RNPS No. 2142
ISSN 2072-6287
Versión Digital

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

REPORTE TÉCNICO
Reconocimiento de Patrones

SERIE AZUL

Reconocimiento automático de rostros usando FPGAs

Ing. Leonardo Chang Fernández,
Dr. C. José Hernández Palancar

RT_013 **octubre 2010**

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

RNPS No. 2142
ISSN 2072-6287
Versión Digital

SERIE AZUL

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

REPORTE TÉCNICO
Reconocimiento de Patrones

Reconocimiento automático de rostros usando FPGAs

Ing. Leonardo Chang Fernández,
Dr. C. José Hernández Palancar

RT_013 **octubre 2010**

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

Índice

1. Introducción 1
2. Diseño de algoritmos sobre FPGAs 2
   - 2.1. FPGAs 2
   - 2.2. FPGAs y su relación con el procesamiento paralelo 3
3. Reconocimiento de rostros usando FPGAs 5
   - 3.1. Reconocimiento de rostros en paralelo no basado en FPGAs 6
   - 3.2. Arquitecturas FPGA para el reconocimiento automático de rostros 7
     - 3.2.1. Arquitecturas FPGA para la detección de rostros 7
     - 3.2.2. Arquitecturas FPGA para la extracción de rasgos del rostro 11
4. SIFT 14
   - 4.1. Detección de puntos claves candidatos en el SIFT 15
   - 4.2. Algoritmo paralelo propuesto para la detección de puntos claves candidatos en el SIFT 16
     - 4.2.1. Explotando el paralelismo de datos 16
     - 4.2.2. Explotando la propiedad de separabilidad del kernel Gaussiano . . . 16
     - 4.2.3. Entrelazando el procesamiento de octavas 17
   - 4.3. Arquitectura hardware propuesta 18
   - 4.4. Implementación en FPGA y resultados experimentales 19
     - 4.4.1. Características de la implementación 19
     - 4.4.2. Confiabilidad del sistema 20
5. Conclusiones 21

Índice de figuras

1. Estructura de los FPGAs 2
2. Filtro espacial 4
3. Diferencias entre modelo de (a) memoria compartida y (b) de memoria distribuida 4
4. Diferencias entre (a) procesamiento secuencial, (b) paralelismo de tareas y (c) paralelismo de datos 5
5. Flujo de procesamiento de un sistema de reconocimiento de rostros 6
6. Estructura general del algoritmo de detección de rostro propuesto por [22] 7
7. Estructura general del algoritmo de detección de rostro propuesto por [22] 9
8. Esquema a alto nivel de la arquitectura propuesta por [34] 10
9. Red Neuronal de Perceptrón Multicapa implementada por [33] 11
10. Arquitectura hardware propuesta por [33] 12
11. Arquitectura propuesta en [25] para la implementación del WMPCA 13
12. Arquitectura tunelizada propuesta. 18
13. Correspondencias entre la implementación híbrida (usando los resultados de nuestro sistema) y la implementación de Lowe para una imagen de ejemplo. 20
14. Errores en coordenadas, escala y orientación en los puntos detectados en una imagen de ejemplo. 21

Reconocimiento automático de rostros usando FPGAs

Ing. Leonardo Chang Fernández, Dr. C. José Hernández Palancar

Centro de Aplicaciones de Tecnología de Avanzada, 7a #21812 e/ 218 y 222, Siboney, Playa, Habana, Cuba
lchang@cenatav.co.cu

RT_013 CENATAV
Fecha del camera-ready: 31 de julio de 2009

Resumen: El reconocimiento automático de rostros es un campo que ha recibido mucha atención en los últimos años. La mayoría de los investigadores se han dedicado al diseño de algoritmos eficaces en software, dedicándole poca atención al diseño de algoritmos eficientes bajo restricciones de tiempo y consumo de energía. En este trabajo se estudia el uso de los FPGA en la aceleración del proceso de reconocimiento de rostros. También, se propone un algoritmo para el procesamiento paralelo de una de las partes más costosas computacionalmente del popular algoritmo SIFT, así como una arquitectura hardware que lo implemente.
Palabras clave: reconocimiento de rostros en FPGA, arquitectura paralela SIFT

Abstract: Face recognition during the past several years has received significant attention. Most of the researches has focused on the design of accurate software based algorithms, while the implementation cost in the form of, for example, speed and power consumption has been on a lower priority. In this work, the used of FPGA in the face recognition process speeding up is studied. Also, a parallel algorithm and architecture for one of the most computationally expensive parts of SIFT is proposed.
Keywords: FPGA-based Face Recognition, Parallel SIFT Architecture

1. Introducción

Los seres humanos usan la visión como medio principal para obtener información de su entorno. Darle esta misma habilidad a los sistemas automatizados sería un gran paso de avance en aras del desarrollo de la convivencia hombre-máquina. El área de investigación que aborda el análisis automatizado del rostro humano ha atraído particular atención en la última década. Un gran número de aplicaciones comerciales, de seguridad y de defensa requieren de sistemas automáticos de reconocimiento de rostros, por ejemplo: tarjetas inteligentes, sistemas de entretenimiento, autenticación, seguridad, vigilancia e interacción hombre-máquina [39] [13].

La mayoría de los investigadores del reconocimiento automático de rostros se han concentrado en el desarrollo de algoritmos de segmentación, y de extracción y comparación de rasgos, los cuales son generalmente implementados en software en computadoras personales. Sin embargo, muchas aplicaciones como control de acceso y seguridad, interacción hombre-máquina y codificación de video basada en modelos demandan tiempos de respuesta muy cortos, por lo que necesitan trabajar a gran velocidad y en tiempo real, por ejemplo, permitir el acceso a una instalación a un individuo sin demoras ni retrasos y al mismo tiempo comprobando su identidad con o sin su concentimiento. El creciente número de sujetos

a controlar y la cantidad de tareas que son necesarias realizar en tiempo real han llevado a que las soluciones software no sean computacionalmente viables. Una solución que un gran número de investigadores ha aplicado satisfactoriamente con el objetivo de acelerar distintos algoritmos de cómputo es el uso de los FPGAs (del inglés *Field Programmable Gate Array*).

Este reporte estudia la aplicación de la tecnología FPGA en el reconocimiento automático de rostros. Algunos de conceptos básicos de la computación paralela y su relación con los FPGAs son tratados. Por otra parte se propone un algoritmo paralelo y su arquitectura hardware de una de las etapas más costosas computacionalmente del algoritmo SIFT.

2. Diseño de algoritmos sobre FPGAs

2.1. FPGAs

El concepto original de FPGA fue desarrollado por Ross Freeman uno de los fundadores de Xilinx[1] en 1984. La idea de Ross Freeman fue que en vez de usar un procesador genérico y escribir software para que corriese sobre él, valdría la pena personalizar chips electrónicos para realizar tareas específicas programando directamente sobre estos chips. El nombre genérico de este tipo de chip es Dispositivo Lógico Programable (PLD) de los cuales los FPGAs son un subconjunto.

Un dispositivo FPGA es un arreglo de unidades de procesamiento de bits cuya interconexión y funcionalidad se puede programar después de su fabricación. [8]

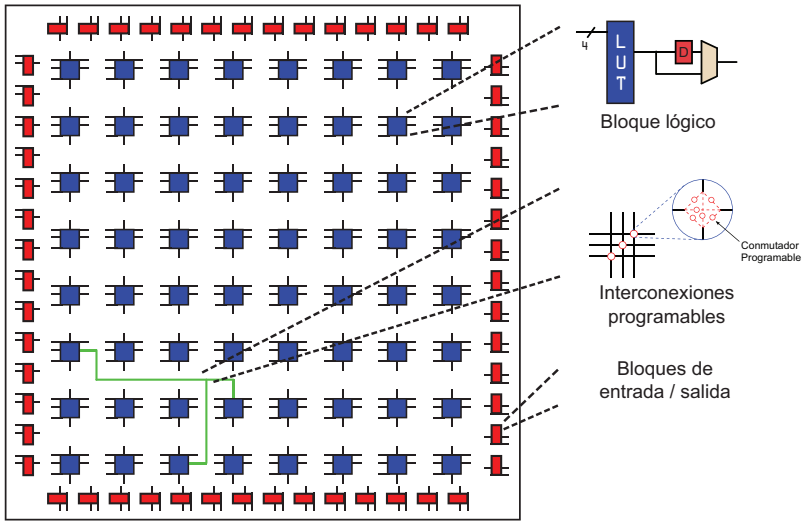

Fig. 1: Estructura de los FPGAs

La mayoría de los FPGAs tradicionales están formados a partir de un bloque lógico, duplicado cientos y miles de veces. Un bloque lógico es básicamente una pequeña tabla de

[1] Compañía líder mundial en la fabricación de FPGAs y CPLDs

verdad (LUT), un biestable tipo D y un multiplexor 2-a-1. La LUT es como una pequeña RAM y generalmente en estos dispositivos tiene 4 entradas, por lo que se puede implementar cualquier compuerta lógica de hasta 4 entradas. Por ejemplo, una compuerta AND con 3 entradas cuyo resultado es luego OR-ado con otra entrada cabría en una de estas LUT. En un FPGA cada bloque lógico puede ser conectado a otros a través de recursos de interconexión (cables alrededor de las bloques lógicos). Un bloque por sí solo no puede hacer mucho, pero varios de ellos interconectados pueden crear complejas funciones lógicas. Los cables de interconexión también llegan hasta los bordes del dispositivo, donde los bloques de entrada / salida son implementados y conectados a los pines del FPGA. En adición a los medios de interconexión de propósito general, los FPGAs tienen líneas dedicadas muy rápidas entre los bloques vecinos. El tipo más común de este tipo de líneas son las *carry chains*, las que permiten crear funciones aritméticas (como contadores y sumadores) eficientemente (mínima cantidad de bloques y alta velocidad en las operaciones) [8]. Un diagrama de esta estructura se muestra en la Figura 1.

2.2. FPGAs y su relación con el procesamiento paralelo

La mayoría de los procesadores de propósito general (CPUs) cuentan con relojes hasta 50 veces más rápidos que los de un FPGA. La diferencia radica en que un CPU convencional solo puede procesar una instrucción por ciclo de reloj. Por el contrario, un FPGA puede ser configurado como varios procesadores capaces de funcionar en paralelo. Algunos FPGAs pueden contener hasta millones de bloques lógicos, por lo que para una sencilla tarea como la comparación de una cadena de texto se podría contar con un FPGA con miles de procesadores trabajando simultáneamente. Una configuración como esta sin lugar a dudas haría despreciable una desventaja de velocidad de reloj de 50 o 100 veces.

Por tanto, para alcanzar tales provechos es indispensable explotar el paralelismo de los algoritmos. En esta sección, dada su importancia, se analizan algunos conceptos básicos del paralelismo.

El paralelismo se puede dividir en dos grandes grupos: el paralelismo temporal y el paralelismo espacial [11]. El paralelismo temporal o tunelizado se refiere a ejecutar una tarea como una cascada de sub-tareas, donde existe una unidad de procesamiento para realizar cada sub-tarea. Estas unidades trabajan al mismo tiempo de modo solapado, como una línea de producción de una fábrica. El paralelismo espacial hace mención a la ejecución simultánea de tareas por varias unidades de procesamiento, donde estas unidades ejecutan la misma tarea o diferentes tareas en un instante dado.

En un FPGA se pueden explotar ambos tipos de paralelismo, veamos el siguiente ejemplo de un filtro espacial $y_i = \sum_{k=0}^{3} w_k x_{i-k}$. En la Figura 2 se puede observar el paralelismo espacial en los diferentes multiplicadores o sumadores, donde simultáneamente una misma operación es realizada por distintas unidades de procesamiento. En este ejemplo también está presente el tunelizado, que es una técnica muy usada en el diseño de arquitecturas para este tipo de dispositivos, logrando retornar un resultado por ciclo de reloj luego de una latencia inicial. En este ejemplo notamos como para obtener un resultado todas sus sub-tareas no son realizadas concurrentemente, sino desplazadas en el tiempo, manteniéndose

todas las unidades de procesamiento en funcionamiento generando sub-resultados de distintas entradas.

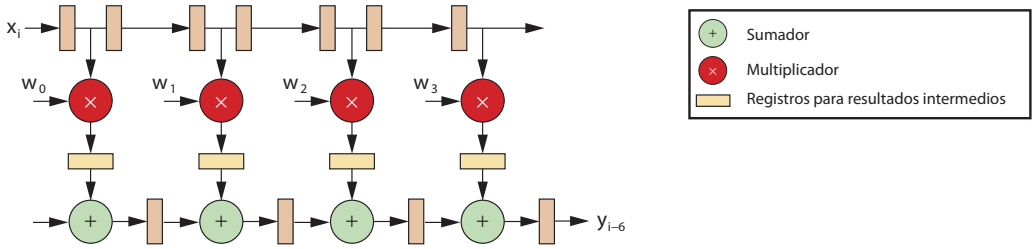

Fig. 2: Filtro espacial

En un algoritmo paralelo una propiedad significativa es el tipo de modelo de memoria empleado. Existen fundamentalmente dos esquemas de memoria: compartida y distribuida. En un modelo de memoria compartida cada unidad de procesamiento tiene acceso directo a un espacio de memoria global, mientras que en el modelo de memoria distribuida cada procesador tiene acceso directo a su correspondiente espacio de memoria local. Esta diferencia se puede apreciar en la Figura 3.

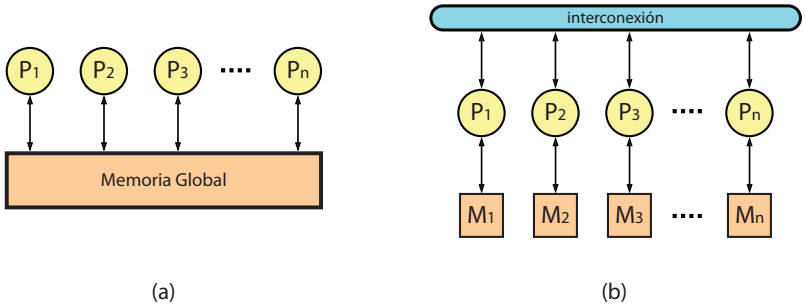

Fig. 3: Diferencias entre modelo de (a) memoria compartida y (b) de memoria distribuida

Para la descomposición del problema existen dos modelos fundamentalmente, estos son el paralelismo de datos y el paralelismo de tareas. El paralelismo de datos se refiere a la ejecución de la misma operación sobre varios subconjuntos de datos al mismo tiempo, mientras que el paralelismo de tareas se refiere a la ejecución concurrente de varias tareas. Ambos modelos se pueden juntar para alcanzar un mayor rendimiento. Las diferencias fundamentales entre estos modelos y el procesamiento secuencial pueden ser observadas en la Figura 4. En los FPGAs el uso del paralelismo de datos es el más sencillo, solo requiere de replicar una unidad de procesamiento y darle como entrada a cada replica una partición del conjunto de datos de entrada y tener a salida un bloque que recolecte los resultados. El paralelismo de tareas generalmente se ve reflejado en el uso de diseños tunelizados o simplemente diferentes bloques de procesamiento realizando distintas tareas al mismo tiempo. El uso masivo del paralelismo en los FPGAs generalmente está acotado por el área del dispositivo ocupada, el tamaño del bus de datos del mismo y el consumo de corriente del sistema.

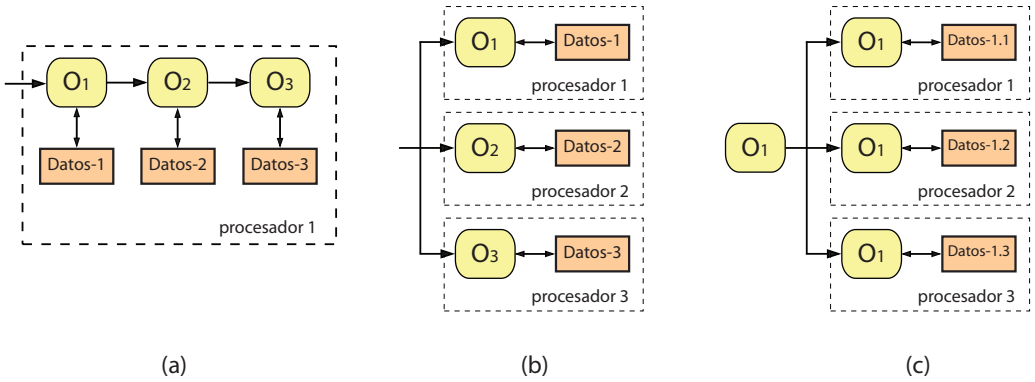

Fig. 4: Diferencias entre (a) procesamiento secuencial, (b) paralelismo de tareas y (c) paralelismo de datos

En la Figura 4 a) se muestra un flujo de procesamiento secuencial donde las operaciones son ejecutadas una a la vez sobre sus datos correspondientes. En la Figura 4 b) se muestran estas mismas operaciones realizadas usando paralelismo de tareas, donde, asumiendo que existen 3 unidades de procesamiento, las 3 operaciones son realizadas al mismo tiempo sobre su conjunto de datos correspondiente. En la Figura 4 c) se muestra la operación O_1 siendo ejecutada usando paralelismo de datos donde cada procesador aplica la misma operación sobre una parte del conjunto de datos.

Para evaluar la calidad de un sistema paralelo se utilizan las medidas: aceleración, eficiencia y escalabilidad. Para calcular la aceleración (Sp) tomemos t_1 como el tiempo total que se toma el algoritmo secuencial más rápido en un sistema de un solo procesador y t_p como el tiempo que se toma el algoritmo paralelo corriendo en una máquina paralela de p procesadores. Sp es la relación entre estos tiempos: $Sp = t_1/t_p$. Idealmente, un sistema paralelo debiera tener Sp lineal. Generalmente, para calcular la aceleración de un sistema FPGA se usa la relación entre el tiempo que se toma el algoritmo secuencial más rápido en un sistema de un solo procesador y el tiempo que se toma el sistema FPGA.

Nótese que Sp solo mide la efectividad en la explotación del paralelismo, sin tomar en cuenta la eficiencia de este proceso. Esta eficiencia (Ef), en un sistema de p procesadores, está dada por: $Ef = Sp/p$. En un sistema sobre FPGA esta medida de la forma anterior no cobra sentido, midiéndose entonces la relación entre la aceleración alcanzada y el área del dispositivo empleada.

La medida de escalabilidad (Sc) evalúa el comportamiento del sistema con respecto a un aumento en el tamaño del dispositivo y en el tamaño del problema a resolver.

3. Reconocimiento de rostros usando FPGAs

El reconocimiento de rostros es una actividad sencilla e intuitiva para los humanos, pero extremadamente complicada para un sistema por computadoras viéndose afectada en mayor parte por las variaciones del entorno y del propio sujeto.

Un enunciado para el problema del reconocimiento automático de rostros podría ser: dadas imágenes de una escena (estáticas o de video), identificar o verificar la identidad de

una o más personas en la escena usando una base de datos previamente almacenada[39]. La solución a este problema requiere de detección de rostros en la escena, extracción de rasgos y verificación o identificación (Fig. 5).

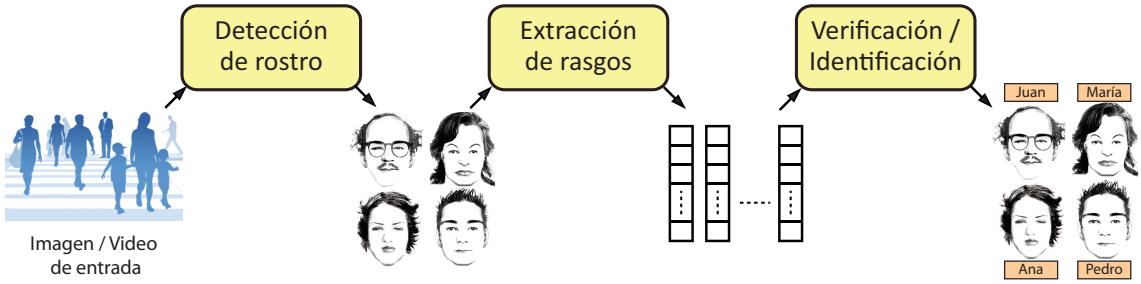

Fig. 5: Flujo de procesamiento de un sistema de reconocimiento de rostros

Las investigaciones en reconocimiento de rostros más recientes se han enfocado en desarrollar eficientes algoritmos secuenciales y sus respectivas implementaciones en software ya que la mayoría no cuentan con los recursos o experiencias requeridos para construir algoritmos paralelos y hardware personalizado. Sin embargo, algunos pocos investigadores se han dedicado a desarrollar algoritmos paralelos con estos fines y sus implementaciones tanto en hardware como software.

3.1. Reconocimiento de rostros en paralelo no basado en FPGAs

Algunos sistemas en software han sido propuestos con el fin de acelerar el reconocimiento de rostros, estos principalmente hacen uso del paralelismo de datos, siendo implementados en su mayoría sobre clusters de computadoras. A continuación se comentan algunos de ellos.

En [18] los autores presentan un sistema paralelo de reconocimiento de rostros basado en clusters con el objetivo de acelerar el cotejo de rasgos, usando paralelismo de datos. En este trabajo la base de datos es particionada en pequeñas sub-bases de datos y distribuida a lo largo del cluster. El tiempo de reconocimiento en una base de datos de 100 000 individuos descendió de 166.059 segundos en un solo nodo a 40.172 segundos en un cluster de cinco nodos, mostrándose una aceleración de aproximadamente 4x.

En [37] se propone un nuevo método, *eigenblock*, con el objetivo de lidiar con la oclusión y la distorsión local en las imágenes de rostro. Este método se basa en el popular algoritmo PCA y las imágenes de rostro son particionadas en bloques sin solapamiento y luego PCA es aplicado sobre estos bloques. En la descripción de su algoritmo los autores señalan el paralelismo intrínseco de su propuesta, dando la posibilidad de explotar el paralelismo de datos ya que, luego de particionada la imagen de rostro el procesamiento en cada bloque es el mismo. Basado en este mismo algoritmo en [30] los autores presentan el método *Parallel Gabor Principal Component Analysis (PGPCA)*. La esencia del método es un proceso de reducción de dimensionalidad usando componentes principales, sin embargo, a diferencia del método PCA tradicional, a la base de datos se le aplica un conjunto de convoluciones

de wavelet de Gabor con el objetivo de extraer información de la frecuencia y orientación en las imágenes. Son usadas 40 wavelets: 8 orientaciones y 5 frecuencias, aumentando la dimensionalidad del problema 40 veces. Para tratar este aumento de dimensionalidad los autores procesan las imágenes convolucionadas con la misma wavelet en paralelo e independientemente. En este caso se hace uso también del paralelismo de datos.

3.2. Arquitecturas FPGA para el reconocimiento automático de rostros

En esta sección se discuten algunos trabajos que demuestran la eficacia y eficiencia en el uso de FPGAs para la solución de problemas de reconocimiento automático de rostros, reportándose aceleraciones superiores a las 10x y con muy poca pérdida de precisión en los resultados. Se analizan arquitecturas e implementaciones de algoritmos como Adaboosting, Redes Neuronales Artificiales, Análisis de Componentes Principales, wavelets y procesamiento de imágenes en general utilizados en la detección y extracción de rasgos del rostro. Evidenciándose también el uso de diversas arquitecturas como SoPC (*System on Programmable Chip*), NoC (*Network on Chip*) y SHcoD (*Software/Hardware co-Design*) entre otros.

3.2.1. Arquitecturas FPGA para la detección de rostros

La detección de rostros es el primer paso dentro del reconocimiento automático de rostros. Un detector de rostros ideal debería, dada una imagen o video, localizar todos los rostros presentes independientemente de su escala, posición, orientación, edad, sexo y expresión. Además que al ser el primer paso del proceso debe realizarse particularmente rápido.

Uno de las primeras implementaciones hardware propuestas en la literatura para la detección de rostros es la de McCready [22]. El autor propone un algoritmo de detección tratando que sea robusto y al mismo tiempo eficiente su implementación hardware. La estructura general de este algoritmo puede ser vista en la Figura 6.

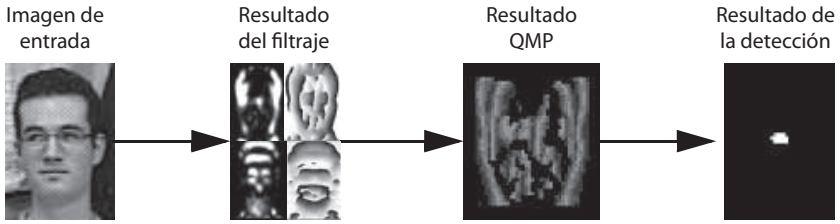

Fig. 6: Estructura general del algoritmo de detección de rostro propuesto por [22]

El primer paso es el filtrado de la imagen (una simple operación de convolución 2D) donde los filtros usados son los llamados "G2" y "H2"debido a sus propiedades de análisis al ser usados en conjunto según [10]. Luego la imagen es llevada a un formato llamado QMP (del inglés *Quantized Magnitude/Phase*) el cual es crítico para la eficiencia del sistema, ya que no solo es fácil de calcular y almacenar sino que reduce grandemente el tamaño del hardware eliminando la necesidad de multiplicaciones en la red de clasificadores. El clasificador de rostro utilizado es una red neuronal de una sola neurona. Juntos, el uso de

QMP y una sola neurona ahorran gran cantidad de recursos de hardware: planteando el autor que solamente con el uso de QMP se reduce en un 75 % el área del hardware.

Un diagrama en bloque del hardware propuesto por McCready se muestra en la Figura 7. El bloque de entrada recibe los datos de una tarjeta de video externa que contiene un chip Brooktree 812. A medida que los píxeles van llegando de la cámara pasan por una etapa de escalado la que reduce la imagen por un factor de 1.25 hasta que se alcanza el rango de tamaño requerido. De ahí, esos píxeles son pasados a través de los filtros G2 y H2. Para el filtrado de la imagen los autores se apoyan en un conjunto de herramientas de software. Una herramienta genera un conjunto de filtros de un tamaño específico y optimiza los coeficientes de los filtros de modo que cada coeficiente pueda ser expresado como la suma o diferencia de dos potencias de dos, lo que les permite no usar más de un sumador o substractor para calcular la multiplicación de un pixel con un coeficiente. Otra de las herramientas toma los coeficientes optimizados y crea una descripción hardware del filtro en el lenguaje AHDL de Altera. El bloque de conversión QMP toma los resultados del filtrado y calcula la representación QMP. El resultado final es dividido en 4 depósitos de fase, por lo que solo dos multiplicaciones y tres comparaciones son requeridas para encontrar el depósito de fase correcto. Para comprobar bajas magnitudes se necesitan dos unidades de 3 bits de elevación al cuadrado, un sumador de 7 bits y tres comparadores. Los dos bloques QMP pasan sus resultados al bloque de detección de rostros (FDU) donde se realiza la clasificación de rostro. Este bloque es por mucho el mayor del sistema, está compuesto por 32 unidades en forma de filas distribuidas en ocho FPGAs. El FDU se ocupa del área rectangular de 30×32 de resultados de detección, los que serán afectados por el próximo valor de QMP, estos valores se almacenan en un gran banco de registros. Cuando el nuevo valor QMP llega los valores de detección son actualizados y el contenido de los registros son desplazados hacia la izquierda, lo que lo deja listo para el próximo valor. El bloque de detección de máximos recibe las probabilidades de rostros del FDU y encuentra los máximos en el área de resultados ya que cada rostro producirá cierta respuesta en varias localizaciones adyacentes. Este bloque usa un matriz de comparadores de 3×3, si el valor del centro es mayor que sus ocho vecinos se toma como máximo y es pasado al bloque de visualización. El bloque de visualización marca estas localizaciones en la imagen de entrada y la muestra en un monitor conectado al mismo. El sistema trabaja a 30 cuadros por segundos y puede detectar cualquier cantidad de rostros en la imagen a varios tamaños. Esta implementación resultó 1000 veces más rápida que su versión en software. Se usaron nueve tarjetas de un sistema TM-2a [17], ocupó 31500 celdas lógicas de Altera, aproximadamente 14000 usadas por el FDU y 5000 por los bloques de filtrado.

En [34], los autores presentan una arquitectura la cual implementa el algoritmo propuesto por Viola y Jones en [36] que es uno de los algoritmos de detección de rostros más usados y ponderado por la comunidad científica en el estado del arte por sus índices de velocidad y eficacia. Este método presenta dos ventajas principales, es capaz de descartar rápidamente falsos positivos y el proceso de clasificación es extremadamente rápido. Cada rasgo propuesto por Viola y Jones consiste en un conjunto de rectángulos blancos y negros, donde su valor está dado por la diferencia de sumas de píxeles en cada una de las regiones rectangulares, dependiendo de la estructura del rasgo. Con el objetivo de acelerar el cálculo de los rasgos se introduce el uso de la imagen integral, donde el valor en el punto

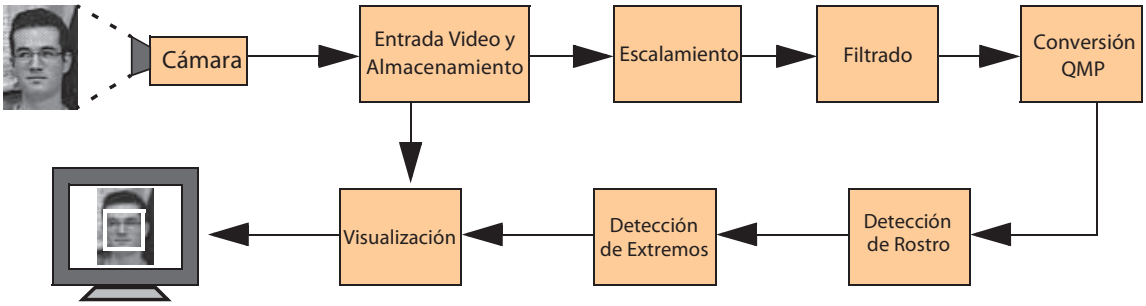

Fig. 7: Estructura general del algoritmo de detección de rostro propuesto por [22]

(x, y) es la suma de todos los píxeles hacia arriba y hacia la izquierda. Un grupo de rasgos componen una etapa, cuyo valor es el resultado de la suma de sus rasgos. El resultado de una etapa determina cuando una región de la imagen contiene un rostro o no. El tamaño del rasgo determina el tamaño de la región de búsqueda en la imagen, por lo que a partir del tamaño base son escalados aumentando su tamaño y evaluados para cada escala, permitiendo detectar rostros de mayor tamaño. Más detalles del algoritmo pueden ser vistos en [36].

Con el objetivo de contar con un acceso paralelo a la imagen integral, en [34] se propone como estructura de su arquitectura, el uso de un arreglo de procesadores en forma de rejilla. Este arreglo es usado como memoria para almacenar los datos de los cálculos y como unidad de transferencia de datos. El sistema consta de tres componentes principales: los bloques de transferencia y recopilación de datos (CDTUs), los bloques de multiplicación y evaluación (MEUs) y los bloques de control (CUs), los cuales colaboran entre sí. En esta arquitectura, y lo que consideramos constituye su mayor desventaja, el tamaño del arreglo depende del tamaño de la imagen, ya que una imagen de tamaño $m \times n$ requiere de un arreglo de $m \times n$ CDTUs, lo que indica que la arquitectura requiere excesivos recursos, dependiendo directamente de las dimensiones de la imagen de entrada. Sin embargo, esto se puede manejar usando técnicas como el escalado y particionado de la imagen. En la Figura 8 se brinda un esquema a alto nivel de dicha arquitectura mostrando la distribución de los bloques y el flujo de datos a través del sistema.

Cuando el tamaño de la ventana de búsqueda cambia, el cálculo es el mismo, se trata como si fuera una nueva. Al ser procesada la imagen a todos los tamaños de ventana el sistema está listo para una próxima imagen. La primera etapa es el cálculo de la imagen integral, el que consiste en adiciones y desplazamientos verticales y horizontales. Los píxeles que van llegando son desplazados al interior del arreglo en cada fila. En dependencia de la columna, cada unidad de cálculo realiza una de tres operaciones posibles. El cálculo de la imagen integral se toma $2 * [(m + (m - 1) + (n - 1)]$ ciclos de reloj para una imagen de $m \times n$. Luego, se realiza el cálculo de los rectángulos. Cuando finaliza la recopilación de las sumas de los rectángulos para un rasgo, estas son almacenadas en el CDTU que representa la esquina inicial para rasgo. Después, todas las sumas recopiladas son desplazadas a la izquierda hacia los MEUs, una a la vez por MEU; aquí, las sumas de los rectángulos son usadas junto a los datos obtenidos del entrenamiento en aras de evaluar el rasgo. El resultado de este rasgo es desplazado de forma anular hasta el CDTU en el extremo derecho

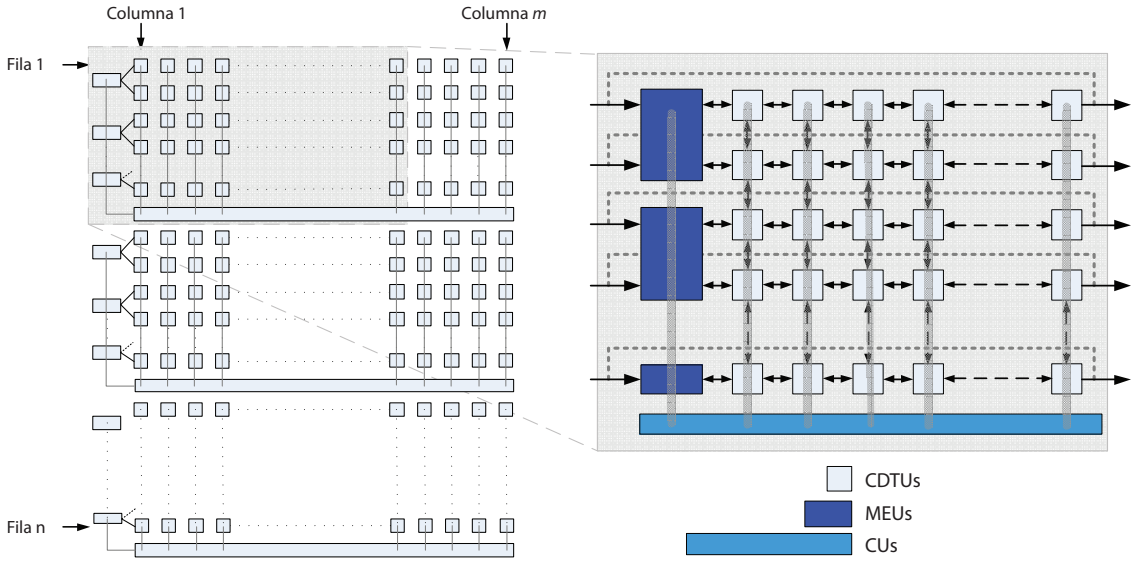

Fig. 8: Esquema a alto nivel de la arquitectura propuesta por [34]

del arreglo para continuar el cálculo. Cuando todos los valores de ese rasgo son calculados, son nuevamente almacenados en los CDTUs y el proceso recomienza con el próximo rasgo. Cuando todas las etapas son pasadas para una escala, las posiciones marcadas contienen un rostro. Si la escala calculada es la última, el proceso termina. Según pruebas realizadas su sistema logra trabajar a 52 cuadros por segundo bajo una frecuencia de reloj de 500 Mhz, aunque no se menciona el tamaño de las imágenes utilizadas para las pruebas, de lo que depende en gran medida este resultado.

Otras implementaciones hardware han sido propuestas basadas en el algoritmo de Adaboost. En [38] se presenta un sistema de detección de rostros con el objetivo de controlar automáticamente la exposición en una cámara digital portátil o en una cámara de un teléfono celular. Este sistema logra trabajar en tiempo real, para esto usan un diseño de hardware tunelizado para el cálculo de los rasgos. Aunque asumen que el ancho de banda, la latencia del bus y las interfaces están garantizadas. La arquitectura fue sintetizada a un FPGA Altera Cyclone II y los resultados de las simulaciones arrojaron una tasa de detección del 80 %. Recientemente en [15] se presenta un diseño hardware con una arquitectura NoC (*Network-on-Chip*) para este algoritmo. La arquitectura NoC brinda un alta capacidad de comunicación [2].El diseño fue verificado en un FPGA Xilinx Virtex II Pro, donde los resultados de la simulación mostraron una tasa de detección del 70 %, procesando imágenes de 320×240 a una velocidad de 40 imágenes por segundo. Otro ejemplo de diseño hardware de este algoritmo puede ser visto en [16] .

Las Redes Neuronales Artificiales (ANN) han sido ampliamente usadas en la detección de rostros. Estos sistemas fueron implementados en software por lo que son poco eficientes (tiempos de detección de alrededor de 10 segundos para imágenes de 320×240). En [33] se implementa un clasificador basado en una red neuronal MLP (Perceptrón Multi-Capa)

de 3 capas (capa de entrada, capa oculta y capa de salida). La capa de entrada de esta red es un vector constituido por n^2 neuronas (imágenes de entrada de $n \times n$ píxeles), donde cada una de ellas está conectada a las n neuronas de la capa oculta y la capa de salida está compuesta por una sola neurona la cual es activada a 1 si se detectó una cara (Fig. 9).

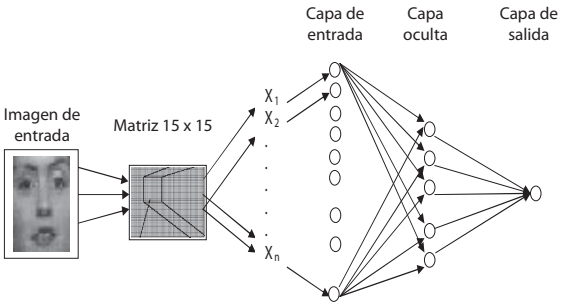

Fig. 9: Red Neuronal de Perceptrón Multicapa implementada por [33]

La actividad de una neurona j en la capa oculta está dada por $S_j = \sum_i w_{ji} x_i$, $x_i = f(s_j)$, donde w_{1i} es el conjunto de pesos de la neurona i, x_i es una entrada de la neurona j y f es una función sigmoidea. La actividad de la capa de salida está dada por la misma fórmula. En su sistema [33] usa una retina de dimensión 15×15 que determinan rostros y no rostros, el vector de entrada está formado por 225 neuronas y la capa oculta por 15 neuronas.

La arquitectura de este sistema está formada por cuatro bloques principales: banco de registros de entrada, unidades de control, neuronas y registros de salida (Ver Fig. 10). El cálculo de la actividad de cualquier neurona es realizado usando un método de multiplicación y acumulación donde los productos parciales son calculados por separado y luego adicionados. Cada elemento del vector de entrada de cada neurona es multiplicado por un valor de peso fijo, el cual es determinado en la fase de entrenamiento por una implementación software, estos valores se almacenan en una ROM en cada neurona. El resultado de la operación MAC se le pasa como entrada a una función de activación (tangente hiperbólica) y el resultado retornado. Dado que la implementación hardware de esta función en su forma conocida es muy complicada, los autores linealizan esta función en varios intervalos.

Los autores plantean que el método implementado presenta una eficacia del 90 % probado en 500 imágenes de la base de datos FERET. El dispositivo utilizado fue un Xilinx Virtex v1000bg560 trabajando a 52 Mhz, el que contiene 12 248 slices y el diseño ocupó un 99.67 % del dispositivo.

3.2.2. Arquitecturas FPGA para la extracción de rasgos del rostro

La extracción de rasgos es un paso imprescindible dentro del reconocimiento automático de rostros. Se realiza para obtener la información de cada sujeto diferenciando rostros de

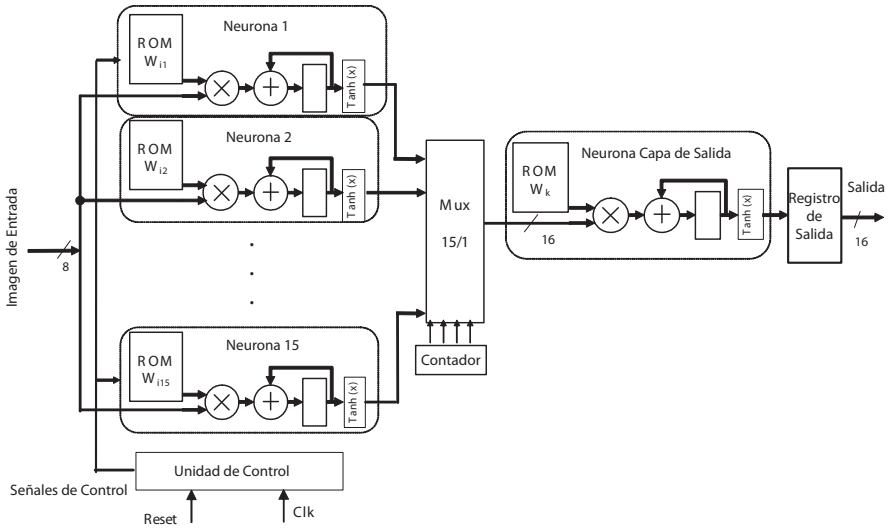

Fig. 10: Arquitectura hardware propuesta por [33]

distintas personas, esta información debe ser lo más robusta ante variaciones geométricas y fotométricas. En la literatura se han reportado varios sistemas que implementan algoritmos de extracción de rasgos sobre FPGA, en estos se proponen nuevos algoritmos y variantes paralelas que demuestran la efectividad del uso de este tipo de dispositivo en sistemas con restricciones de tiempo.

En [25] los autores presentan el diseño de una arquitectura para un sistema de reconocimiento de rostros en línea sobre un chip programable. Esta arquitectura implementa el algoritmo PCA Modular Pesado (WMPCA, del inglés Weighted Modular Principal Component Analysis). Dicho algoritmo es una ampliación del popular método PCA. El WMPCA se basa en la suposición de que las variaciones de expresión facial e iluminación no afectan a todas las regiones del rostro o por lo menos no en igual medida. Con este método el rostro es dividido en regiones horizontales, las cuales son analizadas de modo independiente calculando los rasgos propios para cada región y los que son comparados con sus correspondientes regiones en los rostros de la base de datos. La decisión final del reconocimiento se basa en la suma pesada de los errores calculados de cada una de las regiones. Para el desarrollo de su arquitectura explotan el paralelismo en este algoritmo, principalmente, el más evidente paralelismo de datos sobre las distintas regiones, donde el mismo procesamiento es realizado en las distintas regiones. También es explotado el paralelismo dentro del cálculo de los rasgos propios. La Figura 11 muestra un diagrama de la arquitectura propuesta.

En la figura se observa como la imagen se le pasa a la unidad de recorte, encargada de dividir el rostro en r partes iguales. Cada región pasa a su correspondiente unidad de procesamiento $\mathrm{PE}_{\mathrm{PCA}}$ a través de un bus de datos. Cada una de estas unidades de procesamiento está diseñada para realizar Análisis de Componentes Principales en la región de entrada y generar su correspondiente vector de error. Cada unidad de este tipo con anterioridad carga los vectores propios de la región correspondiente, los rasgos propios de las regiones usadas en la fase de entrenamiento y los pesos a utilizar para calcular el error

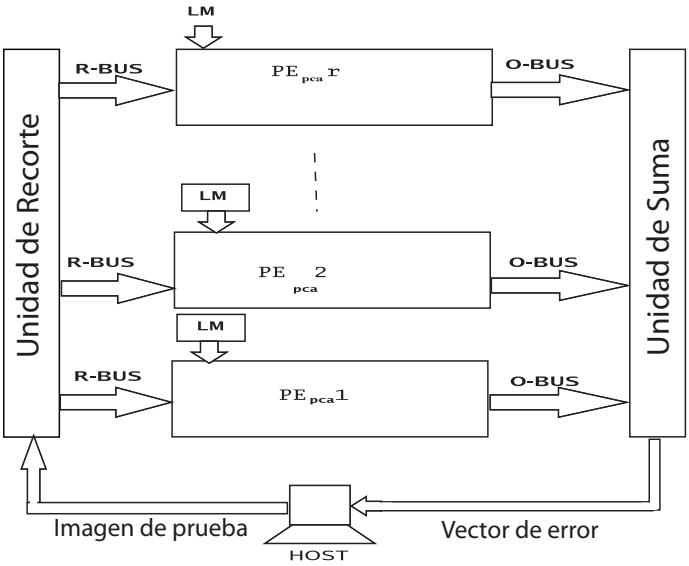

Fig. 11: Arquitectura propuesta en [25] para la implementación del WMPCA

total. La región de entrada es comparada con cada una de las muestras usadas en la fase de entrenamiento y el valor de error se pone en OBUS. El bloque de suma calcula el error total sumando los errores de cada región. El *host* ordena el vector de errores totales y el mínimo de estos valores que se encuentre por debajo del umbral de rechazo es identificado como la mejor correspondencia.

Para las pruebas que le fueron realizadas a este sistema, se llevó a cabo un entrenamiento fuera de línea, donde solo se entrenó con 5 sujetos distintos disponiendo de 3 imágenes de rostro por sujeto debido a las limitaciones de espacio del dispositivo FPGA usado. Según estas pruebas, realizadas sobre un dispositivo Altera de alrededor de 8000 compuertas lógicas a 33.3 Mhz, el sistema es capaz de reconocer un rostro en 38 ms, aunque no se menciona el tamaño de la imagen de rostro.

En [31] se propone un algoritmo de reconocimiento de rostros basado en las wavelets de Daubechies. A este trabajo le precede una implementación hardware presentada en [27] de un algoritmo de detección de rostros basado en clasificación de máximo rechazo y modelos de color de piel, el que es computacionalmente optimizado usando detectores de bordes y redes neuronales. El algoritmo usa PCA para la reducción de dimensionalidad, mientras que las wavelets son usadas como método de preprocesamiento para incrementar el poder discriminante de estos métodos basados en subespacios. La implementación hardware este sistema esta formada por 4 bloques: 1- recepción de la señal de video, 2- cálculo de la Transformada Discreta de Wavelet (DWT) en 2D, 3- cálculo de PCA y 4- clasificación. Para el cálculo de la 2D DWT se propone el diseño presentado en [21]. El análisis de componentes principales es realizado sobre los coeficientes obtenidos de la 2D DWT, usando solamente los coeficientes del primer nivel de descomposición. Este sistema es capaz de reconocer rostros a una velocidad de 12 cuadros por segundo para imágenes de 125 $\times$ 125 píxeles, en un dispositivo trabajando a una frecuencia de reloj de 25 MHz. El sistema completo ocupó menos de 300 mil compuertas de un FPGA Xilinx Spartan IIE.

4. SIFT

En los últimos años el uso de características locales o regiones de interés (más conocido como *local features* en inglés) se ha vuelto muy popular gracias a sus prometedores resultados. Estos han demostrado considerables resultados en una gran variedad de aplicaciones como reconocimiento de objetos [19], recuperación de imágenes [28], navegación de robots [24], construcción de panoramas [5] y también reconocimiento de rostros [3][20][23] [7]. La idea tradicional de estos métodos es primero detectar estructuras o puntos significativos en la imagen y obtener una descripción discriminante de estas estructuras, la cual será utilizada luego para la comparación, a partir de sus alrededores.

Es casi seguro que el método de características locales más popular y ampliamente utilizado es el SIFT (del inglés: *Scale Invariant Features Transform*) propuesto por Lowe [19]. Las características extraídas por el SIFT son en gran medida invariantes a la escala, rotación, cambios de iluminación, ruido y pequeños cambios de pose o perspectiva. Este método ha sido aplicado con éxito en los campos de aplicación mencionados arriba.

Lowe dividió su método en cuatro grandes etapas:

1. Detección de extremos en el espacio-escala.
2. Localización de puntos claves.
3. Asignación de orientación.
4. Descripción de puntos claves.

Algunos trabajos han sido reportados en la literatura con el objetivo de incrementar el rendimiento del SIFT, por ejemplo, mediante el uso de Unidades de Procesamiento Gráfico (GPU) en PCs [32] o simplificando el algoritmo a través de aproximaciones [12][14]. Además de un número reducido de implementaciones en FPGA, las que se comentan a continuación.

En [29] se afirma contar con un sistema que implementa el SIFT en un FPGA para ayudar a la navegación de un robot. Este sistema utiliza 60 ms para procesar una imagen de 640 $\times$ 480, sin embargo no se presentan detalles de la arquitectura utilizada ni una discusión sobre los resultados obtenidos ni las pruebas realizadas al sistema.

En [26] el SIFT es usado para hallar puntos correspondientes entre imágenes de dos cámaras en un sistema de calibración estéreo en línea. Aquí se implementan las partes más costosas del SIFT (DoG y Sobel) para aliviar la carga computacional del sistema. Los autores enfrentan el problema del costo computacional de la convolución usando la propiedad de separabilidad del kernel de convolución Gaussiano y sustituyendo las multiplicaciones de la convolución por el uso de tablas de verdad, un modo eficiente de realizar esto se describe en [1] y [9]. Los autores plantean que su sistema es capaz de procesar imágenes a 60 cuadros por segundo, sin embargo no especifican el tamaño de la imagen. Tampoco se presentan detalles del área del dispositivo ocupada por este sistema.

En [6] los autores presentan un co-diseño hardware/software para el algoritmo SIFT, llevando a hardware las partes de mayor grado de paralelismo. Ellos plantean usar una gran matriz o ventana de desplazamiento para lograr producir todas las escalas al mismo tiempo. Sin embargo no brindan ninguna información acerca del funcionamiento de este método. En este trabajo se mencionan algunas adecuaciones al algoritmo para su funcionamiento en paralelo, pero no se brindan detalles de la arquitectura del sistema. El dispositivo utilizado

fue un FPGA Xilinx Virtex II Pro a 100 MHz donde el sistema logra procesar imágenes de tamaño 320 × 240 en 0.8 ms.

En [4] se propone una arquitectura para la implementación completa del SIFT en un FPGA para su aplicación en la solución del problema de Localización y Mapeo Simultáneo. En este trabajo se presentan las adecuaciones al algoritmo para su implementación en hardware, así como detalles de la misma. El sistema logra procesar imágenes de 320 × 240 píxeles en 33 ms sobre un dispositivo Altera Stratix II.

4.1. Detección de puntos claves candidatos en el SIFT

En la primera etapa propuesta por Lowe, en aras de detectar puntos de interés invariantes a la escala, Lowe propuso usar los extremos del espacio-escala en la función de diferencias de Gaussianas (DoG) convolucionada con la imagen. Para obtener la DoG es necesario realizar varias convoluciones con Gaussianas, lo que representa un alto costo computacional. En esta sección se realiza un estudio de esta etapa del algoritmo.

Para una imagen $I(x, y)$, los puntos claves del SIFT son obtenidos a partir de su espacio-escala Gaussiano, $L(x, y, \sigma)$, el que se obtiene a partir de la convolución de $I(x, y)$ con una Gaussiana de escala variable:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y),$$

donde $G(x, y, \sigma)$ es el filtro o ventana Gaussiana y $*$ es el operador de convolución en x y y. Este espacio-escala se construye generando una serie de imágenes filtradas a valores discretos de σ, donde su dominio es dividido en intervalos logarítmicos organizados en O octavas y cada octava es luego dividida en S subniveles. El valor de σ para una octava o y un subnivel s está dado por:

$$\sigma(o, s) = \sigma_0 2^{o+s/S}, \quad o \in o_{\text{mín}} + [0, ..., O-1], \quad s \in [0, ..., S-1],$$

donde σ_0 es la escala base, por ejemplo $\sigma_0 = 1{,}6$. Luego de calculada cada octava, la imagen pasada hacia la próxima es sub-escalada por un factor de 2.

$Im(x, y)$: Imagen de entrada

```
1  I(x,y) =DoubleSize(Im(x,y));
2  foreach octava o do
       // construir espacio-escala
3      foreach escala s do
4          L(x,y,o,s) = I(x,y) * G(x,y,σ(o,s));
5          I(x,y) = L(x,y,o,s);
6      end
       // obtener espacio-scala DoG
7      foreach escala s, 1 ≤ s ≤ S−1 do
8          D(x,y,o,s) = L(x,y,o,s) − L(x,y,o,s−1);
9      end
10     encontrar extremos locales en el espacio-escala DoG D(x,y,σ);
11     I(x,y) =HalveSize(L(x,y,o,S−2));
12 end
```

Algoritmo 1. Detección de extremos en el espacio-escala DoG

Con el fin de detectar la posición de puntos claves estables en el espacio-escala, Lowe propuso usar los extremos del espacio-escala de diferencias Gaussianas, $D(x, y, o, s)$, calculado a partir de la diferencia de imágenes de escalas adyacentes:

$$D(x, y, o, s) = L(x, y, \sigma(o, s + 1)) - L(x, y, \sigma(o, s)).$$

En aras de detectar los máximos y mínimos locales en $D(x, y, \sigma)$, cada píxel en las imágenes DoG es comparado con sus ocho vecinos en la misma imagen, más los nueve correspondientes vecinos en las escalas adyacentes. Si el valor del píxel es mayor o menor que su 26 vecinos es seleccionado como un punto candidato.

Los pasos propuestos por Lowe para detectar los extremos en el espacio-escala para una imagen $I(x, y)$ son resumidos en el Algoritmo 1.

4.2. Algoritmo paralelo propuesto para la detección de puntos claves candidatos en el SIFT

Habiendo analizado a profundidad el algoritmo SIFT y específicamente el cálculo del espacio de diferencias de Gaussianas, resulta ser que esta tarea tiene un alto grado de paralelismo, resultando su implementación en FPGA muy prometedora.

En esta sección se propone un algoritmo paralelo para la detección de extremos en el espacio-escala de diferencias Gaussianas. El algoritmo propuesto fue diseñado para ser utilizado por una arquitectura hardware con estos propósitos la cual se presenta en la próxima sección. Los detalles del algoritmo propuesto son discutidos a continuación.

Como se puede observar en el Algoritmo 1, la convolución es una operación que es ejecutada en repetidas ocasiones. Esta es una de las operaciones más costosas que comúnmente se usa en el procesamiento digital de imágenes y particularmente en el método SIFT, por lo que constituye un importante aspecto a tratar.

4.2.1. Explotando el paralelismo de datos

Sea I una imagen bidimensional y g una máscara de convolución de tamaño impar $k \times k$, entonces la convolución de I y g está dada por:

$$f(x, y) = \sum_{-i}^{i} \sum_{-j}^{j} I(i, j)g(x - i, y - j), \tag{1}$$

donde $i, j = \lfloor \frac{k}{2} \rfloor$.

En la convolución en 2D un alto potencial para el paralelismo de datos está disponible, específicamente del tipo SPMD (del inglés *Single Process, Multiple Data* que se refiere a un solo proceso, múltiples datos). Como se puede observar en la Ecuación (1), para el cálculo de $f(x_1, y_1)$ solo una vecindad de tamaño $k \times k$ de centro (x_1, y_1) es necesaria.

4.2.2. Explotando la propiedad de separabilidad del kernel Gaussiano

Con el objetivo de disminuir el número de operaciones aritméticas, la propiedades de separabilidad de la Gaussiana es aprovechada.

Un filtro 2D es separable si puede ser dividido en dos señales de 1D: una proyección vertical y otra horizontal. El filtro Gaussiano puede ser separado del siguiente modo:

$$G(x, y, \sigma) = h(x, \sigma) * v(y, \sigma),$$

donde

$$h(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}, \text{and } v(y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-y^2/2\sigma^2}$$

Por lo tanto, la convolución 2D de una imagen con un filtro Gaussiano puede ser llevada a cabo primero convolucionando la imagen con $h(x, \sigma)$ en la dirección vertical y luego con $v(y, \sigma)$ en la dirección vertical o viceversa. La convolución en 1D para obtener un valor de la salida necesita k operaciones MAC (multiplicación - acumulación) en comparación con las k^2 operaciones MAC requeridas por la variante 2D. Por lo tanto, la ventaja computacional de la convolución separable sobre la no-separable es $k^2/2k$.

4.2.3. Entrelazando el procesamiento de octavas

Como se explicó en la Sección 4.1, luego de procesar cada octava, la imagen es subescalada por un factor de dos, tomando todo segundo píxel en cada fila y en cada columna, o sea $I_o(x, y) = I_{o-1}(2x, 2y)$. Luego de escalar una imagen a la mitad, el número total de píxeles es reducido en cuatro. En hardware, para reducir la cantidad de datos, su frecuencia de muestreo es reducida por el mismo factor. Si luego de procesar cada octava la cantidad de datos es reducida por un factor de cuatro, el período de muestro τ de una octava o está dado por

$$\tau(o) = \tau_0 4^o, \tag{2}$$

donde τ_0 es el período de muestreo de la primera octava. Por lo tanto, luego de subescalar, hay un gran porciento de tiempo de procesamiento inactivo $\hat{i}$ con respecto al período de procesamiento de la primera octava. Este se define como

$$\hat{i} = \frac{\tau(o) - 1}{\tau(o)}.$$

Aprovechar estos intervalos de inactividad hace posible procesar las O octavas de una escala en un solo bloque de convolución. Esto se logra entrelazando el procesamiento de las O convoluciones de modo tal que para todas las octavas en un tiempo t dado, el número de elementos procesados $p(o, t)$ cumpla que

$$p(o, t) = \left\lfloor \frac{t + \varepsilon(o)}{\tau(o)} \right\rfloor, \tag{3}$$

donde $\varepsilon(o)$ es la latencia de la octava o en la línea de entrelazado.

Para entrelazar el procesamiento de las O octavas se asume que el período de muestro de la primera octava es mayor o igual a dos ciclos de reloj. En caso contrario, $O-1$ octavas son entrelazadas y τ_0 sería el período de muestreo de la segunda octava.

4.3. Arquitectura hardware propuesta

En las arquitecturas propuestas en [26] y [4], se usa un bloque de convolución para cada operación de este tipo que se realiza; dividiendo por octavas el procesamiento por lo que se necesitan $O \cdot S$ bloques de convolución. En la arquitectura aquí presentada, proponemos usar solamente S bloques de convolución para realizar las $O \cdot S$ convoluciones, dividiendo el procesamiento en escalas y manteniendo el mismo rendimiento del sistema.

Un diagrama de la arquitectura del sistema se muestra en la Figura 12 a). Este diagrama muestra un sistema de cuatro octavas, cinco escalas y un kernel de convolución de siete coeficientes ($O = 4, S = 5, k = 7$). Esta arquitectura puede ser generalizada para cualquier configuración de estos parámetros.

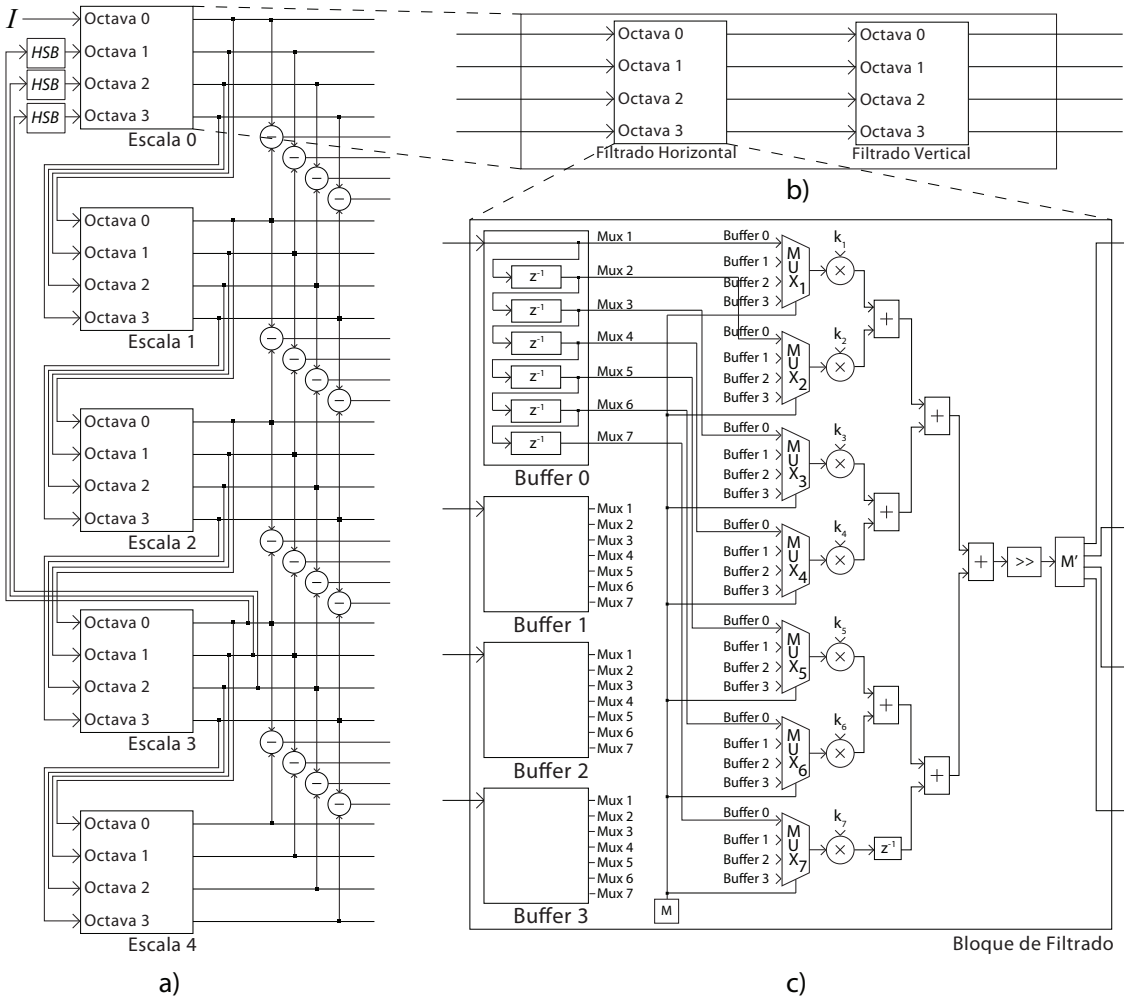

Fig. 12: Arquitectura tunelizada propuesta.

Esta arquitectura está formada, en mayor medida, por bloques de cálculo de escala (SCB). Un bloque SCB realiza las O operaciones de filtrado Gaussiano para una escala dada según se describe en la Sección 4.2.3. Por lo tanto, cada SCB tiene O puertos de

entrada y O de salida, uno para cada octava respectivamente, donde el período de muestreo para cada octava está definido por la ecuación (2).

Los bloques SCB están conectados en cascada con el fin de usar un kernel de convolución de tamaño fijo y así evitar las convoluciones con kernels de gran tamaño. Esto se puede observar en la Figura 12 a).

Un bloque SCB, para realizar el filtrado Gaussiano, aprovecha la propiedad de separabilidad del kernel Gaussiano como se describe en la Sección 4.2.2. Aprovechando dicha propiedad, este bloque realiza dicho filtrado primero en la dirección horizontal y luego en la vertical lo que se puede observar en la Figura 12 b).

La organización interna del bloque de filtrado horizontal se detalla en la Figura 12 c). Cada señal de entrada es desplazada a través de k registros, donde k es el número de coeficientes del kernel de convolución. Las k señales correspondientes a las O octavas son multiplexadas con el objetivo de controlar el orden de procesamiento de las octavas y lograr el entrelazado del mismo. La lógica de los multiplexores en una instante t está determinada por el bloque M, el que implementa la función $m(t)$ y cumple la condición establecida en (3). El orden de entrelazado se define a continuación:

$$m(t) = \begin{cases} o_0 & \text{if } t \equiv \varepsilon(o_0) \mod \tau(o_0) \\ o_1 & \text{if } t \equiv \varepsilon(o_1) \mod \tau(o_1) \\ \vdots & \vdots \\ o_{O-1} & \text{if } t \equiv \varepsilon(o_{O-1}) \mod \tau(o_{O-1}). \end{cases}$$

La estructura del bloque de filtrado vertical es la misma que la del horizontal, con la diferencia de que cada buffer almacena las k últimas líneas de la imagen en vez de los k últimos píxeles.

Entrelazando el procesamiento de las octavas en un solo procesador de convolución es posible ahorrar gran cantidad de área del dispositivo y la consecuente reducción de consumo de corriente. Además, con el fin de evitar trabajar con valores de punto fijo, los coeficientes del kernel de convolución son multiplicados por una constante determinada. Luego, los resultados filtrados son normalizados dividiéndolos por esta misma constante. Preferiblemente esta constante debe ser una potencia de dos y así poder reemplazar la operación de división por un simple corrimiento.

El bloque HSB en la Figura 12 a) realiza el subescalamiento de la imagen. Con este fin, se usa un registro de desplazamiento direccionable y un contador.

4.4. Implementación en FPGA y resultados experimentales

4.4.1. Características de la implementación

Un sistema con $O = 4$, $S = 6$ y $k = 7$ para procesar imágenes de 320×240 se implementó en un FPGA Xilinx Virtex II Pro (XC2VP30-5FF1152). Este sistema se implementó usando System Generator + Simulink. Los recursos estimados usados por esta implementación y su comparación con el sistema de Bonato et al. [4] se muestran en la Tabla 1.

Como fue discutido en secciones anteriores, el sistema devuelve un resultado cada dos ciclos de reloj. Bajo esta implementación, con una restricción de reloj de 50 MHz, el sistema

Tabla 1: Características de la implementación y comparación con [4].

Recursos	Nuestro sistema $O = 4$, $S = 6$ and $k = 7$	DoG en [4] $O = 3$, $S = 6$ and $k = 7$
Slices	5068	-
Flip-flops	6028	7256
Look Up Tables	6880	15137
Blocks RAM	120 (2.1 Mb)	0.91Mb

se toma 3 ms para detectar extremos en el espacio de la escala para una imagen de 320×240, por lo tanto es posible procesar 330 cuadros por segundo. Este resultado fue comparado, en términos de eficiencia, con la implementación software de Vedaldi [35] corriendo en una PC (1.8 GHz Core Duo y 1 GB RAM). Nuestro sistema mostró una significativa aceleración de 250x.

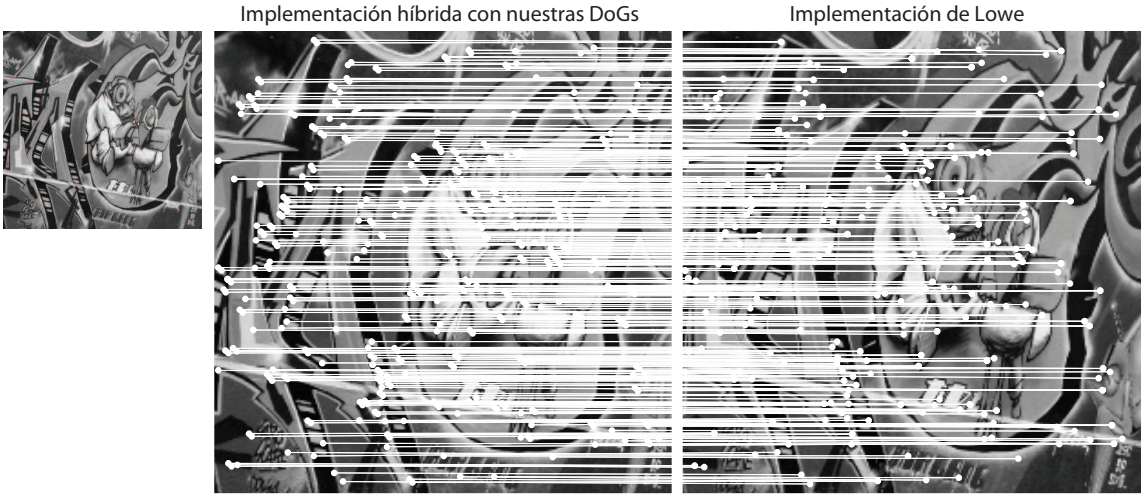

Fig. 13: Correspondencias entre la implementación híbrida (usando los resultados de nuestro sistema) y la implementación de Lowe para una imagen de ejemplo.

4.4.2. Confiabilidad del sistema

Con el fin de probar la confiabilidad de nuestra implementación, se chequearon las correspondencias entre los puntos detectados por una implementación software y una implementación híbrida donde los extremos en el espacio de la escala fueron obtenidos por nuestro sistema. La implementación software del SIFT utilizada fue la de Lowe [19]. La implementación híbrida fue creada a partir de la implementación software de Vedaldi, donde la primera etapa del SIFT fue ejecutada por nuestro sistema. Estas pruebas se realizaron en 26 imágenes. Las principales diferencias en los resultados obtenidos por nuestra implementación híbrida se deben a las aproximaciones en el cálculo de la DoG. Sin embargo estas aproximaciones no afectaron en gran medida en resultado final. Los errores medios en coordenadas, escala y orientación en los puntos detectados se muestran en la Tabla 2.

La Figura 13 muestra un ejemplo de los puntos detectados por las dos implementaciones y sus correspondencias. Los errores para esta imagen se muestran en la Figura 14.

Tabla 2: Error medio en coordenadas, escala y orientación en los puntos detectados.

Parámetro	Error Medio
Δx	1.127
Δy	1.441
$\Delta \sigma$	0.149
$\Delta \theta$	0.047

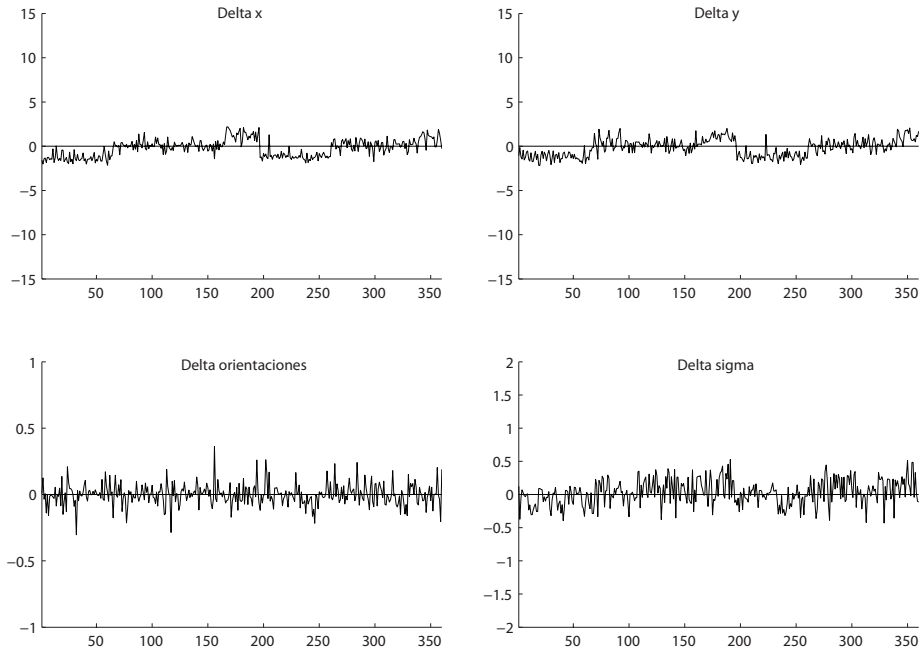

Fig. 14: Errores en coordenadas, escala y orientación en los puntos detectados en una imagen de ejemplo.

5. Conclusiones

En este trabajo se realizó un estudio del estado del arte del empleo de los FPGAs en la solución de problemas de reconocimiento de rostros. En el mismo se muestra la eficiencia del uso de está tecnología en la solución de problemas de esta rama, principalmente en la aceleración computacional de este costoso proceso.

También se analizaron algunos conceptos básicos del procesamiento paralelo, mostrándose su fuerte relación con el diseño de arquitecturas para FPGAs.

Se estudió el algoritmo SIFT debido a su sólida presencia en el estado del arte de distintos campos de aplicación del reconocimiento de patrones, incluyendo el reconocimiento de rostros. Se realizó un análisis de este algoritmo, específicamente la detección de extremos en el espacio-escala, uno de los procesos más costosos desde el punto de vista computacional. Se propuso un algoritmo que explotase el paralelismo presente en esta parte del algoritmo, así como una arquitectura hardware que lo implementase.

Como trabajo futuro continuaremos analizando el algoritmo SIFT y las posibilidades de su diseño e implementación sobre una arquitectura de hardware reconfigurable, extendiendo la investigación al diseño de una arquitectura eficiente para todo el algoritmo SIFT, con la que se esperan aceleraciones superiores a 10x.

Referencias bibliográficas

1. Atmel. 3x3 convolver with run-time reconfigurable vector multiplier in atmel at6000 fpgas. *[Online]. Available: http://atmel.com/dyn/resources/prod documents/DOC0764.PDF*, 1999.
2. Luca Benini and Giovanni De Micheli. Networks on chips: A new soc paradigm. *IEEE Computer*, 35(1):70–78, 2002.
3. Manuele Bicego, Andrea Lagorio, Enrico Grosso, and Massimo Tistarelli. On the use of sift features for face authentication. In *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, page 35, Washington, DC, USA, 2006. IEEE Computer Society.
4. V. Bonato, E. Marques, and G. A. Constantinides. A parallel hardware architecture for scale and rotation invariant feature detection. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(12):1703–1712, 2008.
5. Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vision*, 74(1):59–73, 2007.
6. H. Djakou Chati, F. Muhlbauer, T. Braun, C. Bobda, and K. Berns. Hardware/software co-design of a key point detector on fpga. In *FCCM '07: Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 355–356, Washington, DC, USA, 2007. IEEE Computer Society.
7. Claudia Cruz, Luis Enrique Sucar, and Eduardo F. Morales. Real-time face recognition for human-robot interaction. In *FG*, pages 1–6. IEEE, 2008.
8. André DeHon. The density advantage of configurable computing. *Computer*, 33(4):41–49, 2000.
9. J. Evans. Efficient fir filter architectures suitable for fpga implementation. Online, 1994.
10. W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(9):891–906, 1991.
11. Alex A. Freitas and S. H. Lavington. *Mining Very Large Databases with Parallel Processing*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.
12. Michael Grabner, Helmut Grabner, and Horst Bischof. Fast approximated sift. In P. J. Narayanan, Shree K. Nayar, and Heung-Yeung Shum, editors, *ACCV (1)*, volume 3851 of *Lecture Notes in Computer Science*, pages 918–927. Springer, 2006.
13. Anil K. Jain and Stan Z. Li. *Handbook of Face Recognition*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
14. Yan Ke and R. Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 506–513, 2004.
15. Hung-Chih Lai, Radu Marculescu, Marios Savvides, and Tsuhan Chen. Communication-aware face detection using noc architecture. In Antonios Gasteratos, Markus Vincze, and John K. Tsotsos, editors, *ICVS*, volume 5008 of *Lecture Notes in Computer Science*, pages 181–189. Springer-Verlag, 2008.
16. Hung-Chih Lai, M Savvides, and Tsuhan Chen. Proposed fpga hardware architecture for high frame rate (¿¿100 fps) face detection using feature cascade classifiers. In *Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007. First IEEE International Conference on*, pages 1–6, 2007.
17. David M. Lewis, David R. Galloway, Marcus van Ierssel, Jonathan Rose, and Paul Chow. The transmogrifier-2: A 1 million gate rapid prototyping system. *Field-Programmable Gate Arrays, International ACM Symposium on*, 0:53–61, 1997.
18. Xiaodong Liu and Guangda Su. A cluster-based parallel face recognition system. In Hamid R. Arabnia, editor, *PDPTA*, pages 737–743. CSREA Press, 2006.
19. David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
20. Jun Luo Ma, Y. Takikawa, E. Lao, S. Kawade, and M. Bao-Liang Lu. Person-specific sift features for face recognition. In *Acoustics, Speech and Signal Processing, ICASSP 2007. IEEE International Conference on*, 2007.
21. P. McCanny and J. McCanny. Design and implementation of an efficient multi-resolution two dimensional dwt architecture. In *3rd Intl. CSNDSP Conference*, 2002.
22. Rob McCready. Real-time face detection on a configurable hardware system. In *FPL '00: Proceedings of the The Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications*, pages 157–162, London, UK, 2000. Springer-Verlag.

23. Ajmal Mian, Mohammed Bennamoun, and Robyn Owens. An efficient multimodal 2d-3d hybrid approach to automatic face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1927–1943, 2007.
24. A. C. Murillo, José Jesús Guerrero, and Carlos Sagüés. Surf features for efficient robot localization with omnidirectional images. In *ICRA*, pages 3901–3907, 2007.
25. A. Pavan Kumar, V. Kamakoti, and Sukhendu Das. System-on-programmable-chip implementation for on-line face recognition. *Pattern Recogn. Lett.*, 28(3):342–349, 2007.
26. Niklas Pettersson and Lars Petersson. Online stereo calibration using fpgas. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 55–60, 2005.
27. M. S. Sadri, N. Shams, M. Rahmaty, I. Hosseini, R. Changiz, S. Mortazavian, S. Kheradmand, and R. Jafari. An fpga based fast face detector. In *Global Signal Processing Expo and Conference*, 2004.
28. Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(5):530–535, 1997.
29. Stephen Se, Ho kong Ng, Piotr Jasiobedzki, and Tai jing Moyung. Vision based modeling and localization for planetary exploration rovers. In *55th International Astronautical Congress 2004*, 2004.
30. Ángel Serrano, Isaac Martín de Diego, Cristina Conde, Enrique Cabello, LinLin Shen, and Li Bai. Influence of wavelet frequency and orientation in an svm-based parallel gabor pca face verification system. In Hujun Yin, Peter Tiño, Emilio Corchado, William Byrne, and Xin Yao, editors, *IDEAL*, volume 4881 of *Lecture Notes in Computer Science*, pages 219–228. Springer, 2007.
31. Nasim Shams, Iraj Hosseini, Mohammad S. Sadri, and Ehsan Azarnasab. Low cost fpga-based highly accurate face recognition system using combined wavelets with subspace methods. In *ICIP*, pages 2077–2080. IEEE, 2006.
32. Sudipta Sinha, Jan-Michael Frahm, Marc Pollefeys, and Yakup Genc. Feature tracking and matching in video using programmable graphics hardware. *Machine Vision and Applications*.
33. F. Smach, M. Atri, Johel Mitéran, and M. Abid. Design of a neural networks classifier for face detection. In Cemal Ardil, editor, *WEC (5)*, pages 274–277. Enformatika, Çanakkale, Turkey, 2005.
34. T. Theocharides, N. Vijaykrishnan, and M. J. Irwin. A parallel architecture for hardware face detection. In *ISVLSI '06: Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, page 452, Washington, DC, USA, 2006. IEEE Computer Society.
35. A. Vedaldi. An open implementation of the SIFT detector and descriptor. Technical Report 070012, UCLA CSD, 2007.
36. Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.
37. Huiyuan Wang and Xiaojuan Wu. Eigenblock approach for face recognition. *International Journal of Computational Intelligence Research*, 3(1):72–77, 2007.
38. Ming Yang, Ying Wu, Jim E. Crenshaw, Bruce Augustine, and Russell Mareachen. Face detection for automatic exposure control in handheld camera. In *ICVS*, page 17. IEEE Computer Society, 2006.
39. W. Zhao, R. Chellappa, A. Rosenfeld, and P. J. Phillips. Face recognition: A literature survey. *ACM Computing Surveys*, pages 399–458, 2003.

RT_013, octubre 2010
Aprobado por el Consejo Científico CENATAV
Derechos Reservados © CENATAV 2010
Editor: Lic. Lucía González Bayona
Diseño de Portada: Di. Alejandro Pérez Abraham
RNPS No. 2142
ISSN 2072-6287
Indicaciones para los Autores:
Seguir la plantilla que aparece en www.cenatav.co.cu
C E N A T A V
7ma. No. 21812 e/218 y 222, Rpto. Siboney, Playa;
Ciudad de La Habana. Cuba. C.P. 12200
Impreso en Cuba