UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 5

Título: Algoritmo de codificación para biblioteca de autenticación biométrica a partir del reconocimiento de patrones del iris.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

Autor

Ariel Peláez González.

Tutor

Ing. Alejandro González Abascal.

Alejandro Alfonso Fuster

Co-Tutor

Ing. Enrique Reyes Bermúdez.

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Firma del Autor: Ariel Peláez González.

_____ _____

Firma del Tutor: Alejandro González Abascal. Firma del Tutor: Alejandro Alfonso Fuster.

_____

Firma del Co-Tutor: Enrique Reyes Bermúdez

Datos de contacto

Tutor: Ing. Alejandro González Abascal.

Edad: 24 años.

Ciudadanía: cubano.

Institución: Universidad de Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

Categoría Docente:

E-mail: agabascal@uci.cu

Tutor: Alejandro Alonso Fuster.

Edad: 30 años.

Ciudadanía: cubano.

Institución: Universidad de Ciencias Informáticas (UCI).

Título: Licenciatura en Matemática.

Categoría Docente: Profesor asistente.

E-mail: alejandroaf@uci.cu

Co-Tutor: Ing. Enrique Reyes Bermúdez.

Edad: 26 años.

Ciudadanía: cubano.

Institución: Universidad de Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

Categoría Docente: Profesor Instructor.

E-mail: ereyes@uci.cu

Dedicatoria

A mis padres,

Que han sido mi punto de apoyo

Y son la causa de todo lo bueno que he alcanzado en la vida.

Agradecimientos

A mis padres por confiar siempre en mí y apoyarme en los buenos y malos momentos.

A mi familia por educarme y hacerme sentir que siempre tendré con quien contar.

A Yareivys por convertirse en parte de mí.

A Abascal porque siempre estuvo ahí cuando necesité apoyo, por ser el que dio los primeros pasos en este trabajo y por ser mi amigo.

A Fuster por ponerle el corazón para que este trabajo saliera adelante.

A Kike, Yolier y todos los que me apoyaron cuando lo necesité.

A Hardys, Tony, Maikel y todo el tribunal que me ayudó a corregir errores.

A mis amigos que siempre estuvieron a mi lado durante estos 5 años, a Orea, a Arian, a Osvaldo, a Yanet, y a todos los demás.

A mi gente de la FEU, a Felipe, a Livan, a Noly, a David, a Bencomo y a todos con los que compartimos la organización de cada evento.

Y a todo el que de una forma u otra colaboró con la realización de este sueño.

Resumen

Los sistemas de seguridad de los diferentes complejos industriales en el mundo actual se encuentran en un desarrollo creciente producto al gran número de ataques a los que están sometidos de manera constante los sistemas informáticos modernos. Existen varias formas de fortalecer estos sistemas y una de las más eficientes es contar con diversas vías de autenticación, entre las que se destacan las que se llevan a cabo a partir de parámetros biométricos. Teniendo en cuenta lo anterior surge la necesidad de incrementar estas técnicas en nuestro país para evitar posibles vulnerabilidades. Para dar solución a esta cuestión se realiza el siguiente trabajo que tiene como objetivo solucionar la fase de codificación para la elaboración de una biblioteca de autenticación mediante el análisis de patrones oculares en el proyecto Seguridad del Centro de Informática Industrial (CEDIN), de la Universidad de Ciencias Informáticas (UCI), sobre plataforma libre capaz de brindarles confianza a los usuarios durante su autenticación al sistema.

Se realizó una investigación de las principales tecnologías de desarrollo y bibliotecas existentes en el mundo y se hizo una propuesta de las más importantes todo esto para cumplir el objetivo. Además, se muestra la modelación del sistema a partir de diferentes diagramas y se analiza la implementación del mismo. Por último, se muestran las diferentes pruebas que se le realizaron al algoritmo para determinar su efectividad.

Palabras clave: Autenticación, biometría, codificación, seguridad, sistema.

Índice

Índice de Tablas

Índice de Figuras

Introducción

La informatización de la sociedad es un proceso global y natural consecuencia de los cambios tecnológicos ocurridos en las últimas décadas. En la actualidad se construyen aplicaciones de alta complejidad con relativa facilidad. Surgen fenómenos nuevos como la web 2.0, incrementa el número de aplicaciones para la industria e inteligencia artificial. Las transacciones bancarias en un 97% se hacen de forma digital. La seguridad de instalaciones y/o grandes servidores de información expanden sus horizontes de forma multidireccional. Cuba forma parte de este proceso, en la UCI lo evidencia la creación de los diferentes centros productivos. En el CEDIN existen varios proyectos que al integrarse logran la construcción de SCADAs[1]. Uno de estos proyectos es Seguridad. En vísperas de fortalecer su sistema autenticación y analizando las vulnerabilidades del mismo surgen ideas renovadoras de profundizar en el método de autenticación biométrica.

El proyecto cuenta actualmente con dos formas de autenticación de usuarios: mediante el uso de usuario y contraseña y por el reconocimiento de huellas digitales, luego se encuentra en desarrollo una biblioteca de autenticación biométrica a partir de reconocimiento de patrones del iris para fortalecer el sistema de seguridad del mismo.

Durante el desarrollo de la biblioteca se conoce que una vez solucionada la fase de encontrar la zona del iris en una imagen capturada o pre procesamiento, restan las fases de codificación y reconocimiento. Teniendo en cuenta que en el proyecto se cuenta con un algoritmo que da solución a la fase de pre procesamiento podemos definir la siguiente **situación problémica**: No existe en el módulo de seguridad la fase de codificación para la biblioteca de autenticación biométrica a partir del reconocimiento de patrones del iris con que cuenta el mismo.

Ante esta situación se puede definir el siguiente **problema científico**: ¿Cómo proveer al módulo de seguridad un mecanismo de codificación por reconocimiento de iris?

[1] Siglas en Inglés: Supervisory Control And Data Adquisition (Control Supervisor y Adquisición de Datos).

Para ello, el **objeto de estudio** de esta investigación, se traduce en el reconocimiento de iris como técnica de autenticación de usuarios, centrándose específicamente en el campo de los algoritmos matemáticos para el tratamiento de imágenes de iris.

Para dar respuesta al problema de la investigación se definió que el **objetivo general** consistiera en: Incorporar a la biblioteca de autenticación de usuarios por reconocimiento de iris el algoritmo que dada una correcta localización de la región del iris codifique una imagen determinada.

Conforme a lo planteado como objetivo general se derivan las siguientes **tareas investigativas**:

- Estudio del estado del arte sobre el contenido, que sirva de base para encontrar una solución eficiente al problema en cuestión.
- Búsqueda de elementos reutilizables para la implementación del algoritmo de codificación.
- Obtención de casos de uso y realización del diseño correspondiente.
- Implementación del algoritmo partiendo de los elementos del diseño.
- Definición de los casos y procesos de prueba.
- Comparación de resultados obtenidos con resultados esperados.

Métodos científicos utilizados.

En el transcurso de la investigación llevada a cabo para alcanzar las tareas propuestas se identificaron 3 métodos de tipo teóricos, ellos son:

- **Método Histórico Lógico:** para constatar teóricamente cómo ha evolucionado el desarrollo de los sistemas de autenticación biométrica mediante el análisis de patrones oculares, específicamente mediante reconocimiento de iris, a través de los años.

- **Método Analítico – Sintético:** con el objetivo de analizar la información y la documentación relevante para el desarrollo del software enfatizando en los elementos más importantes que se relacionan con el objeto de estudio.

- **Método Comparativo:** las comparaciones han sido utilizadas de forma exhaustiva en la investigación para lograr tener una visión clara de la complejidad real del problema pues un análisis superficial podría traer serias consecuencias en un ambiente de real.

Resultados esperados al concluir el trabajo de Diploma:

Un componente capaz de construir el iriscode[2], a partir de imágenes segmentadas.

La investigación estará estructurada de la siguiente forma:

Capítulo 1: Se sintetiza sobre los sistemas de autenticación biométrica por reconocimiento de iris, sus fases y características. Se describe el estado del conocimiento de los métodos de codificación como una etapa dentro de un sistema de autenticación por patrones de iris.

Capítulo 2: Se explican las metodologías, tecnologías y herramientas que se van a utilizar para el desarrollo de la aplicación.

Capítulo 3: Se describe el desarrollo de la solución propuesta a partir de la modelación de diagramas de casos de uso, diagramas de secuencia, diagramas de clases. Se plantea el modelo de datos y se especificarán los diferentes patrones utilizados para enriquecer la arquitectura del sistema. Se describe de forma general el diseño estructural de la biblioteca de reconocimiento de iris.

Capítulo 4: Se describen detalles de la implementación del algoritmo y así como los casos y resultados de prueba para los casos de uso críticos. Se especifican los estándares de codificación y documentación utilizados en el desarrollo de la aplicación.

[2] Muestra de información obtenida del iris humano a partir de transformaciones matemáticas.

Capítulo 1: Fundamentación Teórica.

1.1 Introducción al capítulo

El reconocimiento del iris es uno de los avances más interesantes y confiables dentro del reconocimiento de personas. Dentro de un sistema de reconocimiento de iris la codificación es el proceso por el cual se analiza la textura del iris y se extrae información. En este capítulo se expondrán algunas características de iris humano y las fases con que cuenta un sistema de reconocimiento de iris. Además, se hará un repaso de algunos de los métodos de codificación existentes y se describiría el método implementado en el sistema desarrollado.

1.2 El iris humano

El iris está compuesto de tejido fibroso llamado estroma, que conecta músculos encargados de a contraer y dilatar la pupila en respuesta a los cambios de iluminación. El mismo tiene un diámetro aproximado de 11 milímetros, y en su parte externa está rodeado de una capa blanca denominado la esclera o esclerótica.

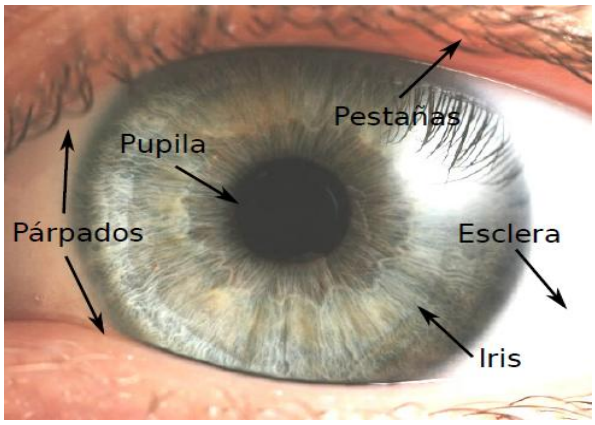


Figura 1: Diagrama del ojo humano.

El tejido que compone el iris genera, como se puede ver en la *figura 1*, un patrón muy complejo, que es generado en forma completamente aleatoria durante los primeros meses de gestación y permanece constante a lo largo de toda la vida. Inclusive, el patrón del iris es independiente en cada ojo: no hay diferencia entre comparar la textura del iris del ojo derecho y del ojo izquierdo de una persona, con comparar la textura del ojo de dos personas diferentes.

Se estima que el iris es una de las características biométricas más ricas en cuanto a información que permite identificar un individuo: por ejemplo, mientras una huella digital tiene aproximadamente entre 40 y 60 características distinguibles, un iris tiene más de 240. Es esta complejidad y aleatoriedad de este patrón la que permite que el iris sea utilizado como característica biométrica.

Otras cualidades que permiten utilizar el iris como característica biométrica son:

1. Es externamente visible: al ser parte del ojo humano, el iris es visible en condiciones normales, lo que permite que la textura sea capturada mediante una simple fotografía.

2. Es sumamente estable a lo largo del tiempo.

3. Es imposible de modificar por medios no-quirúrgicos. Inclusive algunos métodos quirúrgicos en el ojo como operaciones de cataratas mantienen constante la textura del iris. [1]

1.3 Sistemas de reconocimiento de iris

Si bien la idea de utilizar el iris como característica biométrica tiene más de 100 años (propuesta por Bertillon en 1885), la idea de implementar un sistema de reconocimiento automático de iris surge en 1987 por Flom y Safir. La idea fue patentada, pero no se conoce ninguna implementación. La primera implementación de un sistema de reconocimiento de iris fue realizada por Daugman. Este trabajo definió las bases matemáticas y estadísticas que luego se utilizarían en casi todas las implementaciones siguientes. La mayoría de los sistemas de reconocimiento de iris disponibles comercialmente a la fecha utilizan los algoritmos desarrollados por Daugman.

Se pueden mencionar también los sistemas desarrollados por Wildes, Boles y Boashash, Lim y más recientemente, el sistema desarrollado por Monro.

Todos los sistemas de reconocimiento de iris funcionan capturando una imagen de ojo de la persona y, a partir de la textura del iris, generan un código de iris que luego es almacenado en una base de datos y utilizado para la identificación. Todo este proceso está dividido en varias etapas:

1. Primero, se hace una captura de una imagen del ojo de la persona.

2. Una vez capturada la imagen, es necesario aislar la textura del iris y descartar el ruido producido por los párpados, las pestañas y cualquier reflejo ambiente. Esta etapa se denomina segmentación.

3. La textura del iris pasa por un proceso de normalización que debe permitir invarianza[3] frente a variaciones en el proceso de captura, como por ejemplo el tamaño del iris en la imagen o el diámetro de la pupila. Esta textura normalizada es codificada, resultando una representación de la textura del iris que simplificará el proceso de identificación y verificación.

4. Una vez generado el código de iris, se pueden realizar dos acciones:

   Almacenar el código en una base de datos para referencia futura, o bien realizar una búsqueda del código en una base de datos existente con el objetivo de identificar o verificar la identidad de la persona.

[3] Propiedad de ser invariante. Invariante, algo que no cambia al aplicarle un conjunto de transformaciones.

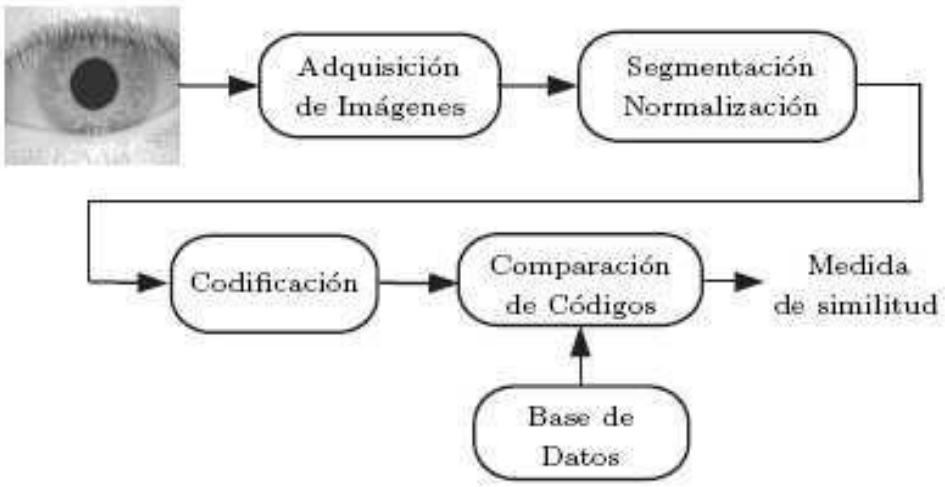


Figura 2: Diagrama en bloque del sistema de reconocimiento del iris.

1.4 Segmentación

En esta etapa se separa la zona que interesa ser procesada, el iris. Hay que seguir algunas premisas de interés para lograr el éxito en esta fase. La primera es el color, la segunda es la diferencia de colores y la tercera es la forma geométrica de lo que se va a extraer. El iris la pupila y la esclera o esclerótica tienen diferente intensidad de gris[4], pero además el iris y la pupila tienen forma casi circular, lo que se asumirá como tal. Además, se nota a simple vista que la mayor variación del gris está justamente en las fronteras de la pupila y el iris. Si se encuentra la forma de determinar la mayor variación de intensidad de gris en una región circular se habrá hallado una de las fronteras circulares observadas. Si se pueden encontrar las dos fronteras ya estamos en presencia de los límites interiores y exteriores del iris.

[4] Referente a escala de grises. Escala empleada en la imagen digital en la que el valor de cada píxel posee un valor equivalente a una graduación de gris. El gris es un color que se encuentra en la naturaleza, se crea mediante la mezcla de blanco y negro en diferentes proporciones.

John Daugman físico-matemático y profesor de la universidad de Cambridge, propone el operador íntegro-diferencial, este operador devuelve un valor máximo de variación para una circunferencia cuyo radio se agranda constantemente y para un punto en el espacio bidimensional. Demostrado empíricamente, el centro de la pupila es el punto donde el operador íntegro-diferencial distingue la frontera entre la pupila y el iris, y aunque estas regiones generalmente no son concéntricas, ese mismo punto, por la relativa cercanía con el centro del iris, es un punto de partida eficaz para la búsqueda de las fronteras externas del iris.

Sabiendo esto se puede decir que la segmentación como proceso es la búsqueda del centro de la pupila y proceder a delimitar las fronteras interiores y exteriores del iris. Una vez lograda esta fase se debe entregar la región del iris a la siguiente fase, la codificación del iris.

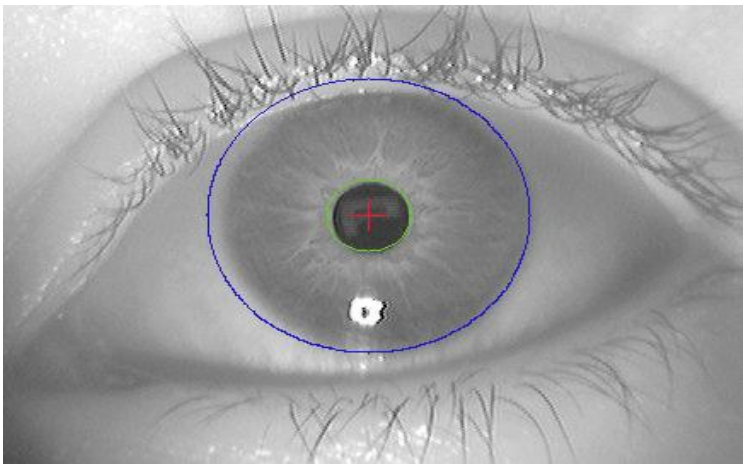


Figura 3: Imagen segmentada.

1.5 Normalizar

Después de tener una imagen segmentada y antes de comenzar la codificación de la misma es necesario pasar por la fase de normalización la cual se encarga de transforma la región circular del iris en una representación rectangular de tamaño fijo, para que de esta manera, diferentes iris de diferentes tamaños puedan ser comparados. El método más común para normalizar es el modelo “rubber sheet”, o “lámina

de goma”, ya que el proceso equivale a extraer la textura del iris y deformarla para que quede rectangular, como se muestra en la figura.

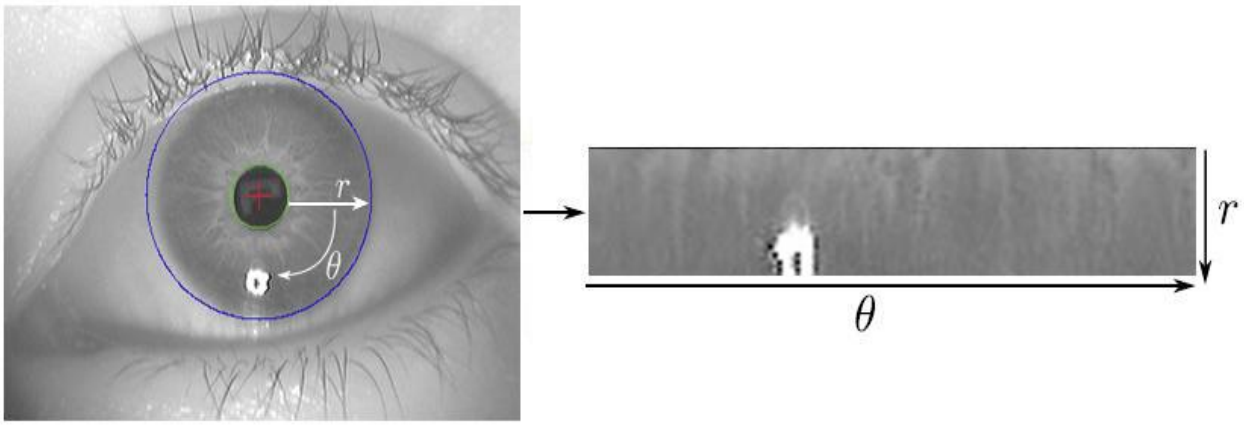


Figura 4: Imagen normalizada.

1.6 Codificar

La idea fundamental en esta etapa es generar una matriz binaria a partir de la imagen de entrada. Antes de realizar este paso la imagen debe ser filtrada para lograr un acercamiento de todas las intensidades a valores extremos. Básicamente la codificación comienza con el filtrado del segmento de iris con el objetivo de asociar los valores reales de la imagen a valores extremos, que facilitan la construcción de un iriscode.

1.6.1 Tipos de filtrados.

Filtros en el dominio de la frecuencia y espacio

Existen básicamente tres tipos distintos de filtros que pueden aplicarse:

- Filtro paso bajo: atenúa las frecuencias altas y mantiene sin variaciones las bajas. El resultado en el dominio espacial es equivalente al de un filtro de suavizado, donde las altas frecuencias que son

filtradas se corresponden con los cambios fuertes de intensidad. Consigue reducir el ruido suavizando las transiciones existentes.

- Filtro paso alto: atenúa las frecuencias bajas manteniendo invariables las frecuencias altas. Puesto que las altas frecuencias corresponden en las imágenes a cambios bruscos de densidad, este tipo de filtros es usado, porque entre otras ventajas, ofrece mejoras en la detección de bordes en el dominio espacial, ya que estos contienen gran cantidad de dichas frecuencias. Refuerza los contrastes que se encuentran en la imagen.

- Filtro paso banda: atenúa frecuencias muy altas o muy bajas manteniendo una banda de rango. medio. [2]

1.6.2 Métodos de codificación existentes.

La codificación del iris es probablemente el área que más se ha estudiado y sobre el que existen mayor cantidad de trabajos. Entre todos los métodos existentes, los más comunes y aquellos en los que se han obtenido mejores resultados son los que utilizan filtros de Gabor o Log-Gabor y los que utilizan wavelets para codificar la textura del iris. Se hará a continuación un repaso de los métodos más utilizados y los más interesantes desde el punto de vista de la originalidad y eficacia.

La codificación mediante wavelets fue propuesta por Boles y Boashash. Su sistema codifica anillos del iris, denominados círculos virtuales utilizando la representación por cruces de ceros de la transformada wavelet diádica del anillo. Ma también utiliza anillos del iris y la transformada wavelet para generar un código de iris. Sin embargo, en vez de utilizar los cruces de ceros para generar el código, codifica las posiciones de los máximos y mínimos locales consecutivos de la transformada wavelet en dos niveles. Lim en cambio, utiliza la wavelet bidimensional de Haar para realizar una descomposición en cuatro niveles, y genera un código sumamente compacto de 87 bits, utilizando una red neuronal[5]. El método de codificación

[5] Paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida.

por Laplaciano de Gaussianas fue introducido porWildes, consiste en generar representaciones de la imagen del iris en varias escalas. Cada escala es filtrada con un filtro cuasi-Gaussiano. Monro utiliza la transformada discreta del coseno (DCT) para codificar “parches” de la textura del iris. Cada parche se codifica utilizando la representación por cruce de ceros y luego es cuantizado en un código binario.

Daugman propone los filtros de Gabor bidimensionales como método de codificación del iris. Los filtros de Gabor son un tipo de filtro paso banda (filtro que deja pasar un determinado rango de frecuencias de una señal) que determina el rango de frecuencias utilizando una función Gaussiana. Debido a los buenos resultados obtenidos con la utilización de estos filtros, son los utilizados en este trabajo.

1.7 Comparación

La comparación es la etapa final del proceso. Lógicamente se comparan los datos de la entrada con los que ya existen y se valida la existencia de un usuario. Sin embargo, cuando se ejecuta la codificación raramente se obtiene una matriz igual a otra. Este fenómeno ocurre porque en condiciones naturales las imágenes obtenidas en la captura no son iguales. Aunque se ajuste un ambiente de iluminación y posicionamiento muy cercano al ideal, siempre hay diferencias. A veces también un mal posicionamiento puede provocar que el corte del iris no se realice en mismo lugar, lo que da lugar a corrimientos de los patrones distintivos. En fin, la solución a este inconveniente es la aplicación de un método sencillo y eficiente, el método de la distancia de Hamming. La distancia de Hamming se describe como:

$$\mathrm{HD} = \frac{\|(codeA \bigotimes codeB) \bigcap maskA \bigcap maskB\|}{\|maskA \bigcap maskB\|}$$

Asociada a cada código, existe una máscara de ruido marcando los puntos que fueron afectados por la interferencia de parpados y pestañas por lo que codeA, codeB se refieren a los códigos que se van a comparar y maskA, maskB sus respectivas máscaras de ruido. El resultado numérico de dicha fórmula es un número entre 0 y 1. Mientras más pequeña es la distancia el número es más cercano a 0 y hay más probabilidades de que las matrices hayan sido generadas por el mismo iris. Se considera que por encima

de 0.45 es un valor de distancia lo suficientemente grande como para afirmar que los iriscode no fueron generados por el mismo iris.

1.8 Conclusiones.

En este capítulo se determinaron las etapas en las que se divide el proceso de reconocimiento de un iris humano que luego serán las utilizadas en el desarrollo del componente propuesto. Además, se repasaron algunas de las técnicas de codificación más comunes en la literatura. La cantidad de trabajos realizados en este aspecto es muy grande, aunque la mayoría suele derivar de alguna u otra forma las técnicas descritas. Finalmente, se analizó la opción implementada.

Capítulo 2: Herramientas y tecnologías usadas.

2.1 Introducción al capítulo

En este capítulo se brinda una síntesis de las herramientas y tecnologías usadas en la construcción del sistema. Se caracterizara desde el sistema operativo usado, las herramientas de modelado, diseño y codificación, hasta los lenguajes de programación usados.

2.2 Sistema Operativo: GNU/Linux.

Debido a la gran cantidad de cálculo que requieren estos sistemas de autenticación se decidió utilizar GNU/Linux con su distribución de Ubuntu 10.04 como Sistema Operativo (SO) por ser muy eficiente en lo referido al procesamiento de información. Es multitarea, multiusuario, multiplataforma y multiprocesador y en las plataformas Intel se ejecuta en modo protegido; protege la memoria para que un programa no pueda hacer caer el resto del sistema; carga solo las partes de un programa que se usan lo que proporciona una mayor eficiencia; comparte la memoria entre programas aumentando la velocidad y disminuyendo el uso de memoria; utiliza toda la memoria libre para caché; permite usar bibliotecas enlazadas tanto estática como dinámicamente, además de ser un SO libre por el que no se debe pagar una patente y se distribuye con su código fuente lo que permite adaptar el sistema a las necesidades de cada desarrollador.

2.3 Lenguaje de Modelado: UML.

Se decidió utilizar el Lenguaje Unificado de Modelado (UML) por ser el lenguaje de modelado de sistemas de software más conocido y utilizado tanto en nuestra Universidad como a nivel mundial. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Está compuesto por elementos gráficos que se combinan para conformar diagramas. El desarrollo de sistemas con UML incluye actividades especificas, cada una de ellas contiene a su vez otras sub-actividades las que sirven como una guía de cómo deben ser las actividades desarrolladas y secuenciadas con el fin de obtener

sistemas exitosos. UML no tiene propietario y está basado en el común acuerdo de la comunidad informática.

2.4 Metodología de Desarrollo de Software: RUP.

De las metodologías que utilizan UML como lenguaje de modelado se decidió usar RUP (Rational Unified Process, Proceso Unificado de Racional) por ser con la que estamos más familiarizados, además de que provee un entorno de proceso de desarrollo configurable, basado en estándares; permite tener claro y accesible el proceso de desarrollo que se sigue; puede ser configurado de acuerdo con las necesidades de la organización y del proyecto; provee a cada participante con la parte del proceso que le compete directamente. RUP se caracteriza por ser:

- Dirigido por casos de uso: los casos de uso son los artefactos primarios para establecer el comportamiento deseado por el sistema.
- Centrado en la Arquitectura: la arquitectura es utilizada para conceptualizar, construir, administrar y evolucionar el sistema en desarrollo.
- Iterativo e incremental: maneja una serie de entregas ejecutables; integra continuamente la arquitectura para producir nuevas versiones mejoradas.

Las principales disciplinas o ciclos de esta metodología son:

Disciplinas	Descripción
Modelado del negocio	Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización. Propone comprender los problemas de la organización e identificar posibles mejoras, evaluar el impacto del cambio introducido con la automatización, asegurar que todos los miembros tienen una misma visión de la organización.

Requerimientos	Esta disciplina se encarga de convertir las peticiones de los clientes en un conjunto de requerimientos de software, que definan el alcance y lo que debe hacer el producto a ser construido. Propone mantener un acuerdo a los interesados de lo que el sistema debe hacer, provee a los desarrolladores de una mejor visión de los requerimientos del sistema, define la frontera del sistema, crea las bases para la planificación y estimación.
Análisis y Diseño	Esta disciplina define como transformar artefactos resultantes del flujo de requerimientos de software en especificaciones del diseño del proyecto a desarrollar, en él se hace evolucionar la arquitectura de modo que se adapte al entorno del usuario final.
Implementación	Define como desarrollar, organizar realizar pruebas de unidad e integración de todos los componentes implementados basado en las especificaciones del sistema.
Pruebas	Este flujo se centra en encontrar y documentar los defectos en la calidad del software, validar y probar todos los supuestos hechos durante el diseño, verificar que el producto se desempeña como fue diseñado y cubre todos los requisitos pactados durante el flujo de “Captura de Requerimientos”.
Instalación	Se encarga de garantizar que el producto está disponible para los usuarios en su ambiente, se realizan actividades como empaque, instalación, asistencia a usuarios.
Administración del proyecto	Se centra los aspectos esenciales del desarrollo iterativo como son la planificación, captura de riesgos, seguimiento del proceso de desarrollo

	de software y aplicación de métricas. Las restantes serán utilizadas durante el proceso de administración.
Administración de configuración y cambios	Se encarga de capturar y sincronizar la evolución de todos los productos generados, introduce el uso de herramientas que faciliten el trabajo colaborativo en el equipo.
Ambiente	Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Tabla 1: Disciplinas de RUP.

RUP divide el proceso de desarrollo en ciclos teniendo un producto final al concluir cada ciclo.

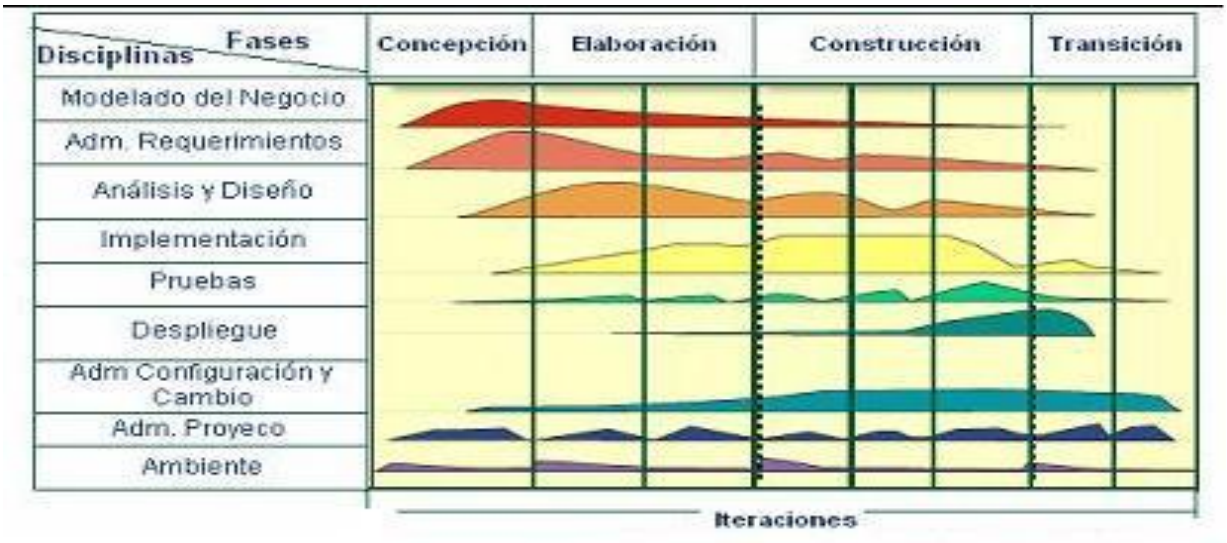


Figura 5: Disciplinas y fases de RUP.

Cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- Conceptualización (Concepción o Inicio): Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

- Elaboración: Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

- Construcción: Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios entregables del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.

- Transición: El entregable ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

2.5 Lenguaje de Programación: C++.

El uso de C++ como lenguaje para el desarrollo de este algoritmo viene dado por su robustez y eficiencia a pesar de sus más de 20 años. Además, su increíble versatilidad que permite programar desde el software más simple a los programas más complicados como incluso sistemas operativos. Tiene la ventaja de ser portable, lo que quiere decir que un programa con el código escrito en C++ se podrá compilar en cualquier sistema operativo sin necesidad de cambiar mucho el código fuente.

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.

2.6 Biblioteca: OpenCV.

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD[6], que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

Open CV es multiplataforma, existiendo versiones para Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos, calibración de cámaras, visión estéreo y visión robótica.

Como metas el proyecto pretende proveer un "Tool-Kit" o Marco de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo. Open CV puede además utilizar el sistema de las Primitivas de Rendimiento Integradas de Intel, que es un conjunto de rutinas de bajo nivel específicas para procesadores Intel.

2.7 IDE: Eclipse.

Un entorno de desarrollo integrado (IDE por sus siglas en inglés) es un programa compuesto por una serie

[6] Licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Es una licencia de software libre permisiva como la licencia de OpenSSL o la MIT License.

de herramientas para un programador. Puede estar dedicada a un lenguaje de programación en específico o bien puede utilizarse para varios. Entre las herramientas más comunes que poseen los IDEs están: un editor de código, un compilador, un intérprete, un depurador y un constructor de interfaces gráficas de usuario.

En la selección del IDE para desarrollar influyeron varios requisitos: se necesitaba que este permitiera la programación en C++, que contara con herramientas integradas para la gestión de la configuración tanto del código como de la documentación, que presentara un buen completamiento de código y que brindara facilidades en la vinculación de bibliotecas y en la configuración del compilador.

2.8 CASE: Visual Paradigm.

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Dentro de las herramientas CASE se utilizó el Visual Paradim, las principales características que se necesitaban y que en específico cumple esta herramienta son que: es multiplataforma y modela todos flujos de trabajo de RUP. Visual Paradigm para UML es una Herramienta Case Cruzado de Ciclo de Vida, lo cual significa que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue, incluye también actividades como la gestión de proyectos y la estimación. El software de modelado UML, ayuda a una más rápida construcción de aplicaciones de mejor calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Incluye además importación desde Rational Rose, exportación/importación XMI, generador de informes, editor de figuras, integración con MS Visio, plug-in, integración IDE con Visual Studio, IntelliJ IDEA, Eclipse, NetBeans y otros. Apoya

un conjunto de lenguajes tanto en la generación del código como en la Ingeniería Inversa por ejemplo Java, C + +, CORBA IDL, PHP, XML Schema, Ada y Python.

2.9 Generador de documentación: Doxygen.

El paquete Doxygen contiene un sistema de documentación para C++, C, Java, Objective-C, Corba IDL y, en parte, PHP, C# y D. Es útil para generar documentación HTML y/o un manual de referencia a partir de un grupo de ficheros fuente documentados. También soporta generación de salida en RTF, PostScript, PDF con hiperlinks, HTML comprimido y páginas de manual Unix. La documentación se extrae directamente de las fuentes, lo que hace mucho más fácil mantener consistente la documentación con el código fuente.

2.10 Conclusiones.

En este capítulo se hace un resumen de las tecnologías y herramientas escogidas en el desarrollo del trabajo, haciendo énfasis en la razón por la que fueron seleccionadas, teniendo en cuenta como premisa fundamental la utilización del Software Libre y las políticas del Centro de Producción.

Capítulo 3: Descripción de la solución propuesta.

3.1 Introducción al capítulo

En el presente capítulo se lleva a cabo el proceso de modelación del sistema y se ofrece una descripción de las características del mismo. Para ello se especifican los requerimientos funcionales y no funcionales que debe tener la aplicación, se identifican los actores, se describen los principales casos de usos y se construyen varios diagramas que posibiliten un mejor entendimiento del sistema.

3.2 Especificación de los requisitos de software

Los requisitos se clasifican en funcionales y no funcionales. Los funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los no funcionales son propiedades o cualidades que hacen al producto atractivo, usable, rápido y confiable.

3.2.1 Requisitos funcionales del sistema.

- Llevar el anillo del iris a una imagen rectangular.
- Crear el iriscode asociado a dicha imagen.

3.2.2 Requisitos no funcionales del sistema.

- Software
  - Por políticas del centro productivo de Informática Industrial se usó GNU/Linux.
- Requerimientos de hardware.
  - Se recomienda usar procesadores INTEL ya que OpenCV está optimizada para este tipo de procesador.

- Requerimientos en el diseño y la implementación.
    - La solución cuenta con un conjunto de patrones que permite la reutilización del código así como facilidad en la actualización del mismo.
    - El código cumple con los estándares de codificación establecidos por el centro productivo de Informática Industrial.
    - La biblioteca OpenCV garantiza la implementación eficiente de algoritmos necesarios para la construcción de este componente.
- Requerimientos de Soporte.
    - Se ofrece servicios de mantenimiento y actualización.
- Requerimientos de Usabilidad.
    - La biblioteca podrá ser usada por cualquier persona que posea conocimientos básicos en el manejo de la computadora, programación y el sistema operativo GNU/Linux.
- Requerimiento asociado al Licenciamiento.
    - Se debe garantizar que el sistema de desarrolle bajo los principios del software libre y por tanto cualquier componente de software que se utilice también lo debe ser.

3.3 Modelación del Sistema.

La modelación de un sistema, es una técnica cognitiva que permite abstraer la situación real a una ideal, desechando el conocimiento innecesario y aprovechando el esencial. De esta forma, el trabajo se facilita, logrando una mayor concentración en el objetivo u objetivos finales.

3.3.1 Casos de Uso.

Un caso de uso está determinado por un flujo de actividades que el sistema debe ser capaz de cumplir y que conceptualmente están relacionadas. En la gran mayoría de las ocasiones un caso de uso está condicionado por uno o más requerimientos funcionales. Dadas las circunstancias se puede asumir que los requisitos anteriores se pueden unir en los casos de usos *Normalizar Imagen y Codificar imagen.*

3.3.2 Actores.

Un actor representa una persona o un sistema externo que interactúa con este sistema. La codificación de la imagen podrá ser usada por un programa, aplicación de software. Este grupo de elementos puede definirse como un actor *Aplicación.*

3.3.3 Diagrama de casos de uso del sistema

Partiendo de la base de que cada requisito funcional se convierte en un caso de uso del sistema es que se definen los actores y casos de uso a partir de la captura de los requisitos.

A continuación se modela la relación existente entre los actores y casos de uso definidos. En este diagrama se pueden observar las responsabilidades que tienen los actores sobre los casos de uso, así como la relación entre ellos.

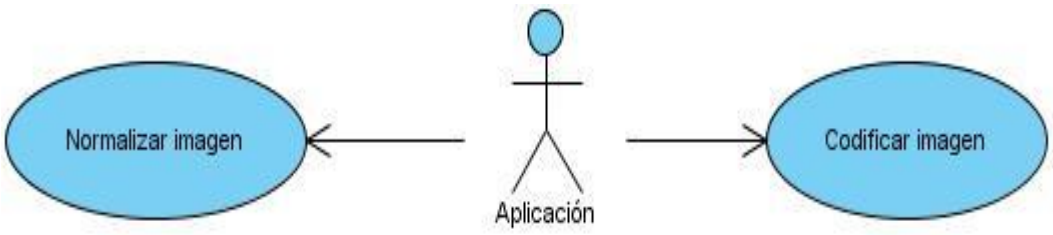


Figura 6: Diagrama de Casos de Uso del Sistema.

3.3.4 Descripción de Casos de Uso del Sistema.

Caso de uso	
CU-1	Normalizar imagen.
Objetivo	Obtener una imagen rectangular de dimensiones fijas con la textura del iris, que posibilite el comienzo de la etapa de codificación.
Actores	Aplicación.
Resumen	La normalización de la imagen constituye la creación de una imagen rectangular de dimensiones fijas a partir de una imagen segmentada.
Complejidad	Alta.
Prioridad	Critico.
Precondiciones	Debe existir algún algoritmo o método que proporcione la imagen segmentada.
Poscondiciones	Si el usuario pertenece al sistema se le debe permitir la entrada. De lo contrario el sistema debe avisarle de que no ha logrado identificarlo.
Flujo básico.	
Aplicación.	Biblioteca.
1. Pide la segmentación de la imagen de un ojo del usuario.	2. Brinda una imagen segmentada.
3. Pide la conversión del anillo del iris obtenido a una imagen rectangular.	4. Convierte la región del iris en una imagen rectangular.
	7. Notifica la normalización del iris.
Flujos alternos.	
Relaciones.	

Tabla 2: Descripción del Caso de Uso Normalizar Imagen.

Caso de uso	
CU-2	Codificar imagen.
Objetivo	Obtener un iriscode a partir de los patrones distintivos del iris que posibilite el comienzo de la etapa de comparación.
Actores	Aplicación.
Resumen	La codificación de la imagen constituye la generación de un código binario denominado iriscode a partir de una imagen normalizada.
Complejidad	Alta.
Prioridad	Critico.
Precondiciones	Debe existir algún algoritmo o método que proporcione la imagen normalizada.
Poscondiciones	Si el usuario pertenece al sistema se le debe permitir la entrada. De lo contrario el sistema debe avisarle de que no ha logrado identificarlo.
Flujo básico.	
Aplicación.	Biblioteca.
1. Pide la normalización de una imagen segmentada.	2. Brinda una imagen normalizada.
3. Pide el iriscode asociado a la imagen.	4. Genera el iriscode.
	5. Notifica la codificación del iris.
Flujos alternos.	
Relaciones.	

Tabla 3: Descripción del Caso de Uso Codificar Imagen.

3.3.5 Análisis y diseño.

A continuación se desarrollará la modelación de las diferentes clases utilizadas en la aplicación así como sus relaciones e interacción.

3.3.6 Diagrama de componentes.

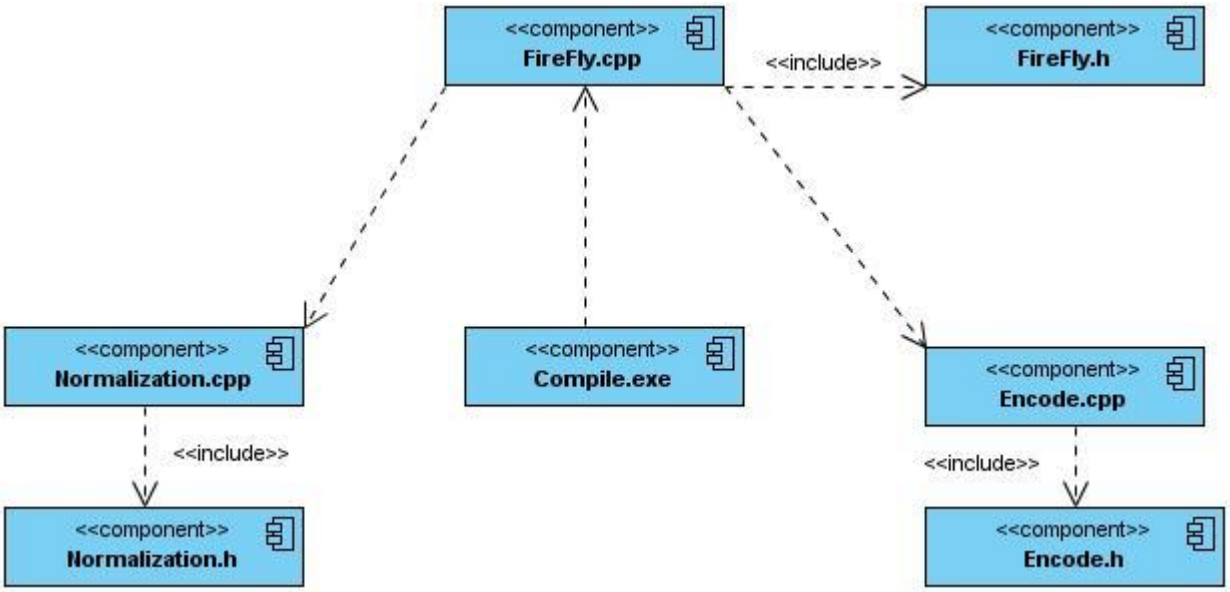


Figura 7: Diagrama de Componentes.

3.3.7 Diagramas de interacción.

Los diagramas de interacción reflejan el flujo de mensajes que existen entre las diferentes interfaces de un sistema. Se dividen en dos tipos: diagramas de secuencia, que se encargan de visualizar la vida de los mensajes y sus efectos y los diagramas de colaboración, donde se representa la relación estática y direccional de los mensajes por orden de ocurrencia.

3.3.8 Codificar Imagen (Diagrama de secuencia).

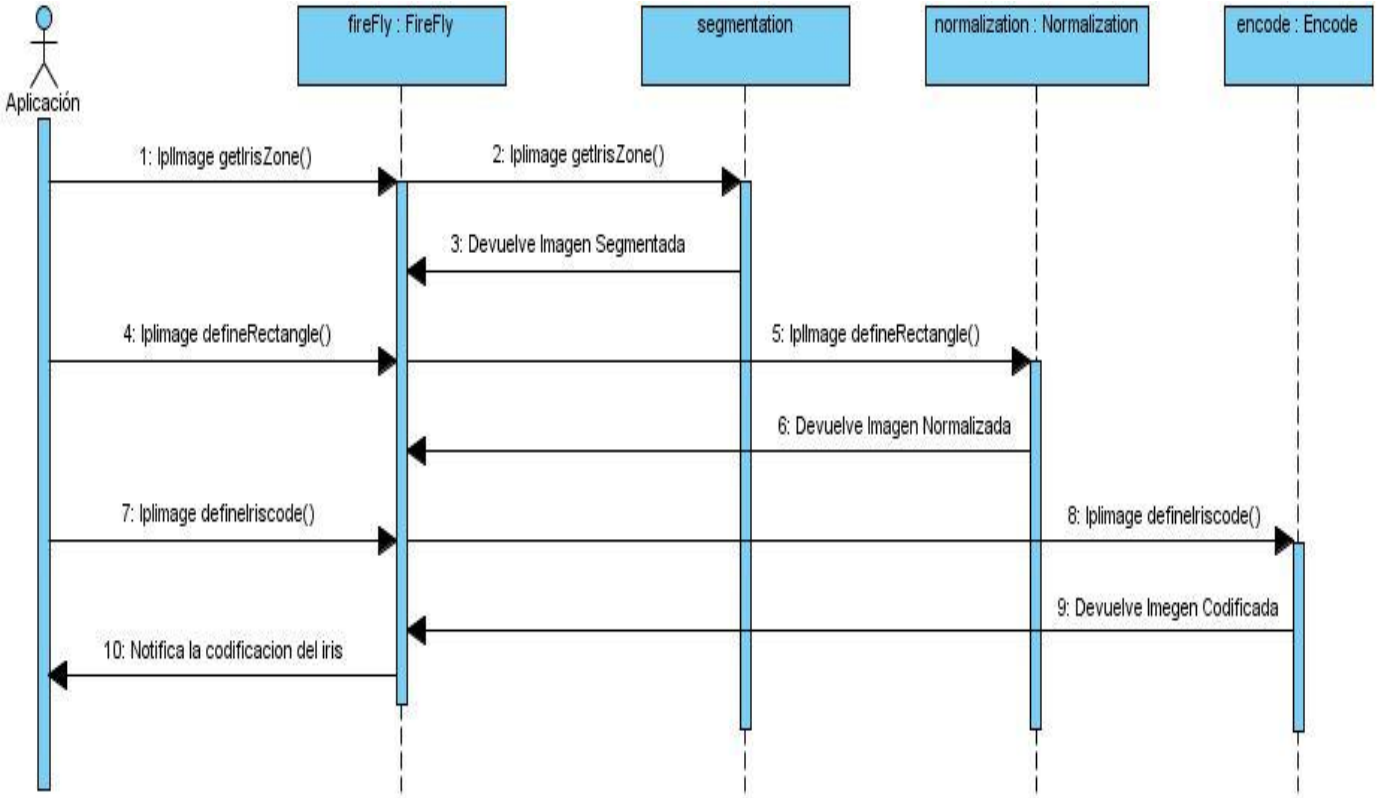


Figura 8: Diagrama de Secuencia.

3.3.9 Diagrama de despliegue.

Figura 9: Diagrama de Despliegue.

3.3.10 Diseño de clases.

El diseño de clases se encapsula los principales conceptos del sistema en clases que constituyen la base para la futura implementación.

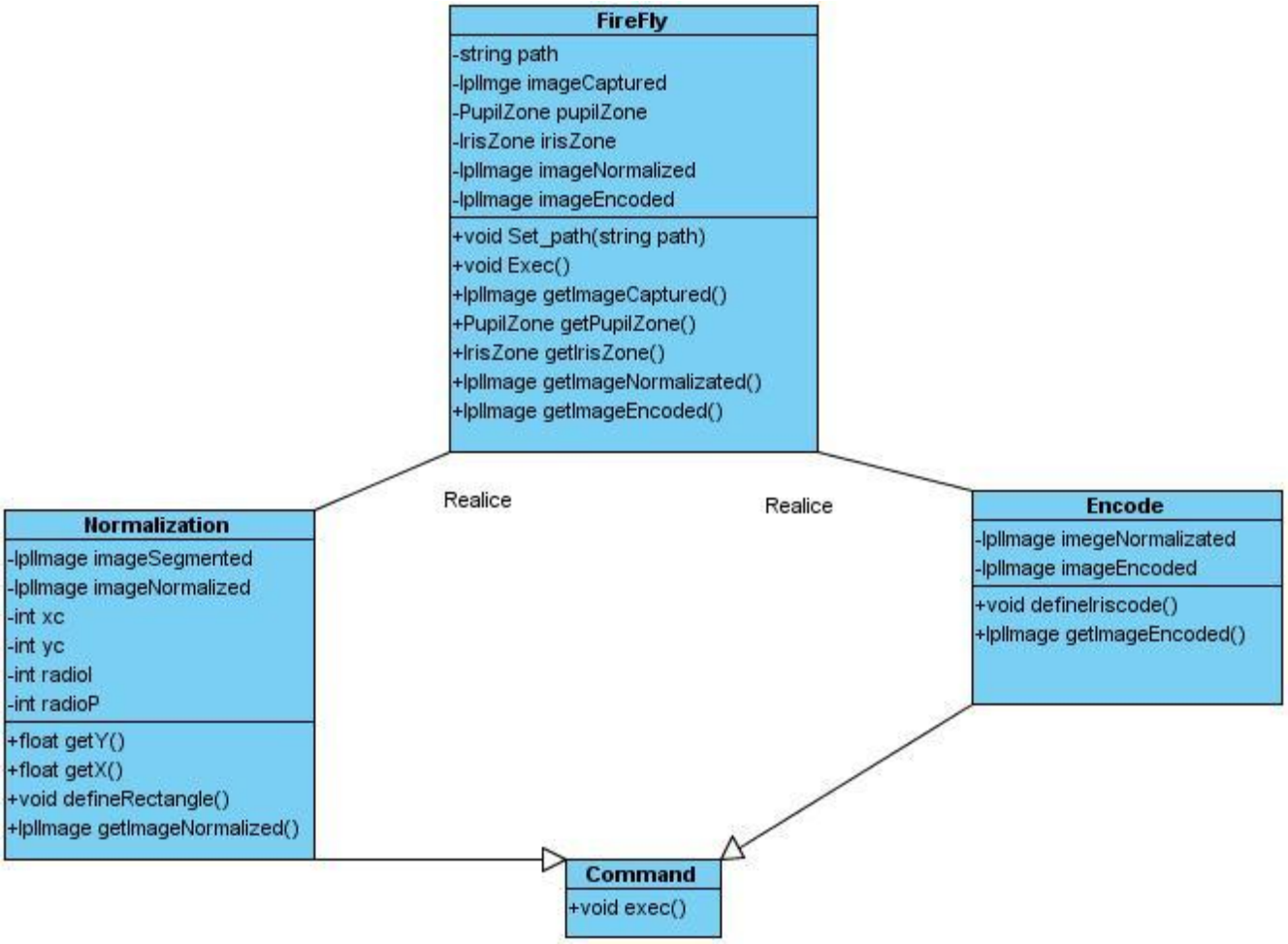


Figura 10: Diagrama de Clases.

3.3.11 Descripción de las principales clases.

A continuación se realizará la descripción de las principales clases, detallando sus funcionalidades y atributos que las componen.

Nombre: FireFly	
Tipo de clase: Interfaz.	
Atributo	**Tipo**
path	string
imageCaptured	IplImage
pupilZone	PupilZone
irisZone	IrisZone
imageNormalized	IplImage
imageEncoded	IplImage
Nombre:	**FireFly (string path)**
Descripción:	Constructor por defecto de la clase con la dirección de la imagen.
Nombre:	**~ FireFly ()**
Descripción:	Destructor de la clase.
Nombre:	**Set_path(string path)**

Descripción:	Modifica la dirección donde se encuentra la imagen.
Nombre:	**exec()**
Descripción:	Se ejecuta la aplicación obteniendo como resultado el iriscode de la imagen segmentada.
Nombre:	**getPupilZone()**
Descripción:	Devuelve la zona de la pupila.
Nombre:	**getIrisZone()**
Descripción:	Devuelve la zona del iris detectada.
Nombre:	**getImageNormalizated ()**
Descripción:	Devuelve una imagen con el iris normalizado o sea una barra rectangular con la textura del iris segmentado.
Nombre:	**getImageEncoded ()**
Descripción:	Devuelve una imagen con el iriscode generado.

Tabla 4: Descripción de la clase FireFly.

Nombre: Normalization	
Tipo de clase: Entidad	
Atributo	**Tipo**
imageSegmented	IplImage
imageNormalized	IplImage
xc	int
yc	int
radioI	int
radioP	int

Nombre:	**Normalization (IplImage * imageSegmented,int xc,int yc,int radioI,int radioP)**
Descripción:	Constructor por defecto de la clase con la imagen obtenida en el proceso de segmentación, así como con los valores de las coordenadas del centro de la pupila y el radio del iris y la pupila.
Nombre:	**~ Normalization ()**
Descripción:	Destructor de la clase
Nombre:	**float getY(int angulo,int radio)**
Descripción:	Devuelve el valor de y en coordenadas cartesianas de un punto dado el valor de

	su radio y ángulo en coordenadas polares.
Nombre:	**float getX(int angulo,int radio)**
Descripción:	Devuelve el valor de x en coordenadas cartesianas de un punto dado el valor de su radio y ángulo en coordenadas polares.
Nombre:	**defineRectangle()**
Descripción:	Devuelve el iris normalizado o sea una barra rectangular con la textura del iris segmentado.
Nombre:	**getImageNormalized()**
Descripción:	Devuelve una imagen con la imagen del iris normalizado.
Nombre:	**exec()**
Descripción:	Ejecuta el proceso de normalización completo obteniendo la imagen del iris normalizado en una barra rectangular.

Tabla 5: Descripción de la clase Normalization.

Nombre: Encode
Tipo de clase: Entidad

Atributo	**Tipo**
imegeNormalizated	IplImage
imageEncoded	IplImage

Nombre:	**Encode(IplImage* imegeNormalizated)**
Descripción:	Constructor por defecto de la clase con la imagen obtenida en el proceso de normalización.
Nombre:	**~Encode()**
Descripción:	Destructor de la clase
Nombre:	**defineIriscode()**
Descripción:	Devuelve el iriscode, o sea, una estructura binaria obtenida a partir de patrones del iris.
Nombre:	**getImageEncoded()**
Descripción:	Devuelve la imagen del iriscode obtenido.
Nombre:	**exec()**
Descripción:	Ejecuta el proceso de codificación completo obteniendo la imagen del iriscode.

Tabla 6: Descripción de la clase Encode.

3.4 Patrones de diseño.

Un patrón de diseño es un mecanismo de la ingeniería de software que brinda robustez y flexibilidad a un conjunto de clases. Constituye una solución predeterminada a un problema de diseño común para las aplicaciones de software.

En la elaboración de este trabajo se usaron dos patrones. El Primero fue el *Facade* o *Fachada.* Este patrón pertenece al grupo de los patrones estructurales y provee una interfaz que posibilita a que cualquier aplicación use las funciones que dicha interfaz es capaz de brindarle. La clase Firefly, actúa como una interfaz y provee las funcionalidades de las clases Normalization y Encode. El Segundo patrón que se usó fue el *Command* o *Comando.* Un comando es un patrón de comportamiento y encapsula una función específica, permitiendo que esta función se ejecute en cualquier comando sin que importe el comportamiento del mismo. La clase Command define una función *execute* permitiendo que cada una de sus clases derivadas actúe como un comando.

El sistema de clases es flexible. Está diseñado para crecer sin afectar a los componentes que ya existen. Las clases comparten sus funcionalidades con la interfaz para que estas puedan ser exportadas. Al mismo tiempo, se comportan como comandos. Este funcionamiento será el mismo para los nuevos paquetes que se decidan agregar.

3.5 Arquitectura del sistema.

La Arquitectura de Software es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. [3]

En este trabajo se aplica una arquitectura en capas. Presenta una capa de presentación constituida por la interfaz, la clase FireFly. Una capa de lógica donde se encuentran las clases Normalization y Encode, que manejan el desarrollo del algoritmo. Y para finalizar una capa de acceso a datos representada por la clase Capture donde se accede a las imágenes.

3.6 Conclusiones.

En este capítulo se abordó un enfoque de la solución desde el diseño. Se puede concluir que los requerimientos de la aplicación quedan resueltos con el conjunto de clases propuesto. Sus descripciones son claras, la arquitectura es sencilla y robusta y patrón de diseño permite el crecimiento del componente o su decrecimiento sin afectar los paquetes que persisten.

Capítulo 4: Desarrollo y pruebas.

4.1 Introducción.

Este capítulo trata sobre el estándar de codificación y documentación y la implementación del algoritmo de normalización, así como da la transformada de Fourier y el filtro de Gabor utilizados para la codificación. También se explicarán diferentes pruebas de sistemas realizadas al software.

4.2 Estándar de codificación

4.2.1 Nombres

- Los nombres de las clases son sustantivos singulares.
- Los nombres de clases y objetos deben reflejar qué hacen y no cómo lo hacen.
- Escoger nombres lo suficientemente largos que expresen correctamente el sentido de lo que se quiere, pero evitando manejar nombres que dificulten la labor de implementación.
- Evitar nombre que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).
- Evitar redundancia no repitiendo nombre de clases en sus elementos.
- Dado que los nombres generalmente son el producto de concatenar varias palabras, se debe emplear la primera palabra en minúscula, mayúscula para denotar la letra de inicio de cada una de las palabras restantes por las que esté formado y minúscula para las letras intermedias en el caso de los nombres de métodos y funciones. Para el caso de los nombres de variables y atributos debe aplicarse la misma convención.
- Las variables booleanas deben contener la palabra *is* en su nombre.

- Los nombres de constantes deben contener solo letras mayúsculas.
- Minimizar el uso de abreviaturas. En caso de ser requeridas, se debe ser consistente en su uso y cada abreviatura debe significar solo una cosa.
- Los nombres de los métodos son frases que incluyen verbos.
- Los nombres de los atributos y parámetros son frases con sustantivos.

4.2.2 Codificación

- Se establece un tamaño de indentación [7]estándar de tres espacios, sin tabulaciones.
- Alinear secciones del código.
- Alinear verticalmente llaves de apertura y cierre, la llave de apertura.
- Usar espacios antes y después de los operadores que el lenguaje de programación permita.
- Emplear líneas en blanco para organizar el código, permitiendo crear párrafos de código para una mejor lectura.
- Evitar colocar más de una sentencia por línea.
- Emplear constantes en sustitución de números o cadenas de caracteres literales.
- Minimizar el alcance de las variables para evitar confusión y facilitar el mantenimiento.
- Emplear cada variable y rutina solo para un propósito.
- Evitar el uso de variables públicas, sustituirlas por variables privadas y métodos que provean el

[7] Mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente. La indentación se utiliza para mejorar la legibilidad del código fuente por parte de los programadores.

valor de tal variable, para mantener el encapsulamiento.

- Emplear i, j, k, l, m, p, q, r para contadores en ciclos.
- Mantener la modularidad del código bajo el criterio de la lógica que encierra, no exagerar la modularidad.
- Emplear correctamente los tipos de ciclos: si es al menos una vez usar *do-while*, si es ninguna o más veces usar *while-do*, y si se conoce el número exacto de ciclos usar *for*.
- Inicializar todas las variables.
- Emplear líneas en blanco para separar pasos lógicos (Ej.: declaraciones, lazos).

4.3 Estándar de documentación

Los formatos brindados por la herramienta Doxygen empleada para la documentación del sistema se explican a continuación:

4.3.1 Estilo de bloques de documentación.

Se adopta el estilo de bloques de documentación JavaDoc, el cual consiste en un bloque de comentario de estilo C, los asteriscos que se encuentran en la línea de la mitad son opcionales. Ejemplo:

```
/**
 * Texto
 */
```

4.3.2 Descripción breve con el comando \brief o @brief.

Para hacer una descripción breve se adopta el uso del comando \brief o @brief en el bloque de comentarios ya descrito.

La acción de este comando termina al final de un párrafo, de tal manera que la descripción detallada sigue después de una línea vacía, tal como lo muestra el ejemplo:

```
/**
 * @brief descripción breve.
 * Continuación de la descripción breve.
 *
 * La descripción detallada comienza aquí, nótese
 * que se debe dejar una línea en blanco para lograr
 * tener las dos descripciones (breve y detallada)
 */
```

Nota: Si no se utiliza el comando @brief Doxygen tomará la descripción hecha en el bloque como una descripción detallada y no habrá descripción breve.

4.3.3 Descripción de argumentos y métodos.

A la hora de hacer una descripción de los argumentos de métodos y funciones ésta se hará en línea, es decir, luego de la declaración de cada uno de los argumentos se da una breve descripción de cada uno de ellos, en el siguiente ejemplo se muestra el formato a utilizar:

```
int multFunction (     int c , /**<Primer operando de la operación */
int d , /**<Segundo operando de la operación */
int e /**<Tercer operando de la operación */ );
```

4.3.4 Documentación de tipos de datos.

Para describir la función y los elementos que componen los tipos de datos definidos se pueden utilizar los formatos especificados en los apartados anteriores. A continuación se muestra un ejemplo:

```
/**
 * @brief Enumerado de los tipos de datos admitidos
 *
 * Descripción más detallada de la función de este tipo de dato.
 */
enum DataTypes{ INTEGER, /**<Puede ser un valor de tipo entero */
DOUBLE, /**<Puede ser un valor de tipo doublé */
STRING /**<Puede ser un valor de tipo string */ };
```

Observamos que se genera una descripción breve seguida de una descripción más detallada para la función, luego se explica brevemente el valor de retorno de la misma y la descripción de los parámetros usando el formato de documentación en línea.

4.3.5 Comandos @author y @date.

Es importante que se especifique el nombre del autor y la fecha de creación de cualquier estructura en un código, para ello se utilizan los comandos @author y @date para el nombre del autor y la fecha respectivamente, en el siguiente ejemplo se puede ver la acción de estos comandos:

```
/**@brief Descripción breve
```

*

* Aquí comienza la descripción detallada

* @author Rosalina Puerto rpuerto@estudiantes.uci.cu

* @date 13-04-2010

*/

4.3.6 Comando @see.

Existe otro comando útil que permite hacer referencias a otras clases o métodos cuando esto sea necesario, es importante usar este comando en la documentación a la hora de implantar los métodos o funciones propias de alguna clase, este comando es @see y su uso se muestra en el siguiente ejemplo:

/**

* Constructor de la clase

* @see Test::Test()

*/

Test1();

Esto generará en la documentación para el constructor de la clase de prueba Test1 una referencia hacia la documentación existente para el constructor de la clase de prueba Test.

Nota: Si se quiere hacer una referencia desde cualquier estructura hacia la documentación completa de una clase ya documentada, solo se debe utilizar el comando @see seguido del nombre de la clase de esta forma:

```
/**
* Constructor de la clase
* @see Clase
*/
Test1();
```

4.4 Implementación del algoritmo para la normalización del iris.

Como se explica en capítulos anteriores, una etapa intermedia entre la etapa de segmentación y codificación es la etapa de normalización, donde se convierte el iris en una imagen rectangular de dimensiones fijas que sirve para resolver problemas como la dilatación o contracción de la pupila, así como posibles ruidos ocurridos en la etapa de captura.

Para solucionar esta etapa se utiliza en este trabajo un algoritmo que se basa en la transformación de la imagen del iris segmentado de coordenadas cartesianas a coordenadas polares y manteniendo dimensiones fijas.

En primer lugar, y con el propósito de no tomar datos dentro de la zona correspondiente a la pupila, el radio de la primera circunferencia virtual a transformar deberá ser como mínimo igual al radio de la pupila más un margen de seguridad, con el cual se asegura que toda la zona de la pupila quede en la parte interior de la circunferencia y no interfiera en ningún caso en los datos que se extraen para determinar la señal procedente del análisis del iris del usuario. Para determinar esta zona en este trabajo se ha definido una distancia fija d=8 pixeles. Luego, se define el radio inicial de la siguiente manera:

Radio inicial=Radio pupila+8.

Para el desarrollo del algoritmo se toman 32 circunferencias virtuales concéntricas al iris a partir del radio inicial descrito anteriormente, cada circunferencia incrementa un valor Δr respecto a la anterior, que

vendrá definido por la diferencia existente entre el radio del iris y el radio inicial. De este modo para poder tomar 32 circunferencias virtuales y admitiendo un margen de seguridad sobre el radio del iris se define el incremento radial como: Δr= (Radio del iris-Radio inicial)/32.

Del mismo modo, se toman 180 ángulos distintos, en este sentido, al trabajar con algoritmos protegidos frentes a la rotación comenzaremos en el mismo ángulo φ inicial, y los ángulos elegidos en cada caso serán tales que $\Delta\varphi=2\pi/180$; sin embargo, solo se reconstruye el 75% de la información del iris eliminando los ángulos correspondientes al intervalo [π/4; 3π/4] logrando de esta manera un ruido casi nulo por intromisión de parpados y pestañas como se puede apreciar en la figura 11.

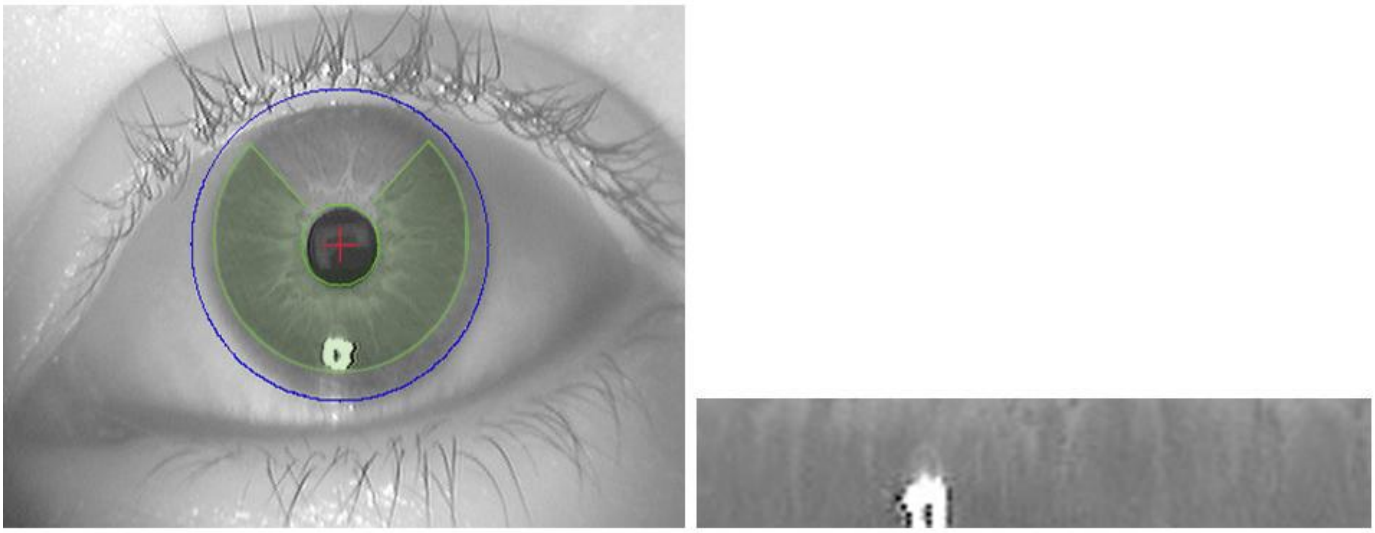


Figura 11: (a) Región de interés para el algoritmo de codificación. (b) Región de interés normalizada.

Una vez solucionado el problema de fijar las dimensiones del rectángulo bastaría reconstruir la nueva

imagen a partir de la imagen segmentada para lo cual se transforman las coordenadas cartesianas a coordenadas polares como se describe a continuación:

$J(x,y)= (x0+ r\cos\varphi, y0+ r\,\mathrm{sen}\varphi)$.

Donde (x_0, y0) son las coordenadas del centro del iris y r es el radio en cuestión, ver figura 12.

Figura 12: Representación del proceso de normalización.

4.5 Implementación de la transformada de Fourier y filtro de Gabor para la codificación.

La codificación es el objetivo central de este trabajo. La idea fundamental en esta etapa es generar una matriz binaria a partir de la imagen de entrada. Antes de realizar este paso a la imagen se debe aplicar una transformada para trabajar en el espacio de frecuencia y una vez filtrada lograr un acercamiento de todas las intensidades a valores extremos.

4.5.1 Filtrado en el dominio de la frecuencia

Los filtros de frecuencia procesan una imagen trabajando sobre el dominio de la frecuencia en la Transformada de Fourier de la imagen. Para ello, ésta se modifica siguiendo el Teorema de la Convolución correspondiente:

1. Se aplica la Transformada de Fourier,

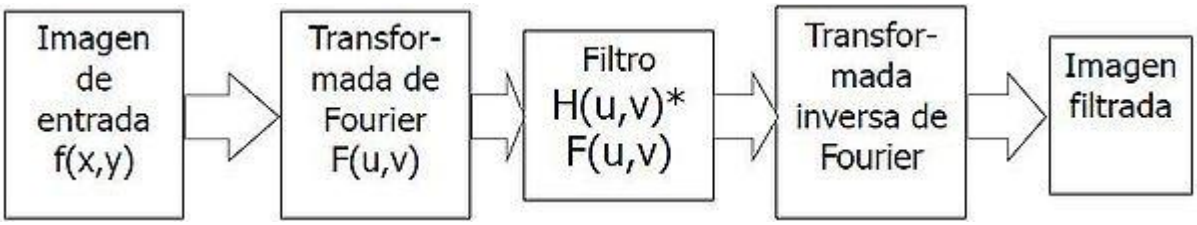
2. Se multiplica posteriormente por la función del filtro que ha sido escogido,

3. Para concluir re-transformándola al dominio espacial empleando la Transformada Inversa de Fourier.[2]

Teorema de la Convolución (frecuencia): ***G*(*u*,*v*) = *F*(*u*,*v*) * *H*(*u*,*v*)**

F(u,v): transformada de Fourier de la imagen original

H(u,v): filtro atenuador de frecuencias



4.5.2 Transformada de Fourier.

La transformada de Fourier es una técnica que consiste en cambiar la representación de los datos del plano espacial al de frecuencia. Cuando hablamos del dominio frecuencial en el procesamiento de imágenes, se hace referencia a una representación de la variación espacial de los píxeles de la imagen. Cabe destacar que la representación de la transformada de Fourier es otra señal bidimensional, en este caso compleja, y por lo tanto está compuesta de una parte real y otra imaginaria.
Cuando se menciona altas frecuencias espaciales se hace referencia a la ocurrencia de cambios rápidos en los niveles de gris entre los píxeles cuando se recorre la imagen espacialmente. Como por ejemplo, los que ocurren en los bordes, las líneas y cierto tipo de ruido en las imágenes. Por el contrario, las bajas frecuencias son producidas por cambios leves en el brillo de una imagen.

Aunque existen varias opciones para realizar este proceso se escogió aplicar la transformada de Fourier, la cual es una operación matemática que transforma una señal de *dominio de tiempo* a *dominio de*

frecuencia y viceversa. En el dominio de frecuencia, una señal es expresada con respecto a la frecuencia. La imagen de entrada, se traslada a un espacio de dominio de frecuencia de la siguiente forma:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}.$$

Donde M y N son las dimensiones de la imagen, u y v se obtienen como combinación de todos los pixeles de la imagen origen.

4.5.3 Filtro de Gabor.

Después de tener la imagen en dominio de frecuencia el siguiente paso es aplicar un filtro para acercar las intensidades a valores extremos, para lo cual se utilizan los filtros de Gabor o sea un filtro lineal cuya respuesta de impulso es una función sinusoidal multiplicada por una función gaussiana.

En el dominio de las frecuencias, el filtro de Gabor G(u, v) se define como la función Gaussiana:

$$G(x,y) = \mathcal{F}^{-1}\{\hat{G}\} = \exp\left(-\pi\left(\frac{x^2}{\alpha^2}+\frac{y^2}{\beta^2}\right)\right)\exp\left(-2\pi i\left(u_0x+v_0y\right)\right)$$

El filtro resultante es entonces una onda sinusoide compleja modulada por una función Gaussiana.

La frecuencia espacial del filtro es $\omega_0 = \sqrt{u_0^2+v_0^2}$ y su orientación es θ = arctan (v0/u0), mientras que el ancho de banda está dado por el radio de la Gaussiana (definido por α y β).

4.5.4 Generación del iriscode:

Para generar el iriscode asociado a una imagen determinada es necesario utilizar la imagen transformada y filtrada. Esta imagen tiene los valores extremos de la intensidad de gris en sus pixeles resaltada, luego se le debe asignar 0 ó 1 a cada pixel en dependencia del signo de la parte real e imaginaria de cada uno de ellos, generando así el iriscode.

4.6 Pruebas

4.6.1 Descripción de los casos de prueba.

La realización de pruebas es una actividad en la cual un sistema o componente es ejecutado bajo ciertas condiciones o requerimientos específicos, los resultados son observados y registrados, y se realiza una evaluación de algún aspecto del sistema o componente.

Uno de los niveles de pruebas que existen son las pruebas de sistema, las cuales tienen como objetivo fundamental verificar el software ejerciendo como un todo. A continuación se detallan los casos de prueba elaborados por cada caso de uso para probar el correcto funcionamiento del sistema bajo las diferentes condiciones a las que puede someterse.

Nombre del caso de uso:	Normalizar imagen.	
Entrada	**Resultados esperados**	**Resultados obtenidos**
Caso de Prueba #1		
Se introduce una incorrecta imagen de entrada por mala segmentación de la imagen.	Acción fallida. El sistema muestra el mensaje “Error: Segmentation failure”.	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #2		
Se solicita normalizar una imagen segmentada cuyo iris contiene insuficiente	Acción fallida. El sistema muestra el mensaje “Error: Capture failure or Segmentation failure”.	Prueba fallida. El sistema retorna una imagen

información para ser procesado.		del iris rectangular de dimensiones fijas.
Caso de Prueba #3		
Se solicita normalizar una imagen segmentada.	El sistema retorna una imagen del iris rectangular de dimensiones fijas.	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #4		
Se solicita normalizar varias imágenes segmentadas con diferentes dimensiones y radios del iris.	El sistema retorna siempre una imagen del iris rectangular de dimensiones fijas asociada a la imagen de entrada.	Prueba exitosa. Se obtienen los resultados esperados.

Tabla 7: Casos de prueba para CU Normalizar Imagen.

Nombre del caso de uso:	Codificar imagen.	
Entrada	**Resultados esperados**	**Resultados obtenidos**

Caso de Prueba #1		
Se solicita codificar una imagen normalizada.	El sistema retorna una imagen del iriscode asociado a la misma.	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #2		
Se solicita codificar dos veces la misma imagen normalizada.	El sistema retorna el mismo iriscode en ambos casos.	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #3		
Se solicita codificar dos imágenes normalizadas diferentes.	El sistema retorna dos iriscode diferentes, cada uno asociado a una imagen.	Prueba exitosa. Se obtienen los resultados esperados.

Tabla 8: Casos de prueba para CU Codificar Imagen.

4.7 Conclusiones.

Los estilos de codificación y documentación explicados en este capítulo son aplicados en el Centro de Informática Industrial. Esto justifica el empleo de los mismos en la biblioteca que se desarrolla, lo cual evidencia la intención de lograr una biblioteca que pueda ser comprendida por usuarios comunes sin necesidad de tener un amplio conocimiento respecto al tema.

Además, en este capítulo se exponen los resultados obtenidos a partir de la aplicación de los diferentes casos de prueba efectuados a las funcionalidades del sistema, de los cuales se deriva la solidez del mismo. Como conclusión de estas pruebas se puede decir que de un total de 7 casos, 6 resultaron satisfactorios y 1 demostró vulnerabilidad del software. Todos los errores detectados fueron solucionados, lo cual demuestra que el proceso de validar la etapa de codificación de la biblioteca concluye de manera exitosa.

Conclusiones generales

Con el objetivo de incorporar a la biblioteca de autenticación de usuarios por reconocimiento de iris del proyecto SCADA el algoritmo que dada una correcta localización de la región del iris codifique una imagen determinada, todas las tareas investigativas propuestas se cumplieron exitosamente, por lo que al culminar este proceso de desarrollo se puede concluir lo siguiente:

- Se sentaron las bases para la mejora de la etapa de segmentación a través del uso de filtros pasa alto para intensificar la variación de intensidades en una imagen.
- Se dio solución a la etapa de normalización para la biblioteca de autenticación de usuarios por reconocimiento de iris del proyecto SCADA.
- Se dio solución a la etapa de codificación para la biblioteca de autenticación de usuarios por reconocimiento de iris del proyecto SCADA con el uso de la transformada de Fourier y el filtro de Gabor.
- De manera general se cumplió con el objetivo trazado al inicio de la investigación y se alcanzaron los resultados esperados.

Recomendaciones

A continuación se recomiendan posibles aspectos a tener en cuenta para continuar con la mejora y perfeccionamiento de la etapa de codificación y normalización de la biblioteca:

- Utilizar filtros pasa alto antes de segmentar las imágenes para obtener mejores resultados en esta etapa y en las que la suceden.

- Experimentar el proceso de filtrado con los filtros de Log-Gabor, ya que estos tienen, al igual que los filtros de Gabor, una función de transferencia[8] Gaussiana pero en escala logarítmica[9] y pudieran codificar mejor las imágenes naturales que los filtros de Gabor.

[8] Modelo matemático que a través de un cociente relaciona la respuesta de un sistema a una señal de entrada o excitación.
[9] Escala de medida que utiliza el logaritmo de una cantidad física en lugar de la propia cantidad.

Referencias bibliográficas

[1] Mortalli, M. L. (Octubre de 2008). Implementación de un sistema de identificación de personas en tiempo real por reconocimiento de iris. Universidad de Buenos Aires. Buenos Aires, Argentina. Recuperado el 15 de Noviembre de 2010, de Implementación de un sistema de identificación de personas en tiempo real por reconocimiento de iris.: www-2.dc.uba.ar/grupinv/imagenes/archivos/tesisMottalli2008.pdf

[2] Procesamiento digital de imágenes. (2010). Recuperado el 5 de Diciembre de 2010, de Procesamiento digital de imágenes: http://es.scribd.com/doc/55307088/Procesamiento-digital-de-imagenes.

[3] autores., C. d. (2011, Enero). Tycon Software Ingineering. Recuperado el 10 de Marzo de 2011, from Tycon Software Ingineering: http://www.tycon.com.ar/Tecnolog%C3%ADa/tabid/58/Default.aspx

Bibliografía

1. Ana Fred, J. F. (2009). Biomedical Engineeiring Systems and Tecnologies . Porto, Portugal.

2. Belmonte, R. C. (2006). Sistema de reconocimiento de personas mediante su patrón de iris basado en la transformada de wavelet. Madrid, España.

3. Daugman., J. G. (1988). Complete Discrete 2D Gabor Transforms by Neural Networks for Image Analysis and Compression. Cabridge: Invited Paper.

4. García, J. O. (2010). Reconocimiento Automático de Patrones del Iris. Madrid, España.

5. Glyn James, D. B. (2002). Matemática avanzada para ingeniería. México.

6. González, M. (2010). Reconocimiento de iris. Barcelona.

7. Laura Florian Cruz, F. C. (2006). Reconocimiento del iris. Trijullo, Perú.

8. Masek, L. (2003). Recognition of human iris patterns for Biometric Identification. Western, Australia.

9. Monro, D. M. (2007). DCT Based Iris Reconiction.

10. Mortalli, M. L. (2008). Implementación de un sistema de identificación de personas en tiempo real por reconocimiento de iris. Buenos Aires, Argentina.

11. Movellan, J. R. (2008). Tutorial on Gabor Filter.

12. Muller, C. (2007). IV Congreso Latinoamericano de ingeniería biométrica. Islas Margaritas, Venezuela.

13. Rafael C Gonzalez, R. E. (1992). Digital Image Processing. New Jersey: Prentice Hall .

14. Rosario Almeyda, P. L. (2004). Reconocimiento de iris. Recuperado el 20 de Enero de 2011, de Reconocimiento de iris.: http://iie.fing.edu.uy/investigacion/grupos/gti/timag/trabajos/2004/recon_iris

15. Stan Z Li, Z. S. (2005). Advances in biometric Person Authentication. Beijin.

16. Teressi, L. D. (2000). Sistema de Reconocimiento de Iris. Rosario, Argentina.

Anexos

Anexo 1: Imágenes de los resultados arrojados por la biblioteca.

Ejemplo 1.

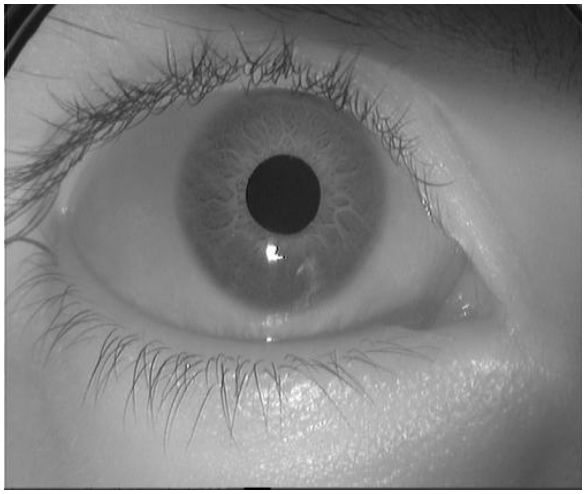


Figura 13: Imagen de entrada ejemplo 1.

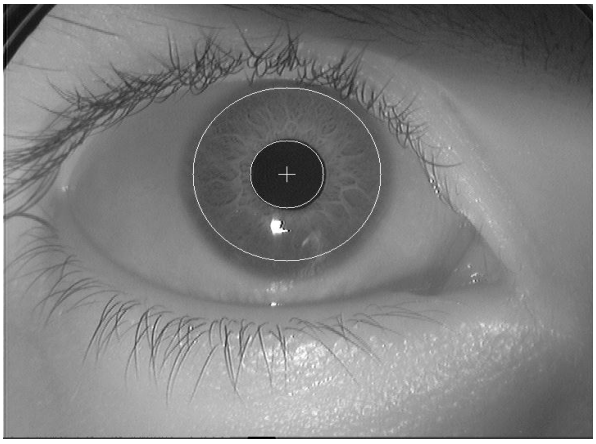


Figura 14: Imagen segmentada ejemplo 1.

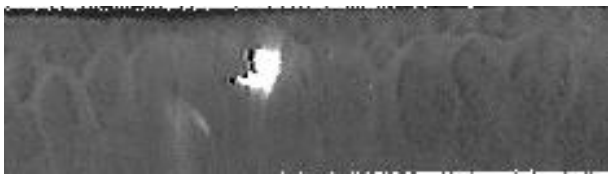


Figura 15: Imagen Normalizada ejemplo 1.

Figura 16: Imagen de Codificada ejemplo 1.

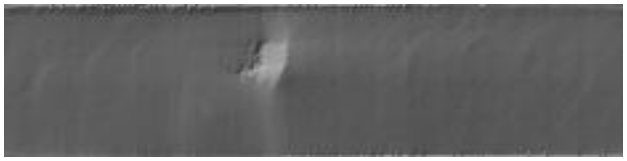


Figura 17: Imagen reconstruida ejemplo 1.

Ejemplo 2.

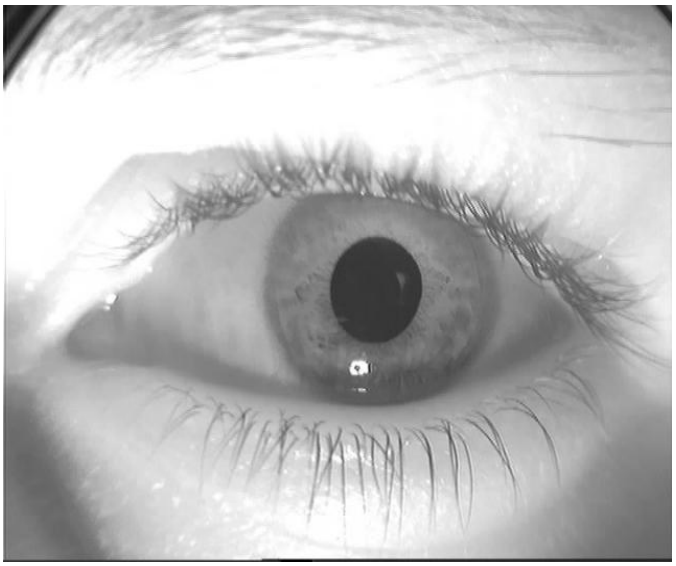


Figura 18: Imagen de entrada ejemplo 2.

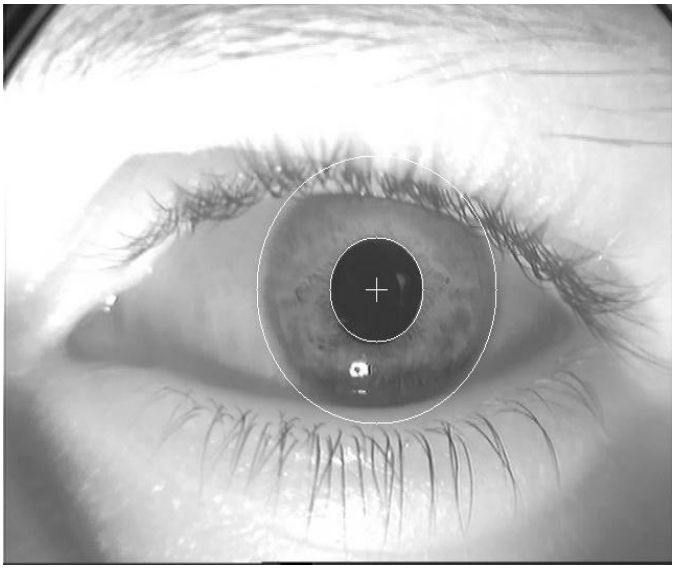


Figura 19: Imagen segmentada ejemplo 2.

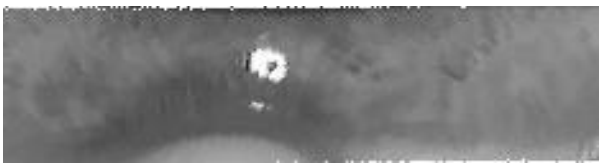


Figura 20: Imagen Normalizada ejemplo 2.

Figura 21: Imagen de Codificada ejemplo 2.

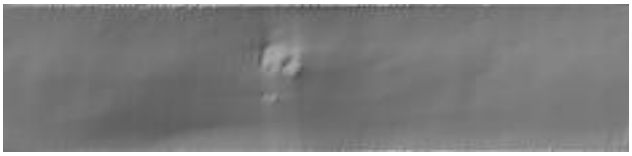


Figura 22: Imagen reconstruida ejemplo 2.

Glosario de términos

A

- Aplicación: Terminología técnica para referirse a un programa de software.
- Arquitectura: Indica la estructura, funcionamiento e interacción entre las partes del software.
- Autenticación: Verificación de la identidad de una persona o de un proceso para acceder a un recurso o poder realizar determinada actividad.

C

- CEDIN: Siglas que identifican el Centro de Informática Industrial de la facultad 5 de la Universidad de las Ciencias Informáticas.
- Cliente: Un sistema o proceso que solicita a otro sistema o proceso que le preste un servicio.
- Contraseña: Información confidencial constituida por una cadena de caracteres, usada para la autenticación de un usuario o en el acceso a un recurso.
- Control de acceso: Mecanismo que en función de la identificación ya autenticada permite acceder a datos o recursos.

H

- Hardware: Componente físico de una computadora o de una red, en contraposición con los programas o elementos lógicos que lo hacen funcionar.

I

- IDE: Siglas en inglés que identifican un Entorno de Desarrollo Integrado (*Integrated Development Environment)*. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.
- Identificación del usuario: Procedimiento de reconocimiento de la identidad de un usuario.

P

- Programa: Conjunto de instrucciones que una vez ejecutadas realizan una o varias tareas en una computadora.

R

- Recurso: Cualquier parte componente de un sistema de información.

S

- SCADA: Siglas que identifican a un sistema desarrollado para el control y la supervisión de datos (Supervisory Control And Data Adquisition).
- Software: Conjunto de programas, documentos, procesamientos y rutinas asociadas con la operación de un sistema de computadoras, es decir, la parte intangible o lógica de una computadora.

U

- Usuario: Sujeto o proceso autorizado para acceder a datos o recursos.