Neiva, 25 de octubre de 2019

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

Miguel Ángel Javela Peña, con C.C. No. 1075299716,

Víctor Hernando Moreno Perdomo, con C.C. No. 1075298928,

Autor(es) de la tesis y/o trabajo de grado o _____

Titulado Diseño e implementación de un prototipo para el reconocimiento de iris

presentado y aprobado en el año 2019 como requisito para optar al título de Ingeniero electrónico;

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:

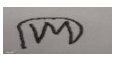
- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con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

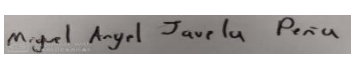
- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

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: 

EL AUTOR/ESTUDIANTE:

Firma: 

TÍTULO COMPLETO DEL TRABAJO: Diseño e implementación de un prototipo para el reconocimiento de iris.

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
Javela Peña Moreno Perdomo	Miguel Ángel Víctor Hernando

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: Ingeniero Electrónico

FACULTAD: **Ingeniería**

PROGRAMA O POSGRADO: Ingeniería electrónica

CIUDAD: Neiva **AÑO DE PRESENTACIÓN:** 2019 **NÚMERO DE PÁGINAS:** 81

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

Diagramas x Fotografías x Grabaciones en discos___ Ilustraciones en general___ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros x

SOFTWARE requerido y/o especializado para la lectura del documento:

MATERIAL ANEXO: Código fuente y manual de usuario.

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

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Inglés
1.	Visión por computador	Computer vision
2.	Inteligencia artificial	Artificial intelligence
3.	Transformada de wavelet	Wavelet Transform
4.	Iris	Iris
5.	Segmentación	Segmentation

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

El fin de este proyecto de grado fue el diseño e implementación de un prototipo funcional para el reconocimiento de personas por medio del iris en un ambiente semi-controlado. El algoritmo fue implementado utilizando el lenguaje de programación de código abierto PYTHON.

El prototipo inicialmente adquiere imágenes de los usuarios que serán registrados en el sistema, cada imagen es pre-procesada adecuadamente para luego pasar por un algoritmo de procesamiento donde la zona correspondiente al iris es segmentada satisfactoriamente, es decir, se obtienen los parámetros necesarios para delimitar la zona correspondiente al iris, la cual es normalizada para obtener una plantilla que solo incluye la zona de interés (el iris), y con estas realizar un proceso de extracción de características de cada plantilla mediante la transformada de Wavelet.

Finalmente, con estas plantillas se alimentan dos algoritmos de clasificación pertenecientes al área de inteligencia artificial (*machine learning*), los cuales son las redes neuronales y las máquinas de soporte vectorial, y a partir de los resultados que se obtuvieron se escogió el de mejor rendimiento.

ABSTRACT: (Máximo 250 palabras)

The aim of this project is designing and implementation of a functional iris recognition prototype for people in a semi-controlled environment. The algorithm was created on programming language of open source known as *PYTHON.*

Initially, the prototype gets images from users who will be registered in the system, each image is pre-processed adequately and then this image will be segmented successfully by a processing algorithm, which is normalized to obtain a template where only is interest area (The Iris), and with these to carry out a characteristics extraction process from each template using the Wavelet Transform.

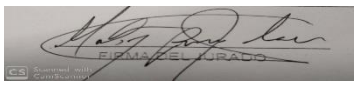
Finally, the resulting templates feed the two classification algorithms of artificial intelligence area, which are the artificial neural networks and the machines of vectorial support. Also looked at results of both classification algorithms for select of the method with the best performance

APROBACION DE LA TESIS

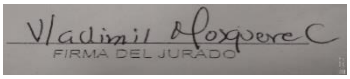
Nombre Presidente Jurado:

Firma:

Nombre Jurado: Martin Diomedes Bravo Obando

Firma: 

Nombre Jurado: Vladimir Mosquera Cerquera

Firma: 

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA EL RECONOCIMIENTO DE IRIS

MIGUEL ÁNGEL JAVELA PEÑA
VICTOR HERNANDO MORENO PERDOMO

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA
2019

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA EL RECONOCIMIENTO DE IRIS

MIGUEL ÁNGEL JAVELA PEÑA
VICTOR HERNANDO MORENO PERDOMO

Trabajo de tesis

Director

JOSÉ DE JESÚS SALGADO PATRÓN
MSc. Ingeniería Electrónica y de computadores

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA, HUILA
2019

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma del Jurado

_____

Firma del Jurado

Neiva, Julio 30 de 2019

Agradezco primero a Dios por darme la salud y las fuerzas para llevar durante este tiempo mi carrera universitaria y este proyecto de grado, a mi madre Nibia por todas sus oraciones, por siempre estar para mí cuando la necesito y por ser el motivo por quien quiero superarme y ser mejor, a mis tías, Rosa y Laurentina, por siempre encomendarme en sus oraciones y guiarme por el mejor camino, a mi primo Alexis que en estos momentos está pasando por un momento difícil en su vida pero que va a superar, además, siempre me impulsa a luchar por mis sueños porque siempre manifiesta su orgullo y admiración por mí, a mi novia Camila por brindarme todo su amor y apoyo en cada momento difícil que he tenido. Así mismo a toda mi familia y amigos que de una y otra manera ayudaron en este proceso de formación profesional, muchas gracias.

Miguel Ángel Javela Peña

Primeramente, agradezco a Dios por ayudarme a superar todas las dificultades encontradas al largo de mi vida y carrera. A mi padre Medardo por siempre estar apoyándome y enseñándome las experiencias de la vida, a mi madre Mercedes por ser esa compañera fiel que siempre he tenido y que ha confiado en mí sin importar nada, a mi hermana Ximena por todo su amor y bondad brindada; esto es de todos nosotros porque todos hemos luchado, solo ustedes saben todas las dificultades por las que hemos pasado, siempre los llevare en mi corazón. A toda mi familia porque cada uno ha aportado un granito de arena con esa ilusión de verme realizado en lo que siempre he querido. A todos mis amigos porque de ellos he aprendido y recibido ayuda en los momentos que he necesitado.

Víctor Hernando Moreno Perdomo

AGRADECIMIENTOS

Mostramos nuestros más sinceros agradecimientos a nuestro director de tesis José de Jesús Salgado por siempre estar atento a nuestras dudas, solicitudes y guiarnos de manera correcta, a nuestros jurados de tesis Vladimir Mosquera y Martin Diomedes Bravo por brindarnos de manera muy cordial sus conocimientos académicos a lo largo de toda la carrera que fueron de mucha ayuda para la realización de este proyecto.

TABLA DE CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

GLOSARIO

Iris: es uno de los elementos que componen el sistema óptico, localizado en el ojo entre la córnea y el cristalino. Su morfología es la de una membrana circular y coloreada en cuyo centro se encuentra la pupila y limitado en su parte exterior por la esclerótica.

Pupila: círculo negro ubicado en el centro del iris, cuya función es regular la cantidad de luz que ingresa al ojo aumentando o disminuyendo su tamaño.

Esclerótica: tejido blando que cubre la parte exterior del globo ocular.

Segmentación: proceso que consiste en agrupar partes de la imagen, hasta llegar a un nivel donde se aíslen las regiones u objetos de interés.

Algoritmo: concatenación de pasos lógicos que permiten llegar a la solución de un determinado problema.

Inteligencia artificial: rama de las ciencias computacionales que se ocupa de los símbolos y métodos no algorítmicos para la resolución de problemas con el propósito de crear máquinas que imiten la capacidad de razonamiento del ser humano.

Umbral: valor numérico representativo de la intensidad de un pixel usado para diferenciar entre el fondo y el objeto de una imagen digital.

Pixel: es la parte homogénea en color más pequeña de una imagen digital generalmente compuesta por 8 bits

RESUMEN

El fin de este proyecto de grado fue el diseño e implementación de un prototipo funcional para el reconocimiento de personas por medio del iris en un ambiente semi-controlado. El algoritmo fue implementado utilizando el lenguaje de programación de código abierto PYTHON.

El prototipo inicialmente adquiere imágenes de los usuarios que serán registrados en el sistema, cada imagen es pre-procesada adecuadamente para luego pasar por un algoritmo de procesamiento donde la zona correspondiente al iris es segmentada satisfactoriamente, es decir, se obtienen los parámetros necesarios para delimitar la zona correspondiente al iris, la cual es normalizada para obtener una plantilla que solo incluye la zona de interés (el iris), y con estas realizar un proceso de extracción de características de cada plantilla mediante la transformada de Wavelet.

Finalmente, con estas plantillas se alimentan dos algoritmos de clasificación pertenecientes al área de inteligencia artificial (*machine learning*), los cuales son las redes neuronales y las máquinas de soporte vectorial, y a partir de los resultados que se obtuvieron se escogió el de mejor rendimiento.

PALABRAS CLAVES: Visión por computador, Inteligencia artificial, Transformada de Wavelet, Iris y Segmentación.

ABSTRACT

The aim of this project is designing and implementation of a functional iris recognition prototype for people in a semi-controlled environment. The algorithm was created on programming language of open source known as *PYTHON.*

Initially, the prototype gets images from users who will be registered in the system, each image is pre-processed adequately and then this image will be segmented successfully by a processing algorithm, which is normalized to obtain a template where only is interest area (The Iris), and with these to carry out a characteristics extraction process from each template using the Wavelet Transform.

Finally, the resulting templates feed the two classification algorithms of artificial intelligence area, which are the artificial neural networks and the machines of vectorial support. Also looked at results of both classification algorithms for select of the method with the best performance.

Keywords: computer vision, artificial intelligence, Wavelet Transform, Iris and segmentation.

INTRODUCCIÓN

Cada día se incrementa la movilización de personas a distintos lugares y por ende muchas empresas quieren llevar un registro de personas que acceden a determinados lugares o restringirlo a solo personal autorizado, esto ha llevado que se busquen diferentes métodos para automatizar este tipo de procesos pero sin dejar de brindar una seguridad muy confiable, debido a esta necesidad se ha incurrido en diferentes técnicas de reconocimiento biométrico como las de [1],[2],[3],[4] que presentan buenas prestaciones para el objetivo mencionado.

El reconocimiento de iris como se mencionó anteriormente es una técnica de la biometría que ha tomado auge en los últimos años, este posee características de medición en la biometría que sobresalen de las demás técnicas entre ellas como la invarianza en el tiempo lo que sí sucede en el reconocimiento facial o cambios temporales de su composición como sucede con el reconocimiento de voz que puede verse afectada, por ejemplo, cuando el usuario tiene resfriado. Este método de identificación ya está siendo implementado por grandes empresas tecnológicas como Samsung, Microsoft, Fujitsu, ZTE, LG, entre otras las cuales apuestan por un sistema de identificación más seguro que los que se han venido implementando anteriormente como el reconocimiento facial o la huella dactilar.

Uno de los grandes impulsores para el desarrollo de esta rama de la biometría han sido las técnicas de *machine learning*, entre las más usadas están las redes neuronales y las máquinas de soporte vectorial debido a su buena fiabilidad en el proceso de identificación de los usuarios puesto que tienen la capacidad de imitar el proceso de aprendizaje del cerebro humano, lo que ayuda a que el sistema sea lo bastante robusto en ambientes con gran cantidad de interferencia además posee una de las ventajas de la visión por computadora la cual es, que no se equivoca ante ilusiones ópticas lo que sí sucede con el cerebro humano. El desafío de la visión por computadora en este caso es obtener solamente la información necesaria donde se hace implementando herramientas matemáticas como la *transformada de Wavelet*, la *transformada del coseno*, *transformada de Fourier* entre otras.

En este proyecto de grado se explica el funcionamiento de un prototipo para el control de acceso de personas por medio de reconocimiento de iris, el desarrollo de este proyecto de grado puede ser la base e inicio de un sistema de reconocimiento de iris para la universidad Surcolombiana que se pueda aplicar en diferentes lugares

[1] Soliman, Naglaa F. and Mohamed, Essam and Magdi, Fikri and El-Samie, Fathi E Abd and AbdElnaby, M, 2017.
[2] Cordea, Marius and Ionescu, Bogdan and Gadea, Cristian and Ionescu, Dan, 2019.
[3] Ghorbani, Mohammadjavad and Alizadeh, Mahdi and Omran, Alireza Esfahani and Asem, Morteza Modarresi, 2018.
[4] Poornima, S and Rajavelu, C and Subramanian, S, 2010.

como laboratorios para llevar un control detallado del acceso de estudiantes, o también para el ingreso de personas a la universidad o para la venta de boletas del restaurante universitario ayudando así al avance en la investigación y aplicación de la inteligencia artificial y visión por computador en diferentes problemáticas que requieran los nuevos avances de la sociedad científica y de esta manera acercar este tipo de tecnología a nivel regional para impulsar e incentivar a los estudiantes en nuevas áreas del conocimiento.

PLANTEAMIENTO DEL PROBLEMA

La seguridad siempre ha sido un tema de interés para la humanidad, más cuando se quiere proteger o limitar el acceso de individuos a determinadas zonas u objetos que requieren gran protección por su importancia. Este control de acceso ha venido evolucionando y ha tenido un rápido avance en los últimos años donde se han utilizado diferentes métodos y sistemas para lograr una identificación exacta y precisa de los individuos.

Estos avances se deben a la investigación y validación de los diferentes algoritmos desarrollados para la identificación, donde una de las técnicas en la actualidad es la autentificación biométrica[5], resaltando el reconocimiento de iris a través de *machine learning*, ofreciendo una seguridad muy confiable debido a que el objeto de identificación (ojo humano), es algo único para cada individuo y no se puede plagiar fácilmente[6],[7].

Un área de las ciencias es la inteligencia artificial, la cual está en crecimiento y es una de las apuestas más prometedoras de las grandes empresas tecnológicas[8], donde se ha encontrado que la investigación de técnicas de *machine learning*[9] aplicadas a esta problemática todavía no se ha explotado completamente en la comunidad científica[10]. Por el lado de la universidad Surcolombiana, se han hecho trabajos de investigación acerca de métodos para el procesamiento de imágenes y técnicas de reconocimiento, pero no han estado enfocadas en el reconocimiento de iris[11],[12],[13].

Ante el conocimiento de este tipo de avances tecnológicos que están en constante desarrollo y la necesidad de proteger las instalaciones de la Universidad Surcolombiana, ¿Cómo puede un estudiante aportar resultados significativos en la investigación, desarrollo e implementación de un prototipo confiable y de bajo costo para el reconocimiento de iris que mejore el cuidado de las instalaciones de la Universidad Surcolombiana?

[5] Anil k., Jain and Arun, Ross and Salil, Prabhakar, 2004
[6] Sinha, Vijay Kumar and Gupta, Anuj Kumar and Mahajan, Manish, 2018
[7] Wildes, Richard P, 1997
[8] Garcìa, 2012
[9] Anzola, 2016
[10] De Marsico, Maria and Petrosino, Alfredo and Ricciardi, Stefano, 2016
[11] Robayo, Faiber Ignacio and Barrera, Ana Marìa and Polanco, Laura Camila, 2015
[12] Giraldo Calderon, 2018
[13] Ossa Vargas, Manuel Fernando and Díaz Vargas, Jonathan Miguel, 2015

JUSTIFICACIÓN

Hoy en día en lugares donde hay una constante circulación de personas, se debe permitir el ingreso solo a individuos previamente seleccionados, esto con el fin de brindar la mayor seguridad posible en dichos sitios. Para lograr este objetivo, se necesita un sistema de seguridad para el control de acceso a personas.

El desarrollo de este objetivo no es una tarea sencilla y aunque la visión por computadora ha dado grandes avances para este tipo de sistemas, mediante métodos capaces de hacer contraparte a las diferentes perturbaciones que puedan presentarse como problemas en la etapa de procesamiento, los cuales no han sido solucionados completamente; por su parte, el iris ofrece características como la universalidad, distinción, permanencia y coleccionabilidad[14],[15],[16] , que son de gran ayuda para enfrentar las diferentes perturbaciones causante de estos problemas. Por lo tanto, es importante investigar nuevos métodos de trabajo que faciliten la realización de proyectos con visión artificial que no requieran supervisión humana y por el contrario sean autónomos, igualmente hacer aportes tecnológicos que contribuyan al beneficio de una región y así darle reconocimiento a nivel de investigación.

Una problemática que presentan los sistemas de seguridad es su alto costo de implementación, puesto que, si se necesita brindar seguridad a un determinado objeto o lugar, es necesario un hardware de alta precisión lo que conlleva una gran inversión para adquirir los equipos adecuados, por lo que todos los sistemas de seguridad no son accesibles para todas las empresas o personas del común que no cuenten con los recursos necesarios para adquirir un sistema de estas características.

Por esto es importante implementar un prototipo de identificación mediante técnicas de *machine learning*, descubriendo nuevos saberes de gran aporte a la ciencia. Además de que se implementará este prototipo con software libre lo que ayuda a reducir costos de implementación y a la vez brindando la alta confiabilidad requerida en este sistema, siendo novedoso para la UNIVERSIDAD SURCOLOMBIANA.

[14] Anil k., Jain and Arun, Ross and Salil, Prabhakar, 2004
[15] De Marsico, Maria and Petrosino, Alfredo and Ricciardi, Stefano, 2016
[16] Sibai, Fadi N and Hosani, Hafsa I and Naqbi, Raja M and Dhanhani, Salima and Shehhi, Shaikha, 2011

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Diseñar e implementar un prototipo para identificación de personas mediante el reconocimiento de iris.

1.2 OBJETIVOS ESPECÍFICOS

- Capturar imágenes del ojo humano que permitan obtener de manera óptima la información del iris.

- Desarrollar un algoritmo de procesamiento capaz de extraer la matriz de pixeles que abarca la región del iris.

- Realizar la extracción de características por diferentes métodos y seleccionar el de mejor desempeño para el proceso de identificación.

- Utilizar las técnicas de *machine learning* como clasificadores en el proceso de reconocimiento de iris para evaluar y seleccionar el de mejor desempeño.

- Probar y evaluar el rendimiento del prototipo en un ambiente de trabajo real.

- Validar la implementación del prototipo comparándolo con el trabajo titulado Iris Recognition based on kernels of support vector machine de Saminathan k., Chakravarthy T. y Chithra Devi.

2. SISTEMAS BIOMÉTRICOS

Un sistema de biometría puede ser definido como un sistema de reconocimiento de patrones presentes en los rasgos físicos intrínsecos de una persona, además este sistema tiene la capacidad de adquirir estos rasgos en forma de datos y posteriormente compararlos con plantillas almacenadas anteriormente en una base de datos[17].

Este tipo de sistemas puede operar en dos modos:

- **Verificación**: en este modo el usuario reclama una identidad y cada vez que este desea ser reconocido por el sistema, este brinda su identidad para que luego el sistema adquiera los rasgos físicos del usuario y los verifique con las propias plantillas del usuario anteriormente almacenadas en la base de datos.

- **Identificación**: en este modo el usuario no reclama una identidad solo es registrado en la base de datos del sistema y cuando un usuario desea ser reconocido por el sistema este adquiere sus rasgos físicos y pasa a hacer una comparación uno a uno con todas las plantillas almacenadas para luego informar si se encontró alguna coincidencia con estas[18] .

2.1 REQUISITOS DE UNA CARACTERÍSTICA BIOMÉTRICA

Cualquier característica fisiológica y/o conducta humana puede ser una característica biometría si cumple las siguientes condiciones:

- **Universalidad**: la característica biométrica debe ser común a toda la población.

- **Distinción**: se refiere al hecho de que la característica biométrica de una persona debe ser lo suficientemente diferente a la de otra persona para identificarlas.

- **Permanencia**: la característica biométrica debe permanecer lo suficientemente invariante en el tiempo para seguir distinguiendo a la persona.

- **Cobrabilidad**: la característica biométrica puede ser medida cuantitativamente[19].

Además de estos requerimientos hay otras cuestiones que debe tener en cuenta un sistema biométrico y estas son:

[17] Anil k., Jain and Arun, Ross and Salil, Prabhakar, 2004
[18] Wayman, 2001
[19] Anil k., Jain and Arun, Ross and Salil, Prabhakar, 2004

- **Rendimiento**: esto indica hasta dónde puede llegar la precisión y velocidad del sistema teniendo en cuenta los recursos disponibles para el diseño deseado y superando así los factores que afectan la operación y el ambiente de trabajo.

- **Aceptabilidad**: disposición que van a tener los usuarios para usar el dispositivo en su vida diaria.

- **Elusión**: indica la facilidad con que se puede engañar al sistema y así obtener resultados erróneos[20].

2.2 CARACTERÍSTICAS BIOMÉTRICAS DEL IRIS

El iris posee un gran potencial para ser una característica biométrica de alta calidad y con gran eficiencia. Por esta razón se entrará en detalle a explicar cómo este tejido humano cumple de gran manera con los requerimientos para un sistema biométrico.

Para iniciar, este tejido se encuentra muy bien protegido por la córnea, lo que hace que tenga mayor permanencia en el tiempo[21] que por ejemplo la huella dactilar que se desgasta después de muchos años en ciertos trabajos manuales. Además el iris en su mayoría es plano y su dilatación y contracción ante la incidencia de luz solo es controlada por un par de músculos complementarios (esfínter que cierra la pupila y el músculo dilatador de la pupila)[22] haciendo que el tamaño del iris sea más predecible que imágenes de la cara por ejemplo.

La distinción en el iris de cada ser humano viene dado desde la gestación embrionaria donde se forman finas texturas aleatoriamente que aseguran un patrón único para cada individuo[23],[24],[25].

A continuación, se presenta la Tabla 1 donde se muestra los parámetros que se tienen en cuenta en un identificador biométrico y su respectiva calificación.

[20] Jan, 2017.
[21] Nedjah, Nadia and Wyant, Rafael Soares and Mourelle, LM and Gupta, BB, 2017.
[22] Bowyer, Kevin W and Hollingsworth, Karen and Flynn, Patrick J, 2008.
[23] Nedjah, Nadia and Wyant, Rafael Soares and Mourelle, LM and Gupta, BB, 2017.
[24] Bowyer, Kevin W and Hollingsworth, Karen and Flynn, Patrick J, 2008.
[25] Jillela, Raghavender Reddy and Ross, Arun, 2015.

Tabla 1. Comparación de varias tecnologías biométricas.

Identificador biométrico	Universalidad	Distinción	Permanencia	Cobrabilidad	Rendimiento	Aceptabilidad	Elusión
ADN	A	A	A	B	A	B	B
Oído	M	M	A	M	M	A	M
Cara	A	B	M	A	B	A	A
Termograma facial	A	A	B	A	M	A	B
Huella Dactilar	M	A	A	M	A	M	M
Modo de Andar	M	B	B	A	B	A	M
Geometría de la mano	M	M	M	A	M	M	M
Vena de la Mano	M	M	M	M	M	M	B
Iris	A	A	A	M	A	B	B
Pulsación de Tecla	B	B	B	M	B	M	M
Olor	A	A	A	B	B	M	B
Impresión de la Palma	M	A	A	M	A	M	M
Retina	A	A	M	B	A	B	B
Firma	B	B	B	A	B	A	A
Voz	M	B	B	M	B	A	A

Fuente: Anil, K., Ross, A. y Salil, P. (2004). “An Introduction to Biometric Recognition”. [Tabla].

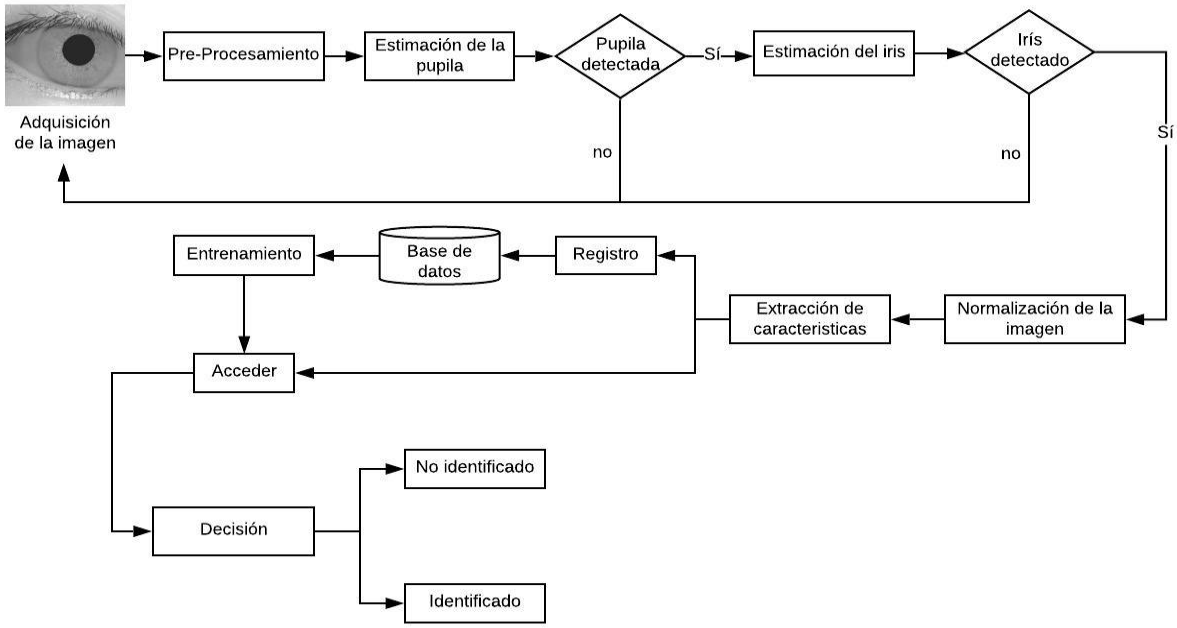
En esta tabla se observa que el iris tiene una buena percepción por parte de los autores de ANIL K., JAIN AND ARUN, ROSS AND SALIL[26] donde A, M y B corresponden a alto, medio y bajo respectivamente.

[26] Anil k., Jain and Arun, Ross and Salil, Prabhakar, 2004

3. DESCRIPCIÓN GENERAL DEL PROTOTIPO

La Figura 1 muestra cómo está constituido el prototipo, a continuación, en las subsecciones 3.1 y 3.2 de describirá detalladamente el hardware y software respectivamente de este prototipo.

Figura 1. Diagrama del prototipo diseñado.



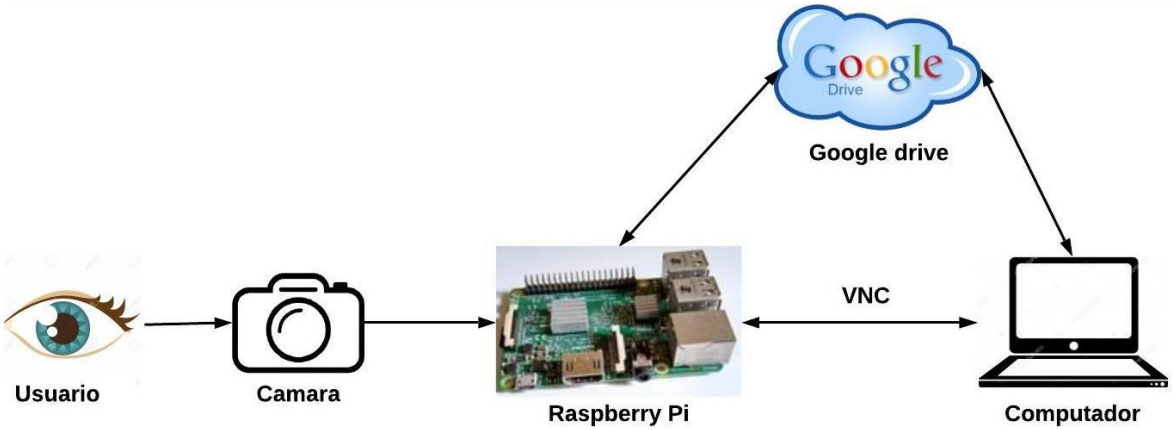
3.1 HARDWARE DEL PROTOTIPO

En esta parte del prototipo se ven involucrados 4 elementos, estos son la *raspicam*, la *rasperry pi*, un computador y el servidor, de los cuales se mostrará su respectiva conexión y algunas de sus características.

3.1.1 Conexión del prototipo

Los dispositivos nombrados anteriormente se conectan como lo evidencia la Figura 2.

Figura 2. Conexión del Hardware.



Fuente: Mora, C. (2019), Freepik Company S.L. (2019), Arauz, C. (2013), Depositphotos Inc. (2019).

Inicialmente se conectó la *raspberry* al computador por medio de la aplicación VNC (Viewer for Google Chrome), donde la *raspberry* es el servidor y de forma pasiva permite que el cliente tome el control que en este caso es un computador. Lo que es muy útil para el siguiente paso pues una vez conectada la *raspberry* con la *raspicam* por medio del bus de cinta de la cámara que ira conectado al puerto especial J3 de la *raspberry* y cuando se enciende la *raspberry* para comprobar la correcta conexión de la cámara se introduce en la terminal de la *raspberry* la siguiente línea de código:

raspistill -o image.jpg

Esta línea de código captura una imagen y la almacena en el directorio de imágenes de la raspberry.

3.1.2 Características del hardware

Raspberry: se escogió este ordenador de placa reducida porque permite trabajar con las librerías de código abierto de Python necesarias para este proyecto, también, cuenta con un puerto ethernet que facilito la conexión inicial entre la raspberry y el computador por medio del software VNC Viewer de Google usado para el monitoreo y usa de la raspberry, además, su estructura pequeña y portable permitió que se adaptara a la estructura diseñada y por lo tanto a diferentes entornos de trabajo.

Raspicam: se utilizó una cámara que consta de las siguientes características:

- Trae incorporado un filtro IR-CUT extraíble para eliminar la distorsión del color en la luz del día.

- Led infrarrojo que soporta visión nocturna.
- Sensor de 5 megapíxeles OV5647.
- Distancia de enfoque ajustable.
- Tamaño CCD: 0.25 pulgadas.
- Apertura (f): 1.2.
- Longitud focal: 8mm.
- Angulo de visión (diagonal): 40 grados.
- Sensor de mejor resolución: 1080p.

Figura 3. Raspicam.

Iluminación: fue necesario contar con una iluminación para poder hacer una buena captura a los detalles del iris, para esto se utilizó una iluminación tipo cálida direccional, el elemento utilizado para esto se observa en la Figura 4.

Figura 4. Iluminación tipo cálida.

Algunas de las características de este elemento son:

- Modelo No.: L3WFSS
- Potencia: 3W
- Corriente: 32mA
- Controlador de led incluido
- Glóbulo ocular LED 3x1W

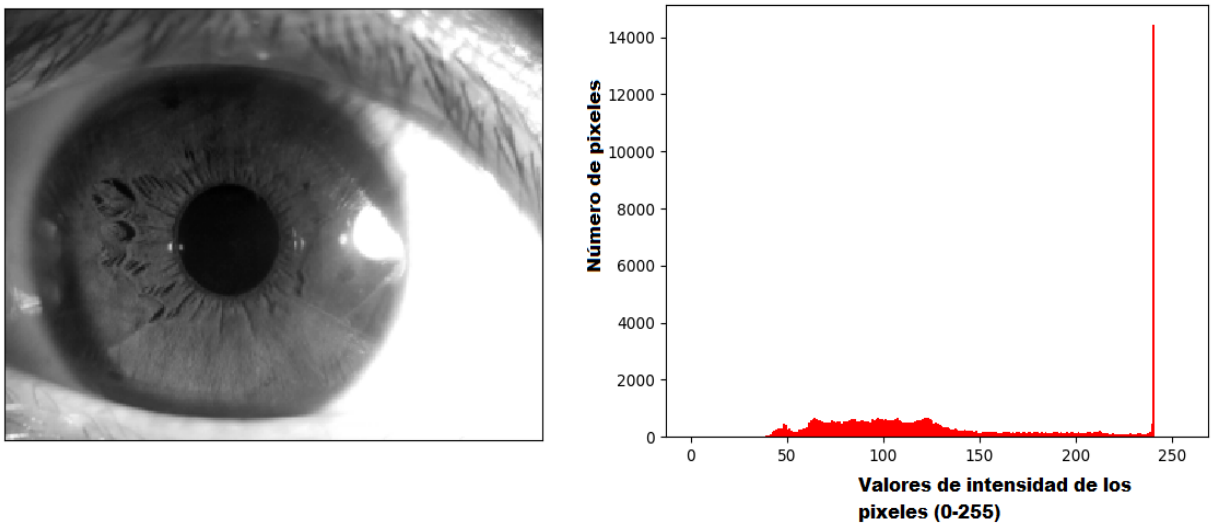
3.2 SOFTWARE DEL PROTOTIPO

A continuación, se describirán las diferentes etapas del algoritmo desarrollado para este prototipo además se explicarán brevemente las diferentes técnicas que fueron requeridas para este propósito.

3.2.1 Pre-procesamiento

Una vez adquirida la imagen de prueba, fue necesario hacer una ecualización adaptativa debido a lo encontrado en el histograma de la respectiva imagen como se observa en la Figura 5.

Figura 5. Imagen de prueba con su respectivo histograma.

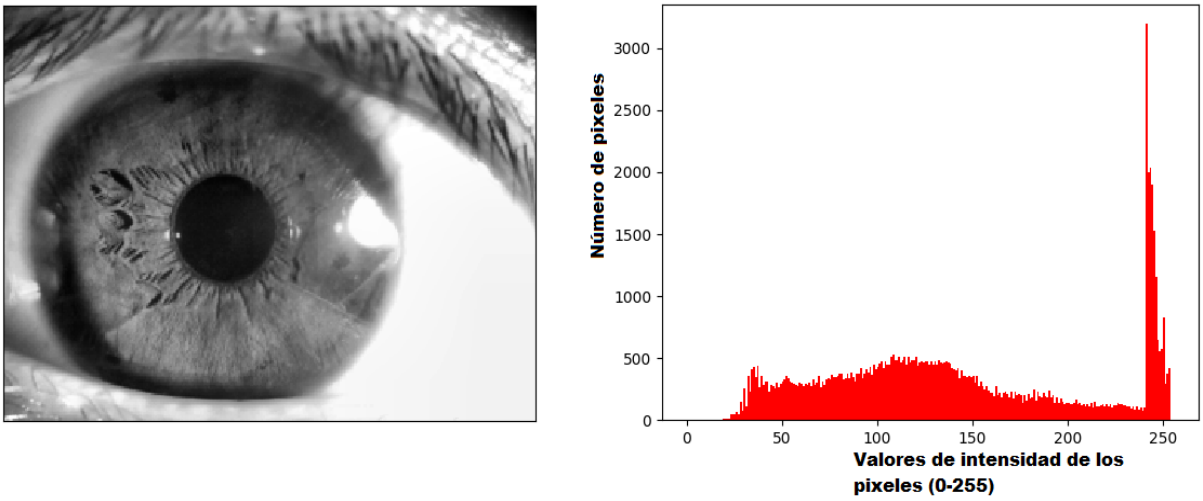


Se observa claramente en la Figura 5 que la imagen en la región comprendida por el iris resultó muy oscura y esto generaba errores en la estimación de los círculos, tanto de la pupila como del iris porque la intensidad en los pixeles era muy cercana como se observa en el histograma de la Figura 5 donde no había una buena

distribución en la intensidad de los pixeles de la imagen, lo que hacía muy difícil hacer una buena estimación del umbral para aislar la pupila y el iris.

La solución a esto fue aplicar un ecualizador adaptativo el cual consiste en estirar el histograma de una imagen por regiones y no de manera global como sucede con la ecualización normal de una imagen. Con este método de ecualización se evita perder información en valores de intensidad cercanos a 255 (en formato RGB corresponde al color blanco), ya que estos pueden presentar un exceso de brillo después de la ecualización global[27]. En la Figura 6 se puede observar claramente como la imagen es más clara y resalta detalles de la imagen, el histograma de la Figura 6 muestra como el contraste mejoró, realzó la región del iris y aunque no es perceptible al ojo humano hizo más oscura la región de la pupila lo que hace más fácil aislarla del resto de la imagen.

Figura 6. Imagen de prueba ecualizada con su respectivo histograma



3.2.2 Estimación de la pupila

Antes de describir el proceso de estimación de la pupila representado en la Figura 7 al que fue sometida cada imagen adquirida, se dará una breve explicación de cada técnica implementada y así entender con mayor claridad el propósito con el que se utilizó cada método.

- **Umbralización**: matemáticamente se define mediante la ecuación:

$$A(i,j)' = \begin{cases} 0 & si\ A(i,j) < U \\ L-1 & si\ \ A(i,j) \geq U \end{cases} \qquad \text{Ec. 1}$$

Donde A es la imagen original, A' es la imagen resultante luego de la umbralización, U es el umbral establecido, i indica las filas de la imagen y j las columnas. “Esta

[27] Yan Chai, Hum and Khin Wee, Lai and Maheza Irna, Mohamad Salim, 2014.

técnica permite separar el objeto de interés del resto de la imagen (fondo), obteniendo una imagen binarizada"[28].

- **Mediana: "**consiste en asignar a cada pixel de la imagen la mediana local resultante del elemento estructurante o pixeles alrededor del pixel de interés. Este es un filtro pasa bajo que ayuda a suavizar las imágenes eliminando detalles pequeños y ruido, esta técnica no es sensible a valores extremos por lo que es muy efectiva para eliminar el ruido de sal y pimienta" [29].

- **Erosión:** "es una operación morfológica en la cual se toma un elemento estructurante (*kernel*) de una dimensión y forma determinada que recorre la imagen original reduciendo el tamaño de los objetos que se encuentren en esta imagen de la siguiente manera: se erosionará el punto de estudio si alguno de los puntos del elemento estructurante coincide con un cero de alrededor del punto de estudio limitado por el tamaño del elemento estructurante" [30]. Vale la pena aclarar que si se erosiona un punto de estudio este tomará el valor de cero por ende la erosión agrando el objeto.

- **Dilatación**: "esta operación morfológica se basa en el efecto contrario al de la erosión, aumentando el tamaño de los objetos que se encuentren en la imagen original, de la siguiente manera: se dilatará el punto de interés si al menos uno de los puntos del elemento estructurante coincide con un uno, es decir pertenece al objeto. Para este caso cuando se dilata un punto de estudio este tomará el valor de uno"[31].
- **Detector de bordes Canny**: este algoritmo fue desarrollado por (OpenCV, 2015)[32] y consta básicamente de 4 etapas:

Etapa 1: consiste en la eliminación de ruido de la imagen con un filtro gaussiano 5x5.
Etapa 2: teniendo la imagen suavizada, la segunda etapa es filtrar la imagen con un filtro *sobel* para obtener la primera derivada en dirección horizontal y vertical para encontrar el gradiente del borde y la dirección de cada pixel.

Etapa 3: la tercera etapa realiza un escaneo completo de la imagen para eliminar píxeles no deseados que pueden no constituir un borde.

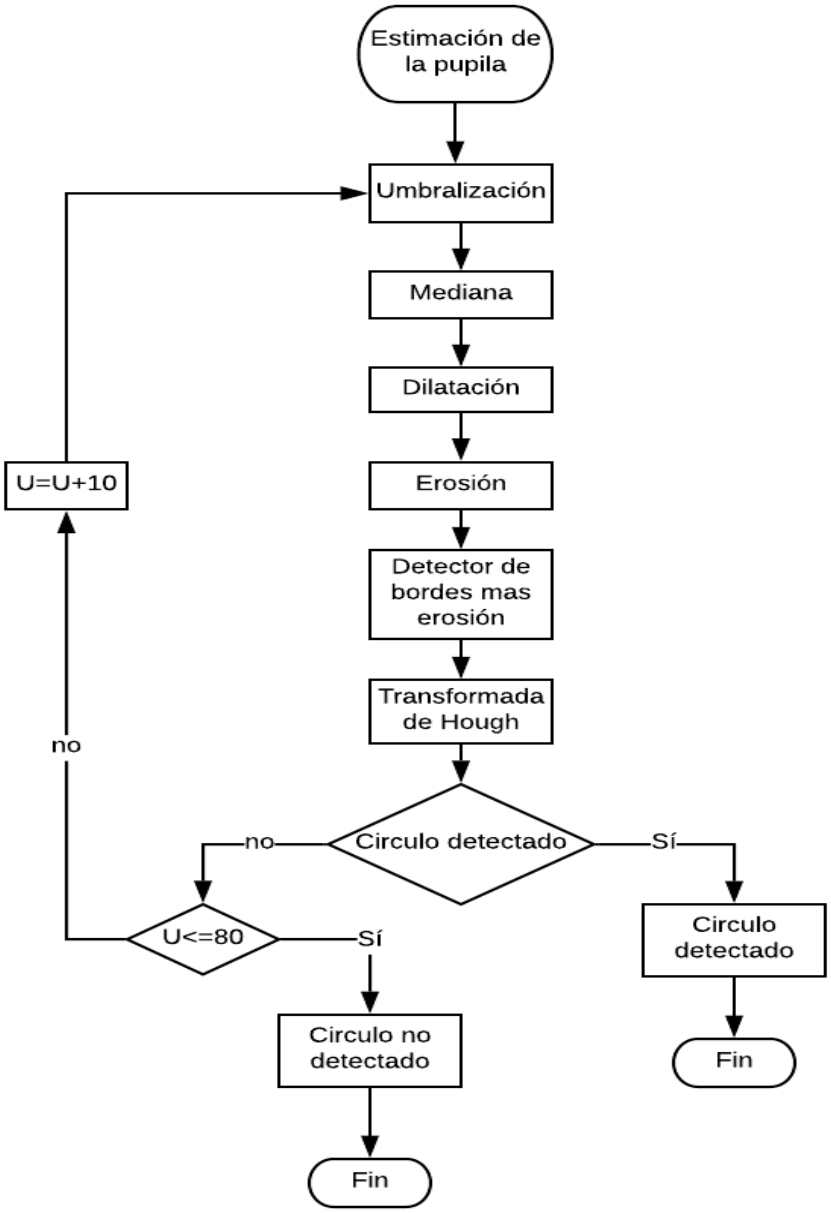
Etapa 4: esta es la etapa de decisión entre que bordes se consideran bordes reales y cuales no a partir de dos valores de umbral maxVal y minVal, donde los bordes con un gradiente de intensidad mayor a maxVal serán considerados bordes seguros y los que estén por debajo de minVal serán descartados y para clasificar los bordes

[28] Alegre, Pajares y De la Escalera, 2016, pág. 32
[29] Ibid.,pág.40
[30] Alegre, Pajares y De la Escalera, Op. Cit., pág. 81

[31] Ibid., pàg. 80.
[32] OpenCV, 2015.

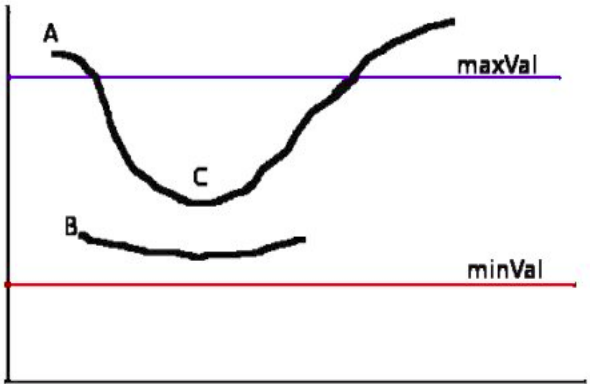
que se encuentran entre estos dos umbrales se tienen en cuenta su conexión con bordes seguros siendo considerados así bordes de lo contrario serán descartados[33].

Figura 7. Diagrama flujo en el proceso de estimación de la pupila.



[33] Parker, 2010.

Figura 8.Umbral de histéresis.



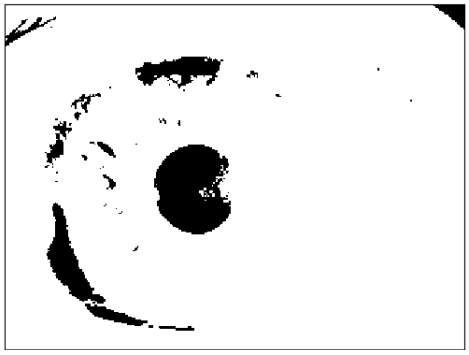
Fuente: Mordvintsev, A. y Abid K. (2013). “Canny Edge Detection”. [Figura]. Recuperado de https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html

- **Transformada de Hough**: este algoritmo es ampliamente utilizado en el procesamiento de imágenes para estimar ya sea líneas rectas, círculos y/o elipses. En este trabajo el interés fue determinar círculos de una imagen, donde el algoritmo se implementó de la siguiente manera: detección de bordes, extracción de curvas, estimación de radio de curvatura, encuentra el centro del círculo usando la acumulación en capas, estimación de radio y la recogida de pruebas para la comprobación del círculo[34] .

Una vez teniendo claro los conceptos básicos de las diferentes técnicas de procesamiento utilizadas en la estimación del círculo de la pupila, se dará una breve explicación de cómo fueron aplicados cada uno de estos algoritmos mostrando los resultados en cada paso. Cabe resaltar que todas estas técnicas de procesamiento fueron tomadas de la librería de código abierto *opencv-python*. Como primer paso se leyó la imagen para obtenerla en un formato matricial y así poder hacer las operaciones matemáticas con ella, seguidamente se aplicó la técnica de umbralización con el objetivo de separar la pupila del resto de la imagen, vale la pena aclarar que el contador que aumenta con la letra U en la Figura 7 indica el umbral que se aplica a la imagen, que inicia en 30 y se intenta hasta un umbral de máximo 80 donde se considera que esta imagen no fue apta para el proceso; el resultado se puede ver en la Figura 9.

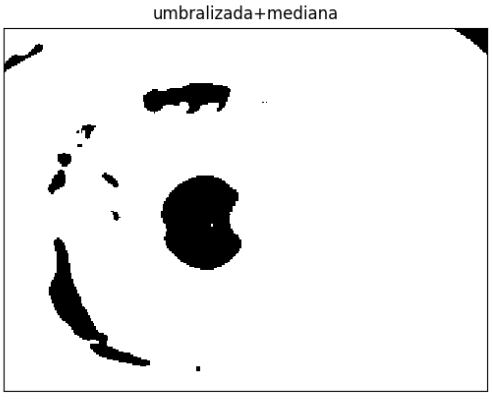
[34] Yao, Z and Yi, W, 2016

Figura 9. Umbralización de la imagen de prueba.



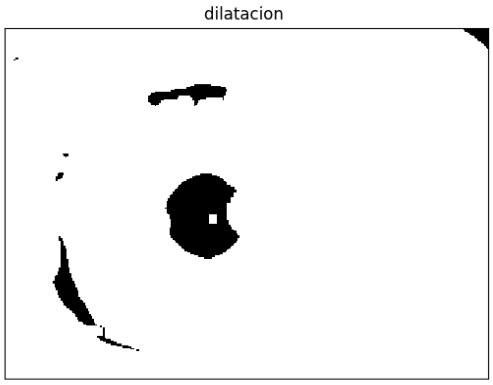
Como se observa en la Figura 9, la técnica implementada pudo separar la pupila del resto de la imagen, pero aun dejo ruido producto de las pestañas que puede ocasionar la detección de círculos erróneos, es decir círculos que no corresponden exactamente al borde de la pupila. Por esta razón fue necesaria hacer una limpieza de ruido en la imagen, para este propósito se utilizaron dos técnicas seguidamente; Filtro de mediana y la dilatación. El filtro de mediana ayuda a eliminar el ruido de “sal y pimienta" que se presenta en la Figura 9 y la dilatación ayuda a borrar el ruido “más pesado" (el ruido que no puede eliminar el filtro de mediana"), esto se puede apreciar en las Figuras 10 y 11. Es importante avisar al lector que para este caso la dilatación, dilató el fondo de la imagen (fondo blanco) más no el objeto de color negro por eso su actuación fue la eliminación de las pequeñas manchas negras (ruido)[35].

Figura 10.Mediana de la imagen de prueba procesada



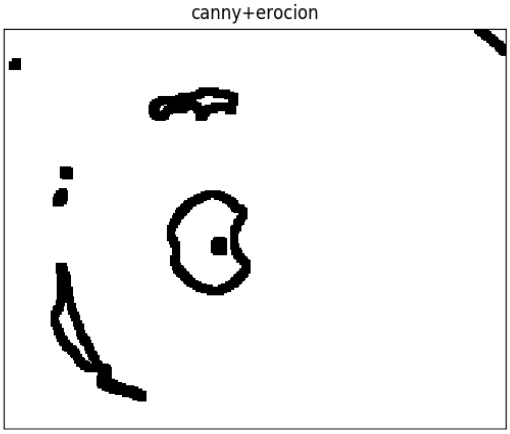
[35] OpenCV, Morphological Transformations, 2014

Figura 11.Dilatación de la imagen de prueba procesada



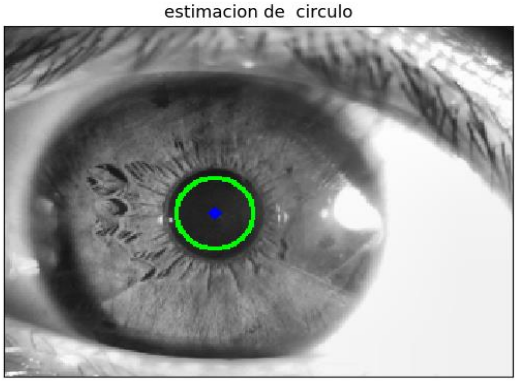
Después de hacer el proceso de limpiado de la imagen, el siguiente paso fue la detección de bordes y para esto se implementó el detector de bordes *canny* de la librería *opencv*, seguido a este se aplicó la técnica de erosión para que los bordes detectados quedaran bien robustos y no tener complicaciones en la estimación del círculo de la pupila, el resultado de este procedimiento se observa en la Figura 12.

Figura 12. Detección de bordes aplicado a la imagen procesada dilatada.



Una vez obtenido los bordes de interés, el siguiente paso fue la estimación del círculo mediante la transformada de *Hough*, donde se tuvo que configurar los parámetros a la función correspondiente, con el fin de evitar falsas estimaciones del círculo de la pupila, ya que como se observa en la Figura 12, por ejemplo, dentro del círculo de interés se encuentra otro círculo de menor tamaño que se da por la reflexión de la iluminación en el momento de la captura de imagen, entonces se debe condicionar la función para que descarte los círculos estimados con radio pequeño, para este caso un radio menor a 10 posiciones de pixel. En la Figura 13 se observa la correcta estimación del círculo correspondiente a la pupila.

Figura 13.Estimación de la pupila



3.2.3 Estimación del iris

En esta parte de procesamiento, se utilizaron algunas técnicas también utilizadas en la estimación de la pupila, por esta razón en este ítem solo se darán las definiciones básicas de las técnicas utilizadas en esta sección que no fueron implementadas en la sección anterior y el proceso de esta etapa es mostrado en la Figura 14.

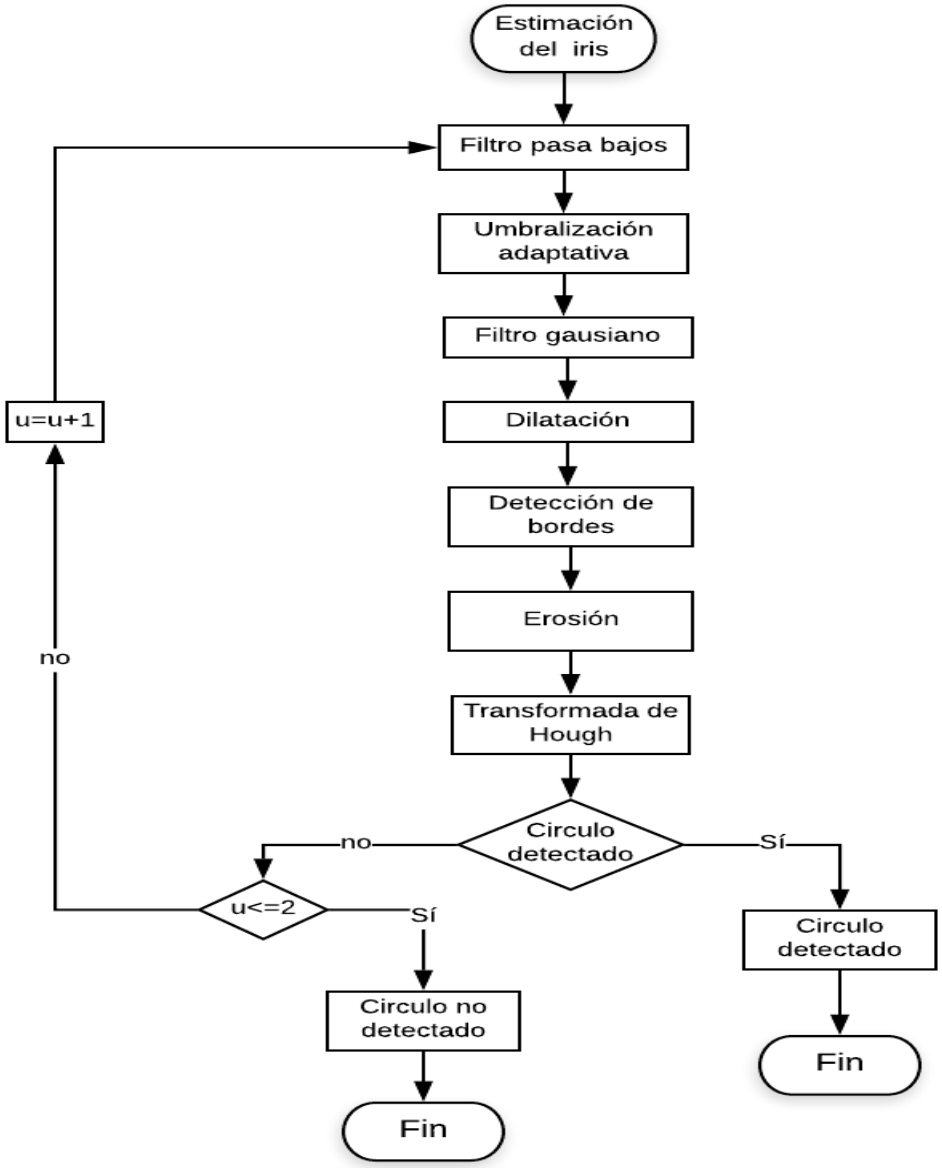
- **Filtros pasa bajos**: las imágenes se pueden considerar como señales bidimensionales las cuales se pueden filtrar con filtros pasa-altos (HPF) y pasa-bajos (LPF). Los filtros pasa-altos sirven para detectar bordes y los filtros pasa-bajos sirven para suavizar y difuminar imágenes. Dentro de los filtros pasa-bajos se encuentran los siguientes tipos:

- **Filtro Gaussiano**: este filtro calcula un promedio ponderado que tiene como objetivo de que un píxel determinado de la imagen se vea como sus vecinos, por lo tanto, el promedio nombrado da mayor peso a los pixeles que se encuentran cerca al punto de estudio y no lo hace así con los pixeles más distantes suprimiendo así el ruido[36].

- **Filtro de convolución 2D**: este filtro pasa un *kernel* determinado sobre la imagen, durante el proceso el *kernel* se ubica encima de cada píxel y obtiene el promedio de los píxeles que rodean al píxel de interés siendo este reemplazado por el promedio.

[36] David, Forsyth and Jean, Ponce, 2003

Figura 14. Diagrama de flujo de la estimación del iris.

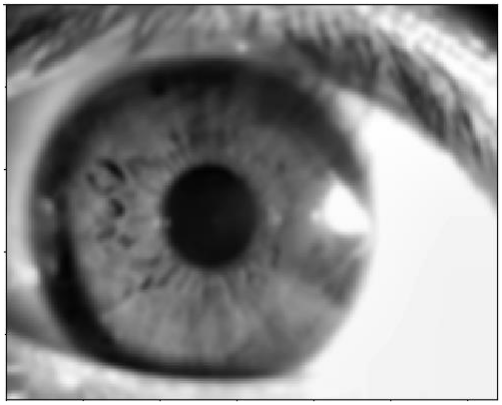


- **Umbralización adaptativa**: mientras que en la umbralización simple se usaba un umbral global, aquí se utilizan diferentes umbrales que el algoritmo calcula para regiones pequeñas de la imagen que pueden verse afectadas principalmente por las condiciones de iluminación.

Luego de aclarar brevemente las técnicas utilizadas para la estimación del iris, se procede a mostrar cómo se estructuro el algoritmo y el respectivo resultado en cada caso. Nuevamente se leyó la imagen (imagen ecualizada en el pre-procesamiento) para obtenerla en un formato matricial, seguidamente se le aplicó una serie de filtros

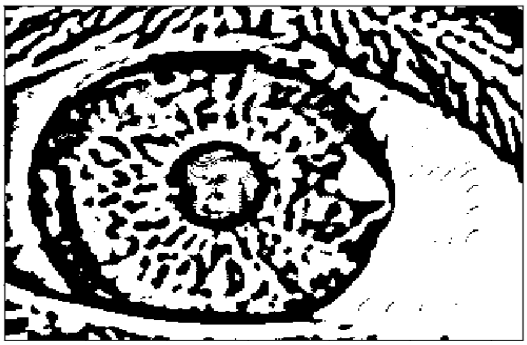
pasa bajos (filtro gaussiano y filtro de convolución 2D) con el objetivo de suavizar la imagen y eliminar el ruido presente en esta como se observa en la Figura 15.

Figura 15. Imagen de prueba suavizada con los filtros pasa bajos



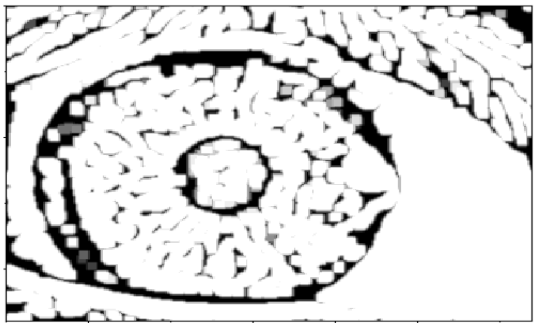
En el paso anterior también se consiguió algo muy importante y es que los filtros de pasa bajo también hacen homogénea la imagen por zonas lo cual es crucial para el siguiente paso evitando dejar regiones heterogéneas que pueden generar umbralizaciones no deseadas entorpeciendo la estimación del iris. A continuación, a la imagen se le aplicó una umbralización adaptativa para segmentar el iris y se mostrará el resultado suavizando la imagen en la Figura 16.

Figura 16.Umbralización adaptativa de la imagen de prueba suavizada.



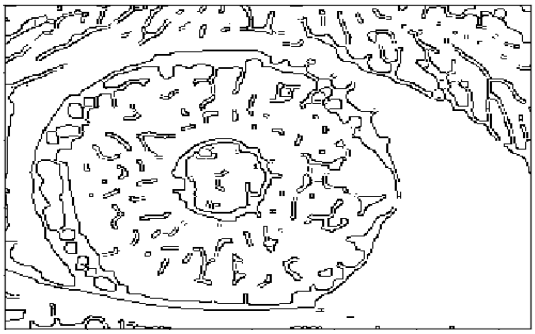
Aunque como se observa en la Figura 16 hubo una buena segmentación de la región del iris, pero aún sigue presentando ruido la imagen por esto fue necesaria aplicar un filtro gaussiano junto con una dilatación para de esta forma eliminar la mayor cantidad de pixeles que generan conflicto en la estimación del círculo, el resultado de este paso es mostrado en la Figura 17.

Figura 17. Imagen suavizada con filtro gaussiano más dilatación



La imagen obtenida en la Figura 17 se considera apta aplicarle el detector de bordes *canny* porque el borde externo del iris ya se ve definido y lo más importante, se eliminó una gran cantidad de ruido evitando la estimación de bordes innecesarios. El resultado del detector de bordes *canny* se presenta en la Figura 18.

Figura 18. Detección de bordes aplicado a la imagen de prueba suavizada

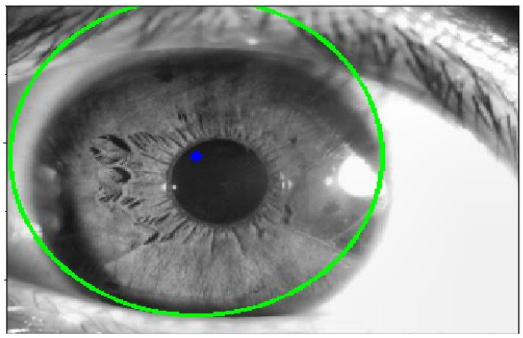


Para aplicar la transformada de Hough es necesario tener una imagen con bordes gruesos y bien definidos y no como los que presenta la Figura 18 donde hay bordes finos por lo que fue necesario aplicar una erosión a la imagen dando mayor grosor a los bordes obteniendo al tiempo un resultado contraproducente y fue que el ruido también aumentaba su grosor por lo que fue necesario buscar un punto de equilibrio con las técnicas de filtro gaussiano y dilatación, la Figura 19 evidencia el resultado del proceso.

Figura 19.Imagen procesada erosionada

Después de erosionar la imagen se hicieron reiteradas pruebas y con diferentes imágenes aplicándoles la transformada de *Hough,* donde se obtuvieron estimaciones del radio y centro del iris, en la Figura 20 se ve un ejemplo claro de esto.

Figura 20.Estimación del iris



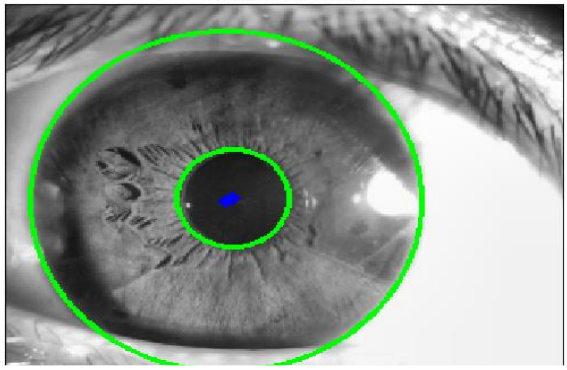
La estimación del iris se ve afectada por las sombras y reflexiones presentes en la imagen, en especial la sombra presente en el límite del iris con la esclerótica porque produce falsos bordes que al ser erosionados produce espacios en blanco que afectan gravemente a la transformada de *Hough* puesto que esta reconoce los círculos con los bordes más gruesos. Una vez definidas las técnicas para la correcta estimación del círculo de la pupila y del iris, se juntaron ambos métodos para de esta forma determinar únicamente la región comprendida por el iris, esta unificación se realizó mediante la implementación de una clase[37], la cual da como resultado la estimación de ambos círculos.

Se decidió presentar el resultado de la estimación de la pupila e iris con una imagen en la que se obtenían resultados un poco imprecisos con respecto a la segmentación de la pupila y el iris, pero no siempre es así, con algunas imágenes

[37] Gonzàles Duque, 2003

se obtenían segmentaciones con muy buena precisión como se observa en la Figura 21.

Figura 21. Estimación correcta del iris y la pupila.



3.2.4 Normalización de la imagen

La normalización de la imagen consiste en convertir las coordenadas cartesianas (x, y) de esta, en coordenadas polares (r, θ).

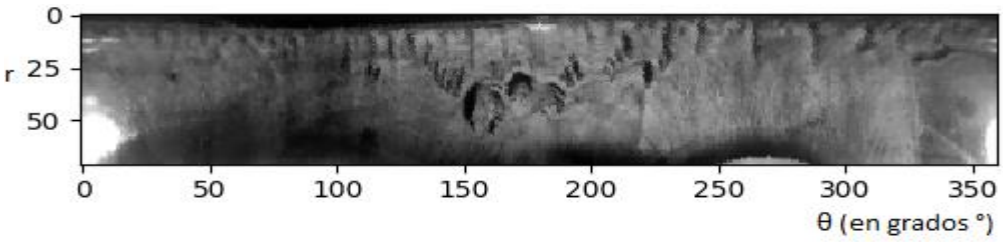
Esto matemáticamente se puede expresar mediante la ecuación 2[38].

$$I\big(x(r,\theta), y(r,\theta)\big) \rightarrow (r,\theta) \qquad \text{Ec. 2}$$

Para este caso es muy útil pues una vez obtenidos el centro y los radios de los círculos correspondientes a la pupila y el iris, se puede extraer fácilmente la región comprendida por el iris. Con este método se estandarizan todas las imágenes a un mismo tamaño para luego poder ser comparadas entre sí ya que puede haber diferencias entre los tamaños de los ojos de las personas.

El resultado de una imagen normalizada es ilustrado en la Figura 22.

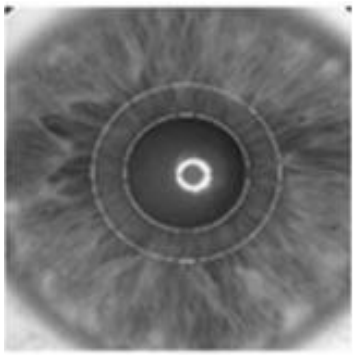
Figura 22. Normalización de la imagen de prueba



[38] Tossy, Thomas and Anu, George and Indira, Devi, 2016

Los patrones más complejos se encuentran en la zona llamada "*zigzag collarette*" del iris[39] como se evidencia en la Figura 23, además de esto esta zona es la menos afectada por las pestañas y con el propósito de trabajar con la menor cantidad de información posible, después de muchas pruebas se estimó un radio con origen un poco afuera de la pupila y con fin en la zona "*zigzag collarette*".

Figura 23. Área del *zigzag collarette*.



Fuente: Rai, H. y Yadav, A. (2014). "Iris recognition using combined support vector machine and Hamming distance approach". [Figura].

En esta etapa al hacer diferentes pruebas y analizar los diferentes resultados de la estimación de la pupila y el iris se decidió que para la normalización no se tomaría como punto de inicio la longitud del radio de la pupila estimada por el algoritmo, porque, en ocasiones esta longitud era más corta que la medida real de la pupila lo que ocasionaba que se incluyera parte de esta, lo que es considerado ruido así que luego de varios intentos se definió sumarle al punto de inicio de la normalización 10 unidades de pixel con respecto al radio de la pupila estimado evitando incluir zonas correspondientes a la pupila en el proceso de normalización, este proceso es necesario, a menos que, el algoritmo de segmentación de la pupila tuviera una efectividad del 100%, es decir, el circulo estimado por el algoritmo correspondiente a la pupila sea exactamente igual al círculo real de la pupila.

Como se mencionó anteriormente, los patrones más complejos del iris se encuentran en la zona "*zigzag collarete*" y con el fin de utilizar la mínima información posible se hicieron pruebas para determinar si era suficiente con una magnitud de radio de 25 unidades de pixel para la etapa de clasificación la cual resultó ser óptima por el hecho de que los algoritmos de clasificación lograron distinguir cada usuario. También se definió el rango de grados tomados desde 90 a 250 grados puesto que es la región menos afectada por las reflexiones y obviamente donde se observan más claramente los detalles. La imagen normalizada queda como se observa en la Figura 22 donde se observa que esta contiene solamente la información necesaria es decir no presenta ruido.

[39] Himanshu, Rai and Anamika, Yadav, 2014

3.2.5 Extracción de características

Esta es la etapa de minería de datos porque se necesita extraer los datos de la imagen que representan las características más relevantes y necesarias para poder diferenciar con alta eficiencia un iris de otro.

Los métodos que se probaron en la siguiente etapa de clasificación, fueron el filtro de *Gabor*[40] y la transformada de *Wavelet*[41], donde se optó por seleccionar como método de extracción de características la transformada de *Wavelet* porque el *filtro de Gabor* no generó una correlación de los datos y por lo tanto no fue posible que los métodos de clasificación convergieran o dicho de otra manera los métodos de clasificación no aprendieron a separar correctamente cada una de las imágenes de entrenamiento. Mientras que con la transformada de *Wavelet* si hubo una convergencia, aplicando la transformada hasta el segundo nivel de descomposición con la función de ventana *Haar* seleccionando como espacio de trabajo la ventana del filtro pasa bajo-bajo[42], se llegó a esta selección de parámetros luego de realizar pruebas con diferentes niveles obteniendo que este fue el de mejor rendimiento para que los algoritmos de clasificación categorizaran correctamente a los usuarios registrados, los resultados se mostrarán en la siguiente etapa.

3.2.6 Clasificadores con inteligencia artificial

Son algoritmos pertenecientes al campo de la inteligencia artificial, lo que implica que tienen la capacidad de “aprender" en base a un conjunto de datos que se han procesado debidamente logrando así que puedan reconocer patrones y tomar una decisión acertada. Para este proyecto se escogieron dos técnicas de *machine learning* ampliamente utilizadas en el reconocimiento de iris; las máquinas de soporte vectorial y las redes neuronales.

Para el método de las máquinas de soporte vectorial[43],[44], se escogió el parámetro de penalización de error en 1000 denotado como C, este parámetro debe ser considerado cuando se entrena la SVM con un núcleo de base radial (RBF) que fue el utilizado en este método. Además a esto, se debe definir un factor gamma el cual define cuánta influencia tiene un solo ejemplo de entrenamiento, por último se definió una tolerancia de criterio de parada, es decir hasta que valor del error debe llegar el entrenamiento para que el método deje de entrenar[45]. Este método se

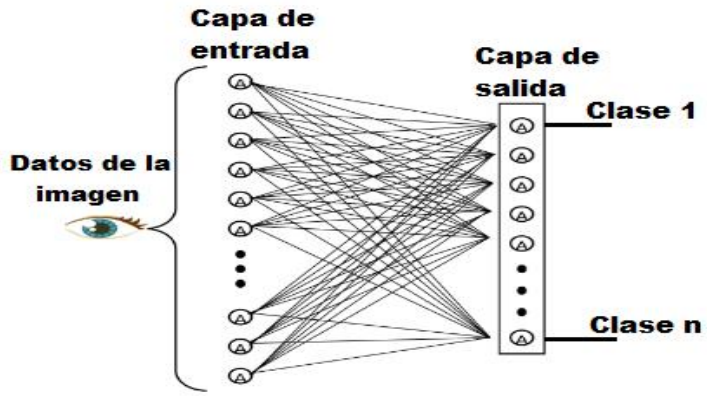
[40] Libor, Masek, 2003
[41] Samir, Kouro and Rodrigo, Musalem, 2002
[42] Gregory R. Lee, Ralf Gommers, Filip Wasilewski, Kai Wohlfahrt, Aaron O'Leary , 2019
[43] (Aurélien, 2017)
[44] (Alegre, Pajares y De la Escalera, 2016)
[45] Chih-Chung, Chang and Chih-jen, Lin, 2002.

implementó a través de la librería de código abierto disponible para *Python sklearn*[46].

La implementación para la red neuronal[47] consiste en una arquitectura de red con dos capas conectadas mediante los pesos[48] de cada capa, una de entrada con n*10 unidades y una de salida con n unidades, donde n es un número entero que corresponde al número de usuarios por registrar. Las funciones de activación para las unidades de la primera capa, fue *RELU* (rectifed linear unit) la cual es una de las más implementadas para la clasificación de imágenes ya que obtiene un rendimiento ligeramente mayor que la función *sigmoide*. La capa de salida es una capa donde la función de activación es *softmax* donde está, permite obtener un vector de salida con n puntuaciones de probabilidad que suman uno. Cada nodo en la capa de salida tiene una puntuación de probabilidad que indica que tan probable es que la imagen de entrada pertenezca a cada una de las clases ya registradas. En la Figura 24 se observa la estructura topológica de esta red.

El optimizador utilizado fue el *ardamax*[49] con una taza de aprendizaje lr=0.002. El parámetro de cálculo de pérdidas utilizado fue la clase *sparse categorical cross entropy,* la cual es adecuada cuando los objetivos del modelo de entrenamiento están en formato categórico, por ejemplo: en un conjunto de datos con diez clases, el objetivo para cada muestra debe ser un vector de 10 posiciones que sea todos ceros, excepto un 1 correspondiente a la clase de la muestra. La implementación de este modelo de red neuronal fue desarrollada sobre *tensorflow*, que es una librería de código abierto creada por google para implementar algoritmos de inteligencia artificial[50].

Figura 24. Topología de la red neuronal implementada.



[46] Lars Buitinck and Gilles Louppe and Mathieu Blondel and Fabian Pedregosa and Andreas Mueller and Olivier Grisel and Vlad Niculae and Peter Prettenhofer and Alexandre Gramfort and Jaques Grobler and Robert Layton and Jake VanderPlas and Arnaud Joly and Bri, 2011.
[47] (Pedro Ponce, Cruz, 2010)
[48] (Toral Barrera)
[49] (Diederik P., Kingma and Jimmy Lei, Ba, 2014)
[50] Attribution, 2017.

3.2.7 Diseño de software

En esta etapa se encuentra el diagrama de flujo de la interfaz gráfica que contiene un algoritmo que permite el manejo de la cámara conectada a la *raspberry pi* y seguidamente los correspondientes casos de uso.

Figura 25. Diagrama de flujo para la interfaz de la raspberry

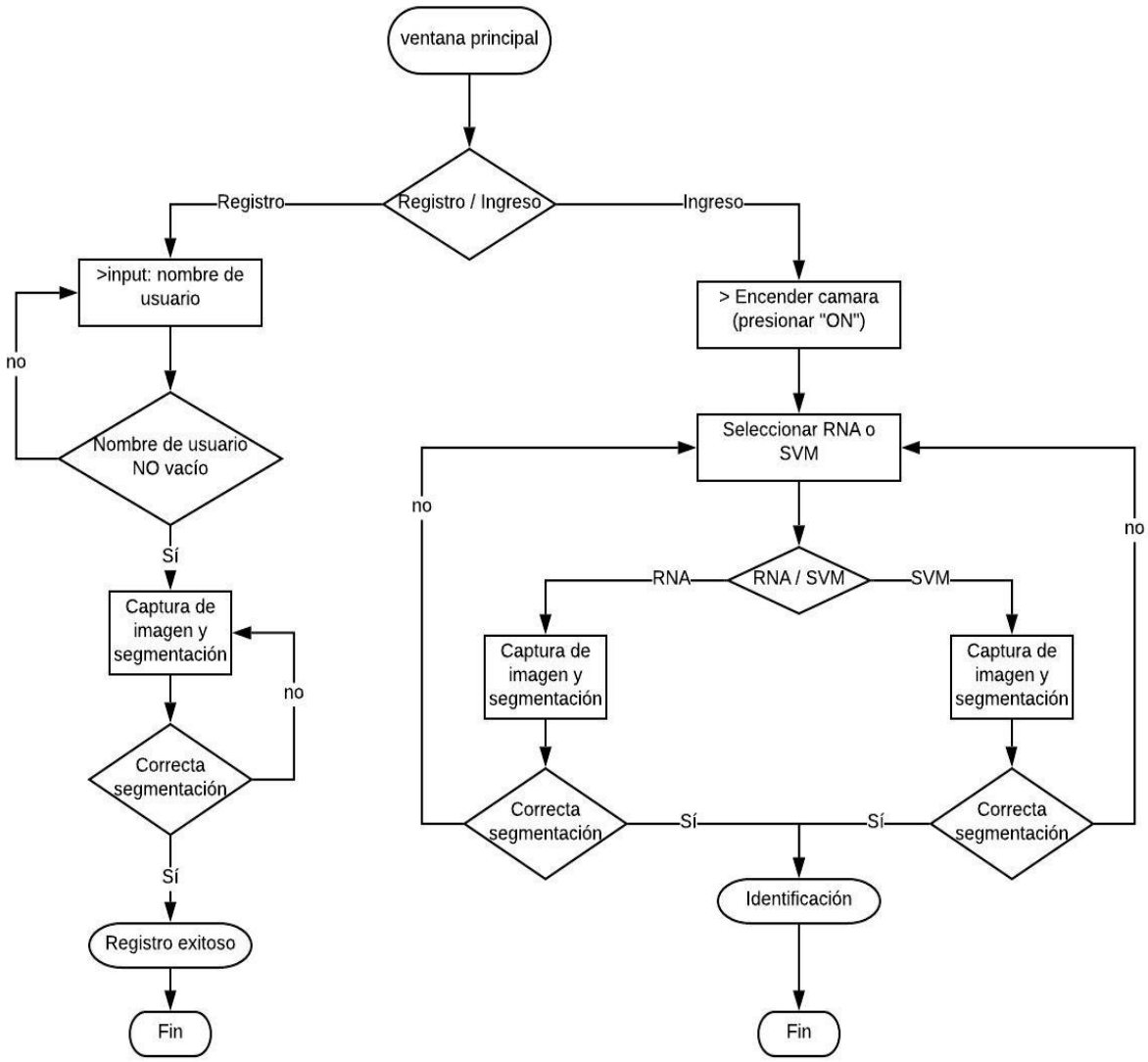
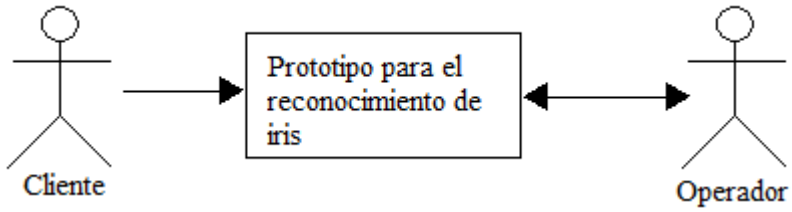


Figura 26. Actores que interactúan con el prototipo

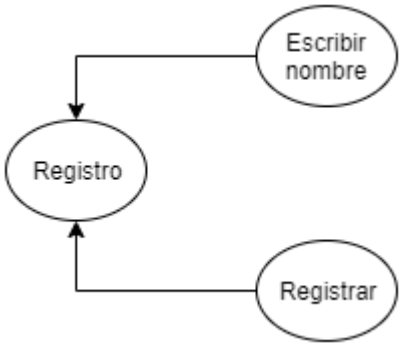


En la Figura 26 se muestra de manera general los actores que interactúan con el prototipo.
Luego, se tiene un cliente al cual se le toma la fotografía del iris y un operador que controla las opciones de verificar el ingreso o registrar el cliente en caso de que no esté registrado previamente como se observa en la Figura 27.

Figura 27. Casos de uso para el cliente y el operador

Si previamente el operador selecciono la opción Registro, debe asignarle el nombre al cliente y registrarlo como se observa en la Figura 28.

Figura 28. Caso de Registro para el operador



En el caso de Ingreso de la Figura 29, el operador puede encender o apagar la cámara según las circunstancias, además, tiene dos opciones para verificar el ingreso del cliente ya sea por redes neuronales o máquinas de soporte vectorial (MSV).

Al final de los casos de Registro o Ingreso, el prototipo muestra un mensaje informando el resultado del proceso como se observa en la Figura 30.

Finalmente se muestra en la Figura 31 un diagrama completo de los casos de uso con este prototipo.

Figura 29. Caso de Ingreso para el operador

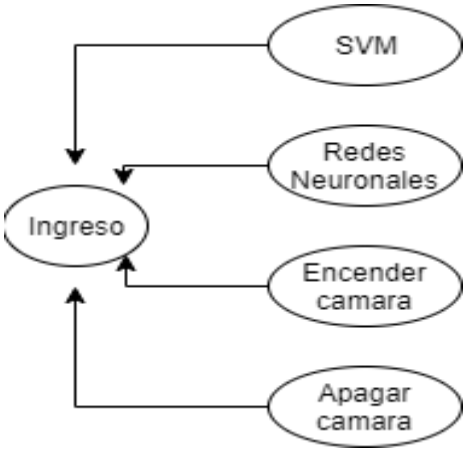


Figura 30. Visualización de los mensajes de Ingreso y Registro

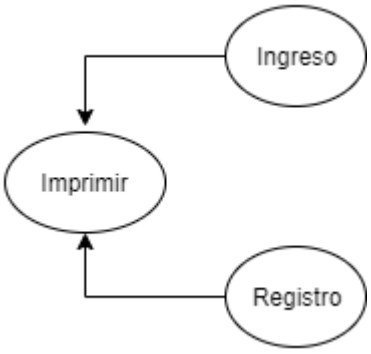
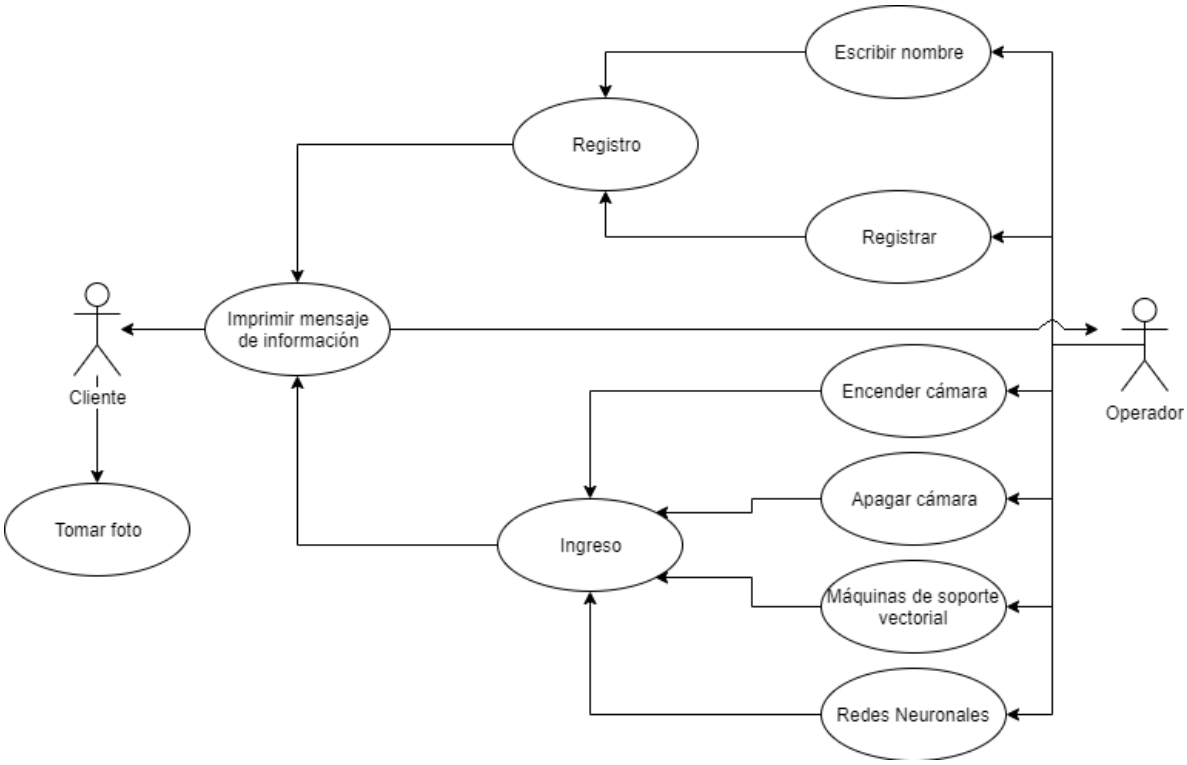


Figura 31. Diagrama completo de los casos de uso



4. CONDICIONES DE OPERACIÓN

Las fotografías deben ser tomadas con una buena fuente de iluminación tipo cálida para que de esta forma puedan ser contrarrestados los efectos de las reflexiones en el ojo producto de la iluminación exterior que crean interferencia en el proceso de extracción de características. Una vez ajustado el enfoque de la cámara, este debe permanecer estático ya que todo el algoritmo está configurado para su correcto funcionamiento a una distancia que determina el mismo enfoque.

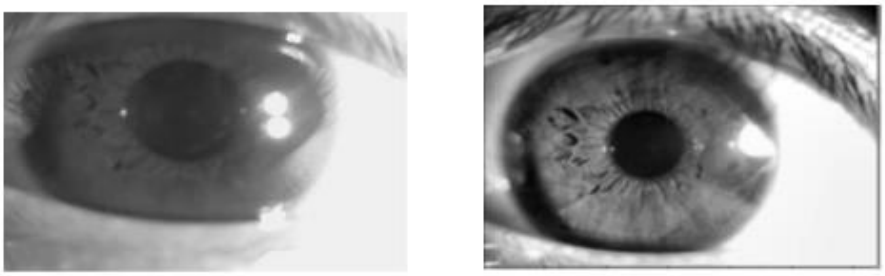
Es necesario crear una cuenta en drive que sea dedicada a almacenar las imágenes y modelos correspondientes al prototipo, un tema muy importante a tener en cuenta es que si se desea convertir el prototipo en un producto comercial es necesario encriptar las plantillas de los iris porque esta información es personal además es necesario tener disponible una señal wifi para conectar la *raspberry* y poder controlarla desde un computador portátil que contenga la interfaz que permite descargar las imágenes contenidas en el drive, actualizar los modelos y subirlos nuevamente para que la *raspberry* los actualice con los nuevos usuarios.

5. ANÁLISIS DE RESULTADOS

5.1 ILUMINACIÓN

La iluminación en este proyecto fue un factor clave para el correcto desarrollo de las etapas comprendidas en este, puesto que están interconectadas en cascada siendo la salida de una etapa la entrada de la etapa posterior. Se hicieron muchas pruebas con diferentes tipos de iluminación led y aunque eran suficientes para que el algoritmo de segmentación funcionara correctamente no era posible resaltar todos los detalles del iris por lo que no era posible obtener suficiente información para diferenciar entre un usuario y otro por lo que con esta iluminación el prototipo no cumplía con su objetivo. La iluminación con mejor desempeño fue de tipo cálida porque este si resaltó los detalles suficientes del iris para que el prototipo pudiera tener un buen desempeño en el proceso de reconocimiento de usuarios. En la Figura 32 se observa una imagen con iluminación led blanca y una imagen con la iluminación tipo cálida.

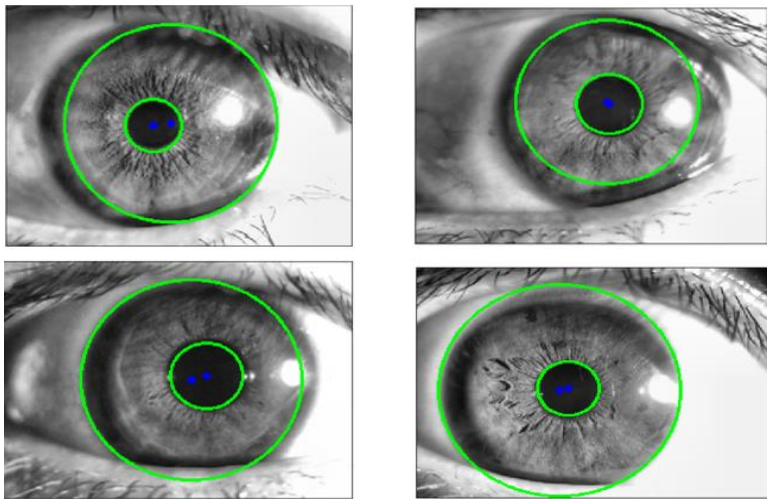
Figura 32. Iluminación led blanca vs Iluminación cálida



5.2 SEGMENTACIÓN

Debido a las limitaciones que presenta la cámara por su enfoque manual, fue tedioso y un poco complicado situar de manera adecuada a los usuarios, dado que si la imagen es tomada a una distancia donde la cámara no enfoca correctamente, esta queda distorsionada y el proceso de segmentación es afectado haciendo falsas estimaciones de círculos diferentes al de la pupila y el iris afectando negativamente el desempeño de la etapa de reconocimiento. La Figura 33 muestra el resultado de segmentación para 4 usuarios diferentes.

Figura 33. Segmentación en algunos usuarios.



Por el lado de la base de datos de iris CASIA, diseño su propia cámara de iris de primer plano, en la cual, entre sus características resalta su matriz circular de LED NIR, con flujo luminoso adecuado para imágenes de iris lo que le permite capturar imágenes muy claras del iris y estudiar características detalladas de textura, este factor tecnológico influye notoriamente en el porcentaje de acierto que pueda tener un algoritmo cuyo propósito sea la identificación de personas, por esta razón son ampliamente utilizada las imágenes provenientes de esta base de datos en los proyectos de investigación . En la Figura 34 se muestra la cámara que utiliza CASIA y en la Figura 35 algunas imágenes capturadas con esta cámara.

Figura 34. Captura del iris con la cámara especial de CASIA

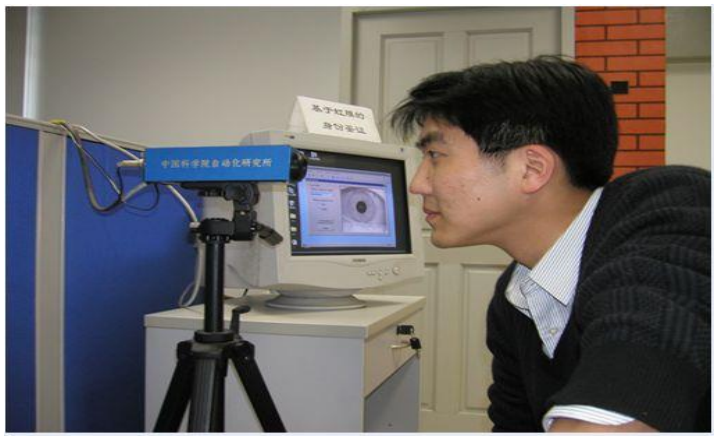
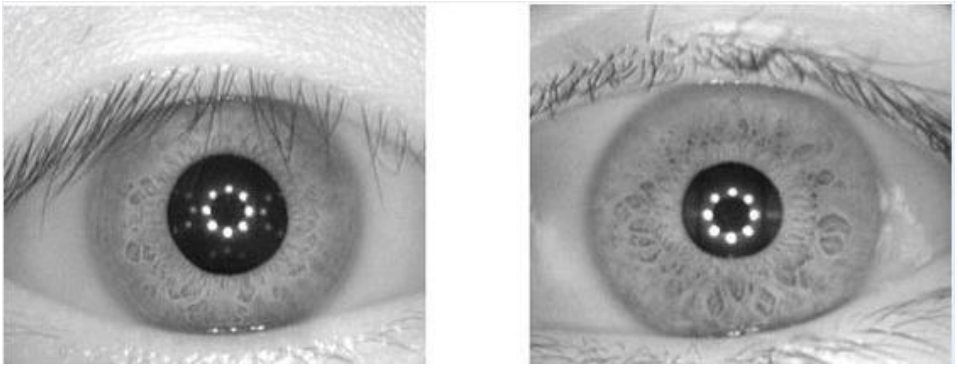


Figura 35. Ejemplo de imágenes de iris en CASIA



5.3 TIEMPOS DE EJECUCIÓN

A continuación, se mostrará en la tabla 2 los tiempos promedios de ejecución (promediado de 10 medidas) por parte de las redes neuronales y las máquinas de soporte vectorial, donde se observa una diferencia de 14.53 segundos en el proceso de entrenamiento entre ambos métodos siendo más favorable para las máquinas de soporte vectorial, pero en el proceso de validación de usuarios, las redes neuronales presentaron un tiempo menor que las máquinas de soporte vectorial teniendo una diferencia de tiempo de 0.3 segundos.

Tabla 2. Tiempos medidos para ambos métodos

Nombres de los métodos	Tiempo de validación de usuario (s)	Tiempo de entrenamiento del modelo (s)
Redes Neuronales	3.87	14.55
Máquinas de soporte vectorial	4.17	0.02

5.4 VALIDACIÓN DE RESULTADOS

Para el proceso de entrenamiento en este proyecto se tomaron 5 fotos por persona de las cuales 4 fueron asignadas para el entrenamiento y 1 para la evaluación, el tamaño de las imágenes fue normalizado obteniendo una plantilla de 25x160 pixeles que luego fue reducida en la extracción de características mediante el segundo nivel de la transformada de *Wavelet Haar* obteniendo una plantilla de 7x40 pixeles. Mientras que en el *paper* referenciado[51], se utilizaron 9 fotos por persona de las cuales 3 fueron seleccionadas para el entrenamiento y 6 para la evaluación, el

[51] Saminathan, K and Chakravarthy, T and Devi, M Chithra, 2015.

tamaño de sus imágenes normalizadas es de 20x240 pixeles que luego fueron reducidas a imágenes de 10x240 donde se registraron 50 usuarios, los resultados de los ambos trabajos se presentarán en las tablas 2, 3 y 4 respectivamente.

Tabla 3. Resultados con personas registradas.

Método	Numero de intentos	Taza de falsos rechazos (%)	Taza de aceptaciones verdadera (%)	Cruce de usuarios (%)
Redes neuronales	48	16.67	83.33	0
SVM	48	20.83	77.09	2.08

Tabla 4. Resultados con personas no registradas.

Método	Numero de intentos	Taza de verdaderos rechazos (%)	Taza de falsas aceptaciones (%)
Redes Neuronales	8	100	0
SVM	8	100	0

Tabla 5. Exactitud del trabajo referenciado.

<table>
<tr><td colspan="6">Método propuesto</td></tr>
<tr><td>1.</td><td colspan="3">Distancia de Hamming</td><td rowspan="11">Imagen del iris (Collarette, criptas, surcos entre la región baja a media)</td><td>76.8</td></tr>
<tr><td>2.</td><td colspan="3">Patrón binario local</td><td>83.3</td></tr>
<tr><td>3.</td><td colspan="3">Red Neuronal feed forward</td><td>87.0</td></tr>
<tr><td rowspan="9">4.</td><td rowspan="9">SVM (Características espectrales)</td><td rowspan="3">Lineal</td><td>SMO</td><td>97.0</td></tr>
<tr><td>QP</td><td>96.0</td></tr>
<tr><td>LS</td><td>97.5</td></tr>
<tr><td rowspan="3">Polinomial</td><td>SMO</td><td>96.0</td></tr>
<tr><td>QP</td><td>95.0</td></tr>
<tr><td>LS</td><td>97.2</td></tr>
<tr><td rowspan="3">Cuadrático</td><td>SMO</td><td>98.0</td></tr>
<tr><td>QP</td><td>97.0</td></tr>
<tr><td>LS</td><td>98.5</td></tr>
</table>

Aunque las tablas evidencian que el trabajo referenciado obtuvo una mejor eficiencia vale la pena resaltar que ellos trabajaron con la base de datos *CASIA* donde se encuentran imágenes con detalles más definidos y donde se evidencia que utilizaron un método de iluminación más sofisticado porque el reflejo de su

iluminación queda contenido en la zona de la pupila que no es de interés para estos proyectos además de que ellos utilizaron más imágenes para el realizar el test del algoritmo, también es importante decir que cada imagen de ellos tenía más pixeles, estos dos factores son cruciales y hacen que su rendimiento sea superior.

El método de entrenamiento tanto para las redes neuronales como para las SVM fue la separación del conjunto de datos en 80% para el conjunto de entrenamiento y el 20% restante para el conjunto de prueba con el fin de que el sistema tendiera a estar sobre ajustado, puesto que uno de los principales objetivos del prototipo es tener en lo más mínimo el porcentaje de falsas aceptaciones y no aceptar a personas que no estén registradas. El optimizador escogido para el entrenamiento de las redes neuronales fue *Adamax*, este obtuvo el mejor desempeño en cuanto a cantidad de unidades de activación en la red y exactitud en el reconocimiento de cada usuario. Por el lado de las SVM los parámetros utilizados para el mejor desempeño fueron los siguientes:

- C: parámetro de penalización del término del error (1000).
- tol: tolerancia para el criterio de parada (0.01).
- gamma: factor de taza de aprendizaje (Automatico generado por la libreria).
- núcleo o kernel: rbf.

5.5 COMPARACIÓN DE PRECIOS

A continuación, en la tabla n se mostrará el precio del prototipo desarrollado en comparación con algunos sistemas existentes en el mercado para el reconocimiento de iris donde los precios fueron tomados del sitio web *folcrum biometrics.*

Tabla 6. Precios de dispositivos de reconocimiento de iris

Referencia / nombre	**Precio (COP)**
Prototipo desarrollado	300000
IrisShield-USB MO 2120	620000
IrisShield-USB MO 2120 con EVM	639000
IrisShield-USB MO 2121	967000
Escáner de iris IrisShield-USB MK 2120U	656000

Fuente: Folcum biometrics (2019). https://www.fulcrumbiometrics.com/IrisCameras-s/36.htm?searching=Y &sort=5&cat=36&show=10&page=1

6. CONCLUSIONES

- Para una buena captura del ojo es recomendable utilizar una cámara que pueda obtener los detalles del iris, además, de que cuente con un enfoque variable el cual permita modificar la distancia de captura y de esta forma obtener la suficiente información en el proceso de reconocimiento.

- La calidad de la imagen tiene una gran influencia en el rendimiento del prototipo puesto que la mayoría de trabajos consultados utilizan la base de datos CASSIA por su gran resolución y nitidez facilitando su trabajo mientras que en este trabajo se obtuvieron imágenes propias que contaban con menos características y menor información evidenciando la robustez del sistema diseñado.

- Se decidió utilizar la zona comprendida entre el borde externo de la pupila y la zona llamada “zigzag collarete" y no toda la región que comprende el iris con el fin de reducir la información necesaria para alimentar los clasificadores y extraer solamente esta región, la cual, contiene la información más relevante del iris y es la menos afectada por reflexiones y las pestañas.

- Por la ubicación que se dio a la iluminación en este prototipo se decidió utilizar la parte comprendida entre los 90° y los 240° (parte izquierda del iris) debido a que la porción restante del iris fue afectada gravemente por las reflexiones y sombras producto de la iluminación haciendo inútil esta zona.

- El factor tecnológico limitó bastante el desarrollar un prototipo más automatizado debido a la falta de una cámara con enfoque automático porque de haber contado con esta, quizás al agregar un sensor de distancia hubiera sido posible automatizar la toma de imágenes cuando alguien se acerque a cierta distancia y no presionando un botón en la interfaz gráfica.

- El tipo de iluminación causó que el prototipo redujera su eficiencia pues al no ser diseñada especialmente para esta aplicación molestaba a los usuarios provocando que no siempre se ubicaran de la mejor manera para tomar las fotografías.

- El número de unidades en la red neuronal tiene una relación inversamente proporcional al tiempo implementado por esta en el proceso de aprendizaje, pero también aumenta el tamaño en memoria del modelo y puede presentar un sobreajuste haciendo más complicado el proceso de reconocimiento, es importante hacer un balance entre el número de unidades (perceptrón), tiempo de entrenamiento y el grado se ajuste de precisión en la red.

- Por el lado de las SVM, el tiempo y sobreajuste del modelo va a depender del factor de aprendizaje (gamma) y la tolerancia de parada de entrenamiento (tol) entre

más grande sea el factor gamma y la tolerancia de parada se acerque a cero, más grande es el tiempo requerido para que el modelo termine con el entrenamiento, pero hará mayor la exactitud en los resultados.

- Al aplicar el filtro de Gabor para la extracción de características, el sistema no fue capaz de separar correctamente las clases para cada usuario, dicho de otra manera, no pudo distinguir las diferencias entre los iris de las personas, por ejemplo, asignaba la misma etiqueta para usuarios diferentes, lo que quería decir, no había homogeneidad entre las plantillas para cada clase y por ende no permitía que los dos modelos de aprendizaje automático convergieran (apuntaran a la minimización del error).

- Los resultados validan que el prototipo trabajó bien en un ambiente real obteniendo un porcentaje de rendimiento de 83% con las redes neuronales que fue el método más efectivo.

BIBLIOGRAFÍA

Alegre, Pajares y De la Escalera. (2016). *Conceptos y Metodos en Vision por computador.* España: Grupo de visiòn del comitè español de automàtica (CEA).

Anil k., Jain and Arun, Ross and Salil, Prabhakar. (2004). An Introduction to Biometric Recognition. *IEEE Transactions On Circuits Ans Systems For Video Technology*.

Anzola, N. (2016). Màquinas de soporte vectorial y redes neuronales artificiales en la predicciòn del movimiento USD/COP spot intradiario. *ODEON*, 113-172.

Arauz, C. (2013). "¿Qué es Google Drive? La nube del Google". [Figura]. Recuperado de http://co.globedia.com/google-drive-nube_2

Attribution, C. (junio de 2017). *Train your first neural network: basic classification*. Obtenido de Train your first neural network: basic classification: https://www.tensorflow.org/tutorials/keras/basic_classification?hl=es#explore_the_data.

Aurélien, G. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. O'Reilly Media, Inc.

Barrera, Jamie Areli-Toral. (s.f.). Redes Neuronales.

Bowyer, Kevin W and Hollingsworth, Karen and Flynn, Patrick J. (2008). Image understanding for iris biometrics: A survey. *Computer vision and image understanding*, 281-307.

Chih-Chung, Chang and Chih-jen, Lin. (2002). LIBSVM: A Library for Support Vector Machines. *ACM transactions on intelligent systems and technology (TIST)*, 27-.

Cordea, Marius and Ionescu, Bogdan and Gadea, Cristian and Ionescu, Dan. (2019). DynFace: A Multi-label, Dynamic-Margin-Softmax Face Recognition Model. *Springer*, 535-550.

David, Forsyth and Jean, Ponce. (2003). *Computer Vision A Modern Approach.*

De Marsico, Maria and Petrosino, Alfredo and Ricciardi, Stefano. (2016). Iris recognition through machine learning techniques: A survey. *Pattern Recognition Letters*, 106-115.

Depositphotos Inc. (2019). Recuperado de https://sp.depositphotos.com/180878578/stock-illustration-laptop-computer-icon-image.html

Diederik P., Kingma and Jimmy Lei, Ba. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Freepik Company S.L. (2019). [Figura] Recuperado de https://www.freepik.es/iconos-gratis/foto-variante-camara_705846.html

Garcìa, A. (2012). Inteligencia Artificial. Fundamentos, pràctica y aplicaciones. Rc Libros.

Ghorbani, Mohammadjavad and Alizadeh, Mahdi and Omran, Alireza Esfahani and Asem, Morteza Modarresi. (2018). An Investigative Review of Human Authentication Based on Fingerprint. *IEEE*, 1359-1366.

Giraldo Calderon, V. (2018). Diseño de un sistema de seguimiento del ojo (pupila), usando técnicas de visión artificial. Neiva.

Gonzàles Duque, R. (2003). *Python Para Todos.*

Gwak, JunYoung and Blevins, Scott and Nabel, Robin. (junio de 2016). *Welcome to PyDrive's documentation!* Obtenido de Welcome to PyDrive's documentation!: https://pythonhosted.org/PyDrive/

Himanshu, Rai and Anamika, Yadav. (2014). Iris recognition using combined support vector machine and Hamming distance approach. *Expert Systems with Applications*, 588-593.

Jan, F. (2017). Segmentation and localization schemes for non-ideal iris biometric systems. *Signal Processing*, 192-212.

Jillela, Raghavender Reddy and Ross, Arun. (2015). Segmenting iris images in the visible spectrum with applications in mobile biometrics. *Pattern Recognition Letters*, 4-16.

Lars Buitinck and Gilles Louppe and Mathieu Blondel and Fabian Pedregosa and Andreas Mueller and Olivier Grisel and Vlad Niculae and Peter Prettenhofer and Alexandre Gramfort and Jaques Grobler and Robert Layton and Jake VanderPlas and Arnaud Joly and Bri. (23 de Mayo de 2011). Scikit - aprender : Machine Learning en Python . *Journal de Machine Learning Research*, 2825-2830. Obtenido de API design for machine learning software: experiences from the scikit-learn.

Libor, Masek. (2003). Recognition of Human Iris Patterns for Biometric Identification.

Mora, C. (2019). "Ojo femenino humano de dibujos animados con ilustración de vector de pestañas". [Figura]. Recuperado de https://es.123rf.com/photo_74741034_ojo-femenino-humano-de-dibujos-animados-con-ilustraci%C3%B3n-de-vector-de-pesta%C3%B1as.html

Nedjah, Nadia and Wyant, Rafael Soares and Mourelle, LM and Gupta, BB. (2017). Efficient yet robust biometric iris matching on smart cards for data high security and privacy. *Future Generation Computer Systems*, 18-32.

OpenCV. (28 de Mayo de 2014). *Morphological Transformations*. Obtenido de Morphological Transformations: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

OpenCV. (23 de Mayo de 2015). *Canny edge detection*. Obtenido de Canny edge detection: https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html

Ossa Vargas, Manuel Fernando and Díaz Vargas, Jonathan Miguel. (2015). *Diseño de algoritmo para detección de tráfico vehicular usando técnicas de visión artificial.* Neiva.

Parker, J. (2010). *algorithms for image processing and computer vision.*

Pedro Ponce, Cruz. (2010). *Inteligencia artificial con aplicaciones a la ingeniería.* Alfaomega.

Poornima, S and Rajavelu, C and Subramanian, S. (2010). Comparison and a neural network approach for iris localization. *Elsevier*, 127-132.

Robayo, Faiber Ignacio and Barrera, Ana María and Polanco, Laura Camila. (2015). Desarrollo de un controlador basado en redes neuronales para un sistema multivariable de nivel y caudal. *Ingenierìa y Regiòn*, 43-54.

Saminathan, K and Chakravarthy, T and Devi, M Chithra. (2015). Iris recognition based on kernels of support vector machine. *ICTACT J. Soft Comput.*, 889-895.

Samir, Kouro and Rodrigo, Musalem. (2002). Tutorial introductorio a la Teoría de Wavelet.

Sibai, Fadi N and Hosani, Hafsa I and Naqbi, Raja M and Dhanhani, Salima and Shehhi, Shaikha. (2011). Iris recognition using artificial neural networks. *Expert Systems with Applications*, 5940-5946.

Sinha, Vijay Kumar and Gupta, Anuj Kumar and Mahajan, Manish. (2018). Detecting fake iris in iris bio-metric system. *Elsevier*, 97-104.

Soliman, Naglaa F. and Mohamed, Essam and Magdi, Fikri and El-Samie, Fathi E Abd and AbdElnaby, M. (2017). Efficient iris localization and recognition. *Elsevier*, 469-475.

Toral Barrera, J. (s.f.). *Redes Neuronales.*

Tossy, Thomas and Anu, George and Indira, Devi. (2016). Effective Iris Recognition System. *Global Colloquium in Recent Advancement and Effectual Researches in Engineering, Science and Technology (RAEREST 2016)*, 464-472.

Wayman, J. L. (2001). Fundamentals of biometric authentication technologie. *International Journal of Image and Graphics*, 93-113.

Wildes, Richard P. (1997). Iris recognition: an emerging biometric technology. *Proceedings of the IEEE*, 1348-1363.

Yao, Z and Yi, W. (2016). Curvature aided Hough transform for circle detection. *Expert Systems With Applications*, 26-33.

ANEXOS

A continuación, se anexa loa códigos implementados desde la segmentación hasta el entrenamiento de las técnicas de *machine learning* implementados en Python 3.6 y 3.5.

ANEXO A. Estimación del círculo de la pupila e iris

'''' código para la estimación del círculo correspondiente al iris y la pupila implementado por medio de la clase bordes1 la cual consta de dos funciones Borde_interno y Borde_externo donde el borde interno corresponde a la pupila y el borde externo al iris.
NOTA: como parámetro de entrada de la clase se requiere dos imágenes en formato matricial una leída en escala de grises y otra a color en formato RGB de tamaño 320x240 aunque se puede modificar el tamaño modificando los parámetros minRadius y maxRadius de la función cv2.Houghcircles .

ejemplo de uso:
```
>circle_detection=bordes1(A,B)
----Donde A es la imagen en escala de grises y B la misma  imagen a
        color-------
>borde_interno=circle_detection.Borde_interno()
>borde_externo=circle_detection.Borde_externo()
```

La salida de las funciones borde_interno y borde_externo es una matriz de 1x3 donde retorna las coordenadas del centro y radio encontrado [x,y,r]. ‘’’

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import math

class bordes1:
  def __init__(self,img,imgc):
        self.img=img
        self.imgc=imgc

  def Borde_interno(self):
      n = True
      umbral = 20
      con = 1
      coor=[]
      while n == True and con <= 6:
        t, dst = cv2.threshold(self.img,umbral,255,cv2.THRESH_BINARY)
```

```
# umbralizacion
        kernel = np.ones((5,5),np.uint8)# mascara para hallar la
            dilatación con el fin de quitar ruido de las pestañas
        dilatacion = cv2.dilate(dst, kernel, iterations=1)# operación
                                                    dilatación
        dstmed = cv2.medianBlur(dilatacion, 5)#mediana de la imagen
        edgesmed = 255 - cv2.Canny(dstmed, 10, 200)  # detectar bordes
                                                        de la imagen dilatada
        edgesmed = cv2.erode(edgesmed, kernel, iterations=1)

        circlesi = cv2.HoughCircles(edgesmed, cv2.HOUGH_GRADIENT, 1, 20,
                                    param1=50, param2=30, minRadius=20,maxRadius=0)
                                                    #se estima el circulo y radio
        tipo = str(type(circlesi))
          if tipo[8:len(tipo) - 2] == 'NoneType':
            n = True
          else:
             n = False
             if circlesi.size > 3:
                n = True
          umbral += 10
          con += 1

       circlesi = np.uint16(np.around(circlesi))
       coor.append(circlesi[0])
       coor=np.array(coor) #la matriz coordenadas del centro y radio interno (de la
pupila)
       for i in circlesi[0, :]:
          cv2.circle(self.imgc, (i[0], i[1]), i[2], (0, 255, 0), 2)
          cv2.circle(self.imgc, (i[0], i[1]), 2, (0, 0, 255), 3)

       return coor[0][0]

    def Borde_externo(self):
       gausiana = cv2.GaussianBlur(self.img, (5, 5), 0)
       kernel1 = np.ones((5, 5), np.float32) / 25
       kernel = np.ones((5, 5), np.uint8)
       dst = cv2.filter2D(gausiana, -1, kernel1)
       matriz_ce = [] #lista para guardar las coordenadas de los centros
       aux1=[] #matriz auxiliar que me guarda los objetos circles con          los
dos metodos de umbral
       i = 1
       umbral = 1
       while i<=2:
          th2 = cv2.adaptiveThreshold(dst, 255,
```

```
                    cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,
11,umbral)
            th2 = cv2.GaussianBlur(th2, (3, 3), 0)
            th2 = cv2.dilate(th2, kernel, iterations=1)
            th2 = 255 - cv2.Canny(th2, 20, 200)
            th2 = cv2.erode(th2, kernel, iterations=1)
            th2 = cv2.medianBlur(th2, 5)
            circles = cv2.HoughCircles(th2, cv2.HOUGH_GRADIENT, 1, 100,
                                       param1=50, param2=30, minRadius=80,
maxRadius=140)
            tipo=str(type(circles))
            if tipo[8:len(tipo)-2]=='NoneType':
                matriz_ce.append([[0,0,0]])
                aux1.append([[0,0,0]])
            else:
                circles = np.uint16(np.around(circles))
                aux1.append(circles)
                matriz_ce.append(circles[0])
            umbral += 1
            i += 1

        matriz_ce = np.array(matriz_ce)
        if sum(sum(sum(matriz_ce))) != 0:
            if matriz_ce[0][0][2] > matriz_ce[1][0][2]:  #aux[0][0][2]=>radio del metodo 1
, aux[0][1][2]=>radi método 2
                r = 1
            else:
                r = 2

            cir_int = self.Borde_interno()  # llama las coordenadadas del bordes interno
            xi = int(cir_int[0])  # se llama de esta manera porque mete un objeto en una
posicion
            yi = int(cir_int[1])
            xe1 = int(matriz_ce[0][0][0])
            xe2 = int(matriz_ce[1][0][0])
            ye1 = int(matriz_ce[0][0][1])
            ye2 = int(matriz_ce[1][0][1])
            d1 = math.sqrt(
                ((xi - xe1) ** 2) + ((yi - ye1) ** 2))  # calculo las distancias de los centros
dist.interno-met1
            d2 = math.sqrt(
                ((xi - xe2) ** 2) + ((yi - ye2) ** 2))  # calculo las distancias de los centros
dist.interno-met2
            y = 52.2820 * d1 + (-69.9397 * d2) + 178 * r - 168
            if y > 0:
```

```
            out = matriz_ce[1][0]  #  escoge las coordenadas del umbral 2
            circles = aux1[
                1]  # escogo de aux el metodo uno para asignarle circles
            print('segundo metodo')
        else:
            out = matriz_ce[0][0]  #  escoge las coordenadas del umbral 1
            circles = aux1[0]
            print('primer metodo')

        for i in circles[0, :]:
            cv2.circle(self.imgc, (i[0], i[1]), i[2], (0, 255, 0), 2)
            cv2.circle(self.imgc, (i[0], i[1]), 2, (0, 0, 255), 3)

        '''plt.subplot(1, 2, 1), plt.imshow(self.img, cmap='gray')
        plt.title('Original Image'), plt.xticks([]), plt.yticks([])
        plt.subplot(1, 2, 2), plt.imshow(self.imgc, cmap='gray')
        plt.title('circulo detectado'), plt.xticks([]),                    plt.yticks([])
        plt.show()'''
        return out
    else:
        return 'error'
```

ANEXO B. Normalización de la imagen

'''

Este código corresponde se encarga de realizar la normalización de la imagen por medio de la clase polcart a partir de las coordenadas x,y correspondientes al centro y con un radio r.

NOTA: la iamgen debe ser en escala de grises formato RGB

ejemplo de uso:

>img_normalizada=polcart().Normalizar(img,cx,cy,rext,rint)

cx=centro en x

cy=centro en y

rext=radio externo (radio correspondiente al iris)

rint=radio interno (radio correspondiente a la pupila

la saalida es una imagen convertida de coordenadas cartesianas a coordenadas polares. '''

```
import math
import numpy as np
from matplotlib import pyplot as plt
```

```
class polcart:
    def __init__(self):
        self.p=[]
    def  Poltocart(self,radio,angle,ingrades):
        r=radio
        ang=angle
        n=ingrades
        if n==0:
            ang=(ang*2*math.pi)/360
        else:
            ang=ang
        x=math.trunc(r*math.cos(ang))
        y=math.trunc(r*math.sin(ang))
        return x,y

    def Normalizar(self,imagen,cx,cy,r_ext,r_int):
        r=r_ext-r_int
        im_out=np.zeros((r,360),np.uint8)
        [m,n]=imagen.shape[0],imagen.shape[1]
        the=0
        while the<360:
            ra=1
            while ra<=r:
                [x,y]=self.Poltocart(ra+r_int,the,0)
                f=math.trunc(int(cy)-y)
                c=math.trunc(int(cx)+x)
                if f<0:
                    f=0
                if c<0:
                    c=0
                if f>m-1:
                    f=m-1
                if c>n-1:
                    c=n-1
                if f>=0 and c>=0:
                    im_out[ra-1,the]=imagen[f,c]
                ra+=1
            the+=1
        '''plt.subplot(1, 2, 1), plt.imshow(imagen, cmap='gray')
        plt.title('Original Image'), plt.xticks([]), plt.yticks([])
        plt.subplot(1, 2, 2), plt.imshow(im_out, cmap='gray')
        plt.title('Normalizar Image'), plt.xticks([]), plt.yticks([])
        plt.show()'''
        return im_out
```

ANEXO C. Extracción de características

'''La función de este código es buscar las imágenes normalizadas almacenadas en la carpeta de trabajo y hacer la extracción de características mediante la transformada de Wavelet y estirar cada imagen de dos dimensiones a una sola dimensión. Además crea los diccionarios correspondientes a los nombres de los usuarios asignando un número a cada usuario por orden alfabético.

-----la función intrain retorna la matriz con las imágenes estiradas correspondientes al grupo de imágenes de entrenamiento.
-----la función intest retorna la matriz con las imágenes estiradas correspondientes al grupo de imágenes de evaluación o testeo.
-----la función targetrain retorna un vector con los números asignados para cada usuario para las imágenes de entrenamiento.
-----la función targetest retorna una vector con los números asignados para cada usuario para las imágenes de testeo.
-----La función labels retorna un diccionario con los números y nombres que corresponde a cada usuario. '''

```
import numpy as np
import cv2
import pywt
from matriz import umbmatriz as umb
import glob

class data:
    def __init__(self):
        self.carpeta=glob.glob('*.bmp')
        self.extension=('.bmp')
    def intrain(self):
        carin=[self.carpeta[i].replace('.bmp','')[:-1] for i in range(len(self.carpeta))]
        auxlis=[]
        for i in carin:
            con=carin.count(i)
            if con>=5:
                auxlis.append(i)
        auxlis.sort()
        k=4
        ca=0
        for j in range(int(len(auxlis)/5)):
            auxlis.pop(k-ca)
            k+=5
            ca+=1
        inputsh=np.zeros([len(auxlis),280],dtype='float')
```

```
        c=0
        n=0
        for j in auxlis:
            rut=j+str(n)+self.extension
            img = cv2.imread(rut,0)
            coeffs=pywt.wavedec2(img, 'haar',level=2)
            ca,(ch,cv,cd)=coeffs[0:2]
            ca=umb().splitarray(ca,'float')
            inputsh[c][:]=ca[:]
            c+=1
            n+=1
            if n>3:
                n=0
        return inputsh/512
    def intest(self):
        carin=[self.carpeta[i].replace('.bmp','')[:-1] for i in range(len(self.carpeta))]
        lis=[]
        auxlis=[]
        for i in carin:
            con=carin.count(i)
            if con>=5:
                auxlis.append(i)
        auxlis.sort()
        k=4
        for j in range(int(len(auxlis)/5)):
            lis.append(auxlis[k])
            k+=5
        inputsh=np.zeros([len(lis),280],dtype='float')
        c=0
        n=4
        for j in lis:
            rut=j+str(n)+self.extension
            img = cv2.imread(rut,0)
            coeffs=pywt.wavedec2(img, 'haar',level=2)
            ca,(ch,cv,cd)=coeffs[0:2]
            ca=umb().splitarray(ca,'float')
            inputsh[c][:]=ca[:]
            c+=1
        return inputsh/512
    def targetrain(self):
        carin=[self.carpeta[i].replace('.bmp','')[:-1] for i in range(len(self.carpeta))]
        target=[]
        auxlis=[]
        for i in carin:
            con=carin.count(i)
```

```
            if con>=5:
                auxlis.append(i)
        auxlis.sort()
        k=4
        ca=0
        for j in range(int(len(auxlis)/5)):
            auxlis.pop(k-ca)
            k+=5
            ca+=1
        l=0
        lc=0
        for c in range(len(auxlis)):
            target.append(l)
            lc+=1
            if lc>3:
                l+=1
                lc=0
        target=np.array(target)
        return target
    def targetest(self):
        carin=[self.carpeta[i].replace('.bmp','')[:-1] for i in range(len(self.carpeta))]
        auxlis=[]
        for i in carin:
            con=carin.count(i)
            if con>=5:
                auxlis.append(i)
        auxlis.sort()
        target=np.arange(int(len(auxlis)/5))
        return target
    def labels(self):
        carin=[self.carpeta[i].replace('.bmp','')[:-1] for i in range(len(self.carpeta))]
        claves=[]
        auxlis=[]
        for i in carin:
            con=carin.count(i)
            if con>=5:
                auxlis.append(i)
        auxlis.sort()
        k=4
        for j in range(int(len(auxlis)/5)):
            claves.append(auxlis[k]) #adiciono los nombres
            k+=5
        keys=np.arange(int(len(claves)))
        dic=dict(zip(keys,claves))
```

```
        return dic
```

ANEXO D. Entrenamiento de la red neuronal

```
'''
En este codigo se hace la implementacion de la red neuronal, donde se entrena
con la matriz de caracteristicas de las imagenes, ademas sube el modelo
entrenado, los nombres con sus respectivos numeros asignados al google drive.

'''

import tensorflow as tf
from tensorflow import keras
from dataraspy import data
import os

class network:
    def __init__(self):
        self.img_train=data().intrain()
        self.img_test=data().intest()
        self.labels_train=data().targetrain()
        self.labels_test=data().targetest()
        self.label=data().labels()
        self.outs=len(self.labels_test)
    def training(self):
        model=keras.Sequential()
        model.add(keras.layers.Dense(10*self.outs,activation=tf.nn.relu))#primera
#capa de n*10 unidades
        model.add(keras.layers.Dense(self.outs,activation=tf.nn.softmax))#capa de
#salida
optimizer=keras.optimizers.Adamax(lr=0.002, beta_1=0.9, beta_2=0.999,
epsilon=None, decay=0.0)#paramtros de optimizacion

model.compile(optimizer=optimizer,loss='sparse_categorical_crossentropy',metrics
=['accuracy'])
        test_loss=1
        cn=0
        while test_loss>0.15 and cn<101 :
            model.fit(self.img_train, self.labels_train, epochs=100)
            test_loss, test_acc = model.evaluate(self.img_test,self.labels_test)
            cn+=1
            print('_______',cn)
        model.save('modelraspi.h5')
        primera=str(list(self.label.keys()))
```

```
        segunda=str(list(self.label.values())) #nombres de los usuarios
        file=open('bloc.txt','w')
        file.write(primera+os.linesep)
        file.write(segunda+os.linesep)
        file.close() #subir las mejores puntuaciones
        predictions=model.predict(self.img_test)
        max_values=[]
        for i in range(len(predictions)):
            max_values.append(max(predictions[i]))
        segundab=str(max_values)
        file=open('bloc_scoresNN.txt','w')
        file.write(segunda+os.linesep)
        file.write(segundab+os.linesep)
        file.close()
```

ANEXO E. Entrenamiento de la máquina de soporte vectorial

```
'''En este código se implementa el entrenamiento de las máquinas de soporte
vectorial  donde también se sube el modelo al google drive y los puntajes de
aceptación para cada usuario.
'''
from dataraspy import data
from sklearn.svm import SVC
from sklearn.externals import joblib
import os

class svm:
    def __init__(self):
        self.img_train=data().intrain()
        self.img_test=data().intest()
        self.labels_train=data().targetrain()
        self.labels_test=data().targetest()
        self.label=data().labels()
        self.outs=len(self.labels_test)
    def trainsvm(self):
        clf = SVC(C=1000,gamma='auto',probability=True,verbose = True,tol=0.01)
        clf.fit(self.img_train,self.labels_train)
        primera=str(list(self.label.keys()))
        segunda=str(list(self.label.values())) #nombres de los usuarios
        file=open('bloc.txt','w')
        file.write(primera+os.linesep)
        file.write(segunda+os.linesep)
        file.close() #subir las mejores puntuaciones
        predictions=clf.predict_proba(self.img_test)
```

```
        max_values=[]
        for i in range(len(predictions)):
            max_values.append(max(predictions[i]))
        segundab=str(max_values)
        file=open('bloc_scores_SVM.txt','w')
        file.write(segunda+os.linesep)
        file.write(segundab+os.linesep)
        file.close()
        joblib.dump(clf,'modelo_SVM.pkl')
        return clf
```

ANEXO F. Interfaz de entrenamiento en el computador

```
'''
Este es el código principal para el computador donde se va a entrenar ambos
métodos, por medio de la interfaz gráfica realizada con las librerías de pyqt donde
llama  a las clases de training( para entrenar las redes neuronales) y trainsvm(para
entrenar las máquinas de soporte vectorial).'''
import sys
from PyQt5.QtWidgets import QApplication,QMainWindow,QDialog
from PyQt5 import uic
import glob
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from training import network
from trainsvm import svm



class Dialogo(QDialog):
    def __init__(self):
        QDialog.__init__(self)
        uic.loadUi("window2.ui",self)
        self.setWindowTitle("Usuarios registrados")


class Ventana(QMainWindow):
    def __init__(self):
        QMainWindow.__init__(self)
        uic.loadUi("Mainwindow.ui",self)
        self.setWindowTitle("Entrenamiento")
        self.buscar.clicked.connect(self.fbuscar)
        self.mostrar_todo.clicked.connect(self.fmostrar_todo)
        self.drive.clicked.connect(self.fdrive)
```

```
        self.actualizar.clicked.connect(self.factualizar)
        self.actualizar_svm.clicked.connect(self.factualizarsvm)
        self.con=0
    def fbuscar(self):
        files=glob.glob('*.bmp')
        files=[files[i].replace('.bmp','')[:-1] for i in range(len(files))]
        nom=self.lineEdit.text()
        auxlis=[]
        for i in files:
            con=files.count(i)
            if con>=5:
                auxlis.append(i)
        auxlis.sort()
        if nom in auxlis:
            a='SI esta registrado'
            self.label.clear()
            self.label.setText(a)
            self.label.setStyleSheet('color:green; font: 75 italic 14pt "URW Chancery
L";')
        else:
            a='No esta registrado'
            self.label.setText(a)
            self.label.setStyleSheet('color:red; font: 75 italic 14pt "URW Chancery L";')
    def fmostrar_todo(self):
        d=Dialogo()
        files=glob.glob('*.bmp')
        files=[files[i].replace('.bmp','')[:-1] for i in range(len(files))]
        auxlis=[]
        lis=[]
        for i in files:
            con=files.count(i)
            if con>=5:
                auxlis.append(i)
        auxlis.sort()
        k=4
        for j in range(int(len(auxlis)/5)):
            lis.append(auxlis[k])
            k+=5
        d.lista.addItems(lis)
        #d.lista.setVerticalScrollBarPolicy(d.verticalScrollBar)
        d.exec_()
    def fdrive(self):
        self.gauth = GoogleAuth()
        self.gauth.LocalWebserverAuth()
        self.con=1
```

```
    #return gauth
def factualizar(self):
    if self.con==1:
        self.label_2.clear()
        self.label_2.setText('actualizando ...')
        drive = GoogleDrive(self.gauth)
        file_list = drive.ListFile({'q': "'root' in parents and trashed=false"}).GetList()
        listitle=[]#LISTA con los titulos de las imgenes
        listid=[]#lista con los ids de las imagenes
        for c in range(len(file_list)):
            listitle.append(file_list[c]['title'])
            listid.append(file_list[c]['id'])
        dic=dict(zip(listitle,listid))
        dic1=dic
        files=glob.glob('*.bmp')
        for i in listitle:
            if i in files:
                dic.pop(i)
        if len(dic)>0:
            for i in list(dic.keys()):
                file=drive.CreateFile({'id': dic[i]})
                file.GetContentFile(i)
        for i in list(dic1.keys()):
            file=drive.CreateFile({'id': dic1[i]})
            if i=='bloc.txt' or i=='bloc_scoresNN.txt' or i=='modelraspi.h5':
                file.Delete()
        network().training()
        self.label_2.setText('se actualizaron todos los archivos')
        fileb=drive.CreateFile()
        fileb.SetContentFile ( 'bloc.txt' )
        fileb.Upload()
        filebs=drive.CreateFile()
        filebs.SetContentFile('bloc_scoresNN.txt')
        filebs.Upload()
        filem=drive.CreateFile()
        filem.SetContentFile('modelraspi.h5')
        filem.Upload()
    else:
        self.label.clear()
        self.label.setText('click en boton rojo')
        self.label.setStyleSheet('color:red; font: 75 italic 14pt "URW Chancery L";')

def factualizarsvm(self):
    if self.con==1:
        self.label_2.clear()
```

```
            self.label_2.setText('actualizando...')
            drive = GoogleDrive(self.gauth)
            file_list = drive.ListFile({'q': "'root' in parents and trashed=false"}).GetList()
            listitle=[]#LISTA con los titulos de las imgenes
            listid=[]#lista con los ids de las imagenes
            for c in range(len(file_list)):
                listitle.append(file_list[c]['title'])
                listid.append(file_list[c]['id'])
            dic=dict(zip(listitle,listid))
            dic1=dic
            files=glob.glob('*.bmp')
            for i in listitle:
                if i in files:
                    dic.pop(i)
            if len(dic)>0:
                for i in list(dic.keys()):
                    file=drive.CreateFile({'id': dic[i]})
                    file.GetContentFile(i)
            for i in list(dic1.keys()):
                file=drive.CreateFile({'id': dic1[i]})
                if i=='bloc.txt' or i=='bloc_scores_SVM.txt' or i=='modelo_SVM.pkl':
                    file.Delete()
            svm().trainsvm()
            self.label_2.setText('se actualizaron todos los archivos')
            fileb=drive.CreateFile()
            fileb.SetContentFile ( 'bloc.txt' )
            fileb.Upload()
            filebs=drive.CreateFile()
            filebs.SetContentFile('bloc_scores_SVM.txt')
            filebs.Upload()
            filem=drive.CreateFile()
            filem.SetContentFile('modelo_SVM.pkl')
            filem.Upload()
        else:
            self.label.clear()
            self.label.setText('click en boton rojo')
            self.label.setStyleSheet('color:red; font: 75 italic 14pt "URW Chancery L";')



app=QApplication(sys.argv)
_ventana=Ventana()
_ventana.show()
app.exec_()
```

ANEXO G. Interfaz de la raspberry

```
'''
código ejecutado sobre la raspberry donde incluye las librerías para la
segmentación del iris, almacena las fotos en el google drive, hace la captura de la
imagen y carga los modelos para el proceso de identificación de usuarios
'''

import sys
from PyQt5.QtWidgets import QApplication,QMainWindow,QDialog,QMessageBox
from PyQt5 import uic
from PyQt5.QtGui  import QPixmap
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from class_inter_exter import bordes1
from pol_cart import polcart
import glob
from picamera import PiCamera
import time
import numpy as np
import cv2
import tensorflow as tf
from tensorflow import keras
from sklearn.externals import joblib
import pywt
from bloc import blocs



class Dialogo_ingreso(QDialog):
    def __init__(self):
        QDialog.__init__(self)
        uic.loadUi("ingreso.ui",self)
        self.setWindowTitle("Ingreso")
        self.ON.clicked.connect(self.fON)
        self.OFF.clicked.connect(self.fOFF)
        self.Redes.clicked.connect(self.fredes)
        self.SVM.clicked.connect(self.fSVM)
        self.conc=True
        self.newmodel_redes=keras.models.load_model('modelraspi.h5')
        self.newmodel_svm=joblib.load('modelo_SVM.pkl')
        pixmap=QPixmap('interrogacionf.jpg')
        self.label.setPixmap(pixmap)
    def fredes(self):
        if self.conc==False:
            image = np.empty((240*320*3,), dtype=np.uint8)
```

```
            self.camera.capture(image, 'bgr')
            image=image.reshape((240,320,3))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            clahe = cv2.createCLAHE(clipLimit=2, tileGridSize=(5,5)) #mascara para la
umbralizacin
            cl1 = clahe.apply(gray)
            x=bordes1(cl1)
            coorin=x.Borde_interno()
            coorex=x.Borde_externo()
            cx=coorin[0]
            cy=coorin[1]
            rint=coorin[2]
            rext=coorex[2]
            if rext=='r':
                self.label_2.clear()
                self.label_2.setText('Intente de nuevo')
                pixmap=QPixmap('interrogacionf.jpg')
                self.label.setPixmap(pixmap)
            else:
                manor=polcart().Normalizar(cl1,cx,cy,rext,rint)
                if manor.shape[0]==25:
                    coeffs=pywt.wavedec2(manor, 'haar',level=2)
                    ca,(ch,cv,cd)=coeffs[0:2]
                    ca=polcart().splitarray(ca,'float')
                    inputsh=np.zeros([1,280],dtype='float')
                    inputsh[0][:]=ca[:]
                    inputsh=inputsh/512
                    scores=blocs().scores_nn()
                    nombres=blocs().nombres_nn()
                    prednew=self.newmodel_redes.predict(inputsh)
                    v=max(prednew[0])
                    vl=nombres[np.argmax(prednew[0])]
                    vls=scores[vl]
                    if v>=(vls-0.15):
                        self.label_2.clear()
                        self.label_2.setText('El usuario es:'+vl)
                        pixmap=QPixmap(vl+'f.jpg')
                        self.label.setPixmap(pixmap)
                    else:
                        self.label_2.clear()
                        self.label_2.setText('Usuario no registrado')
                        pixmap=QPixmap('noregistradof.jpg')
                        self.label.setPixmap(pixmap)
                else:
                    pixmap=QPixmap('interrogacionf.jpg')
```

```
                self.label.setPixmap(pixmap)
                self.label_2.clear()
                self.label_2.setText('Intente de nuevo')

        else:
            self.label_2.clear()
            self.label_2.setText('Presione ON')
    def fSVM(self):
        if self.conc==False:
            image = np.empty((240*320*3,), dtype=np.uint8)
            self.camera.capture(image, 'bgr')
            image=image.reshape((240,320,3))
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            clahe = cv2.createCLAHE(clipLimit=2, tileGridSize=(5,5)) #mascara para la
umbralizacin
            cl1 = clahe.apply(gray)
            x=bordes1(cl1)
            coorin=x.Borde_interno()
            coorex=x.Borde_externo()
            cx=coorin[0]
            cy=coorin[1]
            rint=coorin[2]
            rext=coorex[2]
            if rext=='r':
                self.label_2.clear()
                self.label_2.setText('Intente de nuevo')
                pixmap=QPixmap('interrogacionf.jpg')
                self.label.setPixmap(pixmap)
            else:
                manor=polcart().Normalizar(cl1,cx,cy,rext,rint)
                self.label_2.clear()
                self.label_2.setText('Paso r')
                if manor.shape[0]==25:
                    coeffs=pywt.wavedec2(manor, 'haar',level=2)
                    ca,(ch,cv,cd)=coeffs[0:2]
                    ca=polcart().splitarray(ca,'float')
                    inputsh=np.zeros([1,280],dtype='float')
                    inputsh[0][:]=ca[:]
                    inputsh=inputsh/512
                    scores=blocs().scores_svm()
                    nombres=blocs().nombres_svm()
                    prednew=self.newmodel_svm.predict_proba(inputsh)
                    v=max(prednew[0])
                    vl=nombres[np.argmax(prednew[0])]
                    vls=scores[vl]
```

```
                if v>=(vls-0.1):
                    self.label_2.clear()
                    self.label_2.setText('El usuario es:'+vl)
                    pixmap=QPixmap(vl+'f.jpg')
                    self.label.setPixmap(pixmap)
                else:
                    self.label_2.clear()
                    self.label_2.setText('Usuario no registrado')
                    pixmap=QPixmap('noregistradof.jpg')
                    self.label.setPixmap(pixmap)
            else:
                pixmap=QPixmap('interrogacionf.jpg')
                self.label.setPixmap(pixmap)
                self.label_2.clear()
                self.label_2.setText('Intente de nuevo')

        else:
            self.label_2.clear()
            self.label_2.setText('Presione ON')
    def fON(self):
        self.conc=False
        self.camera=PiCamera()
        self.camera.resolution = (320, 240)
        self.camera.framerate = 20
        self.camera.start_preview(fullscreen=False,window=(250,260,320,240))
    def fOFF(self):
        if self.conc==False:
            pixmap=QPixmap('interrogacionf.jpg')
            self.label.setPixmap(pixmap)
            self.label_2.clear()
            self.label_2.setText('Nombre de usuario')
            self.camera.close()
            self.conc=True


    def closeEvent(self,event):
        if self.conc==False:
          self.camera.close()


class Dialogo_registro(QDialog):
    def __init__(self):
        QDialog.__init__(self)
        uic.loadUi("registro.ui",self)
        self.setWindowTitle("Registro")
```

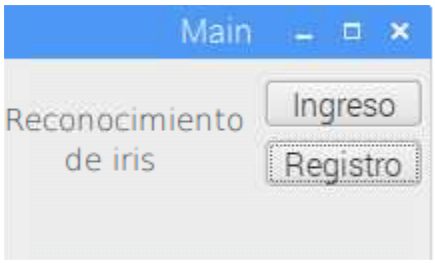
```
        self.camera=PiCamera()
        self.camera.resolution = (320, 240)
        self.camera.framerate = 20
        self.gauth = GoogleAuth()
        self.gauth.LocalWebserverAuth()
        self.camera.start_preview(fullscreen=False,window=(350,300,320,240))
        self.registrar.clicked.connect(self.registro)

    def registro(self):
        nom=self.lineEdit.text()
        self.per=0
        if nom!='':
            image = np.empty((240*320*3,), dtype=np.uint8)
            self.camera.capture(image, 'bgr')
            image=image.reshape((240,320,3))
            rut=nom+'.bmp'
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            clahe = cv2.createCLAHE(clipLimit=2, tileGridSize=(5,5)) #mascara para la
umbralizacion
            cl1 = clahe.apply(gray)
            x=bordes1(cl1)
            coorin=x.Borde_interno()
            coorex=x.Borde_externo()
            cx=coorin[0]
            cy=coorin[1]
            rint=coorin[2]
            rext=coorex[2]
            if rext=='r':
                self.label.clear()
                self.label.setText('intente de nuevo')
            else:
                manor=polcart().Normalizar(cl1,cx,cy,rext,rint)
                if manor.shape[0]==25:
                    self.label.clear()
                    self.label.setText('paso 25')
                    self.per+=1
                    cv2.imwrite(rut,manor)
                    time.sleep(0.2)
                    #sube la imagen a drive
                    drive=GoogleDrive(self.gauth)
                    file5 = drive.CreateFile()
                    file5.SetContentFile(rut)
                    file5.Upload()
                    print(manor.shape[:])
                    conteo="imagen guardada" + str(self.per)
```

```
                self.label.clear()
                self.label.setText(conteo)
            else:
                self.label.clear()
                self.label.setText('intente de nuevo')
        else :
            self.label.clear()
            self.label.setText('debe ingresar un nombre')

    def closeEvent(self,event):
        self.camera.close()


class Ventana(QMainWindow):
    def __init__(self):
        QMainWindow.__init__(self)
        uic.loadUi("principal.ui",self)
        self.setWindowTitle("Main")
        self.Ingreso.clicked.connect(self.fingreso)
        self.Registro.clicked.connect(self.fregistro)
        self.gauth = GoogleAuth()
        self.gauth.LocalWebserverAuth()
        self.cargar()
    def fingreso(self):
        d=Dialogo_ingreso()
        d.exec_()
    def cargar(self):
        drive=GoogleDrive(self.gauth)
        file_list = drive.ListFile({'q': "'root' in parents and trashed=false"}).GetList()
        for c in range(len(file_list)):
            if file_list[c]['title'][-3:]=='txt' or file_list[c]['title']=='modelraspi.h5' or
file_list[c]['title']=='modelo_SVM.pkl':
                file=drive.CreateFile({'id': file_list[c]['id']})
                file.GetContentFile(file_list[c]['title'])
    def fregistro(self):
        r=Dialogo_registro()
        r.exec_()




app=QApplication(sys.argv)
_ventana=Ventana()
_ventana.show()
app.exec_()
```

ANEXO H. Manual de usuario

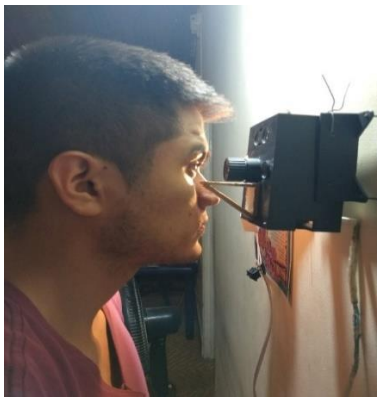
A continuación, se dará una descripción detallada de las funcionalidades de cada uno de las partes del software para dar un correcto uso a todo el sistema de reconocimiento.

Figura 36. Ventana principal



Los botones “Ingreso” y “Registro” de la Figura 36 abren su respectiva ventana de Ingreso o Registro, cada una con su funcionalidad y será explicada más adelante, pero ambas comparten el hecho de que es necesario adquirir una imagen y en la Figura 37 se evidencia como el usuario se ubica para la toma de la fotografía.

Figura 37. Adquisición de la imagen



Los botones presentes en esta ventana principal tienen como función abrir dos ventanas secundarias, donde una permitirá el registro de usuarios (botón de registro) y la otra la identificación de usuarios registrados (botón de ingreso), cada ventana se explicará individualmente a continuación:

- **Ventana de registro**: cuando en la ventana principal se presiona el botón “registro", emerge la ventana mostrada en la Figura 38, la cual automáticamente enciende la cámara para la captura de imágenes, la ventana está compuesta

además de un *edittext*, un *pushbutton* y un *label*. El registro de un usuario es muy sencillo e intuitivo, se inicia escribiendo en el *editext* el nombre del usuario acompañado de un número que irá de 0 a 4 que se aumentará manualmente con la captura de cada foto, luego se pide al usuario que se ubique frente al prototipo con su ojo mirando fijamente a la cámara donde se le darán indicaciones para tomar correctamente la foto, cuando se encuentre en una postura adecuada se presiona el botón “Registrar" el cual captura la imagen, le asigna el nombre escrito en el *edittext* y además sube la imagen a google drive, este proceso se repite 5 veces con cada usuario en caso de que se pueda hacer la estimación de la zona correspondiente al iris y el *label* imprime el mensaje “imagen guardada n" donde n es un número que indica la cantidad exitosa de imágenes guardadas, en caso contrario la captura será descartada y en el *label* se imprime el mensaje “intente de nuevo".

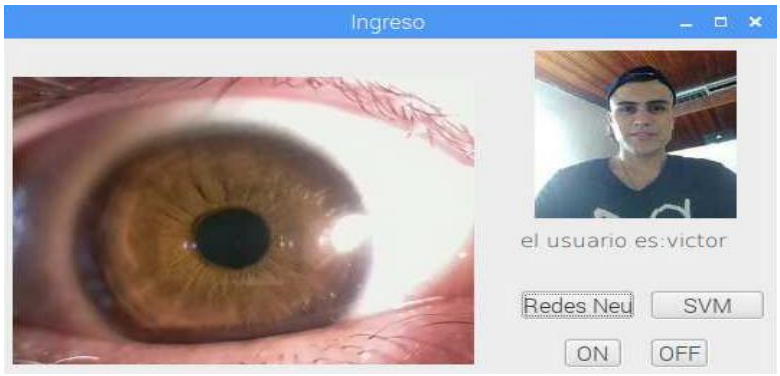
Figura 38. Ventana de registro

- **Ventana de ingreso**: cuando en la ventana principal se presiona el botón “Ingreso", emerge la ventana mostrada en la Figura 39, la ventana está compuesta de cuatro *pushbutton* y dos *label.*

En esta ventana el proceso inicia cuando se presiona el botón “On" el cual enciende la cámara fotográfica, luego se pide al usuario que se ubique frente a la cámara y que mire fijamente a esta, para luego darle indicaciones y ubicarlo en la mejor posición para capturar una buena imagen, la captura de la imagen y su respectiva validación puede ser realizada con dos botones, “Redes Neuronales" y “SVM", el primero toma la imagen y valida al usuarios mediante redes neuronales mientras que el segundo lo hace por máquina de soportes vectoriales, en caso de que la persona sea reconocida, se mostrará una imagen de la persona que se subió anteriormente a la cuenta de drive y un *label* imprimirá el nombre del usuario, en caso de no reconocer la persona se mostrará una imagen predeterminada para este

caso y en el *label* se imprimirá el mensaje “usuario no reconocido", vale la pena aclarar que esta imagen se sube manualmente al drive y el algoritmo diseñado solo la descarga de drive y la muestra en la ventana de ingreso, el otro caso que se puede presentar es que la foto tomada no sea bien segmentada por lo tanto el *label* informará con el mensaje “intente de nuevo" y finalmente si se desea apagar la cámara se presiona el botón “OFF".

Figura 39. Ventana de ingreso



Interfaz gráfica para el computador

Se creó también una interfaz diseñada para el computador mostrada en la Figura 40, La interfaz de entrenamiento consta de cinco *pushbutton*, dos *label* y una caja de texto (*editText*) de los cuales se explicarán el funcionamiento a continuación.

Para iniciar el botón “Conectar drive" utiliza el archivo de credenciales el cual se creó como lo indica[52] Gwak, Blevins, y Nabel para acceder a la cuenta de drive, el botón “Actualizar NN" entrena y crea el modelo para redes neuronales mientras que el botón “Actualizar SVM" entrena y crea el modelo para máquinas de soporte vectorial pero además los dos botones suben los modelos a google drive junto con el documento de los nombres de los usuarios registrados; cuando se presiona el botón “buscar", este toma el nombre escrito en la caja de texto (*EditText*) y busca una coincidencia entre los usuarios registrados, en caso de ser así en el *label* se imprime el mensaje “Si está registrado”, de lo contrario imprime el mensaje “No está registrado", finalmente el botón “Mostrar todo" crea una ventana secundaria donde se pueden observar todos los usuarios registrados.

[52] Gwak, JunYoung and Blevins, Scott and Nabel, Robin, 2016

Figura 40. Interfaz gráfica del computador