INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Secretaría de Investigación y Posgrado

Implementación de algoritmos de procesamiento de imágenes en FPGA

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS EN INGENIERÍA DE CÓMPUTO
CON OPCIÓN EN SISTEMAS DIGITALES

PRESENTA:

Ing. Germán Oswaldo López Verástegui

DIRECTORES DE TESIS:

Dr. Edgardo Manuel Felipe Riverón

Dr. Sergio Suárez Guerra

México, D.F.

Junio de 2014

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 13:00 horas del día 9 del mes de mayo de 2014 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Implementación de algoritmos de procesamiento de imágenes en FPGA"

Presentada por el alumno:

Lopez	**Verastegui**	**German Oswaldo**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: **A 1 2 0 5 7 3**

aspirante de: **MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON OPCIÓN EN SISTEMAS DIGITALES**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. Edgardo Manuel Felipe Riverón

Dr. Sergio Suárez Guerra

Dr. Oleksiy Pogrebnyak

Dr. Ricardo Barrón Fernández

M. en C. Pablo Manrique Ramírez

Dr. Salvador Godoy Calderón

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de **México, D.F.** el día **19** del mes **Mayo** del año **2014**, el (la) que suscribe **German Oswaldo Lopez Verastegui** alumno (a) del Programa de **Maestría en Ciencias en Ingeniería de Cómputo con opción en sistemas digitales** con número de registro **A120573**, adscrito al **Centro de Investigación en Computación del Instituto Politécnico Nacional,** manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de **Dr. Edgardo Manuel Felipe Riverón y Dr. Sergio Suárez Guerra** y cede los derechos del trabajo intitulado **"Implementación de algoritmos de procesamiento de imágenes en FPGA"**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección **glopezv0300@gmail.com.** Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

López V. Germán O.

German Oswaldo Lopez Verastegui

Resumen

En este trabajo de investigación se presenta el desarrollo e implementación en FPGA de un algoritmo de segmentación y normalización de iris aplicado a imágenes de iris humanos provenientes de la base de datos CASIA versión 1, mediante el uso de 3 técnicas de implementación en sistemas digitales:
Procesamiento en hardware, Procesamiento en un procesador suave (usando NIOS II y programación en C/C++), y Procesamiento con un procesador suave y módulos de hardware. Dicha implementación se realiza en la tarjeta de desarrollo FPGA DE2-115 de Altera y lenguajes de descripción de hardware (VERILOG), junto con la herramienta Qsys (proporcionada en el software de desarrollo de Altera) para el desarrollo de la arquitectura base.

El algoritmo desarrollado para el procesamiento de las imágenes está compuesto por procesamientos sencillos que permiten segmentar y normalizar el iris de manera correcta en un 96%.

Con la implementación en FPGA de los algoritmos de segmentación y normalización del iris, se obtienen las imágenes y los resultados esperados, es decir, dadas las imágenes de entrada, el algoritmo, al igual que en una PC, ejecuta la tarea de procesamiento y como salida da la imagen del iris normalizada, con las mismas características (formato.BMP, 8 bits por color, tamaño) que las obtenidas en la PC. En promedio, hay un error cuadrático medio de 4.53 entre el histograma de la imagen resultado obtenida en la PC y la obtenida en el FPGA causado por la representación numérica en ambas plataformas; a pesar de este valor de error, al obtenerse la plantilla de iris o matriz binaria de identificación, hay un porcentaje de diferencia menor al 10% (0.0822) entre las mismas.

Con la implementación de procesamiento de imágenes en hardware, se gana en velocidad de ejecución.

La implementación de algoritmos en FPGA tiene como gran ventaja que el algoritmo se ejecuta y da los mismos resultados que su implementación en la PC, pero con un consumo mucho menor de energía que la PC, ya que son sistemas dedicados compuestos de circuitos integrados de bajo consumo en comparación con lo que consume una PC; además, un circuito integrado FPGA y sus periféricos, requieren mucho menos espacio y pueden ser utilizados en aplicaciones donde las restricciones de espacio y tiempo son especiales.

Abstract

In this research work we present the development and implementation of an algorithm for segmenting and normalizing the iris in images provided by the CASIA V1 database using three techniques to implement digital systems: processing in hardware (using concurrency), processing in a soft processor (using NIOS II and C/C++ language), and by means of the soft processor and hardware modules. This implementation is done in the FPGA development board Altera DE2-115 and the hardware description language VERILOG, together with the Qsys tool (provided in the development software Altera) to develop the base architecture.

The algorithm developed for the image processing is tested using simple processing methods that allow to segment and normalize correctly the iris in 96% of cases.

With the implementation of the segmentation algorithm and normalization of iris images in FPGA , the results are as expected, i.e., given the input images, the algorithm performs as well as on a PC, runs the processing task and outputs the normalized iris image with the same characteristics (format: BMP, 8 bits per color) than those in the PC. There is a root mean square error of 4.53 between the histogram of the resulting image obtained in the PC and that obtained in the FPGA caused by the nature of the calculations on both platforms; despite this error value, the obtained template or binary iris identification matrix has a percentage difference between them less than 10% (0.0822).

With the implementation of the image processing algorithms in hardware, there is a gain in execution speed.

The implementation of algorithms on FPGA has the great advantage that the algorithm runs and gives the same results as its implementation in the PC, but with a much lower power consumption than the PC because it is a dedicated system composed of integrated circuits of low energy consumption compared to the energy consumed by a PC; besides this, an integrated FPGA and its peripheral circuits require much less space and can be used in applications where space and time constraints are specials.

Agradecimientos

Principalmente quiero agradecer a Dios y a esas 2 personas que me dieron la vida, que han velado por mí y me han apoyado en todo momento para alcanzar mis metas, aquellas personas que han trabajado incansablemente para formarme y darme lo mejor, mis Padres quienes son el pilar que sostiene todo lo que soy y he logrado, Gracias.

Gracias a mis hermanos por su paciencia y apoyo en todo momento. Así también gracias a mi Familia que me ha apoyado en todo y en todo momento han creído en mí.

Gracias a todas aquellas personas que han compartido parte de su camino conmigo, que me han brindado su amistad y compañerismo, y me ayudaron y ayudan a disfrutar del camino en cada etapa.

Agradezco a cada uno de los Profesores con los que he tenido la oportunidad de tomar una clase, de todos he aprendido mucho más que lo académico, gracias a ustedes este trabajo ha sido posible. En especial, quiero agradecer a mis directores de tesis, al Dr. Edgardo Manuel Felipe Riverón y Dr. Sergio Suárez Guerra, quienes me dieron su apoyo, sus consejos, sus conocimientos y experiencias para la realización de este trabajo.

Agradezco al Instituto Politécnico Nacional por brindarme la oportunidad de crecer y prepararme de forma académica y personal desde el nivel medio superior hasta el posgrado.

Agradezco al Centro de Investigación en Computación por la oportunidad y el apoyo recibido para la realización de este trabajo de tesis.

Agradezco al Consejo Nacional de Ciencia y Tecnología por el apoyo otorgado para la realización de este trabajo de tesis.

Agradezco a Fundación Telmex por todo el apoyo otorgado para mi formación académica desde el nivel superior hasta este nivel de posgrado.

Muchas gracias a Todos

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE SIGLAS Y ACRÓNIMOS

ASIC	*Application-Specific Integrated Circuits* (Circuitos integrados de aplicación específica)
BMP	*BitMap* (Mapa de bits)
CCM	*Custom Computing Machine* (Máquina computadora del cliente)
CPLD	*Complex Programmable Logic Device* (Dispositivo lógico complejo programable)
DAC	*Digital to Analogue Converter* (Conversor digital analógico)
DFFC	*Data Flow Functional Computer* (Computadora funcional de flujo de datos)
DSP	*Digital Signal Processor* (Procesador de señales digitales)
EDA	*Electronics Design Automation* (Automatización del diseño electrónico)
EM	*Expectation-Maximization* (Expectación-Maximización)
E/S	Entrada/Salida
FFT	*Fast Fourier Transform* (Transformada rápida de Fourier)
FPGA	*Field Programmable Gate Array* (Arreglo de compuertas programables en campo)
FPID	*Field Programmable Interconnect Devices* (Dispositivos de interconexiones programables en campo)
FPOA	*Field Programmable Operator Array* (Arreglo de operadores de campo programable)
GPU	*Graphics Processing Unit* (Unidad de procesamiento gráfico)
HDL	*Hardware Description Language* (Lenguaje de descripción de hardware)
IP	*Intellectual Property*(Propiedad intelectual)
JPG	*Joint Photographic Experts Group* (Grupo conjunto de expertos en fotografía).
NIR	*Near Infrared* (Infrarrojo cercano)
PC	*Personal Computer* (Computadora personal)
PCI	*Peripheral Component Interconnect* (Interconexión de componentes periféricos)
PLD	*Programmable Logic Device* (Dispositivo lógico programable)
RBF	*Radial Basis Function* (Función de base radial)
SRAM	*Static Random Access Memory* (Memoria de acceso aleatorio estática)

SOC	*System On Chip* (Sistema en un chip)
SVM	*Support Vector Machine* (Máquina de soporte vectorial)
USB	*Universal Serial Bus* (Bus en serie universal)
VERILOG	*Verify Logic* (Lógica verificable)
VHDL	VHSIC es el acrónimo de *Very High Speed Integrated Circuit* y HDL, que es a su vez el acrónimo de *Hardware Description Language*.
VGA	*Video Graphics Array* (Arreglo gráfico de video)
VHSIC	*Very High Speed Integrated Circuit* (Circuito integrado de muy alta velocidad)
VLSI	*Very Large Scale Integrated* (integración en escala muy grande)

CAPITULO I

Introducción

Para comprender mejor el propósito y la temática de este trabajo, en este primer capítulo se exponen la introducción, fundamentación y organización del trabajo.

1.1. Aspectos generales

Uno de los principales objetivos de las aplicaciones de la visión artificial es la descripción de forma automática de una determinada escena. Así, se entiende como descripción a la identificación y localización de los objetos que la forman con base en sus características [1]. Frecuentemente, el principal problema de las aplicaciones de visión artificial es el tiempo de ejecución de los algoritmos y el costo de los equipos. La mayor demanda de prestaciones de los algoritmos de procesamiento de imágenes, unido a la mayor resolución espacial, hace que cada vez sean mayores las demandas de carga computacional. Si además se desea trabajar en tiempo real, las exigencias de los tiempos de ejecución de los algoritmos son aún más críticas.

Habitualmente, las plataformas elegidas para realizar estos algoritmos son las basadas en programas secuenciales. Sin embargo, éstas no son las óptimas en muchas aplicaciones desde un punto de vista de rendimiento. Por tanto, se justifica la búsqueda de nuevos sistemas segmentados para procesamiento de imágenes en paralelo. Así, según [2], con base en las características de computación, la mayoría de los sistemas de visión se pueden dividir en tres niveles tal y como se muestra en la Figura 1.1: en el nivel inferior (bajo nivel), se incluyen operaciones al nivel de píxel, tales como filtrado o detección de bordes. Este nivel se caracteriza por manejar un elevado número de datos (pixeles) y de operaciones sencillas, como sumas y multiplicaciones. El procesado de nivel medio, se caracteriza por realizar operaciones más complejas de bloques de pixeles. Dentro de este nivel se pueden incluir operaciones de segmentación o etiquetado de regiones. Por último, las tareas pertenecientes al nivel alto, las cuales se caracterizan por una alta complejidad en la algoritmia. Dentro de este último nivel, se podrían incluir, por ejemplo, todos los algoritmos de pareo (*matching*). Se puede observar en la Figura 1.1 cómo a medida que crece la altura de la pirámide, el volumen de datos a manejar disminuye, incrementándose la complejidad computacional de los algoritmos. La Tabla 1.1 muestra un resumen de las características de cada nivel.

Tabla 1.1 Características y ejemplos de los diferentes niveles de complejidad computacional [1].

Nivel	Características	Ejemplos
Bajo	Acceso a los pixeles vecinos cercanos, operaciones simples, manejo de gran cantidad de datos	Detección de bordes, filtrado, operaciones morfológicas simples
Medio	Acceso a bloques de pixeles, operaciones más complejas	Transformada de Hough, conectividad
Alto	Acceso aleatorio, operaciones complejas	Pareo (*matching*)

El procesado digital de imágenes requiere habitualmente el manejo de un alto número de operaciones al nivel de bits que se desean ejecutar en el menor tiempo posible. Debido a la arquitectura secuencial de los ordenadores convencionales, no se pueden ejecutar concurrentemente muchas operaciones. Por ello, cuando el número de datos a manejar es grande, el rendimiento del sistema es muy bajo. Sin embargo, una plataforma de hardware de propósito específico (diseñada expresamente para una aplicación), puede dar excelentes resultados. Así, operaciones relativamente sencillas como filtros, descompresión de imágenes, etc., se realizan con un rendimiento elevado. Por otro lado, si

se desea implementar algoritmos más complejos, estos hay que reformularlos para explotar las características de la plataforma sobre la que se ejecutará.

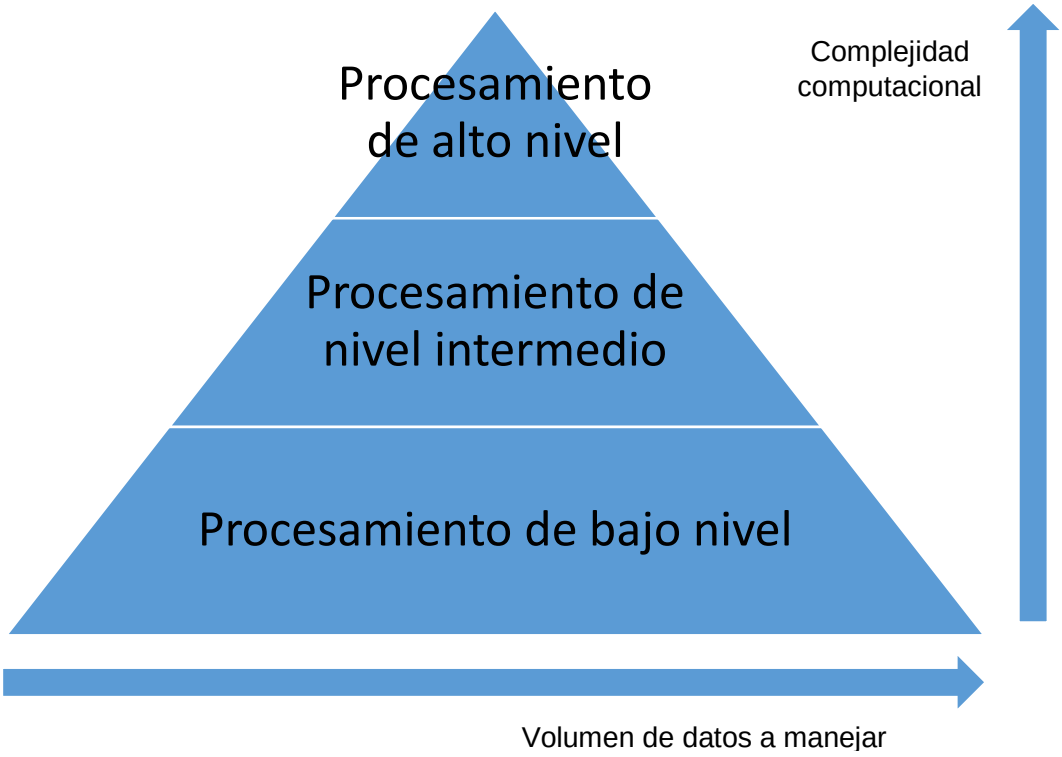


Figura 1.1. Pirámide ilustrativa del volumen de datos vs. complejidad computacional del procesamiento de imágenes [2].

1.1.1. Aspectos generales sobre segmentación de iris en imágenes de iris humanas

A lo largo de la historia, el ser humano ha implementado distintos métodos para reconocerse y poder distinguirse entre las demás personas. Esto nace debido al interés natural del hombre por querer proteger lo que le pertenece y mantener privacidad en sus acciones. Dado eso, es que un reconocimiento confiable de las personas ha sido una meta a alcanzar.

El patrón del iris del ojo humano utiliza parámetros para la identificación de las personas. El pionero en esta técnica es John Daugman que tiene varios trabajos publicados y métodos patentados sobre este tema [28, 32, 36, 37].

En general, las características de un sistema biométrico son:

- Universalidad: cualquier persona posee esa característica;
- Unicidad: la existencia de dos personas con una característica idéntica tiene una probabilidad muy pequeña;
- Permanencia: la característica no cambia en el tiempo; y
- Cuantificación: la característica puede ser medida en forma cuantitativa.

El patrón del iris posee las características mencionadas anteriormente. Cualquier persona lo tiene, la probabilidad de que dos personas tengan un mismo patrón es ínfima debido a que la formación del patrón se realiza aproximadamente durante el primer año de vida de la persona mediante un proceso aleatorio morfogenético. Una vez formada la textura es estable, lo que le da permanencia. En cuanto a la cuantificación, existen diferentes metodologías, la mayoría orientada al análisis de las frecuencias de los patrones presentes en el iris.

El diseño del proceso de captura presenta ciertas dificultades tanto para que no sea invasivo para la persona como también para proporcionar muestras homogéneas debido al ángulo de captura, iluminación, etc.

Otro problema interesante es la segmentación del iris, ya que se debe extraer quitando de la imagen los demás elementos que no son de interés para el problema.

1.1.2. Aspectos generales sobre procesamiento de señales e imágenes en hardware

En cualquier desarrollo práctico de un sistema electrónico, existen diferentes alternativas para intentar aumentar las prestaciones sin incrementar el precio del mismo, preservando en gran medida los requisitos del diseño. Esto es particularmente aplicable a sistemas de procesado de señales o imágenes, donde se requiere en algunos casos una alta carga computacional y al mismo tiempo mayor potencia de cálculo. Para esas aplicaciones existen numerosas alternativas de hardware/software entre las que se pueden destacar: los dispositivos de procesamiento digital de señales DSP (del inglés *Digital Signal Processor*), los circuitos integrados de aplicación específica ASIC (del inglés *Application-Specific Integrated Circuits*) y los arreglos de compuertas programables en campo FPGA (del inglés *Field Programmable Gate Array*). Estas alternativas ofrecen diferentes grados de eficiencia que deben ser sopesados con respecto a factores tales como costos, potencia consumida o tiempo de diseño (Ver la subsección 2.1.1 **Antecedentes**).

Otro aspecto importante que caracteriza la mejora de prestaciones que ofrecen los dispositivos es su evolución tecnológica. Hoy en día, la generación de nuevos dispositivos de hardware se ajusta totalmente a la conocida ley de Moore [3] (Figura 1.2). Esta evolución también es aplicable a los dispositivos FPGA. El número de recursos internos de estos ha crecido notablemente en los últimos años, generándose dispositivos cada vez más preparados hacia el cómputo de imágenes por hardware. Gracias a ello, es posible que estos dispositivos hayan pasado de ser usados inicialmente como elementos coprocesadores, a funcionar actualmente como verdaderos procesadores de aplicaciones de sistemas en tiempo real.

Un FPGA es un dispositivo que un diseñador de sistemas digitales puede programar, después que está soldado en el circuito impreso, para que funcione de un modo determinado. Los FPGA son fabricados con conexiones y lógica programables. El diseñador desarrolla su sistema digital usando herramientas tipo EDA (*Electronics Design Automation* – Automatización de diseños electrónicos), ya sean mediante dibujos esquemáticos o de algún lenguaje de descripción de hardware para poder plasmar el sistema en lógica digital. Luego de simular satisfactoriamente el sistema digital para comprobar su funcionalidad se usan herramientas específicas del vendedor del FPGA para crear un archivo de configuración del mismo, el cual describe todas las conexiones, interconexiones y lógica que necesita ser programada dentro del FPGA para implementar el sistema digital desarrollado. Entonces, a través de un cable USB se conecta el FPGA o el circuito impreso al cual está soldado el FPGA a una PC y con la ayuda del software de configuración del

FPGA, se descarga el archivo de configuración. Una vez comprobado el correcto funcionamiento del sistema en el FPGA, se graba el archivo de configuración en una memoria no-volátil que el FPGA leerá y usará para autoconfigurarse, cada vez que se aplica la tensión de alimentación al FPGA o cada vez que se desee reconfigurar el FPGA.

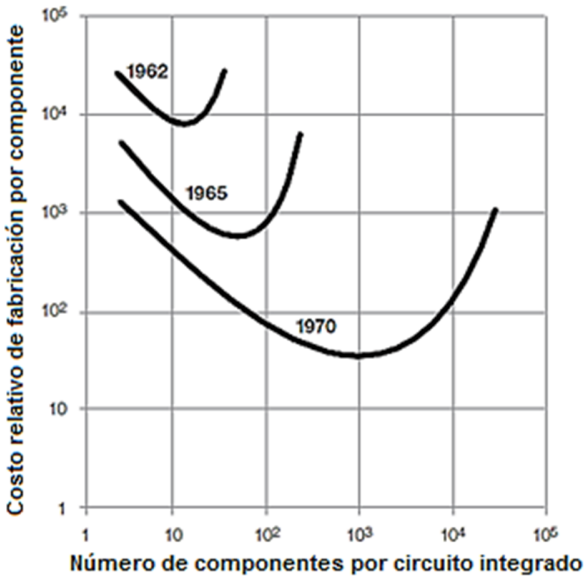


Figura 1.2. Evaluación del crecimiento de los circuitos integrados [3].

Todos los FPGA, independientemente del fabricante, tienen ciertos elementos en común: tienen un arreglo tipo matricial de elementos lógicos, como *flips-flops* y lógica combinatoria, que se configuran usando cierta tecnología de programación. Los terminales de entrada y salida del FPGA usan celdas especiales de E/S que son diferentes de las celdas de elementos lógicos. Además, tienen un esquema de interconexión programable que permite la conexión entre las celdas de elementos lógicos entre sí y con las celdas de E/S. La programación de las interconexiones y de los elementos lógicos puede o no ser permanente, pues depende de la tecnología de programación usada.

La mejora en los dispositivos reconfigurables ha posibilitado la implementación de sistemas digitales de alto nivel, comúnmente denominados SOC (*System On Chip*), con un ahorro económico y en tiempo de diseño. Económico, porque el número de circuitos integrados se reduce al poder integrar en un FPGA numerosos sistemas digitales (incluso microprocesadores de alto nivel). Si el análisis se realiza desde un punto de vista de tiempo de diseño, los tiempos de elaboración de un proyecto también se reducen al integrar todo en un FPGA.

Sin embargo, el empleo de estos dispositivos también conlleva ciertos problemas. Uno de ellos y quizás el más importante, es el error producido por el empleo de datos habitualmente codificados en punto fijo. Actualmente, pocas son las aplicaciones desarrolladas en FPGA implementadas en punto flotante.

Otro problema importante son las herramientas de diseño y de lenguajes de programación. Éstas no permiten trabajar con lenguajes de alto nivel, lo que obliga a abordar el diseño mediante lenguajes de descripción de hardware de bajo nivel o de nivel intermedio, por ejemplo VHDL (VHSIC *Hardware Description Language*, donde el vocablo VHSIC se refiere a Very High Speed Integrated Circuit – Circuito Integrado de muy alta velocidad) o VERILOG (Verify Logic - Lógica verificable).

1.2. Justificación

La implementación de algoritmos de alto nivel sobre FPGA tiene ventajas frente a otras arquitecturas. A continuación se exponen algunas de ellas:

- El hecho de poder ejecutar algoritmos de forma concurrente en un dispositivo reconfigurable logra acelerar los tiempos de ejecución con respecto a los ejecutados de forma secuencial.
- Debido a la característica de reconfigurabilidad de los FPGA, ésta permite un alto grado de flexibilidad, lo que conlleva a que cualquier variación del diseño no tenga que llevar implícito cambio alguno en función de la plataforma empleada.
- El reducido precio de la plataforma donde se desarrollará la aplicación, pues el costo de una plataforma basada en un FPGA es notablemente inferior al que puede tener una computadora personal (PC) o un DSP.

Todas estas razones nos conducen al intento de adaptar algoritmos de alto nivel sobre dispositivos reconfigurables.

1.3. Hipótesis

Es posible crear un sistema específico para procesar imágenes con dispositivos FPGA, que brinde ventajas con respecto a sistemas similares desarrollados mediante software en computadoras personales de propósito general en relación con el costo y rapidez.

1.4. Objetivos

El objetivo general y los objetivos particulares del presente trabajo son los siguientes:

1.4.1. Objetivo general

Desarrollar una arquitectura para procesar imágenes digitales de iris humano, segmentarlos y normalizarlos en un dispositivo FPGA.

1.4.2. Objetivos particulares

Los objetivos particulares son los siguientes:

- Desarrollar una técnica de segmentación del iris.
- Evaluar la técnica de segmentación de iris desarrollada.
- Desarrollar una arquitectura base para lectura, escritura y despliegue de imágenes, así como para su manipulación y procesamiento en el FPGA.
- Implementar la técnica desarrollada en un dispositivo FPGA con los módulos necesarios para el manejo de datos de la imagen y su despliegue en un visualizador (display) por medio del puerto del Adaptador Gráfico de Video (VGA-Video Graphics Array).
- Evaluar los resultados obtenidos con la implementación de la técnica desarrollada en el FPGA.

- Comparar los resultados obtenidos con el FPGA con respecto a los resultados que se obtienen de una computadora personal (PC).

1.5. Alcances

Los alcances del trabajo son los siguientes:

- Los que permite la tarjeta de desarrollo FPGA utilizada para el desarrollo del trabajo.
- Poder desplegar y procesar imágenes en colores hasta de 1024 x 768 pixeles de tamaño.
- Cada píxel tendrá asociado un máximo de 8 bits por color, por lo que los pixeles se representarán en color verdadero (true color) con 24 bits/píxel.
- Manejo de sistema de archivos FAT16 y FAT32 para manipulación de archivos de imagen en una tarjeta externa de memoria SD (*Secure Digital*)
- Manejo de formato de imágenes en BMP y JPG.

1.6. Contribuciones

Se proporcionaron las herramientas de hardware/software necesarias para procesar imágenes en colores de un tamaño máximo de 1024 x 768 en la tarjeta de desarrollo FPGA DE2-115 de Altera, con todas las ventajas que estos pueden proporcionar, a saber:

- La posibilidad de reducir el tiempo de procesamiento del algoritmo implementado en hardware, lo que permite su utilización en sistemas con restricciones de tiempo (“tiempo real”).
- La portabilidad del sistema, ya que el algoritmo queda implementado en un circuito integrado que es fácilmente portable y adaptable a cualquier sistema embebido.
- Debido a que el algoritmo se implementa en hardware en la tarjeta de desarrollo FPGA, el consumo de energía con respecto a la ejecución del algoritmo en una computadora personal es mucho menor.

Con dicho objetivo, en este proyecto se ha desarrollado un algoritmo de segmentación de iris que puede emplearse tanto en procesos de reconocimiento biométrico como en otro tipo de aplicaciones, como por ejemplo pueden ser pruebas médicas (existen en la actualidad, pruebas de análisis de imagen oftalmológica en cuyo proceso interviene la segmentación de iris para detección de glaucoma), mecanismos de detección de sujeto vivo. etc.

1.7. Método de investigación y desarrollo utilizados

El trabajo a realizar, dentro del ámbito del reconocimiento de las personas mediante el análisis del iris, está centrado en el procesamiento y la segmentación de la imagen para la obtención de la subimagen del iris ya normalizado, lista para la extracción posterior de sus características.

1.8. Organización del trabajo

La tesis está compuesta de 6 capítulos. El Capítulo 1, **Introducción**, describe los aspectos generales sobre el procesamiento del iris humanos para la identificación de las personas, la implementación de algoritmos de procesamiento de señales e imágenes en plataformas de hardware, el objetivo general y los específicos del trabajo así como la fundamentación y organización del trabajo.

El Capítulo 2, **Estado del arte**, presenta el estado del arte de los dispositivos FPGA ya desarrollados para procesar imágenes digitales, así como de las técnicas y algoritmos actuales para la segmentación de iris.

Los conceptos generales utilizados en su desarrollo, están en el Capítulo 3 intitulado **Marco teórico**.

El Capítulo 4, **Solución del problema**, expone con detalles la forma en que se ha procedido para lograr el cumplimiento del objetivo general y los objetivos particulares propuestos para el desarrollo.

En el Capítulo 5, **Resultados**, mediante la implementación y prueba de los algoritmos en Matlab y en lenguaje C/C++ se evalúan los resultados obtenidos así como el tiempo de procesamiento en diferentes computadoras personales, del mismo modo se evalúan los resultados y se obtienen los tiempos de la implementación en el FPGA, comparando los resultados de dichas evaluaciones.

En el Capítulo 6, **Conclusiones y trabajos futuros**, se exponen los resultados, recomendaciones y trabajos futuros que se proponen a partir del desarrollo logrado.

CAPÍTULO II

Estado del arte

En este capítulo trataremos algunos los antecedentes del procesamiento de imágenes digitales en FPGA y algunos de los trabajos relevantes en el mismo ámbito; así como de trabajos desarrollados de procesamiento de imágenes para la segmentación del iris humano.

2.1. Procesamiento de imágenes en los FPGA

El procesado de imágenes en tiempo real requiere en muchas aplicaciones de un rápido y continuo cálculo de operaciones a la hora de ejecutarse en un sistema determinado.

Esto es debido fundamentalmente a la elevada carga computacional que llevan asociada los algoritmos de procesado de imágenes. Recientemente los dispositivos con hardware reconfigurables están siendo una alternativa en el procesado de imágenes debido a su bajo precio, así como sus altas prestaciones. Sin embargo, este tipo de dispositivos presenta grandes problemas desde el punto de vista de implementación de algoritmos, ya que las herramientas para su sintetizado no están desarrolladas todo lo deseable ni están estandarizadas al día de hoy.

Otras alternativas al procesado de imágenes, además de la PC cuyo uso está ampliamente extendido, pueden ser el empleo de estaciones de trabajo en paralelo (procesamiento paralelo), unidades de procesamiento grafico (GPU – *Graphical Processing Units*), procesadores de señales digitales (DSP – *Digital Signal Processors*) o circuitos integrados de aplicaciones específicas (ASIC - *Application-Specific Integrated Circuits*), entre otros.

Una GPU es una CPU dedicada exclusivamente al procesamiento gráfico, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos. De esta forma, mientras gran parte de lo relacionado con la gráfica se procesa en la GPU, la CPU puede dedicarse a otro tipo de cálculos.

El procesamiento con GPU hace posible usar la tarjeta gráfica para este fin, en vez de la CPU. Esto puede acelerar el procesamiento, ya que las GPU modernas están diseñadas para realizar un gran número de cálculos numéricos. Por otro lado, éstas también tienen algunas limitaciones para procesar escenas complejas, debido a su limitada memoria y a la interactividad cuando se usa la misma tarjeta gráfica para el monitor y para el procesamiento.

El uso de los DSP en el procesado de señales digitales es la alternativa más empleada. Las diferentes plataformas basadas en ellos proporcionan una buena autonomía, así como una elevada tasa de operaciones por unidad de tiempo. Esto es debido a la arquitectura interna de la CPU cuyas características se adaptan perfectamente al procesado de imágenes. Otra gran ventaja de esta solución es la posibilidad de desarrollar diferentes algoritmos en lenguajes de alto nivel, dada la existencia de múltiples compiladores con elevado grado de prestaciones. No obstante, su naturaleza secuencial y de propósito específico no permite en muchas ocasiones alcanzar un procesamiento en tiempo real con estos dispositivos.

Otra alternativa al procesado de imágenes, es el uso de dispositivos hardware de propósito específico (ASIC). Este tipo de circuitos pueden proporcionar soluciones particulares a determinados algoritmos de procesamiento de imágenes. No obstante, presentan algunas desventajas que pueden resumirse como:

- Esta solución puede tener un costo elevado si el volumen de elementos necesarios no justifica los costos de fabricación.

- El diseño y desarrollo de este tipo de circuitos requiere un tiempo excesivo tanto desde el punto de vista de validación del circuito, como por su fabricación. Esta causa demora notablemente los proyectos, por lo que actualmente se está trabajando en el desarrollo de herramientas que realicen la implementación física automáticamente, partiendo de una *netlist* como la usada por las herramientas de emplazamiento de los FPGA.
- Si bien el rendimiento es óptimo, ya que se diseñan para una aplicación concreta, la falta de flexibilidad para modificar los algoritmos implementados constituye una desventaja.

Con objeto de evitar la rigidez de los circuitos ASIC, actualmente la alternativa hacia la que se están dirigiendo numerosas líneas de investigación es la de emplear dispositivos con mayor flexibilidad y que al mismo tiempo mantengan las ventajas de los elementos ASIC en cuanto a rendimiento. Un ejemplo de estos son los dispositivos FPGA, los cuales permiten reconfigurarse o reprogramarse tantas veces como se desee y de forma sencilla (Fig. 2.1).

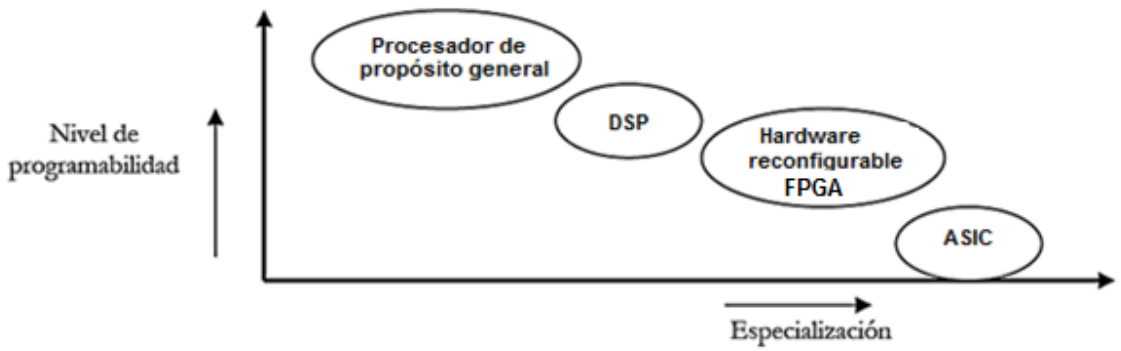


Figura 2.1. Relación entre nivel de especialización y el nivel de programabilidad, para diferentes sistemas de procesado de imágenes [4].

En la Figura 2.1 se muestra la relación existente entre la elección de un sistema que posea ciertas facilidades para poder programarse (programabilidad), con respecto al nivel de especialización del sistema, es decir, la flexibilidad en cuanto a las posibilidades de variar la funcionalidad de un sistema. La opción de hardware reconfigurable ofrece buenos resultados para los diferentes índices evaluados. Una de las grandes características de este tipo de sistemas, es la posibilidad de ejecutar algoritmos de forma concurrente y no de manera secuencial. Esto aplicado a algoritmos de procesado de imágenes, que permitan particionar su ejecución, hace que este tipo de arquitecturas sea muy propicio para la ejecución de estas aplicaciones. A ello se le pueden añadir las tendencias actuales de introducir módulos de procesado de señal dentro de los FPGA, por ejemplo, unidades aritméticas de alta velocidad o *IP* (*Intellectual Properties*), con lo que el empleo de esta opción se acentúa aún más.

Dentro del procesamiento de imágenes con dispositivos reconfigurables el número trabajos realizados hasta hace unos años era reducido, pero poco a poco ha ido en aumento. Esto es debido principalmente a dos razones: por una parte, a que la aparición de estos dispositivos es reciente (aproximadamente hace unos veinte años), y por otra, la falta de herramientas de programación de alto nivel. No obstante, el hecho de que muchos algoritmos de procesado de imágenes permiten particionar su ejecución secuencial, facilita

el que se puedan diseñar diferentes elementos de procesado que puedan funcionar en paralelo, acelerando así su tiempo de ejecución.

Tabla 2.1 Comparación entre DSP, ASIC, hardware reconfigurable (FPGA) y procesadores de propósito general [4].

	Rendimiento	Costo	Consumo	Flexibilidad	Esfuerzo de diseño
ASIC	Alto	Alto	Bajo	Baja	Alto
DSP	Medio	Medio	Medio	Media	Medio
Procesador de propósito general	Bajo	Bajo	Medio	Alta	Bajo
Hardware reconfigurable (FPGA)	Medio	Medio	Alto	Alta	Medio

2.1.2. Estado del arte

Desde la aparición de los FPGA a finales de la década de los ochenta, su uso en el procesado de imágenes ha ido incrementándose continuamente. Inicialmente, los primeros trabajos de procesamiento con FPGA usaban como plataforma una máquina genérica (*Custom Machine*) [5] [6] [7]. De manera simultánea, otros trabajos como los expuestos en [2] y [8], comenzaban a explotar las características de estos dispositivos en algoritmos sencillos de preprocesamiento de imágenes.

La evolución tecnológica de los FPGA provocó la desaparición paulatina de las *máquinas genéricas* construidas con múltiples FPGA, en favor de sistemas con una única FPGA funcionando como coprocesador. Así, éstas coexistían con una PC o un DSP, encargándose estos últimos de manejar las operaciones computacionales complejas para dejar al FPGA el procesado de bajo nivel. Este tipo de plataformas sigue estando aún muy vigente.

2.1.2.1. Máquinas genéricas

Con la aparición a principios de los años 90 de los FPGA de capacidad media, en cuanto al número de recursos internos y con una cierta complejidad en su estructura interna, se crearon las primeras máquinas reconfigurables (SPLASH 2, G-800, PIPS, etc.) [9].

La mayoría de estas máquinas no fue diseñada con un propósito específico, sino que fueron creadas con la idea de ser empleadas en varias aplicaciones y áreas de trabajo. Una de estas áreas fue la del procesado de imágenes. La posibilidad de ejecutar en paralelo algoritmos de este tipo permitiría alcanzar el grado de tiempo real necesario en muchas aplicaciones, el cual era difícil de conseguir con plataformas convencionales de propósito general.

Por esta última razón se crearon a finales de la década de los 80 las denominadas Máquinas CCM (*Custom Computing Machine*), con base en tarjetas compuestas de varios FPGA. Estas máquinas son sistemas “embebidos” que pueden ser usados para construir arquitecturas especializadas en diferentes aplicaciones. Los FPGA revolucionaron este campo, ya que gracias a su reprogramabilidad permitían disponer de una máquina de propósito general, que se podía utilizar en diferentes áreas sin necesidad de cambiar la plataforma hardware.

SPLASH 2 fue una de las máquinas más importantes que se desarrollaron con arquitectura genérica y no exclusivamente orientada al procesamiento de imágenes. Esta máquina constaba de un arreglo de FPGA XC4010, y surge como evolución de la *SPLASH 1*, que estaba basada en dispositivos de la familia XC3000. La arquitectura *SPLASH 2* estaba conectada al *host* a través de una interfaz que expandía los buses. El *host* (una máquina SUN) podía leer/escribir en memoria y mapear los registros de control de la *SPLASH 2* gracias a estos buses expandidos. Cada tarjeta de procesado *SPLASH 2* tenía 16 FPGA 4010 que funcionan como elementos de proceso. Además, un FPGA adicional controlaba el flujo de datos dentro de la tarjeta. Cada tarjeta poseía 512 KB de memoria RAM.

Con esta plataforma se desarrollaron varios trabajos de procesado de imágenes, con diferente complejidad computacional. En [1] se presentan algunos de ellos ordenados acorde al nivel de complejidad algorítmica. Así, dentro de los trabajos que pueden clasificarse como de bajo nivel, se implementaron operaciones de convolución y erosión/dilatación. De nivel medio, se implementó la segmentación de imágenes y de alto nivel se realizó el pareo (*matching*) aplicado a la detección de huellas dactilares.

DFFC (*Data Flow Functional Computer* – Computadora funcional de flujo de datos)***: Es una plataforma pensada para el procesamiento de imágenes.*** Fue diseñada para funcionar como plataforma de propósito general para algoritmos de visión y tenía como objetivo trabajar en tiempo real en operaciones morfológicas con imágenes [10] [11].

Esta plataforma estaba formada por un conjunto de FPGA especialmente desarrollados para este sistema, denominados FPOA (*Field Programmable Operator Array* – Arreglo de operadores ce campo programable). Las FPOA tenían una arquitectura similar a la de las FPGA híbridas (intentan combinar los beneficios de los FPGA y los Dispositivos Lógicos programables o PLD (por sus siglas en inglés, *Programmable Logic Device*). Estos circuitos ASIC fueron construidos con un potente bloque de interconexiones para poder asociarse en estructuras tridimensionales.

La máquina DFFC estaba formada por un arreglo de 8 filas x 8 columnas x 8 tarjetas FPOA.En él se podía conectar directamente a la red de FPOA las entradas y salidas de vídeo digitalizadas. Una estación de trabajo SPARC2 se encargaba de la programación de las FPOA.

Para poder trabajar con la máquina DFFC se diseñaron bibliotecas de programación de bajo y mediano nivel, que posteriormente se compilaban a un formato por el hardware reconfigurable. Las bibliotecas de bajo nivel estaban compuestas por un conjunto de primitivas o macros que contenían operadores lógico-aritméticos, multiplexores, contadores, etc. Por su parte, las de nivel medio contenían agrupaciones de primitivas que formaban elementos para el procesado con una complejidad mayor. Dentro de este grupo estaban las LUT (*Look Up Tables*) y las máscaras de convolución o histogramas.

La versatilidad aportada con estas bibliotecas posibilitó el desarrollo de aplicaciones de filtrado, detección de bordes o seguimiento de un objeto de color dentro de una escena. El principal problema de esta máquina era el excesivo número de recursos hardware consumidos para poder ejecutar algoritmos de complejidad baja [10].

Virtual Computer**:** Fue una máquina desarrollada al inicio de los años noventa por la empresa *Virtual Computer Corporation*. Esta máquina surgió como alternativa al procesado de algoritmos complejos ejecutados sobre PC, teniendo presente la idea de que cualquier algoritmo puede alcanzar su máxima velocidad si es implementado en hardware [12].

Internamente estaba formado por 52 unidades FPGA (Xilinx 4010), 24 unidades FPID (*Field Programmable Interconnect Devices,* I-CUBE IQ160), memoria SRAM (8 MB) y Dual-Port (16K x 8K x 16 bits).

Con respecto a su utilización, inicialmente fue diseñado como una plataforma académica genérica, donde a partir de una biblioteca codificada en C++ se proporcionaba un conjunto de funciones de medio nivel (convoluciones, operaciones morfológicas, etc.). El usuario podía trabajar en alto nivel o bajo nivel ya que una herramienta de síntesis/implementación de propósito específico se encargaba de generar el *bitstream* correspondiente.

Máquinas Monotarjetas. Posterior a la aparición de las máquinas de propósito general, las cuales contenían un gran número de dispositivos FPGA, aparecieron nuevos trabajos donde el objetivo era la implementación de máquinas más simples que las de propósito general. Para ello se construyeron máquinas formadas por una única tarjeta de FPGA.

De esta forma se conseguía reducir notablemente los recursos consumidos, con lo que el costo disminuía. Sin embargo, esa disminución de recursos se traducía en un potencial de cálculo inferior a las de propósito general. Por ello, su ámbito de actuación se centró en aplicaciones de procesamiento de bajo nivel. Estábamos ante las primeras tarjetas coprocesador, donde se delegaba la complejidad computacional a un *host* al que estaba conectado la tarjeta.

Ejemplos de esta línea de trabajo fueron la tarjeta WILDFORCE o SPECTRUM RC. WILDFORCE fue una tarjeta de propósito general, diseñada por la empresa Annapolis Micro Systems. Poseía un conector PCI (*Peripheral Component Interconnect* - Interconexión de Componentes Periféricos) con lo que dicha tarjeta podía ser insertada en una PC. Inicialmente el sistema fue diseñado con un FPGA de Xilinx, de la familia 4000, concretamente el XC4013, permitiendo desarrollar algoritmos de visión de bajo nivel (filtrado y convoluciones) con lo que se alcanzó un alto grado de rendimiento. Posteriormente, se diseñó una herramienta de particionado y síntesis que permitía el diseño inicial en un lenguaje de alto nivel y su posterior traslado al FPGA. Ello posibilitó la implementación de algunos algoritmos de complejidad media, como la compresión de imágenes mediante la técnica JPEG (*Joint Photographic Experts Group* - Grupo conjunto de expertos en fotografía).

2.1.2.2. Los FPGA como coprocesadores

Esta nueva configuración se caracterizó principalmente por la coexistencia de un procesador secuencial, típicamente una PC o un DSP, encargado de realizar operaciones aritméticas complejas y de un reducido número de FPGA (típicamente entre 1 y 3). El hecho de existir un procesador secuencial, provocaba el funcionamiento del FPGA como un dispositivo esclavo. Así, el procesador se liberaba de carga computacional, delegando ciertas tareas al FPGA. Es por esta razón que el nombre adoptado para estos sistemas reconfigurables sea el de tarjetas con coprocesador.

Algoritmos simples, como filtrados o convoluciones, eran aplicaciones habituales de estos sistemas. Algunos ejemplos de este tipo de operaciones se muestran en [2], donde se realizan con FPGA operaciones de cálculo de medias aritméticas sobre imágenes de resolución espacial de 256 x 256 pixeles. Otro ejemplo es el mostrado en [8], en el que la tarjeta con FPGA se introducía como un elemento de preprocesado (media, convolución o filtrado) dentro de un sistema completo de captura y procesamiento de imágenes.

En [13] se muestra el diseño de un multiplicador por hardware implementado en un FPGA para realizar convoluciones sobre imágenes. Dentro de esta línea existen numerosos trabajos como los mostrados en [14] [15] [16], donde se realizan aplicaciones de máscaras

de visión (normalmente de 3x3), sobre imágenes monocromáticas de resolución espacial de 256x256 pixeles.

Las tarjetas que alojan los FPGA solían disponer de buses estándar de comunicación para conectarse con el procesador central. Esto permitía una cómoda expansión en el número de tarjetas.

La complejidad que desarrollaba el sistema compuesto por las tarjetas de FPGA y el *host* central solía ser elevada, debido a la partición de las tareas entre el procesador y el FPGA. Un ejemplo de estos trabajos es el mostrado en [17], el cual constaba de una tarjeta con 4 FPGA y un bus PCI para comunicarse con una PC. Este diseño estaba orientado a la reconstrucción 3D de imágenes médicas.

Otras tarjetas interesantes en las que se han experimentado técnicas de preprocesado digital son las desarrolladas en [18] [19] [20]. Por su parte, los trabajos desarrollados en [21] [22] [23], muestran plataformas que combinan DSP con FPGA. En estos últimos casos, la complejidad algorítmica que se alcanzaba era mayor que en los trabajos previos. Esto era debido al reparto de tareas hardware (FPGA)/software (DSP) y a la alta tasa de transferencia de datos alcanzada.

Varios son los problemas que este tipo de arquitecturas tenían:

a) *Separación de las tareas*: La división de las tareas que debía ejecutar el *host* y el dispositivo reconfigurable era un problema importante. En función de la tarea, cada una de las plataformas era óptima para realizar una u otra operación. Ello implicaba un alto grado de sincronización entre las dos plataformas, complicando notablemente el diseño. Actualmente, la aparición de técnicas y herramientas de codiseño permiten separar de manera óptima estas facetas.
b) *Programación de dos dispositivos diferentes*: El hecho de disponer siempre de un *host* (PC, DSP, etc.) y de un dispositivo reprogramable (por ejemplo un FPGA) conllevaba la programación de dos elementos con lenguajes y métodos diferentes. Esto provoca un doble grado de especialización.

2.1.2.3. Implementación de algoritmos de procesamiento de imágenes sobre FPGA

Con la aparición de las últimas familias de FPGA a principios del año 2000 y hasta el día de hoy, se incorporaron nuevos recursos internos orientados hacia el procesamiento de imágenes. Esto ha provocado que dentro de esta área, se haya producido un incremento en el desarrollo de algoritmos de visión de alto nivel. Sirvan como ejemplo los trabajos mostrados en [22] [23]. En ellos se desarrollaban plataformas hardware orientadas a la segmentación de imágenes mediante la Transformada de Hough. No obstante, la ejecución [35] de dicha transformada no era ejecutada íntegramente dentro del FPGA, sino que también se usaba un DSP. En [24] se propone una alternativa eficaz dónde se realiza íntegramente el cálculo de esta transformada en un FPGA.

La aparición de nuevos lenguajes de programación para FPGA con un nivel de abstracción mayor que VHDL o VERILOG, como pueden ser Okapi, Impulse-C, System-C y sobre todo Handel-C, han permitido trasladar algoritmos codificados en lenguajes de alto nivel para PC a hardwares reconfigurables. En la mayoría de estos trabajos la optimización del sistema se basa en la partición del algoritmo en elementos que puedan ejecutarse en paralelo, o bien, en el diseño de arquitecturas sistólicas donde los elementos de procesado pueden trabajar en paralelo. Un ejemplo de esta readaptación puede ser el mostrado en [25]. En este caso, se presentaba una arquitectura basada en FPGA orientada a la obtención de flujo óptico, alcanzando velocidades de detección de movimientos de hasta

60 marcos/s (*frames/s*) para imágenes de 256x256 pixeles de 8 bits de resolución de nivel de gris.

Por último, dentro del campo del flujo óptico, cabe destacar los trabajos desarrollados en [26]. En ellos se ha desarrollado un sistema completo de captura y procesado de imágenes basado en un FPGA Virtex-II de 6 millones de compuertas. La arquitectura propuesta ha sido orientada a la detección de vehículos por el espejo retrovisor de un automóvil. Mediante este sistema se ayuda a evitar accidentes de tráfico por invasión de un carril, cuando éste está ocupado por otro vehículo. Con este sistema se consigue una alta velocidad de procesado de imágenes en un automóvil en movimiento.

2.2. Segmentación del iris

Con el motivo de satisfacer una creciente demanda en cuanto a sistemas de seguridad, la identificación personal basada en características biométricas ha estado recibiendo gran atención durante las últimas décadas. Los sistemas biométricos tienen como objetivo la exacta identificación de cada individuo mediante diferentes características fisiológicas o de comportamiento, tales como huellas digitales, rostro, patrón de escritura, iris, retina, geometría de la mano, etc. Debido a las características únicas, estables y accesibles del patrón de iris, la identificación personal basada en el patrón de iris se ha convertido en una de las técnicas más confiables [36]. Recientemente, han sido desarrollados varios trabajos para la segmentación exitosa del iris debido a su importancia, ya que si el iris no es correctamente localizado, las etapas posteriores utilizarán datos erróneos, el código generado contendrá errores y el rendimiento del sistema será muy bajo.

2.2.1. Antecedentes

Como se ha demostrado a lo largo de numerosos estudios, los ojos representan una fuente confiable de patrones para la identificación biométrica de personas, tanto mediante el análisis de la retina como del iris. Si indagamos un poco en los antecedentes del reconocimiento biométrico basado en patrones oculares (iris, retina, etc.) encontramos que el primero de estos métodos fue un sistema fundamentado en el análisis de la retina patentado por Robert Hill en 1978 [45]. Este sistema hacía uso de imágenes oftalmoscópicas de los patrones internos de distribución de la formación vascular del fondo del ojo. A pesar de que el color característico del ojo de una persona fue usado como identificador en el siglo XIX, por el físico francés Alphonse Bertillon [47], la idea de que la compleja disposición de patrones del iris pudiera usarse como una especie de huella óptica, fue propuesta por primera vez por Frank Burch, un importante cirujano ocular y oftalmólogo de St. Paul, en el año 1936, quien sugirió en uno de sus discursos dirigido a sus colegas de profesión en el congreso anual de la *American Academy of Ophthalmology* [47]. El concepto que se presentaba en aquel entonces, se reprodujo en diversos libros de texto y la idea fue igualmente usada en películas de ciencia ficción, pero no se produjo ningún progreso en más de medio siglo. Fue en 1987 cuando dos oftalmólogos, Leonard Flom y Aran Safir [48], rescataron la idea patentándola y presentándosela dos años más tarde al profesor por aquel entonces de la Harvard University John G. Daugman, con el objetivo de crear los algoritmos necesarios para desarrollar correctamente un sistema eficaz de reconocimiento de iris. Pese a su duda inicial a participar en el proyecto, debido a su inmediato traslado a la Universidad de Cambridge, los dos oftalmólogos consiguieron atraer la atención del profesor Daugman, presentando sus ideas a través de fotografías del iris de diversos pacientes recogidas en su clínica. En estas fotografías, los iris presentaban complejos patrones aleatorios, creados por ligamentos arqueados, surcos, criptas, anillos, diversidad

de pecas, una corona y un característico zigzag denominado collarete [46]. En 1994, se concedió al Dr. Daugman una patente para sus algoritmos automatizados de reconocimiento de iris [37].

Los trabajos realizados por Daugman, que se recogen parcialmente en su trabajo titulado *High Confidence visual recognition by test of statistical independence* [36], supusieron un avance definitivo en el campo de la identificación personal basada en los patrones de iris. Los algoritmos que desarrolló posteriormente fueron patentados en 1994 (*Biometric Personal Identification System Based on Iris Analysis*) [37], sentando las bases de los actuales sistemas y productos de reconocimiento de personas mediante el iris. Este hecho dio lugar a que posteriormente Aran Safir y Leonard Flor, fundaran conjuntamente con John Daugman la compañía Iris Corporation, poseedora de la patente y encargada de distribuir las licencias a las diferentes compañías desarrolladoras e integradoras de sistemas de reconocimiento que quisiesen hacer uso del patrón del iris. Una de estas empresas es Sensar Corporation, que creó una cámara especial capaz de obtener imágenes de iris en los cajeros automáticos. Estas dos compañías (Iriscan Corp. y Sensar Corp.) se unieron creando Iridian Technologies. En 2005, la amplia patente que cubría el concepto básico de reconocimiento del iris expiró, proporcionando oportunidades comerciales a otras compañías que han desarrollado sus propios algoritmos para el reconocimiento del iris [53].

Agencias relacionadas con los cuerpos de seguridad y justicia de los Estados Unidos comenzaron a utilizar este tipo de dispositivos en 1994, siendo la prisión del estado de Lancaster (Pennsylvania) la pionera en el uso de esta tecnología para la identificación de los reclusos [53]. Otro ejemplo puede encontrarse en el condado de Berkshire, donde la prisión estatal utiliza sistemas de reconocimiento basados en el iris para el control de acceso y seguridad de sus empleados. En un entorno más comercial y enfocado al público en general, se da en el aeropuerto internacional Charlotte/Douglas en Carolina del Norte o en el aeropuerto alemán de Frankfurt, donde se permite a los viajeros habituales registrar su patrón de iris con el objetivo de reducir al máximo posible el tiempo necesario dada la complejidad del proceso de embarque [53]. También, dentro del campo de la seguridad aeroportuaria, tenemos los ejemplos de las instalaciones de los aeropuertos Schiphol en Ámsterdam, JFK en Nueva York y Heathrow en Londres, donde son empleados tanto para realizar el embarque de viajeros como en los controles de pasaportes e inmigración. Una de las aplicaciones relacionadas con los patrones de iris que está experimentando un estudio más exhaustivo, es la utilización de estos dispositivos en los cajeros automáticos. También se considera muy interesante la utilización de estos sistemas de reconocimiento biométrico para el control de acceso y seguridad en entornos demóticos. Entre las empresas que desarrollan este tipo de dispositivos y sistemas pueden destacarse Panasonic, LG, Oki, British Telecom, Unisys, Siemens o IBM.

2.2.2. Estado del arte

Son dos los procedimientos básicos para llegar a la normalización del iris, el primero es la localización y segmentación del iris, donde lo que se busca es la localización del iris en la imagen y su separación del resto de la información; y el segundo es su normalización en sí, es decir, disponer en una imagen de la información del iris de una forma adecuada para los procedimientos posteriores.

2.2.2.1. Localización y segmentación del iris

Para la localización del iris se han propuesto diferentes aproximaciones, unas con mayor fundamento teórico y otras más empíricas, pero ambas con resultados muy satisfactorios.

Se aplicaron en [28] operadores integro-diferenciales para identificar bordes circulares en las imágenes, los cuales resultaron útiles para detectar los límites interno y externo del iris. Este método tiene en cuenta la geometría del iris (forma circular) para encontrar su correcta ubicación, acompañado de una maximización de la derivada parcial con respecto al radio *r* que se incrementa progresivamente. Luego, se hizo una búsqueda más fina para localizar el borde de la pupila. Este método se aprovechó de las ventajas que brinda las diferencias en el tono entre las partes involucradas.

En [27] se localizó el iris a través de operaciones de filtrado e histogramas simples. Primero se aisló el iris con filtros Gaussianos pasa bajas seguidos de un submuestreo espacial. Posteriormente, la segmentación se ejecutó utilizando un procedimiento muy simple bastante similar a la transformada generalizada de Hough. En [29] se empleó un filtrado de tipo diferencial pasa altas con convoluciones Laplacianas o Gaussianas, que brindaba información de la variación de la intensidad en la imagen.

Se utilizó en [27] un método similar al [32], pero más eficiente, mediante el uso de una estrategia de identificación del paso de una textura gruesa a fina en los bordes, para su proceso de búsqueda del contorno. Para lograr esto se aplica el reescalado de la imagen, filtrado y extracción del bordes con un operador Canny, hasta conformar una imagen binaria; luego, se localizan los bordes usando un operador integro-diferencial sobre los bordes gruesos hallados previamente. En [31] fue empleado un método basado en un filtrado simple de detección de bordes y la transformada de Hough que proporcionan resultados eficientes y seguros.

Otras aproximaciones interesantes desarrolladas para la localización del iris son las referidas en [30] y [33]. En el primero de ellos todos los componentes del ojo (pupila, párpados, ojo) fueron segmentados usando un método de estimación paramétrico, detectando los límites del iris con los valores de la distribución de intensidad de la imagen, la cual aparece como una mezcla de tres distribuciones Gaussianas (oscuro, intermedio y brillante), cuyos parámetros se estiman usando el algoritmo EM (*Expectation-Maximization*, Expectación-Maximización) [30]. Por su parte, en [33], fue empleado un algoritmo basado en la apariencia para la detección del iris, detectando primero la pupila, la cual se asume como más oscura que el iris (particularmente en ojos claros) y utilizando posteriormente una SVM (*Support Vector Machine* - Máquina de Soporte Vectorial) con 12 vectores que irradiaban desde el centro de la pupila, los cuales se ingresaban a una red de RBF (*Radial Basis Function* - Función de base radial), para aprobar o no la existencia del iris en la imagen. Posiblemente, este método podría no funcionar bien en los ojos oscuros, donde la pupila no se diferencia mucho del iris.

2.2.2.2. Representación estructural del iris

Para la codificación del patrón del iris, usualmente se venía realizando una conversión de la imagen del iris de coordenadas cartesianas a polares para facilitar la extracción de información, al pasar de una forma circular a una rectangular, tal como se observa en la figura 2.2. A la nueva representación, la mayoría de los autores aplican filtros multicanal de Gabor, Fourier o Wavelet, para extraer los coeficientes que finalmente conformaran el código del iris.

Estas imágenes transformadas del iris son entonces codificadas, en donde los bits más significativos conforman el código bidimensional denominado IrisCode (figura 2.3), con un tamaño de 256 bytes (después de la aplicación de los filtros bidimensionales de wavelets tipo Gabor y la posterior compactación de los coeficientes). Estos brindan información de alta resolución sobre la orientación y la frecuencia espacial de la estructura del iris.

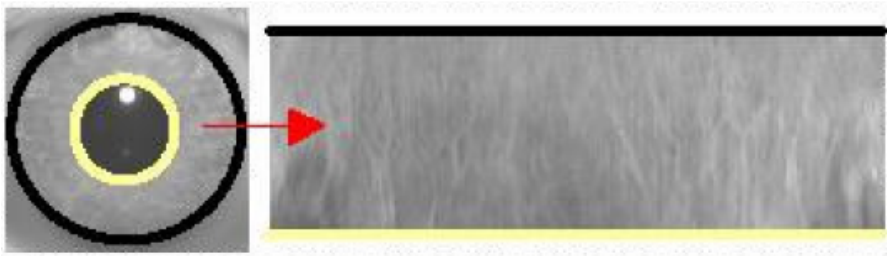


Figura 2.2. Transformación del iris a coordenadas cartesianas.

En [27] se consideró que la información distintiva se encuentra tanto en el área completa del iris, como en las áreas pequeñas, por lo cual se propuso una descomposición multiescala piramidal Laplaciana para representar las características distintivas espaciales del iris humano. En [29], por su parte, el centro de la pupila fue escogido como punto de referencia y a partir de allí se construyeron círculos concéntricos para extraer la información de cada franja circular del iris, utilizando los niveles de gris de las imágenes para obtener señales unidimensionales (1D) que se convierten de esta forma en la firma del iris. Posteriormente, se calcula la representación de cruces de ceros con base en la transformada Wavelet, cuyos coeficientes resultantes conforman el patrón en el proceso de emparejamiento. De esta forma, el autor expone que su método está libre de la influencia de ruidos, debido a que los cruces de ceros no son afectados por estos. Por otro lado, las transformaciones unidimensionales arrojan un menor número de cruces, lo cual podría acelerar la velocidad del proceso. Un aspecto interesante de esta propuesta fue la habilidad de la transformada Wavelet para eliminar el efecto de los destellos producidos por la reflexión de la fuente de luz sobre la superficie del iris, aspecto que no había sido resuelto por las anteriores propuestas.

Se empleó una FFT (*Fast Fourier Transform* - Transformada Rápida de Fourier) en [34] para la extracción de la información, aunque se consideró adicionalmente que la transformada Wavelet podría arrojar mejores resultados. Éste utilizó una conversión en espiral logarítmica a intervalos de 50 pixeles, donde los picos que se observaron en las bandas más externas fueron producidos por los párpados, como se observa en la figura 2.4.

En [30], los autores consideraron que la información de alta frecuencia del iris era sensible a los ruidos, por lo cual usaron los componentes de baja frecuencia en dirección radial y los de frecuencia baja a media en dirección angular, permitiendo mayor robustez frente al ruido. En [35] se empleó un método basado en ACI (Análisis de Componentes Independientes). En primer lugar fueron calculadas las componentes independientes para cada ventana de tamaño N en la representación rectangular del iris, luego se estimaron los coeficientes para cada ventana, se cuantificaron y finalmente fue construido el código del iris con todos los coeficientes de cada ventana. También utilizaron un mecanismo de aprendizaje competitivo para determinar el centro de cada clase que puede almacenarse.

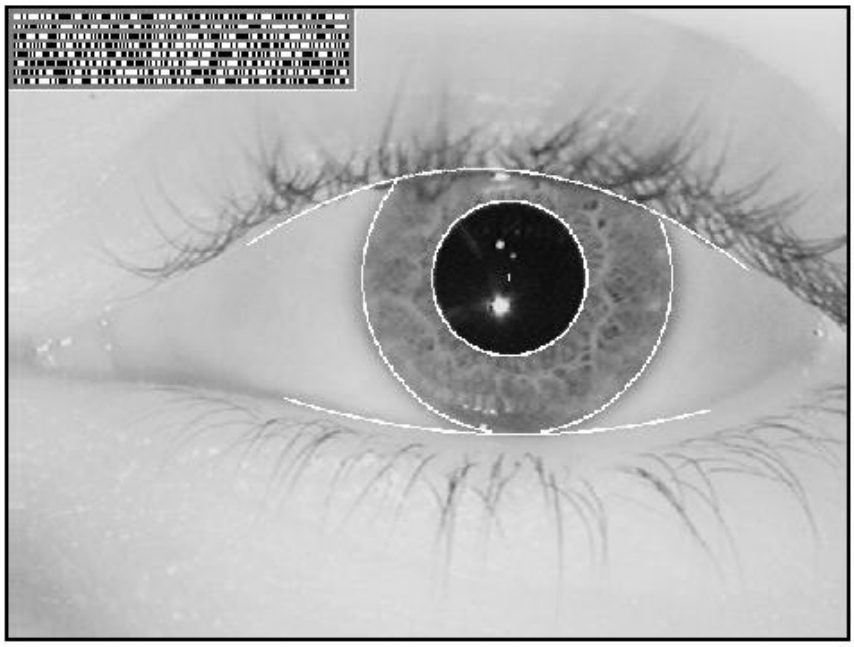


Figura 2.3. Iris localizado mediante IrisCode.

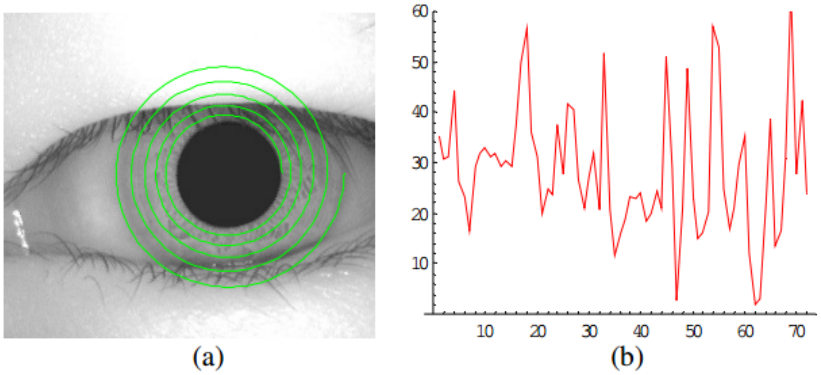


Figura 2.4. Método de espiral logarítmica para la conversión del sistema de referencia.

CAPÍTULO III

Marco teórico

En este capítulo se brindan brevemente los conceptos básicos y las definiciones necesarias para comprender adecuadamente este trabajo. Se ha dividido en tres secciones según los tres campos fundamentales que se tratan. La sección 3.1 contempla la disciplina de procesamiento digital de imágenes; en la sección 3.2 se tratan las generalidades de la conformación del ojo, en especial el iris y las imágenes que de él se obtienen; en la sección 3.3 se tratan las generalidades y fundamentos de los FPGA, así como las características de la tarjeta de desarrollo FPGA utilizada en este trabajo.

3.1. Procesamiento digital de imágenes

El procesamiento digital de imágenes abarca el conjunto de técnicas que se aplica a las imágenes digitales con el objetivo de mejorar la calidad, hacer más evidentes en ellas ciertos detalles que se desean hacer notar o facilitar la búsqueda de información dentro de las imágenes. La imagen puede haber sido generada de muchas maneras, por ejemplo, fotográfica o electrónicamente, por medio de monitores de televisión. El procesamiento de las imágenes se puede llevar a cabo por medio de métodos ópticos, o bien por medio de métodos digitales en una computadora.

Para que una imagen analógica sea útil se requiere que sea muestreada y cuantificada. Estas operaciones se realizan normalmente por medio de dispositivos conocidos como digitalizadores o cuantificadores. El resultado de digitalizar una imagen según la definición anterior da como resultado una imagen digital [38].

3.1.1. Representación y definición de una imagen

Una imagen digital puede ser definida como una función de dos dimensiones, $f(x, y)$, donde *x* y *y* son las coordenadas espaciales y el valor de *f* en un punto *(x, y)* es llamado intensidad, valor o nivel de gris de la imagen en ese punto.

La imagen digital se puede considerar como una matriz cuyos índices de filas y columnas identifican un punto de la imagen y el valor del elemento correspondiente. Los miembros de una distribución digital de este tipo se denominan elementos de la imagen o más comúnmente pixeles (proviene del término en inglés *picture element*).

Por defecto, la representación espacial de un píxel con coordenadas (0,0) es localizado en la esquina superior izquierda de la imagen. En nuestro caso el valor de *x* se incrementa de izquierda a derecha y el valor de *y* de arriba hacia abajo (figura. 3.1) [39].

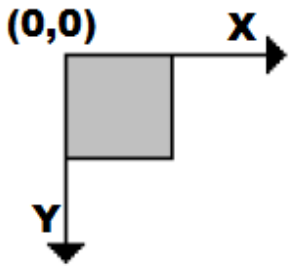


Figura 3.1. Representación de la convención de imágenes digitales.

3.1.2. Propiedades de las imágenes digitales

Hay tres conceptos íntimamente relacionados con una imagen digital: la resolución espacial, la resolución de niveles y el número de planos.

3.1.2.1. Resolución espacial

La resolución espacial de una imagen es el número de pixeles por unidad de longitud. Una imagen de *m* filas y *n* columnas tiene un total de *n x m* pixeles. La resolución espacial se relaciona con los detalles que pueden hacerse visibles en una imagen: mientras mayor sea la resolución espacial, menor será el área representada por cada píxel en una imagen digital y mayor será el número de detalles que puede ser apreciado en la misma. El píxel representa el detalle más pequeño discernible en una imagen.

3.1.2.2. Resolución de niveles

La resolución de niveles de una imagen digital, también llamada profundidad del píxel, indica el número de bits por píxel que brinda el número de niveles de gris que pueden apreciarse en la imagen. La profundidad del píxel es el número de bits usado para definir la intensidad que representa. Para una cantidad de bits *n*, el píxel puede tomar 2^n valores diferentes. Por ejemplo, si *n* es igual a 8 bits, un píxel puede tomar 256 valores distintos en el rango de 0 a 255. Representa el cambio más pequeño discernible de entre los niveles de gris que conforman la imagen.

3.1.2.3. Planos de una imagen

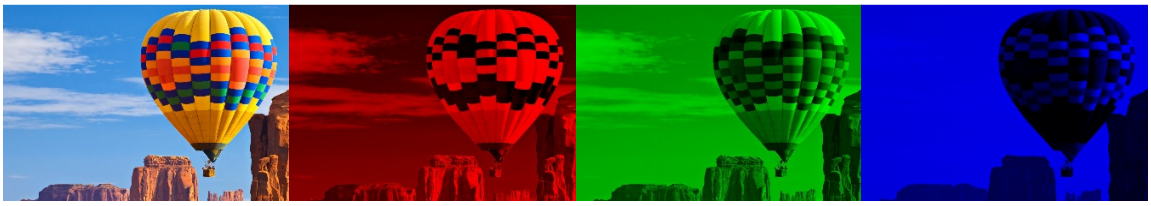
El número de planos en una imagen es el número de arreglos de pixeles que la componen.

Una imagen en tonos de gris está compuesta de un solo plano (en realidad son tres planos, pero iguales), mientras que una imagen de color verdadero (*true color*) está compuesta por tres planos: el de la componente roja, la verde y la azul.

3.1.3. Imágenes en colores

El ojo humano, al tener tres receptores de color distintos (los conos), uno para el rojo, otro para el verde y otro para el azul, es capaz de captar una gran variedad de colores diferentes. Además, posee un tipo de receptor de intensidad: los bastoncillos.

Una imagen RGB es una imagen multibanda, cada una de las cuales está asociada a los colores rojo (R), verde (G) y azul (B) (figura 3.2).



(a) *(b)* *(c)* *(d)*

Figura 3.2. (a) Imagen en colores en el espacio RGB. (b) Plano rojo. (c) Plano verde. (d)Plano azul.

3.1.4. Imágenes en tonos de gris

Si las correspondientes intensidades de la componente roja, verde y azul de cada píxel en una imagen son iguales, se forma una imagen acromática en tonos o en niveles de gris.

Una imagen en niveles de gris es una imagen bidimensional donde cada píxel sólo representa un valor de intensidad acotado entre 0 y $2^n - 1$, donde *n* es la cantidad de bits utilizados para representar cada uno de los valores de intensidad. Por conveniencia los valores extremos de este rango representan el negro y el blanco, respectivamente.

Una forma aproximada de lograr una imagen en niveles de gris se obtiene promediando los valores de las tres componentes R, G y B de todos los pixeles de la imagen en colores.

Formalmente, una imagen digital en niveles de gris es una función bidimensional de la intensidad de luz *f*: $ZxZ \rightarrow Z^2$ cuyos valores se han obtenido muestreando la intensidad sobre una retícula rectangular. Por lo tanto, una imagen digital la denotaremos como *f(x, y)*, donde *x* y *y* son las coordenadas espaciales y el valor de *f* en cada punto (*x, y*) es proporcional a la intensidad de luz de ese punto.

Podemos decir que una imagen *f*(*x, y*) está formada por dos componentes: una es la cantidad de luz incidente en la escena y la otra es la cantidad de luz reflejada por los objetos. Estas dos componentes se llaman: iluminación, que denotaremos por *i*(*x, y*) y reflectancia, que denotaremos por *r*(*x, y*).

3.1.5. Imágenes binarias

Los pixeles en una imagen binaria contienen solo dos valores de intensidad en forma normalizada: 0 para el negro y 1 para el blanco, los que equivalen a los niveles 0 y 255, respectivamente, por estar cada píxel asociado a un byte (Figura 3.3).

(a) **(b)**

Figura 3.3. (a) Imagen en niveles de gris. (b) Imagen binaria.

3.1.6. Relaciones de vecindad y conectividad

En una imagen *f(x, y)*, denotaremos los puntos que la componen mediante las letras minúsculas *p* o *q* y denotaremos por *S* al conjunto de todos ellos.

3.1.6.1. Vecinos de un píxel

Un punto *p* de coordenadas (*x, y*) tiene 4 vecinos verticales y horizontales cuyas coordenadas son: *(x + 1, y), (x – 1, y), (x, y + 1), (x, y - 1)*. A este conjunto de puntos se le llama los 4- vecinos de *p* y lo denotaremos como $N_4(p)$.

Los 4 vecinos diagonales de *p* tienen como coordenadas: *(x + 1, y + 1), (x + 1, y - 1), (x – 1, y + 1), (x – 1, y - 1)* y los denotaremos como $N_D(p)$.

A estos 4 vecinos diagonales junto con $N_4(p)$ les llamaremos los 8-vecinos de *p*, denotado como $N_8(p)$, como se puede observar en la figura 3.4.

(x – 1, y – 1)	(x, y - 1)	(x + 1, y - 1)
(x – 1, y)	(x, y)	(x + 1, y)
(x – 1, y + 1)	(x, y + 1)	(x + 1, y + 1)

Figura 3.4. Coordenadas de los vecinos de un píxel de coordenadas (x, y).

3.1.6.2. Conectividad

Para establecer si dos puntos están conectados debemos determinar si son adyacentes en algún sentido (por ejemplo los 4-vecinos) y si su nivel de gris satisface algún criterio de similitud. Sea *V* el conjunto de niveles de gris usados para definir la conectividad.

Consideramos dos tipos de conectividad:

- 4-Conectividad: dos puntos *p* y *q* con niveles de gris en *V* están 4- conectados si *q* está en el conjunto $N_4(p)$.
- 8-Conectividad: dos puntos *p* y *q* con niveles de gris en *V* están 8- conectados si *q* está en el conjunto $N_8(p)$.

3.1.7. Operaciones lógicas y aritméticas

Las operaciones básicas con imágenes se dividen en dos grupos: lógicas y aritméticas.

Las operaciones lógicas son útiles en la definición de máscaras, detección de características y análisis de formas.

Las operaciones aritméticas son utilizadas generalmente para eliminar ruido e información inútil (sumas y restas), así como para efectuar correlaciones entre imágenes (multiplicación y división).

3.1.7.1. Operaciones lógicas

Se pueden realizar operaciones lógicas AND, OR, XOR, etc. con las imágenes discretas, de tal forma que mediante máscaras y un proceso de convolución, cubrir ciertas áreas de una imagen, efectuar comparaciones, etc. [40]. Para definirlas, sean I_1 e I_2 dos imágenes cualesquiera. Las operaciones se llevan a cabo píxel a píxel.

Intersección (AND). Sobre las imágenes, la operación respeta la lógica convencional y se define de la siguiente manera:

$$O(x,y) = I_1(x,y) and\ I_2(x,y) \qquad \text{(Ec. 3.1)}$$

Para imágenes en niveles de gris se toma el mínimo valor de los pixeles entre los cuales se realiza la operación; es definida de la siguiente manera:

$$O(x,y) = \min\{I_1(x,y), I_2(x,y)\} \qquad \text{(Ec. 3.2)}$$

Unión (OR). Para imágenes binarias, la operación OR se comporta de igual forma a la lógica convencional. Para dos imágenes I_1 e I_2 se define de la siguiente manera:

$$O(x,y) = I_1(x,y)\ or\ I_2(x,y) \qquad \text{(Ec. 3.3)}$$

Para imágenes en niveles de gris, la operación toma el máximo valor de los pixeles involucrados en la operación, por lo que se define como:

$$O(x,y) = \max\{I_1(x,y), I_2(x,y)\} \quad \text{(Ec. 3.4)}$$

Complemento (NOT). Para imágenes binarias, podemos definir esta operación $O(x,y)$ de una imagen de entrada $I(x,y)$ de la siguiente manera:

$$O(x,y) = \text{not } I(x,y) \quad \text{(Ec. 3.5)}$$

Por su parte, en escala de grises el resultado de esta operación (complemento) es igual a la diferencia del máximo valor posible que puede alcanzar un píxel menos el valor del píxel involucrado, definiéndose por lo tanto de la siguiente manera:

$$O(x,y) = (2^n - 1) - I(x,y) \quad \text{(Ec. 3.6)}$$

3.1.7.2. Operaciones aritméticas

Las operaciones aritméticas implican dos imágenes y se efectúan píxel a píxel de la primera imagen con los de la segunda. Las operaciones más comunes son la suma y la resta. Las operaciones aritméticas son relativamente rápidas, pues tan solo se han de realizar *nxm* operaciones, donde *n* es el ancho y *m* es el alto de la imagen en pixeles [40].

Suma (Resta): Podemos definir la suma (resta) de dos imágenes I_1 e I_2 de la siguiente manera:

$$S(x,y) = I_1(x,y) \pm I_2(x,y) \quad \text{(Ec. 3.7)}$$

También podemos sumar (restar) una constante C a una imagen:

$$S(x,y) = I(x,y) \pm C \quad \text{(Ec. 3.8)}$$

La imagen de salida dependerá de la implementación. Puesto que pueden presentarse problemas al obtenerse pixeles fuera del rango dinámico posible, en general hay que ajustar previamente el comportamiento deseado de la operación. Este puede ser limitado, cíclico o escalado.

La diferencia de dos imágenes es útil para detectar cambios producidos en la misma escena después de un determinado intervalo de tiempo, o para eliminar defectos de captura, entre otras aplicaciones.

3.1.8. Etapas del análisis digital de imágenes

La primera etapa en el análisis de imágenes la constituye la captura de la imagen. Para ello se requieren los detectores adecuados que pueden ser una cámara fotográfica de color, una cámara monocromática, o un escáner. Si la imagen de salida de la cámara no está en formato digital, es necesario usar un conversor analógico-digital para digitalizarla. El escáner, por su diseño, ya entrega directamente una imagen digital.

Una vez obtenida la imagen en forma digital, la etapa siguiente es la de preprocesamiento (si fuese necesario). El preprocesamiento puede consistir en mejorar el contraste, suprimir el ruido, modificar la brillantez, su tamaño, etc. Hay que señalar que el preprocesamiento de la imagen depende de cuál sea el objetivo final que se quiere lograr al analizar la imagen, por lo que una misma imagen puede sufrir distintos procesamientos previos.

La tercera etapa es la segmentación. Su objetivo es dividir la imagen en las partes que la constituyen o en los objetos que la forman y el fondo. La salida del proceso de

segmentación son imágenes que contienen la frontera de las regiones de interés, o los pixeles que conforman la región misma. En general la imagen que finalmente se obtiene del proceso de segmentación es una imagen binaria, la que se obtiene mediante un operador de umbralado (Ver 3.1.9.1).

La imagen segmentada es procesada más tarde en un proceso denominado de descripción y representación, que constituye la cuarta etapa. La descripción está dirigida a extraer los rasgos de los objetos que permitirán diferenciar una clase de objetos de otras. La representación le asigna una etiqueta a cada objeto basándose en la información numérica que proporcionan los descriptores.

La última etapa es la de reconocimiento e interpretación. En esta etapa se asigna una etiqueta con un significado a los objetos encontrados en la segmentación con ayuda de sus rasgos descriptores [41].

3.1.9. Histograma de una imagen

El histograma de una imagen digital es una representación estadística que muestra la probabilidad con que un determinado nivel de gris aparece en la imagen. Se representa por el número de pixeles que tienen el mismo nivel de gris dentro del rango dinámico de la imagen.

En general se representa como un gráfico de barras en el que las abscisas son los distintos niveles de gris (o colores RGB) de la imagen y las ordenadas la frecuencia relativa con la que cada color aparece en la misma. El histograma proporciona información global sobre el brillo y el contraste de la imagen.

Una definición más formal del histograma es la siguiente: El histograma de una imagen $f(x,y)$ con *L* niveles de intensidad o de gris en el rango $[0, L-1]$, denotado como $h(r_k)$ es una función discreta.

$$h(r_k) = \frac{n_k}{N} \qquad \text{(Ec. 3.9)}$$

donde r_k es el *k*-ésimo nivel de gris, n_k es el número de pixeles en la imagen con el nivel de intensidad r_k y *N* el número total de pixeles en la imagen.

En la Fig. 3.5 se muestra una imagen en niveles de gris y su histograma. La función $h(r_k)$ proporciona la probabilidad de ocurrencia de un nivel de gris dado n_k. De igual forma, $h(r_k)$ también describe de manera global la apariencia de una imagen [38].

El histograma proporciona una descripción de la apariencia global de una imagen. Si los niveles de gris están concentrados hacia el extremo oscuro del rango dinámico, la apariencia global de la imagen será oscura; si sucede justo lo contrario, la imagen correspondiente será clara. Por su parte, un histograma que presente un perfil estrecho corresponderá a una imagen de bajo contraste y un histograma con una dispersión considerable de sus niveles de gris, corresponderá a una imagen de alto contraste.

En el caso de que la imagen sea en colores, se tendrán tres histogramas, de forma que el tratamiento de imágenes en colores se complicará por la aparición de nuevas componentes. En este caso el histograma no representará el número de pixeles con los tonos del negro al blanco, sino del negro al color correspondiente (rojo, verde o azul para el caso RGB) (figura.3.6).

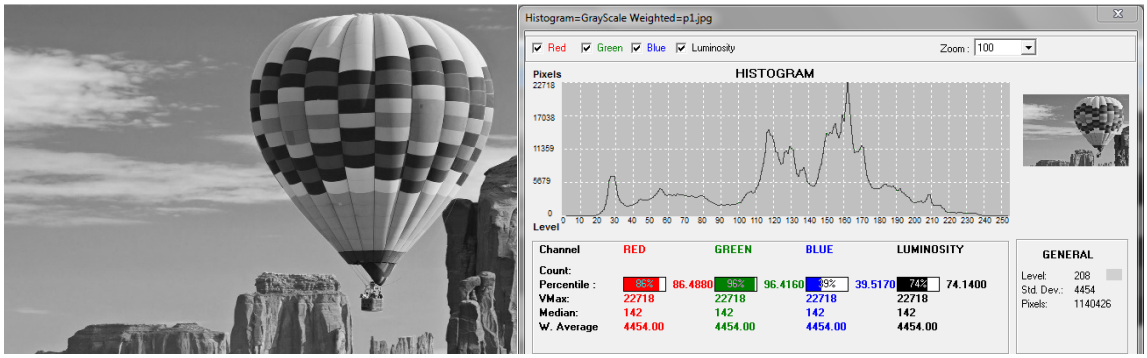


Figura 3.5. Imagen en niveles de gris y su correspondiente histograma.

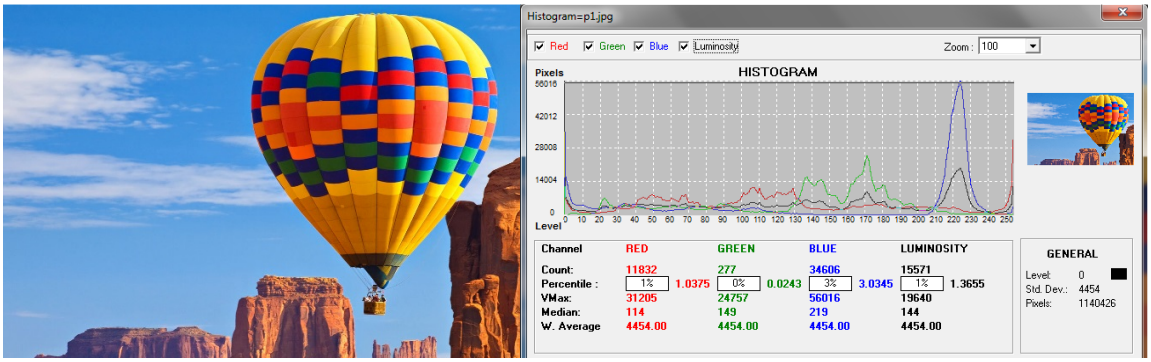


Figura 3.6. Imagen en colores y su correspondiente histograma en colores para cada canal RGB.

3.1.10. Métodos de segmentación basados en el umbralado

Las diferentes técnicas para segmentar una imagen mediante umbralado permiten separar un objeto dentro de la imagen del fondo que lo circunda. Se basan en la comparación de alguna propiedad de la imagen mediante un umbral fijo o variable: si el valor de la propiedad de un píxel supera el valor del umbral, entonces el píxel pertenece al objeto (o al fondo); en caso contrario, el píxel pertenece al fondo (o al objeto) [52].

Cuando la segmentación se realiza sobre la base del nivel de gris, el valor del nivel de gris de cada píxel debe ser comparado con el umbral para decidir si tal píxel pertenece al objeto o al fondo. La imagen de salida es siempre una imagen binaria en la cual aquellos pixeles cuyo valor (normalizado) es uno (claro), pertenecen al objeto y los pixeles cuyo valor normalizado es cero (oscuro), pertenecen al fondo.

La selección del valor del umbral se realiza generalmente a partir del histograma de la imagen. Si una imagen está compuesta de objetos que aparecen en la escena sobre un fondo más o menos homogéneo, entonces es de esperar que el histograma sea bimodal, es decir, si por ejemplo los objetos son más claros que el fondo, en el histograma aparecerán dos picos (modas), el ubicado en los valores de gris mayores correspondiente al objeto y otro pico para niveles de gris menores, correspondientes al fondo. En la Fig. 3.7 se muestra un histograma bimodal, en el cual el umbral se ubica en algún lugar entre los dos picos del histograma.

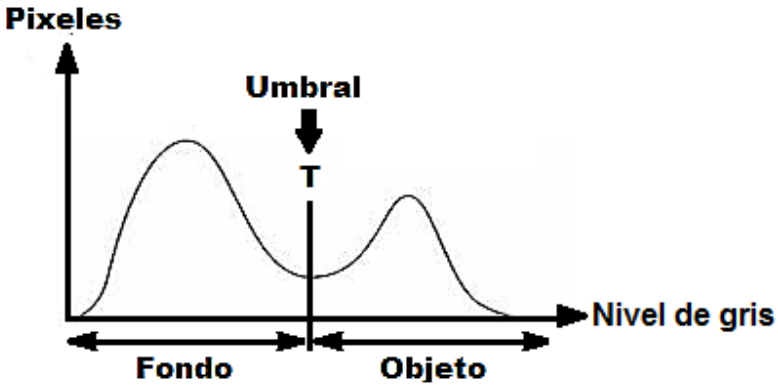


Figura 3.7. Ejemplo de un histograma bimodal.

La selección automática del umbral es un problema difícil, debido a que el histograma no siempre es bimodal, en cuyo caso resulta necesario combinar la información espacial presente en la imagen con la información referente a los niveles de gris.

3.1.11. Filtrado de una imagen

Uno de los principales objetivos del procesamiento digital de imágenes es el del realce o mejoramiento de las imágenes. Comprende un conjunto de técnicas tendientes a mejorar la calidad visual de una imagen. Estas operaciones permiten realzar las características de brillo y contraste de una imagen, reducir su contenido de ruido, agudizar o intensificar detalles presentes en ella, etc.

En este proceso, tal como en otras operaciones del procesamiento digital de imágenes, intervienen una imagen de entrada y una imagen de salida. La primera constituye la imagen cuyos datos serán procesados, es decir, sometidos al realce, y la segunda es la resultante de tal procesamiento.

Las operaciones que componen la técnica de realce pueden dividirse en dos tipos:

a) Operaciones de procesamiento puntual o píxel a píxel

b) Operaciones de vecindad o procesamiento de grupo de pixeles.

Las operaciones mencionadas en a) tienden a mejorar el contraste tonal en la imagen, esto es, la diferencia entre los valores más oscuros y más claros que se visualizan en la imagen. El procesamiento puntual altera de forma independiente los niveles de gris de los pixeles de una imagen. El nivel de gris de cada píxel en la imagen de entrada es modificado por un nuevo valor mediante operaciones matemáticas o relaciones lógicas. El valor resultante es ubicado en la imagen de salida en la misma posición espacial, esto es, en la misma posición (x, y) del arreglo rectangular de pixeles correspondiente a la imagen de entrada, donde x indica columna y y fila. Esto se realiza píxel a píxel, de manera individual, por lo que los pixeles vecinos no tienen influencia.

Las operaciones del tipo b) mejoran el contraste espacial en la imagen, es decir, la diferencia entre el valor digital de brillo de un determinado píxel y el de sus vecinos. Pretenden suavizar o reforzar estos contrastes espaciales de forma tal que, los valores de brillo en cada píxel de la imagen se asemejen o diferencien más de los correspondientes a los pixeles que los rodean. El procesamiento por grupo de pixeles opera sobre un grupo de pixeles de entrada que circundan a un píxel central. Los pixeles vecinos proveen información valiosa sobre las tendencias del brillo en el área bajo procesamiento.

Una imagen está formada por componentes de frecuencia que varían de bajas frecuencias a altas frecuencias. Donde prevalecen transiciones rápidas de brillo, hay altas frecuencias espaciales, mientras que las transiciones de brillo que apenas cambian representan bajas frecuencias. Las altas frecuencias en una imagen aparecen cuando están presentes bordes abruptos, como una transición del blanco al negro en dos regiones adyacentes.

Una imagen puede filtrarse para acentuar o eliminar una banda de frecuencias espaciales, tales como las altas frecuencias o las bajas frecuencias. Estas operaciones de procesamiento digital de imágenes se conocen como operaciones de *filtrado espacial* o *filtros en el dominio del espacio*. Otras operaciones de filtrado espacial permiten resaltar solamente las transiciones abruptas en la imagen, tales como los bordes de los objetos. Estas constituyen un subconjunto de las operaciones de filtrado espacial y se conocen como *operaciones de detección y realce de bordes*.

3.1.11.1. Implementación de filtros espaciales lineales

Los filtros espaciales se implementan mediante un proceso llamado *convolución espacial*. Es un método matemático utilizado en el procesamiento y análisis de señales; se le conoce también como filtro de respuesta finita al impulso (*Finite Impulse Response Filter*).

En el proceso de convolución espacial se recorre la imagen de entrada, píxel a píxel, ubicando los pixeles resultantes de la operación en la imagen de salida. El valor digital de brillo de cada píxel en la imagen de salida depende, en primer lugar, del grupo de pixeles de entrada que rodean al píxel que se está procesando. Al almacenar la información del brillo de los pixeles vecinos en el píxel central, la convolución espacial calcula la actividad de frecuencia espacial en esa área y por lo tanto, es capaz de filtrar con base en el contenido de la frecuencia espacial existente.

El proceso de convolución espacial utiliza un promedio ponderado del píxel de entrada y el de sus vecinos inmediatos para calcular el valor de brillo del píxel de salida. El grupo de pixeles utilizados en el cálculo del promedio ponderado se conoce como *núcleo* (*kernel*). El núcleo es una matriz móvil, generalmente cuadrada, con un número normalmente impar de valores en cada dimensión. Si la dimensión del núcleo es 1 x 1 se trata del procesamiento digital píxel a píxel; las dimensiones usuales en el procesamiento de vecindad son 3 x 3 y 5 x 5. Cuanto mayor es el tamaño del núcleo de pixeles que se emplea en el cálculo, más grados de libertad posee el filtro espacial.

Un cálculo de promedio ponderado es un *proceso lineal,* puesto que involucra la suma de elementos multiplicados por valores constantes. En la convolución espacial los elementos son los valores digitales de brillo de los pixeles de la imagen abarcados por el núcleo y los valores constantes son los pesos dados a los elementos del núcleo, llamados *coeficientes de convolución*. En el caso más simple donde todos los pesos del núcleo son iguales a 1, el proceso consiste en el cálculo de un promedio convencional, es decir, se promedian los valores de brillo de los pixeles de la imagen abarcados por el núcleo. Si los pesos se alteran, algunos pixeles tendrán más o menos influencia en el resultado general. La elección de estos pesos determina directamente el efecto del filtrado espacial, por ejemplo, *filtrado pasa altas*, *filtrado pasa bajas, filtrado para realce de bordes, etc*.

La mecánica de la convolución espacial consiste en aplicar junto con el núcleo de convolución, los coeficientes de convolución en forma de arreglo o matriz; esta matriz se conoce con el nombre de *máscara de convolución*. El píxel correspondiente al centro del núcleo y sus vecinos, al recorrer la imagen de entrada, se multiplican por los respectivos coeficientes de convolución y luego estos valores se suman. El resultado se ubica en la

imagen de salida en la misma posición que la del píxel central. Este proceso ocurre píxel a píxel para cada píxel en la imagen de entrada. Dado que el área filtrada se limita a los pixeles centrales, bajo ciertas condiciones los pixeles de los bordes de la imagen de entrada no se afectan por el proceso, puesto que no tienen los vecinos necesarios para realizar el cálculo. No obstante, puede realizarse la convolución espacial a todos los pixeles de la imagen (incluidos los de los bordes) si se toman oportunamente las medidas adecuadas para ello.

Las máscaras de convolución pueden tomar cualquier valor numérico. Sin embargo, cuando se ejecuta el proceso de convolución, el valor final resultante debe hallarse entre 0 y 255 (para una imagen de salida de 8-bits). Para ello, generalmente suelen reemplazarse por 255 los valores mayores que 255 y por 0 los valores menores que 0.

Si se considera un núcleo de pixeles de dimensión 3 x 3

$I =$

(x-1,y-1)	(x,y-1)	(x+1,y-1)
(x-1,y)	(x, y)	(x+1,y)
(x-1,y+1)	(x,y+1)	(x+1,y+1)

y una máscara de convolución cuyos nueve coeficientes son:

a	b	c
d	e	f
g	h	i

entonces la ecuación para el proceso de convolución espacial resulta como:

$$O(x,y) = a\,I(x-1,y-1) + b\,I(x,y-1) + c\,I(x+1,y-1) + d\,I(x-1,y) + e\,I(x,y) + f\,I(x+1,y) + g\,I(x-1,y+1) + h\,I(x,y+1) + i\,I(x+1,y+1) \quad \text{(Ec. 3.10)}$$

donde los prefijos *I* (*input*) y *O* (*output*) indican imagen de entrada e imagen de salida, respectivamente [42].

Filtros Pasa Bajas

Tienen por objeto suavizar los contrastes espaciales presentes en una imagen. Un filtro espacial pasa bajas tiene por efecto dejar pasar o mantener intocables las componentes de baja frecuencia espacial de una imagen. Las componentes de alta frecuencia son atenuados o virtualmente ausentes en la imagen de salida.

Un filtro pasa bajas muy utilizado es aquel cuya máscara de convolución tiene dimensión 3 x 3 y sus nueve coeficientes son iguales a 1/9, es decir:

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

Esta máscara produce un simple promedio de los valores de brillo de los pixeles y se conoce como filtro de la media. La suma de sus coeficientes es igual a 1 y todos ellos son

positivos. Estas dos características son válidas para todas las máscaras de filtros pasa bajas. Si este filtro se aplica a una región de una imagen donde cada píxel del núcleo tiene el mismo valor de brillo, es decir, un área de baja frecuencia espacial, el resultado es ese mismo valor de brillo. Esto es, el valor de brillo resultante en una región de pixeles con valor de brillo constante es el mismo que el de entrada. Esto se corresponde con el hecho de que no existe actividad espacial en la región (falta de cambios en los niveles de gris), lo que indica que existe frecuencia espacial 0. Si se aplica en una región donde los valores de brillo de los pixeles cambian rápidamente del blanco al negro y/o viceversa, es decir, un área de alta frecuencia espacial, el resultado será un valor medio de gris entre los negros y los blancos. Esto produce una imagen de salida compuesta por valores medios de gris que varían levemente. Las transiciones de altas frecuencias (bordes), del blanco al negro (o del negro al blanco), de la imagen de entrada son atenuadas a valores intermedios de niveles de gris.

Filtros Pasa Altas

Este tipo de filtros pretende aislar las componentes de alta frecuencia presentes en una imagen. El filtro pasa altas tiene un efecto opuesto al filtro pasa bajas, pues acentúa las componentes de alta frecuencia espacial mientras que deja sin tocar las componentes de baja frecuencia espacial.

Una máscara de filtro pasa altas muy común de dimensión 3 x 3, es aquélla que contiene un 9 en la posición del centro y -1 en las posiciones que lo rodean, es decir:

-1	-1	-1
-1	$+9$	-1
-1	-1	-1

La suma de los coeficientes es 1 y los coeficientes más pequeños rodean al coeficiente del centro que es positivo y el más grande. Esta disposición de los coeficientes indica que el píxel central del grupo de pixeles de entrada que se procesa aporta una alta influencia, mientras que los pixeles que lo rodean actúan oponiéndose a él. Si el píxel central posee un valor de brillo muy diferente al de sus vecinos inmediatos, entonces el efecto de estos últimos es despreciable y el valor de salida es una versión acentuada del valor original del píxel del centro. Esa diferencia grande indica una marcada transición en los niveles de gris, lo que indica la presencia de componentes de alta frecuencia. Por consiguiente, en la imagen de salida se espera que la transición aparezca acentuada. Por el contrario, si los valores de brillo de los pixeles vecinos son lo suficientemente grandes como para contrarrestar el peso del píxel del centro, entonces el resultado final se basa más en un promedio de los pixeles involucrados.

Si el valor de brillo de cada uno de los pixeles de un núcleo 3 x 3 es igual, el resultado es simplemente el mismo valor. Es decir, produce la misma respuesta que el filtro pasa bajas aplicado sobre regiones constantes. Esto significa que el filtro pasa altas no atenúa las componentes de baja frecuencia espacial. Más precisamente enfatiza las componentes de alta frecuencia, mientras que deja sin tocar las de baja frecuencia.

3.1.12. Metodología de Daugman: Filtros de Gabor

La metodología seguida por Daugman [28], se basa en la aplicación de filtros de Gabor para la extracción y codificación de características. Un filtro de Gabor se construye a través de la modulación de una onda senoidal / cosenoidal con una gaussiana, siendo capaz de proporcionar la localización óptima tanto en el espacio como en la frecuencia. Una onda

senoidal está perfectamente localizada en la frecuencia, pero no en el espacio. La modulación con una gaussiana proporciona localización espacial, aunque conlleva cierta pérdida de localización en frecuencia. La descomposición de la señal se logra mediante una cuadratura de filtros de Gabor, con una parte real especificada por un coseno modulada por una gaussiana y una parte imaginaria especificada por un seno modulado por una gaussiana. Los filtros reales e imaginarios son también conocidos como la componente simétrica impar y la simétrica par, respectivamente. La frecuencia central del filtro es especificada por la frecuencia de la onda senoidal / cosenoidal, y el ancho de banda del filtro se especifica por la anchura de la gaussiana.

Daugman hace uso de una versión 2D de los filtros de Gabor con el fin de codificar datos de patrones de iris. Un filtro de Gabor 2D del dominio de la imagen (x, y) se define como:

$$G_{\theta,f}(x,y) = exp\left\{\frac{-1}{2}\left[\frac{x^2}{\sigma_{x'}^2} + \frac{y^2}{\sigma_{y'}^2}\right]\right\} \cdot exp\{2\pi f j x'\} \qquad \text{(Ec. 3.11)}$$

$$x' = x\, sen\theta + y\, cos\theta$$

$$y' = x\, sen\theta - y cos\theta$$

Donde (x, y) especifica la posición en la imagen, $f = \frac{1}{T}$ es la frecuencia de la onda senoidal en la dirección θ respecto al eje x, y los parámetros $\sigma_{x'}$ y $\sigma_{y'}$ especifican el ancho y largo de la envolvente gaussiana a lo largo de los ejes x' y y' respectivamente.

Daugman demodula la salida de los filtros de Gabor, a fin de comprimir los datos. Esto se hace por cuantificación de información de fase en cuatro niveles, uno para cada posible cuadrante en el plano complejo. Se ha demostrado por Oppenhein y Lim [56] que la información de fase, en lugar de la amplitud, proporciona la información más significativa dentro de una imagen. Tomando sólo la fase de codificación permitirá discriminar la información del iris, mientras que es descartada la información redundante, tal como iluminación, que es representada por la componente de amplitud.

Estos cuatro niveles están representados utilizando dos bits de datos, por lo que cada pixel en el modelo normalizado del iris corresponde a dos bits de datos en la plantilla de iris o matriz binaria de identificación.

3.2. El ojo y el iris [54]

El ojo, a través de sus distintos elementos, recibe los estímulos luminosos externos, los codifica y transmite a través del nervio óptico al cerebro, lugar donde se produce el fenómeno de la visión. El ojo descansa sobre una cavidad ósea, en la mitad anterior de la órbita, rodeado de músculos extra oculares, grasa y tejido conectivo. Sólo está expuesta su parte más anterior, y está protegido por el reborde orbitario óseo.

Como se puede observar en la Figura 3.8, a grandes rasgos las partes que componen el ojo son:

- La pupila. Es un agujero central, de color negro, que permite la entrada de luz al globo ocular y cuyo tamaño varía dependiendo de la cantidad de luz que llega a éste. El color negro que presenta se debe a los pigmentos retinianos.
- El iris. Es una membrana de color y circular cuya coloración le da la característica más visible de los ojos humanos: la diversidad de colores que presenta dependiendo del individuo. Su apertura central es la pupila. Esta

membrana presenta un músculo de estructura circular que permite modificar el tamaño de la pupila.

- Un epitelio transparente, la córnea, que recubre tanto al iris como a la pupila. Esta es la primera y más poderosa lente del globo ocular, que junto con el cristalino, son las lentes que permiten la visión nítida de las imágenes.
- Una zona de color blanco, la esclerótica, que forma parte de los tejidos de soporte del globo ocular. La esclerótica se continúa con la córnea por delante y con la duramadre del nervio óptico por su parte posterior. Posee función de protección y en su zona exterior está recubierta por una mucosa transparente.

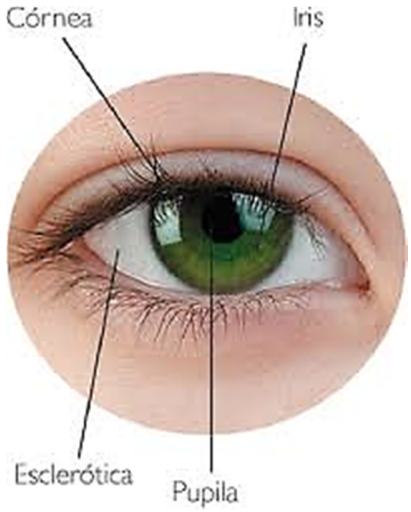


Figura 3.8. Esquema del ojo humano.

3.2.1. Características del iris

El iris comienza a formarse durante el tercer mes de gestación y su estructura se completa en el octavo mes. Durante el primer año de vida un conjunto de células produce cambios normales del color del iris, pero la evidencia clínica disponible indica que el propio modelo del iris es estable a lo largo de toda la vida.

Es un órgano interior protegido del ojo, detrás de la córnea y el humor acuoso, es visible externamente a una distancia cómoda puesto que los medios de comunicación ópticos delante de él son transparentes. El iris está compuesto de tejido conjuntivo elástico. Al ser un órgano interior del ojo, el iris es inmune a influencias medioambientales.

El iris es la membrana coloreada y circular del ojo que separa la cámara anterior de la cámara posterior del ojo. Posee una apertura central de tamaño variable que comunica las dos cámaras: la pupila. El iris tiene como funcionalidad regular la cantidad de luz que entra en el interior del ojo, la que varía su tamaño (diámetro) según la intensidad de luz que le llegue a la retina.

El iris está ubicado en la porción más anterior de la túnica vascular, la cual forma un diafragma contráctil delante del cristalino. Se ubica tras la córnea, entre la cámara anterior y el cristalino, al que cubre en mayor o menor medida en función de su dilatación.

Esta parte del polo anterior del ojo está constantemente activa, permitiendo a la pupila dilatarse (midriasis) o contraerse (miosis), de acuerdo a la intensidad que posea la fuente luminosa. Esta función tiene como objetivo regular la cantidad de luz que llega a la retina.

Las fibras musculares del iris se agrupan formando dos músculos: el esfínter del iris que produce la miosis, y el dilatador de la pupila que produce la midriasis.

Por lo tanto, se puede concluir que el iris es un tejido pigmentado de alta movilidad que se encuentra visible desde el exterior debido a la transparencia de la córnea; gracias a ello se encuentra perfectamente protegido de agentes externos.

Una propiedad que el iris comparte con las huellas dactilares es la morfología aleatoria de su estructura, lo que quiere decir que su textura es distinta para cada persona. El iris presenta ciertas características en favor de su potencial aplicación en la identificación biométrica de la persona. Algunas de ellas son:

- La facilidad de registrar su imagen a cierta distancia sin necesidad de contacto físico.
- El alto nivel de aleatoriedad en su estructura.
- Es estable y no cambia durante la vida del sujeto.
- Es estable frente a cambios que pudieran originarse por accidentes debido a la protección que le brinda la córnea.
- El intento de falsificar el iris de una persona conllevaría operaciones quirúrgicas que podrían dañar severamente la visión.

No obstante, la característica principal es que, basándose en los estudios realizados por Daugman, el iris contiene más información para identificar unívocamente a la persona que una huella dactilar.

El propósito del reconocimiento del iris es obtener, con alto grado de seguridad, la identidad de una persona empleando análisis matemático y de procesamiento de imágenes. Debido a que el iris es un órgano interno protegido, con textura aleatoria y estable, puede ser usado como una clave viva que no necesita ser recordada, pero que siempre estará ahí.

3.3. Generalidades de los FPGA

Los FPGA son actualmente los componentes más representativos de los dispositivos lógicos programables. Un FPGA es un dispositivo cuyas características pueden ser modificadas, manipuladas o almacenadas mediante programación aun después de haber sido puestos en funcionamiento en algún dispositivo, de ahí el término *Field Programmable* (Programable en campo). La configuración de un FPGA es generalmente especificada usando un lenguaje de descripción de hardware (HDL) como los usados para la programación de los ASIC, tal como puede ser el VHDL o VERILOG. Los FPGA pueden ser usados para implementar cualquier función lógica que un ASIC puede desempeñar. La habilidad de poder ser reprogramados después de su instalación y el reducido costo del ciclo de prueba, debido a que el mismo dispositivo puede ser reprogramado sin necesidad de fabricar uno nuevo, ofrecen muchas ventajas en muchas aplicaciones.

3.3.1. Historia

La industria de los FPGA comenzó a surgir a partir de las PROM (*Programmable Read Only Memory*) y los PLD (*Programmable Logic Device*). Ambos dispositivos pueden ser programados, ya sea en lotes en el sitio de fabricación o en campo (*Field Programmable*). No obstante, las interconexiones entre los componente lógicos están establecidas de una manera fija en el dispositivo. Los cofundadores de Xilinx, Ross Freeman y Bernard Vonderschmitt, inventaron el primer arreglo programable en campo en 1985, el XC2064. Este dispositivo contaba con compuertas programables y con interconexiones entre estas compuertas que eran programables. Este primer dispositivo contaba con solo 64 CLB (*Configurable Logic Block* – Bloque lógico configurable) con tres LUT (*Look Up Table* – Tabla de búsqueda). Más de veinte años después, Freeman fue elegido para pertenecer al *National Invertors Hall of Fame* (Salón de la fama de inversores nacionales) por esta invención.

Después de este evento, los FPGA fueron evolucionando en cuanto a la complejidad de los bloques lógicos que los conforman y a la cantidad de estos bloques con los que cuenta un dispositivo. La década de los noventa fue un periodo muy importante para el desarrollo de los FPGA, tanto en el avance de la tecnología como en el volumen de producción; a inicios de dicha década, la principal aplicación de estos era en el área de las telecomunicaciones y las redes de computadoras. Al final de la década su área de aplicación se había extendido a aplicaciones industriales, automotrices y del consumidor.

3.3.2. Comparación de los FPGA con tecnologías similares

Circuitos integrados de aplicación específica (ASIC)

A lo largo de la historia, los FPGA han sido más lentos, menos eficientes en cuanto al consumo de energía y generalmente, de un menor rendimiento que su contraparte los ASIC. No obstante, las ventajas de usar los FPGA son que tienen un ciclo más corto de desarrollo-fabricación, la habilidad de poder ser reprogramados en campo para corregir fallas o incluir mejoras y un menor costo del ciclo de diseño. Algunos fabricantes hacen uso de ambas tecnologías, al desarrollar y probar sus diseños en FPGA para construir después la versión final de manera que pueda ser modificada después de construida.

Dispositivo lógico complejo programable (CPLD)

Las diferencias principales entre los CPLD y los FPGA son a nivel arquitectural. Un CPLD está conformado por una estructura consistente en uno o más arreglos programables de suma de productos más o menos restringida cuyo bloque de salida son unos pocos registros controlados por una señal de reloj. Esto trae como resultado una flexibilidad limitada, con la ventaja de que los retrasos de propagación pueden ser estimados de manera más precisa. Por otra parte, en la arquitectura de los FPGA el elemento dominante son las interconexiones. Esto los hace dispositivos mucho más flexibles en cuanto al rango de diseños cuya implementación resulta práctica en ellos, pero es más complejo realizar el diseño de los dispositivos. Otra diferencia remarcable es la presencia en la mayoría de los FPGA de funciones embebidas de alto nivel, como multiplicadores y sumadores, así como también memorias y bloques lógicos para implementar decodificadores o funciones matemáticas.

3.3.3. Aplicaciones

Las aplicaciones de estos dispositivos incluyen áreas como el procesamiento digital de señales, sistemas aeroespaciales y de defensa, prototipos ASIC, equipos médicos, visión

por computadora, reconocimiento del habla, criptografía, bioinformática, emulación de hardware computacional, radioastronomía, y algunos otros campos en los que el uso de los FPGA se encuentra en crecimiento.

Originalmente, los FPGA comenzaron como competidores de los CPLD y fueron empleados en áreas similares, constituyendo la parte central de la lógica en las tarjetas de circuito impreso (PCB del inglés *Printed Circuit Board*).

Al incrementarse su tamaño, capacidades y velocidad, comenzaron a ser usados en aplicaciones que requerían funciones más complejas, al punto que algunas de ellas son completamente SoC (*System on Chip, Sistema en un Chip*). Particularmente, al introducirse multiplicadores dedicados en la arquitectura de los FPGA a finales de la década de los 90, aplicaciones que hasta ese momento habían sido dominio de los DSP, comenzaron a incorporar FPGA.

Los FPGA son especialmente usados en cualquier área o algoritmo que pueda sacar ventaja del gran paralelismo que ofrece de manera natural su arquitectura, como es por ejemplo, los algoritmos criptográficos. Asimismo los FPGA están siendo cada vez más usados en aplicaciones donde algoritmos como el de la FFT o la convolución de señales, son ejecutados en el FPGA en lugar de hacerse en un microprocesador.

El paralelismo inherente en los recursos lógicos del FPGA permite un gran volumen de procesamiento incluso a una baja frecuencia de reloj. La flexibilidad de los FPGA permite aun un mayor incremento en el rendimiento de los algoritmos, al sacrificar un poco la precisión y el rango en el formato de los números por un número mayor de unidades aritméticas, lo que ha llevado a un nuevo tipo de procesamiento llamado cómputo reconfigurable, en donde tareas intensivas que consumen mucho tiempo son ejecutadas en el FPGA en lugar de procesarlas por software.

Los FPGA también son ampliamente usados para la validación de sistemas, lo cual permite a las compañías comprobar sus diseños antes de comenzar la producción en fábrica, lo cual reduce el tiempo de salida al mercado y los costos de diseño y verificación.

3.3.4. Arquitectura interna

Los FPGA están construidos básicamente a partir de componentes lógicos llamados bloques lógicos, que se unen mediante una jerarquía de interconexiones programables que hacen posible su interconexión (figura 3.9). Los bloques lógicos que conforman el FPGA pueden ser configurados para ejecutar funciones complejas de lógica combinatoria, o simples compuertas AND u OR. En la mayoría de los FPGA actuales los bloques lógicos incluyen componentes de memoria, que varían desde arreglos de *flip-flops* hasta arreglos de memorias más completos. La arquitectura más común de los FPGA consiste en un arreglo de CLB (*Configurable Logic Block*), bloques de Entrada/Salida y canales de interconexión. Generalmente todos los canales de interconexión tienen la misma cantidad de líneas en todo el dispositivo. Un diseño debe ser trazado en el FPGA con los recursos adecuados, mientras que el número de CLB y de puertos de Entrada/Salida pueden ser determinados fácilmente; el número de interconexiones necesarias varía considerablemente aun entre diseños que requieren la misma cantidad de lógica. Debido a que las interconexiones no usadas causan un incremento en el costo y un decremento en el desempeño sin proveer ningún beneficio, los fabricantes de FPGA intentan incluir solo las interconexiones necesarias, de tal manera que la mayoría de los diseños puedan ser trazados en términos de LUT y de puertos de E/S.

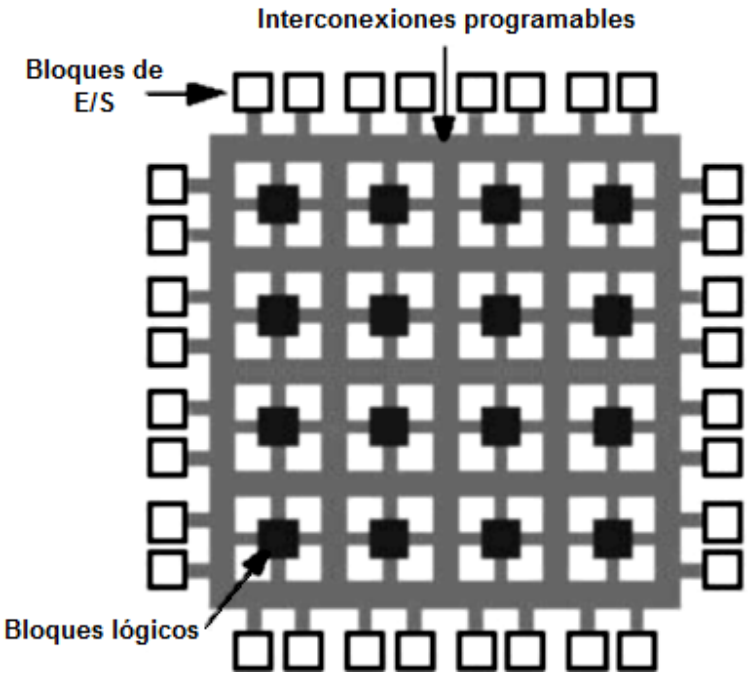


Figura 3.9. Arquitectura interna de una FPGA.

Un bloque lógico básico de un FPGA consiste en una LUT de 4 entradas y un *flip-flop*, como se indica en la figura 3.10. Recientemente se han comenzado a fabricar LUT de 6 entradas en los dispositivos de alto rendimiento. A la salida del bloque se tiene una única línea, que puede ser o no, una salida registrada. El bloque cuenta con cuatro entradas para la LUT y una para el reloj. Debido a que las señales de reloj y algunas otras señales con una alta carga de salida (*fan-out)* son interconectadas normalmente usando líneas dedicadas, éstas y otras señales son administradas por separado.

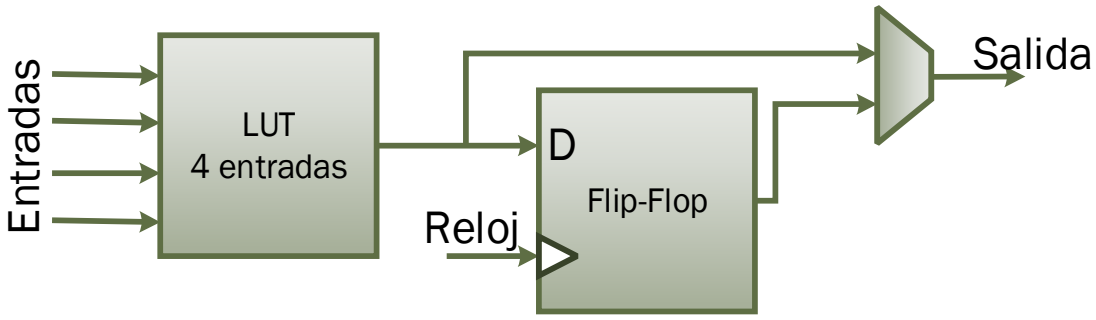


Figura 3.10. Bloque lógico básico.

De manera general, las líneas de interconexión de un FPGA son no segmentadas, esto es, cada segmento de conexión se conecta a un solo bloque antes de terminar en un bloque de interconexión (figura 3.11); cerrando los interruptores de estos bloques de interconexión, pueden formarse pistas de conexión mayores. En cada punto donde se interconectan una línea vertical y una horizontal, existe un bloque de interconexión. En este esquema, por cada línea que entra al bloque, hay tres interruptores que permiten la conexión con las tres líneas adyacentes al bloque. Esta topología de interruptores es llamada topología de

interruptores basada en dominios. En esta topología, las líneas de la sección 1 se conectan con las líneas adyacentes en la sección 1, las líneas de la sección 2 se conectan con las líneas de la sección 2, y así sucesivamente. En los últimos años, las familias de FPGA se expanden más allá de estas características para incluir funciones de más alto nivel de manera fija en los bloques que constituyen el dispositivo. Al tener estas funciones embebidas en el chip, se reduce el área que éstas ocupan, lo que hace dichas funciones más rápidas en comparación a las construirlas con primitivas. Algunos ejemplos de esto incluyen bloques DSP genéricos, procesadores embebidos y memorias embebidas de alta velocidad.

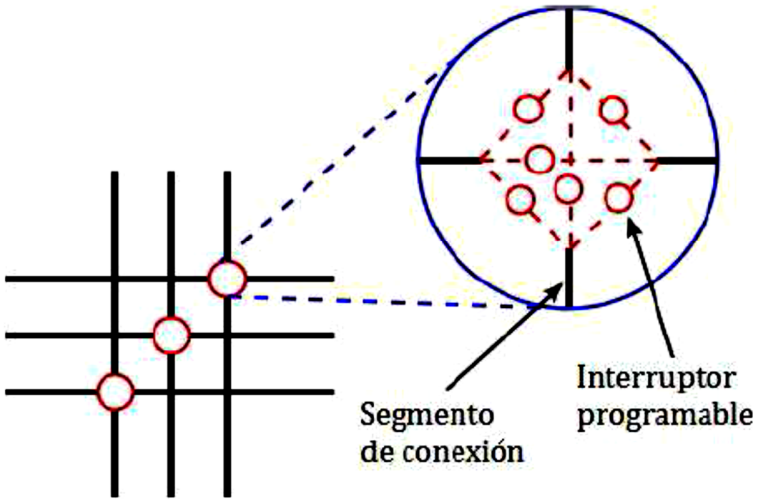


Figura 3.11. Bloque de interconexión.

3.3.5. Diseño y verificación

Para definir el comportamiento de un FPGA, el usuario debe escribir un programa en algún lenguaje de descripción de hardware (HDL) o hacer un diseño esquemático. El usar un HDL funciona mejor si lo que se diseña es una estructura grande y compleja, mientras que para diseños sencillos es más conveniente usar el diagrama esquemático que puede ayudar a visualizar el diseño. Luego, haciendo uso de alguna herramienta de software que es generalmente administrada por la compañía fabricante del FPGA, se genera una lista de conexiones. Después, esta lista es acomodada de acuerdo al dispositivo usado mediante un proceso llamado “sitúa y rutea” (*place-and-route)*; este mapa de conexiones es validado con análisis de tiempos, simulación y otros métodos de verificación. Una vez que el mapa ha sido validado, se genera un archivo binario que luego servirá para configurar el FPGA.

Para simplificar el diseño de sistemas complejos en el FPGA, existen bibliotecas de funciones predefinidas y circuitos que han sido probados y optimizados para acelerar el proceso de diseño. Estos bloques son comúnmente llamados *IP cores* (*Intellectual Property cores* - Núcleos de propiedad intelectual). Estos bloques están disponibles por medio de los fabricantes, raramente de manera gratuita, e incluso por la comunidad de código abierto (*Open Source)*. En un proceso típico de diseño, un desarrollador de aplicaciones en FPGA realiza simulaciones del diseño en varias etapas a lo largo del proceso de diseño.

Inicialmente la descripción RTL (*Register Transfer Level* – Nivel de transferencia de registro) en HDL es simulada y los resultados son estudiados, seguidamente, el software de síntesis crea la lista de interconexiones que es interpretada en una descripción a nivel de compuertas, donde nuevamente se realizan simulaciones para verificar que no se produzcan errores. Por último se coloca en el FPGA y nuevamente se realiza la simulación para obtener datos acerca de los retardos de propagación, los cuales quedan anotados dentro de la lista de interconexiones.

3.3.6. Panorama general de los lenguajes de descripción de hardware

Un HDL es un lenguaje usado para modelar en una forma textual la operación funcional de una pieza de hardware (circuitos electrónicos y/o sistemas completos). Al igual que en los lenguajes de programación comunes, se observan ciertas diferencias entre los HDL que van desde la sintaxis del código hasta los métodos de simulación y síntesis, pasando por su capacidad de compatibilidad. Por lo mismo, cualquiera de ellos, implica un aprendizaje formal [43] [44].

Se conocen dos tipos generales de HDL por algunos diseñadores: los de bajo modelado y los de alto modelado. Los de bajo modelado son capaces de realizar descripciones simples y no permiten la jerarquización de módulos; los de alto modelado son aptos para diseños más complejos.

En el contexto tecnológico mundial, los HDL más comunes e importantes para alto modelado son dos: VERILOG y VHDL. Ambos permiten diseñar de acuerdo a ecuaciones, tablas de verdad y diagramas de estado; su sintaxis es particular para cada uno con palabras reservadas para indicar al sintetizador el método de diseño que se elija.

VHDL (estándares IEEE 1076 – 1987 y IEE 1076 – 1993) presenta cierta complejidad debido a la sintaxis propuesta, aunque en otro sentido está diseñado para trabajar preferentemente en escala VLSI. Este lenguaje de descripción es ampliamente utilizado en la actualidad al igual que VERILOG (estándar IEEE 1364 – 1995), con la salvedad de que éste es más reciente y presenta menor complejidad de sintaxis, debido a que sus raíces sintácticas se basan en el lenguaje C estándar.

Las características que comparten ambos lenguajes son:

- El diseño puede descomponerse jerárquicamente, por lo que la modularización puede ser realizada desde un nivel bajo de abstracción.
- Soportan cualquier nivel de modelado y de abstracción. VERILOG, contiene todas las funciones cubiertas por el VHDL, e incluye la capacidad de poder diseñar al nivel de descripción estructural (SDL- *Structural Description Level*) dirigiéndose a un nivel de transistor, que es aún más bajo que el nivel de las primitivas.
- Cada elemento de diseño tiene una interfaz bien definida (para conectarse a otros elementos) y una especificación precisa de su comportamiento (funcionamiento del circuito).
- La concurrencia, el retardo y el tiempo de sincronía pueden ser modelados. Ambos lenguajes permiten estructuras síncronas, así como asíncronas.

3.3.7. Procesadores suaves (*Soft-processors*)

Actualmente ha emergido una nueva tecnología para el diseño electrónico digital basado en procesadores. Esta tecnología utiliza a los FPGA para albergar arquitecturas de núcleos de procesadores comerciales con propiedad intelectual (*IP Cores*) y efectuar el cómputo con hardware personalizable en los sistemas embebidos.

Los procesadores son clasificados en dos categorías *Hard y Soft* (Duro y Suave); esta clasificación se refiere a la flexibilidad y la capacidad de configuración de su arquitectura.

El procesador *Hard* es dedicado e incrustado físicamente en el silicio, tales como los procesadores convencionales que vemos en las computadoras, microcontroladores, DSP; por mencionar algunos, están los procesadores Pentium de Intel, Opteron de AMD, ARM, CORTEX, etc., los cuales tienen una arquitectura fija que no se puede modificar o alterar.

El procesador *Soft* o procesador suave (*Softcore*) usa como base los recursos de lógica programable existente en el FPGA para implementar la lógica de la arquitectura de un procesador, además de ser personalizable por software, con la gran ventaja de poder agregar módulos tales como IIC, Ethernet, PCI Express, contadores, USB, memoria, coprocesadores, VGA, etc. Muchos de ellos son encapsulados en *IP Cores* o hardware diseñado por el usuario mediante lenguajes de descripción de hardware (HDL); dentro de estos procesadores tenemos el Microblaze de Xilinx, Nios II de Altera, LatticeMicro32 de Lattice y PowerPC440 de IBM.

A continuación se presentan la arquitecturas del procesador Nios II de Altera utilizado en este trabajo.

3.3.7.1. Microprocesador embebido Nios II

El fabricante Altera proporciona una infraestructura completa para crear sistemas con microprocesadores embebidos, completamente a la medida según las necesidades del diseñador, por medio de la combinación de una serie de componentes configurables sobre sus FPGA.

Para ello proporciona un entorno específico, al que denomina SOPC Builder (*System on a Programmable Chip Builder*, Generador de sistema en un chip programable) que permite la definición y configuración a la medida del sistema microprocesador Nios I, y que gracias a la herramienta de síntesis Quartus II puede ser implementado directamente sobre un FPGA de la misma marca.

Al tratarse de un sistema flexible que puede ser adaptado a las necesidades específicas de cada diseño, no solo posibilita ajustar su tamaño al de un determinado dispositivo, sino que deja decidir al diseñador cuándo una implementación se debe realizar en software y cuando en hardware, lo que permite elevar el rendimiento.

El Nios II es un núcleo procesador configurable que se puede implementar en alguna de sus tres versiones disponibles, según se busque minimizar el consumo de recursos del FPGA o maximizar el rendimiento del procesador:

- El NIOS II/f (*fast*) es la versión diseñada para alto rendimiento, la que con un *segmentado* de 6 etapas proporciona opciones específicas para aumentar su desempeño como memorias caché de instrucciones y de datos, o como una unidad de manejo de memoria (MMU - *Memory Management Unit*).
- El NIOS II/s (*standard*) es la versión con *pipeline* de 5 etapas que dotada de una unidad aritmético-lógica (ALU) busca combinar rendimiento y consumo de recursos.

- El NIOS II/e (*economy*) es la versión que requiere menos recursos del FPGA, sin *pipeline* ya que es muy limitada dado que carece de operaciones de multiplicación y división.

Cada una de estas versiones se completa con una serie de componentes, memorias y periféricos mediante su interconexión a través de un bus de sistema denominado *Avalon Switch Fabric*, para obtener un sistema Nios II completo en un chip (SOC - *System On Chip*).

El NIOS II consiste en un procesador RISC de 32 bits de propósito general basado en una arquitectura tipo Harvard (usa buses separados para instrucciones y para datos). Entre sus características principales encontramos:

- Juego completo de instrucciones, datos y direcciones de 32 bits.
- 32 registros de propósito general.
- Juegos opcionales de registros *shadow*.
- 32 fuentes de interrupción externa.
- Una interfaz de control de interrupciones externas para fuentes adicionales.
- Instrucciones dedicadas para multiplicaciones de 64 y 128 bits.
- Instrucciones para operaciones de punto flotante con precisión simple.
- Operaciones de multiplicación y división de 32 bits.
- Acceso a una variedad de periféricos integrados e interfaces para el manejo de memorias y periféricos externos.
- Una Unidad de Manejo de Memoria (MMU) opcional para soportar sistemas operativos que la necesiten.
- Una Unidad de Protección de Memoria (MPU - *Memory Protection Unit*) opcional.
- Entorno de desarrollo de software basado en la herramienta GNU C/C++.

3.3.8. Tarjeta de Desarrollo DE2-115

La tarjeta de desarrollo DE2-115 de la marca Altera utilizada en este trabajo, tiene características que permiten implementar un amplio rango de circuitos diseñados, desde los más simples hasta aplicaciones multimedia.

Las características de las tarjetas DE2-115, mismas que se observan en las figuras 3.12 y 3.13 son:

- Dispositivo FPGA Cyclone® IV 4CE115 FPGA,
- Dispositivo Altera de configuración serial – EPCS64,
- USB Blaster (en la tarjeta) para la programación; se admiten tanto el modo de programación JTAG como el AS (*Active Serial*),
- 2MB de memoria SRAM,
- Dos unidades de 64MB de memoria SDRAM,

- 8MB de memoria Flash,
- Socket para memoria SD,
- 4 *push-bottons*,
- 18 interruptores,
- 18 leds rojos,
- 9 leds verdes,
- Oscilador de 50MHz para entradas de reloj,
- *Codec* de audio de 24 bits, con entrada y salida de líneas, así como entrada de micrófono,
- Conector VGA con 3 convertidores digital-analógicos de alta velocidad,
- Conector y decodificador de video (NTSC/PALM/SECAM),
- 2 Gigabit Ethernet PHY con conector RJ45,
- Conector de 9 pines para comunicación RS-232,
- Conector PS/2 para ratón/teclado,
- Receptor infrarrojo,
- 2 conectores SMA para entrada/salida de señal de reloj,
- 1 conector de 40 pines con diodos de protección para expansión,
- 1 conector HSMC (*High Speed Mezzanine Card*- Tarjeta intermedia de alta velocidad),
- 1 módulo LCD de dos líneas por 16 caracteres por cada línea.

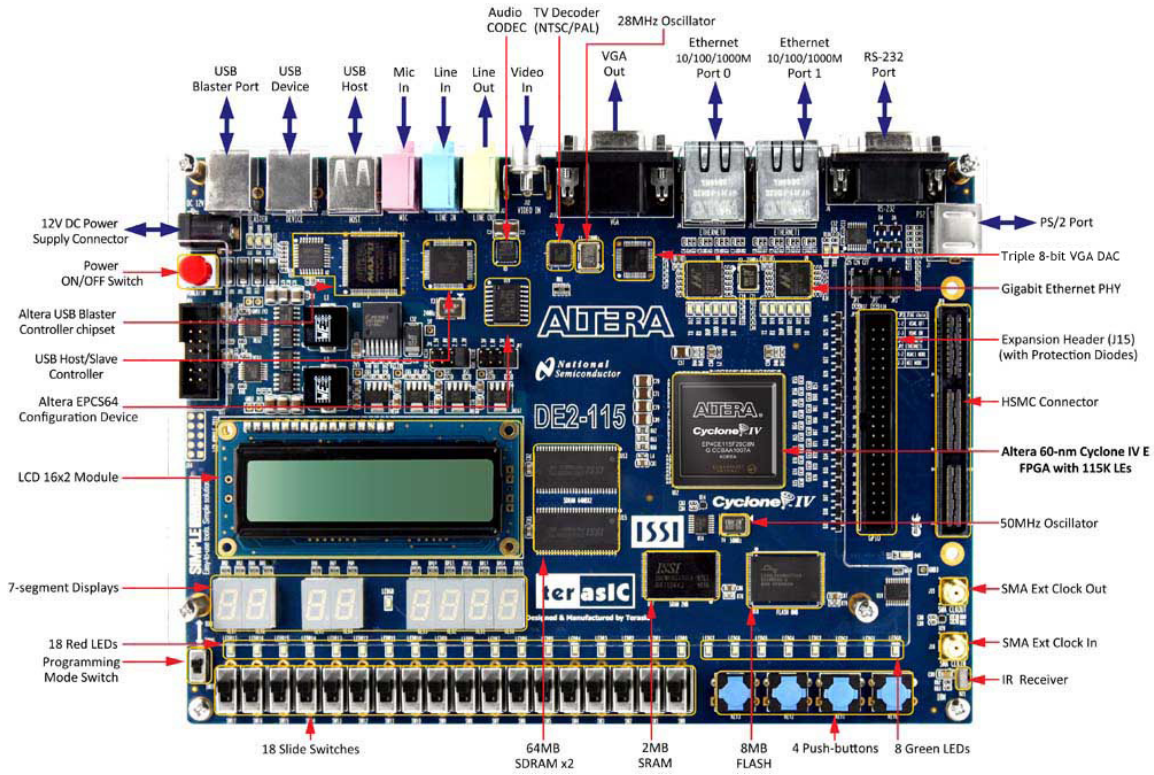


Figura 3.12. Tarjeta de desarrollo DE2-115 (vista frontal).

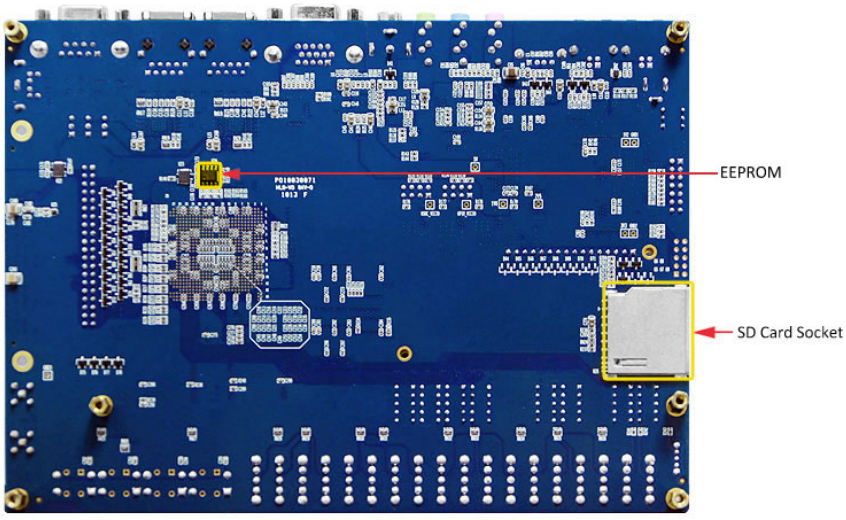


Figura 3.13. Tarjeta de desarrollo DE2-115 (vista trasera)

CAPÍTULO IV

Solución del problema

En este capítulo se describe la técnica desarrollada para la segmentación y normalización del iris en imágenes de iris humanos. Así mismo, se expone la metodología aplicada para la implementación de esta técnica en la tarjeta de desarrollo FPGA DE2-115.

4.1. Metodología general de la propuesta

El propósito esencial de este trabajo es el procesamiento de imágenes de iris en una plataforma FPGA; en nuestro caso se utilizó la tarjeta de desarrollo FPGA DE2-115 de Altera.

Para el desarrollo del algoritmo de segmentación y normalización de iris, se seleccionó la base de datos de imágenes CASIA-Iris V1 [49] cuyas características se describen en la sección 4.2.1.

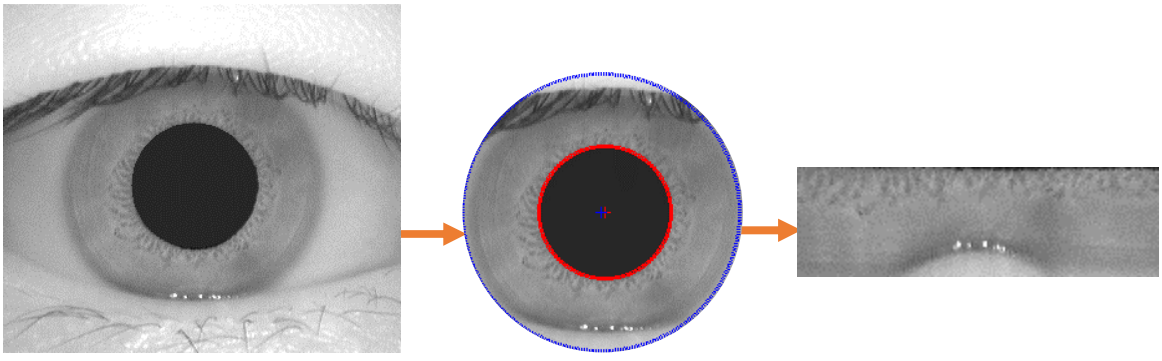
De forma general, la metodología utilizada en este trabajo es la siguiente:

- Desarrollo del algoritmo de segmentación y normalización del iris. Se obtiene como resultado el iris normalizado mediante la aplicación de técnicas de procesamiento de imágenes. El algoritmo se desarrolló tomando en cuenta los recursos de software (paquetería) disponibles para la programación de la aplicación en FPGA y hardware (tarjeta de desarrollo) donde se va a implementar el mismo.
- Implementación del algoritmo en la tarjeta de desarrollo FPGA. Se obtiene como resultado la implementación del algoritmo en una plataforma FPGA con las ventajas y desventajas que esto conlleva.

4.2. Algoritmos de segmentación y normalización de iris

El desarrollo del algoritmo de segmentación y normalización de iris es una de las partes esenciales de este trabajo, ya que según la hipótesis planteada mediante herramientas sencillas de procesamiento de imágenes y con base en sus características, podíamos segmentar y normalizar el iris de manera correcta con un alto porcentaje de efectividad.

Lo que se quiere lograr en este trabajo es la segmentación y normalización del iris de las imágenes de la base de datos utilizada; es decir, partiendo de la imagen de entrada (figura 4.1a), aproximar mediante circunferencias la información relacionada con el iris (figura 4.1b), para posteriormente normalizarla, esto es, transformarlo de coordenadas polares a coordenadas cartesianas o rectangulares (figura 4.1c).



(a) (b) (c)

Figura 4.1. Imágenes resultantes, (a) imagen de entrada, (b) aproximación y segmentación del iris. (c) iris normalizado.

La hipótesis base en la cual se ha basado el algoritmo desarrollado es que una vez que se obtuvo el centro de la pupila y su contorno ha sido aproximado mediante una circunferencia, si se logra obtener una imagen binaria del ojo completo donde la mayor cantidad de información del iris se encuentre con valor 0 (negro) y la mayor cantidad de pixeles que no son parte del iris se encuentren en valor 1 (Blanco) (figura 4.2), entonces podríamos aproximar el borde exterior del iris mediante una circunferencia.

Figura 4.2. Ejemplo de imagen umbralada para la segmentación del iris.

De manera general se proponen los siguientes pasos para segmentar y normalizar el iris (figura 4.3).

El primer paso consiste en obtener el borde interior del iris, es decir, detectar y aproximar la pupila mediante una circunferencia. Posteriormente, como segundo paso, la imagen es procesada y umbralada para mantener con valor de intensidad '0' en una imagen binaria la información que corresponde al iris y con valor '1' todo lo demás. En el tercer paso, la imagen resultante del paso anterior (imagen binaria) es analizada (tomando como referencia el centro de la circunferencia del borde interior) para encontrar todos aquellos puntos limite o frontera por cada fila, donde termina la información del iris (pixeles con valor 0) en dicha imagen. Una vez obtenidos dichos puntos frontera, de acuerdo a la posición de éstos, se obtiene el borde exterior del iris mediante su aproximación con una circunferencia. Como paso final, una vez que se cuenta con ambos bodes (interior y exterior) del iris se hace una conversión de coordenadas polares a rectangulares, lo que es denominado Normalización del iris.

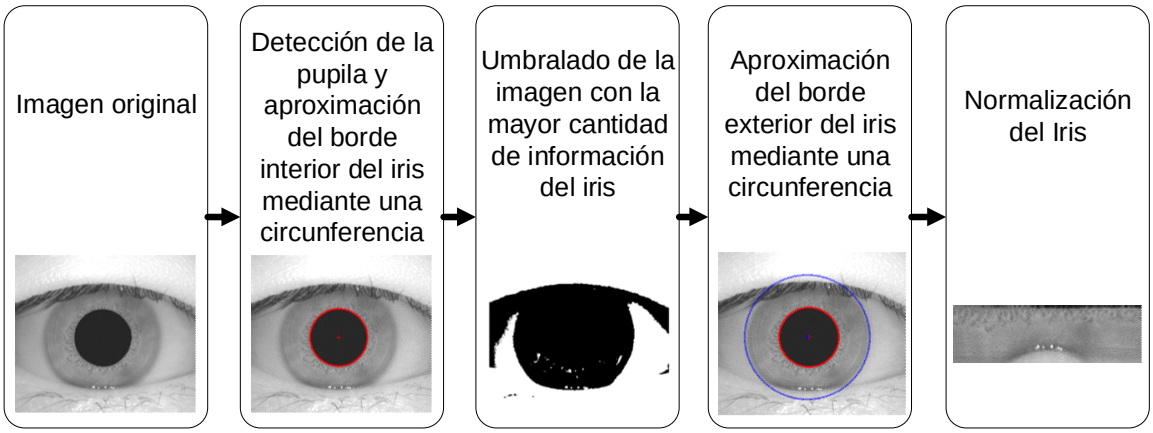


Figura 4.3. Pasos del algoritmo propuesto para la segmentación y normalización del iris.

4.2.1. Base de datos IRIS

Para promover la investigación, el *National Laboratory of Pattern Recognition* (NLPR) afiliado a la *Chinese Academy of Sciences Institute of Automation* (CASIA) provee bases de datos de iris libres para investigadores en reconocimiento de iris.

Las imágenes de iris de CASIA v1.0 (CASIA-IrisV1) fueron capturadas con una *cámara de iris* casera (figura 4.4) [49]; ocho iluminadores NIR (*Near Infrared*, Cercano al infrarrojo) de 850nm se disponen circularmente alrededor del sensor para asegurar que el iris quede uniformemente iluminado.

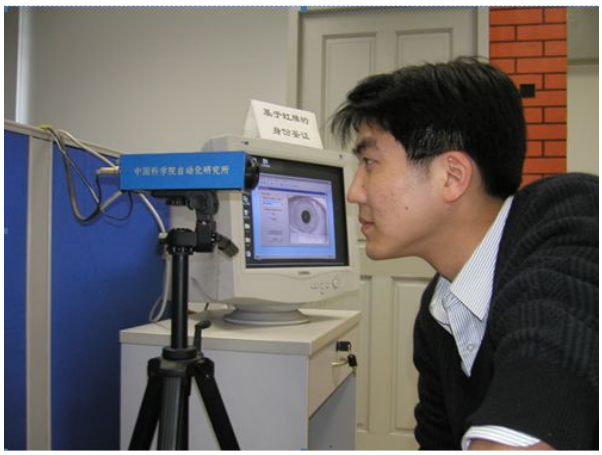


Figura 4.4. Sesión de captura con la cámara desarrollada por la *Chinese Academy of Sciences* para la creación de la base de datos CASIA-Iris V1.

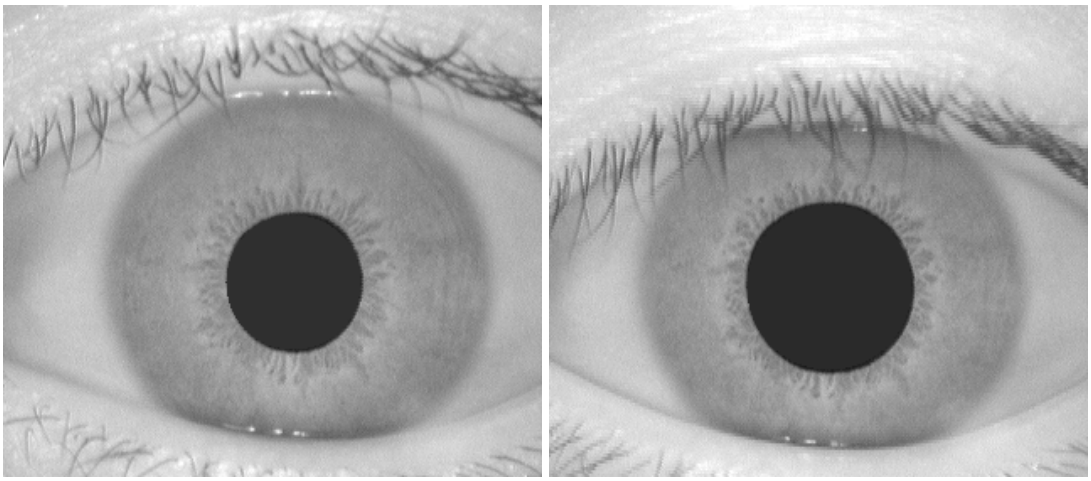
La base de datos de imágenes Casia versión 1 (CASIA-Iris V1) está compuesta de 756 imágenes, provenientes de 7 imágenes capturadas de 108 ojos distintos de 54 personas. De cada ojo se capturaron siete imágenes en dos sesiones con la cámara desarrollada por CASIA, tres capturas en la primera sesión y cuatro en la segunda (figura 4.5). Todas las imágenes se guardaron en formato gráfico BMP con una resolución espacial de 320 x 280 pixeles cada una.

Con el fin de ocultar el diseño de esta cámara de iris (especialmente el esquema de disposición de los iluminadores NIR), las regiones de la pupila de todas las imágenes de iris en la base de datos CASIA-IrisV1 fueron automáticamente detectadas y reemplazadas por una región circular de intensidad constante, con el fin de enmascarar los reflejos de los iluminadores NIR antes del lanzamiento al público.

Claramente este proceso puede afectar la detección de la pupila, pero no tiene efectos en otros componentes de un sistema de reconocimiento de iris, como la extracción de características, siempre y cuando para ello se usen solo datos de la imagen en la región comprendida entre la pupila y la esclerótica.

4.2.2. Cálculo del centro de la pupila y la aproximación de su borde mediante una circunferencia

La detección de la pupila y la aproximación de su borde mediante una circunferencia es el primer paso y una de las bases para la posterior aproximación, segmentación y normalización del iris, ya que de esta etapa se obtiene la ubicación del centroide (punto central) el cual es utilizado en los pasos posteriores.



(a) (b)

Figura 4.5. La imagen numerada 17 en la base de datos, capturada en ambas sesiones. (a) Imagen correspondiente a la primera sesión y (b) a la segunda sesión.

Una característica muy notable en todas las imágenes de la base de datos elegida y que se puede observar claramente en el histograma de cada imagen (figura 4.6), es que la mayor concentración de pixeles con más bajo nivel de intensidad se encuentra en la pupila, aunque también en algunos pixeles relacionados con las pestañas.

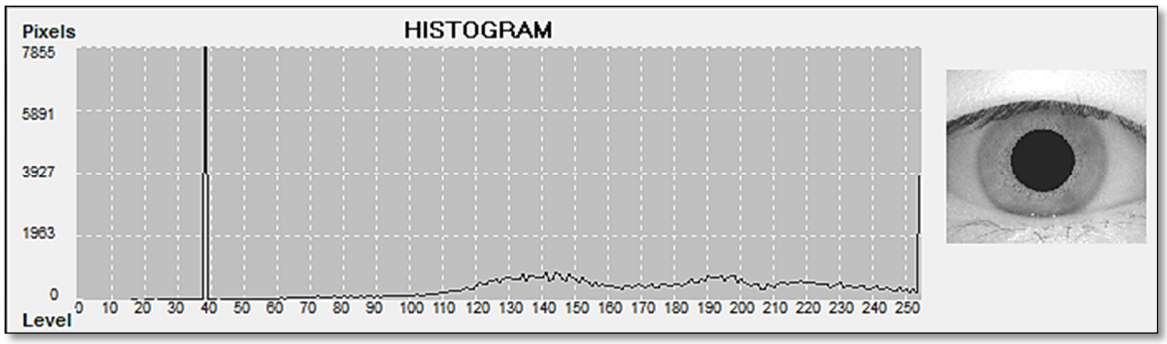


Figura 4.6. Histograma de la imagen numerada 34_2_2 en la base de imágenes CASIA V1.

Dada esta característica inherente a todas a las imágenes de la base de datos, a partir del histograma de la imagen de entrada se obtiene el nivel donde se encuentra la primera cresta; ya con el valor de ese nivel de intensidad, al umbralar la imagen se obtiene una imagen binaria con solo la información de la pupila junto con alguna información de las pestañas (figura 4.7).

Como se observa en la figura 4.7., como resultado de umbralar la imagen con el valor de umbral ya mencionado, en la mayoría de los casos no solo aparece la pupila, sino también aparece información referente a las pestañas. En todos los casos estos pixeles aparecen en forma vertical. Si ahora hacemos un análisis fila a fila, la mayor cantidad de pixeles continuos con valor 0 nos indica la posición de la pupila, la que, como su forma se asemeja a un círculo, en la fila donde se encuentre el máximo número de pixeles continuos con nivel 0 podemos asegurar que se encuentra su centro. Por medio del número de pixeles continuos y los valores de inicio y fin de esta concentración, se puede obtener el centro de la pupila, cuyo borde lo aproximamos posteriormente mediante una circunferencia. Este proceso se muestra en el diagrama de bloques de la figura 4.8.

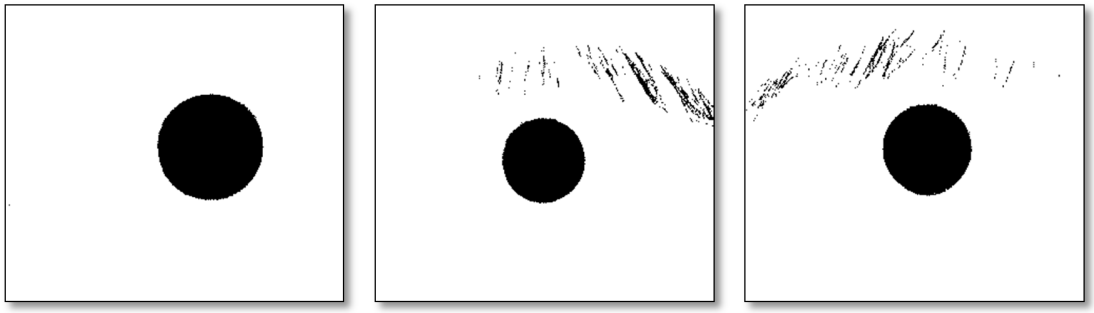


Figura 4.7. Imágenes resultado de umbralar las imágenes de iris con el valor del primer pico o cresta dentro del histograma.

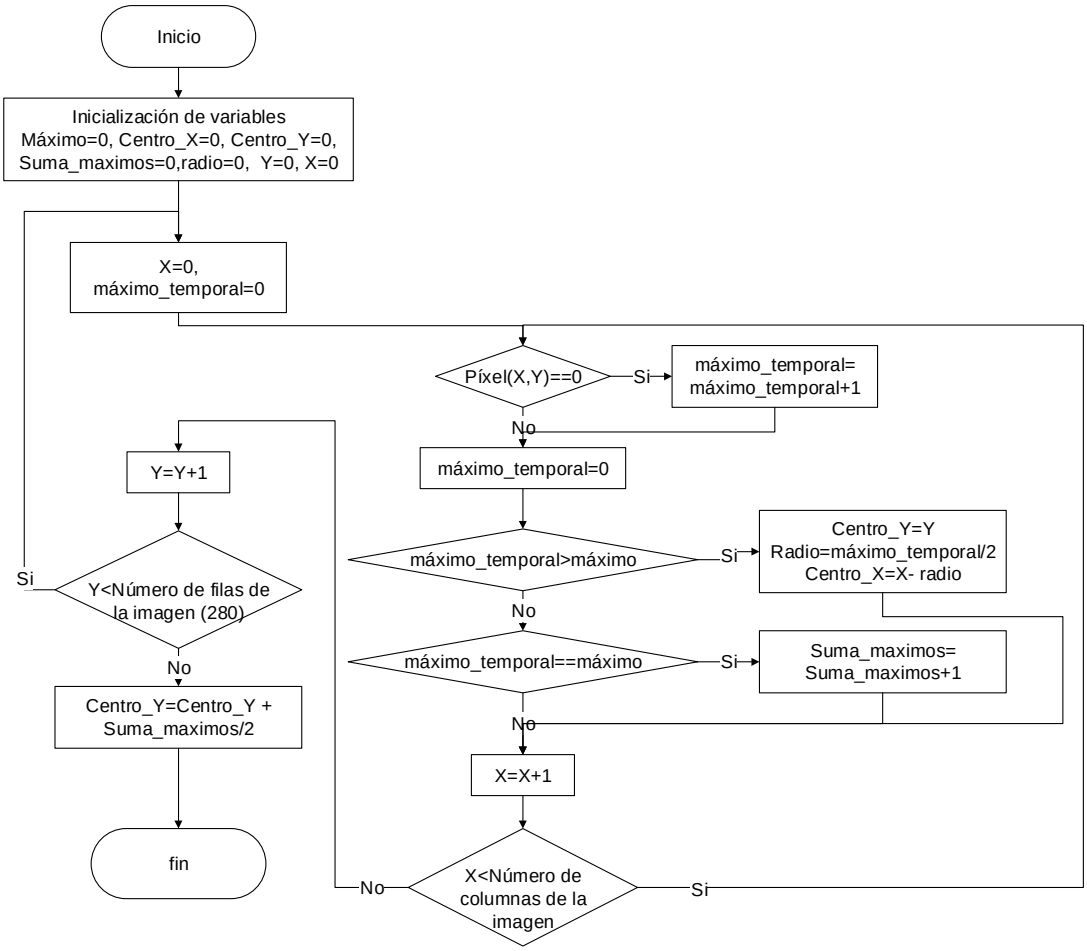


Figura 4.8. Diagrama de bloques para la detección del centro de la pupila.

Como se puede observar en el diagrama de bloques, una vez detectadas las filas con la máxima cantidad de pixeles continuos con valor 0, eventualmente podría haber más de una

fila con ese valor máximo, en cuyo caso mediante un promedio se elige la fila más centrada dentro de la pupila. Sobre esta fila se encontrará finalmente el centroide de la misma. Algunos ejemplos de la detección y aproximación del centro de la pupila se muestran en la figura 4.9.

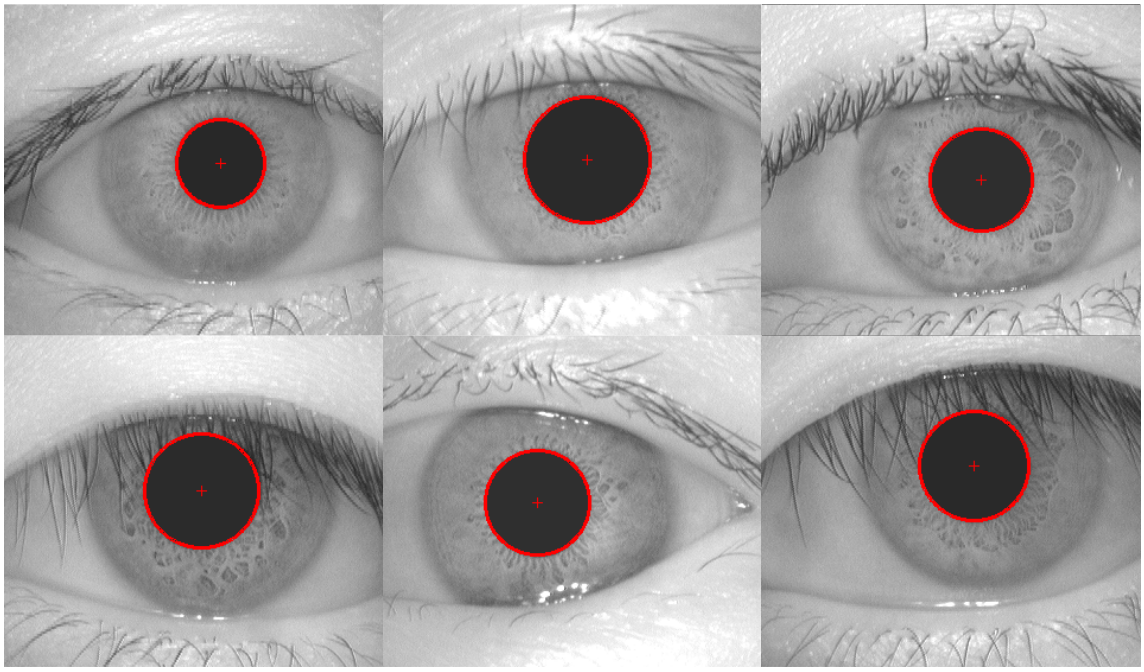


Figura 4.9 Resultados de la detección y aproximación del centro de la pupila mediante el algoritmo propuesto, en seis imágenes de la base de datos.

Es oportuno hacer notar que la forma de la pupila se considera normalmente como un círculo, pero en realidad no es así, pues en muchos casos presenta forma elíptica; debido a esto, al aproximar su borde mediante una circunferencia, hay casos en que partes de ella se salen de la circunferencia de aproximación, como los casos que se muestran en la figura 4.10. Esta característica ha sido reportada por la mayoría de los trabajos que detectan y aproximan la pupila mediante una circunferencia, por lo que no es tomado como un problema mayor.

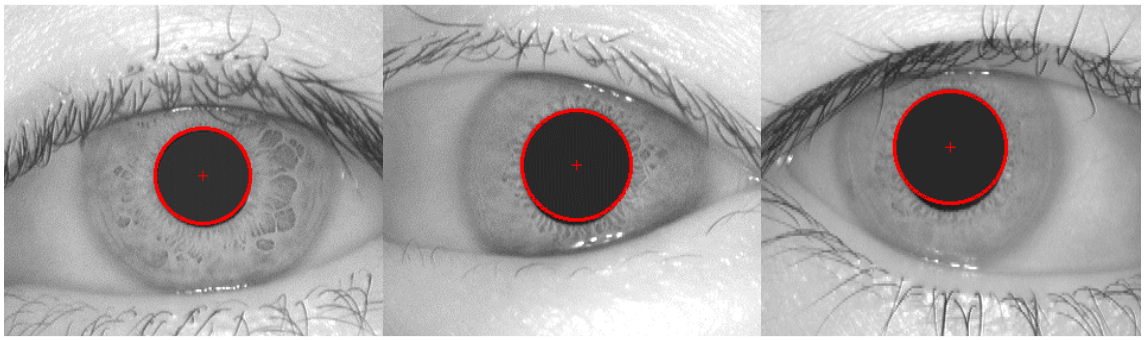


Figura 4.10. Pupilas no circulares, con su borde aproximado por medio de una circunferencia.

4.2.3. Segmentación del iris

La hipótesis base de esta parte del algoritmo para segmentar el iris en la imagen de entrada es que si podemos obtener una imagen binaria en donde los pixeles con valor 0 sean todos

aquellos que en su mayoría pertenezcan al iris y los pixeles con valor 1 sean todos aquellos pixeles que no pertenezcan al iris (ver figura 4.2). Dado el centro de la pupila previamente obtenido, podemos aproximar el iris mediante una circunferencia siempre y cuando obtengamos de la imagen binaria un punto frontera del iris.

Un punto importante e indispensable tener en cuenta en esta parte del algoritmo consiste en filtrar la imagen de entrada mediante un filtro promedio (o de promediado) con una máscara de 3x3 pixeles, donde el píxel de la imagen toma el valor del píxel central el que se hace igual al promedio de sus 8-vecinos y de él mismo; con ello se reduce el ruido aditivo presente en la imagen. A este proceso se le llama suavizado de la imagen. Así, al umbralar se obtiene una imagen más limpia (figura 4.11b) mientras que si no se realiza el filtrado se obtiene una imagen umbralada más ruidosa (figura 4.11a).

(a) (b)

Figura 4.11. Imagen de iris umbralada. Corresponde (a) a una imagen que fue umbralada sin ser suavizada previamente y (b) a una imagen que primero fue suavizada y después umbralada.

4.2.3.1. Umbralado de la imagen

Una de las características de las imágenes de iris con las que se realizó este trabajo, es que en promedio los niveles de intensidad de los pixeles que pertenecen al iris y los pixeles que lo rodean por los costados (pertenecientes a la esclerótica) están muy cercanos, por lo que para obtener un umbral adecuado para la segmentación, primeramente hay que mejorar el contraste entre lo pixeles que pertenece al iris y los que no pertenecen.

Tomando en cuenta lo anterior, se obtuvo un buen mejoramiento de contraste realizando una resta limitada de la imagen original filtrada y su imagen negativa; ya que con esta operación se deja en nivel 0 a todos aquellos pixeles que se encuentran por debajo del nivel de intensidad 128, mientras que los pixeles que tienen un valor de intensidad mayor que o igual a 128, se les resta un valor cada vez menor mientras más alto es su nivel, es decir, si al nivel 128 se le resta 127, el valor del píxel resultante es 1, mientras que si el valor del píxel es 255, se le resta 0 por lo que el valor del píxel queda igual.

Esta forma de mejoramiento de contraste dio buenos resultados en las imágenes de la base de datos utilizada, ya que en promedio los valores pertenecientes al iris se encuentran rondando el valor 128, mientras que todos lo demás pixeles que lo rodean se encuentran por encima de esos valores. Como resultado de este mejoramiento de contraste se obtienen

imágenes donde se distingue más claramente los valores de intensidad entre lo que es el iris y todo lo que lo rodea (figura 4.12).

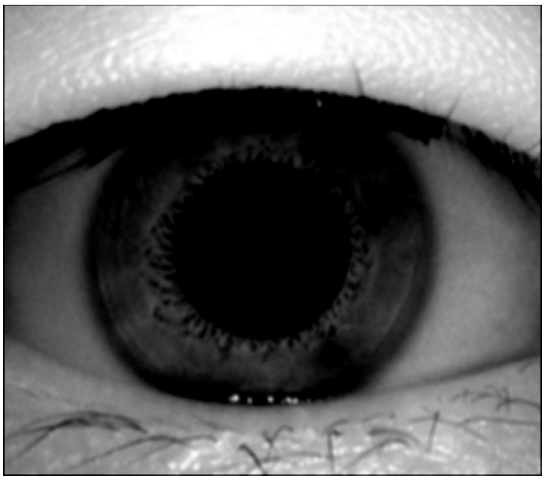


Figura 4.12. Resultado del mejoramiento de contraste de la imagen 34_2_2 filtrada.

En la figura 4.13 se muestra el histograma de la imagen a la que se le realizó el mejoramiento de contraste, donde además de los picos de los niveles mínimo (0) y máximo (255), se aprecian 2 nuevos picos, el primero debido a los pixeles que son parte del iris y el segundo a todo lo demás que lo rodea (esclerótica) ya que los niveles de intensidad de la piel de los párpados en la imagen original tienen los niveles más altos, esos pixeles que lo representan se quedan prácticamente con los mismos altos niveles de intensidad.

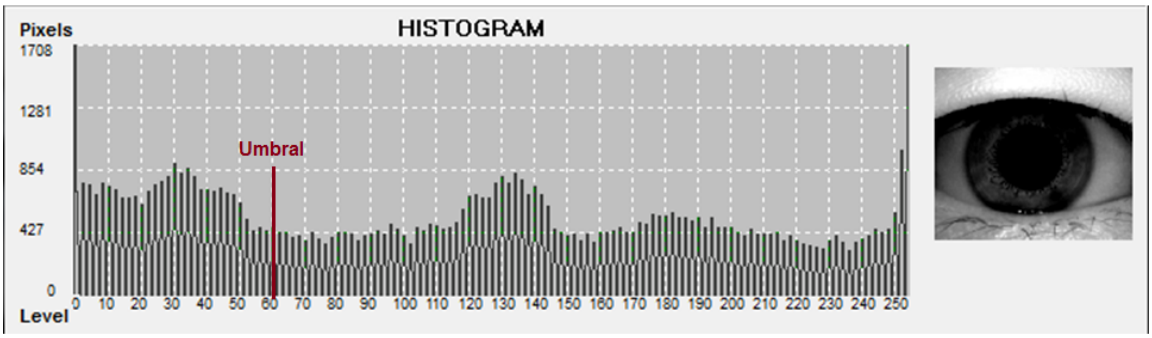


Figura 4.13. Histograma de la imagen resultante del mejoramiento de contraste de la imagen 34_2_2 filtrada.

Para determinar el valor de umbral automáticamente en cada imagen de entrada, se requiere un valor de umbral que no se encuentre muy lejano de la concentración de pixeles de la primera cresta, ya que mientras más alto es el valor de umbral, se incluye en la imagen binaria pixeles con valor 0 que no son parte del iris; sin embargo, si se toma un valor de umbral muy cercano al valor que corresponde a la máxima concentración de pixeles de la primera cresta, el efecto es contrario, es decir, muchos pixeles que son parte del iris toman el valor 1, como si no fueran parte del iris. De manera experimental se obtuvieron los mejores resultados en el umbralado de las imágenes determinando automáticamente como valor de umbral el valor obtenido al detectar la pupila (sección 4.2.2) más $^{1}/_{4}$ del mismo valor, es decir, $^{5}/_{4}$ del valor de umbral utilizado en la detección de la pupila.

Una vez obtenido el umbral, se umbrala la imagen dando como resultado imágenes binarias donde el valor de 0 es lo que está por debajo del nivel de umbral (idealmente sólo la información del iris), mientras que los pixeles con nivel 1 son los pixeles cuyo valor de intensidad está por encima del valor de umbral (idealmente todos aquellos pixeles que no pertenecen al iris), figura 4.14.

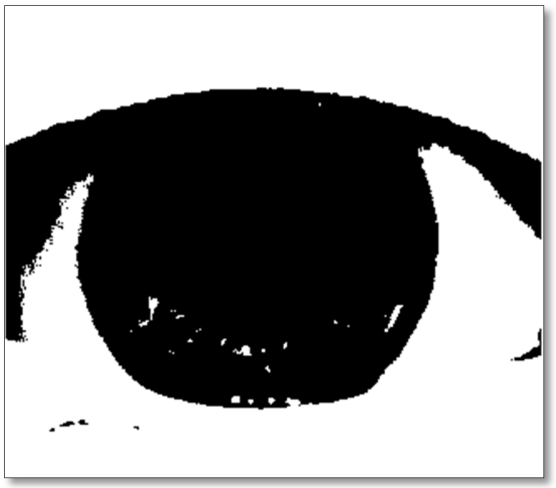


Figura 4.14 Umbralado de la imagen 34_2_2.

Es oportuno hacer notar que esta manera de umbralar las imágenes da buenos resultados (sección 5.1.2), ya que debido a la diversidad de tomas y de ojos de la cual se conforma la base de datos utilizada, es posible por lo menos detectar un punto frontera del iris en cada imagen y así aproximar el iris mediante la circunferencia correspondiente.

4.2.3.2. Segmentación y aproximación del iris mediante una circunferencia

Retomando la hipótesis para la segmentación y aproximación, si al umbralar una imagen en niveles de gris de la base de datos logramos en la imagen binaria que la mayor cantidad de información que compone al iris tenga valor 0 (en negro) y todo lo demás el valor 1 (en blanco) (figura 4.14), al detectar los puntos de la frontera del iris éste se puede aproximar mediante una circunferencia lo que concluye su segmentación.

Para esto, una vez obtenida la imagen umbralada, se hace una búsqueda fila a fila del punto donde se pasa de una secuencia de 4 pixeles con valor 1 a una secuencia de 4 pixeles con valor 0; a este punto de transición lo denotamos como punto frontera del iris. Para limitar la búsqueda de los puntos frontera del iris, la búsqueda se realiza partiendo de la fila en la que se encuentra el centro de la pupila más la mitad del valor de radio de la circunferencia de aproximación de la pupila (debido a que la parte inferior del iris está menos contaminado por el ruido de las pestañas), finalizando en la fila 230 de la imagen (este valor es fijo y sólo para limitar la búsqueda), Por otro lado,se limita la búsqueda solo a los costados del iris (figura 4.15) donde del lado izquierdo la búsqueda comienza desde la columna $X_c - 120$ hasta $X_c - 80$ (X_c es la columna correspondiente al centro de la pupila), y de manera análoga del lado derecho la búsqueda empieza de la columna $X_c + 120$ y termina en $X_c + 80$. Los valores 120 y 80 han sido tomados de [50], donde al llevar a cabo su algoritmo los resultados que obtienen sobre esta base de datos es que el mayor radio del borde exterior del iris es de 120 pixeles mientras que el menor es de 80 pixeles.

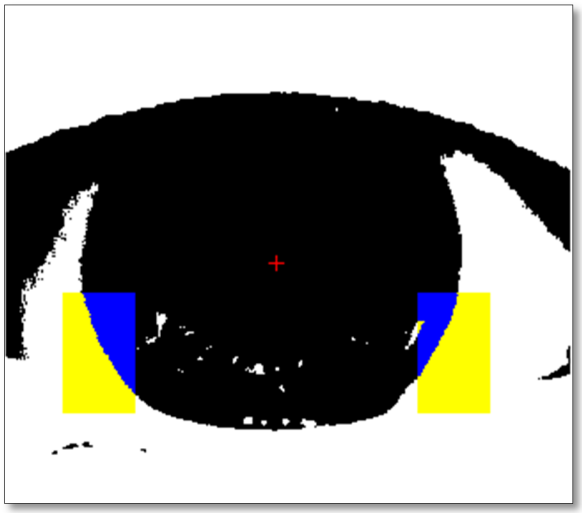


Figura 4.15. Área de búsqueda de los puntos frontera del iris.

Dadas las características de las imágenes de la base de datos y el algoritmo propuesto para umbralarlas (sección 4.2.3.1), en las imágenes resultantes se presentan 3 casos base para lograr la aproximación requerida para trazar la circunferencia que limita el iris exteriormente:

- Si se encuentra un punto frontera del iris en ambos lados sobre la misma fila, se traza entonces una línea recta entre ellos la cual constituye una recta secante; por definición y por las propiedades de la secante y de la circunferencia que la contiene, es posible aproximar el centro del iris a partir de la componente Y del centro de la pupila (fila) y determinando la componente X como la distancia media entre los 2 puntos de la secante.

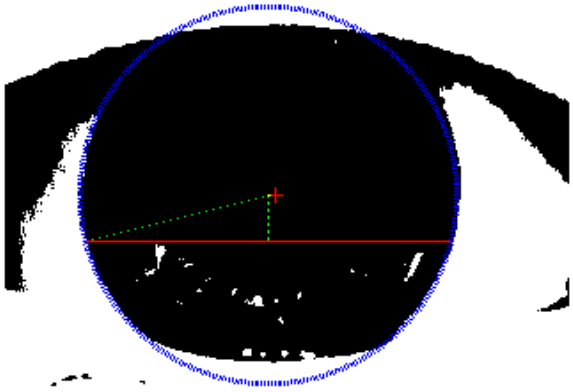


Figura 4.16. Aproximación del borde externo del iris mediante una circunferencia cuando se tienen 2 puntos frontera en la misma fila.

Una vez que se tiene el centro y ambos puntos, mediante el teorema de Pitágoras (Ec. 4.1), es posible determinar el radio de la circunferencia.

$$radio = \sqrt{(X_p - X_c)^2 + (Y_p - Y_c)^2} \qquad \text{(Ec. 4.1)}$$

Donde X_c y Y_c son las componestes X y Y respectivamente de las coordenadas del centro del iris y X_p y Y_p de uno de los puntos obtenidos.

Como se observa en la figura 4.16, la línea roja es la recta secante que se traza entre los 2 puntos frontera encontrados en la misma línea; sobre ella y en el punto medio se traza una perpendicular (que se muestra punteada), la que al interceptarse con la recta que se traza desde uno de los puntos frontera hacia el centro de la pupila, posiciona el nuevo centro. El radio de la circunferencia exterior del iris está dado entonces por la línea punteada (verde) trazada desde el nuevo centro hasta alguno de los puntos frontera. En la figura se muestra el radio trazado hacia el punto frontera izquierda.

- Si no es posible encontrar un punto frontera en ambos lados en la misma fila, pero se encuentra al menos un punto frontera en uno de los lados, entonces el centro de la papila previamente determinado se toma como centro del iris. Dado el punto encontrado y mediante la Ec. 4.1, se obtiene el radio del círculo con el cual se aproxima la circunferencia exterior del iris (figura 4.17).
- Si no se lograron puntos frontera en ningún barrido en ambos lados, entonces el centro de la pupila se deja como referencia y centro del iris; en este caso, el radio de la circunferencia exterior del iris toma el valor de 120 pixeles, ya que es el valor máximo de radio reportado en la mayoría de los trabajos realizados [50].

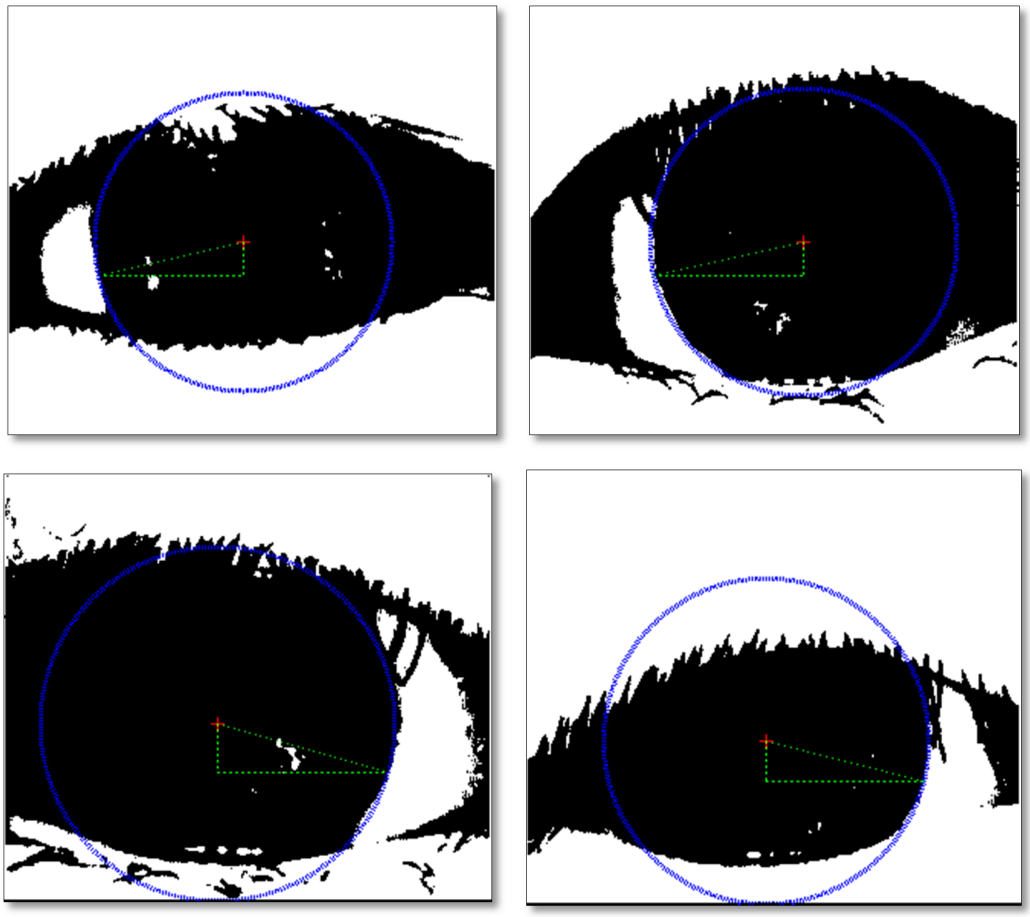


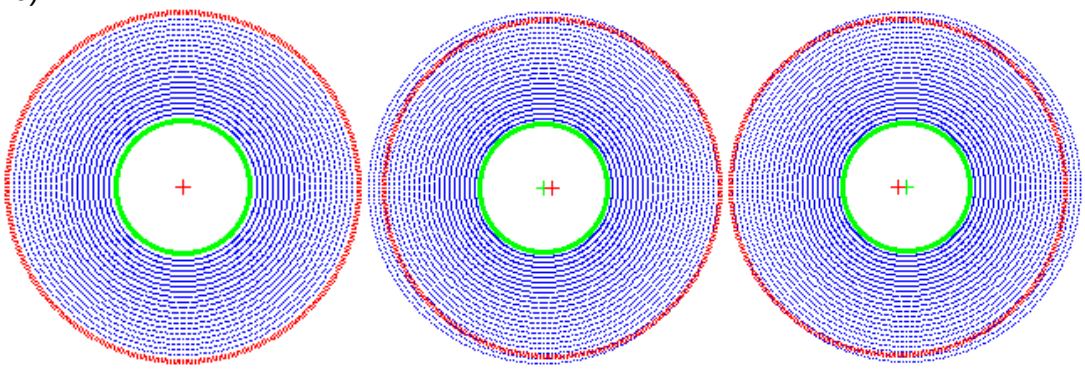
Figura 4.17. Ejemplos de aproximación de la circunferencia exterior del iris cuando solo se tiene un punto lateral.

4.2.4. Normalización del iris

Una vez localizado el iris en la imagen de entrada, se genera una nueva imagen con sólo los pixeles que lo componen. Esto se logra transformando las coordenadas de la imagen capturada (x, y) a un sistema de coordenadas polares (r, θ) y determinando para todos los ángulos posibles los respectivos radios. Con estos valores se crea una nueva imagen a la cual se le denomina plantilla.

La plantilla de identificación adquiere su tamaño de acuerdo a la información del iris que se segmenta en cada una de las muestras posibles; para ello es preciso definir el tamaño de la misma en cada imagen, el cual depende directamente del radio de la circunferencia exterior del iris y el radio de la circunferencia que bordea la pupila obtenidos previamente; donde cada fila representa un radio *r* el cual está dentro de la región del iris limitada por ambos bordes, mientras que las columnas representan los diferentes ángulos θ que forman el círculo correspondiente al radio *r* mencionado. Este concepto se ilustra en la figura 4.18 donde se muestra con un círculo verde la circunferencia que bordea la pupila y su respectivo centro (pequeña cruz también en verde), la circunferencia roja que bordea exteriormente el iris y su respectivo centro (mostrado con una cruz roja), y las diferentes circunferencias azules referidas todas al centro de la pupila. Ahora debemos definir el rango de radios entre los cuales las muestras serán tomadas; tomando como radio máximo, la distancia más grande entre el centro de la pupila y los puntos de la circunferencia exterior del iris a 0° y 180°. De esta manera se garantiza que de no ser concéntricos el iris y la pupila, se tomarán todas las muestras posibles de información del iris, aunque en algunos casos un bajo porcentaje de muestras cayera fuera de la región localizada del iris.

La imagen normalizada o plantilla de identificación sólo se obtiene de los 180° de la parte baja del iris, es decir de 180° a 360°, debido a que en un alto porcentaje de las imágenes la parte superior del iris se ve afectada por el ruido de las pestañas y el párpado (figura 4.19).



(a) (b) (c)

Figura 4.18. Conversión de coordenadas polares a coordenadas cartesianas. (a) Cuando las circunferencias que bordean la pupila y el iris son concéntricas; (b) y (c) Cuando las circunferencias que bordean la pupila y el iris no son concéntricas.

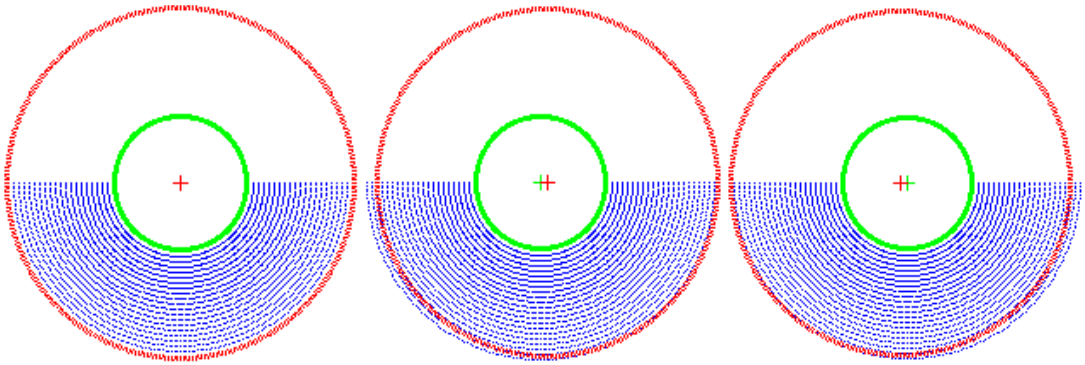


Figura 4.19. Conversión de coordenadas polares a coordenadas cartesianas (de 180° a 360°). (a) Cuando las circunferencias de la pupila y el iris son concéntricas; (b) y (c) Cuando las circunferencias de la pupila y el iris no son concéntricas.

4.3. Implementación del algoritmo en FPGA

Retomando al objetivo de este trabajo de implementar el algoritmo de procesamiento de imágenes de iris humanos desarrollado y presentado en la sección 4.2 en un FPGA deben crearse las condiciones más ventajosas. Para lograr este objetivo, el trabajo se desarrolló bajo la metodología de crear un sistema de codiseño de hardware y software mediante las herramientas de desarrollo de sistemas en chips programables (SOPC, del inglés *System on a Programmable Chip*) proporcionadas por la empresa Altera para el desarrollo de aplicaciones con sus tarjetas FPGA.

El desarrollo de esta parte del trabajo fue dividido en 2 partes: primero el desarrollo de una arquitectura base que permite la lectura y escritura de imágenes en formato BMP y JPG a través de un medio externo como lo es una tarjeta de memoria SD (del inglés *Secure Digital*), la visualización de las imágenes por medio de una pantalla o *display* a través del puerto VGA y la medición de los ciclos de reloj que conlleva el procesamiento a través de un módulo contador de ciclos. Esto se logró por medio de una combinación de módulos de hardware y la implementación de un procesador suave (NIOS II), este último programado en el lenguaje de programación C/C++ (software).Como segunda parte se llevó a cabo el desarrollo de módulos (memorias, operaciones, máquinas de estados) que realizan las operaciones de procesamiento y manipulación de los datos de la imagen en hardware; finalmente, ambas partes se unieron mediante puertos paralelos de entrada y salida.

4.3.1. Arquitectura base

El desarrollo de una arquitectura base para la implementación del algoritmo de procesamiento de imágenes en FPGA es esencial, ya que permite la entrada, salida, manipulación y visualización de los datos (imágenes) en el sistema implementado en el FPGA.

Lo que se requiere de esta arquitectura base es la posibilidad de introducir (leer) y sacar (escribir) imágenes en el FPGA, para lo cual se hace uso del *socket* para memoria SD que posee la tarjeta de desarrollo, del módulo de hardware proporcionado por el programa universitario de Altera (*University Program*) [51], así como de las bibliotecas necesarias proporcionadas por este programa para el manejo de este módulo y de los archivos para los sistemas de archivos FAT16 y FAT32 en las tarjetas de memoria SD.

También se requiere visualizar las imágenes que se introducen y las resultantes del procesamiento. Para esto, al igual que el módulo anterior, se incluyeron en la arquitectura base los módulos necesarios proporcionados por el software de desarrollo (Quartus II) que manejan la memoria SRAM como *buffer* de la imagen de salida, esto es, mediante el procesador NIOS II se escriben los datos en la memoria SRAM y mediante los módulos de acceso directo a memoria y las memorias FIFO (Primero en entrar, primero en salir; del inglés *First In, First Out*), y de acuerdo a la posición del píxel requerido por las señales de sincronía para el manejo del *display*, se va accediendo a las localidades de memoria requeridos y mediante la sincronía con estas memorias se visualiza en la pantalla la imagen almacenada en dicho *buffer* (figura 4.20).

Los módulos de hardware requeridos para el manejo de la memoria SRAM, del acceso directo a memoria, las memorias FIFO y la generación de los pulsos de sincronía para el manejo del *display* con la resolución de 640x480 pixeles, a 24 bits por color (8 bits para cada plano) y la lectura de los datos del *buffer*, son proporcionados por el software de desarrollo Quartus II a través de su programa universitario [51]. En este caso, se generó la biblioteca "vga.h" para el manejo de la memoria SRAM (*buffer*), con las funciones principales para limpiar la pantalla (que consiste en escribir en todas las localidades de memoria de los pixeles correspondientes el mismo valor, ya sea 0 para el caso del color negro u otro de los 2^{24} valores de color posibles) y escribir y leer una imagen en la memoria y por lo tanto mostrarla en la pantalla.

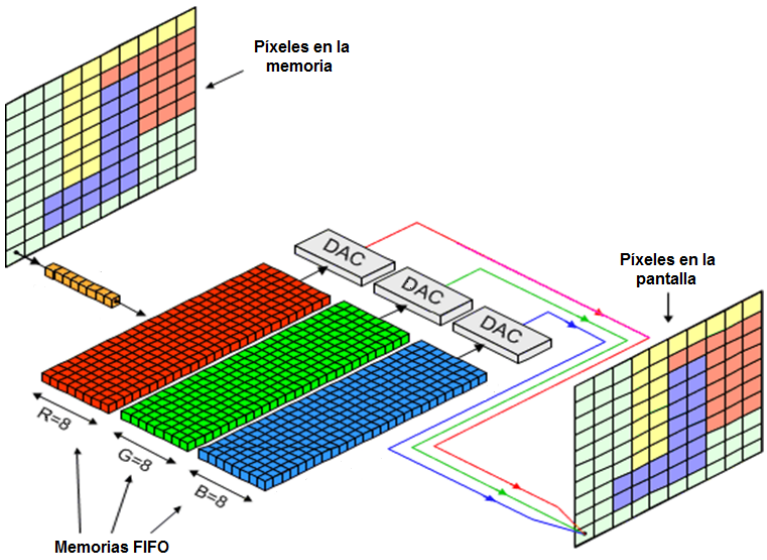


Figura 4.20. Esquema general del controlador de video VGA y sus módulos.

Para corroborar que "…cualquier algoritmo puede alcanzar su máxima velocidad si es implementado en hardware…" [12], es necesario implementar en la arquitectura base y por lo tanto en el FPGA un módulo que permita medir la velocidad o los ciclos de reloj requeridos por el procesamiento. Este módulo consta, de 3 entradas (figura 4.21), la primera de ellas es la entrada de reloj del sistema, la segunda la entrada de habilitación, la cual cuando se encuentra con valor alto ('1' lógico) se activa el contador y cuenta todos los ciclos a partir del flanco de subida de la señal, mientras que si el valor de entrada está en bajo ('0' lógico) se desactiva el contador y se retiene su ultimo valor; y finalmente la entrada *Reset* que regresa el contador a 0. El contador de 64 bits está dividido en 2 registros y por lo tanto tiene 2 salidas separadas de 32 bits, lo que nos da la posibilidad de contar hasta 2^{64} ciclos

de reloj. Para la visualización rápida y sencilla de cada procesamiento, aparte de los 2 registros del contador, se implementaron 8 decodificadores de 7 segmentos para poder desplegar el valor del registro de los primeros 32 bits en hexadecimal con los *displays* de 7 segmentos que tiene la tarjeta de desarrollo.

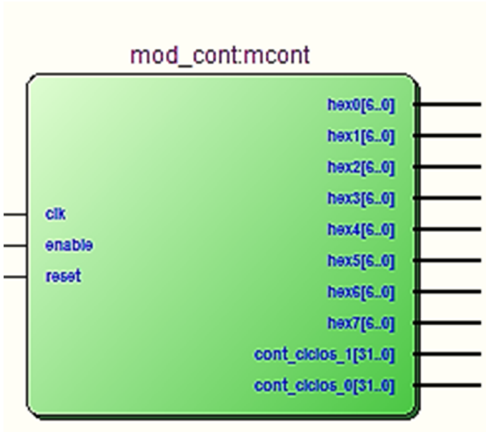


Figura 4.21. Bloque funcional del módulo contador de pulsos de reloj.

La integración e interconexión de cada uno de los módulos con el procesador suave NIOS-II de la arquitectura base, donde la memoria SDRAM es utilizada como memoria de instrucciones y de datos del procesador NIOS II, lo que de manera general se describe en la figura 4.22, se realizó con la herramienta Qsys que se encuentra dentro del software de desarrollo Quartus II proporcionado por Altera. La integración del software que gobierna al procesador y el manejo de todos los módulos se realizó con la herramienta *Eclipse IDE for C/C++ Developers*, también proporcionada como software de desarrollo.

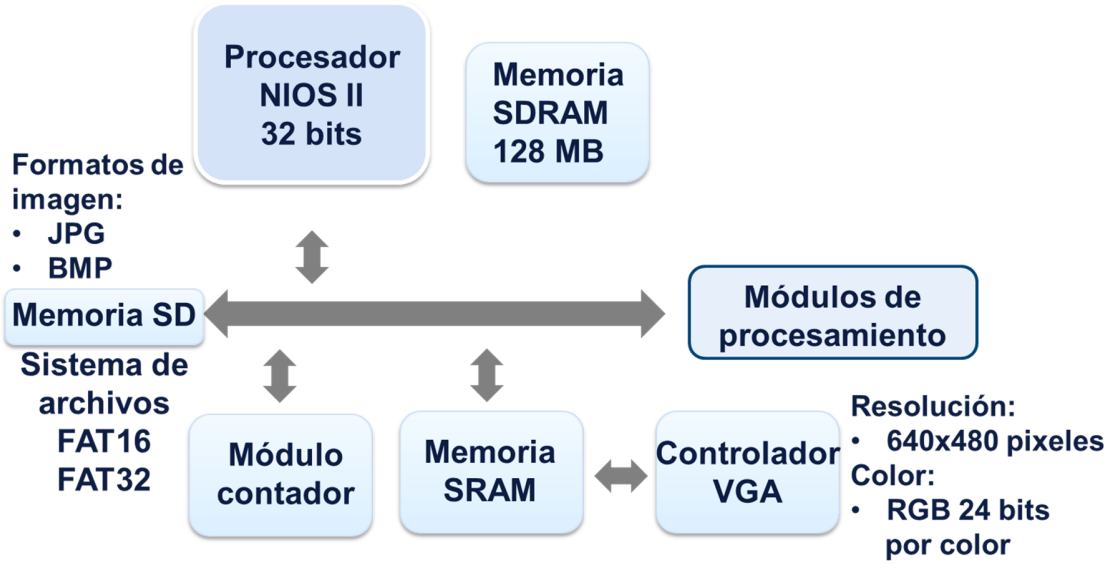


Figura 4.22. Esquema general de la arquitectura base y sus módulos.

En la figura 4.22 se describe de forma general la arquitectura base para la implementación del algoritmo en FPGA, la que consta de los módulos ya mencionados y los módulos de procesamiento para la implementación y realización del algoritmo en FPGA.

4.3.2. Módulos de hardware

Dada la arquitectura base, donde por medio del procesador NIOS II y la lectura de las imágenes de iris extraídas de la tarjeta de memoria SD, mismas que son almacenadas en la memoria SDRAM, pueden implementarse tres formas de hacer el procesamiento con o sin la necesidad de integrar más módulos externos. 1. Se puede hacer implementando el algoritmo o parte del mismo en software de manera secuencial, donde el tiempo de ejecución depende la los ciclos de reloj que se requieran para cada función que se implementa; 2. Para mejorar el tiempo de ejecución implementar parte del procesamiento conjugando el software con el hardware, es decir, ejecutando un programa en el procesador y al mismo tiempo haciendo escrituras y lecturas de módulos externos de hardware que le ayudan a hacer cálculos; y 3. Creando módulos en hardware que dada una señal de inicio reciba los datos necesarios hasta que mediante una bandera o interrupción se le dé aviso de que ha finalizado su tarea. Estas 3 formas de realizar e implementar el algoritmo en FPGA se describen en las siguientes secciones, cuyos resultados se detallan en el capítulo 5.

El módulo de filtrado y el umbralado de la imagen, el módulo para obtener el centro de la pupila y el módulo para obtener los puntos frontera del iris en la imagen binaria, se implementaron como módulos externos al procesador, descritos por completo en el lenguaje de descripción de hardware VERILOG, dependiendo solo del procesador para recibir la imagen en una memoria externa y para activar la entrada de inicio del procesamiento. Estos 3 módulos, acceden a la misma memoria externa, cada uno de ellos en su respectivo turno (controlado por un multiplexor). Primero se realiza la convolución y el umbralado, y mientras este módulo está trabajando, mediante su salida *Working* se le indica al multiplexor que la memoria está siendo utilizada por él; cuando este módulo culmina su tarea desactiva su salida *Working* y activa su salida de *Done*; misma que es la entrada *Start* del módulo de centro de pupila, que al igual que el primer módulo, tiene las salidas *Working* y *Done* , con las cuales le indica a los multiplexores que es él quien está accediendo a la memoria; finalmente, su salida *Done* da el Inicio del tercer módulo que detecta los Puntos de iris.

Las imágenes utilizadas en este trabajo provenientes de la base de datos son de 320x280 pixeles. Para la implementación de los primeros 3 módulos de hardware, fue necesaria la implementación de una memoria o memorias externas donde es almacenada la imagen a procesar y es posible acceder a su contenido de manera concurrente para poder realizar varias operaciones al mismo tiempo; para ello se implementaron 280 memorias de 512 localidades de memoria de 8 bits cada una, debido a que la imagen es en niveles de gris.

Las 280 memorias contienen dos puertos de lectura/escritura, es decir tienen dos puertos con los cuales se puede tanto leer como escribir al mismo tiempo, lo que nos permite acceder a las 280 filas de la imagen simultáneamente. Esta característica junto con el módulo de filtrado y umbralado, permite pasar 278 máscaras de convolución (la convolución no se realiza en los 2 pixeles de los limites tanto superior como inferior de la imagen) al mismo tiempo para lograr la convolución de la imagen completa en tan solo 327 ciclos de reloj.

Una característica importante que es posible gracias a la implementación del algoritmo con la arquitectura base y una memoria externa, es que la imagen puede ser cargada en la memoria externa al mismo tiempo que es leída y extraída de la tarjeta de memoria SD, así como también de manera paralela es obtenido su histograma y el valor de umbral para la segmentación de la pupila.

4.3.2.1. Filtrado y umbralado

Dada la imagen en la memoria externa (conjunto de 280 memorias, una por cada fila), mediante una máquina de estados con 4 estados (figura 4.23.) se realiza la convolución para filtrar la imagen y al mismo tiempo mediante los sumadores y comparadores de magnitud implementados en hardware obtener la imagen binaria con el iris segmentado. Se implementó una máquina de estados por cada memoria o fila.

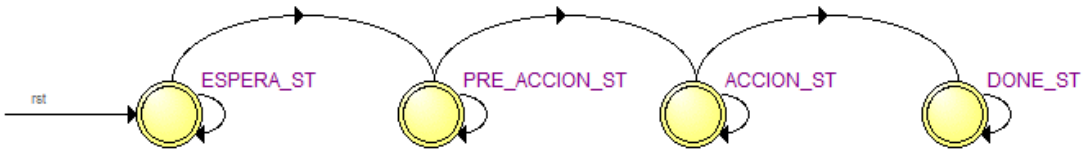


Figura 4.23. Máquina de estados implementada para el filtrado y el umbralado de la imagen.

La máquina de estados se encuentra inicialmente en el estado ESPERA_ST, en el que se inicializan todas las variables a utilizar y se espera a que la entrada Inicio (*go*) sea activada. Una vez que esto tuvo lugar la máquina pasa al estado PRE_ACCION_ST, en el que se extraen los valores de las primeras 2 columnas de las 280 filas de la imagen al mismo tiempo, las que son introducidas en un registro de corrimiento de 3 localidades con 8 bits por localidad; hay un registro de este tipo por cada memoria, es decir, 280 registros. Una vez que se extraen las primeras 2 columnas de cada fila, se pasa al estado ACCION_ST, donde se carga la tercera columna y ya habiendo llenado las 3 localidades, con la ayuda de los 280 registros se realiza simultáneamente la convolución y se hacen las sumas y restas necesarias para obtener en un solo ciclo de reloj la imagen umbralada (figura 4.14). Este proceso se repite hasta meter el valor de la última columna de la imagen (la columna 320) al registro y realizar la última convolución y el último umbralado. Posteriormente se pasa al estado *DONE_ST* donde se detiene la máquina, se activa la salida Hecho y se desactiva la salida Trabajando (*Working*).

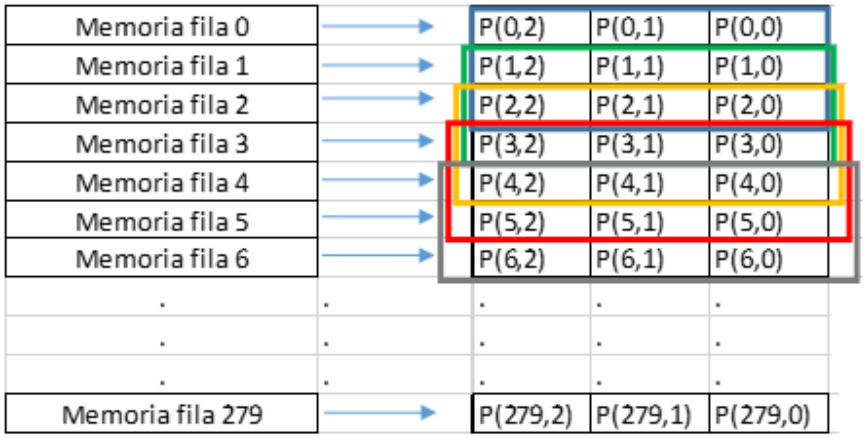


Figura 4.24. Ejemplo general del paso de datos de las memorias a los registros y las 278 convoluciones concurrentes.

En la figura 4.24 se muestra esquemáticamente el proceso en el que se realizan las 278 convoluciones y umbralados de la imagen de manera concurrente, lo que permite realizar este procesamiento en alrededor 327 ciclos de reloj. Así mismo, simultáneamente, con la misma máquina de estados, pero con otro circuito comparador y dado previamente el umbral para obtener la pupila (obtenido al leer y cargar la imagen en la memoria externa), mediante un circuito comparador y contadores por cada fila (memoria), se obtiene el número

máximo de pixeles continuos por fila cuyo valor está por debajo o es igual al valor de umbral, para que en el próximo módulo se pueda determinar el centro y el radio de la circunferencia que bordea la pupila.

La obtención de los pixeles continuos para determinar el iris, así como la extracción de datos para incluirlos en los registros de corrimiento y las convoluciones con las máscaras de 3x3 para obtener el promedio de los 8-vecinos de un píxel, se realiza también de manera concurrente, es decir, simultáneamente en las 280 memorias y registros, gobernado todo por la misma entrada de reloj.

4.3.2.2. Centro de la pupila

Del proceso del módulo precedente se obtiene el número máximo de pixeles continuos con niveles menores o iguales que el umbral de la pupila y con esta información por fila, se obtiene como se menciona en la sección 4.2.2., el centro y el radio de la circunferencia para aproximar el borde de la pupila con una circunferencia.

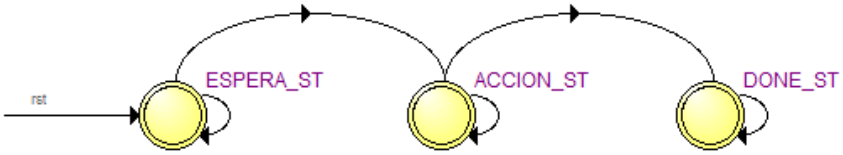


Figura 4.25 Máquina de estados implementada para la obtención del centro de la pupila.

El centro y el radio de la circunferencia para la aproximación del borde interno del iris se obtiene mediante una máquina de estados que contiene 3 estados (figura 4.25); en el primero de ellos ESPERA_ST, la máquina asigna a todas las variables a utilizar su valor inicial y se mantiene ahí hasta que recibe el valor “1” lógico en su entrada de Inicio (*go*); cuando esto sucede se pasa de inmediato al estado ACCION_ST donde con cada ciclo de reloj se lee el valor máximo de pixeles continuos cuyo valor está por debajo del umbral de la pupila (obtenido con el módulo anterior y almacenado en la localización de memoria 321 de cada memoria); este valor almacenado se utiliza para calcular, al igual que como se describe en la sección 4.2.2., el centro y el radio de la circunferencia externa de la pupila; una vez que se revisaron los valores de las 280 filas, se pasa al estado DONE_ST, dando como terminada la tarea de esta máquina y por lo tanto de este módulo al activarla salida Hecho (DONE).

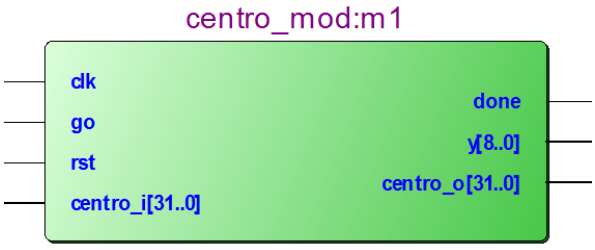


Figura 4.26. Bloque funcional del módulo con el que obtiene el centro de la pupila.

En la figura 4.26 se observa el bloque funcional del módulo a describir en esta sección; consta de sus entradas de reloj (clk), *reset* (rst), Inicio (*go*), la entrada centro_i, por donde se recibe el valor de la cuenta de cada fila; la salida *done* para indicar que terminó su tarea, la salida *y* para hacer referencia a qué fila se está refiriendo y de la cual se requiere la cuenta, y la salida *centro_o*, donde al final se asigna el resultado; en los primeros 8 bits (del 0 al 7) se almacena el radio de la circunferencia, en los bits del 8 al 15 se almacena el

número de la columna donde se encuentra el centro y en los bits del 16 al 23 se almacena la fila.

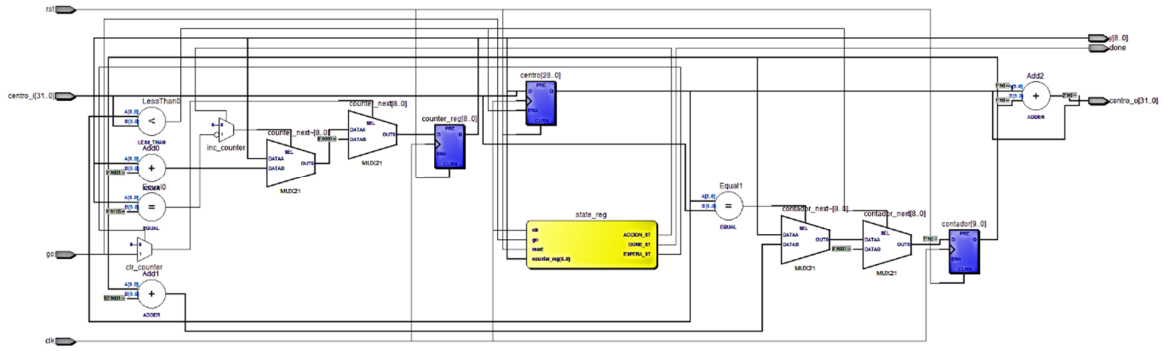


Figura 4.27. Hardware sintetizado para la implementación del módulo en FPGA.

En la figura 4.27., se muestra el circuito sintetizado, que consta de la máquina de estados descrita, sumadores, comparadores de magnitud, multiplicadores y registros donde se almacenan los resultados.

4.3.2.3. Puntos del iris

Como resultado de la tarea realizada por el módulo de filtrado y umbralado, en las memorias queda almacenada la imagen binaria descrita en la sección 4.2.3.1., la que como se muestra en la figura 4.15, se definen zonas de búsqueda dentro de la misma imagen que son analizadas en busca de los puntos frontera. Al utilizar las posibilidades de concurrencia que proporciona una plataforma de hardware FPGA y debido a que por cada fila tenemos una memoria, es posible analizar todas las filas al mismo tiempo y encontrar esos puntos por fila de manera concurrente.

Al igual que el módulo de filtrado y umbralado, se implementó una máquina de estados por cada memoria (figura 4.28). Al igual que en las otras máquinas de estados implementadas, en el primer estado (ESPERA_ST) se inicializan las variables, los registros y las conexiones utilizadas; cuando se activa la entrada Inicio (*GO*), se pasa al segundo estado donde el valor del píxel de la imagen binaria (‘0’ o ‘1’) se va almacenando en un registros de corrimiento a la derecha e introduciendo valores por la izquierda y como el análisis de la imagen en ambos lados se hace de afuera hacia adentro, cuando el valor del registro es “00001111”, indica que pasamos de una concentración de niveles de blanco (unos) a una concentración de niveles de negro (ceros), ese punto se toma como punto frontera del iris. Si se cumple esta condición se pasa al tercer estado (TERMINA_ST) donde solo se dejan tener efecto los ciclos para completar todos los ciclos requeridos, antes de pasar al estado final al mismo tiempo que todas las demás máquinas de estados; de lo contrario, si se llega al límite de la zona de análisis sin cumplirse la condición, se pasa directamente al estado final dejando las salida del punto en cero, pero la salida Hecho (DONE) en alto para indicar que terminó su tarea.

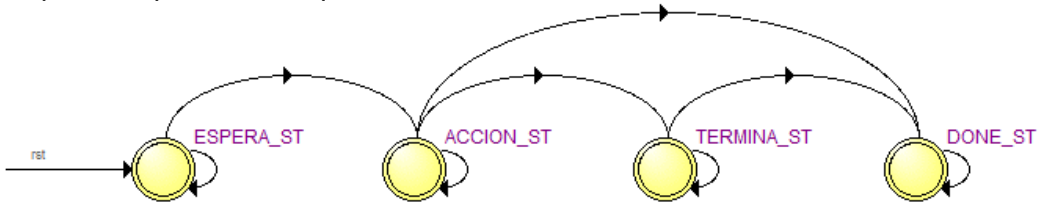


Figura 4.28. Máquina de estados implementada para la obtención de los puntos frontera del iris.

En la figura 4.29., se muestra el bloque funcional del módulo implementado, en el que del lado izquierdo se encuentran las entradas y del lado derecho sus salidas. Como entrada tiene el centro de la pupila para obtener los límites de la búsqueda; el valor del píxel dada la dirección en la imagen, es decir, por la salida *pixel_x* se tiene la salida de la memoria de imagen; y finalmente las salidas *done* y *working* son banderas para indicar que el procesador está realizando su tarea o que ya ha terminado de realizarla. La posición de los puntos frontera resultantes en la fila se guardan en el registro de salida llamado *cord*, donde las coordenadas del punto de la izquierda se guarda en los 16 bits más significativos, mientras que las coordenadas del punto de la derecha se guardan en los 16 bits menos significativos.

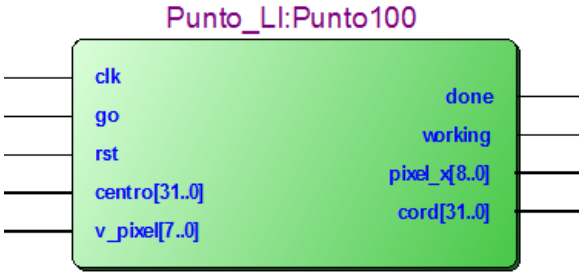


Figura 4.29. Bloque funcional del módulo con el que se obtienen los puntos frontera del iris en la imagen binaria.

Una vez obtenidos los puntos frontera del iris, así como el centro y la circunferencia de la pupila que constituye el borde interior del iris, la circunferencia con la que se aproxima el borde exterior del iris se implementó por completo en software, es decir, esa tarea es ejecutada por medio del procesador suave (NIOS II) integrado en la arquitectura base.

4.3.2.4. Normalización (Software-Hardware)

Una manera sencilla de implementación de los algoritmos es mediante la combinación de software/hardware, es decir, programar en C/C++ el algoritmo en el procesador, pero utilizando un módulo de hardware auxiliar que haga los cálculos, y solamente leer y escribir en el módulo auxiliar para enviar y recibir datos para seguir procesándolos con el procesador.

El control de los ángulos y los radios para muestrear la imagen se implementó en el procesador NIOS II, por medio del lenguaje C/C++, pero para que el procesador no realice los cálculos de senos y cosenos de acuerdo al ángulo seleccionado, así como las multiplicaciones requeridas y para realizar las operaciones de manera más rápida, se implementaron en hardware memorias ROM con los valores de los senos y cosenos de 0° a 90°; mediante el mismo módulo de hardware, cuando el procesador cambia el valor del ángulo y/o del radio se calculan las localizaciones de memoria a las cuales el procesador debe acceder dentro de la imagen para su muestreo.

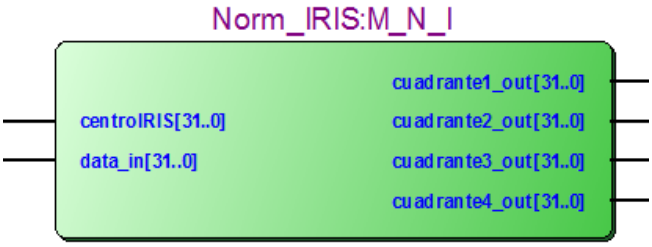


Figura 4.30. Bloque funcional del módulo auxiliar para la normalización de la imagen.

En la figura 4.30 se muestra el bloque funcional del módulo externo implementado para la realización de los cálculos de las direcciones para el muestreo de la imagen original. Debido a que en este trabajo se propone solo trabajar, muestrear y normalizar solo los 180° de la parte inferior del iris, el cuadrante 3 y el cuadrante 4 se dividen en dos partes, es decir, se muestrea al mismo tiempo los 2 cuadrantes, pero con dos muestras al mismo tiempo por cuadrante, uno que empieza en 0° y otro en 90° dentro de cada cuadrante. El módulo consta de dos entradas, en la primera de ellas recibe los datos correspondientes al centro de la pupila y en la segunda recibe los datos correspondientes al radio y al ángulo para hacer la conversión a las direcciones de memoria para muestrear la imagen. Como salida tiene las cuatro direcciones para tomar cuatro muestras al mismo tiempo.

Como se puede observar en la figura 4.19, de acuerdo al valor del ángulo se obtiene el valor del seno o coseno correspondiente. Es importante hacer notar que solo se muestrea y se implementa la memoria para valores de ángulos positivos y enteros de 0° a 90°.Con esos valores y las coordenadas del centro de la pupila se calculan las localizaciones de la memoria a muestrear mediante el uso en hardware de los multiplicadores y sumadores (figura 4.31).

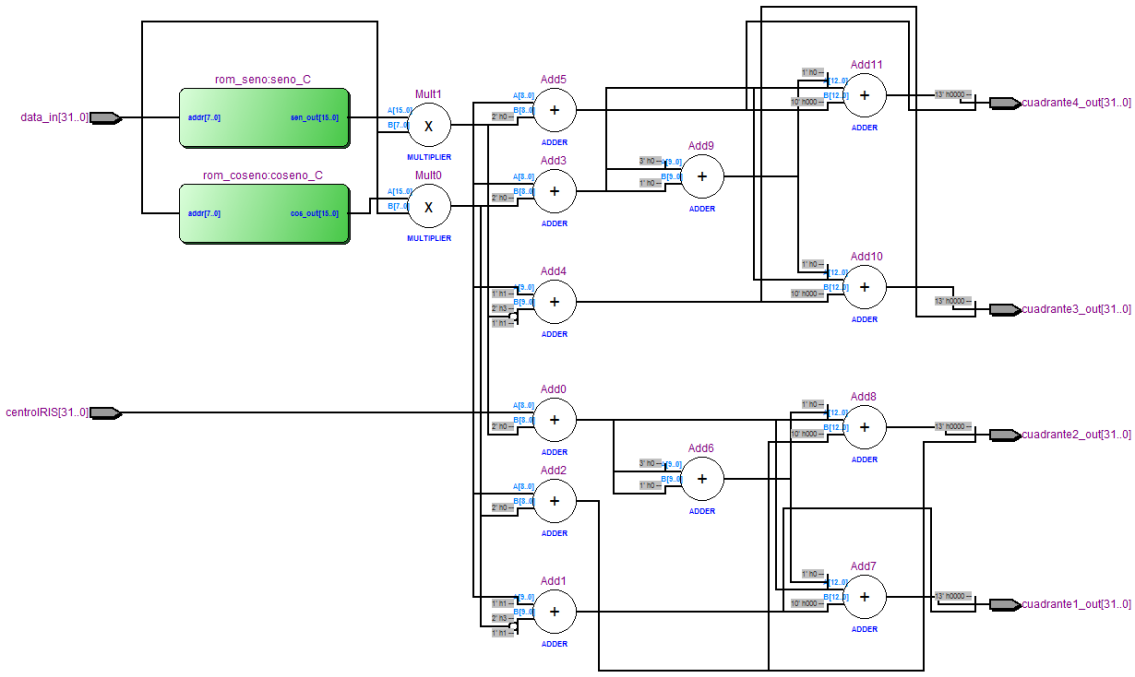


Figura 4.31. Hardware sintetizado para la implementación del módulo auxiliar de normalización en FPGA.

4.3.2.5. Normalización (Hardware)

Debido a que la etapa de normalización consiste en hacer un cambio de coordenadas polares a cartesianas, esto implica que por cada radio muestreado se muestrean 180 pixeles más (cada muestra correspondiente a 1°, de un total de 180°) y considerando que aproximadamente el mínimo número de radios a muestrear en las imágenes es de 70, el valor mínimo de operaciones de muestra y cálculo de coordenadas de las muestras dentro de la imagen es de aproximadamente 12600. Como se muestra en las secciones 5.2.1,

5.2.2 y 5.2.3, la etapa de normalización en cada una de las plataformas en las que fue implementada, es la que requiere de mayor tiempo de procesamiento.

Mediante la implementación mostrada en la sección 4.3.2.4, los resultados que se obtienen en cuanto a tiempo de procesamiento son buenos a comparación de hacer la implementación solamente en software y ejecutada en el procesador suave; la desventaja de esta implementación es que el procesador al llevar el control de los valores del radio y el ángulo a muestrear, hace escritura y lectura de los datos en los puertos paralelos con el que se comunica con el modulo y es en estos pasos donde se consume la mayor cantidad de tiempo de procesamiento; no obstante, es posible mejorar el tiempo de procesamiento si esta etapa es implementada y ejecutada totalmente en hardware.

Para realizar esta etapa del procesamiento totalmente en hardware es necesario primero realizar unos pequeños ajustes a las 280 memorias externas donde se almacena la imagen (sección 4.3.2), el cambio que se hace es cambiar de tamaño cada localidad de memoria, es decir, pasar de 8 bits por cada localidad de memoria a 9 bits por localidad; esto nos permite no modificar el valor original del píxel, y en el bit extra almacenar su valor binario resultado del procesamiento de filtrado y umbralado de la imagen.

El cambio realizado al módulo de filtrado y umbralado (sección 4.3.2.1) es que al leer los valores de los pixeles de las respectivas memorias, el resultado de umbralar la imagen se guarda en el bit más significativo de la localidad de memoria, dejando intacto el valor original de cada píxel de la imagen ya no sustituyendo el valor original del píxel. Por otro lado, el módulo de “Puntos de iris” (Sección 4.3.2.3), la única modificación que se sufre es que en vez de tomar en cuenta el bit menos significativo, ahora sólo toma en cuenta el bit más significativo de la localidad de memoria de cada píxel, mismo valor que corresponde a el valor del píxel en la imagen binaria.

El módulo de normalización, implementado completamente en hardware se muestra en la figura 4.32. Este módulo tiene como entradas, una señal de reloj, de *reset* y una señal de inicio o *go* con la que se le indica al módulo que realice su tarea. Por medio de las entradas *Radio_i* y *Radio_f* que tienen una longitud de 8 bits, se le indica al módulo los valores de los radios de inicio y final respectivamente; mediante las entradas de *Centro_X* y *Centro_Y*, ambos de 9 bits, el modulo recibe los valores que corresponden a las coordenada del píxel central o de referencia (centro del plano polar); la entrada *data_mem_in* es la entrada de los valores de los pixeles provenientes de la memoria donde estos están almacenados de acuerdo a las coordenadas del píxel apuntado por las salidas *Read_X_add_mem* y *Read_Y_add_mem*. Dentro del módulo es implementada también una memoria de 16 Kbytes de 8 bits por localidad en donde es almacenada la imagen resultante, misma que es accedida por el procesador suave a través de la entrada *add_norm_CPU* donde el procesador indica al módulo la localidad de memoria que desea leer y el valor de salida correspondiente a esa localidad es obtenida por el procesador mediante la salida del módulo *datao_norm_CPU*. La saida *working*, que el modulo está realizando su tarea (cuando su valor es un 1 lógico) y además indica a los multiplexores que controlan la lectura de las memorias donde se encuentra la imagen completa que este módulo es quien está trabajando y accediendo a ella. Por último la salida *Norm_Done* toma un valor de 1 lógico cuando el modulo ha terminado de realizar su tarea.

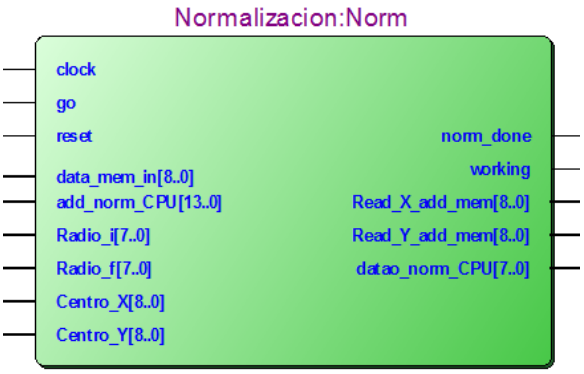


Figura 4.32. Bloque funcional del módulo para la normalización de la imagen en hardware.

Como se observa en la figura 4.33, el módulo de Normalización, cuyo bloque funcional se muestra en la figura 4.32 se compone de 2 módulos principales: 1) el módulo *mod_norm*, el cual contiene una máquina de estados y el hardware necesario para llevar el control de las direcciones de memoria a leer y escribir (figura 4.35) y 2)el módulo *Mem_Norm* que es la memoria donde se almacenan los resultados; esta última es una memoria RAM de 16384 localidades de memoria, donde cada localidad se compone de 8 bits. El tamaño de la memoria RAM es debido a que se están muestreando 180° por cada valor de radio, y limitando la normalización a un valor máximo de 90 radios a muestrear, el tamaño de la memoria necesaria para almacenar el resultado de la normalización es de 16384 localidades (2^{14}).

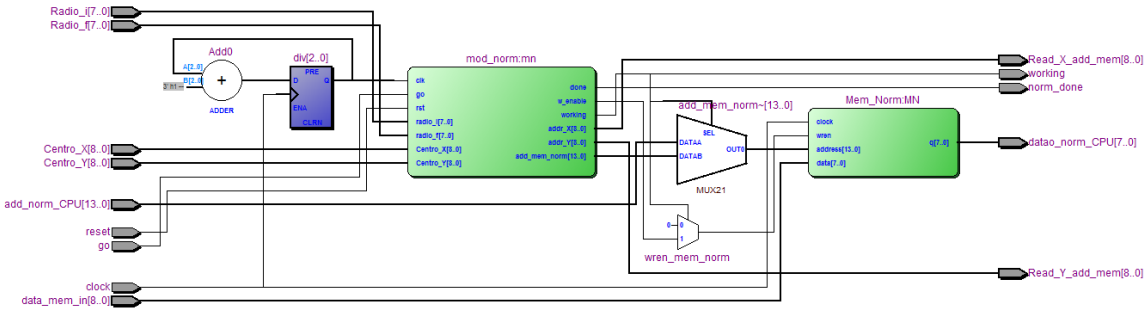


Figura 4.33. Hardware sintetizado para la implementación del módulo de normalización en FPGA.

Para llevar el control de los radios y ángulos a muestrear, el modulo consta de una máquina de estados como se muestra en la figura 4.34; ésta consta de 3 estados, el primero de ellos (ESPERA_ST) es donde la máquina y por lo tanto el modulo esta solo a la espera de la señal de inicio para la realización de la tarea, una vez que se da la señal de inicio, la maquina pasa al segundo estado (ACCION_ST) donde por cada ciclo de reloj de entrada el valor de los ángulos de muestreo y en su momento el valor del radio de muestreo son actualizados partiendo del valor de radio inicial obtenido de la entrada *Radio_i* y culminando con el valor de radio final obtenido de la entrada *Radio_f*; de manera concurrente se obtiene el valor de seno y coseno de cada ángulo (previamente obtenidos y almacenados en una memoria ROM) y los cálculos de la localidad del píxel a muestrear dentro de la imagen. Finalmente una vez que el valor del radio a muestrear es igual al valor del radio final y el ángulo muestrear es 180, la maquina pasa a su ultimo estado (DONE_ST) donde desactiva la salida *working* y habilita la salida de *done* indicando que ha culminado su tarea y quedándose en ese estado hasta recibir una señal de *reset*. El circuito sintetizado donde se

incluye la máquina de estados y los componentes necesarios para la realización de los cálculos se muestra en la figura 4.35.

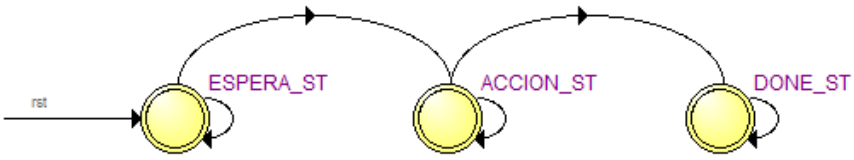


Figura 4.34. Máquina de estados implementada para la normalización del iris

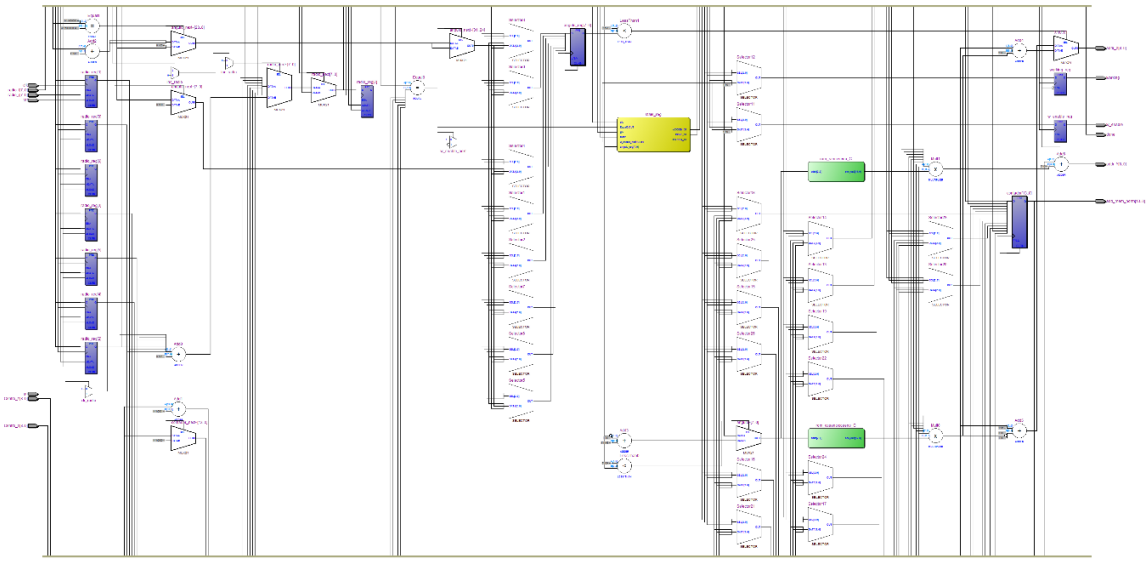


Figura 4.35. Hardware sintetizado para la implementación del módulo *mod_norm* contenido dentro del módulo de normalización.

El circuito sintetizado de la por el software de desarrollo de la interconexión de todos los módulos hasta aquí descritos se muestra en la figura 4.36

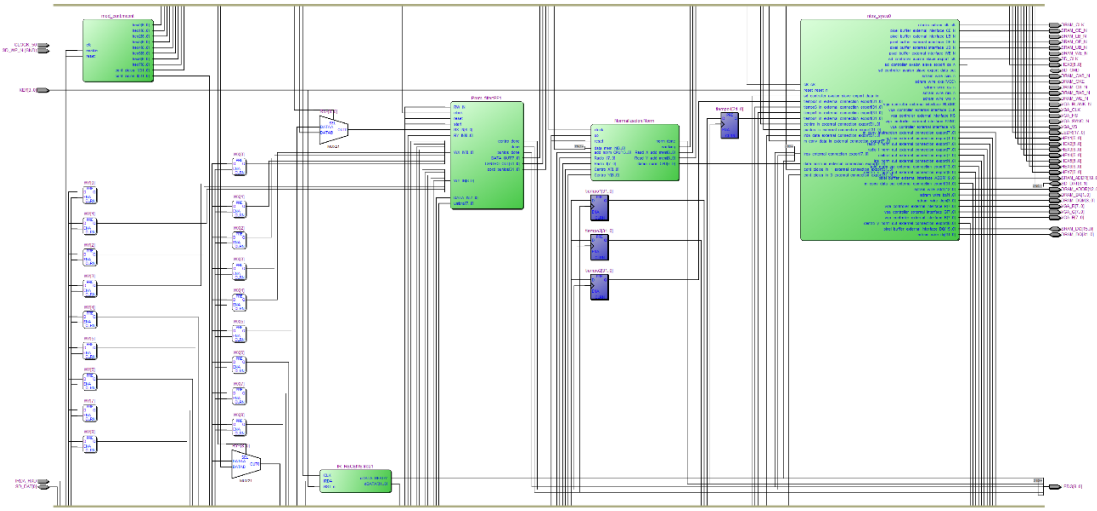


Figura 4.36. Hardware sintetizado para la implementación del módulo *mod_norm* contenido dentro del módulo de normalización.

CAPÍTULO V

Evaluación de los resultados

En el presente capítulo se detallan los resultados de la implementación del algoritmo de segmentación y normalización del iris, desarrollados en Matlab, C++ Builder 6 y en la tarjeta de desarrollo FPGA DE2-115; también se incluye la comparación de los resultados obtenidos en el tiempo de procesamiento para el caso de Matlab y C++ Builder 6, al correrlo en diferentes equipos de cómputo con características distintas con diferentes sistemas operativos.

5.1. Resultados del algoritmo

En esta parte se muestran los resultados de cada una de las etapas del algoritmo descrito en la sección 4.2; se destacan los resultados satisfactorios, así como los de las imágenes conflictivas, donde los resultados obtenidos no siempre fueron los esperados.

5.1.1. Aproximación de la pupila

El algoritmo de ubicación y aproximación de la pupila presentado en la sección 4.2.2 siempre da como resultado un centro y la aproximación adecuada de la pupila mediante una circunferencia (figura 5.1.); no obstante, una característica de la pupila en algunas de las imágenes de la base de datos utilizada es que en algunos de los casos la pupila no es circular, tiene forma elíptica, de manera que al aproximarla mediante una circunferencia hay puntos de la pupila que quedan fuera de la circunferencia de aproximación como se muestra en la figura 5.2. Estos casos han sido reportados en diversos trabajos que aproximan la pupila mediante una circunferencia, y no supone un problema al momento de obtener el código de identificación de la imagen normalizada para su posterior identificación.

En todas las imágenes de la base de datos, los resultados obtenidos son buenos, pues siempre se aproxima la pupila adecuadamente con una circunferencia. De la base de datos, en el 100% de las imágenes se obtuvo el centro de la pupila y se aproximó la circunferencia que la bordea de manera correcta; de acuerdo a las condiciones que se dan el esta parte del algoritmo, no hubo ninguna imagen en la que el centro de la pupila y la circunferencia se hayan ubicado en otra parte que no sea el centro y alrededor de la pupila, respectivamente.

5.1.2. Segmentación de iris

El algoritmo presentado en la sección 4.2, se centra en la segmentación de la imagen para la obtención de la subimagen que esté compuesta únicamente por el iris. Algunos ejemplos de los resultados de la segmentación del iris se muestran en la figura 5.4., donde se obtienen 2 circunferencias con las cuales se limita la información del iris para su posterior normalización.

Como se mencionó en la sección 4.2.1., la base de datos consta de 756 diferentes imágenes, de las cuales se obtiene la aproximación y segmentación del iris de manera correcta en **703** imágenes, que equivale al **93%,** mientras que en las otras **33** imágenes (**4.37%**) la segmentación es considerada incorrecta debido a que el procedimiento aproximó el iris con un círculo de radio mayor o menor (figura 5.5.); sin embargo, al normalizar la zona segmentada se obtiene información del iris, misma que puede considerarse válida y mínima para ser utilizada en etapas posteriores.

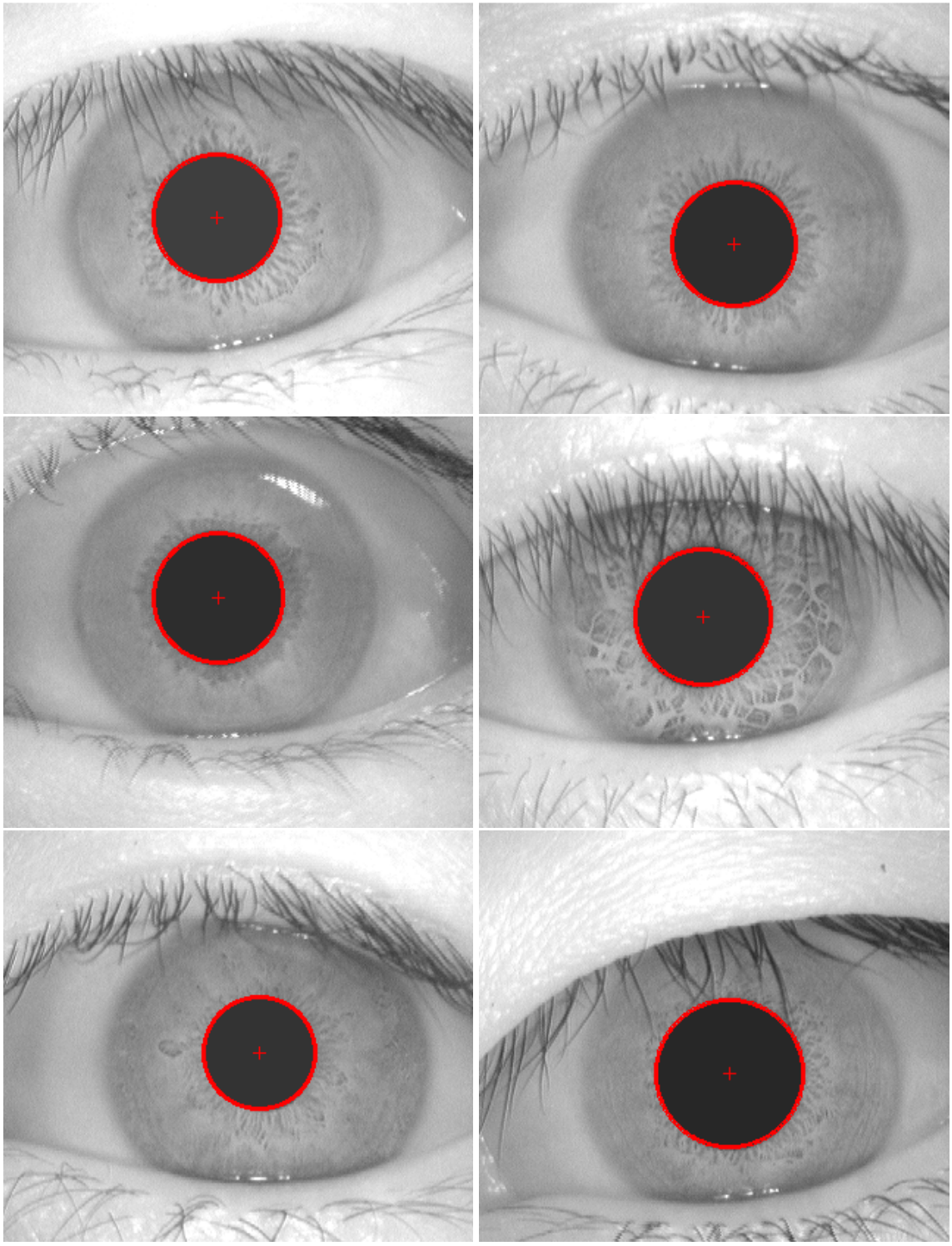


Figura 5.1. Ejemplo de los resultados de la aproximación de la pupila mediante una circunferencia.

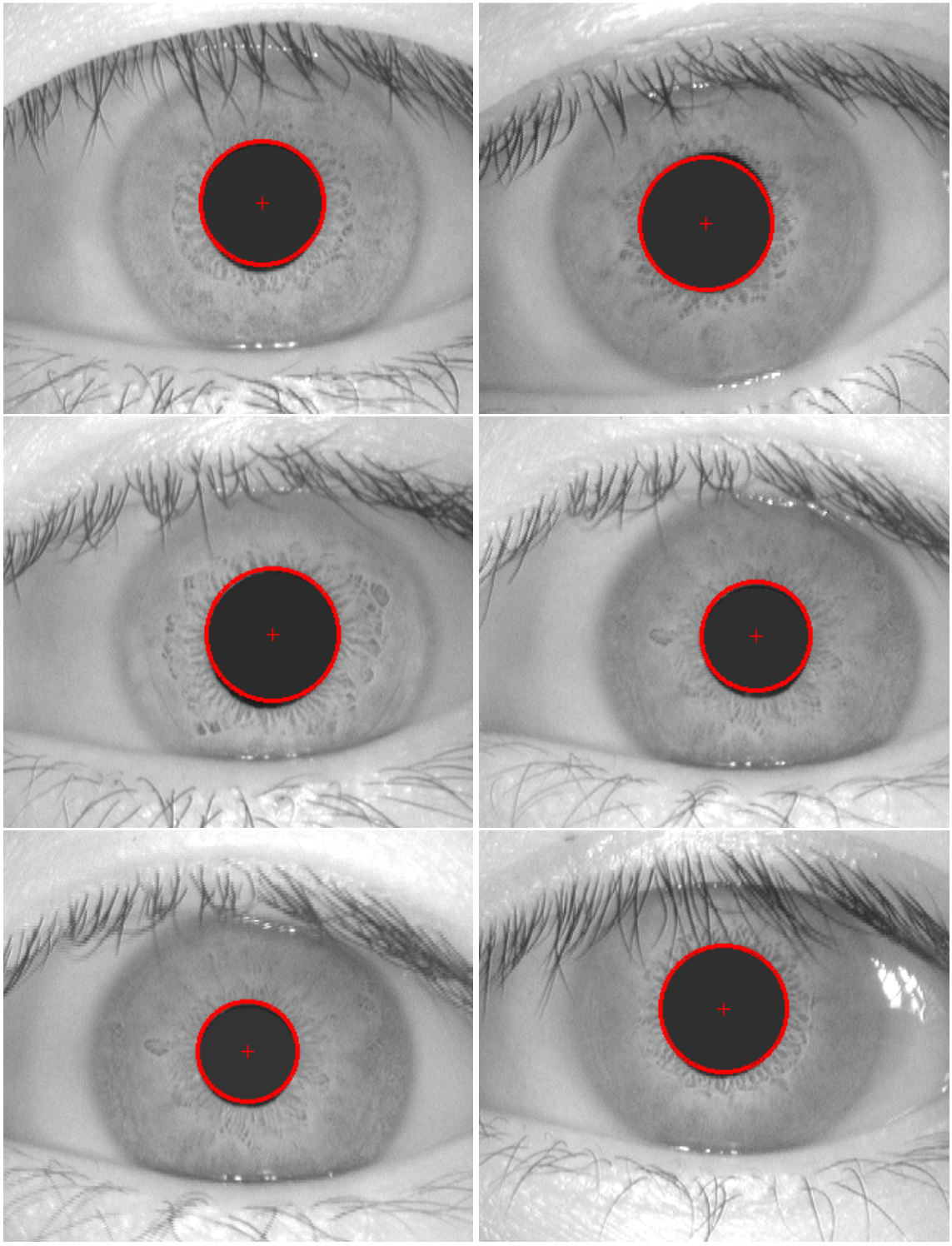
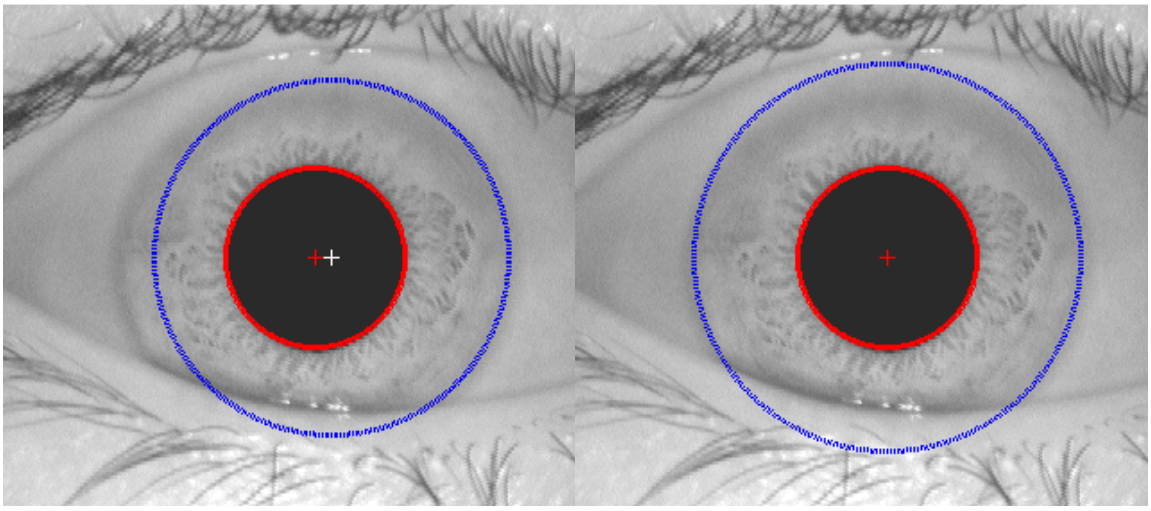


Figura 5.2. Ejemplos de los resultados de la aproximación de la pupila mediante una circunferencia en imágenes donde la pupila no es circular.

En el restante **2.6%** de las imágenes, en primera instancia, la segmentación del iris parece incorrecta debido a que de un costado la aproximación es correcta pero del otro costado es incorrecta (figura 5.3a), debido a que las características de la imagen no permiten una mejor aproximación con el algoritmo propuesto; no obstante y como se mencionó en la sección 4 donde se describe el algoritmo, para la etapa de normalización se toma como referencia un nuevo centro de la pupila y como radio máximo, la máxima distancia entre este centro y la intersección de la circunferencia del iris a 0° y 180° con el eje horizontal que pasa por el nuevo centro; de este modo, como se observa en la figura 5.3b se mejora la aproximación y la segmentación. Por lo tanto, al ser ya correctas este 2.6% pasa a ser parte del porcentaje de las imágenes correctamente segmentadas, lográndose así un porcentaje total de **95.6%**.



(a) (b)

Figura 5.3. Ejemplo de una de las imágenes del 2.6% cuya segmentación requirió corrección antes de la normalización. La figura (a) corresponde a la salida de la aproximación del iris mediante la circunferencia; (b) Resultado final de la segmentación justo antes de la etapa de normalización.

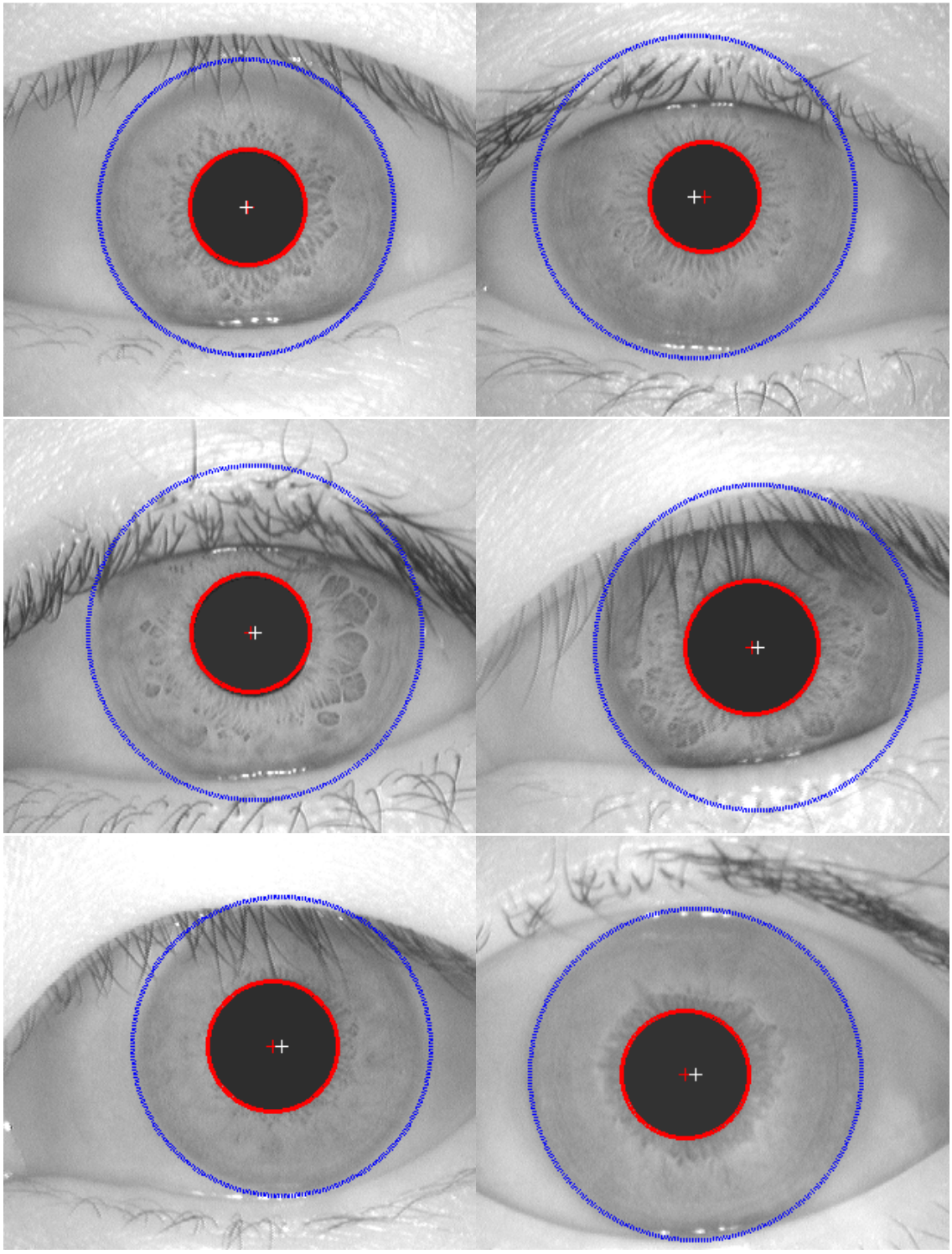


Figura 5.4 Ejemplos de resultados de la segmentación del iris mediante circunferencias dentro del 93% de imágenes correctamente segmentadas.

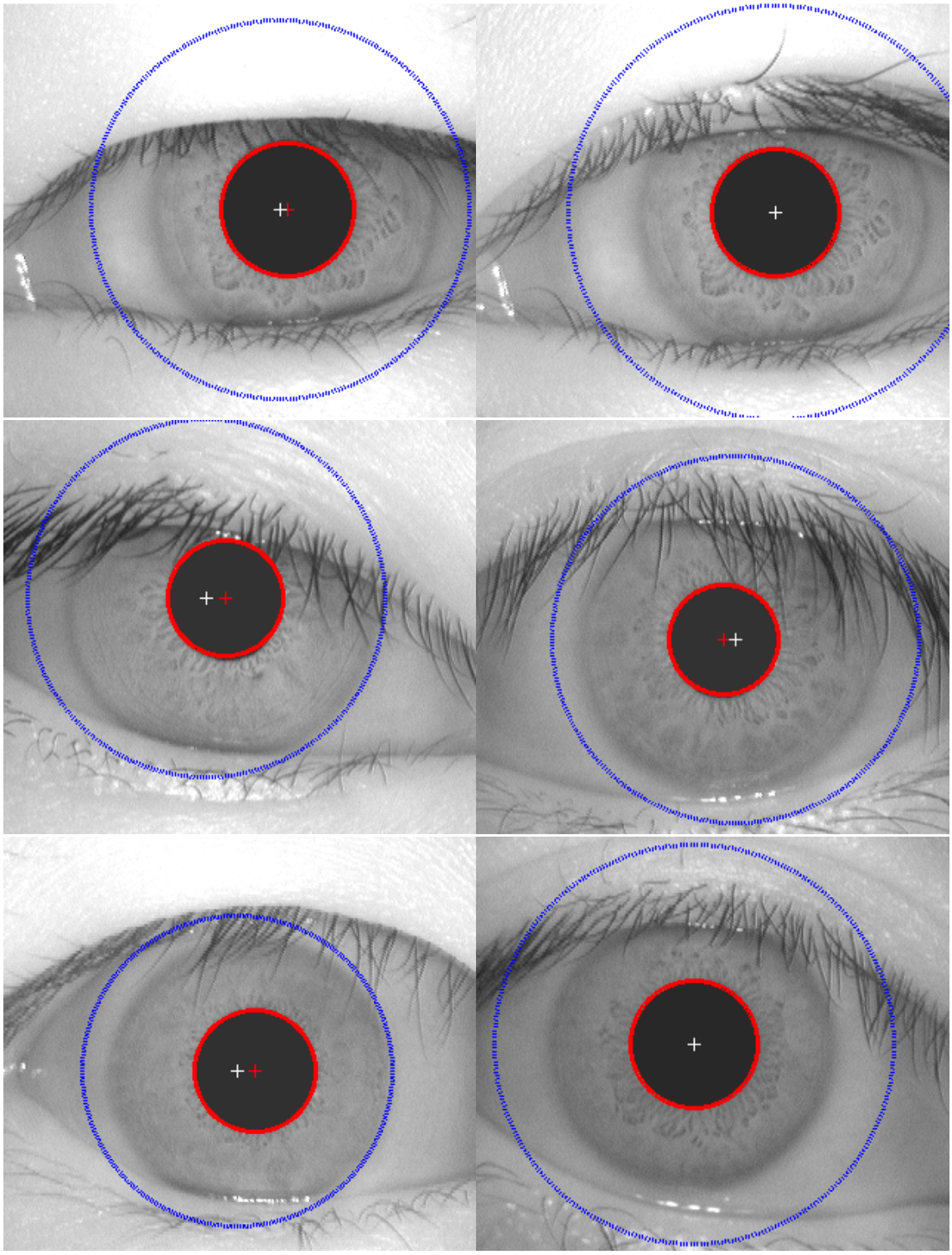
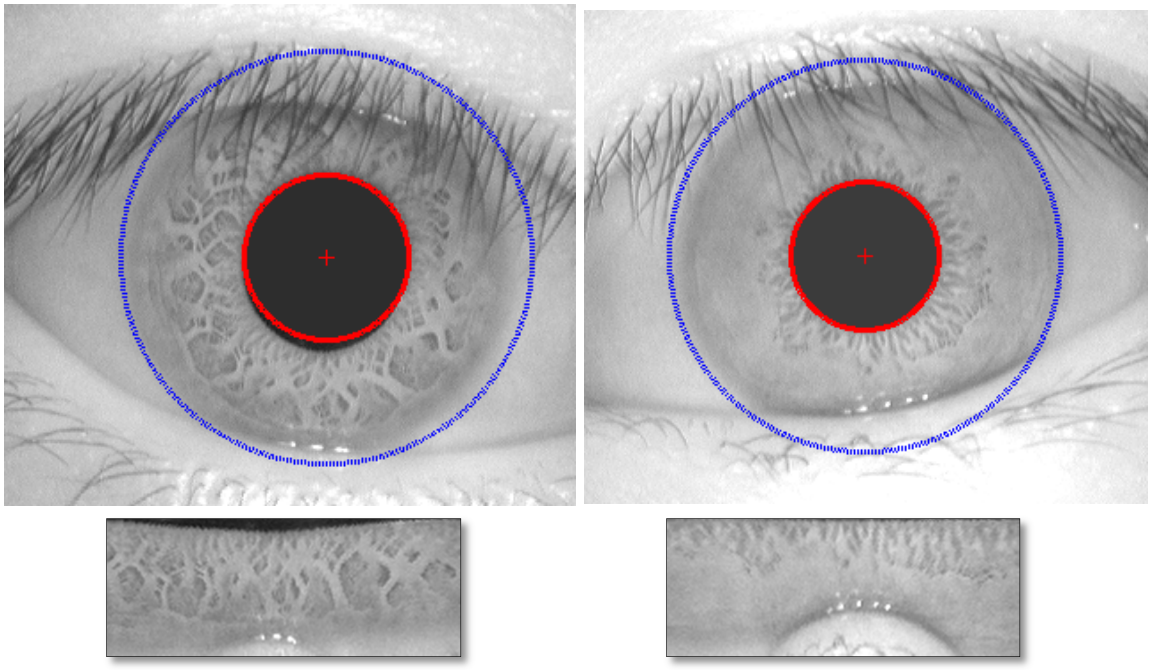


Figura 5.5. Ejemplos de resultados de la segmentación del iris mediante circunferencias dentro del 4.37% de imágenes conflictivas.

5.1.3. Normalización

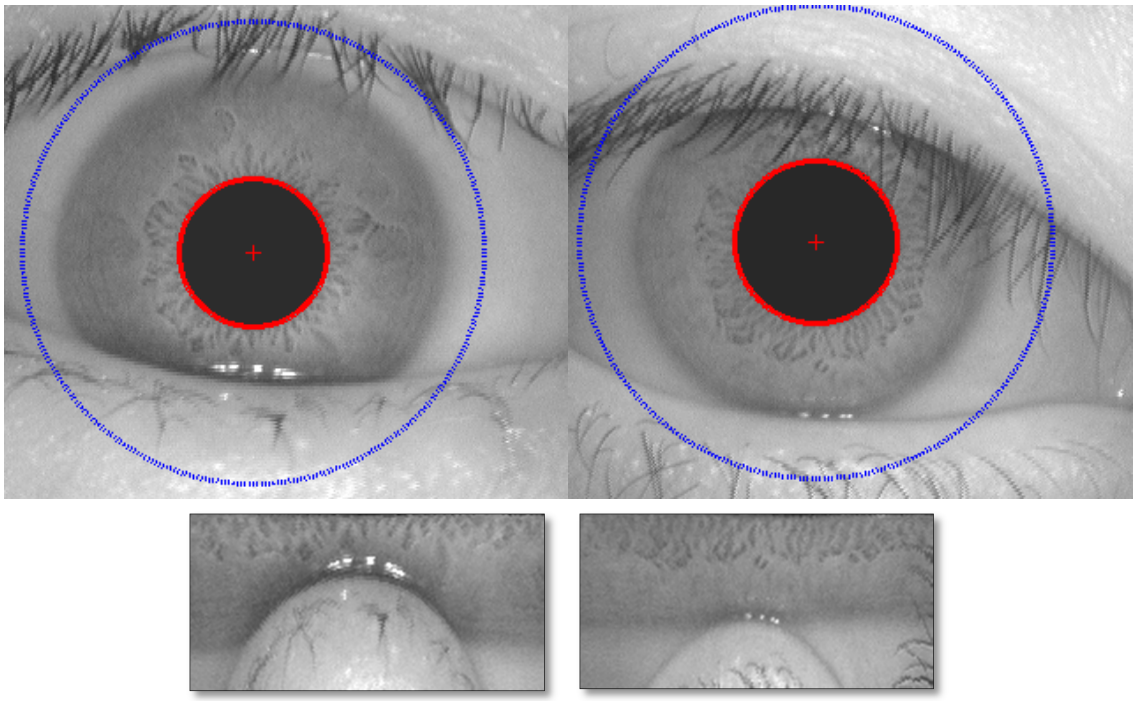
La normalización consiste en un cambio de coordenadas polares a cartesianas, de todos los puntos de la corona que resulta de la segmentación del iris. La zona del iris que se normaliza está ubicada en la parte inferior del iris (tercero y cuarto cuadrantes, debajo de los 180°) y como restricción, para limitar el ancho máximo de las imágenes resultantes, el ancho de la imagen normalizada es limitada a un tamaño máximo de 90 pixeles.

La figura 5.6 muestra ejemplos de imágenes con los iris segmentados y la correspondiente normalización de una parte del mismo. Todas las imágenes resultantes contienen completamente la información del iris, no obstante, en muchos de los casos debido a la naturaleza de las imágenes la parte inferior del iris se ve afectada por el párpado inferior; también hay casos de imágenes conflictivas donde el radio de la circunferencia con la que se segmenta el iris es más grande que la circunferencia del iris (figura 5.4.) y por lo tanto, en la imagen resultante de la normalización además de la información del iris aparece un poco de información de la esclerótica (figura 5.7), misma que en una etapa posterior (que no es parte de este trabajo) en donde se obtiene un código de identificación de cada imagen, es eliminada mediante los procesamientos ahí utilizados principalmente basados en la búsqueda de texturas.



(a) (b)

Figura 5.6. Ejemplos de la normalización del iris segmentado. Las imágenes inferiores (a) y (b) corresponden a la normalización de los iris segmentados de las imágenes de la base de datos 78_2_3 en el caso de (a) y 14_2_2 en el caso de (b).



(a) (b)

Figura 5.7. Ejemplos de la normalización del iris de dos imágenes conflictivas. Las imágenes (a) y (b) muestran la normalización de los iris segmentados de las imágenes de la base de datos 30_1_2 en el caso de (a) y 71_1_2 en el caso de (b).

5.2. Resultados de la Implementación

El algoritmo descrito en la sección 4.2 se implementó en Matlab 2012, en lenguaje C/C++ dentro del ambiente del C++ Builder 6 y en FPGA mediante la implementación de los módulos descritos en la sección 4.3.

Lo que se evaluó en cada una de las implementaciones es el tiempo de ejecución del algoritmo, así como la cantidad de ciclos de reloj requeridos. El algoritmo completo consta de 3 etapas principales:

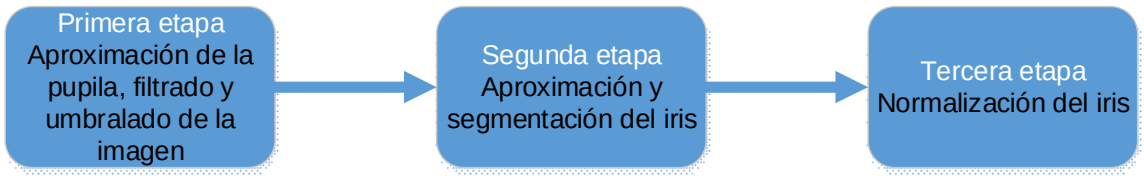


Figura 5.8. Etapas del algoritmo.

En la primera etapa se detecta y se contornea (aproxima) la pupila, se filtra y finalmente se umbrala la imagen; esta etapa se realizó totalmente mediante módulos de hardware, es decir, cuando el procesador da la señal de inicio, estos módulos empiezan y terminan dichas tareas. En la segunda etapa se lleva a cabo la obtención de los puntos frontera de la parte exterior del iris a partir de la imagen binaria obtenida de la primera etapa, la obtención del centro y del radio de la circunferencia exterior que aproxima al iris; los puntos frontera se hallan mediante hardware, mientras que la tarea de analizar esos puntos y determinar el

centro y el radio exterior la realiza el procesador suave. En la última etapa, se lleva a cabo la normalización, la cual se hace con el trabajo conjunto del software y el hardware, es decir, el software para llevar el control de los ciclos y de la memoria y un módulo de hardware que realiza las operaciones matemáticas que regresa las localizaciones de memoria a ser leídas en la imagen original con el fin de normalizar el iris, y mediante un módulo de hardware que realiza todo el trabajo de dicha tarea.

Para obtener los resultados que se muestran en las siguientes secciones, se ejecutó el algoritmo sobre todas las imágenes de la base de datos, es decir, 756 veces, de las cuales se guardaron los tiempos de procesamiento para posteriormente obtener los valores mínimos, máximos y promedio de cada una de las etapas.

5.2.1. Resultados de la implementación del Algoritmo en Matlab

El algoritmo implementado en Matlab fue ejecutado en dos equipos de diferentes capacidades, características y sistemas operativos. Los resultados de los tiempos y el número de ciclos de reloj utilizados por cada etapa se muestran en las tablas 5.1., 5.2. y 5.3.

Tabla 5.1. Tiempos y ciclos de reloj de la ejecución de la primera etapa del algoritmo ejecutada en matlab bajo el sistema operativo Windows.

1ra. Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3 Matlab 2011a	32.0	32.6	57.7	80,985,000	82,495,000	93,458,200
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Microsoft Windows 7 Home Premium Matlab 2012a	240.9	252.1	890	602,132,500	630,130,000	2,226,600,000

Tabla 5.2. Tiempos y ciclos de reloj de la ejecución de la segunda etapa del algoritmo ejecutada en Matlab bajo el sistema operativo Windows.

2da. Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3 Matlab 2011a	0.3	3.6	34	738,760	8,999,700	86,047,830
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Microsoft Windows 7 Home Premium Matlab 2012a	0.307	3.8	53.9	767,500	9,482,100	134,650,000

Tabla 5.3. Tiempos y ciclos de reloj de la ejecución de la tercera etapa del algoritmo ejecutada en Matlab bajo el sistema operativo Windows.

3ra. Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3 Matlab 2011a	86.1	135.5	203	217,750,000	342,770,000	513,582,410
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Microsoft Windows 7 Home Premium Matlab 2012a	83.9	134.2	208.7	209,740,000	335,380,000	521,700,000

Los resultados de los tiempos y el número de ciclos de reloj utilizados por la ejecución total del algoritmo ejecutado bajo el sistema operativo Windows se muestran en la tabla 5.4.

Tabla 5.4. Tiempos y ciclos de reloj de la ejecución de la ejecución completa del algoritmo en Matlab bajo el sistema operativo Windows.

Total						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3 Matlab 2011a	127.1	171.7	244.2	321,480,000	434,260,000	617,729,860
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Microsoft Windows 7 Home Premium Matlab 2012a	341.7	390	1138	854,302,500	975,000,000	2,846,700,000

Los resultados obtenidos en cuanto a tiempo y ciclos de reloj de la misma implementación en Matlab, pero ejecutados en 2 versiones diferentes de sistema operativo Ubuntu son los siguientes:

Tabla 5.5. Tiempos y ciclos de reloj de la ejecución de la primera etapa del algoritmo ejecutada en Matlab bajo el sistema operativo Ubuntu.

1ra Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Ubuntu 11.10 Matlab 2011a	31.9	32.7	57.6	80,634,000	82,683,000	145,692,580
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Ubuntu 12.04 Matlab 2011a	40.7	45.9	60.4	101,715,000	114,680,000	150,910,000

Tabla 5.6. Tiempos y ciclos de reloj de la ejecución de la segunda etapa del algoritmo ejecutada en Matlab bajo el sistema operativo Ubuntu.

2da. Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Ubuntu 11.10 Matlab 2011a	0.5	6.5	73.3	1,282,700	16,413,000	185,545,140
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Ubuntu 12.04 Matlab 2011a	0.345	4.7	48	862,500	11,641,000	119,785,000

Tabla 5.7. Tiempos y ciclos de reloj de la ejecución de la tercera etapa del algoritmo ejecutada en Matlab bajo el sistema operativo Ubuntu.

3ra. Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Ubuntu 11.10 Matlab 2011a	178.6	281.5	400	451,800,000	712,190,000	1,013,500,000
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Ubuntu 12.04 Matlab 2011a	102.1	171	308	255,300,000	427,750,000	770,612,500

Tabla 5.8. Tiempos y ciclos de reloj de la ejecución total del algoritmo ejecutada en Matlab bajo el sistema operativo Ubuntu.

Total						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Ubuntu 11.10 Matlab 2011a	230	320.7	467.4	581,105,580	811,280,000	1,182,500,000
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Ubuntu 12.04 Matlab 2011a	170	222	368	424,252,500	554,070,000	919,845,000

5.2.2. Resultados de la implementación del algoritmo en C++ Builder 6

El algoritmo implementado en lenguaje C/C++ fue desarrollado utilizando el entorno de desarrollo C++ Builder 6. Se ejecutó en cuatro equipos con diferentes capacidades y características, así como con diferentes versiones del sistema operativo Windows. Los resultados de los tiempos utilizados se muestran en la tabla 5.3.; así mismo los resultados en cuanto a ciclos de reloj por cada etapa y en total se muestran en la tabla 5.4.

Tabla 5.9. Tiempos y ciclos de reloj de la ejecución de la primera etapa del algoritmo ejecutada en C++ Builder 6.

1ra. Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3	2.0	2.1	2.5	5,108,200	5,386,900	6,334,900
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Microsoft Windows 7 Home Premium	1.6	2.7	44.2	3,888,675	6,819,000	110,509,750

Intel Core i7 CPU **2.10 GHZ** 6 GB de RAM Microsoft Windows 7 Home Premium	1.8	2.3	7.1	3,873219	4,828,100	14,969,430
Mobile AMD Sempron Processor 3500+ **1.8GHz** 3 GB RAM Windows 7 Ultimate	5.2	6.1	37.3	9,285,768	10,907,000	67,066,000

Tabla 5.10. Tiempos y ciclos de reloj de la ejecución de la segunda etapa del algoritmo ejecutada en C++ Builder 6.

2da. Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de Reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3	0.001	0.010	0.062	29,348	26,047	157,140
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Microsoft Windows 7 Home Premium	0.002	0.013	0.145	30,750	33,779	362,300
Intel Core i7 CPU **2.10 GHZ** 6 GB de RAM Microsoft Windows 7 Home Premium	0.001	0.016	0.185	2,058	34,255	389,990
Mobile AMD Sempron(tm) Processor 3500+ **1.8GHz** 3.00 GB RAM Windows 7 Ultimate	0.006	0.030	0.586	10,548	54,017	1,056,000

Tabla 5.11. Tiempos y ciclos de reloj de la ejecución de la tercera etapa del algoritmo ejecutada en C++ Builder 6.

3ra. Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de Reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3	3	4.2	6	7,581,600	10,744,000	15,303,000
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Microsoft Windows 7 Home Premium	5.4	9.6	12.3	13,578,000	23,951,000	30,750,000
Intel Core i7 CPU **2.10 GHZ** 6 GB de RAM Microsoft Windows 7 Home Premium	6.8	11.3	28.1	14,213,000	23,676,000	59,108,000
Mobile AMD Sempron(tm) Processor 3500+ **1.8GHz** 3.00 GB RAM Windows 7 Ultimate	12.0	19.1	109.9	21,605,238	34,390,000	197,760,000

Tabla 5.12. Tiempos y ciclos de reloj de la ejecución completa del algoritmo ejecutada en C++ Builder 6.

Total						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft	5.1	6.4	8.1	13,025,705	16,157,000	20,616,000

Windows XP 2002 SP3						
Intel Core i5 CPU **2.50 GHZ** 12 GB de RAM Microsoft Windows 7 Home Premium	7.1	12.3	93.9	17,796,175	30,804,000	234,626,250
Intel Core i7 CPU **2.10 GHZ** 6 GB de RAM Microsoft Windows 7 Home Premium	8.7	13.6	35.3	18,243,288	28,538,000	74,146,170
Mobile AMD Sempron(tm) Processor 3500+ **1.8GHz** 3.00 GB RAM Windows 7 Ultimate	17.3	25.2	115.6	31,124,340	45,351,000	208,150,200

5.2.3. Resultados de la implementación del algoritmo en FPGA

El algoritmo implementado en FPGA utilizando la tarjeta de desarrollo DE2-115, se llevó a cabo implementando los módulos descritos en la sección 4.3.

La primera etapa es ejecutada completamente en hardware en los módulos de filtrado y umbralado (sección 4.3.2.1.) y centro de pupila (sección 4.3.2.2.).

La segunda etapa se ejecuta primero en el módulo de Puntos del iris (sección 4.3.2.3.) que obtiene todos los puntos frontera del iris y posteriormente el procesador suave ejecuta la tarea de leer esos puntos y obtener el centro y el radio de la circunferencia que aproxima al iris por su parte interior.

Por último, la tercera etapa la llevan a cabo el procesador FPGA y un módulo de hardware externo al procesador; es decir, el procesador lleva el control del radio y los ángulos a muestrear, mientras que el módulo de hardware a través de memorias ROM obtiene los valores senos y cosenos de dichos ángulos; por medio de estos valores y el valor del centro de referencia realiza los cálculos para regresarle al FPGA los valores de los índices del arreglo de la imagen a muestrear; el muestreo de esos índices en la memoria de la imagen y la asignación a la memoria de la imagen resultante la realiza el procesador suave.

Los resultados de la implementación del algoritmo de estas 3 etapas se muestran en las tablas 5.13., 5.14. y 5.15.

Es de hacer notar que debido a que el procesamiento de esta primera etapa se realiza completamente en los módulos de hardware, el tiempo y la cantidad de ciclos utilizados es constante en todas sus ejecuciones. El módulo de filtrado y umbralado realiza su tarea en **327 ciclos de reloj,** tiempo en el que lee el valor de los 320 pixeles de cada fila (memoria) y realiza las 280 convoluciones de manera recurrente por cada columna. Por otra parte, al

mismo tiempo que se leen los pixeles, se calcula cuantos pixeles continuos con intensidad menor que el umbral de la pupila hay por cada fila (obtenido al mismo tiempo que la lectura del archivo de la memoria SD); posteriormente estos 280 valores (uno por cada fila de la imagen) son analizados por el módulo de centro de pupila, de lo que se obtiene la o las filas con mayor concentración de pixeles continuos que cumplen dicha condición; esto permite la obtención del centro de la pupila y el radio para la aproximación de la pupila en **282 ciclos de reloj** para todas las ejecuciones.

Como se observa en la tabla 5.13., la implementación en hardware de un algoritmo o de una parte del mismo tiene como una característica importante el que siempre la tarea se realiza en el mismo tiempo, o durante la misma cantidad de ciclos de reloj, independientemente de la frecuencia de reloj.

En la tabla 5.14. se muestran los resultados en cuanto a tiempo de ejecución y los ciclos de reloj utilizados por la segunda etapa del algoritmo.

Tabla 5.13. Tiempos y ciclos de reloj de la ejecución de la primera etapa del algoritmo ejecutada en la tarjeta de desarrollo FPGA DE2-115.

1ra. etapa						
Características del equipo	Tiempo (ms)			Ciclos de reloj		
	Min.	Prom.	Max.	Mínimo	Promedio	Máximo
DE2-115 50 MHz.	0.012	0.012	0.012	609	609	609

Tabla 5.14. Tiempos y ciclos de reloj de la ejecución de la segunda etapa del algoritmo ejecutada en la tarjeta de desarrollo FPGA DE2-115.

2da etapa						
Características del equipo	Tiempo (ms)			Ciclos de reloj		
	Min.	Prom.	Max.	Mínimo	Promedio	Máximo
DE2-115 50 MHz.	0.057	0.087	0.344	2,877	4,372	17,200

En esta segunda etapa del algoritmo se obtienen los puntos frontera del iris en la imagen binaria mediante el módulo de hardware Puntos del iris (sección 4.3.2.3.), que accede a las 280 filas (memorias) al mismo tiempo para encontrar dichos puntos; esto permite obtener todos los puntos en un promedio de 75 ciclos de reloj; en comparación con el módulo de la etapa anterior, en esta etapa este módulo no ejecuta su tarea en la misma cantidad de ciclos de reloj en todas sus ejecuciones, debido a que depende del lugar donde se

encuentra el centro de la pupila en la imagen; de acuerdo a los limites propuestos en la sección 4.2.3.2 para la obtención de dichos puntos, en algunas ocasiones la zona de búsqueda queda fuera de los límites de la imagen y por lo tanto la misma es de menor dimensión reduciéndose en estos casos la cantidad de ciclos de reloj requeridos.

La tarea mayor de esta etapa la realiza el procesador suave implementado en el FPGA (NIOS II), ya que una vez que se obtienen los puntos frontera mediante el análisis de la imagen binaria, la parte del análisis de dichos puntos y la aproximación del iris con una circunferencia es llevada a cabo en lenguaje C en el mismo, mediante operaciones matemáticas de sumas, multiplicaciones y raíces cuadradas.

Tabla 5.15. Tiempos y ciclos de reloj de la ejecución de la tercera etapa del algoritmo ejecutada en la tarjeta de desarrollo FPGA DE2-115.

3ra. Etapa						
Características del Equipo	Tiempo (ms)			Ciclos de reloj		
	Min.	Prom.	Max.	Mínimo	Promedio	Máximo
DE2-115 50 MHz. (Hardware)	0.336	0.5272	0.800	16833	26359	40006
DE2-115 50 MHz. (Software-Hardware)	11.9	16.7	27.3	595,812	833,580	1,364,520

Los resultados de la última etapa del algoritmo, la etapa de normalización se muestran en la tabla 5.15.En esta etapa, la tarea de normalización es realizada de 2 formas diferentes, 1) de manera conjunta entre el procesador y un módulo externo que realiza las operaciones matemáticas, mientras que el procesador lleva el control, mediante 2 ciclos implementados en lenguaje C, de los radio y ángulos a muestrear; 2) mediante un módulo en hardware.

En comparación, cuando la normalización se realiza sólo en el procesador NIOS II (software), en promedio la cantidad de ciclos de reloj que requiere el procesador para realizar dicha tarea es de 8,809,500, mientras que con el apoyo del módulo de hardware externo que realiza las operaciones, y el procesador NIOS II (Software-Hardware) solo se limita a leer y escribir datos en el módulo, en promedio es de 838,570 ciclos, es decir, el uso de un módulo externo de apoyo, que realice las operaciones matemáticas reduce en 10.5 veces la cantidad de ciclos de reloj del procesamiento y por lo tanto también disminuye el tiempo. No obstante, con la implementación en hardware de la misma tarea se reduce el tiempo a ser 334.2 veces más rápido que la implementación completamente en Software y ejecutada por el procesador NIOS II y es también 31.6 veces más rápido que la implementación conjunta entre Software y Hardware.

El tiempo total de la implementación total del algoritmo se muestran en la tabla 5.16.

Tabla 5.16. Tiempos y ciclos de reloj de la ejecución total etapa del algoritmo ejecutada en la tarjeta de desarrollo FPGA DE2-115.

Tiempo total						
Características del Equipo	Tiempo (ms)			Ciclos de reloj		
	Min.	Min.	Min.	Min.	Prom.	Max.
DE2-115 50 MHz. (Hardware)	0.411	0.623	1.2	20572	31144	57758
DE2-115 50 MHz.	12	16.8	27.5	599,685	838,570	1,373,024

5.3. Comparación de los resultados

Como lo muestran las tablas de los resultados de las implementaciones en software, independientemente de la etapas y de los sistemas operativos, los mejores resultados obtenidos corresponden al equipo que posee una mayor frecuencia de reloj en su procesador. Pasemos a comparar los resultados obtenidos en las implementaciones desarrolladas.

En la primera etapa y como se muestra en la tabla 5.17, la tarea de esta etapa es realizada en promedio **175 veces** más rápido (tiempo de ejecución en la PC en lenguaje C/C++ entre el tiempo de ejecución en FPGA) en la implementación de hardware en el FPGA con respecto a la implementación en una PC en lenguaje C/C++, y **2716.6 veces** más rápido (tiempo de ejecución en la PC en Matlab entre el tiempo de ejecución en FPGA) con respecto a la implementación en Matlab.

Al comparar el número de ciclos de reloj utilizados por cada implementación, al igual que en la comparación de tiempo, vemos que la implementación de hardware en el FPGA requiere de menos ciclos de reloj para la realización de la tarea. La implementación en FPGA en promedio requiere de **8,845.5 veces** menos ciclos de reloj (en la ejecución en la PC en lenguaje C/C++ entre la cantidad de ciclos de reloj que en la ejecución en FPGA) que la implementación en C/C++, y en promedio **135,459.7 veces** menos ciclos de reloj (requeridos en la ejecución en la PC en Matlab entre la cantidad de ciclos de reloj requeridos en la ejecución en FPGA) que la implementación en Matlab.

Tomando como referencia la cantidad de ciclos de reloj promedio que requiere la implementación en C/C++, para que esta implementación realice la tarea en el mismo tiempo en que la realiza la implementación en FPGA, la frecuencia de reloj del procesador necesitaría ser de aproximadamente 450 GHz.

Tabla 5.17. Comparación de tiempos y ciclos de reloj de la ejecución de la primera etapa del algoritmo.

1ra. Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
DE2-115 50 MHz.	0.012	0.012	0.012	609	609	609
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3	2.0	2.1	2.5	5,108,200	5,386,900	6,334,900
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3 Matlab 2011a	32.0	32.6	57.7	80,985,000	82,495,000	93,458,200

En el caso de la segunda etapa del algoritmo las condiciones cambian con respecto a la primera etapa, ya que la implementación en el FPGA de esta etapa difiere con respecto a la primera; en esta etapa quien realiza más del 98% de la tarea es el procesador suave implementado en el FPGA desde la arquitectura base. Esta es una de las formas más sencillas y más utilizadas al implementar algoritmos en un FPGA, ya que solo es necesario crear mediante el software de desarrollo, una arquitectura base en un procesador suave, y posteriormente solo implementar el algoritmo en lenguaje C, mismo que ejecuta el procesador. La tabla comparativa de los tiempos y ciclos de reloj de cada implementación en la segunda etapa se muestra en la tabla 5.18.

Como se puede observar en la tabla 5.18, el mejor tiempo se obtiene con la implementación en una PC en C/C++, pero en comparación con la primera etapa donde la implementación en FPGA es de 175 veces más rápido, en esta segunda etapa la implementación en C/C++ es sólo en promedio **8.7 veces** más rápida. A pesar de estos resultados en tiempo, en cuento a ciclos de reloj requeridos para la ejecución de esta etapa del algoritmo, la implementación que tiene los mejores resultados, es decir, que requiere menos ciclos de reloj, sigue siendo la implementación en FPGA con una relación promedio de **5.95 veces** con respecto a la implementación en C/C++ en la PC.

Tomando en cuenta la cantidad promedio de ciclos de reloj que conlleva la ejecución de la segunda etapa en el FPGA, con la misma implementación se requeriría aumentar la frecuencia de reloj del FPGA a 437 MHz; sin embargo, esta frecuencia sigue siendo menor que la frecuencia a la que trabaja la PC donde se obtuvo el mejor resultado.

Tabla 5.18. Comparación de tiempos y ciclos de reloj de la ejecución de la segunda etapa del algoritmo.

2da. Etapa						
Características del equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3	0.001	0.010	0.062	29,348	26,047	157,140
DE2-115 50 MHz.	0.057	0.087	0.344	2,877	4,372	17,200
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3 Matlab 2011a	0.3	3.6	34	738,760	8,999,700	86,047,830

La comparación de los resultados en cuanto a tiempo y ciclos de reloj de la tercera y última etapa del algoritmo se muestran en la tabla 5.19.

Dadas las características de las implementaciones de esta etapa en el FPGA, los mejores resultados en cuanto a tiempo y ciclos de reloj los tiene la implementación en FPGA con la realización de la etapa de normalización en hardware en una razón promedio de **7.97 veces** más rápido con respecto a la mejor implementación en software (C/C++) implementada en la PC.

Considerando la realización de esta etapa mediante software-hardware en el FPGA, a comparación de la etapa anterior donde la implementación en C/C++ en la PC es 5.95 veces más rápido, aquí ese cociente es mucho menor debido a que la tarea es realizada de manera conjunta entre el procesador y el módulo de hardware y no solo hecha por el procesador.

Tabla 5.19. Comparación de tiempos y ciclos de reloj de la ejecución de la tercera etapa del algoritmo.

3ra. Etapa						
Características del Equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
DE2-115 50 MHz. (Normalización mediante hardware)	0.336	0.5272	0.800	16833	26359	40006
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3	3	4.2	6	7,581,600	10,744,000	15,303,000
DE2-115 50 MHz. (Normalización mediante software-hardware)	11.9	16.7	27.3	595,812	833,580	1,364,520
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3 Matlab 2011a	86.1	135.5	203	217,750,000	342,770,000	513,582,410

La tabla comparativa de los resultados de la ejecución total del algoritmo se muestra en la tabla 5.20.

En la ejecución total del algoritmo que consiste en la unión de los resultados de las 3 etapas en las que se dividió el algoritmo, la implementación donde se obtiene el mejor tiempo de ejecución es en la implementación en FPGA con la realización de la etapa de normalización en hardware siendo en promedio **10.27 veces** más rápido en cuanto al tiempo de ejecución que el mejor tiempo obtenido en lenguaje C/C++ implementado en la PC.

En cuanto a la cantidad de ciclos de reloj requeridos para la ejecución total del algoritmo, en promedio hay una diferencia de $\mathbf{15 \times 10^6}$ **ciclos de reloj,** es decir, la implementación en FPGA (con la etapa de normalización en hardware) en total requiere de **19.26 veces** menos ciclos de reloj para la ejecución total del algoritmo.

Tabla 5.20. Comparación de tiempos y ciclos de reloj de la ejecución total del algoritmo.

Ejecución Total						
Características del Equipo	**Tiempo (ms)**			**Ciclos de reloj**		
	Min.	**Prom.**	**Max.**	**Mínimo**	**Promedio**	**Máximo**
DE2-115 50 MHz. (Normalización mediante hardware)	0.411	0.623	1.2	20572	31144	57758
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3	5.1	6.4	8.1	13,025,705	16,157,000	20,616,000
DE2-115 50 MHz. (Normalización mediante software-hardware)	12	16.8	27.5	599,685	838,570	1,373,024
Intel Core Duo CPU **2.53 GHz** 1 GB de RAM Microsoft Windows XP 2002 SP3 Matlab 2011a	127.1	171.7	244.2	321,480,000	434,260,000	617,729,860

Es importante hacer notar que la tarjeta de desarrollo FPGA es alimentada y trabaja con una frecuencia de reloj de 50 MHz, mientras que la PC donde se obtuvieron los mejores resultados trabaja a 2.53 GHz, es decir 50.6 veces más; y aún así lo resultados obtenidos en ambas plataformas muestran que llega a haber una diferencia promedio de 5.95 veces en cuanto al tiempo de ejecución cuando el tiempo de la implementación en PC es menor (segunda etapa), pero el tiempo de ejecución en FPGA llega a ser 175 veces menor que la implementación en la PC cuando se implementa el algoritmo puramente en hardware (primera etapa).

5.3.1. Cálculo de Errores

Dadas las características de la ejecución del algoritmo en las dos plataformas diferentes (PC y FPGA), es probable que las imágenes resultantes del algoritmo en ambas plataformas no sean iguales, debido principalmente a la precisión de los cálculos. En el FPGA los cálculos se realizan en gran parte con números enteros, donde se utilizó para los cálculos

de la etapa de normalización (para obtener los valores seno y coseno del ángulo a muestrear) una notación de punto fijo con 14 bits para la parte fraccionaria. Por otro lado en la PC, para esto mismo se realizó con variables de tipo flotante. Esta diferencia hace que las imágenes resultantes no sean exactamente iguales.

El histograma de una imagen proporciona una descripción de la apariencia global de misma, por lo que una forma sencilla de determinan cuan parecidas son las imágenes resultantes del procesamiento en el FPGA y la PC es mediante el error cuadrático medio (Ec. 5.1) tomando como valor esperado el histograma de la imagen resultante de la PC y como muestra el histograma de la imagen resultante en el FPGA. Otra forma de determinar la semejanza o diferencia entre los resultados obtenidos en ambas plataformas es mediante la distancia que existe entre las plantillas de iris o matrices binarias de identificación mismas que se obtienen en una etapa posterior a la normalización haciendo uso de un filtro de Gabor.

$$Error = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - x'_i)^2} \qquad \text{(Ec. 5.1)}$$

En promedio, el error cuadrático medio que se obtiene al comparar los histogramas de las imágenes resultantes del procesamiento en el FPGA y las imágenes resultantes de los procesamientos en la PC, con la misma imagen de entrada, es de **4.53 pixeles**, que ha sido obtenido promediando el error cuadrático medio de los histogramas de cada par de resultados en las 756 imágenes.

En una etapa posterior a la normalización del iris (misma que no es parte de este trabajo), las imágenes resultantes de la normalización son procesadas para obtener un código o una matriz binaria de identificación (figura 5.9), que permite la identificación mediante la medición de distancias entre matrices de identificación.

Figura 5.9. Matriz binaria de identificación

En [55] se hace uso de un filtro de Gabor para la obtención de la matriz binaria de identificación (figura 5.9) y de la distancia de Hamming como forma de medición de la distancia entre las matrices binarias de identificación. En el mismo trabajo son expuestos los resultados obtenidos proponiendo el valor de **0.4** de distancia de Hamming como valor de umbral para determinar si las dos matrices de identificación han sido obtenidas del mismo iris; si la distancia de Hamming es menor a este valor de rechazo, se trata del mismo iris, mientras que si es mayor a éste se trata de un iris diferente.

Aplicando el filtro de Gabor a las 756 imágenes resultantes del procesamiento en FPGA y a las 756 imágenes resultantes del procesamiento en la PC y midiendo la distancia entre ambos resultados con la misma imagen de entrada, el promedio de las distancias de Hamming calculadas de las 756 imágenes es de **0.0822**.

Dado que la distancia de Hamming es una medida de cuántos valores difieren ambas matrices binarias de identificación (figura 5.10) entre el número total de bits de las imágenes (ambas del mismo tamaño), en promedio las imágenes resultantes de la codificación de los resultados obtenidos en el FPGA y la PC difieren en un **8.22%.**

CAPÍTULO VI

Conclusiones y trabajos futuros

En el presente capítulo se detallan las conclusiones a las que se llegaron después de desarrollar el trabajo y analizar los resultados obtenidos, además se dan propuestas para trabajos futuros y las recomendaciones sobre el tema.

6.1. Conclusiones

Los algoritmos propuestos y más utilizados hallados en la literatura abierta y detallados en el estado del arte, hacen uso en su gran mayoría de técnicas de procesamiento de imágenes robustas, pero con un alto consumo computacional, ya que se basan en hacer circunferencias de diferentes dimensiones hasta encontrar la circunferencia que más coincide con los bordes que rodean al iris; sin embargo, en nuestro trabajo hemos logrado una técnica y unos algoritmos de procesamiento de imágenes de iris para su segmentación y su normalización sencillos que están a la altura de los ya desarrollados hasta este momento.

Se desarrolló una arquitectura de 3 etapas para la segmentación y normalización del iris humano de imágenes de la base de datos CASIA V1 en FPGA, dicho desarrollo lleva a cabo la tarea de segmentación y normalización del iris 10.27 veces más rápido que la mejor ejecución obtenida en una PC en lenguaje C/C++. Obteniendo como salida las imágenes de iris normalizados con una diferencia de menos del 10% de pixeles que difieren en cuanto a los resultados obtenidos en la PC y el FPGA.

Se desarrolló la técnica de segmentación y normalización del iris, donde se propone una técnica de segmentación y aproximación del borde exterior de iris. La evaluación de dicha técnica da como resultado un porcentaje de imágenes bien segmentadas de 95.6%, que compite de buena manera con los resultados obtenidos en diversos trabajos del ámbito, con la ventaja de que es un algoritmo sencillo de implementar y rápido en cuanto a tiempo de ejecución.

El desarrollo de la arquitectura base es esencial para la implementación del algoritmo en alguna plataforma FPGA, porque nos permite controlar el flujo de los datos de entrada y salida, el control y manejo de los periféricos de la implementación, así como también la comunicación entre el usuario y el sistema. En este trabajo se desarrolló la arquitectura base en un procesador NIOS II para la lectura y escritura de imágenes en formato BMP de una tarjeta de memoria SD, para el manejo y control de una memoria o *buffer* de video para la visualización de las imágenes en un *display* VGA y para el manejo y control de datos para la realización del procesamiento en módulos de hardware. Esta arquitectura base tiene como principal característica que está basada en un procesador NIOS II y las tareas mencionadas son programadas en lenguaje C/C++, lo que también posibilita la implementación de algunas otras tareas de procesamiento en ella.

Se implementó en FPGA la técnica desarrollada para la segmentación y normalización de iris humano. Para la implementación de la técnica o algoritmo desarrollado, existen diversas maneras de implementarlo en FPGA. En este trabajo se implementaron tres de las formas más utilizadas que son: la implementación de módulos de hardware que ejecutan completamente el algoritmo; la implantación del algoritmo en un procesador suave haciendo uso del lenguaje C/C++ en dicha implementación; y la combinación de software (C/C++) y hardware, es decir la implementación del algoritmo en software ejecutándose en el procesador suave, auxiliándose de módulos de hardware que realizan al mismo tiempo cálculos matemáticos o tareas específicas, liberando así al procesador de carga computacional. Los resultados obtenidos en cuanto a tiempo muestran que la mejor de estas tres implementaciones es la implementación de la tarea completamente en hardware,

mientras que la que mostró ser la más lenta es la implementación del algoritmo completamente en el procesador suave (NIOS II).

Se evaluaron las imágenes resultantes del desarrollo e implementación de la tecnicaq en FPGA obteniendo las imágenes y los resultados esperados, es decir, dadas las imágenes de entrada, el algoritmo al igual que en una PC, ejecuta la tarea de procesamiento y como salida da la imagen del iris normalizado, con las mismas características (formato BMP, 8 bits por color) que las obtenidas en la PC.

Se compararon las imágenes resultantes de la PC y de desarrollo en FPGA de la implementación de la técnica desarrollada. En promedio hay un error cuadrático medio de 4.53 pixeles entre el histograma de la imagen resultado obtenida en la PC y la obtenida en el FPGA. Este valor de error se debe a representación numérica en ambas plataformas, Punto fijo y enteros para el caso de la plataforma FPGA y punto flotante y enteros en el caso de la PC; a pesar de este valor de error, al obtenerse la plantilla del iris o matriz binaria de identificación, hay un porcentaje de diferencia menor al 10% (0.0822, que equivale al 8.22%) entre las mismas.

Con la implementación de procesamiento de imágenes en hardware, se gana en velocidad.

Este trabajo se realizó teniendo como hipótesis e idea base que cualquier algoritmo puede alcanzar su máxima velocidad si es implementado en hardware [12]; los resultados obtenidos en este trabajo así lo demuestran.

La implementación completa de una tarea en hardware reconfigurable mediante algún lenguaje de descripción de hardware, se ejecuta por completo en un menor tiempo y en una menor cantidad de ciclos de reloj. Como se observa de los resultados de la primera etapa y la tercera etapa, la implementación de ambas etapas fue implementada en hardware, siendo desarrollada la arquitectura de los módulos en el lenguaje de descripción de hardware VERILOG. Como se mostró en el capítulo 5, la tarea de esta etapa se ejecuta 175 veces más rápido que el mejor resultado obtenido en una PC (primera etapa); esto corrobora la idea e hipótesis base de este trabajo, "…cualquier algoritmo puede alcanzar su máxima velocidad si es implementado en hardware [12]".

La implementación realizada de la segunda etapa (software) y tercera etapa (software-hardware) en el FPGA, a pesar de haberse ejecutado en un tiempo mayor al de la mejor implementación en la PC, hay que hacer notar que la PC es alimentada por una señal de reloj 50 veces mayor que la frecuencia de reloj que alimenta al FPGA y aún con esta gran diferencia, el mejor tiempo obtenido en la PC es tan solo 8.7 veces mayor, ya que al ser la PC un equipo de propósito general, está gobernado por un sistema operativo que ejecuta diversas tareas aparte de la de procesamiento. Para hacer más equitativa la comparación, se compara la cantidad de ciclos de reloj que requiere la ejecución de las tareas y ahí es donde el FPGA muestra grandes ventajas, ya que requiere de mucho menos ciclos de reloj para la ejecución del algoritmo (del orden de aproximadamente 20 veces menos).

Una característica importante que se obtiene de la implementación de tareas o algoritmos en FPGA, es que como todo se diseña con circuitos digitales, la cantidad de ciclos de reloj y por lo tanto el tiempo de ejecución es el mismo en todas las ejecuciones, porque está dedicado a hacer solo esa tarea siempre de la misma manera; esto no sucede en una PC, donde al ejecutarse un ciclo, una iteración puede ejecutarse en un tiempo X, pero la siguiente iteración que realiza la tarea puede ejecutarse en un tiempo Y, es decir, no siempre se ejecutan dos tareas iguales en el mismo tiempo.

La implementación de algoritmos en FPGA tiene como gran ventaja que el algoritmo se ejecuta y da los mismos resultados que su implementación en la PC, pero con un consumo mucho menor de energía que la PC, ya que son sistemas dedicados compuestos de circuitos integrados de bajo consumo en comparación con lo que consume una PC, además de que un circuito integrado FPGA y sus periféricos, requieren mucho menos espacio y pueden ser utilizados en aplicaciones donde las restricciones de espacio y tiempo son especiales.

Los algoritmos de procesamiento de imágenes tienen características ideales para su implementación en FPGA, ya que los procesamientos se realizan de manera puntual en cada píxel; esto permite segmentar las imágenes y realizar varios procesamientos idénticos al mismo tiempo (paralelizar el procesamiento), lo que permite obtener los resultados en muy poco tiempo, con menos consumo de energía (en comparación con una PC) y a un menor costo.

6.2. Trabajos futuros

Realizar la segmentación del iris haciendo usos de otras herramientas computacionales como la implementación de una red neuronal o un clasificador.

Implementar todas las etapas del algoritmo en módulos de hardware externos e independientes al procesador suave, utilizando este último sólo para el control de los datos tanto en la entrada, salida y activación de los módulos (señal de inicio, "*start"*).

Realizar la implementación de la propuesta en este trabajo en una tarjeta de desarrollo FPGA, donde se encuentre integrado un procesador físico (ARM, Intel, etc.),utilizando este procesador y sus periféricos como arquitectura base y solo implementar los módulos propuestos en los recursos de lógica programable (FPGA).

En el caso de la implementación de un sistema de reconocimiento de personas por medio de imágenes de iris en FPGA, implementar 2 arquitecturas igualitas, debidamente sincronizadas para el procesamiento de las imágenes de ambos ojos de manera concurrente.

Implementar los módulos propuestos y desarrollar una arquitectura base con las herramientas de desarrollo proporcionadas por las plataformas y tarjetas de desarrollo FPGA de otros distribuidores como es Xilinx.

6.3. Recomendaciones

Tomar en cuenta desde la etapa de desarrollo del o los algoritmos a implementar en FPGA los recursos de hardware y software con los que se cuenta para el desarrollo de la implementación, es decir, si se cuenta con licencias o herramientas de desarrollo de aplicaciones en FPGA, como pueden ser herramientas para convertir algoritmos en un lenguaje de alto nivel como puede ser Matlab a lenguajes de descripción de hardware (para el caso de Altera, la herramienta es llamada "dsp builder"), se pueden desarrollar algoritmos con procesamientos más elaborados de una manera más sencilla, ya que estas herramientas se encargan de generar la arquitectura necesaria para llevar a cabo la tarea programada. Por otro lado, es importante tener en cuenta los recursos de hardware en el FPGA (en nuestro caso la tarjeta de desarrollo DE2-115), así como los puertos de entrada y salida con los que se cuenta, al momento de desarrollar el algoritmo.

Implementar la mayor parte del algoritmo en módulos de hardware, de manera que las múltiples operaciones a realizarse en dicho algoritmo se puedan ejecutar de manera concurrente, es decir la mayor cantidad de operaciones al mismo tiempo.

En el caso de realizar implementaciones en FPGA de algoritmos de procesamiento de imágenes, usar lo mostrado en este trabajo como base, ya que como se puede observar en los resultados, por medio de implementación de módulos en hardware, se pueden crear subconjuntos de pixeles de la imagen y hacer el procesamiento de estos subconjuntos de manera concurrente, lo que posibilita obtener los resultados en un tiempo cortos y una cantidad de ciclos de reloj muy reducida.

Actualmente se han venido desarrollando herramientas para la programación de sistemas embebidos en tarjetas FPGA. Hacer uso de estas herramientas como por ejemplo LABVIEW, que permite la realización de sistemas híbridos, permite programar instrumentos virtuales en una PC y también mediante su herramienta llamada “LABVIEW FPGA” permite implementar arquitecturas en tarjetas FPGA que realizan el procesamiento de señales, imágenes, así como el control, de algunos procesos sin depender de la PC, pero teniendo la posibilidad de comunicar la PC con esta implementación para crear un sistema más grande y robusto. Esta herramienta permite hacer uso de su programación gráfica para el diseño e implementación de aplicaciones embebidas en FPGA.

Referencias

[1] Ratha, N.K.; Jain, A.K.; Computer Vision Algorithms on Reconfigurable Logic Arrays. IEEE, Transactions on Parallel and Distributed Systems, Vol. 10, No. 1, pp. 29-43, 1999.

[2] Hamid, G.; An FPGA-Bases Coprocessor for Image Processing. IEEE, Colloquium on Integrated Imaging Sensors and Processing, pp. 6/1-6/4, 1994.

[3] Moore, G.; Cramming more components onto integrated circuits. Proceedings of the IEEE, vol. 86, pp: 82-85, 1965.

[4] Tessier, R.; Burleson, W.; Reconfigurable Computing for Digital Signal Processing: A survey. Journal of VLSI Signal Processing 28, pp: 7-27, 1999.

[5] Arnold, J. M.; Buell, D. A.; Davis, E. G.; SPLASH 2, Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures, pp: 316-322, 1992.

[6] Ziegler, H.; So Byoungro; Hall, M.; Diniz, P.C., Coarse-grain pipelining on multiple FPGA architectures. Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 02), pp: 77–86, 2002.

[7] Piacentino, M.R.; van der Wal, G.S.; Hansen, M.W.; Reconfigurable elements for a video pipeline processor. Proceedings of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 99), pp: 82–91, 1999.

[8] Wiatr, K.; Pipeline architecture of specialized Reconfigurable Processors in FPGA Structures for Real Time Image Pre-Processing. Proceedings of the 3rd International Conference on Signal Processing, vol. 1, pp: 131-138, 1996.

[9] Bouridane, A.; Crookes, D.; Donachy, P.; Alotaibi, K.; Benkrid, K.; A high level FPGA-based abstract machine for image processing. Journal of Systems Architecture, vol. 45, pp: 809-824, 1999.

[10] Quenot, G.M.; Kraljic, I.C.; Serot, J.; Zavidovique, B.; A reconfigurable compute engine for real-time vision automata prototyping. Proceedings of the IEEE Workshop on FPGA for Custom Computing Machines (FCCM 94), pp: 91–100, 1994.

[11] Boluda, J.A.; Arquitectura de procesamiento de imágenes basada en lógica reconfigurable para navegación de vehículos autónomos con visión foveal. Tesis Doctoral. Departamento de Informática de la Universidad de Valencia, 2000.

[12] Casselman, S.; Virtual Computing and the Virtual Computer. Proceedings IEEE Workshop on FPGAs for Custom Computing Machines, pp: 43-48, 1993.

[13] Jamro, E.; Wiatr, K.; Convolution Operation Implemented in FPGA Structures for Real-Time Image Processing. Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis (ISPA 2001), pp: 417-422, 2001.

[14] Dawood, A.S.; Visser, S.J.; Williams, J.A.; Reconfigurable FPGAs for real time processing in Space. Proceedings of the 14th International Conference on Digital Signal Processing (DSP 2002), vol. 2, pp: 845-848, 2002.

[15] Draper, B.A.; Beveridge, J.R.; Willem Böhm, A.P.; Ross, C.; Chawathe, M.; Accelerated Image Processing on FPGAs. IEEE Transactions on image processing, vol 12, Issue: 12, pp: 1543–1551, 2003.

[16] Kessal, L.; Abel, N.; Demigny, D.; Real-time image processing with dynamically reconfigurable architecture. Journal real-time imaging, vol. 9, pp: 297-313, 2003.

[17] Knittel, G.; A PCI-compatible FPGA-Coprocessor for 2D/3D Image Processing. Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines, pp: 136–145, 1996.

[18] Talu, H.M.; Igci, E.; Tekin, M.E.; Sevtekin, H.S.; Genç, B.Ç, Heywood, M.I.; Reconfigurable computing implementation of binary morphological operators using 4-, 6- and 8- connectivity. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '00), vol. 6, pp: 3386–3389, 2000.

[19] Arias-Estrada, M.; Torres-Huitzil, C.; Real time field programmable gate array architecture for computer vision. Journal of Electronic Imaging, vol. 10, pp: 289-296, 2001.

[20] Arias-Estrada, M.; Rodríguez-Palacios, E.; An FPGA co-processor for Realtime visual tracking. Proceedings of the 12th International Conference on Field-Programmable Logic and Applications (FPL 01), pp: 710–719, 2001.

[21] Battle, J.; Martí, J.; Ridao, P.; Amat, J.; A new FPGA/DSP- Based Parallel architecture for real-time image processing. Journal Real-Time imaging, vol.8, pp: 345-356, 2002.

[22] Meribout, M.; Nakanishi, M.; Ogura; T.; Accurate and real-time image processing on a new PC-Compatible Board. Journal Real-Time imaging, vol. 8, pp: 35-51, 2002.

[23] Meribout, M.; Nakanishi, M.; Ogura; T.; A parallel algorithm for real-time object recognition. Journal Pattern Recognition, vol. 35, pp: 1917-1931, 2002.

[24] Hernández, A.; Gardel, A.; Mateos, R.; Bravo, I.; Andrés, A.; VHDL Based Hough transform specification using CORDIC. Actas Semanario Anual de Automática e Instrumentación (SAAEI 04), ponencia: 261, 2004.

[25] Martín, J.L.; Zuloaga, A.; Cuadrado, C.; Lázaro, J.; Bidarte, U.; Hardware implementation of optical flow constraint equation using FPGAs. Computer Vision and Image Understanding, vol. 48, issue 3, pp: 462-490, 2005.

[26] Díaz, J.; Multimodal bio-inspired vision system, High performance motion and stereo processing architecture. Tesis doctoral. Departamento de Arquitectura y Tecnología de Computadores, Universidad de Granada, 2006.

[27] Wildes, Richard P;Iris Recognition: An Emerging Biometric Technology. Proceedings of the IEEE, vol. 85, pp: 1348–1363, 1997.

[28] Daugman, John; How iris recognition works. International Conference on Image Processing, Vol.1, pp: I-33 - I-36, 2004.

[29] Boles, W. y B. Boashash; A human identification technique using images of the iris and wavelet transform. IEEE Transactions on Signal Processing vol. 46, pp: 1185-1188, 1998.

[30] Kim, J.; Cho, S.; Choi, J. y R. ;Marks; Iris recognition using Wavelets features, Journal of VSLI Signal Processing vol.38, pp: 147-157, 2004.

[31] Ma, L.; Wang y T. Tan; Iris recognition based on multichannel Gabor filtering. 5th Asian Conference on Computer Vision, Melbourne, Australia, 2002.

[32] Daugman, John;The importance of being random: statistical principles of iris recognition. PatternRecognition vol. 36, pp: 279-291, 2003.

[33] Cui, J.; Y. Wang, T. Tan y S. Sun; An appearance-based method for iris detection. National Laboratory of Pattern Recognition, Chinese Academy of Sciences, 2003.

[34] Greco, J.; Kallenborn, D.; Nechyha, M.; Statistical pattern recognition of the iris, 2003.

[35] Huang, Y; Luo, S y E. Chen; An efficient iris recognition system. Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, 2002

[36] Daugman, J.; High confidence visual recognition of persons by a test of statistical independence, IEEE Trans. Pattern Analysis Machine Intelligence, Vol. 15, no. 11, pp. 1148-1161, 1993.

[37] Daugman, J. G.; Biometric personal identification system based on iris analysis, U.S. Patent No. 5291560, 1994.

[38] Sossa-Azuela; Rasgos descriptores para el reconocimiento de objetos. SEP. Instituto Politécnico Nacional. Primera edición. México, 2006.

[39] González, R. C.; Woods R. E.; Digital Image Processing. Prentice Hall. Third Edition, New Jersey, 2008.

[40] Chablé-Razo, T. I.; Felipe-Riverón, E. M.; Sánchez-Garfias, F. A.;Caracterización de la papila óptica en imágenes oftalmoscópicas de retina humana. Tesis. Instituto Politécnico Nacional, Escuela Superior de Cómputo. México, 2006.

[41] Mijail A. del Toro-Céspedes, Edgardo M. Felipe-Riverón, Roberto Rodríguez Morales.;Medición de parámetros del disco óptico en imágenes de retina. Tesis. Universidad de La Habana. Facultad de Matemática y Computación, 2003.

[42] Baxes, G. A.; Digital Image Processing - Principles and Applications. USA, John Wiley & Sons, 1994.

[43] Ghosh,Sumit.;Hardware Description Languages; IEEE Press, 2000.

[44] Coffman, ken.; Real World FPGA Design with Verilog; Prentice Hall, 2000.

[45] Hill, R. B.; Apparatus and method for identifying individuals through their retinal vasculature patterns, U.S. Patent No. 4109237, 1978.

[46] Rathgeb, C.; A. Uhl, A.; and Wild, P.; Iris Recognition: From Segmentation to Template. Advances in Information Security. Springer, New York, 2012.

[47] Bertillon,A.; la couleur de l'iris, Revue scientifique, France, 1885.

[48] Leonard Flom, Aran Safir; Iris recognition system, U.S. Patnt No. 4641349,1986.

[49] Chinese Academy of Sciences, CASIA-IrisV1, http://biometrics.idealtest.org/

[50] Ajay Kumar and Arun Passi; Comparison and combination of IRIS matchers for reliable personal authentication Pattern recognition, vol. 43, PP: 1016–1026, 2010

[51] http://www.altera.com/education/univ/unv-index.html

[52] Passarello, G.; Imágenes médicas, Adquisición, Análisis. Editorial Equinoccio, 1999.

[53] Raúl Sánchez Reillo; El iris ocular como parámetro para la identificación biométrica. Universidad Politécnica de Madrid, España, 2000.

[54] Paul L. Kaufman, Albert Lam. Adler; Fisiología del Ojo. Décima edición. Editorial Elsevier. Madrid, España, 2003.

[55] Libor Masek, Peter Kovesi. MATLAB Source Code for a Biometric Identification System Based on Iris Patterns. The School of Computer Science and Software Engineering, The University of Western Australia. 2003.

[56] Oppenheim, A. V.; Lim, J. S.; The importance of phase in signals. Proceedings of the IEEE, vol. 69, pp: 529-541, 1981