Centro de Investigación Científica y de Educación Superior de Ensenada

PROCESAMIENTO PARALELO DE IMAGENES CON FILTROS MORFOLOGICOS

TESIS

MAESTRÍA EN CIENCIAS

ARIEL ARTURO QUEZADA PINA

Ensenada Baja California, México Agosto del 2007

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA

PROGRAMA DE POSGRADO EN CIENCIAS
EN CIENCIAS DE LA COMPUTACIÓN

PROCESAMIENTO PARALELO DE IMÁGENES CON FILTROS MORFOLÓGICOS

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de
MAESTRO EN CIENCIAS

Presenta:

ARIEL ARTURO QUEZADA PINA

Ensenada, Baja California, México, Agosto del 2007.

TESIS DEFENDIDA POR

Ariel Arturo Quezada Pina

Y APROBADA POR EL SIGUIENTE COMITÉ

Dr. Andrei Tchernykh
Codirector del Comité

Dr. Vitaly Kober
Codirector del Comité

Hugo Homero Hidalgo Silva
Miembro del Comité

Dr. Josué Álvarez Borrego
Miembro del Comité

Dr. Gilberto López Mariscal
Coordinador del programa de posgrado en Ciencias de la Computación.

Dr. Edgar Gerardo Pavía López
Director de Estudios de Posgrado

22 de Agosto de 2007

Resumen de la tesis de **Ariel Arturo Quezada Pina,** presentada como requisito parcial para la obtención del grado de **MAESTRO EN CIENCIAS** en **CIENCIAS DE LA COMPUTACIÓN.** Ensenada, Baja California. Agosto del 2007.

PROCESAMIENTO PARALELO DE IMÁGENES CON FILTROS MORFOLÓGICOS.

Resumen aprobado por:

Dr. Andrei Tchernykh
Codirector de Tesis

Dr. Vitaly Kober
Codirector de Tesis

En este trabajo se presenta el diseño, implementación y análisis de filtros morfológicos para procesamiento de imágenes. Los filtros morfológicos son filtros no lineales, basan su funcionamiento en teoría de conjuntos y realizan las operaciones sobre la imagen de forma local en base a un elemento estructural. Las operaciones básicas de filtros morfológicos son la erosión y la dilatación. A partir de estas operaciones se pueden realizar otras más complejas y con ellas una gran variedad de aplicaciones como filtrado de ruido impulsivo, esqueleto morfológico, detección y localización de objetos, búsqueda de contornos y transformada *top hat*. Los filtros morfológicos procesan imágenes binarias. Para emplearlos en imágenes en escala de gris, se muestra el uso de la propiedad de apilamiento. Ésta consiste en descomponer una imagen en diferentes planos binarios para llevar a cabo el procesamiento con filtros morfológicos para imágenes binarias. Terminado el procesamiento binario, se emplea la misma propiedad para obtener la imagen en escala de gris filtrada. Los algoritmos discutidos se implementaron utilizando lenguaje C y la programación paralela se llevó a cabo con biblioteca de funciones basadas en MPI. Éste es un estándar de facto, bajo el paradigma de paso de mensajes que permite portabilidad de código entre diferentes sistemas paralelos, como supercomputadoras y cluster. El esquema de balanceo de cargas utilizado fue maestro-esclavo. El sistema propuesto utiliza técnicas de alto y bajo nivel de paralelización. En alto nivel se empleó segmentación de imagen de entrada, traslape y distribución de imágenes binarias. En bajo nivel se utilizó una técnica llamada nivel de instrucción, la cual hace uso de todos los bits disponibles por palabra en los registros del procesador utilizado. Se presentan resultados de experimentos en dos sistemas diferentes: en Origin2000 y en Calafia. El primero posee modelo de memoria compartida-distribuida y el segundo de memoria distribuida. Ambos sistemas pertenecen a la red de supercómputo de CICESE. Se presenta un análisis de complejidad de los distintos algoritmos implementados.

También, los cálculos de desempeño de los algoritmos ejecutados en los experimentos y análisis de los mismos. En base a los resultados obtenidos se concluyó que el sistema algorítmico propuesto presenta características satisfactorias para el desarrollo de aplicaciones de filtros morfológicos con cómputo paralelo.

Palabras clave: Filtros Morfológicos, MPI, Programación Paralela, Cluster.

Abstract of the Thesis of **Ariel Arturo Quezada Pina**, presented as partial requirement to obtain the **MASTER in SCIENCES** degree in **COMPUTER SCIENCES**. Ensenada, Baja California. August of 2007.

PARALLEL IMAGE PROCESSING USING MORPHOLOGICAL FILTERS.

In this work is presented the design, implementation and analysis of morphologic filters for image processing. The morphologic filters are nonlinear filters, base their operation on set theory and conduct the operations on the image of local form on the basis of a structural element. The basic operations of morphologic filters are erosion and dilation. From these operations more complex operations can be made and with them a great variety of applications like impulsive noise filtering, morphologic skeleton, detection and location of objects, search of contours and top hat transform. The morphologic filters process binary images. In order to use them in gray scale images, the stacking property was used. This one consists of decomposing an image in different binary planes to carry out the processing with morphologic filters for binary images. Finished the binary processing, the same property was used to recompose the filtered image in gray scale. The algorithms discussed were implemented using language C and the parallel programming was carried out with function library based on MPI. This one is a standard de facto, under the message passing paradigm that allows portability of code between different parallel systems, like supercomputers and cluster. The scheme for load balancing for the processes was master-slave. Techniques of high and low level of parallel programming were used. In high level was used segmentation of image, overlapping and distribution of binary images. In low level, a technique called instruction level was used, which makes use of all the bits available in a word in the registers of the used processor. Experiments were made in two different systems: in Origin2000 and Calafia. The first system has shared-distributed model memory and the second distributed memory model. Both systems belong to the network of supercomputing of CICESE. An analysis of complexity of the different implemented algorithms is shown. Calculations of performance of the algorithms executed in the experiments and analysis of such were made. On the basis of the obtained results we concluded that the proposed algorithmic system presents satisfactory characteristics for the development of applications of morphologic filters with parallel computing.

Keywords: Morphological Filters, MPI, Parallel Programming, Cluster.

Dedicado

A mi Mamá, América Pina Palacios

De Diodoman

Agradecimientos

A Dios y mi familia, por estar siempre conmigo. Mi Amá, Nando, Dora y Lilián. A mis cuñados Lidia y Robert. A mis sobrinos: Fernando Isaac, Liria Estéfany, Camila, Rebeka y Leo. A mi "Brother" Rogelio Cano Cetina.

Al CICESE y CONACYT, por permitirme realizar mis estudios de posgrado y por su apoyo durante estos dos años.

A mi comité de tesis. Al Dr. Andrei Tchernykh por invitarme a trabajar con él. Al Dr. Vitaly Kober por su paciencia y estar siempre al pendiente durante todo este trabajo de tesis. Al Dr. Hugo Hidalgo por sus consejos especialmente en cada inicio de cursos. Al Dr. Josué Álvarez Borrego, por sus recomendaciones más allá del aspecto académico.

A Jorge Niebla, Lidia y Carolina, quienes se encargan de que todo funcione en el Departamento y me sacaron de más de un apuro. Muchas Gracias.

A mis compañeros de maestría, de Computación y de Electrónica. TODOS.

A los Scouts, al grupo Capoeira Brasil, los oceanólogos, a la Legión Extranjera, al Foro de Electrónicos ITSon y a todos los que en algún momento estuvieron aquí.

A Nadia Elisa Suárez Bosché, mi compañera y amiga en buena parte de esta etapa.

Contenido

Contenido(continuación)

Lista de Figuras

Página

Lista de Figuras(continuación)

Página

Lista de Tablas

Página

Primera Sección

Fundamentos Teóricos

Capítulo I

Introducción

I.1 Procesamiento y Análisis de Imágenes

El procesamiento y análisis de imágenes es importante ya que es común tener como respuesta de un sistema de información o instrumento una imagen. El procesamiento de imágenes puede llevarse a cabo con una diversidad de objetivos. Entre ellos tenemos mejora de características para entendimiento humano, compresión de información y otros. En cambio, el análisis de imágenes tiene como fin la extracción de características de los objetos contenidos en la imagen, localización de objetos, clasificación y reconocimiento de patrones, por mencionar algunos.

I.1.1 Representación de imágenes

En nuestro contexto, una imagen es una representación digital de una escena [Marion, 1991]. Cada uno de los puntos que componen la imagen, o píxeles, tienen asociado una posición y un valor. El tamaño del vector que representa la posición varía si la imagen es de 2 o 3 dimensiones, a 2 o 3 elementos respectivamente. El valor de cada píxel varía según el tipo de representación de la imagen. Así, se pueden tener imágenes binarias, en escala de gris y a color. Si la imagen es binaria, el valor que puede tomar el píxel es 1 o 0. Si la imagen es en escala de gris, el valor que puede tomar el píxel estará dado por la escala de cuantificación utilizada. Por ejemplo, es común utilizar un byte para la

representación del valor de un píxel. De esta forma, el píxel puede tomar valores que van desde 0 a 255. Si la imagen es a color, es común representar el valor de cada píxel como un vector de tres elementos, donde cada elemento representa la composición de los colores verde, azul y rojo, respectivamente, de dicho píxel [González, 2002]. Este modelo de representación de una imagen a color se le conoce como *RGB*.

I.1.2 Filtros

El filtrado de una imagen es la operación más común en procesamiento de imágenes [Marion, 1991]. Los filtros se emplean en una gran cantidad de aplicaciones. Entre ellas se encuentran: remoción de ruido impulsivo, remoción de ruido aditivo, mejoramiento de contrastes y otras.

Los filtros pueden clasificarse en dos grandes grupos:

- Lineales
- No lineales

Los filtros lineales son aquéllos que cumplen con las propiedades de proporcionalidad y superposición. Éstos han tenido gran aceptación en aplicaciones de procesamiento de imágenes porque han demostrado ser efectivos y son relativamente fáciles de implementar. Sin embargo, existen aplicaciones en las cuales los filtros lineales no resuelven adecuadamente los problemas que se le presentan. De ahí que la teoría de filtros se haya extendido al área de filtros no lineales, los cuales han demostrado ser más robustos en aplicaciones de procesamiento de imágenes. Este tipo de filtros se distingue porque no cumple con las propiedades de proporcionalidad y

superposición. De esta forma, la teoría matemática que justifica sus bases es más complicada que su contraparte lineal. Entre este tipo de filtros tenemos: filtros de mediana, filtros de rango, filtros de pila y filtros morfológicos.

I.2 Cómputo Paralelo

I.2.1 Paralelismo en Procesamiento de Imágenes

Las áreas de procesamiento de imágenes digitales y visión por computadora han exhibido un gran crecimiento en las décadas pasadas [Pitas, 1993]. Este proceso ha mostrado la necesidad de procesamiento rápido y confiable. También, ha aprovechado el paralelismo en las tareas de procesamiento de imágenes y en la naturaleza de las imágenes digitales. Así, se pueden identificar dos dominios en el paralelismo en procesamiento de imágenes digitales: dominio de datos y dominio de operaciones.

Dentro del dominio de datos podemos distinguir diferentes tipos de paralelismo. Uno de ellos consiste en la segmentación de la imagen de entrada. Las imágenes digitales son muestreadas comúnmente en una rejilla rectangular, y almacenadas en un arreglo bidimensional. De esta forma, los datos de entrada para un algoritmo de procesamiento de imágenes poseen un paralelismo por geometría. Este paralelismo puede ser explotado utilizando un arreglo bidimensional de procesadores, asignando un procesador por cada pixel de la imagen. Sin embargo, esto no es posible para fines prácticos debido al tamaño de las imágenes. Pero, la imagen segmentada puede ser dividida en bloques o tiras de tamaño *NxM*. Este tipo de segmentación presenta problemas al momento de procesar los bordes de cada uno de los bloques. Esto se debe a que información

requerida de la imagen para llevar a cabo procesamiento local puede estar dividida en dos bloques diferentes.

Otro tipo de paralelismo explota el hecho de que una imagen puede ser descompuesta en b planos de bits. Aquí, b es el número de bits en un pixel de la imagen. Si la imagen es binaria, b=1; y si es en escala de grises, generalmente b=8 [Pitas, 1993]. Existen varios operadores de procesamiento de imágenes, especialmente del tipo lineal, que pueden ser ejecutados en cada plano de bits de forma independiente. La aritmética que es llevada a cabo en planos de bit es llamada aritmética distribuida.

Dentro del dominio de operaciones existen diferentes tipos de paralelismo. La operación en tubería o *pipelining* es la más comúnmente utilizada y puede ser expresada como:

$$Y = F(X) = F_n(F_{n-1}(...F_2(F_1(X))...)) \tag{1}$$

Donde X es la imagen, o subárea de la imagen, de entrada, Y es la imagen de salida, F es un operador y F_i, $i = 1, 2,\ldots, n$ es su descomposición en cascada. Una aplicación típica de operador en tubería es la ejecución en cascada de filtros lineales. Otro ejemplo es la computación de operaciones morfológicas, erosión y dilatación:

$$Y = X - B = (...(X - B_1) - B_2) - ...) - B_n \tag{2}$$

El procesamiento digital de bajo nivel generalmente implica el uso de operadores en tubería homogéneos. Mientras que procesamiento de alto nivel involucra el uso de

operadores en tubería heterogéneos en la mayoría de los casos. La descomposición paralela involucra operadores de la forma:

$$Y = F(X) = F_1(X) \parallel F_2(X) \parallel ... \parallel F_n(X) \tag{3}$$

En esta expresión || denota una ejecución paralela. Una aplicación típica de este tipo de paralelismo para algoritmos de visión de bajo nivel es la ejecución paralela de filtros lineales de dos dimensiones. Otra aplicación típica en reconocimiento de patrones es la decisión estadística hecha de la forma:

$$\min_{1 \le i \le n}(d(x, x_i)) \tag{4}$$

Aquí, $\boldsymbol{x}$ denota el vector de características, $\boldsymbol{x_i}$, $i = 1,\ldots,n$ denota los vectores de referencia y $d(\boldsymbol{x},\boldsymbol{x_i})$ denota la distancia entre $\boldsymbol{x}$ y $\boldsymbol{x_i}$. En este caso, el cálculo de las diferentes distancias, o inclusive el cálculo del mínimo, puede llevarse a cabo en forma paralela.

Dentro del dominio de operaciones, existe un paralelismo que se explota en operaciones de bajo nivel dentro del procesador del sistema de cómputo. Para este caso, se representa cada uno de los pixeles de una imagen como un bit de la palabra que utiliza el procesador para realizar sus operaciones. De esta forma, se pueden procesar un número determinado de pixeles en paralelo. Y, el número de pixeles que se pueden procesar de manera simultánea está limitado tamaño de palabra del procesador. Este tipo de paralelismo también es llamado píxel-bit.

1.2.2 Cluster

Una implementación de un sistema de cómputo paralelo con modelo de memoria distribuida es el cluster de computadoras. Un cluster consiste en una colección de computadoras independientes interconectadas para trabajar como un recurso de computación único e integrado. La principal ventaja de un cluster es la relación costo/beneficio. Esto significa que se puede realizar tarea de súper cómputo utilizando computadoras de bajo costo.

El paradigma de programación más eficiente y ampliamente utilizado en sistemas de memoria distribuida, como el cluster, es el paso de mensajes. Las bibliotecas basadas en este paradigma proveen rutinas para iniciar y configurar el ambiente de paso de mensajes así como el envío y recepción de paquetes de datos. Actualmente, los sistemas de paso de mensajes de alto nivel más utilizados para cómputo científico y de ingeniería son: PVM (Parallel Virtual Machine) y MPI (Message Passing Interface). Estos fueron definidos por Oak, de Ridge National Laboratory, y por el Foro MPI, respectivamente.

I.3 Antecedentes

Aún cuando se han reportado trabajos de procesamiento de imágenes satelitales en paralelo desde 1965 [2], en los últimos años han continuado como un área de estudio importante. A continuación se mencionan algunos trabajos previos relacionados con el área de procesamiento de imágenes en paralelo.

- **Procesamiento Morfológico en Tiempo Real usando el Autómata Celular 2-D Altamente Paralelo CAM^2 [Ikenaga, 2000].** Presentan algoritmos paralelos para la utilización de filtros morfológicos en tiempo real para

aplicaciones de procesamiento de imágenes en video. Sin embargo, este trabajo muestra una gran dependencia de la arquitectura de supercómputo utilizada. Por esta razón, se trata de un trabajo limitado a la plataforma física donde se ejecutan los algoritmos.

- **Algoritmos Morfológicos Binarios Eficientes en Procesadores Paralelos Masivos. [Svolos, 2000].** En este trabajo presentan algoritmos para las operaciones de erosión y dilatación en imágenes binarias. Reportan factor de aceleración lineal, pero en simulación, para una arquitectura avanzada de procesador asociativo.

- **Arquitectura de Software para Procesamiento de Imágenes en Paralelo Transparente al Usuario [Seinstra, 2002].** El núcleo de esta arquitectura es una librería que contiene un conjunto de tipos de datos abstractos y operaciones a nivel de píxel ejecutadas en datos en forma paralela. El aspecto más distintivo de esta propuesta de implementación de librería es tomar una posición minimalista. Esto significa en esencia que se trata de maximizar la reutilización de operaciones y evitar redundancia de código tanto como sea posible. Además de ser relativamente fácil de implementar, una librería paralela construida de esta forma es extensible, fácilmente mantenida y aún así, alta en desempeño. Se implementó el siguiente conjunto de algoritmos comunes en el procesamiento de imágenes: operación unaria de píxeles, operación binara de píxeles, reducción global, operación de vecindario (mediana y porcentajes) y convolución generalizada.

I.4 Importancia de la investigación

I.4.1 Motivación y Justificación

El procesamiento de imágenes es un área importante en las ciencias computacionales porque es común representar información como una imagen. Además, resulta un área muy atractiva para el desarrollo de cómputo paralelo por diferentes razones. Entre estas razones podemos mencionar:

- Los algoritmos de procesamiento de imágenes pueden tratar problemas de gran complejidad. Una imagen puede contener millones de píxeles a los que se debe realizar algún tipo de operación, o conjunto de ellas, para procesarla.
- Calidad creciente de las imágenes. Cada vez se requieren más imágenes, de mayor tamaño y de mayor resolución. De esta forma, se necesitan implementaciones que permitan realizar el procesamiento de imágenes en el menor tiempo posible.
- Paralelismo inherente en operaciones a imágenes. Una operación puede llevarse a cabo en todos los píxeles que forman la imagen. Hasta donde la falta de dependencia entre píxeles lo permite, se puede dividir la imagen para procesar los diferentes píxeles de la imagen de manera simultánea.

Los filtros morfológicos son una herramienta útil para el procesamiento de imagen. Esto se debe a que tienen una gran diversidad de aplicaciones las cuales se llevan a cabo de manera efectiva. Además, existe una baja dependencia entre píxeles que permite la implementación de algoritmos paralelos eficientes.

El uso de cluster se ha hecho cada vez más común por diferentes ventajas que tiene sobre el uso de supercomputadoras. Entre estas ventajas se encuentran la escalabilidad y la relación costo/beneficio del mismo. De esta manera, el uso de cluster es una opción viable para realizar tareas de supercómputo a bajo costo. Otro punto a considerar es que el uso de cluster se ha extendido de gran manera en todo el mundo [Hwang 1997]. Esto garantiza la disponibilidad de recursos para el desarrollo de aplicaciones basadas en cluster. También, provee la portabilidad del código desarrollado a prácticamente cualquier plataforma que esté basada en cluster.

I.4.2 Limitaciones y Suposiciones Fundamentales

Como primera limitación está el análisis de topologías paralelas. Sin duda alguna, la arquitectura física del sistema paralelo determina en buena medida el desempeño de los algoritmos paralelos. Sin embargo, la investigación de arquitecturas de sistemas paralelos queda fuera del alcance del presente trabajo. Se tienen dos posibilidades para la implementación del cluster pero ninguna permite la manipulación de la topología.

Se utilizará MPI para la distribución de procesos en los nodos del sistema. No se harán comparaciones con diferentes tipos de programación paralela.

Otra limitante será el área de aplicación. En este trabajo se limitará la aplicación de los filtros morfológicos a imágenes en dos dimensiones. Además, no se considerarán imágenes a color y se limitará la aplicación a imágenes binarias y en escala de gris.

I.4.3 Contribución al Conocimiento

La principal contribución de este trabajo será un modelo algorítmico para la implementación de filtros morfológicos en imágenes de dos dimensiones binarias y en escala de gris. Las funciones incluidas estarán caracterizadas en cuanto a desempeño en una comparación hecha con su contraparte de implementación secuencial en un cluster.

Existe una gran variedad de aplicaciones que se pueden desarrollar utilizando filtros morfológicos. En este trabajo se tratarán a las siguientes:

- Filtrado de ruido impulsivo
- Extracción de contornos
- Reconstrucción de textos
- Detección y localización de objetos
- Esqueleto morfológico de estructuras geométricas
- Aplicación de transformada *Top Hat*

I.5 Objetivos

I.5.1 Objetivo General

Diseñar un modelo algorítmico paralelo de filtros morfológicos para el procesamiento de imágenes e implementar dicho modelo en un sistema de cómputo paralelo basado en cluster.

I.5.2 Objetivos Específicos

1) Estudiar la naturaleza de los filtros morfológicos para procesamiento de imágenes binarias y en escala de gris

2) Estudiar sistemas de cómputo paralelo basados en cluster

3) Implementar algoritmos secuenciales de filtros morfológicos para procesamiento de imágenes

4) Estudiar diferentes técnicas de paralelización de algoritmos para aplicaciones de procesamiento de imágenes

5) Proponer técnicas de paralelización para implementación de filtros morfológicos

6) Implementar algoritmos paralelos de filtros morfológicos en un cluster de computadoras

7) Realizar estudio comparativo de tiempos de ejecución entre implementaciones secuenciales y paralelas, de filtros morfológicos

8) Implementar aplicaciones de sistema desarrollado para procesar imágenes del mundo real

I.6 Metodología

A continuación se detalla la metodología a seguir, incluyendo una breve descripción de las actividades a realizar en cada una de sus partes.

- Estudio de Filtros Morfológicos. En esta parte se realizará la investigación de antecedentes teóricos de los filtros morfológicos, incluyendo la base

matemática, descripción de operaciones comunes y las distintas aplicaciones. Se basará en búsqueda bibliográfica entre libros sobre el tema y artículos de divulgación científica.

- Implementación y Experimentos de Algoritmos de Filtros Morfológicos Secuenciales. Una vez atendida la parte teórica de filtros morfológicos, se implementarán en programas secuenciales para realizar experimentos de diferentes aplicaciones de los mismos. Se determinarán imágenes de prueba y se ejecutarán los diferentes algoritmos implementados. Los parámetros a evaluar serán la efectividad de los algoritmos para las distintas aplicaciones y el tiempo de ejecución.

- Estudio y Construcción de un Cluster. Aquí se estudiarán las formas más convenientes de implementar un cluster. Esta parte incluye el estudio de implementación de redes de computadoras, estudio de sistemas operativos y modelos de programación utilizados en cluster. Se pretende la implementación de un cluster utilizando PC comunes ubicadas en el centro de cómputo, o bien, un cluster ya instalado y disponible. También se incluye una parte de experimentos del cluster para comprobar su funcionamiento, utilizando pequeños programas de prueba y algún *benchmark*, como *LINPACK*.

- Estudio de Paralelización en Procesamiento de Imágenes. Se estudiarán las técnicas de paralelización específicas que se pueden aplicar a procesamiento de imágenes, determinar aquéllas que son útiles para la implementación de los filtros morfológicos bajo el paradigma de paso de mensajes.

- Implementación y Experimentos de Algoritmos Paralelos de Filtros Morfológicos. Los algoritmos de filtros morfológicos implementados en forma secuencial, serán implementados en paralelo para su ejecución en cluster. Se utilizarán las mismas imágenes de prueba ya establecidas en la parte de implementación secuencial para poder realizar un análisis comparativo. De igual forma, se caracterizarán los algoritmos paralelos en cuanto a los diversos factores de desempeño.

- Análisis de Resultados. Esta etapa constará del análisis comparativo de los resultados obtenidos en los experimentos de implementación secuencial y paralela de los filtros morfológicos.

- Escritura. Esta parte comprende la escritura de la tesis y las distintas revisiones de la misma.

I.7 Organización del Documento

El documento está organizado en tres secciones:

Primera sección. Capítulos I, II, III y IV. Consta de la base metodológica para llevar a cabo el presente trabajo, la teoría relacionada con los filtros morfológicos y los fundamentos de cómputo paralelo relevantes para esta investigación.

Segunda sección. Capítulos V y VI. Está constituida por la explicación detallada de la implementación de los algoritmos morfológicos de forma secuencial y paralela. También incluye un capítulo de resultados experimentales y cálculo de parámetros de desempeño de los algoritmos implementados.

Tercera sección. Capítulos VII y VIII. En esta sección se presenta un análisis de complejidad de los distintos algoritmos de filtros morfológicos, secuenciales y paralelos. Y termina con las conclusiones de este trabajo.

Capítulo II

Teoría de Filtros Morfológicos

Los filtros morfológicos son transformaciones no lineales que modifican las características geométricas de señales u objetos en una imagen. Están basados en teoría de conjuntos y procesan imágenes de acuerdo a criterios que pueden ser fácilmente interpretados por los seres humanos como: forma, tamaño, orientación, textura, etc. [Serra, 1988].

La operación de este tipo de filtros es realizada de manera local. Los filtros morfológicos hacen uso de un elemento estructural en base al cual las operaciones son llevadas a cabo. De manera general, el elemento estructural debe cumplir con ciertas características para ser adecuado para las operaciones morfológicas. Una de ellas es que se debe de tratar de un elemento lo más isotrópico posible [Jain, 1986]. La utilización de un elemento estructural no simétrico da por resultado imágenes con desplazamientos lo cual es una condición no deseada. Ejemplos de estos elementos estructurales se muestran en la Figura 1.

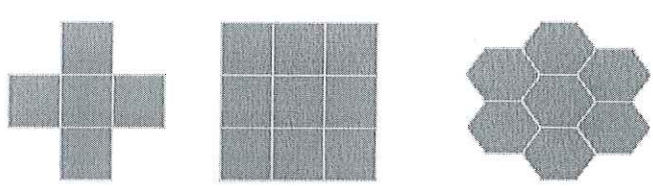


Figura 1. Elementos estructurales comunes.

Otra característica deseable del elemento estructural es que su tamaño sea lo más pequeño posible. Esto propicia que las transformaciones de una imagen sean sutiles. El uso de un elemento estructural grande propiciará la pérdida de información importante en la aplicación de alguna transformación morfológica. Sin embargo, estas dos características no siempre se cumplen. El tamaño y forma del elemento estructural dependerán dos cosas. Del tamaño de los objetos que queremos preservar después de las operaciones morfológicas y la operación morfológica en sí.

II.1 Filtros Morfológicos para Procesamiento de Imágenes Binarias

Una imagen binaria es aquélla en la que los píxeles están representados por una variable que puede tomar dos valores, 1 y 0. De una manera práctica, podemos dividir las operaciones realizables con filtros morfológicos en dos grupos: operaciones básicas y operaciones compuestas. Si establecemos una imagen binaria dada por X y un elemento estructural B, podemos definir dos operaciones básicas de los filtros morfológicos: erosión y dilatación.

II.1.1 Operaciones Básicas

II.1.1.1 Erosión

La erosión está definida como [Heijmans, 1997]:

$$E(X,B) = X - B = \{x : B_X \subset X\} \tag{5}$$

Es decir, un píxel x en una imagen dada X, existe como resultado de la operación morfológica si el elemento estructural B existe como un subconjunto de la imagen. En la Figura 2 se observa el elemento estructural y la imagen de entrada. Los puntos negros en la tercera imagen de la figura, son los puntos que existirán en la imagen después de una operación de erosión con el elemento estructural dado.

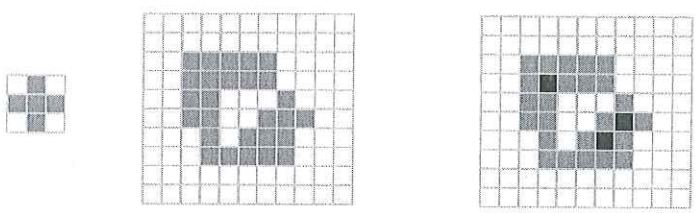


Figura 2. Erosión de una imagen con un elemento estructural.

II.1.1.2 Dilatación

La dilatación se define como [Heijmans, 1997]:

$$D(X,B) = X + B = \{x : B_X \cap X \neq 0\} \tag{6}$$

Esto significa que un píxel existe siempre y cuando exista al menos un punto de intersección entre el elemento estructural B y la imagen X. En la Figura 3 se muestran 3 imágenes. Las dos primeras corresponden al elemento estructural y a la imagen de entrada. La tercera imagen muestra con puntos negros la dilatación llevada a cabo a la imagen original.

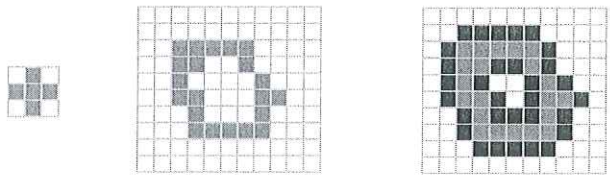


Figura 3. Dilatación de una imagen con un elemento estructural.

En base a estas operaciones básicas y otras operaciones de conjuntos, podemos definir operaciones más complejas.

II.1.2 Operaciones Compuestas

II.1.2.1 Apertura y Cerradura

Existen dos transformaciones morfológicas comunes en base a ejecuciones consecutivas de operaciones de erosión y dilatación. Estas operaciones son de apertura y, su contraparte, de cerradura. La operación de apertura está definida como:

$$X_B = (X - B) + B \tag{7}$$

Esto significa que a una imagen X primero se le aplicará la operación de erosión, y después la operación de dilatación, utilizando el elemento estructural B. Esta definición representaría una sola iteración de la transformación de apertura. Sin embargo, es común realizarla iterativamente un número n de veces para obtener resultados requeridos por alguna aplicación. En la Figura 4 se muestran tres imágenes. Primeramente se encuentra el elemento estructural, seguido de la imagen de entrada. La tercera imagen muestra con

puntos negros aquellos píxeles que existirán después de la erosión, y la cuarta imagen en puntos negros los píxeles que prevalecerán después de aplicar la operación de apertura.

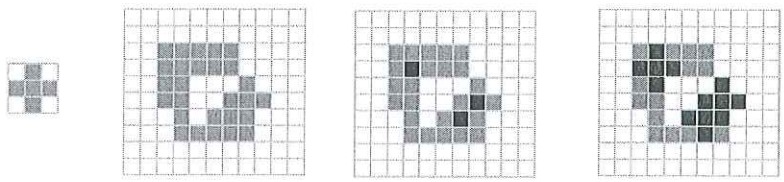


Figura 4. Operación de apertura

Al aplicar la operación de erosión sobre la imagen, en primera instancia, se suavizan los contornos de los objetos que se encuentran en la imagen y se pierden pequeños objetos de la imagen. Es decir, se eliminan todos aquellos detalles que sean de menor tamaño que el elemento estructural. Después, al aplicar la operación de dilatación, los objetos recuperan parte de su tamaño original y los contornos se ven influenciados por la forma del elemento estructural.

La transformación de cerradura está definida como:

$$X_B = (X + B) - B \tag{8}$$

En este sentido, la primera operación a realizar sobre la imagen será la dilatación, y en seguida la erosión. De igual manera, es común realizar esta transformación iterativamente para obtener el resultado esperado. La Figura 5 muestra al elemento estructural seguido de la imagen de entrada. La tercera imagen muestra con puntos negros los píxeles que se adicionan a la imagen después de aplicar la operación de

dilatación, y la cuarta imagen muestra con puntos negros los píxeles de la imagen que prevalecerán después de completar la operación de cerradura.

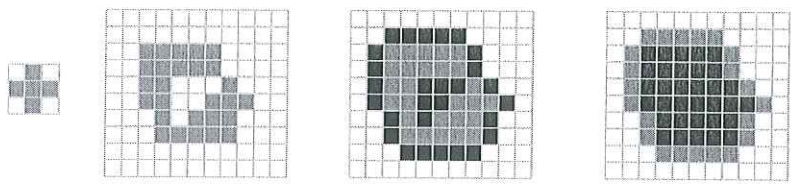


Figura 5. Operación de cerradura

Como puede observarse en el ejemplo, la primera operación elimina objetos delgados de la imagen original que sean más angostos que el elemento estructural. Y al aplicar la segunda operación, une objetos cuando la separación entre ellos es menor al tamaño del elemento estructural.

II.1.2.2 Esqueleto Morfológico

Una transformación morfológica muy útil consiste en obtener el esqueleto de los objetos contenidos en una imagen. Esta información es útil para el cálculo de longitudes y áreas de objetos. De manera general, el esqueleto de un objeto se refiere a la representación lineal de un objeto que es:

a. De un píxel de anchura,

b. Que pasa a través de la parte central del objeto, y

c. Conserva la topología del objeto.

El esqueleto morfológico se define formalmente como:

$$S(X) = \bigcup_{k=0}^{k} S_k(X) \tag{9}$$

Donde $S_k(X)$ se define como el subconjunto de esqueletos dado por:

$$S_k(X) = [(X - kB)/(X - kB)_B \quad k = 0,1,2,...,K \tag{10}$$

Aquí, la operación $(X\text{-}kB)$ implica $((((X\text{-}B)\text{-}B))\ldots\text{-}B)$, la operación $(X\text{-}kB)_B$ denota apertura de $(X\text{-}kB)$ con respecto a B. El elemento estructural B se escoge, al trabajar con imágenes de dos dimensiones, como una aproximación a un disco circular que sea convexo, cerrado y simétrico. Algunos ejemplos de esqueletos morfológicos se muestran en la Figura 6, donde los objetos son de color gris y en color blanco el esqueleto morfológico obtenido.

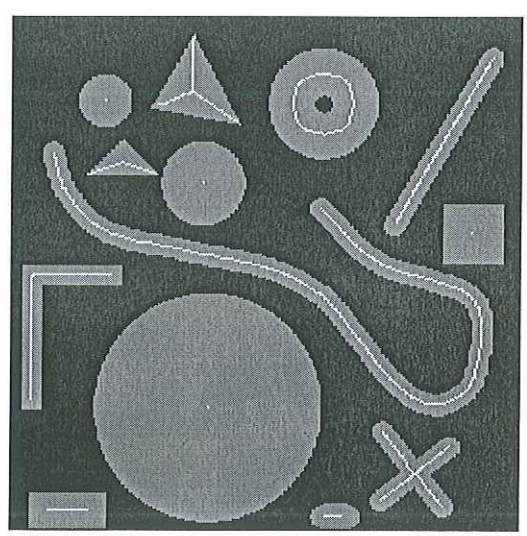


Figura 6. Ejemplos de esqueletos morfológicos

II.1.2.3 Operación *Hit or Miss*

La operación *Hit or Miss* se utiliza para localizar objetos con una forma definida contenidos en una imagen. Esta operación encuentra conjunto de píxeles con ciertas propiedades geométricas, como pueden ser esquinas o bordes. También sirve de base para otras operaciones morfológicas más complejas, como adelgazamiento y engrosamiento. Si tenemos una imagen X y un elemento estructural compuesto, de la forma $B = (B_1, B_2)$, podemos definir formalmente la operación *Hit or Miss* como:

$$X \otimes B = \left\{ x : B_1 \subset X, B_2 \subset X^C \right\} \tag{11}$$

Esto significa que la operación nos da como resultado aquellos objetos B_1 que se encuentran en X y de manera simultánea, aquellos objetos B_2 que se encuentran en el fondo de la misma imagen X.

Utilizando filtros morfológicos, la operación *Hit or Miss* puede llevarse a cabo como:

$$X \otimes B = (X - B_1) \cap (X^C - B_2) \tag{12}$$

En la Figura 7 se detalla el proceso de *hit or miss* de forma gráfica.

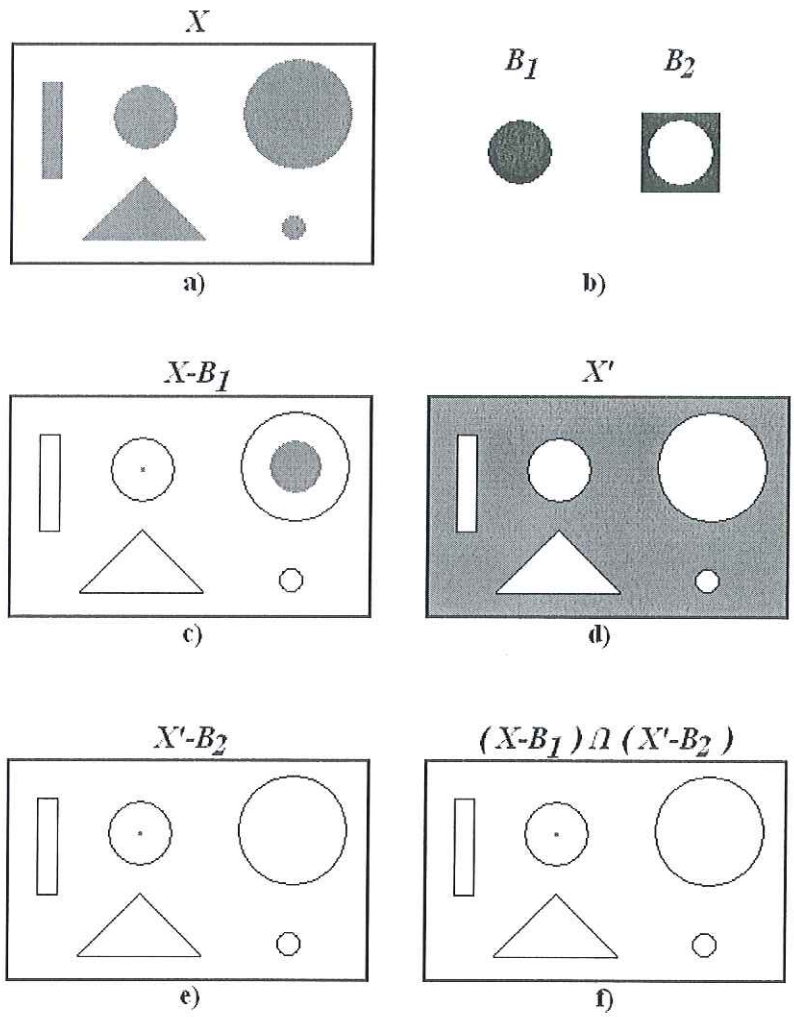


Figura 7. Operación Hit or Miss. a) Imagen original. b) Objeto compuesto B. c) Erosión de la imagen original con B1. d) Complemento de la imagen original. e) Imagen de d) erosionada con B2. f) Intersección de c) y e).

II.1.2.5 Rellenado de Objetos

Existe un proceso iterativo para rellenar objetos huecos contenidos en una imagen. Para llevarlo a cabo se requiere calcular el contorno de dichos objetos. Después de ubicar un punto dentro del objeto a rellenar, se va dilatando dicho punto utilizando la ecuación:

$$X_k = (X_{k-1} \oplus B) \cap A^C \qquad k = 1,2,3\ldots \tag{13}$$

Donde A^C denota el contorno del objeto a rellenar, B el elemento estructural y X la imagen completa donde se encuentra el objeto. La primera iteración, con $k=1$, está dado con el punto de inicio dentro del objeto a rellenar. Este proceso se repite hasta que no existan cambios en la imagen.

Las operaciones aquí descritas son aplicables a imágenes binarias. Sin embargo, el uso de filtros morfológicos puede ser extendido para imágenes en escala de gris y a color.

II.2 Filtros Morfológicos para Procesamiento de Imágenes en Escala de Grises

Las transformaciones morfológicas explicadas en la sección anterior son aplicables a imágenes binarias. Sin embargo, estas técnicas pueden extenderse para el procesamiento de imágenes en escala de gris. Esto es importante ya que las imágenes en aplicaciones reales comúnmente son en escala de gris y no binarias. Esta extensión se lleva a cabo representando a la función de la imagen como un conjunto ordenado de señales binarias. Para lograr lo anterior se utiliza la propiedad de apilamiento o *stacking*.

II.2.1 Propiedad de Apilamiento o Stacking

La sección transversal de una función a un nivel particular es un conjunto binario, y el conjunto de todas las secciones transversales forma la representación de la función.

Este proceso se conoce como descomposición por umbral [Vincent, 1993] y se ilustra en la Figura 8 para una señal unidimensional *f(x)*. Los umbrales están determinados por t_i, y la función queda representada por el conjunto *X*.

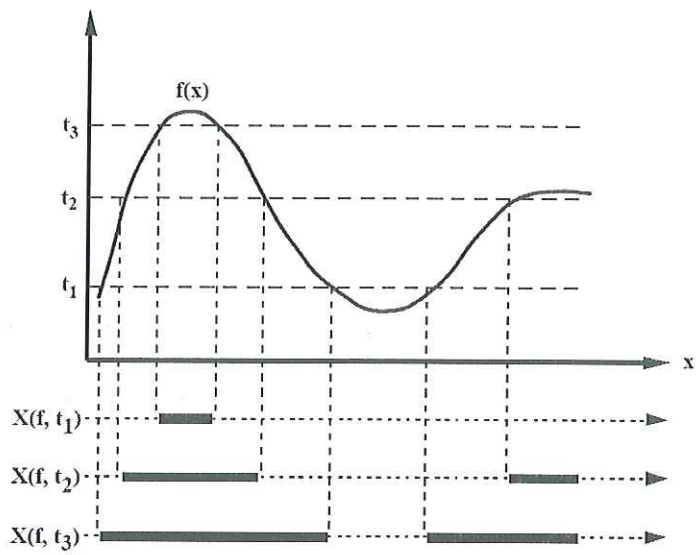


Figura 8. Descomposición por umbral en una dimensión.

Un filtro basado en este principio de descomposición de umbral debe cumplir la propiedad de apilamiento. Esta propiedad consiste en que para dos conjuntos X y Y cualesquiera, tal que:

$$X \subseteq Y \tag{14}$$

Los conjuntos filtrados conservan la misma relación, es decir, que:

$$f(X) \subseteq f(Y) \tag{15}$$

Un filtro de procesado de conjuntos binarios posee la propiedad de apilamiento si y sólo si la salida de dicho filtro puede ser expresado como una función booleana que carezca del complemento de cualquiera de sus variables de entrada. Este procedimiento nos permite procesar una imagen en escala de gris como un conjunto de imágenes binarias, a las cuales se les puede procesar utilizando filtros morfológicos para imágenes binarias. Entonces, una imagen de entrada en escala de gris puede ser expresada como la suma de todas sus versiones descompuestas por umbral en imágenes binarias. Así, la descomposición por umbral de una imagen en escala de gris puede expresarse como [Maragos, 1990]:

$$X_a(m,n) = \begin{cases} 1, X(m,n) \geq a \\ 0, X(m,n) < a \end{cases} \tag{16}$$

Y, la imagen de salida de cualquier filtro morfológico aplicado a dicha imagen de entrada puede expresarse como la suma de las imágenes binarias descompuestas por umbral filtradas. De esta forma, tenemos [Maragos, 1990]:

$$X(m,n) = \sum_{a=1}^{A} X_a(m,n) = \max\{a : X_a(m,n) = 1\} \tag{17}$$

II.2.2 Erosión y Dilatación

Los conceptos de erosión y dilatación de imágenes binarias se pueden extender para imágenes en escala de gris. Para lograrlo, se hace uso de la propiedad de apilamiento. De esta forma, podemos realizar una descomposición por umbral de la imagen en escala de gris. Esto dará como resultado un conjunto de imágenes binarias a las cuales podemos aplicar las operaciones de erosión o dilatación binarias según convenga. Como último paso, simplemente se utiliza la ecuación de apilamiento para obtener la imagen filtrada en escala de gris nuevamente.

$$X_G(m,n) - B = \max\{a : X_a(m,n) - B = 1\} \tag{18}$$

Análogamente, para la dilatación se tiene:

$$X_G(m,n) \oplus B = \max\{a : X_a(m,n) \oplus B = 1\} \tag{19}$$

II.2.3 Apertura y Cerradura

Se puede definir las operaciones de apertura y cerradura para imágenes en escala de gris en base a las transformaciones morfológicas de erosión y dilatación de imágenes en escala de gris.

Para la apertura tenemos que:

$$X_G \circ B = (X_G - B) \oplus B \tag{20}$$

Y para la cerradura:

$$X_G \bullet B = (X_G \oplus B) - B \tag{21}$$

II.2.4 Transformada Sombrero de Copa o Top Hat

La transformada Top Hat se utiliza para segmentar objetos en imágenes en escala de gris que difieren en brillo con respecto al fondo [Sonka, 1998]. Si tenemos una imagen X y un elemento estructural B, matemáticamente podemos definir esta transformada como:

$$X_{TH} = X_G \setminus (X_G \circ B) \tag{22}$$

Es decir, la transformada top hat de la imagen X es el residuo de la resta entre la imagen original y la imagen con apertura del elemento estructural B. Esta transformada es útil para extraer objetos claros de imágenes con fondos oscuros que cambian lentamente. El concepto de esta transformada puede observarse gráficamente en la Figura 9. En este caso, la imagen en escala de gris es expresada como una función en una dimensión para mayor claridad. Una vez realizada la transformación, puede aplicarse un umbral, como se ve en la Figura 9. Ésto permite obtener los objetos que sobresalgan del fondo con un brillo determinado.

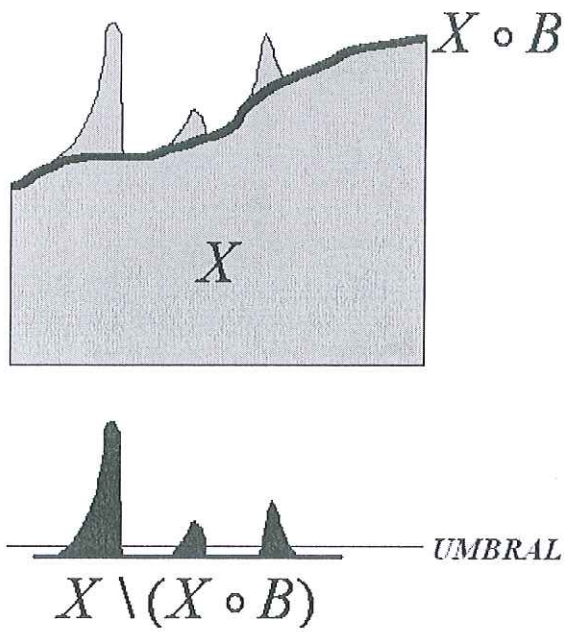


Figura 9. Transformada Top Hat con umbral.

Capítulo III

Aplicaciones con Filtros Morfológicos

Los filtros morfológicos se pueden utilizar para una gran variedad de aplicaciones. En este capítulo se detallan algunas de ellas llevadas a cabo en el presente trabajo.

III.1 Reconstrucción de Textos

Una aplicación de filtros morfológicos tiene que ver con la pérdida de información visual de caracteres en imágenes que contienen textos. Esta pérdida de información puede darse por el daño físico presente en los documentos de texto o por una baja calidad en la imagen obtenida de dicho documento, entre otros. Para mejorar la calidad de los caracteres contenidos en la imagen, puede realizarse la operación dilatación como se observa en la Figura 10.

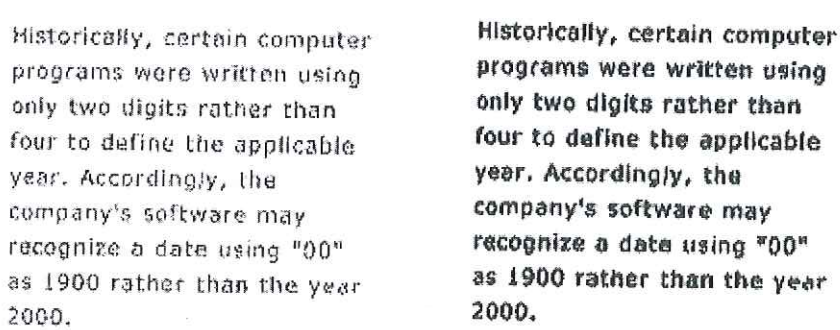


Figura 10. Imagen de texto original y reconstruido.

III.2 Contorno

Existe un procedimiento para encontrar el contorno de objetos contenidos en una imagen binaria. Este procedimiento consiste en calcular primeramente la erosión de la imagen binaria. Después se calcula la resta de la imagen original con la imagen erosionada. El residuo dará como resultado los contornos de los objetos en la imagen. De manera formal, podemos definir esta transformación morfológica como:

$$X_C = X \setminus (X - B) \tag{23}$$

En la Figura 11 se observan dos imágenes. La primera es una imagen binaria y la segunda, el contorno obtenido de los objetos de dicha imagen.

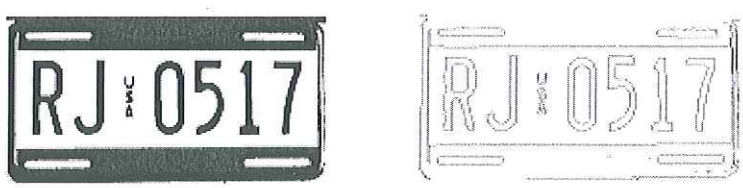


Figura 11. Contorno de una imagen binaria.

III.3 Filtrado de Ruido Impulsivo

El ruido impulsivo en una imagen es una condición indeseable y se presenta en una imagen como pixeles que presentan gran diferencia en brillo con respecto a los pixeles de su vecindario. Para eliminarlo, se puede hacer uso de las transformaciones morfológicas de apertura y cerradura. Así, tenemos que:

$$X = (X \circ B) \bullet B \tag{24}$$

En una imagen binaria, blanco y negro, el ruido impulsivo se observa como puntos negros y blancos localizados al azar sobre la imagen. El aplicar una operación de apertura tendrá como objetivo eliminar el ruido correspondiente a los puntos negros. Los puntos blancos desaparecerán con la operación de cerradura. De esta forma, se logra una imagen filtrada de ruido impulsivo, como se observa en la Figura 12.

Figura 12. Imagen binaria filtrada de ruido impulsivo.

Se pueden aplicar las mismas operaciones morfológicas con el fin de eliminar ruido impulsivo en una imagen en escala de gris. Esto puede observarse en la Figura 13.

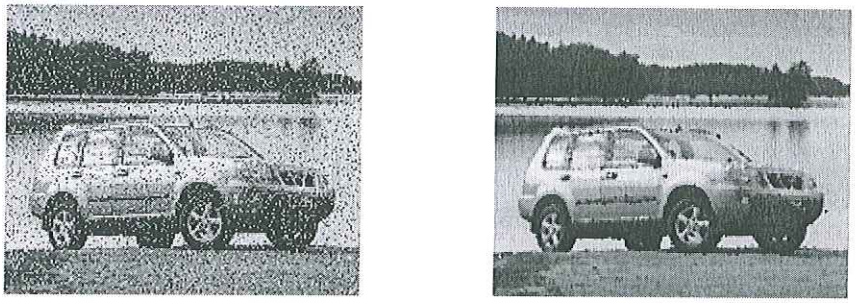


Figura 13. Imagen en escala de gris filtrada de ruido impulsivo.

III.4 Detección de Objetos

La detección de objetos tiene como objetivo determinar la presencia de objetos en una imagen que cumplan con alguna característica. Para el ejemplo de la Figura 14, se busca en la imagen células que cumplan con un tamaño mínimo. Este tamaño está dado por el tamaño seleccionado del elemento estructural. Las tres imágenes de la Figura 14 corresponden a la imagen original, el elemento estructural y la imagen original erosionada con el elemento estructural.

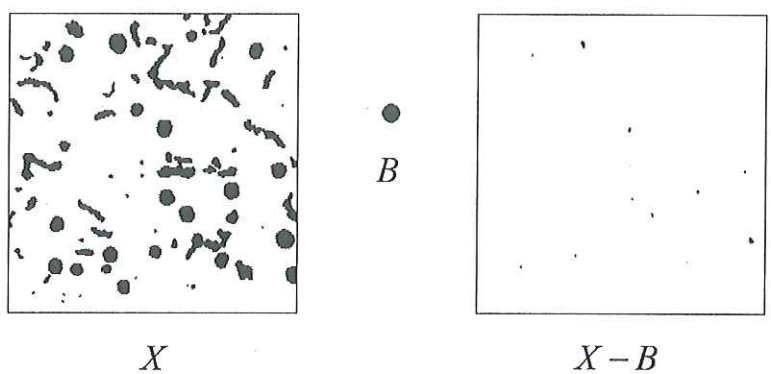
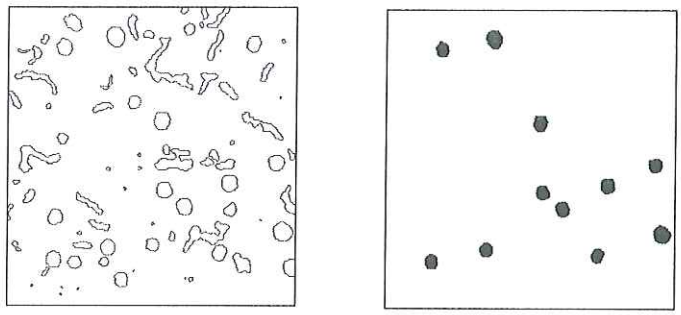


Figura 14. Imagen original, elemento estructural y resultado de erosión.

Para obtener las originales, en este caso células, se hace uso de otras transformaciones morfológicas. Para este caso, se calcula primero el contorno de los objetos de la imagen. Después se utiliza el algoritmo de rellenado de objetos, tomando como puntos de inicio los puntos remanentes de la imagen erosionada. Como límite del algoritmo se toma la imagen de contornos obtenida anteriormente. El contorno y el resultado final se muestran en la Figura 15.



$C = X \setminus (X - B)$ $\qquad$ $X_K = (X_{K-1} \oplus B_2) \cap C^C$

Figura 15. Contorno de objetos en imagen original e imagen después de realizar el rellenado de contornos a partir de la imagen erosionada de la figura anterior.

III.5 Localización de Objetos

En la Figura 16 aparecen diferentes imágenes que ilustran la forma en que el proceso se lleva a cabo. Primeramente se tienen la imagen original y el objeto del cual realizaremos la búsqueda. En la segunda imagen, aparecen únicamente los píxeles donde se ha encontrado el objeto buscado, por medio de la operación *Hit or Miss*. Para generar la ubicación de los objetos solo es necesario observar las coordenadas dentro de la imagen en las que corresponde un "1" después de la operación *Hit or Miss*. Para clarificar la ubicación de dichos píxeles, se realiza la operación de dilatación de la imagen con el elemento estructural utilizado. Esto produce una imagen con la presencia de los objetos que correspondan al objeto buscado, como se observa en la Figura 17.

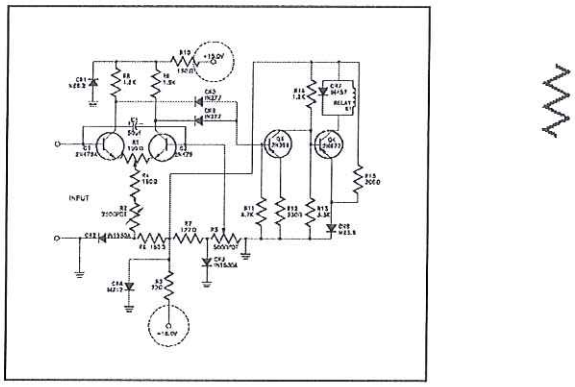


Figura 16. Imagen binaria y elemento estructural a localizar en la imagen

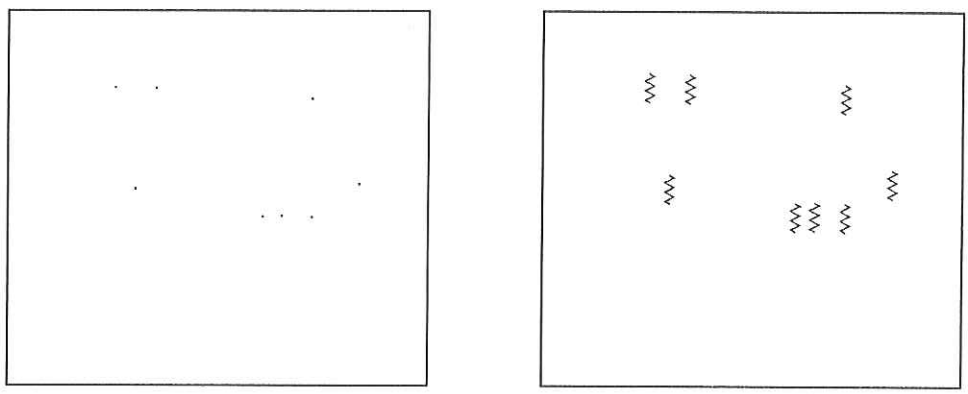


Figura 17. Localización de objetos. Puntos de localización del objeto buscado y puntos dilatados con elemento estructural.

III.6 Cálculo de Dimensiones

Una forma de calcular el perímetro puede hacerse obteniendo primeramente el contorno de objetos en una imagen. Esto da por resultado un cojunto de pixeles que describen el perímetro del objeto. Después de esto, se realiza el conteo de dichos pixeles y esto expresa el perímetro de dicho objeto.

Una forma de llevarse a cabo el cálculo de área de un objeto es contando el número de píxeles que forman dicho objeto. Esto se realiza al hacer el rellenado de dicho objeto. Los pixeles se van contando según vayan conformando el relleno del objeto en cuestión.

III.7 Transformada Sombrero de Copa o Top Hat

Existen imágenes en las cuales la diferencia de brillo entre objetos nos proporciona información específica del proceso observado. En este ejemplo de aplicación, se tiene un proceso industrial en el cual se tiene que controlar una válvula que permite la salida de gas hacia una flama de acetileno. Utilizando la transformada *top hat* y un umbral adecuado, un algoritmo puede determinar fácilmente la apertura correcta de la válvula. La apertura de dicha válvula puede determinarse buscando que el área de la flama, después de aplicar las transformaciones morfológicas, sea cuando mucho un valor determinado.

La imagen original de la Figura 18 muestra 4 flamas. Se encuentran ordenadas de izquierda a derecha, con flamas de menor a a mayor calidad.

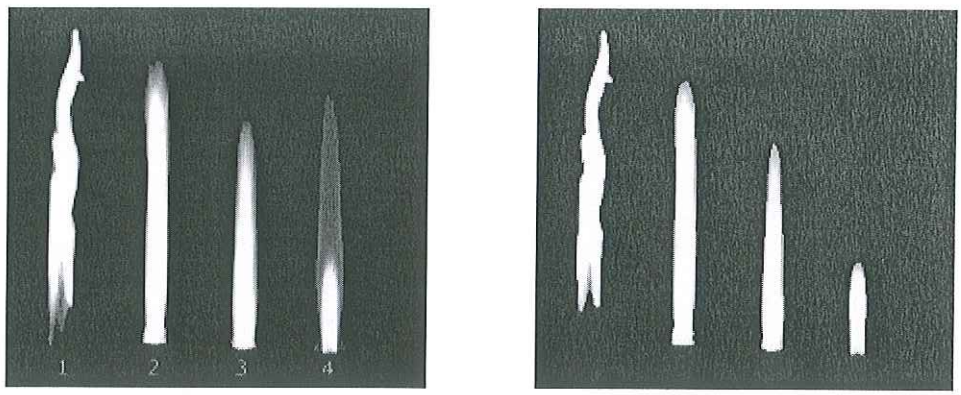


Figura 18. Imagen orginal y su transformada *Top Hat.*

Capítulo IV

Computación Paralela

La computación paralela surgió de la necesidad de procesar información para brindar una solución adecuada a problemas en los que el cómputo secuencial no podía. Esto puede deberse a que la complejidad del problema a tratar impide que se tenga una respuesta en el tiempo requerido. Si bien la computación paralela abarca una gran cantidad de información, en el presente capítulo se tratan sólo los tópicos relevantes para este trabajo de tesis.

IV.1 Arquitecturas Paralelas

En el presente trabajo se hicieron uso de dos diferentes arquitecturas paralelas: supercomputadora y cluster. Una supercomputadora es un sistema de cómputo multiprocesador integrado, de alto rendimiento. Su sistema de comunicaciones entre procesadores y periféricos, acceso a memoria y la arquitectura de los procesadores que la integran, están diseñados y optimizados para utilizarse como una herramienta conjunta. Por otro lado, un cluster consiste en una colección de computadoras independientes interconectadas para trabajar como un recurso de computación único e integrado. Por el contrario de la supercomputadora, el sistema de comunicación entre procesadores y periféricos, acceso a memoria y la arquitectura de los procesadores, son de uso común en la industria de computadoras personales.

A continuación, se hace una comparación entre sistemas basados en una supercomputadora y sistemas basados en cluster, mostrados en la Tabla I.

Tabla I. Comparación entre supercomputadoras y cluster.

SUPERCOMPUTADORA	CLUSTER
El costo económico para adquirir una es alto	Se obtienen a bajo costo
La escalabilidad es económicamente costosa y también en función de la eficiencia de todo el sistema	Es relativamente barato agregar una mayor cantidad de nodos sin que esto afecte la eficiencia
Se requiere una menor cantidad de nodos que en un cluster para obtener el mismo desempeño	Se requiere de un mayor número de unidades de procesamiento para equiparar el desempeño de una supercomputadora
Requiere de equipo específico al modelo de supercomputadora para poder escalarla	Se puede escalar con equipo heterogéneo, diferente al ya existente.
Existe un fuerte compromiso con el proveedor del equipo para soporte y mantenimiento	El soporte se encuentra hasta de forma gratuita y el mantenimiento es prácticamente el de una computadora personal de uso común
Generalmente es físicamente un equipo monolítico que requiere comparativamente poco espacio y energía	Se trata de varios equipos interconectados por lo que requiere de mayor espacio y energía

IV.2 Programación Paralela

Existen diferentes modelos de implementación de algoritmos en sistemas de cómputo paralelo. Estos modelos de programación paralela se realizan en sistemas reales generalemente extendiendo lenguajes como Fortran o C [Hwang 1998]. Esta extensión puede tomar tres caminos que son: directivas al compilador, constructores nuevos y bibliotecas.

Cuando se utiliza directivas al compilador, el lenguaje de programación es el mismo, pero se agregan a los programas comentarios con formato especial, llamados directivas al compilador o pragmas. En cambio, en el caso de constructores nuevos, el lenguaje de programación es extendido con algunos constructores nuevos que soportan paralelismo e interacción. Para la extensión por medio de bibliotecas, además de las bibliotecas que ya existen para el lenguaje secuencial, se agregan bibliotecas con un conjunto nuevo funciones las cuales dan soporte a paralelismo y operaciones de interacción. Dentro de la programación paralela con el uso de bibliotecas se encuentra el estándar MPI que se detallará en la siguiente sección.

IV.2.1 Paso de mensajes, MPI

En el paradigma de paso de mensajes, se hace uso de paralelismo al crear diferentes procesos que se llevan a cabo por los diferentes procesadores, o hilos de ejecución, disponibles. Sin embargo, el paralelismo es explícito y a cargo del programador. Esto significa que los diferentes procesos no se crean solos, sino que el programador debe dividir las diferentes actividades.

De manera general, el modelo de paso de mensajes tiene las siguientes características [Hwang 1998]:

- Multihilo. Un programa bajo este modelo consiste en procesos múltiples, cada uno de los cuales tienen su propio hilo de control y puede ejecutar código diferente de ser necesario.
- Paralelismo asíncrono. Los procesos de un programa basado en paso de mensajes se ejecutan de forma asíncrona. Operaciones especiales son utililizadas para sincronizar los procesos cuando así se requiera.
- Espacios de direcciones separados. Los procesos de un programa paralelo residen en diferentes espacios de direcciones. Las variables de un proceso no son visibles para otro de los procesos. Como consecuencia, un proceso no puede leer o escribir el valor de la variable de otro proceso.
- Interacciones explícitas. El programador tiene que resolver todas las situaciones de interacción de procesos, tales como comunicación, sincronización y partición de datos.
- Localización explícita. Las tareas, y los datos para llevarlos a cabo, son asignadas a procesos por parte del programador. Es común que el programador escriba aplicaciones bajo el esquema de código único para reducir complejidad en el diseño y codificación de las mismas.

Dentro del modelo de programación de paso de mensajes, se encuentra el estándar MPI (*Message Passing Interface* o Interfase de Paso de Mensajes, por sus siglas en inglés). MPI es una especificación de biblioteca para proporcionar la funcionalidad de paso de mensajes a algún lenguaje de programación secuencial ya

definido. Es decir, MPI no es un lenguaje de programación en sí, sino que define la sintaxis y la semántica de un conjunto de rutinas útiles para un amplio grupo de usuarios. De esta manera, podemos encontrar implementaciones de MPI en C o Fortran, por mencionar algunos. La primera versión final de MPI se presentó en 1994 y surgió de la necesidad de contar con un estándar para la programación de sistemas de cómputo con memoria distribuida.

El objetivo principal de MPI es lograr la portabilidad a través de diferentes sistemas de cómputo paralelo. Este grado de portabilidad debe de ser comparable al de un lenguaje de programación que permita ejecutar, de manera transparente, aplicaciones sobre sistemas heterogéneos. Y esto se debe de lograr sin hacerlo a expensas del rendimiento. Entre las características generales de MPI tenemos:

- Ofrece seguridad en la manipulación de aplicaciones multihilo.
- Permite heterogeneidad para búferes estructurados y tipos de datos derivados.
- Comunicación punto a punto, con y sin bloqueo.
- Capacidad de comunicaciones colectivas con operaciones propias o definidas por el usuario.
- Definiciones para manejo del ambiente como, temporizadores, sincronizadores y control de errores.
- Provee soporte para topologías virtuales para procesos.

IV.3 Desempeño de Algoritmos Paralelos

Existen diferentes parámetros para determinar el desempeño de algoritmos paralelos. Los más comunes son factor de aceleración, eficiencia y costo.

El factor de aceleración puede tomarse desde dos puntos de vista. Por lo que se tiene el factor de aceleración absoluto y el relativo. El factor de aceleración absoluto está definido como la relación entre el tiempo requerido por el algoritmo secuencial más eficiente para realizar una tarea de cómputo y el tiempo requerido para realizar la misma tarea en una máquina incorporando paralelismo [QUINN, 1994]. Formalmente podemos describirlo como:

$$S(p) = \frac{Ts}{Tp} \tag{25}$$

De forma ideal, se espera un factor de aceleración lineal. Es decir, igual al número de procesadores. Sin embargo, existe la Ley de Amdahl la cual establece que todo algoritmo paralelo tendrá una fracción estrictamente secuencial, no paralelizable. Y por esta razón, el factor de aceleración es menor que un factor de aceleración lineal.

El factor de aceleración relativo es la relación entre el tiempo de ejecución de un programa paralelo utilizando un solo procesador y el tiempo de ejecución utilizando p procesadores. Esto es:

$$Sr(p) = \frac{T_{P=1}}{T_P} \tag{26}$$

El factor de aceleración relativo puede utilizarse para realizar una estimación de desempeño de un algoritmo paralelo conforme el número de procesadores utilizados se incrementa.

La eficiencia se define como la relación entre el factor de aceleración absoluto y el número de procesadores empleados. Así, se tiene que:

$$E = \frac{S(p)}{p} \tag{27}$$

Generalmente la eficiencia se expresa como porcentaje y nos proporciona información acerca del uso de los procesadores para la solución de una tarea. Una eficiencia del 75% indica que los procesadores, en promedio, son utilizados para cálculos tres cuartas partes del tiempo total de ejecución de un algoritmo.

El costo de un programa paralelo es:

$$Cp = pT_P \tag{28}$$

El costo nos da una medida del poder de cómputo utilizado al resolver un problema con un sistema de cómputo paralelo. De forma general, las técnicas de

paralelización se utilizan para maximizar el factor de aceleración y la eficiencia. Y para el costo, el objetivo es minimizarlo.

Segunda Sección

Implementación y Resultados Experimentales

Capítulo V

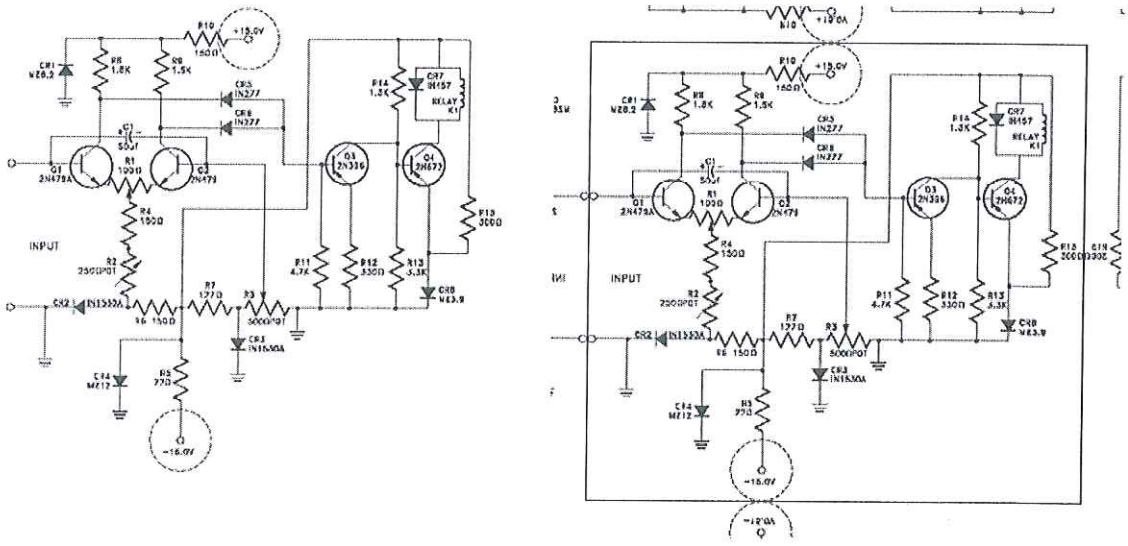
Implementación de Algoritmos de Filtros Morfológicos

En este capítulo se detalla la forma en que se llevó a cabo la implementación de los filtros morfológicos tanto de forma secuencial como paralela. También se describe las técnicas de paralelización utilizadas para la implementación de los distintos algoritmos.

V.1 Implementación de Algoritmos Secuenciales

V.1.1 Reflexión de Imagen

El procesamiento de imágenes con filtros morfológicos se lleva a cabo de manera local. Por esta razón es necesario un ajuste en los bordes de las imágenes de entrada. De esta forma se tiene información suficiente para realizar las comparaciones píxel a píxel entre imagen y elemento estructural, y llevar así a cabo la operación morfológica requerida.



A R_A

Figura 19. Imágenes, original y con reflexión de bordes.

En la Figura 19 se observa como se llevó a cabo la reflexión de bordes en una imagen binaria de un diagrama esquemático de un circuito electrónico. La reflexión es tipo espejo. Esto significa que los píxeles se repiten conservando la misma distancia respecto al borde al cual son reflejados.

V.1.2 Operaciones Matriciales

Las operaciones matriciales son intrínsecas en el procesamiento de imágenes. Esto se debe a que una imagen digital, en principio, se representa como una matriz de datos. De ahí que la mayoría de las operaciones se realicen en forma matricial. Además, el elemento estructural es una imagen en sí, y por lo tanto una matriz con la cual hay que realizar operaciones. Para el presente trabajo se implementaron las siguientes operaciones matriciales: unión, intersección, resta, copia, impresión en pantalla,

reflexión de bordes, reducción de bordes, descomposición por umbral de imagen en escala de gris y composición de imagen de en escala de gris a partir de imágenes binarias.

V.2 Implementación de Algoritmos Paralelos

V.2.1 Segmentación

Existe un paralelismo implícito en el procesamiento de una imagen. Teniendo en cuenta el paralelismo en el dominio de datos, una imagen puede ser segmentada en bloques. Así, de manera simultánea se puede procesar cada uno de los bloques y aprovechar dicho paralelismo.

En el uso de filtros morfológicos existe dependencia de datos ya que las transformaciones morfológicas se llevan a cabo de manera local. Generalmente, el procesamiento local de imágenes se representa como una ventana deslizante sobre la imagen. En el contexto de filtros morfológicos, el tamaño de dicha ventana es el tamaño del elemento estructural utilizado. Esta dependencia impide que la segmentación se lleve a cabo de manera sencilla ya que existirá información en los bloques vecinos necesaria para realizar el procesamiento de la imagen. Este problema se resuelve con traslape de datos. Dicho traslape se lleva a cabo generando información redundante en la vecindad de los bloques. De esta forma, se elimina la necesidad de comunicación entre procesos acerca de la información en los bordes del bloque de la imagen recibida. Este proceso de segmentación con traslape se observa claramente en la Figura 20. La imagen original se encuentra a la izquierda. El proceso de reflexión se llevó a cabo a lo largo del eje X de la

imagen. La segmentación en bloques se hizo a lo largo del eje *Y*. Se aprecia claramente como en los bordes superior y/o inferior se repite parte de la imagen debido al traslape.

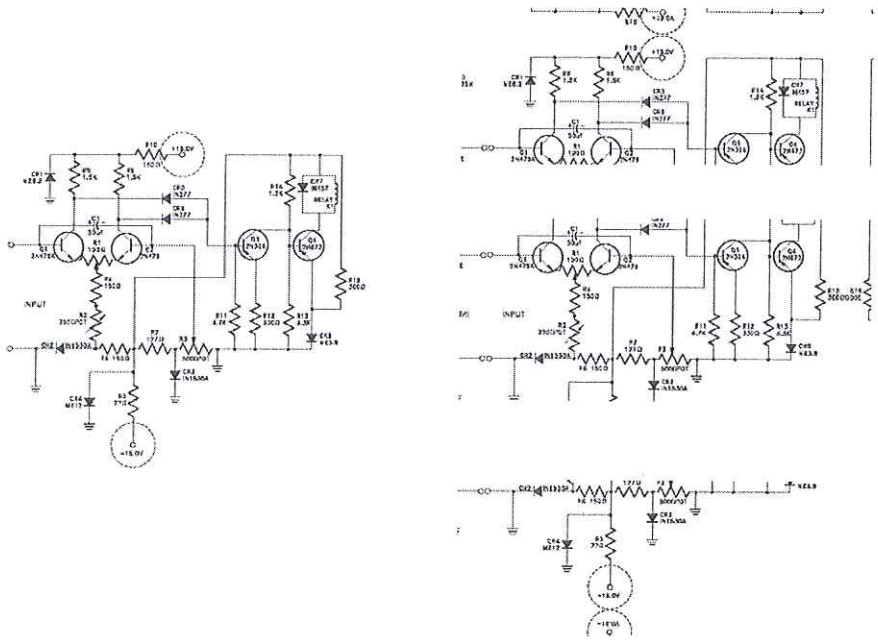


Figura 20. Original y segmentada, con reflexión de bordes y traslape.

V.2.2 Planos Binarios

Esta técnica de paralelismo se utiliza en el procesamiento de imágenes en escala de gris y color. Al utilizar la propiedad de apilamiento, comentada en el la sección II.2.1, una imagen puede dividirse en diferentes planos de imágenes binarias. Para una imagen en escala de gris representada con un byte por píxel, se generarán 256 planos binarios por medio de este procedimiento. Una imagen a color de 24 bits, generará 768 planos binarios. Es decir, 3 veces más que en una imagen en escala de gris.

Las diferentes imágenes binarias son distribuidas en los diferentes procesos del sistema paralelo utilizado, como se observa en la Figura 21.

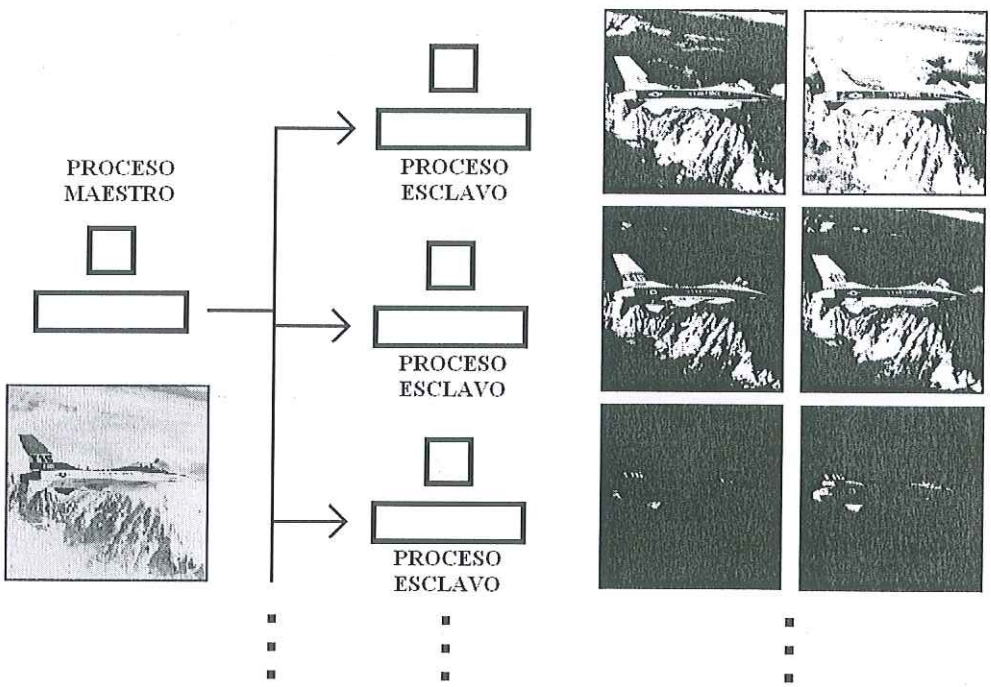


Figura 21. Distribución de planos binarios de una imagen en escala de gris.

Entonces, cada proceso tiene la responsabilidad de: recibir un conjunto de imágenes binarias, procesarlas con la transformación morfológica requerida y enviar los resultados al proceso maestro. Una vez recibidas todas las imágenes binarias procesadas, el proceso maestro utilizada nuevamente la propiedad de apilamiento para formar la imagen en escala de gris o color, según sea el caso, ya procesada.

V.2.3 Nivel de Instrucción

Los procesadores de cualquier sistema de cómputo utilizan internamente registros para realizar las operaciones requeridas por cualquier programa que se ejecute. Se llama

"tamaño de palabra" al número de bits que forman estos registros. Las operaciones de filtros morfológicos puede paralelizarse haciendo uso del tamaño completo de estos registros. Para llevarse a cabo una operación morfológica, se realizan comparaciones píxel a píxel entre la imagen de entrada y el elemento estructural. Sin embargo, un píxel de una imagen binaria solo requiere de un bit para ser representado. De esta forma, en lugar de realizar una comparación píxel a píxel de manera secuencial, se pueden realizar tantas comparaciones a la vez como el tamaño de la palabra de los registros del procesador. Este proceso se ilustra en la Figura 22.

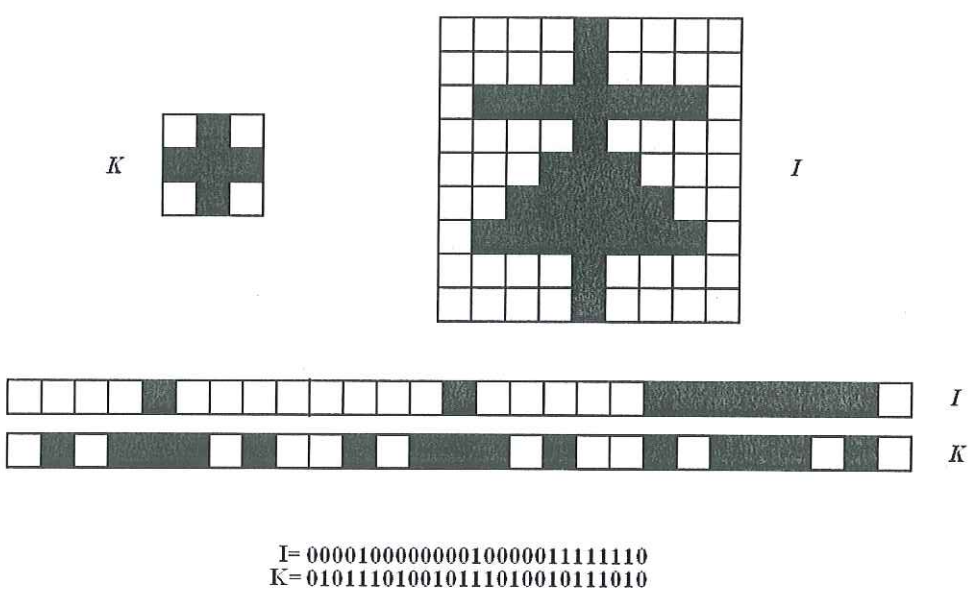


Figura 22. Paralelización en nivel de instrucción. a) Imagen de elemento estructural e imagen original. b) Tres primeras filas de imagen y tres veces el elemento estructural, empatados como arreglo lineal. c) Su representación como una palabra binaria.

Actualmente se cuenta con procesadores de 32 bits de uso común en computadoras personales. Procesadores de 64 bits se encuentran generalmente en supercomputadoras y ya pueden encontrarse también en algunas computadoras personales. Por esta razón se implementaron los algoritmos paralelos utilizando el paralelismo de nivel de instrucción, con palabras de 32 bits.

Capítulo VI

Resultados de Experimentos con Algoritmos Secuenciales y Paralelos de Filtros Morfológicos

VI.1 Características de Sistemas de Cómputo Paralelo Utilizados

Los equipos utilizados para realizar los experimentos de procesamiento morfológico de imágenes en paralelo, fueron Origin2000 y Calafia. Ambos equipos pertenecen a la red de supercómputo de CICESE. Sus características principales pueden observarse en la Tabla II.

Tabla II. Comparación de características de equipos paralelos.

	ORIGIN2000	CALAFIA
Número de Nodos	4	2
Procesadores por Nodo	2	4
Procesador	R10000 195 MHZ	UltraSPARC III 900 MHZ
Memoria por Nodo	256 MB	4 GB
Modelo de Memoria	Compartida/ Distribuida	Distribuida
Sistema Operativo	IRIX 6.4	SunOS 5.8

VI.2 Experimentos Realizados

Se realizaron dos tipos de experimentos con imágenes binarias. El primero fue la búsqueda de contorno de objetos en una imagen utilizando el procedimiento explicado en la sección III.2. La imagen es de una placa automotriz con tamaño 328x640 píxeles. El segundo experimento fue la localización de un objeto dentro de una imagen. Para este experimento se utilizó el procedimiento de la sección III.5. La imagen es un el diagrama de un circuito electrónico de tamaño 515x552. El objeto a localizar es el símbolo de una resistencia eléctrica representada por una imagen de tamaño 51x19 píxeles.

Con imágenes en escala de gris se realizaron dos tipos de experimentos. El primero fue filtrado de ruido impulsivo, visto en la sección III.3. Para este experimento se utilizó la imagen de un automóvil, de tamaño 288x358. El segundo tipo de experimento fue la transformada *top hat* descrita en la sección III.8.

VI.3 Tiempos de Ejecución

Se muestra en Figura 23 y Figura 24 los ejemplos más representativos respecto al comportamiento de los equipos, en cuanto a tiempos de ejecución, en los distintos experimentos. La primera corresponde a los tiempos de ejecución con Origin2000 con el algoritmo para la solución del problema de búsqueda de contorno de una imagen binaria. La segunda pertenece a los tiempos de ejecución del algoritmo para el mismo problema con Calafia.

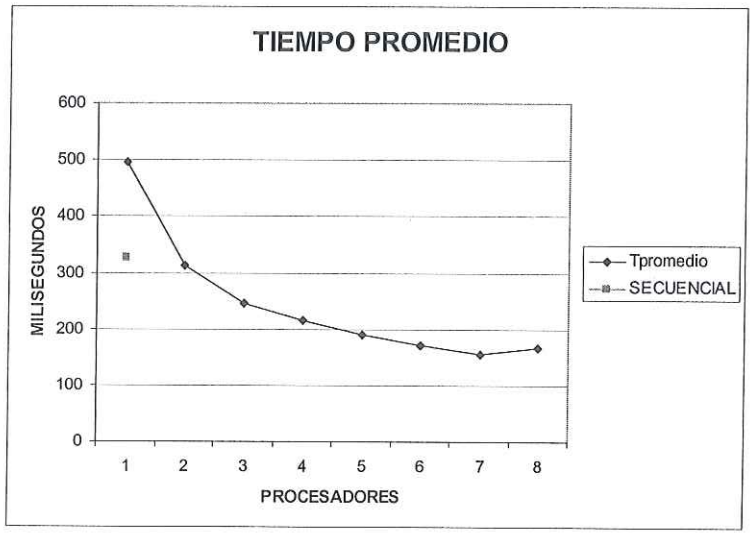


Figura 23. Tiempo de ejecución para búsqueda de contorno en Origin2000.

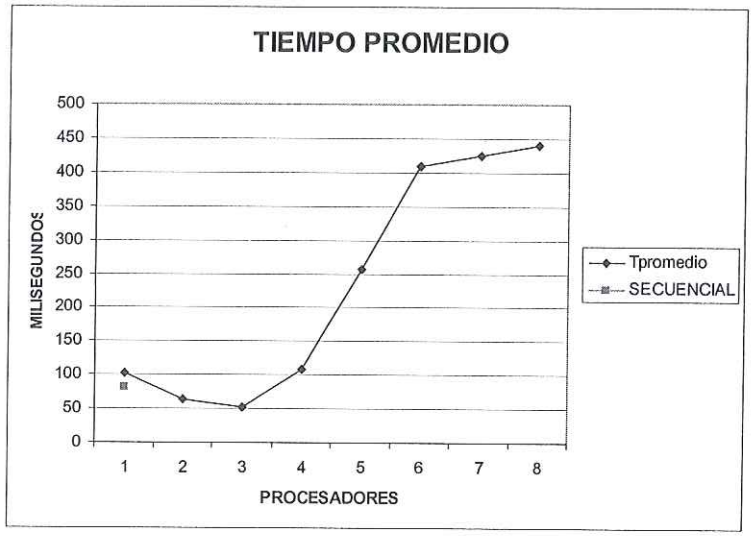


Figura 24. Tiempo de ejecución para búsqueda de contorno en Calafia.

En las dos figuras anteriores se observa como el programa secuencial siempre es más veloz que su contraparte paralela para un solo procesador. Esto era de esperarse ya que en el programa paralelo se realizan operaciones para ejecutar paralelismo, pero que en realidad sólo consumen tiempo de ejecución ya que se cuenta con un solo procesador.

Sin embargo, al utilizar más de un solo procesador, se observa como el tiempo de ejecución va disminuyendo con el aumento de procesadores. Para los experimentos con imágenes binarias con algoritmos paralelos ejecutados en Calafia, se observó un comportamiento como se muestra en la Figura 24. Se llega un punto en el cual, al aumentar el número de procesadores, el tiempo de ejecución aumenta. Esto puede deberse a que los procesos distribuidos a los diferentes procesadores son de tiempo semejante al tiempo requerido para distribuir dichos procesos. De tal forma, que es más costoso comunicar dichos procesos que procesarlos con un menor número de procesadores. También se observa como la arquitectura del sistema utilizado afecta el rendimiento de dichos sistemas. Para Calafia se tienen dos nodos, como se observó en la Tabla II, cada una con 4 procesadores. Por esta razón, se puede decir que la comunicación entre nodos es más costosa que la comunicación entre procesadores de un mismo nodo. Esto se ve reflejado en un aumento mayor del tiempo de ejecución al pasar de 4 a 5 procesadores, que el aumento que se tiene al pasar de 3 a 4 procesadores.

VI.3 Factor de Aceleración

Se realizó el cálculo de factor de aceleración absoluto, relativo y esperado para los diferentes experimentos realizados. Los resultados para los experimentos con imágenes binarias se muestran en las siguientes 4 figuras.

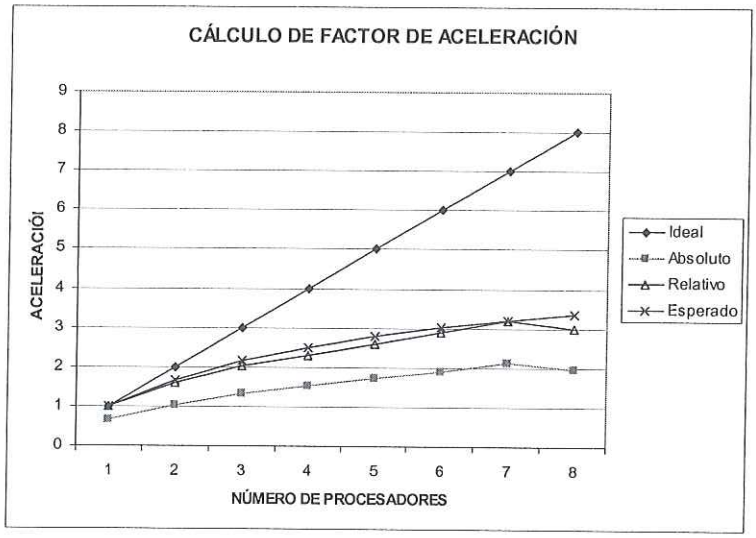


Figura 25. Factor de aceleración para búsqueda de contorno en Origin2000.

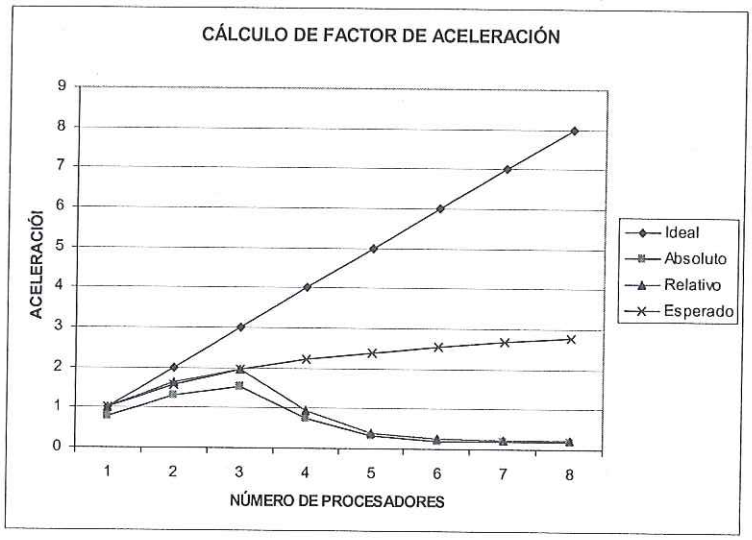


Figura 26. Factor de aceleración para búsqueda de contorno en Calafia.

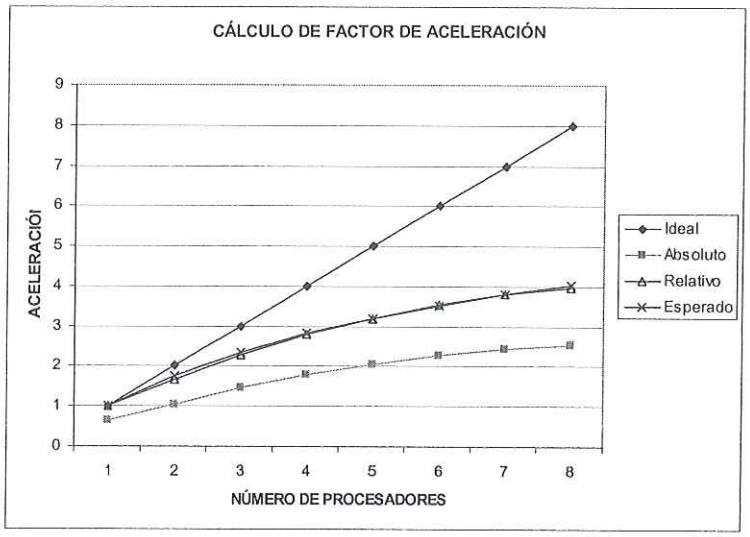


Figura 27. Factor de aceleración para localización de objetos en Origin2000.

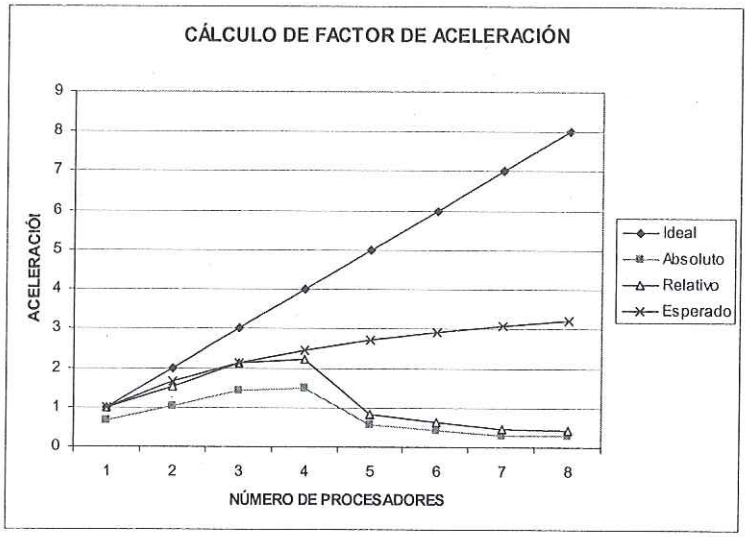


Figura 28. Factor de aceleración para localización de objetos en Calafia.

El cálculo de los diferentes factores de aceleración depende directamente del tiempo de ejecución. Por esta razón, los factores de aceleración calculados para los experimentos con Calafia sufren una disminución al aumentar el número de

procesadores, especialmente al cambiar de 4 a 5 procesaores. Esto simplemente es reflejo de lo explicado en la sección anterior.

A continuación se muestran las gráficas del cálculo de factores de aceleración para los experimentos con imágenes en escala de gris.

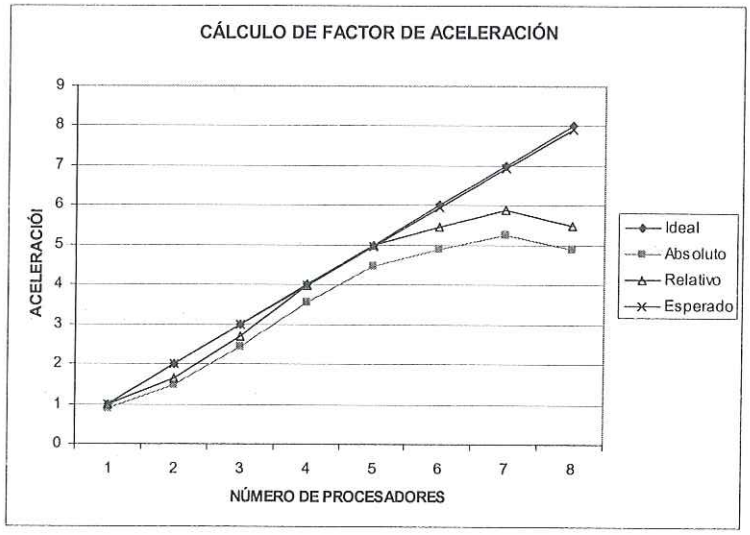


Figura 29. Factor de aceleración para filtrado de ruido impulsivo en Origin2000.

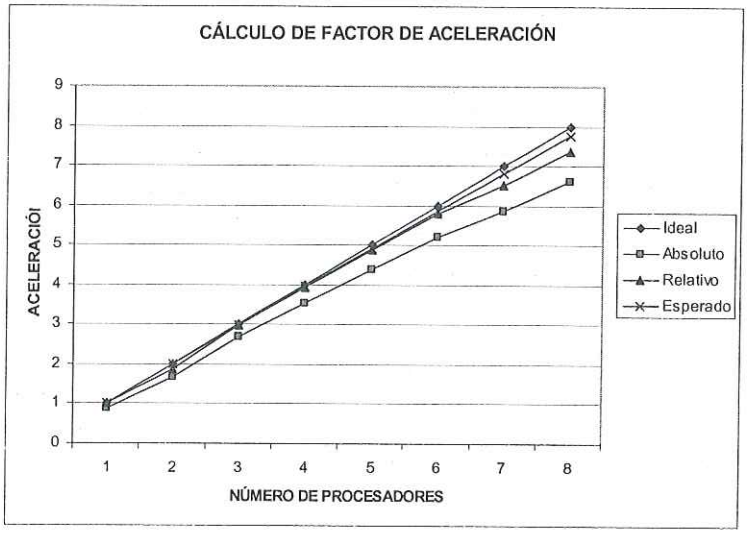


Figura 30. Factor de aceleración para filtrado de ruido impulsivo en Calafia.

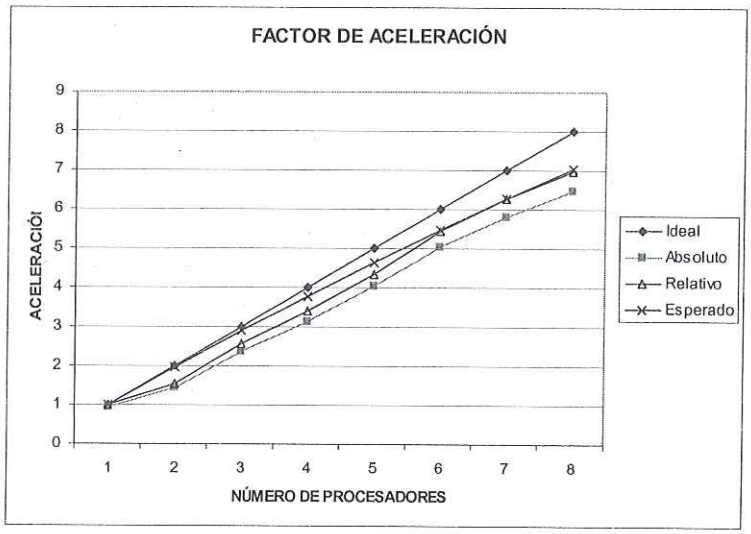


Figura 31. Factor de aceleración para transformada top hat en Origin2000.

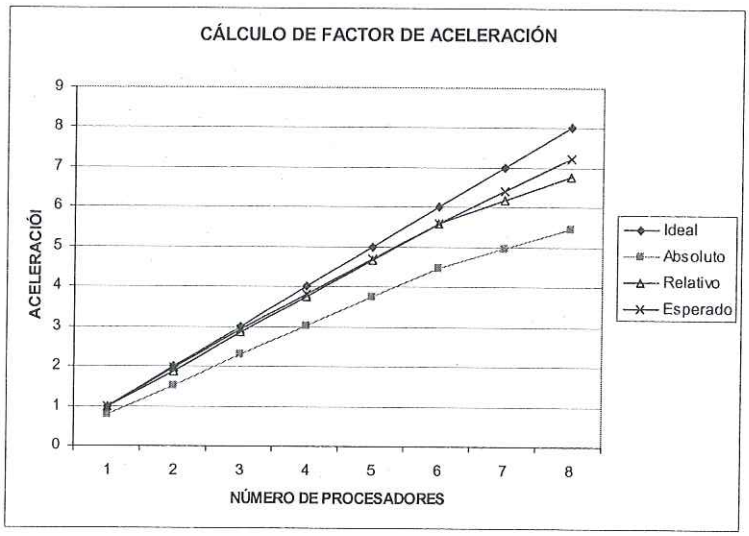


Figura 32. Factor de aceleración para transformada top hat en Calafia.

En estas gráficas se aprecia como el aumento en el número de procesadores es conveniente para este tipo de tareas en ambos equipos. Esto se debe a que realmente se están utilizando los procesadores para labores de cómputo y el tiempo de espera para la

comunicación de procesos es relativamente pequeño en comparación con el tiempo de procesamiento.

Tercera Sección

Análisis y Conclusiones

Capítulo VII

Análisis de Algoritmos Implementados

En este capítulo se hace el análisis teórico de complejidad de los distintos algoritmos implementados. Para llevarlos a cabo se hacen las siguientes consideraciones.

La imagen de entrada es un arreglo matricial de n elementos. Para las imágenes binarias, cada pixel es representado por un byte que puede tener alguno de dos valores posibles, 1 y 0. Para imágenes en escala de gris, cada elemento de la imagen es un byte que puede tomar un valor entre 0 y 255. El elemento estructural está representado por una imagen de k elementos.

VII.1 Análisis Teórico de Complejidad de Algoritmos Secuenciales de Filtros Morfológicos

Es importante establecer la complejidad de los algoritmos implementados de forma secuencial, porque esto nos permite conocer el tiempo de ejecución de los mismos [Jajá, 1992]. Como primer paso para llevar a cabo este análisis, se toman en cuenta las operaciones matriciales requeridas por los distintos algoritmos. Estas operaciones son: suma-unión, resta, intersección y umbralización. Todas estas operaciones son elemento a elemento. Por esta razón el orden de todas estas operaciones es:

$$O(n) \tag{29}$$

Además se tienen otras operaciones necesarias para procesar las imágenes que son reflexión y reducción. Dichas operaciones se consideran elemento a elemento pues prácticamente son una copia, aumentada o reducida, de la imagen original.

Las siguientes operaciones a considerar son erosión y dilatación de imágenes binarias. Como se vio en el Capítulo II, estas operaciones se llevan a cabo de manera local utilizando un elemento estructural. Siguiendo las consideraciones al inicio de este capítulo, la complejidad está dada por el número de comparaciones requeridas por el elemento estructural con los píxeles de la imagen. Esto es:

$$O_{E_B} = O_{D_B} = O(kn) \tag{30}$$

Para la apertura y cerradura se requiere sumar la complejidad de las dos operaciones básicas. Esto da por resultado:

$$O_{A_B} = O_{C_B} = O(kn) + O(kn) = O(2kn) \tag{31}$$

Aún cuando $2k$ es constante, se expresa la complejidad de esta forma para tener en cuenta el tamaño posible de esta constante.

Haciendo uso de las expresiones anteriores, podemos establecer la complejidad de las aplicaciones con filtros morfológicos. Como ejemplos, en la Figura 33 se muestran los diagramas de flujo de los algoritmos de búsqueda de contorno y filtrado de ruido impulsivo.

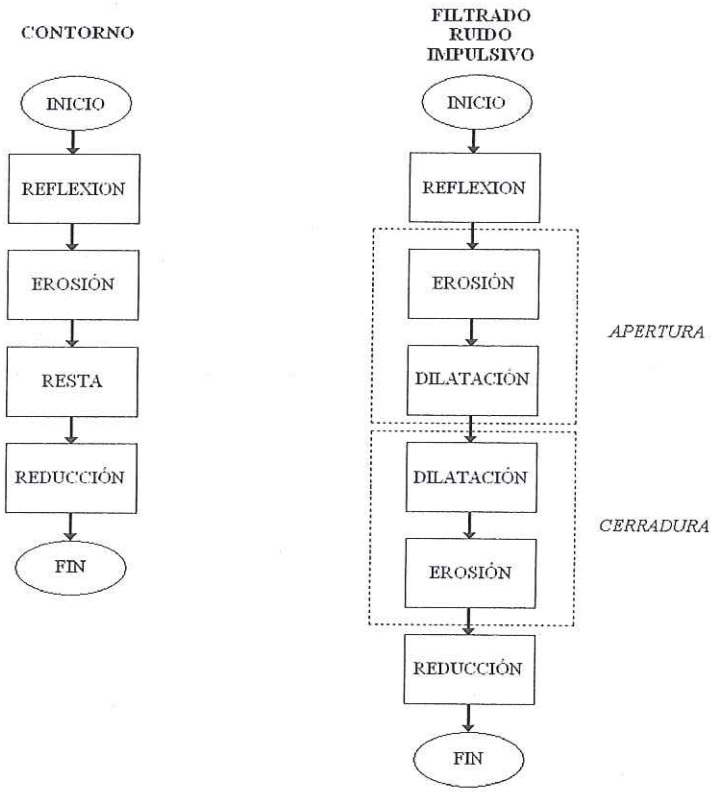


Figura 33. Diagramas de flujo de contorno y de filtro de ruido impulsivo.

Siguiendo las operaciones requeridas para estos dos casos, se puede establecer que la complejidad de la transformación de contorno es:

$$O_{C_B} = O(n) + O(kn) + O(n) + (On) = O((3+k)n) = O(kn) \qquad (32)$$

Y para la aplicación de filtrado de ruido impulsivo tenemos que:

$$O_{FRI_B} = O(n) + O(kn) + O(kn) + O(kn) + O(kn) + O(n) = O((2+4k)n) = O(4kn) \quad (33)$$

Las expresiones obtenidas en la sección anterior nos sirven para establecer la complejidad de los algoritmos secuenciales para su aplicación en imágenes en escala de gris. Esto se debe a que las operaciones de imágenes en escala de gris para este trabajo se basan en procesamiento de imágenes binarias.

Para la operación de erosión, se establece que:

$$O_{E_G} = O_{D_G} = mO_{E_B}\big|_{m=256} = O(256kn) \tag{34}$$

La dilatación tendrá la misma complejidad. Y en base a esta expresión, podemos decir que la complejidad para la apertura en una imagen en escala de gris está dada por:

$$O_{A_G} = O_{C_G} = mO_{A_B}\big|_{m=256} = O(512kn) \tag{35}$$

Y la operación de cerradura tendrá la misma complejidad. En la Figura 34 se muestra el diagrama de flujo para obtener la transformada *top hat*.

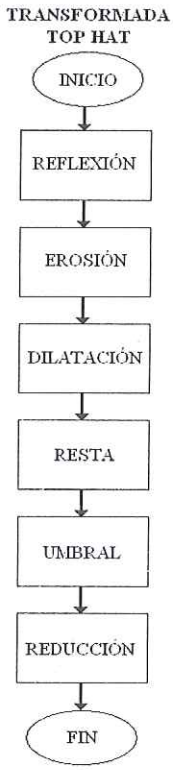


Figura 34. Diagrama de flujo de transformada *Top Hat*.

Tomando en cuenta las operaciones necesarias para llevar a cabo la transformada *top hat*, se puede establecer la complejidad de esta transformación morfológica como:

$$O_{TH_B} = O(n) + O(256kn) + O(256kn) + O(n) + O(n) + O(n) = O(512kn) \quad (36)$$

En la Tabla III, se plasman las expresiones reducidas para las complejidades de las distintas operaciones morfológicas de imágenes binarias y en escala de gris. En esta tabla, dos operaciones morfológicas cuentan con un factor i en la expresión de su complejidad, rellenado de objetos huecos y esqueleto morfológico. Este factor expresa un proceso iterativo. El número de iteraciones, o criterio de paro, dependerá de los objetos contenidos en la imagen.

Tabla III. Complejidades de algoritmos secuenciales.

Operación	Imagen Binaria	Escala de Gris
Erosión	*O(kn)*	*O(256kn)*
Dilatación	*O(kn)*	*O(256kn)*
Apertura	*O(2kn)*	*O(512kn)*
Cerradura	*O(2kn)*	*O(512kn)*
Filtrado de Ruido Impulsivo	*O(4kn)*	*O(1024kn)*
Contorno	*O(kn)*	*O(256kn)*
Hit or Miss	*O(2kn)*	*O(512kn)*
Rellenado de Objetos Huecos	*O(ikn)*	*O(256ikn)*
Esqueleto Morfológico	*O(ikn)*	*O(256ikn)*
Transformada *Top Hat*		*O(512kn)*

VII.2 Análisis Teórico de Complejidad de Algoritmos Paralelos de Filtros Morfológicos

Las técnicas de paralelización utilizadas en este trabajo dependen de la imagen de entrada. Esto es, si la imagen es binaria o en escala de gris. Por esta razón se dividió esta sección en dos análisis separados, uno para imágenes binarias y otro para imágenes en escala de gris. Estos análisis se refieren únicamente al número de operaciones requeridas para llevar a cabo alguna transformación morfológica. No se toma en cuenta, por

ejemplo, el tiempo requerido para envío y recepción de los datos por los distintos procesos, o el efecto de barreras de sincronización, por mencionar algunos.

VII.2.1 Complejidad en Imágenes Binarias

Para las operaciones básicas de erosión y dilatación, se tiene la expresión de O_{E_B} y O_{D_B}. Las opciones que se tienen para paralelizar este tipo de operaciones son segmentación de la imagen de entrada y paralelismo a nivel de instrucción. Tomando en cuenta esto, se puede decir que:

$$O_{E_B} = O_{D_B} = O(\frac{kn}{wp}) \tag{37}$$

En la expresión anterior, w simboliza el tamaño de la palabra dentro del procesador. La variable p se refiere al número de procesadores. La expresión anterior nos indica que el problema de entrada sigue siendo el mismo, con complejidad $O(kn)$. Sin embargo, el tiempo de ejecución se ve disminuido por un factor de $O(w^{-1}p^{-1})$. Esto se debe a que la segmentación de la imagen de entrada reduce el tiempo de ejecución al dividir la imagen con p procesadores. Y, de forma simultánea, se pueden procesar w bits a la vez, donde cada bit representa un píxel de la imagen. Esto reduce el número de operaciones por el factor w.

Para las operaciones de apertura y cerradura, la complejidad la podemos expresar como:

$$O_{A_B} = O_{C_B} = O(\frac{kn}{wp}) + O(\frac{kn}{wp}) = O(\frac{2kn}{wp}) \quad (38)$$

Esto significa que el mismo factor de reducción en el tiempo de ejecución $O(w^{-1}p^{-1})$ se mantiene para las diferentes operaciones morfológicas de apertura y cerradura, en comparación con su contraparte secuencial. De manera general, se puede decir que este factor se presenta en todas las demás operaciones morfológicas ya que tienen como base el uso de erosión y dilatación.

VII.2.2 Complejidad en Imágenes en Escala de Gris

Para imágenes en escala de gris, las opciones de paralelización son procesamiento de planos binarios y nivel de instrucción. Teniendo en consideración esto y la complejidad del problema en secuencial, dada por la expresión de O_{E_G}, se puede establecer que para la erosión y la dilatación se tiene:

$$O_{E_G} = O_{D_G} = O(\frac{256kn}{wp}) \quad (39)$$

Para las operaciones de apertura y cerradura, la complejidad la podemos expresar como:

$$O_{A_G} = O_{C_G} = O(\frac{256kn}{wp}) + O(\frac{256kn}{wp}) = O(\frac{512kn}{wp}) \quad (40)$$

Como se explicó en el Capítulo III, no todas las operaciones de un algoritmo pueden paralelizarse. Indudablemente esto afecta el tiempo de ejecución de un algoritmo paralelo. Para ejemplificar ésto, nuevamente se presenta el algoritmo de filtrado de ruido impulsivo en la Figura 35. En esta ocasión, se muestran sombreadas las operaciones que pueden realizarse en paralelo.

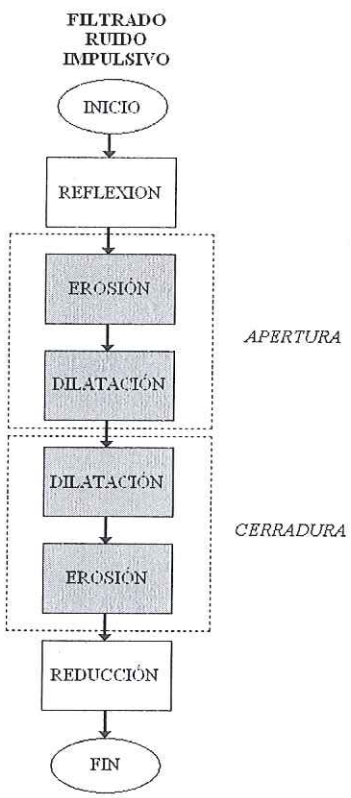


Figura 35. Diagrama de flujo para algoritmo paralelo de filtrado de ruido impulsivo.

Entonces, el tiempo de ejecución de este algoritmo puede expresarse como:

$$O_{FRI_G} = O(n) + O(\frac{512kn}{wp}) + O(\frac{512kn}{wp}) + O(n) = O(2n + \frac{1024kn}{wp}) \qquad (41)$$

A pesar de que en el presente trabajo se conoce el valor de w y se sabe que $p<<1024$, es conveniente dejar sin reducción esta expresión. De esta forma, todavía será válida para el caso en que se varíe el valor de w o que $p \approx 1024$. También, es más claro qué operaciones se están realizando en código secuencial y cuáles han sido paralelizadas. En la Tabla IV se muestran los distintos algoritmos paralelos con sus distintas complejidades para imágenes binarias y en escala de gris.

Tabla IV. Complejidades de algoritmos paralelos.

Operación	Imagen Binaria	Escala de Gris
Erosión	$O(2n+\frac{kn}{wp})$	$O(2n+\frac{256kn}{wp})$
Dilatación	$O(2n+\frac{kn}{wp})$	$O(2n+\frac{256kn}{wp})$
Apertura	$O(2n+\frac{2kn}{wp})$	$O(2n+\frac{512kn}{wp})$
Cerradura	$O(2n+\frac{2kn}{wp})$	$O(2n+\frac{512kn}{wp})$
Filtrado de Ruido Impulsivo	$O(2n+\frac{4kn}{wp})$	$O(2n+\frac{1024kn}{wp})$
Contorno	$O(2n+\frac{kn}{wp})$	$O(2n+\frac{256kn}{wp})$
Hit or Miss	$O(2n+\frac{2kn}{wp})$	$O(2n+\frac{512kn}{wp})$
Rellenado de Objetos Huecos	$O(2n+\frac{ikn}{wp})$	$O(2n+\frac{256ikn}{wp})$
Esqueleto Morfológico	$O(2n+\frac{ikn}{wp})$	$O(2n+\frac{256ikn}{wp})$
Transformada *Top Hat*		$O(2n+\frac{512kn}{wp})$

Capítulo VIII

Conclusiones

En este trabajo se cubrieron dos áreas de las ciencias de la computación: procesamiento de imágenes y cómputo paralelo. La principal aportación de este trabajo es un sistema algorítmico paralelo de procesamiento de imágenes utilizando filtros morfológicos. En base a este sistema, se pueden desarrollar un mayor número de aplicaciones a las ya implementadas en esta tesis.

Se hizo uso en la programación paralela de *MPI*, que permite una gran portabilidad entre distintos sistemas de cómputo paralelo. Esto se llevó a cabo al hacer implementaciones que se ejecutaron en la supercomputadora Origin2000 y en el cluster Calafia, del departamento de supercómputo de CICESE. Para el balanceo de cargas se utilizó el paradigma maestro-esclavo, el cual permitió la distribución de los procesos en el total de los procesadores disponibles para cada experimento.

En este trabajo se implementaron las siguientes aplicaciones:

- Filtrado de ruido impulsivo
- Transformada *top hat*
- Localización de objetos
- Detección de objetos
- Reconstrucción de textos
- Búsqueda de contorno
- Cálculo de dimensiones

- Esqueleto morfológico

Se observó que un algoritmo paralelo se ve beneficiado al aumentar el tamaño de las imágenes o la cantidad de procesamiento requerido por las mismas. Esto se debe a que aumenta la fracción paralela de dicho algoritmo, o bien, que disminuye la fracción secuencial. Como consecuencia se logra un mayor factor de aceleración lo cual es una característica deseada en todo algoritmo paralelo. Esto se observó claramente en los experimentos realizados. Para los casos de imágenes binarias el máximo factor de aceleración absoluto fue de 2.5 para 8 procesadores. En cambio, para los experimentos con imágenes en escala de gris fue de 6.61. La relación de porcentajes de fracción paralelizable para estos casos fue de 85% y 99% respectivamente.

Se implementaron algoritmos con alto y bajo nivel de paralelismo. Esto permite mejorar el desempeño de dichos algoritmos que el utilizar simplemente una sola técnica de paralelización. Se utilizaron como técnicas de paralelización: segmentación y traslape, en alto nivel, y nivel de instrucción, en bajo nivel.

Se dedujo la necesidad de encontrar un balance entre el tamaño del problema, el algoritmo a utilizar y el sistema paralelo donde se implementa dicho algoritmo. Esto se debe a que si el tamaño del problema no es adecuado, se puede perder eficiencia significativa al aumentar el número de procesadores de nuestro sistema de cómputo paralelo, aún cuando se esperaría exactamente lo opuesto. Esto se dedujo al observar los experimentos en imágenes binarias donde el sistema Calafia bajaba su factor de aceleración hasta 0.19 al aumentar el número de procesadores. En cambio, esto no sucedía con el sistema Origin2000 el cual llegaba a un valor 2.53 de factor de aceleración absoluto para los mismos experimentos.

Se realizó un análisis teórico de complejidades de los distintos algoritmos de filtros morfológicos mostrados en este trabajo. Este estudio incluyó el análisis de algoritmos tanto secuenciales como paralelos. Tanto en el análisis de experimentos como en el análisis teórico se observó que mientras el tamaño del problema aumente, por una imagen de mayor tamaño o bien, con mayor información, existe la posibilidad de explotar una mayor fracción de ejecución paralela dentro de un algoritmo.

VIII.1 Trabajo Futuro

- La creación de una interfaz de programación funcional para el sistema algorítmico creado. Esto permitiría su utilización didáctica dentro de los cursos de procesamiento de imágenes.
- El desarrollo de nuevas funciones en base a este sistema para hacerlo más robusto y permitir atacar un mayor número de problemas.
- La implementación de aplicaciones en tiempo real a fin de aprovechar el desempeño logrado por las funciones ya implementadas en paralelo.

REFERENCIAS

Buyya, R. 1999. High Performance Cluster Computing, Vol I: Architectures and Systems. Ed. Prentice Hall. Primera Edición. NJ, EE.UU. 849 p.

Buyya, R. 1999. High Performance Cluster Computing, Vol II: Programming and Applications. Ed. Prentice Hall. Primera Edición. NJ, EE.UU. 664 p.

González, R. y R. Woods. 2002. Digital Image Processing. Ed. Prentice Hall. Segunda Edición. Boston, EE.UU. 793 p.

Heijmans, H. J. A. M.. 1997. Composing Morphological Filters. IEEE Transactions on Image Processing. 6(5). 713-723 p.

Hwang, K. y Z. Xu. 1998. Scalable Parallel Computing. Ed. Wcb/Mcgraw-Hill. Primera Edición. New York, EE. UU. 802 p

Ikenaga, T. y T. Ogura. 2000. Real-Time Morphology Processing Using Highly Parallel 2-D Cellular Automata Cam2. IEEE Transactions on Image Processing. 47(7): 788-801 p.

Jackway, P.T. y M. Deriche. 1996. Scale-Space Properties of the Multiscale Morphological Dilation-Erosion. IEEE Transactions on Pattern Analysis and Machine Intelligence. 18(1): 38-51 p.

Jain, A. 1989. Fundamentals of Digital Image Processing. Ed. Prentice Hall. Primera Edición. NJ, EE.UU. 569 p.

Jájá, J. 1992. An Introduction to Parallel Algorithms. Ed. Addison-Wesley. Primera Edición. New York, EE.UU. 365 p.

Kober, V., M. Mozerov y J. Alvarez-Borrego. 2002. Unsharp Masking By The Rank-Order Filters With Spatially Adaptive Neighborhoods. Pattern Recognition and Image Analysis. 12(1): 46-56 p.

Koivisto, P. y A. Antti. 2004. Breakdown Probabilities of Recursive Soft Morphological Filters. ACM International Conference Proceeding Series. Proceedings of the Winter International Synposium on Information and Communication Technologies. 58: 1-6 p.

Kumar, V.K.P 1991. Parallel Architectures and Algorithms for Image Understanding. Ed. Academic Press. New York, EE.UU. 562 p.

Maragos, P 1989. A Representation Theory for Morphological Image and Signal Processing. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2(6): 586-599 p.

Maragos, P. y R. Ziff. 1990. Threshold Superposition in Morphological Image Analysis Systems. IEEE Transactions on Pattern Analysis and Machine Intelligence. 12(5): 498-504 p.

Marion, A. 1991. An Introduction to Image Processing. Ed. Chapman and Hall. Londres, Inglaterra. 424 p.

Parhami, B. 1999. Introduction to Parallel Processing. Ed. Plenum. Primera Edición. Norwell, EE.UU. 556 p.

Pitas, I. 1993. Parallel Algorithms For Digital Image Processing, Computer Vision And Neural Networks. Ed. Prentice-Hall. Primera Edición. NJ, EE.UU. 350 p.

Quinn, M.J. 1994. Parallel Computing. Ed. Mcgraw-Hill. Segunda Edición. New York, EE. UU. 446 p.

Seinstra, F.J. D. Koelma y J.M. Geusebroek. 2002. A Software Architecture for User Transparent Parallel Image Processing. Elsevier Science Publishers. 28(7-8): 967-993 p.

Serra, J. 1983. Image Analysis and Mathematical Morphology. Ed. Academic Press. Primera Edición. Orlando, EE.UU. 310 p

Shih, F. y Y.T. Wu. 2005. Decomposition of Binary Morphological Structuring Elements Based on Genetic Algorithms. Computer Vision and Image Understanding. 99(2): 291-302 p.

Svolos, A. C., G. Konstantopoulos y C. Kaklamanis. 2000. Efficient Binary Morphological Algorithms on a Massively Parallel Processor. Proceedings of the 14th International Symposium on Parallel and Distributed Processing. 281 p.

Vincent, L. 1993. Morphological Grayscale Reconstruction in Image Analysis. Applications and Efficient Algorithms. IEEE Transactions on Image Processing. 2(2): 176-201 p.

[1] http://www.top500.org/stats/26/archtype/. Consultado: 20 de agosto de 2006.

[2] http://ei.cs.vt.edu/~history/Parallel.html. Consultado: 20 de agosto de 2006.