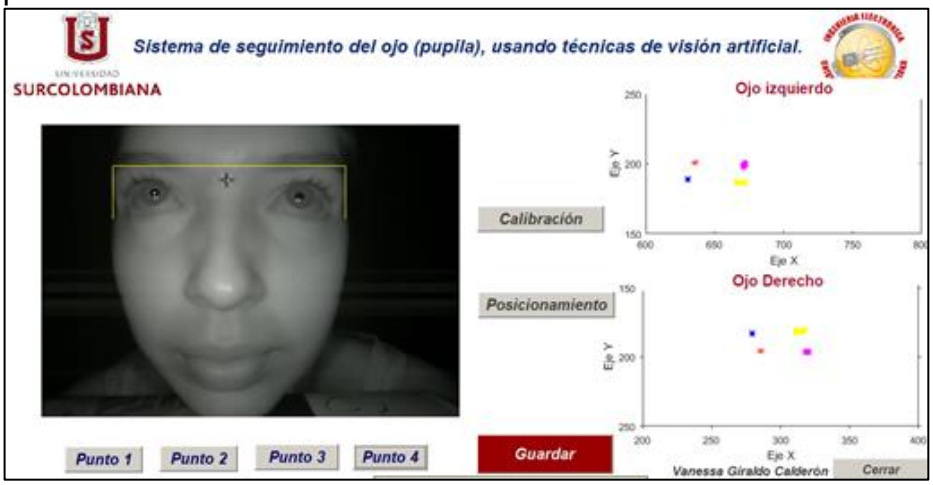
5. Colocar en el rostro el marcador en forma de cruz en la mitad de los dos ojos y posteriormente ubicar el rostro en el soporte cumpliendo con las condiciones de distancia y altura establecidas.

6. En la aplicación presionar el botón *Calibración* donde aparece un video en vivo de detección y seguimiento de las pupilas, donde se resaltaban con una cruz blanca los centros de las pupilas y el centro del marcador del rostro en forma de cruz.

7. Con los cuatro botones que aparecen en pantalla, se deben calibrar los puntos máximos del tablero de posicionamiento; observando cada una de las equis (X) rojas se acumulan los centros de las pupilas al presionar cada botón, donde:
    - *Punto 1* se calibra mirando por unos segundos la equis (X) superior izquierda.
    - *Punto 2* se calibra mirando por unos segundos la equis (X) superior derecha.
    - *Punto 3* se calibra mirando por unos segundos la equis (X) inferior derecha.
    - *Punto 4* se calibra mirando por unos segundos la equis (X) inferior izquierda.

Nota: Para identificar cuando se calibró el punto, al finalizar la calibración de cada uno suena un *beep.*

8. Cuando se tienen los cuatro puntos máximos calibrados, se debe observar que en los planos del recorrido de los ojos (ubicados a la derecha de la pantalla) los puntos formen las esquinas de un rectángulo y posteriormente presionar el botón *Guardar* para almacenar los resultados.



Nota: si los puntos no formaban un rectángulo se debían volver a calibrar todos los puntos nuevamente.

9. Presionar el botón *Posicionamiento* y mirar uno de los seis cuadrantes del tablero fijamente por algunos segundos hasta escuchar un *beep.* Inmediatamente después el programa muestra una imagen con el cuadrante seleccionado en rojo y un mensaje indicando el número del cuadrante que se observó.

Nota: El posicionamiento se puede repetir todas las veces que se desee, siempre y cuando no se produzcan movimientos en la posición del rostro, de ser así es necesario recalibrar la cámara para obtener resultados correctos.

10. Para cerrar el programa en cualquier momento, se presiona el botón *Cerrar* ubicado en la parte inferior derecha de la pantalla.