DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

TESIS PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

TEMA: IMPLEMENTACIÓN DE UN SISTEMA PARA EL MANEJO DEL MOUSE DE UN PC A TRAVÉS DEL MOVIMIENTO OCULAR

AUTORAS: ITÁS MAZÓN, KAREN SULAY
SANGO QUIMBITA, KARINA MIREYA

DIRECTOR: ING. SÁENZ, FABIÁN
CODIRECTOR: ING ROMERO, CARLOS

SANGOLQUÍ – ECUADOR

2015

Certificado de Tutoría

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICADO

Ing. Fabián Sáenz
Ing. Carlos Romero

CERTIFICAN

Que el trabajo titulado "IMPLEMENTACIÓN DE UN SISTEMA PARA EL MANEJO DEL MOUSE DE UN PC A TRAVÉS DEL MOVIMIENTO OCULAR" realizado por Karen Sulay Itás Mazón y Karina Mireya Sango Quimbita, ha sido guiado y revisado periódicamente y cumple normas estatuarias establecidas por la ESPE, en el Reglamento de Estudiantes de las Fuerzas Armadas ESPE.

Debido a que el presente trabajo se basa en el diseño de un sistema para el manejo del mouse de un PC a través del movimiento ocular, es una herramienta amigable y de fácil manejo para el usuario. Se recomienda su publicación.

Sangolquí, 18 de Mayo del 2015

Autoría de Responsabilidad

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

KAREN SULAY ITÁS MAZÓN

KARINA MIREYA SANGO QUIMBITA

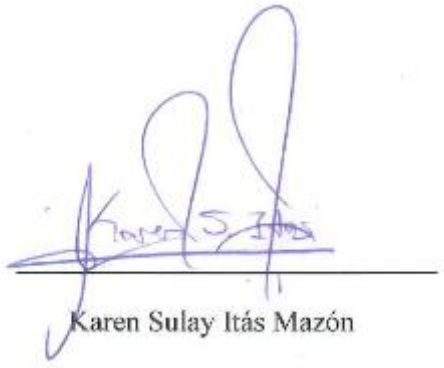
DECLARAMOS QUE:

El proyecto de grado denominado "IMPLEMENTACIÓN DE UN SISTEMA PARA EL MANEJO DEL MOUSE DE UN PC A TRAVÉS DEL MOVIMIENTO OCULAR", ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente el diseño e implementación del presente trabajo es de nuestra total autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, 18 de Mayo del 2015



Karen Sulay Itás Mazón

Karina Mireya Sango Quimbita

Autorización de Publicación

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN

Nosotras, Karen Sulay Itás Mazón y Karina Mireya Sango Quimbita

Autorizamos a la Escuela Politécnica del Ejército la publicación, en la biblioteca virtual de la Institución, del trabajo "IMPLEMENTACIÓN DE UN SISTEMA PARA EL MANEJO DEL MOUSE DE UN PC A TRAVÉS DEL MOVIMIENTO OCULAR", cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Sangolquí, 18 de Mayo del 2015

Karen Sulay Itás Mazón

Karina Mireya Sango Quimbita

DEDICATORIA

A mi mamá Paña, por sus bendiciones, sus sabios consejos, su comprensión, por ser ejemplo de mujer para sus hijos, por su forma tan carismática y graciosa de demostrar su amor incondicional por nosotros.

A mi papá Beto, por sus bendiciones, su paciencia, por dejarme el recuerdo de su amor constante durante toda mi vida y por siempre haber estado ahí apoyándome en mis buenos y malos momentos.

A mi preciosa hija Sarita Michelle, por inspirarme para ser mejor cada día, por su amor incondicional que se ha convertido en mi motor y motivo; y por su sonrisa angelical que alumbra hasta el día más sombrío. Ningún sacrificio es suficiente.

A mi amado esposo y amigo Dany, por ser fuente de sabiduría y calma, por ser el pilar fundamental de mi vida, que con su apoyo y amor incondicional se ha convertido en mi compañero de vida.

A mi ñaña Kary, por ser un ejemplo de perseverancia, por ser una buena hija, madre y hermana, por ser la luz de esperanza en el hogar.

A mi ñaño Roly, por siempre desbordar alegría en el hogar, por mostrar fuerza y humildad ante cualquier adversidad.

A mi sobrina Savie, por haberse convertido en mi primera hija, le agradezco por el amor y cariño que nos brinda a mí y a mi pequeña.

Karen Sulay Itás Mazón

DEDICATORIA

A mi madre Edito, por haberme dado la vida, su apoyo incondicional, sus bendiciones y todo su amor y cariño brindado día a día.

A mi padre Julay, quien siempre ha sabido guiarme por el buen camino, inculcándome valores, enseñándome a luchar ya que él es el pilar fundamental en mi vida profesional.

A mis hermanos Gaby y Alex, quienes siempre han estado a mi lado con sus consejos, sus locuras, acompañándome en mis noches de desvelo.

A mi sobrinita Julianita, quien cada día alegra mi vida y por quien nunca rendirme y seguir cumpliendo cada una de mis metas, y ser ejemplo para su vida estudiantil.

A mis abuelitos, por haberme dado sus bendiciones y permitirme estar en sus oraciones.

A Andrés, por su apoyo incondicional, por demostrar que siempre puedo contar con él y por compartir conmigo alegrías y tristezas.

A mi familia y amigos, por haberme brindado su apoyo y su amistad incondicional, compartiendo buenos y malos momentos.

Karina Mireya Sango Quimbita

AGRADECIMIENTO

Primeramente a Dios, por bendecirme de mil formas y permitirme realizar éste sueño tan anhelado.

También a toda mi familia por sus palabras de aliento durante el transcurso de mi formación académica.

A mi amiga Karina, por el trabajo en equipo realizado y por su muestra de perseverancia para lograr la culminación del presente proyecto.

A la Universidad de las Fuerzas Armadas – ESPE, por abrirme las puertas y permitirme adquirir conocimientos que me serán útiles para la realización de mi vida profesional.

A los ingenieros Fabián Sáenz y Carlos Romero, por la confianza depositada para la realización de éste proyecto.

A mis maestros, que me impartieron sus conocimientos, experiencias y anécdotas en las aulas de la universidad.

A todos mis amigos, en especial a los que me acompañaron durante ésta etapa académica; pues gracias a su compañía, humor y conocimiento la vida universitaria fue la mejor.

Karen Sulay Itás Mazón

AGRADECIMIENTO

A Dios por haberme permitido cumplir con una meta más de mi vida, por protegerme y guiarme en cada momento.

Agradezco a mi familia y a Andrés, por su apoyo incondicional durante el transcurso de mi vida profesional.

A mi amiga Karen, que gracias al trabajo realizado juntas, logramos culminar este proyecto.

A los Ingenieros Fabián Sáez y Carlos Romero, por haber confiado en mí para desarrollar el presente proyecto.

A todos mis maestros, por compartir sus conocimientos, gracias por su paciencia y finalmente un agradecimiento a esta Universidad de las Fuerzas Armadas – ESPE la cual abrió sus puertas a jóvenes como yo, preparándonos para un futuro competitivo y formándonos para ser excelentes profesionales.

Karina Mireya Sango Quimbita

PRÓLOGO

La accesibilidad digital es uno de los mandatos fundamentales que se encuentran establecidos y promovidos internacionalmente por las Naciones Unidas. Además la Unión Internacional de Telecomunicaciones, respecto a la: "Situación de acceso a las TIC por personas con discapacidad", recomienda la adopción de instrumentos legales para establecer el acceso a las TIC como un derecho y obligación de los Estados. En este contexto, el acceso al mundo computacional y virtual se ha hecho prácticamente indispensable para la ejecución de tareas diarias.

En nuestro país el 13,2% del total de la población, de acuerdo al Consejo Nacional de Discapacidades (CONADIS), sufre algún tipo de discapacidad, esto significa que más de 1,6 millones de ecuatorianos requieren protección especial. Entre los dispositivos y equipos de cuidado que han sido desarrollados para ayudar a las personas discapacitadas, se encuentra el de apoyo para personas con discapacidades severas, éste es un instrumento de comunicación con computadores o dispositivos similares., su principal problema radica en controlar las dificultades debido a las habilidades físicas limitadas de los usuarios.

Existe una multiplicidad de programas y herramientas modernas que están abriendo cada día más las puertas del mundo virtual a los discapacitados. Existen sistemas y programas basados en software libre, así como algunos sitios que proporcionan herramientas gratuitas, pero en general los altos precios de estas tecnologías especializadas, si bien han disminuido bastante en los últimos años, siguen siendo una barrera importante al acceso.

El Ecuador busca dar pasos hacia adelante en la inclusión social, por lo que la importancia del desarrollo del presente proyecto radica en realización de una herramienta que permita accesibilidad a una computadora a través del mouse, empleando costos relativamente económicos.

Ésta tesis presenta la propuesta de realización de un sistema que permita la interactividad entre el usuario, persona discapacitada, y el computador a través del

mouse, usando elemento económicos que den como resultado una herramienta de bajo costo y eficiente.

A lo largo del desarrollo se encuentra una breve descripción de requerimientos, elección de la cámara web, descripción del uso, diseño de la herramienta, implementación de la herramienta y pruebas. Además se puede encontrar un marco teórico que abarca desde conceptos básicos, como la definición del ojo, hasta conceptos complejos, como la visión artificial y sus funciones. También se halla las decisiones de diseño, las cuales en su momento fueron analizadas y ejecutadas de acuerdo a la necesidad que se planteó en el presente proyecto.

Cabe aclarar que la herramienta permite solamente realizar algunas de las funcionalidades del mouse: movimiento del cursor, clic izquierdo y doble clic; más no funcionalidades como clic derecho, arrastrar una región, etc. Dichas funcionalidades pueden ser implementadas, por lo que se deja la posibilidad de que el sistema pueda ser desarrollado a futuro.

Este documento, en primera instancia, presenta una introducción donde se muestra una visión general del proyecto, donde consta de antecedentes, justificación e importancia, el alcance del mismo; y objetivos tanto general como específicos. A continuación se define mediante fundamento teórico la información sobre la investigación en lo que respecta al proyecto, es decir conceptos básicos asociados al estudio como lo relacionado con el ojo humano y el mouse de una computadora; además se indagó en la visión artificial, el cómo se forman las imágenes y su respectivo procesamiento. Además se trató como otro punto a la cámara web como sensor y se hace una recopilación de proyectos similares realizados en beneficio de personas con discapacidad motriz.

Seguidamente, se desarrolló un diseño previo a la parte experimental. Entre los temas a tratados están las modificaciones respectivas de la cámara web, incluyendo estudios de características y posicionamiento de la cámara; la metodología de las funciones de visión artificial entre las cuales se encuentran funciones de: morfología binaria, umbralización y conversión de imágenes en escala de grises; es decir los

algoritmos de reconocimientos necesarios para emular el movimiento del puntero del mouse. Se definió funciones y procedimientos, tratamiento de las imágenes para la obtención de parámetros acordes a la necesidad del usuario.

Seguido al diseño se procedió a implementar el mouse con los algoritmos obtenidos previamente, se realizó un análisis en las imágenes captadas del ojo del usuario, de acuerdo con los algoritmos se logró posicionar el puntero del mouse y seleccionar el clic o doble clic.

A continuación se procedió a realizar un análisis de resultados a partir de las pruebas hechas. Esto se lo hizo mediante la representación del algoritmo en diagrama de flujo, se detallan los resultados obtenidos en la ejecución del proyecto; además se realiza una evaluación del funcionamiento del sistema, enfocándose principalmente en el cumplimiento de que haya existido un nivel de error inferior al 20% en las tareas que conlleva el movimiento y selección del cursor de navegación por parte del usuario; y finalmente se muestra un análisis de factibilidad económica del proyecto. Por último se presenta conclusiones de los resultados obtenidos, se resalta los resultados más notorios durante la realización; y recomendaciones para futuros trabajos.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En Ecuador el 13,2 % del total de la población sufre alguna discapacidad según el CONADIS; y el acceso al mundo computacional y virtual se ha hecho prácticamente indispensable para la ejecución de tareas diarias. En este contexto, se han desarrollado instrumentos e interfaces que facilitan la comunicación de dichas personas con computadores y similares, su principal problema es controlar las dificultades debido a las habilidades físicas limitadas de los usuarios. El presente proyecto está orientado al diseño e implementación de una herramienta que permita usar el mouse de un computador a través de la detección de movimiento ocular, la importancia de la realización del presente proyecto radica en mejorar la calidad de vida de usuarios que posean discapacidad motriz. Ésta herramienta hará posible que con recursos relativamente económicos el usuario pueda interactuar con el computador y tener acceso a diversas tareas computacionales. La realización del actual proyecto consta de las siguientes etapas: descripción de requerimientos, donde se analiza las características con las cuales se desea trabajar; en elección de la cámara web se describe los motivos para elegir la cámara a implementar; descripción del uso, donde se argumenta los posibles escenarios en los cuales se utilizará la herramienta; diseño de la herramienta, etapa donde se propone ideas de realización de la herramienta, se muestra como ensamblar el periférico alternativo y las respectivas modificaciones hechas a la cámara web; implementación, se encuentra funciones y procedimientos de OpenCV, API Windows y los creados; y pruebas, donde se expone la efectividad, factibilidad económica y resultados que otorga la herramienta.

PALABRAS CLAVES:

- **DISCAPACIDAD**
- **MOUSE**
- **MOVIMIENTO OCULAR**
- **OPENCV**
- **API WINDOWS**

ABSTRACT

In Ecuador 13.2% of the population have a disability according to CONADIS; and access to computing and virtual world has become practically indispensable for the performance of daily tasks. In this context, it has been developed tools and interfaces that facilitate communication of these people with computers and the like, their main problem is to control difficulties due to limited physical abilities of users. This project is aimed to design and implement a tool to use the mouse of a computer through the detection of eye movement; the importance of the implementation of this project is to improve the quality of life of users with motor disabilities. This tool will enable relatively economic resources the user can interact with the computer and have access to various computational tasks. The realization of this project consists of the following stages: description of requirements, where features with which you want to work is analyzed; in choosing the webcam describes the reasons to choose the camera to implement; description of the use, it shows the possible scenarios in which the tool is going to be used; design, stage where we propose ideas how to make the tool, it shows how to assemble alternative peripheral and respective modifications to the webcam; implementation, functions and procedures of OpenCV, API Windows and the created were found; and testing, where the effectiveness, economic feasibility and results gives the tool are exposed.

KEYWORDS:

- **DISABILITY**
- **MOUSE**
- **EYE MOVEMENT**
- **OPENCV**
- **API WINDOWS**

GLOSARIO DE TÉRMINOS

API	Application Programming Interface
BGR	Red,Green and Blue colors.
BSD	Liberada bajo la licencia
CCD	Charge Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
CONADIS	Consejo Nacional de Discapacidades
DivX	Formato de vídeo que funciona sobre sistemas operativos: Windows, MacOS y GNU/Linux actuales
EOG	Electrooculograma
FPS	Frames per Second
GNU	GNU's Not UNIX
GUI	Graphic User Interface
IPL	Intel Image Processing Library
IRLED	Infrared Light Emitting Diode
LED	Light-Emitting Diode
MPX	Megapixel
ROI	Region of Interest
SDK	Software Development Kit
USB	Universal Serial Bus
VHS	Video Home System

CAPÍTULO 1

INTRODUCCIÓN

1.1 Antecedentes

Hoy en día, el acceso al mundo computacional y virtual se ha hecho prácticamente indispensable para la ejecución de tareas diarias. La tecnología informática ha llegado a ser un universo de aplicaciones que permiten hacer procesos más eficientes y comunicaciones más sencillas, las cuales dan acceso al conocimiento y entretenimiento.

Más de 500 millones de personas, 10 por ciento total de la población mundial, sufre algún tipo de discapacidad; en el Ecuador la cifra representa el 13,2% del total de la población de acuerdo al Consejo Nacional de Discapacidades (CONADIS), esto significa que más de 1,6 millones de ecuatorianos requieren protección especial [1] [2].

Actualmente diferentes dispositivos y equipos de cuidado han sido desarrollados para ayudar a las personas discapacitadas. Uno de los más importantes dispositivos de apoyo para personas con discapacidades severas es un instrumento de comunicación con computadores o dispositivos similares. Sin embargo el principal problema con estos dispositivos es el controlar las dificultades debido a las habilidades físicas limitadas de los usuarios. Es así que diferentes tipos de interfaces han sido desarrolladas para facilitar la comunicación entre el usuario discapacitado y dichos dispositivos.

1.2 Justificación e importancia

En el Ecuador, de acuerdo al Plan Nacional del Buen Vivir, se busca implementar políticas para superar limitaciones arquitectónicas, implementar ayudas técnicas, medicamentos, unidades de rehabilitación especializadas y facilidades para la inserción laboral de las personas discapacitadas que desean incorporarse al trabajo.

Es por esto que una de las importancias de este proyecto radica sobre el Objetivo 1 del Plan Nacional de Buen Vivir: “Auspiciar la igualdad, cohesión e integración social y territorial en la diversidad”, específicamente en la Política 1.6 que habla sobre reconocer y respetar las diversidades socioculturales y erradicar toda forma de discriminación por diferentes motivos o condiciones, entre los cuales se encuentra la discapacidad. Además la realización de éste proyecto aportará al cumplimiento de una de las metas planteadas referentes al Objetivo 1, que dice: “Alcanzar el 40% en la tasa de ocupación plena en personas con discapacidad al 2013” [2].

El manejo de un computador tradicional implica el uso de dispositivos de acceso, como los son el teclado y el ratón; y requiere la capacidad física del movimiento de, por lo menos, una de las extremidades del ser humano. En este sentido, la disposición de una herramienta que permita controlar el cursor del ratón a través del movimiento ocular es de suma importancia ya que permite ampliar el tipo de usuarios. Esto hace posible que con recursos relativamente económicos cualquier persona discapacitada pueda interactuar con el computador y con esto tener acceso a tareas que le aseguraran un mejoramiento en su calidad de vida. Entre la variedad de oportunidades que brinda la herramienta está la comunicación, entretenimiento, acceso a la información; todo esto dependiendo de las condiciones del usuario.

Existen usos de la visión artificial para desarrollar sistemas que pueden ser utilizados mediante un ordenador, pero presentan ciertas desventajas como costos elevados y, en ciertos casos, la necesidad de que el usuario tenga control de movimiento voluntario de su cabeza; además requieren dispositivos adicionales para detectar el movimiento de los ojos [3].

Respecto a lo antes mencionado surgió la inquietud por realizar un estudio e implementación utilizando técnicas de visión artificial para proporcionar a pacientes tetrapléjicos la posibilidad de realizar tareas computacionales, a través de imágenes del movimiento de sus ojos capturadas por medio de una cámara web, ésta disminuirá el costo económico puesto que eliminará la utilización de equipos complejos y costosos. El objetivo es crear una herramienta para personas con tetraplejia que permita en primera instancia detectar el cambio del movimiento ocular

para identificar las intenciones del usuario, y con esto que el usuario mueva el cursor en las direcciones deseadas.

La implementación de la herramienta brindará a las personas tetrapléjicas la oportunidad de interactuar con un computador para realizar diversas actividades como: acceso a Internet, manejo de una variedad de programas con distintos propósitos y cualquier tarea que se pueda realizar usando un ratón tradicional de un computador. Es notorio el beneficio que ofrece a personas con este tipo de discapacidad, ya que esta herramienta ofrece la posibilidad de interactuar con el mundo exterior de una manera económica y simple.

1.3 Alcance del proyecto

El documento del proyecto, en primera instancia, presentará una introducción donde se mostrará una visión general que constará de antecedentes, justificación e importancia del proyecto, el alcance del mismo, objetivos tanto general como específicos.

A continuación se definirá mediante fundamento teórico la información sobre la investigación en lo que respecta al proyecto, es decir conceptos básicos asociados al estudio como lo relacionado con el ojo humano y el ratón de una computadora; además se indagará en la visión artificial, el cómo se forman las imágenes y su respectivo procesamiento. Además se tratará como otro punto a la cámara web como sensor. Además se hace una recopilación de proyectos similares realizados de mouse en beneficio de personas con discapacidad motriz.

En este punto, se procederá a desarrollar un diseño previo a la parte experimental. Entre los temas a tratar estará las modificaciones respectivas de la cámara web, incluyendo estudios de características y posicionamiento de la cámara; la metodología de las funciones de visión artificial entre las cuales se encontrará funciones de: morfología binaria, umbralización, segmentación, histogramas y conversión de imágenes en escala de grises; es decir los algoritmos de reconocimientos necesarios para emular el movimiento del puntero del ratón. Se definirá funciones y procedimientos, tratamiento de las imágenes para la obtención de parámetros acordes a la necesidad del usuario.

Seguido al diseño se procederá a implementar el ratón virtual con los algoritmos obtenidos previamente, se hará un análisis en las imágenes captadas del ojo del usuario, de acuerdo con los algoritmos se logrará posicionar el puntero del mouse y seleccionar el clic, doble clic según sea la necesidad del usuario.

A continuación se procede a realizar un análisis de resultados a partir de pruebas hechas. Esto se lo hará mediante la representación del algoritmo en diagrama de flujo, se detallará los resultados obtenidos en la ejecución del proyecto; además se realizará una evaluación del funcionamiento del sistema, enfocándose principalmente en el cumplimiento de que haya existido un nivel de error inferior al 20% en las tareas que conlleva el movimiento y selección del cursor de navegación por parte del usuario; y finalmente se hará un análisis de factibilidad económica del proyecto. Por último se presentará conclusiones de los resultados obtenidos, se resaltará los resultados más notorios durante la realización; y recomendaciones para futuros trabajos.

1.4 Objetivos

1.4.1 Objetivos General

Realizar un sistema que permita el manejo del mouse de un PC a partir de la detección del movimiento ocular garantizando el uso de personas discapacitadas.

1.4.2 Objetivos Específicos

- Realizar pruebas de funcionamiento y corrección de errores. Realizar un estado del arte sobre estudios previos relacionados a sistemas creados en base a la detección de movimiento ocular.
- Desarrollar el algoritmo pertinente mediante un software especializado.
- Procesar imágenes en base a datos obtenidos por la cámara web.
- Implementar el sistema en base a los algoritmos desarrollados, características y posicionamiento de la cámara.
- Efectuar las pruebas correspondientes y un análisis de los resultados obtenidos del sistema.
- Evaluar el sistema de modo que garantice resultados con niveles de error inferiores al 20%

CAPÍTULO 2

FUNDAMENTO TEÓRICO

Este capítulo contiene información sobre la recopilación de información teórica necesaria para la realización de este proyecto. Se incluyen algunos conceptos básicos relacionados al proyecto, como lo son el ojo humano y sus partes, el ratón en el PC, la visión artificial, la cámara web como sensor, sumado a esto se incluyen un breve resumen de proyectos previos realizados con similitud al actual.

2.1 El ojo humano

2.1.1 Definición

El ojo humano es un órgano esférico detector de luz que posibilita la visión, también conocido como globo ocular. Su función es la de captar colores y formas de objetos presentes en el exterior, por medio de estímulos luminosos que son transportados al centro nervioso de la visión en el cerebro. El ojo es la base del sentido de la vista, éste es uno de los cinco sentidos del ser humano que le permite comprender el mundo que le rodea. Las partes básicas visibles del ojo se pueden observar en la Figura 2.1.

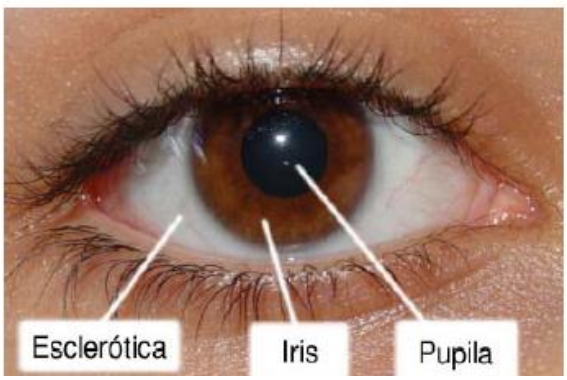


Figura 2.1 Partes básicas visible del ojo

Para la realización del proyecto es necesario conocer las partes fundamentales del glóbulo ocular, que son: pupila, iris, esclerótica, córnea, cristalino, retina, coroides, nervio óptico y cuerpo vítreo. (Véase Figura 2.2). La membrana que recubre al ojo es llamada esclerótica (en su zona opaca) y la córnea (en su zona

transparente). La pupila es un círculo negro, ubicado en el centro del ojo, y enmarcado por el iris (membrana de forma circular). Detrás de la pupila se encuentra el cristalino, que ayuda al enfoque de los objetos. Para que nos sea posible visualizar un objeto, la luz reflejada sobre éste ingresa por nuestras pupilas, y a continuación se proyecta en la retina (membrana). Luego, por medio del nervio óptico, esa información se dirige hacia el cerebro para ser analizada. Cuando se habla del color de los ojos de una persona, se refiere al color del iris, que puede ser: azul, verde, o marrón, en sus distintas tonalidades, y en ocasiones, combinado con matices grisáceos.

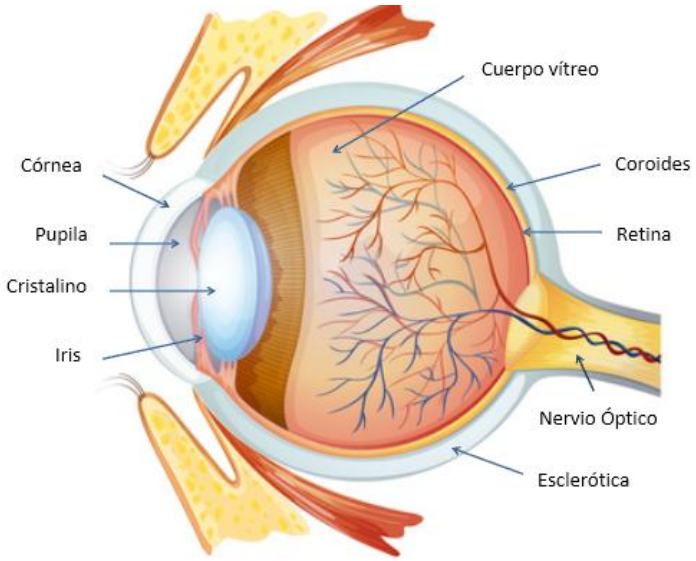


Figura 2.2 Ojo humano y sus partes [4]

2.2 Funcionamiento del ratón en la PC

2.2.1 Conceptos básicos

Mouse: Ratón en español. El mouse es un periférico de entrada para interactuar con la computadora a través de un puntero mostrado en pantalla en sistemas GUI (gráficos). [5]

GUI: Interfaz Gráfica de usuario (Graphic User Interface). Conjunto de formas y métodos que posibilitan la interacción de un sistema con los usuarios utilizando formas gráficas e imágenes. Con formas gráficas se refiere a botones, íconos, ventanas, fuentes, etc. los cuales representan funciones, acciones e información. [5]

API: Conjunto de funciones (API, WinAPI, Windows API) contenidas en bibliotecas que permiten que una aplicación se ejecute en el sistema operativo Microsoft Windows. Son diseñadas en C y C++. Sus versiones son Win16, Win32, Win32s, Win64. Microsoft desarrolló un SDK (kit de desarrollo de software), que provee la documentación y las herramientas que permite a los desarrolladores crear software usando las API de Windows y otras tecnologías asociadas a este sistema operativo. [5]

2.2.2 Funciones y procedimientos API

El presente proyecto se realizó bajo el sistema operativo Windows 7, se utilizó la versión Win32 de API, que son un conjunto de funciones y procedimientos que se ejecutan bajo el sistema operativo antes mencionado.

Entre los beneficios que presenta la librería mencionada se encuentran la posibilidad de trabajar con los eventos del ratón, posición del cursor y resolución de la pantalla, por lo que a continuación se detalla los eventos de importancia para el proyecto.

MOUSE _EVENT: Esta función sintetiza los movimientos y clics de los botones del ratón. Respecto a los cinco argumentos que se encuentran en la función se destacan los detallados en el Tabla 2.1.

Tabla 2.1

Eventos de MOUSE_EVENT

EVENTO	SIGNIFICADO
DWORD dwFlags	Son banderas que especifican variantes de movimiento y clics.
DWORD dx	Se encarga de la posición horizontal del ratón o cambio de posición.
DWORD dy	Se encarga de la posición vertical del ratón o cambio de posición.
DWORD dwData	Se encarga de la cantidad de movimiento de rueda.
DWORD dwExtraInfo	Contiene la información de 32 bits definida por aplicación.

DWORD dwFlags: Es un conjunto de bits de bandera que especifica varios aspectos del movimiento o clic de botones del ratón. Los bits en este parámetro

pueden ser combinaciones razonables de los siguientes valores que se observan en el Tabla 2.2.

Tabla 2.2

Descripción de bits en el parámetro dwFlags

BITS	SIGNIFICADO
MOUSEEVENTF_MOVE	Específica qué ha ocurrido un movimiento.
MOUSEEVENTF_LEFTDOWN	Específica que el botón izquierdo ha cambiado ha pulsado.
MOUSEEVENTF_LEFTUP	Específica que el botón izquierdo ha cambiado ha no pulsado.
MOUSEEVENTF_RIGHTDOWN	Específica que el botón derecho ha cambiado ha pulsado.
MOUSEEVENTF_RIGHTUP	Específica que el botón derecho ha cambiado ha no pulsado.
MOUSEEVENTF_MIDDLEDOWN	Específica que el botón central ha cambiado ha pulsado.
MOUSEEVENTF_MIDDLEUP	Específica que el botón central ha cambiado ha no pulsado.
MOUSEEVENTF_WHEEL	Sólo en Windows NT: especifica que la rueda se ha movido, si el ratón tiene una rueda.
MOUSEEVENTF_ABSOLUTE	Especifica que los parámetros dx y dy contienen coordenadas absolutas normalizadas. Si no está activa, los parámetros contienen datos relativos: el cambio de posición desde la última informada. Esta bandera puede activarse o no, independientemente del tipo de dispositivo de ratón

Los valores que especifican el estado botón del ratón se configuran para indicar cambios en el estado y no las condiciones en curso.

DWORD dx: Especifica la posición absoluta del ratón a lo largo del eje x o su cantidad de movimiento desde el último evento de ratón, dependiendo de la configuración de MOUSEEVENTF_ABSOLUTE. Los datos absolutos se especifican como la coordenada-x actual del ratón.

DWORD dy: Especifica la posición absoluta del ratón a lo largo del eje y o su cantidad de movimiento desde el último evento de ratón, dependiendo de la configuración de MOUSEEVENTF_ABSOLUTE. Los datos absolutos se especifican como la coordenada-y actual del ratón.

DWORD dwData:

- Si dwFlags contiene MOUSEEVENTF_WHEEL, entonces dwData especifica la cantidad de movimiento de la rueda. Un valor positivo indica que la rueda se gira hacia adelante; un valor negativo indica que la rueda se gira hacia atrás. Un clic de rueda se define como WHEEL_DELTA, que toma el valor de 120.
- Si dwFlags contiene MOUSEEVENTF_HWHEEL, entonces dwData especifica la cantidad de movimiento de la rueda. Un valor positivo indica que la rueda se inclina hacia la derecha; un valor negativo indica que la rueda se inclina hacia la izquierda.
- Si dwFlags contiene MOUSEEVENTF_XDOWN o MOUSEEVENTF_XUP, entonces dwData especifica que fueron presionados o liberados los botones X. Este valor puede ser cualquier combinación de las siguientes banderas, ver Tabla 2.3.
- Si dwFlags no contiene MOUSEEVENTF_WHEEL, MOUSEEVENTF_XDOWN, o MOUSEEVENTF_XUP, a continuación, dwData debe ser cero.

Tabla 2.3

Valores de las banderas del botón X

VALOR	SIGNIFICADO
XBUTTON1	Establece si el primer botón X se ha pulsado o liberado.
XBUTTON2	Establecer si el segundo botón X se ha pulsado o liberado.

DWORD dwExtraInfo: Especifica un valor adicional de 32 bits asociado con el evento del ratón. Una aplicación puede llamar a GetMessageExtraInfo para obtener esta información extra.

2.3 Visión artificial

La visión artificial es un campo de la inteligencia artificial que ha estado ganando popularidad en los últimos años debido a las aplicaciones que de ella se han derivado. Actualmente, las aplicaciones de la visión artificial están muy extendidas y van desde el campo de la industria (contar botellas o comprobar defectos en una

cadena de montaje) y el campo de la medicina (recuento y búsqueda de células), hasta los sistemas más complejos, que permiten a los robots orientarse en un entorno desconocido, pasando por el reconocimiento de patrones de la realidad aumentada, entre otras muchas aplicaciones.

Una de las características que vuelven atractiva a la visión artificial es que permite optimizar procesos donde se requiere el uso del ojo humano, de éste modo se ha logrado mejorar la calidad y velocidad de los procesos. Otro factor importante es la alta velocidad de respuesta, lo que hace posible mediciones sin contacto en tiempo real.

El concepto de la palabra visión es: proceso que produce, a partir de las imágenes del mundo exterior, una descripción que es útil para el observador y que no tiene información irrelevante. [6]. Partiendo de ésta definición, la visión artificial cuenta con tres aspectos importantes: La visión es un proceso computacional, la descripción a obtener depende del observador y respecto a la reducción de información; y es necesario eliminar la información que no sea útil. Entonces se puede definir la "Visión Artificial" como un campo de la "Inteligencia Artificial" que, mediante la utilización de las técnicas adecuadas permite la obtención, procesamiento y análisis de cualquier tipo de información especial obtenida a través de imágenes digitales.

2.3.1 Formación de imágenes

Una imagen es una representación visual de un objeto iluminado por una fuente radiante. Las que se perciben en las actividades visuales cotidianas provienen por lo general de la luz reflejada por los objetos. La naturaleza básica de una imagen se caracteriza por dos componentes que son iluminación y reflactancia. Iluminación es la cantidad de luz incidente que procede de la fuente de la escena contemplada; notándose como i(x,y). Reflactancia es la cantidad de luz reflejada por los objetos de la escena; notándose como r(x,y). Ambas se combinan como producto para dar como resultado la imagen y notándose como f(x,y). [7]

En el proceso de formación de imágenes intervienen los siguientes elementos: objeto, fuente radiante y sistema de formación de la imagen; ésta última consiste,

básicamente, en un sistema óptico, un sensor y un digitalizador. La imagen digital que se forma puede ser representada por una matriz f de dimensiones NxM de la siguiente forma:

$$f = \begin{bmatrix} f(1,1) & \cdots & f(1,M) \\ \vdots & \ddots & \vdots \\ f(N,1) & \cdots & f(N,M) \end{bmatrix} \tag{2.1}$$

Donde cada elemento o pixel da la intensidad de la imagen en ese punto, (Véase Figura 2.3; una representación matricial, de una parte de la imagen, de dimensiones 5x5.

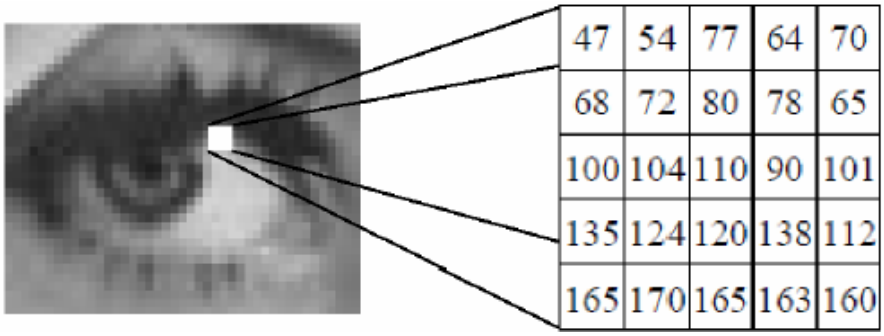


Figura 2.3 Representación matricial de una imagen [7]

2.3.2 Procesamiento de imágenes

El procesamiento de imágenes tiene como objetivo mejorar el aspecto de las imágenes y hacer más evidentes en ellas ciertos detalles que se desean hacer notar.

El paso posterior a haber obtenido la imagen, a través de la cámara, es realizar el procesamiento de ésta; de modo que se pueda utilizar detalles o áreas útiles para el presente proyecto. Las técnicas a usarse se detallan a continuación.

2.3.2.1 Segmentación de imágenes

La segmentación es uno de los pasos para analizar una imagen consiste en separar los distintos objetos, que aparecen en la misma, del fondo de la imagen; en nuestro caso dichos objetos serán áreas con significado que representan el primer paso del procesamiento de imágenes y el más importante ya que todo el proceso dependerá del nivel de robustez de la segmentación. [8]

Existe una gran variedad de técnicas de segmentación, que se adaptan al tipo de imágenes y al objetivo que se persigue, entre las cuales se distinguen básicamente:

- Umbralización: los píxeles se clasifican atendiendo únicamente a su nivel de gris.
- Detección de regiones: los píxeles con características similares se agrupan en regiones, siendo cada región un objeto diferente.
- Detección de fronteras: se realiza una búsqueda de los píxeles que corresponden a fronteras que separan dos objetos diferentes. Una vez detectados, la segmentación es sencilla.

Segmentación mediante Umbrales: Esta técnica se basa en determinar ciertos umbrales de forma que delimiten la imagen en zonas, en lo que a valores de intensidad se refiere. Posteriormente, dependiendo del nivel de gris de cada píxel, éste se clasificará en uno u otro objeto. Una herramienta muy útil para las técnicas de umbralización es el estudio del histograma. De esta forma, si r representa el valor umbral considerado y f(x,y) el nivel de gris para el punto (x,y), la imagen resultante tras la segmentación se define como:

$$g(x,y) = \begin{cases} 1 & si\ f(x,y) > T \\ 0 & si\ f(x,y) \leq T \end{cases} \tag{2.2}$$

Se muestra un ejemplo de segmentación típico mediante umbralización, imagen a) a escala de grises; b) con umbral bajo; c) con umbral alto; d) con umbral intermedio. (Véase Figura 2.4)

Segmentación basada en Detección de Regiones: La segmentación orientada a las regiones tienen su base en las reglas de similitud y en la conectividad de los píxeles. Las regiones se forman mediante píxeles que tengan conectividad y presenten alguna propiedad de similitud y discrepancia respecto al resto de los píxeles que no pertenecen a la región (Véase Figura 2.5).

La técnica a experimentar se basa en el crecimiento de regiones. Se elige un píxel semilla de la región a obtener y se les aplica a sus vecinos la regla de similitud, es decir aquellos píxeles que cumplan se añadirán a la región creciente. La regla de

similitud empleada se basa en que la diferencia del nivel de gris del píxel a estudiar y el brillo de la región creciente sea menor a un determinado umbral.

Figura 2.4 Ejemplo de segmentación típico mediante umbralización [9]

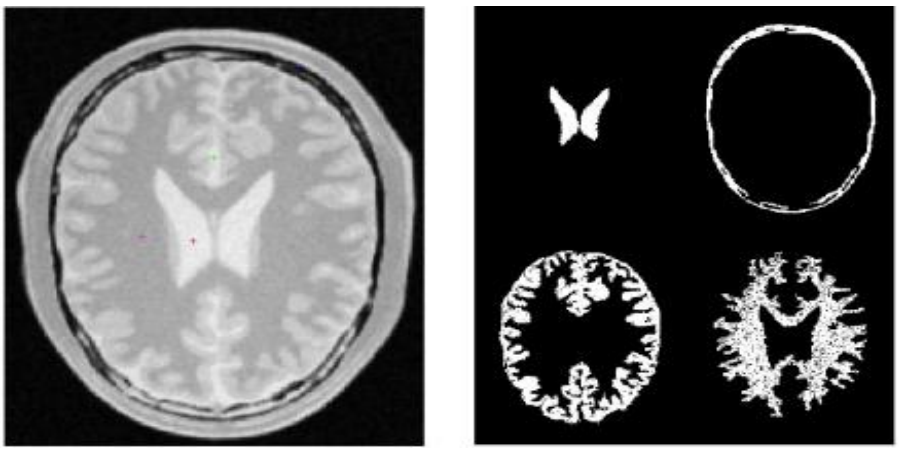


Figura 2.5 Segmentación de la imagen basada en detección de regiones [10]

Segmentación basada en Detección de Fronteras: Los bordes de una imagen digital se pueden definir como transiciones entre dos regiones de niveles de gris significativamente distintos, éstos suministran una valiosa información sobre las fronteras de los objetos y puede ser utilizada para segmentar la imagen, reconocer objetos, etc.

La mayoría de las técnicas para detectar bordes emplean operadores locales basados en distintas aproximaciones discretas de la primera y segunda derivada de los niveles de grises de la imagen.

2.3.2.2 Morfología

Las tareas de segmentación de imágenes no suelen dar un resultado exacto de la delimitación de los objetos o regiones de interés, aparecen píxeles mal clasificados, bordes imprecisos de los objetos o regiones que están solapadas. Por tanto, antes de extraer más características de medio nivel se requiere de una etapa de post-procesamiento, a la cual se la denomina como morfología. Ésta técnica de procesamiento no lineal de la señal, está caracterizada en realzar la geometría y forma de los objetos. De forma general, para realizar la morfología de una imagen es necesario realizar una erosión y dilatación, que se describen a continuación:

Erosión: Es una transformación anti-extensiva, su utilidad consiste en definir una geometría determinada al elemento estructurante y pasarlo sobre la imagen, en donde los objetos menores al elemento estructurante no aparecerán en la imagen resultante; se observa un ejemplo de cómo cambia una imagen mediante la erosión. (Véase Figura 2.6).

Figura 2.6 Ejemplo de proceso de erosión binaria [10]

Dilatación: Es la transformación dual a la erosión. Ésta técnica es el valor máximo del entorno de vecindad definido por el elemento estructurante, las aplicaciones de las operaciones de erosión seguida con una dilatación no son conmutativas, ya que los resultados son diferentes dando paso a las aperturas y cierres morfológicos. La Figura 2.7 representa una imagen a la cual se le aplica binarización y luego dilatación.

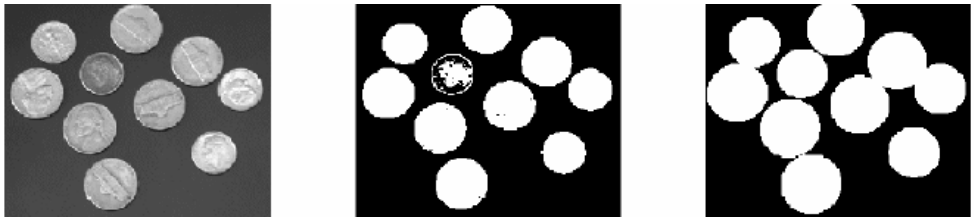


Figura 2.7 Ejemplo de proceso de dilatación binaria [10]

2.3.2.3 Procesamiento de histograma

Ésta técnica permite mejorar la calidad de una imagen. El histograma es la base de numerosas técnicas de procesamiento de la imagen en el dominio espacial, además la información inherente en el histograma también puede ser utilizada en otras aplicaciones útiles como compresión de imágenes y segmentación; los histogramas son muy fáciles de calcular por lo que es una herramienta candidata a ser calculada en tiempo real. Existen diversos métodos para manipular histogramas, entre ellos la ecualización y la especificación del histograma; en ambos métodos los pixeles se modifican utilizando una función de transformación que se basa en la distribución de los niveles de gris en toda la imagen [11].

2.4 OpenCV

OpenCV es una biblioteca de visión artificial desarrollada en el año 1999, liberada bajo la licencia BSD y es libre para usos académicos y comerciales. Es utilizado para crear aplicaciones donde la seguridad y el control de procesos son primordiales. OpenCV aprovecha las capacidades de los procesadores multi-núcleo para permitir una programación optimizada en lenguajes como C o C++. OpenCV es popular en el mundo de los desarrolladores ya que 47 mil personas son usuarios de

sus beneficios y se estima que el número de descargas es de 7 millones. Su logo se puede apreciar en la Figura 2.8

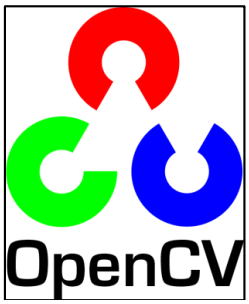


Figura 2.8 Logo de OpenCV [12]

2.4.1 Características

OpenCV cuenta con características que ayudan al programador a crear aplicaciones multiplataforma y en tiempo real, a continuación las más importantes:

- Soporta interfaces como C, C++, Python y Java
- Soporta plataformas como Windows, Linux, Mac OS, iOS y Android.
- Diseñado para eficiencia computacional y en tiempo real.
- Tiene funciones de captura y presentación de imágenes.
- Utiliza operaciones con matrices para el procesamiento de imágenes.

2.4.2 Arquitectura

En la versión 2.2 de OpenCV los creadores subdividieron las funcionalidades y obteniendo así una arquitectura de diseño que muestra así una relación entre las librerías de imágenes y video. [13] Las funcionalidades de los módulos principales son las siguientes:

- **Core:** Es la definición de las clases y templates.
- **Imgproc:** Es el procesado de imagen.
- **Highgui:** Permite la E/S de las imágenes y videos.
- **Ml:** Es la máquina que permite el conocimiento.
- **Calib3d:** Calibración de imagen y reconstrucción 3D.
- **Features2d:** Características de las imágenes 2D.

- **Objdetect:** Detección de objetos.
- **Video:** Análisis de los videos.

2.5 Estudios previos

Existen cuatro estudios básicos de medición de movimiento ocular recopilados: a) electrooculograma (EOG), b) lentes de contacto, c) foto o video oculograma; y d) detección de video basada en pupila y reflexión cornea; cada una de éstos tiene sus particularidades. [14]

A continuación se exponen dichos estudios básicos, según Turégano (2006), y sus principales particularidades:

La electro-oculograma es una técnica para medir el potencial de reposo de la retina, para medir los movimientos del ojo se usa cuatro electrodos colocados en los cuatro puntos cardinales del ojo; éstos captan el movimiento de la pupila hacia cualquiera de los electrodos mediante una diferencia de potencial con el cual se logra distinguir la posición. Ésta técnica mide la posición relativa del ojo respecto a la cabeza, más no a lo que esté mirando.

La técnica de lentes de contacto es una de las más precisas para medir el movimiento del ojo, consiste en un lente de contacto con un pequeño hilo de metal en él; éste hilo está enrollado y mide las variaciones de campo magnético; al igual que la electro-oculograma mide la posición del ojo respecto a la cabeza y no de lo que se está observando.

La técnica de video-oculograma o foto-oculograma representa una gran variedad de técnicas bajo ésta denominación, las cuales se encargan de extraer las características de los ojos, entre ellas: rotación y traslación, es decir la forma de la pupila, unión entre el iris y la esclerótica, reflexiones córneas, etc. Sin embargo ninguna de éstas características proporciona la posición del ojo respecto hacia donde está mirando.

La detección por video basada en la pupila y reflexión córnea, permite medir la posición del ojo respecto de donde se está mirando, para poder medir dicha posición

se necesita que la cabeza permanezca estable, de modo que la posición del ojo pueda coincidir con el punto de interés.

2.6 Cámara web

2.6.1 Definición y generalidades

Definición: Es un periférico de entrada usada para capturar imágenes y videos con la finalidad de almacenarlos, editarlos o transmitirlos a otro equipo o red. [15]

Generalidades: Entre las principales y las de incumbencia para el presente proyecto se encuentran:

- Generalmente usadas en servicio de mensajería instantánea y videoconferencias.
- Se conecta al PC, generalmente, mediante el puerto USB; puede ser inalámbricamente (WiFi, zig-bee o bluetooth).
- La resolución de imagen se mide en megapíxeles (mpx), esto es resultado de sumar todos los píxeles que es capaz de capturar en una imagen.
- La resolución de video se mide en frames por segundo (fps), los frames son imágenes capturadas; los fps dependen en gran medida de la resolución de la cámara web

2.6.2 La cámara web como sensor

De acuerdo a lo expuesto en el ítem 2.5, se pudo contemplar que para la realización del presente proyecto, la técnica apta es la de detección por video basada en pupila y reflexión córnea, pues a diferencia de las demás permite medir el punto de interés, es decir el lugar exacto a donde el ojo está mirando.

Respecto a ésta técnica cabe acotar que existen dos tipos de seguidores basados en luz infrarroja: los de pupila oscura y los de brillante; cuando el haz de luz ilumina el ojo es coaxial al camino de visión, el ojo actúa como retro-reflector produciendo una pupila brillante en la captura de la misma; si existe un desplazamiento entre la luz y el camino de visión, la pupila será capturada con un color muy oscuro generando una pupila oscura.

Todo lo mencionado anteriormente fue de vital importancia para elegir a la técnica de detección por video basada en pupila como la apropiada para trabajar y la reflexión cornea para la creación de la herramienta. Por lo tanto la cámara web intervendrá como sensor del movimiento del ojo.

CAPÍTULO 3

DISEÑO DE LA HERRAMIENTA

En este capítulo se muestra el proceso para diseñar e implementar la herramienta para ayuda a personas con capacidades especiales. Es de mucha importancia el diseño de la herramienta de ello depende la creación de un buen producto, realizando así varias pruebas que sean satisfactorias.

En la Figura 3.1 se muestra el plan de desarrollo que consta de 6 fases: Descripción de Requerimientos, Elección y modificación de cámara web, Descripción del Uso, Diseño e Implementación. Las pruebas realizadas serán detalladas en el Capítulo 4 del documento.

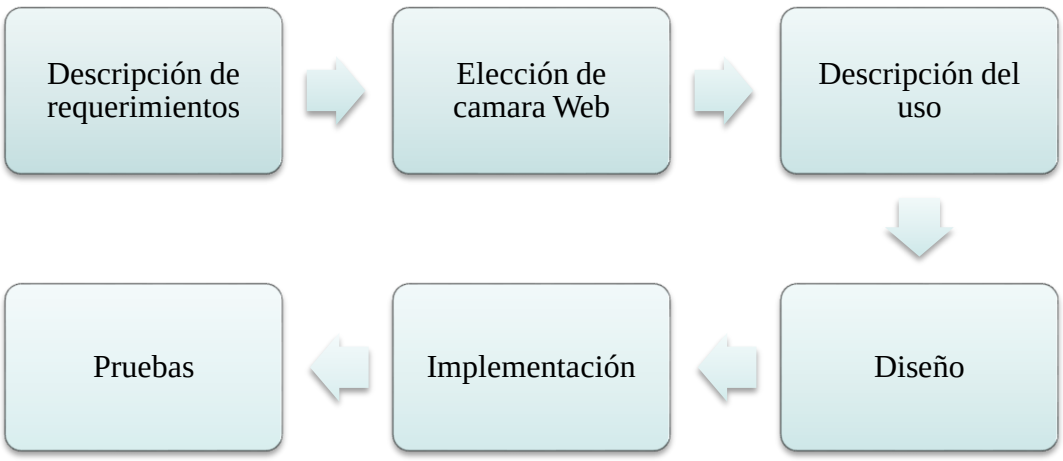


Figura 3.1 Técnica de desarrollo

Tres aspectos importantes a recalcar en el diseño de la herramienta son: que se utiliza una cámara web que permita la captura de imágenes en tiempo real, la iluminación de la zona de imagen donde se usa ocho diodos LED y que esto ayuda a diversas técnicas de procesamiento de imágenes que se efectuará.

3.1 Creación del periférico alternativo

Es de mucha necesidad la creación de un periférico alternativo que sustituya al mouse tradicional del computador, para ello se ha propuesto la creación de un mouse para personas con capacidades especiales sin movilidad motriz, aprovechando así el movimiento de los ojos, buscando la mejor posición del dispositivo para poder capturar la zona de interés, es decir que la región del ojo de interés sea localizada de manera precisa; todo esto mediante una cara web y procesamiento de imágenes mediante un computador y la utilización de la biblioteca OpenCV.

La mayoría de los sistemas de búsqueda de la mirada desde el diseño lógico es un diseño usual, sin embargo hay muchas transiciones en las herramientas utilizadas en estos sistemas. Por esta razón es necesario, primeramente, elegir una cámara web que sea económica y la nitidez de la imagen como es la resolución.

3.1.1 Decisiones del diseño

Comúnmente estos sistemas de la detección de los ojos tienen varios problemas que se deben tener en cuenta para calcular el punto del enfoque o para localizar el ojo. Entre estos tenemos los siguientes:

- Movimiento de la cabeza: es una gran desventaja ya que se requiere que el usuario mantenga la cabeza firme con el fin de captar la exactitud de los datos [16].
- Otro problema es la forma en el que el ojo se mueve, como es común el usuario opta por mover su mirada a donde se muestra interés, por ello el sistema podría terminar actuando sobre comandos que el usuario no tenía la intención de realizar por lo tanto, es inevitable resolver los comandos significativos a partir de observaciones simples, o movimientos no significativos. [17].
- Otra solución es manejar la detección de parpadeo como la base para los comandos de interacción, como los son clic y doble clic.

3.1.2 Elección cámara web

La elección de una adecuada cámara web es la segunda etapa de este proceso, se mostrará a continuación algunas cámaras web con el propósito de elegir el dispositivo que cuente con el requerimiento del proyecto. En la Figura 3.2 se muestra una reseña de características de diferentes cámaras web.

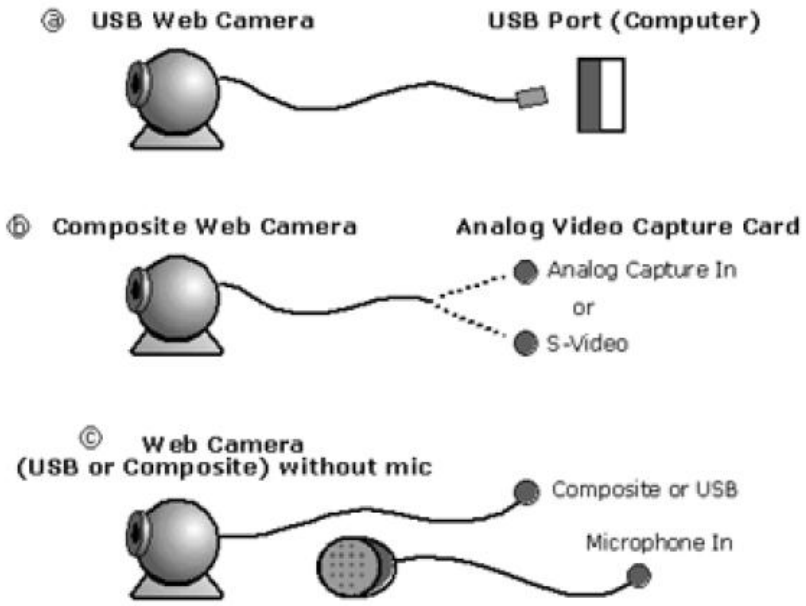


Figura 3.2 Cámara Web con diferentes conectores[18]

3.1.2.1 Cámara webcam

La cámara web utilizado para este proyecto es la cámara web Omega. Esta es una cámara web relativamente barata y, como tal, convenía a los fines del proyecto. La cámara web mencionada puede ser observada en la Figura 3.3.

3.1.2.2 Características

Características:

- Sensor: (CMOS) *Complementary Metal Oxide Semiconductor* los fotones que recibe cada celda son convertidos en carga eléctrica y en voltaje en la misma celda receptora, realiza la digitalización píxel a píxel, ello conlleva que el resultado que entrega a la circuitería de la cámara este ya

digitalizado, en el proyecto se retiró el filtro mencionado y se reemplazó por un filtro infrarrojo en este caso papel fotográfico.

- Grabación de video DivX (empleado software), es un formato de vídeo que funciona sobre los sistemas operativos Windows, MacOS y GNU/Linux actuales y que, combinado con la compresión de audioMP3, consigue una alta calidad de imagen superior a la del VHS con un caudal inferior a 1 Mbit/s. (VHS) *Video Home System* es un popular sistema doméstico de grabación y reproducción analógica de video. [19]
- Video con 30 y 15 fps (frames por segundo)
- Compatible con S.O. Windows XP-SP2/Vista/Win7, por lo que se utilizó en sistema operativo Win7
- Max. resolución dinámica: 5.0M píxeles 320x240, resolución del video, el cual fue considerado para obtener el punto centro y poder realzar la calibración para iniciar la ejecución del herramienta (Ver Anexos).
- Max. resolución estática: 30.0M pixeles 640x480, resolución de las fotos, no se consideró esta característica para el proyecto.

Figura 3.3 Cámara Web

3.1.3 Modificación de la cámara

Debido a varias necesidades para obtener una imagen del ojo con áreas significativas que adquiriesen ser tratadas con el resultado de lograr el movimiento del cursor, ya que existen varios parámetros que hacen que dichas áreas cambien. Como bien se trata de la detección por video establecida en la pupila y reflexión de la córnea para evaluar el movimiento ocular, es necesario realizar una modificación a la cámara y se logre captar la luz infrarroja, pues las cámaras digitales comunes contienen un sensor, que generalmente es un *Complemetary Metal Oxide Semiconductor* (CMOS), el cual detecta toda luz visible tanto para la luz infrarroja como para el ojo humano. Por esta razón la cámara se la transformara en infrarroja, esto se obtiene retirando el filtro que viene en la cámara y reemplazándolo por un filtro infrarrojo este filtro lo que hace es bloquear el espectro de luz visible y deja pasar únicamente luz infrarroja, con esto logramos evitar que la luz infrarroja sature los colores y conciba una imagen inexistente.

Como bien es cierto existen variedad de filtros, la desventaja es los cotos que son elevados, y con ello no se lograra economizar el proyecto, teniendo este antecedente se ha optado por conseguir algo económico, como es un negativo fotográfico, el cual permite tener una imagen de buena calidad y además es fácil de conseguir.

A continuación se procederá explicar el proceso de modificación de la cámara para transformarla en infrarroja, cabe acotar que para la realización del presente proyecto se utilizó un total de cinco cámaras.

1. Retiramos los tornillos y abrimos la cámara.
2. Desmontar con precaución la parte frontal hasta mirar el filtro CMOS o CCD despendiendo de la cámara web. (Véase Figura 3.4)

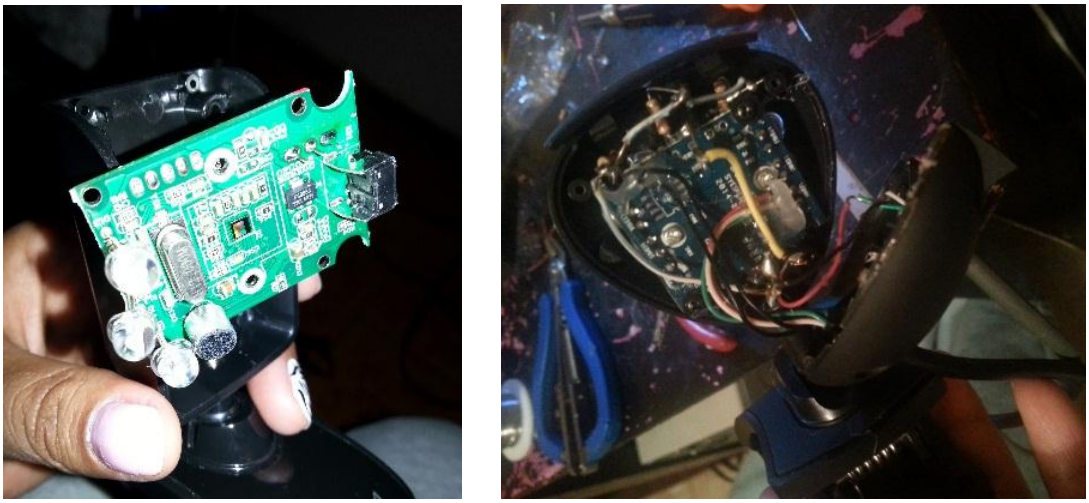


Figura 3.4 Apertura de la cámara

3. Localizar el lente del sensor de luz y se procede a retirar el lente donde se encuentra ubicado el filtro, como se observa en la Figura 3.5.

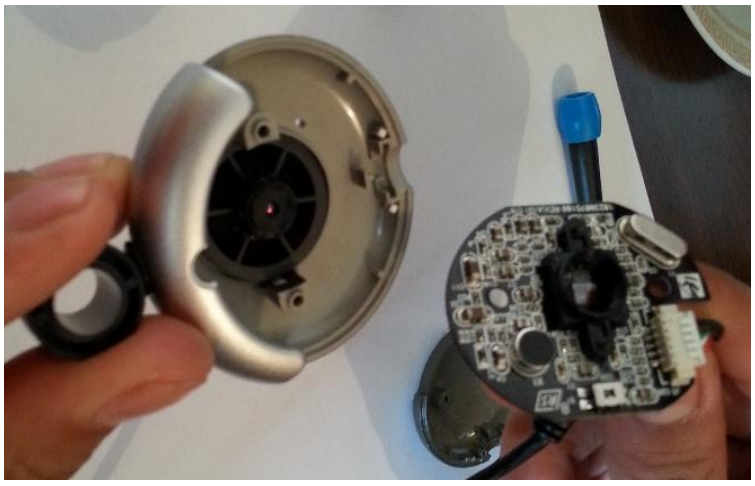


Figura 3.5 Lente y sensor de luz

4. El filtro se encuentra ubicado dentro del lente, primero retirar lente y luego el filtro (similar a un cristal) (Véase Figura 3.6).

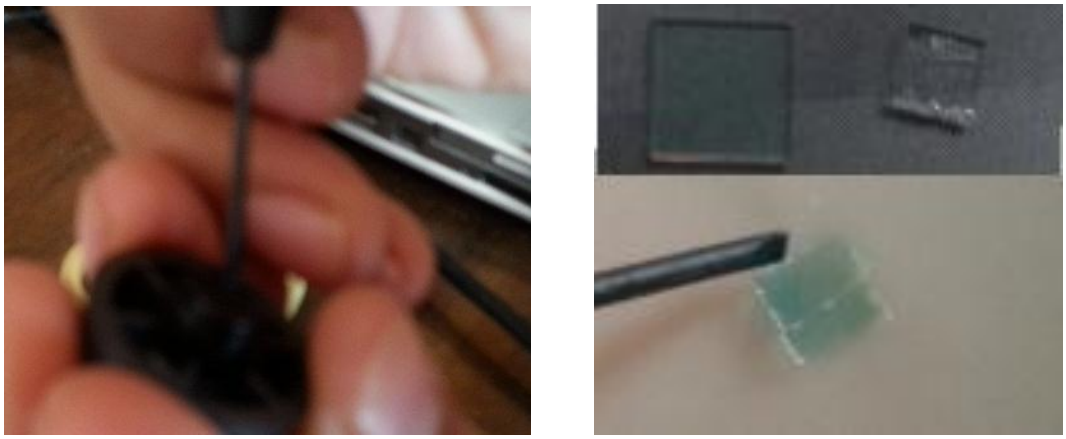


Figura 3.6 Retiro de filtro

5. El negativo fotográfico debe sustituir al cristal retirado, para este proceso se debe obtener dicho negativo fotográfico, previamente, listo, como se puede apreciar en la Figura 3.7.

Figura 3.7 Colocación del negativo fotográfico

6. Posteriormente se realizó el re ensamblaje de la cámara y se puede apreciar el resultado de la modificación, ver Figura 3.8.

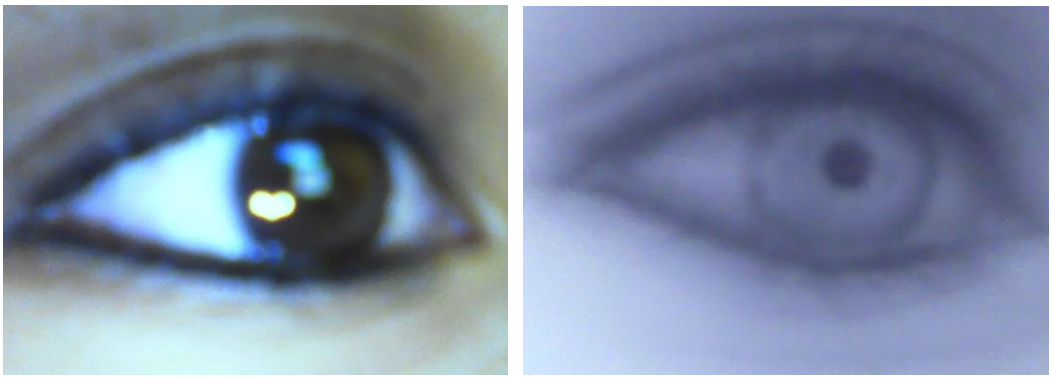


Figura 3.8 Resultado de la modificación de la Cámara web

3.1.4 Iluminación de la imagen

La iluminación de la imagen en uno de los puntos más críticos del proyecto, ya que ésta es muy dependiente del factor ambiente para la detección de la imagen del ojo, es notorio que se tendrá varios escenarios dependiendo si se usa la herramienta durante el día o la noche. Por todo lo antes mencionado se procedió con la construcción de un dispositivo para iluminar la zona del ojo y no depender del factor ambiente. Se optó por utilizar un LED emisor de infrarrojos.

El diodo (IRLED) del inglés *lnfrared Light Emitting Diode*, es un emisor de rayos infrarrojos que son una radiación electromagnética situada en el espectro electromagnético, en el intervalo que va desde la luz visible a las microondas. Estos diodos se diferencian de los LED por el color de la cápsula que los envuelve que es de color azul o gris. El diámetro de ésta es generalmente de 5 mm. [20]

Se acopló varios LED's emisores de infrarrojo en la cámara con la cual se está trabajando, a éstos se les proporciona energía a través de un cable (USB) *Universal Serial Bus* conectado al computador, proporcionando a los diodos LED 5 Voltios, necesarios para encenderlos. Además se integró una resistencia para evitar que se queme el LED.

Con todos estos parámetros se consiguió un correcto funcionamiento de los diodos LED y una iluminación adecuada para la realización del proyecto. En la Figura 3.9 se puede observar el esquema de conexión para el diodo LED y en la Figura 3.10 como éste fue acoplado a la cámara.

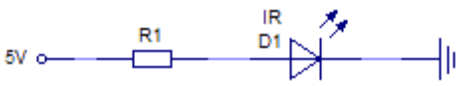


Figura 3.9 Esquema de conexión para el Diodo LED.

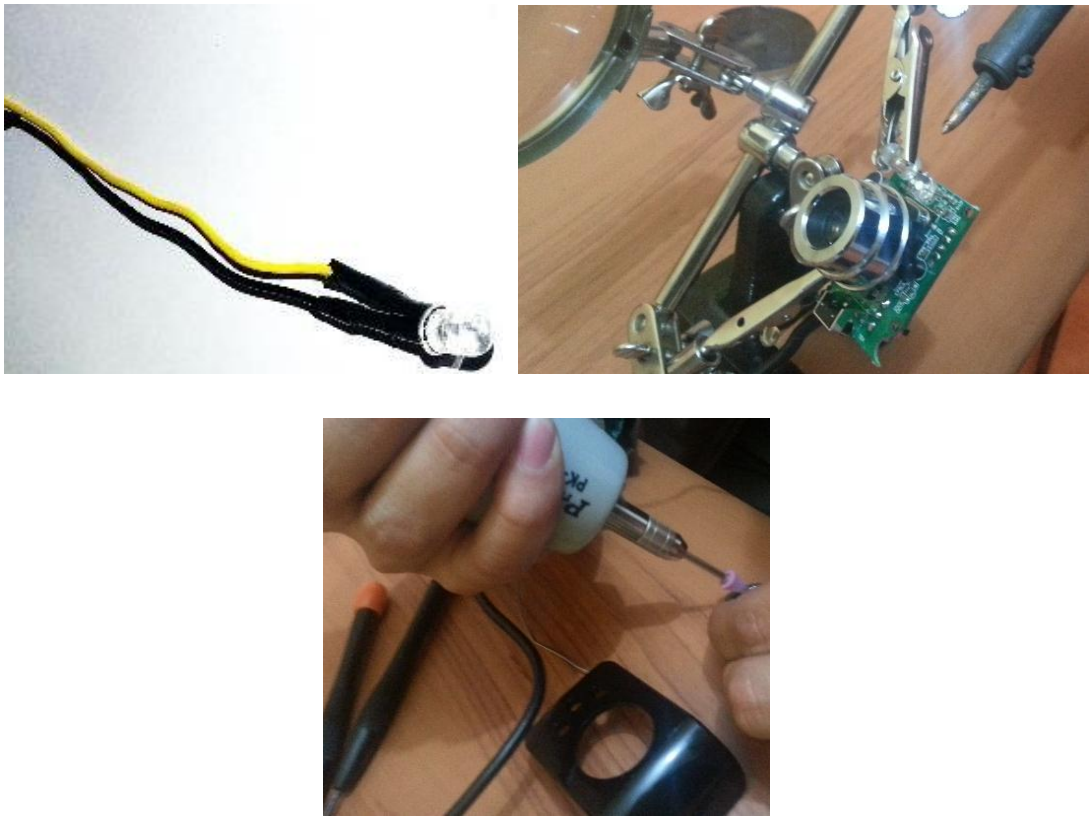


Figura 3.10 Acoplamiento del Diodo LED a la cámara

3.1.5 Ensamblaje del periférico

Para el ensamblaje del periférico tomar en cuenta la comodidad del usuario así como la posición del dispositivo para un correcto funcionamiento de la herramienta, de modo que enfoque la región de interés, se procedió a ubicar a la cámara web en la visera de una gorra, logrando mantenerla fija y considerando que esta posición este direccionado a la región del ojo. Ver Figura 3.11.

Figura 3.11 Ensamblaje del periférico

3.2 Lenguaje de programación

La idea inicial del proyecto fue realizarlo sobre un software especializado que diera las facilidades para la realización del proyecto, por éste motivo se trabajó a la par con dos software especializados; MATLAB y OpenCV (Open Source Computer Vision Library). Debido a circunstancias que se detallan a continuación se optó por trabajar con el segundo.

Tomando en cuenta que la herramienta requiere procesamiento en tiempo real para su ejecución, de manera que el cursor del mouse se mueva coordinadamente con el movimiento ocular del usuario, con los mejores tiempos de respuesta posible, resulta necesario utilizar un lenguaje de programación de bajo nivel OpenCV, sin lugar a dudas es la más conocida en lo que a visión artificial respecta, pues contiene más de quinientos algoritmos entre los que se incluye funciones de propósito general para procesamiento de imágenes, descripciones geométricas, segmentación, seguimiento, etc. Además está diseñada para el tratado de imágenes en tiempo real; y es mucho más rápido al momento de compilar el código.

De éste modo se determinó que OpenCV es el software especializado necesario para el desarrollo del proyecto, y con ella el lenguaje de programación C++, C, utilizado sobre Microsoft Visual Studio.

Ahora que ya se tiene el periférico de la herramienta, se procede a desarrollar el software para la funcionalidad del proyecto.

3.3 Algoritmo general

El algoritmo general se divide en tres partes: el bucle continúo, para conseguir la mirada; programa principal; y movimiento del mouse.

3.3.1 Pasos de bucle

Los pasos de bucle son necesarios e importantes, para tener toda la información consolidada y no tener inconvenientes al momento de compilar el código, En la figura que se muestra a continuación se puede apreciar los pasos:

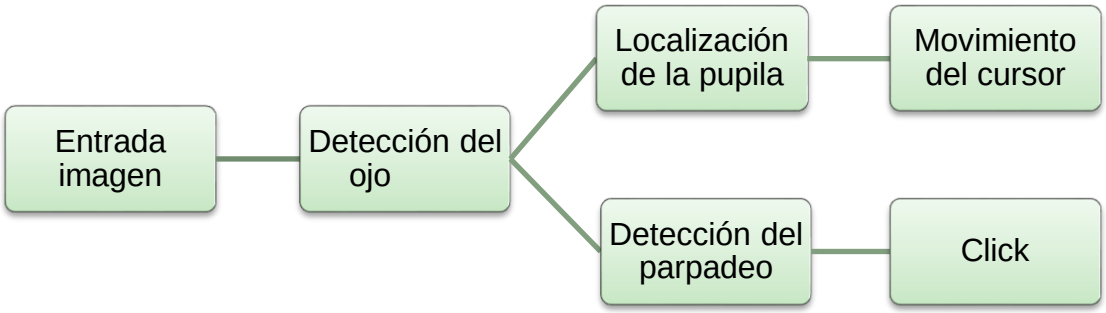


Figura 3.12 Diseño lógico

En la actualidad, todos estos procesos se pueden utilizar con eficacia para el seguimiento de la mirada y un buen funcionamiento de la herramienta.

Con una elección de una webcam con la mejor calidad, es decir tener la mejor nitidez y resolución, también con los principales procesos de programación que existe hoy en día, logrando el funcionamiento de la herramienta con seguir ciertos pasos de procesamiento, de una imagen capturada con un dispositivo de grabación de video.

CAPÍTULO 4

IMPLEMENTACIÓN DE LA HERRAMIENTA

Una vez realizadas las respectivas modificaciones a la cámara web, la creación del periférico alternativo; y de haber analizado los requerimientos para el diseño de la herramienta, se procede a la implementación de la misma.

Para el proceso de implementación de la herramienta se utilizó el sistema operativo *Microsoft Windows 7*, usando el software *Microsoft Visual Studio 2010* y como lenguaje base *C++ y C*; se usó la librería *OpenCV*, con ella el procesamiento en tiempo real de las imágenes que ingresaron y el procesamiento de las funciones de Visión Artificial. Además se usó funciones API *Win32 para Windows 7* para poder trabajar con los eventos del ratón. En adición, se creó funciones, donde se realizó tratamiento de imágenes para lograr un óptimo funcionamiento de la herramienta.

A continuación se detalla las funciones y procedimientos utilizados en el desarrollo de la herramienta.

4.1 Definición de funciones y procedimientos

4.1.1 Metodología de las funciones de visión artificial

4.1.1.1 Morfología

OpenCV proporciona una cómoda interfaz rápida para hacer transformaciones morfológicas. Las transformaciones morfológicas básicas se llaman dilatación y la erosión, y que se presenten en una amplia variedad de contextos, tales como la eliminación de ruido, aislamiento de elementos individuales, y uniéndose a elementos diversos en una imagen. [21]

Se manejara por la función `cvMorphologyEx()`.

```
void cvMorphologyEx(
const CvArr*  src,
CvArr*        dst,
CvArr*        temp,
IplConvKernel*  element,
```

```
int           operation,
int           iterations = 1
);
```

Parámetros:

- `src` - imagen de entrada.
- `dst` – imagen de salida del mismo tamaño y el mismo número de canales.
- `IplConvKernel` es una máscara de convolución o filtro de coeficientes enteros o caracteres que está definida en IPL como un tipo específico. Este argumento es nulo por defecto, lo cual significa que el núcleo es 3x3.

Además de los argumentos src, dst, elemento, e iteraciones, que utilizamos con los operadores anteriores, `cvMorphologyEx()` tiene dos parámetros nuevos. La primera es la matriz temporal, que se requiere para algunas de las operaciones ver en el siguiente cuadro. Cuando sea necesario, esta matriz debe ser del mismo tamaño que la imagen de origen. El segundo argumento es la operación morfológica que vamos hacer. [21]

Tabla 4.1

`cvMorphology()` opciones de operación

Operación	Operador Morfológico	Requiere imagen Temp
CV_MOP_OPEN	Opening	No
CV_MOP_CLOSE	Closing	No
CV_MOP_GRADIENT	Morphological gradient	Siempre
CV_MOP_TOPHAT	Top Hat	For in-place only (src = dst)
CV_MOP_BLACKHAT	Black Hat	For in-place only (src = dst)

Las dos primeras operaciones de la Tabla 4.1, apertura y cierre, son combinaciones de los operadores de erosión y dilatación. En este caso se trabajó con `CV_MOP_OPEN`, ya que es utilizada para representar las regiones en una imagen binaria, la apertura primero se realiza la erosión y luego dilatación, el efecto es eliminar valores distintos que son más altos que sus vecinos (Véase Figura 4.1).

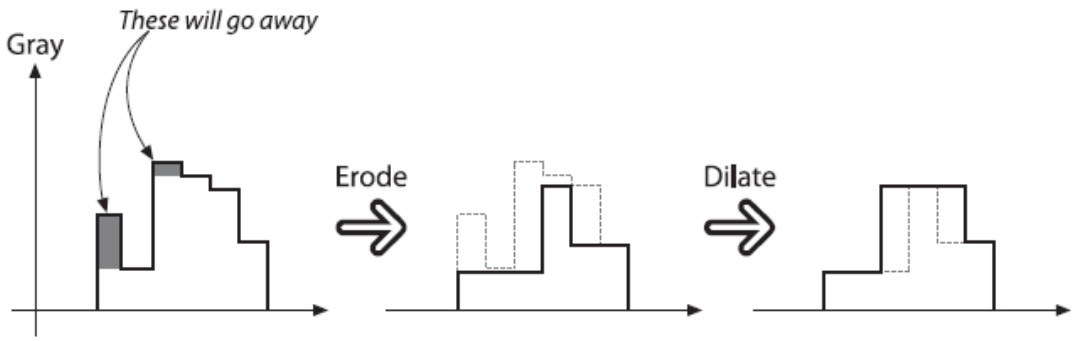


Figura 4.1 Operación morfológica de apertura [22]

Aunque el resultado final de la utilización de apertura o cierre es similar al uso de erosionar o dilatar, estas nuevas operaciones tienden a preservar el área de las regiones conectadas con mayor precisión. [23]

`IplConvKernel` en una máscara de atenuación o núcleo de coeficientes enteros o caracteres definidos en IPL como un tipo específico.

Para no estar limitados al núcleo sencillo de 3 por 3 cuadrados. Se puede realizar kernels personalizados morfológicamente utilizando `IplConvKernel`. Estos núcleos se distribuyen utilizando cvCreateStructuringElementEx() y se liberan utilizando `cvReleaseStructuringElement()`.Un núcleo morfológico, a diferencia de un kernel de convolución, no requiere ningún valor numérico. [23]

```
IplConvKernel* cvCreateStructuringElementEx(
int cols,
int rows,
int anchor_x,
int anchor_y,
int shape,
int* values=NULL
);
```

Parámetros:

- `Cols`: columna del elemento estructurante
- `Rows`: fila del elemento esctructurante
- `Anchor y / Anchor x`: coordenadas (X, Y) donde está situado el elemento.

- `Shape:` la forma que tiene el núcleo en este caso se usó CV_SHAPE_CROSS un elemento estructurante en forma de cruz. [23]

4.1.1.2 Umbralización

Esta operación transforma la imagen a solo dos colores blanco y negro. El proceso para binarizar la imagen es el siguiente:

- Se encuentra la mediana (dato de en medio) de los valores permitidos para cada píxel.
- Se toma uno por uno los píxeles de la imagen.
- Se compara cada píxel: si el valor del píxel es menor a la mediana entonces ese píxel toma el valor mínimo permitido, caso contrario toma el valor máximo permitido. [24]

Para realizar la umbralización OpenCV utiliza la función `cvThreshold`

```
double cvThreshold(
CvArr* src,
CvArr* dst,
double threshold,
double max_value,
int threshold_type
);
```

Donde `CvArr` es un estructura cuyos parámetros hacen que sea equivalente a pasar la estructura `IplImage`, ya que esta se deriva de `CvMat,` que a su vez se deriva de `CvArr.` Tenemos `threshold` es un variables flotante que representa el valor máximo (es 255 generalmente) que se usa en los tipos de umbral mencionados en el cuadro 5. Y `threshold_type` representa el tipo de umbral a utilizar.

Como se muestra en la tabla siguiente, cada tipo de umbral corresponde a una operación de comparación en particular entre el pixel i-ésima fuente (scr_i) y el umbral (indicado en la tabla por T). Dependiendo de la relación entre el píxel de origen y el umbral, el pixel destino dst_i, puede ser fijado a 0, el scr_i, o la *max_value* (indicado en la tabla por M). [25]

Tabla 4.2

`cvThreshold()` threshold_type optio

TIPO DE UMBRAL	OPERACIÓN
CV_TRESH_BINARY	$dst_i = (scr_i > T)?\, M : 0$
CV_TRESH_BINARY_INV	$dst_i = (scr_i > T)?\, 0 : M$
CV_TRESH_TRUNC	$dst_i = (scr_i > T)?\, M : scr_i$
CV_TRESH_TOZERO_INV	$dst_i = (scr_i > T)?\, 0 : scr_i$
CV_TRESH_TOZERO	$dst_i = (scr_i > T)?\, scr_i : 0$

En el proyecto a presentarse se utiliza `CV_THRESH_BINARY` para obtener dos niveles de imagen en escala de grises o para eliminar un ruido, es decir el filtrado de pixeles con valores demasiado pequeños o demasiado grandes. Los resultados antes mencionados se los puede observar en la Figura 4.2, conjuntamente con el modo de operación de la función.

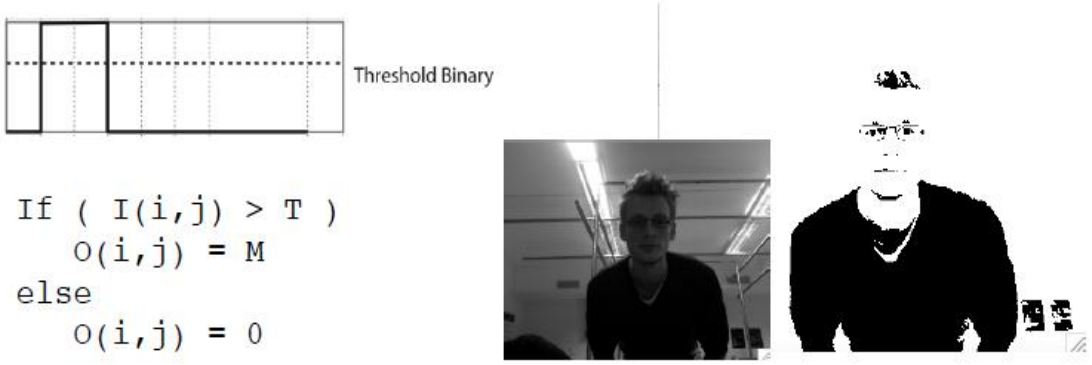


Figura 4.2 Threshold Binary [26]

4.1.2 Funciones y procedimientos de opencv

A continuación se explican las funciones y procedimientos de OpenCV asociados con la implementación de la herramienta que se utilizaron para el tratamiento de imágenes.

La función `cvtColor()`, para convertir una imagen a escalas de grises e invertirlas, requiere de tres parámetros las dos primeras son las imágenes que deben

ser del mismo número de canales y el tercero es el parámetro que representa es tipo de conversión. [3]

```
void cvCvtColor(
   const CvArr* src,
   CvArr* dst,
   int code
);
```

Parámetros:

- `Src y dst` son variables de tipo apuntador a `CvArr` que representa la imagen fuente y la imagen destino
- `code` – Representa el tipo de conversión que se desea realizar, entre los cuales se utilizó: `CV_RGB2GRAY` y `CV_BGR2GRAY`, que transforma una imagen de colores a escala de grises; `CV_GRAY2BGR`, que transforma una imagen de escala de grises a colores.

```
void cvZero (CvArr* arr);
```

Este procedimiento recibe un parámetro de tipo apuntador a `CvArr`, el cual representa una imagen. `cvZero` establece todos los valores de todos los canales de una matriz a 0 (color negro). [3]

```
void cvSub (
const CvArr* src1,
const CvArr* src2,
CvArr* dst,
const CvArr* mask = NULL
);
```

El procedimiento `cvSub()` recibe cuatro argumentos, su función es la de restar los dos primeros argumentos, para retornar un resultado en el tercer argumento. La variable mask es de tipo apuntador a y constituye una imagen para almacenar una máscara, por lo general éste última se la deja en cero; para efectos del proyecto se tomará como nulo NULL. Considerar que `src1, src2` y `dst` debe tener el mismo tipo, tamaño y número de canales; máscara, si se usa, debe ser una matriz de 8 bits del mismo tamaño y número de canales como dst. [25]

```
void cvRectangle(
CvArr* array,
CvPoint pt1,
CvPoint pt2,
CvScalar color,
int thickness = 1,
);
```

El procedimiento `cvRectangle` se usa para dibujar un rectángulo en la imagen, éste recibe cinco argumentos array es de tipo apuntador a y hace referencia a la imagen fuente `CvArr*`. Los dos siguientes argumentos `pt1` y `pt2` hacen referencia a las coordenadas `(x,y)` de la esquina superior izquierda y la inferior derecha del rectángulo a dibujar; es decir son las que dan el tamaño al rectángulo. El siguiente argumento, color, representa el color de la línea a ser dibujada; se hace uso de `CV_RBG (R, G, B)`; rojo verde y azul respectivamente; para definir el color. El parámetro thickness indica el grosor con el que se desea trabajar, por defecto se trabaja con 1. [3]

```
int cvWaitKey(int time);
```

La función `cvWaitKey()` es usada para que el programa se detenga y espere el tiempo que viene dado por el parámetro time; si time es cero o negativo el tiempo de espera es indefinido hasta que se presione cualquier tecla, retornando el valor de dicha tecla. [21]

```
Void cvShowImage(
const char* name,
const CvArr* image
);
```

El procedimiento `cvShowImage()` es usado para mostrar una imagen en una determinada ventana. El primer argumento, name, es de tipo constante apuntador a carácter y especifica el nombre de la ventana en el cual se mostrará la imagen. El argumento `image` es de tipo constante apuntador a `CvArr,` con el cual se indica la imagen a mostrar. [21]

OpenCV utiliza una entidad denominada una memoria de almacenamiento como su método de manipulación de asignación de memoria para objetos dinámicos (por lo

general esto incluye cualquier función cuya salida es de tamaño variable) se utiliza función de `cvCreateMemStorage()`. [21]

Almacenamientos de memoria se manejan con los siguientes cuatro rutinas:

- La función `cvReleaseMemStorage()` toma un puntero a un almacenamiento de memoria válida y luego re-asigna el almacenamiento.
- Puede vaciar un almacenamiento de memoria llamando `cvClearMemStorage()`, que también tiene un puntero a un almacenamiento válido. Usted debe ser consciente de una característica importante de esta función: es la única manera de liberar (y, posteriormente, reutilizar) memoria asignada a un dispositivo de almacenamiento de memoria.
- `cvReleaseMemStorage()` reciclar la memoria de almacenamiento. Eliminación de cualquier estructura dinámica (`CvSeq`, `CvSet`, etc.) nunca regresa de nuevo a cualquier memoria de almacenamiento (aunque las estructuras son capaces de reutilizar parte de la memoria una vez extraído del almacenamiento para sus propios datos). [21]

```
int cvNamedWindow(
const char* name,
int flags = CV_WINDOW_AUTOSIZE
);
```

La función `cvNamedWindow()` crea una venta donde se muestra una imagen, para ello recibe dos argumentos. El primero de ellos es una variable de tipo constante apuntador a carácter llamada `name`, que representa el nombre con el cual se identificará a la ventana. El siguiente argumento es una bandera, `flags`, que permite autoajustar la ventana a la imagen que se está mostrando, usando el valor de 1; si se deja el valor en 0, esto le permite al usuario decidir el ajuste de ventana [3]. Las banderas soportadas son:

`WINDOW_NORMAL` Si se establece, el usuario puede cambiar el tamaño de la ventana (sin limitación).

`WINDOW_AUTOSIZE` Si se establece, el tamaño de la ventana se ajusta automáticamente para adaptarse a la imagen mostrada (ver `imshow ()`), y no se puede cambiar el tamaño de la ventana manualmente, esta bandera se usó para la herramienta.

`WINDOW_OPENGL` Si se establece, la ventana se creará con soporte OpenGL.

```
void imshow(const string& winname,
InputArray mat);
```

Parámetros:

- `winname` - Nombre de la ventana.
- `image` – Imagen para mostrar.

La función `imshow` muestra una imagen en la ventana especificada. Si la ventana se creó con la bandera `CV_WINDOW_AUTOSIZE`, la imagen se muestra con su tamaño original, sin embargo, todavía está limitada por la resolución de la pantalla. De lo contrario, la imagen se escala para ajustarse a la ventana [3]. La función puede escalar la imagen, dependiendo de su profundidad:

Si la imagen es de 8 bits sin signo, que se muestra como es.

Si la imagen es de 16 bits sin signo o entero de 32 bits, los píxeles se dividen por 256. Es decir, el rango de valores [0,255 * 256] se asigna a [0,255].

Si la imagen es de 32 bits de punto flotante, los valores de los píxeles se multiplican por 255. Es decir, el rango de valores [0,1] se asigna a [0,255].

Si la ventana no se creó antes de esta función, se supone la creación de una ventana con `CV_WINDOW_AUTOSIZE`.

Si usted necesita para mostrar una imagen que es más grande que la resolución de la pantalla, usted tendrá que llamar `namedWindow` ("", `WINDOW_NORMAL`) antes de la `imshow`.

```
int cvDestroydWindow(char* name);
```

La función `cvDestroydWindow()` cierra la ventana una vez que se ha detenido el programa y libera la memoria usada por ella; recibe un solo parámetro apuntador llamado `name`, que indica el nombre de la ventana a cerrar. [21]

```
IplImage* cvCreateImage(
CvSize size,
int depth,
int channels)
```

Esta función `cvCreateImage` crea imágenes. Esto se debe cuando se crea una imagen tipo `IplImage,` se puede seleccionar el tipo de imagen que va almacenar en dicha variable. La misma recibe los siguientes argumentos:

- `CvSize` tamaño de la imagen- Ancho y la altura de la imagen
- `depth` define la profundidad de la imagen en el proyecto se eligió de 8. Ver `IplImage` para profundidades válidas.
- `channels` – Número de canales por píxel. Ver `IplImage` para más detalles. Esta función sólo crea imágenes con canales intercalados. [21]

```
cvReleaseImage(IplImage** image)
```

Esta función `ReleaseImage()` se usó para liberar memoria ocupada por una determinada imagen. Solamente recibe un argumento que representa la imagen a la cual se liberaría la memoria.

Esta función retornara un apuntador a `IplImage`, la cual tendría trasformaciones requeridas. [21]

`IplImage` es la estructura que se utiliza para la interpretación de imagen en OpenCV, imagen de cabecera IPL (*Intel Image Processing Library*).

- `int nSize: sizeof (IplImage)`
- `int nChannels` Número de canales. La mayoría de las funciones de OpenCV apoyan 1-4 canales tomamos el 1.
- `int alphaChannel` Ignorado por OpenCV
- `int depth` Profundidad del canal en bits +bit de signo opcional `(IPL_DEPTH_SIGN)`. Se trabajó con la profundidad:

IPL_DEPTH_8U - entero sin signo de 8 bits. Equivalente a CV_8U en tipos de matriz.

La función cvGetSize Devuelve el tamaño de la matriz o la imagen ROI.

La función devuelve el número de filas (CvSize :: altura) y el número de columnas (CvSize :: ancho) de la matriz de entrada o la imagen. En el caso de la imagen se devuelve el tamaño de la ROI. [27]

ROI tienen gran importancia práctica, ya que en muchas situaciones que aceleran las operaciones de visión por ordenador, estableciendo una imagen Región de interés (ROI) de un rectángulo dado. Para activar o desactivar el retorno de la inversión, utilice las funciones cvResetImageROI() y cvSetImageROI(). Dada una subregión rectangular de interés en la forma de un CvRect, puede pasar un puntero de la imagen y el rectángulo cvSetImageROI () para "encender" retorno de la inversión; "Apagar" ROI al pasar el puntero en la imagen para cvResetImageROI ().[21]

```
void cvSetImageROI(
   IplImage* image,
   CvRect rect
);
```

Parámetros:

- image - Un puntero a la cabecera de la imagen
- rect - El rectángulo ROI

Si el ROI imagen original era NULL y el rect no es toda la imagen, la estructura ROI se asigna.

La mayoría de las funciones de OpenCV apoyan el uso de ROI y tratar el rectángulo imagen como una imagen separada. Por ejemplo, todas las coordenadas del pixel se cuentan a partir de la esquina superior izquierda (o de abajo a la izquierda) del retorno de la inversión, no la imagen original. [3]

```
void findContours(
InputOutputArray image,
```

```
OutputArrayOfArrays contours,
OutputArray hierarchy,
int mode,
int method,
Point offset=Point()
);
```

Esta función encuentra los contornos de una imagen binaria.

Parámetros:

- `image` – Fuente, una imagen de un solo canal de 8 bits. Los píxeles no son cero son tratados como 1. Si con cero píxeles permanecen como 0 es decir no detecta nada, por lo que la imagen se trata como binario.
- `contours` - Detectado contornos. Cada contorno se almacena como un vector de puntos.
- `Hierarchy` – vector de salida opcional, que contiene información sobre la topología de la imagen.
- `mode` - Modo de recuperación del contorno que se uso es `CV_RETR_CCOMP` el cual recupera todos los contornos y los organiza en una jerarquía de dos niveles. En el nivel superior, hay límites externos de los componentes. En el segundo nivel, hay límites de los agujeros. Si hay otro contorno interior de un agujero de un componente conectado, todavía se pone en el nivel superior.
- `method` – Método de aproximación a utilizarse es `CV_CHAIN_APPROX_SIMPLE` comprime segmentos horizontales, verticales y diagonales y deja sólo sus puntos finales. Por ejemplo, un contorno rectangular arriba-derecha se codifica con 4 puntos.
- `Offset` – Opcional proporcionado por la que se desplaza cada punto del contorno. Esto es útil si los contornos se extraen de la `ROI` imagen y entonces deben ser analizados en el contexto global de la imagen. [3]

```
CvRect cvBoundingRect(
CvArr* points,
int update=0 )
```

Parámetros:

- `points` – conjunto de puntos 2D de entrada, almacenados en `std::vector o Mat`.

La función calcula y devuelve el mínimo rectángulo de delimitación arriba-derecha para el conjunto de puntos especificado.

La técnica más simple es llamar `cvBoundingRect ();` devolverá un `CvRect` que limita el contorno. `CvContour` es un miembro `CvRect` para referirse a su propio cuadro de límite. Si llama `cvBoundingRect ()` con la actualización se establece en 0, entonces usted acaba de obtener el contenido de ese miembro de datos; pero si llamas con la actualización se establece en 1, el cuadro de límite se computará (y el miembro de datos asociada también se actualizará).

Un problema con el rectángulo delimitador de `cvBoundingRect ()` es que es un `CvRect` por lo que sólo puede representar un rectángulo cuyos lados están orientadas horizontal y verticalmente. Ver Figura 4.3.

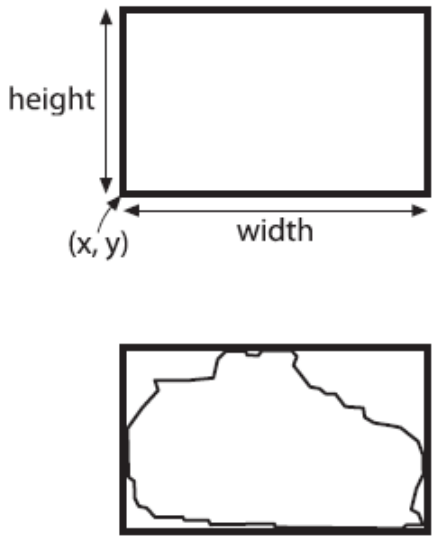


Figura 4.3 CvRect [21]

```
cvHoughCircles(
CvArr* image,
void* circle_storage,
int method,
double dp,
double min_dist,
double param1 = 100,
```

```
double param2 = 300,
int min_radius = 0,
int max_radius = 0
);
```

Esta función `cvHoughCircles()` se la ha utilizado con el fin de detectar círculos en una imagen. En el caso de detección de línea, una línea se define por dos parámetros (r, θ). En el caso del círculo, necesitamos tres parámetros para definir un círculo: $C: (x_{center}, y_{center}, r)$

Donde x_{center}, y_{center} definir la posición central (punto de grado) y r es el radio, lo que nos permite definir completamente un círculo, como se puede ver en la siguiente Figura:

Figura 4.4 Circulo definido [28]

Parámetros:

- `image`: de 8 bits, la imagen de un solo canal, imagen de entrada. Si se trabaja con una imagen en color, convertir a escala de grises primero.
- `circle_storage`: Un vector que almacena conjuntos de 3 valores: $(x_{center}, y_{center}, r)$ para cada círculo detectado, puede ser una matriz o de almacenamiento de memoria, dependiendo de cómo le gustaría que los resultados devueltos. En este caso se utiliza la memoria de almacenamiento, entonces los círculos se harán en una secuencia de OpenCV y un puntero a esa secuencia se devolverán por `cvHoughCircles()`. (Dado un valor de puntero matriz para

`circle_storage`, el valor de retorno de `cvHoughCircles ()` es NULL.)

- `method`: Define el método para detectar los círculos en las imágenes. Actualmente `CV_HOUGH_GRADIENT` esta es la única disponible en OpenCV.
- `dp`: Este parámetro es la relación inversa de la resolución del acumulador a la resolución de la imagen. Por ejemplo, si `dp = 1`, el acumulador tiene la misma resolución que la imagen de entrada. Si `dp = 2`, entonces la resolución del acumulador será más pequeño por ese factor, en este caso media, se usó un `dp=2` en el proyecto.
- `min_dist`: Distancia mínima entre el centro *(x,y)* de las coordenadas de los círculos detectados. Si el parámetro es demasiado pequeño, múltiples círculos de vecinos pueden ser detectados falsamente además de una verdadera. Si es demasiado grande, algunos círculos se pueden perder.
- `param1`: umbral superior para el detector de borde
- `param2`: valor umbral del acumulador para el método `HOUGH_GRADIENT`. Cuanto menor sea el umbral, se detectarán más círculos (incluyendo falsos círculos).Cuanto mayor sea el umbral, potencialmente se devolverán más círculos.
- `minRadius`: Tamaño mínimo del radio (en píxeles). Si desconoce, poner a cero por defecto
- `maxRadius`: Tamaño máximo del radio (en píxeles). Si desconoce, poner a cero por defecto [29]

```
void GaussianBlur ( InputArray src ,
outputArray dst ,
Tamaño ksize ,
doble Sigmax ,
doble sigmaY = 0,
int borderType = BORDER_DEFAULT )
```

Se trabajó con el filtro de Gauss con el fin de reducir el ruido, este filtro se realiza mediante la convolución de cada punto de la matriz de entrada con un *kernel de Gauss* y luego sumando todos ellos para producir la matriz de salida.

Parámetros:

- `src`: imagen de entrada; la imagen puede tener cualquier número de canales, que se procesan de forma independiente, pero la profundidad debe ser:
  `CV_8U, CV_16U, CV_16S, CV_32F o CV_64F.`
- `dst`: imagen de salida del mismo tamaño y tipo que `src.`
- `ksize` – tamaño Gaussian kernel (núcleo) `ksize.width` y `ksize.height` pueden diferir, pero ambos deben ser positivos y extraño. O bien, pueden ser cero y después de que se calculan a partir `sigma *.`
- `sigmax`: desviación estándar del núcleo de Gauss en dirección X.
- `sigmaY`: desviación estándar del núcleo de Gauss en la dirección Y; Si `sigmaY` es cero, se establece para que sea igual a `Sigmax`, si ambos están sigmas ceros, que se calculan a partir `ksize.width y ksize.height ,` respectivamente; para controlar totalmente el resultado independientemente de posibles modificaciones futuras de todos esta semántica, se recomienda especificar todos `ksize` , `Sigmax` y `sigmaY .`
- `borderType`: método de extrapolación de píxeles, este valor sólo afecta a los píxeles en la frontera. Utilizaremos `BORDER_DEFAULT`. [28]

4.1.3 Funciones y procedimiento de API

Como se mencionó en el ítem 2.2.2 del presente proyecto, para trabajar con los eventos del mouse en el PC se recurrió a la Librería de Interfaz de Programación (API), Win32. En el ítem antes mencionado se indicó las rutinas que forman parte de Win32; sin embargo para la realización de éste proyecto se utilizó solo algunas de ellas, y pasaron a formar parte del presente proyecto.

```
void mouse_event
{
DWORD dwFlags
DWORD dx,
DWORD dy,
DWORD dwData,
DWORD dwExtraInfo
};
```

El procedimiento `mouse_event()` permite ejecutar las diferentes acciones del mouse, como el desplazamiento y los clic. Para el proyecto, cabe destacar que, se usó tres de los primeros eventos mencionados.

`dwFlags` usa banderas para indicar que ha sucedido eventos, no se utilizó todas las banderas, las que si fueron utilizadas se detallan a continuación:

- `MOUSEEVENTF_MOVE`, para especificar que ha ocurrido un movimiento.
- `MOUSEEVENTF_LEFTDOWN`, para especificar que el botón izquierdo ha cambiado ha pulsado.
- `MOUSEEVENTF_LEFTUP`, para especificar que el botón izquierdo ha cambiado ha no pulsado.
- `MOUSEEVENTF_ABSOLUTE`, para indicar que los parámetros `dx` y `dy` contienen coordenadas absolutas normalizadas. Si no está activa, los parámetros contienen datos relativos: el cambio de posición desde la última informada

`dx` se encarga de la posición horizontal del mouse.

`dy` se encarga de la posición vertical del mouse.

```
int SendInput();
```

La función `SendInput()` sintetiza movimientos que ha realizado el mouse y sus pulsaciones.

```
int GetSystemMetrics(int nIndex);
```

La función `GetSystemMetrics()` recupera varios valores de medidas y parámetros de configuración del sistema, las medidas del sistema son las

dimensiones de ancho y alto del *display* de *windows*, expresado en pixeles, es decir la resolución. El parámetro tipo entero que recibe ésta función denominado nIndex especifica la configuración del sistema a recuperar. En el proyecto se utilizó `SM_CXSCREEN` y `SM_CYSCREEN`, que representan ancho y alto respectivamente.

4.2 Proceso para la implantación de la herramienta

En el marco de la implementación de la herramienta, se decidió separar en dos partes para probar su óptimo funcionamiento, la primera de ellas es el movimiento del cursor del mouse, la cual se detalla a continuación.

El primer paso a seguir es captar el video en formato digital que ingresa de la cámara web, que se muestra en la Figura 4.5. Esto se lo hace mediante la función `VideoCapture cap()`. En otras palabras obtiene la imagen proveniente de la cámara web.

Figura 4.5 Imagen de entrada a la cámara web

En este punto, la imagen que ingresa será invertida y transformada a escala de grises, como es bien sabido el tratamiento de imágenes en este formato resulta más fácil de trabajar y visualizar la gama de los medios tonos, que para el presente proyecto es de gran utilidad. Los resultados de la función utilizada `cvCvtColor()` se muestra en la Figura 4.6. Es decir la imagen que ingresó por la cámara web entro en formato `BGR` (azul, verde y rojo), colores, será transformado a blanco y negro e

invertido (similar a un negativo fotográfico) para continuar con el tratamiento de la imagen.

Figura 4.6 Imagen invertida y formato escala de grises

En este punto se obtiene una imagen con bordes que comúnmente se lo llama basura, por lo que es necesaria una función que elimine dichos bordes; de este modo se obtiene una imagen con objetos fuertes. El siguiente paso consiste en aplicar un suavizado Gaussiano a la imagen, con el fin de eliminar los bordes e irregularidades menores y mantener únicamente los contornos fuertes, mediante la función, `GaussianBlur()`, el resultado de éste proceso se lo puede apreciar en la Figura 4.7.

Figura 4.7 Imagen con suavizado Guassiano

Luego de haber obtenido una imagen en escala de grises y mejorada con suavizado gaussiano, por razones explicadas en el presente documento, se procede a realizar la umbralización, para que se pueda determinar ciertos umbrales de forma que delimiten la imagen en zonas, en lo que a valores de intensidad se refiere; dependiendo del nivel de gris de cada píxel. En otras palabras se busca objetos que tengan un determinado tipo de intensidad, para el caso del proyecto la pupila. En la Figura 4.8 se puede observar el resultado de la aplicación del umbral; se lo hará mediante la función `cvthreshold ()`.

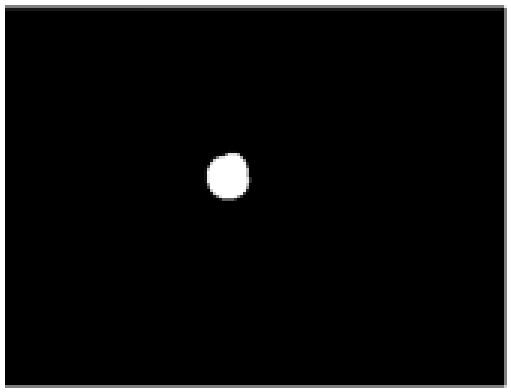


Figura 4.8 Imagen con umbral a 245

Ahora, después de haber aplicado umbralización a la imagen, es notorio que habrá objetos que tendrán la misma intensidad que la pupila, por lo que es necesario buscar la manera de que ésta quede aislada y sea el único objeto a tratar, esto se lo hace mediante el detector de Hough, que implícitamente busca círculos en la imagen. Se lo realiza mediante la función `HoughCircles()` que detecta círculos en la imagen. La función `center()`, creada, devuelve dos parámetros importantes, como el centro en coordenadas (x,y) y radio del círculo que forma la pupila. Una vez detectado el círculo de la pupila, conociendo su centro y radio, se procede a dibujar dicho círculo con la función `circle()`(Véase Figura 4.9). El centro y radio obtenidos serán usados en las funciones que se describen más adelante.

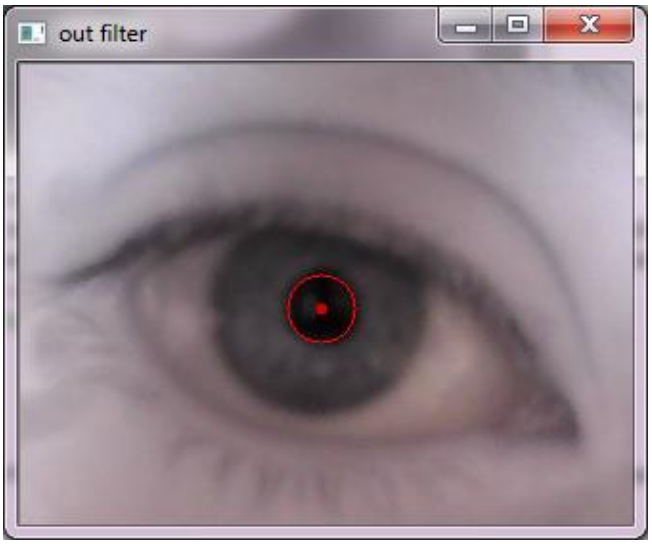


Figura 4.9 Pupila detectada

Cabe indicar que de este punto en adelante solo se utiliza funciones API Win32 y las creadas por las autoras del proyecto. La función `MouseMove(double x, double y, int mode)` fue creada bajo la finalidad de ajustar las medidas de la pantalla; y verificar si el movimiento que se realiza con el mouse es absoluto o relativo.

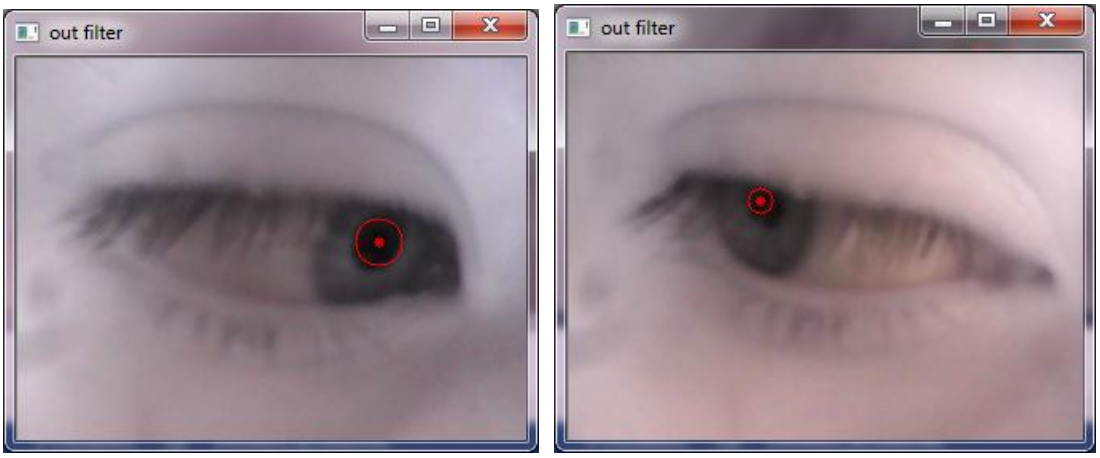
Para empezar, las resoluciones de pantalla diferentes pueden provocar pantallas que aparecen fuera de proporción. Se puede obtener el ancho y el alto de los distintos elementos de la pantalla de la ventana mediante la función de interfaz `GetSystemMetrics()` de API. Es decir se obtendrá las dimensiones, en pixeles, del display de Windows.

Una vez que se ha obtenido las dimensiones de la pantalla, se procede a condicionar los movimientos del mouse, absolutos o relativos, mediante una serie de combinaciones de eventos que se detallaron previamente en el capítulo II. Cabe indicar que para el caso se utilizó únicamente movimientos relativos, puesto que el usuario parte de cualquier punto de la pantalla para mover el cursor.

La función `MouseMove (double x, double y, int mode)` es la encargada de mover el puntero del mouse. Ingresan tres parámetros; el primero `double x` indica que tanto de pixeles se desea que se recorra en el eje *x*, lo mismo sucede con el parámetro `double y`, con la diferencia que se lo hace respecto al eje

y. El parámetro `int mode` es una bandera, que da la opción de saber si el movimiento del mouse que se está efectuando es absoluto o relativo.

La función `MouseLeftRight(int LeftH,int data,int RightH)` indica el sentido con el que se moverá el puntero del mouse, en este caso de izquierda a derecha y viceversa. Ingresan tres parámetros, el primero y últimos de ellos especifican a partir de cuantos pixeles se empezará a mover el puntero, ya sea a la izquierda o derecha; mientras que el parámetro int data es el que almacena la posición de donde se encuentra el puntero en el eje x y lo va actualizando, es decir en éste parámetro se ingresó las coordenadas `(x,y)` obtenidas de la función `center()` descrita anteriormente. En resumen la función se utiliza para mover el puntero del mouse en el `eje x` de acuerdo a las coordenadas obtenidas en la función `center ()`. Dentro de ésta función se llama a la función `MouseMove ()`, con el ingreso de los tres parámetros establecidos `(0.1, 0,0)`, el primero con un avance de *0.1* o *-0.1*, dependiendo si es izquierda o derecha, en pixeles; el segundo en *0* ya que no se desea que recorra verticalmente; y el tercero es *0* debido a que se necesita que la posición del mouse sea relativa. En la Figura 4.10 se puede observar el resultado de aplicar la función creada `MouseLeftRight(int LeftH,int data,int RightH)`, donde a) movimiento a la izquierda b) movimiento a la derecha.

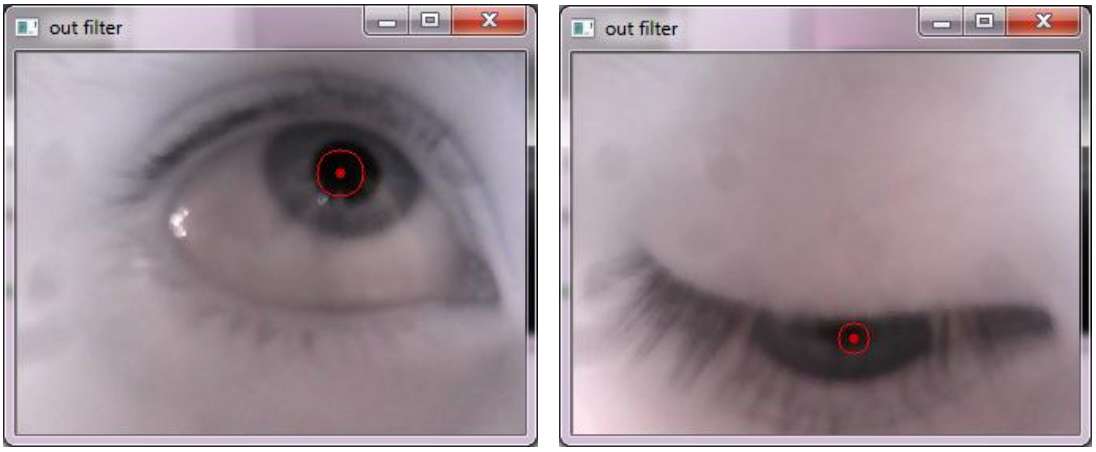


a) **b)**

Figura 4.10 Detección del movimiento ocular en el eje X

La función `MouseUpDown(int DownH,int data,int UpH)`, sigue el mismo lineamiento que la función `MouseLeftRight()`, con la diferencia que lo

hace a lo largo del eje *y*. De igual manera llama a la función `MouseMove ();` pero sus tres parámetros establecidos vienen dados de la siguiente manera: (0,0.1, 0); el primero de ellos *0* debido a que no se desea que haya movimiento horizontal; el segundo *0.1* o *-0.1* debido a que se le asigna un avance vertical, *0.1* para arriba y *-0.1* para abajo; y el último *0* ya que el movimiento que se desea es relativo. La Figura 4.11 muestra la resultante de aplicar la función `MouseUpDown(int DownH,int data,int UpH)`, donde a) movimiento hacia arriba b) movimiento hacia abajo



a) **b)**

Figura 4.11 Detección del movimiento ocular en el eje Y

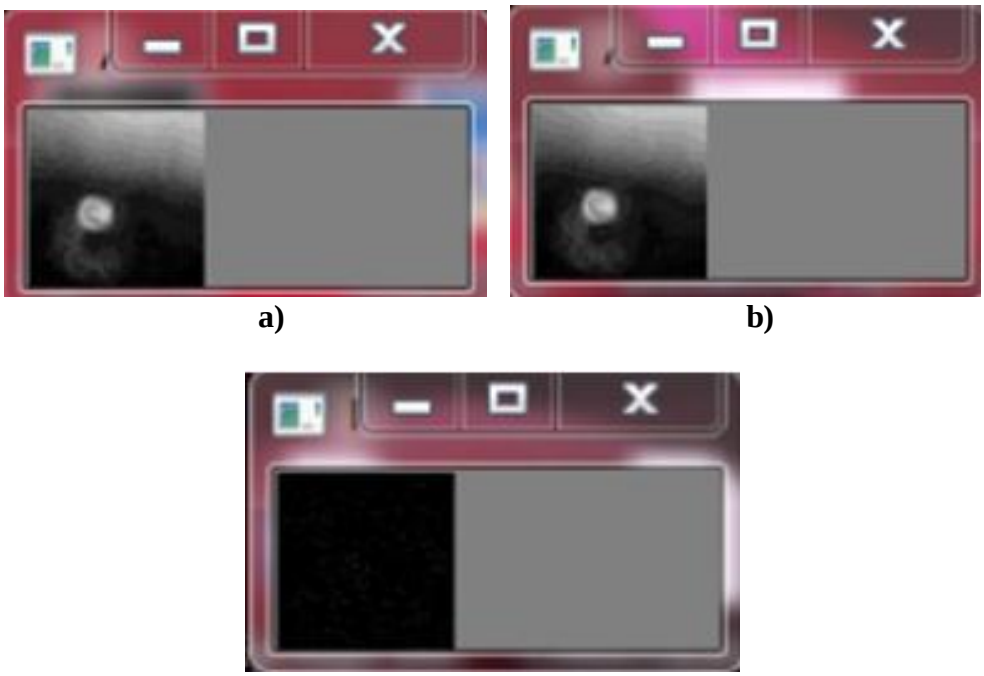
Luego de conseguir el movimiento del cursor, se procedió con la implementación del clic, con esto se lograría que el usuario simultáneamente con el movimiento, pueda acceder a las aplicaciones del computador.

Se utilizó la función `cvCreateMemStorage()` que es una entidad denominada una memoria de almacenamiento como su método de manipulación de asignación de memoria para objetos dinámicos, en la cual se almacenará la imagen actual, imagen previa y la imagen deferencial, las dos primera imágenes son la imagen que es capturada y la otra es la imagen en ese instante, las tercera es la imagen donde se mostrara el resultado de la resta de la 2 imágenes anteriores, las 3 variable se inicializo creando una imagen en blanco con el tamaño del video y en el punto de origen de acuerdo a las propiedades de `IplImage` siempre se inicializa en el punto de origen, mediante la función `cvCreateImage`, y se utiliza `cvGetSize` ya que es adecuada para la construcción de una nueva matriz y

devuelve el tamaño de una matriz, es decir las matrices tendrán las misma características.

Se define el área de trabajo, esto se realizó mediante la función `cvSetImageROI` el cual nos permite procesar una pequeña subregión de la imagen con esto, logramos modificar una región de la imagen que requerimos; definiendo el tamaño del rectángulo. Definida el área de trabajo se procedió a realizar la resta entra la imagen actual y la imagen anterior con el uso de la función `cvSub`, considerar las tres variables tiene que tener el mismo tipo, tamaño y número de canales, en el proyecto de uso una matriz de 8bits. Como resultado se podrá obtener dos tipos de resultados, dependiendo de si exista o no movimiento.

El primer caso, al tener dos imágenes iguales no se presentar ningún cambio a la salida, es decir el umbral sigue siendo el mismo, ver Figura 4.12, donde a) Imagen actual, b) imagen anterior, c) imagen diferencia sin que exista detección de movimiento.:

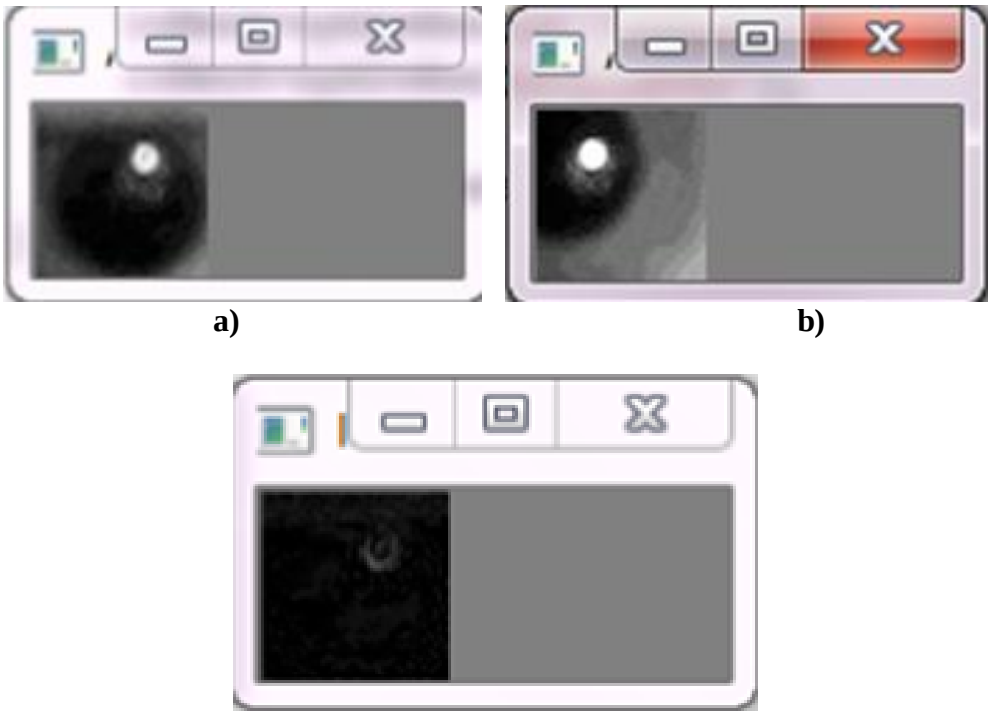


a) **b)**

c)

Figura 4.12 Diferencia de imágenes

Y cuando se tiene las imágenes diferentes se ejecuta la resta es decir se tiene umbral de la imagen, como se observa en la siguiente Figura donde a) Imagen actual, b) imagen anterior, c) imagen al existir detección de movimiento:



a) **b)**

c)

Figura 4.13 Detección de diferencia de imágenes

Observar que en la figura 4.12 no se tiene ningún resultado ya que las dos imágenes son iguales es decir no se tiene cambio en el umbral, y el figura 4.13 se observa una diferencia teniendo como resultado una imagen en la cual se tiene datos diferentes se observa que el umbral.

A esta imagen resultado de la resta, se la aplica el umbral; con la función `cvThreshold` para que no sea detectado el mínimo movimiento a cada momento, es decir por cualquier movimiento detectado tener resultados que no se los requiera. De igual manera aplicamos la morfología con el uso de la función `cvMorphologyEx;` en este caso se efectuó la transformación de apertura `CV_MOP_OPEN` la cual tiende a resaltar el área de las regiones conectadas con mayor precisión, es decir acentuar más las áreas reconocidas por el umbral, en la siguiente figura se muestra la morfología de la imagen diferencia de la Figura 4.13.

Figura 4.14 Morfología de la imagen diferencia

Con la obtención de una imagen binaria con la parte que se mueve en blanco, copiamos la imagen para mostrar los contornos en la región de trabajo con la función `cvFindContours()`, se localiza los contornos en la imagen binaria, estos contornos detectados son almacenados como un vector de puntos en un puntero `CvSeq`, en la función `cvFindContours()`, se utilizó el modo de recuperación del contorno `CV_RETR_CCOMP` este método recupera todos los contornos hallados, y el método de aproximación que se uso es `CV_CHAIN_APPROX_SIMPLE`. En el proyecto comprime los segmentos, es decir el contorno rectangular de arriba, derecha y se codifica con 4 puntos. El resultado de lo antes expuesto se puede observar en la Figura 4.15.

Figura 4.15 Detección de contornos

Con cada contorno localizado; se usó el método `cvBoundingRect` para que nos devuelva un objeto `CvRect` que limita el contorno, el vector donde se encuentran todos los contornos localizados es miembro de `CvRect,` mismo que se utiliza para dibujar el rectángulo alrededor de ese objeto, y tener los contornos detectados dentro de su propio rectángulo. Lo importante es las dimensiones de la figura que está incluida en este contorno, mas no lo contornos como tal para estimar la apertura del grado del ojo.

Creamos una función para el parpadeo del ojo que es `blink` creado por las autoras del proyecto, el cual muestra el resultado de las imágenes filtradas mediante `nameWindows`, crea una ventana, se usó la bandera `CV_WINDOW_AUTOSIZE` en la que el tamaño se ajusta automáticamente para adaptarse a la imagen mostrada y poder encajar la imagen, y con la función `imshow` que muestra una imagen en la ventana específica, después la ventana cambia automáticamente si una nueva imagen se carga en la ventana.

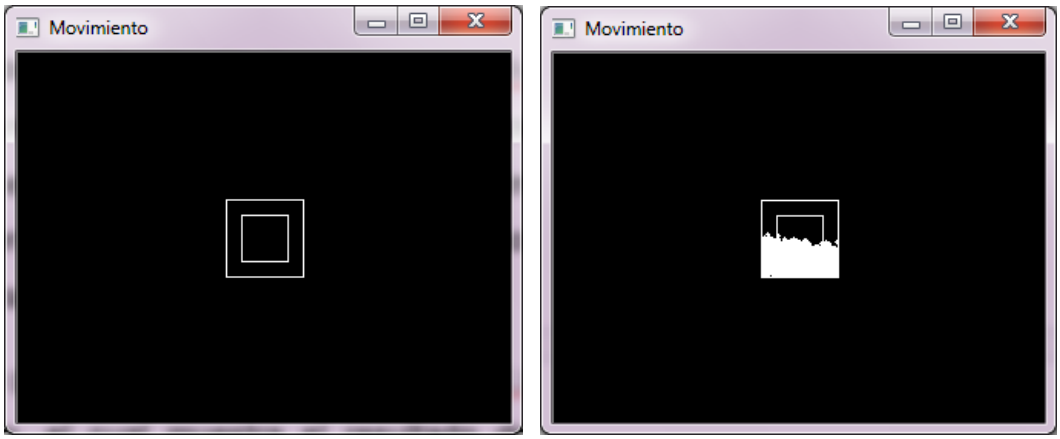


Figura 4.16 Detección del parpadeo

Obteniendo así la diferencia cuando se tenga en la una ventana la pupila y en la otra se observe que no hay pupila es decir ha realizado un parpadeo ver en la Figura 4.16. Se tiene que considerar que existe un breve espacio de tiempo que transcurre mientras se cierran los ojos y cuando se abren, es por ello que durante este transcurso de tiempo no está completamente cerrado ni abierto, entonces para detectar si el parpadeo fue voluntario o involuntario dependería de la tiempo en cual se lo mantenga cerrado, es decir cuando el ojo se encuentra abierto queda a la vista el cirulo que detecta la pupila; y cuando un ojo se encuentra cerrado se mira las

pestañas y el parpado, la una obstrucción es la captura del color blanco al tener el parpado y la otra va ser cuando se encuentre totalmente en color negro el cuadro, esto sucederá cuando los ojo se encuentren mirando a un punto fijo y se tendrá los parpadeos normales.

Como se debe diferencia cuando esté cerrado y cuando esté abierto el ojo se procede a cerrar el ojo durante 3 segundo aproximadamente para realizar un clic y la pantalla permanecerá en color negro ya que no detectara ningún movimiento, y para el doble clic se a largo el tiempo a 5 segundos.

Los eventos de mouse se realizara las acciones principales que ejecuta un mouse convencional que es el clic y doble clic, los cuales serán controlados mediante un gesto facial, como es el mantener cerrado el parpadeo durante cierto tiempo y tener la acción de un clic y doble clic, considerando que cada evento se lo realiza uno a la vez.

4.3 Tratamiento de la imagen para la obtención de parámetros óptimos

Una vez explicadas las funciones con las cuales se desarrolló el proyecto, se explica cómo se determinó el valor de umbral, mediante el método más efectivo, ensayo y error. Para empezar, en la Figura 4.15 se puede observar la imagen resultante de la cámara modificada.

Posteriormente se realiza el proceso para mover el cursor, explicado anteriormente; en este proceso se observa que para aplicar el umbral se usa la función `threshold();` para la realización de la herramienta con parámetros óptimos se realizó el método de prueba y error con diferentes valores de umbrales. Los resultados se muestran en la siguiente Figura 4.17.

El valor de n=220 es óptimo para el funcionamiento de la herramienta, ya que si tenemos un valor menor el umbral muestra otros objetos, en este caso las pestañas, y si es un valor mayor el círculo de la pupila empieza a deformase hasta el punto de desaparecer. Por lo que n quedará establecido como variable fija, es decir que el momento que se ejecute el programa el valor de n=220 estará establecido por defecto. Cabe recalcar que a pesar de tener un parámetro óptimo de umbral, se creó

una ventana para permitir el ingreso del valor del umbral y que el usuario, en caso de no encontrarse satisfecho con lo que la herramienta muestra, pueda modificar dicho valor. En la Figura 4.17 se observa los resultados de aplicar diferentes umbrales con valor n.

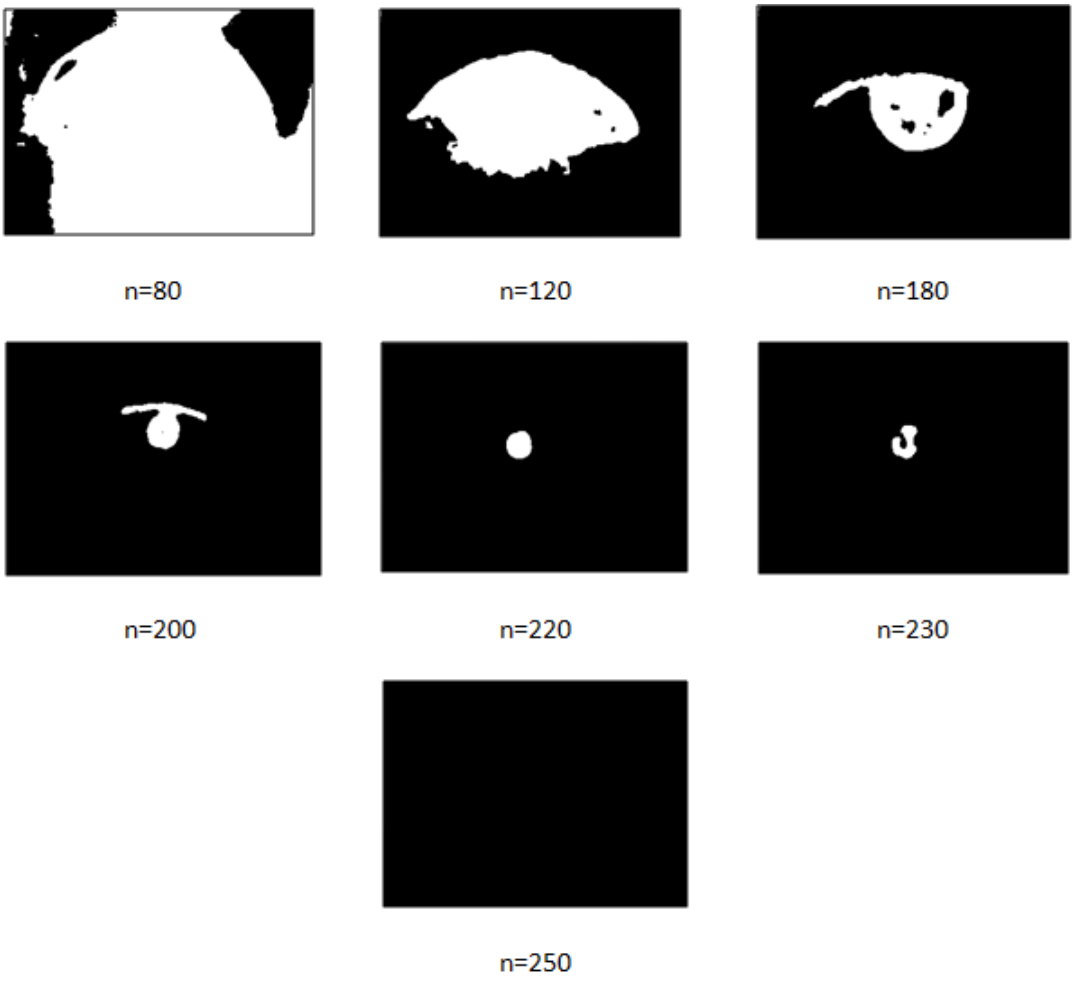


Figura 4.17 Variación del umbral

4.4 Representación de algoritmos-diagrama de flujo

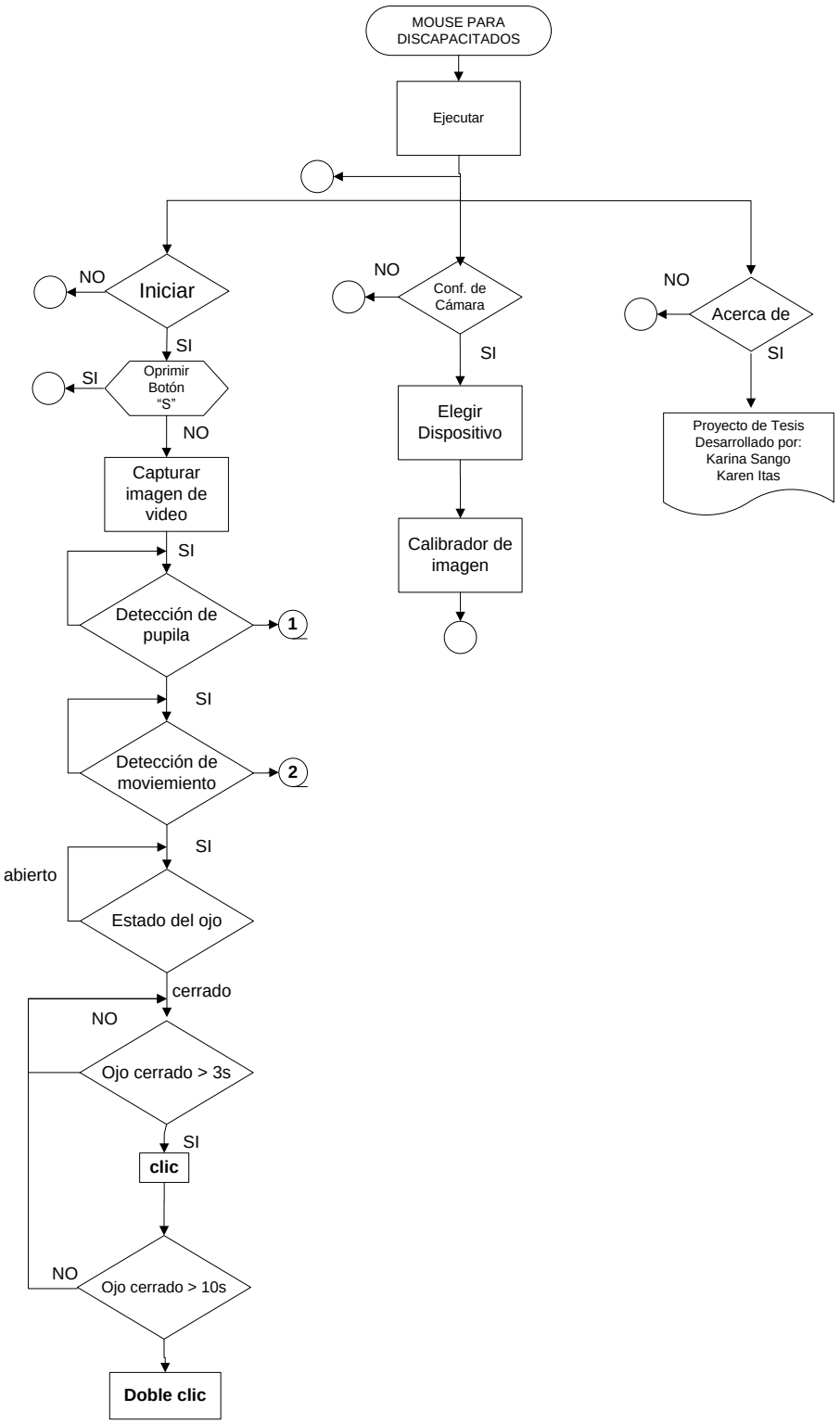


Figura 4.18 Algoritmo principal

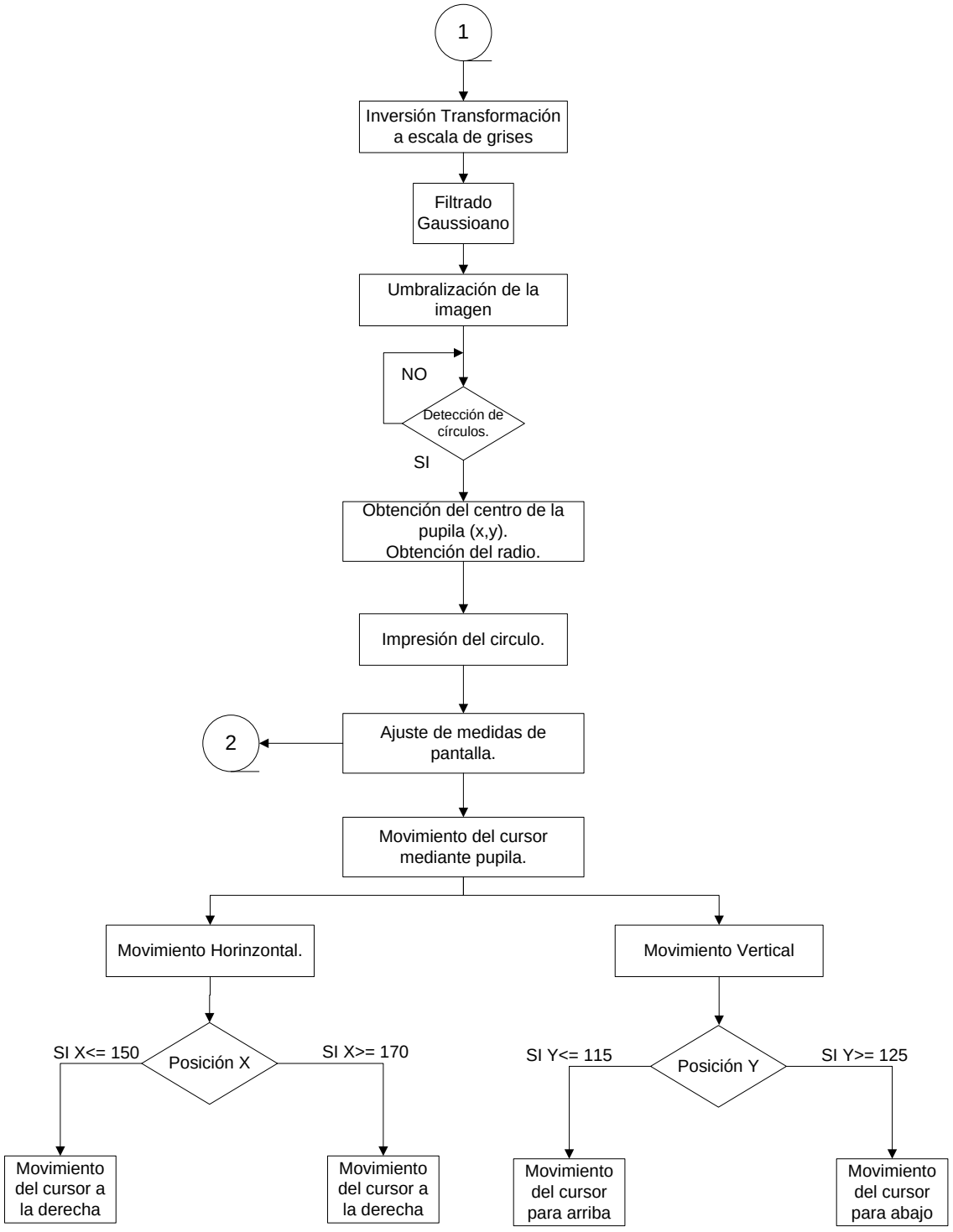


Figura 4.19 Algoritmo detección de la pupila

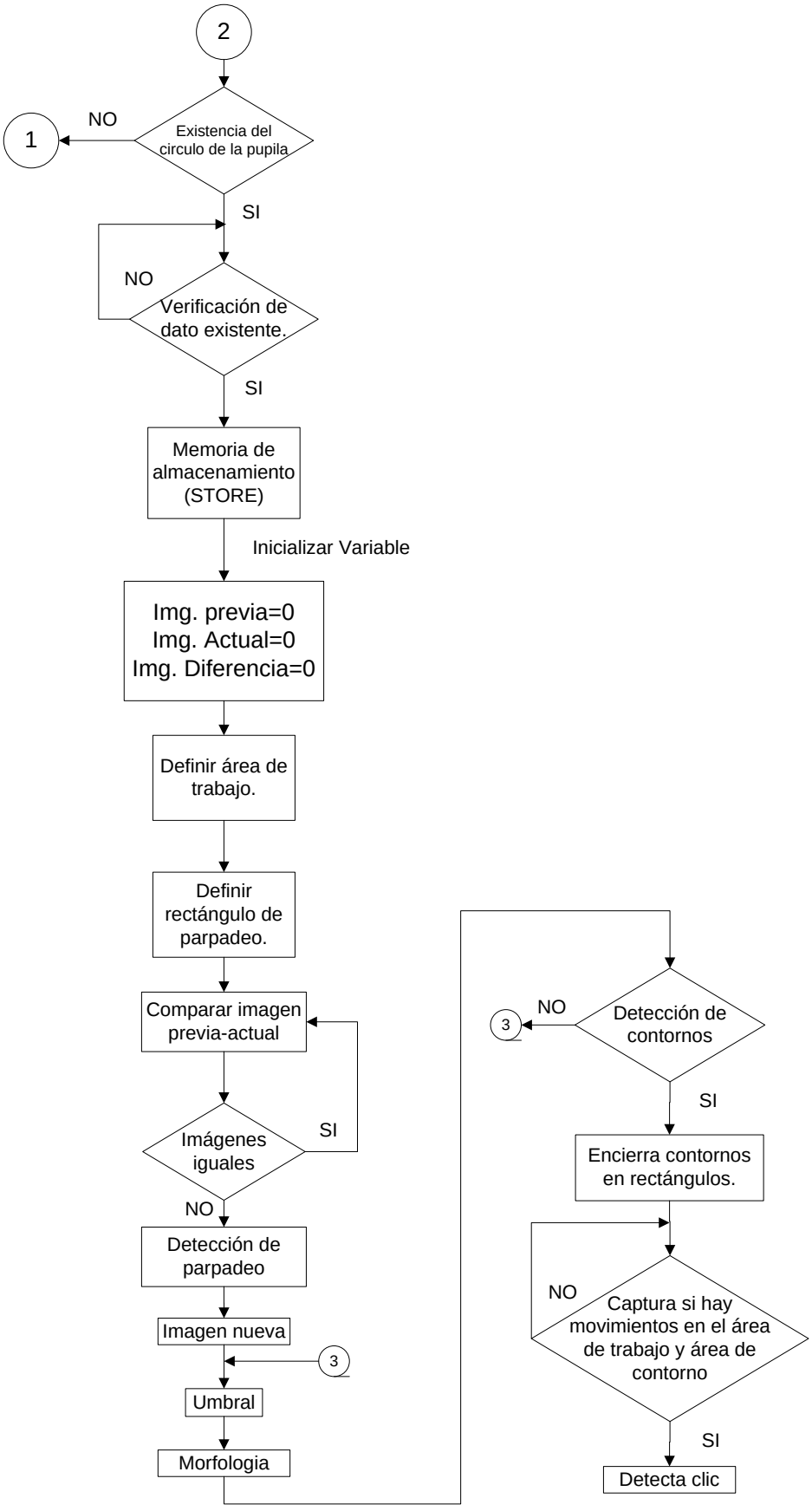


Figura 4.20 Algoritmo detección movimiento y estado del ojo

4.5 Resultados de las pruebas

Se presentó como objetivo específico garantizar resultados con niveles de error inferiores al 20%, por lo que la fase de Resultado de las Pruebas tomó mayor tiempo de lo previsto.

Inicialmente, se realizaron diferentes pruebas que garantice el nivel de error antes mencionado en la herramienta, después se planteó que la herramienta sea cómoda y tenga una interfaz amigable, con la cual el usuario pueda interactuar sin complicaciones.

El primero de los resultados notorios en la realización de la herramienta fue que la cámara web acoplada en la gorra debía tener un ángulo adecuado, es decir estar frente a frente con el ojo, sin ningún tipo de inclinación, ya que esto influye directamente en el sentido del movimiento del puntero del mouse.

La herramienta fue probada con personas de ojos claros y oscuros, en ambos casos se puede apreciar que funciona, cabe acotar que para las personas con ojos claros el umbral que se requería era menor; partiendo de ésta característica se ajustó el umbral de modo que la herramienta funcionará para ambos casos. Se menciona este punto debido a que durante la investigación para la realización del presente proyecto se encontró con herramientas que tenían el limitante de ser solo para ojos oscuros. La misma característica ocurría con el maquillaje que se usará durante las pruebas, es decir si se usaba un maquillaje oscuro esto afectaba directamente al umbral de trabajo, por lo que es indispensable recalcar que el umbral ajustado funciona para ojos claros y oscuros, ambos sin maquillaje oscuro. En caso de que el usuario llevara algún tipo de maquillaje oscuro, la herramienta permite el ingreso del valor del umbral que se ajuste a las características necesarias para el correcto funcionamiento de la herramienta.

Respecto al proceso de calibración, basta con que la pupila del usuario coincida con el centro de la pantalla establecido, el cual se muestra en el botón Configuración de la herramienta (Ver Anexos). Sin embargo este proceso en un comienzo se mostró complicado debido al factor ambiente, el cual lo controlamos modificando la cámara

con la colocación de Led's infrarrojos alrededor de la misma para no depender del escenario en el que nos encontremos sea día o noche.

Para el movimiento del mouse se tiene que introducir el movimiento de la pupila al movimiento del puntero del mouse, aquí se tiene una desventaja, pues el movimiento de los ojos es más rápido que los del mouse y se pierde un poco la exactitud.

4.6 Factibilidad económica de la herramienta

Un estudio de factibilidad económica se realiza bajo el denominante de estimar costos asociados a la implementación de la herramienta; por esto es necesario mencionar que los gastos extras hechos durante el desarrollo no son considerados; sino los gastos asociados a la creación del periférico y desarrollo del software, básicamente lo que incluye lo mencionado en el ítem

4.6.1 Creación del periférico alternativo

Como base para la ejecución de la herramienta, el usuario debe contar con una computadora personal, con requisitos mínimos establecidos en cuanto a componentes y sistema operativo instalado.

La Tabla 6.1 detalla los componentes utilizados para la implementación de la herramienta con sus respectivos costos, para posteriormente llegar a establecer el precio de mercado de dicha herramienta.

Tabla 6.1

Costo de Desarrollo de la Herramienta

CANTIDAD	OBJETO	COSTO	TOTAL
1	Gorra	$ 15,00	$ 15,00
1	Cámara Web	$ 25,00	$ 25,00
9	Diodos LED	$ 0,40	$ 3,60
1	Licencia de Microsoft Visual Studio 2010 Express	$ 0,00	$ 0,00
1	Varios	$ 10,00	$ 10,00
	TOTAL		$53,60

A continuación, en la Tabla 6.2, se detalla los costos para determinar un Valor Total de Mercado.

Tabla 6.2

Costo de Mercado de la Herramienta

CANTIDAD	OBJETO	COSTO	TOTAL
1	Periférico Alternativo	$ 53,60	$ 53,60
1	Licencia de Microsoft Visual Studio 2010 Professional	$ 597,74	$ 597,74
1	Desarrollo de Software	$ 400,00	$ 400,00
	TOTAL		$ 1051,34

Es de suma importancia realizar un análisis comparativo del costo de la herramienta creada respecto a otras herramientas con similares características que actualmente se encentran en el mercado. De aquí se puede apreciar el principal beneficio de la herramienta creada en el presente proyecto, pues tiene un costo inferior a las demás. En la Tabla 6.2 podemos observar dicho análisis comparativo.

Tabla 6.3

Costos de Herramientas Similares

APLICACIÓN / HERRAMIENTA	TOTAL
Iriscom QG2	$ 7390,00
Iriscom QG3	$ 8477,00
Eye Mouse	$ 3043,00
Sistema I4Control	$ 1739,00
Ukuchañawi V1.0 (Herramienta desarrollada por las autoras)	$ 1051,34

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

En la realización del proyecto se efectuó un estudio de visión artificial, abarcando conceptos, herramientas, metodologías de las funciones y técnicas para la ejecución exitosa de la herramienta. El desarrollo de dicha herramienta permitió simular las funcionalidades del mouse a través del movimiento ocular, los cual permite concluir que:

- El presente proyecto permite el manejo del mouse a través de la detección de movimiento ocular en tiempo real, para personas con capacidades especiales, esto se logró realizando pruebas sin la necesidad de que los usuarios tengan movilidad de su cuerpo.
- La programación está formado por dos etapas; en la primera de ellas se realizó la detección del movimiento de la pupila con lo cual se logró el desplazamiento del cursor del mouse, desarrollado por algoritmos en OpenCV; y en la segunda etapa se obtuvo la detección del clic; esto mediante el tratamiento de algoritmos hechos en OpenCV, utilizando programación básica en C++ y aplicando funciones API.
- Para la implementación de la herramienta no fue necesario el uso de cámaras sofisticadas o de alto costo, la cámara utilizada fue de 5Mpx puesto que esta resolución tuvo gran influencia en el desarrollo del proyecto.
- La etapa para la obtención de parámetros óptimos fue de vital importancia ya que permitió establecer un valor de umbral mediante el cual la herramienta funcione con la mayor exactitud posible y el usuario puede hacer uso de la misma sin inconvenientes. Cabe recalcar que dicha obtención de parámetros permitió alcanzar uno de los objetivos planteados, el cual fue regirse a un margen de error inferior al 20%.

- La herramienta brinda a personas con capacidades especiales la posibilidad de interactuar con la computadora para la realización de diversas tareas, esto haciendo alusión a los objetivos planteados en el Plan Nacional del Buen Vivir, mencionados en el capítulo I.
- El presente proyecto brinda grandes beneficios económicos, puesto que es menos costoso respecto a otros productos que se encuentran actualmente en el mercado y que realizan funciones similares.
- Durante el desarrollo del proyecto se cumplió los objetivos trazados, concluyendo la elaboración de la herramienta con las características planteadas, esto mediante la realización de pruebas y análisis de los resultados obtenidos en cada etapa del proyecto.

5.2 Recomendaciones

- La herramienta seria mejorada si se trabajara con una cámara infrarroja, sin embargo el precio aumentaría y la herramienta no tuviera el mismo beneficio económico.
- Es necesario en el momento de la calibración el usuario mire directamente al lente de la cámara web y que la ubicación de la pupila del ojo este situada en el círculo de referencia (ver anexos).
- Desarrollar la herramienta para que permita la realización de funciones adicionales del mouse, ya que en el presente proyecto solo se enfatizó en dos de ellas: clic izquierdo y doble clic, mas no la ejecución de clic derecho, selección de una región, mover, arrastrar, etc.
- Realizar pruebas con más personas con discapacidades motrices y analizar los resultados buscando mejoras en la herramienta.

REFERENCIAS BIBLIOGRÁFICAS

[1] O. Krejcar, University of Hradec Kralove, Department of Information Technologies., 2013.

[2] República del Ecuador, *Plan Nacional del Buen Vivir*, 2009-2013.

[3] F. Justo, *"Creación de una herramienta que permita mover el cursor de un computador a partir del movimiento ocular, utilizando técnicas de visión artificial",* Mérida: Universidad de los Andes, 2009.

[4] J. Masdeu, "Blog de Naturotopia," 10 Enero 2014. [Online]. Available: http://www.naturopatamasdeu.com/la-degeneracion-macular-y-sus-cuidados/human-eye-cross-section/.

[5] L. Alegsa, «Alegsa,» 2014. [En línea]. Available: http://www.alegsa.com.ar/Dic/api%20de%20windows.php . [Último acceso: 11 11 2014].

[6] D. Marr, *Vision*, San Francisco: Freeman, 1982.

[7] M. F. P. A. A. F. C. M. O. J. V. E. Gonzalez A., TECNICAS Y ALGORITMOS BASICOS DE VISION ARTIFICIAL, Universidad de la Rioja, 2006.

[8] R. Llobregat, *Procesado digital de imágenes de video para la detección de humo*, Gandia, Valencia: Universidad Politecnica de Valencia, 2011.

[9] C. Alberola, «Laboratorio de Porcesado de Imagen,» 2014. [En línea]. Available: http://tanis.lpi.tel.uva.es/~nacho/docencia/ing_ond_1/trabajos_03_04/sonificacion/cabroa_archivos/umbralizacion.html. [Último acceso: 13 12 2014].

[10] U. P. d. Madrid, *Prácticas de Segmentación de las imágenes,* 2013.

[11] Gonzalez R. y Woods R., *Tratamiento Digital de Imágenes*, Madrid: Addison Wesley, 1996.

[12] E. Di Nardo, "OpenCV / Python – Primi passi," 30 Mayo 2013. [Online]. Available: http://emanueldinardo.com/wp/opencv-python-primi-passi/.

[13] E. Momma and T. Minagawa, "Introducing OpenCV for Developers," 9 Noviembre 2011. [Online]. Available: http://www.jsk.t.u-tokyo.ac.jp/rsj2011/_downloads/2Q2-3.pdf. [Accessed 9 Julio 2014].

[14] E. Turégano, «Eyeboard: Un Periférico Alternativo Visual,» 2006. [En línea]. Available: http://robolab.unex.es/research/doc/libro.pdf.

[15] J. Gallego, *FP Básica - Montaje y mantenimiento de sistemas y componentes informáticos,* 2015.

[16] Z. W. a. A. P. Ying Huang, Non-contact Gaze Tracking with Head Movement Adaptation based on Single Camera, International Scholarly and Scientific Research & Innovation, 2009.

[17] J. S. H. M. E. B. M. T. San Agustin, Evaluation of a low-cost open-source gaze tracker. In Proceedings of the 2010 Symposium on Eye-Tracking Research, New York: ETRA '10, ACM, pp. 77-80, 2010.

[18] J. P. DINA, MOUSE PARA PERSONAS CON DISCAPACIDAD MOTRIZ, ECUADOR, 2009.

[19] kioskea, "kioskea.net," Junio 2014. [Online]. Available: http://es.kioskea.net/contents/717-el-formato-divx-divx-xvid-3ivx-vp3.

[20] D. H. Halbritter, "Osram Opto Semiconductors," Marzo 2014. [Online]. Available: http://www.osram-os.com/Graphics/XPic5/00135349_0.pdf/High-Speed%20Switching%20of%20IR-LEDs%20(Part%20I).pdf.

[21] Bradski and Kaehler, Learning OpenCV, United States: Oreilly, 2008.

[22] A. K. G. Bradski, Learning OpenCV, United States: Oreilly, 2008.

[23] Bradski and Kaehler, Learning OpenCV, United States: Oreilly, 2008.

[24] L. M. Pechuán, "Uso de ontologías para guiar el proceso de segmentacion de imagenes," febrero 2007. [Online].

[25] Kaebler y Bradski, Learning OpenCV, United States: Oreilly, 2008.

[26] G. Lisanti, "Image Processing with OpenCV," New Ypork, 2010.

[27] "Documentación," 25 02 2015. [Online]. Available: http://docs.opencv.org/genindex.html. [Accessed 01 03 2015].

[28] Anónimo, "Image Filtering, GaussianBlur," 25 02 2015. [Online]. Available: http://docs.opencv.org/modules/imgproc/doc/filtering.html#gaussianblur.

[29] J. P. J. I. a. J. K. H.K. Yuen, "A COMPARATIVE STUDY OF HOUGH TRANSFORM METHODS FOR CIRCLE FINDING," University of Surrey, Guildford, GU2 5XH. U.K., South East of England, 1989.

ACTA DE ENTREGA

El proyecto fue entregado al Departamento de Eléctrica y Electrónica y reposa en la Universidad de las Fuerzas Armadas – ESPE, desde:

Sangolquí, 19 de Mayo de 2015

ELABORADO POR:

Karen Sulay Itás Mazón
171860672-4

Karina Mircya Sango Quimbita
172233799-3

AUTORIDAD

Ing. Paúl Bernal, MSc.
DIRECTOR DE LA CARRERA DE ELÉCTRICA Y ELECTRÓNICA, TELECOMUNICACIONES