Universidad Autónoma de Baja California

Facultad de Ciencias

IMPLEMENTACIÓN PARALELA DE LA CORRELACIÓN MORFOLÓGICA MEDIANTE LA GPU: RECONOCIMIENTO FACIAL COMO CASO DE ESTUDIO

TESIS

que presenta para obtener el grado de MAESTRO EN CIENCIAS

Sixto Lázaro Martínez

DIRECTOR DE TESIS:

DR. JOSÉ ÁNGEL GONZÁLEZ FRAGA

Ensenada, Baja California Enero del 2014

Universidad Autónoma de Baja California

Facultad de Ciencias

IMPLEMENTACIÓN PARALELA DE LA CORRELACIÓN MORFOLÓGICA MEDIANTE LA GPU: RECONOCIMIENTO FACIAL COMO CASO DE ESTUDIO

TESIS

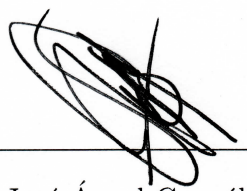
que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN CIENCIAS

Presenta

Sixto Lázaro Martínez

Aprobada por:



Dr. José Ángel González Fraga

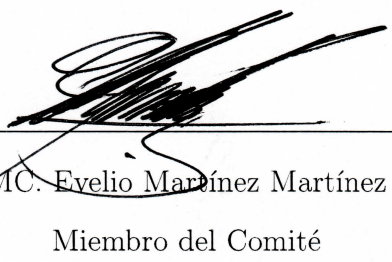
Director de Tesis

MC. Sergio Omar Infante Prieto

Miembro del Comité

Dr. Juan José Tapia Armenta

Miembro del Comité



MC. Evelio Martínez Martínez

Miembro del Comité

Ensenada Baja California, México. Enero del 2014.

RESUMEN de la tesis de **Sixto Lázaro Martínez**, presentada como requisito parcial para la obtención del grado de MAESTRO EN CIENCIAS. Ensenada Baja California, México, Enero del 2014.

IMPLEMENTACIÓN PARALELA DE LA CORRELACIÓN MORFOLÓGICA MEDIANTE LA GPU: RECONOCIMIENTO FACIAL COMO CASO DE ESTUDIO

Resumen aprobado por:

Dr. José Ángel González Fraga

Director de Tesis

En la actualidad, el tiempo de ejecución de muchos algoritmos puede ser mejorado si se emplean sus adecuadas versiones paralelas. Los algoritmos de detección y reconocimiento facial, basados en correlación, son buenos candidatos para implementarse en paralelo, debido al número de operaciones independientes que son aplicadas sobre cada píxel en las imágenes.

En este trabajo, se presenta un análisis comparativo sobre los tiempos que consumen las versiones secuencial y paralela de los procesos de correlación morfológica y el de construcción de un filtro compuesto. Las versiones paralelas se implementaron a través de la plataforma CUDA, con unidad de procesamiento gráfico GeForce 590.

Para el proceso de construcción del filtro compuesto, se consideraron tres tamaños de las imágenes de entrenamiento: 180×180, 360×360 y 640×640 píxeles. Además, se varió la cantidad de imágenes de entrenamiento que se incorporaron al diseño del filtro,

se empezó por emplear 2 y se llegaron a usar hasta 200 imágenes. Se alcanzó una tasa de aceleración de hasta 23.44 para imágenes de 180×180 píxeles, 25.85 para imágenes de 360×360 píxeles y 26.05 para imágenes de 640×640 píxeles, presentando un ahorro de tiempo por encima del 90 %.

Para el proceso de correlación morfológica, se realizaron varios experimentos aplicados a la detección y reconocimiento facial en imágenes en escala de gris, que varían en tamaño de 4×4 a 4096×4096 píxeles. El análisis realizado, muestra que la mejora de los tiempos de ejecución puede alcanzar una tasa de aceleración de 29.79 y un ahorro de tiempo de hasta 96 % para una imagen de escena de 4096×4096 y un filtro del mismo tamaño.

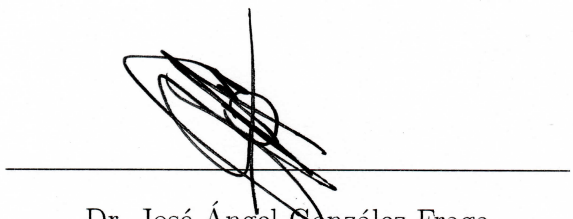
Como puede observarse, los resultados muestran un claro ahorro en el tiempo de ejecución al emplear los algoritmos paralelos estudiados.

Palabras Clave: *Correlación Morfológica, Filtros Compuestos de Correlación, CUDA, GPU, Tasa de Aceleración, Procesamiento Facial.*

ABSTRACT of Master of Science´s Thesis, presented by **Sixto Lázaro Martínez**, in order to obtain the MASTER of SCIENCE. Ensenada Baja California, México. January, 2014.

PARALLEL IMPLEMENTATION OF MORPHOLOGICAL CORRELATION THROUGH GPU: FACIAL RECOGNITION AS A CASE STUDY

Approved by:



Dr. José Ángel González Fraga

Chair of Advisory Committee

Nowadays, the execution time of many algorithms might be improved by using an appropriate parallel version. Face detection and recognition algorithms, based on correlation, are well situated to take advantage of parallel implementation, this is because of the large amount of independent operations required to apply to each pixel while processing images.

In this study, we present a comparative analysis of the execution time between the sequential and parallel versions of the processes of morphological correlation and construction of a composite filter. Parallel versions were implemented through the CUDA platform, which carries a GeForce 590 graphics processing unit.

For the construction process of the composite filter, three different sizes of the training images were considered: 180×180, 360×360 and 640×640 pixels. In addition, the number of training images incorporated for the design of the filter varied, ranging

from two images to up to 200 images. We achieved an acceleration rate of 23.44 for images of 180 $\times$ 180 pixels, 25.85 for images of 360 $\times$ 360 pixels and 26.05 for images of 640 $\times$ 640 pixels, obtaining a reduction in the execution time above 90 %.

Regarding the morphologic correlation process, several experiments applied to face detection and recognition in gray scale images were conducted, varying in size from 4 $\times$ 4 to 4096 $\times$ 4096 pixels. The analysis conducted, shows that the improvement of the execution time can reach an acceleration rate of 29.79 and a saving of running time of up to 96 % for 4096 $\times$ 4096 scene images and a filter of similar size. As can be seen, our results show a clear savings of the execution time while using the studied parallel algorithms.

Keywords: *Morphological Correlation, Composite Correlation Filters, CUDA, GPU, speedup, Facial Processing.*

Dedicada

A mis padres, por su apoyo incondicional.

Agradecimientos

A mi director de tesis Dr. José Ángel Gonzales Fraga, por su ayuda, consejos, tiempo y el gran apoyo brindado para realizar este trabajo de investigación.

A los miembros de mi comité de tesis Dr. Juan José Tapia Armenta, MC. Sergio Omar Infante Prieto y MC. Evelio Martínez Martínez, cuyos comentarios y consejos contribuyeron al desarrollo de este trabajo de investigación.

Al Dr. José Ignacio Ascencio López por su amistad y todo el apoyo brindando durante el inicio de mis estudios de maestría.

A la Facultad de Ciencias de la UABC y al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico y todos los recursos brindados durantes mis estudios de maestría.

A mis amigos y compañeros del MYDCI.

Índice general

Índice de figuras

Índice de tablas

Lista de algoritmos

Capítulo 1

Introducción

El reconocimiento de patrones es un área de la computación que ha despertado un gran interés debido a que tiene una amplia gama de aplicaciones, por ejemplo la detección y el reconocimiento facial. En este proyecto de investigación, se presentan dos temáticas de la computación: el procesamiento facial y la paralelización de los algoritmos del procesamiento facial a través de la unidad de procesamiento gráfico (GPU por sus siglas en inglés).

En la primera temática, el procesamiento facial, se involucra la detección para determinar la presencia de un rostro y el reconocimiento, para la identificación de un individuo particular en una escena.

La segunda temática tiene que ver con el uso de conceptos de programación paralela, a través del uso de la computación de propósito general en unidades de procesamiento gráfico, para acelerar (en términos de tiempo de ejecución) los algoritmos de procesamiento facial que se presentan. Esta segunda temática involucra también el uso de una métrica conocida como tasa de aceleración para medir el desempeño, en término de los tiempos de ejecución presentados por estos algoritmos frente a sus versiones secuenciales.

1.1. Definición del problema

El proceso de la correlación morfológica, también conocida como correlación no lineal, presenta una gran complejidad computacional en términos de tiempo de ejecución, debido a la gran cantidad de operaciones que se desarrollan por cada píxel en una imagen de prueba. Si la imagen de prueba es de tamaño $N \times N$ y el filtro de $M \times M$ la cantidad máxima de operaciones que se espera es de $N^2 \times M^2$, por lo que puede ser reducido de manera significativa a M^2 por hilo mediante una implementación paralela utilizando la GPU.

1.2. Justificación

El proceso de la correlación morfológica involucra una escena de prueba y un filtro de correlación, el cual es centrado en una posición inicial (x, y) de la escena de prueba y a continuación se realiza una operación conocida como suma de los mínimos entre los píxeles correspondientes de la imagen de prueba y el filtro. Esta suma da como resultado un valor que es almacenado en la posición centrada actual del plano de correlación de salida. Esta operación es realizada en cada píxel de la imagen de escena mediante el desplazamiento entre sus columnas y filas, por lo que el tiempo de ejecución de la versión secuencial de este algoritmo puede ser mejorado si se paralelizan estas operaciones utilizando una GPU.

En adición, se requiere la construcción de un filtro compuesto SDF no lineal (NSDF), que permite la detección de un objetivo en la escena de prueba. Este filtro involucra un conjunto con N imágenes de entrenamiento, en donde cada imagen de entrenamiento se separa en 256 capas binarias por medio de una descomposición por umbral y entonces se localizan las intersecciones de los valores de los píxeles de cada una de estas capas tanto de la imagen de entrenamiento actual como en las imágenes subsecuentes. Este algoritmo también presenta un tiempo de ejecución viable a ser mejorado.

1.3. Objetivos

Se propuso alcanzar el objetivo general y los objetivos específicos siguientes.

1.3.1. Objetivo general

Comparar los tiempos de ejecución de las implementaciones secuencial y paralela, implementados a través de la GPU, de la correlación morfológica aplicada al problema del reconocimiento facial.

1.3.2. Objetivos específicos

1. Implementar las versiones, secuencial y paralela de la operación de correlación morfológica.

2. Realizar un análisis comparativo del desempeño, en términos de la métrica de aceleración, entre las versiones secuencial y paralela de la operación de correlación morfológica.

3. Paralelizar el algoritmo de síntesis del filtro NSDF.

4. Realizar un análisis comparativo del desempeño, en términos de la métrica de aceleración, entre las versiones secuencial y paralela del filtro de correlación NSDF.

1.4. Trabajos previos relevantes

Actualmente existen investigaciones en el área de detección y reconocimiento facial que están enfocados al aprovechamiento de la arquitectura provista por NVIDIA, algunas de las últimas implementaciones realizadas a lo largo de la última década son las siguientes:

Viola and Jones (2004), presentan un sistema de reconocimiento facial frontal, que consigue una detección y tasas de falsos positivos equivalentes a los mejores resultados publicados en ese año, una de sus mejores características es que consigue detectar rostros de una forma extremadamente rápida, fue probado en imágenes de 384×288 píxeles, las caras son detectadas a una velocidad de 15 marcos por segundo en un procesador convencional Intel Pentium III a 700 MHz. Introducen una nueva representación de imágenes a la que denominan imagen integral, un método para la construcción de un clasificador utilizando *AdaBoost* mediante la selección de un número pequeño de características y por ultimo un método para combinar clasificadores sencillos en uno más complejo, utilizando una estructura de cascada que permite la aceleración de la velocidad del detector.

Park et al. (2009) presentan un algoritmo en tiempo real que estima la pose que tiene un rostro semi-oculto en imágenes 3D. Basado en una nueva forma característica para identificar la nariz (de personas) en imágenes. El método consiste en generar ciertos candidatos para las posiciones y entonces generaba y evaluaba algunas poses como hipótesis (de posicionamiento), en paralelo utilizando la GPU. Desarrollaron una nueva función de error que compara las imágenes de entrada con imágenes de poses precalculadas de una cara humana promedio. El algoritmo es muy robusto a variaciones de poses y rotaciones, caras parcialmente ocultas y trabaja para múltiples caras desde un punto de vista. El algoritmo esta implementado en una tarjeta NVIDIA GeForce 8800 GTX, utilizando CUDA para explotar las capacidades de la GPU como un dispositivo de computo paralelo. La GPU cuenta con 16 núcleos, donde cada uno puede ejecutar uno o más bloques con más de 512 hilos cada uno.

Hefenbrock et al. (2010) nos presentan una implementación del algoritmo Viola-Jones utilizando GPUs, comparándolo con la mejor implementación hasta entonces, realizado sobre circuitos FPGA (field-programmable gate array). En cuanto a rendimiento notaron que utilizando 4 GPUs la detección se realiza a 15.2 FPS, valor bas-

tante cercano al mejor rendimiento de la FPGA, en el que se realiza a 16 FPS. Otras diferencias que mencionan son la complejidad de programación, la cual es mucho más sencilla realizarlo en la GPU y el precio de los dispositivos que son muy variables, pero que las tarjetas de video de baja gama son mucho más accesibles que las FPGA y en cuanto a consumo de energía la GPU consume mucho más.

Devrari and Kumar (2011) proponen una alternativa a Viola-Jones, que es bastante bueno para imágenes de media y baja resolución, la alternativa de reconocimiento de imágenes utiliza características haar-like, la imagen integral, y como algoritmo de entrenamiento y clasificación la Máquina de Vector de Soporte (SVM), aprovechan la arquitectura CUDA para la detección rápida de rostros en imágenes de gran resolución.

1.5. Organización de la tesis

El resto de este documento está dividido en 5 capítulos y 2 apéndices, el contenido de cada uno de ellos se describe de forma breve a continuación.

En el capítulo 2 se presentan los fundamentos teóricos en los que se basa este trabajo de investigación, se abordan los sistemas lineales y no lineales, el reconocimiento basado en correlación, así como los filtros basados en correlación más conocidos, los filtros no lineales compuestos y su proceso de filtrado y los dos tipos de reconocimiento que se aplican entre las clases de objetos.

En el capítulo 3 se presentan la introducción al reconocimiento de patrones y su aplicación en el procesamiento facial, así como algunas aplicaciones dentro de esta, como la detección, reconocimiento y seguimiento facial y como los filtros de correlación son aplicados a estas tareas además de las ventajas que ofrecen.

El capítulo 4 muestra una introducción a CUDA, la plataforma presentada por NVIDIA para el cómputo paralelo utilizando unidades de procesamiento gráfico y se muestra lo que es la tasa de aceleración, la métrica utilizada para conocer el ahorro

en el tiempo de ejecución de los algoritmos ejecutados de forma paralela y comparados contra sus versiones secuenciales.

En el capítulo 5 se muestra la implementación de la correlación morfológica y el filtro compuesto NSDF tanto secuencial como de forma paralela utilizando la unidad de procesamiento gráfico, así como la descripción de los experimentos realizados y los resultados obtenidos al medir sus tiempos de ejecución.

En el capítulo 6 se presentan los resultados obtenidos al realizar los experimentos de estos algoritmos sobre utilizando imágenes en escala de gris y midiendo las tasas de aceleración y ahorro de tiempo.

Y finalmente en el capítulo 7 se presentan las conclusiones y reflexiones derivadas de este trabajo, así como las áreas de oportunidad identificadas durante su desarrollo.

Capítulo 2

Fundamentos teóricos

En el procesamiento de imágenes, una imagen puede ser definida como una función bidimensional $f(x, y)$, en donde x y y representan coordenadas espaciales, y la amplitud de f recibe el nombre de *intensidad* o *nivel de gris* de la imagen en el punto (x, y). Si los valores de x, y y f son finitos, es decir valores discretos, se le denomina *imagen digital* (Gonzalez et al., 2009).

Una imagen digital está conformada por un número finito de elementos, cada uno con una ubicación y un valor particular. Una imagen digital es comúnmente representada como un arreglo matricial de valores numéricos, en donde cada valor discreto de este arreglo matricial representa una región de la imagen continúa y es conocido como píxel.

Los valores de los píxeles son obtenidos mediante muestreo (digitalización de las coordenadas) y cuantificación (digitalizar los valores de la amplitud) de una imagen continua.

Para procesar estos arreglos de píxeles es necesario utilizar diferentes herramientas matemáticas derivadas de la teoría de sistemas lineales, sistemas no lineales y estadística. En este capítulo se presenta la teoría básica en la que se basa el desarrollo de esta tesis.

2.1. Sistemas lineales

Dentro del tratamiento digital de señales, los sistemas se clasifican de acuerdo a una serie de categorías o propiedades características tales como la linealidad, causalidad, estabilidad e invariancia en el tiempo; dentro de esta clasificación se encuentra los sistemas lineales que son aquellos que satisfacen el *principio de superposición* (Proakis and Manolakis, 2012).

Este principio exige que la respuesta de un sistema a una suma ponderada se señales sea igual a la correspondiente suma ponderada de las respuestas del sistema a cada una de las señales individuales de entrada. Por lo que se tiene la siguiente definición de linealidad, un sistema $\mathscr{T}$ es lineal si y solo si:

$$\mathscr{T}[a_1x_1(n) + a_2x_2(n)] = a_1\mathscr{T}[x_1(n)] + a_2\mathscr{T}[x_2(n)] \tag{2.1}$$

para cualquier secuencia arbitraria $x_1(n)$ y $x_2(n)$ y cualquier constante arbitraria a_1 y a_2.

Las dos propiedades que constituyen el principio de superposición son

$$\mathscr{T}[a_1x_1(n)] = a_1\mathscr{T}[x_1(n)] = a_1y_1(n) \tag{2.2}$$

si suponemos que $a_2 = 0$, entonces la ecuación (2.1) se reduce a la ecuación (2.2), en donde $y_1(n) = \mathscr{T}[x_1(n)]$, esto se conoce como propiedad *multiplicativa*, la segunda propiedad es la *aditiva* y es cuando $a_2 = a_1 = 1$, su relación se muestra en la siguiente ecuación

$$\mathscr{T}[x_1(n) + x_2(n)] = \mathscr{T}[x_1(n)] + \mathscr{T}[x_2(n)] = y_1(n) + y_2(n) \tag{2.3}$$

Se dice que un sistema lineal está en reposo, si con una entrada cero produce una salida cero. Si un sistema genera una salida distinta de cero, cuando se aplica una entrada cero, entonces se dice que el sistema no está en reposo o es un sistema no

lineal. Si un sistema en reposo no satisface el principio de superposición entonces es un sistema no lineal.

Una vez expuesto lo que es un sistema lineal, vamos a enfocarnos en la categoría de los sistemas lineales invariantes en el tiempo (sistemas LTI). Estos sistemas se caracterizan en el dominio del tiempo mediante su respuesta a una serie de impulsos unitarios. Cualquier señal de entrada arbitraria se puede descomponer y representar como una suma ponderada de secuencias de impulsos unitarios. La respuesta del sistema a cualquier señal de entrada arbitraria se puede expresar en función de la respuesta del sistema al impulso unitario.

La forma general de la expresión que relaciona la respuesta al impulso unitario del sistema con las señales de entrada y salida es conocida como convolución, dada por la ecuación (2.4), la ecuación de la convolución nos proporciona un medio para calcular la respuesta de un sistema LTI en estado de reposo a cualquier señal de entrada $x(n)$:

$$y(n) = x(n) * h(n) \equiv \sum_{k=-\infty}^{\infty} x(k)h(n-k) \tag{2.4}$$

en donde la secuencia que sigue la respuesta al impulso $h(n)$ se refleja y se desplaza, $x(n)$ es la entrada al sistema. La convolución tiene las propiedades de identidad e invariancia al desplazamiento y satisface las leyes conmutativa, asociativa y distributiva. Es importante hacer notar que la convolución es el núcleo de los sistemas LTI.

En general, una señal discreta puede ser expresada siempre como la suma de los pesos de funciones delta desplazadas, lo cual podemos observar en la ecuación (2.4), en donde $x(n) * h(n)$ se usa como una notación abreviada para la operación de la suma. Es importante notar que en la suma de la convolución, el signo de k es diferente en $x(k)$ y $h(n-k)$, si utilizamos el mismo signo para k, lo que obtenemos es la ecuación (2.5) de correlación:

$$c(n) = \sum_{k=-\infty}^{\infty} x(k)h(n+k) = x(n) \otimes h(n) \tag{2.5}$$

en donde $\otimes$ denota la operación de correlación, estas operaciones son semejantes pero arrojan resultados totalmente distintos. Como puede notarse, es muy semejante al de la convolución, al igual implica dos señales, y su principal diferencia es que tiene el objetivo de medir el grado de semejanza, cuando estas señales son distintas se genera una señal denominada *correlación cruzada* y cuando son idénticas *autocorrelación*

2.1.1. Correlación lineal

El concepto de correlación tal y como lo plantea la teoría de la probabilidad indica que, dos variables aleatorias están correlacionadas si conociendo como varía una podemos saber acerca de cómo varía la otra. Existen grados de correlación y esta puede ser positiva o negativa, su aplicación dentro del reconocimiento de patrones, es mostrar que tan similares o diferentes son un objeto de prueba de objetos de entrenamiento.

Sin embargo, la correlación directa funciona solamente cuando un objeto de prueba coincide bien con los objetos de entrenamiento, por lo que se han desarrollado diversos métodos para mejorar esta correlación básica y lograr atributos para tolerar ciertas diferencias y/o distorsiones tales como rotaciones de imagen, cambios en la escala (redimensión), variaciones en iluminación y la discriminación a otros tipos de clase (Kumar et al., 2005).

Como se ha visto, la correlación es utilizada en el reconocimiento de patrones, en la correlación se involucran dos imágenes, en donde una de ellas es la imagen de referencia $r[m, n]$ y la otra una imagen de prueba $t[m, n]$, y la correlación entre la imagen de referencia y la imagen de prueba es calculada por medio de la suma de la multiplicaciones elemento por elemento, iniciando en una posición en la cual se centra la imagen de referencia sobre la posición $(0, 0)$ de la imagen de referencia y desplazándose de izquierda

a derecha sobre todos los píxeles de la imagen de referencia. Como resultado se obtiene una salida de correlación con valores grandes en donde existe alguna coincidencia de la imagen de referencia.

La ecuación (2.6) presenta el proceso de correlación cruzada entre las imágenes de referencia y de prueba:

$$c(m,n) = \sum_{k} \sum_{l} t(k,l) r(k+m, l+n) \tag{2.6}$$

en donde se puede ver que la salida de correlación $c(m,n)$ es el resultado de dos sumas, por lo que la correlación es una operación integrativa.

Maximizando la correlación entre dos señales se minimiza el Error Cuadrático Medio (MSE), el MSE entre dos señales se obtiene por medio de la ecuación (2.7), la manera de minimizar la expresión, es maximizando el último término (Díaz Martínez, 2008).

$$\begin{aligned} MSE(m,n) &= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} (g(k,l) - (h(k+m, l+n)))^2 \\ &= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} g(k,l)^2 + \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h(k+m, l+n)^2 \\ &= -2 \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} g(k,l) \cdot h(k+m, l+n) \end{aligned} \tag{2.7}$$

2.1.2. Transformada de Fourier

Las computadoras digitales pueden manejar señales discretas a diferencia de los procesadores ópticos o sistemas analógicos, que pueden manejar señales e imágenes continuas. Una herramienta para analizar señales en el dominio de las frecuencias es la transformada de Fourier, la transformada discreta de Fourier (DFT) de una señal bidimensional se puede expresar por medio de la ecuación (2.8) (Kumar et al., 2005). Esta tranformada consta de dos componentes, una real conocida como *fase* y una compleja conocida como *magnitud*. La transformada de Fourier es muy utilizada porque

tanto la convolución como la correlación se pueden llevar a cabo en el dominio de las frecuencias.

En ocasiones es muy útil recuperar una señal de su transformada de Fourier, esto se lleva a cabo por medio de su inversa (IDFT) mostrada en la ecuación (2.9) (Kumar et al., 2005).

$$I[k,l] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} i[n,m]e^{-j2\pi(\frac{nk}{N}+\frac{ml}{M})},$$
$$k = 0,1,2,...,(N-1) \text{ y } l = 0,1,2,...,(M-1) \tag{2.8}$$

$$i[m,n] = \frac{1}{MN}\sum_{k=0}^{N-1}\sum_{l=0}^{M-1} I[k,l]e^{j2\pi(\frac{nk}{N}+\frac{ml}{M})},$$
$$n = 0,1,2,...,(N-1) \text{ y } m = 0,1,2,...,(M-1) \tag{2.9}$$

2.2. Sistemas no lineales

Los sistemas lineales no producen respuestas satisfactorias en aplicaciones donde existe ruido de naturaleza no aditiva, además de que algunos problemas tienen soluciones de naturaleza no lineal (Díaz Martínez, 2008). Estos filtros tienen la forma:

$$g(m,n) = G[f(m,n)] \tag{2.10}$$

en donde G es una función no lineal (Pitas and Venetsanopoulos, 1990).

Como se vio en secciones anteriores, este tipo de sistemas ya no cumplen con los principios de superposición, por lo que la salida de este tipo de filtros no se puede expresar como la convolución de la señal de entrada y de la función de transferencia del sistema. Es posible utilizar técnicas estadísticas para el análisis de estimadores robustos.

Existen diversos tipos de filtros no lineales, entre ellos se encuentran los filtros morfológicos.

2.2.1. Filtros morfológicos

Originalmente la morfología matemática se desarrolló para el procesamiento de imágenes binarias, se utilizan operaciones que se basan en la conectividad entre pixeles que se supone son de la misma clase, este procesamiento modifica la forma espacial o la estructura de los objetos dentro de una imagen. El elemento estructural es la herramienta que se utiliza para modificar las imágenes y tiene una forma definida, como por ejemplo un cuadrado. Este elemento se desliza dentro de la imagen y se aplica alguna operación lógica entre los elementos de ambos objetos. Como resultado se obtiene alguna operación morfológica (Díaz Martínez, 2008).

Las dos operaciones morfológicas básicas aplicables a una imagen son la dilatación y la erosión, estas operaciones (y sus modificaciones y combinaciones) se basan en la conectividad en el vecindario de píxeles y no son reversibles, una vez aplicadas a una imagen no es posible aplicar otra operación inversa para obtener la imagen original.

El objetivo de la erosión es eliminar ciertos píxeles de ciertas características de una imagen, la dilatación hace lo contrario, es decir, puede ser usado para agregar píxeles. Hay una variedad de reglas para decidir qué píxeles añadir o eliminar y para la formación de sus combinaciones.

La operación de erosión se puede utilizar para efectuar reconocimiento de objetos dentro de una imagen binaria, porque da por salida los puntos centrales en los cuales el elemento estructural coincide con algún objeto dentro de la imagen.

Sea X un objeto binario dentro de una imagen, la cual es erosionada con el elemento estructural A y sea X^c el complemento de X, el cual es erosionado por el elemento B; el conjuto de puntos en los cuales la versión desplazada de (A, B) coinciden dentro de la imagen X es llamado la transformada acierto-falla de X por (A, B), la transformación se denota por $\odot$

$$X \odot (A, B) = x : A_{+x} \subseteq X, B_{+x} \subseteq X^c \tag{2.11}$$

La operación ha sido utilizada en la detección de objetos en imágenes binarias, como por ejemplo la mejora en la calidad de texto en imágenes binarias que presentan una pobre resolución, son mejoradas utilizando la dilatación de los píxeles correspondientes al texto.

2.3. Filtros de correlación

2.3.1. Reconocimiento basado en correlación

La popularidad del uso de métodos de correlación en el reconocimiento de patrones se debe principalmente al rol que los filtros de acoplamiento han jugado en la detección de señales corrompidas por ruido blanco aditivo.

Una vez que se detecta el objetivo, se puede usar la salida del filtro de acoplamiento, para estimar el tiempo de desplazamiento relativo entre las señales transmitidas y recibidas. Siempre que la velocidad de la señal de propagación sea constante y conocida, de este tiempo de retardo se obtiene el la distancia del objetivo. Mediante el uso de al menos tres señales transmitidas, cuyas locaciones de transmisión son conocidas, se puede estimar el alcance del objetivo a las tres posiciones conocidas y así conocer la posición del objetivo por medio de triangulación. Es posible estimar este desplazamiento en la frecuencia y por lo tanto la velocidad de un objetivo (Kumar et al., 2005).

El proceso basado en correlación involucra dos imágenes, en donde uno se desempeña como imagen de referencia (o filtro) y una imagen de prueba (escena de prueba). Para detectar y/o localizar la imagen de referencia dentro de la escena de prueba se aplica la operación de correlación. En dado caso de que la imagen de referencia exista dentro de la imagen de escena se presenta una correlación alta en el punto central donde el objeto se empalme totalmente con el objeto de referencia. Por lo que este proceso consta de una entrada (escena de prueba), una plantilla almacenada de referencia o filtro, y una salida en forma de plano de correlación. La figura 2.1 muestra de forma

sencilla como es llevado a cabo este proceso.

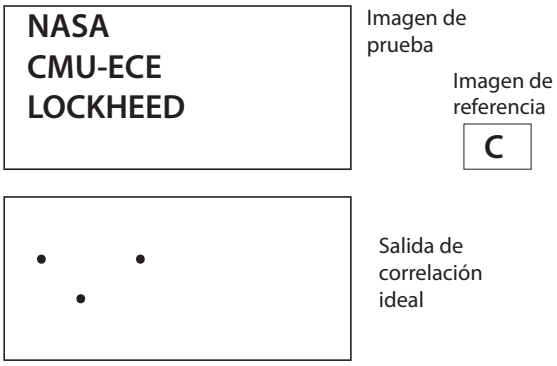


Figura 2.1: Esquema de la correlación de imágenes. Tomado de Kumar et al. (2005).

2.3.2. Filtro de acoplamiento

El filtro de acoplamiento (MF, del inglés Matched Filter), es el filtro de correlación más básico, es óptimo en la detección de imágenes de referencia conocidas que han sido corrompidas por ruido blanco aditivo, tiene un desempeño pobre cuando la imagen presenta distorsiones tales como rotaciones o redimensiones en la escala, el MF hizo su aparición de la maximización de la relación señal-ruido (SNR) entre señales de radar (Turin, 1960). Los filtros de acoplamiento en dos dimensiones son comúnmente utilizados en el procesamiento digital de imágenes.

Sea $s(t)$ una señal transmitida y $r(t)$ su correspondiente señal recibida que contiene efectos tales como la atenuación, tiempo de retardo, desplazamiento Doppler y ruido. Con el modelo de las ecuaciones (2.13) y (2.12), podremos considerar solamente los efectos del ruido aditivo, el tiempo de retardo y los desplazamientos de frecuencia se pueden estimar de la señal recibida. La atenuación causa un decremento en la SNR, la cual afecta el rendimiento de la detección. Sin embargo la atenuación no afecta la optimización del filtro SNR máximo.

Para el modelo de ruido aditivo, el problema de detección se simplifica a la elección

entre las siguientes dos hipótesis:

$$H_0 : r(t) = n(t) \tag{2.12}$$

$$H_1 : r(t) = s(t) + n(t) \tag{2.13}$$

en donde la ecuación (2.12) solo contempla una señal compuesta únicamente por ruido $n(t)$, y la ecuación (2.13) se contempla además al objeto de referencia $s(t)$. Este ruido se modela como un proceso aleatorio estacionario, con una media de 0 y una densidad de potencia espectral (PSD) $P_n(f)$.

La SNR permite presentar el efecto de ruido en el pico de salida, y se define como:

$$SNR = \frac{|E\{\eta/H_1\}|^2}{var\{\eta\}} \tag{2.14}$$

en donde η es el pico en la salida y, $E\{\cdot\}$ y $var\{\cdot\}$ representa al valor esperado y la varianza. En esta ecuación se desea maximizar al numerador.

El filtro de acoplamiento está dado por la siguiente ecuación (VanderLugt, 1964)

$$H(u, v) = \frac{kS^*(u, v)}{N(u, v)} \tag{2.15}$$

en donde $S^*(u, v)$ representa el complejo conjugado de la imagen de referencia, $N(u, v)$ es la densidad espectral del ruido blanco de fondo y k es una constante para hacer que $|H(u, v)| < 1$ para filtros espaciales pasivos.

La construcción de un filtro de acoplamiento en el dominio de las frecuencias se realiza de la siguiente manera, si $f(u, v)$ es una imagen de referencia entonces $H^*(u, v) = \mathcal{F}\{f(u, v)\}$ es el filtro de acoplamiento, en donde $\mathcal{F}$ es la transformada de Fourier. Para realizar la operación de correlación, se tiene que calcular también la $\mathcal{F}$ de la imagen de prueba $g(u, v)$ de la siguiente manera $G(u, v) = \mathcal{F}\{g(u, v)\}$, la correlación se calcula por medio de la siguiente ecuación:

$$y(x,y) = \mathcal{F}^{-1}[H^*(u,v)]\,[G(u,v)] \tag{2.16}$$

en donde $\mathcal{F}^{-1}$ representa a la inversa de la transformada de Fourier, y $(H^*)(G)$ es la multiplicación elemento a elemento de H y G.

2.3.3. Filtros de funciones Sintéticas Discriminantes

Hester and Casasent (1980) introdujeron el concepto de filtro Función Sintética Discriminante (SDF, del inglés Synthetic Discriminant Function) con el objetivo de manejar un patrón de variabilidad con menos filtros de correlación. Este filtro asume la existencia de un conjunto de imágenes de entrenamiento representativos de una clase.

El primer filtro SDF requirió que el modelo asociado fuera la suma ponderada de MFs , con los pesos elegidos de tal modo que los valores de la salida de correlación correspondientes a las imágenes de entrenamiento darían valores previamente especificados en el origen.

Por ejemplo, para todas las imágenes de entrenamiento de la misma clase, el valor de correlación debería ser igual, tomando el nombre de ECP SDF. Con esto se espera que con el filtro de correlación resultante se producirían valores para el pico cercanos a uno para imágenes no utilizadas en el entrenamiento pero que son de la clase que se desea reconocer y valores cercanos a cero para imágenes de la clase impostora (Kumar and Savvides, 2006; Kumar et al., 2005).

Un conjunto de ecuaciones lineales que describen la limitación de los picos de correlación con restricciones en el origen se pueden escribir como:

$$\boldsymbol{X}^{+}\boldsymbol{h} = \boldsymbol{u} \tag{2.17}$$

en donde $\boldsymbol{h}$ es un vector columna del filtro con d elementos, + denota la transpuesta de la conjugada. Además, $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N}]$ es una matriz con una dimensión

de $d \times N$ con N imágenes de entrenamiento en forma de vectores (cada uno con d pixeles) como sus columnas, y $\boldsymbol{u} = [u_1, u_2, \ldots, u_N]^T$ es un vector columna de $N \times 1$ que contiene los valores del pico deseados para las imágenes de entrenamiento. Sin embargo el número de imágenes de entrenamiento N es generalmente más pequeño que la dimensión d de los filtros. Para encontrar una unica solución, suponemos que $\boldsymbol{h}$ es una combinación lineal de la imágenes de entrenamiento, es decir $\boldsymbol{h} = \boldsymbol{X}\boldsymbol{a}$ en donde $\boldsymbol{a}$ es el vector de pesos para las combinaciones lineales de las columnas de la matriz de datos $\boldsymbol{X}$, $\boldsymbol{X}^+\boldsymbol{X}\boldsymbol{a} = \boldsymbol{u} \Rightarrow \boldsymbol{a} = (\boldsymbol{X}^+\boldsymbol{X})^{-1}\boldsymbol{h}$, entonces ECP SDF tiene la siguiente expresión (Vijaya Kumar et al., 2004).

$$\boldsymbol{h_{SDF}} = \boldsymbol{X}(\boldsymbol{X}^+\boldsymbol{X})^{-1}\boldsymbol{u} \tag{2.18}$$

Para construir un filtro SDF en el dominio de las frecuencias, se lee el conjunto de imágenes de entrenamiento, entonces a cada imagen de ese conjunto se le aplica la transformada discreta de Fourier, para después al resultado de esa operación(a cada matriz) convertirlo en un vector columna con N elementos, en donde N es igual al número de píxeles que contiene la imagen. X es una matriz que se construye a partir de los vectores columna de las imágenes de entrenamiento, por lo que el número de columnas de X será N y su número de columnas será igual a la cantidad de imágenes de entrenamiento; y es sobre esta matriz en donde se lleva a cabo la transpuesta de la conjugada (X^+), para después multiplicar esta transpuesta por la X e invertir la matriz que resulta de este producto, para después calcular el producto de X por la matriz inversa resultante y finalmente por el vector u. El vector u contiene valores que indican la salida deseada.

2.3.4. Filtro del mínimo promedio de la energía de correlación

El filtro SDF sólo controla la salida para imágenes de entrenamiento centradas en el origen. Y dado a que los patrones de entrada no están necesariamente centradas, es casi imposible saber dónde están estos valores predefinidos dentro del resultado.

El filtro del mínimo promedio de la energía de correlación (MACE, del inglés Minimum Average Correlation Energy) tiene como objetivo hacer que los valores predefinidos, aquellos centrados en el origen para las imágenes de entrenamiento de clase verdadera tengan los valores más grandes en el plano de correlación de salida y valores pequeños en respuesta a imágenes de otras clases. Las salidas de correlación de filtros MACE bien diseñados por lo general muestran picos agudos para las imágenes de entrada de la clase verdadera, por lo que la detección y la ubicación es sencilla y robusta (Kumar and Savvides, 2006).

Este filtro compuesto fue diseñado para minimizar la energía de correlación promedio (ACE, del inglés Average Correlation Energy) que resulta de las imágenes de entrenamiento y restringir el valor en el origen de los valores predefinidos. La principal diferencia entre el filtro MACE y los anteriores es la aparición de las matrices $\boldsymbol{D_i}$ de tamaño d^2 que contiene la potencia del espectro de la imagen de entrenamiento i a lo largo de su diagonal, y la matriz diagonal $\boldsymbol{D}$ es el promedio de todas las $\boldsymbol{D_i}$. Entonces el filtro MACE esta dado por la siguiente ecuación.

$$\boldsymbol{h_{MACE}} = \boldsymbol{D}^{-1}\boldsymbol{X}(\boldsymbol{X}^{+}\boldsymbol{D}^{-1}\boldsymbol{X})^{-1}\boldsymbol{u} \tag{2.19}$$

En este filtro el principal cálculo computacional es llevada a cabo al calcular la inversa de la matriz $N \times N$.

Para construir un filtro MACE en el dominio de las frecuencias, se lee el conjunto de imágenes de entrenamiento, y se calcula la transformada discreta de Fourier de cada imagen, a continuación se crea un vector columna de la imagen. Calculamos la

magnitud y la elevamos al cuadrado, entonces se le suma este resultado a la matriz D. Después de esto construimos la matriz X, que cuya cantidad de columnas es la cantidad de imágenes de entrenamiento, y filas la cantidad de elementos de la imagen de entrenamiento. A continuación calculamos la matriz D^{-1} , que contiene el promedio de las energías espectrales de cada imagen de entrenamiento.

2.3.5. Filtro no lineal ley K

Con el fin de mejorar el rendimiento de los filtros compuestos es términos de discriminación (cuando son comparados con clases similares a la clase objetivo), agudeza en el pico de correlación y una robustez al ruido, se aplican técnicas de filtrado no lineal a los filtros compuestos (Javidi et al., 1997). Una de estas técnicas consiste en aplicar la transfomada no lineal ley k-ésima a cada una de las transformadas de Fourier de las imágenes de entrenamiento, para entonces utilizar estos filtros no lineales y formar un filtro compuesto. Cuando se aplica la no linealidad a una matriz, ésta se aplica a cada elemento.

La no linealidad es aplicada a la ecuación (2.18) del filtro SDF, cuando se aplica esta operación a las transformadas de Fourier de todas las imágenes, se obtiene el filtro compuesto no lineal en el plano de Fourier.

$$\hat{\boldsymbol{h}}^k = \hat{\boldsymbol{X}}^k(\hat{\boldsymbol{X}}^+\hat{\boldsymbol{X}}^k)^{-1}\boldsymbol{u} \tag{2.20}$$

Este tipo de filtros realizan varias correlaciones no lineales entre una escena de entrada y diferentes imágenes de referencia. Un filtro no lineal se puede expresar de la siguiente forma:

$$H(x,y) = |F(x,y)|^k e^{-i\phi(x,y)} \tag{2.21}$$

en donde $F(x,y)$ es la transformada de Fourier del objeto que deseamos reconocer, ϕ es la fase de la imagen objetivo $|.|$ es el modulo de $F(x,y)$ y k es el factor de no linealidad.

Si se cambia el valor de k a 1 obtenemos un filtro de acoplamiento clásico, si se cambia a 0 obtenemos un filtro de solo fase y si se cambia a -1 obtenemos un filtro inverso. Cuando el operador de no linealidad modifica la transformada de Fourier en ambas imágenes, es decir en la escena de entrada y en la imagen de referencia, se considera que el correlador no es lineal.

Construir un filtro K-law es semejante a construir un filtro SDF, se realiza la siguiente modificación, después de obtener la transformada discreta de Fourier de cada imagen de entrenamiento, se calcula la magnitud y se eleva cada elemento de esa magnitud al valor de K-law, que regularmente es de 0.3, al resultado de esta operación se multiplica por una matriz $e^{-i\phi(x,y)}$, la cual representa la fase de la transformada inversa de la imagen de entrenamiento.

2.4. Filtros no lineales compuestos

2.4.1. Proceso del filtrado no lineal

Díaz Martínez (2008) propone un método de filtrado localmente adaptable de la señal de referencia y una ventana que se desplaza sobre cada píxel en la escena de prueba. El tamaño de esta ventana se ajusta de manera que pueda contener al objeto de referencia. Esta ventana deslizante contiene una vecindad espacial W de los píxeles que rodean geométricamente el píxel central de la ventana. Utilizan una forma de vecindad similar a la forma de la región de soporte del objeto de referencia y su tamaño es aproximadamente el mismo que el del objeto de referencia. Esta región de soporte de define como

$$w(m,n) = \begin{cases} 1 & \text{al interior del objeto de referencia} \\ 0 & \text{fuera del objeto de referencia} \end{cases} \tag{2.22}$$

Este procesamiento se realiza de la siguiente manera, primero la ventana deslizante se centra en cada pixel de la escena de prueba. Después se extrae de la escena de prueba la vecindad de los píxeles que rodean al píxel central y finalmente se aplica la operación de correlación entre el objeto de referencia y la vecidnad extraida.

2.4.2. Descomposición por umbral

Fitch et al. (1984) introduce el concepto de la descomposición por umbral, la cual permite que una imagen $S(k,l)$ con Q niveles de gris pueda ser representada por una suma de las imágenes binarias.

$$S(k,l) = \sum_{k=1}^{Q-1} S^q(k,l) \tag{2.23}$$

en donde $S(k,l)$, $q = 1, ..., Q-1$ es el conjunto de imágenes binarias obtenidas de la descomposición por umbral de una imagen original con un umbral q, de la forma:

$$S^q(k,l) = \begin{cases} 1, & \text{si } S(k,l) \geq q \\ 0, & \text{en cualquier otro caso} \end{cases} \tag{2.24}$$

2.4.3. Correlación morfológica

Las operaciones morfológicas básicas (erosión, dilatación), se pueden extender a imágenes en escala de grises aplicando operaciones de estadísticas de orden:

Sean $S(k,l)$ y $T(k,l)$ dos imágenes en escala de grises con Q niveles de gris y sean $S^q(k,l), q = 1, 2, ...Q-1$ y $T^q(k,l), q = 1, 2, ...Q-1$ las imágenes obtenidas de la descomposición por umbral de las imágenes $S(m,n)$ y $T(m,n)$ respectivamente la correlación no lineal entre las imágenes $S(k,l)$ y $T(k,l)$ puede calcularse utilizando imágenes

binarias como

$$c(k,l) = \sum_{k=m,n\in W} \sum_{k=1}^{Q-1} MIN[S^q(m+k, n+l), T^q(m,n)] \tag{2.25}$$

2.4.4. Funciones Sintéticas Discriminantes no lineales

Los filtros no lineales compuestos se basan en operaciones lógicas y la operación de correlación morfológica, estos filtros incorporan información de varios objetos de la clase falsa y verdadera, de manera similar a una función discriminante sintética. Además la salida es normalizada para obtener el valor de correlación deseado. Con esto se obtiene un nuevo filtro basado en funciones discriminantes sintéticas no lineales (NSDF). La definición formal de los filtros compuestos es el siguiente: Sean $T_i(k,l), i = 1, ..., N$ un conjunto de N imágenes de entrenamiento en escala de gris pertenecientes a la clase que se desea reconocer y sea $P_i(k,l), i = 1, ..., M$ un conjunto de M imágenes pertenecientes a la clase que se desea rechazar. Primero se define una nueva imagen de referencia compuesta :

$$H_{NSDF} = \sum_{q=1}^{Q-1} \left[\bigcap_{i=1}^{N} T_i^q(k,l) \right] \bigcap \left[\overline{\bigcup_{i=1}^{M} P_i^q(k,l)} \right] \tag{2.26}$$

donde $T_i^q(k,l), q = 1, ..., Q-1, i = 1, ..., N$ y $P_i^q(k,l), q = 1, ..., Q-q, i = 1, ..., M$ son las imágenes binarias obtenidas por descomposición por umbral de las imágenes de clase falsa y verdadera respectivamente. Es esta ecuación $\cup_{i=2}^{M}$ representa la unión lógica entre las imágenes binarias. El resultado de la unión en las coordenadas (k,l) es cero si los píxeles correspondientes son cero, de otro modo el resultado es uno. Una vez hecho esto se calcula la correlación utilizando la ecuación 2.25.

2.4.5. Reconocimiento entre dos clases de objetos

Estos filtros pueden ser utilizados para reconocer objetos de dos clases diferentes. Esto asignando una clase al conjunto $T_i(k,l), i = 1, ..., N$ y otra clase al conjunto $P_i(k,l), i = 1, ..., M$. Como resultado al encontrarse un objeto del primer conjunto (clase verdadera), el valor de la correlación sera u mientras que para los demas objetos sera cero.

2.4.6. Reconocimiento entre objetos de la misma clase

Para reconocer objetos de una misma clase se puede utilizar la versión de los filtros compuestos en la cual se incluyan sólo los objetos de la clase buscada, si la siguiente manera:

$$H_{NSDF} = \sum_{q=1}^{Q-1} \left[\bigcap_{i=1}^{N} T_i^q(k,l) \right] \tag{2.27}$$

donde $\cap_{i=1}^{N}$ representa la intersección lógica entre las imágenes binarias. El resultado de la intersección en las coordenadas (k,l) es uno si los píxeles correspondientes son uno, y de cualquier otro modo es cero. $T_i^q(k,l)$ son las imágenes binarias obtenidas al descomponer cada imagen del conjunto de entrenamiento. Para calcular la correlación con la ecuación 2.25.

2.5. Resumen del capítulo

En este capítulo se abordaron los fundamentos teóricos en los que se basó este trabajo de investigación, se define lo que es una imagen y su representación digital, los sistemas lineales y no lineales que fueron utilizados en dos tipos de correlación (correlación lineal y morfológica). Además se abordó la transformada discreta de Fourier, una herramienta muy utilizada en las computadoras digitales para el análisis de señales

discretas.

Se mostró una introducción al reconocimiento basado en correlación, la forma en que son construidos algunos de los filtros básicos y compuestos más conocidos en el dominio de las frecuencias. Para concluir con el proceso de filtrado no lineal y las principales tareas que conlleva, tal como la descomposición por umbral, la construcción de un filtro compuesto SDF y su aplicación al reconocimiento entre dos clases de objetos y entre objetos de la misma clase. Se mostraron también las ecuaciones que representan la correlación morfológica y el proceso de construcción del filtro compuesto NSDF.

Capítulo 3

Procesamiento facial

La mayoría de nosotros aprendemos a leer y a escribir durante las estancias en el jardín de niños y primaria, pasamos por un proceso de entrenamiento en el que adquirimos los conocimientos necesarios para reconocer, casi de forma instantánea (aunque no estemos prestando total atención) objetos, ruidos, números y letras entre otras cosas.

En las ciencias e ingeniería computacional se hace el uso del reconocimiento de patrones para llevar a cabo un proceso similar y así crear aplicaciones específicas para el reconocimiento automático de objetos. Un ejemplo de aplicación en la industria, son las áreas en donde se requiere optimizar procesos para identificar objetos y/o sus características de manera automática, es decir, con poca o nula intervención humana.

El reconocimiento de patrones tiene diversas aplicaciones. En la biometría se utiliza entre otras cosas para la detección, validación, clasificación, identificación y autentificación de rasgos humanos tales como caras, iris, huellas dactilares, etc. En la actualidad, no solo la biometría hace uso del reconocimiento de patrones, también es muy utilizado en áreas que involucran tareas tales como la seguridad de información, aplicación de la ley y vigilancia, tarjetas inteligentes y control de acceso.

El paradigma estándar del reconocimiento de patrones se muestra en el diagrama de la Figura 3.1. Este paradigma consta de cinco pasos, el primero consiste en darle un

patrón de entrada (imagen), el cual es pre-procesado en el paso dos, esto significa aplicarle cambios posibles para que la extracción de características pueda ser más eficiente, estos cambios pueden ser la eliminación de ruido, convertir la imagen a escala de grises o cualquier otra transformación que mejore el patrón de entrada. El tercer paso es la extracción de características, su objetivo es extraer descriptores del patrón de entrada, estos descriptores capturan su esencia, este paso es importante porque es donde se evalúan los descriptores, que pueden resultar de buena o mala calidad, esto será fundamental para el paso siguiente que es la clasificación, si se tienen buenos descriptores habrá una buena clasificación. El último paso es la clasificación cuyo principal objetivo es proveer una clase etiquetada a partir del conjunto de características extraídas y asi proveer una clase de salida.

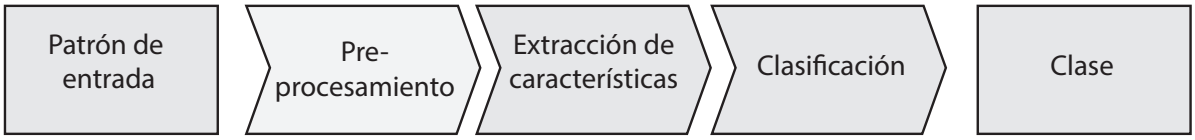


Figura 3.1: Diagrama de bloques que muestra el paradigma del reconocimiento de patrones. Tomado de Kumar et al. (2005).

Un área que hace uso del reconocimiento de patrones, es en el procesamiento facial el cual involucra los procesos de detección, reconocimiento y seguimiento facial, procesos que se detallan a en las siguientes secciones.

3.1. Detección facial

La detección facial es una aplicación del reconocimiento de patrones, es además una de las técnicas fundamentales que permiten una interacción natural humano-computadora. La detección facial es un paso obligado para los algoritmos de análisis facial. El rostro humano es un objeto dinámico y tiene un alto grado de variabilidad en su apariencia,

lo que hace que la detección facial sea un problema difícil en la visión por computadora. Se han propuesto una gran variedad de técnicas que van desde algoritmos basados en los bordes a enfoques compuestos de alto nivel utilizando métodos avanzados de reconocimiento de patrones.

Figura 3.2: Ejemplo de imagen facial típica utilizado en la detección. Tomado de Pesquisa et al. (2006).

El objetivo principal de la detección facial está dado por el siguiente enunciado: dada una imagen arbitraria o escena de video, detectar y localizar un número indefinido de caras (Hjelmå s and Low, 2001). La solución al problema implica la segmentación, la extracción y verificación facial y posiblemente la de rasgos faciales en una escena no controlada.

Algunos retos asociados a la detección facial son atribuidos a variaciones en escala, ubicación, orientación, pose, expresiones faciales, condiciones de iluminación, oclusiones, etc. Debido a estos retos, existen diversos enfoques para llevar a cabo la detección facial. Kriegman and Ahuja (2002), agruparon los métodos en cuatro categorías:

1. Los métodos basados en conocimiento, los cuales utilizan reglas predefinidas del conocimiento humano para determinar una cara.

2. El enfoque invariante a características, se apoya en características de la estructura de la cara robustas a poses y variaciones de iluminación.

3. Los métodos basados en plantillas de acoplamiento, utilizan plantillas de caras pre-almacenadas para juzgar si una imagen es una cara.

4. Los métodos basados en la apariencia, las cuales aprenden modelos de caras de un conjunto de entrenamiento para optimizar la detección.

El algoritmo de detección facial con más impacto en la última década es el trabajo presentado por Viola and Jones (2004), este algoritmo incluso se emplea en cámaras de video, fotográficas, celulares, etc..., dado a que no requiere un equipo de computo con mucho poder de procesamiento. Este algoritmo consta de tres partes, una representación para las imágenes la cual denominan *imagen integral*, un clasificador simple y eficiente el cual hace uso de *AdaBoost* y un método para la combinación de clasificadores cada vez más complejos en una estructura en cascada el cual aumenta drásticamente la velocidad del detector. La Figura 3.3 nos muestra la detección que ofrece este algoritmo.

Figura 3.3: Detección facial utilizando el algoritmo. Tomado de Viola and Jones (2004).

3.2. Reconocimiento facial

A diferencia de la detección facial el reconocimiento no solo involucra localizar rostros sino que también se encarga de clasificarlos. El reconocimiento facial es una de las aplicaciones más exitosas en el análisis y la comprensión de imágenes, ha recibido una atención significativa especialmente durante los últimos años, esto gracias a la amplia gama de aplicaciones tanto comerciales como las requeridas por la biometría y por la disponibilidad de tecnología viables después de treinta años de investigación. La Tabla 3.1 resume algunas de las áreas en las que se hace uso del reconocimiento facial (Zhao et al., 2003).

Tabla 3.1: Algunas aplicaciones de la reconocimiento facial.

Áreas	**Aplicaciones especificas**
Biométrica	Licencias de conducir, programas de ayuda social. Inmigración, Identificaciones, pasaportes, registro del voto. Fraude en servicios de bienestar social.
Seguridad de la información	Aplicaciones de seguridad, seguridad en base de datos, encriptación de archivos. Seguridad en redes intranet, acceso a internet, archivos médicos Terminales de comercio electrónico.
Aplicación de la ley y vigilancia	Vigilancia avanzada de video, control CCTV. Control de portales, análisis post-eventuales. Hurto, seguimiento e investigación de sospechosos.
Tarjetas inteligentes	Seguridad de valores almacenados, autentificación de usuarios.
Control de acceso	Facilidad de acceso, acceso vehicular.

Un enunciado general del problema del reconocimiento facial puede ser formulado como de la siguiente manera: dada una serie de imágenes, identificar a una o más personas en la escena, esto utilizando una base de datos que tiene almacenadas imágenes faciales (Chellappa et al., 1995). La solución para este problema implica la detección

facial de escenas, extracción de características de regiones faciales, reconocimiento y autentificación (Zhao et al., 2003).

En el diagrama de la Figura 3.4 se muestra la configuración genérica de un sistema de reconocimiento facial, la cual consta de una imagen o vídeo de entrada, las fases de detección facial, extracción de características de regiones de la cara y la del reconocimiento facial, que da como salida la identificación o autentificación.

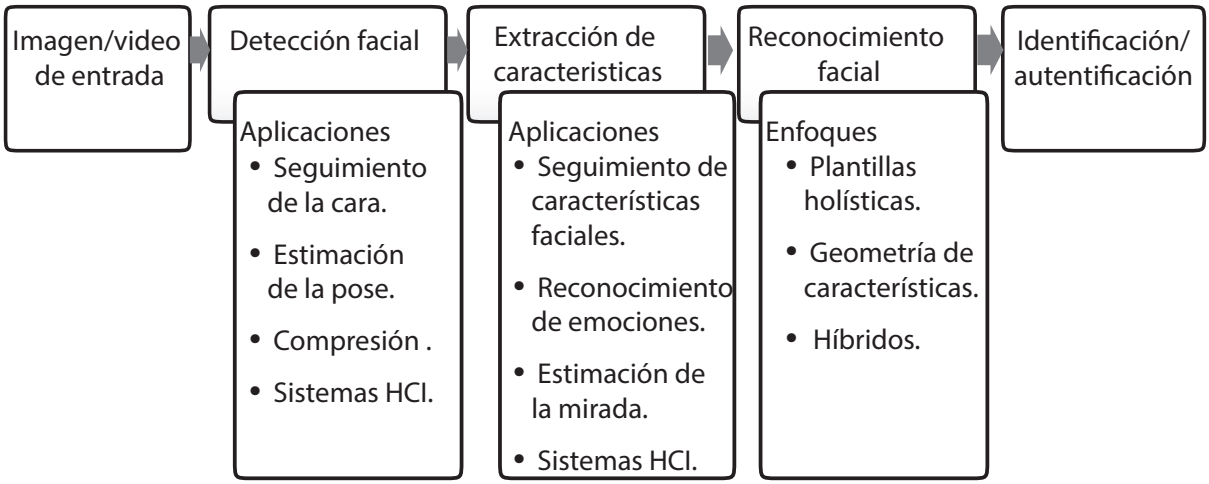


Figura 3.4: Configuración de un sistema genérico de reconocimiento facial. Tomado de Zhao et al. (2003).

3.3. Seguimiento facial

El seguimiento facial es un proceso que consta de dos partes: la primera consiste en la localización facial, mediante un sistema de detección de caras y de un sistema de reconocimiento facial, mientras que la segunda consiste en el seguimiento de estas a lo largo de una secuencia de imágenes.

El seguimiento facial es empleado en aplicaciones que requieren el manejo de fuentes de video, es un proceso iterativo realimentado por imágenes sucesivas (conocidas también como frames) que constituyen una secuencia de video.

Se puede decir que el seguimiento facial no es más que una relocalización continúa del rostro y sus componentes faciales a lo largo de un video, que además de tener el problema de la localización de imágenes estáticas, debe de tener en cuenta una dimensión temporal. Esto quiere decir que existe información adicional sobre las caras que están siendo seguidas, como por ejemplo la posición, orientación, forma, velocidad, etc., que es la historia de cada instancia a lo largo del tiempo (Vega Aquino, 2011).

El último paso del seguimiento facial es la de relocalizar el rostro en la nueva secuencia de video, algunos problemas que se presentan en el seguimiento facial son:

- El movimiento del rostro
- Desenfoque por movimiento
- Tiempo real
- Oclusión y desaparición
- Posición y orientación
- Cambios en la iluminación y expresión facial

3.4. Filtros de Correlación aplicados al reconocimiento y seguimiento facial

Kumar et al. (2005) define el proceso de correlación para el reconocimiento de objetos de la siguiente manera: "se dice que dos variables tienen alguna correlación si conociendo una podemos decir algo de la otra". Existen varios grados de correlación y esta puede ser positiva o negativa. Por lo que su rol en el reconocimiento de patrones es la de encontrar la similitud de un objeto o imagen de prueba.

En la Figura 3.5 se puede apreciar el proceso de correlación en el dominio de las frecuencias, (esto a través de la Transformada de Fourier), aplicada en el reconocimiento

facial. Este proceso involucra el uso de un conjunto de imágenes de entrenamiento para el diseño de un filtro y de una imagen de prueba. Tanto a la imagen de prueba como al conjunto de imágenes de entrenamiento se les aplica la transformada rápida de Fourier (FFT por sus siglas en inglés), para posteriormente realizar la operación de correlación. El plano de correlación producido por esta operación se obtiene con la transformada inversa de Fourier (IFFT), de esta forma al graficar el plano se puede visualizar el pico producido.

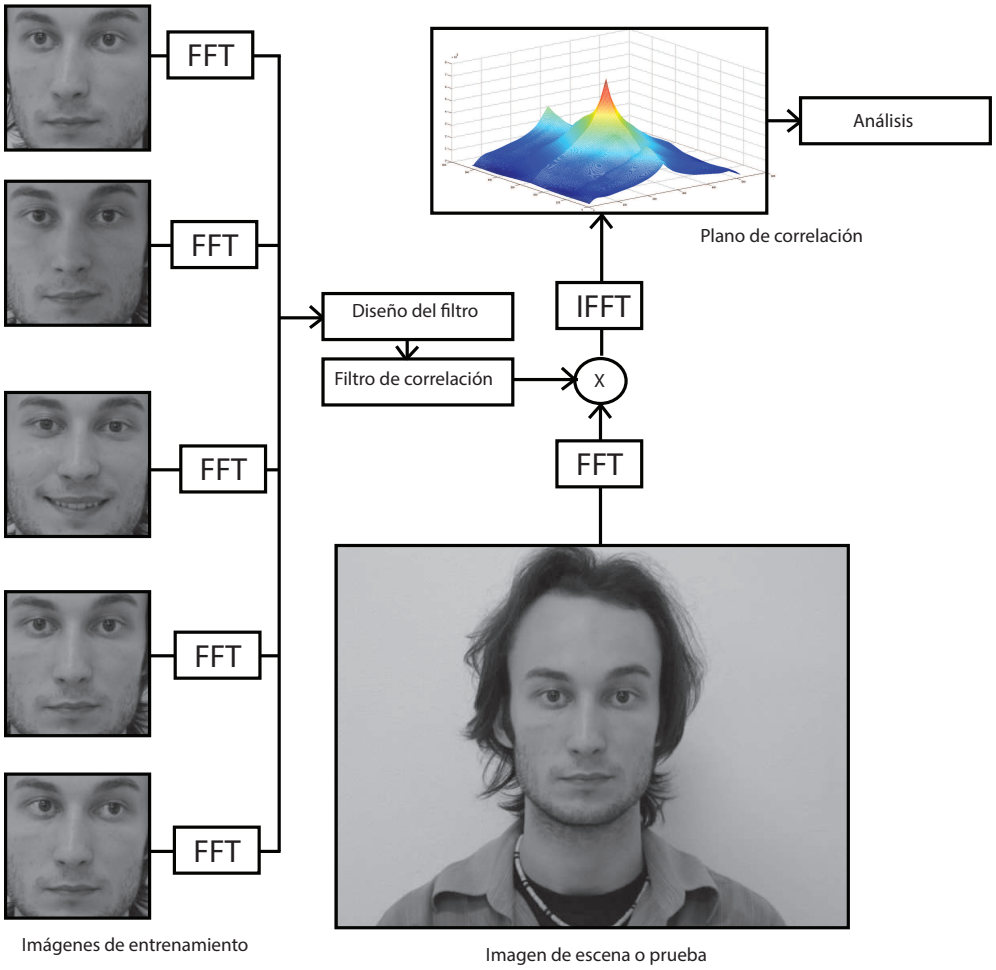


Figura 3.5: Diagrama de bloques del proceso de correlación vía FFT. Imagen tomada de Pesquisa et al. (2006).

En el proceso de análisis de la salida de correlación, se buscan los picos y se deter-

minan sus alturas, o se emplean otras métricas relacionadas con el valor del pico que permiten determinar si un objeto de entrada pertenece a una clase verdadera o a una clase falsa. La localización de esos picos indica la posición de los objetos dentro de la escena.

Algunas ventajas que ofrecen los filtros de correlación son los siguientes:

La invariancia a la traslación en la operación de filtrado, ya que si la imagen de entrada se traslada cierta cantidad de píxeles, entonces la salida del filtro se desplaza por la misma cantidad de píxeles. Cuando hace uso de los filtros de correlación, el desplazamiento se estima fácilmente mediante la localización de los picos de correlación. Los filtros de correlación pueden ser diseñados para tener cierta tolerancia al ruido, buena discriminación a objetos de una clase falsa, etc., se pueden utilizar para el reconocimiento de objetos parcialmente ocultos (González-Fraga and Kober, 2006), además pueden utilizar toda la información del objeto, esto es tanto la forma como el contenido.

Las técnicas basadas en filtros de correlación son buenas candidatas para la detección y reconocimiento facial, ya que ofrecen la precisión que se necesita en la verificación biométrica, en particular las técnicas avanzadas de filtros de correlación tales como las funciones discriminantes sintéticas (SDF), que pueden ofrecer un rendimiento muy bueno de acoplamiento aún con condiciones como la variabilidad de las imágenes biométricas (Vijaya Kumar et al., 2004).

3.5. Resumen del capítulo

En este capítulo se presentó el uso del reconocimiento de patrones en la biometría, enfocándose en el procesamiento facial y las tareas que la forman, se expuso la definición de detección y reconocimiento facial, los diversos enfoques que se siguen para implementarlos; se mostró además la configuración genérica de un sistema de reconocimiento facial, tarea que involucra la detección y algunas de sus aplicaciones prácticas dentro

de la biometría. Se incluyó también la definición de seguimiento facial y algunos de sus principales retos al ser aplicado en un entorno no controlado.

Se mostró también el proceso que sigue el reconocimiento basado en correlación utilizando el diseño de un filtro compuesto, aplicando la transformada rápida de Fourier en el dominio de las frecuencias para generar el plano de correlación y su posterior análisis.

Capítulo 4

GPGPU

El cómputo de propósito general en unidades de procesamiento gráfico (GPGPU), es un concepto reciente dentro de las ciencias de la computación, que trata de estudiar y aprovechar las capacidades de cómputo de la unidad de procesamiento gráfico (GPU). Las unidades de procesamiento gráfico se han especializado en el proceso de vértices y pixeles, que básicamente vienen siendo multiplicaciones masivas de vectores por matrices cuyos elementos son números de punto flotante.

El alto poder de cómputo introducido por las GPUs está permitiendo solucionar problemas desafiantes para los procesadores de las PCs, usando hardware de bajo costo por lo que desde sus inicios ha llamado la atención para el desarrollo de aplicaciones científicas. Recientemente la capacidad de las tarjetas de video ha dejado de crecer linealmente, para comenzar a crecer de forma casi exponencial, se han realizado comparaciones hechas entre los CPUs de las computadoras y las GPUs (ver Figuras 4.1 y 4.2), en cuanto al número de instrucciones de punto flotante que pueden realizar en un segundo (FLOP/s) y la diferencia en cuanto a hardware. En ellas se pueden apreciar que las CPUs se han incrementado en promedio 2.0 veces su poder de procesamiento, mientras que las GPUs lo han incrementado en promedio 3.7 veces cada 18 meses.

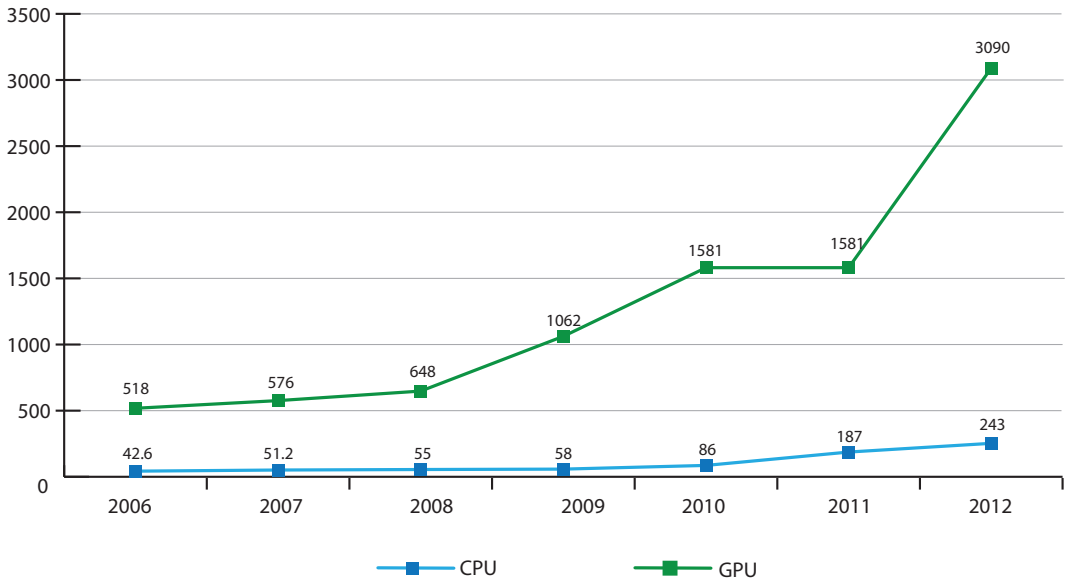


Figura 4.1: Comparación entre el incremento del rendimiento de las CPUs y las GPUs en gigaflops a través de los años. Tomado de Cook (2012).

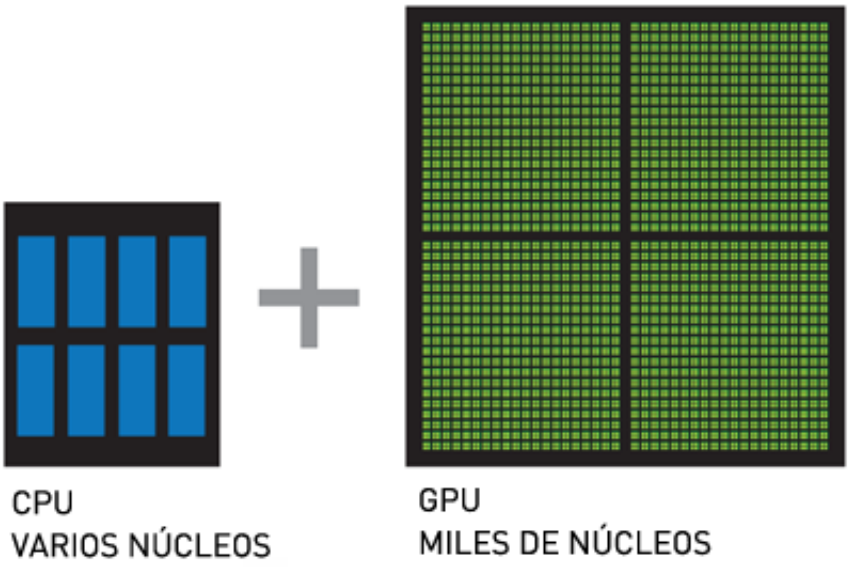


Figura 4.2: Las CPUs y las GPUs tienen fundamentalmente diferentes filosofías de diseño. Tomado de NVIDIA (2013).

4.1. GPU y CPU

Durante los últimos treinta años, uno de los métodos más importantes para mejorar el rendimiento de las computadoras personales ha consistido en el incremento de la velocidad del reloj del procesador, comenzando con las primeras computadoras personales de principios de 1980 cuyas unidades centrales de procesamiento corrían con relojes internos que operaban cerca de 1 MHz hasta hoy en día, en donde la mayoría de las computadoras de escritorio corren a velocidades entre 1 GHz y 4 GHz, esto es cerca de 1000 veces mas rápido que sus antecesores. Aunque el incrementar la velocidad de reloj de la CPU no es el único método por el cual la capacidad de cómputo puede ser mejorado, este ha sido desde siempre uno de los métodos más confiables.

Sin embargo, en años recientes los fabricantes han tenido que buscar opciones alternativas debido a que se han enfrentado a ciertas limitaciones en la fabricación de circuitos integrados, algunas de estas limitaciones son físicas y tienen que ver con la incapacidad de reducir el tamaño de los transistores, por lo que ya no hacen factible el incremento de la velocidad del reloj del procesador.

En la búsqueda de poder de procesamiento para las computadoras personales, a partir de la mejoras logradas y aplicadas a las supercomputadoras se planteó la siguiente cuestión, ¿Por qué no colocar más de un núcleo de procesamiento a una computadora personal?, por lo que en 2005 los principales manufacturadores comenzaron a ofrecer procesadores con dos núcleos, y al pasar de los años se fueron adicionando más, por lo que surgieron procesadores con 3, 4, 6 y hasta 8 núcleos, a esto se le conoce como la revolución multinúcleo, y marcó los indicios de que la computación paralela había arribado a las computadoras personales.

La idea del cómputo utilizando la unidades de procesamiento gráfico no es tan nueva como se cree, a finales de 1980 e inicios de 1990 Microsoft Windows ayudó a crear un mercado para este tipo de procesador con la introducción de la interfaz gráfica

de usuario, en los 90s se comienzan a comprar los primeros aceleradores gráficos 2D. Al mismo tiempo la compañia Silicon Graphics popularizaba el uso de los gráficos en 3D, en 1992 abre la interfaz de programación de su hardware liberando OpenGL. A mediados de los noventas compañías como NVIDIA y ATI comenzaron a producir tarjetas asequibles y que llamaban la atención a los compradores en general. Y es con el lanzamiento del modelo serie GeForce 3 en 2001 en que los desarrolladores comienzan a tener cierto control de los cálculos que se llevaban a cabo en las GPUs.

4.2. Introducción a CUDA

Para poder tener acceso a esta capacidad de cálculo, NVIDIA desarrolla CUDA (acrónimo de *Compute Unified Device Architecture*), el cual es una arquitectura de software que facilita la programación en paralelo utilizando sus GPUs. CUDA es una tecnología reciente, fue presentada formalmente en 2006, junto con la tarjeta de video GeForce 8800 GTX, esta arquitectura incluía nuevos componentes diseñados especialmente para el cómputo en GPU que corregían muchas de las limitaciones presentadas en sus anteriores tarjetas gráficas y que impedían su utilidad al cómputo de propósito general.

En estos en los últimos años CUDA se ha aplicado a diversos campos consiguiendo incrementos en rendimiento de algoritmos y aplicaciones como AMBER en el ámbito científico o Numerix y CompatibL en el mercado financiero logrando una aceleración 18 veces superior (NVIDIA, 2013).

4.2.1. Modelo de programación CUDA

CUDA es un modelo de programación paralela y un entorno de programación que tiene como objetivo facilitar el desarrollo de aplicaciones que puedan ejecutarse en los multiprocesadores de las tarjetas gráficas, sin tener que preocuparse por el modelo de

la tarjeta ni el aprender lenguajes de programación complejos o pensar en términos de primitivas gráficas.

Es una extensión del lenguaje C, por lo que el código GPU puede ser escrito en C estándar. El código se dirige tanto para el procesador central (CPU) y el procesador del dispositivo (GPU). El procesador central genera tareas multiproceso (o kernels como se les conoce en CUDA) en el dispositivo GPU, este tiene su propio programador interno que asigna las tareas al hardware de la GPU.

4.2.2. Estructura de un programa en CUDA

En la estructura de un programa CUDA se refleja la coexistencia de un *host* (CPU) y uno o mas *devices* (GPUs) en una computadora. Un archivo fuente de CUDA puede contener código tanto para el *host* como para el *device*. Se pueden colocar funciones que se ejecutan en GPU y declarar tipos de datos en archivos fuente C a través de palabras reservadas que utiliza CUDA. Las funciones que se ejecutan en GPU no pueden ser compiladas utilizando un compilador C estándar, para esto NVIDIA provee NVCC (*NVIDIA C Compiler*), que procesa el código fuente separando los tipos de código para CPU y GPU, la Figura 4.3, nos muestra como es que se lleva a cabo este proceso.

Durante la ejecución de un programa utilizando CUDA se intercalan o incluso se interponen ejecuciones de código en la CPU, y en la GPU. La inicialización y transferencia de datos, así como la obtención de los resultados y el lanzamiento de kernels por lo general es ejecutada en la CPU, mientras que la GPU es la encargada de realizar los cálculos. La Figura 4.4 muestra el flujo que se sigue la ejecución.

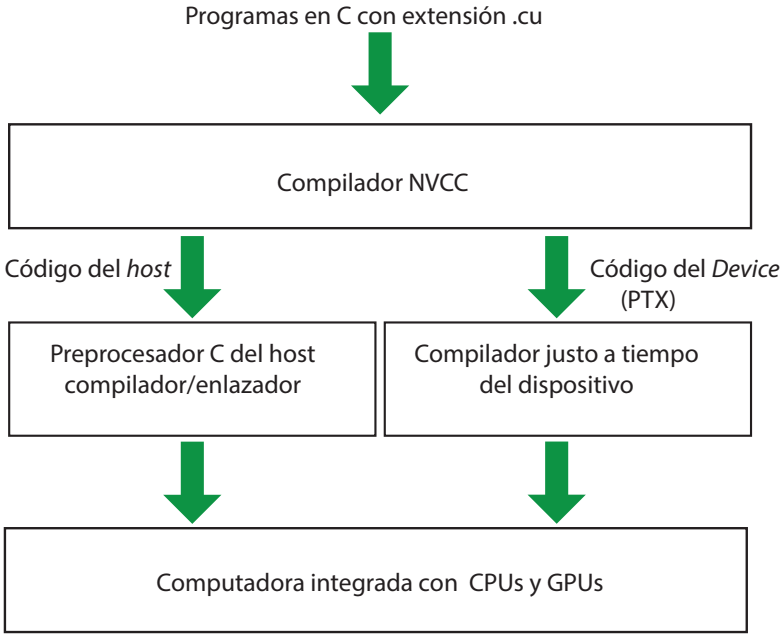


Figura 4.3: Descripción general del proceso de compilación de un programa CUDA. Tomado de Kirk and Wen-mei (2010).

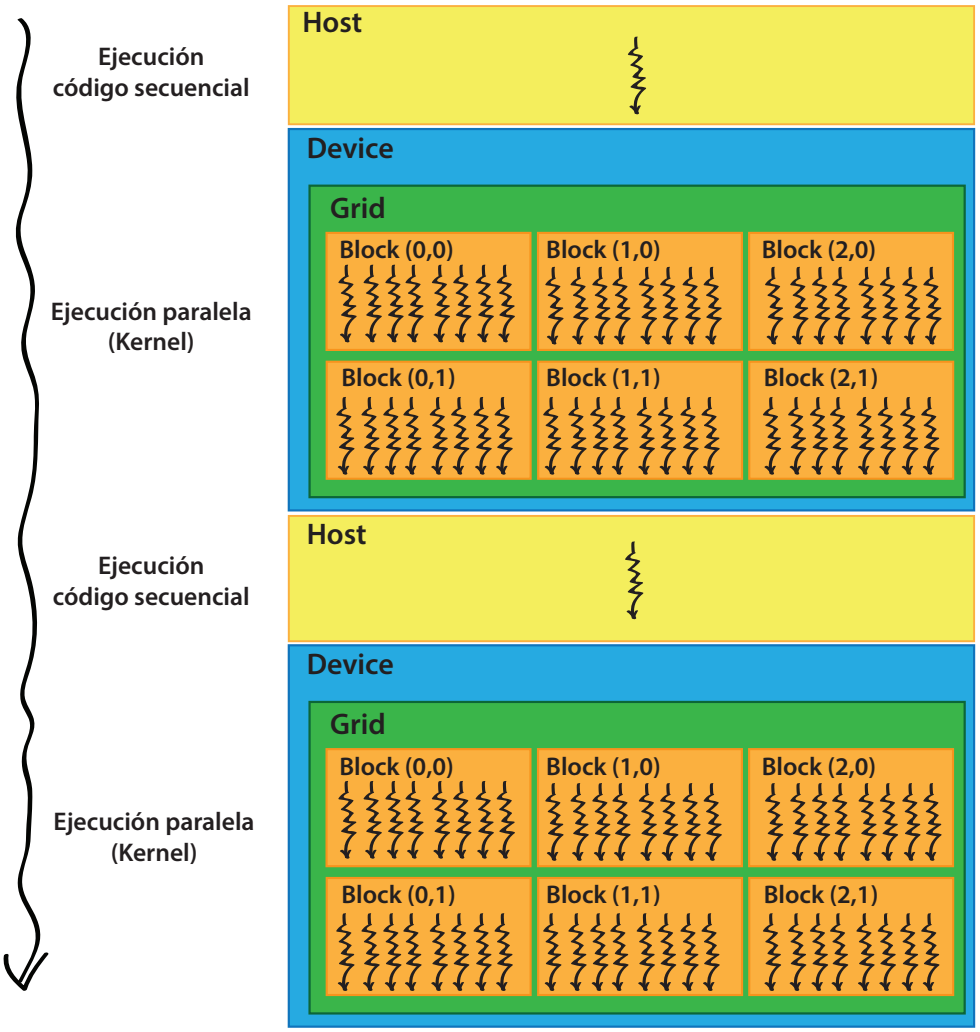


Figura 4.4: Flujo que sigue un programa en CUDA. Adaptado de Kirk and Wen-mei (2010).

4.2.3. Términos utilizados en CUDA (Threads, blocks y Grids)

A continuación se describen algunos términos que se utilizan en la programación utilizando CUDA, estos términos se aplican a componentes de la arquitectura de la GPU. La GPU puede manejar miles de *threads* o hilos, el programador debe de agrupar los hilos que va a utilizar en *blocks* o bloques, que a su vez se agrupan en un *grid* o malla. La imagen de la Figura 4.5 ofrece un acercamiento de la imagen de la Figura

4.4.

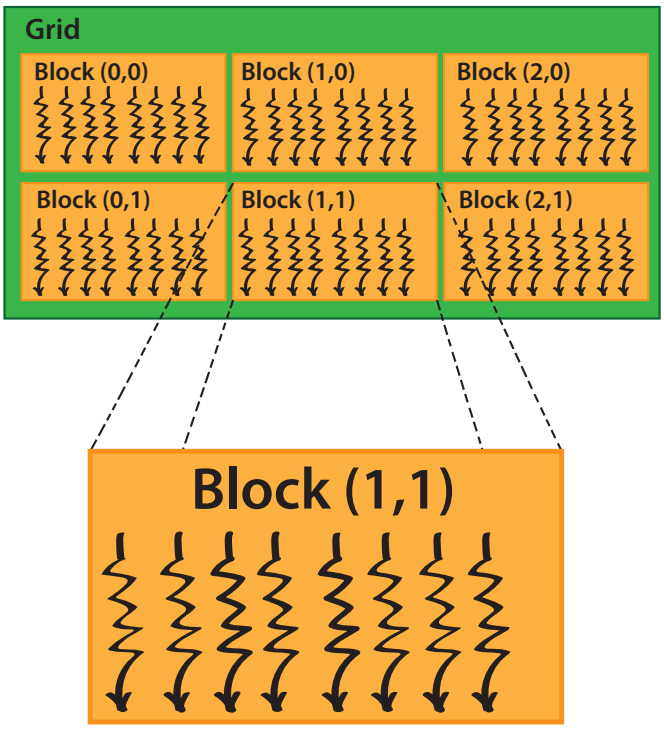


Figura 4.5: Hilos, bloques y mallas.

Los hilos se agrupan en *warps* para su ejecución, un warp en CUDA es un grupo de 32 hilos, el número mínimo de información o datos a ser procesados. En CUDA los programadores no definen el número de warps a ser utilizados, en lugar de eso trabajan con los bloques que pueden contener entre 64 y 1024 hilos. La forma en que estos hilos son ejecutados es en forma de recorrido por filas tanto de los bloques de la malla como de los hilos dentro del bloque. Cuando el bloque tiene más de 2 dimensiones entonces este recorrido pasa primero por las filas de los hilos localizados a menor Z (coordenada).

CUDA cuenta con un tipo de memoria denominada *shared memory* o memoria compartida a la que tienen acceso todos los hilos agrupados dentro de un bloque. Los bloques pueden verse como una matriz de tres dimensiones, un grid solo pueder ser visto como uno de dos dimensiones.

4.3. Métrica para medir el desempeño de algoritmos paralelos

4.3.1. Tasa de aceleración o Speedup

El desarrollo de programas paralelos no solo tiene como objetivo alcanzar el mayor desempeño posible, sino que este se mantenga a proporción, a medida que se agreguen procesadores.

Para poder tener una noción de la medida del desempeño de los algoritmos paralelos se definen las métricas aceleración o *speedup* y eficiencia, esto para evaluar su desempeño, en este trabajo se aborda solamente la primera métrica.

La tasa de aceleración de un programa paralelo, es la relación entre la versión paralela de un algoritmo, para una cantidad dada de procesadores, y el tiempo de ejecución de la versión secuencial del mismo algoritmo, esto considerando clusters de CPUs. Para poder calcularlo, necesitamos definir primero:

- $T_{Secuencial}$: Tiempo necesario para la ejecución de un programa serial.
- $T_{Paralelo}$: Tiempo necesario para la ejecución de la versión paralela del mismo programa.

Pacheco (2011) define a la tasa de aceleración de un programa paralelo ejecutándose en procesadores como:

$$S = \frac{T_{Secuencial}}{T_{Paralelo}} \tag{4.1}$$

Esta medida sirve para comparar el rendimiento del algoritmo paralelo con respecto al secuencial. Cuando $S = p$ (en donde p es el número de procesadores), se dice que se cuenta con un programa que tiene una tasa de aceleración lineal.

Esta tasa de aceleración lineal es el óptimo desde el punto de vista de la paralelización, puesto que indica una proporción directa y sin desperdicios entre el tiempo de ejecución y la cantidad de procesadores que intervienen. Es así que $0 \leq S \leq p$.

Cuanto mayor sea la tasa de aceleración de un algoritmo paralelo, mejor será la rendimiento del mismo en la ejecución con p procesadores.

En este trabajo de investigación la tasa de aceleración se calcula a partir de los tiempos que tardan los programas al ejecutarse en la CPU y la GPU.

A partir de la tasa de aceleración se calcula además el ahorro de tiempo en términos de porcentajes, a través de los siguientes pasos:

1. Obtenemos los tiempos de ejecución secuencial y paralelo de los programas, y con ello calculamos la tasa de aceleración.

2. Consideramos el tiempo de ejecución secuencial del programa como el 100 %, dado a que es el tiempo que usamos de referencia y deseamos disminuir.

3. Calculamos el porcentaje de tiempo que representa el tiempo de ejecución de la versión paralela del algoritmo esto con relación al tiempo secuencial.

4. Sustraemos el porcentaje de la versión paralela al porcentaje de la versión secuencial, el resultado es el ahorro de tiempo en términos de porcentaje, cuando el tiempo de ejecución de la versión secuencial del programa es menor al tiempo de ejecución paralelo este ahorro es negativo y por lo tanto no se realiza esta sustracción.

El objetivo es obtener una noción de cuánto tiempo se ahorra al utilizar la versión paralela del programa.

4.4. Resumen del capítulo

En este capítulo se definen conceptos del cómputo de propósito general en unidades de procesamiento gráfico. Un resumen de cómo el poder de procesamiento de las GPUs y CPUs se han ido distanciando, como la comunidad científica ha aprovechado este poder de cómputo para utilizarlo en sus investigaciones y acelerar algoritmos.

En específico, se aborda la plataforma desarrollada por NVIDIA, conocida como CUDA, algunas de sus propiedades y características tales como la estructura que siguen sus programas, y definiciones utilizadas para referirse a la programación utilizando su arquitectura, como lo son los grids, bloques, hilos y los tipos de memoria con los que cuenta la unidad de procesamiento gráfico.

Para finalizar con la métrica de la tasa de aceleración, utilizada para calcular el rendimiento entre la versión paralela y secuencial de un algoritmo, y el porcentaje de ahorro de tiempo al utilizar la versión paralela de un programa.

Capítulo 5

Desarrollo

En este capítulo se presenta el desarrollo del algoritmo del proceso de correlación morfológica y la del proceso de contrucción de filtro NSDF. Para la implementación de estos algoritmos se hicieron uso de las siguientes herramientas de software:

1. El lenguaje de programación C++, utilizando el compilador proporcionado en Microsoft Visual C++ 2012 en el sistema operativo Windows y GCC 4.6.3 en Ubuntu Linux, estos en conjunto con el compilador NVCC que se utiliza para compilar el código ejecutado en GPU, la ejecución de los experimentos y los resultados obtenidos se realizaron utilizando la plataforma Ubuntu Linux 12.04 LTS y GCC.

2. NVIDIA CUDA 5.0 una plataforma para desarrollo de aplicaciones de propósito general en unidades de procesamiento gráficos (cuyos detalles se mostraron en el capitulo anterior).

3. OpenCV versión 2.4.4, biblioteca de funciones multiplataforma para la visión artificial de código abierto desarrollado por la compañía Intel (Bradski and Kaehler, 2008), escrito en C++ y utilizada para la manipulación de imágenes.

El desarrollo y los experimentos, se llevaron a cabo utilizando un equipo de cómputo con las siguientes características: un procesador Intel Core i7 2600 Sandy Bridge con tecnología de 32 nm a 3.40 Ghz, dos memorias RAM Kingston DDR3 de 4 GB cada uno y una frecuencia de 532.1 MHz, y una tarjeta gráfica NVIDIA GeForce GTX 590 con una tecnología de 40 nm, con memoria GDDR5 de 1536 MB y un ancho de banda de 165.9 GB/s.

Para la compilación del código fuente, se configuraron en un archivo *Makefile* las opciones de compilador GCC de tal manera que redujera el tamaño del código y el tiempo de ejecución, las banderas o *flags* de optimización activadas son hasta el nivel de optimización $-O3$ el cual es la que ofrece la máxima optimización. Para las opciones ofrecidas por el compilador NVCC se eligió también el nivel de compilación $-O3$ la cual viene marcada por defecto y la arquitectura para 64 bits.

La tabla 5.1 presenta algunas de las especificaciones técnicas de la tarjeta de video utilizada y la versión de CUDA soportada.

En las siguientes secciones se presenta el desarrollo de las implementaciones secuenciales y paralelas del proceso de correlación morfológica, el proceso de construcción del filtro NSDF.

5.1. Proceso de correlación morfológica

La correlación morfológica está definida por la Ecuación 5.1, donde $h(k,l)$ es la imagen de referencia y $s(k,l)$ la imagen de prueba, ambos con Q niveles de gris (cuantización), $c(k,l)$ representa el plano de salida de la correlación no lineal en las coordenadas (k,l), $min(x,y)$ es el valor mínimo entre x y y. La suma es realizada sobre el vecindario-W.

$$c(k,l) = \sum_{k=m,n\in W} \left[\sum_{k=1}^{Q-1} [S^q(m+k,n+l) \cap h^q(m,n)] \right] \tag{5.1}$$

Tabla 5.1: Características de la tarjeta de video GeForce 590.

Device 1: GeForce GTX 590	
Versión Driver CUDA / Runtime Version	6.0 / 5.5
CUDA Capability Major/Minor version number	2.0
Cantidad total de memoria global	1536 MBytes
(16) Multiprocesadores (MP), (32) Núcleos CUDA por MP:	512 Núcleos CUDA
Frecuencia de reloj de la GPU:	1260 MHz (1.26 GHz)
Frecuencia de reloj de la memoria:	1728 Mhz
Ancho de bus de la memoria:	384-bit
Tamaño cache L2:	786432 bytes
Cantidad total de memoria constante:	65536 bytes
Cantidad total de memoria compartida por bloque:	49152 bytes
Número total de registros disponibles por bloque:	32768
Tamaño Warp:	32
Número máximo de hilos por multiprocesador:	1536
Número máximo de subprocesos por bloque:	1024
Dimensión del tamaño máximo de hilos en un bloque (x,y,z):	(1024, 1024, 64)
Dimensión del tamaño máximo de una malla (x,y,z):	(65535, 65535, 65535)
CUDA Driver = CUDART, NumDevs = 2, Device0 = GeForce GTX 590, Device1 = GeForce GTX 590	

La imagen de la Figura 5.1, muestra de manera mas ilustrativa este proceso. Como se puede notar, se de involucran una escena de prueba, un filtro de correlación y el plano de salida, illustrados en forma de matrices, y el dezplazamiento sobre las columnas y renglones para cubrir todos los píxeles de la imagen de prueba.

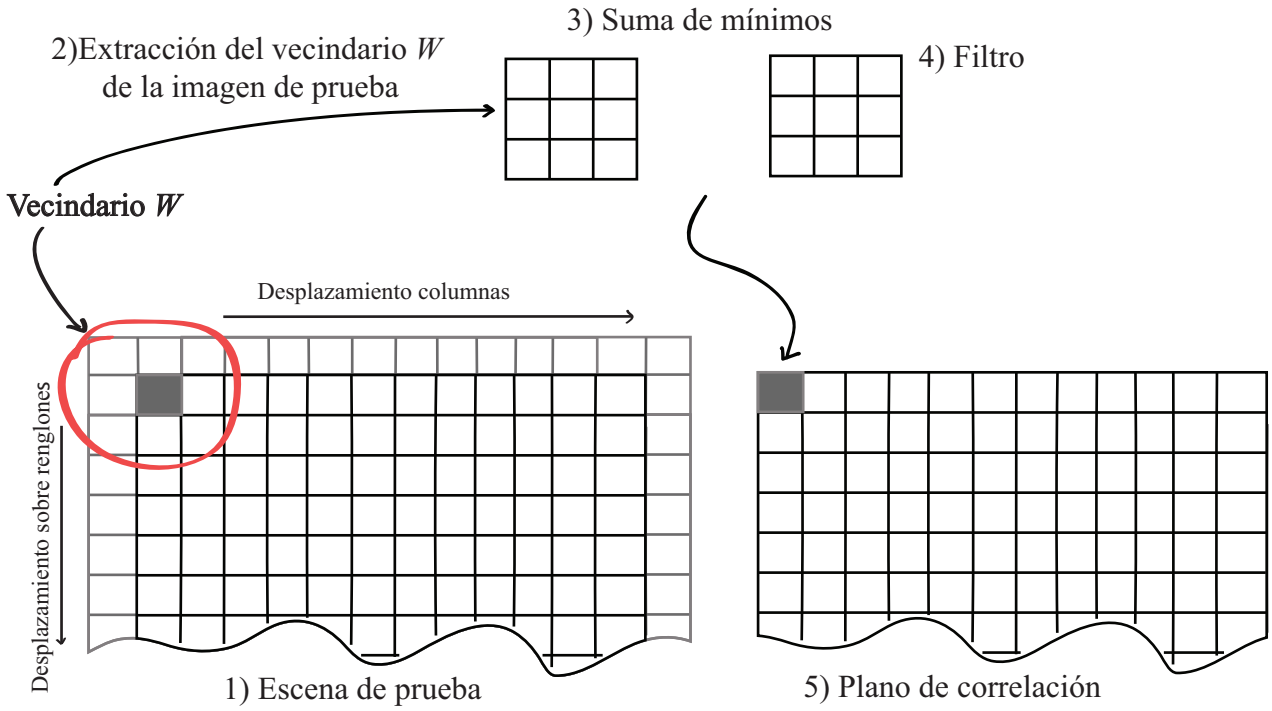


Figura 5.1: Vista general de la forma secuencial del proceso de correlación morfológica.

El Algoritmo 1 muestra los pasos que se siguieron para implementar este proceso en forma secuencial, proceso que sirvió de base para la versión paralela.

5.2. Proceso de construcción del filtro NSDF

El filtro se construye como una combinación lógica de imágenes de entrenamiento. Sea $T_i(k,l), i = 1, \ldots, N$ el conjunto de N imágenes de entrenamiento en escala de gris, que representan las posibles distorsiones que puede tener la imagen objetivo. El filtro compuesto no lineal (NSDF) se puede expresar como:

$$h(k,l) = \sum_{q=1}^{Q-1} \left[\bigcap_{i=1}^{N} T_i^q(k,l) \right] \tag{5.2}$$

donde $\cap_{i=1}^{N}$ representa la intersección lógica de las N imágenes binarias y $T_i^q(k,l), q = 1, \ldots, Q-1, i = 1, \ldots N$ son las imágenes binarias obtenidas por medio de la descom-

Algoritmo 1 Proceso de correlación morfológica secuencial.

Entrada: Leer $imagenPrueba$ y $filtro$.

1: Reservar memoria para el $planoSalida$.
2: Aplicar zero padding a $imagenPrueba$.
3: Hacer cf = columnas del filtro
4: Hacer rf = renglones del filtro
5: **para** $xPrueba$ = cf/2 hasta $numColsPrueba$ - cf/2 **hacer**
6: **para** $yPrueba$ = rf/2 hasta $numRensPrueba$ - rf/2 **hacer**
7: $suma = 0$.
8: **para** $xFiltro = 0$ hasta $numColsFiltro$ **hacer**
9: **para** $yFiltro = 0$ hasta $numRensFiltro$ **hacer**
10: Calcular $xImagenPrueba$ y $yImagenPrueba$.
11: Hacer $minimo = filtro(xFiltro, yFiltro)$.
12: **si** $minimo > imagenPrueba(xImagenPrueba, yImagenPrueba)$ **entonces**
13: Hacer $minimo = imagenPrueba(xImagenPrueba, yImagenPrueba)$.
14: **fin si**
15: $suma = suma + minimo$
16: **fin para**
17: **fin para**
18: $planoSalida(xPrueba, yPrueba) = suma$
19: **fin para**
20: **fin para**

posición por umbral para cada imagen de la clase verdadera.

La figura 5.2, muestra de forma ilustrativa la umbralización de la imagen de entrenamiento, como resultado de esto se producen 255 capas binarias.

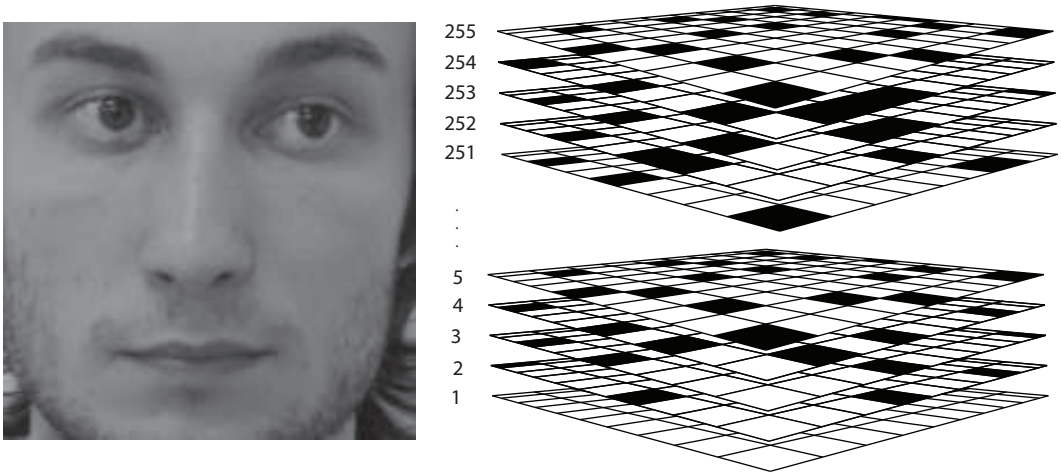


Figura 5.2: Umbralización de imagen de entrenamiento y sus capas binarias. Imagen tomada de Pesquisa et al. (2006).

La figura 5.3, muestra el ejemplo de como es realizada la construcción del filtro, la intersección es por cada capa q umbralizada de los conjuntos imágenes $X1$ y $X2$. La intersección en cada valor de q da como resultado una capa que contiene el valor 1 en donde hubo intersecciones y 0 en cualquier otro caso, el filtro h es el resultado de la suma de estas capas.

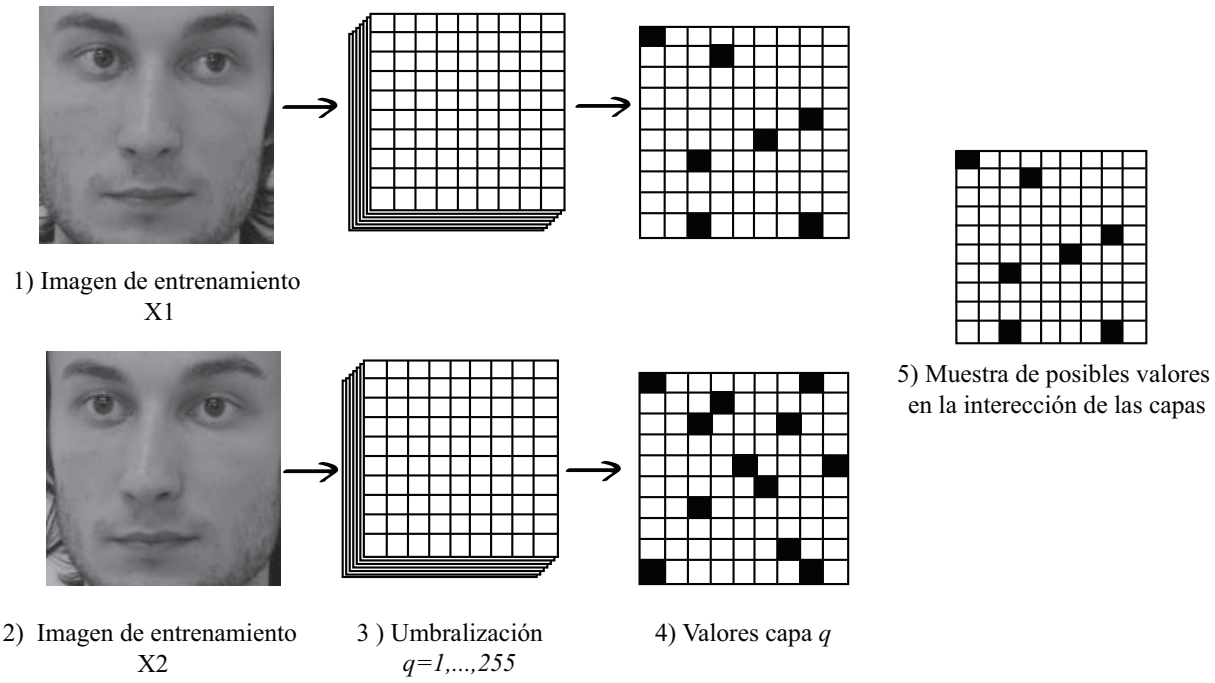


Figura 5.3: Proceso de construcción del filtro NSDF utilizando 2 imágenes de entrenamiento. Imágenes tomadas de Pesquisa et al. (2006).

El Algoritmo 2 muestra la secuencia que se siguió para implementar este proceso en forma secuencial, proceso que sirvió de base para la versión paralela.

5.3. Paralelización del proceso de correlación morfológica utilizando una GPU

El flujo correspondiente a la implementación del proceso de correlación no lineal se presenta en el Algoritmo 3. Como puede notarse, este algoritmo esta conformado principalmente de dos partes, la primera (pasos 1-6) es ejecutada de forma secuencial y la segunda (pasos 7-21) en la GPU.

La primera parte de este algoritmo, toma los datos de entrada, que son las imágenes $imagenPrueba$ y $filtro$, los pasos que la conforman son necesarios para asegurar que

Algoritmo 2 Proceso para la construcción del filtro NSDF.

Entrada: N imágenes de entrenamiento.

1: Calcular el tamaño de la imagen de entrenamiento.
2: Leer $X1$ =primer imagen de entrenamiento.
3: Reservar memoria para h.
4: **para** Cada imagen de entrenamiento **hacer**
5: Leer $X2$ = siguiente imagen de entrenamiento.
6: **para** $x = 0$ hasta $numColsX1$ **hacer**
7: **para** $y = 0$ hasta $numRensX1$ **hacer**
8: **para** $q = 1$ hasta 255 **hacer**
9: Hacer $y1 = falso$ y $y2 = falso$.
10: **si** $X1(x, y) > q$ **entonces**
11: Hacer $y1 = verdadero$.
12: **fin si**
13: **si** $X2(x, y) > q$ **entonces**
14: Hacer $y2 = verdadero$.
15: **fin si**
16: Hacer $h(x, y) = y1\&\&y2$.
17: **fin para**
18: **fin para**
19: **fin para**
20: Hacer $X1 = h$.
21: **fin para**

los datos de entrada y salida tengan las condiciones previas necesarias tales como la reserva de memoria para las variables, establecer el número de hilos, el tamaño de la malla y traslado de datos de memoria local a memoria en GPU para la llamada a la función $\_\_global\_\_\ correlacion()$ del paso 7.

La segunda parte de este algoritmo está constituido por la función ejecutada en GPU, dado que los datos se han copiado a la memoria global se tiene acceso a los datos de la imagen de prueba, al filtro y a la memoria para el plano de correlación de salida; en esta función se ejecuta un hilo por cada píxel de la imagen de prueba. Primero se localiza la posición (x, y) en la malla, estas coordenadas corresponden también a la posición en el píxel actual en $imagenPrueba$ y del plano de correlación de salida, entonces se realiza un desplazamiento a través de las columnas y los renglones $(xFiltro, yFiltro)$ que representa la posición de los pixeles del filtro y por cada desplazamiento calculamos la coordenada $(xImagenPrueba, yImagenPrueba)$ actual en la imagen de Prueba. Enseguida se obtiene el valor mínimo entre el $filtro(xFiltro, yFiltro)$ y la $IimagenPrueba(xImagenPrueba, yImagenPrueba)$. Finalmente se suma este valor a un acumulador que contiene la suma de los mínimos para este filtro, el acumulador es almacenado en la posición (x, y) calculado en el paso 8.

Como resultado se obtiene una matriz con todos los valores del plano de correlación que es copiada nuevamente de la GPU a memoria local, su posterior análisis permite localizar el pico de correlación.

La figura 5.4, muestra de manera illustrativa como es ejecutado un hilo por píxel de la imagen de prueba y como se almacenan los resultados de la suma de los mínimos por hilo en la matriz correspondiente a la salida de correlación.

Algoritmo 3 Implementación paralela del proceso de correlación morfológica.

Entrada: Leer $imagenPrueba$ y $filtro$.

1: Reservar memoria en GPU para el $planoSalida$, $imagenPrueba$ y $filtro$.
2: Aplicar zero padding a $imagenPrueba$.
3: Copiar los datos $planoSalida$, $imagenPrueba$ y $filtro$ a la GPU.
4: Calcular el número de hilos y el tamaño de la malla a partir del arreglo $imagenPrueba$.
5: realizar llamada a $\_\_global\_\_\ correlacion()$;
6: Copiar datos de vuelta de la GPU.
7: $\_\_global\_\_\ correlacion()$
8: Calcular posición (x, y) en el grid.
9: $suma = 0$.
10: **para** $xFiltro = 0$ hasta $numColsFiltro$ **hacer**
11: **para** $yFiltro = 0$ hasta $numRensFiltro$ **hacer**
12: Calcular $xImagenPrueba$ y $yImagenPrueba$.
13: Hacer $minimo = filtro(xFiltro, yFiltro)$.
14: **si** $minimo > imagenPrueba(xImagenPrueba, yImagenPrueba)$ **entonces**
15: Hacer $minimo = imagenPrueba(xImagenPrueba, yImagenPrueba)$.
16: **fin si**
17: $suma = suma + minimo$
18: **fin para**
19: **fin para**
20: $planoSalida(x, y) = suma$
21: **devolver** $planoSalida$

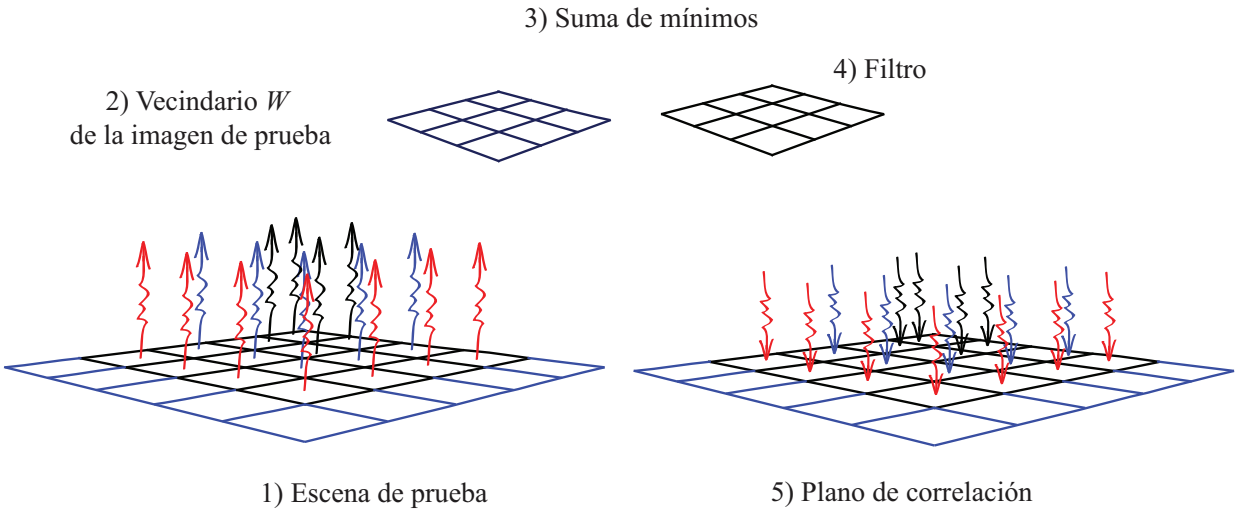


Figura 5.4: Vista general de la forma paralela del proceso de correlación morfológica, en donde cada flecha representa un hilo que ejecuta cálculos por píxel.

5.4. Paralelización del proceso de construcción del filtro NSDF

Esta sección muestra la implementación del filtro no lineal SDF, para ello se siguió el flujo mostrado en el Algoritmo 4. El cual está conformado también por dos partes.

La primera (pasos 1-11) toma como datos de entrada el conjunto de imágenes de entrenamiento y tiene como salida una matriz h que representa al filtro construido. Los pasos son ejecutados de forma secuencial utilizando la CPU y son necesarias para iterar sobre las imágenes de entrada, establecer el número de hilos, el tamaño de la malla, reservar tanto memoria local como en GPU y copiar los datos de memoria local a memoria de la GPU del par de imágenes de entrenamiento $X1$ y $X2$ actuales, en la línea 8 realiza la llamada a la función $\_\_global\_\_\ nsdf()$ cuyas instrucciones son ejecutados en la GPU.

La segunda parte de este algoritmo (pasos 12-25) es la función encargada de rea-

lizar los siguientes cálculos, se determinan las coordenadas del pixel actual (x, y) que indican el posicionamiento actual en las imágenes de entrenamiento $X1$, $X2$ y el filtro h; entonces cada hilo realiza la descomposición por umbral, esto a través de la variable q que nos indica el nivel de gris actual, este valor se incrementa desde $q = 1$ hasta 255 niveles, produciendo con esto 255 capas binarias por pixel en ambas imágenes; por cada nivel de gris o capa binaria se calcula si existe una intersección entre $X1(x, y)$ y $X2(x, y)$ esto a través de las variables $y1$ y $y2$ cuyos valores son de tipo booleano y se inicializan $falso$ y que solo toma el valor *verdadero* si se satisfacen las condiciones de los pasos 16 y 19, si estas condiciones se cumplen, significa que el valor de $X1(x, y)$ o $X2(x, y)$ son mayores al umbral q actual. Por último se almacena en $h(x, y)$ un 1 si existe la intersección y en cualquier otro caso se almacena un 0, h es la salida que representa nuestro filtro, este es copiada nuevamente a memoria local en donde toma el rol de $X1$ en dado caso de que exista otra imagen de entrenamiento.

La figura 5.5, muestra de manera ilustrativa como es ejecutado un hilo por píxel en las imágenes de entrenamiento, estos hilos son las encargadas de calcular las intersecciones entre las imágenes y almacenar la suma de las intercciones la matriz que representa el filtro.

Algoritmo 4 Algoritmo paralelo para implementar la construcción del filtro

Entrada: N imágenes de entrenamiento.

1: Calcular el tamaño de la imagen de entrenamiento.

2: Leer $X1$ =primer imagen de entrenamiento.

3: Reservar memoria en GPU para el $X1$,$X2$ y h.

4: **para** cada imagen de entrenamiento **hacer**

5: Leer $X2$ = siguiente imagen de entrenamiento.

6: Copiar datos $X1$, $X2$ del CPU a la GPU.

7: Calcular el número de hilos y el tamaño del grid a partir del tamaño de $X1$.

8: realizar llamada a $\_\_global\_\_\ nsdf()$;

9: Copiar datos h de GPU a memoria local.

10: Hacer $X1 = h$.

11: **fin para**

12: $\_\_global\_\_\ nsdf()$ {

13: Calcular posición (x, y) en el grid.

14: **para** $q = 1$ hasta 255 **hacer**

15: Hacer $y1 = falso$ y $y2 = falso$.

16: **si** $X1(x, y) > q$ **entonces**

17: Hacer $y1 = verdadero$.

18: **fin si**

19: **si** $X2(x, y) > q$ **entonces**

20: Hacer $y2 = verdadero$.

21: **fin si**

22: Hacer $h(x, y) = y1 \&\& y2$.

23: **fin para**

24: **devolver** h

25: }

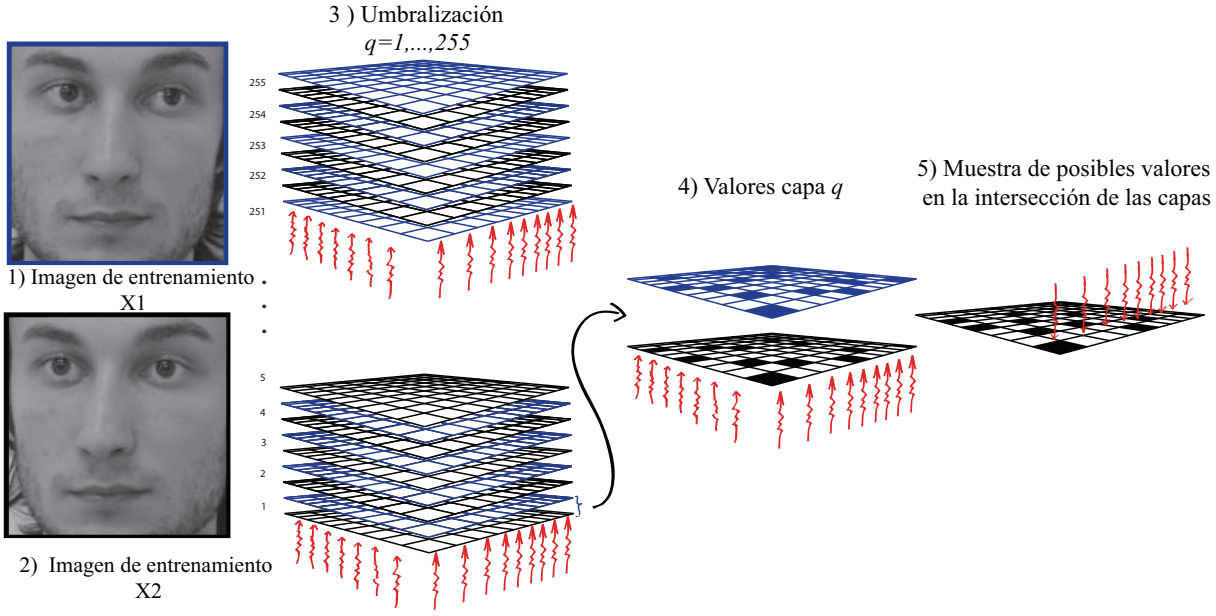


Figura 5.5: Proceso de construcción del filtro utilizando GPU. Imágenes tomadas de Pesquisa et al. (2006).

5.5. Resumen del capítulo

Este capítulo muestra la implementación secuencial y paralela del proceso de correlación morfológica y la construcción del filtro compuesto NSDF utilizando la plataforma CUDA, la librería OpenCV y el lenguaje de programación C++, el desarrollo se llevó a cabo en Linux y Windows.

Primero se implementaron las versiones secuenciales de estos algoritmos, que sirvieron de base para implementar sus versiones paralelas y así distribuir el cálculo de las operaciones entre los hilos que se crean al trabajar con las imágenes de entrada. Además se muestran las características y propiedades de la unidad de procesamiento gráfico utilizado en el desarrollo.

Capítulo 6

Experimentos y resultados

En este capítulo se presentan los experimentos realizados y resultados obtenidos al ejecutar los algoritmos del proceso de correlación morfológica y la del proceso de contrucción de filtro NSDF desarrollados en el capítulo anterior. Cada sección muestra los datos de su respectivo algoritmo.

6.1. Proceso de correlación morfólogica

Para probar la eficiencia del algoritmo del proceso de correlación morfológica se realizó un experimento, con 11 imágenes de distintos tamaños, en la que la imagen de prueba fue utilizada como filtro para detectar un pico de correlación en el centro del plano de correlación, la imagen de prueba fue la misma, variándose solo su tamaño, siendo el número de píxeles de la forma N^2, con $N = 4, ..., 4096$.

La tabla 6.1 muestra el número de píxeles de la imagen de prueba y el filtro, las primeras dos columnas nos muestran el tamaño del filtro y su número de píxeles, las siguientes columnas nos señalan el tamaño de la imagen de escena y el número de píxeles de la imagen de escena (con relleno).

El tamaño de la extensión del relleno en la imagen de prueba es definido por dos

Tabla 6.1: Tamaños de las imágenes de escena y su respectivo filtro.

Imagen	tamaño filtro	# píxeles imagen filtro	tamaño escena	# píxeles escena
1	4	16	8	64
2	8	64	16	256
3	16	256	32	1024
4	32	1024	64	4096
5	64	4096	128	16384
6	128	16384	256	65536
7	256	65536	512	262144
8	512	262144	1024	1048576
9	1024	1048576	2048	4194304
10	2048	4194304	4096	16777216
11	4096	16777216	8192	67108864

variables que son establecidas por la altura y anchura del filtro.

Como puede observarse, el relleno dobla el tamaño de la imagen de escena y por lo tanto el número de píxeles se cuadruplica, las figuras 6.1 y 6.2 nos muestran el ejemplo de cómo es aplicado un relleno que en este caso y para la correlación tiene que ser en forma de reflexiones de espejo sobre la misma imagen, tanto de la parte superior como la inferior, los lados izquierdo y derecho.

Figura 6.1: Imagen utilizada como filtro e imagen de prueba (original cortesía de MIT).

La Fig. 6.3 muestra el plano de correlación de salida en 2 y 3 dimensiones entre estas dos imágenes (6.1 y 6.2).

Figura 6.2: Relleno aplicado a la imagen de prueba (original cortesía de MIT).

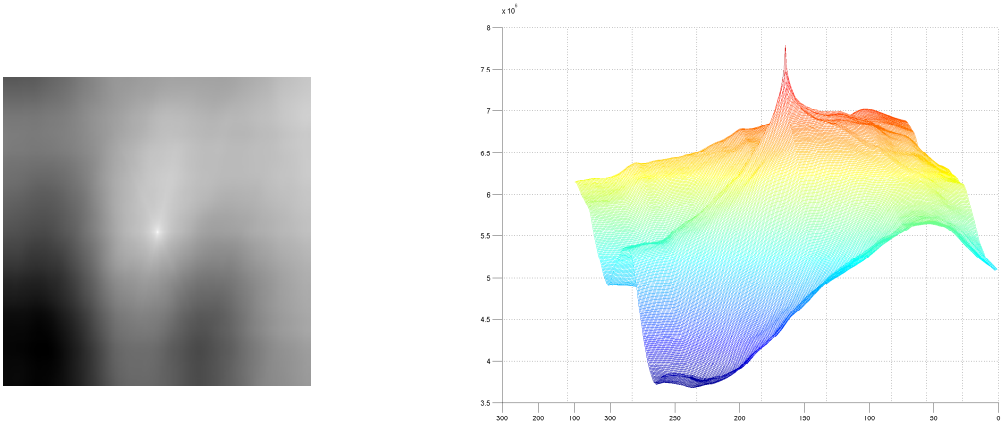


Figura 6.3: Plano de correlación entre las imágenes 6.1 y 6.2.

La tabla 6.2 resume los resultados obtenidos para este experimento, cuyo objetivo

fue determinar la tasa de aceleración alcanzada en cada uno de las ejecuciones, la primera columna representa el tamaño de la imagen, la segunda y tercera muestran los tiempos de ejecución en segundos obtenidos con las versiones secuencial y paralela del algoritmo. Como puede observarse, la tasa de aceleración varía durante la ejecución del algoritmo con los diferentes tamaños de las imágenes de prueba y filtro, siendo esta de 0.0343125 para una imagen de prueba y un filtro de 4×4 pixeles hasta alcanzar los 29.79108811 para un tamaño de 4096×4096 pixeles, es decir la versión paralela del algoritmo es hasta 29.79 veces más rápido que su versión secuencial, también se puede notar que mientras que las ejecuciones en la versión secuencial del algoritmo toma desde segundos hasta varios días, la versión paralela solo toma de segundos a unas cuantas horas.

Tabla 6.2: Comparación y tasa de aceleración obtenida entre el tiempo de ejecución de las implementaciones secuencial y paralela del algoritmo para el proceso de correlación, con una prueba de tamaño N.

Tamaño de la imagen	Versión secuencial	Versión paralela	Tasa de aceleración
4	0.0000011	0.000032	0.034
8	0.0000085	0.000045	0.19
16	0.00011	0.00014	0.78
32	0.0016	0.00047	3.28
64	0.026	0.0035	7.51
128	0.40	0.014	29.31
256	6.24	0.22	29.03
512	100.533	3.42	29.37
1024	1626.03	54.68	29.74
2048	25879.7	874.24	29.60
4096	416849	13992.41	29.79

A continuación se presentan las gráficas en donde se puede observar cómo evoluciona la tasa de aceleración, el eje de las abscisas representa el diferente tamaño de las

imágenes, el eje de las ordenadas nos muestra el tiempo en segundos, en amarillo se representa la evolución del tiempo de ejecución de la versión secuencial del algoritmo y en color verde su contraparte paralela. La gráfica 6.4 presenta los tiempos de ejecución de la imagen de prueba y filtro con un tamaño de 4×4 hasta 32×32 pixeles, se puede notar que a partir del tamaño de la imagen de 16×16 la versión paralela del algoritmo comienza a ser menor que su versión secuencial.

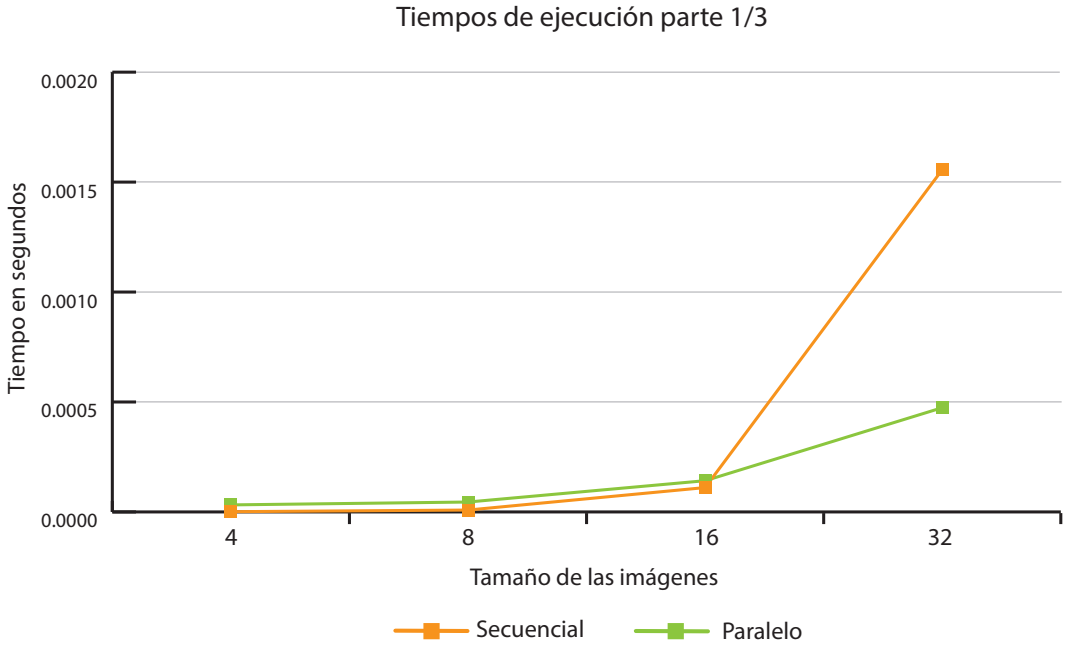


Figura 6.4: Tiempos de ejecución obtenidos utilizando imágenes con tamaño de 4×4 hasta 32×32 píxeles.

La Figura 6.5 presenta los tiempos de ejecución de la imagen de prueba y filtro con un tamaño de 64×64 hasta 256×256 pixeles, y es notable como el tiempo de ejecución de la versión secuencial del algoritmo es cada vez mayor, finalmente la tabla 6.6 muestra los tiempos de ejecución de las imágenes de tamaño 512×512 pixeles hasta 4096×4096 pixeles, aquí podemos notar que el tiempo de ejecución secuencial toma ya varias horas.

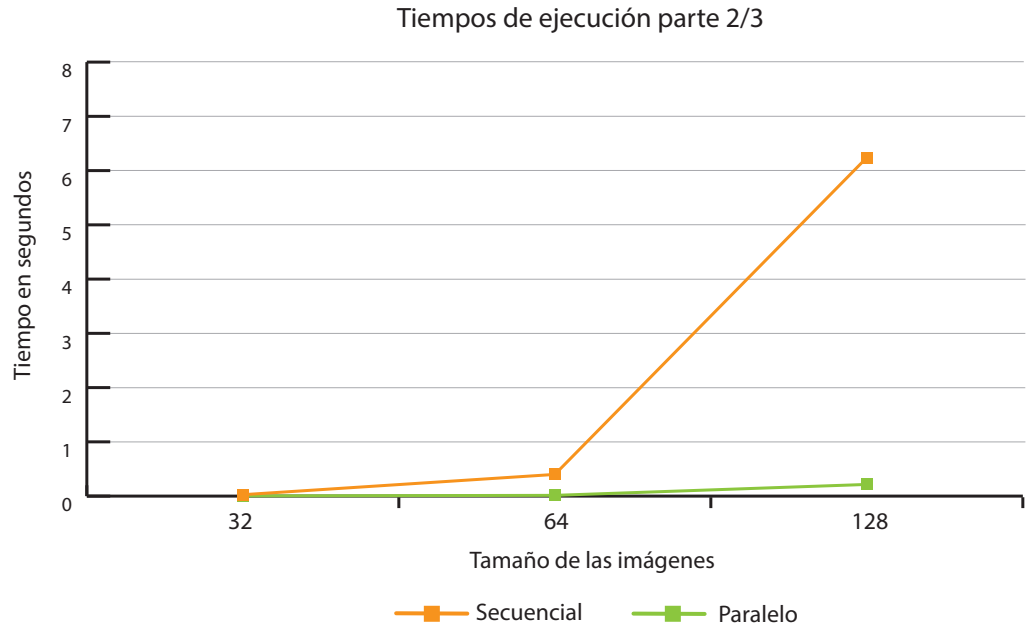


Figura 6.5: Tiempos de ejecución obtenidos utilizando imágenes con tamaño de 64×64 hasta 256×256 píxeles.

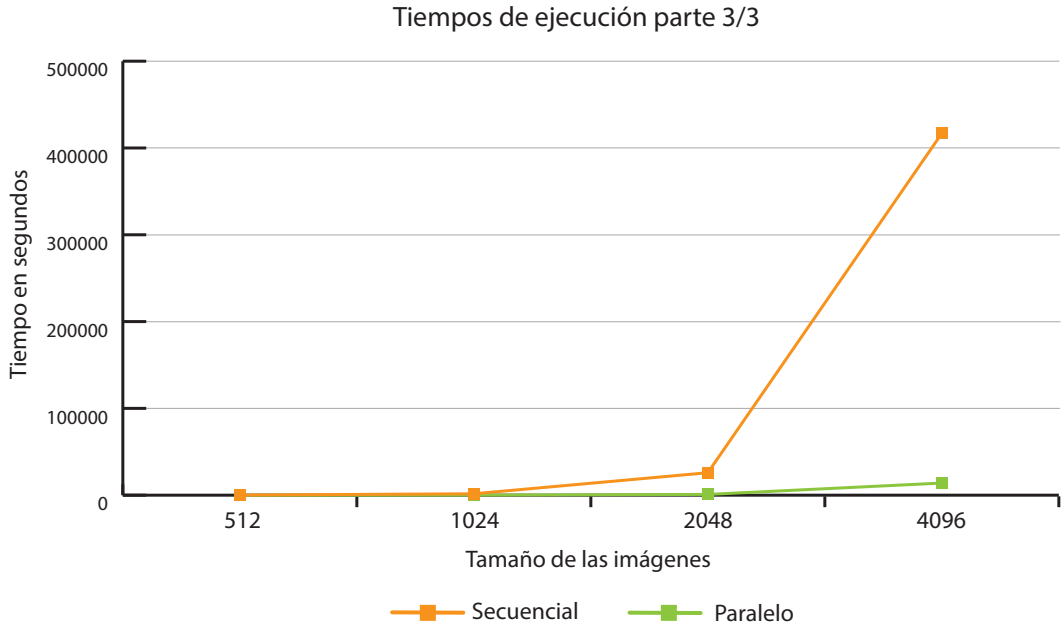
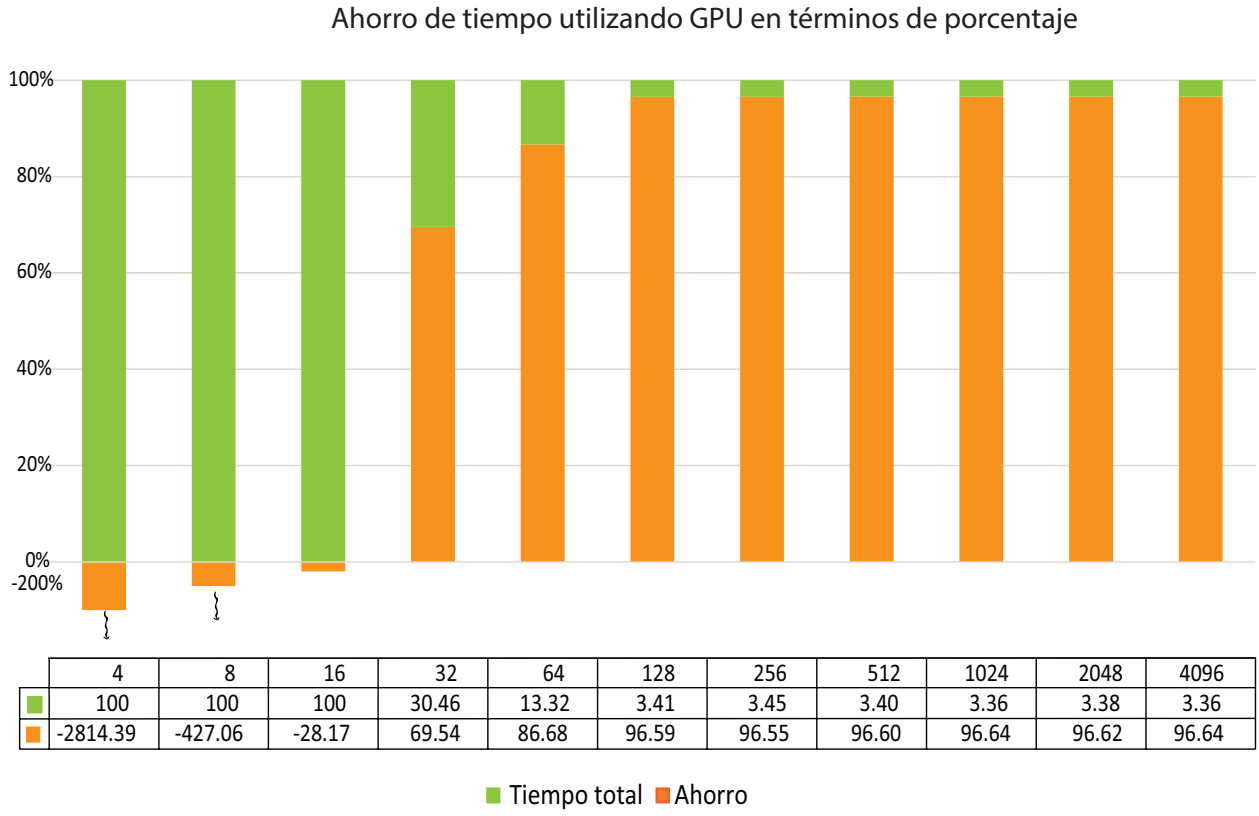


Figura 6.6: Tiempos de ejecución obtenidos utilizando imágenes con tamaño de 512×512 a 4096×4096 píxeles.

En la gráfica de barras 6.7 se muestra el porcentaje de ahorro al utilizar la forma paralela el algorimo del proceso de correlación para las 11 imágenes. En esta gráfica de barras de toma como un 100 % al tiempo de ejecución secuencial medido para cada imágen, como podemos notar para las imágenes cuadradas de 4×4 a 16×16 píxeles no existe tiempo de ahorro, incluso presenta porcentajes negativos dado a que la vesión secuencial presento un mejor rendimiento.



	4	8	16	32	64	128	256	512	1024	2048	4096
	100	100	100	30.46	13.32	3.41	3.45	3.40	3.36	3.38	3.36
	-2814.39	-427.06	-28.17	69.54	86.68	96.59	96.55	96.60	96.64	96.62	96.64

Figura 6.7: Comparación de tiempos de ejecución obtenidos en terminos de porcentajes.

6.2. Proceso de construcción del filtro NSDF

Para crear el conjunto de imágenes de entrenamiento de los experimentos, se utilizaron imágenes de la base de datos de laboratorio de Inteligencia Artificial de FEI en São Bernando do Campo, Brazil (Pesquisa et al., 2006), un ejemplo de esta base de datos

se muestra en la Figura 6.8, las imágenes originales son a color, tienen una dimensión de 640×480 pixeles y en ellas se representan rostros de personas de frente.

El conjunto de imágenes de entrenamiento consta de 200 imágenes, en la que cada imagen fue convertida a escala de gris y recortada a 180×180 píxeles de tal forma que solo se abarcaran las facciones faciales, la imagen 6.9 es una muestra del conjunto de entrenamiento creado.

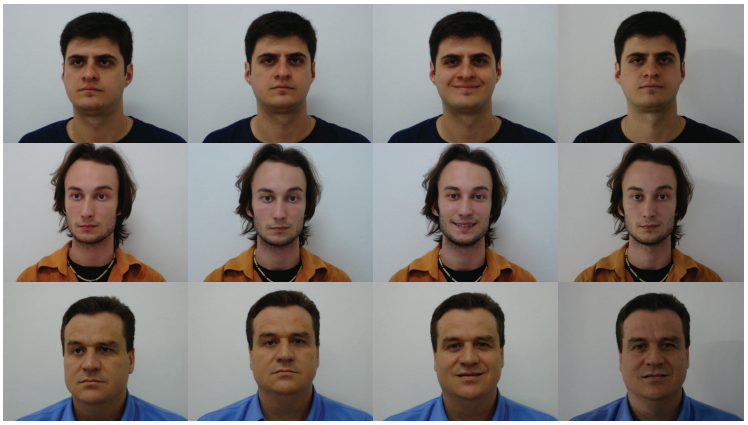


Figura 6.8: Muestra del conjunto de imágenes de la base de datos FEI (Pesquisa et al., 2006).

Figura 6.9: Ejemplo imágenes de entrenamiento faciales recortadas y en escala de gris (Pesquisa et al., 2006).

Para probar la eficiencia de este algoritmo se realizaron tres experimentos con conjuntos de imágenes de tamaño 180×180, 360×360 y 640×640 píxeles respectivamente. Cada experimento consistió en 10 ejecuciones del algoritmo con una cantidad de imágenes distinta, el número de imágenes con las que se construyeron los filtros son de 2, 5, 10, 15, 20, 25, 50, 100, 150 y 200, el objetivo fue determinar la tasa de aceleración alcanzada en cada uno de las ejecuciones.

Los resultados obtenidos se presentan en las tablas 6.3, 6.4 y 6.5 estas tablas siguen la misma nomenclatura, la primer columna representa la cantidad de imágenes de entrenamiento, la segunda y tercera columna se aprecian los tiempos de ejecución obtenidos en las versiones secuencial y paralela del algoritmo y finalmente la tasa de aceleración, la tabla 6.3 presenta los resultados obtenidos correspondientes al primer experimento, la tabla 6.4 para las imágenes de 360×360 y la tabla 6.5 para el último conjunto. Comparando estas tres tablas podemos notar que la tasa de aceleración se incrementa conforme mayor cantidad de imágenes de entrenamiento se utilicen para la construcción del filtro y que es levemente mejor cuando las imágenes son de una mayor dimensión. En cuanto a los tiempos de ejecución podemos notar que las versiones paralelas no sobrepasaron los 2 segundos, mientras que su contraparte secuencial tardo hasta 44.4 segundos.

A continuación se presentan las gráficas en donde se puede observar la tendencia de los tiempos de ejecución obtenidos en estos tres experimentos, el eje de las abscisas representa el número de imágenes de entrenamiento utilizado, el eje de las ordenadas nos muestra el tiempo en segundos, con línea azul se representa la evolución del tiempo de ejecución de la versión secuencial del algoritmo y la línea de color naranja su contraparte paralela. La Figura 6.10 presenta los tiempos de ejecución del primer experimento y las Figuras 6.11 y 6.12 las del segundo y tercer experimento. Como podemos notar en ninguna ejecución el tiempo de ejecución secuencial fue mejor al paralelo, por lo que cuando se incrementaba el número de imágenes de entrenamiento existe una dife-

Tabla 6.3: Comparación y tasa de aceleración obtenida entre el tiempo de ejecución de las implementaciones secuencial y paralela del algoritmo para el proceso de construcción del filtro, con imágenes de 180×180 píxeles.

# imágenes	V. Secuencial	V. paralela	Tasa de aceleración
2	0.019	0.0019	10.36
5	0.078	0.0047	16.51
10	0.18	0.0095	18.55
15	0.28	0.014	19.79
20	0.36	0.019	18.53
25	0.46	0.019	23.94
50	0.95	0.039	24.48
100	1.90	0.085	22.34
150	2.87	0.13	22.81
200	3.82	0.16	23.45

rencia de tiempo fué cada vez mayor, hay un ligero repunte con 25 y 50 imágenes de entrenamiento, esto en los tres experimentos.

Tabla 6.4: Comparación y tasa de aceleración obtenida entre el tiempo de ejecución de las implementaciones secuencial y paralela del algoritmo para el proceso de construcción del filtro, con imágenes de 360×360 píxeles.

# imágenes	V. Secuencial	V. paralela	Tasa de aceleración
2	0.073	0.0047	15.44
5	0.29	0.012	24.23
10	0.65	0.026	25.21
15	1.01	0.045	22.35
20	1.38	0.05	25.90
25	1.72	0.07	23.16
50	3.55	0.144	24.68
100	7.17	0.28	25.29
150	10.79	0.42	25.55
200	14.40	0.56	25.86

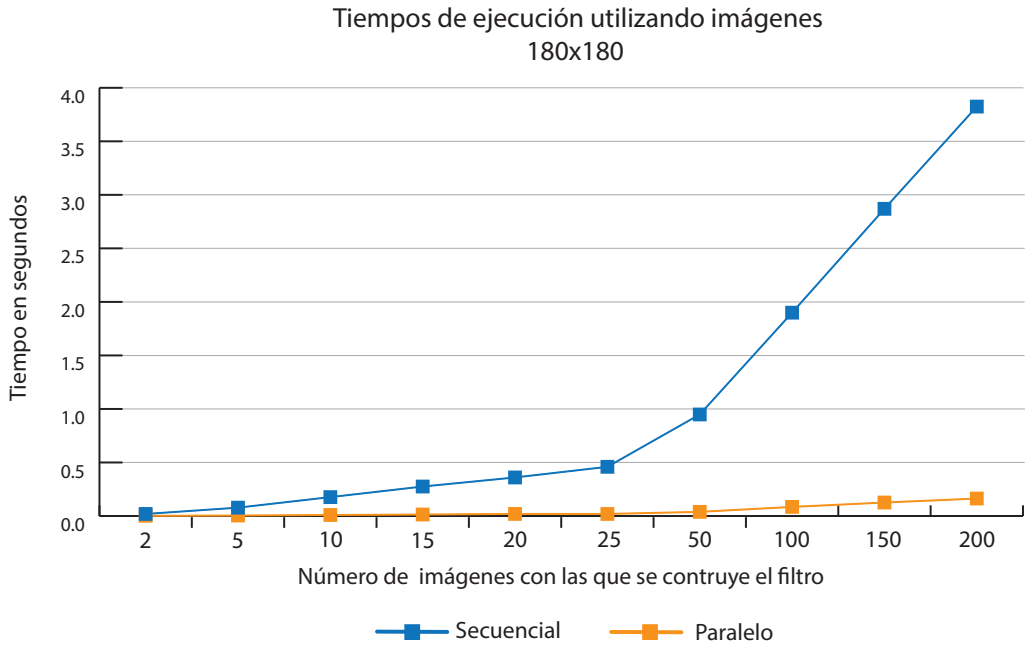


Figura 6.10: Tiempos de ejecución obtenidos en la contrucción del filtro utilizando imágenes de 180×180 píxeles.

Tabla 6.5: Comparación y tasa de aceleración obtenida entre el tiempo de ejecución de las implementaciones secuencial y paralela del algoritmo para el proceso de construcción del filtro, con imágenes de 640×640 píxeles.

# imágenes	V. Secuencial	V. paralela	Tasa de aceleración
2	0.23	0.014	16.85
5	0.89	0.042	21.04
10	2.02	0.085	23.74
15	3.10	0.13	24.46
20	4.24	0.17	24.87
25	5.30	0.21	25.15
50	10.95	0.42	25.98
100	22.12	0.85	26.11
150	33.28	1.27	26.17
200	44.45	1.71	26.05

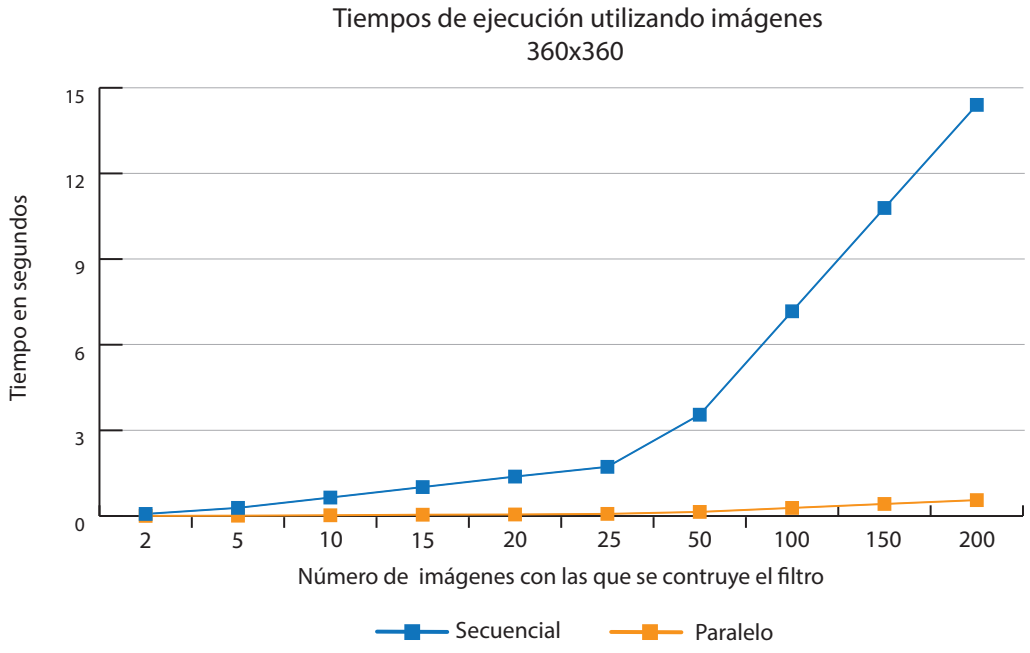


Figura 6.11: Tiempos de ejecución obtenidos en la contrucción del filtro utilizando imágenes de 360×360 píxeles.

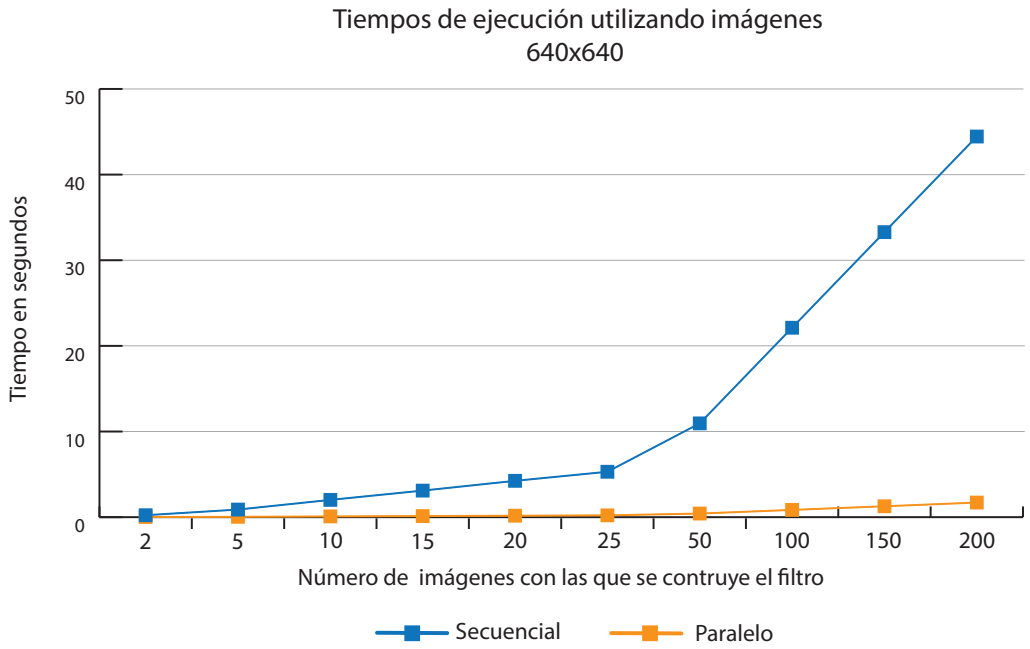


Figura 6.12: Tiempos de ejecución obtenidos en la contrucción del filtro utilizando imágenes de 640×640 píxeles.

En las figuras 6.7, 6.14 y 6.15 se muestra con gráficas de barras el porcentaje de ahorro al utilizar la forma paralela el algoritmo del proceso de construcción del filtro para los 3 grupos de imágenes con diferente dimensión. En estas gráficas de barras de toma como un 100 % al tiempo de ejecución secuencial medido para la cantidad de imágenes utilizadas para la contrucción, como se puede notar, el ahorro en el tiempo en porcentajes fue por encima de los 90 %.

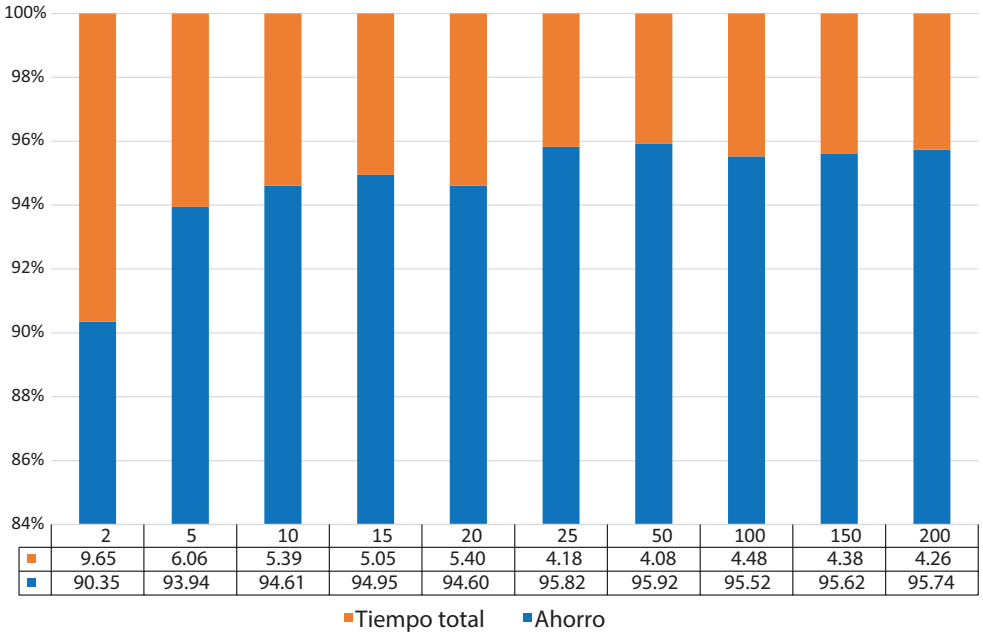


Figura 6.13: Comparación de tiempos de ejecución obtenidos en términos de porcentajes con imágenes de entrenamiento de 180×180 píxeles.

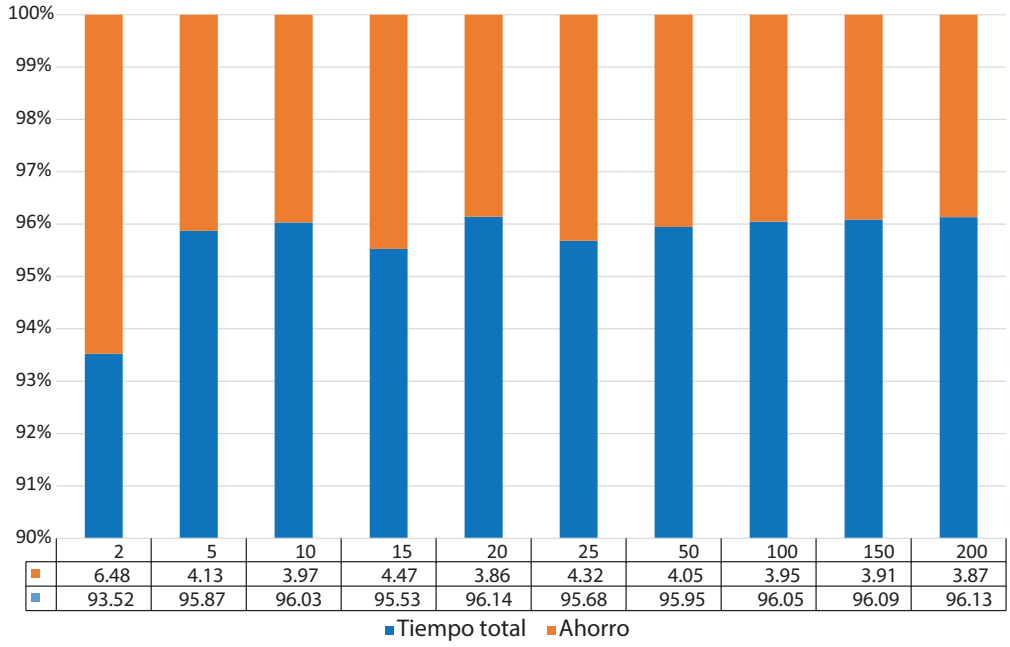


Figura 6.14: Comparación de tiempos de ejecución obtenidos en términos de porcentajes con imágenes de entrenamiento de 360×640 píxeles.

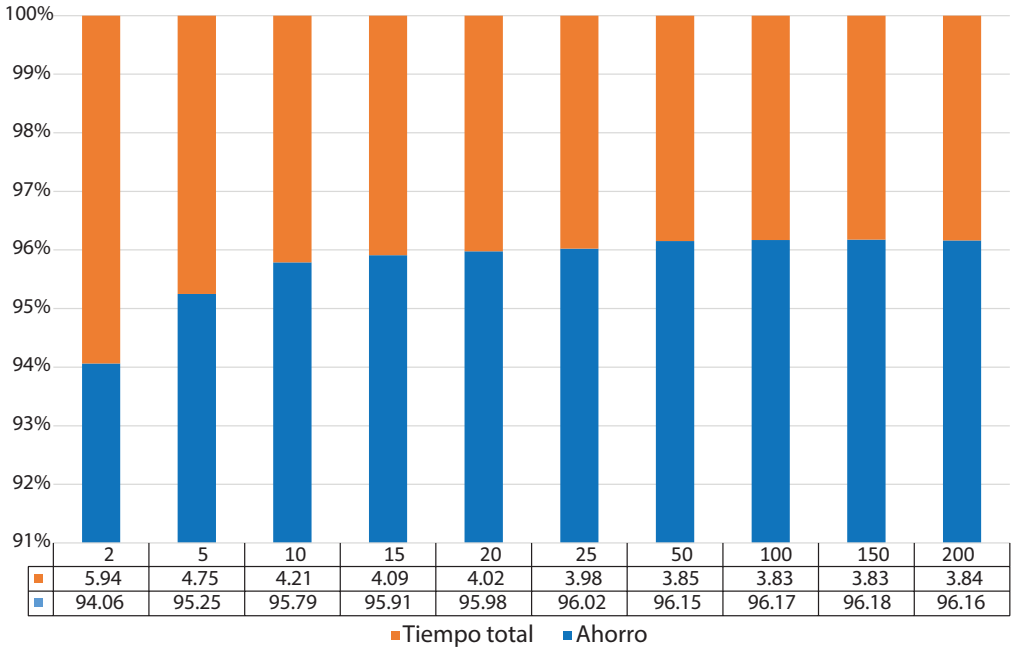


Figura 6.15: Comparación de tiempos de ejecución obtenidos en términos de porcentajes con imágenes de entrenamiento de 640×640 píxeles.

6.3. Resumen del capítulo

Este capítulo muestra los experimentos que se realizaron para medir las tasas de aceleración alcanzadas por los algoritmos paralelos del proceso de correlación y el proceso de construcción del filtro NSDF. La base de datos de imágenes utilizada fue la FEI, utilizando un conjunto de imágenes en escala de gris y con distintos tamaños.

Se presentan los resultados obtenidos al medir los tiempos de ejecución y calcular las tasas de aceleración, en tablas que muestran la información de cada ejecución con el tamaño de la imagen de escena y filtro, número de píxeles en la escena así como los resultados obtenidos en la ejecución tanto del proceso de correlación como la de creación del filtro y de forma visual se muestran graficas que ilustran la ventaja que ofrecen las versiones paralelas de estos algoritmos.

También se muestra el porcentaje de ahorro de tiempo, esto es si consideramos al

tiempo de ejecución secuencial de un algoritmo como un 100 % y colocamos en perspectiva el tiempo de ejecución del algoritmo paralelo en también términos de porcentaje, cubriendo una parte de ese 100 % obtenemos así el porcentaje de ahorro en ese algoritmo.

Capítulo 7

Discusión de resultados y conclusiones.

La mayoría de los algoritmos de reconocimiento y detección facial en imágenes, basados en correlación presentan un tiempo de ejecución bastante elevado si son comparados con sus contrapartes paralelas, aunque generalmente esto depende del equipo de cómputo en donde se ejecuten y el enfoque que se utiliza.

En este trabajo de investigación se abordó el algoritmo del proceso de correlación morfológica junto con la construcción del filtro NSDF aplicado a la detección y reconocimiento facial con el fin de comparar sus tiempos de ejecución.

En ese contexto los resultados obtenidos en este trabajo se enumeran a continuación:

1. La implementación de las versiones secuencial y paralelo de la operación de correlación morfológica.

2. La implementación de las versiones secuencial y paralela del algoritmo de síntesis del filtro NSDF.

3. Análisis comparativo del desempeño, en términos de la métrica de la tasa de aceleración, entre las versiones secuencial y paralela de la operación de correlación

morfológica.

4. Análisis comparativo del desempeño, en términos de la métrica de la tasa de aceleración, entre las versiones secuencial y paralelo del filtro de correlación NSDF.

Los elementos 1 y 2 de esta lista consisten en la implementación secuencial del proceso de correlación morfológica y del proceso de construcción del filtro NSDF, cuyo fin fue obtener las bases y proponer una forma de realizar la implementación de las versiones paralelas de estos dos algoritmos utilizando una GPU y así reducir sus tiempos de ejecución y minimizar la carga de operaciones calculadas cuando este se ejecuta de forma secuencial, distribuyendo esta carga a través de hilos que son ejecutados de forma paralela.

Para el proceso de correlación, la cantidad de operaciones calculadas para una imagen de prueba con dimensiones $N \times N$ y un filtro de dimensiones $M \times M$ en la versión secuencial pasó de $N^2 \times M^2$ a M^2 por hilo en su versión paralela, debido a que se eliminaron las iteraciones realizadas sobre las filas y columnas de la imagen de prueba, lanzando un hilo por píxel en la unidad de procesamiento gráfico, cabe mencionar que aunque los experimentos fueron realizados con imágenes con imágenes con una anchura y longitud igual, el algoritmo también funciona con imágenes de distintas dimensiones.

Para el proceso de construcción del filtro NSDF, la cantidad de operaciones calculadas para una imagen de entrenamiento de dimensiones $N \times N$ en la versión secuencial pasó de N^2 a 1 por hilo en su versión paralela, debido a que se eliminaron las iteraciones realizadas sobre las filas y columnas de la imagen de entrenamiento, lanzando un hilo por píxel en la unidad de procesamiento gráfico y solo iterando sobre las siguientes imágenes de entrenamiento existentes.

Los elementos 3 y 4 de esta lista consisten en el análisis del tiempo de ejecución medido para el proceso de correlación morfológica y del proceso de construcción del filtro

NSDF, cuyo fin fue calcular la tasa de aceleración alcanzada, al comparar los tiempos medidos presentados en las versiones secuenciales y paralelas de estos algoritmos.

Para el proceso de correlación, los tiempos medidos para imágenes pequeñas (4×4 hasta 32×32) fueron menores que los presentados por la versión paralela debido a que el tamaño de las imágenes es pequeño y el tiempo que se toma mover los datos en de memoria local a memoria en GPU es mayor a realizar las operaciones de forma local utilizando la CPU, a partir de las imágenes de 64×64 fue que esta tendencia cambio y los tiempos de ejecución de la versión paralela se mantuvieron menores alcanzando una tasa de aceleración de 29.79 con imágenes de 4096×4096.

Para el proceso de construcción del filtro NSDF, los tiempos medidos se realizaron sobre dos variaciones en los experimentos, la primera de ella tiene que ver con la dimensión de las imágenes de entrenamiento y la segunda con número de imágenes de entrenamiento utilizadas para la construcción del filtro. Por lo que se presentaron dos variaciones en las tasas de aceleración obtenidas, estas crecen cuando se aumenta la cantidad de imágenes de entrenamiento y cuando se utilizan imágenes con dimensiones (ancho, alto) cada vez más grandes. En ninguna medición la versión secuencial presentó un tiempo de ejecución menor a su versión paralela.

Se proponen como trabajo a futuro las siguientes tareas:

1. Explorar la tasa de aceleración que presentan estos algoritmos utilizando memoria distinta al global en la tarjeta de gráfica, tal como la memoria textura.

2. Verificar y utilizar concurrencia de ser posible en el proceso de construcción del filtro al calcular la suma de mínimos, de tal forma que pueda ser lanzado un grid del tamaño del filtro y que los hilos escriban en un mismo espacio de memoria que es el utilizado por el acumulador.

3. Añadir discriminación a objetos de clase falsa al proceso de construcción del filtro compuesto, es decir el filtro solo considera objetos de la clase verdadera.

Bibliografía

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'reilly.

Chellappa, R., Wilson, C., and Sirohey, S. (1995). Human and machine recognition of faces: a survey. *Proceedings of the IEEE*, 83(5):705–741.

Cook, S. (2012). *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*. Morgan Kaufmann.

Devrari, K. and Kumar, K. (2011). Fast Face Detection Using Graphics Processor. *International Journal of Computer Science and Information Technologies*, 2(3):1082–1086.

Díaz Martínez, S. (2008). *Métodos no lineales, invariantes a distorsiones geométricas e iluminación para el reconocimiento de patrones e imágenes*. Tesis doctorado, Centro de Investigación Científica y de Educación Superior de Ensenada.

Fitch, J., Coyle, E., and Gallagher, N. (1984). Median filtering by threshold decomposition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(6):1183–1188.

Gonzalez, R., Woods, R., and Eddins, S. (2009). Digital image processing using MATLAB. page 827.

González-Fraga, J. and Kober, V. (2006). Detección de objetos parcialmente ocultos por métodos de correlación. In *Memorias del Primer Congreso Internacional de Ciencias Computacionales*, volume 1, pages 83–87.

Hefenbrock, D., Oberg, J., Thanh, N. T. N., Kastner, R., and Baden, S. B. (2010). Accelerating Viola-Jones Face Detection to FPGA-Level Using GPUs. In *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 11–18, Charlotte, NC. IEEE.

Hester, C. F. and Casasent, D. (1980). Multivariant technique for multiclass pattern recognition. *Applied optics*, 19(11):1758–61.

Hjelmå s, E. and Low, B. K. (2001). Face Detection: A Survey. *Computer Vision and Image Understanding*, 83(3):236–274.

Javidi, B., Wang, W., and Zhang, G. (1997). Composite Fourier-plane nonlinear filter for distortion-invariant pattern recognition. *Optical Engineering*, 36(10):2690.

Kirk, D. and Wen-mei, W. (2010). *Programming massively parallel processors: a hands-on approach*. Morgan Kaufmann, 2nd edition.

Kriegman, D. and Ahuja, N. (2002). Detecting faces in images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58.

Kumar, B. and Savvides, M. (2006). Correlation Pattern Recognition for Face Recognition. *Proceedings of the IEEE*, 94(11):1963–1976.

Kumar, B. V. K. V., Mahalanobis, A., and Juday, R. D. (2005). *Correlation pattern recognition*. Cambridge University Press.

NVIDIA (2013). What is GPU acelerated computing?

Pacheco, P. (2011). *An introduction to parallel programming*. Morgan Kaufmann, Burlington,MA.

Park, I. K., Germann, M., Breitenstein, M. D., and Pfister, H. (2009). Fast and automatic object pose estimation for range images on the GPU. *Machine Vision and Applications*, 21(5):749–766.

Pesquisa, P. D., Leonel, L., and Junior, D. O. (2006). Relatório Final Captura e Alinhamento de Imagens : Um Banco de Faces Brasileiro. Technical report, Centro Universitário da FEI, São Bernardo do Campo, São Paulo, Brazil.

Pitas, I. and Venetsanopoulos, A. N. (1990). *Nonlinear Digital Filters*, volume 84. Springer US, Boston, MA.

Proakis, J. G. and Manolakis, D. G. (2012). *Tratamiento digital de señales*. Prentice-Hall, Madrid, 4th edition.

Turin, G. (1960). An introduction to matched filters. *IEEE Transactions on Information Theory*, 6(3):311–329.

VanderLugt, A. (1964). Signal detection by complex spatial filtering. *IEEE Transactions on Information Theory*, 10(2):139–145.

Vega Aquino, E. A. (2011). *Detección y seguimiento de rostros*. Tesis licenciatura, Universidad Carlos III de Madrid.

Vijaya Kumar, B. V. K., Savvides, M., Xie, C., Venkataramani, K., Thornton, J., and Mahalanobis, A. (2004). Biometric verification with correlation filters. *Applied optics*, 43(2):391–402.

Viola, P. and Jones, M. J. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154.

Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face Recognition: A Literature Survey. *ACM Computing Surveys*, 35(4):399–458.