UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN ALGORITMO BASADO EN EL DOMINIO DE LAS FRECUENCIAS Y PROCESAMIENTO EN PARALELO PARA LA GENERACIÓN DE UNA IMAGEN MÉDICA EN SUPER RESOLUCIÓN

TESIS

PARA OPTAR EL TITULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

CRISTIAN TINOCO YURIVILCA

PROMOCIÓN
2011-II

LIMA – PERU
2014

IMPLEMENTACIÓN DE UN ALGORITMO BASADO EN EL DOMINIO DE LAS FRECUENCIAS Y PROCESAMIENTO EN PARALELO PARA LA GENERACIÓN DE UNA IMAGEN MÉDICA EN SUPER RESOLUCIÓN

DEDICATORIA

A Dios porque es mi fortaleza y mi guía en la vida, amor y dedicación

A mis padres, por su comprensión y paciencia, ejemplo de vida a seguir, quienes permitieron que cada logro en mi vida sea gracias a ellos.

A mis hermanos, por su paciencia quienes con cada consejo y enseñanza hicieron que lo difícil sea fácil.

SUMARIO

En esta tesis se muestra el desarrollo, implementación y los resultados del algoritmo que permite generar una imagen en súper resolución usando procesamiento en paralelo. Se presenta al médico especialista, una aplicación práctica del algoritmo que mejora la calidad de las imágenes médicas, con el fin de proporcionar al médico, una herramienta útil de imágenes en súper resolución, para facilitar la formulación de un diagnóstico más seguro y más preciso.

El capítulo 1 inicia con la definición de una imagen médica y los métodos que se aplican para su obtención, se presenta la teoría necesaria para la generación de una imagen en súper resolución, termina con la teoría básica de procesamiento en paralelo desde el punto de vista de hardware y software.

En el capítulo 2 se muestra el desarrollo matemático que permite la generación de una imagen en súper resolución a partir de varias imágenes en baja resolución. El capítulo finaliza con el método para el cual, parte del algoritmo se programa para su ejecución en una unidad de procesamiento grafico (GPU) que realiza procesamiento en paralelo.

En el capítulo 3 se muestra las pruebas realizadas aplicando el Toolbox Computer Vision System (herramienta en Matlab), con el fin de optimizar los resultados en la calidad obtenida de una imagen en súper resolución.

En el capítulo 4 se muestran los resultados obtenidos con el algoritmo aplicado sobre las imágenes médicas de ecografía y tomografía, se comparan los resultados con el método de interpolación bicúbica que es ampliamente usado por las aplicaciones de edición de imágenes como por ejemplo Adobe Photoshop.

ÍNDICE

PRÓLOGO 1

CAPÍTULO I

MARCO TEÓRICO 2

1.1 Imágenes Médicas 2
1.1.1 Radiografía Convencional 2
1.1.2 Radiografía Dental 3
1.1.3 Mamografía 4
1.1.4 Tomografía Computarizada 4
1.1.5 Ecografía 5
1.2 Calidad de las Imágenes Médicas Digitales 8
1.2.1 Representación Matemática de una Imagen Digital 8
1.2.2 Resolución Espacial, de Densidad y Relación Señal a Ruido (SNR) 9
1.2.3 Error Cuadrático Medio (MSE) y Relación Señal a Ruido Pico (PSNR) 11
1.2.4 Función Dispersión de un Punto (Point Spread Function - PSF) 11
1.3 Transformada Discreta de Fourier (TDF) 14
1.3.1 La Transformada Discreta de Fourier y su Inversa 14
1.3.2 La 2D Transformada Discreta de Fourier y su Inversa 15
1.3.3 Teorema de Muestreo y Fenómeno de Traslape (Aliasing) 15
1.4 Método de Mínimos Cuadrados 16
1.4.1 Descomposición en Valores Singulares 16
1.5 Correlación Cruzada 17
1.6 Súper Resolución 18
1.6.1 Aplicaciones 18
1.6.2 Problema de Reconstrucción de una Imagen en Súper Resolución 18
1.7 Computación en Paralelo 20
1.7.1 Arquitectura de Computación en Paralelo 21
1.7.2 Organización de Computadores con Memoria Compartida 24
1.7.3 Paralelismo a Nivel de Hilos 26
1.7.4 Procesadores Multinúcleo 27
1.8 Unidades de Procesamiento Gráfico (GPU) 27
1.8.1 Arquitectura de una GPU Moderna 27

1.8.2 Arquitectura del Hardware GPU NVIDIA 28
1.8.3 Computación Heterogénea CPU y GPU 29
1.9 Principios Para el Diseño de Algoritmos que se Ejecutan en Paralelo 30
1.9.1 Tipos de Paralelismo 31
1.9.2 Concurrencia 32
1.9.3 Granularidad 32
1.9.4 Sincronización y Datos Compartidos 33
1.9.5 Procesos e Hilos 33
1.9.6 Ley de Amdahl 34
CAPÍTULO II
DESARROLLO DEL ALGORITMO 36
2.1 Estimación de los Parámetros de Desplazamiento 37
2.1.1 Estimación del Desplazamiento entre dos Imágenes 37
2.1.2 Estimación de la Rotación entre dos Imágenes 41
2.2 Generación de una Imagen en Súper Resolución 45
2.2.1 Fenómeno de Anillos 50
2.2.2 Filtro Bilateral 52
2.3 Software para la Implementación del Algoritmo 54
2.4 Implementación del Algoritmo para la Generación de la Imagen en Súper Resolución Mediante el Procesamiento en Paralelo 54
CAPÍTULO III
RECOLECCIÓN DE DATOS Y PRUEBAS PARA OPTIMIZAR LA EFICIENCIA DE LOS ALGORITMOS UTILIZADOS 57
3.1 Recolección de Datos 57
3.2 Pruebas Sintéticas 57
3.2.1 Efecto del Método Empleado para Incrementar o Disminuir la Tasa de Muestreo 57
3.2.2 Efecto de la Elección de las Componentes en Frecuencia para la Generación del Sistema Lineal de Ecuaciones 58
3.2.3 Efecto de la Precisión en la Estimación de la Rotación 61
3.2.4 Efecto del Número de Imágenes del Registro Sobre la Súper Resolución 64
3.3 Límites de Procesamiento y Medición del Tiempo para Generar la Imagen en Súper Resolución 64
CAPÍTULO IV
RESULTADOS 69
4.1 Resultados Obtenidos Mediante Pruebas Sintéticas 69
4.2 Resultados Sobre Imágenes Médicas 71
CONCLUSIONES Y RECOMENDACIONES 74
ANEXO A: FORMATO DE IMÁGENES 76
ANEXO B: OPENCV – VISIÓN POR COMPUTADORA 80

ANEXO C: PROGRAMA EN PARALELO VIA OPENCV Y CUDA **82**
ANEXO D: CÓDIGO DE LOS PROGRAMAS EJECUTADOS SOBRE UNA GPU **88**
BIBLIOGRAFÍA **96**

PRÓLOGO

El uso de métodos no invasivos para poder observar el interior del cuerpo humano, ha permitido realizar el análisis y estudio del cuerpo humano en una forma más sencilla y menos riesgosa para el paciente.

Desde el descubrimiento de los rayos X y su aplicación en la medicina para el análisis de la estructura ósea del cuerpo humano, se han mejorado las técnicas para adquirir las imágenes proporcionando cada vez mayor información al médico. En los sistemas actuales modernos, la adquisición de una imagen médica se encuentra en formato digital. Esto permite el uso de nuevas técnicas para el análisis de las imágenes y su manipulación como: la segmentación, visualización en 3d para el caso de tomografía, detector de bordes, etc.

Desde hace 20 años se han desarrollado métodos basados en el tratamiento digital de imágenes. En la presente tesis se pretende darle una aplicación específica en el ámbito médico con la finalidad de ser un punto de partida a futuras investigaciones, mediante el método de súper resolución que busca lograr un diagnostico cada vez más temprano de una enfermedad.

Durante su labor, un médico radiólogo analiza cientos de imágenes diarias, haciendo que el tiempo empleado para su análisis sea un parámetro importante y crítico. La aplicación del método de súper resolución en el sistema real debe procesarse de la forma más eficiente posible acelerando el tiempo de procesamiento. Por ende se usa la tecnología del procesamiento en paralelo que permite la ejecución de varios procesos en forma simultánea.

El método de investigación aplicado fue básicamente: búsqueda de información en el internet, revisión de artículos científicos publicados en revistas prestigiosas, lectura de libros para un entendimiento profundo de los conceptos inherentes al ámbito de investigación. En muchas situaciones la solución técnica en la implementación del algoritmo fue solucionado con la información contenida en los foros tecnológicos.

Agradecimiento especial al Ph. D. Aldo Camargo Fernandez Baca por el apoyo durante el desarrollo del proyecto y al Ing. Briceño por su asesoría para la formulación de la presente tesis.

CAPÍTULO I
MARCO TEÓRICO

1.1 Imágenes Médicas

Una imagen médica, es aquella que proviene del conjunto de técnicas y procesos usados para crear imágenes del cuerpo humano o parte de él, esto con propósitos clínicos, es decir, procedimientos médicos que buscan revelar, diagnosticar, examinar enfermedades con propósitos científicos y médicos tales como el estudio de la anatomía física y metabólica. [1]

Dentro del entorno médico, la imagen médica se relaciona y es muy importante para muchas disciplinas médicas como la radiología, la endoscopia, la termografía médica, la fotografía médica y la microscopia. A continuación se detallan algunas de estas disciplinas médicas mencionando, a partir de ellas, que patologías que pueden detectarse para su posterior tratamiento.

1.1.1 Radiografía Convencional

La radiografía convencional refiere a la placa radiográfica donde se forma una imagen en un sustrato fotográfico. Esta imagen se forma mediante la exposición de una parte del cuerpo humano a una fuente de emisión de rayos X. Durante la formación de la imagen, el mineral de plata se precipita en una cantidad proporcional a la intensidad de la radiación incidente. Según la cantidad de mineral depositado varia la intensidad de la zona oscura en la placa, a mayor cantidad de mineral depositado se observa una zona más oscura o negruzca. Los gases, la grasa, los tejidos y huesos producen una zona negra, gris oscura, gris y blanco respectivamente, esto se muestra en la radiografía del pecho de la figura 1.1 que muestra un contraste entre el corazón (H), los pulmones (L) y donde puede apreciarse un tumor (T).

Según la parte del cuerpo puesta en observación para ser examinado y el tipo de análisis que se requiere, se aplican rayos X en distintas proporciones e intensidades. Los tejidos duros tales como el tejido óseo, requieren una fuente de energía elevada.

Para generar la radiación, se utiliza un ánodo de tungsteno alimentado con una alta diferencia de potencial (50 - 150 KVp) en una máquina trifásica de alta frecuencia. Los

tejidos más densos absorben los rayos X, de manera que la película fotográfica queda

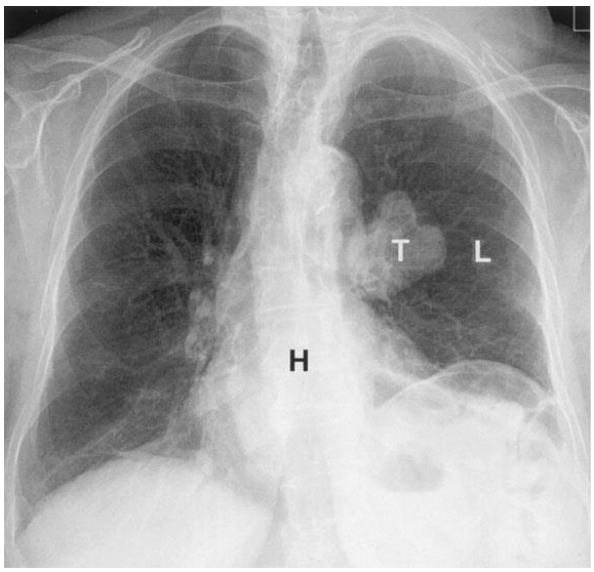

Figura 1.1 Imagen radiográfica

expuesta en la zona correspondiente a los mismos. Sin embargo, en las zonas correspondientes a tejidos blandos, que no absorben tal cantidad de rayos X, la película fotográfica en esta zona queda totalmente expuesta a la radiación y por tanto adquiere el color negro. Esta configuración suele utilizarse para diagnosticar fracturas óseas, objetos ingeridos y patologías como la osteoartritis, osteomielitis, cáncer de huesos, escoliosis, acondroplastia y problemas de crecimiento.

Los tejidos blandos utilizan la misma configuración descrita para tejidos duros, pero utilizando una menor cantidad de energía en el emisor de rayos X. Así puede obtenerse imágenes correspondientes a los pulmones, el corazón, los intestinos, los globos oculares, los tejidos blancos del cuello, así como los tejidos blandos del interior de los huesos para diagnosticar diferentes traumas ocultos.

En la actualidad los equipos médicos modernos que aplican rayos X generan las imágenes que se almacenan en formato digital. Este formato con respecto a las placas radiográficas analógicas posee muchas ventajas como la facilidad de transporte, almacenamiento, etc.

Existen dos métodos para obtener una imagen digital radiográfica: la imagen radiográfica que se obtiene mediante el escaneo o captura fotográfica de una placa radiográfica analógica y la imagen radiográfica digital que se obtiene mediante la captura digital directa de la imagen para convertir los rayos X directamente a señales electrónicas.

1.1.2 Radiografía Dental

La radiografía dental usa una dosis menos de radiación con alta penetración para visualizar los dientes, que son relativamente densos. Suelen utilizarse para tal fin

maquinas monofásicas de radiación. La figura 1.2 muestra un ejemplo de radiografía dental.

Figura 1.2 Imagen médica de radiografía dental

1.1.3 Mamografía

La mamografía es la radiografía proyectiva correspondiente a las mamas. Su uso fundamental es el diagnóstico del cáncer de mama, aunque puede tener otros usos, tales como localización de tejidos sospechosos antes de realizar una biopsia. La figura 1.3 muestra como ejemplo una mamografía.

Se utiliza normalmente un ánodo de molibdeno con una diferencia de potencial de 30KV, lo que proporciona un rango de energías que va desde los 15 hasta los 30KeV. [1]

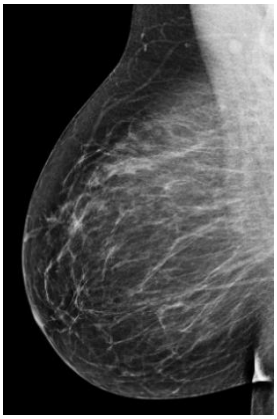

Figura 1.3 Mamografía digital

1.1.4 Tomografía Computarizada

Es una técnica topográfica axial, es decir, mediante esta técnica se generan imágenes que son perpendiculares al eje longitudinal del cuerpo. Los valores de atenuación en la imagen, generados por la tomografía computarizada, reflejan: la densidad y el numero atómico de los diferentes tejidos y se expresa generalmente como los coeficientes de atenuación relativos o Unidades Hounsfield (HU). Por definición, el HU del agua y del aire son 0 y -1000 respectivamente. El HU de la gama de tejidos varía de 10 a 50, con grasa que muestran HU negativos. El hueso es por lo menos 1000HU.

La resolución de contraste de las estructuras vasculares, de los órganos y patologías, pueden mejorarse con la infusión intravenosa de medios de contraste solubles en agua,

agentes solubles en agua o suspensiones de bario, se pueden administrar para una mejor visualización del intestino.

En esta prueba se producen artefactos que provocan distorsiones en la imagen obtenida, los artefactos, se producen por el movimiento de los pacientes o por una alta densidad de cuerpos extraños, como clips quirúrgicos. En la figura 1.4 se confirma un tumor mediante la prueba de tomografía donde el epigastrio muestra dos áreas de atenuación (M) confirmando una múltiple metástasis hepática de un tumor estromal gastrointestinal

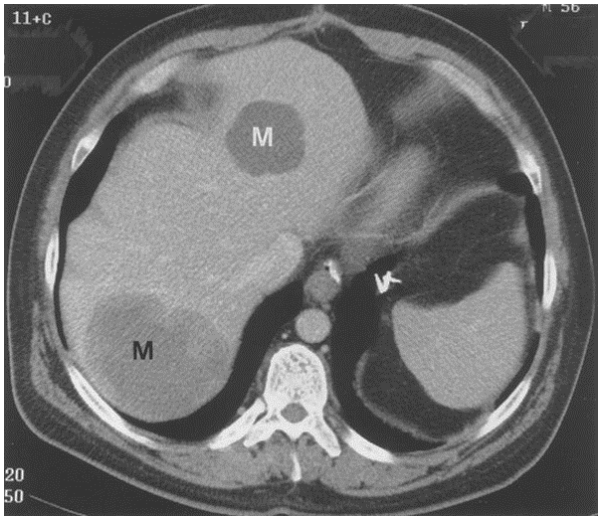

Figura 1.4 Imagen Tomográfica

En la actualidad se dispone de técnicas mejoradas para aplicaciones más específicas como la angiografía y la colonografía.

La angiografía CT, para obtener la distribución vascular, con una reducción de los artefactos producidos, incrementando la resolución. La angiografía se ha vuelto importante para el análisis de las arterias: pulmonar e iliaca y sus ramificaciones, la aorta torácica y la circulación carotidea intracraneal y extracraneal.

La colonografía CT, para obtener imágenes en 2-D o 3-D del colon. Esta tecnología es usada principalmente en la detección y la caracterización de pólipos del colon, estas imágenes muestran la superficie del colon y la densidad interna de las lesiones descubiertas, también la pared del intestino y estructuras fuera del colon, estructuras abdominales/pélvicas.

1.1.5 Ecografía

El diagnóstico por ultrasonido o ecografía es una técnica de obtención de imágenes no invasiva, la frecuencia de las ondas usadas son del orden de los 20 kHz. Un transductor es usado para emitir y recibir las ondas de sonido reflejadas en varios tejidos del cuerpo humano. El transductor es colocado sobre la piel del paciente con una capa fina de gel de acoplamiento. Este gel desplaza todo el aire que de otra manera refleja prácticamente

toda la onda de ultrasonido incidente. Como el sonido viaja en el paciente en forma de frentes de onda, en su trayecto disminuye la intensidad del haz, la atenuación del haz se produce en la absorción del tejido en forma de calor. En las interfaces de cada tejido el rayo es parcialmente reflejado y transmitido. Las ondas reflejadas viajan de vuelta al transductor y son convertidos en señales eléctricas.

La amplitud de la onda que vuelve depende del grado de absorción del haz. Un tono de gris es asignado a cada amplitud, los ecos débiles se asignan a un color negro casi al final del espectro.

La profundidad del tejido que produce la reflexión puede ser calculada, a partir del tiempo de viaje del haz y de la velocidad media del sonido en el tejido humano (1540 m/s)

Las limitaciones son principalmente dependientes del operador, limitaciones adicionales incluyen la visualización de órganos medianos abdominales (páncreas), la vasculatura cuando el intestino es cubierto por un gas, así como la incapacidad de las ondas sonoras para atravesar el gas o el hueso.

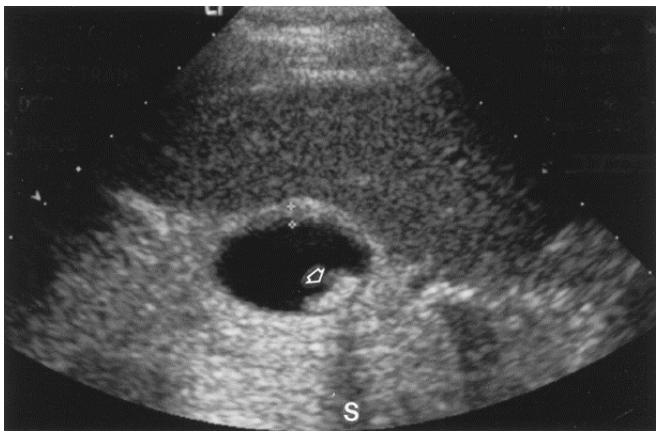

Figura 1.5 Imagen de ecografía

Hay muchas aplicaciones comunes de la ecografía, que incluyen imágenes del abdomen (el hígado, la vesícula, el páncreas, riñones), órganos femeninos reproductivos, el feto (estudios rutinarios fetales para la detección de anomalías), el sistema vascular (aneurismas, comunicaciones arteriales venosas, la trombosis profundamente venosa), testículos, pechos, el cerebro pediátrico (hemorragia, malformaciones congénitas), el pecho (el tamaño y la posición de colecciones pleurales fluidas) e intervenciones dirigidas por ultrasonido (2). La figura 1.5 muestra una imagen de ecografía de la vesícula que muestra un cálculo biliar (la flecha).

En general, las imágenes médicas pueden generarse usando diversos métodos, la ventaja es que no son invasivas. En la siguiente tabla se muestran las modalidades empleadas para obtener una imagen médica, su objeto de estudio, procedimientos, etc.

Tabla Nº 1.1 Características de adquisición de imágenes medicas

Modalidad	Objeto de Estudio	Energía Transmitida	Propiedad Medible	Energía Recibida	Método de Detección	Procesamiento	Visualización	Tamaño NxM (pixeles)	Niveles de Gris	Bits por Pixel
Radiología Convencional	Tejido duro y blando	Rayos X	Absorción	Rayos X	Placas	Química	Placas	Modo analógico		
Tomografía Computarizada	Tejido duro y blando	Rayos X	Absorción	Rayos X	Detectores	Transformada de Radón	Monitor y Placas	512 x 512	4096	**12**
Ultrasonido	Flujo sanguíneo	Sonido	Reflexión	Sonido	Cristal Piezoeléctrico	Barrido en espacio y tiempo	Video, monitor o impresión	512 x 512	256	**8**
Doppler	Tejido blando	Sonido	Dispersión	Sonido (variación frecuencia)	Cristal Piezoeléctrico	Barrido en espacio, tiempo y frecuencia	Video, monitor o impresión	512 x 512	256	**8**
Resonancia Magnética Nuclear	Tejido blando	RF	Respuesta a la señal de resonancia magnética	RF	Antenas	Transformada de Fourier	Monitor y placas	256 x 256	4096	**12**
Radiología Convencional Computarizada	Tejido duro y blando	Rayos X	Absorción	Rayos X	Detectores	Convertidor A/D	Monitor y placas	4000 x 2000	4096	**12**
Endoscopia	Órganos internos	Luz visible	Reflexión	Luz visible	Cámara	Digitalización	Monitor e impresión	512 x 512	2^24 colores	**24**
Microscopia	Muestras	Luz visible	Transmisión	Luz visible	Cámara	Digitalización	Monitor e impresión	512 x 512	2^24 colores	**24**
Medicina Nuclear	Funcionalidad orgánica		Numero de desintegraciones	γ, β	Escintilacion CCD Placas	Análisis de eventos	Monitor, impresión en placas y otros	128 x 128	65536	**16**

1.2 Calidad de las Imágenes Médicas Digitales

La calidad de una imagen es una medida de las diversas distorsiones que sufre la imagen durante su adquisición, procesamiento, compresión, almacenamiento, transmisión y reproducción.

En aplicaciones prácticas se presenta frecuentemente al usuario final como el que califica de forma visual, si una imagen es de buena calidad o no, un método de evaluación subjetiva. Para una evaluación objetiva se mide la calidad de la imagen médica mediante conceptos matemáticos, este método de evaluación objetiva requiere de una imagen original libre de distorsiones usado como imagen de referencia, con el cual las imágenes distorsionadas se comparan. En muchas aplicaciones prácticas, sin embargo, no se dispone de una imagen de referencia. [3]

Una evaluación subjetiva de la calidad, en la práctica es un proceso que requiere más tiempo, tiene como recursos a las características propias de una imagen como: la resolución espacial, resolución de densidad y el ruido de la imagen observable. [4]

Para una evaluación objetiva de la calidad de la imagen, el medida más conocida y ampliamente usada y porque su cálculo es sencillo, es el error cuadrático medio (MSE) junto con la relación señal a ruido pico (PSNR). [3]

Antes de iniciar con una descripción detallada de los conceptos mencionados. Para un mejor entendimiento, se hace una breve introducción al modelo matemático que describe a una imagen digital.

1.2.1 Representación Matemática de una Imagen Digital

Matemáticamente la imagen digital es una matriz bidimensional ordenada de valores.

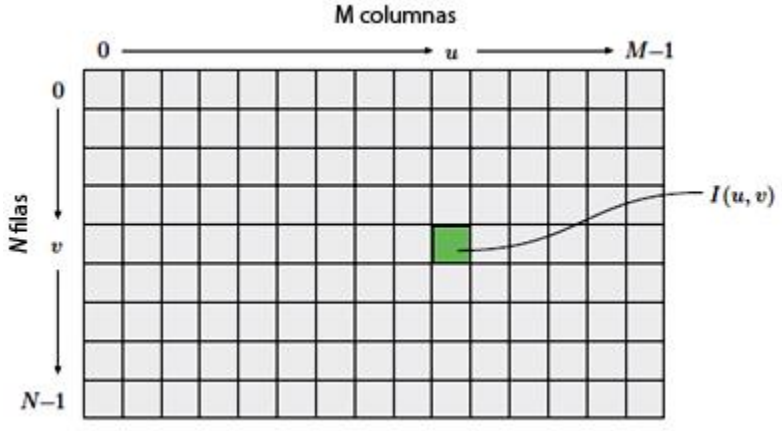

Figura 1.6 Imagen digital

Formalmente una imagen es una función I que asocia las coordenadas enteras NxM a un rango de valores (pixeles)

$$I(u,v) \in P\ ; \qquad u,v \in N \tag{1.1}$$

El sistema de coordenadas definido tal como se muestra en la figura 1.6, ubica al origen en la esquina superior izquierda, la coordenada v se incrementa desde arriba hacia abajo. La coordenada u se incrementa de izquierda a derecha. La numeración de filas y columnas inician a partir de cero.

La información en un pixel depende del tipo de dato usado para representarlo, estos valores se representan en binario, palabras de longitud k tal que un pixel se puede representar por uno de 2^k diferentes valores. El valor de k está definido como profundidad de bits de la imagen. El nivel de bit exacto de un pixel depende del tipo de la imagen, por ejemplo: binario, escala de grises o color RGB. [5]

1.2.2 Resolución Espacial, de Densidad y Relación Señal a Ruido (SNR)

La calidad de la imagen está caracterizada por tres parámetros: resolución espacial, resolución de densidad y relación señal a ruido (S/N). La resolución espacial es la cantidad de pixeles empleados para representar la realidad. La resolución de densidad es la cantidad total de niveles de gris disponible en una imagen digital. Una relación señal ruido S/N elevada indica una imagen agradable al ojo y por lo tanto con una buena calidad de imagen.

Para mostrar el impacto de la resolución espacial en la calidad de la imagen, de la figura 1.7 vemos una imagen obtenida en un ecógrafo con 256 niveles de gris y tamaño 200x200 pixeles (a). Las siguientes imágenes muestran los resultados de reducir la resolución espacial a: 100x100 pixeles (b) 50x50 pixeles (c) y 25x25 pixeles (d). En todos los casos se utilizó 256 niveles de gris.

Figura 1.7 Efectos de reducir la resolución espacial

Realizando una comparación visual, es fácil observar la pérdida progresiva de detalles, desde (a) hacia (d).

La figura 1.8 muestra los efectos producidos al reducir el número de bits empleados para representar los niveles de gris (profundidad de bits), es decir, reduciendo la resolución de densidad. La imagen (a) es una imagen de 100x100 y 8 bits de profundidad, el resto de las imágenes han sido obtenidas reduciendo la profundidad de bits, desde k=5 hasta k=1 manteniendo constante la resolución espacial.

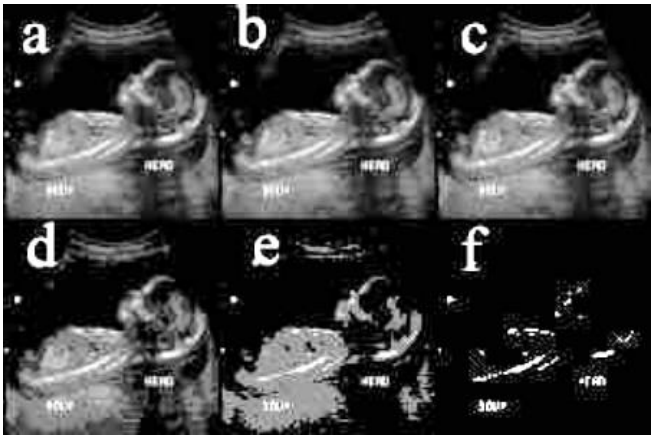

Figura 1.8 Efectos de reducir la densidad espacial

Comparando las imágenes visualmente, (b) y (c) son muy similares a la original (a), sin embargo conforme se reduce k, se hace más notorio un conjunto de estructuras ondulantes en las zonas de niveles de gris suaves. Este efecto es originado por el uso de un número insuficiente de niveles de gris en las áreas más suaves de la imagen, se denomina falso contorno. Por lo general más visibles en las imágenes que emplean un numero de niveles de gris inferior a 16.

En la figura 1.9 se muestra el efecto del ruido sobre una imagen, visualmente la imagen de la izquierda posee menos ruido y es la que ofrece una mejor calidad de la imagen.

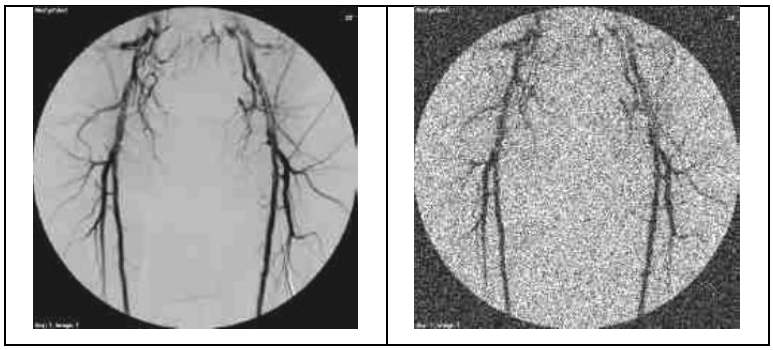

Figura 1.9 Izquierda imagen sin ruido, derecha imagen con ruido

La relación señal a ruido es un parámetro para medir el ruido en una imagen, es la razón de la energía de la señal entre la energía del ruido, por esta razón es conveniente expresarlo en la escala de decibeles.

$$SNR(dB) = 10log\left(\frac{Pseñal}{Pruido}\right) \tag{1.2}$$

Una relación señal a ruido (SNR) elevada implica que la imagen no es ruidosa. Solo en el caso de una imagen de tomografía, el ruido se distribuye uniformemente a través de la imagen.

1.2.3 Error Cuadrático Medio (MSE) y Relación Señal a Ruido Pico (PSNR)

El MSE calcula el cuadrado de la diferencia promedio de la imagen de referencia y la imagen distorsionada.

Entonces, para dos imágenes I, K de ancho M y alto N, de un canal y con una profundidad de bits k. Definimos al error cuadrático medio como:

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\|I(i,j) - K(i,j)\|^2 \tag{1.3}$$

Una variación del SNR y que sirve como medida cuantitativa de la calidad de la reconstrucción en el ámbito de la compresión de imágenes, es la Relación Señal a Ruido Pico (PSNR), su definición es:

$$PSNR(dB) = 10log\left(\frac{MAX_I^2}{MSE}\right) \tag{1.4}$$

$$MAX_I^2 = 2^k - 1 \tag{1.5}$$

Donde k es el número de bits que puede tomar como valor un pixel.

Para imágenes a color en 3 canales RGB, el MSE es la media aritmética del MSE en cada canal.

En el entorno médico, los tres parámetros analizados son modificados durante la adquisición para obtener una imagen médica de alta calidad. Debe mencionarse, que a mayor resolución espacial y/o de densidad, mayor será la capacidad de memoria necesaria y mayor el tiempo de procesamiento y transmisión de la imagen.

1.2.4 Función Dispersión de un Punto (Point Spread Function - PSF)

La función dispersión de un punto (PSF) describe, la respuesta del sistema de adquisición de imágenes ante una fuente puntual. Una fuente puntual idealmente debería representarse por un solo pixel como respuesta del sistema, pero será reproducido en más de un pixel vecino, en la imagen real.

Matemáticamente la imagen de salida $R(x,y)$ del sistema, se determina como la convolución de la imagen de entrada $g(u,v)$ con la PSF, es decir

$$R(x,y) = \int\int PSF(x-u, y-v)g(u,v)dudv + \eta(x,y) \tag{1.6}$$

Donde: $\eta(x,y)$ es la función que modela el ruido aditivo.

La forma del PSF en el sistema real no necesariamente es isotrópico es decir simétrico radialmente, por ejemplo para el caso de imágenes de ecografía, rayos X, tomografía de

radionúclidos el PSF es anisotrópico, en MRI se puede tener PSF isotrópico o anisotrópico dependiendo del rango de frecuencias que se considera. La figura 1.10 muestra el PSF isotrópico (B) y anisotrópico (C) ante una fuente puntual (A)

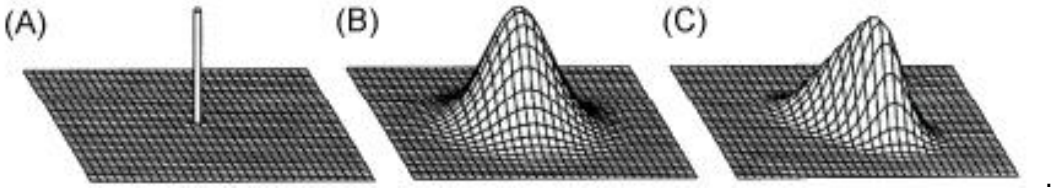

Figura 1.10 Modelos de PSF

La respuesta del sistema ante una fuente puntual (PSF), es el modelo matemático que explica la distorsión causada por el sistema de adquisición de imágenes. En general el PSF varía según el sistema, en el caso de imágenes médicas el PSF varia al pasar de una imagen de ecografía a otra de tomografía o de rayos X, etc.

El PSF nos da información vital para una mejora de la imagen minimizando las degradaciones. Está relacionado con la resolución espacial y de profundidad. [6]

PSF de una Imagen de Ecografía

En una imagen de ecografía, los parámetros que determinan su calidad son:

- Resolución de detalle. Es el parámetro que nos mide el espacio mínimo de objetos puntuales distinguibles y es determinado por el ancho del lóbulo principal del PSF
- Resolución de contraste. Capacidad del sistema para discriminar pequeños cambios de densidad.
- Resolución temporal. Capacidad del sistema para visualizar objetos en movimiento.
- Sensibilidad. Capacidad del sistema para visualizar objetos que débilmente reflejan la onda, este parámetro es cuantificado por la relación señal a ruido. (SNR)

El PSF determina la resolución de detalle y resolución de contraste. [7] Estas dos medidas son importantes para que el médico pueda detectar lesiones y tumores en el cuerpo. En la figura 1.11 se muestra el PSF de un sistema de ecografía.

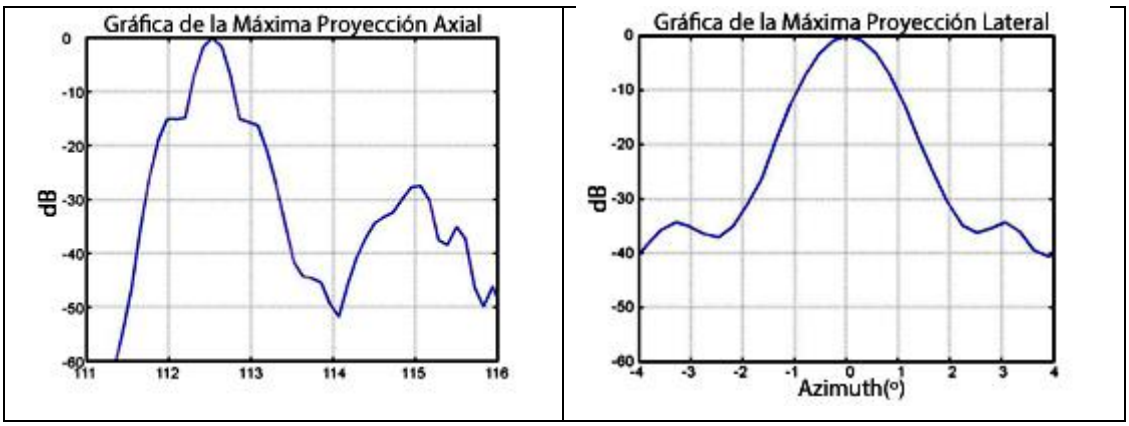

Figura 1.11 PSF de una ecografía, izquierda (PSF axial), derecha (PSF lateral)

Para la prueba con ultrasonido, el PSF se caracteriza por ser variable en el espacio, es decir en cada pixel de la imagen el PSF cambia. La figura 1.12 muestra el PSF para ángulos diferentes.

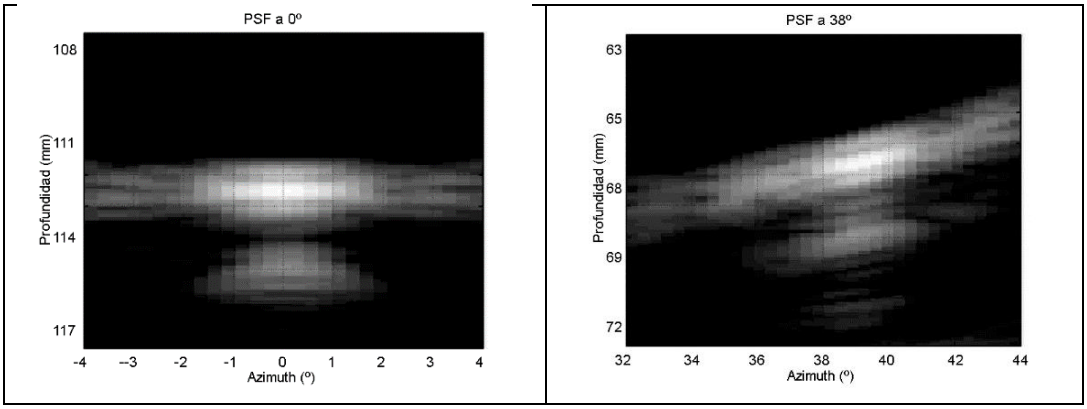

Figura 1.12 PSF de una prueba de ecografía

Mejorar la calidad de una imagen de ecografía conociendo el PSF se puede lograr al incrementar la resolución de detalle o la resolución de contraste. Para incrementar la resolución de detalle, el ancho del haz en el punto focal debe disminuirse, para aumentar la resolución de contraste la intensidad del ancho deben disminuir rápidamente fuera del punto focal.

Una opción para mejorar la imagen de ecografía se realiza controlando las propiedades del haz y así la calidad de las imágenes, para ello es pondera los elementos de la matriz PSF en la ecuación (1.6), sin embargo, las ponderaciones disminuyen el nivel del lóbulo lateral, ampliando el ancho del lóbulo principal y la resolución de detalle se pierde para mejorar la resolución de contraste. Adicionalmente como el PSF es variante en el espacio, en consecuencia la resolución en todos los puntos cambian la ponderación continuamente que es computacionalmente costoso, por otro lado las propiedades del PSF están relacionados a las características del medio a ser fotografiado y las ponderaciones establecidos no estiman el deseado PSF, entonces una elección serían las ponderaciones adaptativas, pero es computacionalmente costoso. Adicionalmente, el rendimiento de los métodos adaptativos es usualmente degradado con errores en el sistema (como errores en la localización de elementos de una matriz), con una incertidumbre en las características del medio (como velocidad del sonido) y ruido del transductor.

Otros métodos adaptativos para mejorar la imagen por ultrasonido es el procesamiento no lineal de los datos provenientes del eco. Estas técnicas intentan estimar la señal filtrando a través de los lóbulos laterales y entonces determinar la correcta amplitud de la señal principal. [8]

PSF de una Imagen de Tomografía

El PSF de una tomografía no es variante en el espacio, comparado con el PSF de ecografía, en la siguiente figura 1.13 se muestra un ejemplo de PSF típico en tomografía.

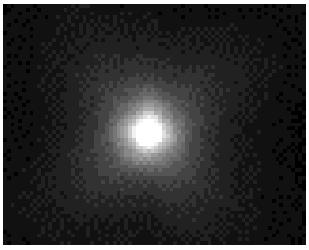

Figura 1.13 PSF de una tomografía

En pruebas como ultrasonido, rayos X y tomografía, es común obtener non-isotrópicos PSF (no son radialmente simétricos). En MRI es posible tener isotrópicos o no isotrópicos PSF dependiendo del sistema específico de adquisición.

1.3 Transformada Discreta de Fourier (TDF)

El uso de la transformada discreta de Fourier es de vital importancia para el tratamiento de las imágenes, permite el análisis para el dominio de las frecuencias.

1.3.1 La Transformada Discreta de Fourier y su Inversa

La Transformada Discreta de Fourier (DFT) de una señal digital $x(n)$ de longitud N es

$$X(\boldsymbol{k}) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) * exp\left(-j\frac{2\pi}{N}kn\right) \tag{1.7}$$

Donde: $k = 0, 1, \dots, N-1$

La correspondiente Transformada Discreta Inversa de Fourier (IDFT) es

$$x(n) = \sum_{k=0}^{N-1} X(k) * exp\left(j\frac{2\pi}{N}kn\right) \tag{1.8}$$

Considerando la fórmula de Euler: $e^{j\Theta} = \cos(\Theta) + jsen(\Theta); \quad con \quad j = \sqrt{-1}$

La ecuación (1.8) puede expresarse en su forma equivalente.

$$X(k) = \frac{1}{N}\sum_{k=0}^{N-1} x(n) * \left[cos\left(\frac{2\pi}{N}kn\right) - jsen\left(\frac{2\pi}{N}kn\right)\right] \tag{1.9}$$

Donde: $k = 0, 1, \dots, N-1$

En algunos casos, es conveniente expresarlo en coordenadas polares.

$$X(k) = |X(k)| * \exp(-j\phi k) \tag{1.10}$$

Donde la magnitud o espectro de Fourier es:

$$|X(k)| = |R^2(k) + I^2(k)|^{1/2} \tag{1.11}$$

El ángulo de fase espectro de fase de la transformada de Fourier es:

$$\phi(k) = tan^{-1}\left(\frac{I(k)}{R(k)}\right) \tag{1.12}$$

1.3.2 La 2D Transformada Discreta de Fourier y su Inversa

La extensión a 2 dimensiones de la transformada discreta de Fourier y su inversa, es sencillo.

La 2D transformada Discreta de Fourier de una función $f(x,y)$ de tamaño $M\,x\,N$ esta dado por la ecuación.

$$X(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) * exp\left(-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right) \tag{1.13}$$

Como en el caso unidimensional, esta expresión es calculada para: $u = 0, 1, \dots, M-1$ y $v = 0, 1, \dots, N-1$

Similarmente, dado $X(u,v)$, obtenemos $f(x,y)$ vía Transformada Inversa de Fourier, dado por la siguiente expresión:

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v) * exp\left(j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right) \tag{1.14}$$

Para: $u = 0, 1, \dots, M-1$ y $v = 0, 1, \dots, N-1$

La definición del espectro de Fourier, el ángulo de fase y el espectro de Energía son respectivamente:

$$|F(u,v)| = |R^2(u,v) + I^2(u,v)|^{1/2} \tag{1.15}$$

$$\phi(u,v) = tan^{-1}\left(\frac{I(u.v)}{R(ku,v}\right) \tag{1.16}$$

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v) \tag{1.17}$$

Donde $R(u,v)$ y $I(u,v)$ son las partes real e imaginaria de $F(u,v)$ respectivamente. (9)

1.3.3 Teorema de Muestreo y Fenómeno de Traslape (Aliasing)

Poder procesar digitalmente una señal continúa en el tiempo usando técnicas de procesamiento de señales, requiere convertir la señal analógica en una secuencia discreta. Una señal analógica, $x_a(t)$ muestreado cada intervalo de tiempo T produce una señal discreta en el tiempo $x(n)$ dado por:

$$x(n) = x_a(nT), \qquad -\infty < n < \infty \tag{1.18}$$

La frecuencia de muestreo $fs = 1/T$ debe ser tan grande como para no causar pérdidas de información espectral.

Teorema de Muestreo

Una señal continua en el tiempo, limitada con una alta frecuencia B (Hz) puede únicamente recuperarse a partir de la señal muestreada, si la tasa de muestreo satisface la siguiente condición

$$fs \geq 2B \tag{1.19}$$

El traslape ocurre cuando $fs < 2B$.

La figura 1.14 muestra el espectro de frecuencias $Xo(f)$ para una señal analógica y lo que le ocurre cuando se muestrea la señal y los espectros se traslapan.

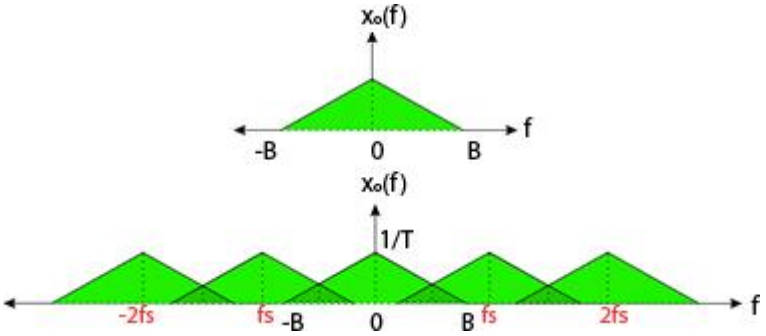

Figura 1.14 Fenómeno de traslape

El contenido de frecuencias en la zona traslapada no es propio de la señal original. [10]

1.4 Método de Mínimos Cuadrados

La solución $x \epsilon R^n$ a un problema de mínimos cuadrados satisface la siguiente ecuación:

$$\|Ax-b\|_2 = min \tag{1.20}$$

Si y solo si se cumple la siguiente condición.

$$A^T(b - Ax) = 0 \tag{1.21}$$

Donde$A\epsilon R^{mxn}$ y $\|\quad\|_2$ es la 2-norma definido por:

$$\|x\|_2 = \left(\sum_{i=1}^{n}|x_i|^2\right)^{1/2}, \qquad 1 \leq p \leq \infty \tag{1.22}$$

1.4.1 Descomposición en Valores Singulares

La descomposición en valores singulares (SVD) de una matriz $A\epsilon R^{mxn}$ es muy importante para el problema de mínimos cuadrados, da como resultado una forma diagonal de A bajo trasformaciones equivalentes ortogonales. En la actualidad es una herramienta valiosa para numerosas aplicaciones como: procesamiento de señales e imágenes, teoría de control, reconocimiento de patrones, etc.

A continuación enunciamos el teorema aplicado sobre elementos complejos (A^H denota la matriz formado por el conjugado de cada elemento y tomado su transpuesta).

Sea $A\epsilon C^{mxn}$ una matriz de rango r, entonces existe las matrices $U\epsilon C^{mxm}$ y $V\epsilon C^{nxn}$ tal que:

$$A = U\Sigma V^H, \quad \Sigma = \begin{pmatrix}\Sigma_1 & 0\\ 0 & 0\end{pmatrix} \tag{1.23}$$

Donde:

$$\Sigma\epsilon R^{mxn}, \qquad \Sigma_1 = diag(\sigma_1,\sigma_2,\dots,\sigma_r) \qquad y \qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$$

Los σ_i son llamados valores singulares de A y si denotamos

$$U = (u_1,\dots,u_m), \quad V = (v_1,\dots,v_n)$$

Los u_i y v_i son los vectores singulares izquierda y derecha asociados respectivamente con σ_i, $i = 1, \dots, r$

La SVD de A puede ser escrita

$$A = U_1 \Sigma_1 V_1^H = \sum_{i=1}^{r} \sigma_i u_i v_i^H \tag{1.24}$$

Dónde: $U_i = (u_1, \dots, u_r), \quad V_i = (v_1, \dots, v_r)$

La matriz A de rango r es descompuesto en una suma de r matrices de rango uno. Realizar el cálculo de la descomposición en valores singulares de A, requiere de métodos numéricos realizados por el computador.

Si la descomposición en valores singulares SVD de $A = U\Sigma V^H \epsilon R^{mxn}$ se tiene entonces, la norma que minimiza la solución de mínimos cuadrados $\|Ax-b\|_2$ esta dado por:

$$x = V \begin{pmatrix} diag(\sigma_1, \sigma_2 \dots, \sigma_r) & 0 \\ 0 & 0 \end{pmatrix} c \qquad c = U^T b \tag{1.25}$$

Donde

$$\text{Rango}(\,\text{A}\,) = \text{r} \leq \text{n}$$

1.5 Correlación Cruzada

La correlación cruzada entre dos señales de entrada es una medida: del grado de interdependencia entre dos procesos o la similitud entre dos señales $x_1[n]$ y $x_2[n]$. En otras palabras determinar la correlacion existente entre dos procesos o señales. [11]

La correlación cruzada puede calcularse para cualquier dimensión, para el caso de una dimensión, definimos la correlación cruzada normalizada entre dos señales como: (12)

$$\text{r}_\text{d} = \frac{\sum_\text{i}(\text{x[i]} - \breve{\text{x}})(\text{y[i} - \text{d]} - \breve{\text{y}})}{\sqrt{\sum_\text{i}(\text{x[i]} - \breve{\text{x}})^2}\sqrt{\sum_\text{i}(\text{y[i} - \text{d]} - \breve{\text{y}})^2}} \qquad \text{para} \quad 1 \leq \text{k} \leq \text{N} \tag{1.26}$$

El coeficiente $\text{r}_\text{d}\,\epsilon[-1{:}\,1]$, es una medida del tamaño y dirección, de la relación lineal entre las variables x e y. Si $\text{r}_\text{d} = +1$ significa un 100% de correlación, mientras que $\text{r}_\text{d} = -1$ significa un 100% de correlación en oposición de fase, $\text{r}_\text{d} = 0$ significa que no existe correlación, por lo tanto, las señales son independientes. Si $\text{r}_\text{d} < 0$ significa correlación negativa y si $\text{r}_\text{d} > 0$, correlación positiva.

El cálculo de la correlación cruzada de dos señales $x_1[n]$ y $x_2[n]$ puede acelerarse con el uso de la siguiente expresión

$$c_{12}[j] = \frac{1}{N}\text{F}_\text{D}^{-1}\{\text{X}_1^*[\text{k}].\text{X}_1\,[\text{k}]\} \tag{1.27}$$

Donde F_D^{-1} representa la IDFT. Esta técnica implica el cálculo de dos DFT y una IDFT, cada una de las cuales puede ser evaluada mediante el correspondiente algoritmo FFT. Cuando el número de elementos de la secuencia es suficientemente grande es mucho más rápido este cálculo que el obtenido directamente. [12]

1.6 Súper Resolución

La súper resolución (SR), se refiere a las técnicas que permiten construir imágenes en alta resolución (HR) a partir de imágenes observadas en baja resolución (LR), incrementando las componentes en alta frecuencia y removiendo las degradaciones causadas en un sistema de adquisición de imágenes (cámara digital, sistemas de adquisición de imágenes médicas o un múltiples imágenes de una secuencia de video). La idea básica es combinar información no redundante contenida en las imágenes de baja resolución.

Es necesario mencionar que una técnica de interpolación produce una imagen de mayor resolución, a partir de, una imagen de menor resolución, sin embargo, esta técnica actúa como filtro pasa bajo, causando pérdidas de frecuencia en consecuencia, sin adicionar nueva información. Esto no lo hace una técnica de súper resolución.

1.6.1 Aplicaciones

- Video vigilancia
- Imágenes Médicas (CT, MRI, Ecografía, etc.)
- Imágenes satelitales
- Conversión estándar de video, desde el formato NTSC a HDTV [13]
- Etc.

1.6.2 Problema de Reconstrucción de una Imagen en Súper Resolución

El modelo matemático que formula el problema de obtener una imagen en súper resolución, está fundamentado completamente con el proceso para obtener una imagen digital.

El sistema para obtener imágenes digitales no es perfecto, debido a las limitaciones en el hardware, las imágenes obtenidas sufren diversos tipos de degradación. Por ejemplo, el desenfoque óptico, el desenfoque por movimiento (muy común en videos), el tamaño del sensor causa también un desenfoque. Algunos de estos fenómenos se muestran en la figura 1.15.

Figura 1.15 Izquierda: desenfoque de movimiento. Derecha: desenfoque óptico

La limitada densidad del sensor conduce al fenómeno de traslape, limitando la resolución espacial.

Estas degradaciones que afectan a la imagen, algunos en mayor grado que otros, son modeladas total o parcialmente dependiendo de la técnica de súper resolución.

La siguiente figura esquematiza un modelo que relaciona la imagen en alta resolución con las imágenes en baja resolución mediante procesos concretos en las cuales se dan las degradaciones en una imagen. La entrada al sistema es una escena natural continua, muy aproximada a una señal en banda limitada. Estas señales están contaminadas por la atmosfera antes de llegar al sistema electrónico de adquisición de imágenes.

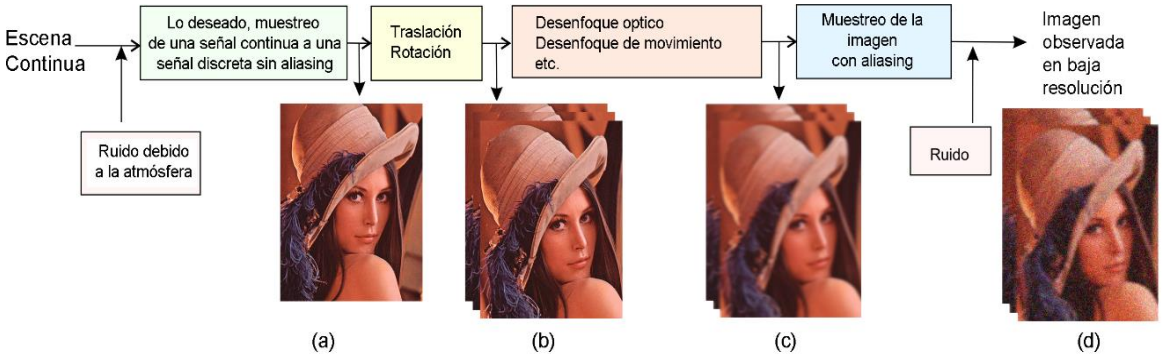

Figura 1.16 Proceso para obtener una Imagen Digital

Muestreando la señal continúa por encima de la frecuencia de Nyquist, genera la imagen digital en alta resolución (a) que es la imagen deseada. La entrada al sistema son múltiples imágenes o tomas de la misma escena. Relacionados entre ellas por desplazamientos relativamente pequeños, mostrado en (b). Siguiendo el proceso, estas imágenes sufren distintos efectos que provocan el desenfoque en la imagen (c). Estas imágenes distorsionadas (c) luego son muestreados por los sensores y convertidos en pixeles. Una sección de la imagen cae en el área del sensor, ocasionando una tasa de muestreo menor al de la imagen deseada. Posteriormente las imágenes serán afectadas por el ruido propio del sistema electrónico. Finalmente, las imágenes capturadas están desenfocadas, disminuidas (menor tasa de muestreo) y con ruido. [13]

Formular el modelo matemático que plantea el problema de reconstrucción de una imagen en súper resolución, acorde al proceso total mostrado en la figura 1.16, requiere modelar cada fenómeno que distorsiona la imagen, el ruido y el desplazamiento entre las imágenes en baja resolución.

Denotemos por $f(x,y)$ la imagen deseada en HR con una tasa de muestreo superior a la frecuencia de Nyquist. También denotemos por $\{g_K(m,n)\}_{k=1}^{N}$ a las N imágenes (con una tasa de muestreo menor al de la imagen deseada) capturadas por el sistema de adquisición de imágenes. Estas imágenes son representaciones distintas de $f(x,y)$.

El modelo matemático formulado por Iraní y Peleg [14] se muestra a continuación y describe el proceso mencionado para obtener la imagen,

$$g_K(m,n) = \sigma_k(\ \mathrm{h}(\ \mathrm{T}_k(f(x,y))\ )\ +\ \eta_k(\mathrm{x},\mathrm{y})\ ) \qquad \text{para} \quad 1 \leq \mathrm{k} \leq \mathrm{N} \qquad (1.28)$$

Dónde:

g_K, es la imagen que se obtiene al final del proceso de un sistema de adquisición de imágenes.

$f(x,y)$, es la imagen deseada en HR, lo cual al ser obtenerse por métodos matemáticos se le denominaría imagen en súper resolución.

T_k es la función geométrica de f a g_K determinado por los parámetros de desplazamiento en 2 dimensiones, que se obtienen de entre las imágenes de entrada g_K.

h, es la función que genera el desenfoque en la imagen, modelado por el PSF. Esto requiere un conocimiento del sistema de adquisición de imágenes.

η_k, es el ruido térmico.

σ_k, es la función que disminuye la tasa de muestreo (downsampling), en consecuencia disminuye el tamaño de la imagen.

Los valores de m y n representan las coordenadas de los pixeles en la imagen g_K, análogamente x e y representan las coordenadas de cada pixel en la imagen f en HR.

A menudo la función h suele representarse como una operación de convolución como se vio en la sección (1.2.3).

También puede formularse el modelo matemático empleando matrices.

$$g_K(m,n) = D_K H_K F_K f(x,y) + \eta_k(\mathrm{x},\mathrm{y}) \qquad \text{para} \quad 1 \leq \mathrm{k} \leq \mathrm{N} \qquad (1.29)$$

Donde F_K tiene la información determinada por los parámetros de desplazamiento, H_K modela los efectos del desenfoque. Adicionalmente D_K disminuye la tasa de muestreo.

Como se observa (1.6.3-2) es una ecuación lineal, entonces puede expresarse de la siguiente manera.

$$\begin{bmatrix} g_1(m,n) \\ g_2(m,n) \\ \vdots \\ g_N(m,n) \end{bmatrix} = \begin{bmatrix} D_1 H_1 F_1 \\ D_2 H_2 F_2 \\ \vdots \\ D_N H_N F_N \end{bmatrix} f(x,y) + \eta_k(\mathrm{x},\mathrm{y})\ ) \qquad \text{para} \quad 1 \leq \mathrm{k} \leq \mathrm{N} \qquad (1.30)$$

En sistemas reales las matrices $D_K H_K F_K$ son desconocidas y necesariamente deben ser estimados a partir de las imágenes capturadas y del sistema particular de adquisición de imágenes.

1.7 Computación en Paralelo

La computación en paralelo es el simultáneo uso de múltiples procesadores o computadoras con el fin de resolver un problema. [15]

Desde el 2003, debido a un incremento del consumo de energía a nivel mundial y al

excesivo calor disipado, el desarrollo en el aumento de la frecuencia del reloj se ha visto dificultado, limitando el nivel de actividad productiva que ahora busca disminuir el periodo del reloj en una CPU.

En la figura 1.17 se muestra a lo largo de los años, el nivel de desarrollo que han tenido en: los transistores, la frecuencia del reloj, las operaciones que realiza una CPU por ciclo de reloj y la energía disipada. Desde el año 2000, se observa un estancamiento en el desarrollo de la frecuencia del reloj al igual que el consumo de energía y de las operaciones por ciclo de reloj, todos se mantienen en un nivel dado excepto el desarrollo de los transistores, en otras palabras ahora caben una mayor cantidad de transistores en un área determinada.

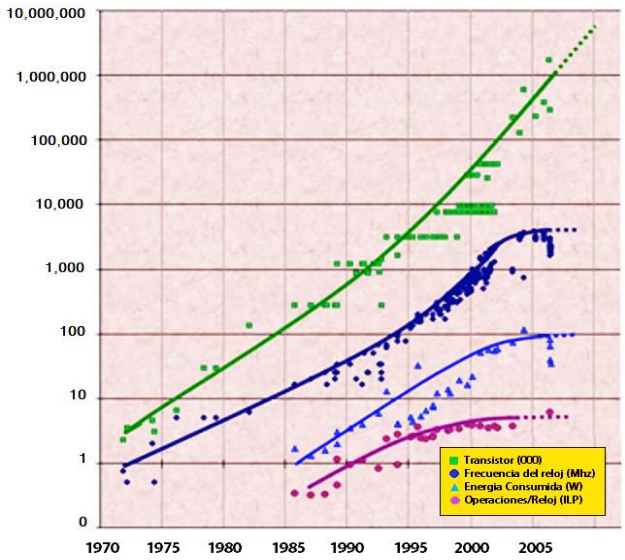

Figura 1.17 Fuentes Intel, Microsoft y Stanford

Con el fin de incrementar el poder de procesamiento sin aumentar la frecuencia del reloj los fabricantes de hardware se han orientado al modelo de usar múltiples procesadores para el procesamiento en paralelo. Estos cambios han producido un enorme impacto en la comunidad de desarrollo del software, permitiendo avances tecnológicos en múltiples áreas, explotando las capacidades de un rango de microprocesadores de múltiples núcleos: CPU, procesador digital de señales DSP, hardware reconfigurable FPGA, unidades de procesamiento gráfico (GPU). Ello ofrece un una gran variedad de aplicaciones, desde simulaciones por computadora con fines científicos y aplicaciones en la ingeniería hasta aplicaciones comerciales. [16]

1.7.1 Arquitectura de Computación en Paralelo

La clasificación muy usada hasta el día de hoy realizada por Flynn la cual se muestra en la siguiente figura, se basa en el número de instrucciones que se ejecuta y también en la cantidad de datos que se procesa.

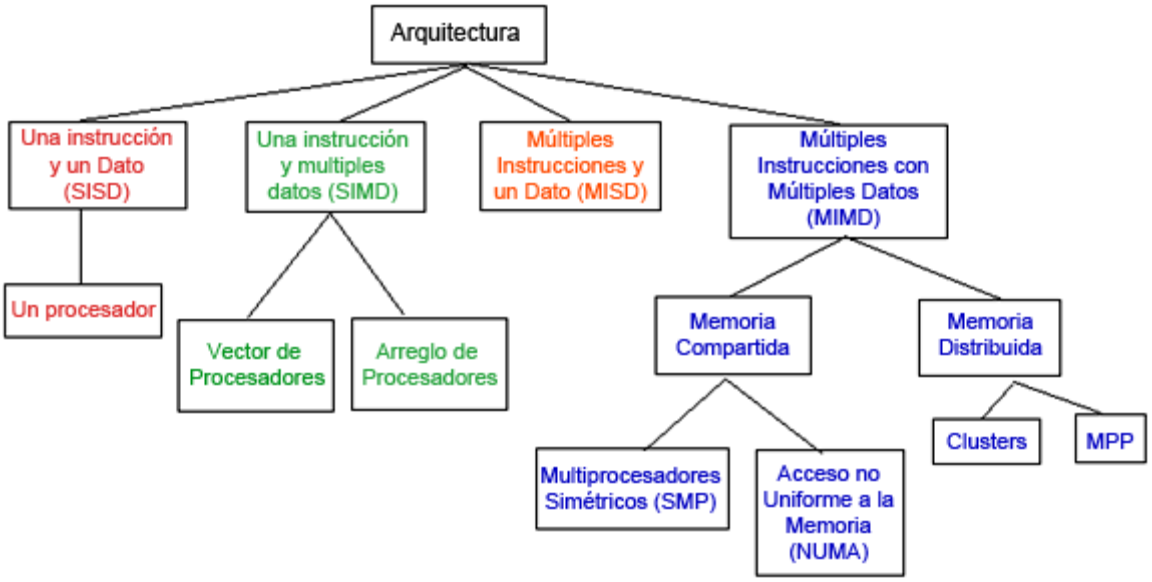

Figura 1.18 Clasificación de Flynn

Esta clasificación tiene 4 tipos de arquitecturas, las cuales se detallan a continuación.

Una Instrucción y Un Dato (Single Instruction, Single Data SISD). Un elemento de procesamiento (núcleo, procesador o un computador) que durante un ciclo del reloj puede ejecutar una instrucción, también durante un ciclo del reloj solo se puede leer un dato. El resultado se guarda de vuelta en solo un dato por cada ciclo del reloj. Este es la computación secuencial o serial y mostrada como ejemplo en la figura 1.19

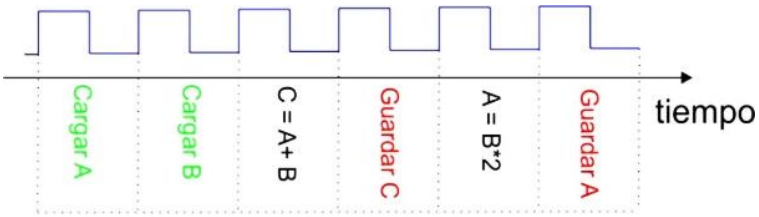

Figura 1.19 Ejemplo de una operación secuencial

En la figura 1.20 se muestra la Intel Pentium I, lanzado al mercado en 1993 cuya frecuencia de reloj era de 60 MHz.

Figura 1.20 Familia Intel Pentium I

Múltiples Instrucciones y Un Dato (Múltiple Instrictión, Single Data MISD). Estos múltiples elementos de procesamiento cada uno de los cuales tiene una memoria

privada, con un acceso común a una memoria global única. En cada paso, cada elemento de procesamiento obtiene el mismo dato único desde la memoria de datos y carga una instrucción desde su memoria privada, posiblemente instrucciones distintas que se ejecutan en paralelo. Este modelo de ejecución es muy restrictivo y no es comercial.

Una Instrucción y Múltiples Datos (Single Instruction, Múltiple Data SIMD). Hay múltiples elementos de procesamiento cada uno de los cuales tiene un acceso privado a una memoria de datos (compartida o distribuida). Pero hay solo un programa en memoria desde el cual un procesador especial actúa como controlador cuya función es buscar y enlazar instrucciones. Cada elemento de procesamiento obtiene desde el procesador especial la misma instrucción y se carga un dato diferente sobre el cual se procesa la instrucción. La instrucción se ejecuta en paralelo por todos los núcleos con diferentes datos. En la figura 1.21 se muestra un ejemplo.

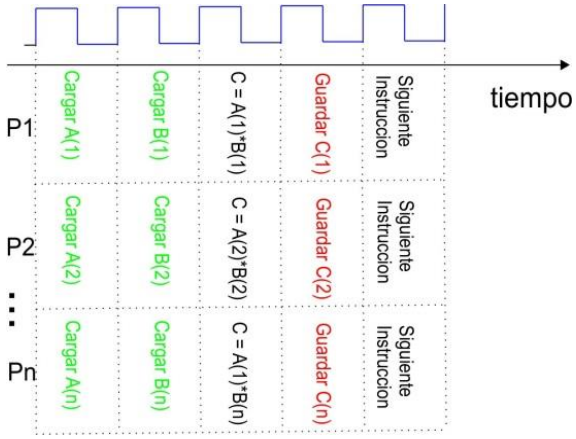

Figura 1.21 Suma en paralelo de múltiples elementos

En la figura 1.22 se muestra la arquitectura hardware GPU cuyo procesamiento se basa en la SIMD.

Figura 1.22 GPU Titán con 2688 núcleos CUDA

Múltiples Instrucciones y Múltiples Datos (Múltiple Instruction, Múltiple Data MIMD). Son múltiples elementos de procesamiento ejecutando una instrucción distinta y con acceso a una memoria de datos distintos. En cada ciclo del reloj, cada elemento de procesamiento carga una instrucción separada y elemento de datos separado, aplica la instrucción a los

datos y almacena el resultado de vuelta en el dato almacenado. Los elementos trabajan asincrónicamente con respecto a los demás tal y como se muestra en la figura 1.23.

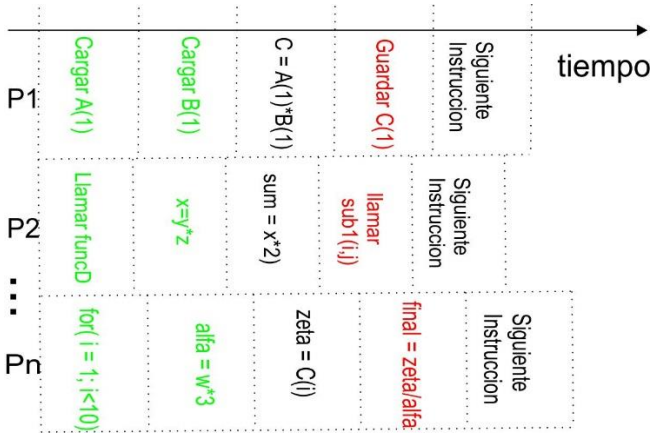

Figura 1.23 Ejecución asíncrona

Los multiprocesadores como la Pentium Core i7 y los multicomputadores como Sistemas Clúster son ejemplos de la arquitectura MIMD la figura 1.24 muestra algunos ejemplos.

Figura 1.24 Sistemas de Computo que usan MIMD

De los 4 sistemas mencionados, solo dos de ellas son importantes para la computación en paralelo: SIMD y MIMD. Los sistemas SIMD son considerados sencillos para programar, porque solo hay un programa y no se requiere de la ejecución sincrónica. Por el otro lado, los sistemas MIMD son más flexibles, cada elemento de procesamiento puede ejecutarse con su propio flujo de programación. [17]

Nuevos sistemas como el CM-5 intentan combinar los beneficios de ambos mundos, usando SIMD en programación mientras la maquina se ejecuta en modo MIMD.

1.7.2 Organización de Computadores con Memoria Compartida

Como se vio en la figura 1.20, de los 4 tipos de arquitectura es de interés para nuestro caso, la arquitectura SIMD que poseen las GPU y la arquitectura MIMD con memoria compartida empleado en el diseño de la Intel Pentium Core i7 o AMD Optetron.

Los computadores con una memoria física compartida se denominan también máquinas de memoria compartida (SMM). Consiste de una cantidad de procesadores o núcleos,

una memoria física compartida (memoria global) y una red de interconexiones que conectan el procesador con la memoria como se muestra en la siguiente figura 1.25.

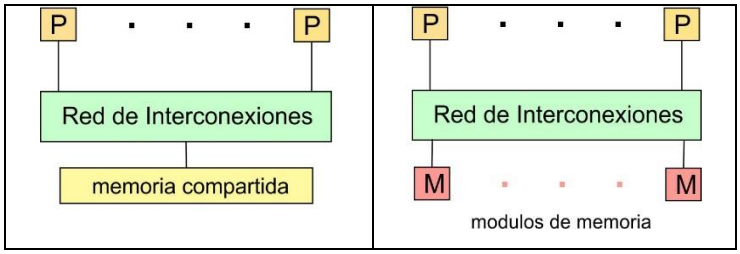

Figura 1.25 Estructura de las Maquinas de Memoria Compartida (SMMs)

Físicamente la memoria global esta implementada como un conjunto de módulos de memoria conteniendo un espacio de direcciones comunes que puede ser accedido por cada uno de los procesadores, esto hace posible que las variables almacenadas en la memoria global sean accedidas por todos los procesadores. La comunicación y cooperación entre los procesadores se organizan mediante la escritura y lectura de variables compartidas, sin embargo debe evitarse acceder a las variables compartidas concurrentemente por muchos procesadores.

Una variante de los SMM son lo multiprocesadores simétricos (SMP) que posee una memoria compartida, que permite un tiempo de acceso uniforme de todos los procesadores para todas las direcciones de memoria. Cada procesador además tiene una jerarquía de memoria cache privada cuyo acceso es mucho más rápido que la memoria global.

Los programas en paralelo para los SMM se basan en la ejecución de hilos (en inglés llamado thread). Un hilo (en este contexto) es un control de flujo particular que comparte datos con otros hilos, a través de un espacio de direcciones globales.

Los procesadores son controlados completamente por el sistema operativo, dando así la facilidad de ejecutar múltiples programas secuenciales, cada uno en diferentes procesadores.

El tiempo de acceso a la memoria es muy importante sobre todo en el rendimiento de un programa. Por ello se han formulado mejoras en la organización de la memoria obteniéndose (figura 1.26) mejoras significativas en el rendimiento del programa, incluso para el programa en paralelo.

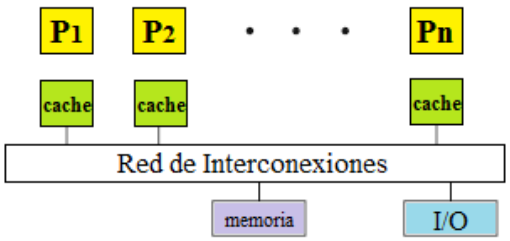

Figura 1.26 Multiprocesadores simétricos (SMP)

Para reducir el tiempo de espera para el acceso a memoria, se tiene dos enfoques: la simulación de procesadores virtuales para cada procesador físico (multithreading) y el uso de caches locales. El multithreading que hace uso de múltiples hilos, se basa en múltiples hilos intercalados (interleaved multithreading) o múltiples hilos en simultaneo (simultaneous multithreading SMT). La idea de múltiples hilos intercalados es de ocultar el tiempo de espera para acceder a memoria, el procesador físico contiene un contador de programa propio (PC) así como un conjunto propio de registros para cada procesador virtual. Después de ejecutar una instrucción (mientras ocurre la espera al acceso de memoria global) se cambia a otro procesador virtual ejecutando una instrucción, el número de procesadores virtuales por cada procesador físico es tal que el tiempo entre ejecuciones de instrucciones sucesivas en un procesador virtual es suficientemente largo como para cargar los datos requeridos desde la memoria global, ocultando los retardos al ejecutar instrucciones desde otros procesadores virtuales. Esta técnica no reduce el total de datos cargados desde la memoria global para ello se usan las memorias cache organizadas de tal forma que se acceda a los datos más frecuentes con mayor rapidez, algunas de estas formas se muestra en la figura 1.27.

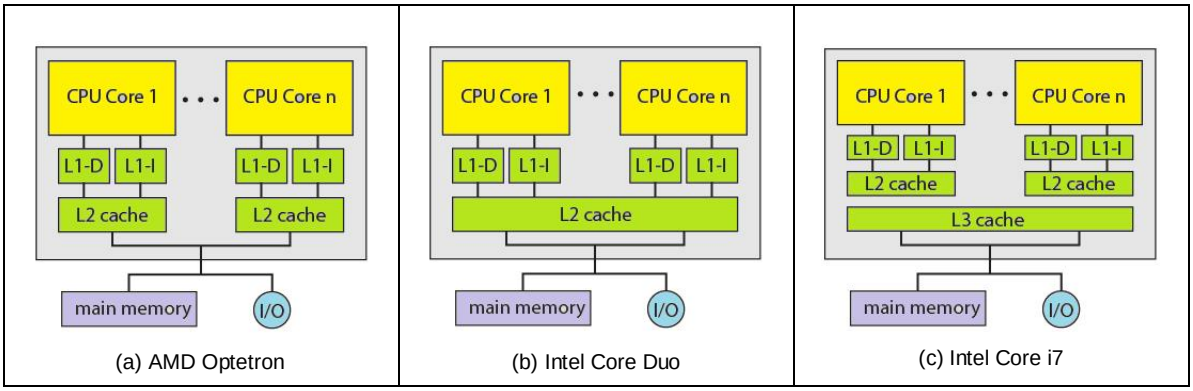

Figura 1.27 Formas de organización de la cache para C.I.

Los sistemas SMP pueden ser parte de una gran computadora en paralelo, es decir la opción de ejecutar muchos computadores con sistemas SMP, para la computación en paralelo, empleando una red de interconexiones que permiten el intercambio de datos entre distintos sistemas SMP. Para ello es necesario definir un conjunto de direcciones usando un protocolo adecuado. El protocolo asegura que debe asegurar que cualquier acceso a la memoria retorna el más reciente valor escrito en la dirección de memoria.

1.7.3 Paralelismo a Nivel de Hilos

Ocultar el tiempo de espera para acceder a la memoria, mejorar el tiempo de acceso según la frecuencia forman una arquitectura organizada para el procesador, queda mencionar que la programación en paralelo requiere el uso eficiente de los recursos disponibles, esto se denomina paralelismo a nivel de hilos (thread level parallelism). Los

chips con esta forma de organización usando hilos se denomina chip multiprocesador (chip multiprocessing CMP), en particular si los procesadores se ejecutan en forma independiente se denominan procesadores multinúcleo como la Pentium Dual Core o Pentium i7, los circuitos integrados de última generación fabricados por Intel o AMD.
La técnica de múltiples hilos en simultaneo (SMT) es una variación de múltiples hilos que usa el paralelismo a nivel de hilos, específicamente ejecuta simultáneamente instrucciones desde varios hilos.

1.7.4 Procesadores Multinúcleo

Existen dos principales razones por la cual la frecuencia del reloj no puede incrementarse significativamente. Primero el incremento del número de transistores en un chip, incrementa el consumo de energía y el aumento del calor producido debido a las corrientes de fuga, adicionalmente se requiere un incremento en la energía para su enfriamiento. Segundo, el tiempo de acceso a la memoria no puede ser reducido en la misma proporción como el periodo del reloj en el procesador, como consecuencia se incrementa el número de ciclos del reloj para acceder a la memoria.
Los procesadores multinúcleo tienen integrado en el chip múltiples núcleos. Para el sistema operativo, la ejecución de cada núcleo representa un procesador lógico con sus propios recursos, cuyo control se realiza en forma independiente. El sistema operativo puede asignar una aplicación a diferentes núcleos obteniendo ejecuciones en paralelo. Usando técnicas de programación en paralelo, es posible ejecutar una aplicación que realiza enormes cálculos (juegos, visión por computadora o simulaciones científicas), reduciendo el tiempo de ejecución comparado al tiempo que emplea un simple núcleo. [17]
Existen variados diseños de procesadores multinúcleo, dependiendo del número de núcleos, la estructura y el tamaño de la cache, el acceso de los núcleos a la cache y el uso de componentes heterogéneos.
Como ejemplo típico de procesadores multinúcleo tenemos: al Intel Xeon, AMD Optetron, las unidades de procesamiento grafico (GPU) como la NVIDIA GeForce 8800 o la AMD Radeon.

1.8 Unidades de Procesamiento Gráfico (GPU)

1.8.1 Arquitectura de una GPU Moderna

La arquitectura de una GPU posee múltiples núcleos, su propósito principal es de realizar operaciones gráficas, muy usado en juegos, pero también puede programarse para diversos propósitos que requieran inmensos procesamientos de datos, realizando cálculos matematicos. Posee la estructura SIMD dentro de la clasificación de Flynn.

El esquema general de una arquitectura GPU se muestra en la siguiente figura 1.28.

Figura 1.28 Arquitectura General de una GPU

En general, una GPU posee grandes archivos de registros para soportar el procesamiento simultáneo de múltiples hilos, una gran cantidad de ALUs (unidad lógico aritmético) con una pequeña memoria cache de núcleo y un bus de memoria optimizado para dar servicio a los ALUs.

En el mercado actual existen dos fabricantes de GPU: NVIDIA y AMD. La comprensión del hardware que se usa permite generar un eficiente código de programación disminuyendo el tiempo de procesamiento.

1.8.2 Arquitectura del Hardware GPU NVIDIA

La empresa NVIDIA posee diversos productos orientados principalmente para PC y Laptops. En los últimos años presenta línea de productos GeForce como tarjeta de video que hace uso de las características del GPU, hasta la fecha el mejor producto es la GeForce GTX TITAN.

Para el entendimiento de la arquitectura GPU de NVIDIA, como ejemplo se toma el modelo de la NVIDIA GeForce 8800 el cual se muestra en la figura 1.29. Posee 112 streaming processor (SP) también denominados núcleos CUDA, organizados en 14 streaming multiprocesor (SM), conteniendo: 8 núcleos SP, dos unidades funcionales (SFUx), cache de instrucciones y cache constante, una unidad de instrucciones multithreaded y memoria compartida.

Cada núcleo SP administra 96 hilos y sus estados en el hardware. Los procesadores se conectan con 4 particiones DRAM con un ancho de 64 bit, vía una red de interconexiones. La figura 1.29 muestra 7 clústeres de 2 SMs compartiendo una texture unit y una cache L1. El arreglo de procesadores se conecta con los procesadores en la operación de rasterización, cache de texturas L2, memorias DRAM externas y sistemas

de memoria, vía una red de interconexiones.

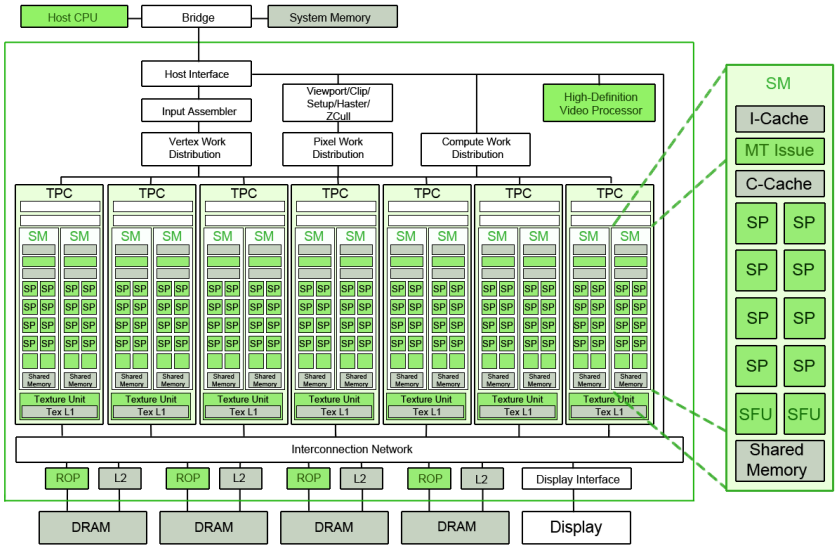

Figura 1.29 Arquitectura de una GPU NVIDIA GeForce 8800

Otro ejemplo aún más moderno, es la arquitectura GTX 480 mostrado en la figura 1.30 posee la arquitectura Fermi cuya mejora principal es la mayor capacidad de hilos que se ejecutan en paralelo y una mayor disposición de registros.

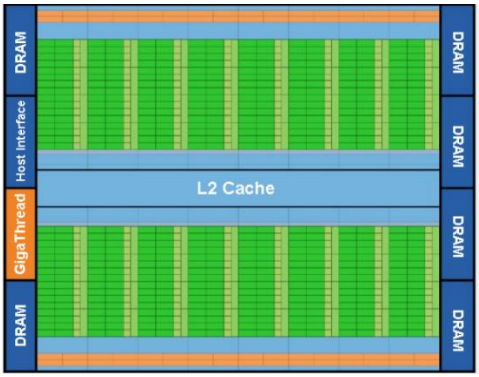

Figura 1.30 Arquitectura de una GPU NVIDIA GeForce GTX 480

Posee 15 streaming multiprocessors (SMs), cada SM posee 32 núcleos CUDA dando en total 480 núcleos CUDA. Cada núcleo CUDA consiste de un ALU y una unidad en punto flotante FPU.

1.8.3 Computación Heterogénea CPU y GPU

Una arquitectura de computación heterogénea se refiere a aquella que usa una GPU y una CPU, puede describirse principalmente, primero por el número de subsistemas y/o chips que usa la red de interconexiones y segundo por cuanta memoria disponen estos subsistemas.

La siguiente figura 1.31, se muestra dos configuraciones, una de Intel y otra de AMD muy usados actualmente, caracterizados por una GPU separada de la CPU con sus

respectivas memorias. En la figura de la izquierda, la CPU Intel se une al GPU vía una PCI-Express 2.0 con una capacidad pico de 16Gb/s de velocidad de transferencia (pico de 8Gb/s en cada dirección). Similarmente la figura de la derecha, con una AMD CPU la GPU está unido al chip también vía PCI-Express con el mismo ancho de banda.

Para ambos casos, la CPU y GPU pueden acceder a la memoria de los otros con menor ancho de banda disponible al acceso de memoria que está conectado directamente a ellos. Para la AMD el controlador de memoria está integrado en el CPU.

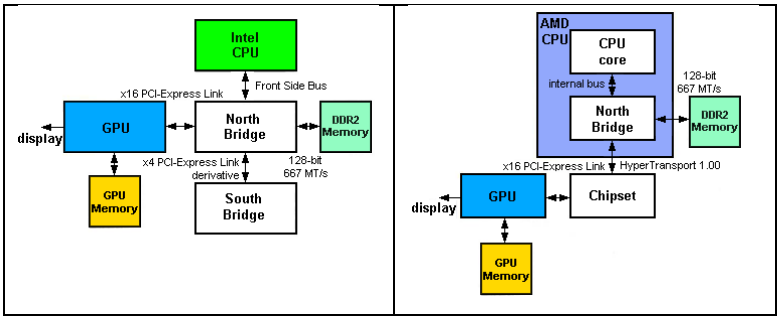

Figura 1.31 Arquitectura Heterogénea

Una variación del sistema heterogéneo mencionado hasta ahora, consiste en usar múltiples GPUs, comúnmente de 2 a 4 trabajando en paralelo. Un ejemplo es el sistema NVIDIA SLI con múltiples GPUs, diseñado para altísimos rendimientos, aquí su potencial de uso para la investigación, ejemplo simulación de modelos matemáticos que consumen un alto poder de procesamiento como la simulación de fenómenos cósmicos, modelos con elementos finitos.

1.9 Principios Para el Diseño de Algoritmos que se Ejecutan en Paralelo

En general conocer desde el punto de vista hardware, el número de núcleos o procesadores de un sistema permite establecer el número óptimo de hilos que se ejecutaran en paralelo, este número se especifica mediante un código de programación. Conocer arquitectura de la memoria (compartida o distribuida) en un hardware especifico, permite usar de forma eficiente la memoria de cache en sus diversos niveles y la memoria global, esto nos da un criterio en el código del programa para que su ejecución y procesamiento sea lo más rápido posible, como se sabe usar la memoria global puede almacenar gran cantidad de data pero la transmisión de datos es muy lento en comparación con la memoria cache, la memoria cache es privada para una GPU y la global puede ser accedido desde cualquier hilo en ejecución.

Existen más criterios útiles para hacer un programa eficiente que depende en gran medida de la arquitectura hardware para su ejecución en paralelo, estos se mencionaran a continuación mediante la inclusión de conceptos muy importantes relacionados a la computación en paralelo desde el punto de vista del programador.

1.9.1 Tipos de Paralelismo

Tenemos 2 diferentes tipos de paralelismo, las cuales son:

- Paralelismo de Tareas. La ejecución de tareas en un problema al mismo tiempo.
- Paralelismo de Datos. La ejecución de partes de una misma tarea (datos diferentes) al mismo tiempo.

Para el entendimiento de estos conceptos de forma sencilla, se hace una analogía con la recolección de manzanas de un árbol.

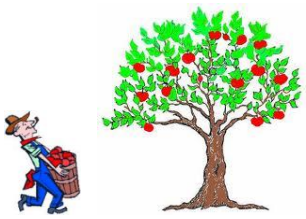

Figura 1.32 Proceso secuencial, una persona recogiendo manzanas una a la vez

El recolector de manzanas representa al hardware (unidad de procesamiento), el árbol representa la tarea a ejecutar y las manzanas representan los datos a operar.

Si el recolector recibe la ayuda de tres granjeros más, tenemos muchos trabajadores recolectando manzanas en un mismo árbol, esto representa el paralelismo de datos.

Figura 1.33 Paralelismo de datos, varias personas recolectando manzanas en un árbol

Otra alternativa para recolectar las manzanas es tener cada trabajador en cada árbol, esto representa el paralelismo de tareas, cada tarea se ejecuta al mismo tiempo.

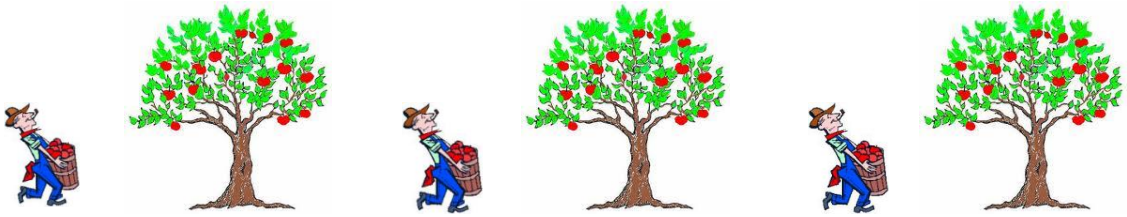

Figura 1.34 Paralelismo de tareas, varias personas recolectando, uno en cada árbol

Con estos dos conceptos se puede disminuir el tiempo de procesamiento. Para paralelizar un programa secuencial debemos pensar en descomponer el problema en sub tareas individuales, analizar cada sub tarea y concluir si es el caso de un paralelismo de tareas o paralelismo de datos.

Si es un paralelismo de tareas, puede aplicarse el procesamiento en paralelo desde con

el procesador principal de una CPU, es decir con una Intel o AMD que posee múltiples procesadores (ejemplo la Intel i7 posee 8 procesadores), hay muchas librerías que se puede emplear como OPENMP.

Si es un paralelismo de datos, puede aplicarse el procesamiento en paralelo desde la GPU, hay muchas librerías para tal fin como el estándar OpenCL o CUDA de NVIDIA.

Si se requiere una combinación de ambas es necesario tener una arquitectura heterogénea.

1.9.2 Concurrencia

La concurrencia es un concepto que está relacionado a dos o más sucesos que ocurren en el mismo tiempo. Encontramos concurrencia en el mundo real todo el tiempo, por ejemplo llevar a un niño en una mano y al mismo tiempo cruzamos la autopista. En términos de computación, concurrencia significa un solo sistema que procesa múltiples tareas por separado, sin embargo, estas tareas pueden o no ejecutarse al mismo tiempo lo cual no es un requerimiento. Por ejemplo, en un programa de dibujo que recibe entradas vía mouse y teclado o actualizando la pantalla con una imagen, conceptualmente recibir y procesar entradas son operaciones distintas para actualizar la pantalla. Estas tareas son concurrentes, pero no necesariamente se calculan en paralelo, es decir, para el caso en el cual lo ejecutamos en un núcleo de CPU, en este caso el sistema debe de intercambiar entre las tareas para procesarlos.

Los programas en paralelo pueden ser concurrentes pero los programas concurrentes no necesariamente están en paralelo, esquemáticamente se muestra en la figura 1.35.

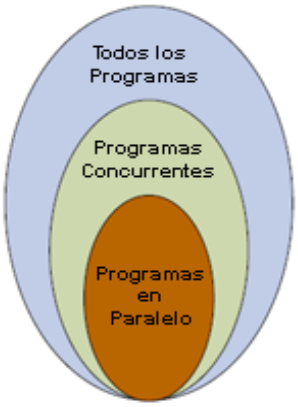

Figura 1.35 Los programas en paralelo son un subconjunto de los programas concurrentes

1.9.3 Granularidad

En la computación en paralelo, la granularidad es una medida de la razón de cálculo a la comunicación. Los periodos de comunicación están separados de los periodos de cálculo mediante eventos sincronizados, un procesador primero tendría que transferir los datos (periodo de comunicación) y luego realizar el procesamiento sobre ellos (periodo

de cálculo). Es importante seleccionar la más adecuada granularidad, bajo una específica plataforma de hardware, con el fin de obtener mayores beneficios, algunas veces, esta selección depende de la cantidad de datos asignados a una tarea específica.
La granularidad se clasifica de la siguiente manera.
Paralelismo de grano fino

- Intensidad de la aritmética baja.
- Si la granularidad es demasiado fino, es posible que el tiempo necesario para la comunicación y sincronización entre las tareas produzca una implementación más lenta en paralelo respecto de la ejecución en serie original.
- Puede no tener suficiente procesamiento como para ocultar la larga duración de la comunicación.
- Facilita el balance de carga, proporcionando un mayor número manejable (es decir, más pequeño) de unidades de trabajo.

Paralelismo de grano grueso

- Intensidad de la aritmética alta.
- Aplicaciones completas pueden servir como el grano de paralelismo.
- Es más difícil de equilibrar la carga de manera eficiente.

La selección de una granularidad más eficiente, depende del algoritmo y del hardware a usar.

1.9.4 Sincronización y Datos Compartidos

Considere el caso en el cual dos aplicaciones se ejecutan y no compartan dato alguno. Estos pueden ejecutarse concurrentemente o en paralelo, pero si la ejecución de una aplicación genera un resultado que pueda ser requerido por la segunda aplicación, tendríamos que introducir alguna forma de sincronización en el sistema y ejecución en paralelo, así podemos paralelizar porciones de una aplicación que no tienen dependencia de datos, pero esto no siempre será posible. (18)

1.9.5 Procesos e Hilos

Los modelos de programación en paralelo están basados frecuentemente en procesos o hilos. Ambos son abstracciones de un flujo de control. La principal idea es descomponer el proceso de una aplicación en tareas y emplear múltiples controles de flujo en diferentes procesadores o núcleos para su ejecución, entonces obtenemos pequeñas ejecuciones en menor tiempo.
En general un proceso está definido como un programa en ejecución. El proceso comprende al programa ejecutable junto con toda la información que es necesaria para la ejecución del programa, esto incluye datos del programa, los valores actuales del

registro, como el contenido del programa que especifica la siguiente instrucción a ser ejecutado. Cada proceso tiene su propio espacio de direcciones.

El modelo de los hilos es una extensión del modelo de proceso, cada proceso consiste de múltiples controles de flujo independientes llamados hilos. La palabra wordthread es usado para indicar una secuencia larga de instrucciones a ejecutar.

Una característica significativa del hilo, es que comparten el espacio de direcciones del proceso. Cuando un hilo almacena un valor en el espacio de direcciones, otro hilo del mismo proceso puede acceder a este valor después. Los hilos son comúnmente usados cuando los recursos de ejecución tienen acceso a una memoria física compartida, como es el caso de un procesador de múltiples núcleos. En este caso el intercambio de información es muy rápido. El uso de hilos es más flexible que el uso de un proceso, dando las mismas ventajas concernientes a la ejecución en paralelo. En particular los diferentes hilos de un proceso pueden asignarse a diferentes núcleos de un procesador multinúcleo. [17]

1.9.6 Ley de Amdahl

Teóricamente si uno duplica el número de procesadores que usa, el tiempo de ejecución se reduce a la mitad, si se dobla el número de procesadores sucesivamente, entonces otra vez se acorta a la mitad el tiempo de ejecución. Podría pensarse que la reducción del procesamiento continúa sin fin conforme aumentamos el número de procesadores, sin embargo existe un límite en la reducción del tiempo, debido al hecho de que todo programa consta de una sección que se puede paralelizar y otra sección que se procesara en forma serial.

La Ley de Amdahl establece la relación entra la aceleración esperada de la implementación paralela de un algoritmo y la implementación serial del mismo algoritmo. Si se considera el número de procesadores (N) como el factor de mejora introducido, la aceleración S sera:

$$S = \frac{F}{Fa} = \frac{1}{(1-P) + \frac{P}{N}} \tag{1.31}$$

Donde:

F = tiempo de ejecución mejorado

Fa = tiempo de ejecución antiguo

P = Porción del cálculo mejorado

Conforme N se vuelve más grande, en el límite se tiene

$$\frac{F}{Fa} = \lim_{N\to\infty} \frac{1}{(1-P) + \frac{P}{N}} = \frac{1}{1-P} \tag{1.32}$$

La siguiente figura.1.36 muestra que la aceleración de un programa en paralelo está limitada por la porción serial que no puede ser paralelizada.

Como ejemplo, si el 95% de un programa puede paralelizarse la máxima aceleración obtenida es de 20x.

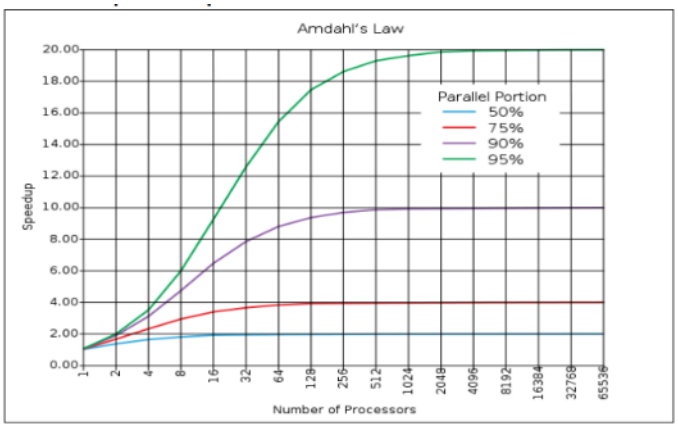

Figura 1.36 Aceleración del procesamiento versus número de procesadores

CAPÍTULO II
DESARROLLO DEL ALGORITMO

Generar una imagen en súper resolución requiere del conocimiento de cada uno de los fenómenos que degradan la imagen, originados en un sistema electrónico particular donde se genera la imagen, esto se vio en la sección 1.6.

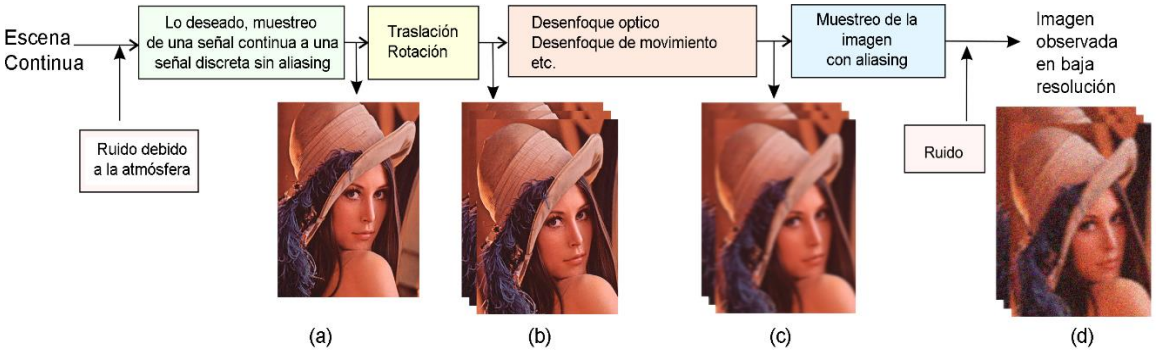

Figura 2.1 Proceso para obtener una imagen digital

Es necesario establecer modelos matemáticos que expliquen los fenómenos, lo más preciso posible, en consecuencia, para generar la imagen en súper resolución es necesario:

- Modelar el ruido.
- Aumentar y disminuir la frecuencia de muestreo mediante un método de interpolación.
- Modelar el PSF del sistema de adquisición de imágenes, característico de cada sistema de adquisición de imágenes.
- Generar una función que desplace y rote una imagen.

Se puede remover el ruido aplicando métodos de procesamiento de imágenes, esto implica usar filtros sobre la imagen. El tipo de filtro depende de la naturaleza del ruido. Para imágenes médicas, el hardware incluye de un método para eliminar el ruido. Es decir, en general la imagen médica obtenida ya fue procesada mediante algún filtro.

Aumentar la frecuencia de muestro en una imagen implica obtener una imagen ampliada con más datos, en forma análoga, disminuir la frecuencia de muestreo da como resultado una imagen más pequeña, sin embargo, el hecho de generar una imagen ampliada o reducida afecta a la imagen de entrada al comportarse como un filtro pasa bajo, el grado

de distorsión depende del método de interpolación que se emplea.

Determinar el PSF del sistema de adquisición de imágenes, requiere un estudio del sistema electrónico. Para el entorno médico, el PSF varía según el tipo de examen médico. Como un aporte adicional y la aplicación específica en las imágenes médicas, se hace un estudio del PSF para pruebas de ultrasonido.

Para determinar en qué medida la función que desplaza o rota una imagen, es necesario determinar los parámetros de movimiento (rotación y desplazamiento) entre las imágenes de entrada con respecto a uno de referencia, esto se hace, haciendo una comparación de cada una de las imágenes de entrada con respecto a la imagen de referencia. Existen diversos métodos para poder determinar los parámetros de desplazamiento y rotación. El algoritmo seleccionado está basado en el análisis del dominio de las frecuencias, cuyos resultados son muy precisos y formulados por Vanderwalle, Susstrunk y Vetterlli. [19]

En base a los modelos matemáticos que explican los fenómenos que degradan la imagen, se aplica un algoritmo para construir la imagen en súper resolución. De los diversos métodos existentes se selecciona el denominado Iterative Backprojection (IBP) basado en iteraciones y formulados por Irani y Peleg. [20]

Las siguientes secciones se hace un análisis más detallado del algoritmo general por partes, divididos en: estimar los modelos que describen los fenómenos que degradan a las imágenes, estimar los parámetros de desplazamiento y rotación así como el IBP.

Finalmente se aplica la tecnología del procesamiento en paralelo que permitirá reducir el tiempo que tarda en procesar los datos para obtener la imagen en súper resolución.

2.1 Estimación de los Parámetros de Desplazamiento

2.1.1 Estimación del Desplazamiento entre dos Imágenes

Estimar el desplazamiento en el dominio de las frecuencias se basa en la propiedad de la transformada de Fourier, en la cual un desplazamiento en el espacio de la señal genera un cambio de fase de su transformada manteniéndose constante su magnitud.

Consideremos 2 señales analógicas.

$$f_1 : R^2 \rightarrow R \tag{2.1}$$

$$f_2 : R^2 \rightarrow R \tag{2.2}$$

Tal que la señal f_2 es una versión desplazada de f_1 en $\Delta d = (\Delta x, \Delta y)$, es decir.

$$f_1(\Delta x + x, \Delta y + y) = f_2(x, y) \tag{2.3}$$

Entonces la 2-D transformada de Fourier de cada señal es:

$$F_1(u, v) = \iint_{-\infty}^{\infty} f_1(x, y) * \exp\big(-j2\pi(ux + vy)\big) \tag{2.4}$$

$$F_2(u,v) = \iint_{-\infty}^{\infty} f_2(x,y) * \exp\big(-j2\pi(ux+vy)\big) \tag{2.5}$$

Reemplazando el valor de f_2, de la ecuación (2.3) en (2.5), se tiene:

$$F_2(u,v) = \iint_{-\infty}^{\infty} f_1(\Delta x + x, \Delta y + y) * \exp\big(-j2\pi(ux+vy)\big) \tag{2.6}$$

Haciendo cambios de variable: $x' = x + \Delta x\,;\quad y' = y + \Delta y$

$$F_2(u,v) = \iint_{-\infty}^{\infty} f_1(x',y') * \exp\big(-j2\pi(ux' - u\Delta x + vy' - v\Delta y)\big) \tag{2.7}$$

Reordenando los términos dentro de la función exponencial

$$F_2(u,v) = \iint_{-\infty}^{\infty} f_1(x',y') * \exp\big(j2\pi(u\Delta x + v\Delta y) - j2\pi(ux' + vy')\big) \tag{2.8}$$

Simplificando

$$F_2(u,v) = \exp\big(j2\pi(u\Delta x + v\Delta y)\big) * \iint_{-\infty}^{\infty} f_1(x',y') * \exp\big(-j2\pi(ux' + vy')\big) \tag{2.9}$$

Reemplazando la ecuación (2.4) en (2.9), se tiene:

$$F_2(u,v) = \exp\big(j2\pi(u\Delta x + v\Delta y)\big) * F_1(u,v) \tag{2.10}$$

Este resultado general demuestra la propiedad de traslación en el tiempo que en el dominio de frecuencia ocasiona un cambio de fase.

En el caso particular en el que f_1 y f_2 son dos imágenes con M de ancho y N de alto, se aproxima el cálculo de F2 y F1 mediante la 2-D transformada discreta de Fourier definidos por:

$$F_1\left(\frac{2\pi}{M}u, \frac{2\pi}{N}v\right) = \sum_{x\prime=0}^{M-1}\sum_{y\prime=0}^{N-1} f_{1p}(x',y') * \exp\left(-j2\pi\left(\frac{ux'}{M} + \frac{vy'}{N}\right)\right) \tag{2.11}$$

$$F_2\left(\frac{2\pi}{M}u, \frac{2\pi}{N}v\right) = \sum_{x\prime=0}^{M-1}\sum_{y\prime=0}^{N-1} f_{2p}(x',y') * \exp\left(-j2\pi\left(\frac{ux'}{M} + \frac{vy'}{N}\right)\right) \tag{2.12}$$

Dónde: u= $0, 1, \ldots, M-1$ y $v = 0, 1, \ldots, N-1$

La tasa de muestreo en la horizontal es $\delta w_x = 2\pi/M$ y en la vertical $\delta w_y = 2\pi/N$

De los resultados en (2.11) y (2.12), basado en el resultado obtenido en (2.10), sin pérdida de generalidad, se tiene:

$$F_2\left(\frac{2\pi}{M}u, \frac{2\pi}{N}v\right) = \exp\left(j2\pi\left(\frac{u\Delta x}{M} + \frac{v\Delta y}{N}\right)\right) * F_1\left(\frac{2\pi}{M}u, \frac{2\pi}{N}v\right) \tag{2.13}$$

La ecuación (2.12) muestra que la diferencia de fases de las TDF está dado por la siguiente ecuación

$$\frac{\angle F_1(u,v)}{\angle F_2(u,v)} = -j2\pi\left(\frac{u\Delta x}{M} + \frac{v\Delta y}{N}\right) \tag{2.14}$$

También

$$|F_1(u,v)| = |F_2(u,v)| \tag{2.15}$$

Para: $u = 0, 1, \ldots, M-1$; $v = 0, 1, \ldots, N-1$

De (2.15) el resultado demuestra que los desplazamientos Δx e Δy pueden determinarse a partir de la diferencia de fases en cada elemento de F_1 y F_2. Para ello se resuelve un sistema de ecuaciones lineales con dos incógnitas.

Generalmente las imágenes presentan el fenómeno de traslape en el dominio de las frecuencias, con una tasa de muestreo menor a dos veces la frecuencia más alta contenida en la imagen. Este fenómeno distorsiona los resultados, por ello es necesario aplicar un filtro pasa bajo.

En general, el contenido de frecuencias de una imagen digital está dentro de la región mostrada en la figura 2.1 (a), la figura (b) muestra traslape en su espectro de frecuencias.

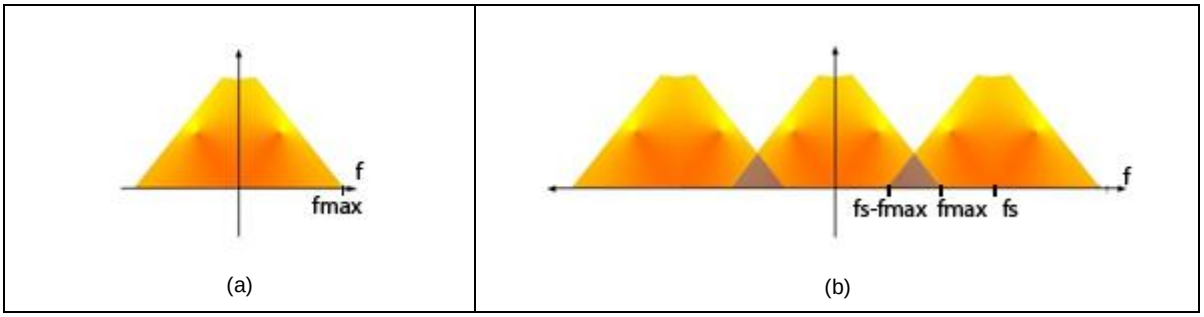

Figura 2.1 Espectro de frecuencia de la señal y de la señal muestreada

Aplicando un filtro pasa bajo se tiene el resultado mostrado en la figura 2.2

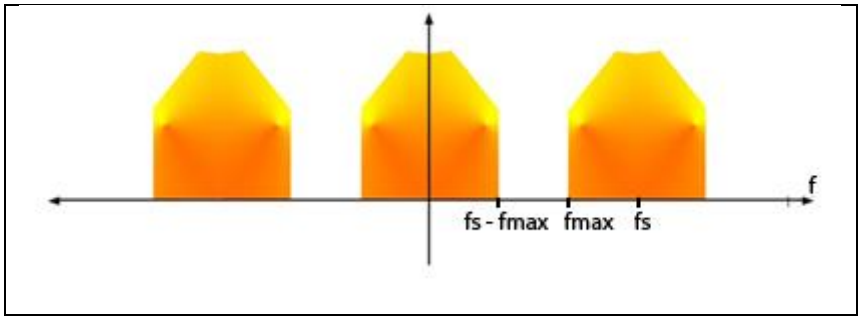

Figura 2.2 Espectro de frecuencias sin presentar fenómeno de traslape

Observamos que los valores de las frecuencias que están libres de traslape se encuentran en el siguiente intervalo

$$-fs + fmax < f < fs - fmax \tag{2.16}$$

Teniendo en cuenta este resultado, se elige un método para aplicar un filtro pasa bajo. Existen diversos métodos para aplicar un filtro pasa bajo sobre una imagen dada como:

filtro pasa bajo ideal, filtro butterworth pasa bajo y filtro gaussiano pasa bajo. [9]

Para el método aplicamos el filtro ideal pasa bajo, pues se desea evitar la interferencia de las frecuencias altas lo más que se pueda El filtro pasa bajo ideal se caracteriza por anular las altas frecuencias que se encuentran a una distancia mayor a un valor $D_0 > 0$ desde el origen.

Definiendo por $H(u,v)$ a la 2-D transformada discreta de Fourier del filtro, entonces:

$$H(u,v) = \begin{cases} 1 & si\ \ D(u,v) \leq D_0 \\ 0 & si\ \ D(u,v) > \ D_0 \end{cases} \tag{2.17}$$

Para una imagen digital dada, en general la 2D-FFT tiene las componentes centrales en las esquinas de la imagen, esto se muestra en la siguiente figura 2.3, donde los valores más altos de las componentes en frecuencia se dan en las frecuencias bajas y se ubican en las esquinas de la imagen (b).

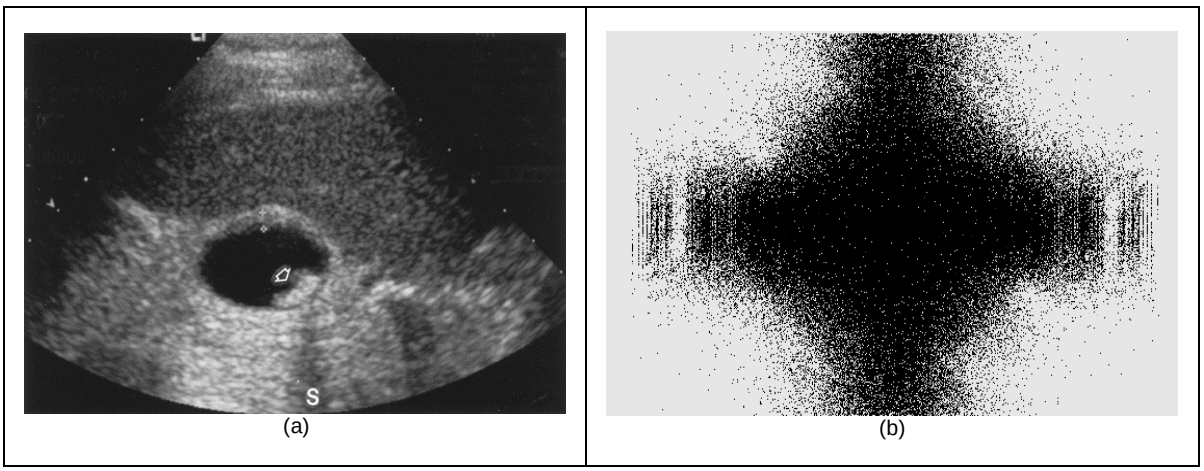

Figura 2.3 Izquierda imagen médica, derecha magnitud del espectro de frecuencias

Entonces para aplicar de forma muy sencilla los valores de las frecuencias más altas se traslada a la parte central de la imagen, como se muestra en la siguiente figura 2.4.

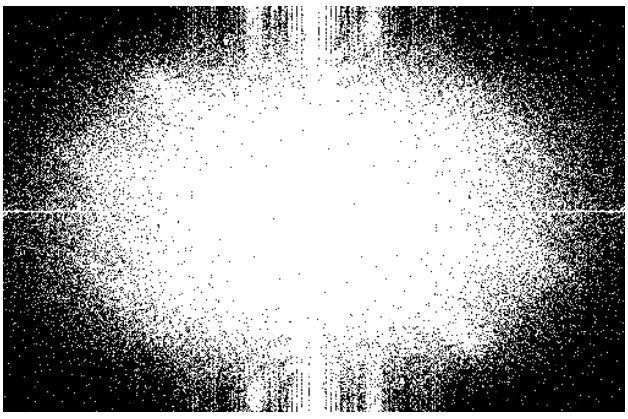

Figura 2.4 Componente de la frecuencia central trasladado a la parte central

Para imágenes de tamaño MxN, el centro del rectángulo se da en $(u,v) = (M/2, N/2)$, como la frecuencia central de la 2D-FFT se ubica en el centro, entonces

$$D(u,v) = [(u - M/2) + (v - N/2)]^{1/2} \tag{2.18}$$

La elección de D_0, el número de armónicos que se consideraran, se basa en la potencia de energía que encierra el circulo generado.

Como dato en promedio un radio de 5 abarca el 94.6% de la potencia total, con un radio de 30 encierra el 96.4%. [9]

La elección adecuada del radio se determinara en forma empírica, dado que el contenido de energía varía de una imagen a otra, esto debido a las condiciones mediante la cual se obtuvo una imagen, el tipo de escenario capturado, etc.

El diagrama de flujo correspondiente es

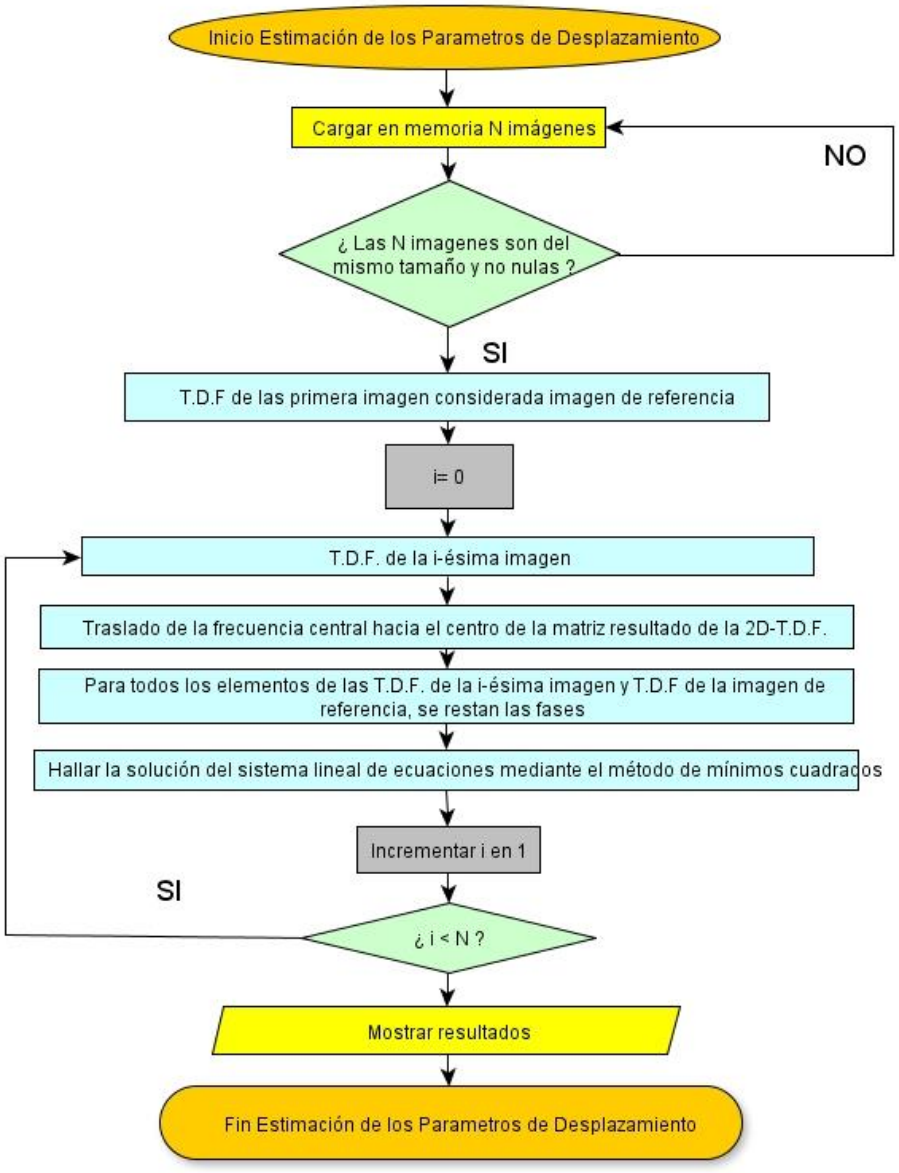

Figura 2.5 Diagrama de flujo para estimar el desplazamiento entre un conjunto de imágenes

2.1.2 Estimación de la Rotación entre dos Imágenes

La idea básica de estimar el ángulo de rotación entre dos imágenes es de rotar una imagen, un ángulo pequeño y comparar el resultado con la otra imagen, luego seguir

rotando y seguir comparando. El ángulo resultante en el cual se produce la máxima semejanza al comparar las dos imágenes, este será el ángulo estimado.

El parámetro que cuantifica en qué medida dos imágenes son semejantes es la correlación cruzada (Véase sección 1.5).

Determinar la correlación cruzada entre dos imágenes requiere de un cálculo en dos dimensiones lo cual computacionalmente seria complejo. Con el fin de disminuir la complejidad computacional y disminuir el tiempo de procesamiento, la correlación cruzada se calcula mediante el uso de la 2D transformada de Fourier.

El ángulo entre $|F_1(u,v)|$ y $|F_2(u,v)|$ puede ser calculado como el angulo para el cual la T.D.F. de la imagen de referencia $|F_1(u,v)|$ y la T.D.F. de la imagen del registro $|F_2(u,v)|$ tienen máxima correlación.

Para el cálculo de la rotación se genera el contenido de frecuencia h como una función del ángulo, en consecuencia determinar en qué ángulo está el contenido de energía más alto, esto es posible mediante la integración de $F(r,\theta)$ alrededor de la línea radial en coordenadas polares como se muestra en la siguiente figura 2.6.

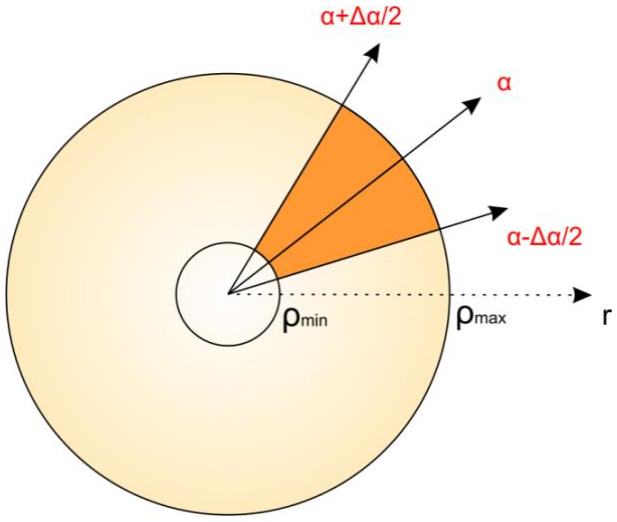

Figura 2.6 Región a considerar para el cálculo de $\boldsymbol{h(\alpha)}$

Para ello primero es necesario hacer el cambio de coordenadas a polares. Sean $F_1(u,v)$ y $F_2(u,v)$ las transformadas de Fourier de dos imágenes, representados en coordenadas cartesianas, entonces mediante el cambio de variables

$$r = \sqrt{u^2 + v^2} \tag{2.19}$$

$$\theta = \begin{cases} tg^{-1}(v/u) & si\ u > 0\ ; v \geq 0 \\ tg^{-1}(v/u) + 2\pi & si\ u > 0;\ v < 0 \\ tg^{-1}(v/u) + \pi & si\ u < 0 \\ \pi/2 & si\ u = 0;\ v > 0 \\ 3\pi/2 & si\ u = 0;\ v < 0 \end{cases} \tag{2.20}$$

Dónde: $\theta\ \epsilon\ [0; 2\pi]$

Se obtiene $F_1(r,\theta)$ y $F_2(r,\theta)$, en coordenadas polares

Con este cambio de variables, entonces la función $h(\alpha)$ estará definido por:

$$h(\alpha) = \int_{\alpha-\Delta\alpha/2}^{\alpha+\Delta\alpha/2} \int_{0}^{\infty} |F(r,\theta)|\, dr\, d\theta \tag{2.21}$$

Esta ecuación se cumple para señales continuas. Para una señal discreta la integral sobre $d\theta$ en el intervalo $\alpha - \frac{\Delta\alpha}{2} < \theta < \alpha + \frac{\Delta\alpha}{2}$ contiene valores finitos de $F(r,\theta)$ entonces

Se determina los valores discretos y extremos de θ tal que:

$$\alpha - \frac{\Delta\alpha}{2} < \theta_{min} < \theta < \theta_{max} < \alpha + \frac{\Delta\alpha}{2}$$

Con el fin de promediar el valor de $|F(r,\theta_i)|$ para $\theta_i \in [\theta_{min}, \theta_{max}]$

También para el caso digital, la variación del radio tomas valores finitos. Si ρ es el máximo valor del radio que se puede obtener al transformar el sistema de coordenadas, entonces el radio máximo y mínimo a considerar puede ser expresado como una fracción de ρ. Entonces la ecuación (2.20) puede expresarse para el caso digital, como:

$$h(\alpha) = \sum_{\theta_{min}}^{\theta_{max}} \sum_{a\rho}^{b\rho} |F(r,\theta)| = \sum_{a\rho}^{b\rho} \sum_{\theta_{min}}^{\theta_{max}} |F(r,\theta)| \tag{2.22}$$

Dónde: $0 < a < b < 1.0$

La precisión al calcular el ángulo de rotación será de $\Delta\alpha/2$ grados, $h(\alpha)$ es calculado cada intervalo $\Delta\alpha$. Como se observa, el promedio es solo evaluado en un disco circular de valores para el cual $a\rho < r < b\rho$ (donde ρ es el radio de la imagen)

El intervalo $\Delta\alpha$, divide el círculo de 360 grados en p espacios iguales, entonces $\Delta\alpha = \frac{360}{p}$.

Luego para la primera imagen la función $h_1(\alpha)$ se determina por:

$$h_1(\alpha) = \sum_{r=\rho min}^{\rho max} \sum_{\theta_{min}}^{\theta_{max}} |F_1(r,\theta)| \tag{2.23}$$

Dónde: $\alpha - \frac{\Delta\alpha}{2} < \theta_{min} < \theta_{max} < \alpha + \frac{\Delta\alpha}{2}$

Análogamente para la segunda imagen.

$$h_2(\alpha) = \sum_{r=\rho min}^{\rho max} \sum_{\theta_{min}}^{\theta_{max}} |F_2(r,\theta)| \tag{2.24}$$

Dónde: $\alpha - \frac{\Delta\alpha}{2} < \theta_{min} < \theta_{max} < \alpha + \frac{\Delta\alpha}{2}$

Con estos resultados se calcula la correlación entre las dos señales, para ello se emplea la Transformada Discreta de Fourier de las dos imágenes

$$H_1(k) = F(h_1(\alpha)) \tag{2.25}$$

$$H_2(k) = F(h_2(\alpha)) \tag{2.26}$$

Aplicando, la ecuación (1.27), la correlación finalmente será

$$c_{12}(\alpha) = \frac{1}{N} \mathrm{F_D^{-1}}\{H_1^*[\mathrm{k}].H_2\ [\mathrm{k}]\} \tag{2.27}$$

Con N el número total de elementos de la matriz.

La rotación exacta se determina para el valor en la cual la correlación es máxima.

Hasta el momento para estimar la rotación no se consideró la frontera de la imagen, obviamente al rotar la imagen se genera una zona negra que adiciona información falsa este fenómeno se muestra en la siguiente figura 2.7.

Figura 2.7 Izquierda imagen original, derecha imagen rotada 30 grados

Como se observa también parte de la imagen se pierde debido a la rotación. Este fenómeno provoca que la estimación en la rotación sea menos precisa. Una forma de evitar este fenómeno, es aplicar un filtro en el espacio del tiempo, con el fin de hacer simétrico a la región, se dispone de muchas herramientas para este fin como la ventana: haming, hanning, triangular, coseno cuadrado, blackman, káiser, gaussiana, etc.

Para nuestro caso se aplica la ventana Tukey o también llamado coseno cónico, se obtiene como la convolución de un lóbulo de coseno de ancho $\propto N/2$ con una ventana rectangular de ancho $\left(1-\propto \frac{1}{2}\right) N$.

Matemáticamente para L puntos digitales, la ventana Tukey se define por:

$$w(x) = \begin{cases} \frac{1}{2}\left[1+\cos\left(\frac{2\pi}{r}[x-r/2]\right)\right] & 0 \le x \le \frac{r}{2} \\ 1 & \frac{r}{2} \le x \le 1-\frac{r}{2} \\ \frac{1}{2}\left[1+\cos\left(\frac{2\pi}{r}[x-1+r/2]\right)\right] & 1-\frac{r}{2} \le x \le 1 \end{cases} \tag{2.28}$$

El diagrama de frecuencias es:

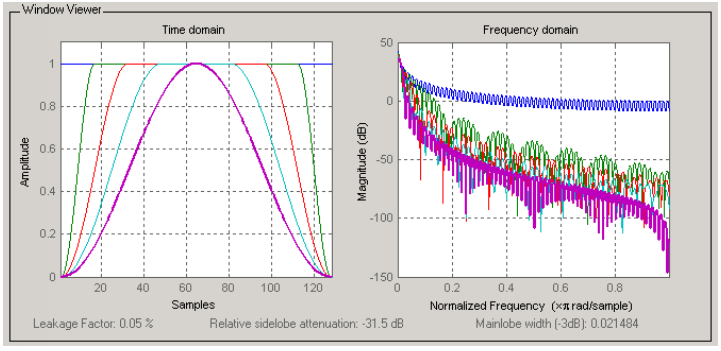

Figura 2.8 Ventana Tukey en el dominio del tiempo, derecha su respuesta en frecuencia

A continuación se muestra el diagrama de flujo del algoritmo.

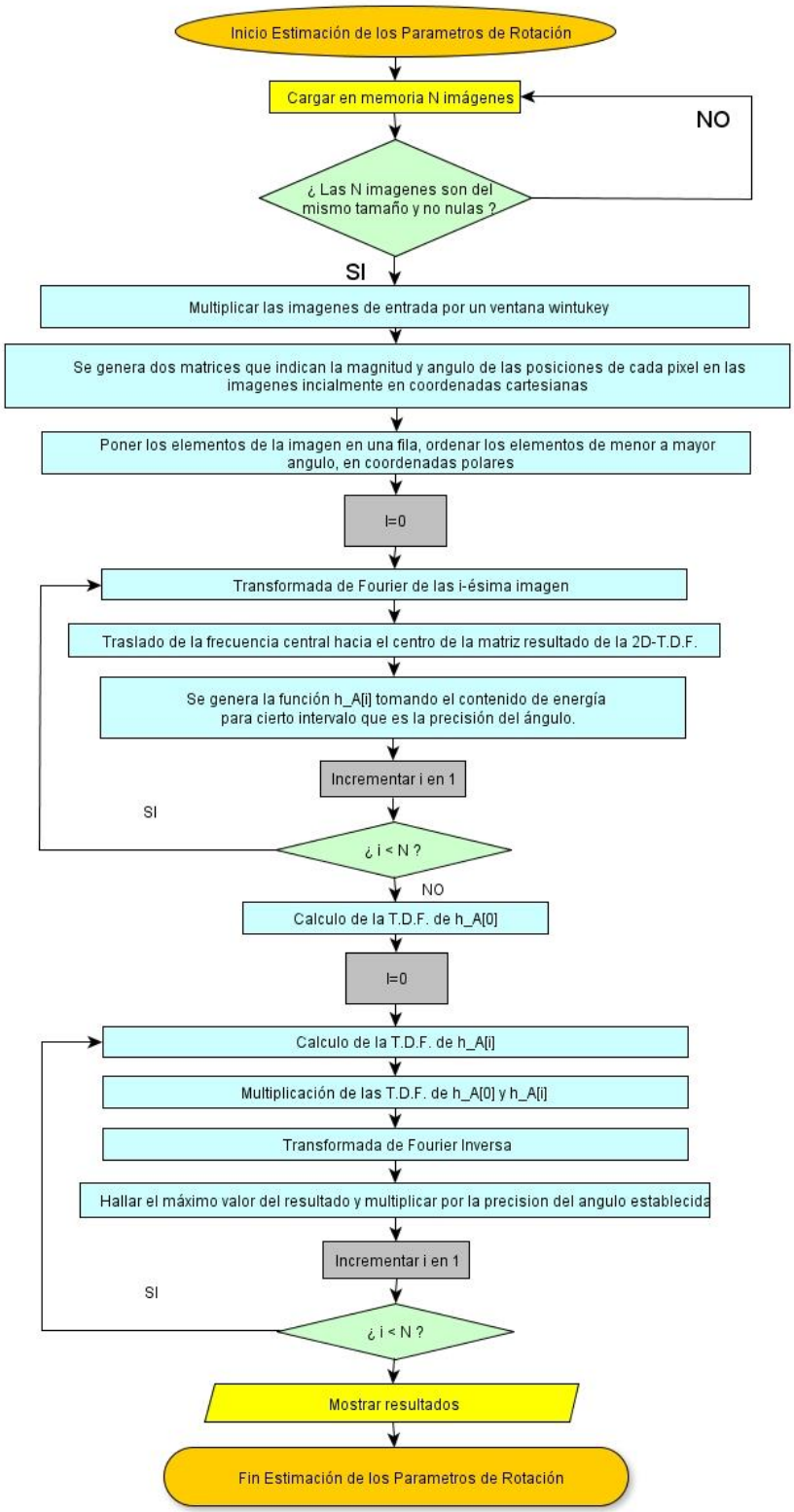

Figura 2.9 Diagrama de flujo para estimar la rotación entre un conjunto de imágenes

2.2 Generación de una Imagen en Súper Resolución

Una vez determinado los parámetros de movimiento (traslación y rotación) representado

por T_k, se usan las muestras de las diferentes imágenes de entrada para la construcción de una imagen en alta resolución.

Se tiene que resolver el problema de súper resolución enunciado en la sección (1.6.3)

$$g_K(m,n) = \sigma_k(\ \mathrm{h}(\ \mathrm{T}_k(f(x,y))\ ) + \eta_k(\mathrm{x},\mathrm{y})\ ) \qquad \text{para} \quad 1 \leq \mathrm{k} \leq \mathrm{N} \qquad (2.29)$$

Dónde:

g_K, es la imagen obtenida en un sistema de adquisición de imágenes.

$f(x,y)$, es la imagen deseada en súper resolución.

T_k, está determinado por los parámetros de desplazamiento y rotación.

h, es la función que genera el desenfoque en la imagen, modelado por el PSF.

η_k, es el ruido térmico.

σ_k, es la función que disminuye la tasa de muestreo, disminuye el tamaño de la imagen.

Existen diversos métodos para resolver la ecuación (1.28) como: el enfoque estadístico, máximum a posteriori, máximum likehood, enfoque bayesiano, etc. De entre muchas opciones se eligió el método por que se basa en el dominio de las frecuencias y fue presentado por Iraní y Peleg [20] y [14]. El método se denomina iterative backprojection (IBP), un método iterativo para la reconstrucción de la imagen en súper resolución.

La siguiente figura muestra esquemáticamente el método para genera iterativamente la imagen en súper resolución a partir de un conjunto de imágenes de entrada en baja resolución.

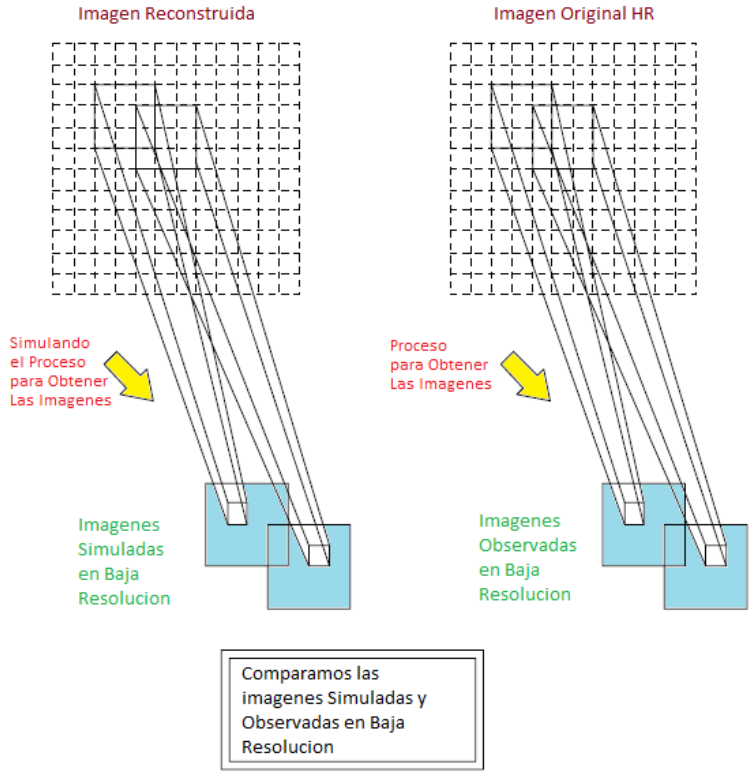

Figura 2.10 Diagrama esquemático del algoritmo IBP

Se comienza con una suposición inicial $f^{(0)}$ para la imagen en súper resolución. El proceso es simular un conjunto de imágenes en baja resolución $\{g_k{}^{(0)}\}$ correspondiente a las imágenes de entrada observadas $\{g_k\}$.Si $f^{(0)}$ es la imagen en súper resolución ideal, entonces las imágenes simuladas $\{g_k^{(0)}\}$ deben de ser idénticos a las imágenes en baja resolución $\{g_k\}$ y la diferencia $\{g_k - g_k^{(0)}\}$ seria cero, pero en realidad esto no sucede entre las imágenes, la diferencia no nula de $\{g_k - g_k^{(0)}\}$ se usa para mejorar la suposición inicial por retro alimentación, cada valor de la diferencia resultante se une a $f^{(0)}$. Este proceso se repite para minimizar la función de error expresado por:

$$e^{(n)} = \sqrt{\frac{1}{K}\sum_{k=1}^{K} \left\| g_k - g_k^{(n)} \right\|_2^2} \tag{2.30}$$

En base a este modelo, las imágenes simuladas, se expresan mediante la siguiente ecuación que tiene en cuenta los efectos que degradan la imagen como el: desenfoque, el traslape, etc.

$$g_k^{(n)} = \left(T_k\left(f^{(n)}\right) * h\right) \downarrow s \tag{2.31}$$

Donde $\downarrow s$ denota al operador que disminuye la frecuencia de muestreo por un factor s y el símbolo $*$ es el operador convolución, k es la imagen en baja resolución numero k.

g_k: Es la imagen sensada del objeto en movimiento en la posición k del registro de imágenes de entrada.

f: Es una imagen de alta resolución del objeto estudiado en una vista de reconstrucción deseada. Encontrar f es el objetivo del algoritmo de reconstrucción

T_k: Es una función geométrica desde f hasta gk determinado por los parámetros de movimiento calculados en 2-D y del objeto estudiado en la imagen plana, es invertible.

h: Función blurring, determinado por el "Point Shared Function" del sensor. Cuando no se tiene conocimiento de las propiedades del sensor, se asume que es gaussiano.

$\downarrow s$ Es una función que simula el efecto de muestrear la señal por debajo de la frecuencia de Nyquist en un factor s.

El esquema iterativo para generar la imagen en súper resolución matemáticamente es:

$$f^{(n+1)} = f^{(n)} + \text{informacion adicional} \tag{2.32}$$

La información adicional parte de la diferencia entre la imagen original y la imagen simulada. Sea $g_k^{(n)}$ la n-esima imagen simulada a partir de $f^{(n)}$, la información entonces la diferencia es.

$$diferencia = g_k - g_k^{(n)} \tag{2.33}$$

Siguiendo el inverso del proceso que sufre la imagen se amplifica en un factor s

$$diferencia\ amplificada = \left(\left(g_k - g_k^{(n)}\right) \uparrow s\right) \tag{2.34}$$

Luego se filtra con un núcleo p y se aplica los parámetros de movimiento mediante T_k^{-1}, luego

$$\text{informacion adicional de una imagen} = T_k^{-1}\left(\left(\left(g_k - g_k^{(n)}\right) \uparrow s\right) * p\right) \tag{2.35}$$

Donde, $\uparrow s$ es una función que aumenta la frecuencia de muestreo

Finalmente la información adicional será la suma total de la información adicional que proporciona cada imagen dividido por un factor K. Si K=N la información adicional seria el promedio.

$$f^{(n+1)} = f^{(n)} + \frac{1}{K}\sum_{k=1}^{K} T_k^{-1}\left(\left(\left(g_k - g_k^{(n)}\right) \uparrow s\right) * p\right) \tag{2.36}$$

El algoritmo es numéricamente similar a los métodos de iteración para resolver un sistema lineal de ecuaciones, entonces tiene propiedades similares como la rapidez de convergencia. Depende de si existe la convergencia para obtener una imagen en súper resolución.

Una especial consideración de la ecuación (2.36) es, analizar bajo qué condiciones este método converge a una solución que sería la imagen en súper resolución.

De (2.36) se incluye el valor de $f^{(n)}$ dentro de la sumatoria.

$$f^{(n+1)} = \frac{1}{K}\sum_{k=1}^{K} f^{(n)} + T_k^{-1}\left(\left(\left(g_k - g_k^{(n)}\right) \uparrow s\right) * p\right) \tag{2.37}$$

Entonces

$$f^{(n+1)} = \frac{1}{K}\sum_{k=1}^{K} T_k^{-1}\left(T_k\left(f^{(n)}\right) + \left(\left(g_k - g_k^{(n)}\right) \uparrow s\right) * p\right) \tag{2.38}$$

Aprovechando que $T_k\left(f^{(n)}\right) = T_k\left(f^{(n)}\right) * \delta$ donde δ denota la función impulso unitario centrado en el origen

$$f^{(n+1)} = \frac{1}{K}\sum_{k=1}^{K} T_k^{-1}\left(T_k\left(f^{(n)}\right) * \delta \ + \left((g_k) \uparrow s\right) * p - \left(\left(g_k^{(n)}\right) \uparrow s\right) * p\right) \tag{2.39}$$

A partir de la ecuación (2.31)

$$T_k\left(f^{(n)}\right) * h = \left(\left(g_k^{(n)}\right) \uparrow s\right) \tag{2.40}$$

Reemplazando (2.40) en el último factor de la sumatoria en (2.39)

$$f^{(n+1)} = \frac{1}{K}\sum_{k=1}^{K} T_k^{-1}\left(T_k\left(f^{(n)}\right) * \delta \ + \left((g_k) \uparrow s\right) * p - T_k\left(f^{(n)}\right) * h * p\right) \tag{2.41}$$

Simplificando

$$f^{(n+1)} = \frac{1}{K}\sum_{k=1}^{K} T_k^{-1}\left(T_k\left(f^{(n)}\right) * (\delta - h * p) \ + \left((g_k) \uparrow s\right) * p\right) \tag{2.42}$$

De esta última expresión, la ecuación es convergente si $\lim_{n\to\infty} f^{(n)} = f$, donde f es la imagen ideal en alta resolución deseada, entonces reemplazando (2.42) en (2.40) se tiene:

$$\mathrm{f} = \frac{1}{K}\sum_{k=1}^{K} T_k^{-1}\left(T_k(\mathrm{f}) * (\delta - h * p) \ + \left((g_k) \uparrow s\right) * p\right) \tag{2.43}$$

En el artículo de Irani y Peleg [14], se demuestra que la convergencia queda asegurada si se satisface la siguiente condición.

$$\|\delta - h * p\|_2 < \ 1 \tag{2.44}$$

Donde δ denota la función de pulso unitario centrado en el origen.

Como se observa, no hay mayores restricciones sobre el núcleo p más que la ecuación 2.44.

Es importante notar que las altas frecuencias originales no siempre permiten una restauración. Por ejemplo, si la función de desenfoque es un filtro ideal pasa bajo y la transformada de Fourier tiene valores de cero para altas frecuencia, es obvio que las componentes de frecuencia que hayan sido filtrado no hayan podido ser restaurados, en consecuencia existen muchas soluciones posibles y el algoritmo puede converger a uno de ellos u oscilar entre uno de ellos, la elección de las condiciones iniciales no influencia el rendimiento del algoritmo (velocidad o estabilidad), sin embargo puede influenciar en que la posible solución se alcance primero. Una buena elección de las condiciones iniciales, es el promedio de las imágenes en baja resolución.

$$f^{(0)} = \frac{1}{K}\sum_{k=1}^{K} T_k^{-1}(g_k) \tag{2.45}$$

Otro es la elección del núcleo de retro alimentación p. Únicamente h que representa las propiedades del sensor (PSF), hay una cierta libertad en la elección de p.

Reducción del Ruido

De la ecuación (2.36), si el nuevo valor del pixel en alta resolución para cada iteración se obtiene tomando el promedio de todas las contribuciones de los diversos pixeles en baja resolución. Tomando el promedio por sí mismo implicaría agregar ruido, por ello para evitar la propagación del ruido, las contribuciones extremas (alto y bajo) se dejan de considerar antes de tomar el promedio. Solo las contribuciones de aquellos cuyos valores son cercanos al máximo y mínimo, son considerados, eliminando el ruido aditivo y multiplicativo. [14]

2.2.1 Fenómeno de Anillos

Como se vio el algoritmo IBP, en cada iteración se realiza un aumento y disminución de la tasa de muestreo sobre una imagen (ecuación 2.31 y 2.43), esto se logra aplicando un método de interpolación, debido a que la posición de los pixeles luego de aumentar o disminuir la tasa de muestreo no siempre se ubica en posiciones discretas y como la imagen es digital con pixeles en posiciones discretas, es necesario un método de interpolación que aproxime el valor del pixel considerando los pixeles vecinos antes del cambio de muestreo.

Otra sección del algoritmo que requiere interpolación y por ende produce también el fenómeno de anillos es cuando una imagen se rota. Tanto el cambio de la frecuencia de muestreo como la rotación forman parte del tema de transformaciones lineales sobre imágenes, más información en [21].

La aplicación de un método de interpolación desde el punto de vista de las frecuencias, se comporta como un filtro pasa bajo, lo cual puede expresarse como la convolución de la imagen con un núcleo o kernel, por ejemplo para la interpolación cubica de Keys. [22]

Sea f(x) una señal discreta y g(x) la función luego de ampliar la tasa de muestreo, entonces.

$$g(x) = \sum_k \mathrm{f}(x_k) * \mathrm{u}\left(\frac{x - x_k}{h}\right) = \sum_k \mathrm{f}(x_k) * \mathrm{u}(s) \tag{2.46}$$

Dónde: h representa el incremento de la frecuencia y la función u denominado kernel o núcleo es:

$$u(s) = \begin{cases} \frac{3}{2}|s|^3 - \frac{5}{2}|s|^2 + 1 & 0 < |s| < 1 \\ -\frac{1}{2}|s|^3 + \frac{5}{2}|s|^2 - 4|s| + 2 & 1 < |s| < 2 \\ 0 & 2 < |s| \end{cases} \tag{2.47}$$

El radio del núcleo es de 2 porque para $2 < |s|$, u(s) toma el valor de 0.

Sin pérdida de generalidad el núcleo de un método de interpolación en el dominio de las frecuencias se comporta como un filtro pasa bajo. Para aplicaciones sucesivas de este algoritmo sobre una imagen dada genera una distorsión en los bordes de los objetos dentro de la imagen, a esta distorsión se le denomina fenómeno de anillos.

Los métodos clásicos de interpolación tanto para aumentar y disminuir la tasa de muestreo son: Interpolación bilineal, bicúbica, del vecino más próximo (nearest neightbor - NN). Con el fin obtener el mejor resultado de aplicar la interpolación, con una menor distorsión, se consideran los métodos: lanczos3, lanczos4, z-chirp.

De [23] se muestra el siguiente resultado mediante la figura 2.11 que compara el efecto de los métodos de interpolación sobre los bordes, generando el fenómeno de anillos.

Como se observa la interpolación bicúbica genera menos distorsión comparado a la interpolación bilineal y NN.

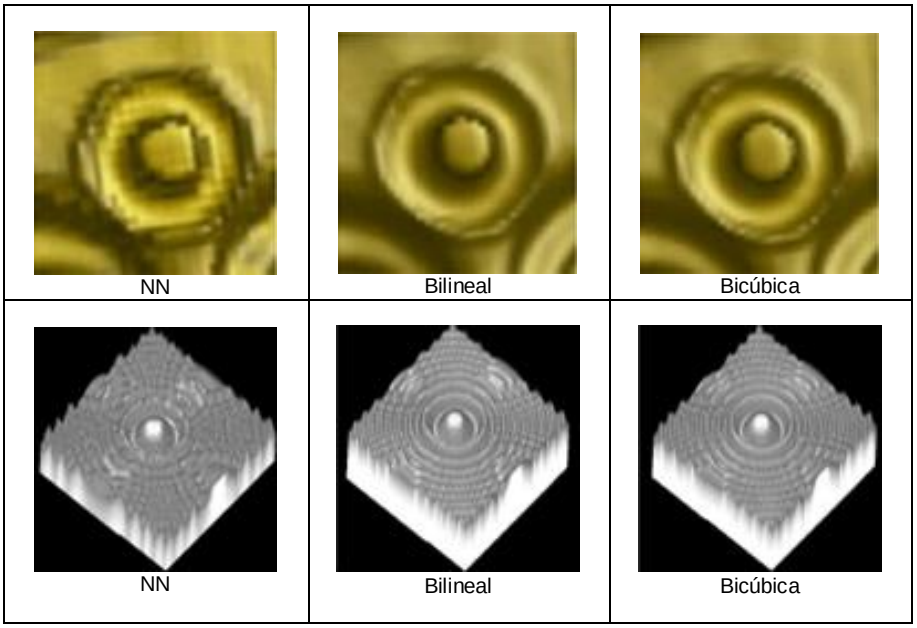

Figura 2.11 Distorsión generada al aplicar los métodos de interpolación

Del artículo científico mostrado en [22] se tiene la respuesta en frecuencia de los 3 métodos comparados en la figura 2.11 además del filtro ideal pasa bajo, esto se muestra en la siguiente figura 2.12

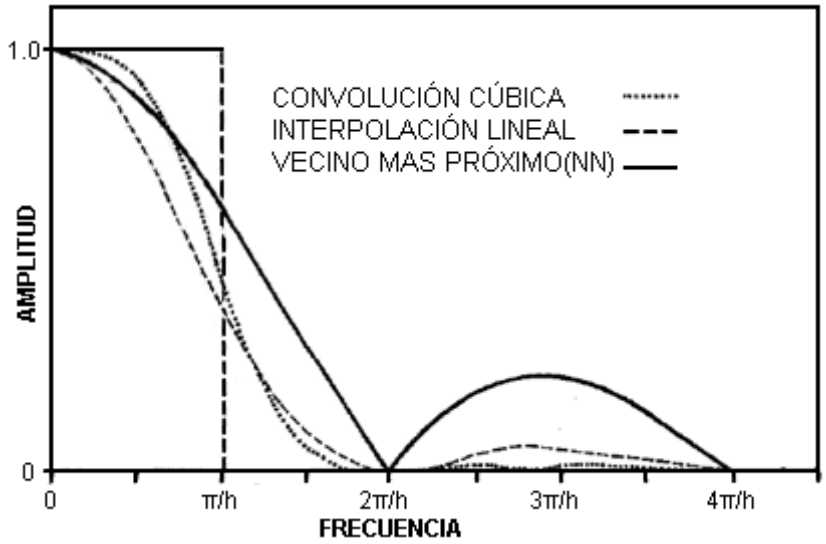

Figura 2.12 Espectro de magnitud correspondiente al método de interpolación

Como se observa la interpolación cúbica se aproxima más al filtro ideal en comparación de los otros métodos. La diferencia entre interpolación cubica y bicúbica es que el primero se aplica en una dimensión y el segundo en dos dimensiones.

Existe otro método de interpolación caracterizado por la función que resulta de la transformada inversa de Fourier del filtro pasa bajo, la función Sinc(x) definido por:

$$Sinc(s) = \begin{cases} 1 & 0 < |s| < 1 \\ \dfrac{\sin(\pi x)}{\pi x} & 1 < |s| < 2 \end{cases} \quad (2.48)$$

El método de interpolación Lanczos toma un intervalo de la función Sinc(x), en otras palabras es una versión truncada de Sinc(x), si es Lanczos3 el radio del núcleo respectivo es de 3 y si es Lanczos4 su radio es de 4. Puede obtenerse una mejor estimación de la imagen incrementando el radio del núcleo, este incremento también lleva a un mayor tiempo de procesamiento.

2.2.2 Filtro Bilateral

De la ecuación mostrada en (2.44), no hay restricción acerca de la elección del núcleo p más que dicha ecuación, como hipótesis se toma el efecto que produce la convolución de p con la imagen correspondiente mediante un filtro bilateral.

El filtro bilateral se caracteriza por suavizar la imagen y preservar los bordes. El método es no iterativo, local y simple. [24]

La razón de porque usar este filtro, es porque preserva los bordes, es en estos bordes donde se tiene la información las frecuencias más altas, conforme se van realizando las iteraciones para generar la imagen en súper resolución, se va perdiendo una parte del contenido de frecuencias altas. Este problema se menciona en el siguiente artículo científico (25)

En la siguiente figura 2.13 se muestran el resultado de aplicar el filtro bilateral

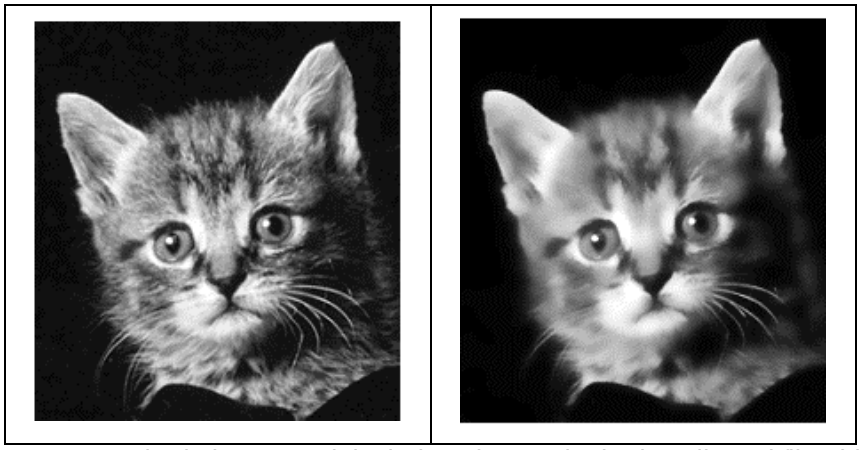

Figura 2.13 Izquierda imagen original, derecha resultado de aplicar el filtro bilateral

Como se observa luego de aplicar el filtro bilateral, no se distorsionan los bordes ni las líneas finas como el caso de los bigotes del gato, conservando así las componentes de frecuencias altas.

Diagrama de Flujo

En la siguiente página se muestra el diagrama de flujo del algoritmo para generar la imagen en súper resolución.

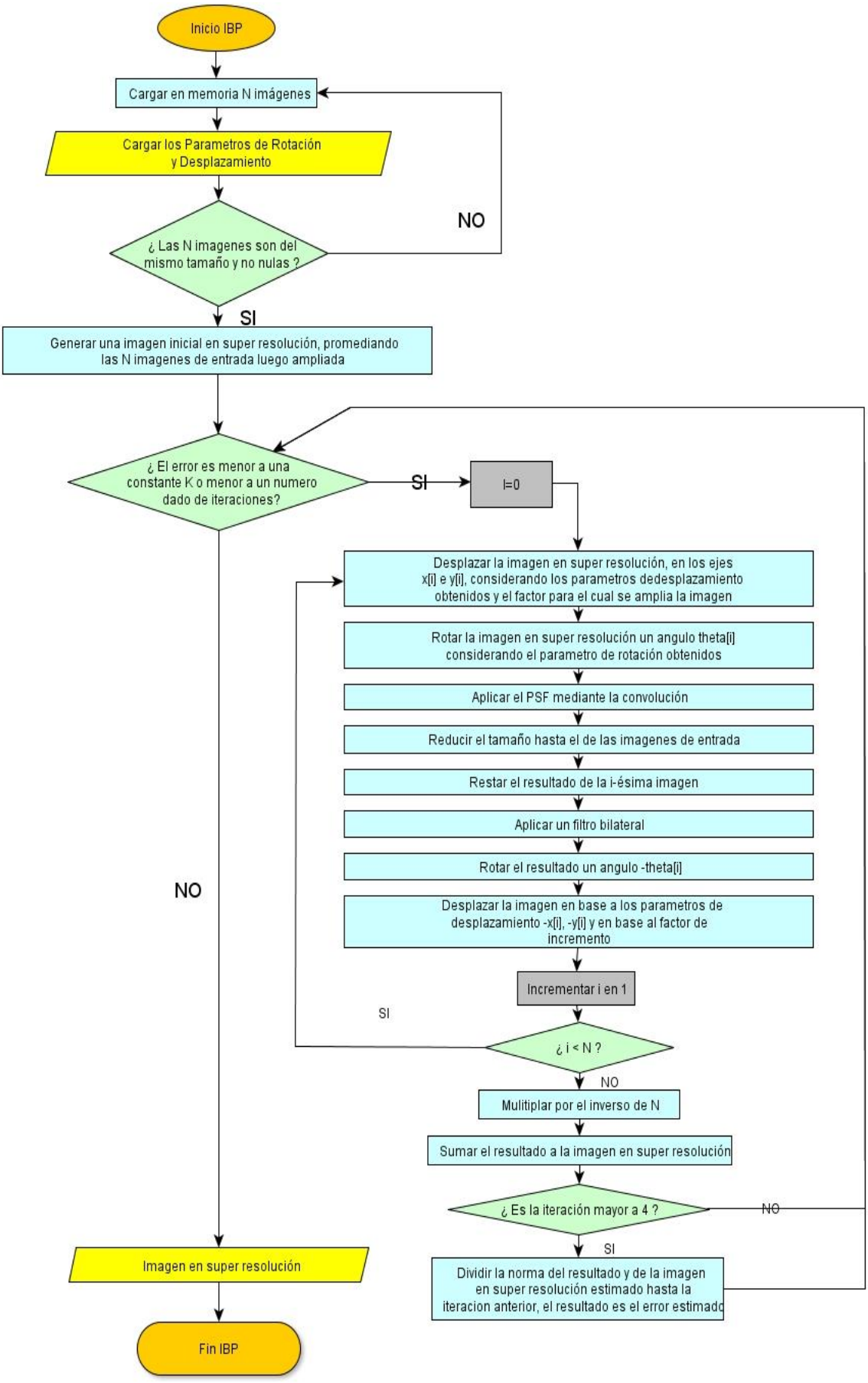

Figura 2.14 Diagrama de flujo propuesto para el algoritmo IBP

2.3 Software para la Implementación del Algoritmo

El desarrollo de este algoritmo se realizó bajo el sistema operativo Windows 7, vía el software Visual Studio 2008 para usar C/C++, debido a que los cálculos realizados operan sobre matrices y debido a que la eficiencia en el tiempo es un parámetro importante, se empleó la librería de código abierto OpenCV 2.3.1. Puede conseguirse gratuitamente desde el internet.

Para la manipulación de imágenes médicas en formato DICOM, se emplea la librería también de código abierto GDCM.

Finalmente para aplicar programación en paralelo sobre el algoritmo, se emplean las librerías CUDA y OpenCL que están disponibles para diversos sistemas operativos. Mayor referencia véase el Anexo C.

2.4 Implementación del Algoritmo para la Generación de la Imagen en Súper Resolución mediante el Procesamiento en Paralelo

Ejecutar los algoritmos usando procesamiento en paralelo, requiere de una serie de análisis para determinar que partes del mismo pueden procesarse en una GPU. Los tipos de análisis a realizar se muestran a continuación.

- Se descompone el algoritmo en sub tareas y se determina la dependencia entre ellas.
- Para cada sub tarea, se analiza si es factible usar paralelismo de tareas o paralelismo de datos.
- Es necesario analizar la granularidad del proceso, teniendo en cuenta la cantidad de datos que se procesara y tomar una decisión de paralelizar el proceso. Cuando se paraleliza un proceso, se procede a programar sobre la GPU. Al código de programación a ejecutar se le denomina kernel.

Antes de iniciar la programación exclusiva del kernel, es necesario detectar la GPU disponible y sobre el cual se ejecutará el kernel. Tanto CUDA como OpenCL poseen funciones para la finalidad. Técnicamente primero se detecta el número de plataformas denominados huéspedes (host) y luego sobre cada plataforma se detecta el número de unidades GPU disponibles.

El siguiente paso principal es transferir los datos desde el huésped hasta el dispositivo GPU.

Después de completar los pasos mencionados llamamos desde el huésped mediante una función en C/C++ al kernel que se ejecuta sobre la GPU.

Luego de ejecutar el kernel se copian los resultados de vuelta al CPU.

A continuación se menciona las secciones que se han programado en paralelo.

Estimación de Desplazamiento

Del diagrama de flujo correspondiente es posible aplicar paralelismo de datos al cálculo de la 2D-T.D.F y al cálculo del desfase entre cada elemento complejo que corresponden al resultado de la 2D-T.D.F.

Sin pérdida de generalidad y por sencillez, mostramos los pasos para la programación en paralelo vía OpenCL del algoritmo que determina el desfase entre 2 elementos complejos.

- Cargar las dos matrices de elementos complejos en la CPU.
- Determinar el número de plataformas.
- Detectar la cantidad de unidades GPU.
- Crear un contexto, que almacena la información de la arquitectura hardware.
- Crear un command queue, un conjunto de comandos para ejecutar el kernel.
- Copiar los elementos de las matrices de entrada al GPU mediante un buffer.
- Ejecutar el kernel, especificando el número de elementos y grupos de trabajo.
- Copiar el resultado de vuelta al CPU.

El diagrama de flujo para determinar el desfase entre dos matrices complejas.

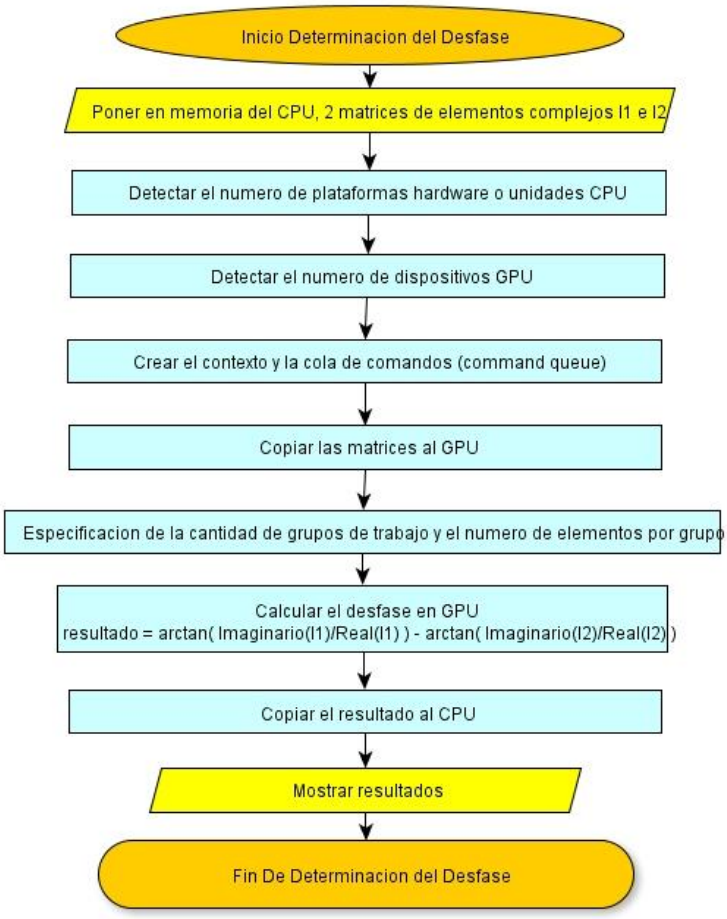

Figura 2.15 Diagrama de flujo para ejecutar un algoritmo con procesamiento en paralelo

También es factible paralelizar el cálculo de la T.D.F. sin embargo, ya existen librerías en OpenCL y CUDA que hace este cálculo, por sencillez y tiempo se emplea esta librería.

Estimación de Rotación

Aparte de la 2D-T.D.F, se encontraron algunos pasos del algoritmo que puede paralelizarse: suma y resta de matrices. El diagrama de flujo para este programa mantiene el esquema general que se muestra en la figura 2.15, solo cambia el código del kernel.

IBP

Para este algoritmo, se paralelizaron las siguientes funciones

- Rotación de una imagen con interpolación bicúbica
- Filtro de una imagen según el PSF
- Filtro bilateral
- Desplazamiento en el eje x e y
- Resta de matrices
- Norma de una matriz
- Ampliación y reducción de una matriz mediante interpolación bicúbica

Muchas de estas funciones están disponibles en la última versión de la librería de OpenCV 2.4.5 que se ejecuta sobre la GPU NVIDIA, de ahí su importancia en el proyecto, sin embargo al inicio del proyecto, en versiones anteriores de esta librería, no se contaba con muchas de las funciones mencionadas, en el anexo se adjunta algunos códigos del kernel.

CAPÍTULO III
RECOLECCIÓN DE DATOS Y PRUEBAS PARA OPTIMIZAR LA EFICIENCIA DE LOS ALGORITMOS UTILIZADOS

3.1 Recolección de Datos

Los datos, imágenes médicas, han sido obtenidos en el Hospital Cayetano Heredia.

3.2 Pruebas Sintéticas

La prueba sintética consiste en generar imágenes de menor resolución desde solo una imagen original de mayor resolución, se simula sobre cada imagen de menor resolución los fenómenos que deterioran la imagen como: el desenfoque, etc, para luego generar la imagen en súper resolución. El objetivo de la prueba sintética es permitir comparar en qué medida la imagen en súper resolución se asemeja a la imagen original, para ello hay diversos indicadores como el PSNR y el MSR.

Condiciones del Experimento

La imagen original se obtiene mediante una cámara digital y usando Matlab se genera 4 imágenes en menor resolución. Se mantiene constante todas las variables de entrada, solo haciendo pequeñas variaciones en la variable de interés.

3.2.1 Efecto del Método Empleado para Incrementar o Disminuir la Tasa de Muestreo

Incrementar la tasa de muestreo aumenta la resolución espacial. Existen diversos métodos de interpolación para realizar esta tarea como: bilineal, bicúbica, lanczos4, etc.

La tabla Nº 3.1 muestra el PSNR obtenido al variar el método de interpolación para incrementar la tasa de muestreo.

Tabla Nº 3.1 PSNR obtenido para diferentes métodos de interpolación

Método para Disminuir la Tasa de Muestreo	Método para Aumentar la Tasa de Muestreo	Original y Súper Resolución
Bilineal	Bilineal	PSNR: 28.404905
Bilineal	Bicúbica	PSNR: 29.187547
Bilineal	Lanczos4	PSNR: 29.293773

La tabla Nº 3.2 muestra el PSNR obtenido luego de variar el método de interpolación utilizado para reducir la tasa de muestreo.

Tabla Nº 3.2 PSNR obtenido para diferentes métodos de interpolación

Método para la Disminución de la Tasa de Muestreo	Método para el Aumento de la Tasa de Muestreo	Original y Súper Resolución
Bilineal	Lanczos4	PSNR: 29.293773
Bicúbica	Lanczos4	PSNR: 29.384956
Lanczos4	Lanczos4	PSNR: 29.545451

Notara que los resultados obtenidos en la Tabla Nº 3.1 y 3.2 nos llevan a la conclusión de que el PSNR más alto se obtiene aplicando el método de interpolación Lanczos4. El valor de PSNR más bajo obtenido corresponde a la interpolación bilineal.

Basados en los resultados que se muestra en la figura 2.11, entre la interpolación bicúbica y bilineal es la última que se acerca más al filtro ideal dejando pasar más contenido de frecuencias altas. Existe la relación entre la respuesta de frecuencia de la interpolación con la calidad del resultado y su relación con el filtro ideal.

3.2.2 Efecto de la Elección de las Componentes en Frecuencia para la Generación del Sistema Lineal de Ecuaciones

La estimación de los parámetros de desplazamiento implica resolver un sistema lineal de ecuaciones definido por la ecuación (2.14). Debido al fenómeno de traslape no es posible seleccionar todos los valores de $F_1(u,v)$ y $F_2(u,v)$ (véase la sección 1.3.3), por ello solo se considera las componentes en frecuencia sin traslape. El parámetro de precisión para estimar el desplazamiento, indica el número de componentes en frecuencia bajas a considerar.

Se realiza una prueba sintética para conocer el valor más apropiados del parámetro de precisión. En este caso se emplea una imagen de 768x768 con desplazamientos en el eje x e y según la tabla Nº 3.3.

Tabla Nº 3.3 Condiciones establecidas para la prueba sintética

Imagen	Desplazamiento eje x	Desplazamiento eje y
1(referencia)	0	0
2	1	1
3	-1	3
4	-6	6

A continuación la tabla Nº 3.4 muestra los resultados obtenidos, mostrando la dependencia entre el parámetro de precisión y el desplazamiento estimado.

Tabla Nº 3.4 Error absoluto, al estimar el desplazamiento en el eje x e y

Parámetro de Precisión	Desplazamiento eje X			Desplazamiento eje Y		
	real	estimado	%error relativo	real	estimado	%error relativo
5	1	0.921616	7.8%	1	0.926366	7.4%
	-1	-0.722575	27.7%	3	2.427411	19.1%
	-6	-4.653667	22.4%	6	4.395611	26.7%
10	1	0.972834	2.7%	1	0.995740	0.4%
	-1	-0.950594	4.9%	3	2.932204	2.2%
	-6	-5.761166	4.0%	6	5.823988	2.9%
15	1	0.982561	1.7%	1	0.987866	1.21%
	-1	-0.981783	1.8%	3	2.970165	1.0%
	-6	-5.945998	0.9%	6	5.951619	0.8%
20	1	1.02377	2.4%	1	0.926307	7.3%
	-1	-0.986713	1.3%	3	2.995819	0.1%
	-6	-5.967621	0.5%	6	6.010841	0.2%
25	1	1.016959	1.7%	1	0.959828	4.0%
	-1	-0.984185	1.6%	3	2.988890	0.4%
	-6	-5.980472	0.3%	6	5.986705	0.2%
30	1	1.010380	1.0%	1	0.980297	2.0%
	-1	-0.985153	1.5%	3	2.994800	0.2%
	-6	-5.991262	0.1%	6	5.990355	0.2%
35	1	1.009038	0.9%	1	0.984580	1.5%
	-1	-0.984682	1.5%	3	2.994765	0.2%
	-6	-5.346663	10.9%	6	5.350919	10.8%
40	1	0.997506	0.2%	1	0.985627	1.4%
	-1	-1.003719	0.4%	3	2.992493	0.3%
	-6	-3.721004	38.0%	6	3.737089	37.7%
45	1	0.999606	0.04%	1	0.989905	1.0%
	-1	-1.002078	0.2%	3	2.992124	0.3%
	-6	-2.355445	60.7%	6	2.351981	60.8%
50	1	0.999410	0.1%	1	0.995925	0.4%
	-1	-1.000778	0.1%	3	2.996062	0.1%
	-6	-1.361764	77.3%	6	1.347400	77.5%

Los resultados muestran que cuando el parámetro de precisión es n = 15, 25, 30, el error relativo en la estimación es menor al 5%. Cuando el parámetro de precisión es menor a 15 y mayor de 30 el error se incrementa sobre todo cuando el desplazamiento entre las imágenes también se incrementa.

A continuación se realiza otra prueba sintética para determinar el máximo desplazamiento que se puede estimar con un error absoluto menor a 1 pixel. Esto puede expresarse en función del porcentaje con respecto a las dimensiones de la imagen. Las condiciones establecidas para la ejecución de la prueba son:

Parámetro de precisión = 25

Ancho de la imagen = 768 y alto de la imagen = 768

La siguiente tabla Nº 3.5 muestran los resultados del experimento.

Tabla Nº 3.5 Error relativo en la estimación de los parámetros de desplazamiento

Imagen	dy1	dy1 estimado	Error Relativo
1	0	0	0
2	1	0.978871	2.1%
3	2	1.965889	1.7%
4	3	2.976160	0.8%
5	4	3.968725	0.8%
6	5	4.962338	0.8%
7	6	5.965326	0.6%
8	7	6.968250	0.5%
9	8	7.929707	0.9%
10	9	8.935896	0.7%
11	10	9.920687	0.8%
12	11	10.926553	0.7%
13	12	11.956691	0.4%
14	13	12.951473	0.4%
15	14	13.802940	1.4%
16	15	14.642874	2.4%
17	16	10.638419	33.5%
18	17	6.877971	59.5%
19	18	4.807096	73.3%

Considerando un error relativo menor a 5% como una estimación valida, de los resultados se obtiene una estimación con buena precisión, de hasta 15 pixeles.

Podemos estimar un porcentaje máximo de desplazamiento.

$$\%\, max.de\ desplazamiento = \frac{15}{786} * 100\% = 1.90\% \tag{3.1}$$

Este resultado implica que obtener múltiples imágenes como registro para la generación de una imagen en súper resolución, requiere que las imágenes deban obtenerse cuidando de un desplazamiento pequeño entre ellas menor al 1.9% del ancho y alto de la imagen. En el caso de cámaras fotográficas, es preferible emplear un trípode sobre el

cual se ponga la cámara fotográfica, otra opción es aplicar fotos simultaneas una herramienta que viene en las cámaras digitales modernas, con esta herramienta es posible tomar 4 o más imágenes en menos de un segundo. Finalmente puede generarse un registro de imágenes desde una secuencia de video ya que en general un video contiene para 1 segundo de reproducción, desde 25 frames a más.

3.2.3 Efecto de la Precisión en la Estimación de la Rotación

El algoritmo que estima la rotación tiene 3 parámetros de entrada que son independientes las cuales son: parámetro de precisión, el radio minimo y máximo de integración a considerar, denotados respectivamente como dmin y dmax.

El método escogido para determinar los valores apropiados de los parámetros de entrada será aproximar el comportamiento del sistema a un sistema lineal, esto es, haciendo variaciones pequeñas en cada parámetro de entrada y a la vez manteniendo las demás constantes.

A continuación se realiza una prueba sintética para el cual, se generan nuevas imágenes generadas por la rotación de una imagen de referencia, el Angulo de rotación se muestra en la tabla Nº 3.6.

En total se usan 13 imágenes al azar, cuyo tamaño y contenido de frecuencias es variable, procurando tener las más diversas situaciones que se tiene en una escena fotografiada.

Tabla Nº 3.6 Resultados en la estimación de la rotación

Imagen	Rotación 1	Rotación 2	Rotación 3
1	4.0	0.5154	0.0000
2	-2.0	2.3454	0.0010
3	5	-3.6554	0.0015
4	-1	5.0423	0.0020
5	10	-7.0145	0.2500
6	-11	9.1333	0.3000
7	12	-8.0796	0.3500
8	20	11.2524	0.4000
9	18	-13.8957	0.4500
10	-20	14.8445	0.5000
11	2.9	-17.2365	0.5500
12	24.1	19.7611	0.6000
13	29.1	-21.3377	0.6500
14	-20.0	23.6598	0.7000

Primero se hace varia el parámetro de precisión, de 0.1 y 0.01 manteniendo las demás variables constantes, los resultados se muestran en las siguientes tablas

Tabla Nº 3.7 Resultados al estimar la rotación con el parámetro de precisión de 0.1

Imagen	Rotación 1	error absoluto	Rotación 2	error absoluto	Rotación 3	error absoluto
1	4.1	0.1	0.1	0.0846	0.6	0.1
2	-1.9	0.1	0.1	0.0546	2.4	0.099
3	5.1	0.1	0.1	0.0554	-3.6	0.0985
4	-1.0	0	0.1	0.1577	5.2	0.098
5	10.0	0	0.3	0.0145	-7.0	0.05
6	-10.9	0.1	0.1	0.0667	9.2	0.2
7	12.0	0	0.3	0.0204	-8.1	0.05
8	19.7	0.3	0.5	0.3524	10.9	0.1
9	18.1	0.1	0.5	0.0043	-13.9	0.05
10	-20.0	0	0.5	0.1445	14.7	0
11	2.8	0.1	0.6	0.4365	-16.8	0
12	23.9	0.2	0.6	0.1611	19.6	0
13	29.1	0	0.7	0.0377	-21.3	0.05
14	-20.2	0.2	0.7	0.2598	23.4	0

La mediana obtenida del error absoluto mostrado en la tabla Nº 3.7 es de 0.0193 y la correspondiente varianza es 0.0127

Tabla Nº 3.8 Resultados al estimar la rotación con el parámetro de precisión de 0.01

Imagen	Rotación 1	error absoluto	Rotación 2	error absoluto	Rotación 3	error absoluto
1	3.98	0.02	-0.01	0.0454	0.47	0.01
2	-2.0	0	-0.01	0.0954	2.25	0.011
3	4.98	0.02	-0.01	0.0146	-3.67	0.0115
4	-1.00	0	-0.01	0.0323	5.01	0.012
5	10.03	0.03	0.24	0.0045	-7.01	0.01
6	-10.98	0.02	0.0	0.0067	9.14	0.3
7	12.03	0.03	0.25	0.0304	-8.11	0.1
8	19.63	0.37	0.47	0.4024	10.85	0.17
9	17.97	0.03	0.45	0.0343	-13.93	0
10	-19.95	0.05	0.47	0.0155	14.86	0.03
11	2.74	0.26	0.54	0.3265	-16.91	0.01
12	23.94	0.16	0.56	0.3711	19.39	0.04
13	29.05	0.05	0.66	0.0423	-21.38	0.01
14	-20.20	0.20	0.1	0.0598	23.60	0.50

La mediana obtenida del error absoluto mostrado en la tabla Nº 3.8 es de 0.0302 y la correspondiente varianza es 0.0351. Comparando los resultados, hay un límite en la precisión, con 0.1 la varianza es menor lo cual lo hace un valor más estable.

A continuación se realiza la prueba sintética para estimar el rango de valores apropiados de Dmin y Dmax, variando una y manteniendo la otra constante. Se genera una imagen rotando una imagen de referencia según la tabla Nº 3.9.

Tabla Nº 3.9 Valores establecidos para la rotación entre 4 imágenes

Imagen	Rotación (1vuelta/360)
1(referencia)	0
2	4.000
3	18.500
4	28.500

Las siguientes tablas muestran los resultados de la prueba para cada caso.

Tabla Nº 3.10 Promedio del error absoluto al variar Dmax

Número de Imágenes	Dmin	Dmax	Imagen 2	Imagen 3	Imagen 4	Promedio del error absoluto
4	0.06	1.0	4.2000	18.5000	29.5000	0.4
4	0.05	0.9	3.8000	18.5000	28.9000	0.2
4	0.06	0.8	3.7000	18.5000	28.5000	0.1
4	0.06	0.7	3.7000	18.5000	28.3000	0.17
4	0.06	0.6	3.5000	18.5000	28.3000	0.23
4	0.06	0.5	3.4000	18.5000	28.2000	0.3
4	0.06	0.4	3.4000	18.5000	28.3000	0.27
4	0.06	0.3	0.1000	19.6000	28.5000	1.67
4	0.06	0.2	0.1000	19.7000	27.0000	2.2
4	0.06	0.1	0.0000	0.0000	0.0000	17

De la tabla la máxima precisión obtenida, se da para Dmax = 0.8

Tabla Nº 3.11 Promedio del error absoluto al variar el Dmin

Número de Imágenes	Dmin	Dmax	Imagen 2	Imagen 3	Imagen 4	Promedio del error absoluto
4	0.03	0.6	0.1000	18.5000	26.7000	1.9
4	0.06	0.6	3.5000	18.5000	28.3000	0.23
4	0.09	0.6	4.0000	18.6000	28.5000	0.03
4	0.12	0.6	4.0000	18.6000	28.5000	0.03
4	0.15	0.6	4.1000	18.5000	28.5000	0.03
4	0.18	0.6	4.1000	18.6000	28.5000	0.07
4	0.21	0.6	4.1000	18.6000	28.6000	0.1
4	0.24	0.6	4.1000	18.6000	28.6000	0.1
4	0.27	0.6	4.1000	18.6000	28.6000	0.1

De la tabla la máxima precisión obtenida, se da para Dmin = 0.21, 0.24 ó 0.27

3.2.4 Efecto del Número de Imágenes del Registro Sobre la Súper Resolución

Para determinar en qué medida influye el número de imágenes de entrada sobre el valor del PSNR, se realiza una prueba sintética. La tabla Nº 3.12 muestra los resultados.

Tabla Nº 3.12 PSNR al aumentar el número de imágenes de entrada

Numero de Imágenes	Original y Súper Resolución
2	PSNR: 29.503356
3	PSNR: 29.315365
4	PSNR: 29.304956
6	PSNR: 29.432950
8	PSNR: 29.567572
10	PSNR: 29.615685
12	PSNR: 29.504682
16	PSNR: 29.675585
32	PSNR: 29.705716
48	PSNR: 29.754699
56	PSNR: 29.658857
64	PSNR: 29.700735
122	PSNR: 29.853493

De los resultados se obtiene la siguiente figura 3.1.

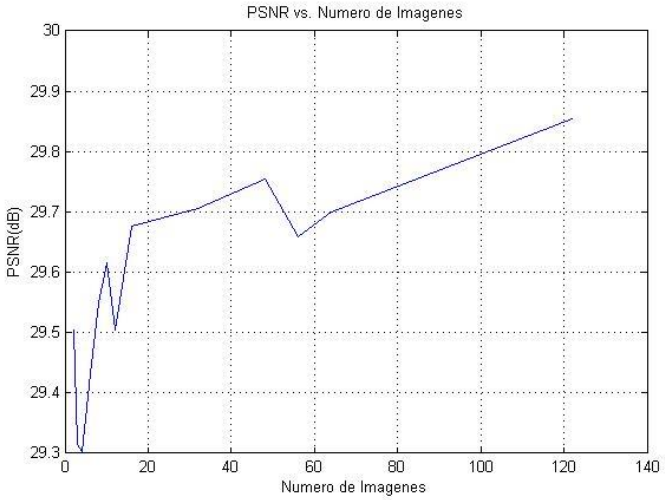

Figura 3.1 Variación del PSNR según el número de imágenes de entrada

Se observa un aumento del PSNR conforme aumenta el número de imágenes, en general esta tendencia se mantiene. Los resultados obtenidos al generar la imagen en súper resolución mejoran conforme el número de imágenes de entrada aumenta.

3.3 Límites de Procesamiento y Medición del Tiempo para Generar la Imagen en Súper Resolución

En esta sección se realizan pruebas con medición del tiempo de procesamiento,

haciendo una comparación de los tiempos obtenidos usando GPU y sin GPU. Se genera una imagen en súper resolución tomando como registro imágenes médicas como: ecografía, rayos X, tomografía.

El hardware empleado para las siguientes pruebas es: Intel Core i7 con una GPU marca NVIDIA GTX 560M de 196 núcleos.

Los parámetros de entrada al algoritmo, basado en los resultados obtenidos en la secciónes 3.1 y 3.2 son:

- Ventana Tukey = 0.35
- Precisión de Desplazamiento = 25
- Precisión de Rotación = 0.1
- Dmax = 0.8 , Dmin = 0.21
- Numero de imágenes = 4

Esta prueba se aplica a una ecografía cuyas dimensiones son de 786 x 554 pixeles. Con 8 iteraciones.

Tabla Nº 3.13 Tiempo de Procesamiento del algoritmo IBP

Factor de Incremento	IBP	
	CPU Tiempo(seg)	GPU Tiempo(seg)
2	19.001	0.626
3	24.955	1.258
4	31.955	2.096
5	54.743	3.196
6	73.538	4.502
7	98.956	6.091
8	125.466	7.902
9	160.025	9.941
10	191.565	12.173
11	Memoria Insuficiente	Memoria Insuficiente

De los resultados en la tabla Nº 3.13, se observa que hay un límite en el factor de incremento con CPU, por insuficiencia de memoria, esto debido a que el algoritmo fue implementado en una plataforma de 32 bits. Solo se tiene disponible 2^{32} direcciones de memoria, se puede ampliar el manejo de datos si la plataforma es de 64 bits, pues se tendrá disponible 2^{64} direcciones de memoria. Finalmente, la siguiente figura 3.2 confirma las diferencias del tiempo obtenido entre el CPU y el GPU.

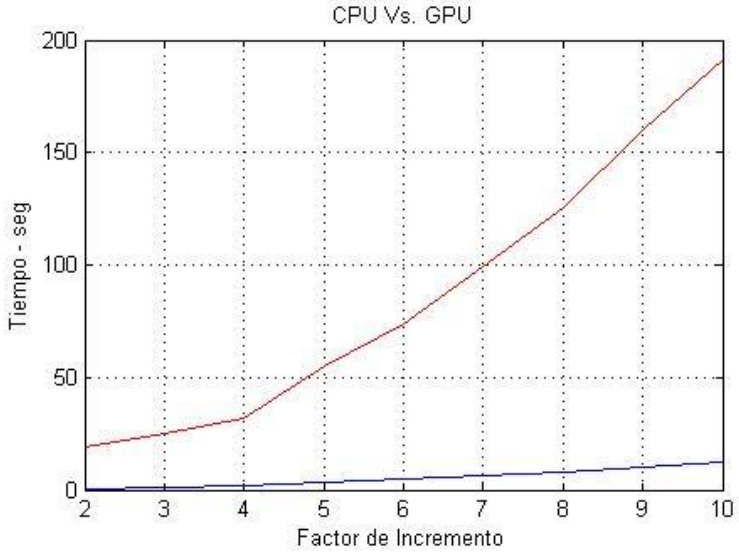

Figura 3.2 Tiempo de procesamiento IBP al procesar con una CPU y una GPU

De la figura 3.2 se observa que hay una reducción en el tiempo de procesamiento del algoritmo usando la GPU. Esta diferencia se incrementa con el aumento del factor de incremento, con mayor data procesada. Con un factor de incremento de 10, la velocidad de procesamiento del GPU es de $\frac{191.565}{12.173} = 15.73$ más veloz que con la CPU.

A continuación se presentan el resultado que se obtiene al medir el tiempo de procesamiento promedio, correspondiente a la estimación de desplazamiento (mediante la CPU y la GPU). Para determinar el tiempo de procesamiento promedio, se ejecuta 100 veces el algoritmo y se mide el tiempo, luego se promedia. Las condiciones de la prueba son: 4 imágenes de 786*554 pixeles.

Tabla Nº 3.14 Tiempo de Procesamiento para la estimación de desplazamiento

Total de data procesada	Estimación de Desplazamiento	
	CPU – mseg	GPU – mseg
428640	759.0	86.4
857280	2387	96.9
1714560	3816.7	150.0
3429120	6417.0	191.0
6858240	6718.3	260.6
13716480	10551.0	341.9
27432960	13796.0	468.6
54865920	18288.3	483.5
109731840	26093.7	593.0
219463680	28308.7	790.6

Los resultados en la tabla Nº 3.14 se obtiene la figura 3.3, claramente se observa la gran diferencia del tiempo de procesamiento entre ambas tecnologías. Por ejemplo para

42.864 millones de datos procesados, la GPU es de $\frac{28308.6667}{2371} = 11.939$ más veloz que con la CPU.

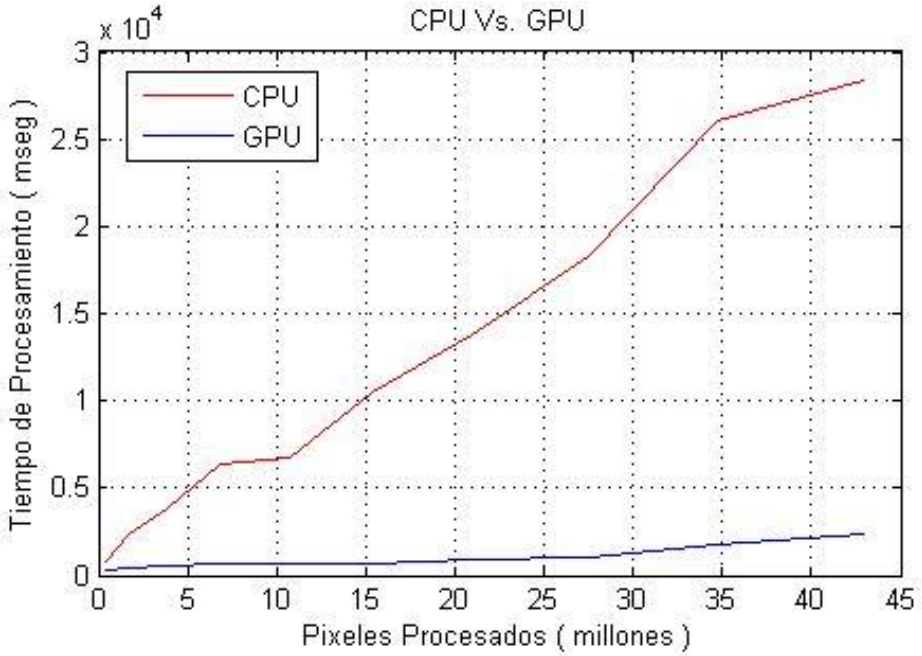

Figura 3.3 Tiempo de procesamiento en la estimación de desplazamiento

Análogo al caso de estimación de desplazamiento se mide 3 veces el tiempo de procesamiento y se considera el promedio para poder hacer su respectiva gráfica.

Tabla Nº 3.15 Resultado del tiempo de procesamiento en la estimación de rotación

Total de data procesada	Estimación de Rotación	
	CPU – mseg	GPU – mseg
428'640	764.3333333	171
1'714'560	2386.666667	499
3'857'760	3973	1237
6'858'240	9037.666667	1390
10'716'000	14055.66667	3743
15'431'040	16910.33333	4203
21'003'360	15859.66667	5436
27'432'960	18263.66667	6300
34'719'840	20805	12701
42'864'000	39099.33333	15684

De los resultados obtenidos en la tabla Nº 3.15 se obtiene la figura 3.3, la diferencia en los tiempos de procesamiento tiende a aumentar conforme se procesa mas datos.

Para 42.864 millones de datos, la GPU es $\frac{39099.33333}{11372.66667} = 3.438$ más veloz comparado al obtenido mediante la CPU.

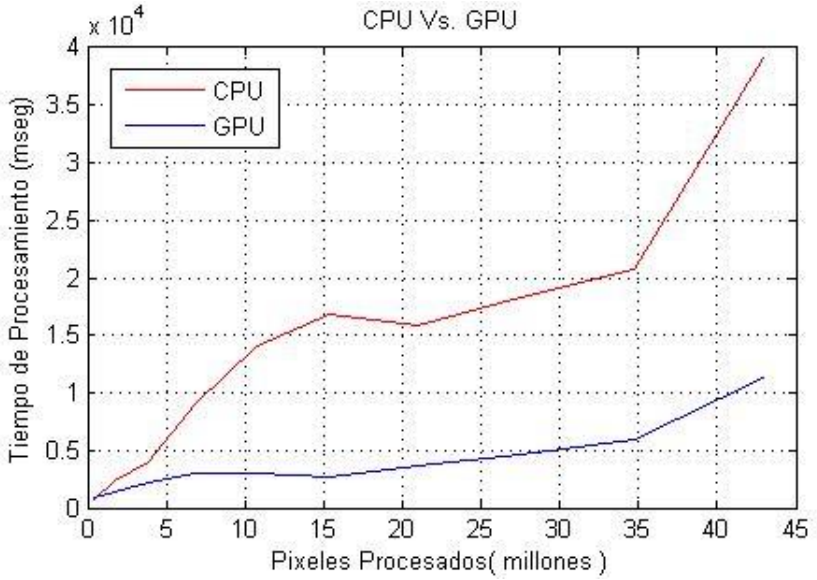

Figura 3.4 Tiempo de procesamiento en la estimación de rotación

Se observa que con la GPU el factor de velocidad superior a la CPU no es una constante, varía en: la estimación de desplazamiento, estimación de rotación e IBP. Esto debido a que solo una parte del algoritmo completo ha sido programado para ejecutarse con procesamiento en paralelo. Con la estimación de desplazamiento el grado de paralización es mayor a los demás algoritmos por lo cual, en este caso, la reducción del tiempo de procesamiento es mayor.

CAPÍTULO IV
RESULTADOS

4.1 Resultados Obtenidos Mediante Pruebas Sintéticas

La forma objetiva de medir la calidad de una imagen como se mencionó en la sección 1.2, es calculando el MSE y PSNR entre dos imágenes. Este método requiere de una imagen original de referencia, por ello se realizan pruebas sintética que generan las condiciones para calcular el PSNR y comparar resultados.

La prueba sintética para este caso consiste en generar un registro imágenes en baja resolución desde una imagen original (referencia) en alta resolución. En base a la ecuación (1.6) cada imagen generada se distorsiona de forma controlada, agregando el ruido y el efecto del desenfoque mediante la convolución de la imagen con el PSF definido y conocido. Luego, a partir de este registro de imágenes, se genera una imagen en súper resolución con el algoritmo propuesto.

Se elige también, en forma aleatoria una imagen del registro para aplicarle interpolación bicúbica

En la figura 4.1 se muestran: la imagen original, la imagen obtenida aplicando interpolación bicúbica y las imágenes en súper resolución obtenida desde un registro de 8 y 16 imágenes respectivamente.

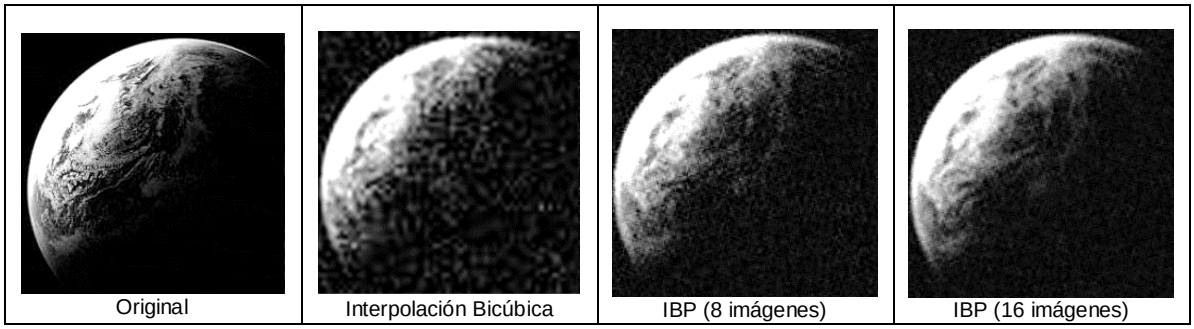

Figura 4.1 Resultado del Algoritmo IBP comparado a la Interpolación Bicúbica

Basado en la ecuación (1.4) se calcula el PSNR entre la imagen original y el obtenido por interpolación bicubica y entre la imagen original con el generado en súper resolución a partir de 8 y 16 imágenes en baja resolución respectivamente. El resultado se muestra en la tabla Nº 4.1.

Tabla Nº 4.1 PSNR obtenido según la ecuación (1.4)

	Interpolación Bicúbica	**IBP (8 imágenes de entrada)**	**IBP (16 imágenes de entrada)**
PSNR	34.242220dB	35.301864dB	36.615494dB

La tabla Nº 4.1 muestra que la imagen en súper resolución se asemeja más al original en comparación con la interpolación bicúbica, esta afirmación puede reforzarse también visualmente por dos factores, disminución del ruido y mayor detalle observable. Se observa también, que con 16 imágenes de entrada la imagen en súper resolución tiene menos ruido en comparación al obtenido usando 8 imágenes de entrada.

En la siguiente figura 4.2 se muestra el resultado de aplicar la misma prueba sintética considerando como imagen de referencia, una imagen de mamografía.

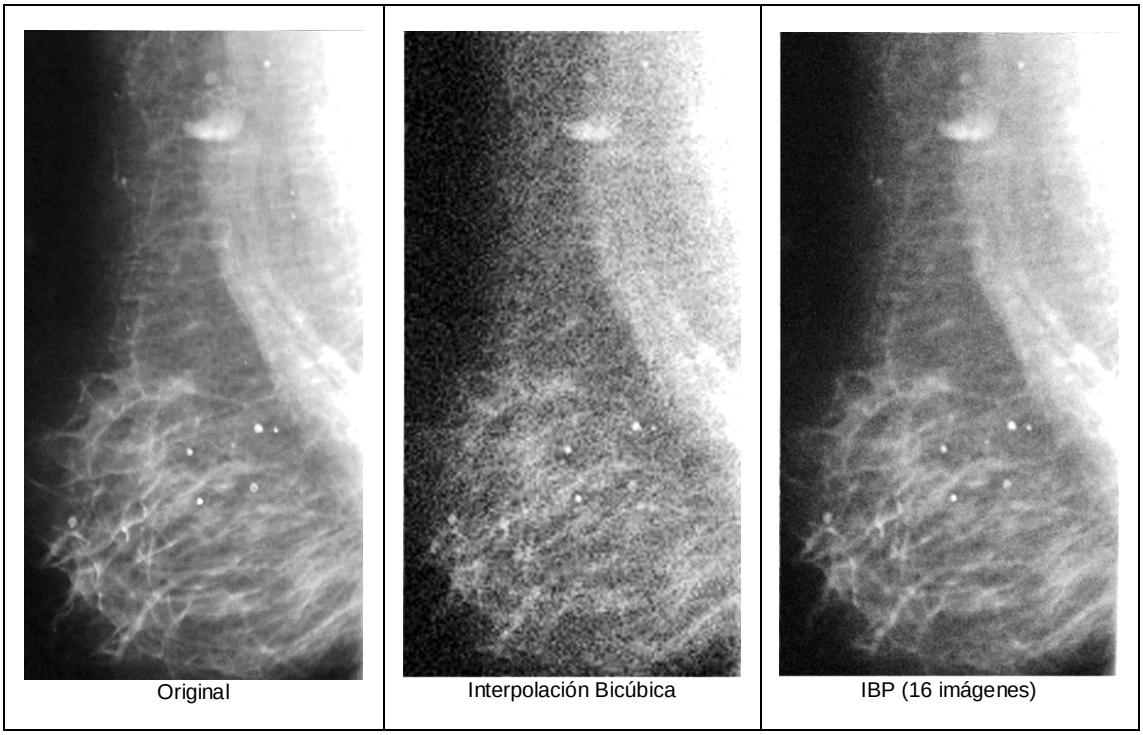

Figura 4.2 Imagen en súper resolución comparado a la interpolación bicúbica

El PSNR respectivo se muestra en la siguiente tabla Nº 4.2

Tabla Nº 4.2 PSNR obtenidas con la interpolación bicúbica y el algoritmo IBP

	Interpolación Bicúbica	**IBP (16 imágenes de entrada)**
PSNR	31.267225dB	34.898541dB

El cálculo PSNR confirma que la imagen en súper resolución se acerca más al original.

De la figura 4.2 el resultado obtenido con la IBP se asemeja más a la imagen original por tener menor ruido y resaltado los puntos blancos que es resultado de la acumulación de micro calcificaciones en la seno del paciente.

Respecto de las micro calcificaciones, con la imagen obtenida mediante la interpolación bicúbica, es más dificultoso detectarlos visualmente debido al ruido en la imagen, esto dificulta al médico a realizar un diagnóstico seguro y preciso. Sin embargo con la imagen en súper resolución, la reducción del ruido con mayor seguridad facilita la detección de las micro calcificaciones. Es importante mencionar que la imagen en súper resolución no muestra puntos blancos que no están en la imagen original lo cual demuestra que el método no produce falsos negativos, es decir no distorsiona la imagen.

Este resultado fue obtenido con una prueba sintética, donde se tiene conocimiento del modelo aplicado para degradar la imagen original (ruido, PSF) y generar las imágenes en baja resolución.

Una prueba real sobre imágenes de mamografía restringe el hecho de que solo se dispone de una imagen como entrada, pues para generar una imagen en súper r muestre nueva información requiere tener al menos de 2 imágenes de entrada, es un factor limitante debido a la tecnología usada para obtener una imagen de mamografía.

4.2 Resultados Sobre Imágenes Médicas

En la sección 4.1 mediante una prueba sintética se tenía una imagen original y se asumía como una imagen de referencia ideal sin distorsiones, esto para poder comparar los métodos mediante el cálculo PSNR, una comparación objetiva de la calidad de las imágenes obtenidas.

En una situación real la situación no es la misma, no se dispone de la imagen de referencia, como se mencionó en la sección (2.1) la evaluación y comparación de la calidad de una imagen en sistemas reales, es en gran parte subjetiva y depende de la aplicación o el nivel de detalle que se requiere para su análisis. Se cuenta con algunas herramientas como la resolución espacial, profundidad y el ruido de la imagen, herramientas que nos permiten establecer un nivel de calidad de la imagen.

El siguiente resultado mostrado se obtiene al generar una imagen en súper resolución a un examen real de ecografía. La imagen fue generada a partir de 4 imágenes de entrada, obtenidas desde un video (en formato .avi) grabado por el ecógrafo de un hospital aplicado a un paciente.

Para fines de comparación, en la figura 4.1 se muestra una sección arbitraria ampliada de la ecografía. En la parte izquierda esta una de las 4 imágenes originales de ecografía y a la derecha la imagen en súper resolución.

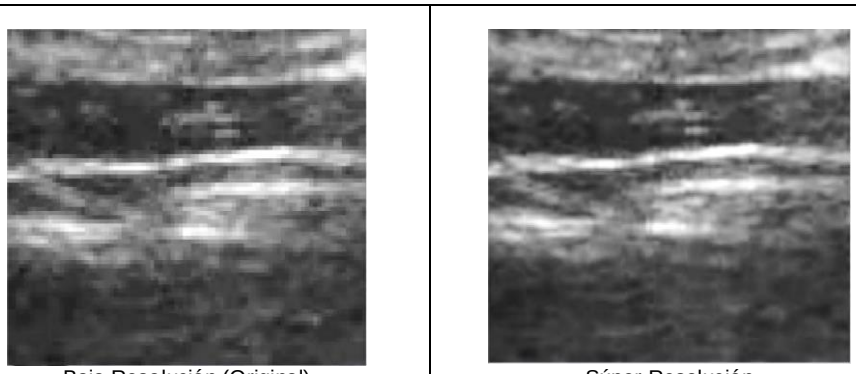

Figura 4.3 Super resolucion aplicado a la imagen de ecografia

Claramente con la imagen en súper resolución se observa una mejora en la resolución espacial, principalmente puede observarse más detalle, lo que permite extraer más información con respecto de la imagen original.

Este resultado es independiente de la tecnología empleada para obtener la imagen original, esta característica hace que la súper resolución sea aplicable a la última tecnología empleada para hacer un examen médico de ecografía.

El tiempo procesamiento total fue de 953 mseg, con una GPU NVIDIA GTX 560M de 192 núcleos.

TOMOGRAFÍA

A continuación se muestra el resultado obtenido en un examen de tomografía. La figura 4.4 muestra el corte transversal en la columna vertebral de una persona. En la parte izquierda, la imagen original y en la parte derecha, la imagen en súper resolución.

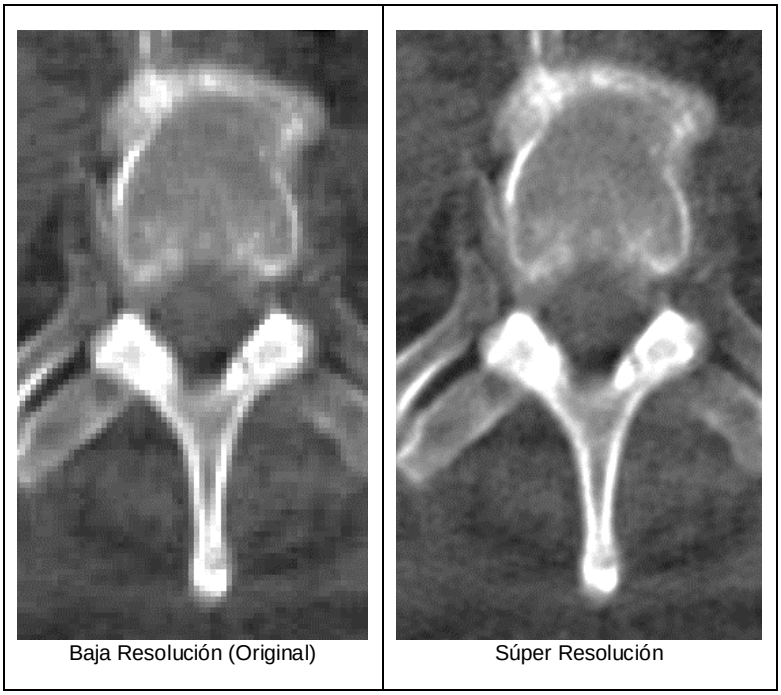

Figura 4.4 Resultado del algoritmo aplicado a una tomografia

Como se observa la imagen de tomografía posee menor ruido con respecto a una de ecografía. En un análisis minucioso entre las dos, puede apreciarse mejoras no solo con respecto a una mayor resolución espacial sino también, es una imagen con mayor información.

Si bien la imagen de tomografía está en formato DICOM, para fines de demostración en la presente tesis, se convierte a formato jpg lo cual hace que la profundidad de bits disminuya. Esto no hace perder el hecho de que la imagen en súper resolución es mejor, sino hace posible que en formato DICOM la imagen en súper resolución sea más aún de mejor calidad.

No fue posible obtener buenos resultados para imágenes médicas de radiografía y mamografía debido a la falta de un buen modelo matemático del PSF.

CONCLUSIONES Y RECOMENDACIONES

1. El algoritmo implementado que genera una imagen en súper resolución usando procesamiento en paralelo, mejora la imagen médica de ecografía y tomografía. Al basarse de un método que requiere múltiples imágenes de entrada, es factible su aplicación a las imágenes médicas de ecografía. También se puede aplicar como un método que reduce el desenfoque de imágenes médicas como: la tomografía, mamografía y rayos X, que tecnológicamente solo se dispone de una imagen de entrada.
2. Se redujo el tiempo de procesamiento del algoritmo, con la arquitectura hardware GPU NVIDIA GTX 560M de 192 núcleos. En una prueba de ecografía, para un incremento en un factor de 2, el tiempo de procesamiento se redujo de 20 segundos (CPU) a un tiempo de procesamiento de 651 mseg. La eficiencia en la reducción del tiempo depende del número de procesadores o núcleos en la GPU. Desde el punto de vista tecnológico, el tiempo puede reducirse aún más empleando una GPU con una mayor cantidad de núcleos y/o usando múltiples unidades de GPU en una sola PC, la última tarjeta de video NVIDIA en el mercado posee 3000 núcleos.
3. El PSF es el parámetro más importante y crítico que determina la calidad de la imagen en súper resolución generada. Mientras el modelo estimado se aproxime más al PSF real del sistema, la imagen obtenida será de mayor calidad. Mientras se cuente con más imágenes de una escena como entrada del algoritmo, la imagen obtenida en súper resolución será de mayor calidad, Estas imágenes de entrada deben obtenerse considerando, un desplazamiento máximo entre ellas del 1% de sus dimensiones, con una rotación entre ellas que sea mayor a -30º y menor a 30º y los objetos en la escena no pueden moverse respecto de los demás objetos y la cámara.
4. Las estimaciones de los parámetros de desplazamiento se obtienen con una precisión de 1 pixel. Análogamente, las estimaciones en los parámetros de rotación se obtienen con una precisión no menor a 0.7º.
5. La técnica propuesta de usar el filtro bilateral reduce el fenómeno de anillos en el resultado, manteniendo el contenido de las altas frecuencias de las imágenes originales en baja resolución.

6. Para continuar con la investigación iniciada, con el fin de optimizar los resultados. Se recomienda aplicar un algoritmo que permita estimar el PSF. Existen varios enfoques para la estimación, muchos de ellos basados en métodos estadísticos y otras basados en pruebas específicas sobre el sistema de adquisición de imágenes para asi, obtener un PSF real propio del sistema. Un buen método encontrado, basado en la derivada sobre las imágenes y que puede añadirse al algoritmo implementado, se encuentra en el artículo “PSF Estimation Using Sharp Edge Predictor” (26), el algoritmo que emplea se basa en la información de los bordes de cada objeto dentro de una imagen.
7. Se recomienda aplicar el concepto de súper resolución en otras áreas de desarrollo y de investigación como: imágenes satelitales, imágenes obtenidas desde un aeroplano, seguridad (cámaras de seguridad, identificación de rostros, detección de placas de automóviles, etc).
8. Mediante el concepto de voxel se puede generar una vista en 3d, a partir de un examen completo de tomografía. El voxel es una ampliación a 3 dimensiones del concepto de pixel. En el plano xyz, para cada valor de z se tiene un conjunto de voxeles ubicados en un plano, se puede aplicar súper resolución en cada plano, mejorando el resultado del objeto 3d generado. Se sugiere investigar en esta área, aplicando el concepto de segmentación, que en medicina se usa principalmente para identificar órganos específicos del cuerpo humano desde un conjunto de datos

ANEXO A
FORMATO DE IMÁGENES

La visualización de una imagen digital requiere cargar en la memoria la imagen desde un archivo. El formato digital permite guardar en memoria, archivarlo, mover su ubicación.
En la actualidad se dispone de una variedad de formatos estandarizados: Estos estándares han facilitado el intercambio de imágenes, así como también, permite que estas imágenes en su mayoría puedan ser leídos por un software. Existen algunos criterios a considerar que nos permite seleccionar el más adecuado formato de archivo.

- Tipo de Imagen. Determinar imágenes en blanco y negro, en escala de grises, escaneados para documentos, imágenes a color, imágenes cuyos valores están en punto flotante. En aplicaciones como imágenes satelitales, el máximo de tamaño es un factor importante.
- Tamaño de almacenamiento y compresión. Considerar cuanta memoria ocupa una imagen y el método de compresión a usar sin olvidar las perdidas por compresión
- Compatibilidad. Establecer cuán importante es el intercambio de la imagen
- Aplicación. Determinar en qué ámbito será principalmente usado la imagen, si es para imprimirse, aplicación web, película, medicina, astronomía, etc.

A.1 JPEG

Es un estándar que define un método de compresión para imágenes en escala de grises e imágenes a color. Fue desarrollado por la Joint Photographic Experts Group (JPEG) con el objetivo de reducir en promedio la activación de los datos en un factor de 1:16 y fue establecido en 1990 como un estándar ISO (IS-10918).
EL JPEG soporta imágenes con 256 colores por componente, es decir con una profundidad de 8 bits/pixel.
Sus desventajas al emplear solo 8 bits, el rendimiento es pobre.

A.2 PNG

Originalmente desarrollado como reemplazo del formato GIF. Fue diseñado como un formato universal especialmente para usarse en el internet, soporta 3 diferentes tipos de imágenes.

- Color verdadero (hasta 3 x 16bits/pixel)
- Escala de grises (hasta 16 bits/pixel)
- Indexado (hasta 256 colores)

A.3 TIFF

Está diseñado para satisfacer las necesidades de los profesionales en diversos campos. El formato soporta un amplio rango de escala de grises, imágenes en true color, también imágenes con una gran profundidad de bits y soporte para elementos en punto flotante.

El formato TIFF usa varios métodos de compresión (LZW, ZIP, CCITT y JPEG), como también varios espacios de colores. Mediante este formato es posible almacenar una cantidad de variaciones de una imagen en tamaños diferentes juntos en una sola imagen.

A.4 BMP

El formato Bitmap (BMP), soporta escala de grises, indexado e imágenes con color verdadero. Soporta imágenes en binario, pero no de una manera eficiente ya que cada pixel se almacena usando un byte completo. BMP soporta imágenes de almacenamiento en un rango en un rango igual al del TIFF, pero es un formato menos flexible.

A.5 DICOM – Digital Imaging and Communications in Medicine

Representa el más universal y fundamental estándar en imágenes médicas. Como tal, proporciona todas las herramientas necesarias para el diagnóstico preciso y el tratamiento de imágenes médicas. Por otra parte, contrario a la creencia popular, DICOM es no solo una imagen o un formato de archivo, engloban transferencia de datos, almacenamiento, protocolos de visualización y diseñado para cubrir todos los aspectos funcionales de la imagen medica digital.

- Las imágenes basados en DICOM soportan 65536 (16 bits) escalas de grises, entonces se puede capturar las más mínimas matices en comparación con una imagen JPEGs o Bitmaps (BMP) que están limitados a 256 niveles.
- Mediante el DICOM no solo es posible almacenar imágenes, permite el almacenamiento de parámetros relacionados con el paciente, posición relativa en 3D, tamaño físico de los objetos en la imagen, espesor de los cortes, parámetros de exposición, etc. Esta información adicional enriquecen el contenido de una imagen DICOM y facilita el procesamiento e interpretación de los datos en la imagen por diversos métodos (ejemplo creación de imágenes en 3D, segmentación).
- Los archivos DICOM usan más de 2000 atributos estandarizados para transmitir diversos datos médicos desde el nombre del paciente hasta la profundidad de bits en la imagen. Estos datos son esenciales para un diagnóstico preciso.
- La claridad en la descripción de una imagen digital.

{COMO FUNCIONA

Como introducción al complejo entorno médico, DICOM usa su propio leguaje, basado en su modelo del mundo real.

Toda la data del mundo real (paciente, estudio llevado, dispositivo médico, etc.). Son vistos en DICOM como objetos con sus respectivas propiedades o atributos. La definición de estos objetos están estandarizados y denominados IODs (Information Object

Definitions). Puede también pensarse en los IODs como colecciones de atributos que describen cada dato particular del objeto, por ejemplo, Un paciente IOD puede describirse mediante el nombre, record medico numérico, sexo, edad, peso, si fuma o no, etc. Esto se muestra en la figura A.1.

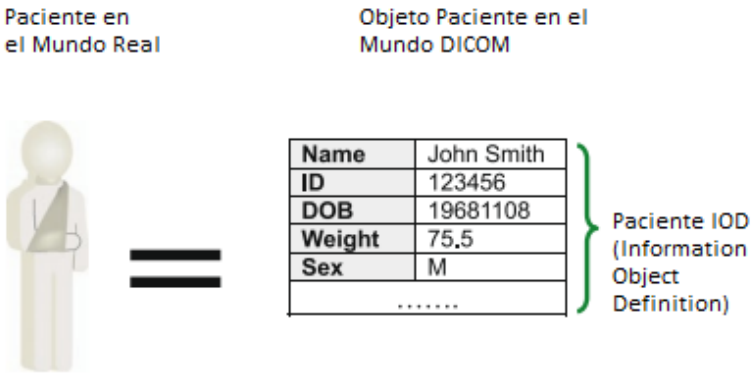

Figura A.1 Desde los datos reales a los DICOM IODs

ANEXO B

VISIÓN POR COMPUTADORA - OPENCV

OpenCV es una librería de fuente libre y sin restricciones. Contiene más de 500 algoritmos optimizados para el análisis de imágenes y video. Desde su introducción en 1999, fue sido ampliamente adoptado como una herramienta de desarrollo principal por la comunidad de investigadores y desarrolladores en visión por computadora.

La librería está escrito en C/C++ y corre bajo Linux, Windows, Mac OS X. Está en desarrollo para Python, Ruby, Matlab y otros lenguajes.

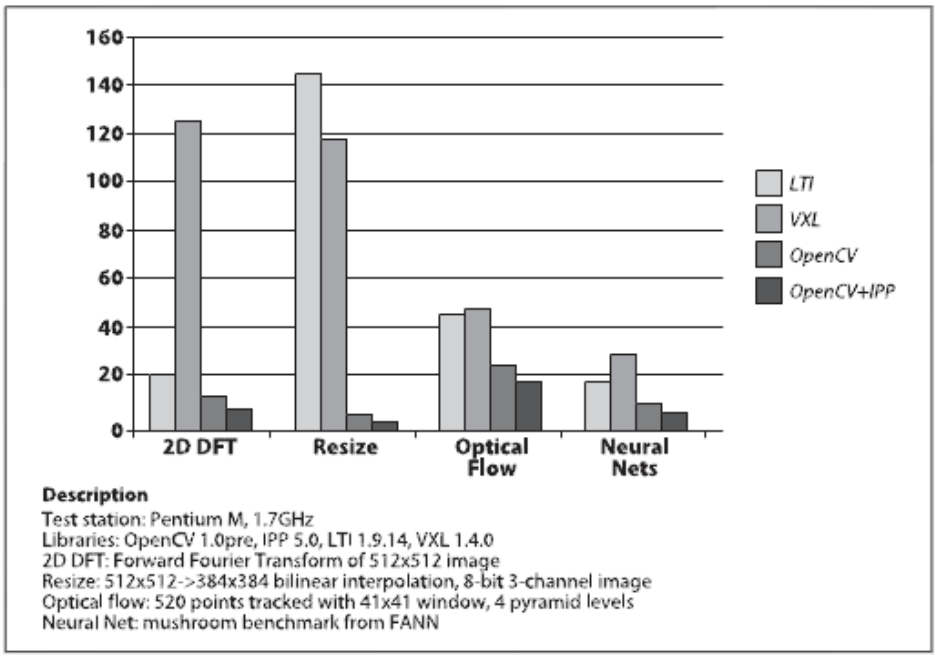

Figura B.1 Velocidad de Procesamiento de OpenCV comparado con productos similares

La librería OpenCV puede también aplicarse en proyecto basados en C# via EmguCV. Puede ser parte de un proyecto basado en Android.

En general la visión por computadora en los últimos años ha dado grandes avances, puede tener numerosas aplicaciones (ejemplo detección de rostros, detector de objetos, lector de códigos, súper resolución, etc.) que dependen del ingenio y creatividad del desarrollador.

ANEXO C

PROGRAMACIÓN EN PARALELO VIA OPENCL Y CUDA

C.1 OPENCL

Es un estándar para la programación en paralelo para sistemas homogéneos. Una aplicación basada en OpenCL ejecuta un conjunto de procesadores heterogéneos conformado básicamente de una CPU (huésped) y uno o más unidades GPU (dispositivos)

Arquitectura OpenCL

La arquitectura está definida en 3 partes o modelos.

- Modelo de plataforma
- Modelo de ejecución
- Modelo de memoria

C.1.1 Modelo de Plataforma

El modelo de plataforma define los roles del huésped y de los dispositivos, en forma concreta se elige que unidades hardware disponibles podrán usarse para el procesamiento en paralelo. En este modelo hay un simple host que coordina la ejecución en uno o más dispositivos.

El modelo de plataforma define un dispositivo como un conjunto de unidades computacionales, cada uno independiente del resto. Cada unidad computacional se divide en elementos de proceso.

Este modelo corresponde al modelo de hardware de algunos GPUs, por ejemplo para el AMD Radeon 6970 Graphics que posee 24 núcleos SIMD (unidades computacionales), cada núcleo SIMD contiene 16 líneas (elementos de proceso). Cada línea del SIMD ejecuta 4 extensas instrucciones (VLIW), en total ejecuta 1536 instrucciones al mismo tiempo. OpenCL usa la función clGetPlatformIDs() para descubrir el conjunto de plataformas disponibles en un sistema dado.

C.1.2 Modelo de Ejecución

Para poder trasladar los datos y comandos al dispositivo, es necesario definir un contexto para coordinar la forma de interacción entre el huésped y los dispositivos. Administra los objetos de memoria que están disponibles en el dispositivo y mantiene los programas y kernels que son creados para cada dispositivo.

La ejecución se basa en el concepto del kernel, un código que representa una simple ejecución y es similar a una función en C, acepta como entrada una lista de parámetros y posee variables locales, este simple kernel es conocido como un elemento de trabajo (work ítem). La diferencia de una función C es que el kernel define solo una tajada de un gran espacio de ejecución en paralelo.

C.1.3 Modelo de Memoria

En general los sub sistemas de memoria varían enormemente entre los fabricantes de hardware. Para soportar el código OpenCL define un modelo de memoria tal que los programadores lo tengan en cuenta. El modelo esta esquematizado según la siguiente figura C.1.

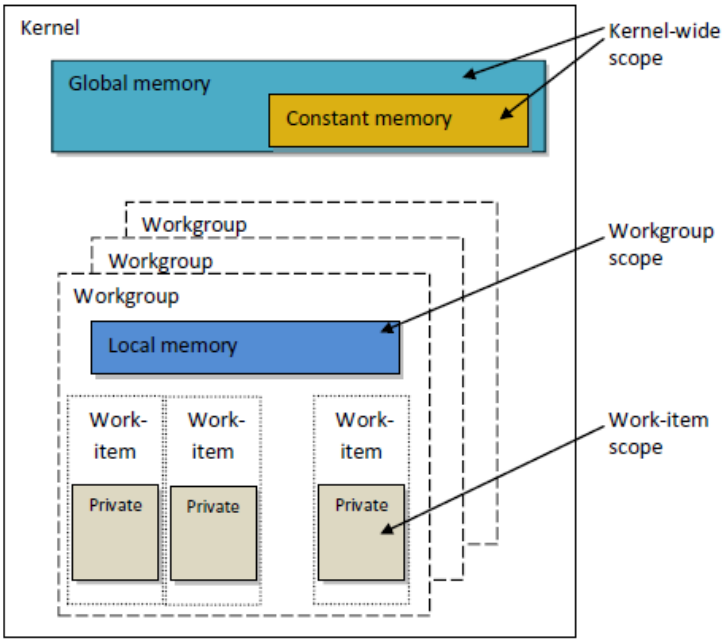

Figura C.1 Modelo de Memoria

La memoria global.- Es similar a la memoria principal del CPU (host), es visible siempre que se transmiten los datos del huésped al dispositivo, los datos residen en la memoria global. La palabra "__global" va junto a un puntero para especificar este tipo de memoria.
Memoria constante.- Está específicamente diseñado para datos de solo lectura, son datos cuyos valores son accedidos por todos los elementos de trabajo simultáneamente, son variables cuyos valores nunca cambian. La memoria constante es parte de la memoria global. Los objetos de memoria que son transferidos a la memoria global pueden especificarse como constantes mediante la palabra "__constant".
Memoria local.- Es una memoria rápida, cuya dirección es única en cada dispositivo y es compartido en un grupo de trabajo, por lo cual, el acceso es más rápido que la memoria global, con un ancho de banda más grande. Llamando a la función "clSetKernelArg()" con un tamaño, pero sin argumentos permite separar espacio en la memoria local especificado con la palabra "__local" junto a un puntero.
Memoria Privada.- Es una memoria única en cada elemento de trabajo. Las variables locales y los argumentos del kernel que no son punteros son privados por defecto. OpenCL encierra un modelo acorde a los GPUs modernos, por ejemplo la siguiente

figura detalla la relación entre el modelo de memoria y el encontrado en un AMD 6970 GPU. [18]

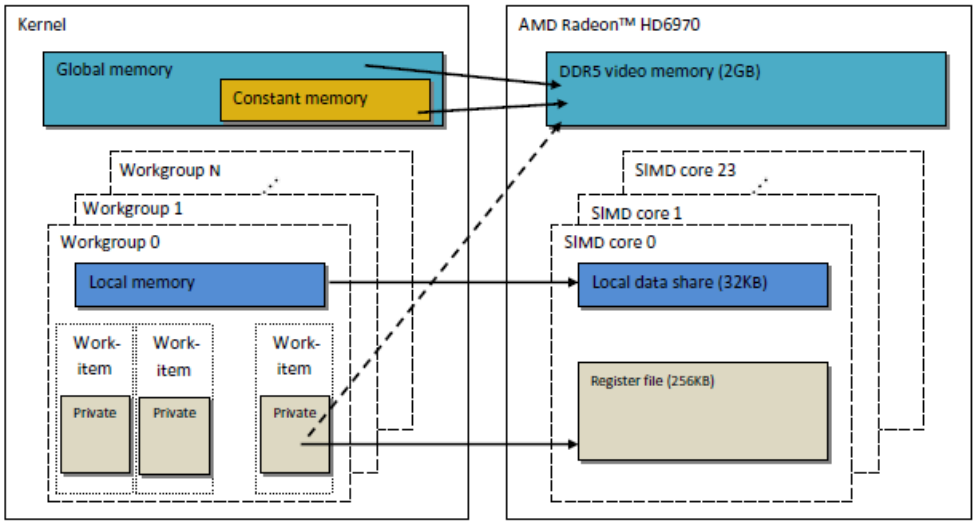

Figura C.2 Modelo de Memoria de OpenCL comparado con el modelo real de una AMD 6970

C.2 CUDA

La arquitectura unificada de dispositivos de cómputo (compute unified device architecture) es un modelo de programación en paralelo y una plataforma para la computación en paralelo. Permite el incremento de la computación en paralelo aprovechando el poder de la unidad de procesamiento grafico (GPU).

Mediante CUDA podemos enviar códigos basados en C/C++ y FORTRAN al GPU, no requiere lenguaje máquina.

En la actualidad se están desarrollando miles de aplicaciones y se han publicado artículos de investigaciones. Se puede obtener gratuitamente librerías matemáticas y herramientas para la construcción y optimización de sus aplicaciones, todo esto vía CUDA Toolkit. [27]

El modelo de programación en paralelo se divide en

- Modelo de Plataforma
- Modelo de ejecución
- Modelo de memoria

C.2.1 Modelo de Plataforma

La arquitectura CUDA está construida en base a un gran número de "Streaming Multiprocessor". Cuando el programa en CUDA ejecuta un kernel, los bloques de la malla son enumerados y distribuidos al microprocesador. Los hilos dentro de un bloque de hilos se ejecutan concurrentemente dentro de un multiprocesador y múltiples bloques de hilos pueden ejecutarse concurrentemente en un multiprocesador. Conforme se terminan

de procesar los bloques de hilos, se ejecutan nuevos bloques de hilos en los multiprocesadores que quedan vacantes.

Un multiprocesador está diseñado para ejecutar miles de hilos. Para manejar efectivamente la cantidad de hilos a ejecutar CUDA emplea la arquitectura SIMT (Simple instrucción múltiples hilos). La arquitectura SIMT es un multi procesador que crea, administra y ejecuta hilos en grupos de 32 llamados warps.

C.2.2 Modelo de Ejecución

CUDA define funciones basados en C, permite a los programadores definir funciones llamados kernels, la diferencia con las funciones clásicas en C es que cuando son llamados, ejecutan N operaciones al mismo tiempo en N hilos diferentes.

Un kernel en CUDA se define con la palabra "__global__" y con el número de hilos a ejecutar mediante la notación "<<<…>>>". Cada hilo que es ejecutado por el kernel se identifica mediante un hilo ID único que es accesible en el kernel mediante la palabra "threadIdx"

El threadIdx es un vector de 3 componentes, estos pueden identificarse mediante índices de hilos en una, dos o tres dimensiones, agrupándose en bloques de hilos de uno, dos o tres dimensiones. Existe un límite con el número de hilos por bloque, se tiene en cuenta que todos los hilos de un bloque están en un mismo núcleo de procesador y comparten los recursos de memoria limitada.

Un kernel puede ejecutarse vía múltiples bloques de igual tamaño, tal que el número total de hilos es igual al número de hilos por bloque multiplicado por el número de bloques. Los bloques están organizados en arreglos de una o dos dimensiones, tal como se muestra en la siguiente figura C.3.

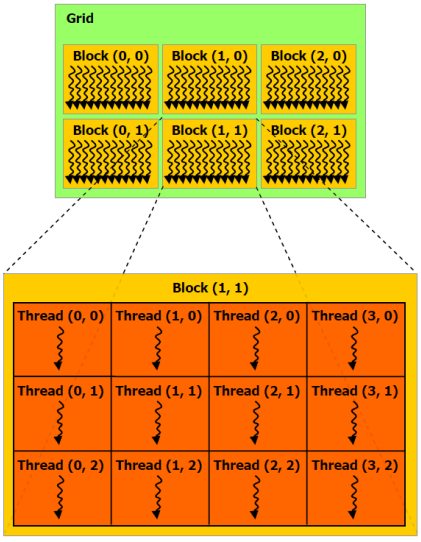

Figura C.3 Esquema del Modelo de Programación

El número de hilos por bloque y el número de bloques se especifican dentro de la notación "<<<número de bloques, hilos por bloque>>>".

C.2.3 Modelo de Memoria

Los hilos acceden a los datos desde múltiples espacios de memoria durante su ejecución, cada hilo posee una memoria local, cada bloque de hilos comparten una memoria visible entre todos. Todos los hilos tienen acceso a la misma memoria global. Esquemáticamente se muestra la siguiente figura C.5.

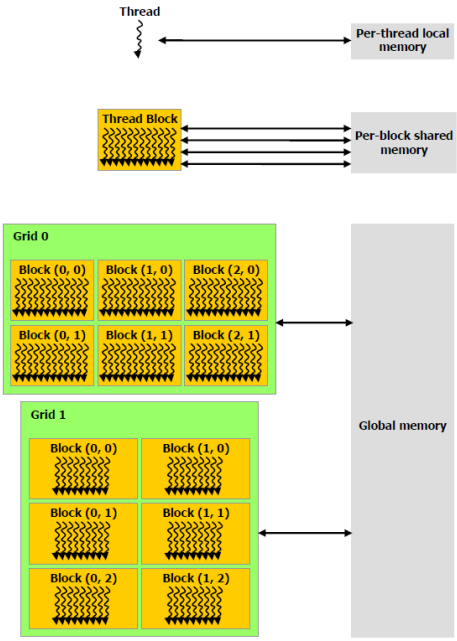

Figura C.4 Modelo de Memoria

Ejemplo

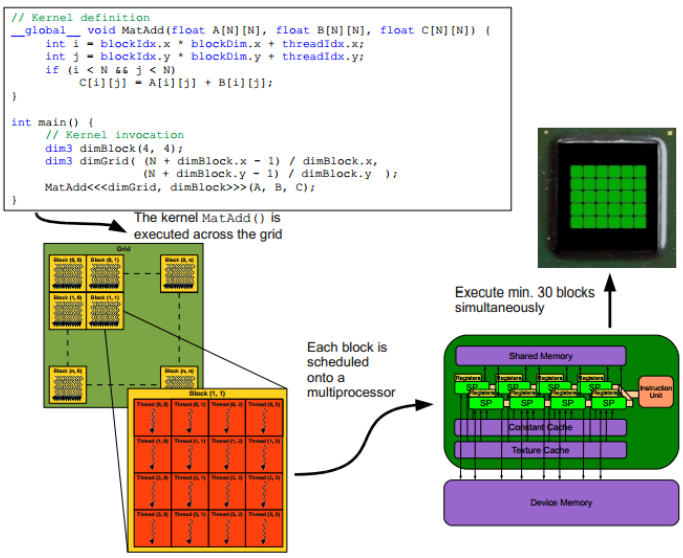

Figura C.5 Ejemplo de programación y como se ejecuta en la GPU

ANEXO D

KERNELS – CÓDIGO DE LOS PROGRAMAS EJECUTADOS SOBRE UNA GPU

D.1 KERNEL – INTERPOLACIÓN BICÚBICA

```
//valor interpolado en un punto de la señal continua\n"
float cubic(float *x)
{
       //considerando a = 1/3 y b = 1/3\n"
  if( (*x) < 0 ) (*x) = -(*x);
  float z = 0;
  const float p1 = *x;
  const float p2 = p1*(*x);
  const float p3 = p2*(*x);
  if( (*x) < 1.f)
         z = (21.f/18.f)*p3 - (2.f)*p2 +16.f/18.f;
  else if ( (*x) < 2.f )

      z = (-7.f/18.f)*p3 + (2.f)*p2 - (10.f/3.f)*p1+(32.f/18.f);
  return z;

}
  //funcion para interpolar
__kernel
void rescubic(
    __global float *p_in,
      const int p_Wold,
      const int p_Hold,
      const int p_Wnew,
      const int p_Hnew,
      const float p_dx,
      const float p_dy,
    __global float *p_out)
{
 // Definiendo los indices
 const int x = get_global_id(0);
 const int y = get_global_id(1);
 const int oldY = (float)y*p_dy;
 const int oldX = (float)x*p_dx;
 const float ty = p_dy*(float)y ;
 const float tx = p_dx*(float)x ;
 //interpolamos
       //obteniendo los valores del pixel en la fuente mas cerca

       float q = 0;   //q es el valor interpolado
       //x y y son coordenadas de la fuente
       //el objetivo es up vp
    float vpp;
       float upp;
       const int oldhh = p_Hold-1; const int oldww = p_Wold-1;
       int vp,up;
       int vptt;
       for(int j = -2; j < 2; j++ )
       {
       vp = oldY + j;
              vpp = ty -(float)vp;
              if( vp < 0 ) vp = 0;
              if( vp > oldhh ) vp = oldhh;
              vptt = vp * p_Wold;
              float pp = 0;
              for(int i = -2; i < 2; i++)
              {
                     up = oldX + i;
                     upp = tx - (float)up;
                     if(up < 0) up = 0;
                     if(up > oldww) up = oldww;

                     pp += p_in[ vptt + up ] * cubic(&upp);
              }
              q += pp * cubic(&vpp);
       }


 //copiamos a la salida
 p_out[y * p_Wnew + x] = q;
 }
```

D.2 KERNEL – INTERPOLACIÓN LANCZOS4

```
//valor interpolado en un punto de la señal continua\n"
float lanczos4(float *x)
{
       //Interpolacion de lanczos 2
  const float pii = 3.14159265358979323846f;
  if( (*x) < 0 ) (*x) = -(*x);
  float z = 0;
  const float tt1 = pii*(*x);
  const float ttd = 1.f/tt1;
  if( (*x) == 0 ) z=1.f;
  else if( (*x) < 4.f)
         z = 4.f*( sin(tt1*0.25f)*sin(tt1) )*(ttd*ttd);
  return z;

}
  //funcion para interpolar
__kernel
void rescubic(
    __global float *p_in,
      const int p_Wold,
      const int p_Hold,
      const int p_Wnew,
      const int p_Hnew,
      const float p_dx,
      const float p_dy,
    __global float *p_out)
{
 // Definiendo los indices
 const int x = get_global_id(0);
 const int y = get_global_id(1);
 const int oldY = (float)y*p_dy;
 const int oldX = (float)x*p_dx;
 const float ty = p_dy*(float)y ;
 const float tx = p_dx*(float)x ;
 //interpolamos
       //obteniendo los valores del pixel en la fuente mas cerca

       float q = 0;   //q es el valor interpolado
       //x y y son coordenadas de la fuente
       //el objetivo es up vp
    float vpp;
       float upp;
       const int oldhh = p_Hold-1; const int oldww = p_Wold-1;
       int vp,up;
       int vptt;
       for(int j = -3; j < 3; j++ )
       {
       vp = oldY + j;
              vpp = ty -(float)vp;
              if( vp < 0 ) vp = 0;
              if( vp > oldhh ) vp = oldhh;
              vptt = vp * p_Wold;
              float pp = 0;
              for(int i = -3; i < 3; i++)
              {
                     up = oldX + i;
                     upp = tx - (float)up;
                     if(up < 0) up = 0;
                     if(up > oldww) up = oldww;

                     pp += p_in[ vptt + up ] * lanczos4(&upp);
              }
              q += pp * lanczos4(&vpp);
       }


 //copiamos a la salida
 p_out[y * p_Wnew + x] = q;
 }

  //Kernel para interpolar,usando memoria local
__kernel
```

```
void rescubicimg2(
    __global float *imagenIn,
    __global float *imagenOut,
      int p_Hold,
      int p_Wold,
      int p_Wdevice,
      int p_Hnew,
      int p_Wnew,
      float p_dx,
      float p_dy,
       __local float * localImage,
         int localHeight,
         int localWidth)
{
 //propiedades propias del lanczos4
 int kernelRadio   = 3;
 int kernelAncho   = 7;

 //determinando el tamaño de los workgroups
 int groupStarCol = get_group_id(0)*get_local_size(0);
 int groupStarRow = get_group_id(1)*get_local_size(1);

 //determinando el ID local de cada work-item
 int localCol = get_local_id(0);
 int localRow = get_local_id(1);

 //determinando el ID global de cada work-item
 int x = get_global_id(0);
 int y = get_global_id(1);

 // Definiendo los indices
 int oldY = (float)y*p_dy;
 int oldX = (float)x*p_dx;
 float ty = p_dy*(float)y ;
 float tx = p_dx*(float)x ;

  //Copiando los datos a memoria local;
 //Bajando en las filas
 int curRow = oldY - kernelRadio;
 int curCol = oldX - kernelRadio;
 bool evaluar = curRow<0 || curCol<0 || curRow>p_Hold-1 || curCol>p_Wold-1;
 if(evaluar){
    localImage[ localRow*localWidth + localCol ] = 0.f;
 }
 else {
        localImage[localRow*localWidth + localCol] = imagenIn[curRow*p_Wdevice+curCol];
 }
 barrier( CLK_LOCAL_MEM_FENCE );

 if( y < p_Hnew && x < p_Wnew ){
       //iniciando desde el filtro izquierdo y arriba
       //obteniendo los valores del pixel en la fuente mas cerca
       float q = 0;   //q es el valor interpolado
       //x y y son coordenadas de la fuente
       //el objetivo es up vp
       float vpp;
       float upp;
       int vk = oldY-3; int uk = oldX-3;
       int vp = vk;
    for(int i = localRow; i <localRow + kernelAncho; i++ )
       {
       int offset = i*localWidth+localCol;

       int up = uk;
              vpp = ty -(float)vp;
              float pp = 0;
           for(int j = localCol; j < localCol + kernelAncho; j++)
              {
                     upp = tx - (float)up;
                     pp += localImage[offset++] * lanczos4(&upp);
           up++;
              }
              q += pp * lanczos4(&vpp);
        vp++;
       }
```

```
     //copiamos a la salida
     imagenOut[y * p_Wnew + x] = q;
  }//end if
}
```

D.3 KERNEL – ROTACIÓN CON INTERPOLACIÓN LANCZOS4

```
//valor interpolado en un punto de la señal continua
float lanczos4(float *x)
{
  //Interpolacion de lanczos 2
  const float pii = 3.14159265358979323846f;
  if( (*x) < 0.0f ) (*x) = -(*x);
  float z = 0;
  const float tt1 = pii*(*x);
  const float ttd = 1.f/tt1;
  if( (*x) == 0 ) z=1.f;
  else if( (*x) < 4.0f)
         z = 4.0f*( sin(tt1*0.25f)*sin(tt1) )*(ttd*ttd);
 return z;

}
//valor interpolado en un punto de la señal continua
//la funcion devuelve el x e y del rango en T-1
void GetRotacion( float *x0, float *y0, float *theta )
{
       //matriz T en coordenadas homogeneas
       const float a11 =  cos(*theta);
       const float a12 =  sin(*theta);
       const float a13 =  0.0f;

       const float a21 = -a12;
       const float a22 =  a11;
       const float a23 =  0.0f;

       const float a31 =  0.0f;
       const float a32 =  0.0f;
       const float a33 =  1.0f;

       //calculo de T^-1
       const float det_inv = 1.f/(   a11*(a22*a33)+ a12*(-a21*a33) );

       const float a11p = (a22              );
       const float a12p = (        - a12    );
       const float a13p = (          0              );

       const float a21p = (        - a21  );
       const float a22p = (a11*a33        );
       const float a23p = (        0      );

       const float a31p = (0                );
       const float a32p = (0                );
       const float a33p = (a11*a22 - a12*a21);

       const float h_inv = 1.f/(  a33p   )*det_inv;
       float x = ( a11p*(*x0) + a12p*(*y0)  )*h_inv;
       float y = ( a21p*(*x0) + a22p*(*y0)  )*h_inv;

       //devolviendo los valores de x0 e y0
       *x0 = x;
       *y0 = y;

}
//funcion para rotar
__kernel
void rotar(__global float *p_in,
 const int w,
 const int h,
 const float theta,
 __global float *p_out)
{

 // Definiendo los indices
 const int u = get_global_id(0);
 const int v = get_global_id(1);
```

```
 //calculamos T^-1
 float x0 = (float)(u-w/2);
 float y0 = (float)(v-h/2);
 GetRotacion( &x0, &y0, &theta );
 x0 += (float)(w/2);
 y0 += (float)(h/2);

 //interpolamos
        //obteniendo los valores del pixel en la fuente mas cerca
        const int u0 = (int)floor(x0);
        const int v0 = (int)floor(y0);

        float q = 0;   //q es el valor interpolado
        //u y v son coordenadas de la fuente
        //el objetivo es u' v'
        const int vphh = h-1, upww = w-1;

        for(int j = -3; j < 3; j++ )
        {
                int vp = v0 + j;
         const float vpp = y0 - (float)vp;
                if( vp < 0 ) vp = 0;
                if(vp > vphh ) vp = vphh;

                float pp = 0;
                int vptt = vp*w;
                for(int i = -3; i < 3; i++)
                {
                        int up = u0 + i;
                        const float upp = x0 - (float)up;
                        if( up < 0 ) up = 0;
                        if(up > upww ) up = upww;
                        pp += p_in[vptt+up] * lanczos4(&upp);
                }
                q += pp * lanczos4(&vpp);
        }


 //copiamos a la salida
 p_out[  v*w+u ] = q;
}

//funcion para rotar
__kernel
void rotarimg2( __global float *imagenIn,
 __global float *imagenOut,
   int h,
   int w,
   int cols_imagen,
   float theta,
 __local float * localImage,
   int localHeight,
   int localWidth)
{
 //propiedades propias del lanczos4
 int kernelRadio   = 3;
 int kernelAncho   = 7;
 int kernelPadding = 6;
 //determinando el tamaño de los workgroups
 int groupStarCol = get_group_id(0)*get_local_size(0);
 int groupStarRow = get_group_id(1)*get_local_size(1);
 //determinando el ID local de cada work-item
 int localCol = get_local_id(0);
 int localRow = get_local_id(1);
 // valor del pixel en la imagen de salida
 int u = groupStarCol + localCol;
 int v = groupStarRow + localRow;
 //calculamos T^-1
 float x0 = (float)(u-cols_imagen/2);
 float y0 = (float)(v-h/2);
 GetRotacion( &x0, &y0, &theta );
 x0 += (float)(cols_imagen/2);
 y0 += (float)(h/2);
 //u0 y v0 es la posicion del pixel mas cercana en la imagen original
 int u0 = (int)floor(x0);
```

```
 int v0 = (int)floor(y0);
 //la interpolacion se hara en los pixeles vecinos de (u0,v0)
 //copiando a memoria local estos pixeles
 for(int i = localRow; i < localHeight; i += get_local_size(1)){
       int curRow = v0 - localRow + i - kernelRadio;
       int tt  = i*localWidth;
       int tt1 = curRow*w;

       //avanzando en las columnas
       for(int j = localCol; j<localWidth; j+=get_local_size(0)){
              int curCol = u0 - localCol + j - kernelRadio;

              //Teniendo en cuenta la frontera
        if(curRow < 0 || curCol < 0 || curRow > h-1 || curCol > cols_imagen-1){
           localImage[ tt + j ] = 0.f;
        }
        else {
                     localImage[ tt + j ] = imagenIn[ tt1 + curCol ];
              }
       }
 }
 barrier( CLK_LOCAL_MEM_FENCE );

 //interpolando
  //Realizando la convolucion
 if( v < h && u < w ){
   //iniciando desde el filtro izquierdo y arriba
   float q = 0;   //q es el valor interpolado
   int vk = v0-3, uk = u0-3;
   int vp = vk-1;
   int up = uk-1;

   for(int i = localRow; i <localRow + kernelAncho; i++ )
   {
      int offset = i*localWidth+localCol;
      vp += 1;
      float vpp = y0 - (float)vp;
         float pp = 0;
      up = uk-1;
         for(int j = localCol; j < localCol + kernelAncho; j++)
         {
                up += 1;
                float upp = x0 - (float)up;

                pp += localImage[offset++] * lanczos4(&upp);
         }
         q += pp * lanczos4(&vpp);
    }


   //copiamos a la salida
   imagenOut[  v*w+u ] = q;
 } //fin de if

}
```

D.4 KERNEL – FILTRO

```
__kernel
void blurring2(
       __global float *imagenIn,
       __global float *imagenOut,
       __constant float *filter,
         int rows,
         int cols,
         int filtroAncho,
       __local float * localImage,
         int localHeight,
         int localWidth )
{

 //determinando el exceso para el filtro
 int filtroRadio = (filtroAncho/2);
 int exceso = filtroRadio*2;
```

```
 //determinando el tamaño de los workgroups
 int groupStarCol = get_group_id(0)*get_local_size(0);
 int groupStarRow = get_group_id(1)*get_local_size(1);

 //determinando el ID local de cada work-item
 int localCol = get_local_id(0);
 int localRow = get_local_id(1);

 //determinando el ID global de cada work-item
 int globalCol = groupStarCol + localCol;
 int globalRow = groupStarRow + localRow;

 //Copiando los datos a memoria local;
 //Bajando en las filas
 for(int i = localRow; i < localHeight; i += get_local_size(1)){
        int curRow = groupStarRow + i;
        int tt  = i*localWidth;
        int tt1 = curRow*cols;

        //avanzando en las columnas
        for(int j = localCol; j<localWidth; j+=get_local_size(0)){
                int curCol = groupStarCol+j;

                //Teniendo en cuenta la frontera
                if(curRow < rows && curCol < cols){
                        localImage[ tt + j ] = imagenIn[ tt1 + curCol ];
                }
        }
 }
 barrier( CLK_LOCAL_MEM_FENCE );

 ////////////////////////////////////////////////
 //Realizando la convolucion
 if( globalRow < rows-exceso && globalCol < cols-exceso ){
        //iniciando desde el filtro izquierdo y arriba
        float sum = 0.0f;
        int filterIdx = 0;

        for( int i=localRow; i< localRow+filtroAncho; i++){
                int offset = i*localWidth + localCol;
                sum += localImage[offset++]*filter[filterIdx++];
                sum += localImage[offset++]*filter[filterIdx++];
                sum += localImage[offset++]*filter[filterIdx++];
                sum += localImage[offset++]*filter[filterIdx++];
                sum += localImage[offset++]*filter[filterIdx++];
                sum += localImage[offset++]*filter[filterIdx++];
                sum += localImage[offset++]*filter[filterIdx++];
        }

        //escrbiendo la salida de vuelta
        imagenOut[(globalRow+filtroRadio)*cols + (globalCol+filtroRadio)] = sum;

 }//fin de if

}
```

BIBLIOGRAFÍA

[1] Fenoll, Ignacio Garcia. ***Aportación a la Segmentación y Caracterización de Imágenes Medicas en 3D.*** Ingenieria de Telecomunicacion, Universidad de Sevilla. Sevilla : s.n., 2010. págs. 15 - 19.

[2] Michael Y. M. Chen, Thomas L. Pope, David J. Ott. ***Basic Radiology*** *2nd Edition.* s.l. : McGraw Hill, 2011. págs. 4-9.

[3] ***Image Quality Assessment: From Error Visibility to Structural SimilarityImage Quality Assessment: From Error Visibility to Structural Similarity.*** Zhou Wang, Alan C. Bovik, Hamid R. Sheikh. 4, s.l. : IEEE Transactions on Image Processing, 2004.

[4] Juan Pablo Graffina, Raul Romo. ***Imagenes en Medicina*** - *UNSJ.* 2003.

[5] Wilhelm Burger, Mark J. Burge. ***Principles of Digital Image - Fundamental Techniques.*** London : Springer, 2009.

[6]. ***A phantom study of the accuracy of CT, MR and PET image registration with a block matching-based algorithm.*** A. Isambert, G. Bonniand, F. Lavielle, G. Malandain, D. Lefkpoulos. s.l. : ELSEIVER MASSON, 2008.

[7] ***Ultrasound Imaging System Performance.*** Kutay F. Ustuner, Gregory L. Holley. s.l. : Siemens Medical Solutions USA, 8 de setiembre del 2003.

[8] ***Parallel Beamformation Method to Enhance Ultrasound Images.*** S. M. Skhaei, A. Mahloojifar, H. Ghassemian. 2, s.l. : Iranian Journal of Electrical and Electronic Engineering, Abril del 2006, Vol. 2.

[9] Rafael C. Gonzales, Richard E. Woods. ***Digital Image Processing.*** New Jersey : Prentice-Hall.

[10] Jhon G. Proakis, Dimitris G. Manolakis. ***Digital Signal Processing: Principles, Algorithms and Applications.*** s.l. : Prentice-Hall, 1995.

[11] Vasco, Universidad del Pais. ***Calculo de la correlacion cruzada mediante la IDFT***. [En línea] [Citado el: 27 de 08 de 2012.] http://www.ehu.es/Procesadodesenales/tema8/corre1k.html.

[12] ***The Discrete Fourier Transform, Part 6: Cross-Correlation.*** Lyon, Douglas. 2, Zurich : Journal of Object Technology, March - April 2010, Vol. 9.

[13] Milanfar, Peyman. ***Super Resolution Imaging.*** Boca Raton, Florida : CRC Press , 2011.

[14] Michal Irani, Samuel Peleg. ***Motion Analysis for Image Enhancement Resolution, Occlusion, and Transparency.*** s.l. : Institute of Computer Science, The Hebrew University of Jerusalem.

[15] Mozdzynski, George. ***Concepts of Parallel Computing.*** s.l. : ECMWF, Marzo del 2012.

[16] Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumer. ***Introduction to Parallel Computing.*** Essex, England : Addison Wesley, 16 de junio del 2003.

[17] Thomas Rauber, Gudula Runger. ***Parallel Programming for Multicore and Cluster Systems.*** Berlin : Springer, 2010.

[18] Benedict R. Gaster, Lee Homes, David Kaeli, Perhaad Mistry y Dana Schaa. ***Heterogeneus Computing OpenCL.*** s.l. : Elseiver, 2012.

[19] ***A Frequency Domain Approach to Registration of Aliased Images with Application to Super-resolution.*** **Patrick Vanderwalle**, Sabine Susstrunk, Martin Vetterli. s.l. : Ecole Polytechnique Federal de Laussane, University of California, 2005.

[20] ***Improving Resolution By Image Registration.*** Peleg, Michael Irani and Samuel. 3, Jerusalem, Israel : CVGIF: Graphical Models and Image Processing, 25 de mayo de 1990, Vol. 53.

[21] Burge, Wilhelm Burger y Mark J. ***Digital Image Processing An Algorithmic Introduction Using Java.*** New York : Springer, 2008.

[22] ***Cubic Convolution Interpolation for Digital Image Processing.*** Keys, Robert G. 6, s.l. : IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, Diciembre de 1981, Vols. ASSP-29.

[23] ***Methods for Efficient, High Quality Volume Resampling in the Frequency Domain.*** Aili Li, Klaus Mueller, Thomas Ernst. s.l. : Center for Visual Computing, Computer Science Department.

[24] ***Bilateral Filtering for Gray and Color Images.*** C. Tomasi, R. Manduchi. 1998, IEEE International Conference on Computer Vision, Bombay, India.

[25] ***Super-Resolution Image with Estimated high Frequency Compensated Algorithm.*** Jong-Tzy Wang, Kai-Wen Liang, Shu-Fan-Chang y Pao-Chi Chang. Department of Electronic Engineering, Jinwen University of Science and Technology,Shindian, Taiwan and Department of Communication Engieenring, National Central University, Jhongli, Taiwan.

[26] ***PSF Estimation using Sharp Edge Prediction.*** Nell Joshi, Richard Szeliski, David J. Kriegman. University of California, San DIego.

[27] NVidia. ***CUDA API REFERENCE MANUAL.*** febrero del 2011.