



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN CONTROL Y REDES INDUSTRIALES

“DISEÑO DE UN SISTEMA DE INTERACCIÓN MEDIANTE VISIÓN ARTIFICIAL PARA PERSONAS CON DISCAPACIDAD MOTORA Y DIFICULTAD DEL HABLA”

TRABAJO DE TITULACIÓN

TIPO: PROYECTO TECNOLÓGICO

Presentado para optar al grado académico de:

**INGENIERO EN ELECTRÓNICA, CONTROL Y REDES
INDUSTRIALES**

**AUTORES: NAVARRETE ESPINOZA DIEGO FABIÁN
PILLAJO ÑAÑAY MARTHA CECILIA**

TUTOR: ING. JAVIER GAVILANES MSc.

Riobamba – Ecuador

2018

©2018, Diego Fabián Navarrete Espinoza y Martha Cecilia Pillajo Ñauñay

Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo a la cita bibliográfica del documento, siempre y cuando se reconozca el Derecho de Autor.

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
ESCUELA DE INGENIERÍA ELECTRÓNICA EN CONTROL Y REDES
INDUSTRIALES

El Tribunal de Trabajo de Titulación certifica que: El trabajo de investigación: “DISEÑO DE UN SISTEMA DE INTERACCIÓN MEDIANTE VISIÓN ARTIFICIAL PARA PERSONAS CON DISCAPACIDAD MOTORA Y DIFICULTAD DEL HABLA”, de responsabilidad del señor Diego Fabián Navarrete Espnoza y la Señorita Marta Cecilia Pillajo Ñauñay, ha sido minuciosamente revisado por los Miembros del Tribunal de Trabajo de Titulación, quedando autorizada su presentación.

NOMBRE	FIRMA	FECHA
Ing. Washington Luna Encalada DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA	-----	-----
Ing. Freddy Chávez V. DIRECTOR DE LA ESCUELA DE INGENIERÍA ELECTRÓNICA EN CONTROL Y REDES INDUSTRIALES	-----	-----
Ing. Javier Gavilanes DIRECTOR DEL TRABAJO DE TITULACIÓN	-----	-----
Ing. Alberto Arellano PRESIDENTE DEL TRIBUNAL	-----	-----

“Nosotros, **DIEGO FABIÁN NAVARRETE ESPINOZA** y **MARTHA CECILIA PILLAJO ÑAÑAY**, somos responsables de las ideas, doctrinas y resultados expuestos en este trabajo de titulación, y el patrimonio intelectual del mismo pertenece a la **ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO**”.

Diego Fabián Navarrete Espinoza
060461431-3

Martha Cecilia Pillajo Ñañay
060424399-8

DEDICATORIA

A mi familia que a lo largo de mi carrera estudiantil me brindaron su apoyo incondicional.

Diego Fabián Navarrete Espinoza.

A mi familia por siempre apoyarme y alentarme a cumplir mis objetivos.

Martha Cecilia Pillajo Ñauñay.

AGRADECIMIENTO

Agradecemos a nuestras familias por su apoyo, dedicación y comprensión hacia nosotros durante todo éste trayecto.

A nuestros profesores que con su conocimiento, dedicación y calidad humana marcaron de manera positiva nuestra experiencia universitaria.

A nuestros compañeros y amigos por los momentos gratos que hemos compartido juntos.

Agradecemos a nuestro tutor, Ing. Javier Gavilanes por su guía en la realización del trabajo de titulación.

ÍNDICE DE ABREVIATURAS

CCD	Charge Coupled Devices (Dispositivos de Acoplamiento de Carga)
CONADIS	Consejo Nacional de Discapacidades
CMOS	Complementary Metal Oxide Semiconductor (Semiconductor Complementario de Óxido Metálico)
CV	Computer Vision
GUI	Graphical User Interface (Interfaz Gráfica De Usuario)
OMS	Organización Mundial de la Salud
RGB	Red, Green, Blue (Rojo, Verde, Azul)
SAAC	Sistemas Aumentativos y Alternativos de Comunicación
WYSIWYG	What You See Is What You Get (Lo que ves, es lo que tienes)

TABLA DE CONTENIDO

INTRODUCCIÓN	1
PLANTEAMIENTO DEL PROBLEMA	3
JUSTIFICACIÓN	3
JUSTIFICACIÓN TEÓRICA	3
JUSTIFICACIÓN APLICATIVA	3
OBJETIVOS	5
OBJETIVO GENERAL	5
OBJETIVOS ESPECÍFICOS	5
CAPÍTULO I	6
1. MARCO REFERENCIAL TEÓRICO	6
1.1 Visión Artificial	6
1.1.1 Definición	6
1.1.2 Componentes de un sistema de visión artificial	6
1.1.2.1 <i>Iluminación</i>	6
1.1.2.2 <i>Cámara</i>	7
1.1.2.3 <i>Sistema de procesamiento</i>	8
1.1.2.4 <i>Actuadores externos</i>	8
1.1.3 Procesamiento digital de imágenes	8
1.1.3.1 <i>Adquisición de imagen</i>	9
1.1.3.2 <i>Preprocesamiento</i>	10
1.1.3.3 <i>Segmentación</i>	12
1.1.3.4 <i>Representación y descripción</i>	14
1.1.3.5 <i>Reconocimiento e interpretación</i>	15
1.2 Métodos de procesamiento de imágenes	15
1.2.1 Filtros	15
1.2.1.1 <i>Filtro suavizante</i>	15
1.2.1.2 <i>Filtro de mediana</i>	15

1.2.1.3	<i>Filtro Gaussiano</i>	16
1.2.1.4	<i>Filtro realzante</i>	16
1.2.2	Clasificador para detección de rostro	17
1.2.3	Algoritmo Haar en Cascada (Viola & Jones)	18
1.3	Discapacidad y competencias comunicativas	18
1.3.1	Discapacidad motora	19
1.3.2	Dificultad del habla	19
1.3.3	Condiciones relacionadas al transtorno de desarrollo motriz y de lenguaje	19
1.3.3.1	<i>Parálisis cerebral</i>	19
1.3.3.2	<i>Discapacidad intelectual</i>	20
1.3.3.3	<i>Autismo</i>	20
1.3.4	Competencias comunicativas	20
1.3.5	Tipos de movimientos oculares	21
1.4	Sistemas alternativos y aumentativos de comunicación (SAAC)	21
1.4.1	Sistemas sin apoyo	21
1.4.2	Sistemas con apoyo	22
1.5	Software	22
1.5.1	Python	22
1.5.2	Open CV	22
1.5.3	Qt designer	23
CAPÍTULO II		23
2.	MARCO METODOLÓGICO	24
2.1	Análisis de la capacidad gestual	24
2.2	Selección de software y hardware	25
2.2.1	Etapas de adquisición	26
2.2.1.1	<i>Cámara</i>	26
2.2.2	Etapas de procesamiento	26
2.2.2.1	<i>Python</i>	26
2.2.2.2	<i>OpenCV</i>	27

2.2.3	Etapa de interpretación de resultados	27
2.2.3.1	<i>QT</i>	27
2.3	Procesamiento digital de imagen	27
2.3.1	Adquisición de la imagen	28
2.3.2	Preprocesamiento de imagen	28
2.3.3	Descripción	31
2.3.4	Reconocimiento e interpretación	33
2.3.4.1	<i>Reconocimiento</i>	33
2.3.4.2	<i>Interpretación</i>	34
2.4	Diseño de la Interfaz gráfica de usuario.	36
2.4.1	Determinación de los elementos de la base de datos del sistema	39
2.5	Integración de la Interfaz con el programa de visión artificial	48
CAPÍTULO III		52
3.	MARCO DE RESULTADOS	52
3.1	Análisis detección movimiento de pupila	52
3.1.1	Detección figura derecha	53
3.1.1.1	<i>Análisis de gráfica</i>	53
3.1.1.2	<i>Análisis de dispersión de datos</i>	53
3.1.2	Detección figura centro	55
3.1.2.1	<i>Análisis de gráfica</i>	55
3.1.2.2	<i>Análisis de dispersión de datos</i>	56
3.1.3	Detección figura izquierda	57
3.1.3.1	<i>Análisis de gráfica</i>	58
3.1.3.2	<i>Análisis de dispersión de datos</i>	58
3.2	Pruebas de precisión del sistema	60
3.2.1	Detección figura derecha	60
3.2.2	Detección figura centro	61
3.2.3	Detección figura izquierda	62
3.3	Análisis de costos	63

3.3.1	Costos de hardware.....	63
3.3.2	Costos de software de desarrollo	63
3.3.3	Costo de instalación.....	63
3.3.4	Costo total de implementación.....	64
3.4	Análisis de resultados obtenidos	64
3.4.1	¿Se pueden aplicar técnicas de visión artificial para distinguir movimientos voluntarios de las pupilas?	64
3.4.2	¿Cómo se puede usar el movimiento de las pupilas como medio para la comunicación alternativa en personas con necesidades especiales?	65
3.4.3	¿De qué manera se logra una correcta segmentación y extracción de características de la pupila?.....	65
	BIBLIOGRAFÍA	68

ÍNDICE DE FIGURAS

Figura 1-1 Visión artificial	6
Figura 2-1 Iluminación.....	7
Figura 3-1 Procesamiento digital de imágenes	9
Figura 4-1 Adquisición de la imagen	10
Figura 5-1 Comparación entre imagen original y filtrada	10
Figura 6-1 Conversión del color	11
Figura 7-1 Histograma de una imagen	12
Figura 8-1 Transformada de Hough para detectar círculos.....	13
Figura 9-1 Filtro suavizante	15
Figura 10-1 Filtro de mediana	16
Figura 11-1 Filtro Gaussiano.....	16
Figura 12-1 Filtro realzante de Sobel.....	17
Figura 13-1 Ejemplo detección de rostro	17
Figura 14-1 Algoritmo Haar en Cascada (Viola & Jones)	18
Figura 15-1 Sistemas alternativos y aumentativos de comunicación (SAAC)	21
Figura 16-1 Entorno de trabajo de Python	22
Figura 17-1 Entorno de trabajo de Qt designer	23
Figura 1-2 Movimientos oculares con los que se presenta dificultades	25
Figura 2-2 Movimientos con los que se procede al desarrollo del sistema	25
Figura 3-2 Esquema básico de un sistema de procesamiento visual.....	26
Figura 4-2 Procedimiento para el procesamiento digital de imagen.....	27
Figura 5-2 Adquisición de la imagen mediante la cámara	28
Figura 6-2 Paletas de colores y tabla de conversión a escala de grises.....	29
Figura 7-2 Imagen Original transformada a escala de grises.....	29
Figura 8-2 Contraste imagen ecualizada vs escala de grises.	30
Figura 9-2 Segmentación del rostro y los ojos.	30
Figura 10-2 Sección del rostro recortado para extraer la información.....	31
Figura 11-2 Imagen recortada y amplificada el ojo para su análisis.	31
Figura 12-2 Código para la determinación de círculos.	31
Figura 13-2 Imagen con bordes y reconociendo la pupila.	33
Figura 14-2 Código de supresión de falsos positivos.....	33
Figura 15-2 Seguimiento de pupila hacia la derecha.	34
Figura 16-2 Seguimiento de la pupila hacia la izquierda.	35

Figura 17-2	Seguimiento de la pupila hacia el centro.....	35
Figura 18-2	Pantalla principal de QT.....	36
Figura 19-2	Botones principales.	37
Figura 20-2	Botones de desplazamiento de imágenes.	37
Figura 21-2	Elementos de señalización del sistema.....	38
Figura 22-2	Interfaz gráfica de usuario finalizada.	38
Figura 23-2	Librerías necesarias para la comunicación Python con QT.....	48
Figura 24-2	Comando para acceder a las funciones que contienen un ciclo while.....	49
Figura 1-3	Ilustración de los límites para la fijación de la mirada.....	52
Figura 2-3	Captura ingreso de conjunto de datos de la pupila en movimiento derecha.....	54
Figura 3-3	Captura ingreso de conjunto de datos de la pupila fijada en el centro	56
Figura 4-3	Captura ingreso de conjunto de datos de la pupila en movimiento izquierda.	59

ÍNDICE DE GRÁFICOS

Gráfico 1-2 Primera parte del diagrama de proceso (Interfaz gráfica de usuario)	50
Gráfico 2-2 Segunda parte del diagrama de proceso (Interfaz gráfica de usuario)	51
Gráfico 1-3 Gráfica de conjunto de datos de las posiciones captadas de la pupila en movimiento derecha	53
Gráfico 2-3 Diagrama de caja y bigotes de datos de la pupila en movimiento derecha	55
Gráfico 3-3 Gráfica de conjunto de datos de la pupila fijada en el centro	56
Gráfico 4-3 Diagrama de caja y bigotes de datos de la pupila fijada en el centro	57
Gráfico 5-3 Gráfica de conjunto de datos de la pupila en movimiento izquierda	58
Gráfico 6-3 Diagrama de caja y bigotes de la pupila en movimiento izquierda	60
Gráfico 7-3 Análisis de aciertos con mirada hacia la izquierda	61
Gráfico 8-3 Análisis de aciertos con mirada hacia la centro	62
Gráfico 9-3 Análisis de aciertos con mirada hacia la derecha	62

ÍNDICE DE TABLAS

Tabla 1-1 Base de datos del sistema	39
Tabla 1-3 Resumen valores estadísticos de conjunto de datos de las posiciones captadas de la pupila en movimiento derecha.....	54
Tabla 2-3 Resumen valores estadísticos de conjunto de datos de las posiciones captadas de la pupila en movimiento centro	57
Tabla 3-3 Resumen valores estadísticos de conjunto de datos de las posiciones captadas de la pupila en movimiento izquierda	59
Tabla 4-3 Costos de hardware	63
Tabla 5-3 Costos de software de desarrollo.....	63
Tabla 6-3 Costo de instalación.....	64
Tabla 7-3 Costo total de implementación.....	64

ÍNDICE DE ANEXOS

- A** Código desarrollado en Python
- B** Manual de usuario del sistema

RESUMEN

En el presente trabajo de titulación se desarrolló un sistema dirigido hacia las personas con discapacidad motora y ausencia de habla que aún conservan la movilidad voluntaria de los músculos de la cara como el seguimiento visual, éste dispositivo permite la selección de imágenes en un monitor utilizando técnicas de visión por computador. El proceso de visión artificial para éste sistema se basa en la identificación del movimiento de los ojos, para lo cual se hace una extracción de las características para identificar el rostro, posteriormente identificar los ojos, reconocer los movimientos de las pupilas y finalmente hacer un reconocimiento automático de éstos patrones. El sistema integra una interfaz gráfica que se compone por un menú principal de imágenes referentes a seis categorías gramaticales, cada categoría despliega un submenú que comprende varias imágenes en función de su clasificación. La pantalla del computador se dividió en tres partes, cada una con una imagen específica, la cámara ubicada en la parte superior del monitor capta la dirección de la mirada del usuario para identificar la imagen que está observando. El objetivo se selecciona mediante la fijación de la mirada en ese punto por un intervalo de tiempo específico, una vez seleccionada una categoría se puede navegar a través de ella mediante un parpadeo corto, y un parpadeo más largo permitirá regresar al menú principal. El algoritmo del sistema se desarrolló en Python por su diversidad de librerías aplicativas, además se utilizó la librería OPEN CV por la variedad de comandos que permite utilizar para el procesamiento de imágenes, la interfaz gráfica de usuario se realizó en QT Creator, por la flexibilidad de su integración con Python, todas éstas son plataformas de código libre lo cual permite que el sistema sea muy económico y pueda tener un mayor alcance social.

Palabras clave: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA>, <VISIÓN ARTIFICIAL>, <DISCAPACIDAD MOTORA>, <DIFICULTAD DEL HABLA>, <OPENCV>, <PYTHON (SOFTWARE)>, <QT CREATOR (SOFTWARE)>

ABSTRACT

In the present study a system was developed focused in people with moto disability and speech absence that still preserve the voluntary movement of face muscles like the visual detection, this device allows the selection of images in a monitor using vision techniques by computer. The artificial vision process for this system is based on the identification of eyes movement, for which an extraction of the characteristics is done to identify the face, later the eyes, recognition of pupils' movements and finally make an automatic recognition of these patterns. The system integrates a graphical interface that consists of a main menu of images referring to six grammatical categories, each category displays a submenu that includes several images according to their classification. The computer screen was divided into three parts, each one with a specific image, the camera located at the top of the monitor captures the direction of the users gaze to identify the image that is being observed. The objective is selects by fixing the gaze at that point for a specific time interval, once selected a category can be navigated through each by a short blink, as a longer blink will allow go back to the main menu, the algorithm of the system was developed in Python due to its diversity of library applications, and the OPENCV library was used for the variety of commands that allows to use for image processing. The graphic user interface was made in QT Creator due to the flexibility of its integration with Python, all these are open source platforms which permits the system to be very reasonably priced and have greater social reach.

KEY WORDS: Engineering sciences and technology, Artificial vision, Motor disabilities, Speech difficulties, OPENCV, PYTHON (SOFTWARE), QTCREATOR (SOFTWARE).

INTRODUCCIÓN

De acuerdo al objetivo planteado en el Plan nacional del buen vivir, auspiciar la igualdad, la cohesión, la inclusión y la equidad social, se desarrolló un sistema dirigido hacia las personas con discapacidad motora y ausencia de habla, que aún conservan la movilidad voluntaria de los músculos de la cara como el seguimiento visual, éste dispositivo permite el manejo de un apuntador en un monitor basado en el movimiento de sus ojos, utilizando técnicas de visión artificial.

Éste sistema se espera tendrá un impacto positivo en las personas con las características mencionadas anteriormente, al lograr un mejor desempeño en la forma de comunicación, lo que ayuda al proceso de inclusión y aprendizaje.

A nivel académico se vienen desarrollando proyectos con tratamientos de imágenes y sistemas sensoriales para detectar movimientos voluntarios de articulaciones del cuerpo humano para controlar la posición del cursor en una pantalla a continuación se presentan algunas de las propuestas más destacadas en el ámbito internacional.

En Estados Unidos, el sistema de “Camera Mouse” ha sido desarrollado para proporcionar acceso a la computadora para las personas con discapacidades severas. El sistema realiza el seguimiento de los movimientos del usuario del ordenador con una cámara de video y las traduce en los movimientos del puntero del ratón en la pantalla (Betke, Gips y Fleming, 2002)

Universidad de Michigan en Estados Unidos, debido a que muchas personas con discapacidad todavía tienen un control significativo de su movimiento de la cabeza, el seguimiento de la misma es una elección lógica. Se ha desarrollado un software de seguimiento no intrusivo de la cabeza para controlar el cursor, junto con el reconocimiento de parpadeo del ojo para emular el clic del ratón. El sistema, llamado NaviGaze, también permite el uso de un ratón y un teclado estándar, por lo que es ideal para su uso en entornos de computación públicos (O'Grady, Cohen, Beach y Moody, 2004)

Universidad Estatal de Georgia en Estados Unidos, “Sistema de control difuso del cursor del ratón para los usuarios de ordenador con daño en la Médula Espinal”. A pesar de no utilizar visión artificial, propone el uso de un control difuso para mejorar el desempeño de un sistema que posiciona el cursor mediante un dispositivo neumático para la boca y utilizando técnicas de escaneo de la pantalla (Surdilovic, 2006)

Universidad Técnica "Gheorghe Asachi" en Rumania, Sistema de comunicación basado en el seguimiento de los ojos para los pacientes con importantes Discapacidades Neuro-locomotoras desarrolla un sistema de comunicación inalámbrica para pacientes con dificultad para hablar, compuesto por un computador de usuario, un servidor de mensajes y terminales móviles. Una

cámara sujeta a unas gafas, capta la dirección de la mirada del paciente para identificar el mensaje que está observando. El mensaje se selecciona mediante un parpadeo y es enviado por el servidor, al dispositivo móvil del profesional a cargo (Lupu, Bozomitu y Ungureanu, 2011)

Así también en la Latinoamérica existen proyectos de éste tipo desarrollados a nivel académico: Universidad Distrital “Francisco José de Caldas” en Colombia, “Dispositivo apuntador mediante visión artificial, adecuado para usuarios de computador con discapacidad motora en miembros superiores” (Peña y Rodríguez, 2015)

En el país se han desarrollado proyectos de éste tipo orientados al ámbito social, donde destaca los siguientes:

Escuela Politécnica Nacional, “Diseño e implementación de un sistema traductor de lenguaje de señas de manos a un lenguaje de texto mediante visión artificial en un ambiente controlado” (Chiguano, Moreno y Corrales, 2011)

Escuela Politécnica Nacional “Mouse para personas con Discapacidad Motriz”.

Escuela Superior Politécnica del Litoral, “Valoración y análisis de los movimientos de las manos de un paciente de Parkinson según la escala UPDRS usando técnicas de visión artificial con Kinect” (Rubio y Vintimilla, 2015)

Escuela Superior Politécnica del Litoral, “Seguimiento y Análisis del Movimiento de las Extremidades Superiores Aplicado a la Rehabilitación Física de un Paciente Usando Técnicas de Visión Artificial” (Velarde, Perugachi, Vintimilla y Romero, 2016)

El problema nace de la necesidad que presentan las personas con discapacidad motora y dificultad del habla al momento de interactuar con su entorno, debido a la limitada la capacidad de comunicación que ofrecen los recursos existentes, además de la completa dependencia que implica utilizar cualquiera de esos recursos, debido a que siempre deberá existir un interlocutor que realice las funciones de intermediario entre el paciente y el medi

PLANTEAMIENTO DEL PROBLEMA

JUSTIFICACIÓN

JUSTIFICACIÓN TEÓRICA

El presente proyecto es un sistema de comunicación para personas con discapacidad motora y dificultad del habla usando técnicas de visión artificial.

El proceso de visión artificial para éste sistema se basa en la identificación del movimiento de los ojos, para lo cual se debe hacer una extracción de las características para identificar el rostro, posteriormente identificar los ojos, reconocer los movimientos y finalmente hacer un reconocimiento automático de los patrones, ésta última etapa puede ser analizada desde varios enfoques; heurística, matemático, determinístico, estadístico, entre otros.

El algoritmo del sistema se desarrollará en Python por su diversidad de librerías aplicativas, además se utilizará la librería OPEN CV dedicada a la visión artificial, la interfaz gráfica de usuario se hará en QT por la facilidad de programación y por la variedad de comandos que se puede utilizar para el procesamiento de imágenes.

JUSTIFICACIÓN APLICATIVA

En el país no se encuentran datos estadísticos específicos sobre personas que tengan ambas condiciones discapacidad motora y dificultad del habla, tanto la Vicepresidencia de la República como el Consejo Nacional de Discapacidades (CONADIS), carecen de esta información. Sin embargo, existen registros de las discapacidades físicas e intelectuales donde se influye esta condición. Según el CONADIS hay un total de 408,021 personas censadas discapacitadas en todo el Ecuador, de las cuales 47% de las discapacidades son físicas, la discapacidad intelectual corresponde al 23% y la discapacidad de lenguaje con un 1%. Se considera una discapacidad física e intelectual, ya que afecta a la distintas áreas del cerebro que condicionan distintas áreas del cuerpo como puede ser el habla, la función motora, cognitiva, perceptiva y sensorial. Además se debe tener en cuenta que la discapacidad se presenta en varios grados de afección por lo que en éste trabajo se considera únicamente al grado más severo, aquel donde la persona necesita de asistencia mecánica o de terceras personas para cualquier actividad.

De acuerdo con los objetivos planteados, su resultado permite dar solución a la limitada interacción debido a múltiples discapacidades, con la ayuda de éste sistema, las personas con discapacidad motora y dificultad de habla, que aún conservan la movilidad voluntaria de los músculos de la cara como el seguimiento visual, podrán comunicarse de mejor forma gracias al movimiento de sus ojos, se elige analizar el movimiento de los ojos debido a que varios de éstos pacientes implican dificultad para mover la cabeza o alguna extremidad, además la libertad de movimiento de la mirada se ve limitada al eje horizontal, el usuario debe estar sentado frente a la computadora a una distancia prudencial de tal forma que la cámara pueda enfocar los ojos de manera objetiva.

El sistema integra una interfaz gráfica que se compone por un menú principal de imágenes referentes a seis categorías gramaticales (Personas, Verbos, Descriptivos, Nombres, Miscelánea, Social) cada categoría despliega un submenú que comprende varias imágenes en función de su clasificación.

La pantalla del computador se dividirá en tres partes, cada una con una imagen específica, una cámara sujeta a la parte superior del monitor o la cámara integrada al monitor en caso de las computadoras portátiles, captará la dirección de la mirada del usuario para identificar la imagen que está observando. La imagen se selecciona mediante la fijación de la mirada en ese punto por un intervalo de tiempo específico, una vez seleccionada una categoría se puede navegar a través de ella mediante un parpadeo de un tiempo corto, y un parpadeo de un tiempo mas largo permitirá regresar al menú principal.

OBJETIVOS

OBJETIVO GENERAL

Diseñar un sistema de interacción mediante visión artificial para personas con discapacidad motora y ausencia de habla.

OBJETIVOS ESPECÍFICOS

Analizar la capacidad gestual de las personas con las que vamos a trabajar para definir las variables de movimiento que se usaran como comandos.

Seleccionar la configuración de la plataforma hardware/software más apropiada para desarrollar el proyecto (tipo de cámara, interface de comunicación, lenguaje de programación).

Desarrollar la parte de software que permita receptar y entender los movimientos oculares que realice el usuario.

Implementar la interfaz gráfica que contiene el conjunto de elementos necesarios para la comunicación fundamentado en sistemas de comunicación aumentativa basado en pictogramas.

Validar el sistema realizando pruebas para observar la factibilidad y desempeño del sistema.

CAPÍTULO I

1. MARCO REFERENCIAL TEÓRICO

1.1 Visión Artificial

1.1.1 Definición

La visión artificial es una disciplina científica y tecnológica cuyo objetivo es obtener información digitalizada que pueda ser procesada por una máquina mediante la adquisición, procesamiento y análisis de imágenes, se ilustra la en la figura 1-1.

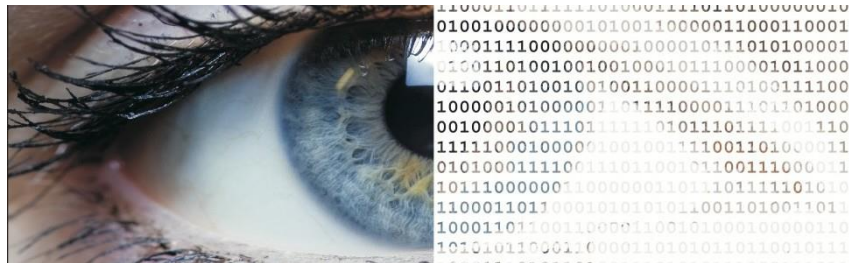


Figura 1-1 Visión Artificial

Fuente: <https://visartblog.wordpress.com/tag/definicion/>

1.1.2 Componentes de un sistema de visión artificial

1.1.2.1 Iluminación

El propósito de la iluminación es el de controlar la forma en que la cámara va a ver el objeto, juega un papel muy importante dentro de la visión artificial, ya que simplifica considerablemente el posterior procesamiento de la escena captada.

Los objetivos de la iluminación son: optimizar el contraste, normalizar cualquier variación de la iluminación ambiente y simplificar el procesamiento posterior de la imagen.

El tipo de iluminación dependerá del entorno en que este implementado el sistema de visión artificial.

En la figura 2-1 se puede apreciar una posible forma de iluminación en un ambiente controlado.

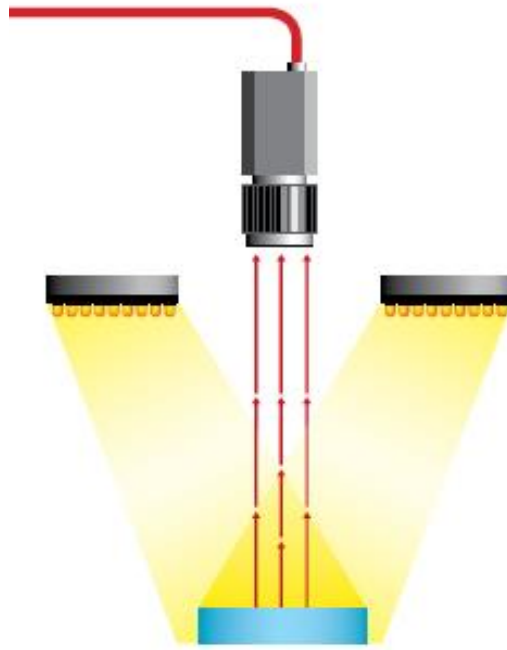


Figura 2-1 Iluminación

Fuente: <https://www.infaimon.com/enciclopedia/iluminacion-frontal/>

1.1.2.2 Cámara

La cámara es un dispositivo que utilizando un juego de lentes y un sensor recibe la luz reflejada por la escena y la utiliza para generar imágenes; una cámara desempeña la función de sensor en un sistema de visión artificial, de acuerdo a la aplicación y a las necesidades se selecciona el tipo de cámara y óptica más adecuado.

El lente en una cámara se utiliza para transmitir la luz al sensor de la cámara de una forma controlada para poder obtener una imagen enfocada de uno o varios objetos.

Los sensores contenidos en una cámara son componentes sensibles a la luz que modifican su señal eléctrica en función de la intensidad luminosa que perciben. La tecnología más habitual en este tipo de sensores es el CCD (Charge Coupled Devices o Dispositivos de Acoplamiento de Carga), en este tipo de sensores la señal eléctrica que transmiten los elementos fotosensibles es función de la intensidad luminosa que reciben, su espectro, y el tiempo de integración (tiempo durante el cual son sensibles a la luz incidente). Otra tecnología son los sensores CMOS (Complementary Metal Oxide Semiconductor), estos son de menor tamaño y precio en comparación con los CCD; este tipo de sensores se adapta mejor al brillo existente en su entorno.

1.1.2.3 *Sistema de procesamiento*

Suele ser una computadora y es el encargado de recibir las imágenes e implementar las funciones de tratamiento de la imagen de acuerdo al tipo de análisis a realizar, está compuesto por:

- **Tarjeta de adquisición**
Permite transferir la imagen de la cámara a la memoria de la computadora con el fin de que ésta pueda realizar el procesamiento adecuado; los aspectos importantes a considerar en la tarjeta de adquisición son: velocidad de transmisión, formato de los datos, la capacidad de preproceso de la imagen, la velocidad de transferencia de la imagen hacia la memoria de la computadora y la capacidad de controlar la cámara en tiempo real.
- **Algoritmos de procesado**
Es la parte inteligente del sistema, consiste en aplicar las transformaciones necesarias y extracciones de información de las imágenes capturadas, con el fin de obtener los resultados para los que haya sido diseñado.
- **Interface**
Después de realizar el procesamiento de la imagen y a través de los algoritmos es común mostrar los resultados obtenidos de acuerdo a las necesidades requeridas a un usuario final, por lo que se crea una interfaz para mostrarlos.

1.1.2.4 *Actuadores externos*

Es el conjunto de elementos que muestran los resultados obtenidos del procesamiento de la imagen; estos pueden ser: monitores, robots, dispositivos neumáticos e hidráulicos, autómatas programables, etc.

1.1.3 *Procesamiento digital de imágenes*

Los sistemas de visión artificial se basan en el procesamiento digital de imágenes, el cual es el conjunto de técnicas que se aplican sobre imágenes digitales con el fin de mejorar la calidad o facilitar la búsqueda de información dentro de las mismas.

En la figura 3-1, se observa los posibles componentes para el procesamiento digital de señales y sus etapas.

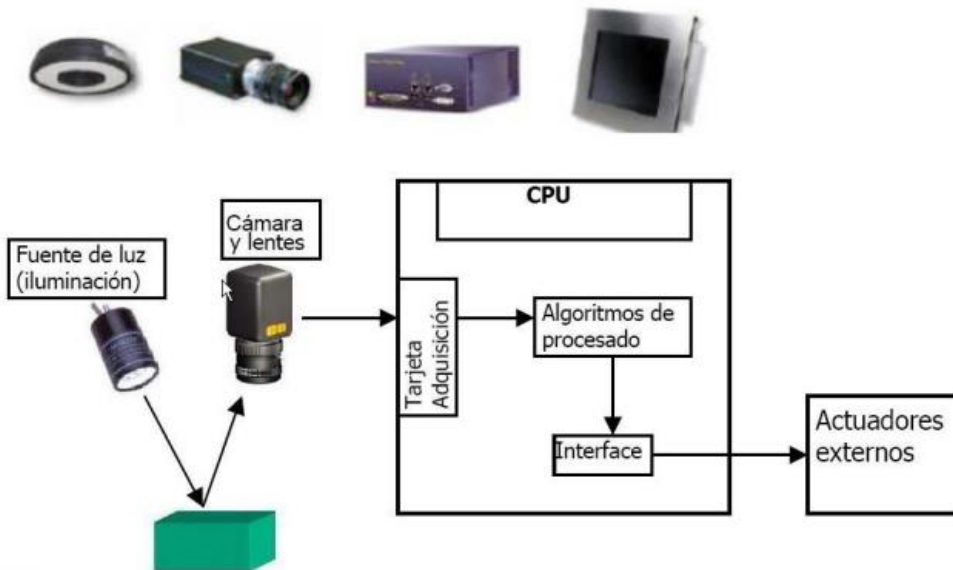


Figura 3-1 Procesamiento digital de imágenes

Fuente: <https://www.infaimon.com/enciclopedia/iluminacion-frontal/>

1.1.3.1 Adquisición de imagen

Ésta etapa es la más sencilla, puesto que en esta se realiza con una cámara la obtención de una imagen y su posterior digitalización.

Las imágenes digitales son señales discretas, que suelen tener origen en una señal continua, por ejemplo: una cámara digital toma imágenes del mundo real que es continuo.

Una imagen digital puede definirse como una matriz bidimensional (x, y) , en la que cada par de coordenadas representa un elemento contenido en la imagen, los elementos que componen a una imagen son llamados píxeles, cada uno de los cuales tiene un valor (propiedad del punto que se representa) y una posición específica. El término píxel (Picture Element/Elemento de Imagen), se trata de la unidad mínima de información de una imagen, la cual aparece como un punto en la pantalla o en una hoja impresa; cada píxel se compone de tres registros de color: rojo, verde y azul; mediante la combinación de estos el píxel adopta un color en particular.

En la figura 4-1 se aprecia un ejemplo de la adquisición de información mediante una cámara y el ojo humano.



Figura 4-1 Adquisición de la imagen

Fuente: <https://goo.gl/SQFA8X>

1.1.3.2 *Preprocesamiento*

Transformaciones y filtrado de imágenes

Cuando se adquiere una imagen, por lo general tiene cierta cantidad de ruido debido ya sea a deficiencias en la iluminación, a la óptica de la cámara, al medio de transmisión, etc. Generalmente el ruido se manifiesta como píxeles aislados que toman un nivel de color diferente al de sus vecinos.

Esta aparición de ruido como se puede apreciar en la figura 5-1 es necesario un preprocesamiento de la imagen con el fin de corregirlo; el objetivo del preprocesamiento es mejorar la calidad de la imagen adquirida y realzar las características de la misma.

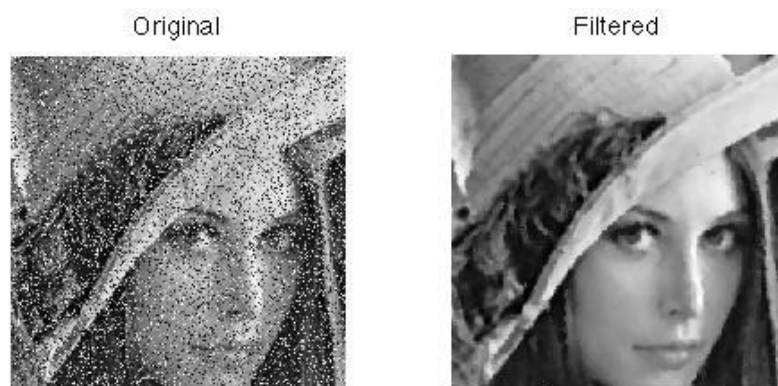


Figura 5-1 Comparación entre imagen original y filtrada

Fuente: : <https://goo.gl/sb0lkz>

Algunas de las técnicas de preprocesamiento de imágenes más habituales son:

- Conversión del color. Dependiendo de la aplicación del sistema de visión artificial la imagen se suele convertir a escala de grises, RGB, binaria o indexada; para resaltar sus características. En la figura 6-1 se observa la conversión de una imagen RGB a escala de grises.



Figura 6-1 Conversión del color

Fuente: <https://goo.gl/y1bj4W>

- Transformaciones aritmético-lógicas. Son las más usadas en un sistema de tratamiento de imágenes, ya que son las que se utilizan para leer y dar valores a los píxeles de las imágenes; las operaciones básicas son:
 - Conjunción. Operación lógica AND entre los bits de dos imágenes; se usa para borrar píxeles en una imagen.
 - Disyunción. Operación lógica OR entre los bits de dos imágenes; se usa para añadir píxeles a una imagen.
 - Negación. Inversión de los bits que forman una imagen; se usa para obtener el negativo de una imagen.
 - Suma. Suma de los valores de los píxeles de dos imágenes.
 - Resta. Resta de los valores de los píxeles de dos imágenes.
 - Multiplicación. Multiplicación de los valores de los píxeles de una imagen por los de otra; se usa para añadir textura a una imagen.
 - División. División de los valores de los píxeles de una imagen entre los de otra.
- Transformaciones geométricas. Modifican el aspecto visual de la imagen transformando los valores de una imagen, tal y como podría observarse desde otro punto de vista; algunas de estas transformaciones son:
 - Rotación. Giro de los píxeles de una imagen en torno a un origen.
 - Traslación. Movimiento de los píxeles de una imagen.
 - Escalado. Cambio del tamaño de una imagen.

- Transformación del histograma. El histograma de una imagen es una gráfica en la que se representa la frecuencia de píxeles existentes para cada nivel de cuantificación determinado: habitualmente en el eje de abscisas se disponen los diferentes niveles de cuantificación de valores que pueden tomar los píxeles de la imagen, mientras el eje de las ordenadas refleja el número de píxeles existentes para cada nivel de cuantificación ejemplificada en la figura 7-1. La transformación del histograma se usa para: aclarar u oscurecer una imagen pero manteniendo la relación entre los valores de cada nivel, mejorar el contraste de la imagen, reducir al máximo el ruido, etc.

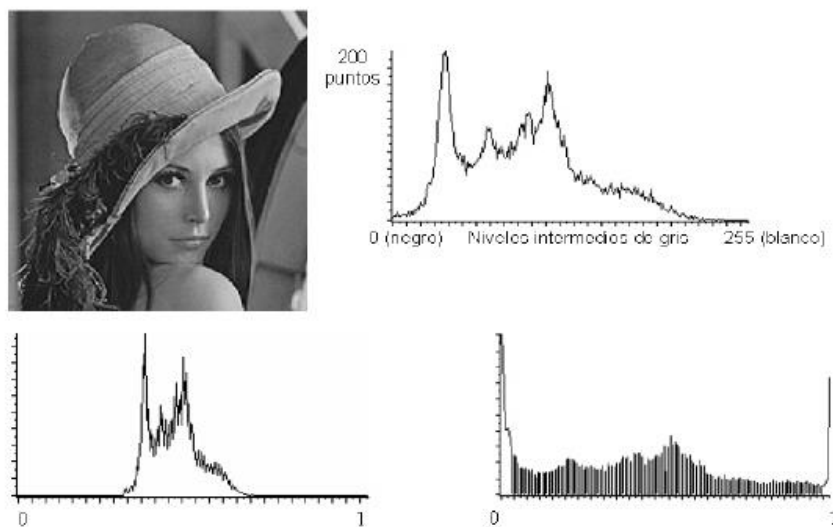


Figura 7-1 Histograma de una imagen

Fuente: <https://goo.gl/mEcxD>

1.1.3.3 Segmentación

Es el proceso que divide a una imagen en objetos que sean de nuestro interés con respecto a una o más características (como por ejemplo el brillo o el color) con el fin de facilitar un posterior análisis.

La segmentación debe verse como un proceso que a partir de una imagen, produce otra en la que cada píxel tiene asociada una etiqueta distintiva del objeto al que pertenece. Así, una vez segmentada una imagen, se podría formar una lista de objetos consistentes en las agrupaciones de los píxeles que tengan la misma etiqueta.

El proceso de segmentación de una imagen depende del problema que se desee resolver; por ejemplo, sobre una imagen de una página de texto se pueden segmentar las líneas de texto (si el objetivo es localizar la estructura de los párrafos), o las palabras y los caracteres que las forman o los logotipos y

membretes (si se desea clasificar el documento), etc. Por ello, dentro de una misma imagen pueden realizarse diversas segmentaciones.

Una forma de segmentación es la detección de círculos o circunferencias, apreciada de mejor forma en la figura 8-1.



Figura 8-1 Transformada de Hough para detectar círculos

Fuente: <https://goo.gl/rtr5jJ>

Los diferentes objetos que aparecen en una imagen pueden ser localizados atendiendo a aspectos como sus contornos o su textura; algunos de los tipos de segmentaciones más comunes se describen a continuación:

- Segmentación basada en umbralización. La umbralización es un proceso que permite convertir una imagen de niveles de gris o de color en una imagen binaria, de tal forma que los objetos de interés se etiqueten con un valor distinto al de los píxeles del fondo.
- Segmentación basada en detección de los contornos. Usa la información proporcionada por las fronteras de los objetos que aparecen en una imagen; al encontrar las fronteras de los objetos se pueden diferenciar estos del fondo.
- Segmentación basada en propiedades locales de las regiones. Determinan zonas dentro de una imagen basándose en criterios de similitud y proximidad entre los píxeles de la misma. Este tipo de segmentación es adecuada en imágenes con ruido y en donde los contornos son difíciles de localizar.
- Segmentación basada en el color. Dentro de esta segmentación se suele convertir la imagen original a otro tipo de imagen requerida (RGB, indexada, intensidad de grises o binaria), de acuerdo a lo que se requiera, para así identificar diversas regiones dentro de una imagen de acuerdo a su color o realizar posteriormente algún otro tipo de segmentación.
- Segmentación basada en la textura. En este enfoque se definen modelos de texturas para reconocer agrupaciones de estos dentro de la imagen.

- Segmentación basada en el movimiento. El movimiento puede ser una potente herramienta para la segmentación de objetos animados sobre fondos estáticos; esta segmentación consiste en el estudio de la imagen resultante de la resta de dos imágenes consecutivas de una secuencia animada.

Cada uno de estos tipos de segmentación suele resultar insuficiente para describir los objetos contenidos en una imagen, es por ello que suelen combinarse de acuerdo a las necesidades requeridas.

1.1.3.4 *Representación y descripción*

Una vez que los objetos de interés contenidos en una imagen son aislados, el siguiente paso es la obtención de características convenientes para describir cada objeto. Las propiedades que deben tener estos descriptores son:

- Ser capaces de hacer una adecuada discriminación, deben proporcionar valores numéricos diferentes para cada objeto.
- Ser fiables, debe de haber cambios numéricos pequeños para objetos iguales.
- Se calculen en un tiempo aceptable, de manera que sean utilizables en problemas de tiempo real.
- La descripción del objeto tiene que ser lo más completa posible y no presentar ambigüedades.

Entre los descriptores comúnmente usados en el procesamiento digital de imágenes se tiene:

- Descriptor de color. El color es una característica importante cuando se requiere de la clasificación y descripción de objetos.
- Descriptor de textura. Se basan en la vecindad, ya que la textura se define para regiones y no para píxeles individuales, es muy utilizado para el reconocimiento de objetos.
- Descriptor de rasgos geométricos de un objeto. Son mediciones que pueden ser usadas para describir un objeto, se obtienen a partir de los píxeles que componen la región del objeto o los píxeles obtenidos del contorno del objeto; dentro de estos descriptores existe:
 - Área. Es igual a la suma de todos los píxeles correspondientes al objeto dentro de la imagen.
 - Perímetro. Se obtiene al sumar solamente los píxeles del contorno del objeto.
 - Centroide. Es un punto geométrico que se refiere a la parte central del objeto.
 - Factor de compacidad. Se define como el perímetro al cuadrado del objeto dividido por 4π veces su área; el factor de compacidad será cercano a 1 para objetos circulares, mayor a 1 para objetos cuadrados y para otro tipo de objetos se obtendrán valores más

grandes. La importancia de este rasgo característico radica en que es invariante a traslaciones, rotaciones y cambios de escala.

1.1.3.5 *Reconocimiento e interpretación*

La última etapa en el procesamiento digital de imágenes es la que comprende el reconocimiento e interpretación de forma; en esta etapa se asigna un significado, se etiqueta o asigna un nombre a los objetos contenidos en la imagen basándose en las características obtenidas en la etapa de descripción.

1.2 Métodos de procesamiento de imágenes

1.2.1 *Filtros*

1.2.1.1 *Filtro suavizante*

Estos filtros atenúan las componentes de alta frecuencia presentes en la imagen, las cuales son responsables de los bordes y de los detalles finos, por este motivo utilizados para la reducción de ruido. M.C. (2011).

La figura 9-1 presenta un ejemplo de los filtros que se puede utilizar dentro de OpenCV para suavizar el ruido.

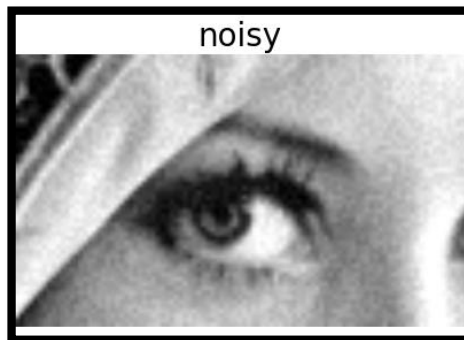


Figura 9-1 Filtro suavizante

Fuente: <https://goo.gl/Xxd57a>

1.2.1.2 *Filtro de mediana*

Para calcular la mediana de un conjunto de valores estos se debe arreglar en orden ascendentes, el color que se situó en la mitad de dicho arreglo es la mediana del conjunto.

En la figura 10-1 se ve representado “Median filter”, un filtro que reemplaza los pixeles obtenidos por la mediana de los pixeles vecinos con una vecindad cuadrada.

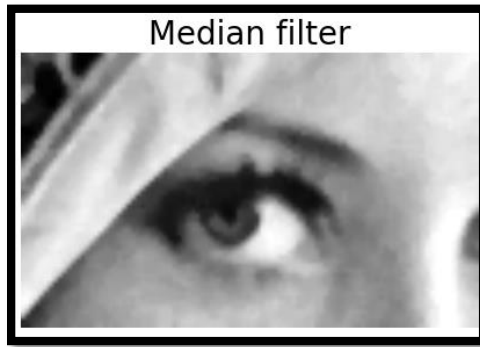


Figura 10-1 Filtro de mediana

Fuente: <https://goo.gl/Xxd57a>

1.2.1.3 *Filtro Gaussiano*

Este filtro tiene un mejor desempeño que el filtro de mediana debido a que da mayor prioridad al pixel central haciendo convolución de cada punto de la matriz de entrada con un núcleo gaussiano apreciada de mejor forma en la figura 11-1.



Figura 11-1 Filtro Gaussiano

Fuente: <https://goo.gl/Xxd57a>

1.2.1.4 *Filtro realzante*

Como se aprecia en la figura 12-1 el objetivo principal de estos filtros es destacar los detalles finos de la imagen, una de sus aplicaciones más importantes es la detección de bordes. En esta investigación se utilizará el filtro: Sobel, Laplaciano Gaussiano y HighBoots. Revathi (2012).



Figura 12-1 Filtro realzante de Sobel

Fuente: http://fileadmin.cs.lth.se/cs/Personal/Calle_Lejdfors/pyip/edges.png

1.2.2 Clasificador para detección de rostro

Inicialmente el problema de detección de rostro en los sistemas de reconocimiento no recibió la atención necesaria y se partía de que el rostro ya había sido detectado, fue solo en la década de los ochenta que surgieron los primeros algoritmos, basados en técnicas heurísticas y antropométricas, y en la década de los noventa cuando el desarrollo de algoritmos de detección de rostros inició su crecimiento, poniéndose una gran variedad de técnicas, desde algoritmos básicos de detección de bordes hasta algoritmos compuestos de alto nivel que utilizan métodos avanzados de reconocimiento de patrones.

En la figura 13-1 se observa un ejemplo de la detección de rostro a través de una cámara en un sistema de visión.

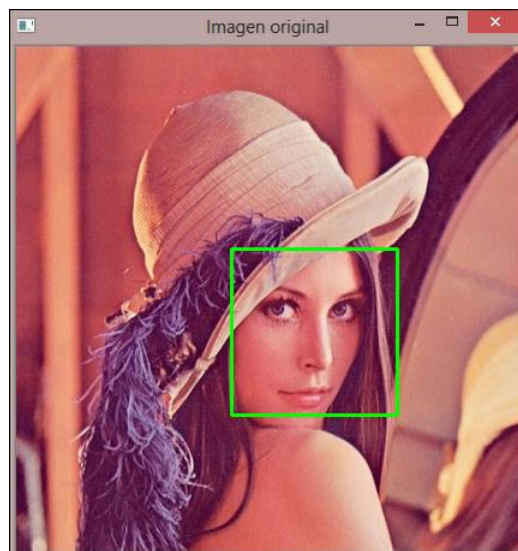


Figura 13-1 Ejemplo detección de rostro

Fuente: <http://acodigo.blogspot.com/2013/06/deteccion-de-rostros.html>

1.2.3 Algoritmo Haar en Cascada (Viola & Jones)

Es un enfoque para la detección de rostros que minimiza el tiempo de cálculo mientras se logra una alta precisión de detección. El enfoque se usó para construir un sistema de detección de rostros que es aproximadamente quince veces más rápido que cualquier enfoque anterior a éste.

La forma en la que trabaja el algoritmo se ve de mejor manera en la figura 14-1.

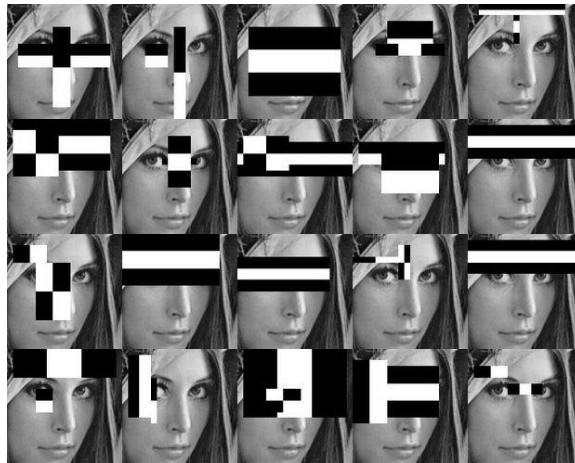


Figura 14-1 Algoritmo Haar en Cascada (Viola & Jones)

Fuente: <https://goo.gl/DVwUzH>

1.3 Discapacidad y competencias comunicativas

Según la OMS (Organización Mundial de la Salud) discapacidad es un término general que abarca las deficiencias, las limitaciones de la actividad y las restricciones de la participación. Las deficiencias son problemas que afectan a una estructura o función corporal; las limitaciones de la actividad son dificultades para ejecutar acciones o tareas, y las restricciones de la participación son problemas para participar en situaciones vitales.

La inclusión educativa es un tema de interés mundial esto hace que se busque desarrollar estrategias de comunicación inclusiva, existen centros especializados para personas en ésta situación, como también centros educativos que atienden a la diversidad. Los métodos utilizados se basan en los Sistemas SAAC. El entrenamiento de una persona para que pueda hacer uso de los SAAC consiste en trabajar su atención visual para que pueda desarrollar la capacidad de observación y así conseguir captar la contingencia causa – efecto y finalmente entrenar su capacidad de señalar, éste último paso estará condicionado por sus habilidad motriz ya que la señalización directa requiere de destrezas motrices estructuradas, de movimientos complejos y coordinados, no siempre posibles en personas que tengan muy afectadas dichas capacidades.

El uso de dispositivos de alta tecnología y de algoritmos especializados, permite hoy en día ampliar la posibilidad de alcanzar objetivos mas ambiciosos en escenarios mas complejos, además el uso de estos, crea un mayor interés en los usuarios gracias a interfaces novedosas y la interpretación del mensaje de manera llamativa como la emisión de sonidos.

1.3.1 *Discapacidad motora*

Por discapacidad motora se entiende aquella que abarca todas las alteraciones o deficiencias orgánicas del aparato motor o de su funcionamiento, que afectan al sistema óseo, articulaciones, nervios y/o músculos. Las personas afectadas por ellas presentan una clara desventaja en su aparato locomotor, determinado por limitaciones posturales, de desplazamiento, coordinación y manipulación, pudiendo integrar dos o más de éstas. Pueden ir acompañadas de alteraciones sensoriales, perceptivas, de lenguaje y, en un porcentaje alto, conservan su capacidad intelectual.

Se puede clasificar en los siguientes trastornos:

- Físico periféricos: Afectación en extremidades, articulaciones huesos y músculos.
- Trastornos neurológicos: Daño que se origina en el cerebro (corteza motora cerebral) que se encarga de procesar y enviar la información de movimiento al resto del cuerpo, por lo tanto origina dificultades en el movimiento, sensaciones y control de ciertas partes del cuerpo.

1.3.2 *Dificultad del habla*

Un “trastorno del habla o lenguaje” se refiere a los problemas de la comunicación u otras áreas relacionadas, tales como las funciones motoras orales. Estos atrasos y trastornos varían desde simples substituciones de sonido hasta la inhabilidad de comprender o utilizar el lenguaje o mecanismo motor-oral para el habla y alimentación. Algunas causas de los impedimentos del habla o lenguaje incluyen la pérdida auditiva, trastornos neurológicos, lesión cerebral, discapacidad intelectual, abuso de drogas, impedimentos tales como labio leporino, y abuso o mal uso vocal. Sin embargo, con mucha frecuencia se desconoce la causa.

1.3.3 *Condiciones relacionadas al transtorno de desarrollo motriz y de lenguaje*

1.3.3.1 *Parálisis cerebral*

En un nuevo consenso internacional, se propone como definición: “La parálisis cerebral describe un grupo de trastornos del desarrollo psicomotor, que causan una limitación de la actividad de la persona,

atribuida a problemas en el desarrollo cerebral del feto o del niño. Los desórdenes psicomotrices de la parálisis cerebral están a menudo acompañados de problemas sensitivos, cognitivos, de comunicación y percepción, y en algunas ocasiones, de trastornos del comportamiento”.

1.3.3.2 *Discapacidad intelectual*

La discapacidad intelectual es una alteración en el desarrollo del ser humano caracterizada por limitaciones significativas tanto en el funcionamiento intelectual como en las conductas adaptativas y que se evidencia antes de los 18 años de edad. Afecta alrededor del 2% de la población general.

Genera anomalías en el proceso de aprendizaje entendidas como la adquisición lenta e incompleta de las habilidades cognitivas durante el desarrollo humano que conduce finalmente a limitaciones sustanciales en el desarrollo corriente.

1.3.3.3 *Autismo*

El autismo infantil produce alteraciones intelectuales que a menudo son muy difíciles de diferenciar del retraso mental. Sus principales características son: ausencia de interacción social, alteraciones profundas en el lenguaje, no acorde con las capacidades intelectuales, insistencia en comportamientos estereotipados.

Los menores que padecen retraso mental suelen exhibir un retraso en el desarrollo lingüístico, pero siguen las mismas etapas del niño normal. El autismo infantil y el retraso mental llegan a estar relacionados y, de hecho, se ha considerado que aproximadamente tres cuartas partes de niños autistas funcionan como adultos con retraso mental.

1.3.4 *Competencias comunicativas*

La comunicación implica la interacción con el medio social y a su vez permite cierta autonomía, ya que a través de la comunicación se puede expresar nuestras ideas, sentimientos, pensamientos y deseos.

La competencia comunicativa es saber comunicarse en un campo del conocimiento y un saber aplicarlo; saberes que comprenden conocimientos, habilidades, actitudes y valores (precondiciones, criterios, usos, reglas y normas) para realizar actos comunicativos eficientes en un contexto determinado, según necesidades y propósitos. Las competencias comunicativas es algo con lo que cuentan todas las personas, y en el caso de las personas con discapacidades que dificultan su comunicación, se busca lograr que se pueda realizar el acto comunicativo, que éstos individuos interactúen; para ello es que se recurre a los sistemas aumentativos y alternativos de comunicación.

1.3.5 *Tipos de movimientos oculares*

- Fijación: Es la habilidad que tiene la fovea de mantener en dicho punto de retina la imagen de un objeto de manera estable, mediante pequeños micromovimientos imperceptibles. De esta manera, permite ver dicha imagen clara.
- Seguimientos: Es la habilidad de seguir un objeto en movimiento con nuestros ojos. Lo perfecto es hacerlo con la fovea para que la imagen se mantenga enfocada.
- Sacádicos: Es una habilidad más compleja, que permite que nuestros ojos salten de un objeto a otro. Si dicho salto se realiza con la fovea, será más preciso y se verá la imagen más clara.

1.4 **Sistemas alternativos y aumentativos de comunicación (SAAC)**

Cualquier forma de comunicación distinta del habla y empleada por una persona en contextos de comunicación cara a cara. El uso de signos manuales y gráficos, el sistema Morse, la escritura, etc., son formas alternativas de comunicación para una persona que carece de la habilidad de hablar.

En la figura 15-1 muestra imágenes pertenecientes al sistema alternativo y aumentativo de comunicación (SAAC),



Figura 15-1 Sistemas alternativos y aumentativos de comunicación (SAAC)

Fuente: <https://goo.gl/V5w4pp>

1.4.1 *Sistemas sin apoyo*

No precisan de ningún elemento físico externo al emisor. Es el propio emisor, quien a través de su propio cuerpo (gestos y signos manuales) configura y transmite el mensaje. Dentro de estos, se encuentran el lenguaje de señas, gestos, el sistema bimodal y la palabra complementada.

1.4.2 *Sistemas con apoyo*

Requieren de algún tipo de apoyo o soporte físico para su utilización, en este caso se usan elementos tangibles y símbolos gráficos (fotografías, pictogramas...), además de los soportes necesarios, los cuales pueden tratarse desde sencillos soportes como cartulinas, tableros de madera o plástico, hasta tecnologías más sofisticadas, como las Tablet y las computadoras. Dentro de estos, se encuentran los pictogramas, elementos no estructurado (por ej: uso de fotografías) y diferentes software.

1.5 **Software**

1.5.1 *Python*

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

En la siguiente figura se ve un ejemplo del código desde el editor de código y texto en el entorno de trabajo Python.



Figura 16-1 Entorno de trabajo de Python

Fuente: <http://programaenlinea.net/optimiza-operaciones-listas-python/>

1.5.2 *Open CV*

OpenCV es una librería abierta de visión artificial que tiene soporte para distintos tipos de sistemas operativos, contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión,

como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estérea y visión robótica.

1.5.3 *Qt designer*

Qt es un entorno de trabajo multiplataforma orientado a objetos ampliamente usado para desarrollar programas (software) que utilicen interfaz gráfica de usuario, así como también diferentes tipos de herramientas para la línea de comandos y consolas para servidores que no necesitan una interfaz gráfica de usuario.

Qt Designer es la herramienta Qt para diseñar y construir interfaces gráficas de usuario (GUI) con Qt Widgets. Puede componer y personalizar sus ventanas o cuadros de diálogo de una manera WYSIWYG ("lo que ves es lo que se obtiene") y probarlos con diferentes estilos y resoluciones.

Los widgets y formularios creados con Qt Designer se integran perfectamente con el código programado, utilizando las señales de Qt y el mecanismo de las ranuras, de modo que usted puede asignar fácilmente el comportamiento a los elementos gráficos. Todas las propiedades establecidas en Qt Designer se pueden cambiar dinámicamente dentro del código como se aprecia en la figura 17-1. Además, las características como la promoción de widgets y los complementos personalizados le permiten usar sus propios componentes con Qt Designer.

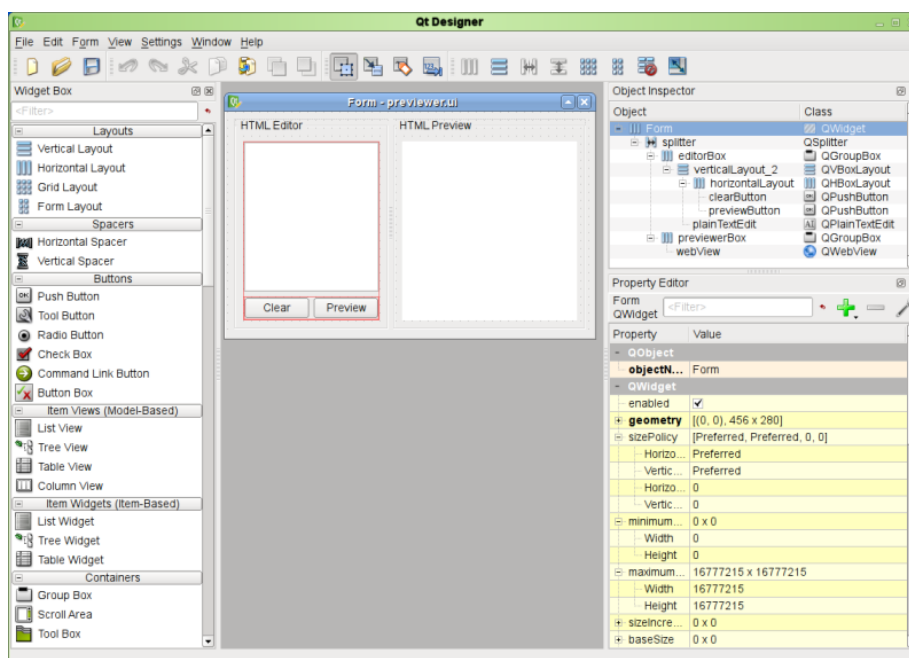


Figura 17-1 Entorno de trabajo de Qt designer

Fuente: <http://doc.qt.io/archives/qt-4.8/designer-to-know.html>

2. MARCO METODOLÓGICO

En el presente capítulo se procede a la descripción detallada del diseño del sistema de visión.

Etapas del diseño del sistema de visión seguimiento de pupilas

- Análisis de la capacidad gestual.
- Selección del hardware y software más apropiado.
- Procesamiento digital de imagen.
 - Adquisición de imagen
 - Preprocesamiento.
 - Segmentación.
 - Descripción.
 - Reconocimiento o interpretación.
- Diseño de la Interfaz gráfica de usuario.
- Integración de la Interfaz con el programa de visión artificial.

2.1 Análisis de la capacidad gestual

En base a la investigación teórica se concluye que existen 3 tipos de movimientos oculares: sacádico, seguimiento y fijación, éste último, en el que los ojos permanecen casi estáticos durante 250 milisegundos, teniendo lugar la extracción de información en esta fase.

El grupo de personas a quien se dirige el sistema generalmente presenta dificultad para realizar movimientos oculares controlados como:

- Esquina superior derecha
- Esquina superior izquierda
- Esquina inferior derecha
- Esquina inferior izquierda

Los movimientos mencionados se pueden observar de mejor manera en la Figura 1-2.

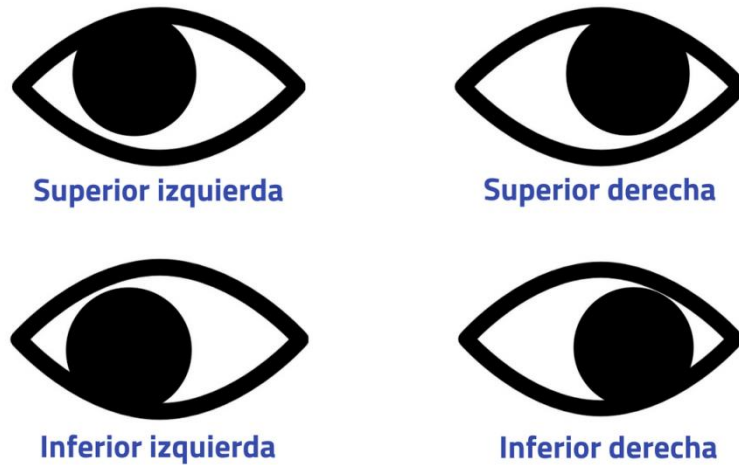


Figura 1-2 Movimientos oculares con los que se presenta dificultades

Fuente: Pillajo, M. Navarrete, D. 2018

Debido a la dificultad que el grupo de personas presenta para realizar los movimientos presentados en la figura 1-2, se decide usar movimientos de fácil ejecución y fijación, como los presentados en la figura 2-2:

- Central
- Central derecha
- Central izquierda

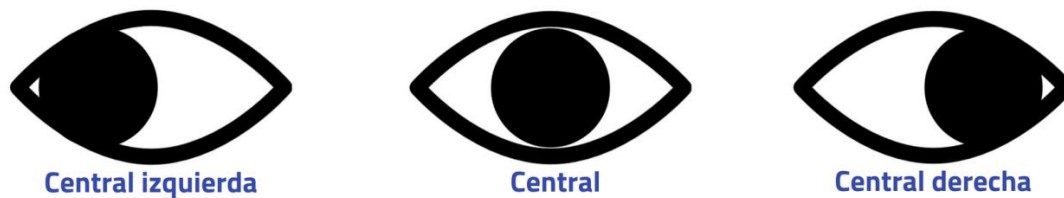


Figura 2-2 Movimientos con los que se procede al desarrollo del sistema

Fuente: Pillajo, M. Navarrete, D. 2018

2.2 Selección de software y hardware

La selección del software y hardware se realiza en función a las etapas del sistema como se observa en la figura 3-2 y a los requerimientos presentes en cada una de ellas.

Esquema básico de un sistema de procesamiento visual

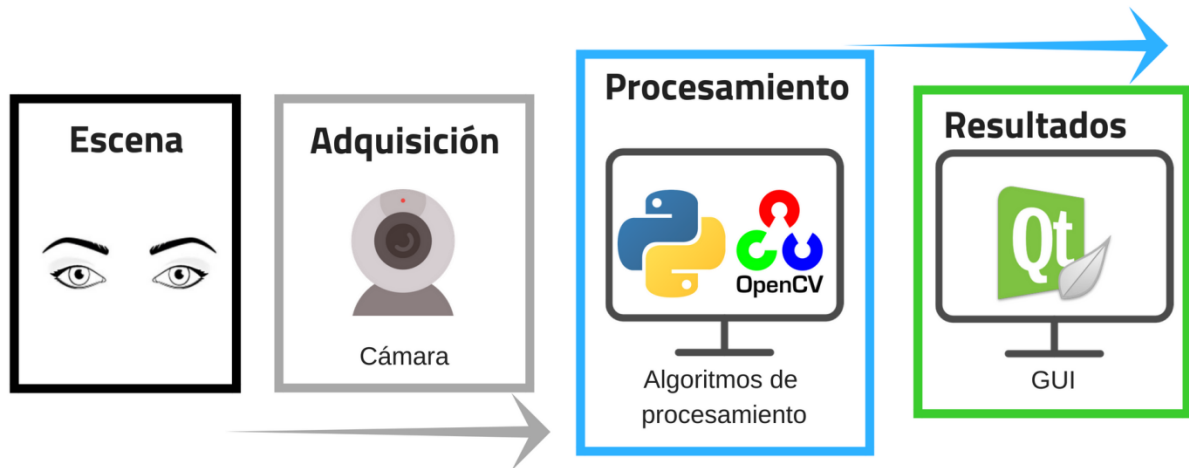


Figura 3-2 Esquema básico de un sistema de procesamiento visual

Fuente: Pillajo, M. Navarrete, D. 2018

2.2.1 *Etapa de adquisición*

2.2.1.1 *Cámara*

Es necesario para la realización del trabajo un ordenador que tenga características estándar del procesador y la cámara del mismo en el caso de una laptop o una cámara web de gama media que oscila por unos 15 a 20 dólares.

2.2.2 *Etapa de procesamiento*

2.2.2.1 *Python*

Python tiene una comunidad considerable que opta de desarrollar sus proyectos en esta plataforma y comparte la información de cómo lo realizaron, problemas que tuvieron el transcurso del proyecto y la solución a los mismos mediante foros de la misma plataforma de programación o ajena a ella.

2.2.2.2 *OpenCV*

OpenCV es una librería multiplataforma que tiene soporte para distintos tipos de sistemas operativos, dedicada al procesamiento digital de imágenes que de igual forma de Python tiene una gran comunidad que aporta con información sin fines de lucro.

2.2.3 *Etapa de interpretación de resultados*

2.2.3.1 *QT*

QT es un software de diseño y programación de GUI, que tiene su versión gratuita y su versión Premium a la que se accede mediante pago, pero su versión gratuita provee de herramientas suficientes para el desarrollo de proyecto.

2.3 **Procesamiento digital de imagen**

A continuación se presenta un esquema del proceso que se realiza para el procesamiento de imagen desde su adquisición hasta su interpretación.



Figura 4-2 Procedimiento para el procesamiento digital de imagen

Fuente: Pillajo, M. Navarrete, D. 2018

2.3.1 *Adquisición de la imagen*

Para poder obtener información del entorno donde se va a desarrollar el proyecto es necesario una cámara de video compatible con el ordenador o la Webcam incluida de las laptops.

Mediante estas se obtiene una imagen de la persona y del ambiente en el que se encuentra entonces, gracias a Python y OpenCV, mediante código se obtiene una imagen de forma digital y transformada en una matriz de información numérica, como se observa en la figura 5-2.

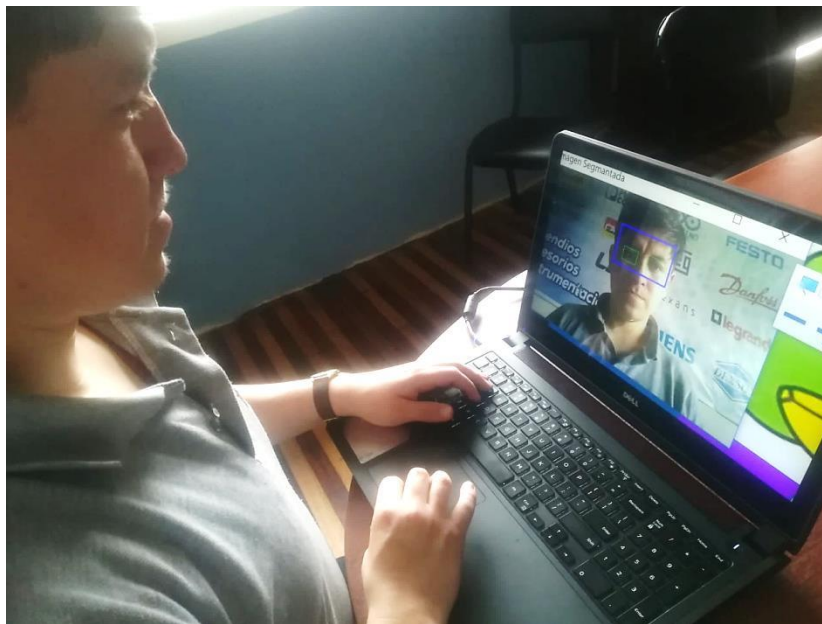


Figura 5-2 Adquisición de la imagen mediante la cámara

Fuente: Pillajo, M. Navarrete, D. 2018

2.3.2 *Preprocesamiento de imagen*

La imagen debe ser transformada en escala de grises para poder extraer la información de mejor manera, debido a la facilidad de distinguir solo en distintas tonalidades de blanco y negro,

$$Y = R*0.3+G*0.59+B*0.11$$

Aplicando la anterior fórmula se puede transformar a cada pixel de la imagen en una matriz de un byte que daría la información de luminancia.

Se puede tomar como referencia la siguiente paleta de colores para saber cómo se representaría un color en escala de grises representada a continuación en la Figura 6-2.

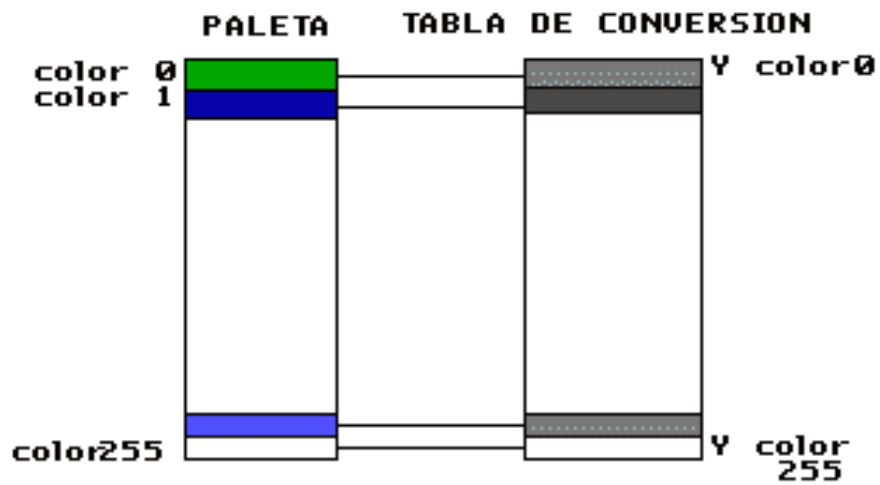


Figura 6-2 Paletas de colores y tabla de conversi3n a escala de grises.

Fuente: Pillajo, M. Navarrete, D. 2018

Mediante el uso de las herramientas de OpenCV junto con el c3digo en Python se obtiene la imagen en escala de grises como se aprecia en la figura 7-2.



Figura 7-2 Imagen Original transformada a escala de grises.

Fuente: Pillajo, M. Navarrete, D. 2018

Para mejorar la calidad de la imagen se requiere de la ecualizaci3n del histograma obtenido y los resultados se ven reflejados en la figura 8-2.

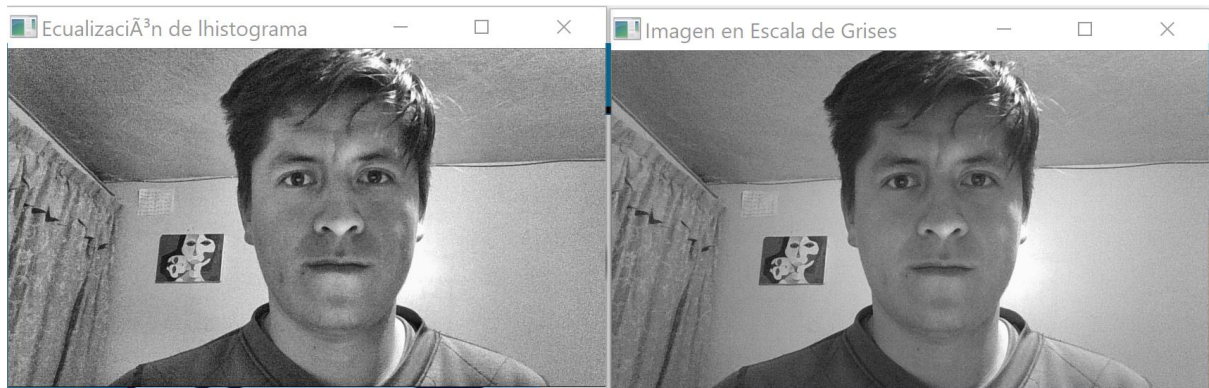


Figura 8-2 Contraste imagen ecualizada vs escala de grises.

Fuente: Pillajo, M. Navarrete, D. 2018

A partir de este punto comienza el reconocimiento y segmentaci3n del 3rea de inter3s empezando desde el rostro luego con los ojos y finalmente con las pupilas.

Para el reconocimiento del rostro se parte de librerías de OpenCV llamados haarcascades que se basa en el algoritmo de Haar.

En la figura 9-2 se observa como trabaja los algoritmos dentro de OpenCV aplicada a la imagen obtenida segmentando los objetos de inter3s.

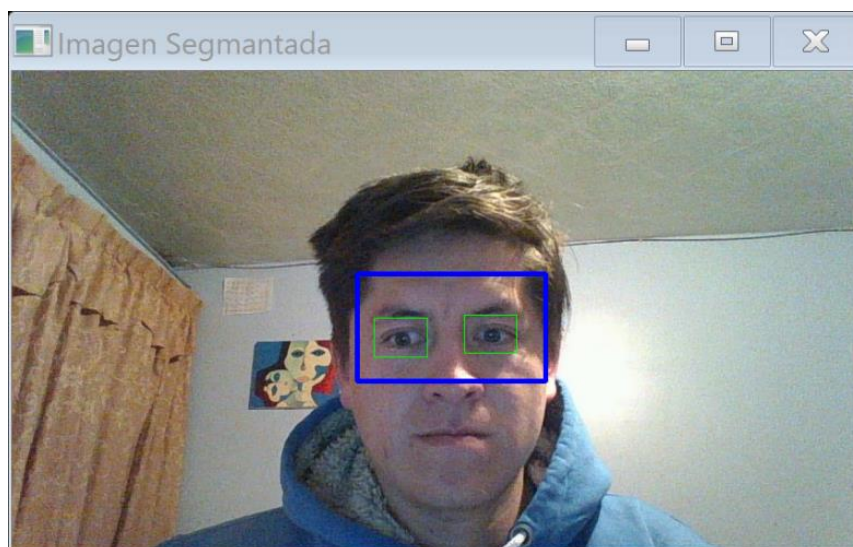


Figura 9-2 Segmentaci3n del rostro y los ojos.

Fuente: Pillajo, M. Navarrete, D. 2018

Para seguir con el procesamiento de imagen, se aísla el 3rea de trabajo para encontrar el objeto de inter3s en este caso el rostro como se aprecia en la figura 10-2.



Figura 10-2 Sección del rostro recortado para extraer la información.

Fuente: Pillajo, M. Navarrete, D. 2018

Seguidamente se procede a realizar lo mismo que con el rostro pero ahora a los ojos para procesar la información que se requiere como se observa en la figura 11-2.

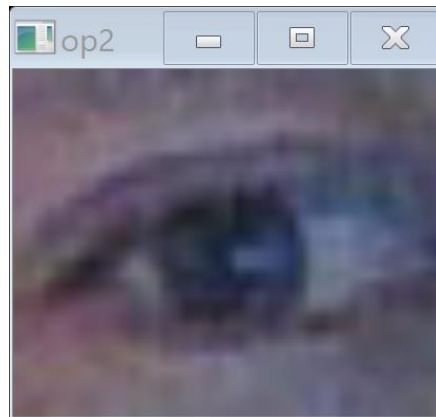


Figura 11-2 Imagen recortada y amplificada el ojo para su análisis.

Fuente: Pillajo, M. Navarrete, D. 2018

2.3.3 Descripción

Una vez obtenida la sección de trabajo se debe segmentar las pupilas para esto se aplicó una máscara para la obtención de bordes de la imagen.

Utilizando el algoritmo de Hugh para el reconocimiento de círculos y mediante el área y la posición se determina el iris y la pupila.

```
e_gauss_rsz = cv2.resize(e_gauss, (0,0), fx = 10, fy = 10)
ret, e_gauss_bw = cv2.threshold(e_gauss_rsz,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
circles1 = cv2.HoughCircles(c_small, cv2.HOUGH_GRADIENT, thrs1+(thrs2/100000000), 150, 500, 200, min_d/2, max_d/2)
circles1 = np.matrix(circles1)
```

Figura 12-2 Código para la determinación de círculos.

Fuente: Pillajo, M. Navarrete, D. 2018

En la figura 12-2 se observa el código con el cual se puede obtener los círculos configurando los parámetros que, de acuerdo al comando, el cual está compuesto de los siguiente:

```
circles = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, dp=1, minDist=20, param1=50, param2=30, minRadius=0, maxRadius=0)
```

Parámetros

- **Circles:** Vector de salida de círculos encontrados. Cada vector está codificado como un vector de coma flotante de 3 elementos (x, y, radio).
- **img:** Imagen de entrada en escala de grises de un solo canal de 8 bits.
- Método de detección. Actualmente, el único método implementado es HOUGH_GRADIENT.
- **dp:** 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 $dp = 2$, el acumulador tiene la mitad del ancho y la altura.
- **minDist:** Distancia mínima entre los centros de los círculos detectados. Si el parámetro es demasiado pequeño, se pueden detectar varios círculos vecinos además de uno verdadero. Si es demasiado grande, algunos círculos se pueden perder.
- **param1:** Primer parámetro específico del método. En el caso de CV_HOUGH_GRADIENT, es el umbral más alto de los dos pasados al detector Canny Edge (el inferior es dos veces más pequeño).
- **param2:** Segundo parámetro específico del método. En el caso de CV_HOUGH_GRADIENT, es el umbral del acumulador para los centros del círculo en la etapa de detección. Cuanto más pequeño es, más círculos falsos pueden detectarse. Los círculos, que corresponden a los valores de acumulador más grandes, se devolverán primero.
- **minRadius:** Radio mínimo del círculo.
- **maxRadius:** Radio máximo del círculo.

Configurando los parámetros anteriores se puede obtener un resultado satisfactorio como se muestra en la figura 13-2.

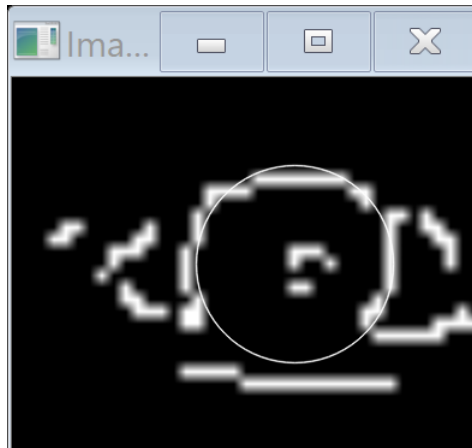


Figura 13-2 Imagen con bordes y reconocimiento de la pupila.

Fuente: Pillajo, M. Navarrete, D. 2018

2.3.4 Reconocimiento e interpretación

2.3.4.1 Reconocimiento

Una vez alcanzado el objetivo se debe eliminar falsos positivos, es decir circunferencias que no forman parte del objeto de estudio con el código mostrado a continuación en la figura 14-2.

```

if np.size(circles1) > 1:
    while i < circles1.shape[0]:
        if (circles1[i,1] > c_small.shape[0]*0.65) or (circles1[i,1] < c_small.shape[0]*0.35):
            circles1 = np.delete(circles1, i, 0)
            i = i - 1
            i = i + 1

##### Eliminar círculos que excedan los límites horizontales
i = 0
while i < circles1.shape[0]:
    if (circles1[i, 0] + circles1[i, 2] >= c_small.shape[1]) or (circles1[i, 0] - circles1[i, 2] <= 0):
        circles1 = np.delete(circles1, i, 0)
        i = i - 1
        i = i + 1

##### Eliminar círculos excedentes
if circles1.shape[0] > 1:
    for i in range(circles1.shape[0]):
        e_gauss_bw_cir = e_gauss_bw[int(circles1[i,1] - circles1[i, 2]): \
                                   int(circles1[i,1] + circles1[i, 2]), \
                                   int(circles1[i,0] - circles1[i, 2]): \
                                   int(circles1[i,0] + circles1[i, 2])]
        pixeles_b.append(cont_pixeles_negros(e_gauss_bw_cir))
    circles1 = circles1[np.argmax(pixeles_b),:]

```

Figura 14-2 Código de supresión de falsos positivos.

Fuente: Pillajo, M. Navarrete, D. 2018

En la detección de circunferencias se encontraron objetos que no pertenecían al objeto de estudio por lo que se opta por eliminarlos para que no afecten con el resultado final.

El análisis comienza preguntando si existen circunferencias que detectar, si existen se realizó una relación de proporción para determinar el tamaño que debería tener la pupila dependiendo de la distancia del rostro hacia la cámara discriminando al resto.

Luego se eliminó circunferencias que se encontraban muy alejadas de donde debe estar pupila, de forma horizontal como vertical, así se asegura en lo posible un mejor funcionamiento del sistema.

2.3.4.2 Interpretación

Finalmente, la dirección de la mirada en el sistema de visión es interpretado como un número del 0 al 2 el cual servirá para el enlace con la Interfaz gráfica.

- El movimiento hacia la derecha es interpretado con un 0 como se ve en la figura 15-2.

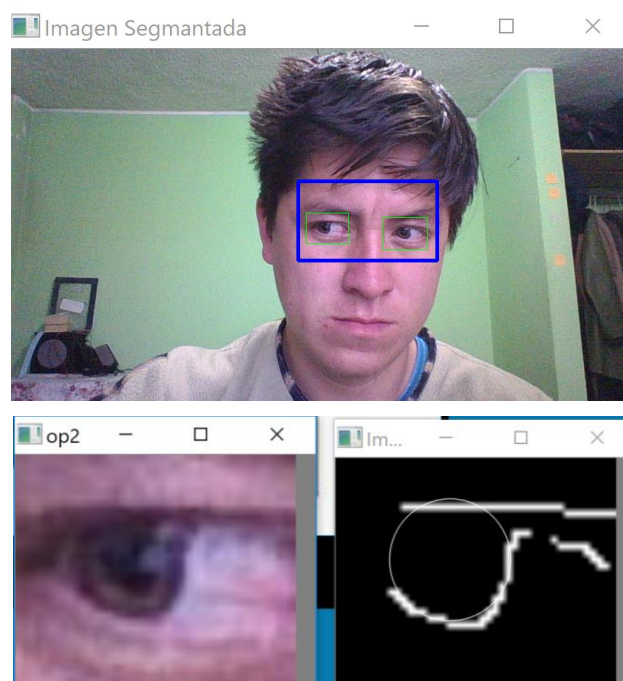


Figura 15-2 Seguimiento de pupila hacia la derecha.

Fuente: Pillajo, M. Navarrete, D. 2018

- El movimiento hacia la izquierda es interpretado con un 2 visto en la figura 16-2.

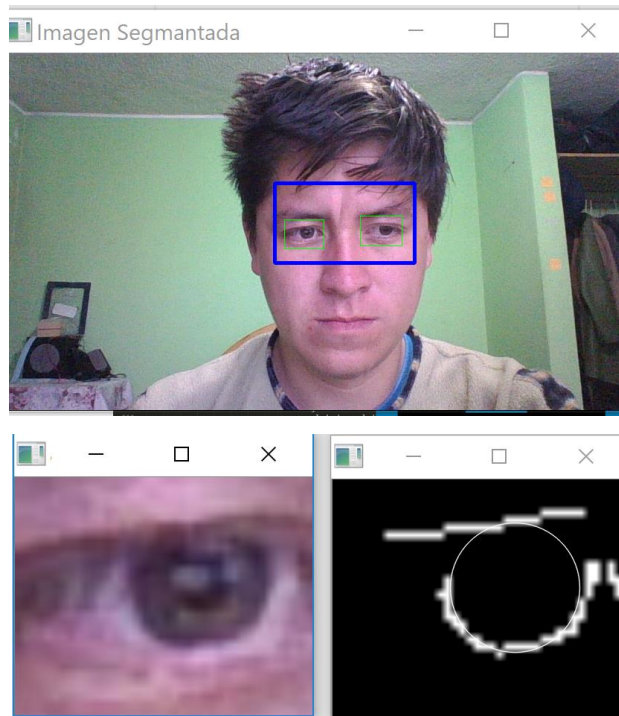


Figura 16-2 Seguimiento de la pupila hacia la izquierda.

Fuente: Pillajo, M. Navarrete, D. 2018

- La mirada al centro es interpretado como un 1 como se aprecia en la figura 17-2.

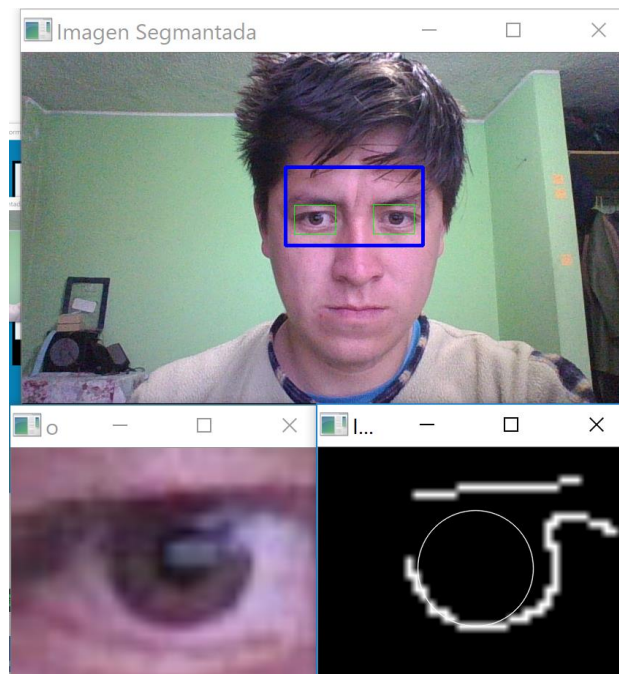


Figura 17-2 Seguimiento de la pupila hacia el centro.

Fuente: Pillajo, M. Navarrete, D. 2018

Debido al error presente en la discriminación de posición de la pupila, producido por los movimientos involuntarios del ojo resulta complicado trabajar con la variable de posición en estado puro, por lo que es necesario aplicar el calculo de un estadístico para el análisis de éstos datos.

Se procede a formar pequeños grupos de datos y obtener el valor más frecuente o repetitivo (moda estadística) para así lograr una mejor selección de la posición elegida.

2.4 Diseño de la Interfaz gráfica de usuario.

Partiendo del análisis gestual se diseñó la GUI mediante QT, debido a su forma interactiva de búsqueda, selección y aplicación de objetos.

El diseño de la GUI como se muestra en la figura 18-2 se empezó por definir los botones que van a ser los que mediante clicks o a través del programa de visión artificial efectue la acción designada al mismo.

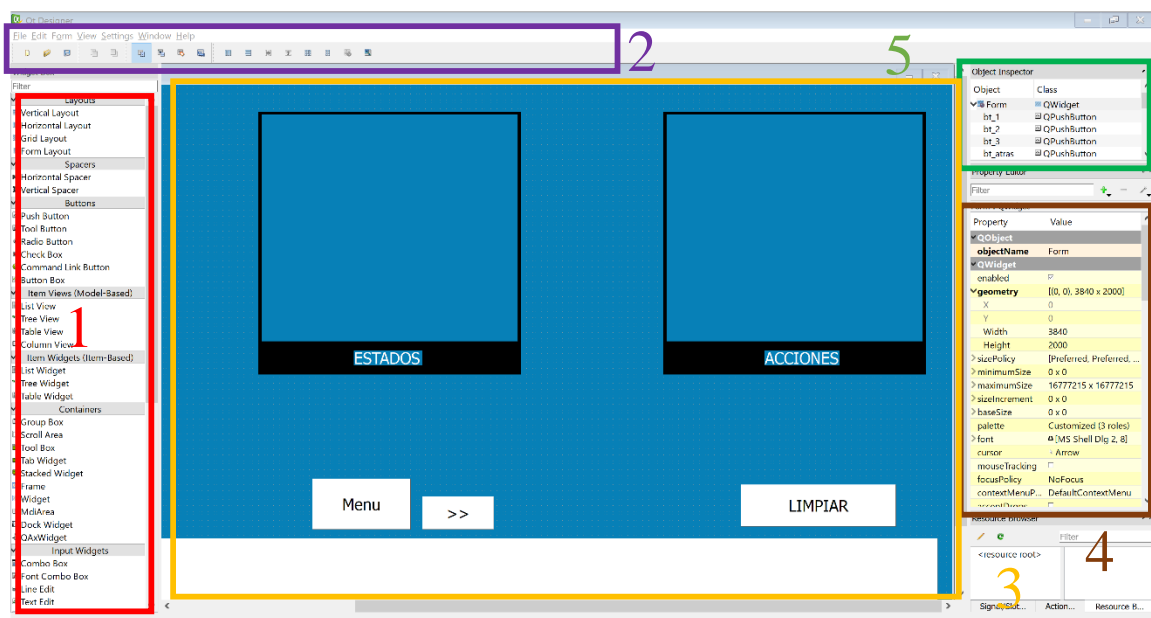


Figura 18-2 Pantalla principal de QT.

Fuente: Pillajo, M. Navarrete, D. 2018

1. Objetos que se utilizan en el diseño de la GUI como botones, labels, textbox, etc.
2. Barra de herramientas para crear una nueva ventana guardar agregar o retirar opciones de diseño.
3. Area de trabajo de QT, donde se coloca los objetos que se desee , vaya en la GUI.
4. Barra de parámetros del objeto, en el cual se pude modificar sus características como el color, tamaño, tipo de fuente.
5. Tabla de objetos es donde se encuentran todos los elementos que se ha utilizado para diseñar la interfza gráfica de usuario.

Primero se ubica los botones de principales que se va a seleccionar y navegar a través de los menús y los pictogramas pre seleccionados que se observa en la figura 19-2.



Figura 19-2 Botones principales.

Fuente: Pillajo, M. Navarrete, D. 2018

Los siguientes botones que muestra en la siguiente figura 20-2 ayudan como un control manual para el sistema en vez del sistema de visión.



Figura 20-2 Botones de desplazamiento de imágenes.

Fuente: Pillajo, M. Navarrete, D. 2018

- **Menu.-** Configura los botones de selección al menú principal o a las imágenes que aparecen por defecto al inicializar el programa.
- **>>.-** Este botón cambia las imágenes dependiendo del menú o submenú en el que se encuentre.
- **LIMPIAR.-** La función de este es borrar las palabras que se encuentran en el cuadro de texto que se encuentra en la parte inferior de la ventana principal del sistema.

La ventana principal también está compuesta por elementos de señalización los cuales ayudan a saber cuando detecta el rostro y los ojos, pero también cuando se da un parpadeo, finalmente el cuadro de texto donde se escribe el significado del pictograma seleccionado.

La figura 21-2 muestra como está constituida la pantalla de señalización.

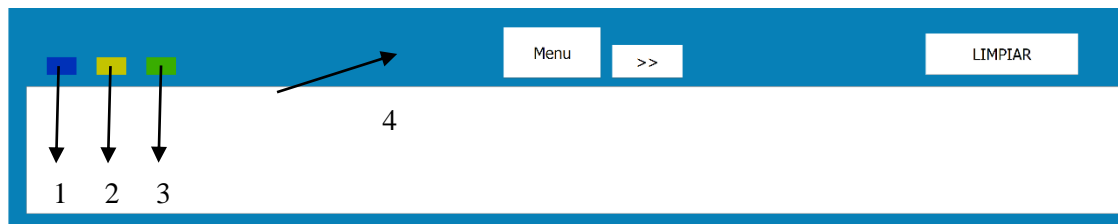


Figura 21-2 Elementos de señalización del sistema.

Fuente: Pillajo, M. Navarrete, D. 2018

1. **Detección de rostro.-** Cambia de color a un azul más clasto cuando detecta el rostro.
2. **Detección de ojos.-** Modifica su color a un amarillo intenso al momento de detectar los ojos.
3. **Detección de parpadeo.-** En el instante del parpadeo altera su color a un verde más claro.
4. **Cuadro de texto.-** Es el sitio donde expresa mediante palabras el pictograma seleccionado.

Finalmente se obtiene el resultado final, la interfaz gráfica de usuario que se observa en la figura 22-2.

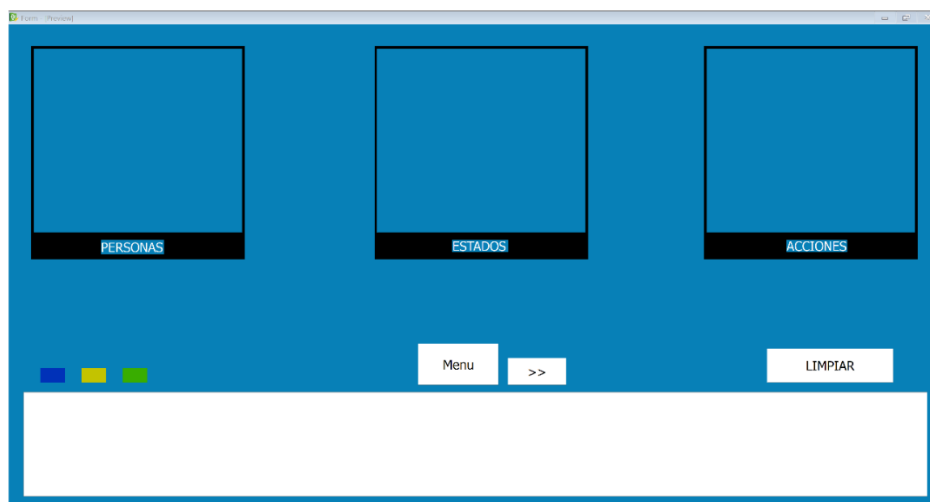








Figura 22-2 Interfaz gráfica de usuario finalizada.









Fuente: Pillajo, M. Navarrete, D. 2018

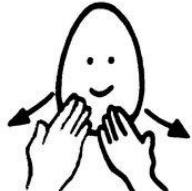



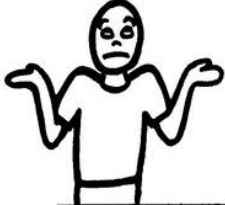

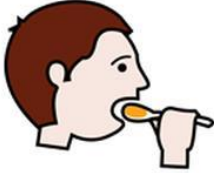
2.4.1 *Determinación de los elementos de la base de datos del sistema*




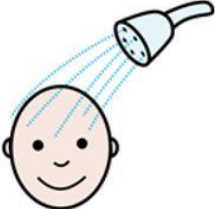
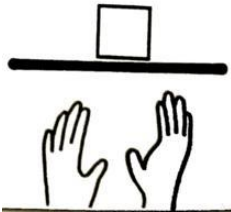

En la tabla 1-1 se muestra el conjunto de datos que usará el sistema, cada registros comprende los siguientes campos: el nombre del archivo con el cual el programa lo identifica, su descripción y la imagen que representa al pictograma.

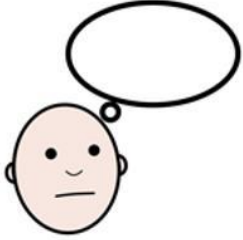
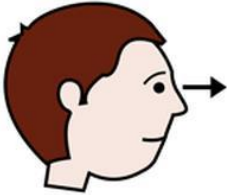




Tabla 1-1 Base de datos del sistema






IDENTIFICADOR	DESCRIPCIÓN	PICTOGRAMA
10	PERSONAS	
11	YO	
12	TU	
13	MAMÁ	
14	PAPÁ	
15	HERMANA	






16	HERMANO	
17	PROFESOR	
18	AMIGO	
19	DOCTOR	
20	ESTADOS	
21	ESTOY BIEN	
22	ESTOY MAL	
23	AYUDA	




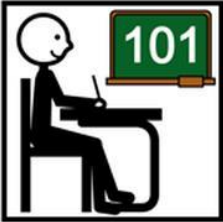
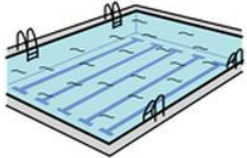

24	GRACIAS	
25	CHAO	
26	HOLA	
27	LO SIENTO	
28	NO SÉ	
29	POR FAVOR	
30	ACCIONES	







31	COMER	
32	DORMIR	
33	JUGAR	
34	BAÑAR	
35	QUERER	
36	ABRAZAR	



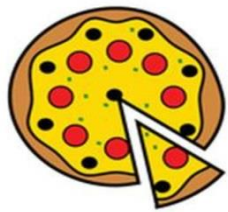

37	PENSAR	
38	MIRAR	
39	ESCUCHAR	
40	SENTIMIENTOS	
41	FRIO	
42	CALIENTE	

43	ENFERMO	
44	FELIZ	
45	TRISTE	
46	ENOJADO	
47	BUENO	

48	MALO	
50	LUGARES	
51	BAÑO	
52	CASA	
53	PARQUE	

54	ESCUELA	
55	CALLE	
56	HOSPITAL	
57	AULA	
58	PISCINA	
59	IGLESIA	

60	COMIDA	
61	AGUA	
62	FRUTA	
63	PAN	
64	ARROZ	
65	SOPA	

66	CARNE	
67	POLLO	
68	PIZZA	
69	CARAMELO	

Realizado por: Pillajo, M. Navarrete, D. 2018

2.5 Integración de la Interfaz con el programa de visión artificial.

El último paso para poder utilizar el sistema es enlazar la Interfaz gráfica de usuario con el programa de visión artificial para lo cual se utiliza lenguaje Python mediante la librería que se muestra en la figura 23-2:

```
3 from PyQt4.QtGui import QApplication, QMainWindow
4 from PyQt4 import QtCore, QtGui, uic
```

Figura 23-2 Librerías necesarias para la comunicación Python con QT.

Fuente: Pillajo, M. Navarrete, D. 2018

Mediante estas librerías se puede simplemente llamar a la interfaz diseñada previamente y ponerla en condiciones iniciales y definiendo que es lo que hará cada botón cuando se ejecute un click.

Cuando este definida las funciones de cada botón resta el enlace mediante código. Debido a que el sistema de visión necesita ejecutarse permanentemente dentro de un ciclo repetitivo, es importante utilizar el comando `self.threadclass` el cual se ejecuta paralelamente con la GUI debido que al usar la librería `thread` la congela inhabilitándola por completo y no permite utilizarla.

En la figura 24-2 muestra parte del código necesario para la ejecución de procesos paralelos con la GUI.

```
self.threadclass.start()
self.connect(self.threadclass,QtCore.SIGNAL('direccion'), self.Update_Signals)
self.connect(self.threadclass,QtCore.SIGNAL('blink_1'), self.Parpadeo)
ejec_cam = True
self.show()
```

Figura 24-2 Comando para acceder a las funciones que contienen un ciclo while.

Fuente: Pillajo, M. Navarrete, D. 2018

Para entender de mejor manera el proceso que realiza el sistema de comunicación mediante visión artificial, se presenta a manera de diagrama de flujo en la gráfica 1-2 donde se ve representado la función que cumple la interfaz gráfica de usuario y la gráfica 2-2 que muestra como se realiza el procesamiento digital de la imagen hasta su interpretación del movimiento del iris y pupila.

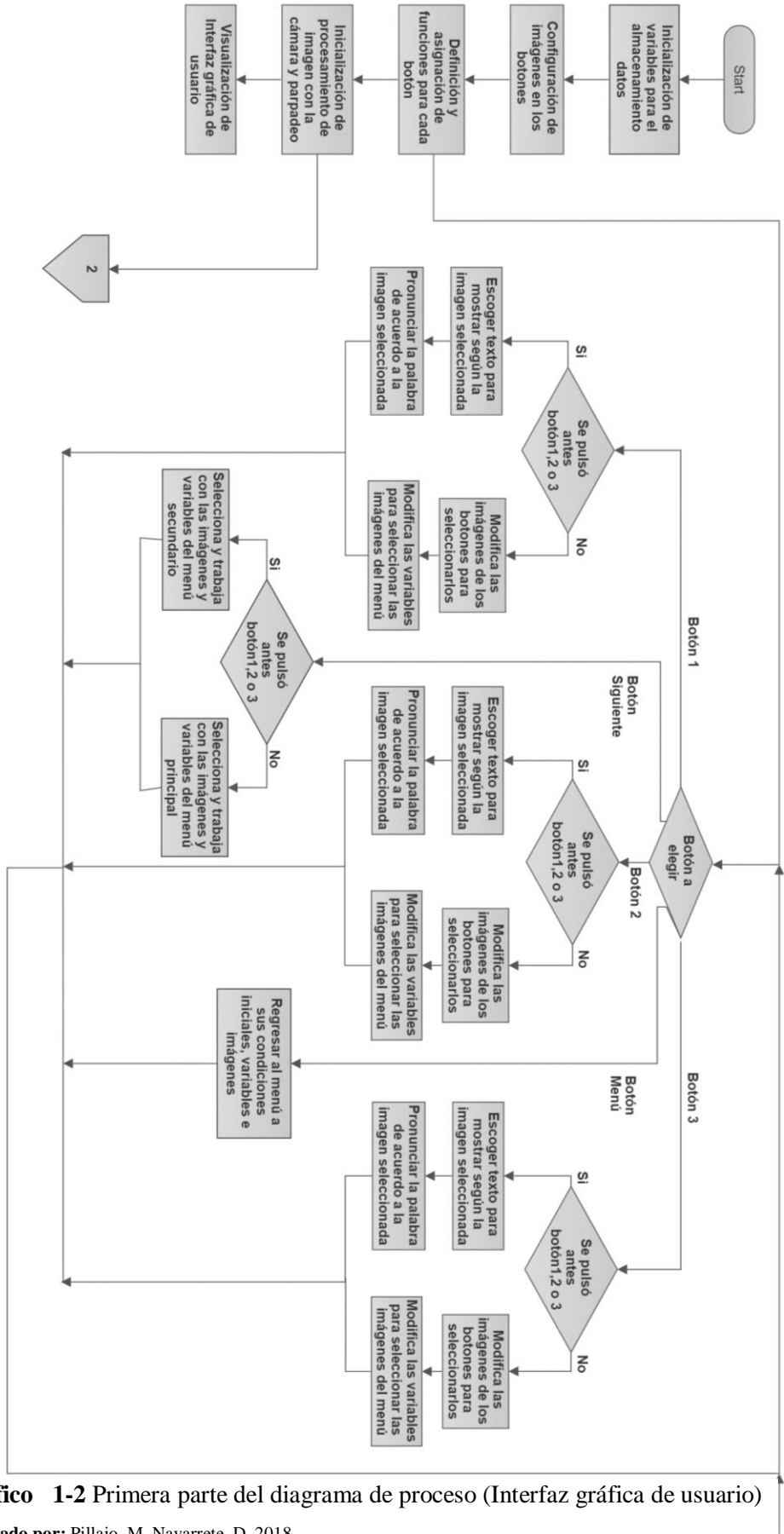


Gráfico 1-2 Primera parte del diagrama de proceso (Interfaz gráfica de usuario)

Realizado por: Pillajo, M. Navarrete, D. 2018

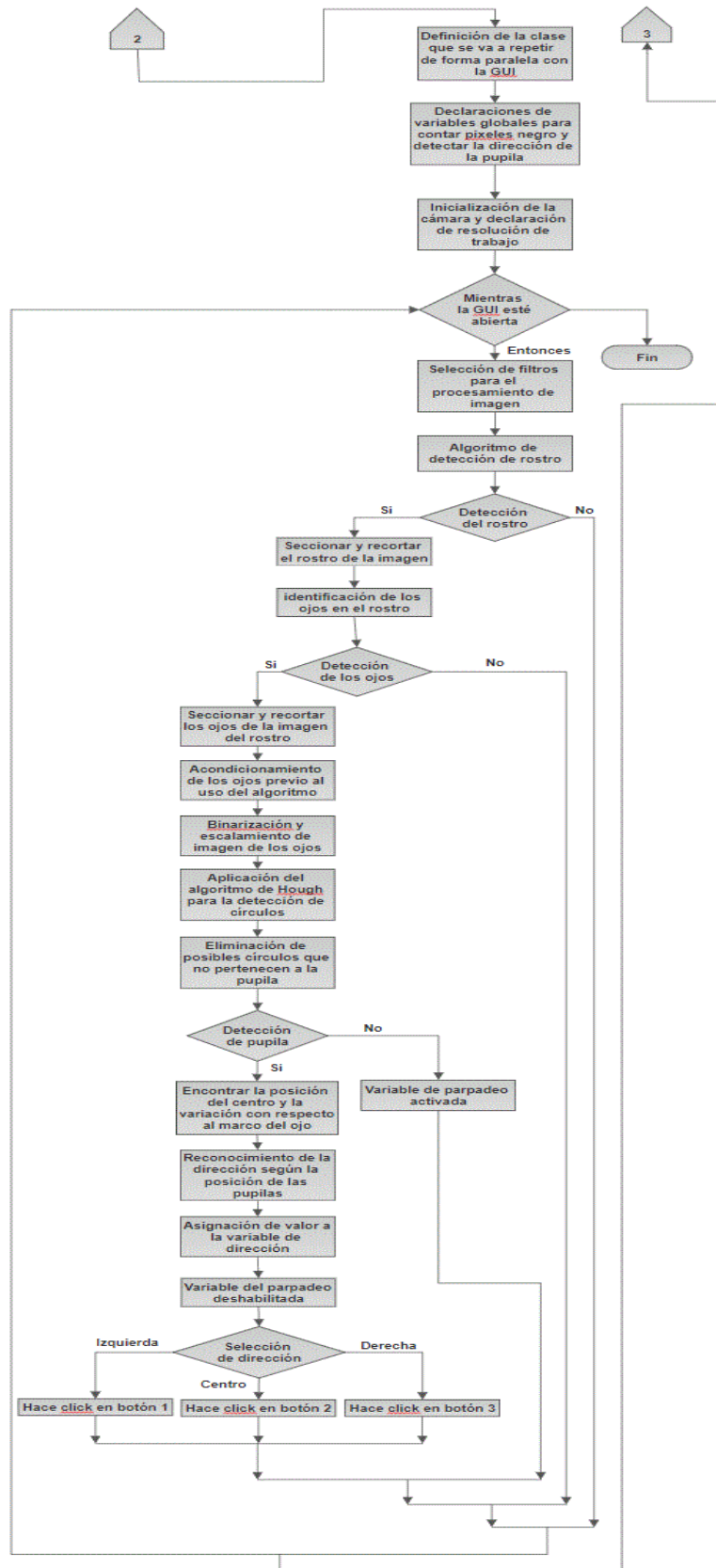


Gráfico 2-2 Segunda parte del diagrama de proceso (Interfaz gráfica de usuario)

Realizado por: Pillajo, M. Navarrete, D. 2018

CAPÍTULO III

3. MARCO DE RESULTADOS

3.1 Análisis detección movimiento de pupila

Debido a la dificultad de obtención de permisos en instituciones públicas para trabajar con las personas con este tipo de discapacidad con el consentimiento de sus representantes se optó por realizar las pruebas del sistema evaluando su desempeño con parámetros de precisión.

Una vez que se obtiene la imagen segmentada del ojo se procede a definir los rangos para discriminar la posición del iris y pupila con respecto al rectángulo que enmarca el ojo, los cuales están definidos por un valor proporcional del total de píxeles en el eje x, donde el límite de la derecha es el valor de los píxeles multiplicado por 0.65 y en cambio el de la izquierda está multiplicado por 0.35, estos valores permiten una mejor discriminación de la posición del iris y la pupila cuando se fija la mirada a la derecha, centro o izquierda como se puede apreciar en la figura 1-3.

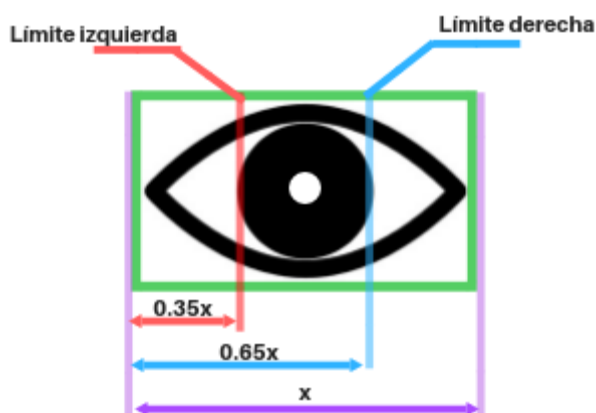


Figura 1-3 Ilustración de los límites para la fijación de la mirada

Fuente: Pillajo, M. Navarrete, D. 2018

Se eligen 72 datos debido a que es el número de muestras necesarias para que el procesamiento arroje un resultado fiable dentro del sistema, si se opta por tomar menos muestras, el sistema gana rapidez pero pierde precisión y confiabilidad en la interpretación de datos, y si se opta por tomar más datos, el sistema es más confiable pero pierde rapidez obligando a que la persona que utiliza el sistema permanezca con su mirada fija por un tiempo excesivo provocando un agotamiento visual.

3.1.1 *Detección figura derecha*

3.1.1.1 *Análisis de gráfica*

Se copia el vector resultante de la posición de la pupila, cuando la visión está dirigida hacia la derecha, en una columna de Excel y se puede observar su curva en relación a los límites superior e inferior del rango establecido para la detección como se observa en la gráfica 1-3.

La gráfica muestra tres líneas de tendencia, una línea gris que representa el límite superior o izquierdo, una línea azul que representa el límite inferior o derecho y una gráfica naranja que presenta una acumulación de valores bajo el límite inferior, esto se debe a que al dirigir la mirada hacia la derecha la pupila presenta estas posiciones.

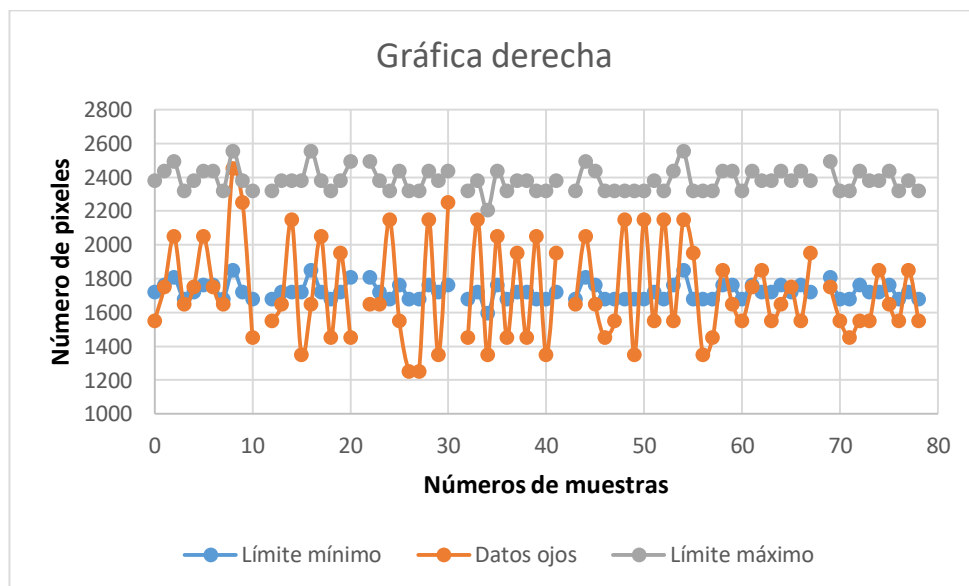


Gráfico 1-3 Gráfica de conjunto de datos de las posiciones captadas de la pupila en movimiento derecha

Realizado por: Pillajo, M. Navarrete, D. 2018

3.1.1.2 *Análisis de dispersión de datos*

Se copia los datos obtenidos en un script del software R como se aprecia en la figura 2-3 y se obtiene los siguientes resultados :

```

R Console
> A<-read.table("D:/TESIS/Estadistica/Pruebas/Derecha.txt",header=T)
> x=as.numeric(unlist(A))
> boxplot(x)
> summary(x)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1250  1550   1650   1727   1950   2450

```

Figura 2-3 Captura ingreso de conjunto de datos de la pupila en movimiento derecha

Fuente: Pillajo, M. Navarrete, D. 2018

La tabla 1-3 muestra la dispersión de los datos y en donde se encuentran más datos acumulados, los cuales fueron extraídos del diagrama de caja y bigotes representada en la gráfica 2-3. La parte superior de la caja es mayor que la inferior; ello quiere decir que los valores comprendidos entre el 75% y el 50% del conjunto de datos (1950 y 1650) están más dispersos que entre el 25% y el 50% (1550 y 1650). El bigote inferior o cuartil 1 es más corto que el superior; por ello el 25% de los datos de menor valor están más concentrados que el 25% de los de mayor valor. El rango intercuartílico = $Q3 - Q1 = 400$; es decir, el 50% de los datos presentan una variación de un valor de 400. La mediana que nos presenta el gráfico es 1650 es decir que la mayoría de los datos tienden a éste valor, con la ayuda de éste dato se puede discriminar un movimiento de la pupila hacia la derecha, comparando su posición con respecto a éste valor.

Tabla 1-3 Resumen valores estadísticos de la pupila en movimiento derecha

Estadístico	Valor
Valor mínimo	1250
Primer cuartil (Q1)	1550
Mediana	1650
Media	1727
Tercer cuartil (Q3)	1950
Valor máximo	2450

Realizado por: Pillajo, M. Navarrete, D. 2018

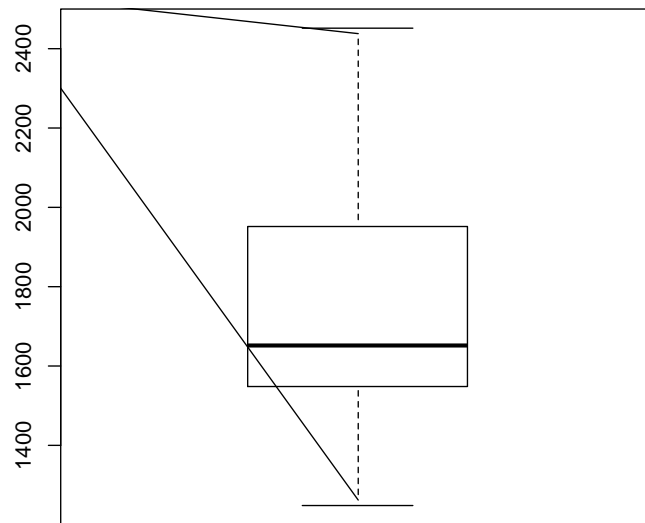


Gráfico 2-3 Diagrama de caja y bigotes de datos de la pupila en movimiento derecha

Realizado por: Pillajo, M. Navarrete, D. 2018

3.1.2 *Detección figura centro*

3.1.2.1 *Análisis de gráfica*

Se copia el vector resultante de la posición de la pupila, cuando la visión está dirigida hacia el centro, en una columna de Excel y se observa su curva en relación a los límites superior e inferior del rango establecido para la detección como muestra la gráfica 3-3.

La gráfica muestra tres líneas de tendencia, una línea gris que representa el límite superior o izquierdo, una línea azul que representa el límite inferior o derecho y una gráfica naranja que presenta una acumulación de valores comprendidos entre el límite superior y el límite inferior, esto se debe a que al dirigir la mirada hacia el centro la pupila presenta estas posiciones.

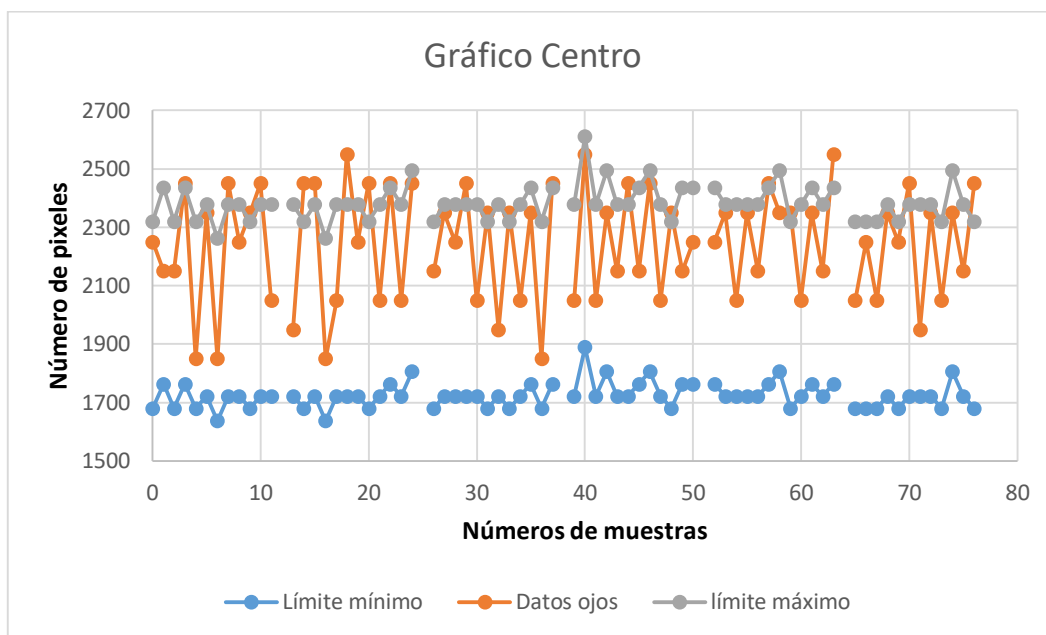


Gráfico 3-3 Gráfica de conjunto de datos de la pupila fijada en el centro

Realizado por: Pillajo, M. Navarrete, D. 2018

3.1.2.2 *Análisis de dispersión de datos*

Se copia los datos obtenidos en un script del software R y se obtiene los siguientes resultados mostrado en la figura 3-3:

```

R Console
> B<-read.table("D:/TESIS/Estadistica/Prueba1/Centro.txt",header=T)
> y=as.numeric(unlist(B))
> boxplot(y)
> summary(y)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1850   2050   2250   2240   2400   2550
> |

```

Figura 3-3 Captura ingreso de conjunto de datos de la pupila fijada en el centro

Fuente: Pillajo, M. Navarrete, D. 2018

Captura ingreso de datos de las posiciones captadas del iris y la pupila manteniendo la mirada al centro, los cuales se ven recopilados en una tabla 2-3, con los cuales es posible construir el diagrama de caja y bigotes de la grafica 4-3.

La parte superior de la caja es menor que la inferior; ello quiere decir que los valores comprendidos entre el 75% y el 50% del conjunto de datos (2400 y 2250) están menos dispersos

que entre el 25% y el 50% (2050 y 2250). El bigote superior o cuartil 3 es más corto que el inferior; por ello el 25% de los datos de mayor valor están más concentrados que el 25% de los de menor valor. El rango intercuartílico = $Q3 - Q1 = 350$; es decir, el 50% de los datos presentan una variación de un valor de 350. La mediana que nos presenta el gráfico es 2250 es decir que la mayoría de los datos tienden a éste valor, con la ayuda de éste dato se puede discriminar un movimiento de la pupila hacia el centro, comparando su posición con respecto a éste valor.

Tabla 2-3 Resumen valores estadísticos de la pupila fijada en el centro

Estadístico	Valor
Valor mínimo	1850
Primer cuartil (Q1)	2050
Mediana	2250
Media	2240
Tercer cuartil (Q3)	2400
Valor máximo	2550

Realizado por: Pillajo, M. Navarrete, D. 2018

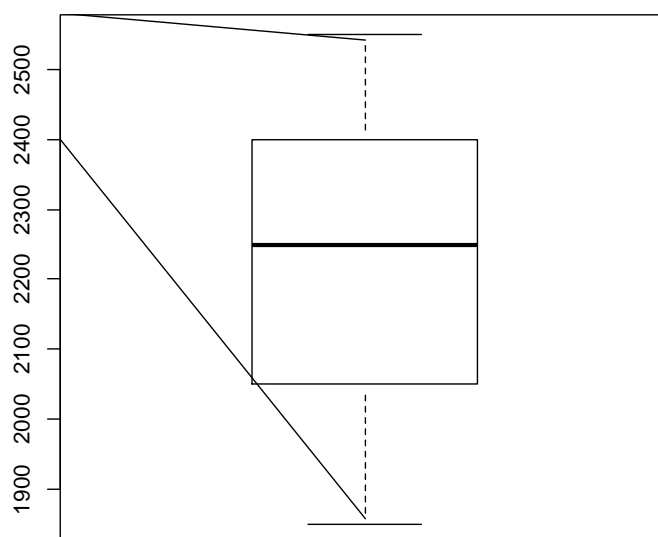


Gráfico 4-3 Diagrama de caja y bigotes de datos de la pupila fijada en el centro

Realizado por: Pillajo, M. Navarrete, D. 2018

3.1.3 *Detección figura izquierda*

3.1.3.1 Análisis de gráfica

Se copia el vector resultante de la posición de la pupila, cuando la visión está dirigida hacia la izquierda, en una columna de Excel y se observa su curva en relación a los límites superior e inferior del rango establecido para la detección representada en la gráfica 5-3.

La gráfica muestra tres líneas de tendencia, una línea gris que representa el límite superior o izquierdo, una línea azul que representa el límite inferior o derecho y una gráfica naranja que presenta una acumulación de valores por arriba del límite superior, esto se debe a que al dirigir la mirada hacia la izquierda la pupila presenta estas posiciones.

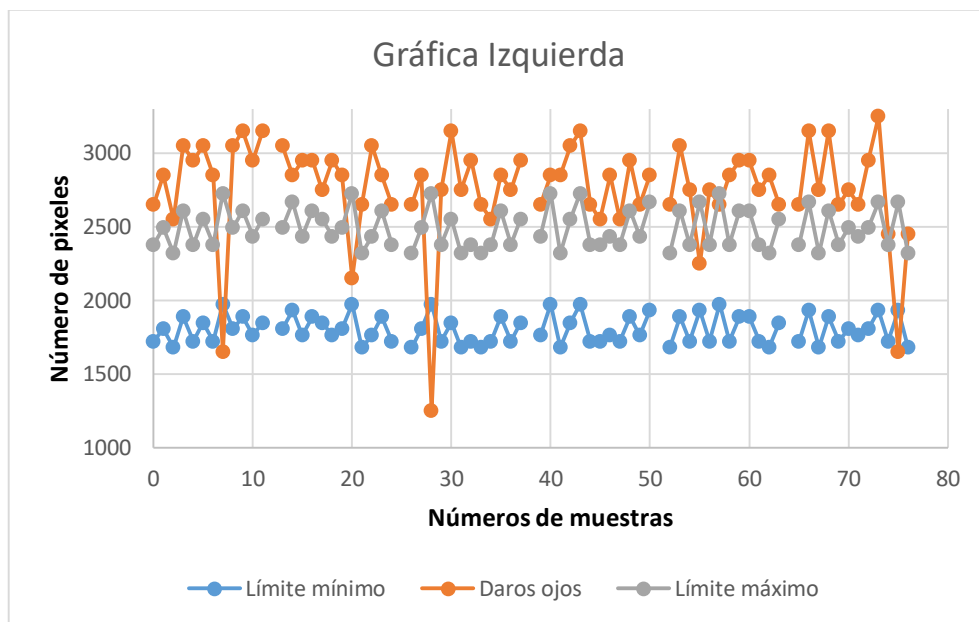


Gráfico 5-3 Gráfica de conjunto de datos s de la pupila en movimiento izquierda

Realizado por: Pillajo, M. Navarrete, D. 2018

3.1.3.2 Análisis de dispersión de datos

Se copia los datos obtenidos en un script del software R y se obtiene los siguientes resultados como se ve en la figura 4-3:

```

R Console
> C<-read.table("D:/TESIS/Estadística/Prueba1/Izquierda.txt",header=T)
> z=as.numeric(unlist(C))
> boxplot(z)
> summary(z)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1250   2650   2850   2764   2950   3250
> |

```

Figura 4-3 Captura ingreso de conjunto de datos de la pupila en movimiento izquierda.

Fuente: Pillajo, M. Navarrete, D. 2018

Los datos que se obtienen, de la fijación de la mirada hacia la izquierda, se ven resumidos en la tabla 3-3 y permite contruir el diagrama de caja y bigotes representado en la gráfica 7-3.

La parte superior de la caja es menor que la inferior; ello quiere decir que los valores comprendidos entre el 75% y el 50% del conjunto de datos (2950 y 2850) están menos dispersos que entre el 25% y el 50% (2650 y 2850). El bigote superior o cuartil 3 es más corto que el inferior; por ello el 25% de los datos de mayor valor están más concentrados que el 25% de los de menor valor. El rango intercuartílico = $Q3 - Q1 = 300$; es decir, el 50% de los datos presentan una variación de un valor de 300. La mediana que nos presenta el gráfico es 2850 es decir que la mayoría de los datos tienden a éste valor, con la ayuda de éste dato se puede discriminar un movimiento de la pupila hacia la izquierda, comparando su posición con respecto a éste valor.

Tabla 3-3 Resumen valores estadísticos de conjunto de datos de las posiciones captadas de la pupila en movimiento izquierda

Estadístico	Valor
Valor mínimo	1250
Primer cuartil (Q1)	2650
Mediana	2850
Media	2764
Tercer cuartil (Q3)	2950
Valor máximo	3250

Realizado por: Pillajo, M. Navarrete, D. 2018

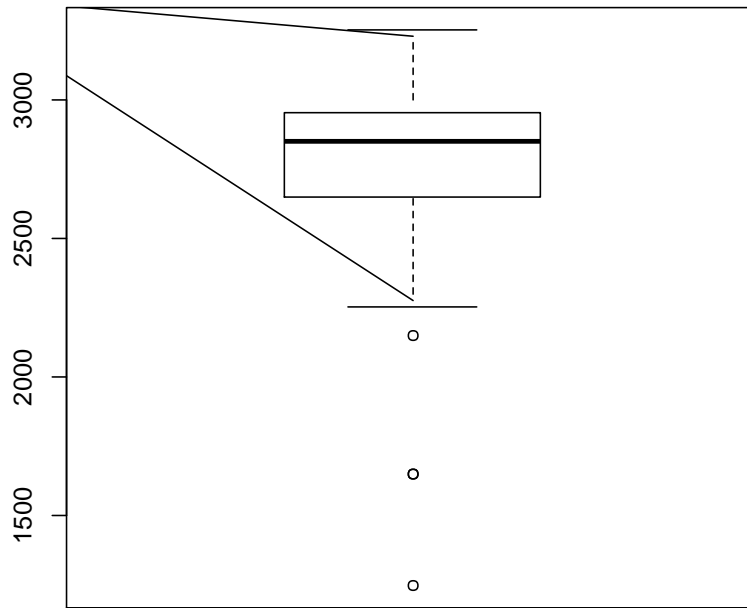


Gráfico 6-3 Diagrama de caja y bigotes de la pupila en movimiento izquierda

Realizado por: Pillajo, M. Navarrete, D. 2018

3.2 Pruebas de precisión del sistema

Utilizando el software R permite realizar gráficas de frecuencia, por una parte los datos antes del procesamiento (1600 aproximadamente) y posteriormente con los datos después del procesamiento (115 aproximadamente) descrito en el numeral 2.3.4.2 del marco metodológico.

3.2.1 *Detección figura derecha*

El contraste evidencia como la precisión mejora después del procesamiento de un 64.3% a un 88.9% debido a que se elimina en gran parte el error de selección con respecto al centro en un 18.6% y con respecto a la izquierda en su totalidad, esto se representa en el gráfico 7-3.

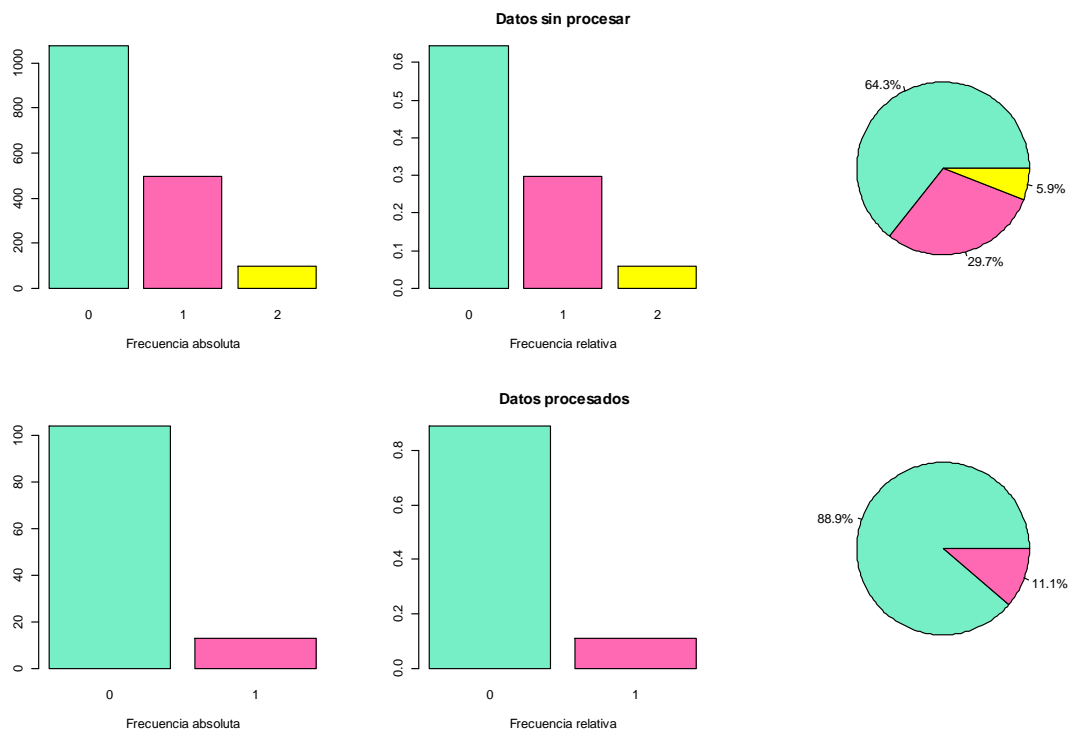


Gráfico 7-3 Análisis de aciertos con mirada hacia la izquierda

Realizado por: Pillajo, M. Navarrete, D. 2018

3.2.2 *Detección figura centro*

El contraste evidencia como la precisión mejora después del procesamiento de un 60.6% a un 78.6% debido a que se elimina en gran parte el error de selección con respecto a la derecha en un 3.2% y con respecto a la izquierda en un 14.8%.

La gráfica 8-3 permite apreciar los resultados obtenidos al realizar la corrección del error cuando se fija la mirada hacia el centro.

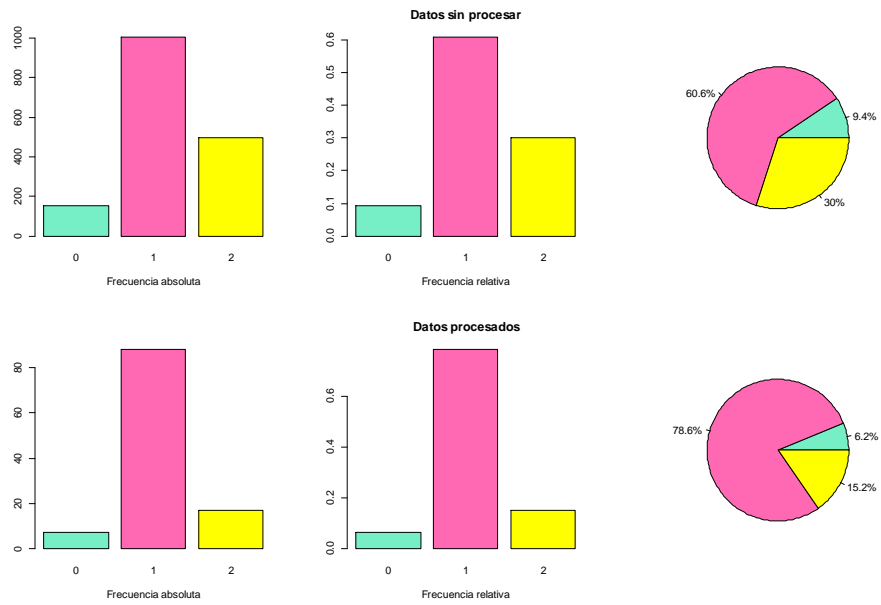


Gráfico 8-3 Análisis de aciertos con ma mirada hacia el centro

Realizado por: Pillajo, M. Navarrete, D. 2018

3.2.3 *Detección figura izquierda*

El contraste evidencia como la precisión mejora después del procesamiento de un 90.1% a un 98.1% debido a que se elimina en gran parte el error de selección con respecto a la derecha en un 4.4% y con respecto a la derecha en su totalidad como se aprecia en la gráfica 9-3.

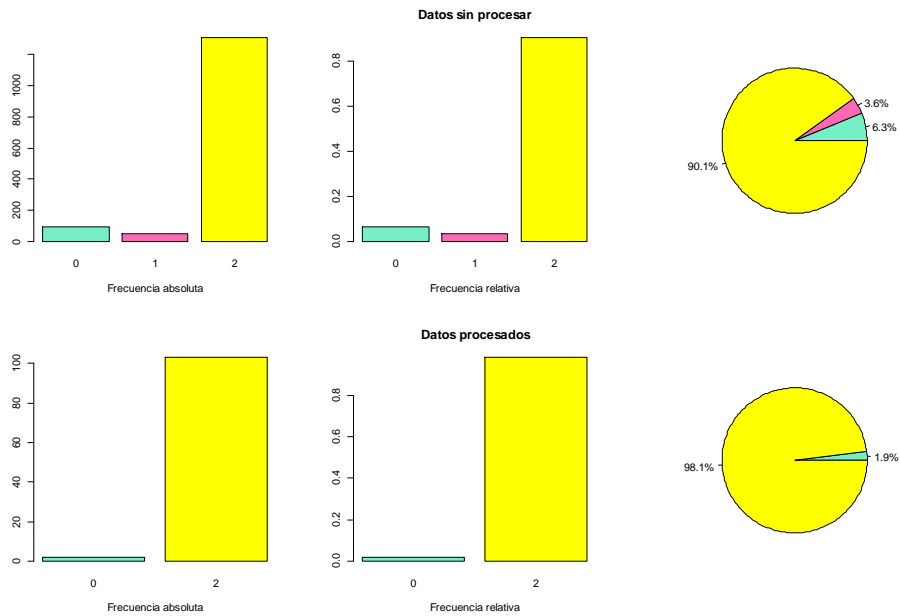


Gráfico 9-3 Análisis de aciertos con ma mirada hacia la derecha

Realizado por: Pillajo, M. Navarrete, D. 2018

3.3 Análisis de costos

3.3.1 Costos de hardware

Los equipos involucrados en el sistema se detalla en la tabla 4-3 a continuación:

Tabla 4-3 Costos de hardware

Cantidad	Detalle	Valor unitario	Total
1	Computadora con procesador Intel I3 o mejor.	\$ 500,00	\$ 420,00
1	WebCam con conexión usb con resolución mínima de 1 Megapixel.	\$ 20,00	\$ 20,00
TOTAL			\$440,00

Realizado por: Pillajo, M. Navarrete, D. 2018

3.3.2 Costos de software de desarrollo

El software involucrado en el sistema se detalla en la siguiente tabla 5-3:

Tabla 5-3 Costos de software de desarrollo

Cantidad	Detalle	Valor unitario	Total
1	Python	\$ 0,00	\$ 0,00
1	Qt Creator	\$ 0,00	\$ 0,00
TOTAL			\$ 0,00

Realizado por: Pillajo, M. Navarrete, D. 2018

3.3.3 Costo de instalación

El tiempo estimado que le tomaría a un técnico informático instalar el sistema es de máximo una hora lo cual se detalla en la tabla 6-3:

Tabla 6-3 Costo de instalación

Cantidad	Detalle	Valor unitario	Total
1	Hora técnico informático	\$ 25,00	\$ 25,00
TOTAL			\$ 25,00

Realizado por: Pillajo, M. Navarrete, D. 2018

3.3.4 *Costo total de implementación*

El costo total de implementación del programa se calcula en función de los costos anteriormente definidos recopilados en la tabla 7-3.

Tabla 7-3 Costo total de implementación

Detalle	Costos
Costos de hardware	\$ 440,00
Costos de software de desarrollo	\$ 0,00
Costos de instalación	\$ 25,00
TOTAL	465,00

Realizado por: Pillajo, M. Navarrete, D. 2018

3.4 **Análisis de resultados obtenidos**

3.4.1 *¿Se pueden aplicar técnicas de visión artificial para distinguir movimientos voluntarios de las pupilas?*

Es posible aplicar técnicas de visión artificial para distinguir movimientos voluntarios de las pupilas debido a que se parametrizan los rangos de desplazamiento mínimo para que sea considerado como un movimiento intencionado, gracias a esto el algoritmo discrimina los movimientos involuntarios al no cumplir con los parámetros establecidos.

3.4.2 ¿Cómo se puede usar el movimiento de las pupilas como medio para la comunicación alternativa en personas con necesidades especiales?

El seguimiento visual es una de las características que se conservan a pesar de un traumatismo motor, por lo que su uso en sistemas alternativos de comunicación resulta bastante conveniente, actualmente los sistemas SAAC utilizan cartillas físicas con pictogramas para interpretar mensajes a través de la fijación visual.

A pesar de sus beneficios éstos sistemas son muy dependientes de un ser humano que identifique, analice y determine el elemento seleccionado por el usuario, al digitalizar éste sistema se puede lograr una mayor independencia del beneficiario.

3.4.3 ¿De qué manera se logra una correcta segmentación y extracción de características de la pupila?

La identificación de las pupilas requiere la segmentación previa del rostro, a partir del cual se enmarcan los ojos utilizando el algoritmo Haarcascade, una vez obtenida ésta sección de interés se procede a definir las pupilas en base a sus características morfológicas usando principalmente el algoritmo de Hough el siguiente paso es determinar el movimiento de las pupilas a partir del desplazamiento de píxeles.

CONCLUSIONES

- Se analizó la capacidad gestual de las personas a quien se dirige el sistema y se observó que presentan un seguimiento visual controlado pero a su vez éste les representa cierto grado de dificultad para efectuar movimientos en sentido diagonal, lo que impidió usar las esquinas como zonas de trabajo para la selección de objetos.
- Se seleccionó un ordenador portátil y su cámara integrada para la implementación hardware por su portabilidad y accesibilidad, además se optó por desarrollar el sistema en los programas: Python, QT Creator y OpenCV debido a su característica de código abierto y sencilla integración entre plataformas.
- Se desarrolló el algoritmo principal en Python haciendo uso de la herramienta OpenCV para el procesamiento de imágenes, posteriormente se integró con la interfaz gráfica de usuario diseñada en QT Creator.
- Se implementó la interfaz gráfica tomando en cuenta la capacidad de direccionamiento visual de personas con discapacidad, así se definió la cantidad, posición y tamaño de los elementos contenidos en la GUI.
- Se validó el sistema realizando pruebas para evaluar su desempeño y precisión, se analizó los resultados y se concluyó que el sistema presenta un nivel de confiabilidad adecuado.

RECOMENDACIONES

- Se recomienda hacer un breve entrenamiento previo con el usuario para lograr un correcto desempeño del sistema, ya que mientras mayor control del seguimiento visual se logre, los movimientos se identificarán con mayor precisión.
- Se recomienda usar un ordenador con una velocidad de procesamiento de 1.5 GHz, para que funcione de forma adecuada, además se recomienda usar la versión 2.7 de Python debido a que posee un mayor desarrollo de librerías en comparación a su versión más actual, en cuanto a Open CV se recomienda usar la versión 3.2 ya que presenta mejoras en relación a su predecesora.
- Se recomienda determinar los valores más adecuados para configurar los algoritmos Haarcascade y Hough en el programa desarrollado en Python, ya que éstos intervienen directamente con el desempeño del sistema.
- Se recomienda que diseñar la interfaz gráfica de usuario con imágenes que representen un concepto o idea de manera clara y concisa, se debe procurar utilizar pictogramas estandarizados para una mejor comprensión.
- Se recomienda realizar las pruebas del sistema en un ambiente preferiblemente cerrado que presente pocas variaciones de luminosidad.

BIBLIOGRAFÍA

ARDILA, Alfredo, ROSSELLI, Mónica & VILLASEÑOR, Esmeralda. *Neuropsicología de los trastornos del aprendizaje*. México-México DF: El manual moderno, 2005. pp. 89-91

Aplicación práctica de la visión artificial en el control de procesos industriales [en línea]. [Consulta: 15 marzo 2018]. Disponible en: http://visionartificial.fpcat.cat/wp-content/uploads/UD_1_didac_Conceptos_previos.pdf

BETKE, GIPS & FLEMING. "The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access for People With Severe Disabilities". *Ieee transactions on neural systems and rehabilitation engineering* [en línea], 2002, (United State of America) 10(1) pp, 1-3 [Consulta: 20 enero 2018]. ISSN 1534-4320. Disponible en: <http://www.cameramouse.org/downloads/CMIEEE.pdf>

CÁCERES TELLO, Jesús. "La visión artificial y las operaciones morfológicas en imágenes binarias". [en línea], 2010, (España) pp. 5-7 [Consulta: 24 febrero 2018]. Disponible en: <https://goo.gl/iqXnS9>

CHIGUANO, Edwin, MORENO, Nathaly & CORRALES, Luis. *Diseño e implementación de un sistema traductor de lenguaje de señas de manos a un lenguaje de texto mediante visión artificial en un ambiente controlado* (tesis).Escuela Politécnica Nacional. 2011. pp. 23-28.

CHIN, A, BARRETO, Armando. "Integrated electromyogram and eye-gaze tracking cursor control system for computer users with motor disabilities", *Journal of Rehabilitation Research & Development* [en línea], 2008, (United State of America) 45 pp. 161-174 [Consulta: 20 enero 2018]. Disponible en: <https://www.ncbi.nlm.nih.gov/pubmed/18566935>

DE LA ESCALERA, A. Visión por computador [en línea]. [Consulta: 28 marzo 2018]. Disponible en: <http://ocw.uc3m.es/ingenieria-de-sistemas-y-automatica/sistemas-percepcion/material-de-clase-1/libro/capitulo-2.pdf>

GONZÁLEZ AGUILERA, D. Procesamiento de imágenes [en línea]. [Consulta: 20 marzo 2018]. Disponible en: <http://ocw.usal.es/eduCommons/enseanzas-tecnicas/procesamiento-avanzado-de-imagenes-digitales/contenidos/Tema2.pdf>

Los Sistemas Aumentativos y Alternativos de Comunicación [en línea]. 2015 [Consulta: 15 febrero 2018]. Disponible en: <http://www.asha.org/public/speech/disorders/AAC.htm?LangType=1034>

LUPU, Robert, BOZOMITU, Radu & UNGUREANU, Florina. "Eye tracking based communication system for patient with major neuro-locomotor disabilities". *Ieee* [en línea], 2011, (United State of America) pp. 1-5 [Consulta: 24 enero 2018]. Disponible en: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6085677&isnumber=6085648>

LUPU, Robert, UNGUREANU, Florina & SIRITEAN, Valentin. "Eye Tracking Mouse for Human Computer Interaction". *Ieee* [en línea], 2013, (United State of America) 10(1) pp, 1-3 [Consulta: 24 enero 2018]. ISSN 4799-2373. Disponible en: <http://www.cameramouse.org/downloads/CMIEEE.pdf>

O'GRADY, R.; COHEN, C.; BEACH, G.; MOODY, G. "NaviGaze: enabling access to digital media for the profoundly disabled". *Ieee* [en línea], 2004, (United State of America) 10(1) pp, 1-3 [Consulta: 27 enero 2018]. ISSN 1550-5219. Disponible en: <https://ieeexplore.ieee.org/document/1409700>

PEÑA BUENO, Leidy Marcela, & RODRIGUEZ LOZANO, Yesid. Dispositivo apuntador mediante visión artificial, adecuado para usuarios de computador con discapacidad motora en miembros superiores [En línea] (tesis).(Maestría) Universidad distrital Francisco José de Caldas, Bogotá, Colombia. 2015. pp. 21-24. [Consulta: 2018-02-01]. Disponible en : <http://repository.udistrital.edu.co/bitstream/11349/2251/1/Pe%C3%B1aBuenoLeidyMarcela2015.pdf>

Python documentation [en línea]. [Consulta: 15 marzo 2018]. Disponible en: <https://docs.python.org/3/>

REYES HIDALGO, Jaime Andrés, & ÁLVAREZ TISCORNIA, Florencia María. *Campaña Ni Más, Ni menos, Yo puedo concientizar sobre la parálisis cerebral* (tesis).Universidad San Francisco De Quito. 2016. pp. 74-79.

RUBIO, Jefferson & VINTIMILLA, Boris. *Valoración y análisis de los movimientos de las manos de un paciente de Parkinson según la escala UPDRS usando técnicas de visión artificial con Kinect* (tesis). Escuela Superior Politécnica del Litoral. 2015. pp. 31-33.

SURDILOVIC, Tihomir. *Fuzzy Mouse Cursor Control System for Computer Users with Spinal Cord Injuries* (tesis).Universidad del Estado de Georgia. 2006. pp. 14-17.

VELARDE, Ma. Paz, PERUGACHI, Erika, VINTIMILLA, Boris & ROMERO, Dennis. *Seguimiento y análisis del movimiento de las extremidades superiores aplicado a la rehabilitación física de un paciente usando técnicas de visión artificial* (tesis). Escuela Superior Politécnica del Litoral. 2016. pp. 29-32.

VIOLA, Paul & JONES, Michael. “Robust Real-Time Face Detection”. *International Journal of Computer Vision* [en línea], 2004, (United State of America) 57(2) pp. 5-8 [Consulta: 20 enero 2018]. ISSN 137–154. Disponible en: <http://www.face-rec.org/algorithms/boosting-ensemble/16981346.pdf>

VIVES, Luis, MEJÍA, Hever, VILCHERREZ, Kevin & VASSALLO, Marcelo. “ Visión artificial: aplicación de filtros y segmentación en imágenes ”. *Ingeniería: Ciencia, Tecnología e Innovación* [en línea], 2014, (España) 1(2) pp. 5-8 [Consulta: 20 enero 2018]. ISSN 2313-1926. Disponible en: <http://revistas.uss.edu.pe/index.php/ING/article/download/198/209>

RODRÍGUEZ ARIAS, M. “Trastornos del Lenguaje: Instrumentos de Recogida de Datos”. *Ieee* [en línea], 2000, (España) 6(4) pp. 6-9 [Consulta: 29 enero 2018]. ISSN 1138-1663. Disponible en: <https://goo.gl/GoLVXp>

SURDILOVIC, Tihomir. *Fuzzy Mouse Cursor Control System for Computer Users with Spinal Cord Injuries* [En línea] (tesis).(Maestría) Georgia State University, Georgia, United State of America. 2006. pp. 15-16. [Consulta: 2018-01-23]. Disponible en: http://scholarworks.gsu.edu/cs_theses/49

ANEXOS

CÓDIGO FUENTE PYTHON

```
from PyQt4.QtGui import QApplication, QMainWindow
from PyQt4 import QtCore, QtGui, uic
import cv2
import numpy as np
import sys
from playsound import playsound
import time
from matplotlib import pyplot as plt

file=open('Expresiones.txt','r')
global texto
texto=file.readlines()
Qt = QtCore.Qt

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

global c1
global c2
global c3
global c4
global c5
global x
global y
global cap
global ejec_cam
global dir_pup_gui
global blink
global cara
global oj
global t
global cont1
global cont2
global cont3
global cont4
cont3 = 0
cont2 = 0
cont1 = 0
cont4 = 0
```

```

proc_cam_paralelo = None
ejec_cam = None
c1 = 0
c2 = 0
c3 = 0
c4 = 0
c5 = 0
x = np.array([1, 2, 3])
y = np.array([0, 0, 0])
i = 0
cap = None
face_cascade =
cv2.CascadeClassifier('C:\opencv\sources\data\haarcascades\haarcascade_frontalface_d
efault.xml')
eye_cascade =
cv2.CascadeClassifier('C:\opencv\sources\data\haarcascades\haarcascade_eye.xml')
t=0

```

```

class Ui_Form(QtGui.QWidget):

```

```

    global blink
    def __init__(self):
        super(Ui_Form, self).__init__()
        uic.loadUi('Menu_1_V5.ui',self)
        global ejec_cam
        i1=str(x[0])+str(y[0])
        i2=str(x[1])+str(y[1])
        i3=str(x[2])+str(y[2])
        l1 = int(i1)
        l2 = int(i2)
        l3 = int(i3)
        self.bt_1.setStyleSheet(("background-image:
url(D:/SV/Pictos/"+i1+".jpg);\n""\n"""))
        self.lb_1.setText(_translate("Form", texto[l1], None))
        self.bt_2.setStyleSheet(("background-image:
url(D:/SV/Pictos/"+i2+".jpg);\n""\n"""))
        self.lb_1.setText(_translate("Form",texto[l2], None))
        self.bt_3.setStyleSheet(("background-image:
url(D:/SV/Pictos/"+i3+".jpg);\n""\n"""))
        self.lb_1.setText(_translate("Form", texto[l3], None))
        self.connect(self.bt_1, QtCore.SIGNAL(("clicked()")), lambda:self.fbt1())
        self.connect(self.bt_2, QtCore.SIGNAL(("clicked()")), lambda:self.fbt2())
        self.connect(self.bt_3, QtCore.SIGNAL(("clicked()")), lambda:self.fbt3())
        self.connect(self.bt_siguiente, QtCore.SIGNAL(("clicked()")),
lambda:self.fbt_siguiente())
        self.connect(self.bt_principal, QtCore.SIGNAL(("clicked()")),
lambda:self.fbt_principal())
        self.threadclass = rec_pupila()
        self.threadclass.start()

```

```

self.connect(self.threadclass,QtCore.SIGNAL('direccion'), self.Update_Signals)
self.connect(self.threadclass,QtCore.SIGNAL('blink_1'), self.Parpadeo)
ejec_cam = True
self.show()

def fbt1(self):
    global c1
    global c2
    global c3
    global c4
    global c5
    global x
    global y
    global texto
    if c1 > 0 or c2 > 0 or c3 > 0 or c4 > 0 or c5 > 0:
        texto_1 = str(x[0])+str(y[0])
        t = int(texto_1)
        U = self.textEdit.toPlainText().split('\n')
        actual = U[0]
        self.textEdit.setText(actual+ " " + texto[t].decode('latin-1'))
        playsound("D:/SV/Palabras/" + str(t) + ".mp3")
    else:
        c1 = 1
        x = (x[0],x[0],x[0])
        y = (1,2,3)
        i1=str(x[0])+str(y[0])
        i2=str(x[1])+str(y[1])
        i3=str(x[2])+str(y[2])
        l1 = int(i1)
        l2 = int(i2)
        l3 = int(i3)
        self.bt_1.setStyleSheet(("background-image:
url(D:/SV/Pictos/" +i1+".jpg);\n""\n"""))
        self.lb_1.setText(_translate("Form", texto[l1].decode('latin-1'), None))
        self.bt_2.setStyleSheet(("background-image:
url(D:/SV/Pictos/" +i2+".jpg);\n""\n"""))
        self.lb_2.setText(_translate("Form", texto[l2].decode('latin-1'), None))
        self.bt_3.setStyleSheet(("background-image:
url(D:/SV/Pictos/" +i3+".jpg);\n""\n"""))
        self.lb_3.setText(_translate("Form", texto[l3].decode('latin-1'), None))
    if len(self.textEdit.toPlainText().split('\n')[0]) >= 248:
        self.textEdit.setText("")
    return (c1,c2,c3,c4,c5)

def fbt2(self):
    global c1
    global c2
    global c3

```

```

global c4
global c5
global x
global y
global texto
if c1 > 0 or c2 > 0 or c3 > 0 or c4 > 0 or c5 > 0:
    texto_1 = str(x[1])+str(y[1])
    t = int(texto_1)
    U = self.textEdit.toPlainText().split('\n')
    actual = U[0]
    self.textEdit.setText(actual+ " " + texto[t].decode('latin-1'))
    playsound("D:/SV/Palabras/" + str(t) + ".mp3")

else:
    c2 = 1
    x = (x[1],x[1],x[1])
    y = (1,2,3)
    i1=str(x[0])+str(y[0])
    i2=str(x[1])+str(y[1])
    i3=str(x[2])+str(y[2])
    l1 = int(i1)
    l2 = int(i2)
    l3 = int(i3)
    self.bt_1.setStyleSheet(("background-image:
url(D:/SV/Pictos/" +i1+".jpg);\n""\n"""))
    self.lb_1.setText(_translate("Form", texto[l1].decode('latin-1'), None))
    self.bt_2.setStyleSheet(("background-image:
url(D:/SV/Pictos/" +i2+".jpg);\n""\n"""))
    self.lb_2.setText(_translate("Form", texto[l2].decode('latin-1'), None))
    self.bt_3.setStyleSheet(("background-image:
url(D:/SV/Pictos/" +i3+".jpg);\n""\n"""))
    self.lb_3.setText(_translate("Form", texto[l3].decode('latin-1'), None))
if len(self.textEdit.toPlainText().split('\n')[0]) >= 248:
    self.textEdit.setText("")
return (c1,c2,c3,c4,c5)

def fbt3(self):
    global c1
    global c2
    global c3
    global c4
    global c5
    global x
    global y
    global texto
if c1 > 0 or c2 > 0 or c3 > 0 or c4 > 0 or c5 > 0:
    texto_1 = str(x[2])+str(y[2])
    t = int(texto_1)

```

```

    U = self.textEdit.toPlainText().split('\n')
    actual = U[0]
    self.textEdit.setText(actual+ " " + texto[t].decode('latin-1'))
    playsound("D:/SV/Palabras/" + str(t) + ".mp3")
else:
    c3 = 1
    x = (x[2],x[2],x[2])
    y = (1,2,3)
    i1=str(x[0])+str(y[0])
    i2=str(x[1])+str(y[1])
    i3=str(x[2])+str(y[2])
    l1 = int(i1)
    l2 = int(i2)
    l3 = int(i3)
    self.bt_1.setStyleSheet(("background-image:
url(D:/SV/Pictos/" +i1+".jpg);\n""\n"""))
    self.lb_1.setText(_translate("Form", texto[l1].decode('latin-1'), None))
    self.bt_2.setStyleSheet(("background-image:
url(D:/SV/Pictos/" +i2+".jpg);\n""\n"""))
    self.lb_2.setText(_translate("Form", texto[l2].decode('latin-1'), None))
    self.bt_3.setStyleSheet(("background-image:
url(D:/SV/Pictos/" +i3+".jpg);\n""\n"""))
    self.lb_3.setText(_translate("Form", texto[l3].decode('latin-1'), None))
if len(self.textEdit.toPlainText().split('\n')[0]) >= 248:
    self.textEdit.setText("")
return (c1,c2,c3,c4,c5)

```

```

def fbt_siguiente(self):
    global c1
    global c2
    global c3
    global c4
    global c5
    global x
    global y
    global s
    global texto
    s = np.array([3,3,3])
    if c1 > 0 or c2 > 0 or c3 > 0 or c4 > 0 or c5 > 0:
        if y[2] < 8:
            y = y + s
        else:
            y = np.array([1,2,3])
    else:
        if x[2] < 5:
            x = x + s
        else:

```

```

        x = np.array([1,2,3])
        fig = np.array(["00", "00", "00"])
        for i in range(len(fig)):
            fig[i]=str(x[i])+str(y[i])
            l1 = int(fig[0])
            l2 = int(fig[1])
            l3 = int(fig[2])
            self.bt_1.setStyleSheet(("background-image:
url(D:/SV/Pictos/"+fig[0]+".jpg);\n""\n"""))
            self.lb_1.setText(_translate("Form", texto[l1].decode('latin-1'), None))
            self.bt_2.setStyleSheet(("background-image:
url(D:/SV/Pictos/"+fig[1]+".jpg);\n""\n"""))
            self.lb_2.setText(_translate("Form", texto[l2].decode('latin-1'), None))
            self.bt_3.setStyleSheet(("background-image:
url(D:/SV/Pictos/"+fig[2]+".jpg);\n""\n"""))
            self.lb_3.setText(_translate("Form", texto[l3].decode('latin-1'), None))
        return (c1,c2,c3,c4,c5)

```

```

def fbt_principal(self):

```

```

    global c1
    global c2
    global c3
    global c4
    global c5
    global x
    global y
    c1 = 0
    c2 = 0
    c3 = 0
    c4 = 0
    c5 = 0
    x = np.array([1, 2, 3])
    y = np.array([0, 0, 0])
    i1=str(x[0])+str(y[0])
    i2=str(x[1])+str(y[1])
    i3=str(x[2])+str(y[2])
    l1 = int(i1)
    l2 = int(i2)
    l3 = int(i3)
    self.bt_1.setStyleSheet(("background-image:
url(D:/SV/Pictos/"+i1+".jpg);\n""\n"""))
    self.lb_1.setText(_translate("Form", texto[l1].decode('latin-1'), None))
    self.bt_2.setStyleSheet(("background-image:
url(D:/SV/Pictos/"+i2+".jpg);\n""\n"""))
    self.lb_2.setText(_translate("Form", texto[l2].decode('latin-1'), None))
    self.bt_3.setStyleSheet(("background-image:
url(D:/SV/Pictos/"+i3+".jpg);\n""\n"""))
    self.lb_3.setText(_translate("Form", texto[l3].decode('latin-1'), None))

```

```

def closeEvent(self, event):
    global ejec_cam
    ejec_cam = False

def Update_Signals(self):
    global dir_pup_gui
    global cont1
    global cont2
    global cont3
    if dir_pup_gui == 2:
        cont2 = 0
        cont3 = 0
        self.lb_bt1.setStyleSheet(("background-color: rgb("+str(cont1*85)+"", 0, 255);"))
        if cont1 == 3:
            cont1 = 0
            self.bt_1.click()
        cont1+=1
    else:
        self.lb_bt1.setStyleSheet(("background-color: rgb(255, 255, 255);"))
    if dir_pup_gui == 1:
        cont1 = 0
        cont3 = 0
        self.lb_bt2.setStyleSheet(("background-color: rgb("+str(cont2*85)+"", 0, 255);"))
        if cont2 == 3:
            cont2 = 0
            self.bt_2.click()
        cont2+=1
    else:
        self.lb_bt2.setStyleSheet(("background-color: rgb(255, 255, 255);"))
    if dir_pup_gui == 0:
        cont2 = 0
        cont1 = 0
        self.lb_bt3.setStyleSheet(("background-color: rgb("+str(cont3*85)+"", 0, 255);"))
        if cont3 == 3:
            cont3 = 0
            self.bt_3.click()
        cont3+=1
    else:
        self.lb_bt3.setStyleSheet(("background-color: rgb(255, 255, 255);"))
    print dir_pup_gui

def Parpadeo(self):
    global t
    global cont1
    global cont2
    global cont3
    global cont4

```

```

if blink == True:
    self.lb_blink.setStyleSheet(("background-color: rgb(85, 255, 0);"))
    cont4+=1
    t=1
    if cont4>=5 and cont4<10:
        if cont4 == 5:
            playsound("D:/SV/Palabras/1.mp3")
    if cont4>=10:
        if cont4 == 10:
            playsound("D:/SV/Palabras/2.mp3")
else:
    self.lb_blink.setStyleSheet(("background-color: rgb(57, 173, 0);"))
    if cont4>=5 and cont4<10:
        self.bt_siguiente.click()
        cont3 = 0
        cont2 = 0
        cont1 = 0
        cont4 = 0
        t = 0
        time.sleep(0.1)
    if cont4>=10:
        self.bt_principal.click()
        cont3 = 0
        cont2 = 0
        cont1 = 0
        cont4 = 0
        t = 0
        time.sleep(0.1)
    if t == 1:
        ##self.bt_siguiente.click()
        t=0
        ##cont3 = 0
        ##cont2 = 0
        ##cont1 = 0
        cont4 = 0
if cara == True:
    self.lb_cara.setStyleSheet(("background-color: rgb(0, 49, 255);"))
    i=0
else:
    self.lb_cara.setStyleSheet(("background-color: rgb(0, 49, 184);"))
if oj == True:
    self.lb_oj.setStyleSheet(("background-color: rgb(255, 255, 0);"))
    i=0
else:
    self.lb_oj.setStyleSheet(("background-color: rgb(195, 195, 0);"))

```

```

class rec_pupila(QtCore.QThread,QtGui.QMainWindow):

```



```
global cap
global ejec_cam
global dir_pup_gui
global cara
global oj
global blink
```

```
def __init__(self, parent = None):
    super(rec_pupila, self).__init__(parent)
```

```
def cont_pixeles_negros(e_matriz):
    (m, n) = e_matriz.shape
    num_pix_w = 0
    for i in range(m):
        for j in range(n):
            if e_matriz[i, j] == 0:
                num_pix_w = num_pix_w + 1
    return num_pix_w
```

```
def dir_pupila(l_d, l_i):
    dir_p = -1
    if (l_i is not None) and (l_d is not None) and (l_d == l_i):
        if l_d == 0:
            dir_p = 0
        if l_d == 1:
            dir_p = 1
        if l_d == 2:
            dir_p = 2
    return dir_p
```

```
def run (self):
    def cont_pixeles_negros(e_matriz):
        (m, n) = e_matriz.shape.
        num_pix_w = 0
        for i in range(m):
            for j in range(n):
                if e_matriz[i, j] == 0:
                    num_pix_w = num_pix_w + 1
        return num_pix_w
```

```
def dir_pupila(l_d, l_i):
    dir_p = -1
    if (l_i is not None) and (l_d is not None) and (l_d == l_i):
        if l_d == 0:
            dir_p = 0
        if l_d == 1:
            dir_p = 1
        if l_d == 2:
```

```

        dir_p = 2
    return dir_p

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 800);
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 600);
thrs1 = 10
thrs2 = 457163531
thrs3 = 243 ##243
thrs4 = 208 ##208
thrs5 = 75
thrs6 = 4
v_pos = []
while ejec_cam:
    ret, img = cap.read()
    ret, org = cap.read()
    ogray = cv2.cvtColor(org, cv2.COLOR_BGR2GRAY)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    clahe = cv2.createCLAHE(clipLimit=1.0, tileGridSize=(8,8))
    cli1 = clahe.apply(gray)
    faces = face_cascade.detectMultiScale(cli1,1.8,5)
    global cara
    if faces is None:
        cara = False
    for (x,y,w,h) in faces:
        cara = True
        pos_eyes = 0
        #### Graficar rectangulo alrededor de la zona de los ojos
        cv2.rectangle(img,(x+int(0.13*w),y+int(0.13*h)), (x+w-int(0.13*w) , y+h-
int(0.45*h)),(255,0,0),4)
        #### Extraer zona de los ojos de imagen original
        roi_img = img[y+int(0.13*h):y+h-int(0.45*h), x+int(0.13*w):x+w-
int(0.13*w)]
        #### Extraer zona de los ojos de imagen ecualizada
        roi_cli1 = cli1[y+int(0.13*h):y+h-int(0.45*h), x+int(0.13*w):x+w-
int(0.13*w)]
        #### Deteccion de los ojos
        eyes = eye_cascade.detectMultiScale(roi_cli1,1.1,8)
        global blink
        global oj
        if len(eyes) == 0:
            blink = True
            oj = False
        if len(eyes) == 2:
            oj = True
            blink = False
            j = 0
            lado_der = None

```

```

lado_izq = None
for (ex,ey,ew,eh) in eyes:
    oj = True
    lado = None
    pixeles_b = []
    cv2.rectangle(roi_img, (ex+int(0*ew),ey+int(0.15*eh)),(ex+ew-
int(0*ew),ey+eh-int(0.15*eh)),(0,255,0),1)
    e_img = roi_img[ey+int(0.20*eh):ey+eh-int(0.20*eh),
ex+int(0.13*ew):ex+ew-int(0.13*ew)]
    e_gray = cv2.cvtColor(e_img, cv2.COLOR_BGR2GRAY)
    erb = roi_img[ey+int(0.20*eh):ey+eh-int(0.20*eh),
ex+int(0.13*ew):ex+ew-int(0.13*ew)]
    er = cv2.medianBlur(erb,15)
    e_gauss = cv2.GaussianBlur(erb, (5,5), 0)
    e_canny = cv2.Canny(e_gauss, thrs3, thrs4)
    c_small = cv2.resize(e_canny, (0,0), fx=10, fy=10)
    ##c_small = cv2.resize(e_canny, (0,0), fx=10, fy=10)
    e_amp = cv2.resize(e_img,(0,0),fx=10,fy=10)
    min_d = np.int(np.round(0.95*c_small.shape[1]/3)
    max_d = np.int(np.round(1.05*c_small.shape[1]/3)

##### Extraer imagen ojo en blanco y negro
    e_gauss_rsz = cv2.resize(e_gauss, (0,0), fx = 10, fy = 10)
    ret, e_gauss_bw =
cv2.threshold(e_gauss_rsz,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    circles1 = cv2.HoughCircles(c_small, cv2.HOUGH_GRADIENT,
thrs1+(thrs2/1000000000), 150, 500, 200, min_d/2, max_d/2)
    circles1 = np.matrix(circles1)

##### Eliminar círculos cerca de los bordes inferior y superior
    i = 0
    if np.size(circles1) > 1:
        while i < circles1.shape[0]:
            if (circles1[i,1] > c_small.shape[0]*0.65) or (circles1[i,1] <
c_small.shape[0]*0.35):
                circles1 = np.delete(circles1, i, 0)
                i = i - 1
                i = i + 1

##### Eliminar círculos que excedan los límites horizontales
    i = 0
    while i < circles1.shape[0]:
        if (circles1[i, 0] + circles1[i, 2] >= c_small.shape[1]) or (circles1[i,
0] - circles1[i, 2] <= 0):
            circles1 = np.delete(circles1, i, 0)
            i = i - 1
            i = i + 1

```

```

##### Eliminar círculos excedentes
if circles1.shape[0] > 1:
    for i in range(circles1.shape[0]):
        e_gauss_bw_cir = e_gauss_bw[int(circles1[i,1] - circles1[i, 2]): \
                                   int(circles1[i,1] + circles1[i, 2]), \
                                   int(circles1[i,0] - circles1[i, 2]): \
                                   int(circles1[i,0] + circles1[i, 2])]
        pixeles_b.append(cont_pixeles_negros(e_gauss_bw_cir))
    circles1 = circles1[np.argmax(pixeles_b),:]

##### Graficar círculos detectados
if circles1 is not None:
    (a, b) = circles1.shape
    for i in range(a):
        cv2.circle(c_small,(circles1[i, 0], circles1[i, 1]), circles1[i,2],
(255, 255, 255), 1, cv2.LINE_AA)
        if (circles1[i, 0] > 0) and (circles1[i,0] <
e_gauss_bw.shape[1]*0.44):##originalmente es 0.42
            lado = 0
        elif (circles1[i, 0] >= e_gauss_bw.shape[1]*0.445) and
(circles1[i, 0] < e_gauss_bw.shape[1]*0.58):
            lado = 1
        else:
            lado = 2
        print e_gauss_bw.shape[1]*0.44, " ",circles1[i, 0], "
",e_gauss_bw.shape[1]*0.58, " ",lado
        if lado is not None:
            if j == 0:
                lado_izq = lado
            if j == 1:
                lado_der = lado
            j = j + 1
        v_pos.append(dir_pupila(lado_der, lado_izq))
if len(v_pos) == 7: ##Normalmente es igual a 8
    global dir_pup_gui
    v_pos_mul = [v_pos.count(0), v_pos.count(1), v_pos.count(2)]
    dir_pup_gui = np.argmax(v_pos_mul)
    v_pos = []
    self.emit(QtCore.SIGNAL('direccion'),dir_pup_gui)
    self.emit(QtCore.SIGNAL('blink_1'), blink)
    cv2.imshow('Imagen Segmentada',img)
    ##cv2.imshow('Rostro extraido',roi_cli1)
    ##cv2.imshow('op2',e_amp)
    cv2.imshow("Imagen con bordes",c_small)
    k=cv2.waitKey(30) & 0xff
##cv2.waitKey(0)
if k == 27:
    break

```