UNIVERSIDAD DON BOSCO

"DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE CAPTACIÓN Y PROCESAMIENTO DE IMÁGENES DIGITALES"

TRABAJO DE GRADUACIÓN
PREPARADO PARA LA FACULTAD
DE INGENIERÍA

PARA OPTAR AL GRADO DE:

INGENIERO EN ELECTRÓNICA

POR

WELMER TOMÁS CRUZ SOSA
NÉSTOR JAVIER MACCAGNO
LUIS ALONSO MARIN VILLANUEVA

19 SEPTIEMBRE, 1998

SOYAPANGO, EL SALVADOR C.A.

UNIVERSIDAD DON BOSCO

RECTOR

ING. FEDERICO MIGUEL HUGUET RIVERA

SECRETARIO GENERAL

PBRO. PEDRO JOSÉ GARCÍA CASTRO S.D.B

DECANO DE LA FACULTAD DE INGENIERÍA

ING. CARLOS GUILLERMO BRAN

ASESOR DEL TRABAJO DE GRADUACIÓN

ING. OSCAR DURÁN VIZCARRA

JURADO EVALUADOR

ING. MAURICIO FLORES ROMERO

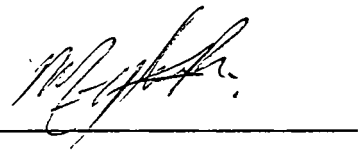
ING. FEDERICO JOSÉ LAÍNEZ

UNIVERSIDAD DON BOSCO

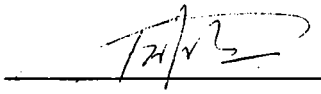
FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

JURADO EVALUADOR DEL TRABAJO DE GRADUACIÓN



ING. MAURICIO ENRIQUE FLORES
JURADO



ING. FEDERICO JOSÉ LAÍNEZ
JURADO

ING. OSCAR DURÁN VIZCARRA
ASESOR

DEDICATORIA

Para emprender cualquier tarea en esta vida es condición necesaria contar con la inspiración de Dios Todopoderoso, la voluntad humana y el apoyo de la gente que tanto nos ama. Nuestro caso no es ninguna excepción, y a pesar de tener que realizar una lista, no queremos olvidarnos de aquellos que en todo momento brindaron incondicionalmente su apoyo para que este proyecto pudiera concretarse y dar los mejores frutos.

De Welmer Tomás Cruz Sosa:
A mis padres Tomás Cruz y Aurora Sosa de Cruz, por su apoyo incondicional, a mis hermanos Francisco Antonio y Maricela E. Cruz, ya que sin ellos esto no hubiera sido posible.

De Néstor Javier Maccagno:
A mi esposa Yolanda Rocío Pando, a mis hijas Lucila Yasmín y Rocío Trinidad, a mis padres Néstor José Maccagno y Catalina Luján Rigo y a mis suegros Eufracio Jaime Pando y Josefina Oliaga.

De Luis Alonso Marín Villanueva:
A mis padres Juan Angel Marín, Reina Villanueva, a todos mis hermanos a Ericka Patricia Hernández por su apoyo incondicional, en fin a todos mis familiares y amigos que siempre aportaron su ayuda de una u otra manera.

Además, a nuestro asesor del proyecto de graduación: Oscar Durán Vizcarra.

ÍNDICE

TABLA DE FIGURAS

Página

INTRODUCCIÓN

En las últimas tres décadas han sido muchos los esfuerzos de los investigadores científicos en materia de procesamiento de imágenes. El procesamiento de imágenes se puede ver aplicado en áreas tales como la industria, la biomédica, el espacio y la geografía física.

En muchas aplicaciones del procesamiento de imágenes se desea resaltar cierto tipo de característica que se encuentra presente en la imagen. Por ejemplo el microbiólogo centra su atención en el realce de la imagen de las células, en tanto que un astrónomo se interesa en el mejoramiento de imágenes de galaxias lejanas.

El interés fundamental de este trabajo es investigar y desarrollar un sistema de procesamiento de imágenes compuesto básicamente por una cámara de inspección, una interfaz o frame grabber y un software de aplicación o programa. El programa de aplicación es un producto original, desarrollado en un lenguaje de alto nivel como el Visual C++, y su finalidad es proveer de herramientas básicas para realizar el procesamiento de imágenes digitales. Si bien es conocido que en el mercado internacional es posible conseguir material de este tipo ya desarrollado, aun así se cree firmemente que el impulsar el desarrollo local de tecnología es imprescindible para que se reduzca la gran brecha que nos separa de los países desarrollados. Más aun si nos concentramos en la cantidad de aplicaciones del procesamiento de imágenes, como se ha expuesto anteriormente.

Los laboratorios de la Universidad Don Bosco no cuentan hasta la fecha con el material físico ni literatura en materia de procesamiento de imágenes como para poder enfrentar los nuevos retos de la institución, marcados por la constante actualización curricular de las carreras. Con el trabajo se pretende cubrir una de estas áreas desprovistas de material didáctico y contribuir al fortalecimiento de la educación integral de los futuros profesionales.

El primer capítulo propone los lineamientos y pilares fundamentales sobre los que se sustenta el trabajo, sus objetivos, su alcance y limitaciones, y una justificación clara de la solución propuesta. En el segundo capítulo se presenta una discusión de los conceptos fundamentales partiendo de las bases del procesamiento de imágenes hasta la extracción de características de una imagen. En el tercer capítulo se desarrollan experimentos y se presentan sus resultados y conclusiones. Todo esto permitirá tener un parámetro para realizar las futuras guías de laboratorio en temas fundamentales como son las técnicas de iluminación de objetos y óptica. Para concluir, en el capítulo cuarto se presentan los programas desarrollados que permitirán realizar las funciones de procesamiento de imágenes y un manual de utilización del sistema completo.

Por último se desea destacar que las diferencias que existen en el planteamiento y desarrollo de este Proyecto de Graduación y el Anteproyecto de Graduación están justificadas en este mismo trabajo, y son por ende la guía para conseguir nuestro objetivo principal, demostrar nuestras habilidades y conocimientos adquiridos en esta etapa de la vida.

Capítulo 1
Bases para el desarrollo del proyecto

1.1 OBJETIVO GENERAL

- Diseñar y construir un sistema de adquisición y procesamiento de imágenes digitales que permita desarrollar tareas orientadas a la comprensión del proceso de percepción visual de las mismas.

- Reconocer las etapas sucesivas a las que puede ser sometida una imagen digital con características determinadas para luego poder incorporarla a algún sistema específico.

1.2 OBJETIVOS ESPECÍFICOS

- Desarrollar experimentos para comprobar técnicas de iluminación.

- Diseñar y editar los programas para el tratamiento de imágenes, apoyándose en herramientas matemáticas y en lenguajes de alto nivel.

- Desarrollar un programa que permita al usuario del sistema la utilización de las herramientas de procesamiento de imágenes disponibles en dicho entorno e interactuar con el resultado obtenido.

- Proveer un manual de operación del sistema como una guía para su buena utilización, comprensión y explotación.

1.3 DESCRIPCIÓN DEL PROBLEMA

Hace un par de décadas únicamente era un sueño pensar que algún día existirían máquinas o computadoras con la capacidad de captar, integrar e interpretar voz e imágenes, y ni siquiera se podía pensar en que ellas realizarían una función tal que pudieran agilizar los procesos de producción. Los avances técnico-científicos lo están haciendo posible.

La adaptación de los planes de estudio en la Universidad Don Bosco a la realidad tecnológica mundial actual la coloca en la necesidad de tener que contar con material didáctico para lograr alcanzar los objetivos planteados. Específicamente en la asignatura donde se desarrollarán las técnicas de percepción visual, la Universidad no cuenta con la tecnología que sustente y aporte los requerimientos mínimos para alcanzar los objetivos de la misma. Si bien el equipamiento existente en los Laboratorios permite formar profesionales con un perfil definido en la experimentación, es importante seguir en la misma línea y contribuir en la medida que cada uno pueda. Es un gran desafío y que sin ninguna duda contribuirá a nuestro crecimiento profesional, humano y espiritual.

1.4 JUSTIFICACIÓN

El campo de Procesamiento de Imágenes Digitales en El Salvador es un campo poco provisto de profesionales relacionados al tema. En consultas a profesionales se ha podido detectar una inquietud a cerca de los alcances y aplicaciones del procesamiento de imágenes. En muchos casos esta tecnología viene a incorporarse a la Industria, Hospitales, etc., en

lugar de transferirse el conocimiento tecnológico. Este trabajo se dispone a romper con esta forma de proceder.

De tal manera que haciéndonos partícipes en el proceso de investigación y desarrollo de un producto con características didácticas y con el complemento del trabajo de futuras generaciones, existirá la posibilidad de enmarcarlo dentro del ámbito industrial y productivo, es así, ya que se está proyectando la necesidad de incorporar el producto final a una maquinaria automática del tipo CNC (Control Numérico Computarizado) o en cualquier otro sistema. Se contribuirá con el desarrollo de un sistema de procesamiento de imágenes digitales que se complementará con lo existente en los Laboratorios de la Universidad Don Bosco.

La puesta a punto de los nuevos planes curriculares y las ambiciones que actualmente tiene la Universidad nos marcan el camino futuro a seguir. Es el momento oportuno para desarrollar elementos de tecnología de punta que sirvan de apoyo a las nuevas generaciones. Este trabajo viene a ocupar un lugar vacante en el plan curricular a implementarse a partir del año 1998. Plan que contempla la incorporación de temas de actualidad tecnológica. La visión por computadora es uno de ellos y nuestro compromiso es enriquecer a la materia afin con un material escrito y de apoyo experimental que ayude a cumplir con los objetivos del curso.

Selección del Equipamiento Adquirido

En vista de no haber encontrado en el mercado local personal capacitado en soporte y ventas de los elementos a cotizar como lo son las cámaras digitales, frame grabbers, tarjetas interfaces y los programas o librerías para control de dicha tarjeta, se hizo necesario realizar las conexiones con proveedores extranjeros. En el anexo pueden ubicarse las direcciones de Internet a través de las cuales han podido localizarse. Cabe destacar que existe una variedad muy extensa de ofertas y oferentes, aunque no todos han respondido de la misma manera

ante los requerimientos realizados. A continuación se presenta un cuadro comparativo de las opciones de compra que se presentaron durante el período enero y febrero del corriente año, y que se han tomado como base para realizar la selección más adecuada con relación a las prestaciones de los elementos, calidad y su costo inicial.

CUADRO COMPARATIVO - OPCIONES DE COMPRA

CÁMARA	*INTERFACE*	*FORMATO*	*SOFTWARE*	*PRECIO*
FUJI FV-7 Photo Imag.	SNAPY	JPEG (LPT)	Del Snapy	$499 + $ 150 = $649
EPSON PHOTO PC 600		JPEG (SERIE)	De Epson	$981.49 (1)
PULNiX TM-7CN				$716
SONY XC-73				$725
SONY XC-73CE				$795
SONY XC-75				$775
SONY XC-75CE				$845
COHU 4910				$860
CRESCENT T 370				$305
	PIXCI - SV4 (EPIX)	8 BITS	XCOBJ	$575 + $495 = 1.070
	FPG - 44 (DIPIX)	Idem	Adicional	$1,600
	IMAQ PCI - 1408	Idem	NI - IMAQ (Incl.)	$1.195
CRESCENT T 330	MTPCI-MN	Idem	MTPCI-MN-95	$1.227.09(2)
CRESCENT T 370	Idem	Idem	Idem	$1.155.38(2)

Notas: (1) Este valor fue tomado de un proveedor local y el valor del cambio adoptado fue de 8.75 colones por dólar norteamericano.

(2) El cambio de esta moneda se tomó en 125.5 yenes por dólar norteamericano.

Finalmente el equipo seleccionado es:

1.- Cámara CRESCENT T-370

2.- Interface PIXCI-SV4

3.- Biblioteca XCOBJ (control de interface)

Este es parte del equipo que definitivamente conformará el sistema de procesamiento de imágenes.

Comparativo de la Opción de Compra y un Desarrollo en Laboratorio

Para clarificar más aún la selección realizada para la compra, realicemos una comparación con un desarrollo en Laboratorio de la tarjeta de adquisición y el software de control para la misma.

Hardware:	
Procesador de Señales Digitales (DSP)	$ 250.00
Hardware adicional (convertidores, periféricos, etc.)	$ 300.00
Mano de obra (150 hs. a 8 $/hora)	$ 1,200.00
Software (100 hs. A 8 $/hora)	$ 800.00
Imprevistos	$ 150.00
TOTAL:	**$ 2,700.00**

En vista de los resultados obtenidos, pueden observarse dos cosas: 1) Que el tiempo invertido para realizar un equipo de características y prestaciones similares a los cotizados es tan significativo como para llevar a cabo un proyecto de graduación aparte de éste, y 2) Que los costos no justifican la inversión del desarrollo, desde este punto de vista.

1.5 ALCANCES Y LIMITACIONES

Alcance de la Solución

- El principal compromiso es la realización de un sistema de procesamiento de imágenes que permita obtener una imagen adecuada partiendo de los elementos básicos de captación de la señal.

- El sistema estará capacitado para operar con imágenes de dos dimensiones.

- Se tratará con elementos de iluminación, óptica, captura de imagen hasta la aplicación de procedimientos de programas que permitan operar características como el contraste, ruido, bordes, textura y realizar transformaciones morfológicas.

- Los experimentos de iluminación quedarán diseñados de forma tal que puedan ser utilizados para comprobar los resultados de la aplicación de las técnicas de iluminación.

- Se pretende desarrollar una interfaz de programas lo más amigable posible de manera que sea amigable la utilización del software.

Limitantes de la Solución

- El sistema no tratará la segmentación de imágenes, el reconocimiento de objetos y mediciones.

- El tratamiento de imágenes en color, y de tres dimensiones no estará comprendido dentro del trabajo, aunque puede considerarse como un complemento a integrarse en un futuro.

1.6 ANTECEDENTES HISTÓRICOS Y TÉCNICOS

El desarrollo en las técnicas del procesamiento de imágenes no es algo nuevo, ni producto de la casualidad. Este surge inicialmente como una inquietud de nuestros antepasados; cuando estos registraban los acontecimientos y eventos importantes en sus vidas a través de dibujos e imágenes. A medida crecieron los pueblos, han sido precisamente estos elementos pictóricos, los que han servido para llegar a conocer antiguas civilizaciones y en general parte de nuestro pasado.

Formalmente el trabajo en el tratamiento de las imágenes comenzó con el descubrimiento accidental del Italiano J.B. Porta, el cual descubrió que un rayo de luz que penetraba un pequeño orificio en la puerta de un cuarto oscuro, producía una imagen invertida de la escena exterior, sobre una pantalla blanca situada enfrente de la puerta. Este descubrimiento es el fundamento de las modernas cámaras de hoy. Pero fue gracias a los notables avances de la química, a mediados del siglo XIX, que dio inicio a lo que se conoce ahora como las antiguas técnicas del *procesamiento de imágenes fotográficas*. Este tipo de procesamiento se realizaba también apoyado por las técnicas de iluminación y óptica, conocidas hasta ese momento. En 1884 George Eastman desarrolló una red de laboratorios fotográficos, haciendo el uso de la fotografía moderna tan sencilla y fácil de usar, como actualmente la conocemos.

Con el pasar del tiempo, las exigencias de calidad en los resultados de la fotografía fueron en aumento, y como es lógico, los avances en las técnicas no fueron lo suficientemente buenas para satisfacer estas exigencias. Por esto fue necesario buscar nuevos métodos para la captura y procesamiento de imágenes. En este momento inician las investigaciones sobre los métodos en el *procesamiento de imágenes digitales*. El interés en estos métodos proviene de dos áreas principales de aplicación: el mejoramiento de la apariencia de las imágenes para que el humano tuviese una mejor apreciación visual como también los estándares y además la posibilidad de dotar a las máquinas con la visión artificial.

Las primeras aplicaciones que utilizaron estos métodos fueron las de transmisión y almacenamiento de imágenes. El sistema Bartlane, primero de este tipo; destinado a la transmisión de fotografías digitalizadas por medio de un cable submarino que cruzaba el océano Atlántico. El tiempo de transmisión de una imagen era de dos a tres horas. Este sistema exploraba la imagen elemento por elemento, representando cada tono de gris como un dato almacenado en una cinta de papel. Luego ésta era codificada, transmitida su información y recuperada a través de una impresora de impacto que reproducía cada tono de gris, con un ancho de impacto diferente.

Un acontecimiento que dio un gran impulso a la investigación del procesamiento de imágenes digitales fue sin duda la segunda guerra mundial, que unido al invento de la computadora y al desarrollo de la microelectrónica, formaron la base para el desarrollo formal de las técnicas y conceptos del procesamiento de imágenes digitales modernos. El reconocimiento de objetivos enemigos, desde el aire, fue una aplicación muy común en el tiempo de la guerra. Se construyeron sistemas especializados, destinados para ese fin, que involucraban a fotógrafos, investigadores y técnicos, los cuales trabajaban primero en el mejoramiento general de la imagen, que involucraba: un incremento del contraste e iluminación y la eliminación de algunos defectos producidos por la baja calidad de las cámaras y las malas condiciones del clima.

En el campo de las investigaciones espaciales, el procesamiento de imágenes sirvió para mejorar cientos de imágenes enviadas por las naves espaciales, y que debido a muchos factores, no era posible apreciar todos los detalles de las imágenes. La distorsión propia de las cámaras de vídeo dentro de la nave, el ruido espacial, la distorsión debido a movimientos bruscos y la pérdida de datos al momento de la transmisión, son algunos de estos factores. Todo el procesamiento de imágenes se realizó inicialmente en el laboratorio de propulsión a chorro de la NASA. Aquí se procesaron imágenes de la superficie lunar provenientes de la nave Ranger 7, que más tarde fueron convertidas a formato digital y procesadas por medio de la aplicación de los algoritmos desarrollados por los investigadores. A mediados de 1960 la NASA tuvo que recurrir a estas técnicas de procesamiento para determinar los sitios válidos para el aterrizaje de la famosa nave espacial del programa Apollo 11. Después de los buenos resultados obtenidos con estas técnicas, se llevaron a cabo nuevas misiones espaciales. Entre estas se encuentra la de la nave Surveyor 7, la cual envió miles de imágenes, las que fueron procesadas más tarde con el propósito de determinar la composición y estructura de la superficie lunar.

También el procesamiento sirvió para crear un mapa de la superficie de Marte, gracias a las fotografías digitalizadas y pre-procesadas por las naves de la serie Mariner. Casi todos los planetas del sistema solar han sido estudiados por medio de las imágenes tomadas por

estas naves, y gracias a las técnicas de procesamiento ha sido posible apreciar detalles de éstas, que de otra manera no se podrían apreciar.

El trabajo en el procesamiento de imágenes, hoy en día se ha extendido a la astronomía, la cuál ha ayudado a estudiar y descubrir nuevas constelaciones; a la meteorología, que constituye la base para obtener la información respecto al clima y los cambios en la superficie de la tierra; la medicina, que hace uso del procesamiento para el análisis y evaluación de las imágenes de rayos X, de resonancia magnética, ultrasonido, etc. En el área de la producción y la robótica, la visión artificial junto con el procesamiento de imágenes han permitido la automatización y control de muchos procesos, como por ejemplo la fabricación de vehículos, la manufactura de circuitos integrados, computadoras, electrodomésticos, etc. [26].

Capítulo 2

Bases Teóricas

2.1 ELEMENTOS DE UN SISTEMA DE PROCESAMIENTO DE IMÁGENES

Existe una serie de elementos de hardware, programas y bases teóricas que comprenden un sistema completo de procesamiento de imágenes. Un sistema como el que se detalla a continuación es un ejemplo: (1) iluminación y óptica, (2) adquisición, (3) almacenamiento, (4) procesamiento, y (5) presentación.

2.1.1 ILUMINACIÓN Y ÓPTICA

La iluminación es una parte fundamental dentro de un sistema de captación y procesamiento de imágenes, por lo que depende en gran medida de ésta para la simplificación del análisis y posterior interpretación de la escena captada. Dicho de otra manera, dependerá de la técnica a utilizar en cuanto a la iluminación, para obtener el realce de las propiedades particulares de los objetos en escena de acuerdo a los resultados esperados. La iluminación es un factor que usualmente afecta la complejidad de los algoritmos aplicados en aquellos procesos en los que se desea modificar alguna o varias propiedades de las imágenes. La iluminación producida en el entorno del objeto a captar, sin la utilización de una fuente dedicada para tal función, hace que se obtenga un mal resultado, bien porque las imágenes poseen un bajo contraste, o se producen sombras, o cuentan con

una baja definición, realce de brillos, reflexiones especulares e ilusiones visuales. Todos estos defectos provocan interpretaciones falsas de la escena con suma facilidad [24].

La iluminación, posee una razón científica y de manera general se conoce como la densidad de la potencia radiada incidente sobre una superficie receptora evaluada a través del espectro, en términos de la eficiencia luminosa espectral. La iluminación es simbolizada por la letra E y no es más que la densidad de flujo luminoso por unidad de área. La unidad de medida de la iluminación en el Sistema Internacional es el Lux. La ecuación que define tal magnitud física se especifica a continuación:

$$E = k\sum_{i=1}^{n} V_i G_i + k\int_0^{\infty} V_\lambda G_\lambda d\lambda \qquad (2.1\text{-}1)$$

donde V y G son: componente de la irradiación y el valor espectral de la eficiencia luminosa respectivamente [21].

Técnicas de Iluminación

a) Iluminación Direccional

Consiste en aplicar una iluminación orientada o dirigida al objeto, usando un haz altamente direccional, así como es el caso de un rayo láser. En este tipo de técnica se requiere medir el grado de dispersión. Es muy simple de realizar y permite la observación cuidadosa de superficies rugosas y la detección de fallos de superficies uniformes. Así como también la alta utilidad en localización y reconocimiento de piezas, inspección de la superficie de los objetos, etc. En general en sistemas de inspección como el tipo realizado en control de calidad.

La forma del haz luminoso depende de la aplicación y del objeto al que se analizará. De acuerdo a lo anteriormente expuesto, es posible mencionar algunos de los casos, como por ejemplo:

- Punto luminoso.
- Plano de luz.
- Corona circular luminosa.

En la figura 2.1.1(a) se trata de representar un ejemplo típico de iluminación, mediante el uso de planos de luz.

La técnica de iluminación anterior es muy utilizada, mas sin embargo, una de las desventajas principales es que por la misma propiedad de la direccionalidad, tienden a aparecer sombras que causan una atenuación de algunas características de los objetos iluminados, por lo tanto, el tamaño de los objetos a iluminar no puede ser demasiado grande, dada la limitación de tener luz colimada para toda la superficie iluminada.

b) Iluminación Difusa

La principal propiedad de este tipo de iluminación es que se pretende que los haces luminosos incidan sobre el objeto desde todas las direcciones y no exclusivamente desde la fuente de luz. Ilumina indirectamente la escena y permite la utilización de técnicas secuenciales, por conmutación de distintos focos luminosos, con el objetivo de obtener distintos contornos de los objetos de las escenas. La combinación de estos contornos, con la información espacial relativa de la cámara-escena y las direcciones de la iluminación, conduce a la eliminación de sombras y la obtención de un modelo tridimensional del objeto con una sola cámara [24].

Esta técnica de iluminación es ampliamente utilizada en aquellos casos en los que la superficie de los objetos a analizar se mantiene suave y regular. Se emplea por lo general en aplicaciones donde las características de las superficies son importantes para el análisis. Una de las propiedades que presenta la misma, es que no proporciona el máximo contraste del objeto.

Las fuentes de luz apropiadas para la aplicación de la técnica, son aquellas como por ejemplo, el fluorescente y las fuentes halógenas con haces de fibra óptica [3]. La figura 2.1.1(b) ejemplifica el método descripto en los párrafos anteriores.

c) Iluminación a Contraluz

Consiste en aplicar al objeto una fuente luminosa desde la parte de atrás, tal que la cámara, el objeto y la fuente de iluminación formen una línea recta. Entre las propiedades que se obtienen cuando se emplea este tipo de iluminación, se pueden mencionar la obtención de imágenes con sólo dos niveles de gris, es decir imágenes en blanco y negro.

Se le utiliza en ocasiones en las que se desean aplicaciones de localización, reconocimiento de piezas pequeñas y ligeras en sistemas industriales donde se realizan tareas de control de calidad, clasificación y análisis dimensional. Donde la silueta es suficiente para el reconocimiento de los objetos. La gran ventaja de este sistema de iluminación, es que permite extraer el contorno o bordes de los objetos con una gran precisión de manera rápida y segura. Es importante mencionar que se pierden todos los detalles, además de la posibilidad de detectar fisuras en el objeto. En la figura 2.1.1(c) se presenta un ejemplo del uso de este tipo de iluminación.

La técnica del contraluz se subdivide en contraluz único centrado, contraluz único desplazado y contraluz doble equilibrado. La utilidad de cada una de las anteriores está sujeta a las características deseadas en cada una de las escenas [13].

d) Iluminación Estructurada

Este tipo de iluminación que en otros casos es conocida como iluminación modulada, hace uso de la proyección de puntos, franjas o rejillas sobre la superficie de trabajo,

generándose de esta manera patrones de iluminación. Está orientada hacia la obtención de las curvaturas de la superficie de los objetos en escena [24]. Presenta dos ventajas con respecto al resto de los métodos de iluminación, que son:

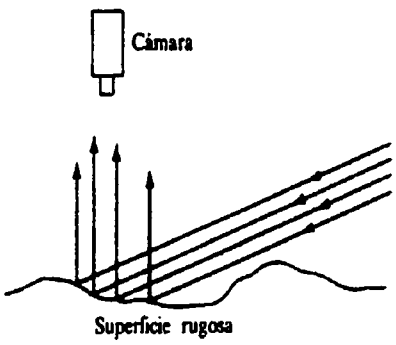
- En primer lugar se establece un patrón de luz conocido sobre la superficie de trabajo y las diferencias con este patrón establecen la presencia de un objeto, de esa manera simplifica el problema de detección de un objeto.

- En segundo lugar, analizando la forma en la que el patrón de luz es distorsionado, es posible obtener la información necesaria de las características tridimensionales del objeto.

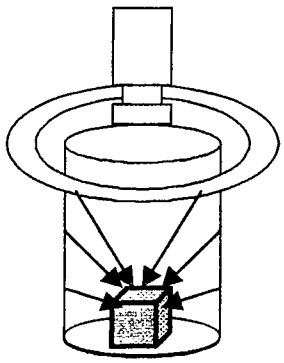
En la figura 2.1.1(d) se presenta un gráfico en el que se puede observar la técnica antes mencionada.

e) Iluminación Oblicua

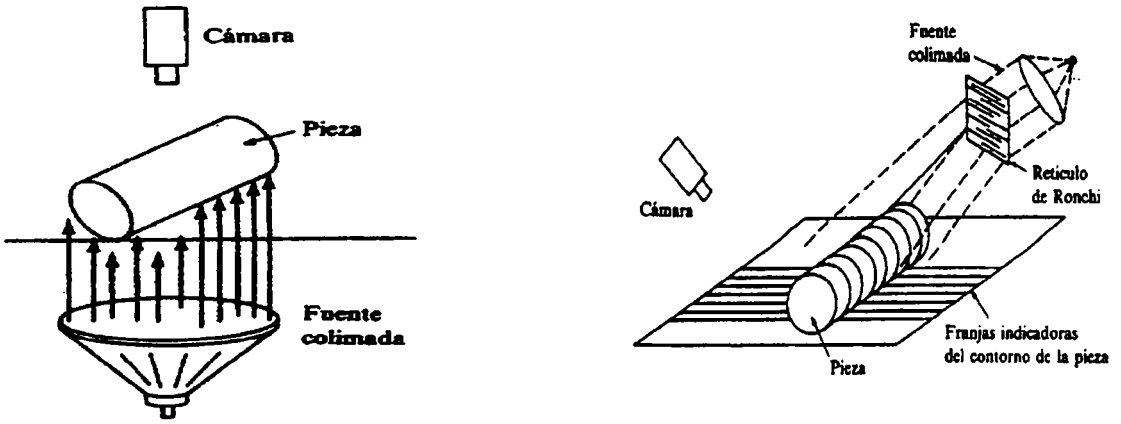
El objetivo principal de la iluminación oblicua, es crear sombras encargadas de aumentar el contraste de las partes tridimensionales de los objetos. Este tipo de iluminación se adapta en aquellos casos en los que se desea generar sombras sobre objetos cuyo contraste es pequeño respecto al fondo [3].



a) Iluminación direccional.



b) Iluminación difusa.



c) Iluminación a contraluz. d) Iluminación estructurada.

Figura 2.1.1. Técnicas de iluminación.

Óptica

La óptica es un elemento muy importante, para la determinación de las imágenes al interior de una cámara ya sea de vídeo o fotográfica. Contiene un convertidor (Sensor) que transforma las variaciones de luminancia en variaciones eléctricas. En su interior se encuentra un complejo sistema óptico-mecánico compuesto por varias lentes y los correspondientes dispositivos para su desplazamiento.

Para la determinación del objeto existen dos magnitudes ópticas involucradas que se pueden definir así:

- La primera, expresada en milímetros define una distancia focal *f*, la cual puede ser fija o variable, a este último se le conoce como zoom, ya que se caracteriza por no perder el enfoque con el cambio de la distancia focal.

- El otro valor es una cantidad *F* máximo, es decir la mayor abertura del diafragma en la escala *F*. Sobre otro anillo similar se encuentra una escala graduada en metros, y que sirve para regular el enfoque según la distancia del objeto encuadrado [3].

Enfoque y Distancia Focal

Se define como enfoque de la lente para una distancia determinada del objeto, la acción de aproximar el plano de elementos sensibles a la lente para hacerlo coincidir con el de la formación de la imagen.

La distancia focal *f* se define como la distancia lente-plano de formación de la imagen cuando el objeto se encuentra situado en el infinito. La definición anterior se refiere a que es la distancia entre la lente y el punto de convergencia de todos los rayos que inciden perpendicularmente a ella. A este punto de convergencia se le conoce como foco de la lente.

Una lente posee dos focos, uno que se encuentra en el lado del objeto, conocido como foco primario, y el otro se encuentra en el lado de la imagen, llamado foco secundario.

Las lentes utilizadas en las cámaras de vídeo o de televisión suelen ser lentes compuestas, que combinan una serie de lentes simples con el objeto de corregir las aberraciones que degradan la definición de la imagen.

Por lo tanto si el objeto se encuentra en una distancia finita *d*, la distancia entre el plano de la imagen y la lente aumentará, es decir el plano que contiene los puntos en los que los haces divergentes de rayos que proceden de cada punto del objeto, se encontrará a una distancia de la lente que será mayor que la distancia focal. La variación de la distancia focal para lentes del mismo diámetro, se traduce en un mayor o menor tamaño de la imagen producida por el objeto.

El Diafragma y la Luminosidad

La luminosidad de un objetivo hace referencia a la máxima cantidad de luz que pasa a través de él, siendo regulada por un diafragma que posee un diámetro variable situado

generalmente entre dos grupos casi simétricos de lentes; el diafragma se encuentra constituido por láminas de metal muy delgadas que forman una abertura aproximadamente circular y que son de accionamiento suave.

Su posición debe ser tal que regule la cantidad de luz proveniente del objeto, pero que no obstruya los rayos luminosos que forman parte de la imagen. La cantidad F es el resultado del cociente entre la distancia focal y el diámetro del diafragma, de modo que cuanto más pequeño sea su valor, mayor es su abertura, valor que es siempre el mismo para un mismo objetivo.

Se recurre al tratamiento antirreflejante, que consiste en depositar sobre las superficies de las lentes por procedimientos al vacío o similares, sustancias tales como floruro de magnesio, para anular los reflejos debidos a las interferencias entre los rayos incidentes y los reflejados que son desfasados ¼ de su longitud de onda por el espesor de la capa antirreflexión.

Es obvio que al reducir la cantidad de luz al interior del objetivo, debida a los reflejos parásitos, mejora notablemente la definición de la imagen, puesto que se aumenta el contraste de la misma y la saturación de tonos y colores.

Profundidad de Campo

Un objeto contenido en un plano perpendicular al eje óptico de la lente, se enfoca sobre un plano de imagen. En realidad si el objeto tiene profundidad, la imagen aparecerá dentro de cierto margen por delante y por atrás del punto sobre el que la lente está enfocada.

La zona por delante y detrás del plano de imagen para la cual el desenfoque es menor que el círculo de confusión permisible (Elemento sensible de la cámara) se llama profundidad de enfoque.

La profundidad de campo es la zona dentro de la cual el objeto forma una imagen que está dentro de la profundidad de enfoque. Cualquier cosa dentro de la profundidad de campo aparecerá nítida como si estuviera enfocada.

La profundidad de campo posee las siguientes características:

- Mayores números F dan mayores profundidades de campo.

- Menores distancias focales dan mayores profundidades de campo.

- Mayor distancia al objeto supone mayor profundidad de campo.

La primera de las características supone que cuanto menor sea la abertura de la lente mayor será la profundidad de campo, luego el diafragma además de limitar la intensidad del haz luminoso, mejora la calidad de la imagen, al aumentar la profundidad de campo; así la nitidez de la imagen mejora notablemente cerrando el diafragma dos o tres puntos, a partir de la máxima abertura.

Distancia Hiperfocal

La distancia hiperfocal H es la distancia de enfoque que tiene un margen desde infinito hasta $\frac{1}{2}H$ dentro de la profundidad de campo. El enfoque es innecesario dentro de este campo.

Ya que la distancia hiperfocal es aquella a la cual el límite lejano de la profundidad de campo está situado en el infinito:

$$H = \frac{f^2}{F \times c} \qquad (2.1\text{-}2)$$

c es el diámetro del círculo de confusión admitido de la focal normal.

Ángulo Visual

Es el tamaño máximo del objeto cuya imagen pueda ser contenida dentro del plano de imágenes en relación con el tamaño de la zona sensible de la cámara y con la distancia a la que dicha zona se encuentra de la lente.

La distancia a la que el plano de formación de la imagen se encuentra de la lente depende de dos factores: la distancia a la que se encuentra el objeto enfocado y la distancia focal misma. Bajo estas condiciones se define el ángulo visual como el ángulo que forman dos rayos que inciden sobre los bordes externos de la zona sensible de la imagen cuando ésta se encuentra enfocada a una distancia infinita.

El ángulo visual se expresa matemáticamente como:

$$w = 2Arctan\left(\frac{y}{2f}\right) \tag{2.1-3}$$

donde y es el tamaño de la zona sensible y f es la distancia focal.

Dado que es usual que la zona sensible no es exactamente cuadrada sino que es por lo general rectangular es apropiado hablar de un ángulo visual vertical y de otro horizontal, así como también de un ángulo con respecto de la diagonal [3].

A continuación se presenta un esquema, el cual ilustra algunas de las variables más usadas en óptica geométrica.

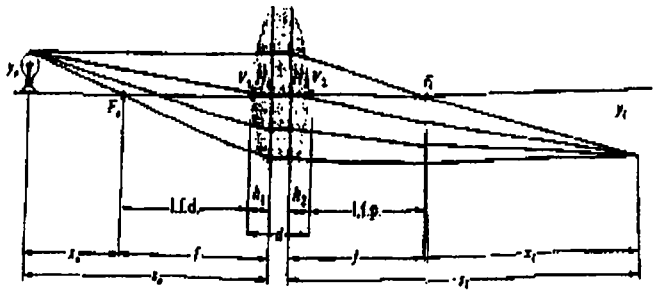


Figura 2.1.2. Algunas variables tratadas en óptica geométrica.

Aberraciones y Distorsión de la Imagen

En el sentido general, el término aberración se refiere a las imperfecciones introducidas en las imágenes, debido a los sistemas ópticos. Se pueden considerar de dos tipos así:

- Cromáticas.
- Geométricas (Seidel).

Mientras que las aberraciones cromáticas se muestran ya en el campo paraxial, las aberraciones geométricas aparecen si nos apartamos de él, es decir si los haces luminosos tienen una abertura importante y proceden de puntos situados fuera de la lente [8].

Aberración Cromática

La aberración cromática se debe al diferente índice de refracción de un rayo luminoso según su longitud de onda. Existen dos tipos de aberraciones cromáticas que son: Longitudinal y Lateral.

Aberración Cromática Longitudinal

Este tipo de aberración provoca que los rayos de distintas longitudes de onda procedentes de un punto sobre el eje óptico de la lente se enfoquen sobre diferentes planos de imagen.

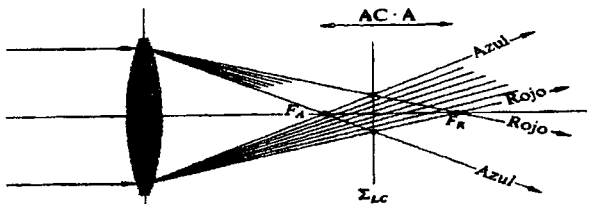


Figura 2.1.3. Aberración cromática longitudinal.

Aberración Cromática Lateral

Ocurre debido a que la magnificación de la lente cambia según la longitud de onda del rayo. Se produce un desplazamiento lateral de los puntos de convergencia de los rayos de diferente longitud de onda.

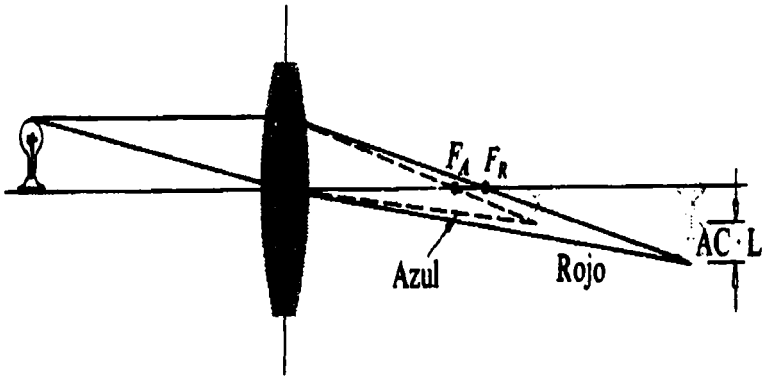


Figura 2.1.4. Aberración cromática lateral.

Aberración Geométrica

Las cinco aberraciones geométricas básicas son conocidas como fueron definidas por Seidel, quien las clasificó de la siguiente manera: aberración esférica, coma, astigmatismo, curvatura de campo y distorsión. Las primeras cuatro afectan la nitidez de las imágenes.

Aberración Esférica

Se debe a una mayor convergencia de los rayos a medida que aumenta la distancia de su punto de incidencia sobre la lente respecto al eje de ésta. Por lo tanto si el plano de la imagen se sitúa a la distancia focal de los rayos que pasan por el eje de la lente, la imagen aparecerá desenfocada lejanos al centro de la imagen.

Si el tamaño de la lente es pequeño, esta aberración se hace casi despreciable. La aberración esférica se expresa como una aberración longitudinal (distancia al eje óptico).

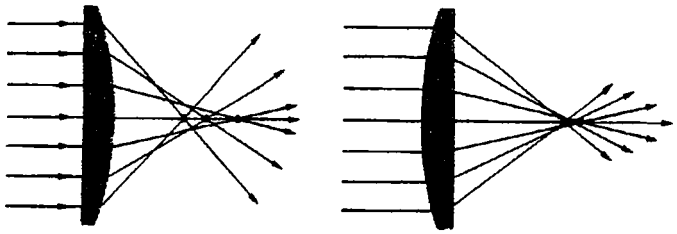
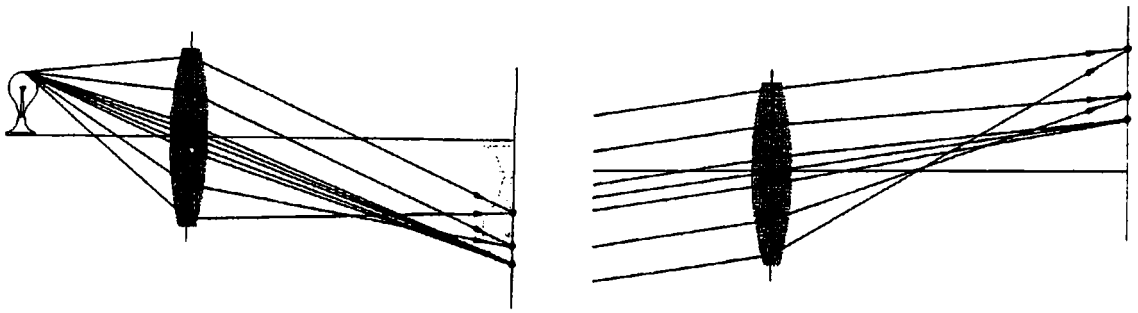


Figura 2.1.5. Aberración esférica.

Coma

En el caso en el que la lente está completamente corregida del fenómeno de aberración esférica, otros tipos de aberraciones aparecen en los puntos no pertenecientes a los ejes, la coma es una de ellas.

Esta aberración se produce por el hecho de que los rayos incidentes bajo un cierto ángulo respecto al eje de la lente, no convergen en un solo punto del plano de imagen sino que forman una imagen alargada de tipo cometa. Este tipo de aberración disminuye mucho cuando la superficie de incidencia de los rayos sobre la lente es pequeña.



a) b)

Figura 2.1.6. a) Coma negativa. b) Coma positiva.

Astigmatismo

El astigmatismo es una aberración que se manifiesta en los rayos oblicuos a causa de la asimetría de la refracción en las distintas secciones de la lente donde inciden los rayos de luz. Esta aberración provoca que los rayos procedentes de un punto que no esté situado sobre el eje de la lente, no converjan en un solo punto de la imagen, sino que forman una elipse.

La forma de estas elipses depende de la posición del plano de la imagen. Cuando éste está situado para que los puntos del eje de la lente se hallen enfocados aparecerán las distintas formas de la elipse para los puntos alejados de dicho eje.

La disminución de la apertura de la lente conduciendo a un aumento en la profundidad de enfoque, absorbe en cierta medida el problema del astigmatismo, no corrigiéndolo de una manera completa.

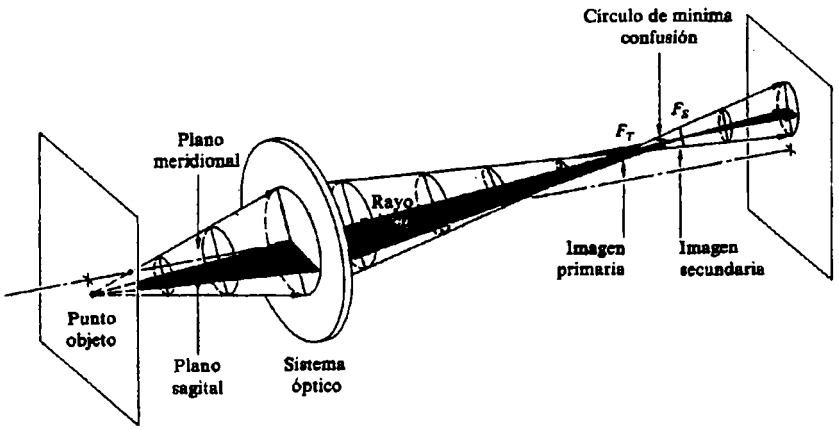


Figura 2.1.7. Astigmatismo.

Curvatura de Campo

La curvatura de campo es el error de una lente que enfoca a un objeto plano como una imagen plana. Por razones de simetría, se puede ver que las imágenes de los puntos del infinito estarán situados en una superficie esférica concéntrica con la lente esférica.

Esta aberración ocasiona que cuando el centro de la imagen esté enfocado, se encuentren desenfocados los bordes y viceversa. El efecto de esta aberración se ve disminuido cuando la superficie de la lente es pequeña.

Al igual como ocurre con el astigmatismo, la curvatura de campo puede ser absorbida por la disminución de la abertura de la lente y el aumento de la profundidad de enfoque.

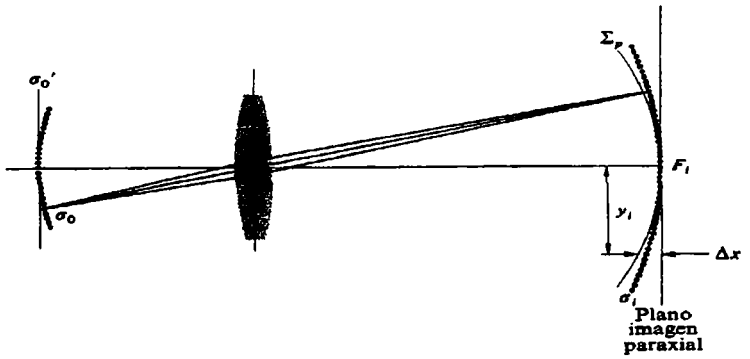


Figura 2.1.8. Curvatura de campo.

Distorsión

El efecto de distorsión provoca una deformación de la imagen, pero manteniendo todos los puntos enfocados. La distorsión de la imagen es debida al distinto aumento de ésta en virtud de la distancia de los rayos incidentes a la lente respecto al centro de ésta. Este hecho provoca dos tipos de efectos: distorsión positiva y distorsión negativa. El efecto de la distorsión positiva se manifiesta mediante la formación de una imagen del tipo cojín a partir de un objeto cuadrado.

El efecto de distorsión negativa se manifiesta mediante la formación de una imagen de tipo barril a partir de un objeto cuadrado. La cuantificación de estas distorsiones se efectúa de manera porcentual respecto a las dimensiones del cuadrado. Para minimizar el fenómeno de la distorsión, se recurre a la utilización de un diafragma situado en medio de dos elementos ópticos idénticos [8].

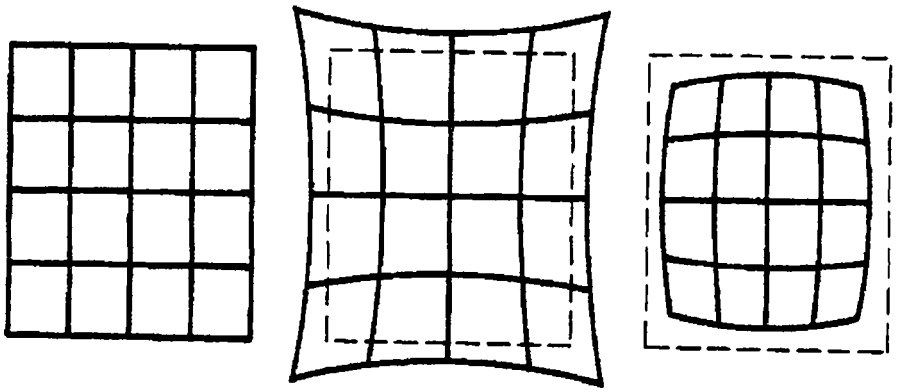


Figura 2.1.9. a) Imagen original. b) Distorsión positiva o tipo cojín. c) Distorsión negativa o tipo barril.

2.1.2 ADQUISICIÓN DE LA IMAGEN

Existen dos elementos fundamentales en la adquisición de la imagen. Primero el elemento físico sensitivo a un determinado espectro de energía electromagnética, como lo es el visible, rayos X, ultravioleta, o infrarrojo, y que será capaz de producir una salida proporcional al nivel de energía detectada. El segundo, llamado digitalizador, es un dispositivo que convierte la salida analógica de los elementos sensores a un formato digital.

Uno de los dispositivos mayormente utilizados para la adquisición de imágenes es la cámara de televisión, que puede ocupar como elemento sensor un tubo como el vidicón o bien un conjunto de elementos de estado sólido, como lo son los dispositivos de carga acoplada (CCD). Estos arreglos de estado sólido se presentan en dos diferentes configuraciones: sensores de exploración de línea y sensores de área. El sensor explorador de línea está compuesto por una hilera de elementos de silicio llamados *photosites*, y producen una imagen bidimensional por el movimiento relativo entre la escena y el detector. Por ejemplo, este tipo de sensores son utilizados en los scanners de página completa. Un

sensor de área está compuesto por una matriz de *photosites* y es capaz de capturar una imagen de la misma forma en la que lo hace un vidicón. La principal ventaja que existe entre los de estado sólido y el vidicón es la elevada velocidad de disparo de los primeros (alrededor de 1/30,000 segundos).

En el mercado pueden encontrarse sensores de exploración de línea con una resolución que va desde 256 hasta 4096 elementos. En tanto que para sensores de área los de baja resolución son del orden de 256 x 256 elementos y los de muy alta resolución llegan hasta 2048 x 2048 elementos [5]. Cabe destacar además que existen cámaras en blanco y negro y en color. Las de blanco y negro se adecuan más a aplicaciones industriales, y muchas veces presentan mejores características en resolución, relación señal a ruido, sensibilidad y contraste que las de color de precio similar. En las de color la mejor opción es la que se basa en tres dispositivos individuales para detectar los colores fundamentales, rojo, verde y azul. La salida RVA de los sensores es considerada superior a los estándares de formato NTSC y YC, ya que la información de color se presenta en tres señales separadamente.

A través del texto se utilizará $f(x,y)$ para indicar la imagen bidimensional obtenida del elemento sensor, donde x e y indican las coordenadas espaciales y el valor de f en cualquier punto (x,y) es proporcional al brillo (intensidad) de la imagen en ese punto. La digitalización de las coordenadas espaciales (x,y) se denomina muestreo de la imagen, en tanto que a la digitalización de la amplitud se la llama cuantización de intensidad o nivel de gris.

2.1.3 ALMACENAMIENTO

El método de almacenamiento que sea necesario adoptar, estará condicionado por algunas de las siguientes características: el tipo de proceso al que será sometida la información almacenada, el tiempo de acceso a dicho medio, la cantidad de accesos a realizar y el costo relativo de la información.

Uno de los métodos de proveer almacenamiento por corto tiempo es la memoria de la computadora. Otro es a través de tarjetas especializadas, llamadas frame buffers o frame grabbers. Estas tarjetas se enchufan en una de las ranuras disponibles en la computadora y proveen una suficiente cantidad de memoria como para almacenar una o varias imágenes completas. Además, de acuerdo a la complejidad del desarrollo de la tarjeta, permiten realizar operaciones de zoom, desplazamientos verticales u horizontales, y hasta otra serie de operaciones pertenecientes al procesamiento de imágenes digitales. Sus capacidades de memoria van desde 256 Kbytes hasta 256 Mbytes.

Existe otra categoría de tipo de almacenamiento, donde la característica principal es el frecuente acceso a los datos, por ello se les denomina almacenamiento en línea. Pueden ser discos Winchester de unos cientos de Megabytes, o discos magneto-ópticos con capacidades de hasta un Gigabyte en 5¼ de pulgada. Existen también unidades que contienen hasta 30 ó 100 de estos discos, denominados en inglés Jukeboxes, permitiendo una solución a almacenamientos de gran escala y con posibilidades de lectura y escritura en línea.

Finalmente los medios de almacenamiento masivos para los casos en que pocas veces será necesario acceder a los datos. Cintas magnéticas y discos ópticos son medios usuales para este tipo de almacenamiento. Existe una tecnología de discos ópticos que se escriben solo una vez y permiten múltiples lecturas (WORM de sus siglas en inglés), con una vida útil de hasta 30 años. La capacidad de almacenamiento de éstos varía de acuerdo a su tamaño, llegando hasta 10 Gigabytes en discos de 14 pulgadas. Una vez escritos estos discos se comportan como si fuera un acceso en línea.

2.1.4 PROCESAMIENTO

Con relación a la magnitud del problema a resolver en el procesamiento de imágenes digitales, encontraremos sistemas de super-computadoras dedicadas, hasta soluciones en una

computadora personal. La mayoría de las funciones de procesamiento de imágenes se pueden implementar a través de programas. La única razón de la existencia de un hardware especializado para realizar dicho procesamiento es la velocidad de proceso o superar alguna limitación importante de la computadora [6].

Actualmente el hardware agregado a las computadoras consiste en tarjetas digitalizadoras y buffers, las que muchas veces cuentan con unidades aritmético/lógicas (ALU de las siglas en inglés) que realizan operaciones aritméticas y lógicas a la velocidad de los cuadros, pudiendo también incorporar algoritmos de procesamiento aliviando así la carga de trabajo de la computadora.

2.1.5 PRESENTACION

Los monitores monocromáticos y en color son los principales dispositivos utilizados en sistemas de procesamiento de imágenes digitales para llevar a cabo la presentación. La salida para los monitores las proveen las cámaras en la mayoría de los casos, sino pueden tomarse de las tarjetas digitalizadoras. Puede tomarse la señal de salida de vídeo para alimentar un equipo que grabe las imágenes y luego reproducirlas, ya sea en diapositivas, transparencias o fotografías. Otros dispositivos como impresores láser, de chorro de tinta puede utilizarse para producir una copia en escala de gris con una resolución razonable.

2.2 FUNDAMENTOS Y CARACTERIZACIÓN DE IMÁGENES DIGITALES

La etapa de pre-procesamiento se puede entender como aquella que inicia con la señal analógica de salida de los sensores de la imagen, realiza el muestreo de la imagen, la cuantización de la intensidad, digitalización de la amplitud y modificación del contraste.

2.2.1 MUESTREO Y CUANTIZACIÓN

El proceso de muestreo de una imagen es el proceso de aplicar una grilla de dos dimensiones a una imagen continua espacialmente para dividirla en elementos de una matriz de dos dimensiones. En la figura 2.2.1 se puede ver una imagen muestreada, que contiene un total de *MxN* elementos utilizando una grilla rectangular.

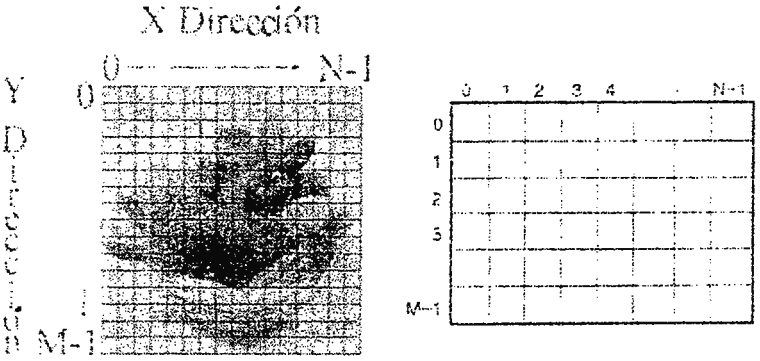


Figura 2.2.1. Imagen muestreada con una rejilla de *MxN* elementos.

Cualquier tipo de grilla se puede utilizar para caracterizar la imagen digital. El elemento fundamental de una imagen de este tipo se denomina pixel. El valor de cada pixel corresponde al promedio de la intensidad de la imagen espacial continua contenida o cubierta por dicho pixel. La siguiente matriz tiene la forma de una imagen digital.

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & f(0,M-1) \\ f(1,0) & f(1,1) & F(1,M-1) \\ & & \\ f(N-1,0) & f(N-1,1) & f(N-1,M-1) \end{bmatrix} \qquad (2.2\text{-}1)$$

Es usual que los valores de M y de N sean potencias exactas de 2, es decir:

$$N = 2^n \qquad M = 2^k \qquad (2.2\text{-}2)$$

Tamaños típicos utilizados para las representaciones matriciales son los siguientes: 256x256, 320x240, 512x512, 640x480 (NTSC o RS 170), 1024x768, 1024x1024, 2048x2048 y 4096x4096. En la figura 2.2.2 se presenta el resultado de tomar diferentes tamaños de matriz para representar la misma imagen [5].

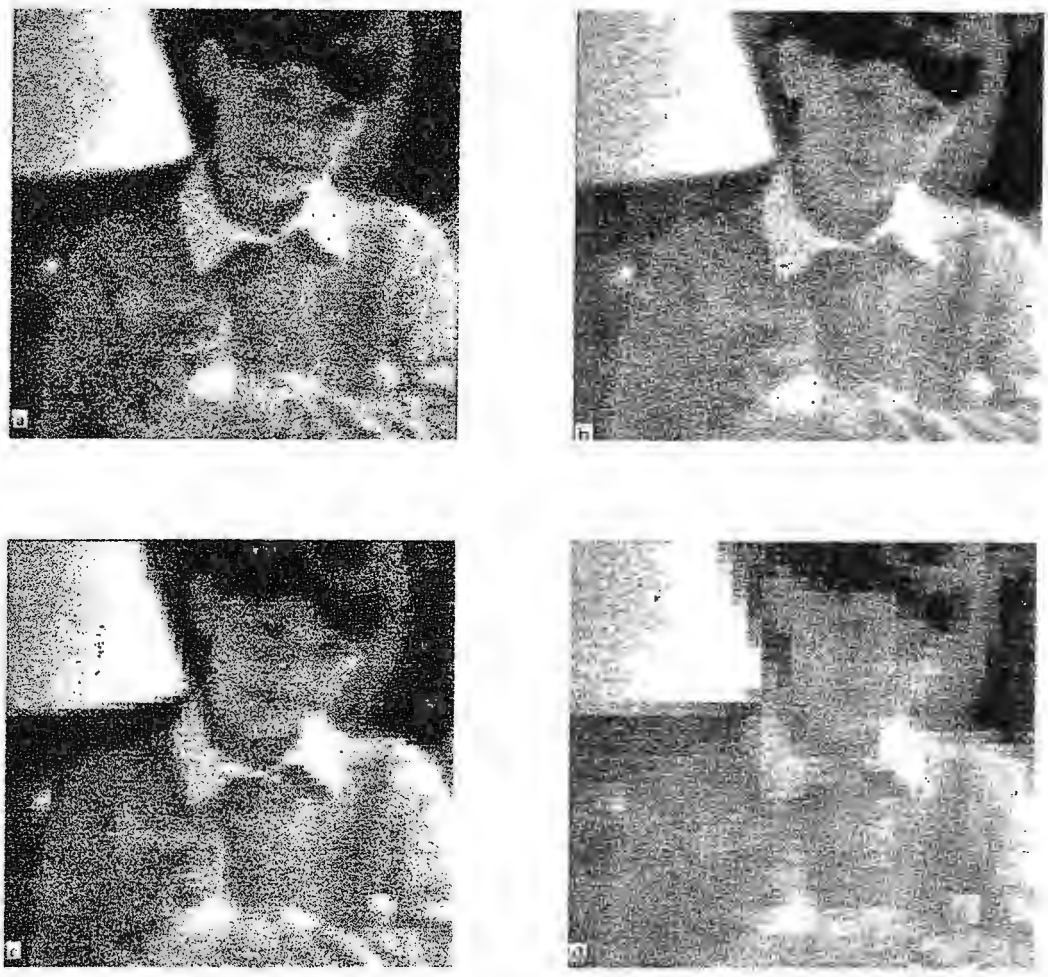


Figura 2.2.2. Efectos de la reducción de la rejilla muestreadora.

Se asume que los niveles de gris están igualmente distanciados en la totalidad de la escala de gris, es decir, la cuantización es uniforme. El proceso de digitalización convierte un valor de intensidad analógico en otro digital que es representativo del nivel de intensidad de la imagen. La calidad del proceso estará íntimamente ligada con la cantidad de dígitos utilizados en la conversión, ocho bits es una medida usual, asignándose así 256 niveles de

gris. Se mostrará a continuación el resultado de variar la cantidad de estados discretos para el mismo rango dinámico de entrada, así lo muestra la figura 2.2.3.

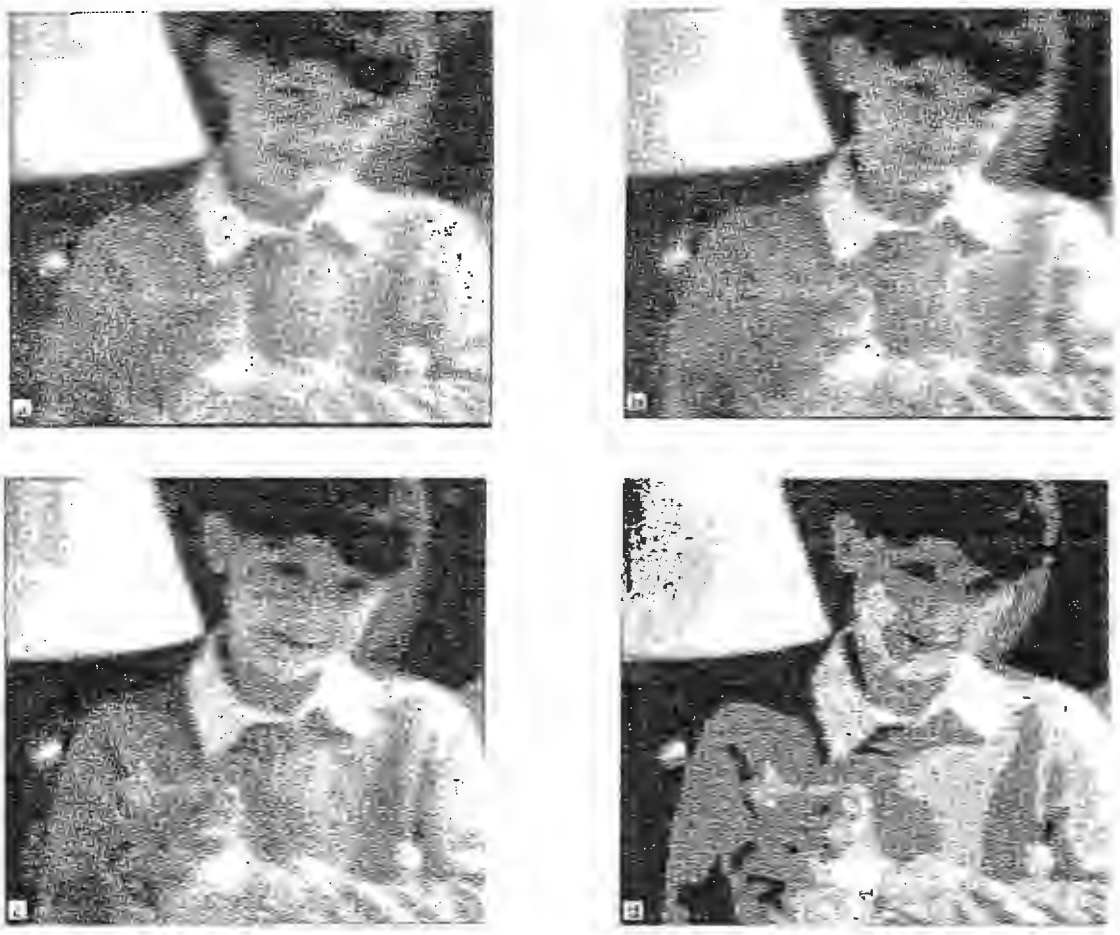


Figura 2.2.3. Representación de una imagen con diferentes niveles de gris.

A estas alturas podemos realizar una apreciación acerca de la resolución requerida para un sistema de procesamiento de imágenes. La resolución está fuertemente ligada a dos parámetros, la cantidad de muestras, que se puede asumir para el caso una matriz cuadrada de N x N elementos, y la cantidad de niveles de gris m. Digamos que entre una fotografía y una imagen de 1024 x 1024 con 256 niveles de gris, el ojo humano no sería capaz de encontrar diferencia alguna. Perceptible empezaría a ser un pequeño cambio en la granularidad de la imagen digital en 512 x 512, claro que siempre es relativo al tamaño de los objetos. Ya para 256 x 256 se comienza a apreciar una impresión cuadriculada en los bordes de la figura. La resolución necesaria en todo sistema de procesamiento de imágenes

está íntimamente relacionada con la aplicación que se desee y a las exigencias de precisión involucradas en el sistema [6].

2.2.2 ALGUNAS RELACIONES ENTRE PIXELES

Vecinos de un pixel

Un pixel p cualquiera de coordenadas (x,y) posee cuatro vecinos, dos horizontales y dos verticales con las siguientes coordenadas:

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

A este conjunto de pixeles se los denota por $N_4(p)$. Cada uno de estos vecinos se encuentra a una distancia unitaria de p en (x,y), y pudiera darse el caso de que algunos de ellos no existieran, como en los casos en que p fuera frontera de la imagen [5].

Los cuatro vecinos diagonales tienen las siguientes coordenadas

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

y se los denota $N_D(p)$. Este conjunto unido al otro se los denomina $N_8(p)$.

Conectividad

La conectividad entre pixeles es un concepto importante utilizado para establecer los límites de objetos o regiones en una imagen. Es necesario establecer cierta relación entre dos pixeles para concluir a cerca de su conectividad.

Sea V un conjunto de valores que representan niveles de gris utilizados para definir la conectividad. Por ejemplo, en una imagen de escala de gris, se podrían considerar valores de

intensidad desde 64 hasta 128 para establecer la conectividad. El conjunto V = {64, 65, 66,..., 127, 128} estaría siendo considerado para el ejemplo. Entonces se pueden definir tres tipos de conectividad:

a) conectividad -4. Dos pixeles p y q con valores de V conectados en 4 si q pertenece a $N_4(p)$.

b) conectividad -8. Dos pixeles p y q con valores en V están conectados en 8 si q pertenece a $N_8(p)$.

c) conectividad -*m* (mixta). Dos pixeles p y q con valores en V están conectados en m si:

I. q está en $N_4(p)$, o

II. q está en $N_D(p)$ y el conjunto $N_4(p) \cap N_4(q)$ es vacío (Es decir, el conjunto de pixeles que son $N_4(p)$ y $N_4(q)$ con valores en V).

La conectividad mixta busca eliminar las conexiones con múltiples caminos al utilizar conectividad-8. La figura que sigue presenta un ejemplo de lo que se ha descrito.

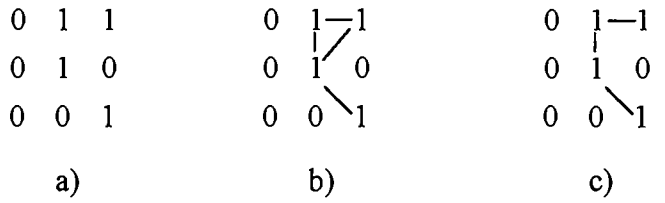


Figura 2.2.4. Conectividad: a) Arreglo, b) Vecindad-8, c) Vecindad-*m*.

Un pixel p es adyacente a otro q cuando están conectados. Se puede definir entonces adyacencia -4, -8, ó -*m* dependiendo del tipo de conectividad especificada. Dos subconjuntos de una imagen S_1 y S_2 son adyacentes si un pixel de S_1 es adyacente a un pixel de S_2.

Un camino desde un pixel p con coordenadas (x,y) al pixel q con coordenadas (s,t), es una secuencia de distintos pixeles con coordenadas:

$$(x_0,y_0),\ (x_1,y_1),\ (x_2,y_2),\ldots,\ (x_n,y_n)$$

en donde $(x_0,y_0) = (x,y)$ y $(x_n,y_n) = (s,t)$, (x_i,y_i) es adyacente a $(x_i\text{-}1,y_i\text{-}1)$, $1 < i < n$, y n es la longitud del camino. Se puede definir caminos -4, -8 ó -m, según sea la adyacencia especificada.

Si p y q son pixeles de un subconjunto S, luego p está conectado a q si existe un camino entre p y q que tiene todos sus pixeles en S. Para cualquier pixel p en S, el conjunto de pixeles en S que están conectados a p se lo denomina componente conectado.

Distancia

Dados tres pixeles p, q y z, con coordenadas (x,y), (s,t) y (u,v) respectivamente, se puede definir una función distancia D si:

a) $D(p,q) \geq 0$ ($D(p,q) = 0$ si $p = q$).
b) $D(p,q) = D(q,p)$.
c) $D(p,z) \leq D(p,q) + D(q,z)$

Las funciones de distancia usadas comúnmente son:

(1) La distancia Euclídea entre p y q, que se define así:

$$D_c(p,q) = \sqrt{(x-s)^2 + (y-t)^2} \qquad (2.2\text{-}3)$$

Para este caso de medida de distancia, todos los pixeles que se encuentren a una distancia menor o igual a r respecto de (x,y) estarán contenidos en un círculo de radio r centrado en (x,y).

(2) La distancia definida de la siguiente manera entre dos pixeles p y q:

$$D_4(p,q) = |x-s| + |y-t| \qquad (2.2\text{-}4)$$

En este caso el pixel que se ubique a una distancia menor o igual a D_4 estará contenido en un rombo o diamante centrado en (x,y).

```
      2
    2 1 2
  2 1 0 1 2
    2 1 2
      2
```

Se presenta el caso de un rombo asociado a una distancia $D_4 \leq 2$. Nótese que todos los pixeles con distancia $D_4 = 1$, constituyen $N_4(p)$.

(3) La distancia D_8 entre dos pixeles se define como:

$$D_8(p,q) = máx\ (|x\text{-}s| + |y\text{-}t|) \qquad (2.2\text{-}5)$$

Para este caso los pixeles con una distancia D_8 respecto de (x,y), menor o igual a un cierto valor r, forman un cuadrado centrado en dicho punto.

```
2 2 2 2 2
2 1 1 1 2
2 1 0 1 2
2 1 1 1 2
2 2 2 2 2
```

También puede apreciarse que los pixeles con una distancia $D_8=1$, constituyen $N_8(p)$.

Operaciones Aritméticas y Lógicas

Estas operaciones son ampliamente utilizadas en el procesamiento de imágenes. Las operaciones aritméticas entre dos pixeles p y q se denotan de la siguiente manera:

Suma: $p + q$

Resta: $p - q$

Multiplicación: $p * q$ (también $p \times q$)

División: p / q

Éstas se llevan a cabo pixel a pixel. La principal aplicación de la suma es para realizar promedios y reducir el ruido. La diferencia se aplica en imágenes médicas para quitar la información estática de fondo. La multiplicación o división se aplican cuando se desea corregir las sombras que resultan de una iluminación no uniforme, por ejemplo.

Las operaciones lógicas también tienen lugar en el procesamiento de imágenes. Las principales operaciones son: AND, OR y COMPLEMENTO, y se denotan así:

AND: pANDq (también $p \cdot q$)

OR: pORq (también $p + q$)

COMPLEMENTO: NOTp (también $\bar{p}$)

Las operaciones lógicas son fundamentales en el procesamiento de imágenes binarias, empleadas en tareas como enmascarar, detección de características, y análisis de formas. El conjunto de operaciones aritméticas y lógicas suele emplearse además en el procesamiento a través de grupos de vecinos.

2.2.3 GEOMETRIA DE IMÁGENES

Algunas Transformaciones Básicas

Comenzaremos definiendo unas transformaciones básicas que podrán ser concatenadas para obtener resultados más elaborados. Se aconseja la representación matricial para facilitar las operaciones y la representación gráfica de los conceptos.

Traslación

Se utilizará la notación cartesiana de tres dimensiones para el cual un punto genérico se denota como (X,Y,Z). Supongamos ahora que este punto se desea desplazar haciendo uso de los valores (X_0,Y_0,Z_0), por lo que se obtiene un nuevo punto con coordenadas

$$\begin{aligned} X^* &= X + X_0 \\ Y^* &= Y + Y_0 \\ Z^* &= Z + Z_0 \end{aligned} \tag{2.2-6}$$

En forma matricial las ecuaciones anteriores pueden expresarse así,

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2-7}$$

En el transcurso de esta sección utilizaremos la siguiente nomenclatura

$$\mathbf{v}^* = \mathbf{A}\,\mathbf{v} \tag{2.2-8}$$

donde **A** representa una matriz de transformación de 4 x 4, **v** es el vector columna que contiene las coordenadas originales

$$\mathbf{v} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2-9}$$

$$\mathbf{v}^* = \begin{bmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{bmatrix} \tag{2.2-10}$$

y **v*** es el vector columna cuyas componentes son las coordenadas transformadas. Con esta notación, la matriz utilizada para la traslación es

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2-11}$$

Escalado

Escalar con factores S_X, S_Y y S_Z a lo largo de los ejes X, Y y Z se obtiene de la siguiente matriz de transformación

$$\mathbf{S} = \begin{bmatrix} S_X & 0 & 0 & 0 \\ 0 & S_Y & 0 & 0 \\ 0 & 0 & S_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2-12}$$

Rotación

La forma más sencilla de rotación es la que se realiza alrededor de los ejes coordenados. Las rotaciones tridimensionales son inherentemente mucho más complejas que las que se presentan aquí, de hecho están fuera del alcance de este trabajo.

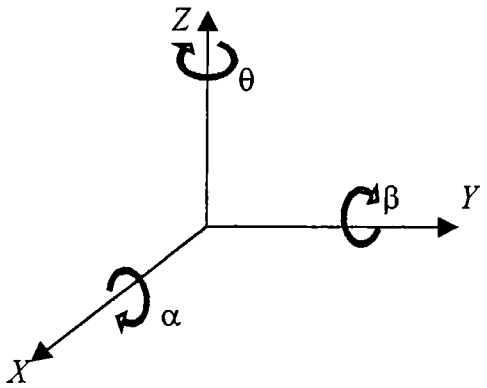


Figura 2.2.5 Rotación de un punto alrededor de los ejes coordenados. Los ángulos se miden en el sentido de las agujas del reloj cuando se mira por el eje hacia el origen.

Seguidamente se presentan las matrices de transformación para cada caso:

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & \text{sen}\,\theta & 0 & 0 \\ -\text{sen}\,\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2\text{-}13) \qquad \mathbf{R}_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \text{sen}\,\alpha & 0 \\ 0 & -\text{sen}\,\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2\text{-}14)$$

$$\mathbf{R}_\beta = \begin{bmatrix} \cos\beta & 0 & -\text{sen}\,\beta & 0 \\ 0 & 1 & 0 & 0 \\ \text{sen}\,\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2\text{-}15)$$

Transformación de Perspectiva

La transformación de perspectiva proyecta puntos en el espacio hacia un plano en dos dimensiones. Estas transformaciones nos presentan una manera de como las imágenes se forman al observar un objeto tridimensional. En la figura 2.2.6 se presenta un modelo básico de la formación de imágenes. El sistema de coordenadas de la cámara queda definido de manera tal que el plano *xy* coincide con el plano de la imagen y el eje óptico (siendo éste el que pasa por el centro de la lente) coincide con el eje *z*. De esta manera el centro del plano de la imagen es el origen de coordenadas espaciales y el centro de la lente tiene como coordenadas (0,0,λ), entonces λ puede entenderse como la longitud focal de la lente.

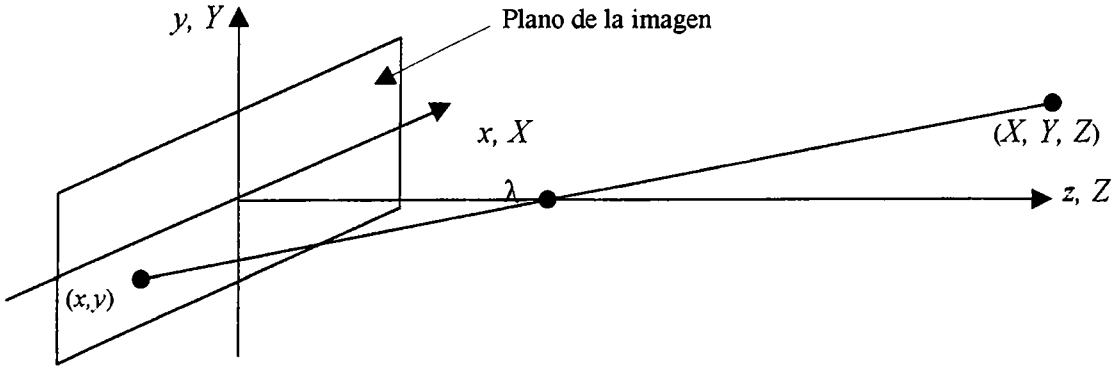


Figura 2.2.6. Modelo básico del proceso de formación de imágenes.

Consideremos un punto de una imagen tridimensional de coordenadas (X,Y,Z), tal que su sistema de coordenadas (X,Y,Z) es el absoluto y coincide con el de la cámara, como lo muestra la figura 2.2.6. Lo que se quiere obtener es una relación que nos dé las coordenadas (x,y) de la proyección del punto (X,Y,Z) en el plano de la imagen. Haciendo uso de los triángulos semejantes y observando la figura 2.2.6 se tiene que

$$\frac{x}{\lambda} = -\frac{X}{Z-\lambda} = \frac{X}{\lambda - Z} \qquad (2.2\text{-}16) \qquad\qquad \frac{y}{\lambda} = -\frac{Y}{Z-\lambda} = \frac{Y}{\lambda - Z} \qquad (2.2\text{-}17)$$

donde los signos negativos indican que los puntos de la imagen son invertidos, como puede observarse en la figura 2.2.6. Así de estas ecuaciones se obtienen directamente lo que se está buscando:

$$x = \frac{\lambda X}{\lambda - Z} \qquad (2.2\text{-}18) \qquad\qquad y = \frac{\lambda Y}{\lambda - Z} \qquad (2.2\text{-}19)$$

Claramente puede verse que estas ecuaciones no son lineales al aparecer la variable Z en el denominador de ambas.

Las coordenadas homogéneas de un punto, con coordenadas cartesianas (X,Y,Z), son (kX,kY,kZ,k), donde k es una constante arbitraria distinta de cero. Expresemos los resultados anteriores en forma matricial:

$$\mathbf{w} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (2.2\text{-}20) \qquad \mathbf{w_h} = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} \quad (2.2\text{-}21) \qquad \mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{bmatrix} \quad (2.2\text{-}22)$$

donde $\mathbf{w}$ es la representación de un punto en coordenadas cartesianas absolutas, $\mathbf{w_h}$ el equivalente en coordenadas homogéneas y $\mathbf{P}$ la matriz de transformación de perspectiva.

Entonces el producto $\mathbf{Pw_h}$ será un vector que indicaremos por $\mathbf{c_h}$, cuyos elementos serán las coordenadas de la cámara en forma homogénea.

$$\mathbf{c}_h = \mathbf{P}\mathbf{w}_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} = \begin{bmatrix} kX \\ kY \\ kZ \\ \frac{-kZ}{\lambda} + k \end{bmatrix} \qquad (2.2\text{-}23)$$

La componente z obtenida no tiene relevancia ya que la proyección se realiza sobre el plano (x,y). Esta variable actúa como una variable libre en la transformación de perspectiva inversa. De la expresión anterior se obtiene que

$$\mathbf{w}_h = \mathbf{P}^{-1}\,\mathbf{c}_h \qquad (2.2\text{-}24)$$

En donde resulta ser:

$$\mathbf{P}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{bmatrix} \qquad (2.2\text{-}25)$$

Si se tiene un punto de la imagen de coordenadas $(x_0,y_0,0)$, se obtiene al aplicar la transformación inversa el siguiente resultado

$$\mathbf{c_h} = \begin{bmatrix} kx_0 \\ ky_0 \\ 0 \\ k \end{bmatrix} \qquad (2.2\text{-}26) \qquad \mathbf{w_h} = \begin{bmatrix} kx_0 \\ ky_0 \\ 0 \\ k \end{bmatrix} \qquad (2.2\text{-}27)$$

Que obviamente no es lo que se esperaba, ya que al no conocerse la coordenada z del punto que generó esta imagen, esta variable desaparece del vector obtenido. Pero si tomamos la precaución de agregarla como variable libre, el resultado es otro

$$\mathbf{w_h} = \begin{bmatrix} kx_0 \\ ky_0 \\ kz \\ \frac{kz}{\lambda} + k \end{bmatrix} \tag{2.2-28}$$

Resultando las ecuaciones:

$$X = \frac{\lambda x_0}{\lambda + z}; \qquad Y = \frac{\lambda y_0}{\lambda + z}; \qquad Z = \frac{\lambda z}{\lambda + z}; \tag{2.2-29}$$

o bien despejando z de la última

$$X = \frac{x_0}{\lambda}(\lambda - Z); \qquad Y = \frac{y_0}{\lambda}(\lambda - Z) \tag{2.2-30}$$

2.2.4 MODELO DE LA CÁMARA

La ecuación (2.2-23) describe un modelo matemático sencillo de una cámara y parte de la suposición de que el sistema coordenado absoluto y la cámara coinciden.

Consideremos ahora un problema más general en el cual ambos sistemas se encuentren por separado. En la figura 2.2.7 se presenta esta situación, donde se emplea un sistema de coordenadas absoluto (X,Y,Z) para situar la cámara y los puntos tridimensionales (**w**). Además, dentro de la cámara puede observarse el plano de la imagen. La cámara puede tomar imágenes con ángulos de visión θ e inclinarse un ángulo α. Tómese el ángulo θ medido entre los ejes x y X y α entre z y Z. El vector $\mathbf{w_0}$ indica la distancia entre el origen de coordenadas absoluto y el soporte de la cámara, y el vector **r** denotará la distancia entre el soporte y el centro del plano de la imagen, de componentes (r_1,r_2,r_3).

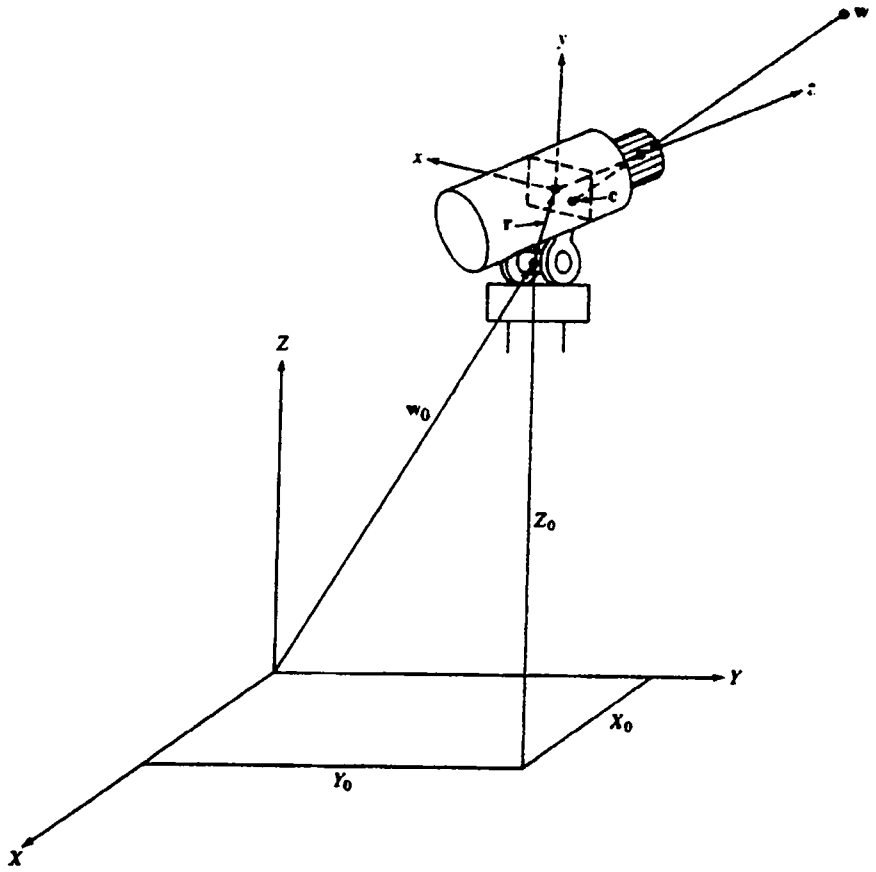


Figura 2.2.7. Geometría de formación de imágenes con dos sistemas de coordenadas.

Supongamos partir de un estado tal que la cámara se encuentra en una posición normal, en el sentido que el centro del soporte y el origen del plano de la imagen coinciden con el sistema de coordenadas absoluto y todos los ejes están alineados. Puede llegarse al esquema de la figura 2.2.7 de la siguiente forma: 1) desplazar el centro del soporte del origen, 2) girar el eje *x*, 3) inclinar el eje *z*, 4) desplazar el plano de la imagen respecto al centro del soporte.

Un punto en el espacio homogéneo que tenía coordenadas (X_0, Y_0, Z_0) está en el origen del nuevo sistema de coordenadas luego de aplicar la transformación $\mathbf{Gw_h}$, donde la matriz de transformación **G** que traslada el origen de coordenadas absoluto al centro del soporte es igual a

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2-31}$$

Para realizar la rotación del eje *x* se procede a emplear la ecuación (2.2-13), aplicando dicha matriz a todos los puntos del plano. Téngase presente que los ángulos se consideran positivos cuando los puntos se giran en sentido de las agujas del reloj, lo cual implica un giro en sentido contrario de la cámara respecto al eje *z*.

Ahora se hace necesario orientar los ejes *z* y *y* realizando una rotación alrededor del eje *x* un ángulo α, aplicando la matriz de transformación $\mathbf{R}(\alpha)$(incluido el punto $\mathbf{R}(\theta)\mathbf{Gw_h}$). Como consecuencia de ambas rotaciones se puede definir una sola matriz de rotación $\mathbf{R} = \mathbf{R}(\alpha)\,\mathbf{R}(\theta)$

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \operatorname{sen}\theta & 0 & 0 \\ -\operatorname{sen}\theta\cos\alpha & \cos\theta\cos\alpha & \operatorname{sen}\alpha & 0 \\ \operatorname{sen}\theta\operatorname{sen}\alpha & -\cos\theta\operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2-32}$$

Finalmente se concluye desplazando el origen del plano de la imagen mediante un vector **r**, al que le corresponde una matriz de transformación [5].

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2-33}$$

En este momento estamos en condiciones de realizar la transformación de perspectiva para obtener las coordenadas del plano de la imagen de un punto $\mathbf{w_h}$ del espacio homogéneo. La ecuación que representa la transformación de perspectiva aplicada a los dos sistemas de coordenadas resulta ser

$$\mathbf{c_h} = \mathbf{PCRGw_h} \tag{2.2-34}$$

En coordenadas cartesianas se pueden obtener las expresiones del punto de la imagen (x,y)

$$x = \lambda \frac{(X - X_0)\cos\theta + (Y - Y_0)\operatorname{sen}\theta - r_1}{-(X - X_0)\operatorname{sen}\theta \operatorname{sen}\alpha + (Y - Y_0)\cos\theta \operatorname{sen}\alpha - (Z - Z_0)\cos\alpha + r_3 + \lambda} \tag{2.2-35}$$

y

$$y = \lambda \frac{-(X - X_0)\operatorname{sen}\theta \cos\alpha + (Y - Y_0)\cos\theta \cos\alpha + (Z - Z_0)\operatorname{sen}\alpha - r_2}{-(X - X_0)\operatorname{sen}\theta \operatorname{sen}\alpha + (Y - Y_0)\cos\theta \operatorname{sen}\alpha - (Z - Z_0)\cos\alpha + r_3 + \lambda} \tag{2.2-36}$$

2.2.5 CALIBRACIÓN DE LA CÁMARA

En la sección anterior se pudo determinar las ecuaciones explícitas para las coordenadas de imagen (x,y) de un punto del espacio **w**. Para poder emplear dichas ecuaciones es necesario conocer la distancia focal, la posición en la que se halla la cámara y los ángulos de panorama e inclinación. Puede utilizarse un método de trabajo que requiere del conocimiento de las coordenadas espaciales de un conjunto de puntos de imagen. El procedimiento de cálculo usado para obtener los parámetros de la cámara es conocido como *calibración de la cámara.*

Partamos de la ecuación $\mathbf{c_h} = \mathbf{PCRGw_h}$ y hagamos $\mathbf{A} = \mathbf{PCRG}$ por lo que **A** incluye todos los parámetros de la cámara, y tomando $k = 1$ en la representación homogénea se puede escribir

$$\begin{bmatrix} c_{h1} \\ c_{h2} \\ c_{h3} \\ c_{h4} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2-37}$$

En las secciones anteriores se obtuvo que las coordenadas de la cámara venían dadas por

$$x = \frac{c_{h1}}{c_{h4}} \qquad y = \frac{c_{h2}}{c_{h4}} \tag{2.2-38}$$

Despejando c_{h1} y c_{h2} de estas dos últimas y reemplazándolas en la matriz y desarrollando el producto de ellas se obtiene (téngase en claro que para el desarrollo c_{h3} no se ha tomado en cuenta por estar relacionado con z)

$$a_{11}X + a_{12}Y + a_{13}Z - a_{41}xX - a_{42}xY - a_{43}xZ - a_{44}x + a_{14} = 0 \tag{2.2-39}$$

$$a_{21}X + a_{22}Y + a_{23}Z - a_{41}yX - a_{42}yY - a_{43}yZ - a_{44}y + a_{24} = 0 \tag{2.2-40}$$

El procedimiento consistirá en: 1) obtener m mayor o igual a 6 puntos del espacio con coordenadas conocidas (X_i, Y_i, Z_i) y con i=1,2,...,m, 2) obtener la imagen de estos puntos con la cámara en una posición dada para corregir los correspondientes puntos de imagen (x_i, y_i), i=1,...,m y 3) usar estos resultados en las ecuaciones (2.2-39) y (2.2-40) para calcular los coeficientes desconocidos.

2.3 PROCESAMIENTO Y EXTRACCIÓN DE CARACTERÍSTICAS

Una de las herramientas básicas de las que hace uso el procesamiento y la extracción de características en una imagen es la matemática. En ciertos casos es necesario realizar modificaciones de las imágenes mediante la aplicación de algoritmos que permitan visualizar características de las mismas, tal es el caso de la extracción de bordes, transformaciones morfológicas, etc. En otros casos es importante obtener un realce de las características que se encuentran contenidas al interior de la imagen, para ello se utiliza la manipulación de contraste y brillo así como también la ecualización de histograma entre otras técnicas.

2.3.1 TÉCNICAS DE HISTOGRAMAS

La técnica de histograma se implementa con el propósito de realzar la brillantez y el contraste de las imágenes. Con esta técnica se logra que una imagen oscura y con bajo contraste incremente su brillantez y contraste, y poder observar detalles de está que no eran visibles en la imagen original.

El histograma se representa como la distribución en los niveles de gris de una imagen. El histograma de una imagen de N x M se define como el porcentaje de pixeles que tienen un determinado nivel de gris:

$$h_i = \frac{n_i}{NM} \qquad \text{para } 0 \le i \le G_{max} \qquad (2.3\text{-}1)$$

Donde n_i es el número de pixeles a un nivel de gris i, NM es el total de pixeles dentro de la imagen y G_{max} es el valor del máximo nivel de gris de la imagen. Para una imagen de 256 niveles de gris, G_{max} = 255. Una propiedad importante de los histogramas es que la suma de los valores del histograma para cada nivel de gris dentro de la imagen es igual a uno:

$$\sum_{i=0}^{G_{max}} h_i = 1 \qquad (2.3\text{-}2)$$

Determinación de Brillo y Contraste por Medio de Histogramas

El histograma nos ayuda también a determinar la distribución de los niveles de gris dentro de una imagen.

A través de éste se puede calcular tanto el nivel promedio como la desviación estándar de los niveles de gris. El nivel promedio de los niveles de gris en términos de histograma es

$$avg = \sum_{i=0}^{G_{max}} i * h_i \qquad (2.3\text{-}3)$$

y su desviación estándar es

$$std = \sqrt{\sum_{i=0}^{G_{max}} i^2 * h_i - avg^2} \qquad (2.3\text{-}4)$$

El promedio obtenido de la ecuación (2.3-3) está directamente relacionado con la brillantez percibida, y la desviación estándar dada en la ecuación (2.3-4), da un estimado de la variación del nivel de gris promedio; lo que es conocido como el nivel de contraste de una imagen.

La figura 2.3.1 muestra cuatro histogramas de imágenes con diferentes características de brillantez y contraste. En el eje horizontal, los niveles de gris van desde 0 (nivel de negro) a G_{max} (nivel de blanco). La altura de cada línea da el valor del histograma (porcentaje de pixeles) a un nivel de gris *i*. Las figuras 2.3.1(a) y 2.3.1(b) muestran la diferencia en los histogramas de dos imágenes, una oscura y una clara. La amplitud de los histogramas describe la variación en niveles de gris de la imagen. Una mayor amplitud significa un mayor contraste, como se muestra en la figura 2.3.1(d), y por el contrario la menor amplitud de la figura 2.3.1(c) representa a una imagen de bajo contraste.

Ecuación de Histograma

Esta técnica es usada para realzar la brillantez y el contraste de una imagen. El objetivo de la ecuación de histograma es distribuir los niveles de gris en toda la imagen, de manera que cada nivel de gris tenga un porcentaje de aparición similar. En otras palabras la ecuación de histograma toma el histograma de una imagen, y produce una nueva imagen con un histograma uniformemente distribuido. Similar al histograma mostrado en la figura 2.3.1(d).

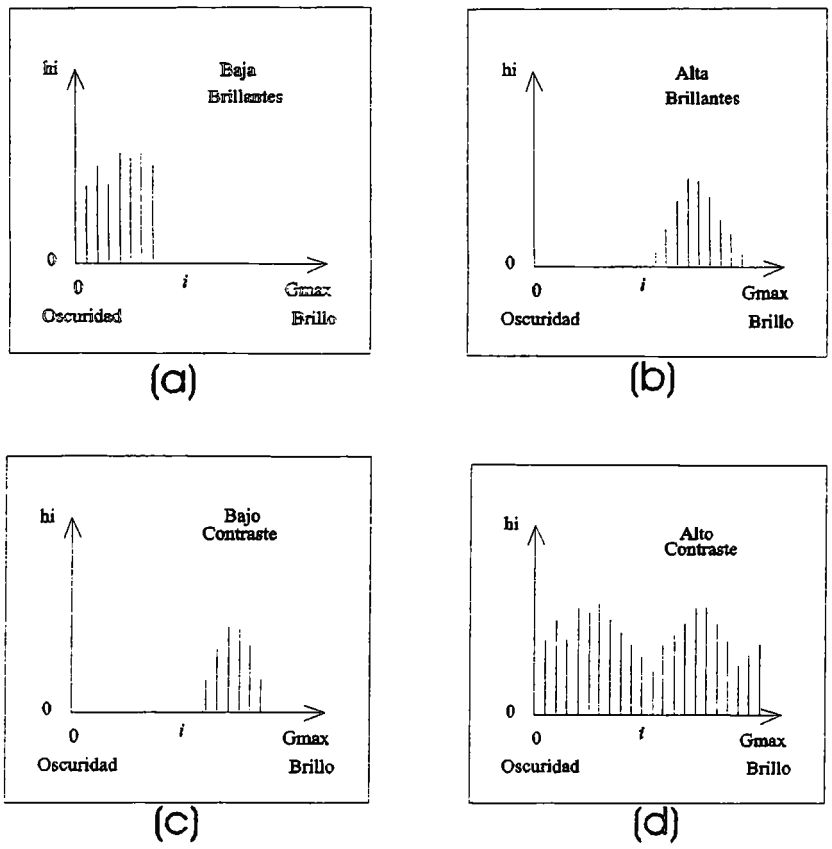


Figura 2.3.1. Histogramas de diferentes imágenes y niveles de gris: a) histograma de imagen oscura, b) histograma de imagen clara, c) histograma de imagen de bajo contraste, y d) histograma de imagen de alto contraste.

Para deducir la ecuación de histograma primero se usará el caso de una imagen continua y después se aplicará el resultado al caso discreto. Partiendo del hecho que las variables de intensidad a y b son cantidades aleatorias en el intervalo [0,1] y, como tales, pueden ser caracterizadas por sus funciones de densidad de probabilidad (FDP) $p_a(a)$ y $p_b(b)$. La función FDP es semejante al histograma de una imagen, la diferencia radica en que la FDP está referida a las imágenes en el caso continuo, y el histograma representa al caso discreto, el cual corresponde al procesamiento de imágenes digitales.

El proceso de la ecuación de histograma es encontrar una función de mapeo $M(a)$ que mapée una entrada FDP $p_a(a)$ a una salida FDP $p_b(b)$ la cual está uniformemente distribuida.

Supóngase que la variable a representa la intensidad de los pixeles en una imagen. Como se trata del caso continuo, a es una variable normalizada y continua en el intervalo $0 \leq a \leq 1$.

Para cualquier valor de a en el intervalo [0,1], usaremos transformaciones de la forma.

$$b = M(a) \tag{2.3-5}$$

Obteniéndose un valor de intensidad b para cada valor de intensidad de pixel a de la imagen de entrada.

Se supone que la función de transformación M satisface las siguientes condiciones:

1.- $M(a)$ no está multivaluada y es monótona creciente en el intervalo $0 \leq M(a) \leq 1$.

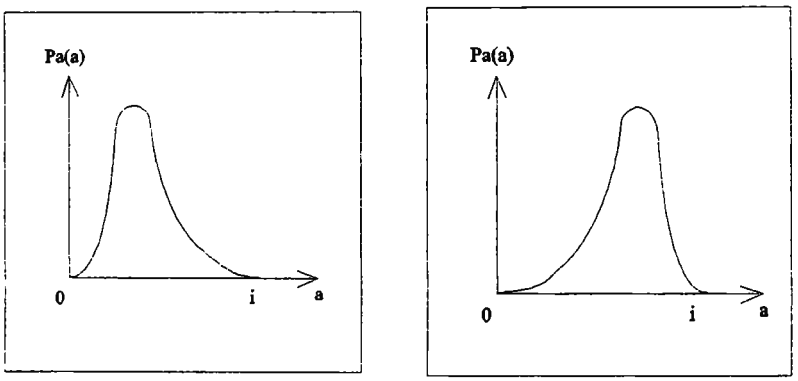
2.- $0 \leq M(a) \leq 1$ para $0 \leq a \leq 1$.

La condición 1 hace que se preserve el orden en la escala de intensidad (de negro a blanco), y la condición 2 garantiza que los valores de b que se obtienen con la función de mapeo son compatibles con el margen de valores de pixeles permitido (de 0 a 1).

La transformación inversa de b que satisface las dos condiciones enunciadas anteriormente tiene la forma

$$a = M^{-1}(b) \tag{2.3-6}$$

Se puede obtener mucha información sobre el aspecto general de una imagen usando su FDP de intensidad. Por ejemplo, una imagen cuyos pixeles tengan la FDP mostrada en la figura 2.3.2(a) sería bastante oscura debido a que la mayoría de los valores de intensidad de los pixeles estarían concentrados en el extremo oscuro de la escala de intensidad. Por el contrario, una imagen cuyos pixeles tuvieran una distribución de intensidad como la mostrada en la figura 2.3.2(b) sería predominante en tonos claros.



a) b)

Figura 2.3.2. a) Intensidad FDP de una imagen "oscura". b) Una imagen "clara".

Según la teoría elemental de las probabilidades, si $p_a(a)$ y $M(a)$ se conocen, y $M^{-1}(b)$ satisface la condición 1, entonces la FDP de las intensidades transformadas viene dada por

$$p_b(b) = \left| p_a(a) \frac{da}{db} \right|_{a=M^{-1}(b)} \tag{2.3-7}$$

Se desea tener una salida FDP uniforme, con una probabilidad igual para cada nivel de gris:

$$p_b(b) = \begin{cases} \dfrac{1}{Z-W}, & W \le b < Z \\ 0, & \text{Para cualquier otro valor} \end{cases} \tag{2.3-8}$$

En donde el parámetro Z es nivel de gris máximo que se desea obtener, W es el valor mínimo y Z - W es el rango de niveles de gris.

La ecuación (2.3-7) se puede volver a escribir de tal manera que las probabilidades diferenciales sean iguales:

$$p_a(a)da = p_b(b)db \tag{2.3-9}$$

Integrando ambos lados de la ecuación (2.3-9)

$$\int_{h\in A} p_a(h)dh = \int_{v\in B} p_b(v)dv \qquad (2.3.10)$$

donde A y B son los valores del rango sobre el cual está definido $p_a(a)$ y $p_b(b)$. Sustituyendo la ecuación (2.3-8) en la ecuación (2.3-10), y desarrollando la integración

$$\frac{b-W}{Z-W} = \int_0^a p_a(h)dh \quad \text{para } W \le b < Z \qquad (2.3\text{-}11)$$

y finalmente despejando b, nos da la función de mapeo requerida:

$$b = M(a) = (Z-W) \int_{h\in A} p_a(h)dh - W \quad para \quad W \le b < Z \qquad (2.3\text{-}12)$$

Para el procesamiento digital, el rango sobre el cual b está definida es entre 0 y G_{max}, así como también los niveles de entrada. Por lo tanto:

$$b = M(a) = G_{max} \int_0^a p_a(h)dh \qquad \text{para} \quad 0 \le b < G_{max} \text{ y } 0 \le a < G_{max} \qquad (2.3\text{-}13)$$

La ecuación (2.3-13) es la función de mapeo M(a) en términos de la integral de la FDP de entrada, pero ésta hace referencia al caso continuo. Para el caso discreto, la cual es una aproximación de esta ecuación, la función FDP de entrada se convierte en el histograma de la imagen original y la integral se transforma en una sumatoria:

$$b_k = M(a) = G_{max} \sum_{i=0}^{k} h_i \quad \text{para} \quad 0 \le k \le G_{max} \qquad (2.3\text{-}14)$$

Para desarrollar el método de igualación de histograma, primero debe obtenerse el histograma de la imagen, deben evaluarse estos valores sustituyéndolos en la ecuación (2.3-14), para obtener la función de transferencia. Después la imagen de entrada es explorada

pixel por pixel, aplicándole la función de mapeo y de esta forma obtener una nueva imagen cuyo histograma estará uniformemente distribuido.

2.3.2 DETECCIÓN DE BORDES

La detección de bordes es una de las tareas más importantes en un proceso de tratamiento de imágenes, sirviendo como paso inicial en el procesamiento previo a la segmentación y reconocimiento de objetos.

Un borde se define como una discontinuidad en los niveles de gris dentro de una imagen, y que delimita la frontera que separa el entorno y los objetos. Existen varios métodos para llevar a cabo la detección de bordes; el más común hace uso de los operadores derivada local (Gradiente y Laplaciano), y un segundo método es aplicar los filtros en el dominio de la frecuencia de Fourier.

2.3.2.1 DETECCIÓN DE BORDES USANDO OPERADORES DERIVADA LOCAL

Estas técnicas se basan en la aplicación de la primera y segunda derivada sobre los elementos de una imagen, con el objeto de detectar los cambios en los niveles de gris, de una región a otra y en una dirección particular. La figura 2.3.3, muestra la aplicación de este concepto. La parte (a) de esta figura muestra una imagen de un objeto claro sobre un fondo oscuro, donde la derivada con respecto a la posición (x,y) servirá como medio para detectar la presencia de un borde.

En la figura se muestra la primera y segunda derivada en la dirección x para la coordenada y = 64 de una imagen de 128x128 pixeles. Es una distribución del perfil

horizontal en los niveles de gris, haciendo la magnitud de la derivada proporcional a la agudeza del borde detectado. Para la primera derivada, se muestra que la magnitud de ésta llega a su máximo positivo, exactamente en el medio de la transición de oscuro a claro, volviendo a disminuir a cero en los niveles constantes de gris, hasta llegar a un mínimo negativo en el cambio de claro a oscuro. La segunda derivada es positiva para la parte de la transición asociada con el lado oscuro del borde, negativa para la parte asociado con el lado claro del borde y cero en las áreas de niveles constantes.

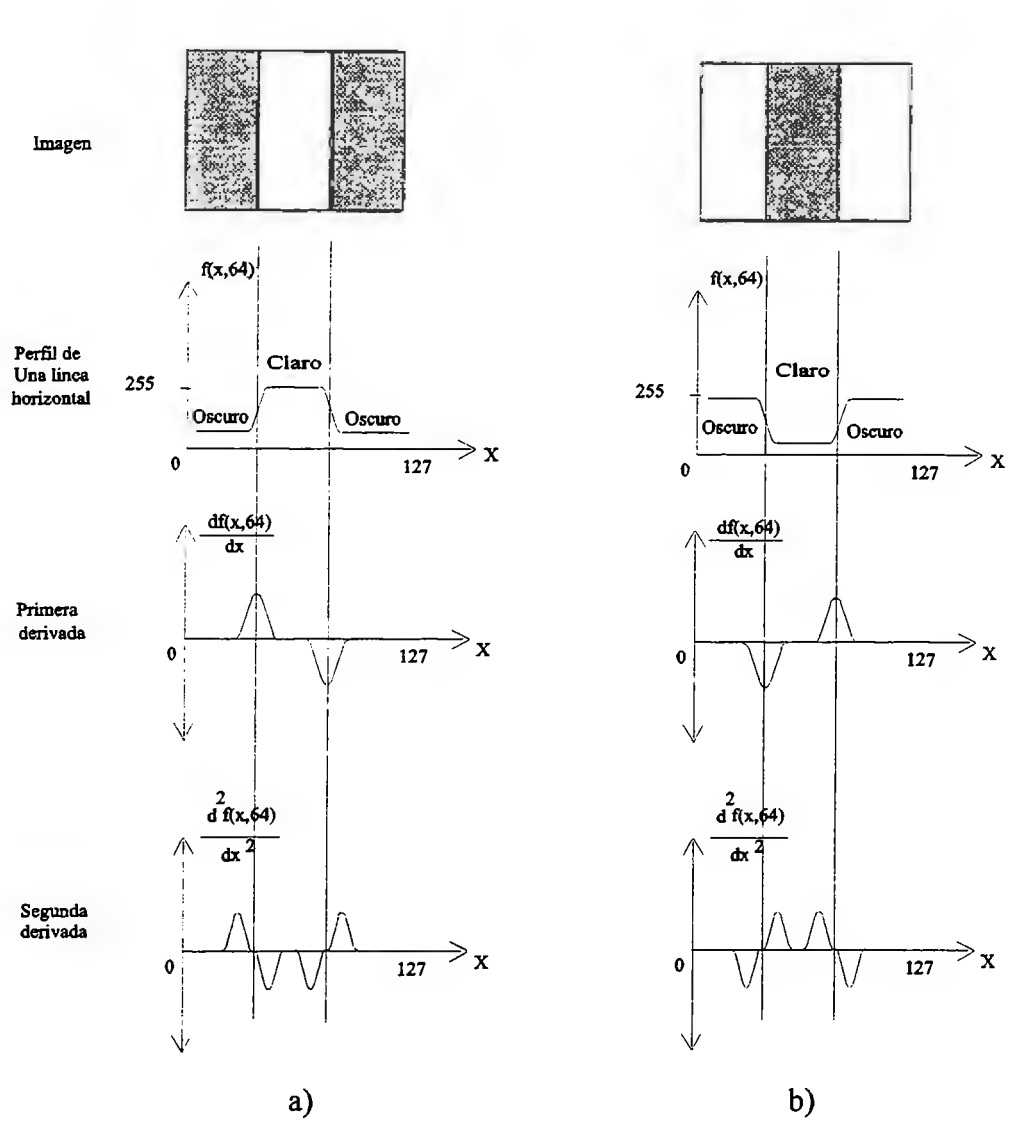


Figura 2.3.3. Elementos de detección de aristas por operadores de derivadas. a) Objeto claro sobre fondo oscuro, b) objeto oscuro sobre fondo claro.

A diferencia de la primera derivada, la cual produce un sólo pulso por borde, la segunda derivada produce dos pulsos; debido a está característica de la segunda derivada, se le utiliza a menudo para delimitar bordes. De estas dos derivadas, la segunda es más sensible a la presencia de un borde, pero también es más sensible al ruido.

A) Operador Gradiente

El análisis anterior se ha limitado a un perfil horizontal (dirección x de la imagen), pero se puede extender a cualquier dirección. Simplemente se definirá un perfil perpendicular a la dirección del borde en cualquier punto e interpretaremos los resultados como en el párrafo anterior. La primera derivada en cualquier punto de la imagen se obtiene empleando el valor del *gradiente* en ese punto.

El gradiente de una imagen $f(x,y)$, en un punto (x,y) se define como un vector en dos dimensiones.

$$\nabla f(x,y) = \begin{bmatrix} f_x(x,y) \\ f_y(x,y) \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} \tag{2.3-15}$$

Del análisis vectorial sabemos que el vector gradiente apunta en la dirección de la máxima razón de cambio de la función $f(x,y)$. En la detección de bordes, la magnitud de este vector es el que generalmente se conoce como el gradiente, denotado por ∇f, donde:

$$\nabla f = \text{mag}(\nabla f(x,y)) = \sqrt{fx(x,y)^2 + fy(x,y)^2} \tag{2.3-16}$$

y su dirección:

$$\theta(x,y) = \tan^{-1}\left[\frac{f_y(x,y)}{f_x(x,y)}\right] \tag{2.3-17}$$

Para simplificar la ecuación (2.3-16), eliminamos la raíz cuadrada y obtenemos la siguiente aproximación:

$$\nabla f = \text{mag}(\nabla f(x,y)) = |f_x(x,y)| + |f_y(x,y)| \qquad (2.3\text{-}18)$$

Para la evaluación del gradiente necesitamos obtener las derivadas de primer orden $\partial f/\partial x$ y $\partial f/\partial y$. Hay muchas maneras de obtener estas derivadas en una imagen digital. Un método es usar las diferencias de primer orden entre los pixeles adyacentes. En la figura 2.3.4, se muestran los ocho pixeles rodeando al pixel de la coordenada (x,y) y a continuación las diferencias entre pixeles y su máscara de filtro espacial.

$f(x-1,y-1)$ ***a***	$f(x,y-1)$ ***b***	$f(x+1,y-1)$ ***c***
$f(x-1,y)$ ***d***	$f(x,y)$	$f(x+1,y)$ ***e***
$f(x-1,y+1)$ ***g***	$f(x,y+1)$ ***h***	$f(x+1,y+1)$ ***i***

Figura 2.3.4. Los ocho pixeles vecinos del pixel $f(x, y)$.

$$f_x(x,y) = f(x,y) - f(x-1,y); \qquad (2.3\text{-}19)$$

$$f_y(x,y) = f(x,y) - f(x,y-1); \qquad (2.3\text{-}20)$$

Nota: El punto dentro de la máscara indica la posición $f(x, y)$ del pixel que se evalúa.

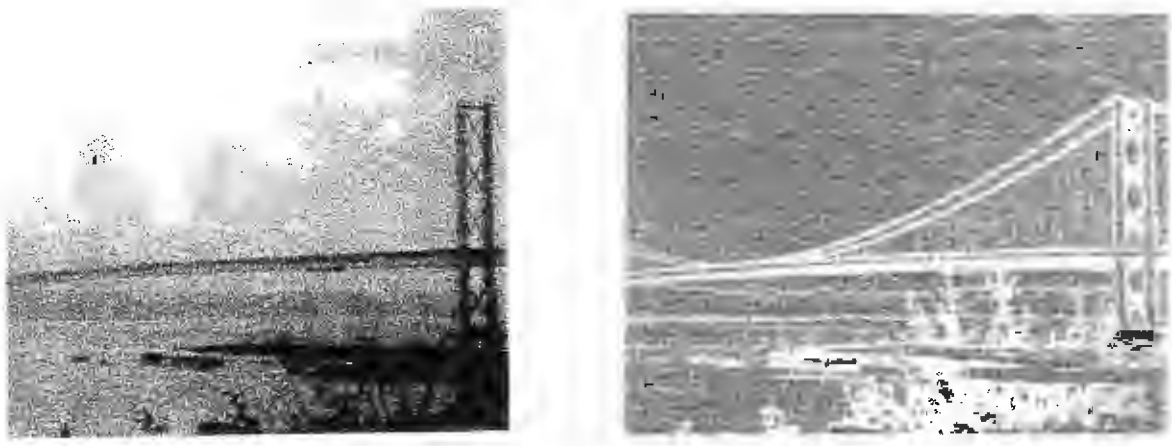
También puede usarse un entorno de vecindad de 3 x 3 pixeles centrados en $f(x, y)$, con lo que se obtiene una definición más compleja. A continuación se muestra cada derivada parcial y su respectivo operador (conocidos como operadores de Sobel).

$$f_x(x,y) = (c + 2e + i) - (a + 2d + g); \qquad \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad (2.3\text{-}21)$$

$$f_y(x,y) = (g + 2h + i) - (a + 2b + c); \qquad \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \qquad (2.3\text{-}22)$$

El detector de borde con el gradiente de Sobel, produce una mejor detección y es algunas veces insensible al ruido presente en la imagen. Esto se debe al promedio que se lleva a cabo durante la evaluación del gradiente.

En la figura 2.3.5 se presenta una imagen y la aplicación del detector de borde de Sobel. Nótese todos los detalles obtenidos por el detector, aun es notable la estructura de las nubes en el cielo.



a) b)

Figura 2.3.5. Ejemplo del uso del gradiente de Sobel para detección de bordes. a) Imagen original, b) Imagen de bordes.

B) Operador Laplaciano

El laplaciano se define como:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{2.3-23}$$

Al igual que el gradiente, el laplaciano puede ser obtenido a partir de las diferencias entre pixeles adyacentes. La definición junto con su máscara de 3 x 3 es:

$$\nabla^2 f(x,y) = 4f(x,y) - (d + e + b + h); \qquad \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array} \tag{2.3-24}$$

El filtro detector Laplaciano es usado junto con otros detectores como el de Sobel para determinar la dirección de los cambios en los niveles de gris, de claro a oscuro o viceversa.

2.3.2.2 DETECCIÓN DE BORDES EN EL DOMINIO DE LA FRECUENCIA DE FOURIER.

Esta técnica se implementa con el uso de un filtro pasa-altos. Primero se obtienen las componentes de la transformada de Fourier $F(n, m)$ de la imagen en escala de grises $f(x, y)$ y después estas componentes se filtran con un filtro pasa-altos, con una función de transferencia $H(n, m)$, para obtener la nueva imagen.

$$G(n,m) = F(n,m)H(n,m) \tag{2.3-25}$$

Finalmente se aplica la transformada inversa de Fourier, para producir la imagen de bordes detectados.

Un filtro comúnmente usado es el filtro pasa-altos Butterworth, con la función de transferencia:

$$H(\rho) = \frac{1}{\sqrt{1 + \left(\frac{\omega_c}{\rho}\right)^{2N}}} \qquad (2.3\text{-}26)$$

donde $\rho^2 = n^2 + m^2$. La figura 2.3.6 es una imagen de bordes detectados usando este filtro con $N = 4$ y $\omega_c = 60$.

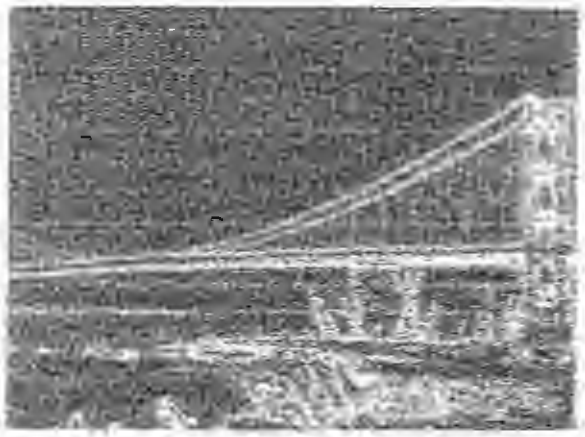


Figura 2.3.6. Imagen de bordes de la figura 2.3.5(a) obtenida usando el filtro pasa-altos Butterworth.

2.3.3 DETECCIÓN DE UMBRAL

La detección de umbral es el procedimiento mediante el cual se encuentran los pixeles frontera entre dos regiones con diferentes niveles de gris, y se distingue de la detección de bordes en que la diferencia en los niveles de gris entre las dos regiones, se define de antemano mediante un valor de intensidad, conocido como nivel de umbral T.

Umbralización de Bordes

A menudo la detección de umbral se lleva a cabo como un proceso posterior a la detección de bordes. Esta técnica sirve para destacar los bordes (determinar si un borde existe) y crear imágenes que puedan ser utilizadas directamente por los sistemas de segmentación y detección de objetos. Definamos la función *BORDE*(x, y) como una imagen de bordes detectados. La umbralización se aplica de la siguiente manera, dando como resultado una nueva imagen.

$$g_e(x,y) = \begin{cases} 0 & \text{Para BORDE}(x,y) \le K \\ 1 & \text{Para BORDE}(x,y) > K \end{cases} \qquad (2.3\text{-}27)$$

donde el valor de K determina la sensibilidad del detector de bordes, es decir el nivel de intensidad de borde que será resaltado. Para imágenes libres de ruido el umbral se determina de forma que todas las discontinuidades con un nivel de contraste mínimo sean consideradas bordes de la imagen. Con las imágenes ruidosas, la selección del valor de K dependerá de un compromiso, de tal manera que se pierda la menor cantidad de bordes válidos y se rechace la mayor cantidad de bordes falsos producto del ruido. Como resultado de la aplicación de la relación (2.3-27) se obtendrá una imagen en dos niveles de gris (imagen binaria), con los bordes en un nivel claro y todo lo demás oscuro.

Umbralización de Imágenes

Es una de las principales técnicas usadas por los sistemas de procesamiento para llevar a cabo la detección de objetos, especialmente en aplicaciones que requieren una cantidad elevada de datos.

La figura 2.3.7(c) representa el histograma de la imagen $f(x, y)$ de la figura 2.3.7(a), compuesta por un cuadro brillante sobre un fondo oscuro. Para poder separar ambos

elementos consideremos al nivel de gris T como valor de referencia, de tal manera que un valor de $f(x, y) > T$ será un punto del objeto; en caso contrario el punto será considerado parte del entorno o fondo de la imagen. La aplicación práctica de esta técnica es llevada a cabo por el rastreo de la imagen pixel por pixel, y luego designar a cada pixel como parte del objeto o del entorno, dependiendo si el nivel de gris es mayor o menor que el valor de T. La figura 2.3.7(b) es la imagen binaria resultante producto de la umbralización. Este criterio puede ser extendido como se muestra en el histograma de la figura 2.3.7(d), donde existen dos niveles de umbral $T1$ y $T2$, para considerar ahora tres regiones distintas dentro de la imagen.

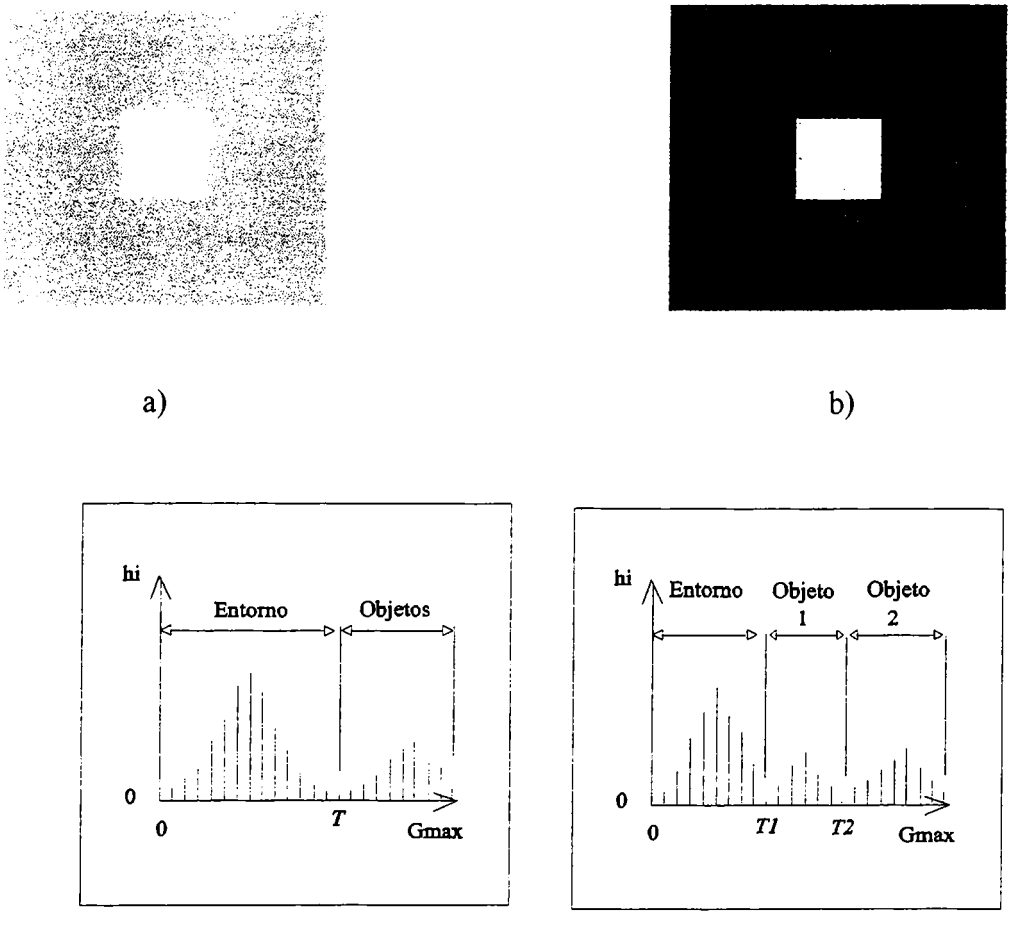


Figura 2.3.7. Ejemplo de umbralización. a) imagen original, b) imagen umbralizada, histogramas de intensidad divididos por: c) umbral único de la imagen de (a) y d) umbral múltiple

La determinación del nivel de umbral T puede llevarse a cabo considerando la siguiente función:

$$T = T[x, y, p(x, y), f(x, y)] \qquad (2.3\text{-}28)$$

donde $f(x,y)$ es la luminosidad del punto (x,y) y $p(x,y)$ es una propiedad local en ese punto; por ejemplo, la intensidad media de un entorno de vecindad centrado en (x,y). Cuando T depende sólo de $f(x,y)$, el umbral se llama global (fig. 2.3.7(a)). Si T depende tanto de $f(x,y)$ como de $p(x,y)$, entonces el umbral se llama local y si T depende de las coordenadas (x,y), se llama umbral dinámico [5].

Selección de Umbral Basados en las Características de Frontera

El umbral T es el elemento principal en la aplicación del método de detección de umbral, y los resultados obtenidos dependen de la buena elección de su valor. La figura 2.3.7(a) muestra que es mucho más sencillo la elección de este valor, si el histograma de la imagen es tal que permita distinguir las distintas zonas de una imagen. Esta figura por ejemplo, concentra la totalidad de los pixeles en dos zonas distintas, la del entorno y la del objeto, y ambas tienen formas similares, pico muy alto, extendida hacia los lados, simétrico y ambas zonas separadas por un valle.

Una forma fácil de trabajar con un histograma con tales características, es considerar primero el tipo de iluminación. Una imagen uniformemente iluminada tendrá un histograma, cuyas regiones serán fácilmente identificables [6]. Después de esto se procede a usar todos aquellos pixeles cercanos a la frontera o que formen parte de ella. Con esto se logra delimitar aún más la imagen procesada y hacer uso de una característica adicional de frontera. Para determinar si un pixel forma parte de la frontera, se utilizan los operadores gradiente y laplaciano, como la característica o propiedad adicional que se requiere al momento de llevar a cabo la detección de umbral. Estas dos cantidades se usan para formar una imagen de tres niveles como sigue:

$$s(x,y)=\begin{cases}0 & \text{si } \nabla f < T \\ + & \text{si } \nabla f \geq T \text{ y } \nabla^2 f \geq 0 \\ - & \text{si } \nabla f \geq T \text{ y } \nabla^2 f < 0\end{cases} \qquad (2.3\text{-}29)$$

donde los símbolos 0, + y – representan cualquier nivel distinto de gris, T es el umbral y además, el gradiente ∇f y el laplaciano $\nabla^2 f$ son evaluados en cada punto (x,y). Como puede notarse en la figura 2.3.3, un signo positivo del operador laplaciano representa a todos los pixeles asociados al lado oscuro del borde y un signo negativo hace referencia a los pixeles del lado claro del borde. Con esto se consigue profundizar más el valle entre la zona de contorno y los objetos, es decir hacer más evidente la separación entre ambas zonas.

2.3.4 FALSO COLOR

El Falso color o seudocolor es el mapeo punto a punto de una imagen en escala de gris, a una nueva imagen con características resaltadas en color, usando el modelo de color RVA (componentes rojo, verde y azul de las imágenes en color), que en inglés se representa por sus siglas RGB. El objetivo de este método es enfatizar a través del color, zonas o características de la imagen que interesan.

El objetivo del *falso color* es agregar color a las imágenes en escala de gris, y no debe confundirse con el proceso de *colorización* de una imagen en escala de gris. La *colorización* consiste en convertir cada objeto de la imagen a su color real, de tal manera que la imagen total pueda ser vista tal como sería sí estuviera a colores. El *falso color* asigna color a las imágenes en escala de gris, basándose en los niveles de gris presentes en la imagen sin considerar el color de los objetos originales.

Para llevar a cabo el falso color, la imagen debe ser mapeada para obtener las tres imágenes componentes RVA de la imagen en color $c(x,y)$. Estas componentes son el rojo $r(x,y)$, el verde $v(x,y)$ y el azul $a(x,y)$, y la información de cada una representa la

contribución de intensidad en el color correspondiente, en cada punto. El mapeo resulta en una nueva imagen de 24 bit usando 256 niveles de intensidad para cada una de las componentes RVA. La figura 2.3.8 muestra un diagrama de bloques del proceso de falso color.

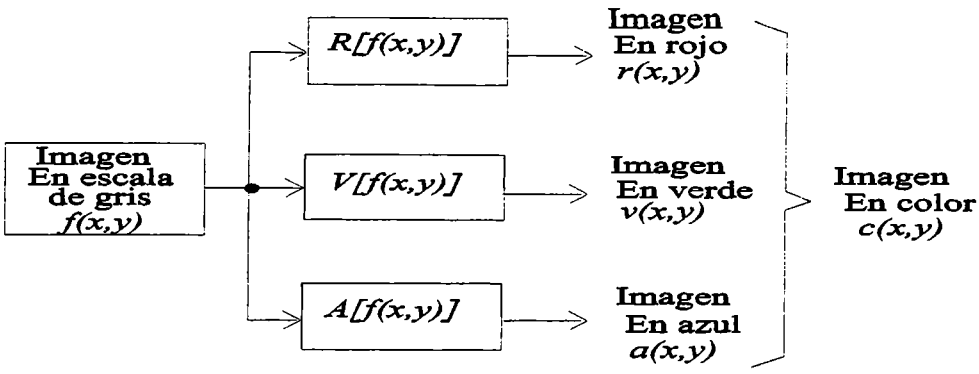


Figura 2.3.8. Diagrama de bloques del proceso de falso color.

A continuación se presenta una serie de relaciones que muestran la aplicación del falso color sobre una imagen en escala de gris $f(x,y)$.

1- La imagen resultante $c(x,y)$ es simplemente la imagen original en escala de gris.

$$c(x,y) = \begin{cases} R[f(x,y)] = f(x,y) \\ V[f(x,y)] = f(x,y) \\ A[f(x,y)] = f(x,y) \end{cases} \tag{2.3-30}$$

2- Imagen resultante con énfasis en un nivel de gris individual.

$$c(x,y) = \begin{cases} R[f(x,y)] = 0; V[f(x,y)] = 0; A[f(x,y)] = b_{max} & \text{para } f(x,y) = T \\ R[f(x,y)] = V[f(x,y)] = A[f(x,y)] = f(x,y) & \text{para cualquier otro valor} \end{cases} \tag{2.3-31}$$

Aquí el nivel de gris T es resaltado en azul y todos los demás pixeles mantienen su nivel de gris original.

3- Imagen que enfatiza una región con niveles de gris determinados.

$$c(x,y)=\begin{cases} R[f(x,y)]=255; V[f(x,y)]=A[f(x,y)]=0 & 105\le f(x,y)\le 185 \\ R[f(x,y)]=V[f(x,y)]=A[f(x,y)]=f(x,y) & \text{para cualquier otro valor} \end{cases} \quad (2.3\text{-}32)$$

Las regiones de la imagen con niveles de gris entre 105 y 185 serán resaltadas con el nivel máximo de rojo (255), y todos los demás pixeles mantendrán su valor original.

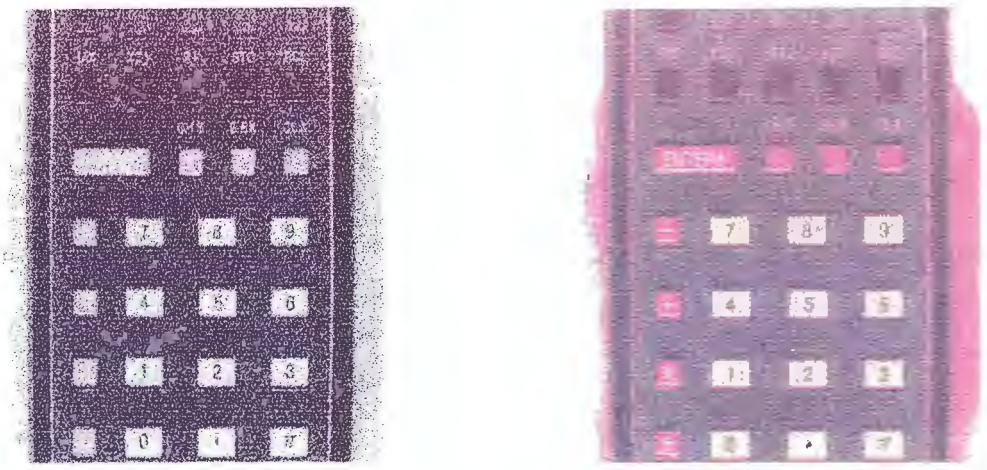
4- Imagen resaltada por completo en el espectro visible del color.

$$c(x,y)=\begin{cases} R=0 & V=254-4f & A=255 & 0\le f\le 63 \\ R=0 & V=4f-254 & A=510-4f & 64\le f\le 127 \\ R=4f-510 & V=255 & A=0 & 128\le f\le 191 \\ R=255 & V=1022-4f & A=0 & 192\le f\le 255 \end{cases} \quad (2.3\text{-}33)$$

Este tipo de funciones se usa para representar los cambios en la escala de grises, como si estos fueran producto de cambios debido a la temperatura [26].

Esta función de mapeo en particular, produce un "arco iris", desde el azul hasta el rojo para una imagen de 256 niveles de gris. Aquí $R = R[f(x,y)]$, $V = V[f(x,y)]$, $A = A[f(x,y)]$, y $f = f(x,y)$. El efecto "arco iris" se debe a la asignación de una combinación específica de colores, sobre cada una de las cuatro regiones en que ha sido seccionada la imagen.

La figura 2.3.9 presenta un ejemplo de aplicación de falso color. La imagen original en escala de gris (a), muestra las teclas de una calculadora. Pero algunos símbolos no son visibles, debido a la intensidad de luz en esa zona. La parte (b) muestra la imagen resaltada por medio de la aplicación de la relación 3 descrita anteriormente.



a) b)

Figura 2.3.9. Ejemplo del uso del falso color para resaltar características. a) imagen original y b) imagen resaltada en color.

2.3.5 TRANSFORMACIONES MORFOLÓGICAS

El procesamiento morfológico de imagen, es un área de procesamiento no lineal que trata de la estructura geométrica de objetos dentro de una imagen. Actualmente se han utilizado algoritmos morfológicos para la eliminación de ruido en la imagen, como también detectores de bordes, en compresión de imágenes y extracción de características [6].

El procesamiento morfológico binario se basa en el tratamiento de un objeto dentro de una imagen binaria como un conjunto compuesto por pixeles. La posición de cada elemento del conjunto define un vector bidimensional que indica la coordenada de cada pixel dentro del objeto. El objeto queda descripto observando las coordenadas de los pixeles que representan al objeto.

Las dos operaciones fundamentales de los filtros del procesamiento morfológico de imágenes binaria son: erosión binaria y dilatación binaria. El concepto del procesamiento

morfológico, se puede extender hacia imágenes representadas en escala de gris donde se definen tres variables de interés, las cuales son: coordenada x, coordenada y y el nivel de intensidad de gris, propio de cada elemento de la imagen [6].

2.3.5.1 DILATACIÓN BINARIA

Considerando dos objetos A y B como imágenes binarias y basándose en la suma de conjuntos, la dilatación del conjunto A debida al conjunto B, se define como todos los desplazamientos posibles de los elementos de A en virtud de los elementos de B, resultando los vectores bidimensionales que componen la imagen dilatada. Simbólicamente se representa así:

$$A \oplus B = \{t \in I^2 : t = a + b, a \in A, b \in B\} \tag{2.3-34}$$

Siendo I^2 el plano en el que se ubica cada elemento t que pertenece a la imagen dilatada.

La ecuación 2.3-34 muestra que la dilatación del objeto A son todos los vectores resultantes de la suma de todos los elementos del objeto A con los de B. La dilatación del objeto A a partir de B, también se puede entender como la unión de todas las traslaciones posibles de los elementos de A debido a cada uno de los elementos del objeto B. Resultando la dilatación como la unión de todos esos traslados así:

$$A \oplus B = \bigcup_{b \in B} \{t \in I^2 : t = a + b, a \in A\} \tag{2.3-35}$$

Un caso particular de dilatación se encuentra cuando el objeto dilatador es un conjunto compuesto únicamente por un elemento ubicado sobre el eje x. En ese caso la imagen dilatada tendrá que ser la imagen original trasladada en la dirección del eje x, de acuerdo a la posición del objeto B. Otro caso similar se encuentra, cuando B está compuesto por dos elementos ubicados sobre el eje x positivo y negativo respectivamente, ya que se trata de dos elementos, el desplazamiento de la imagen original se genera en ambas direcciones y la imagen dilatada tendrá que ser la unión de las dos sub-imágenes de la imagen original A de

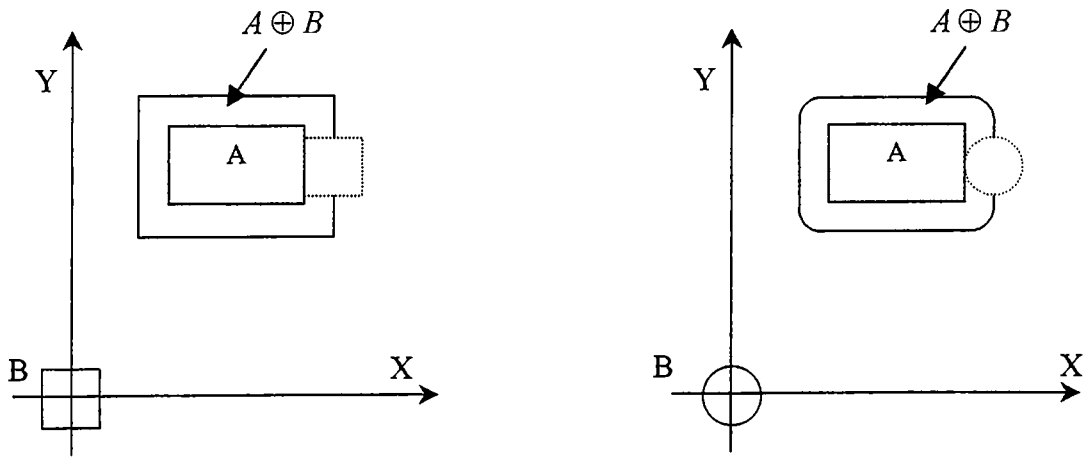
acuerdo a los elementos de B. Finalmente se obtiene una imagen alongada en la dirección del eje x.

Otra forma de interpretar la dilatación binaria es deslizar el objeto dilatador a través de las fronteras externas del objeto a dilatar, haciendo que el movimiento de B sea completamente adyacente en todo el contorno del objeto A. La imagen dilatada resultante será el contorno trazado por el centro del objeto B (si éste se encuentra centrado en el origen del sistema coordenado) más allá de las fronteras del objeto A. Este concepto es fácil de visualizar para figuras geométricas sencillas y tiende a complicarse cuando éstas son estructuras geométricas complejas. En la figura 2.3.10 se presentan dos ejemplos del uso de este concepto, debe destacarse que la forma de la imagen dilatada esta sujeta a la función de estructura (objeto B).

El concepto de rodar el objeto dilatador sobre el objeto a dilatar se puede expresar matemáticamente como:

$$A \oplus B = \bigcup_{x \in I^2} \{Bx \cap A \neq \phi\} \tag{2.3-36}$$

donde $Bx = \{t \in I^2 : t = b + x, b \in -B\}$.



a) Elemento estructural cuadrado. b) Elemento estructural circular.

Figura 2.3.10. Ejemplos de dilatación binaria.

La ecuación 2.3-36 se puede explicar de la siguiente manera: (a) el objeto B se ubica simétricamente en el origen del sistema coordenado para formar el objeto $-B$ a través de la reflexión. (b) este nuevo objeto se traslada por medio de x en toda la imagen. La unión de todos esos traslados se traslapan con el objeto A para obtener como resultado el objeto dilatado. El objeto B actúa como una función filtro que cambia la estructura del objeto A por la operación de dilatación. Por esa razón al objeto B se le conoce como función elemental estructural y la dilatación del objeto A debido a B se llama la dilatación del objeto A por el elemento estructural B.

Existen casos en los que se utiliza un elemento estructural circular para realizar la operación de suavizado de objetos. El suavizado permite eliminar bajas definiciones y cambios bruscos existentes en la imagen original, este efecto se logra visualizar con mayor facilidad en la medida que se aumenta el tamaño de la función del elemento estructural B. La desventaja es que se pierde la forma de la imagen original, tendiendo a tomar la estructura de B. A continuación se presenta la figura 2.3.11 que permite visualizar tal efecto de suavizado.

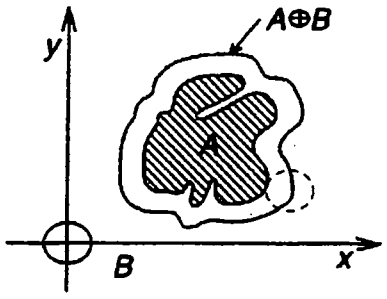


Figura 2.3.11. Ejemplo de dilatación binaria para el suavizado del contorno del objeto.

2.3.5.2 EROSIÓN BINARIA

Al contrario de la dilatación, la erosión binaria disminuye el tamaño del objeto original, ésta última se basa en la substracción de conjuntos y representa la intercepción de todos los

desplazamientos producidos por el elemento estructural sobre el objeto A. En esta ocasión la función estructural se rota 180° con relación al origen. La erosión binaria se define como:

$$A \ominus B = \bigcap_{b \in -B} At$$

ó

$$A \ominus B = \bigcap_{b \in -B} \{t \in I^2 : t = a + b, a \in A\} \qquad (2.3\text{-}37)$$

La figura 2.3.12 presenta un ejemplo de la erosión binaria usando dos puntos como elemento estructural, localizados en (-a,0) y (a,0). El objeto $A1$ resulta del traslado producido por (a,0) ya que B ha sido rotado 180° y el objeto $A2$ se genera de la traslación debida al elemento (-a,0). El objeto erosionado por el elemento estructural B, se obtiene de los elementos comunes tanto para $A1$ como para $A2$. Tal como se observa en la figura el objeto erosionado se ha reducido en tamaño en comparación con el objeto original A.

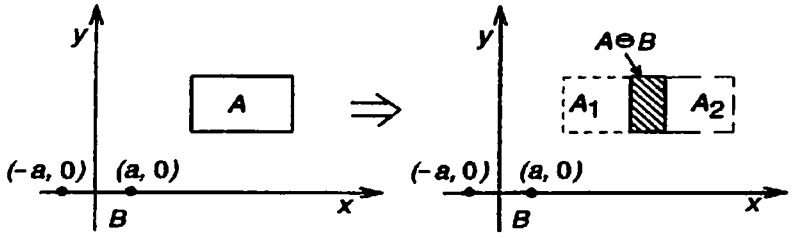


Figura 2.3.12. Erosión binaria del objeto A producida por el objeto B.

Así como la dilatación binaria, la erosión binaria puede ser definida en términos de la aproximación de un círculo rodante. Al contrario de desplazar el objeto circular por el límite externo como se hace en la dilatación, en la erosión se desplaza sobre el límite interno del objeto. El centro del círculo del objeto B genera la frontera para el nuevo objeto erosionado. Si el elemento estructural se centra simétricamente en el origen, el objeto erosionado aparecerá al centro del objeto original A.

El concepto de desplazar el círculo a través del contorno del objeto A, puede ser expresado matemáticamente de la siguiente forma:

$$A \ominus B = \bigcup_{x \in I^2} \{Bx \cup A \neq \phi\}$$

donde

$$Bx = \{t \in I^2 : t = b + x, b \in B\} \qquad (2.3\text{-}38)$$

La ecuación 2.3-38 muestra que la erosión es la unión de todas las traslaciones del elemento estructural en la dirección x y en la zona donde el objeto A encierra por completo al objeto B.

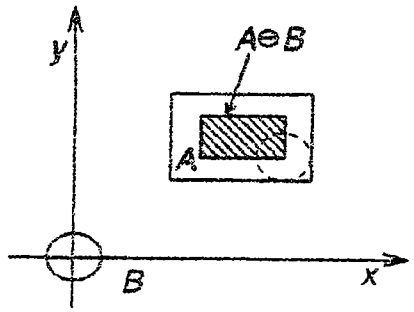


Figura 2.3.13. Erosión de un objeto complejo. Se presenta la reducción del objeto A y alisado de su contorno.

La erosión binaria puede ser expresada en términos de la dilatación utilizando el complemento de A, así:

$$A \ominus B = (A^C \oplus -B)^C \tag{2.3-39}$$

Varias de las propiedades que se cumplen en la teoría de conjuntos, pueden ser definidas y utilizadas para la dilatación y erosión binaria, como por ejemplo:

1°) La dilatación cumple con la propiedad conmutativa:

$$A \oplus B = B \oplus A \tag{2.3-40}$$

2°) La dilatación cumple con la propiedad asociativa de conjuntos:

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) \tag{2.3-41}$$

3°) La erosión y dilatación no varían con la traslación. El resultado es el mismo si primero se traslada y luego se erosiona o dilata, o si primero es erosionado o dilatado y luego trasladado. Propiedad que se aplica a la imagen misma y no al elemento estructural.

$$\begin{aligned} Ax \ominus B &= (A \ominus B)x \\ Ax \oplus B &= (A \oplus B)x \end{aligned} \tag{2.3-42}$$

4°) Cuando la traslación se aplica al elemento estructural, se realizan ciertas modificaciones las que resultan en:

$$A \ominus Bx = (A \ominus B) - x$$
$$A \oplus Bx = (A \oplus B)x \tag{2.3-43}$$

5°) Si al objeto A y al objeto B se les aplica un mismo factor de escala y se efectúa la operación de dilatación o erosión, el resultado final es equivalente si se erosiona o dilata a A por medio de B y posteriormente al objeto dilatado o erosionado se le aplica el mismo factor de escala. Por lo tanto se define el escalado como:

$$A^S = \{t \in R^2, t = sa, a \in A, s \in R\} \tag{2.3-44}$$

donde R es el conjunto de los números reales. La dilatación y erosión cumplen con la propiedad de escalado de la siguiente manera:

$$(A \ominus B)^S = (A^S \ominus B^S)$$
$$(A \oplus B)^S = (A^S \oplus B^S) \tag{2.3-45}$$

Para s>1 el tamaño de A y B se incrementa, de lo contrario el tamaño de éstos se reduce.

6°) La dilatación cumple con la propiedad distributiva de conjuntos para la unión e intercepción. La unión o intercepción para los objetos A y C seguida de la dilatación es equivalente a dilatar ambos objetos y luego realizar su intercepción o unión.

$$(A \cup C) \oplus B = (A \oplus B) \cup (C \oplus B)$$
$$(A \cap C) \oplus B = (A \oplus B) \cap (C \oplus B) \tag{2.3-46}$$

También es aplicable para la erosión de la intercepción de A con C.

7°) Otra de las propiedades importantes es la propiedad distributiva entre la erosión y dilatación:

$$A \ominus (B \oplus D) = (A \ominus B) \ominus D \tag{2.3-47}$$

La erosión del objeto A debido a la dilatación de B por medio de D, es igual a la erosión de A por medio de B seguido de la erosión provocada por D. Esta es una propiedad muy importante porque permite que los elementos estructurales puedan ser descompuestos en elementos estructurales más pequeños. Siendo BT el elemento estructural compuesto así:

$$BT = (...(B1 \oplus B2) \oplus B3) \oplus ... \oplus BN) \tag{2.3-48}$$

Entonces para la erosión se tiene:

$$A \ominus BT = (...(A \ominus B1) \ominus B2) \ominus B3) \ominus ... \ominus BN) \tag{2.3-49}$$

La aplicación es análoga para la dilatación:

$$A \oplus BT = (...(A \oplus B1) \oplus B2) \oplus B3) \oplus ... \oplus BN) \tag{2.3-50}$$

Un caso particular es aquel en el que se tiene la dilatación o erosión de un objeto A por un objeto B de manera iterativa durante n veces:

$$\begin{aligned} A \ominus Bn &= (...(A \ominus B) \ominus B) \ominus B) \ominus ... \ominus B) \\ A \oplus Bn &= (...(A \oplus B) \oplus B) \oplus B) \oplus ... \oplus B) \end{aligned} \tag{2.3-51}$$

donde

$$Bn = (...(B \oplus B) \oplus B) \oplus ... \oplus B) \tag{2.3-52}$$

El elemento Bn define n dilataciones del elemento estructural B. La ecuación 2.3-51 define que la dilatación o erosión de un objeto con el mismo elemento estructural durante n veces es igual a erosionar o dilatar una vez al objeto con la n-ésima dilatación del elemento estructural.

Una aproximación para la realización de los filtros involucrados en la dilatación o la erosión, es el uso de las operaciones lógicas en conjunto con una máscara, con el objetivo de poder representar la función estructural. La erosión y dilatación pueden ser definidas usando una modificación de convolución espacial, así la erosión se puede definir de la siguiente manera:

$$g(x,y) = \begin{cases} 0 & \text{para } \underset{i,j \in H}{\text{AND}}\{f(x-i, y-j)\} = 0 \\ 1 & \text{para } \underset{i,j \in H}{\text{AND}}\{f(x-i, y-j)\} = 1 \end{cases} \tag{2.3-53}$$

donde H define el elemento estructural y $g(x,y)$ la imagen erosionada.

La realización de la dilatación binaria sigue los mismos pasos que se utilizan para la erosión, excepto que la ecuación 2.3-53 se modifica por la inclusión de la operación lógica OR.

$$g(x,y) = \begin{cases} 0 & \text{para } \underset{i,j \in -H}{\text{OR}}\{f(x-i, y-j)\} = 0 \\ 1 & \text{para } \underset{i,j \in -H}{\text{OR}}\{f(x-i, y-j)\} = 1 \end{cases} \tag{2.3-54}$$

donde $-H$ es la rotación de 180° de H alrededor del origen, $f(x,y)$ es la entrada de la imagen original, $g(x,y)$ la imagen dilatada y OR{} emplea la operación lógica OR para todos los

pixeles definidos por elemento estructural *-H*. La rotación de 180° es necesaria para ser consistente con la definición original de dilatación usando la traslación. La salida de la ecuación 2.3-54 será uno si cualquiera de las sub-imágenes generadas por cada vector bidimensional de *-H* produce un uno en cualquiera de las coordenadas respectivas de la imagen dilatada. Debe recordarse que la operación OR es análoga a la operación de la unión en la teoría de conjuntos.

La figura 2.3.14 presenta un ejemplo de aplicación de la dilatación binaria en una imagen total.

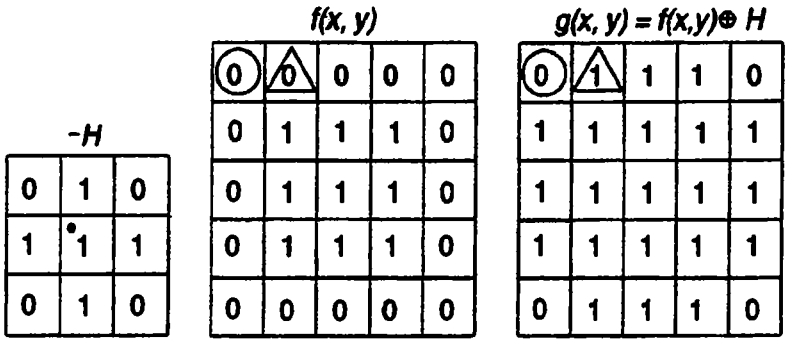


Figura 2.3.14. Dilatación binaria en una imagen completa.

2.3.5.3 APERTURA BINARIA

Como se presentó anteriormente la dilatación binaria tiende a aumentar el tamaño del objeto, el caso contrario es que la erosión hace disminuir el tamaño de éste. La combinación de esas operaciones morfológicas genera una forma y tamaño relativo a la imagen original y es una alternativa de realizar el filtrado geométrico de la imagen. La operación de apertura binaria se define como la erosión del objeto A debida al elemento estructural, seguido de la dilatación usando la misma función estructural, matemáticamente se puede expresar como:

$$A_B = (A \ominus B) \oplus B \qquad (2.3\text{-}55)$$

La ecuación 2.3-55 no permite obtener el objeto original porque la erosión y dilatación no son operaciones inversas. La apertura morfológica se usa para eliminar pequeños objetos geométricos y para suavizar el contorno de la imagen, volviendo circular las esquinas de la misma.

La figura 2.3.15 presenta los pasos a seguir para realizar la operación de abertura usando un elemento estructural circular y desplazando el elemento para efectuar la erosión y dilatación respectivamente.

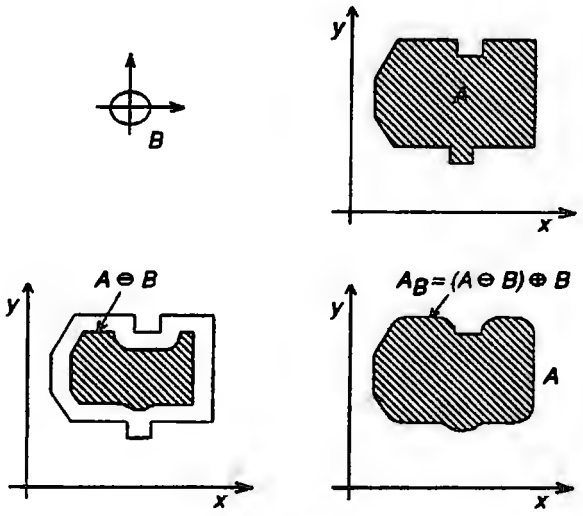


Figura 2.3.15. Apertura binaria de un objeto rectangular.

Debe notarse que las esquinas cuadradas del rectángulo han tomado una forma circular, con un radio igual al radio del elemento estructural *B*.

2.3.5.4 CERRADURA BINARIA

Una operación similar a la apertura binaria es la cerradura binaria, que se define como la dilatación de un objeto seguida por la erosión, bajo la utilización del mismo elemento estructural.

$$A^B = (A \oplus -B) \ominus -B \tag{2.3-56}$$

De la ecuación 2.3-56 se puede observar que en la cerradura, primero se efectúa la dilatación lo que repercute en el aumento del tamaño del objeto original, luego a ese resultado se le aplica la erosión, por tal motivo los objetos no se eliminan por cerradura.

La cerradura es muy usada para eliminar sombras al interior de un objeto y suavizar el contorno de éste reduciendo los vértices formados por esquinas.

En la figura 2.3.16 se presentan los pasos a seguir para realizar la cerradura binaria usando el mismo objeto rectangular y elemento estructural así como en la figura 2.3.15.

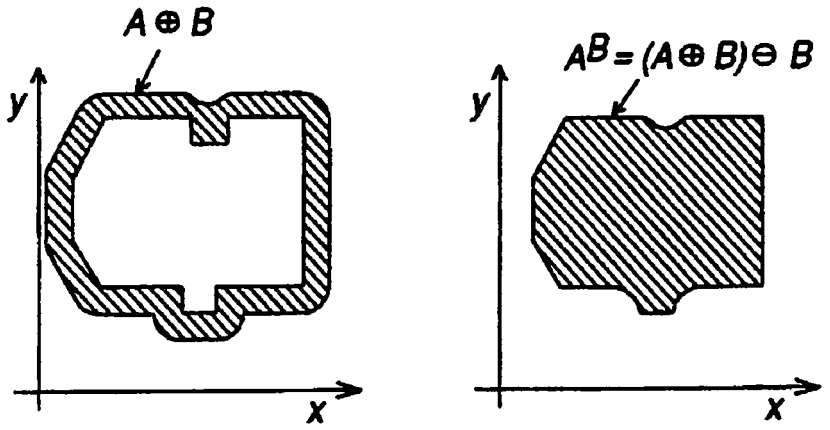
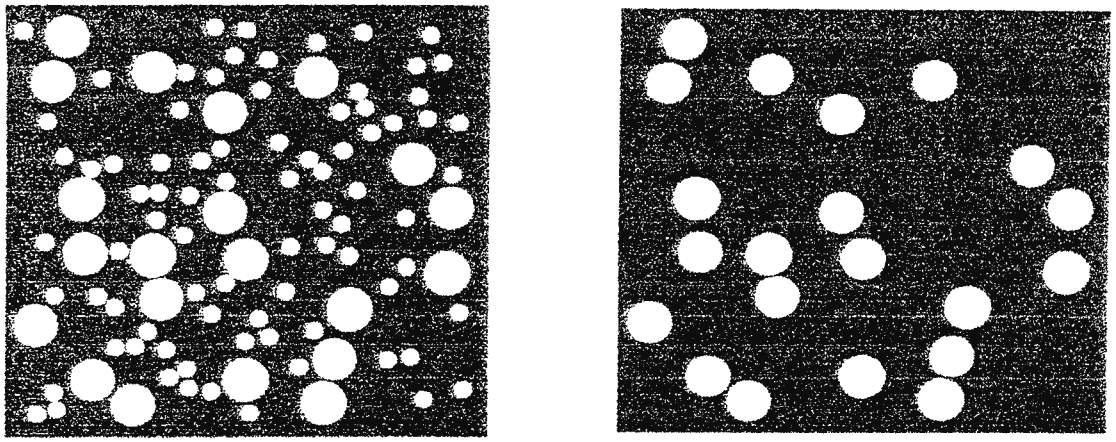


Figura 2.3.16. Cerradura binaria del objeto rectangular *A* presentado en la figura 2.3.15.

De acuerdo a la figura anterior el primer paso a seguir es la rotación de *B* un ángulo de 180° en torno al origen, con el objetivo de generar -*B*. Ya que *B* es simétrico con respecto al origen, la rotación no es necesaria en este ejemplo. El siguiente paso es la dilatación que es la encargada de expandir el objeto, aumentando el tamaño de las irregularidades externas y disminuyendo el tamaño de los vértices internos. El último paso es erosionar el objeto dilatado con el objetivo de reducir el tamaño de éste, produciéndose un objeto rectangular de tamaño parecido al objeto original [26].

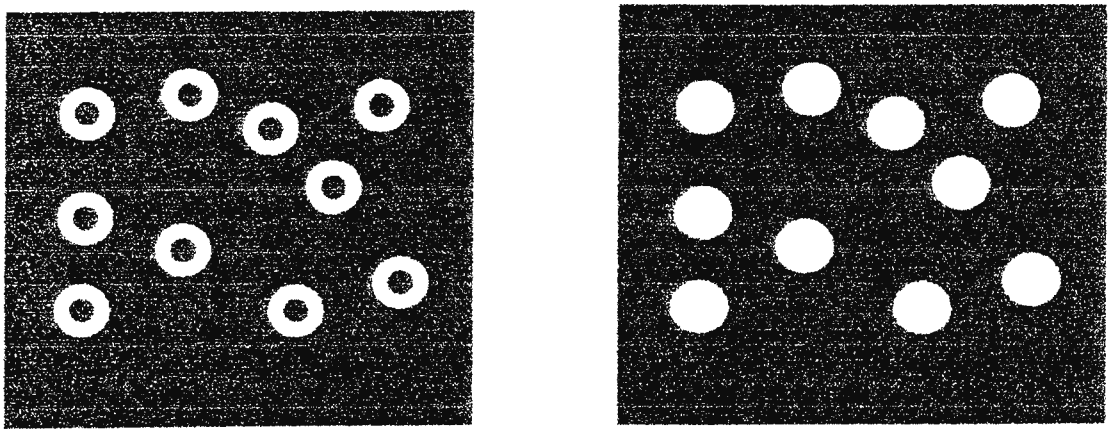
A continuación se presentan las figuras 2.3.17 y 2.3.18 en las que se observa la aplicación de la apertura y cerradura respectivamente.

En la figura 2.3.17 se emplea la apertura para eliminar objetos de diferente tamaño con relación al tamaño del objeto de interés y en 2.3.18 se emplea la cerradura para hacer desaparecer las sombras al interior de la imagen original.



a) b)

Figura 2.3.17. Uso de la apertura binaria para separar dos objetos de diferentes tamaños. a) Imagen original, b) Imagen que resulta de la apertura binaria.



a) b)

Figura 2.3.18. Uso de cerradura binaria para eliminar las sombras al interior de los objetos. a) Imagen original, b) Imagen después de aplicar la cerradura binaria.

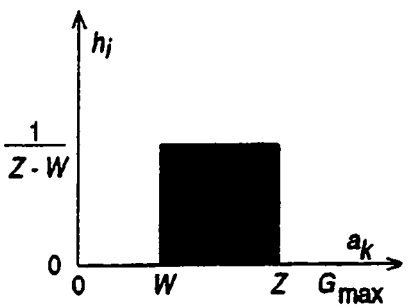
2.3.6 TRATAMIENTO DE RUIDO EN IMÁGENES DIGITALES

Ruido en las Imágenes

El ruido es toda información que degrada una imagen y que no forma parte de la escena original. El ruido hace que muchas características importantes de la imagen no puedan ser observadas. Las fuentes más comunes generadoras de ruido están relacionadas con las condiciones ambientales, pero incluso elementos de la misma cámara son fuente potencial de ruido, como por ejemplo el ruido producido por los sensores sus dispositivos electrónicos asociados [26].

Los tipos de ruido pueden ser clasificados dependiendo de la forma del histograma que representa la imagen de ruido. A continuación se presentan los tipos más comunes de ruido en las imágenes, junto con la definición del valor de histograma h_i, la distribución de histograma y una imagen ejemplo.

Ruido Uniforme



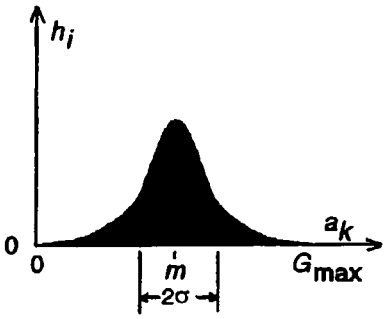
$$h_i(a_k) = \begin{cases} \frac{1}{Z-W} & \text{Para } W \le a_k < Z \\ 0 & \text{Para otros valores} \end{cases}$$

Figura 2.3.19. Distribución de imagen de ruido uniforme.

El ruido uniforme produce igual valor de histograma para cada nivel de gris en un rango determinado. Este tipo de ruido hace más difícil el proceso de segmentación y

detección de regiones sobre una imagen, debido a la exagerada distribución uniforme que presenta la imagen de ruido y que se suma como una característica adicional a la imagen.

Ruido Gaussiano

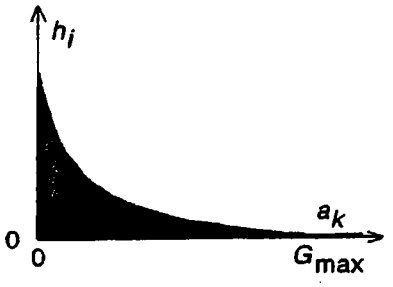


$$h_i(a_k) = \frac{e^{-(a_k-m)^2/\sigma^2}}{\sigma\sqrt{2\pi}} \quad \text{para} -\infty < a_k < \infty$$

Figura 2.3.20. Distribución de imagen de ruido Gaussiano

En la definición m es el valor promedio (*avg*) de la imagen de ruido, dada por la ecuación (2.3-3) y σ es la desviación estándar (*std*) definida por (2.3-4). El valor promedio está justo en la cresta del histograma y representa la más alta probabilidad del ruido Gaussiano, y la desviación determina el grado de dispersión del ruido. El ruido Gaussiano es de los más comunes y su distribución sirve como modelo del comportamiento de las fuentes desconocidas de ruido presentes en un sistema.

Ruido Exponencial Negativo

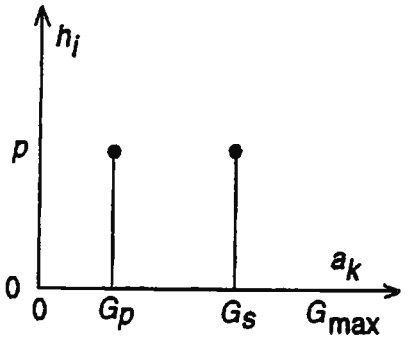


$$h_i(a_k) = \frac{e^{-a_k/a}}{a} \quad \text{para } 0 \le a_k < \infty$$

Figura 2.3.21. Distribución de imagen de ruido exponencial negativo.

Este tipo de ruido aparece en imágenes adquiridas usando un láser como fuente de iluminación. En la ecuación el valor de a es igual al valor promedio y a la desviación estándar. La más alta probabilidad de aparición se encuentra en el nivel de gris cero y decrece a medida que aumenta el valor.

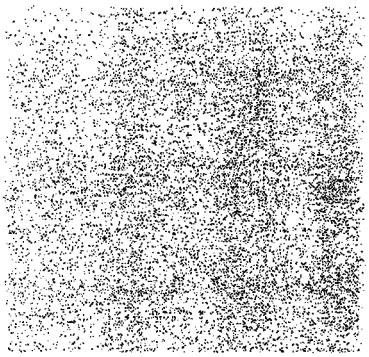
Ruido Tipo Sal y Pimienta



$$h_i(a_k) = \begin{cases} p & \text{(pimienta)} \quad \text{para } a_k = G_p \\ p & \text{(sal)} \quad \text{para } a_k = G_s \\ o & \text{para cualquier otro valor} \end{cases}$$

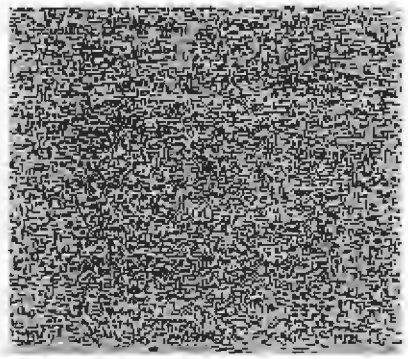
Figura 2.3.22. Distribución de imagen de ruido tipo sal y pimienta.

Es causado por el polvo y la suciedad en el sistema óptico de la cámara. Solamente existen dos niveles que son agregados a la imagen G_p y G_s. Generalmente estos valores difieren mucho y es ésta la causa de la apariencia como de sal y pimienta sobre la imagen [26]. Este tipo de ruido puede ser removido con un filtro espacial basado en la mediana, como se explicará más adelante.



a) Ruido uniforme.

b) Ruido Gaussiano.



c) Ruido exponencial negativo.

d) Ruido sal y pimienta.

Figura 2.3.23. Ejemplos de imágenes con ruido.

Filtrado Espacial

El filtrado espacial permite remover el ruido y afinar las imágenes borrosas. Existen filtros espaciales lineales y no lineales. Un filtro espacial lineal está basado en la convolución bidimensional de una imagen $f(x,y)$ con la respuesta impulso de un filtro $h(x,y)$. En el caso de los no lineales se pueden distinguir tres categorías: los que operan en base a una ventana en forma similar a los lineales, los filtros adaptativos que modifican sus características de acuerdo al tipo de ruido y características de la imagen, y por último los filtros homomorfos utilizados para remover ruido multiplicativo y el sombreado en imágenes. Puede consultarse Pratt para otros tipos de filtros más complejos aplicados a restauración de imágenes [18].

Convolución de dos Imágenes

La convolución en dos dimensiones de una imagen de NxM con un filtro de respuesta impulso definida sobre el conjunto de elementos dados por H se define como

$$g(x,y) = \sum_{i,j \in H} \sum f(x-i, y-j)\, h(i,j) \qquad (2.3.57)$$

donde $0 \leq x - i < N$ y $0 \leq y - j < M$. Para cualquier pixel en una imagen, la ecuación (2.3-57) es la sumatoria ponderada del pixel $f(x,y)$ con sus pixeles vecinos, con el peso dado en cada pixel del filtro $h(x,y)$ [26].

En la figura 2.3.24 se muestra un pixel genérico $f(x,y)$ y sus vecinos, además de un filtro $h(x,y)$ de 3x3 con sus respectivos valores de respuesta impulso dados por los A_i. El centro del filtro está dado por las coordenadas 0,0 y se lo define en el rango de $x,y \in -1,0,1$.

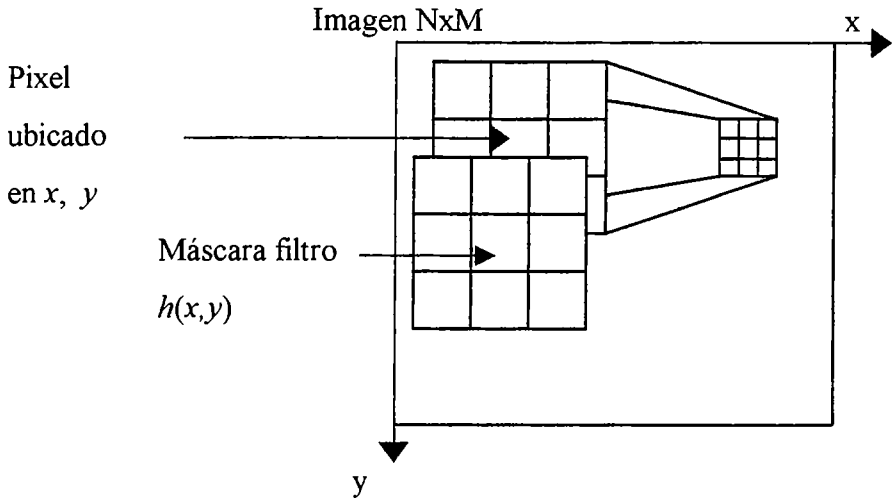


Figura 2.3.24. Ilustración de filtrado espacial haciendo uso de una máscara de vecinos.

EL concepto de filtrado espacial usando máscaras de filtrado puede extenderse a cualquier tamaño y forma deseada. Arbitrariamente máscaras de formas particulares forman un papel importante en el área de filtrado morfológico. Por la facilidad de implementación en la programación y en el hardware es usual utilizar máscaras de forma rectangular o cuadrada de tamaño impar (3x3, 5x5, etc.). Esta primera definición hace que durante el procesamiento no se incluya en la convolución la periferia de la imagen. Esto simplifica el procesamiento reduciendo al máximo el tiempo empleado para la ejecución del mismo.

Algunas veces se hace necesario sacrificar el tiempo de procesamiento a costa de tratar la totalidad de la imagen, por lo que se utilizará otra expresión en la definición de la

convolución. El primer caso que se muestra es aquel en que se justifican los bordes superior e izquierdo. La expresión en forma de series es como la siguiente

$$g(x,y) = \sum_{i,j \in H} \sum f(x,y)\ \delta(x-i+1, y-j+1) \tag{2.3-58}$$

donde

$$\delta(x-i+1, y-j+1) = \begin{cases} 1 & x=i \text{ y } y=j \\ 0 & \text{otro caso} \end{cases} \tag{2.3-59}$$

Los límites de las sumatorias son

$$\text{MAX}[1, x-L+1] \leq i \leq \text{MIN}[N, x] \tag{2.3-60}$$

donde MAX[a,b] y MIN[a,b] son el máximo y el mínimo del argumento. De igual manera se cumple para y. Examinando los índices del vector respuesta impulso y su posición extrema indica que $M = N - L - 1$, y por lo tanto la matriz procesada de salida, g, tiene mayores dimensiones que la de entrada, f. En la figura 2.3.25 se muestra una geometría de la superposición de área finita [18].

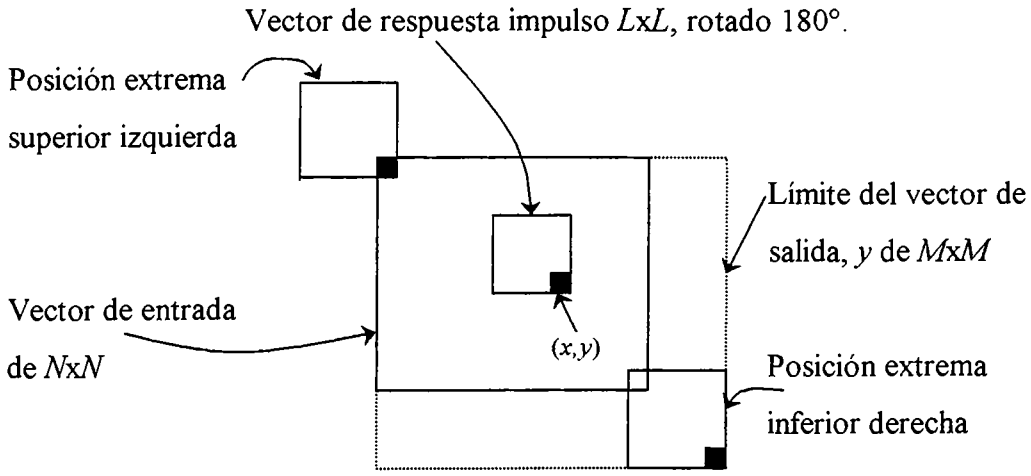


Figura 2.3.25. Relaciones entre vector de entrada, salida y respuesta impulso para superposición de área finita justificado esquina superior izquierda

Por razones de notación es conveniente utilizar una definición en la cual el arreglo de salida esté centrado con respecto al arreglo de entrada. La definición para el caso centrado está dada por

$$g_c(x,y) = \sum_i \sum_j f(x,y)h(x-i+L_c, y-j+L_c; x, y) \qquad (2.3\text{-}61)$$

donde - $(L - 3)/2 \leq x \leq N + (L - 1)/2$ (igual manera para y) y $L_c = (L + 1)/2$. Los límites de la sumatoria son para este caso

$$\text{MAX}[1, x - (L - 1)/2] \leq i \leq \text{MIN}[N, x + (L - 1)/2] \qquad (2.3\text{-}62)$$

De la misma manera se cumple para la variable y. La figura 2.3.26 muestra la relación espacial entre los diferentes arreglos en el caso de tratar con un arreglo de respuesta impulso de 5x5.

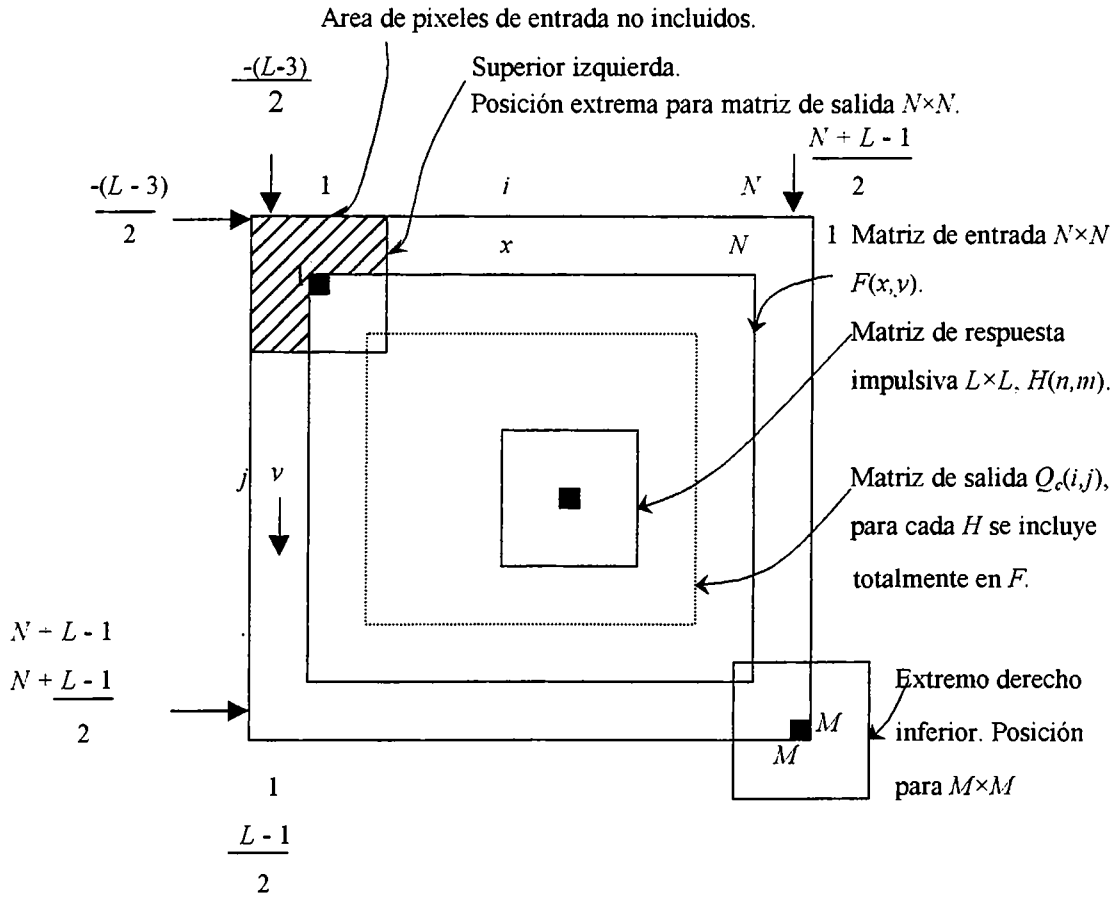


Figura 2.3.26. Relación entre vector de entrada, salida y respuesta impulso para superposición de área finita definición de arreglo centrado

En procesamiento de imágenes digitales es conveniente trabajar con arreglos de igual dimensión. Para aquellos sistemas, la ecuación (2.3-61) debe ser evaluada sólo en el rango $1 \leq x$ ó $y \leq N$. Cuando la matriz de respuesta impulso está ubicada en el borde del arreglo de entrada, el producto de la ecuación (2.3-61) no involucra todos los elementos de la matriz de respuesta impulso. Situación que se ilustra en la figura 2.3.26, y para el caso de los pixeles fuera de cómputo se muestra la figura rayada. Uno de los métodos es seguir la definición dada por la ecuación (2.3-61), y asignar el valor de cero a aquellos pixeles errados. Otro método es asignarle a ellos un valor constante. Una variante de estos métodos es asignar a los pixeles errados una imagen de los pixeles del arreglo de entrada, como se muestra en la esquina inferior derecha de la figura 2.3.26. Para este caso la definición se volvería como sigue

$$g_c(x,y) = \sum_i \sum_j f(i',j')h(x-i+L_c, y-j+L_c; x, y) \tag{2.3-63}$$

donde los límites de la sumatoria son

$$x - (L - 1)/2 \leq i \leq x + (L - 1)/2 \tag{2.3-64}$$

y además

$$i' = 2 - i \qquad \text{para } i \leq 0 \tag{2.3-65a}$$
$$i' = i \qquad \text{para } 1 \leq i \leq N \tag{2.3-65b}$$
$$i' = 2N - i \qquad \text{para } i \geq N \tag{2.3-65c}$$

de la misma forma para j' y y.

Filtro Espacial Lineal Promedio

Es un filtro pasa bajos que remueve las frecuencias espaciales altas de una imagen, reduciendo además el ruido Gaussiano. Su principal desventaja es que degrada la imagen dejándola borrosa reduciendo la definición de los bordes de la misma. La expresión generalizada para un filtro de este tipo es

$$g(x,y) = \frac{1}{N_t} \sum \sum_{i,j \in H} f(x-i, y-j) \tag{2.3-66}$$

donde $0 \leq x - i < N$ y $0 \leq y - j < M$ y N_t es el número de elementos contenidos en la máscara de filtrado H. Estas máscaras por lo general están normalizadas a la unidad por lo que al realizar el proceso no introducen un cambio en la amplitud de la imagen procesada. Abajo se muestran algunos ejemplos de filtros pasa bajos [18].

Máscara 1 Máscara 2 Máscara 3

$$H = \frac{1}{9}\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix} \quad (2.3\text{-}67a) \qquad H = \frac{1}{10}\begin{vmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{vmatrix} \quad (2.3\text{-}67b) \qquad H = \frac{1}{16}\begin{vmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{vmatrix} \quad (2.3\text{-}67c)$$

El último de los casos puede enunciarse como un caso general, conocido como filtro pasa bajos paramétrico de 3x3, el que se define en su respuesta impulso como

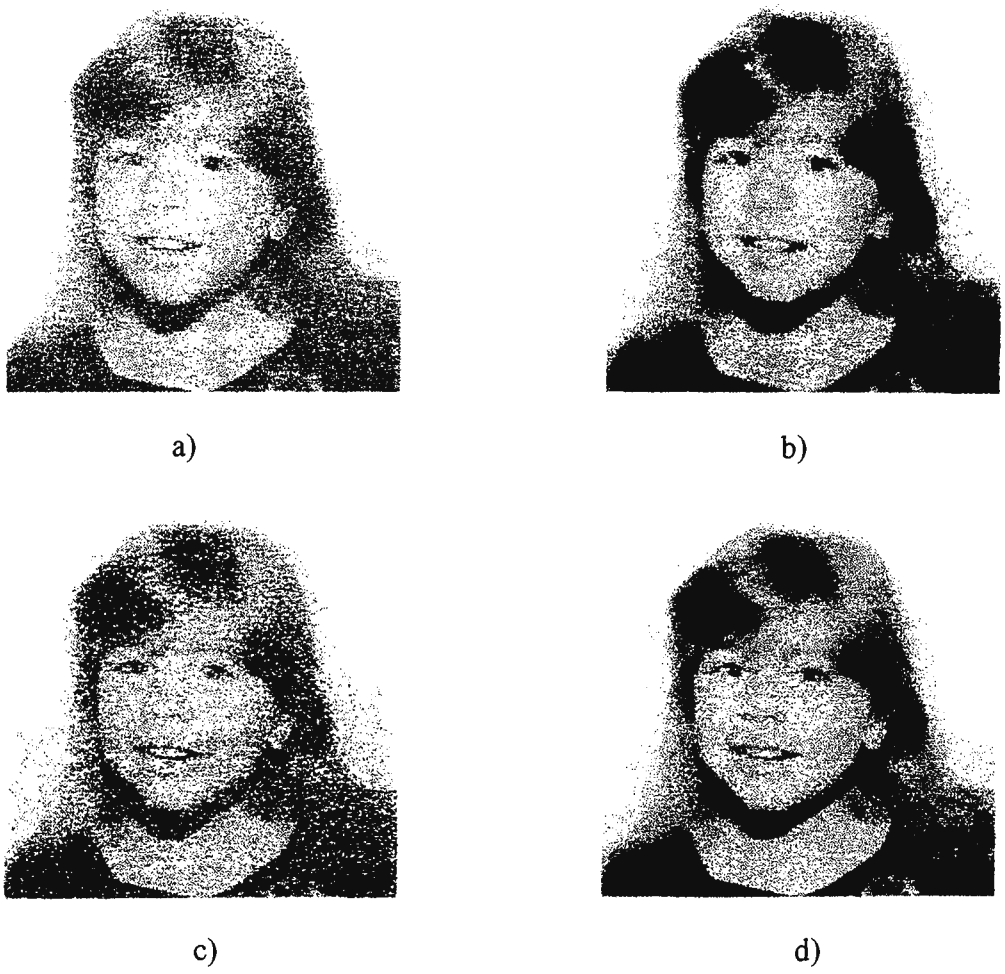
$$H = \frac{1}{(b+2)^2}\begin{vmatrix} 1 & b & 1 \\ b & b^2 & b \\ 1 & b & 1 \end{vmatrix} \quad (2.3\text{-}68)$$

Si se observa cuidadosamente la ecuación (2.3-66) se verá que durante el procesamiento no se incluyen las $(N\text{-}1)/2$ primeras y últimas columnas ni las $(M\text{-}1)/2$ primeras y últimas filas. Este es la mayor limitante de este método, el no incluir la periferia de la imagen, aunque cabe destacar que la mayor información en una imagen se encuentra en la parte central.

El ruido Gaussiano es removido satisfactoriamente con filtros de promedio o pasa bajos. De hecho, la desviación estándar del ruido se reduce en un factor de uno entre la raíz cuadrada del número de elementos del filtro. Para el caso de un filtro de promedio de *nxn*, la desviación estándar cae *n* veces. Un tamaño de ventana menor podría ser utilizado, reduciendo el total de borrosidad, pero también habría menos ruido removido. La regla

general es, cuanto mayor es el filtro, mayor cantidad de ruido se remueve pero más borrosa será la imagen filtrada [26].

El ruido de sal y pimienta, que pertenece al tipo de ruido *outlier*, y se define como un ruido que produce valores que cambian drásticamente de los normales, es pobremente removido por el filtro de promedio. El mejor filtro para eliminar este tipo de ruido es el filtro de mediana, y además mantiene de mejor manera los bordes que el filtro de promedio.



a) b)

c) d)

Figura 2.3.27. Ejemplos de filtrado: a) imagen degradada por ruido Gaussiano con varianza de 600, b) filtro de promedio de 7x7, c) imagen degradada con ruido sal y pimienta con probabilidad de ocurrencia del 15% y d) filtrado de mediana de 3x3 de la figura 2.3.27 (c)

A continuación veamos como se refleja a través de la Transformada Discreta de Fourier de su respuesta impulso el hecho que el filtro de promedio se comporte como un filtro pasa bajos

$$\Im\{h(x,y)\} = H(n,m) = \frac{1}{N^2}\sum_{y=0}^{N-1}\sum_{x=0}^{N-1} h(x,y)e^{-j2\pi(nx+my)/N} \qquad (2.3\text{-}69)$$

Haciendo $h(x,y) = 1/Nt$, que es igual a la constante incluida en la ecuación del filtro de promedio y aplicando la propiedad de desplazamiento de la transformada de Fourier, se llega a

$$H(n,m) = \frac{1}{N_t}\sum_{x,y\ \in\ H}\sum e^{-(j2\pi xn + j2\pi ym)/N} \qquad (2.3\text{-}70)$$

donde n y m son las frecuencias espaciales en las direcciones x y y, respectivamente. La figura 2.3.28 muestra el espectro de magnitud para la respuesta impulso de un filtro de promedio de 7x7. Cuanto más grande sea el filtro pasa bajos, más angosta se volverá la función *sinc* y el filtro conservará menos frecuencias espaciales. El efecto neto es que mayor cantidad de frecuencias espaciales altas son atenuadas, incrementando la borrosidad de la imagen [26].

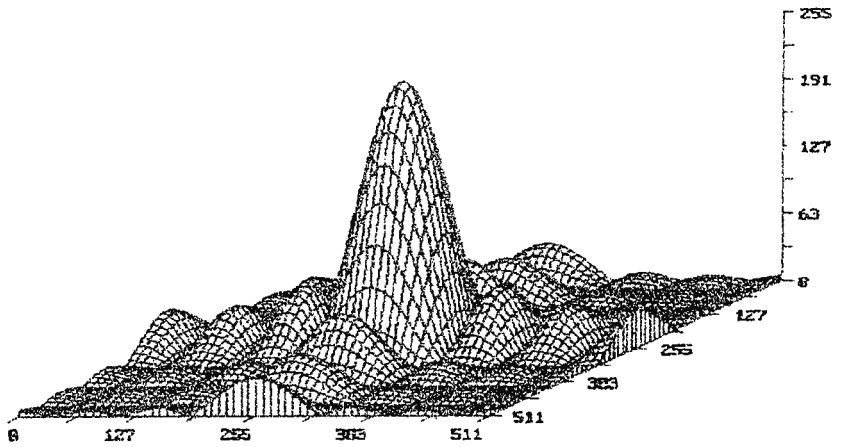


Figura 2.3.28. Respuesta impulso de un filtro de promedio de 7×7.

Otro filtro lineal espacial es el filtro Gaussiano. Este filtro tiene simetría circular y su respuesta impulso varía de acuerdo a la función Gaussiana

$$h(x,y) = e^{-\pi(x^2+y^2)/a^2} \tag{2.3-71}$$

donde el parámetro a determina el ancho del filtro Gaussiano. El espectro de Fourier de magnitud correspondiente es

$$|\Im\{h(x,y)\}| = e^{-\pi a^2(n^2+m^2)/N} \tag{2.3-72}$$

donde n y m son las frecuencias espaciales en las direcciones x y y, respectivamente. Esta última ecuación muestra que el filtro Gaussiano es también un filtro pasa bajos.

Filtrado Espacial de Mediana

El cálculo que se realiza para el filtro de mediana es diferente al expuesto anteriormente, ya que no se utiliza convolución. La máscara de filtrado define cual de los pixeles se van a incluir en el cálculo de la mediana. El proceso de cálculo se inicia ordenando los N pixeles incluidos como lo define la máscara de filtrado desde el mínimo al valor máximo:

$$F(0) \le F(1) \le \ldots \le F(N\text{-}2) \le F(N\text{-}1) \tag{2.3-73}$$

donde $F(0)$ es el valor mínimo y $F(N\text{-}1)$ el máximo de los pixeles incluidos en el cálculo. La salida del filtro de mediana se encuentra usando

$$F_{\text{med}} = \begin{cases} \dfrac{F_{(N/2)} + F_{(N/2-1)}}{2} & \text{Para } N \text{ par} \\ F_{((N-1)/2)} & \text{Para } N \text{ impar} \end{cases} \tag{2.3-74}$$

Los filtros basados en las ecuaciones anteriores se conocen como filtros de orden estadístico. Este tipo de filtro es adecuado para quitar el ruido outlier. Una característica importante de destacar es que estos filtros tienen un valor de corte a partir del cual no podrán remover los pixeles fuera de tono. Es decir, existe un valor porcentual respecto de la cantidad de pixeles del filtro después del cual el filtrado no será exitoso. Este valor de corte es del 50% para los filtros de mediana. En la figura 2.3.27 se pueden ver los resultados del filtrado.

Filtrado Espacial Pasa Altos

Este tipo de filtro afina las imágenes borrosas realzando las altas frecuencias presentes en la misma. Las componentes espectrales de una imagen pueden dividirse en las de baja frecuencia $f_l(x,y)$ y las de alta frecuencia $f_h(x,y)$. Y dado que el filtro de promedio es un pasa bajos, se puede escribir

$$f(x,y) = f_m(x,y) + f_h(x,y) \tag{2.3-75}$$

En muchos casos se quiere realzar las componentes de alta frecuencia sin atenuar las componentes de baja frecuencia para afinar una imagen borrosa. Entonces resulta ser para las frecuencias altas realzadas una cantidad G, que usualmente varía entre 0 y 2, la siguiente expresión

$$f_h(x,y) = (1 + G) f(x,y) - G\ f_m(x,y) \tag{2.3-76}$$

La matriz que describe la ecuación anterior es:

-G/9	-G/9	-G/9
-G/9	(9+8G)/9	-G/9
-G/9	-G/9	-G/9

(2.3-77)

Filtrado Espacial de Frecuencias

Otro método usual es utilizar las componentes real e imaginaria de la imagen a través de la DFT. En el dominio de Fourier podría expresarse el producto de dos funciones correspondientes a la de entrada y la respuesta impulso para obtener la imagen filtrada

$$G(n,m) = F(n,m)\, H(n,m) \tag{2.3-78}$$

donde n y m son las frecuencias espaciales en las direcciones x y y respectivamente. La imagen filtrada se obtiene realizando la antitransformada de Fourier de la anterior

$$g(x,y) = F^{-1}\{G(n,m)\} \tag{2.3-79}$$

El siguiente diagrama de flujo muestra los pasos a seguir para filtrar una imagen haciendo uso de la transformada de Fourier.

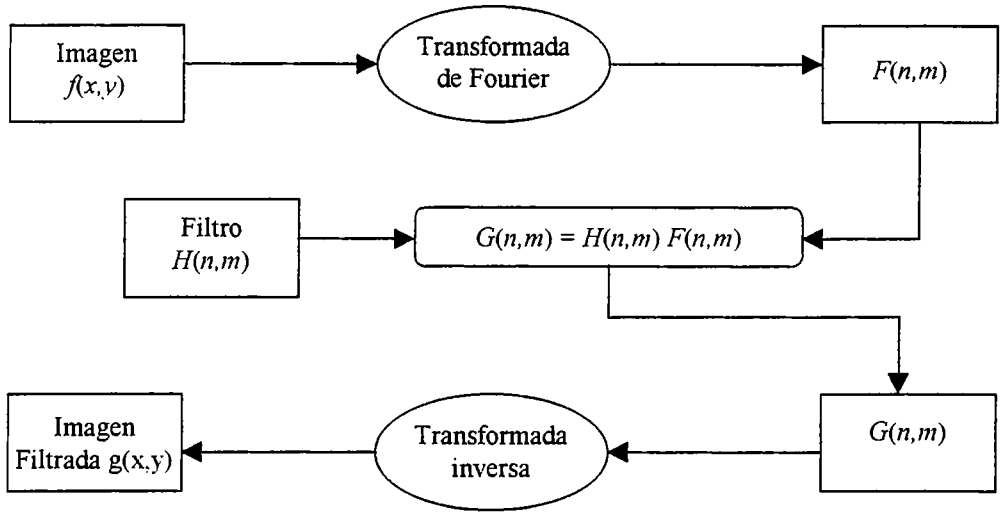


Figura 2.3.29. Diagrama de flujo de utilización de técnicas de filtrado en el dominio de la frecuencia de Fourier.

Existe una gran cantidad de filtros de una dimensión, muchos de ellos vienen del diseño de circuitos electrónicos. Los más comunes son Butterworth, elíptico, Chebyshev y Bessel como los de primer y segundo orden en redes de resistores, capacitores e inductores.

Filtros Espaciales no Lineales

El filtro de mediana descripto anteriormente es uno de los filtros no lineales de orden estadístico más conocidos. Como se ha mencionado estos filtros realizan un mejor trabajo en remover ruido del tipo sal y pimienta y preservan el afinado de las imágenes. También existen filtros de mínimo y de máximo que se los emplea para remover ruido del tipo sal o pimienta respectivamente. Un filtro que posee una respuesta aceptable frente a ruidos Gaussianos y outlier es el alfa-promedio ajustado. Su implementación requiere el ordenamiento de los pixeles a ser filtrados desde un mínimo a un máximo. El filtro alfa-promedio ajustado se define así

$$alfa - promedio[f(x,y)] = \frac{1}{N-2p} \sum_{i=p}^{N-p-1} F_{(i)} \qquad (2.3\text{-}80)$$

donde p = 0,1,2,3...*entero*[N/2] y N-2p es el total de pixeles incluidos en el cálculo del promedio. El valor de p determina el tipo de filtrado a realizar. Para p = 0 la expresión se reduce al filtro de promedio. El límite superior para p de *entero*[N/2] es válido sólo para valores de N impares, en tanto que para valores pares del mismo el límite deberá cambiarse a N/2-1. Para cualquiera de los casos, sea N par o impar, cuando p es el valor máximo el tipo de filtro obtenido es del tipo de mediana. Existen otros tipos de filtros de orden exponencial como los siguientes: filtro alfa-promedio ajustado complementario utilizado para remover ruido uniforme; el filtro promedio ajustado modificado es similar al alfa promedio y produce resultados equivalentes: el filtro promedio ajustado modificado de doble ventana, que utiliza dos ventanas diferentes para el cálculo de la mediana y el promedio; el filtro de rango, que detecta los bordes con diferencias de niveles de gris menores, aunque es muy susceptible al ruido; el filtro cuasi-rango es similar al anterior solo que a través de un parámetro puede modificarse su valor de ruptura y volverlo menos susceptible al ruido outlier.

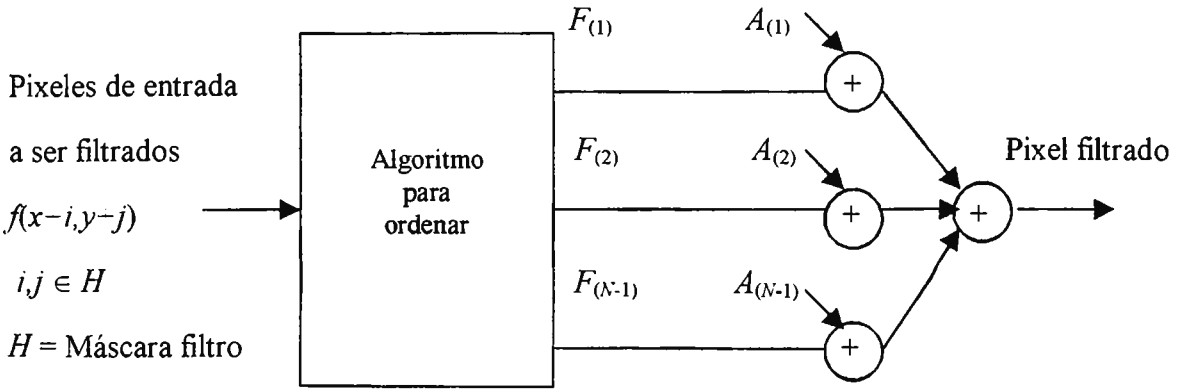


Figura 2.3.30. Diagrama en bloques que ilustra el formato de un filtro de orden estadístico.

Todos estos filtros descriptos no necesitan tener algoritmos por separado. La mayoría de ellos se puede implementar con un algoritmo pero haciendo uso de diferentes parámetros para cada caso. En la figura 2.3.30 se muestra un diagrama en bloques básico para realizar lo anterior. Lo primero es definir dos vectores o arreglos, uno que contiene los pixeles que se van a filtrar y el otro para almacenar los coeficientes, los que definirán el tipo de filtro a implementar. El próximo paso en el filtrado de un pixel de la imagen con coordenadas x,y es colocar en uno de los arreglos los N pixeles que se utilizarán en el filtrado como lo define la máscara de filtrado H. Luego se hace uso de un algoritmo para ordenar los elementos del arreglo desde el mínimo hasta el máximo. Cada elemento del arreglo será multiplicado por el coeficiente del filtro A_i como lo mostró la figura anterior. La salida del filtro será la sumatoria de los pixeles guardados con su peso correspondiente.

La figura 2.3.31 muestra los coeficientes necesarios para desarrollar varios filtros estadísticos. La desventaja de utilizar estos algoritmos es la pérdida de la eficiencia en el procesamiento. EL lector interesado en obtener más información concerniente a los filtros de orden estadístico puede referirse al libro "Filtros Digitales No Lineales de Pitas y Venetsanopoulos" [16].

Filtros Adaptativos

La mayor dificultad de todos los filtros presentados hasta el momento es que operan sobre la totalidad de la imagen de entrada. La mayoría de las imágenes no son homogéneas por lo que son mejor filtradas por filtros que cambian sus características a medida que se mueven por la imagen de entrada. Idealmente el mejor filtrado sería remover el ruido sin causar ningún tipo de degradación.

Considérese la imagen separada en dos regiones, una de fondo y una de bordes. En la región de fondo se pretende realizar el mayor trabajo en la remoción de ruido. Es en estas regiones homogéneas donde el ruido es más notable y objetable. Dentro de la región de bordes se desea mantener la definición de los mismos en toda la imagen. El objetivo del filtro adaptativo es detectar las regiones de bordes y las homogéneas de la imagen y realizar un filtrado exhaustivo en la región homogénea mientras a través de otro filtrado se preservan los bordes.

Tipo de filtro	Coeficientes
Mediana	$A_{N/2} = 0.5$, $A_{N/2-1} = 0.5$ para N par $A_{(N-1)/2} = 1.0$ para N impar
Punto medio	$A_0 = 0.5$, $A_{N-1} = 0.5$ El resto de coeficientes son cero
Máximo	$A_{N-1} = 1.0$ El resto de coeficientes son cero
Mínimo	$A_0 = 1.0$ El resto de coeficientes son cero
Rango	$A_0 = -1.0$, $A_{N-1} = 1.0$ El resto de coeficientes son cero
Quasi-rango(i)	$A_i = -1.0$, $A_{N-1-i} = 1.0$ El resto de coeficientes son cero
Dispersión	$A_i = -1/\text{int}[N/2]$, $A_{N-1-i} = 1/\text{int}[N/2]$ para i=0 hasta int[(N-1)/2]
Alfa promedio ajustado complementario (p)	$A_i = 1/(2p+2)$ para i=0 hasta p e i=N-p-1 hasta N-1 El resto de coeficientes son cero
Alfa promedio ajustado	$A_i = 1/(N-2p)$ para i=p hasta N-p-1 El resto de coeficientes son cero
Promedio	$A_i = 1/N$ para i=0 hasta N-1

Figura 2.3.31. Coeficientes requeridos para realizar varios filtros de orden estadístico

El filtro mínimo error cuadrático medio (MMSE de sus siglas en inglés) cambia sus características en función de la varianza alrededor del pixel $f(x,y)$. La varianza es una buena medida de la presencia de un borde en una región local. Para varianzas locales elevadas, el MMSE no realiza filtrado, en tanto que para varianzas locales bajas aplica un filtrado promedio estándar. Aunque este tipo de filtro falla para los tipo de ruidos outlier, debido a la similitud de características entre éste y los bordes. Las ecuaciones que se presentan a continuación describen este tipo de filtro:

$$m_f(x,y) = \frac{1}{N_t} \sum_{i,j \in H} \sum g(x-i, y-j) \tag{2.3-81}$$

$$\sigma_g{}^2(x,y) = \frac{1}{N_t} \sum_{i,j \in H} \sum [g(x-i, y-j) - m_f(x,y)]^2 \tag{2.3-82}$$

$$\text{est}(x,y) = m_f(x,y) \tag{2.3-83}$$

Los filtros adaptativo medio de doble ventana modificados ajustados (DW-MTM de sus siglas en inglés) y adaptativo medio alfa ajustado utilizan un filtro promedio para las regiones homogéneas, y uno de mediana para las regiones de bordes. De esta manera busca mejorarse las características de los mismos para hacerlos menos sensibles a los más frecuentes tipos de ruido.

Por último mencionaremos al filtro de ventana adaptativa en borde detectado (AWED de sus siglas en inglés), que modifica el tamaño de la ventana siempre que se detecte un borde. Inicialmente el tamaño de la ventana se fija en su valor máximo de *nxn*. Luego, se computa un histograma local y las muestras adyacentes a los niveles de gris mínimo y máximo se investigan para verificar la presencia de cualquier pixel outlier. Si se hallaran de esos pixeles, se marcarían y no se utilizan en la detección de bordes en la ventana local. Si se detectara un borde, el tamaño de la ventana se reduce en uno hasta una de 3x3 o hasta no

detectar bordes. En este momento, si la ventana se encuentra en un tamaño de 3x3 se realiza un filtrado de mediana de 3x3 o un filtrado promedio para el tamaño actual de ventana. En las regiones homogéneas de fondo, el tamaño de la ventana se incrementa y se aplica un filtrado promedio con ventana de gran tamaño para reducir cualquier ruido que se presente.

Para mayor información y una excelente cobertura de los filtros adaptativos, el lector puede referirse a "Filtros Digitales No Lineales: Principios y Aplicaciones de I. Pitas y A. N. Venetsanopoulos" dado en la bibliografía [16].

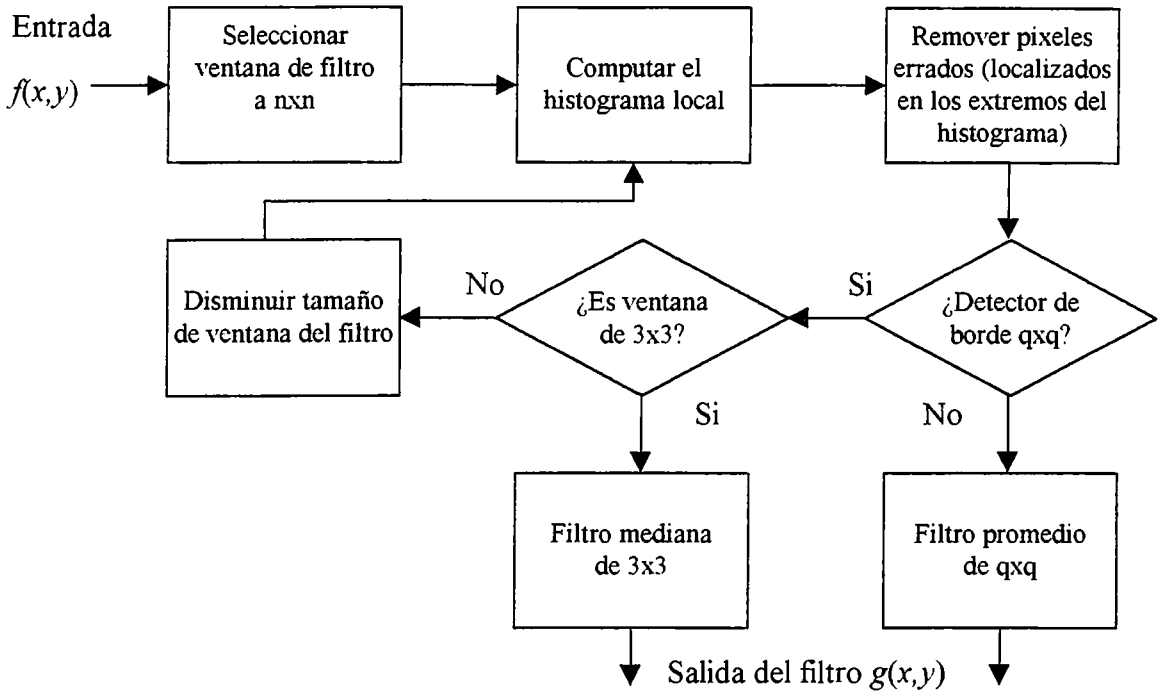


Figura 2.3.32. Diagrama en bloques de un filtro AWED (Adaptado de Pitas y Venetsanopoulos, 1990)

Filtros Homomorfos

Los métodos de Fourier en la frecuencia son usuales para remover el ruido en una imagen. El único requerimiento es que el ruido sea lineal. En otras palabras, el ruido debe

ser aditivo e invariable en el espacio. Las imágenes que han sido degradadas por ruido multiplicativo no pueden tratarse directamente utilizando métodos de Fourier. El objetivo de los filtros homomorfos es transformar la degradación multiplicativa en una aditiva que pueda ser tratada por los métodos de frecuencia de Fourier. Los filtros homomorfos se utilizan para reducir el ruido mutiplicativo y remover las sombras en imágenes pobremente iluminadas.

Considérese la generación de una imagen definida por la siguiente expresión

$$f(x,y) = i(x,y)\, r(x,y) \tag{2.3-84}$$

donde $r(x,y)$ es el poder reflexivo del objeto en cuestión e $i(x,y)$ es la fuente de iluminación. Para una imagen iluminada pobremente, $i(x,y)$ varía espacialmente a través de la imagen $f(x,y)$, haciendo partes de la imagen más oscura y otras más claras. Si consideramos el caso del ruido multiplicativo, $r(x,y)$ puede definir la imagen original y el ruido quedaría definido en $i(x,y)$. El diagrama mostrado en la siguiente figura destaca como un filtro homomorfo cambia el proceso multiplicativo en uno aditivo a través del logaritmo natural. La transformada discreta de Fourier de este logaritmo entrega componentes de frecuencia espacial que pueden ser tratados por el filtro espacial $H(m,n)$.

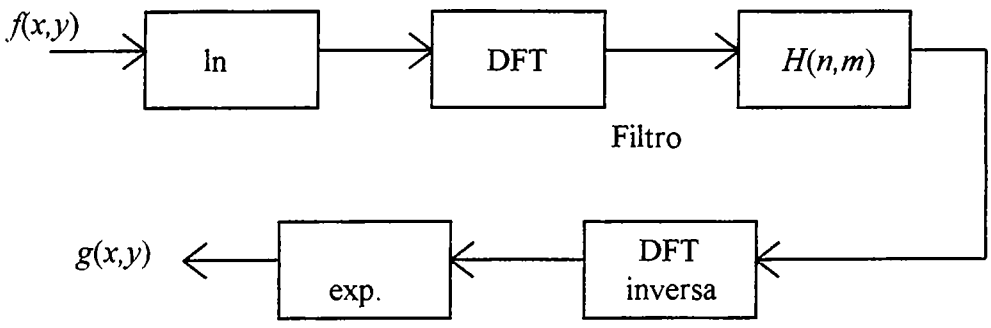


Figura 2.3.33. Diagrama en bloques de un filtro homomorfo.

Capítulo 3

Experimentos de iluminación

En este capítulo se estructurará una serie de lineamientos para verificar cada una de las técnicas de iluminación. Se dará a conocer algunas de las características más sobresalientes de cada una de ellas y además se mencionarán algunos de los resultados que se obtienen cuando se ilumina un determinado objeto con una técnica específica.

3.1 PROPÓSITO Y OBJETIVOS DE LOS EXPERIMENTOS DE ILUMINACIÓN

El propósito que se persigue es obtener diferentes resultados aplicando distintas técnicas de iluminación a escenas particulares, dejando armado una serie de experiencias de laboratorio que permitan comprobar las conclusiones a las que se arribe. Los principales objetivos se mencionan a continuación:

- Reconocer la importancia de la correcta iluminación de la escena en la obtención de una imagen.

- Comprobar la utilidad de las técnicas de iluminación que se desarrollarán.

- Determinar las características más relevantes del ambiente para tenerlas en cuenta al montar una escena.

- Establecer las características que se desean resaltar o analizar de la escena.

- Comprobar las limitantes de las técnicas de iluminación que se experimentan.

3.2 EXPERIMENTACIÓN

Se mostrarán seguidamente tres técnicas de iluminación ampliamente utilizadas, la iluminación direccional, la difusa y por último la de contraluz. Si bien se presenta un desarrollo separado de cada una de las tres experiencias de iluminación, la estructura de éstas será la misma y se detalla a continuación:

- Características esenciales de la técnica de iluminación
- Materiales y equipo
- Procedimiento
- Análisis de resultados
- Conclusiones y recomendaciones

3.2.1 ILUMINACIÓN DIRECCIONAL

Características Esenciales de la Iluminación Direccional

- Los rayos de luz son dirigidos hacia la escena en una sola dirección, por lo general cuentan con una lente colimadora, aunque hoy en día las lámparas halógenas ofrecen una buena solución para aplicaciones en áreas de trabajo de pequeña dimensión.

- Por ende se aplica para trabajar con objetos pequeños.

- Debido a su direccionalidad, el sombreado que produce muchas veces dificulta obtener una aceptable apreciación en ciertos detalles de la escena.

- Se aplica adecuadamente a objetos rugosos.

Materiales y Equipo

- Cámara de CCD con su respectivo pie (soporte para elevar)
- Tarjeta interfaz Pixci-SV4
- Programa de aplicación en procesamiento de imágenes digitales
- Luz direccional halógena
- Objetos a colocar en la escena

Procedimiento

1. Conecte el equipo a su respectiva alimentación y prepárelo para poder realizar las tomas de las imágenes.

2. Coloque el dispositivo de iluminación de forma que ilumine de la forma más directa posible y en el mismo sentido en que se ha ubicado la cámara de CCD.

3. Encienda la computadora, luego la cámara y espere, para que después inicie la sesión del programa de procesamiento de imágenes.

4. Ubique adecuadamente el objeto (disipador) en la escena y ajuste los controles de apertura de diafragma y foco para obtener una imagen adecuada.

5. Capture una imagen y guárdela.

6. Modifique la dirección de iluminación y vuelva a tomar otra imagen.

7. Compare los resultados obtenidos. ¿Observa diferencias en las imágenes?. Explique.

8. Pruebe de colocar objetos con diferentes características volumétricas (objetos planos, con relieves, etc.)

9. Repita los pasos 6 y 7.

Análisis de Resultados

Se han tomado dos imágenes con diferentes objetos en las escenas. Presentamos a continuación cada una de ellas y sus características relevantes.

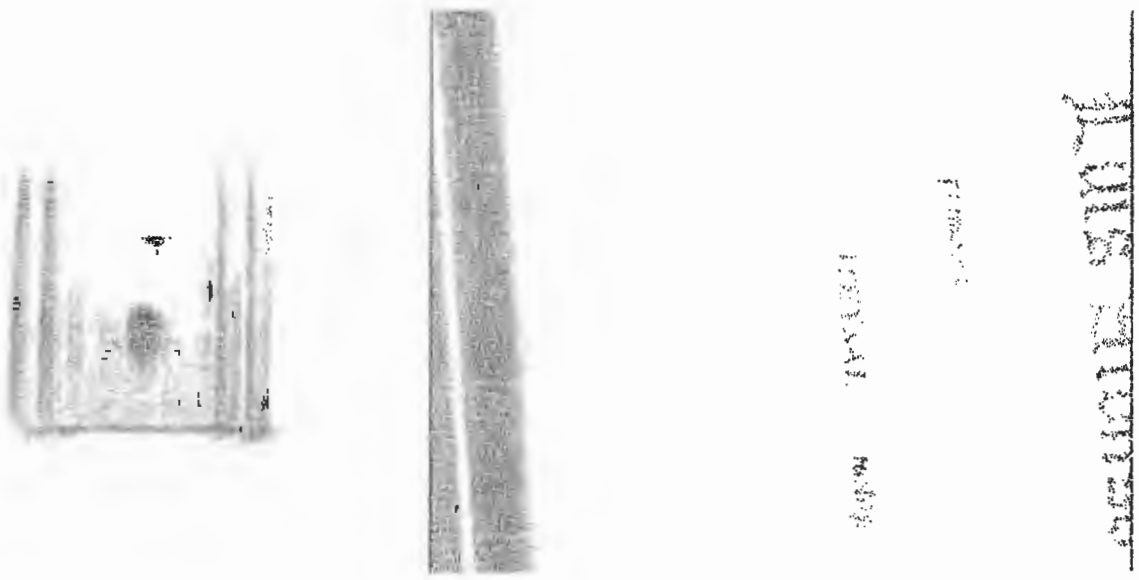


Figura 3.2.1. Iluminación direccional.

Diploma

- Claramente se distinguen los trazos y letras contenidas en él.

- No se perciben reflejos de ninguna clase.

Disipador

- Claramente puede observarse una sombra producida por el relieve del objeto.

- A pesar de la uniformidad del nivel de gris del objeto, pueden distinguirse los relieves dentro del objeto mismo.

- Aparecen algunos reflejos en las aletas del disipador.

Conclusiones y Recomendaciones

- Permite obtener un adecuado contraste de la escena, aunque debe cuidarse de la presencia de sombras que disimulen características importantes presentes en la misma.

- La presencia de un sombreado en la imagen permite conocer orientaciones de los objetos, calcular alturas y distancias mediante la utilización de algoritmos adecuados.

- La iluminación direccional permite minimizar la presencia de reflejos que modifican la imagen real de la escena.

- Es adecuada para aplicaciones en control de calidad de piezas, por ejemplo, realizar un análisis de presencia de etiquetas, códigos de barras, etc.

3.2.2 ILUMINACIÓN DIFUSA

Características Esenciales de la Iluminación Difusa

- Iluminación de la escena en todas las direcciones.

- Debidamente sincronizada permite realizar tomas en distintas direcciones y realzar los contornos.

- Permite obtener un modelo tridimensional de la escena conociendo la posición espacial de la cámara relativa a la escena.

- No proporciona un máximo contraste.

Materiales y Equipo

- Cámara de CCD con su respectivo pie (soporte para elevar)
- Tarjeta interfaz Pixci-SV4
- Programa de aplicación en procesamiento de imágenes digitales
- Luces y difusor
- Objetos a colocar en la escena

Procedimiento

1. Conecte el equipo a su respectiva alimentación y prepárelo para poder realizar las tomas de las imágenes.

2. Coloque el dispositivo de iluminación de forma que ilumine de la forma más homogénea posible la escena.

3. Encienda la computadora, luego la cámara y espere, para que después inicie la sesión del programa de procesamiento de imágenes.

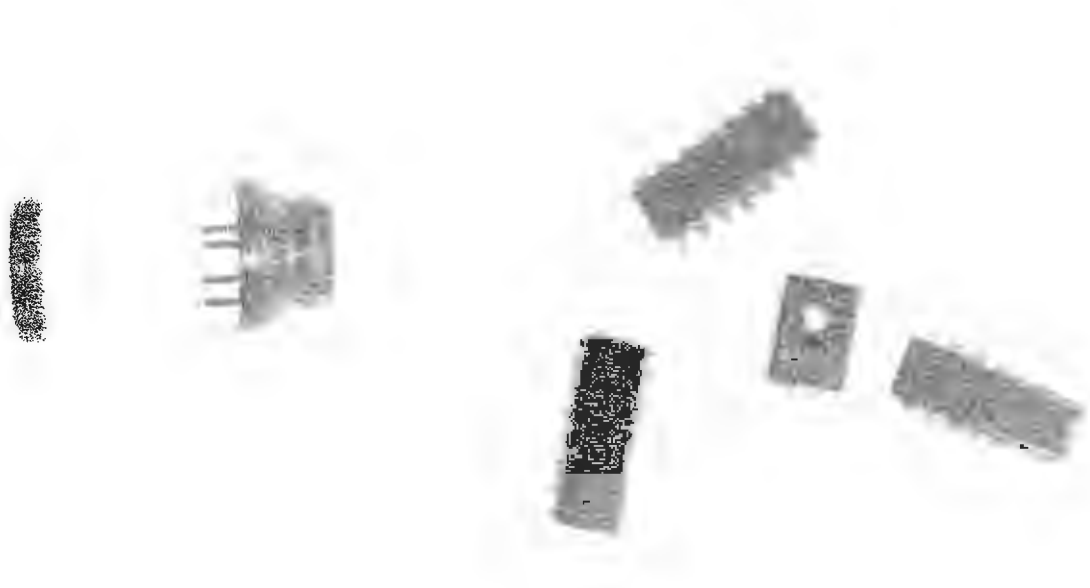
4. Ubique adecuadamente el objeto en la escena y ajuste los controles de apertura de diafragma y foco para obtener una imagen adecuada.

5. Capture una imagen y guárdela.

6. Pruebe de colocar objetos con diferentes características volumétricas (objetos planos, con relieves, etc.). Explique lo que observa.

Análisis de Resultados

Se han tomado tres imágenes con diferentes objetos en las escenas. Presentamos a continuación cada una de ellas y sus características relevantes.



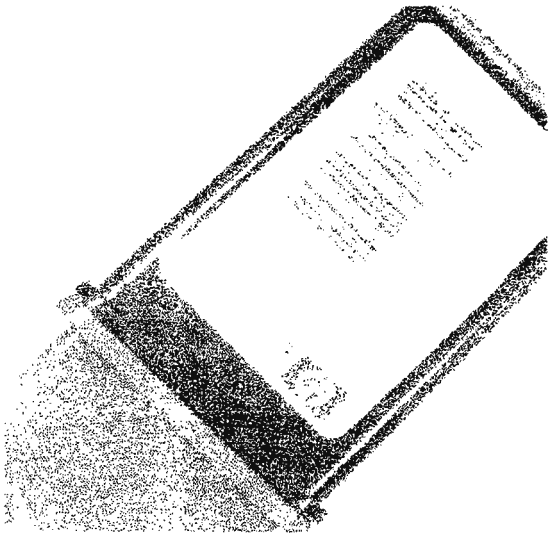


Figura 3.2.2. Iluminación difusa.

Bote de medicina

- Se puede observar que no se percibe la existencia de un tapón.
- Los laterales de la viñeta tampoco llegan a distinguirse con claridad.
- Se presentan ciertos reflejos en el vidrio del envase.

Cartucho de tinta

- Se presentan ciertos reflejos en los laterales del envase.
- Se ha perdido uno de los distintivos de color existente en la etiqueta del envase.
- En la etiqueta puede notarse un buen contraste en el código de barras y letras de tonalidad oscura.

Dispositivos integrados

- Las patillas de los dispositivos apenas pueden distinguirse.

- El contraste con el negro es suficiente para analizar la imagen.

Conclusiones y Recomendaciones

- Debido a la difusión de los haces luminosos que llegan hasta los elementos de la escena, los reflejos hacen que se pierda parte de la información presente en la misma.

- Es importante observar detenidamente el tipo de superficie del objeto en la escena en relación con lo mencionado anteriormente. Por consiguiente, esta técnica es utilizable para iluminar superficies rugosas.

- Los niveles de gris cercanos al blanco fácilmente se saturan y por ende no se los distingue entre ellos.

- En las figuras planas no se justifica la utilización de esta técnica, puesto que de ninguna manera se producen sombras.

3.2.3 ILUMINACIÓN A CONTRALUZ

Características Esenciales de la Iluminación a Contraluz

- Se obtienen imágenes con solo dos niveles de gris (binarias).

- Permite reconocer grietas o roturas en piezas pequeñas y tomar sus dimensiones.

- Es adecuada para extraer los contornos o bordes de objetos con una gran facilidad.

- A partir de ella puede realizarse la segmentación de imágenes.

Materiales y Equipo

- Cámara de CCD con su respectivo pie (soporte para elevar)
- Tarjeta interfaz Pixci-SV4
- Programa de aplicación en procesamiento de imágenes digitales
- Dispositivo para iluminar a contraluz
- Objetos a colocar en la escena

Procedimiento

1. Conecte el equipo a su respectiva alimentación y prepárelo para poder realizar las tomas de las imágenes.

2. Coloque el dispositivo de iluminación de forma que tal que sea mínima la luz que incida desde arriba de la escena.

3. Encienda la computadora, luego la cámara y espere, para que después inicie la sesión del programa de procesamiento de imágenes.

4. Ubique adecuadamente el objeto en la escena y ajuste los controles de apertura de diafragma y foco para obtener una imagen adecuada.

5. Capture una imagen y guárdela.

6. Pruebe de colocar objetos con diferentes características volumétricas (objetos planos, con relieves, etc.). Explique lo que observa.

Análisis de Resultados

Se han tomado dos imágenes con diferentes objetos en las escenas. Presentamos a continuación cada una de ellas y sus características relevantes.

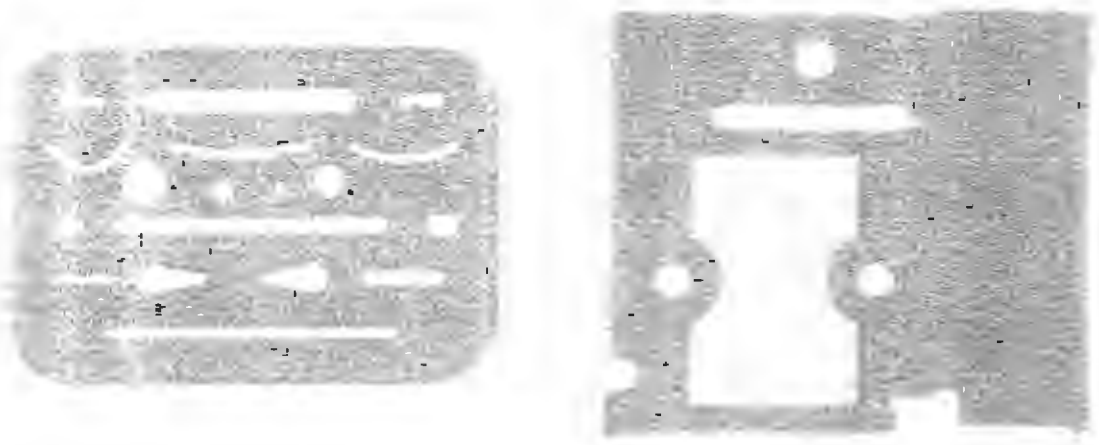


Figura 3.2.3. Iluminación a contraluz.

Plantilla

- Sólo existen dos niveles de gris en la imagen adquirida.
- Se aprecian con una buena claridad los límites del objeto.
- No existen reflejos de ningún tipo.

Disipador

- No se aprecia el volumen del objeto presente en la escena.

- No se perciben sombras.

Conclusiones y Recomendaciones

- Debe minimizarse la iluminación ambiental o proveniente de otras direcciones, de manera tal que sólo predomine la posterior.

- Permite detectar con facilidad las fisuras o roturas presentes en los objetos.

- Es la iluminación recomendada para la detección de bordes.

- Permite realizar el dimensionado de objetos, cálculo de áreas, centros de gravedad, etc.

- Debido al tipo de iluminación se pierde toda la información volumétrica del objeto. Por lo que debe complementarse con otra técnica para poder realizar un control adecuado de la escena.

3.3 EXPERIENCIAS ADICIONALES

Como fue planteado desde un principio en este trabajo de graduación y como último punto de este capítulo se anexará una serie de experiencias con sus respectivos objetivos, las que permitirán, siguiendo el desarrollo consecutivo de las mismas, completar la explotación de las herramientas de procesamiento de imágenes provistas en el programa de aplicación.

Además, se les ha ordenado de forma tal que guardan una relación estrecha con el contenido de la mayoría de los textos que aparecen en las citas bibliográficas.

Experiencia número 1: El sistema de adquisición de imágenes

- Reconocer los elementos del sistema de adquisición y procesamiento de imágenes digitales.

- Familiarizarse con los parámetros básicos del equipamiento para adquisición de imágenes.

- Adquirir experiencia básica en el manejo del programa de aplicación en procesamiento de imágenes digitales.

Experiencia número 2: Óptica e iluminación

- Comprobar las diferencias en la adquisición de imágenes modificando la cantidad de niveles de cuantización y la cantidad de pixeles de la imagen original.

- Reconocer la degradación de la calidad de las imágenes modificando el brillo, contraste, la apertura de diafragma y el foco.

- Comprobar los resultados de aplicar diferentes técnicas de iluminación como: contraluz, direccional y difusa.

Experiencia número 3: Histogramas y su aplicación

- Aplicar las técnicas de histogramas para determinar el contraste y el brillo en imágenes de niveles de gris.

- Reconocimiento del entorno y objeto a partir de la umbralización de histogramas.

- Obtención de una imagen con zonas resaltadas haciendo uso del falso color.

- Diferenciar claramente entre umbralizar y detectar bordes.

Experiencia número 4: Filtrado espacial y en el dominio de la frecuencia

- Identificar algunos de los diferentes tipos de ruido que pueden estar presentes en una imagen.

- Aplicación de técnicas de filtrado espacial y en el dominio de la frecuencia. La transformada rápida de Fourier (FFT) y su inversa.

- Observar las diferencias en las imágenes resultantes de aplicar máscaras de convolución con variadas características.

- Ser capaz de seleccionar un filtro adecuado.

Experiencia número 5: Tratamiento de imágenes binarias

- Aplicación de técnicas de procesamiento de imágenes a imágenes binarias.

- Comparar el resultado obtenido de aplicar diferentes técnicas en la detección de bordes.

- Realizar la segmentación de una imagen a partir de la aplicación de las técnicas de procesamiento de imágenes estudiadas.

Capítulo 4

Desarrollo del programa de aplicación

Se considera muy importante el lenguaje de programación que provee el programa de aplicación para el procesamiento de imágenes digitales. Uno de los aspectos que se han tomado en cuenta para la presentación del producto final, es que en el diseño del sistema se incluya una interfaz gráfica de usuario, con el objeto de que la aplicación sea lo más amigable posible. Una de las alternativas que se adapta en gran medida a tal objetivo es un poderoso lenguaje de programación de alto nivel como lo es Visual C++ versión 5.0, por lo tanto, todas las rutinas que se presentan durante el desarrollo de este capítulo se han editado en el código que corresponde al lenguaje ya mencionado.

4.1 EL BANCO DE TRABAJO DE VISUAL C++ VERSIÓN 5.0

Estrictamente esta versión forma parte de un conjunto de lenguajes de programación de alto nivel, como por ejemplo Visual Basic, Visual Java, etc. Todos ellos agrupados en lo que se le ha dado por llamar *Visual Studio '97.*

Una de las razones más importantes que se han evaluado para elegir tal lenguaje de programación, es el hecho de tener una interfaz gráfica que sólo el sistema operativo de Windows ofrece. Además, las librerías de la tarjeta digitalizadora están desarrolladas en lenguaje "C" y éste último es compatible con la versión sucesora, que en la actualidad es la herramienta que se utilizará para llevar a cabo cada una de las rutinas que se han incluido en

el desarrollo del proyecto. A continuación se presenta el banco de trabajo del lenguaje de programación Visual C++:

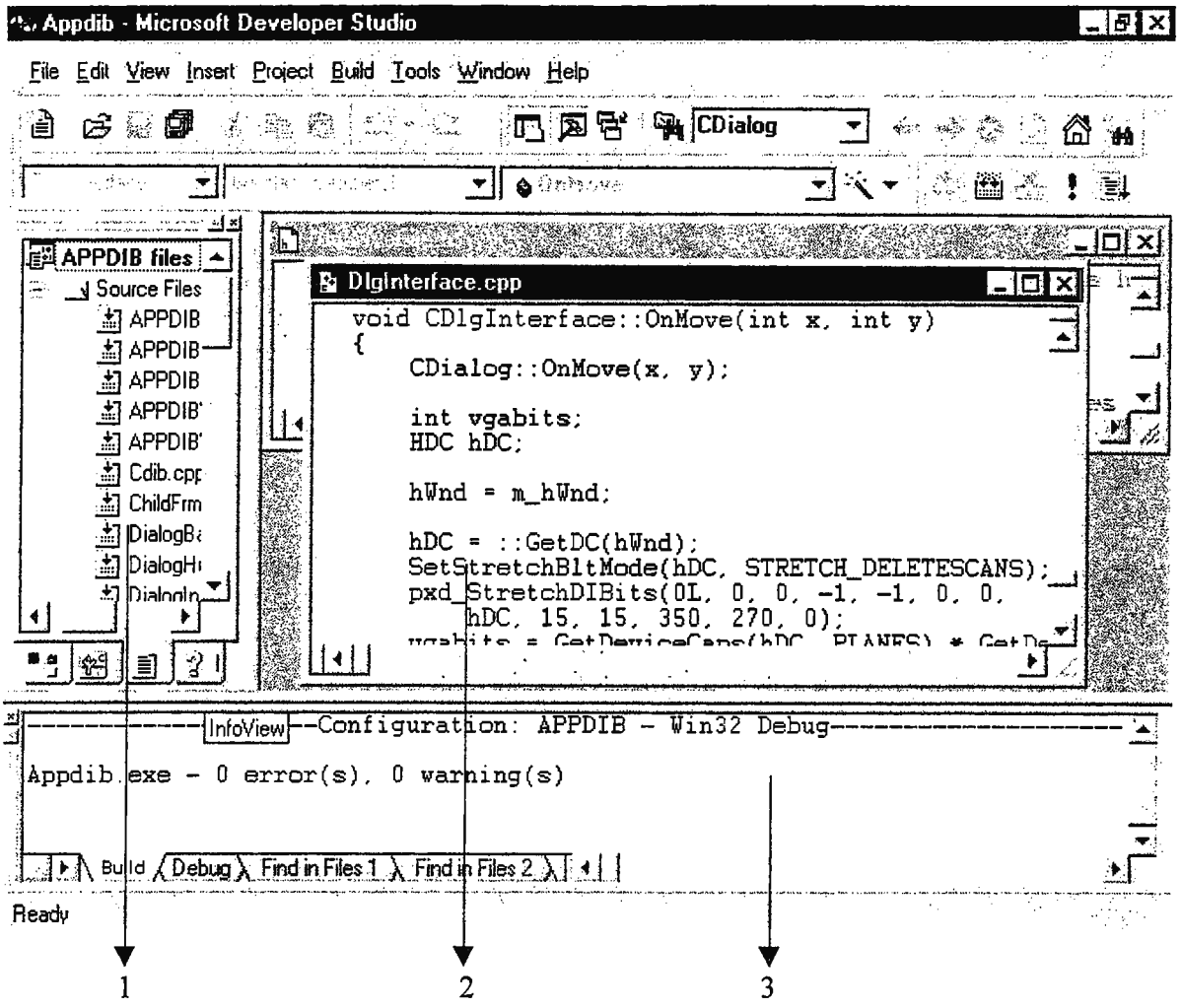


Figura 4.1.1. El banco de trabajo de Visual C++ versión. 5.0.

Básicamente este ambiente de trabajo posee tres ventanas principales que son:

1- Tal como se indica en la figura 4.1.1, en esta ventana se encuentran cuatro posibilidades de vista: a) De izquierda a derecha, en la primera solapa se encuentra el listado de las clases que pertenecen a la aplicación global y el listado de las funciones que forman parte de cada una de las clases, indistintamente de la sección de clase a la que pertenece dicha función. b) En la segunda solapa, se presenta cada uno de los recursos que forma parte de la aplicación.

Se llama recurso a un elemento del programa como lo es el menú principal, un icono, un cuadro de diálogo, una barra de herramientas, etc. c) Como tercera solapa se tiene una vista de los archivos que contienen el código fuente del programa en proceso de diseño. En esta vista se incluyen todos los archivos CPP (codificación) y los H (cabecera). d) Por último la solapa que pertenece a la vista de la ayuda que proporciona el *Developer Studio*. Ésta contiene información valiosa desde el punto de vista del programador ya que en ella se puede encontrar una guía muy completa que respalda la programación en Visual C++.

2- La ventana de edición de los códigos fuentes que forman parte de la aplicación a construir. En esta ventana se editan los archivos *.cpp y los archivos de cabecera *.h. Una clase que define una categoría de objeto está caracterizada por un archivo *.cpp unido con el archivo *.h, término que se utiliza en la programación orientada a objetos; en todos los casos los datos y funciones miembros de la clase madre son heredadas a las clases hijas. Un objeto comparte los mismos atributos y la misma funcionalidad con los demás objetos que pertenecen a la misma clase.

Cada clase está conformada por datos y funciones que se definen en una de las tres secciones de la clase, para el caso las secciones de clase son las siguientes: *privada* en la que sólo las funciones miembro de la clase pueden accesar a los miembros privados. En la sección *protegida* sólo las funciones miembros de la clase y sus clases descendientes pueden accesar a miembros protegidos. En la sección *pública* se especifican miembros que son visibles a las funciones miembros de la clase, a las instancias de la clase, a las funciones miembro de clases descendientes y a sus instancias mismas [23].

3- La ventana de salida. En esta ventana se presentan los errores de construcción o de enlace al momento de producir el archivo ejecutable. Como es posible observar en la figura 4.1.1, esta ventana tiene solapas en las que el usuario tiene la posibilidad de llamar a una sub-ventana asociada a la solapa en cuestión. Por ejemplo, si se está ejecutando la operación de depuración a la aplicación que está siendo diseñada, la ventana asociada es la de depuración o del inglés *Debug*.

Este tipo de programación se basa en la creación de un proyecto, el cual puede ser una aplicación de consola, una aplicación para Windows, o bien el diseño de un archivo ejecutable a partir de las MFC (del inglés Microsoft Fundation Classes) de Windows, etc. Cuando se crea un proyecto, toda la información del mismo se almacena en un archivo *.dsw, construido automáticamente por Developer Studio. Cuando el usuario desee realizar una modificación al proyecto (aplicación en proceso de diseño), deberá abrirlo con anterioridad, éste último se encargará de actualizar el banco de trabajo como el que se presenta en la figura 4.1.1.

4.2 EL BANCO DE TRABAJO DEL MARCO DE LA APLICACIÓN

Aparte de otras ventajas que presenta Visual C++ ante otros lenguajes de programación, se puede hacer mención de la característica modular para la programación basada en la creación de clases, además, porque hace uso de las utilerías de la MFC. Lo antes expuesto significa básicamente dos cosas: una de ellas es que el programador no debe editar tediosamente aquellos códigos que se encargan de la creación de una clase en particular. Y segundo, el proceso antes mencionado se lleva a cabo de forma automatizada por medio de la facilidad que presenta la utilería *ClassWizard*.

Particularmente, es una herramienta muy versátil y que entre otras funciones, se encarga de agregar una clase a un recurso de diálogo recién creado. Además, elimina o agrega funciones miembro de clase, o en todo caso agregar o eliminar variables miembro de clase, etc. Si no se tiene experiencia en esta programación, se recomienda utilizar la opción de automatización para los aspectos antes expuestos [9].

Si se opta por la creación de un nuevo proyecto a través de MFC *AppWizard* (exe) dentro del banco de trabajo del lenguaje de programación, se tendrá una serie de elementos iniciales que pertenecerán al marco de la aplicación. Es decir, con solo haber elegido la

opción antes descripta se tendrá de manera automatizada la creación del banco de trabajo que se presenta en la figura 4.2.1. Esta es una prueba de la potencialidad que nos proporciona este lenguaje de programación.

Una de las características que posee el marco de la aplicación desarrollado, es que posee una interfaz de documentos múltiples o mejor conocido como MDI (del inglés Multiple Document Interface), ésta permite abrir las ventanas hijas que el usuario estime conveniente. Esta característica se puede elegir durante el proceso de la creación de un nuevo proyecto tal como anteriormente se explicó.

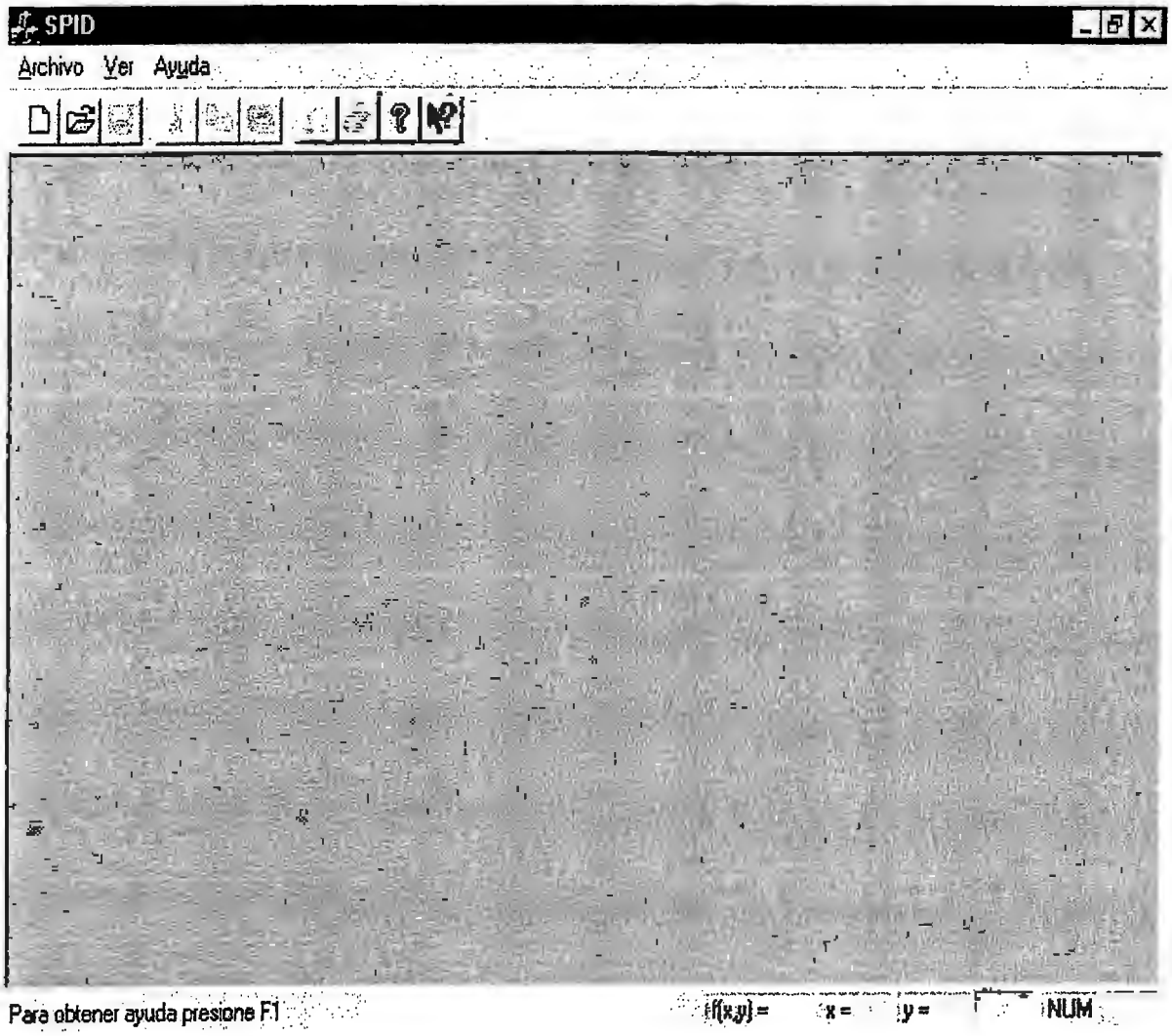


Figura 4.2.1. Banco de trabajo del marco de la aplicación.

El programador es el responsable del resto del código para darle funcionalidad a la aplicación mínima que se crea por medio de las utilerías antes mencionadas y que se ejemplifica en la figura 4.2.1.

4.2.1 MENÚ DEL MARCO DE LA APLICACIÓN

Básicamente se tienen activos dos menús: uno que aparece por defecto dentro del marco de la aplicación así como se presenta en la figura 4.2.1. Éste tiene validez cuando aún no se ha abierto un documento, al cual se le llamará imagen, el menú restante se vuelve activo durante la apertura de una imagen. Se centrará la atención en éste último ya que es en este recurso donde se encuentran las opciones del procesamiento de imágenes digitales.

Del Menú sin Documento

Está constituido por las opciones básicas que contienen la mayoría de las aplicaciones existentes, entre las más comunes se puede mencionar: barra de herramientas, barra de estado, ventanas en mosaico y sin faltar la apertura de un documento ya existente que para el presente caso serán archivos en formato BMP (Mapa de Bits), los cuales están formados por tres secciones principales que son: a) Una cabecera que describe la resolución del dispositivo sobre el cual el rectángulo de pixeles fue creado, así como también las dimensiones de rectángulo y de la matriz de bits, b) Una paleta lógica de colores y c) Una matriz de bits que define la relación entre los pixeles en la imagen BMP y las entradas de la paleta lógica [9].

A continuación se presenta la estructura básica de un archivo BMP. Nótese los tres componentes que forman parte de la estructura del formato, la posición que guardan entre ellos y la forma en la que se relacionan.

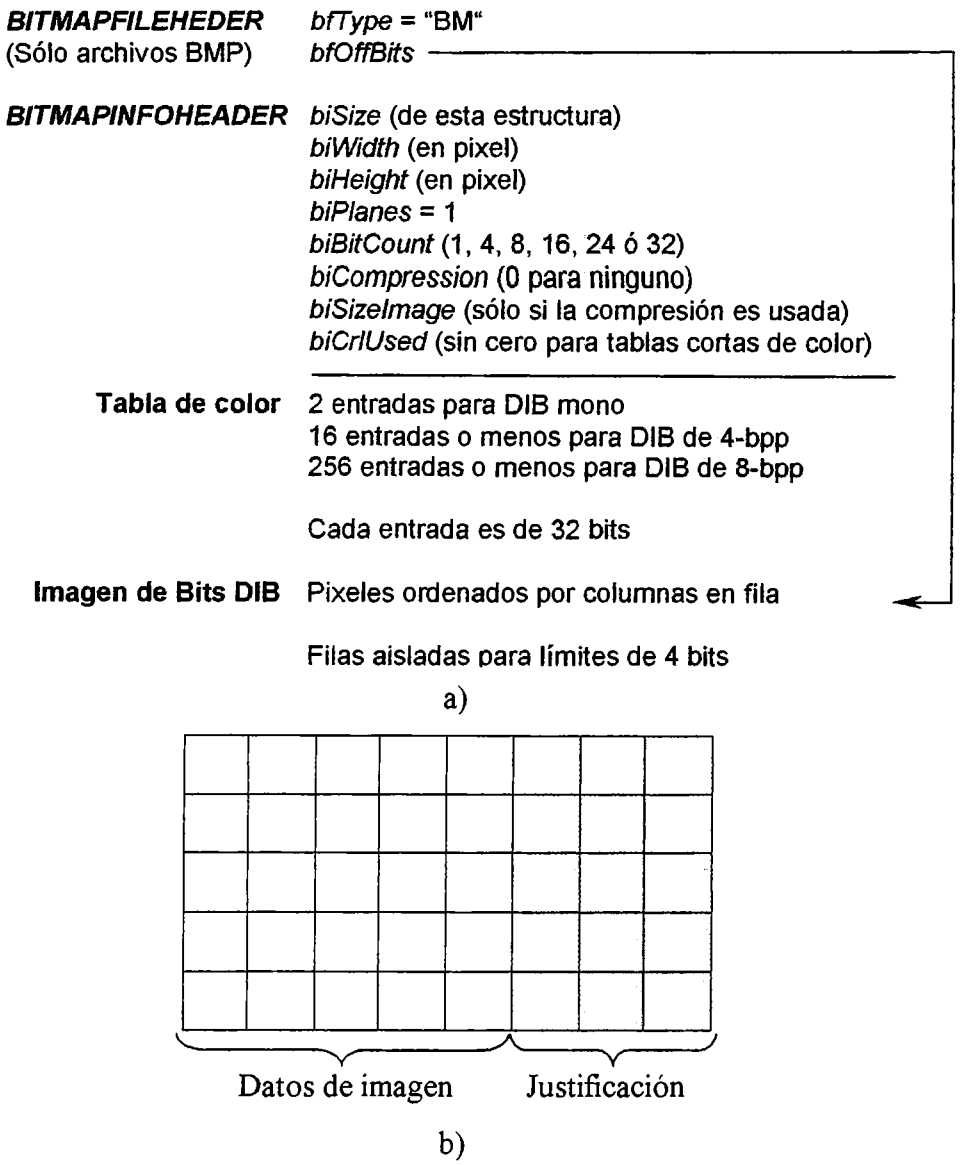


Figura 4.2.2. a) Estructura básica de un archivo BMP. b) Matriz de datos justificados.

Del Menú con Documento

Considerado como el más importante de ambos, ya que éste realiza las llamadas a las rutinas de servicio en virtud de los algoritmos desarrollados para el procesamiento de las imágenes o la llamada a una función propia del sistema. Este menú se activa cuando se abre un documento sobre el área cliente del marco de la aplicación, se le llamará así, al área donde aparecerá las *n* ventanas hijas y se conoce como ventana hija a aquella que contiene la

imagen. En la figura 4.2.3 se destaca el marco de la aplicación y el marco cliente. Puede observarse la modificación de este menú con respecto al menú de la figura 4.2.1.

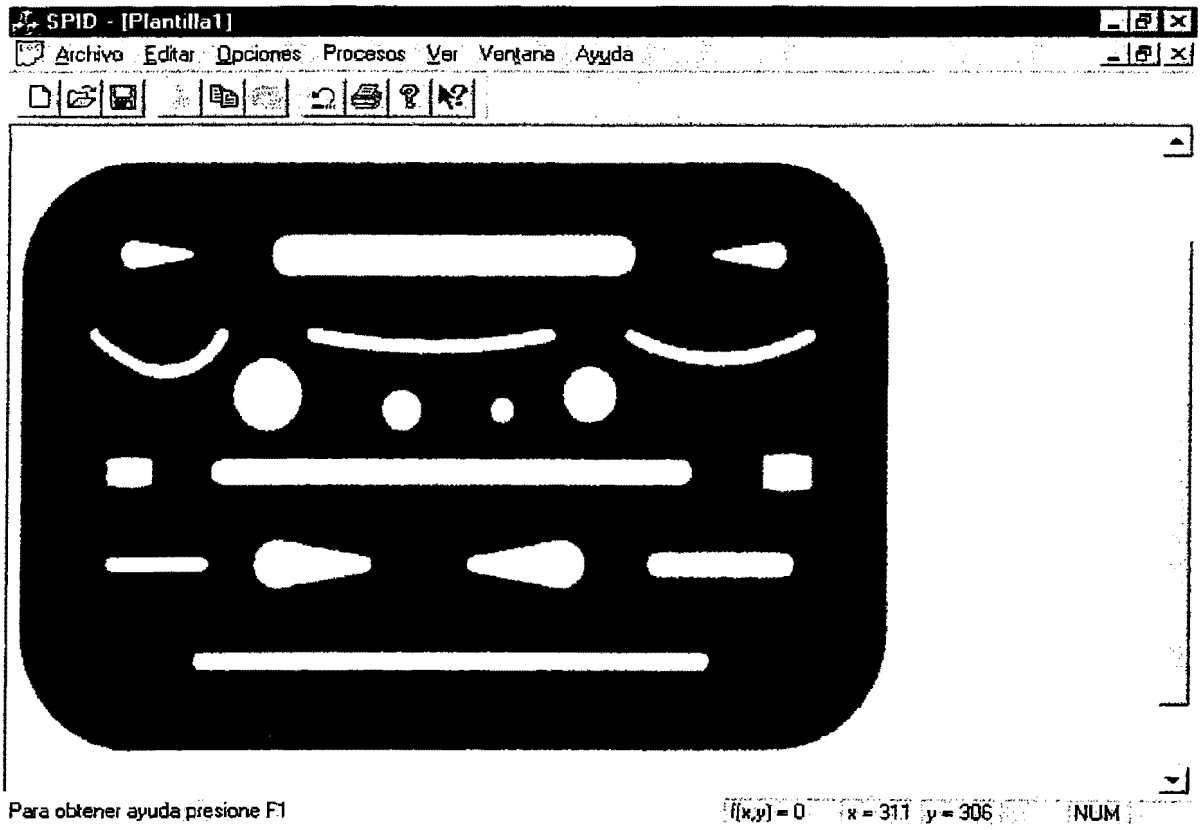
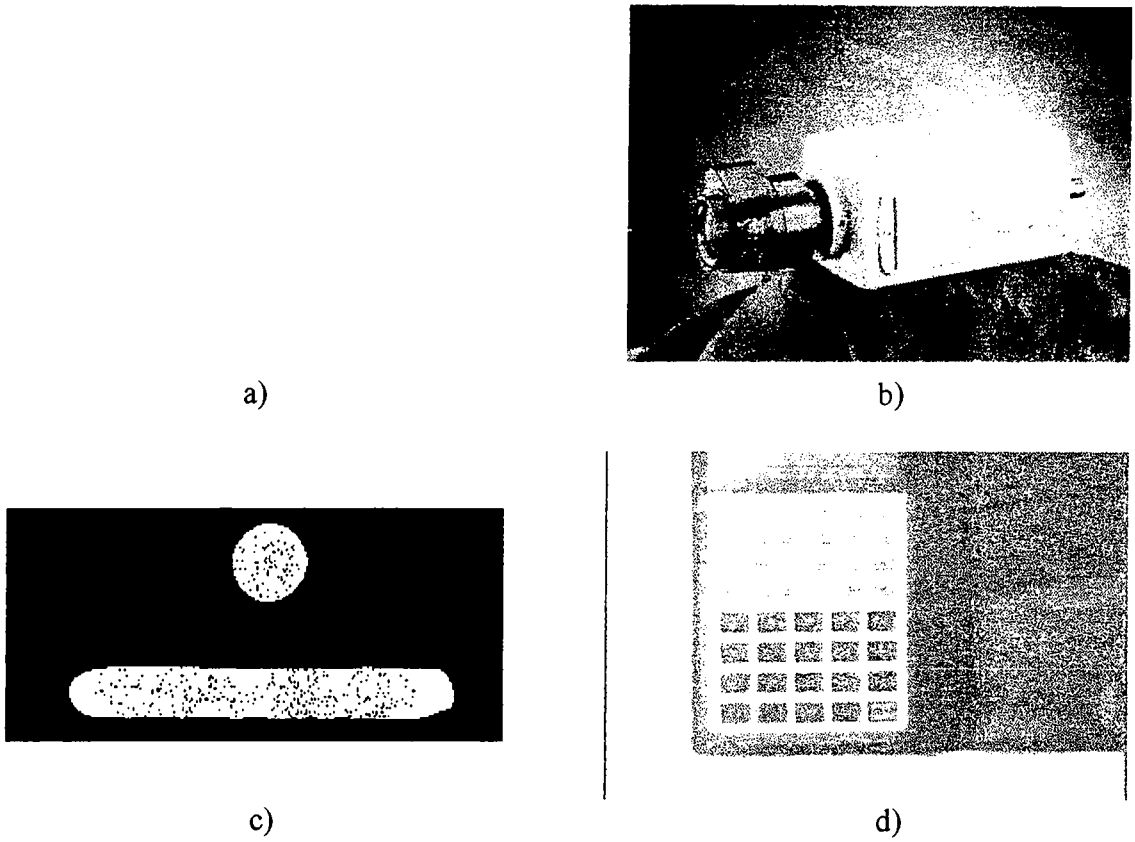


Figura 4.2.3. Marco de la aplicación incluyendo un documento activo.

4.3 DESARROLLO DE ALGORITMOS

Durante esta sección se presentarán los algoritmos que hacen posible la ejecución de cada uno de los procesos aplicados a las imágenes activas dentro del marco de la aplicación. De acuerdo con lo anterior, es necesario evaluar las imágenes que servirán como parámetro de comparación, con el objeto de verificar la evolución de las mismas tras haber aplicado un algoritmo en particular. Un algoritmo cualquiera se ejecuta cuando el usuario elige una opción del menú principal que será aquel que se vuelve y mantiene activo durante la apertura de una imagen con formato BMP.

Algunas de las figuras que se presentan a continuación son imágenes que han sido captadas con el equipo adquirido como parte de la presente solución. Por otra parte, también se tienen imágenes ya existentes como producto de la investigación, pero que se han retomado porque se adaptan en gran medida al modelado de ciertos procesos, como por ejemplo la imagen de la figura 4.3.1(b).



a) b)

c) d)

Figura 4.3.1. Imagen patrón. a) Imagen de plantilla para borrar. b) Imagen de cámara. c) Imagen con ruido de una pieza metálica. d) Imagen de calculadora.

4.3.1 MENÚ ARCHIVO

La sección archivo del menú principal contiene en su interior, funciones que son comunes para la mayoría de los programas de aplicación en procesamiento de imágenes, significa que si el usuario despliega este sub-menú encontrará opciones ya conocidas tal

como se puede observar en la figura 4.3.2. La mayor parte de las rutinas incluidas en la sección archivo, están relacionadas con funciones que manipulan un archivo en particular, desde abrir una imagen ya existente hasta guardar los cambios realizados por medio de las opciones incluidas en otros menús.

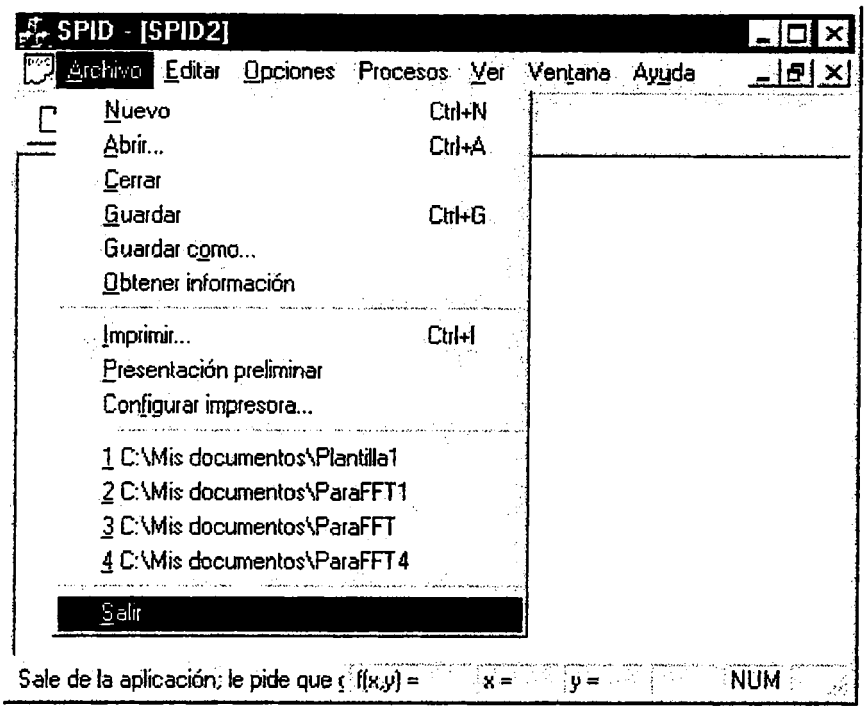


Figura 4.3.2. Despliegue del menú archivo del marco de la aplicación.

4.3.1.1 OBTENER INFORMACIÓN

Información del promedio y la desviación estándar se puede obtener directamente del programa al seleccionar la opción "Obtener información" en el menú "Archivo". Cuando este comando se activa aparece un cuadro de dialogo de información general, como por ejemplo, el número de bits por pixel, la resolución horizontal y vertical, el tamaño en bytes de la imagen, la ruta en la que la imagen ha sido guardada, etc. En la figura 4.3.3 se muestra una imagen del cuadro de diálogo conteniendo el valor del promedio y la desviación

estándar representadas como el brillo y el contraste respectivamente, parámetros que forman parte de la imagen representada por la figura 4.3.1(b). Los componentes de la información de archivo que se consideran más importantes son: el promedio y la desviación estándar, debido a que son funciones utilizadas por otros algoritmos.

Cálculo del Valor Promedio y la Desviación Estándar

Una imagen puede estar caracterizada por su nivel de brillantez y contraste, cuyos valores pueden alterar la percepción que se tenga de ésta en virtud de la manipulación de los niveles de gris de la imagen. El nivel de brillantez es la luminosidad u oscuridad percibida y está definida como el valor promedio de todos los pixeles que componen la imagen. Mientras que el contraste define el rango de variación en los niveles de gris con respecto a su promedio y se mide a través de la variación promediada de los niveles de gris, ésta última es comúnmente conocida como la desviación estándar.

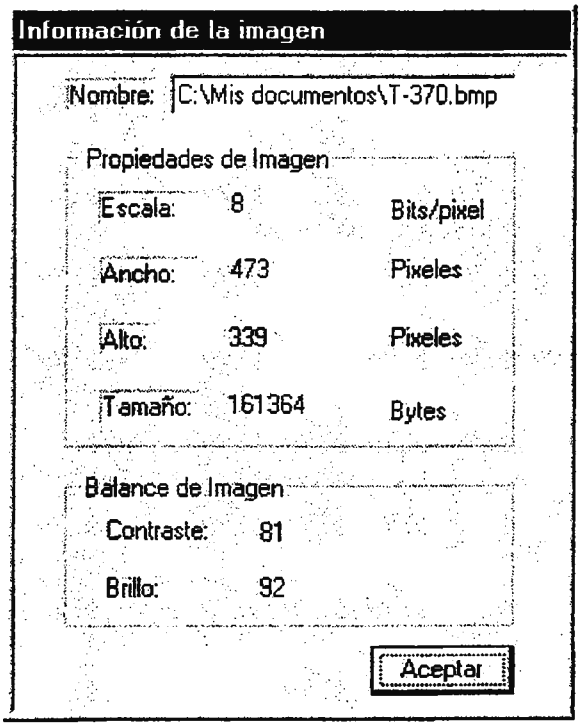
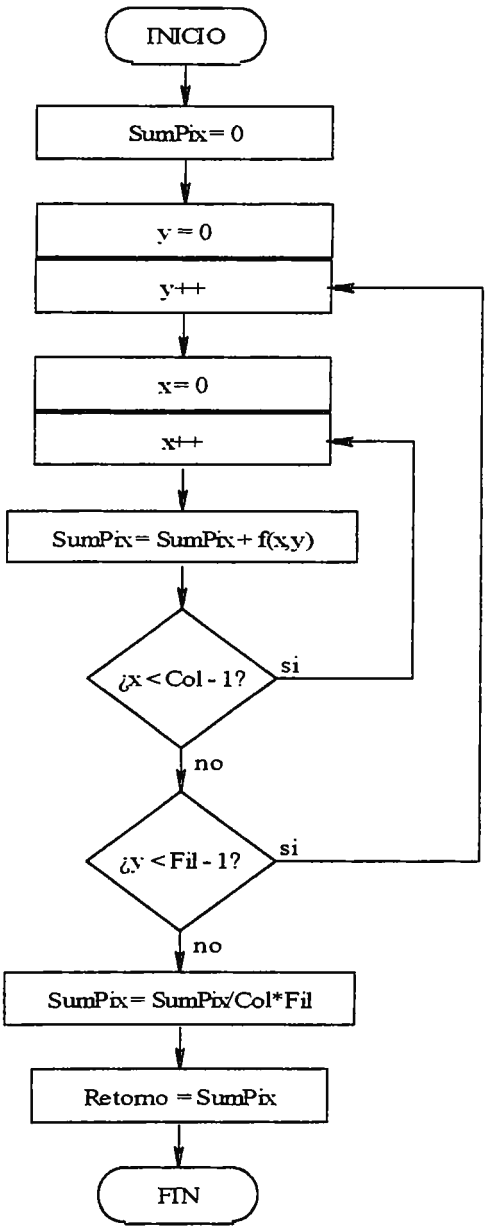
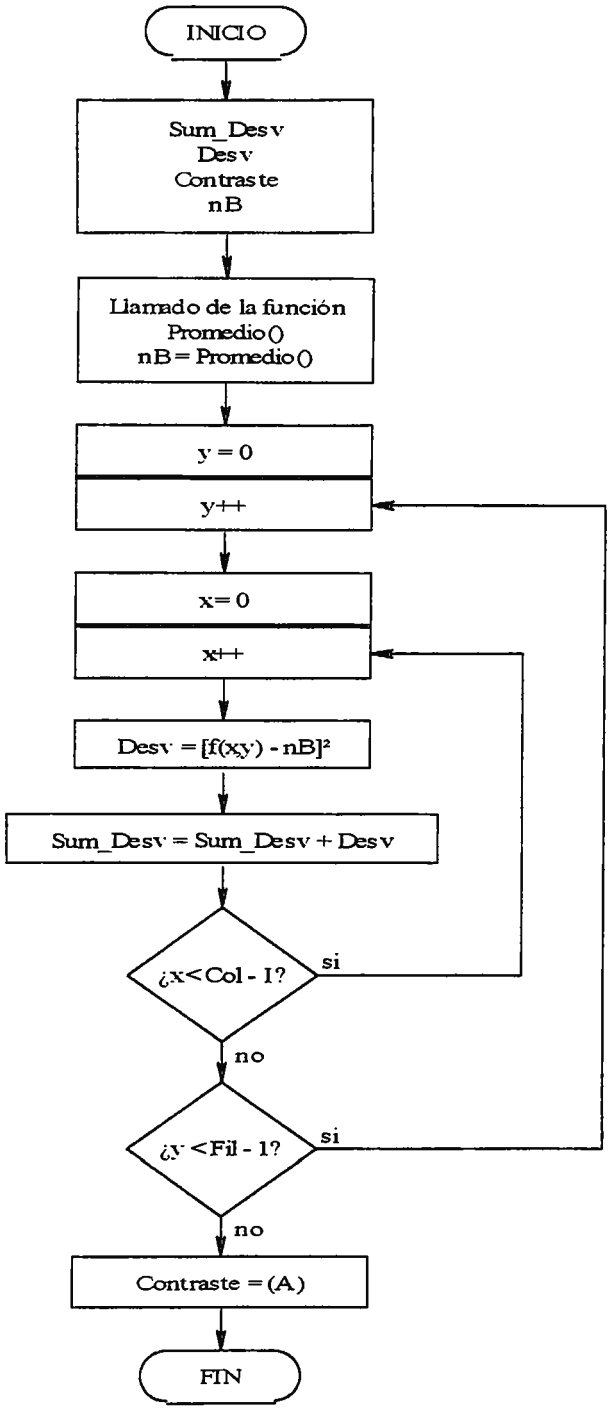


Figura 4.3.3. Cuadro de diálogo de información de imagen.

Para conocer la desviación estándar es necesario con anterioridad conocer el promedio y luego calcular la desviación promedio para cada valor, por lo tanto se vuelve indispensable realizar primero un llamado a la función PROMEDIO(), cuyo retorno es precisamente el valor promedio calculado. Esta forma de programación de hacer llamadas a funciones específicas dentro de otras funciones o procesos es muy común en lenguajes estructurados como el C/C++.



a)



b)

$$(A) = \sqrt{\frac{Sum - Desv}{Fil * Col}} \qquad (4.3\text{-}1)$$

Figura 4.3.4. Diagrama de flujo para obtener a) Promedio y b) Desviación estándar.

En la figura 4.3.4 se muestran ambos flujogramas para obtener el valor promedio y la desviación estándar respectivamente. Nótese que se ha designado a $f(x,y)$ como la función de la imagen y x,y como las coordenadas espaciales. Además, se usan las variables *Fil* (número de filas) y *Col* (número de columnas) para designar las dimensiones de la imagen. La mayoría de los diagramas de flujo utilizan el proceso de notación simplificada así, "*Variable*++" para hacer notar la operación "*Variable* = *Variable* + 1".

4.3.2 MENÚ EDICIÓN

En este menú, se encuentran opciones que se encargan de modificar la imagen para que posteriormente se compruebe un proceso en particular. Por ejemplo, con este menú se tiene una alternativa para agregar un tipo de ruido a la imagen en cuestión y de esa manera poder comprobar el resultado del filtrado espacial que se ubica en el menú procesos.

En la siguiente figura se presentan las opciones que el usuario puede utilizar para realizar una operación de edición.

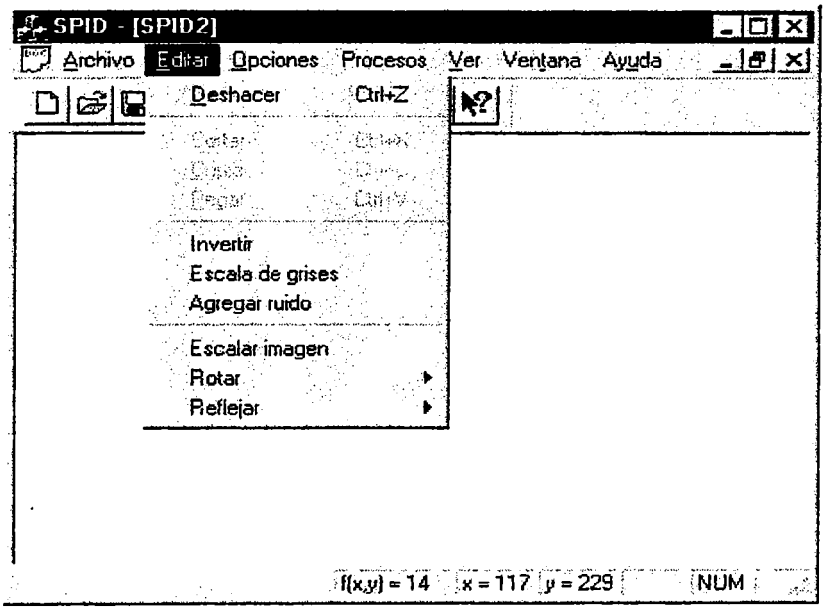


Figura 4.3.5. Despliegue del menú edición del marco de la aplicación.

4.3.2.1 INVERTIR

Es un procedimiento en el que se aplica una función de mapeo a la imagen. Esta función convierte los niveles de gris oscuros de la imagen de entrada en niveles claros, mientras que los niveles claros son convertidos a oscuros en la nueva imagen.

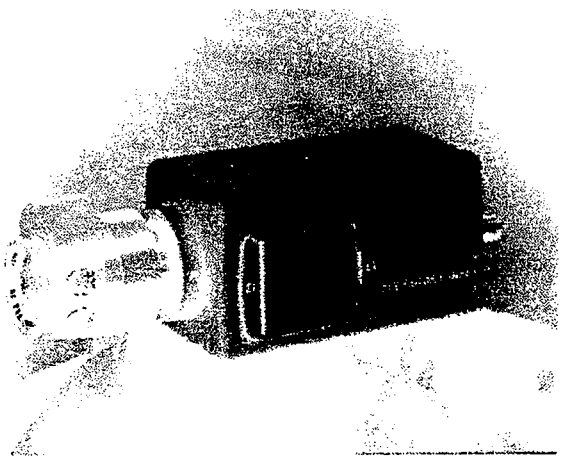
La función de inversión puede implementarse como:

$$P_k = G_{\max} - q_k \tag{4.3-2}$$

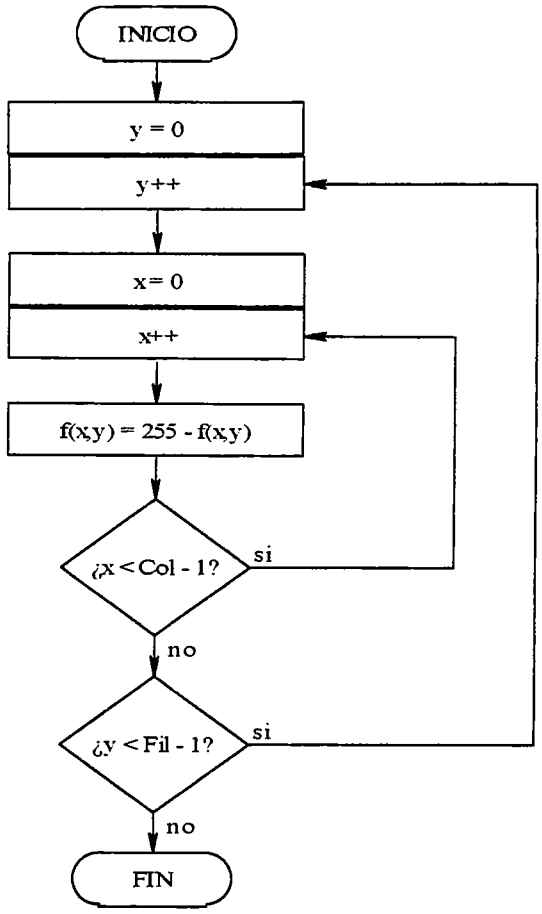
donde G_{max} es igual al máximo nivel de gris permitido (255 para una imagen de 8 bits/pixel). Alternativamente la operación booleana NOT produce los mismos resultados.

Algunas veces a la inversión se le conoce como "el negativo de la imagen" por su similitud con el proceso asociado llevado a cabo en la fotografía. La inversión es útil cuando se desea realzar (incrementar el brillo) áreas oscuras dentro de la imagen.

Al seleccionar la opción invertir del menú principal localizado en "Edición", se ejecuta el procedimiento, cuyo algoritmo se muestra en la figura 4.3.6(b). La figura 4.3.6(a) es el resultado de la aplicación del algoritmo sobre la imagen de la figura 4.3.1 (b).



a)



b)

Figura 4.3.6. a) Imagen resultante de la inversión. b) Diagrama de flujo para la inversión.

4.3.2.2 ESCALA DE GRISES

La percepción de las características de una imagen depende sin ninguna duda de la cantidad de niveles de gris con la que se está representando. Por tal motivo se ha diseñado una rutina que permita al usuario, observar las transformaciones que sufre la imagen al aplicársele cada una de las escalas predefinidas en el algoritmo. Cuando se activa esta opción, se visualiza un cuadro de diálogo en el que se puede definir la escala que se le aplicará a la imagen actualmente activa, éste último se ve representado en la siguiente figura.

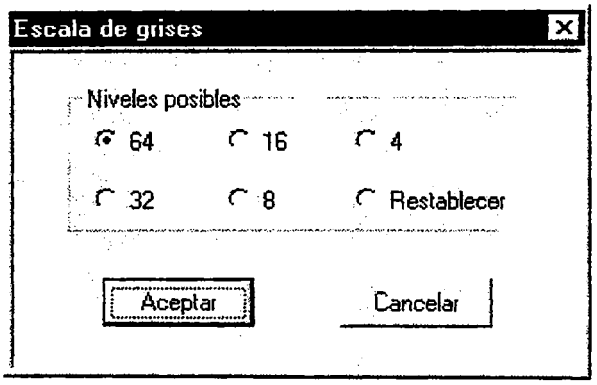
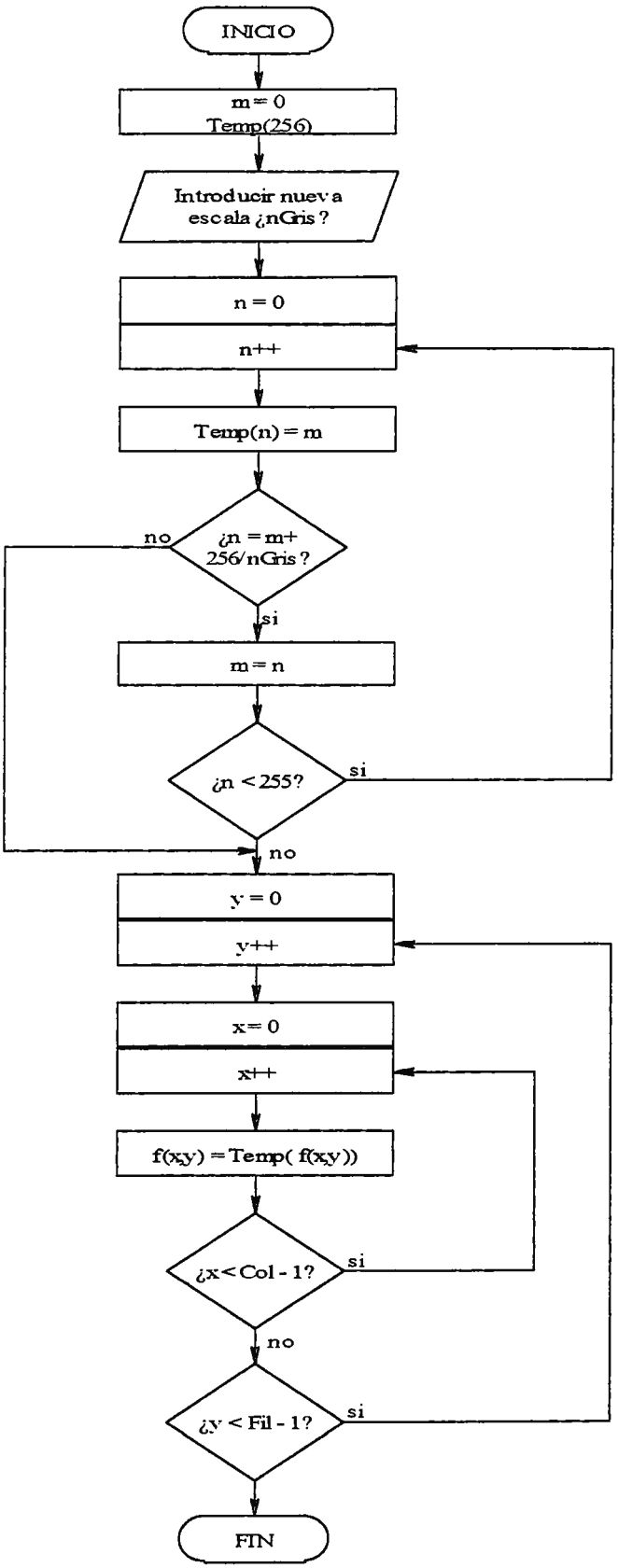


Figura 4.3.7. Cuadro de diálogo para aplicar una nueva escala de grises a una imagen.

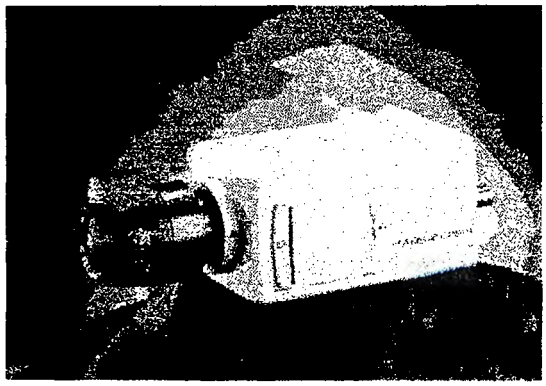
La posibilidad de restablecimiento obedece al hecho de que el usuario desee recuperar la imagen original aunque se haya transformado por pasos sucesivos a otra escala. Una razón de peso que justifica el hecho de haber incluido una escala menor o igual a 64 niveles, es que al experimentar con el programa de aplicación es posible darse cuenta que los resultados obtenidos por aquellas escalas que están contempladas en el rango de 256 a 64, no representan diferencia alguna para la apreciación humana.

En la figura 4.3.8(a) se puede observar el diagrama de flujo que se encarga de establecer los nuevos niveles de gris asociados a la imagen resultante, así como la que se presenta en la figura 4.3.8(b). En este algoritmo se llevan a cabo las siguientes tareas: En primer lugar, se subdivide la escala original en intervalos de acuerdo a la nueva escala. Como segundo paso, cada intervalo está caracterizado por un nivel de gris, por lo tanto se definen conjuntos de pixeles cuyos niveles se encuentran dentro de un intervalo, y a todos ellos les corresponderá un solo nivel de gris de salida. Este método representa una forma sencilla de simular la variación de los pasos de cuantización cuando se digitaliza una imagen con formato originalmente analógico.

De acuerdo con las variables que se manejan en el diagrama de flujo, se puede observar que se define un arreglo denominado *Temp* el cual contiene 256 elementos y que guarda los nuevos niveles de gris permitidos. Posteriormente se le asigna a cada $f(x,y)$ el nivel correspondiente antes de la transformación.


INICIO
m = 0
Temp(256)
Introducir nueva escala ¿nGris?
n = 0
n++
Temp(n) = m
no
¿n = m+ 256/nGris?
si
m = n
¿n < 255?
si
no
y = 0
y++
x = 0
x++
f(xy) = Temp(f(xy))
¿x < Col - 1?
si
no
¿y < Fil - 1?
si
FIN

a)



b)

Figura 4.3.8. Escala de grises. a) Diagrama de flujo. b) Imagen de la figura 4.3.1(b) transformada a una escala de 4 niveles de gris.

4.3.2.3 AGREGAR RUIDO

El algoritmo que se encarga de agregar ruido a una imagen, hace referencia al tipo de distribución y aleatoriedad de aparición de éste dentro de la matriz que compone la imagen. En la sección 2.3.6 se exponen los tipos de ruido más comunes que se encuentran a la hora de realizar la captura o recepción de una imagen y el histograma asociado a cada uno de ellos. Cuando se desee eliminar un tipo de ruido en particular, se debe utilizar para ello, el filtro que se adapte a las condiciones impuestas por la clase de ruido, es decir, es importante conocer un estimado de la distribución del mismo para luego decidir el filtro y los parámetros de éste para que se encargue de eliminarlo o minimizarlo.

La figura 4.3.9 muestra el cuadro de diálogo, en el que el usuario decide el porcentaje de ruido (válido para la distribución uniforme y salpimienta), el tipo, la desviación estándar y el promedio. Se puede comprobar en las figuras 2.3.20 y 2.3.21 de la sección 2.3.6 que los valores de desviación y promedio, son válidos para la distribución de Gauss y Exponencial negativo.

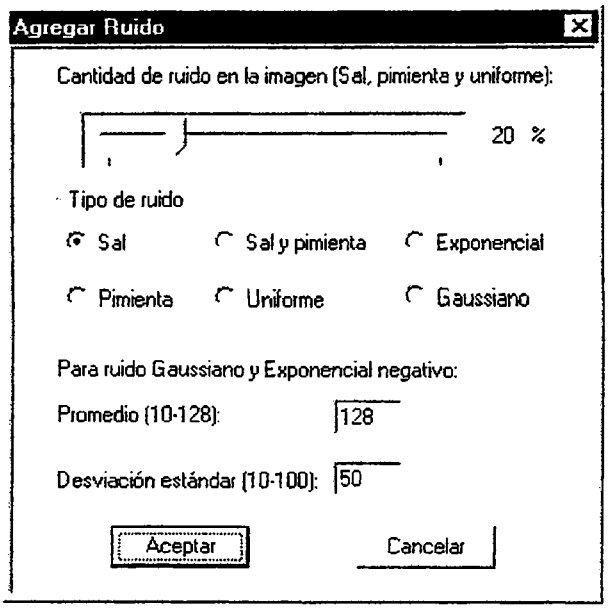


Figura 4.3.9. Cuadro de diálogo para definir los parámetros de ruido agregado a una imagen.

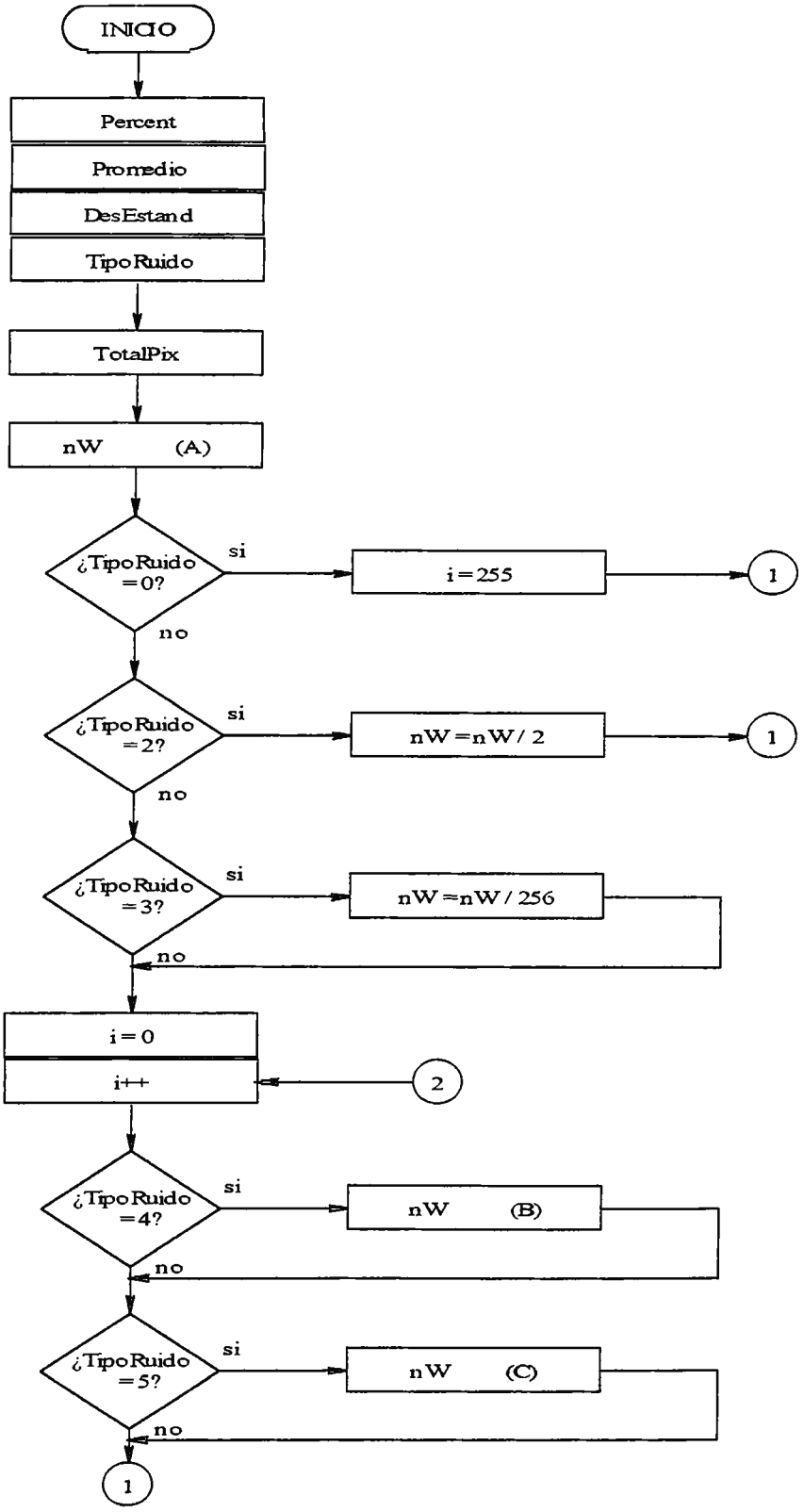
Para generar el ruido por medio de la programación, se ha utilizado una función aleatoria que retorna al azar una fila y una columna de la matriz de la imagen, posteriormente se procede a una substitución directa del valor actual $f(x,y)$, por el nivel que se interpretará como ruido. En todos los casos, la cantidad de pixeles afectados por el ruido está vinculada con los valores de M y N de la matriz de imagen.

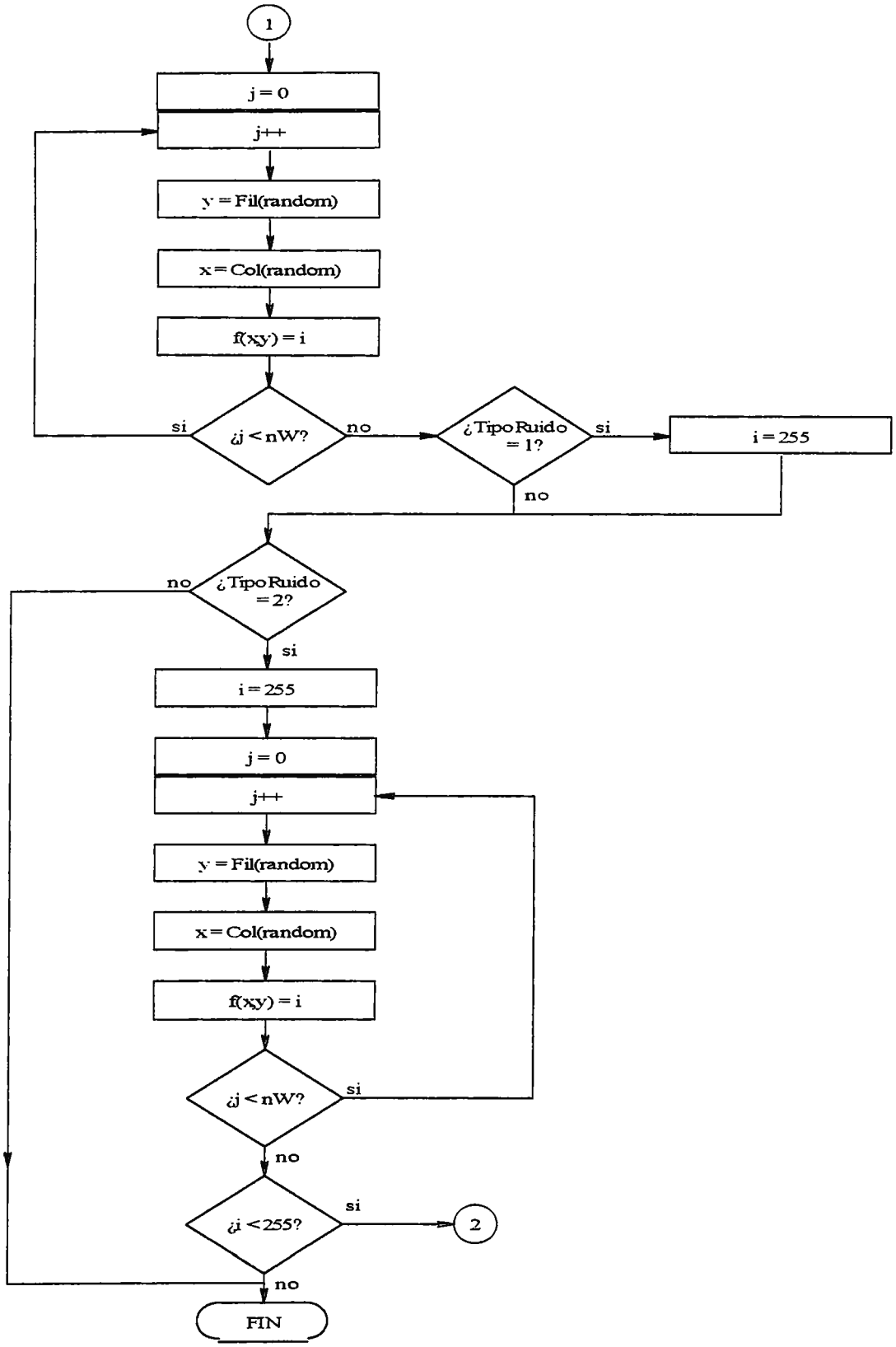
A continuación se presenta una tabla en la que se muestran los valores que toma la variable *TipoRuido* en el diagrama de flujo de la figura 4.3.10(a) y el tipo de ruido al que se refiere cada uno de ellos.

TipoRuido	Tipo de ruido
0	Sal
1	Pimienta
2	Sal y pimienta
3	Uniforme
4	Exponencial negativo
5	Gaussiano

Tabla 4.3.1. Tipos de ruido implementados en el algoritmo.

A continuación se presenta la figura 4.3.10(a) y (b) en las que se puede observar el diagrama de flujo para producir ruido en una imagen y una imagen con ruido.



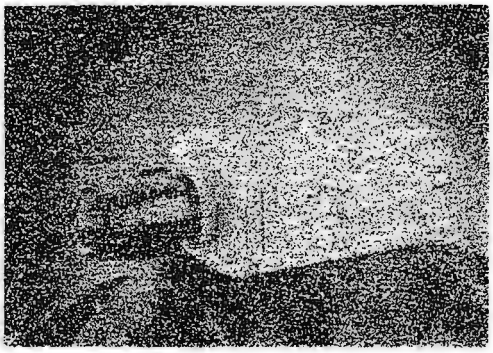


$$(A)\ nW = (Percent * TotalPix) / 100 \qquad (4.3\text{-}3)$$

$$(B)\ nW = ((e^{-i/DesEstand}) * TotalPix)/DesEstand \qquad (4.3\text{-}4)$$

$$(C)\ nW = ((e^{(i - Promedio)^2 / DesEstand^2}) * TotalPix)/\sqrt{2\pi} * DesEstand \qquad (4.3\text{-}5)$$

a)



b)

Figura 4.3.10. a) Diagrama de flujo para agregar ruido a una imagen. b) Imagen con ruido Gaussiano.

Una vez se haya elegido el tipo de ruido en el cuadro de diálogo y validado los parámetros, se realiza el llamado a la función que se encarga de adicionar ruido a la imagen. El ruido se agrega a la imagen a través de una sustitución directa de $f(x,y)$ antiguo por i para puntos selectos de la imagen. *TipoRuido* se encarga de guardar valores enteros que hacen referencia a los niveles de i permitidos y la cantidad de pixeles afectados (nW) por dicho nivel.

Si se desea verificar la cantidad de pixeles que son afectados por i, puede remitirse a la sección 2.3.6, con la particularidad que para la distribución uniforme se ha tomado toda la escala de grises o ver (A), (B), (C) de la figura 4.3.10 (a).

4.3.2.4 ESCALAR IMAGEN

El escalado de una imagen representa uno de los procedimientos que están comprendidos por la geometría de imagen. Este tipo de operación, se encarga de modificar las dimensiones de una imagen hasta un límite predefinido por el espacio de memoria

asignada por el programa. Algoritmos como este son utilizados para hacer más observables algunas características diminutas contenidas al interior de una imagen ya que es posible obtener una magnificación de la misma.

Al igual que la rotación, esta técnica hace uso de los métodos de interpolación que se encargan de asignar coordenadas a aquellos puntos de la nueva imagen que en la original no están definidos. Una ecuación sencilla puede ser implementada para producir los resultados anteriormente expuestos, por lo tanto se pueden definir las relaciones que se han implementado en este algoritmo:

a) Réplica de pixel

Sea r y q dos variables cuyos valores están comprendidos entre los intervalos $0 < r < 1$ y $0 < q < 1$, entonces se define la siguiente relación:

$$f(x+r, y+q) = f(entero[x+r+0.5], entero[y+q+0.5]) \tag{4.3-6}$$

De la ecuación (4.3-6) se puede concluir que si r o q son valores menores que 0.5, entonces el nivel que se le atribuye a $f(x+r,y+q)$ es igual a $f(x,y)$. Además, si r y q son valores mayores o iguales que 0.5, el nivel de gris asociado a ese pixel debe ser el nivel correspondiente a $f(x+1,y+1)$. La pérdida de la definición de la imagen resulta más evidente con la utilización de este método, por tratarse de una simple copia de nivel de gris.

2 - Vecino más cercano

Este tipo de interpolador también es conocido como interpolador lineal o interpolación de orden cero. Se define entonces la siguiente relación:

$$\begin{aligned} f(x+r, y+q) = (1-r).(1-q).f(x,y) + r.(1-q).f(x+1,y) + \\ q.(1-r).f(x,y+1) + r.q.f(x+1,y+1) \end{aligned} \tag{4.3-7}$$

La principal desventaja que presenta este interpolador, es el hecho de asumir que la variación de los niveles de gris entre los cuatro pixeles vecinos al que se evalúa, puede ser modelado como un plano lineal.

c) Promedio

El último de los métodos en el que se apoya la técnica de escalado es el promedio que se encuentra entre los cuatro pixeles vecinos al punto que se está evaluando, así se puede definir la siguiente relación:

$$f(x+r, y+q) = \frac{f(x,y) + f(x+1,y) + f(x,y+1) + f(x+1,y+1)}{4} \qquad (4.3\text{-}8)$$

Este método representa una de las formas más sencillas de llevar a cabo la interpolación, en virtud de los cálculos necesarios y los resultados obtenidos [26]. Para que el usuario pueda decidir entre las tres posibilidades arriba expuestas, se ha diseñado un cuadro de diálogo que se origina cuando se vuelve activa la opción de "Escalar imagen". En la siguiente figura se observan las tres posibilidades para modificar las dimensiones de la imagen. El valor de cada uno de los factores que se encargan de afectar dichas dimensiones, se debe encontrar entre 0.1 y 5, ya que así ha sido definido en el programa. Cada uno de esos factores son totalmente independientes, es decir, el redimensionamiento se puede realizar en la dirección horizontal, en la vertical o bien en ambas.

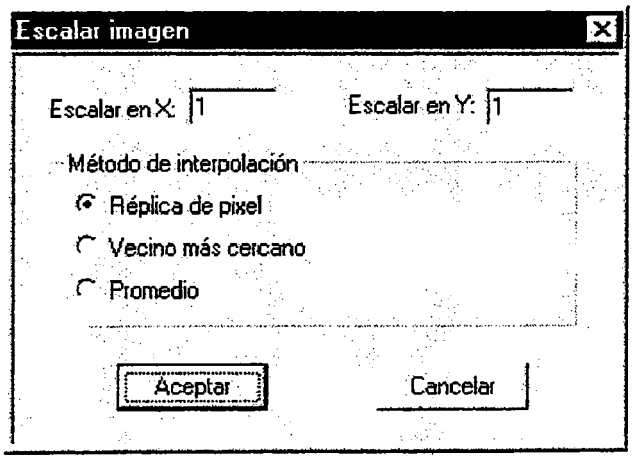
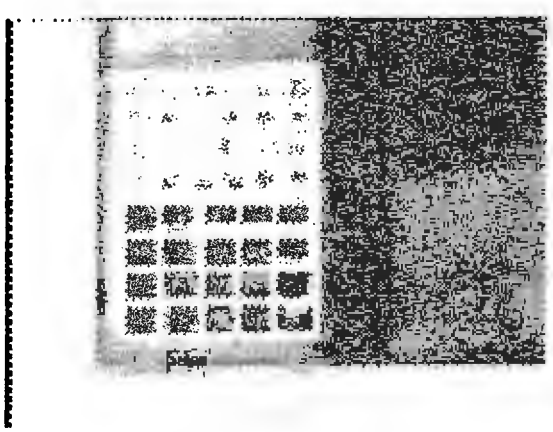


Figura 4.3.11. Cuadro de diálogo para el escalado de imagen.

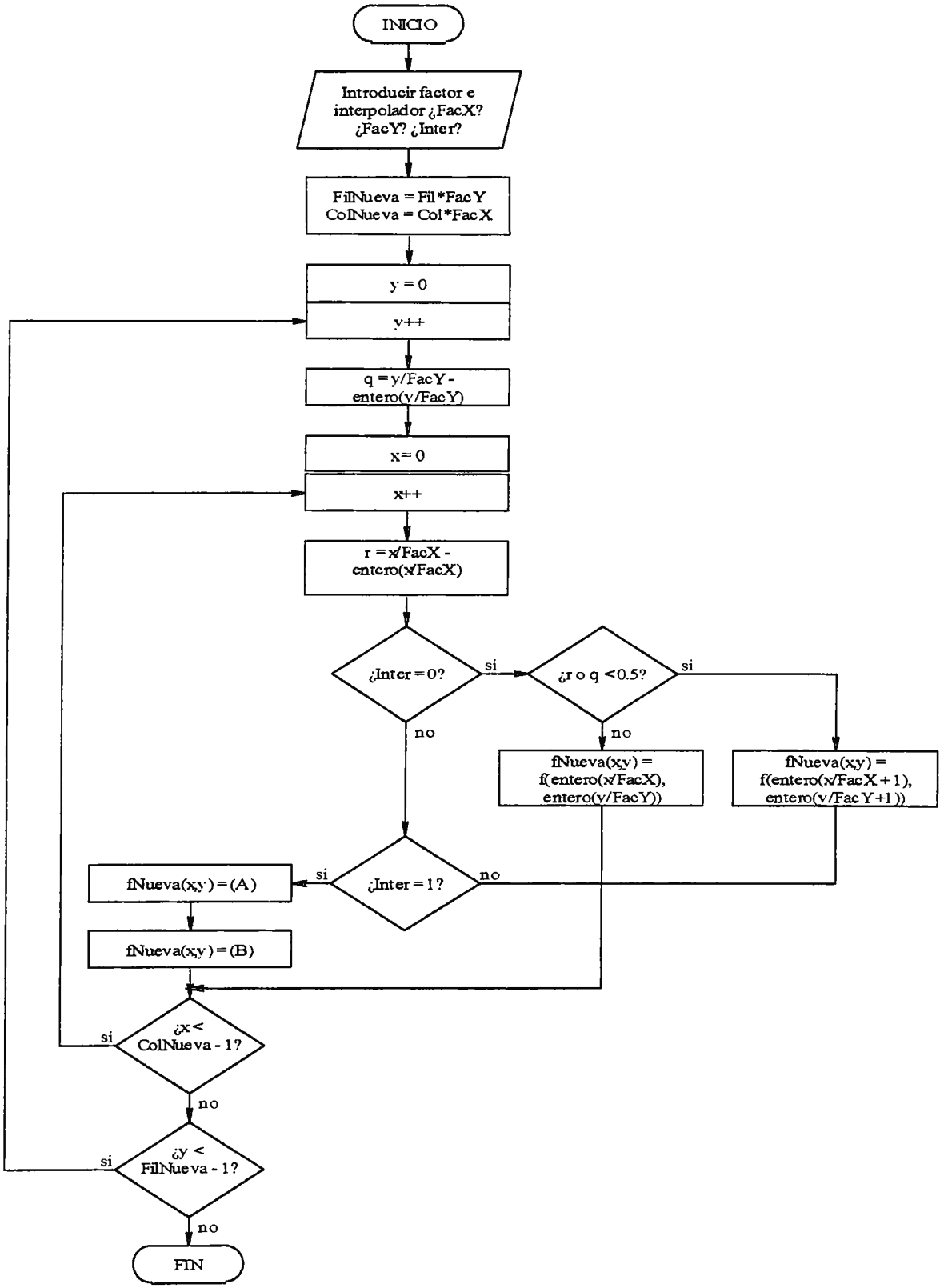
En la figura 4.3.12(a) se tiene el resultado de aplicar el algoritmo presentado en el diagrama de flujo de la figura 4.3.12(b) con el método de interpolación denominado réplica de pixel. Mas que el tamaño, es de hacer notar la pérdida en la definición de la imagen ya que primero se disminuyó su tamaño a un 10% y luego este resultado se multiplicó por un factor de 3 en ambas direcciones.

En cuanto a la definición de la imagen, los mejores resultados se obtienen con la aplicación de la ecuación (4.3-7), ya que considera un cálculo exhaustivo para determinar el nivel de gris que se le atribuirá al punto mapeado en la nueva imagen, pero como en la gran mayoría de los casos, para llegar a ese resultado se tiene que recurrir a una mayor capacidad de procesamiento.

En primera medida, el algoritmo se encarga de calcular las dimensiones de la nueva imagen de acuerdo a los factores de multiplicación que el mismo usuario introduce. La siguiente acción corresponde a la exploración de pixel a pixel de la nueva imagen para determinar los valores de r y q respectivamente y de esa manera poder determinar por comparaciones sucesivas el nivel de gris asociado a cada uno de los pixeles en la nueva imagen. Sucesivamente al paso anterior, se lleva la selección del método de interpolación elegido por el usuario y es ésta la porción del programa la que se encarga de aplicar una de las ecuaciones definidas en (4.3-6) a (4.3-8).



a)



b)

Figura 4.3.12. Escalado. a) Imagen de la figura 4.3.1(d) escalada. b) Diagrama de flujo.

En el diagrama de flujo de la figura 4.3.12(b) se utiliza una función denotada por la siguiente nomenclatura *entero(x/FacX)* además, una función similar de la forma *entero(y/FacY)*, ambas se encargan de determinar y retornar el valor entero del argumento que se encuentra entre paréntesis. Otro aspecto que es necesario aclarar, es que en los procesos donde aparece la simbología (A) y (B) se refiere a las ecuaciones (4.3-7) y (4.3-8) respectivamente. La variable *Inter* representa los tipos de interpoladores utilizados en el algoritmo, según la tabla siguiente:

Inter	Tipo de interpolación
0	Réplica de pixel
1	Vecino mas cercano
2	Promedio

Tabla 4.3.2. Tipos de interpoladores implementados en el algoritmo.

4.3.2.5 ROTAR

La opción de rotar una imagen incluye la rotación de un ángulo (0-90 grados) y la rotación derecha e izquierda. La combinación de estos tres tipos de rotación, puede dar como resultado ángulos de rotación mayores a los 90 grados ya sea en el sentido horario o antihorario.

Rotar un Ángulo θ

Este proceso consiste en rotar la imagen θ grados alrededor de un punto seguido por un desplazamiento, de tal manera que ésta se encuentre completamente enmarcada dentro de un nuevo rectángulo con un fondo de nivel de gris constante. Todo esto es con el propósito

de evitar que la imagen sea recortada en alguna de sus esquinas durante el proceso de rotación y de esta manera preservar todas las características de la misma.

En primera instancia deben calcularse las nuevas dimensiones de la imagen a partir de un ángulo de rotación θ. Las funciones usadas para calcular el nuevo alto y ancho de la imagen son:

$$FilNueva = Fil(\cos\theta) + Col(\text{sen}\theta) \tag{4.3-9}$$

$$ColNueva = Fil(\text{sen}\theta) + Col(\cos\theta) \tag{4.3-10}$$

donde *Fil* es número de filas y *Col* el número de columnas que representan las dimensiones de la imagen original en pixeles. Después de determinar las nuevas dimensiones, se crea una imagen y se rellena en su totalidad con un solo nivel de gris.

Para determinar la ubicación de cada punto de la nueva imagen, se hace uso de la matriz de transformación siguiente:

$$\begin{bmatrix} x_{nuevo} \\ y_{nuevo} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\text{sen}\,\theta \\ \text{sen}\,\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{antiguo} \\ y_{antiguo} \end{bmatrix} \tag{4.3-11}$$

Con la nueva ubicación se hace un mapeo de pixeles desde la imagen original a la nueva imagen, y para el final del rastreo de toda la imagen se tendrá la imagen completamente rotada.

Al seleccionar la opción de rotar un ángulo en el menú "Edición" aparece un cuadro de diálogo como el de la figura 4.3.13. Como se puede verificar, éste permite introducir el nivel de gris del fondo que circunda la nueva imagen y el ángulo a rotar en grados, el cual se mide a partir del eje horizontal, tomando siempre como centro de rotación la esquina superior izquierda de la imagen.

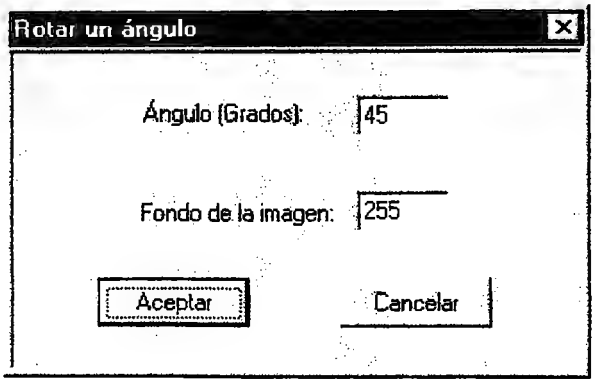
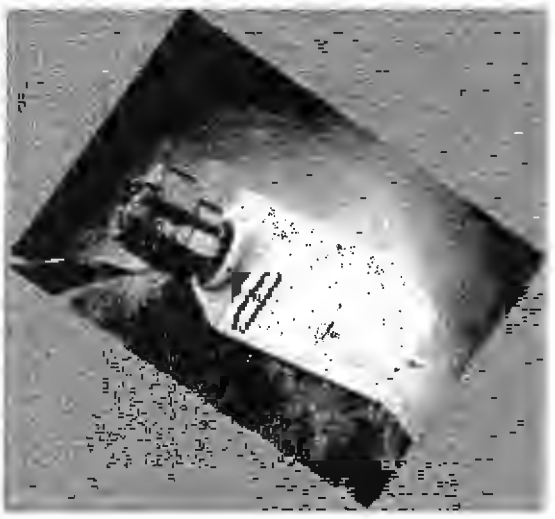


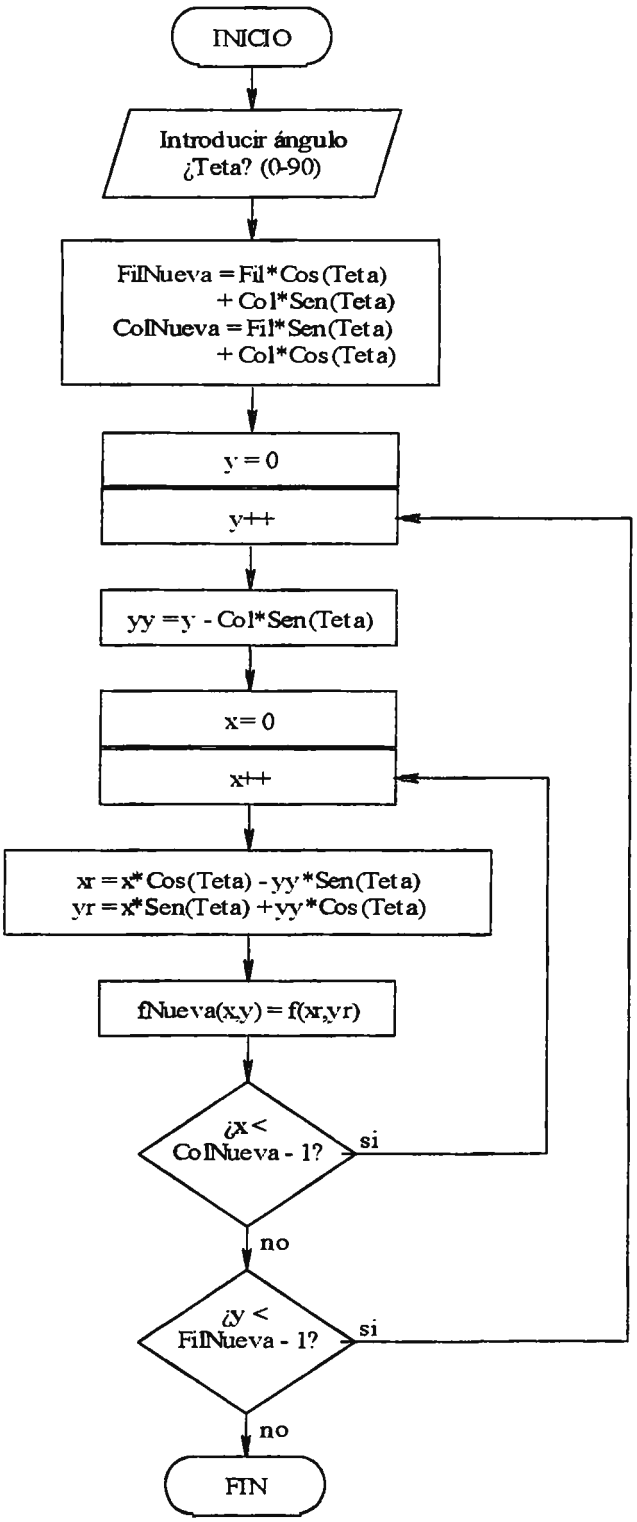
Figura 4.3.13. Cuadro de diálogo para definir el ángulo y el fondo para una imagen rotada.

Para mayor información sobre este efecto, consultar referencia [17]. En la figura 4.3.14(a) se muestra la imagen de la figura 4.3.1(b) rotada 35 grados y el fondo con un nivel de gris igual a 100.

En la figura 4.3.14(b) se muestra el algoritmo relacionado con este procedimiento. En éste se ha usado el proceso inverso de exploración y mapeo de datos, es decir, se explora la imagen rotada y se mapea la original, con lo que se elimina el efecto de textura producido al usar el procedimiento descrito en la página anterior.



a)



b)

Figura 4.3.14. Rotación de imagen. a) Resultado de aplicar b) a la imagen de la figura 4.3.1(b). b) Diagrama de flujo para rotar a una imagen un ángulo.

Rotación Derecha e Izquierda

El procedimiento consiste en un intercambio de filas y columnas seguido por un desplazamiento, de tal manera que el origen de la nueva imagen rotada coincida con el origen de la imagen original. Al activar cualquiera de las opciones, se inicia un procedimiento que ejecuta el algoritmo asociado, mostrado en los diagramas de flujo que se ubican en la figura 4.3.15.

Es necesario contar con una replica de la imagen original para llevar a cabo la rotación, ya que la resultante difiere en ancho y alto en comparación con la primera. Esto indica que algunos parámetros, como por ejemplo, el tamaño de la imagen justificada de la estructura BMP debe modificarse.

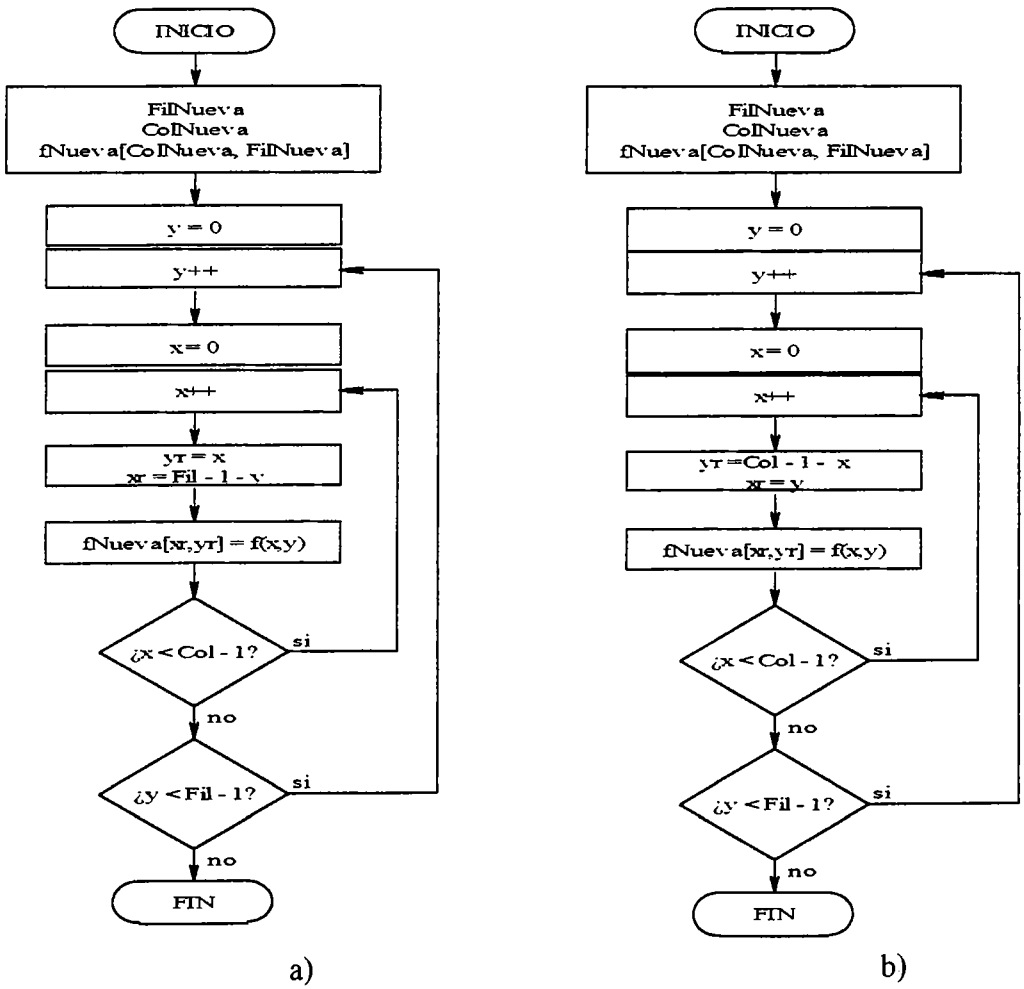


Figura 4.3.15. Diagrama de flujo de la rotación a 90°. a) Derecha y b) Izquierda.

Para observar los resultados obtenidos al aplicar estos procedimientos, se presenta la figura 4.3.16, la que muestra dos imágenes, la primera rotada hacia la derecha y la segunda rotada hacia la izquierda.

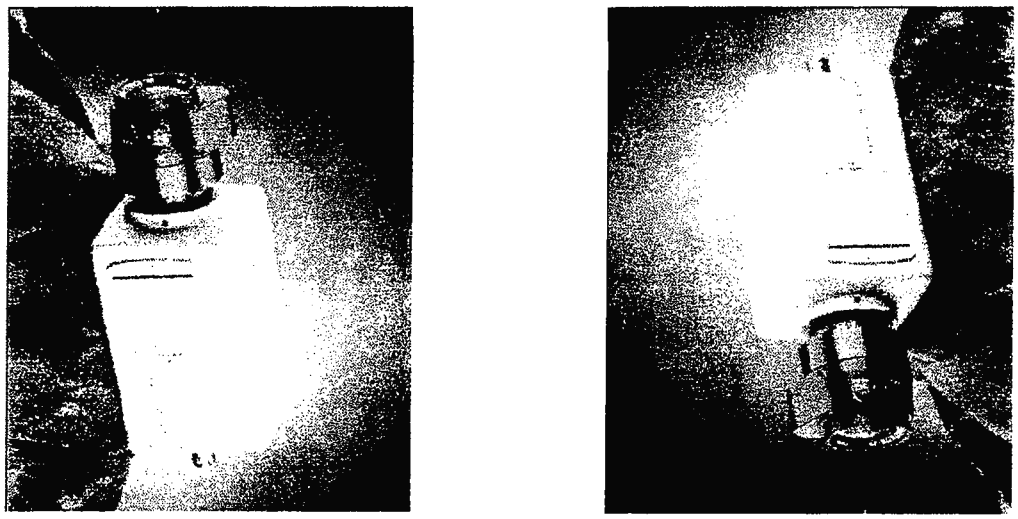


Figura 4.3.16. Resultado de aplicar la técnica de rotación. a) Derecha y b) Izquierda.

4.3.2.6 REFLEJAR

Los dos tipos de reflexión discutidos son el vertical y el horizontal. Al igual que el proceso de rotación, la reflexión está relacionada con la geometría de la imagen, es decir, que no trabaja con el valor de los pixeles sino más bien con su ubicación en la matriz de datos, con el fin de cambiar su forma y posición. El proceso de reflexión consiste en el intercambio de pixeles entre columnas (reflexión horizontal) o bien entre filas (reflexión vertical) simétricamente opuestas respecto a la columna o fila central, desde la fila o ya sea la columna cero hasta la (fila o columna)/2 - 1.

Al activar esta función se procede a ejecutar el diagrama de flujo mostrado, dependiendo si es vertical u horizontal según el caso.

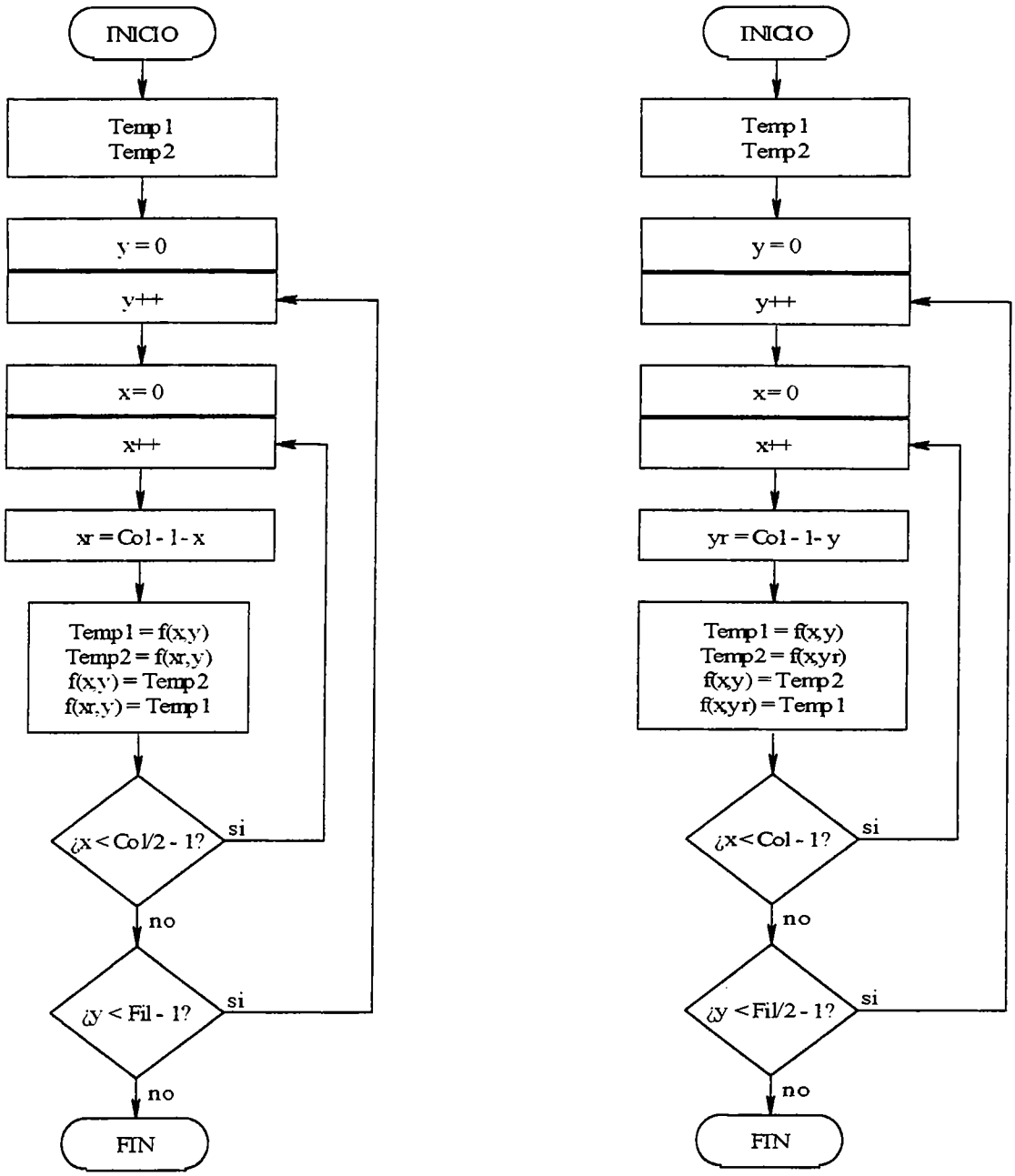


Figura 4.3.17. Diagrama de flujo de la reflexión. a) Horizontal y b) Vertical.

La figura 4.3.18 muestra los resultados del uso de la técnica sobre la imagen de la figura 4.3.1(b). A diferencia de la rotación las dimensiones de la imagen se mantienen, y no es necesario crear una nueva imagen como se hizo en la rotación.

a) b)

Figura 4.3.18. Resultado de la aplicación del algoritmo de reflexión. a) Horizontal, b) Vertical.

4.3.3 MENÚ OPCIONES

Es un elemento del marco de la aplicación que en general, está constituido por opciones que se orientan hacia la obtención de nuevas vistas, como por ejemplo, la llamada hacia la creación del histograma asociado a una imagen, para realizar trabajos con la tarjeta digitalizadora, para transformar una imagen en escala de grises en una imagen con componentes en color, etc.

Las opciones que se encuentran ubicadas dentro de este menú, son opciones que permiten la creación de una nueva imagen mediante la captura producida por la tarjeta o en todos los casos la preparación de una imagen activa en el banco de trabajo. Para que posteriormente se le aplique un proceso en particular, como por ejemplo, la detección de bordes de una imagen binarizada, morfología y también para la preparación de una imagen que va a ser sometida a las operaciones con otras imágenes.

Aquí se tiene una imagen del marco de la aplicación, cuando el menú "Opciones" es activado por el usuario.

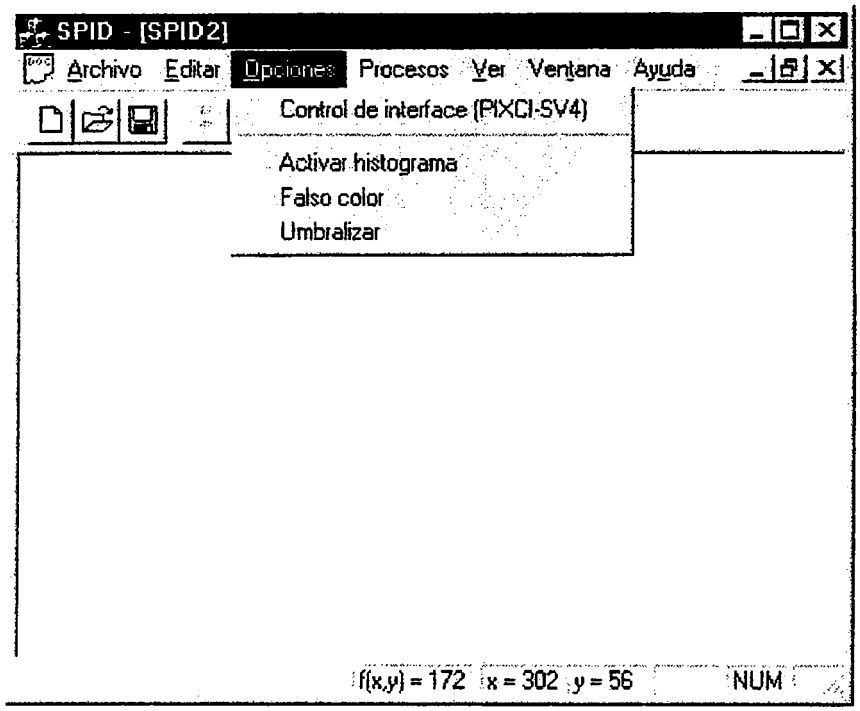


Figura 4.3.19. Despliegue del menú opciones del marco de la aplicación.

4.3.3.1 CONTROL DE INTERFACE

Esta es una opción que permite capturar una imagen, digitalizarla y prepararla para que posteriormente el usuario trabaje con ella haciendo uso del resto de los elementos del programa de aplicación. Entre estas tareas se puede mencionar mejoramiento de la distribución de los niveles de gris, resaltar ciertas características o bien aplicar algún proceso en particular como parte de la experimentación.

Cuando el usuario hace uso de esta capacidad del sistema, se realiza el llamado a una función que se encarga de generar una nueva ventana de trabajo la cual puede ser observada en la figura 4.3.20. Ésta última da servicio a las funciones de la tarjeta interface, es decir, sobre ella se dibujará una secuencia de imágenes o también le permitirá configurar algunos parámetros en la misma.

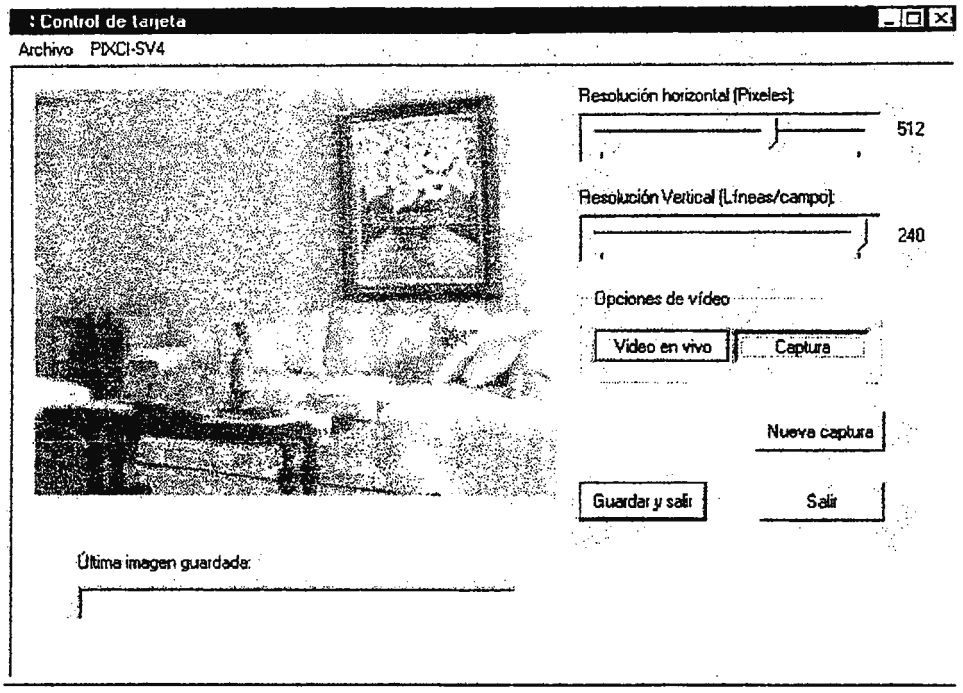


Figura 4.3.20. Ventana de servicio de la tarjeta interface.

Puede observarse que en la figura anterior, se cuenta con algunos de los controles más comunes de Windows. En primer lugar se puede mencionar el menú "Guardar como", en el que se tiene la posibilidad de guardar una imagen y poder continuar con otras tareas. Los tres formatos de archivo con los que se cuenta son: TIFF (del inglés Tagged Image File Format), PCX y BMP. Ahora bien, en el menú "PIXCI-SV4", se pueden encontrar las opciones para configurar los parámetros básicos en la tarjeta o para cerrar la sesión de trabajo.

Los controles de desplazamiento se encargan de modificar los parámetros de resolución tanto vertical como horizontal, dicho sea de paso, el rectángulo donde aparece cada una de las imágenes posee dimensiones constantes, por lo tanto una función de la librería de la tarjeta se encarga de adecuar las dimensiones de la imagen a las dimensiones del rectángulo. Dos controles del tipo botón operan alternadamente para habilitar la posibilidad de vídeo en vivo o para que el usuario realice una secuencia de capturas cada vez que él lo desee. Por otra parte, el botón de "Guardar y salir" se encarga de presentar el

cuadro de diálogo de guardar como y la imagen recién guardada se abre como un documento en el marco principal de la aplicación. "Salir", simplemente cierra la ventana sin la posibilidad de guardar con anterioridad la imagen. Como punto final existe una caja de edición de sólo lectura que se encuentra ubicada en la parte inferior de la ventana y en la que aparece la ruta en la que la última imagen ha sido guardada.

Una sesión de trabajo con la cámara y la tarjeta debe seguir los siguientes pasos: primero, seleccionar del menú "PIXCI-SV4" la opción de "Abrir tarjeta", posteriormente presionar uno de los botones para habilitar vídeo en vivo o capturas individuales. Una vez terminada la sesión se debe ejecutar "Cerrar tarjeta" del mismo menú.

4.3.3.2 ACTIVAR HISTOGRAMA

El histograma es una de las principales fuentes de información, respecto a la distribución de los distintos niveles de gris que componen la imagen. Dado que es una representación gráfica de la imagen, el histograma es muy fácil de interpretar y constituye la herramienta principal en la segmentación y reconocimiento de objetos. Muchos de las técnicas antes mencionadas usan la distribución dada por el histograma para determinar las distintas regiones u objetos dentro de una imagen, así también ayudan a determinar los niveles de umbral óptimos y permiten caracterizar una imagen desde el punto de vista del brillo y el contraste.

El algoritmo del histograma hace uso de la definición, que establece que el valor de histograma h_i es el porcentaje de pixeles a un determinado nivel de gris. Cada valor puede estar representado gráficamente por una barra o una línea vertical con altura h_i. Y la secuencia de barras a lo largo del eje horizontal con respecto al nivel i dentro de la imagen, genera el gráfico de histograma. En el programa, esta opción está activa en el menú "Opciones" bajo el nombre de "Activar histograma". Al seleccionar esta opción aparece una

nueva ventana hija en donde se construirá el gráfico, junto con los valores para cada nivel de gris desde 0 hasta 255.

Una vez obtenidos los valores del histograma para cada nivel de gris y almacenados en un arreglo, denominado como *Histo* dentro del algoritmo, el programa procede a crear el gráfico dentro la nueva ventana hija. El resultado de esta opción es como se muestra en la figura 4.3.21, donde se observa la imagen y su respectivo gráfico de histograma.

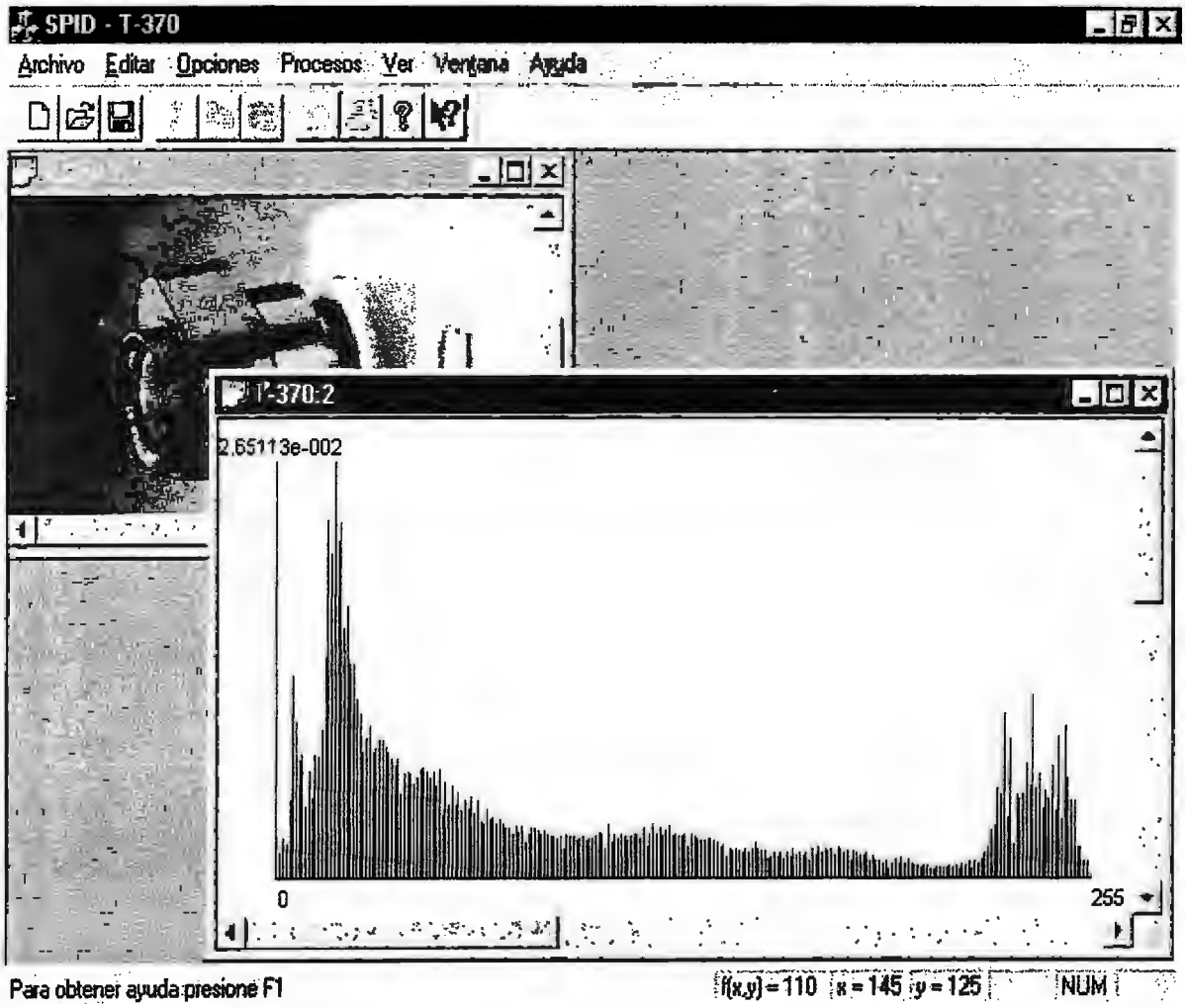


Figura 4.3.21. Gráfico de histograma.

A continuación se puede observar una representación simplificada en la que se indica el proceso desarrollado para obtener el histograma de una imagen cualquiera.

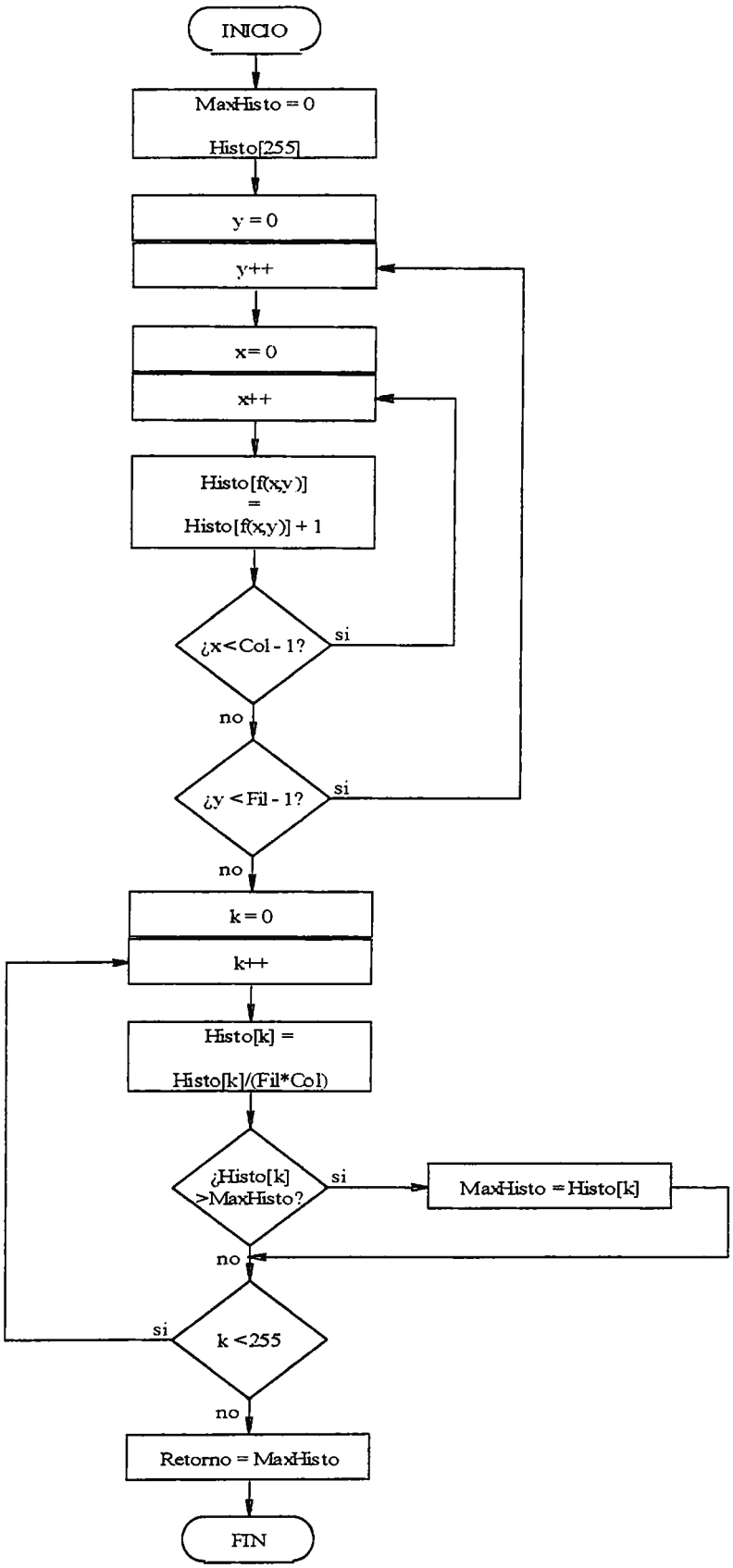


Figura 4.3.22. Diagrama de flujo para obtener el histograma de una imagen.

4.3.3.3 FALSO COLOR

El método usado dentro del programa para llevar a cabo el falso color difiere del expresado en la teoría de la sección 2.3.4, en el que se propone convertir una imagen en escala de gris de 8 bit por cada pixel en una imagen a color de 24 bit por pixel. El proceso que se describe a continuación es una forma fácil y rápida de aplicar falso color a una imagen en escala de gris sin tener que transformarla al formato RVA. El proceso consiste en la modificación de la tabla de color del mapa de bits, con el objeto de alterar la paleta lógica que se encarga de asignar los colores de la imagen en relación con el dispositivo de presentación, el cual puede ser la pantalla o bien la impresora.

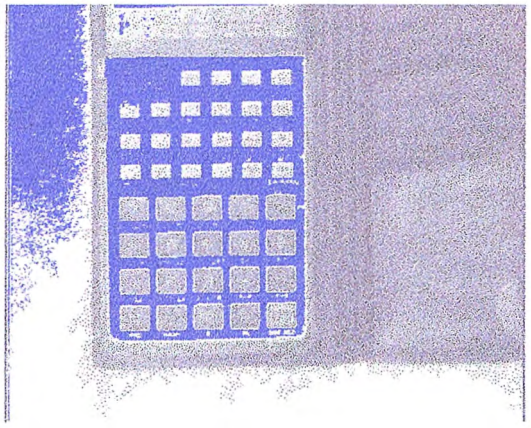
En la tabla de color de una imagen BMP existe una localidad de memoria de 32 bits para cada color. Por lo tanto existen 256 entradas de 32 bits para una imagen de 8 bits por pixel. En una localidad de memoria de 32 bits, 3 bytes son asignados para las componentes de color rojo, verde y azul, el byte restante es reservado como bandera. Para la creación de un color en particular debe hacerse uso de la proporción adecuada de las componentes de color. Así por ejemplo, para el nivel de gris 0 (negro) cada componente debe ser cero, para el nivel de gris 1 las componentes deben ser uno y así sucesivamente, hasta llegar al nivel de gris 255 (blanco) en donde las tres componentes son iguales a 255.

Si se desea cambiar un nivel de gris por un color en particular como por ejemplo el rojo, debe sustituirse esa entrada de color por la combinación: componente rojo igual a 255, verde y azul igual a 0. De esta manera se puede asignar un color determinado a un solo nivel o rango de niveles de gris. Una vez se ha modificado la tabla de color del mapa de bits, debe crearse una paleta lógica por medio de las instrucciones destinadas para ello, como por ejemplo la instrucción Cpalette::CreatePalette de la librería de clases de la MFC para Win32 [9]. Con la paleta lógica creada, el sistema operativo de Windows 95 crea una paleta fisica en relación con el tipo de tarjeta de vídeo, que puede ser de 8, 24 o 32 bits. Y con esto, se está listo para la presentación de las imágenes en pantalla.

La ventaja de este método es que la imagen permanece como una imagen de 8 bits por pixel, lo que permite seguir usando todas las técnicas de procesamiento destinadas a imágenes en escala de gris.

En la figura 4.3.23(a) se muestra un cuadro de diálogo que aparece cuando se selecciona falso color y en la figura 4.3.23(b) el resultado de aplicar falso color a la imagen de la figura 4.3.1(d) para un rango de niveles de 250 a 255.

a)



b)

Figura 4.3.23. Falso color. a) Cuadro de diálogo. b) Imagen con falso color.

4.3.3.4 UMBRALIZAR

Este algoritmo es muy importante ya que ofrece una alternativa para volver binaria una imagen que se constituye por *n* niveles de gris; después de ese proceso el usuario tiene la posibilidad de trabajar con la morfología de imagen entre otros procesos.

Cuando el usuario vuelve activa esta opción desde el menú principal, aparecerá el siguiente cuadro de diálogo si con anterioridad se ha abierto una imagen.

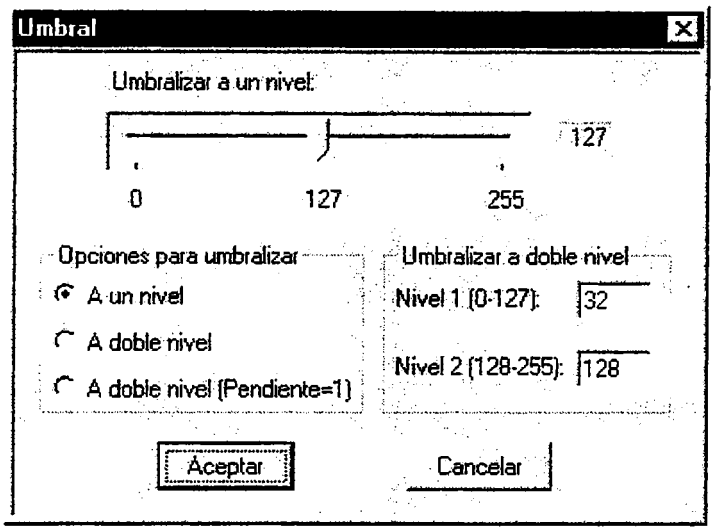


Figura 4.3.24. Cuadro de diálogo que permite elegir el nivel de umbralización "T".

Tal como se indica en la figura 4.3.24, es posible umbralizar utilizando distintas funciones de transferencias, así en un solo nivel se establece una frontera de umbralización que es definida por el interesado. Para umbralización a doble nivel se establecen dos fronteras de discriminación, teniéndose como resultado una imagen binaria. Por último, la opción doble nivel con pendiente igual a uno, permite que aquellos niveles de gris que se encuentran dentro del rango sean modificados a una constante de 255 y el resto de los elementos de la imagen mantienen sus valores originales, por lo tanto la imagen resultante no se considera binaria. A partir del instante en el que el usuario presiona el botón aceptar, se efectúa la llamada al algoritmo asociado que se resume y ejemplifica en el diagrama de flujo de la figura 4.3.25.

En este diagrama de flujo los valores que toma la variable *TipoUmb* pueden ser: cero para la umbralización a un nivel, uno para la umbralización a doble nivel y dos para doble nivel con pendiente igual a uno.

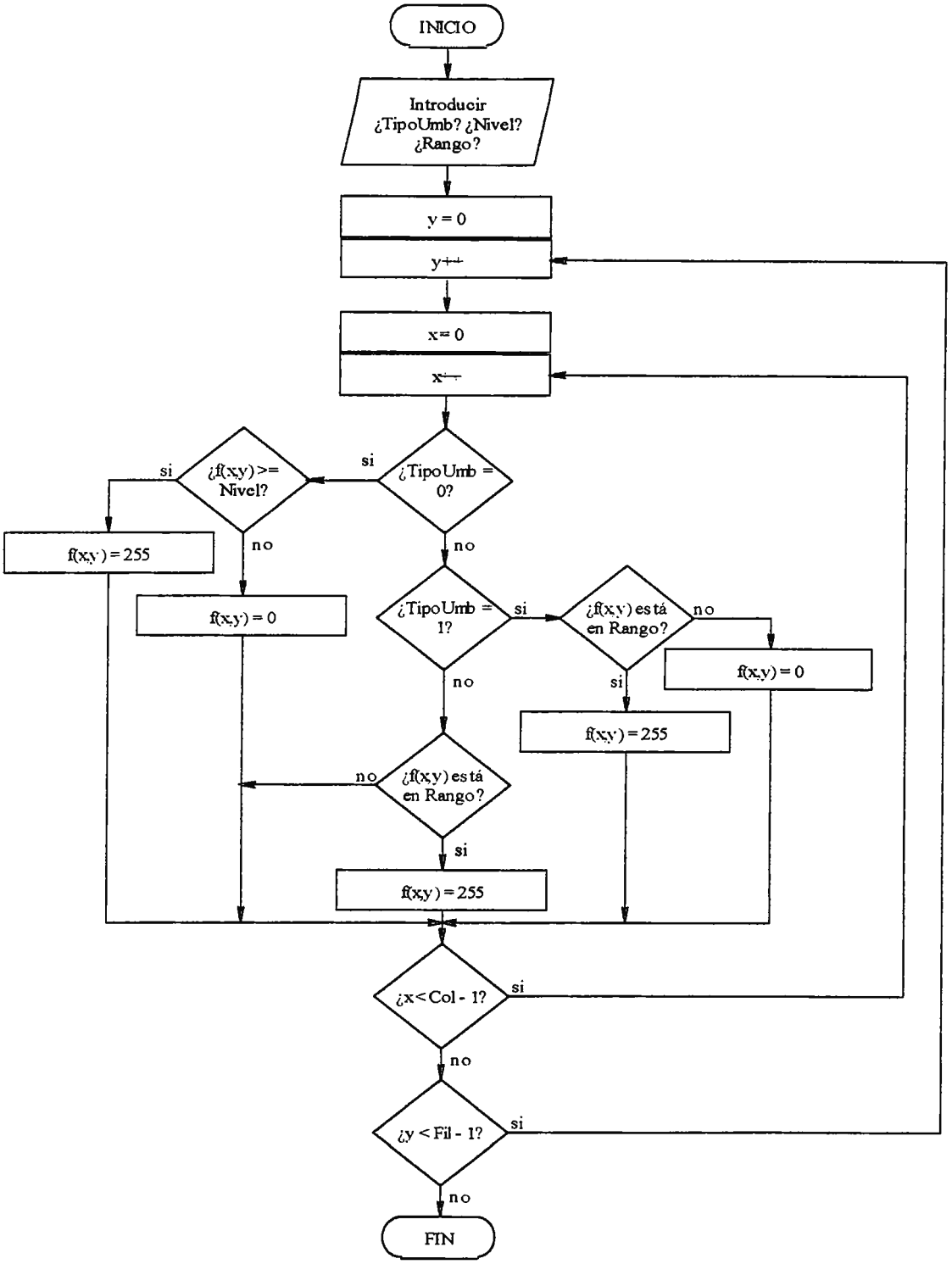


Figura 4.3.25. Diagrama de flujo para obtener la umbralización.

Para hacer descriptiva esta opción, se tomará la imagen de la figura 4.3.1(a), la que se obtuvo a partir del equipo adquirido para la captación y digitalización, además, en ella se puede observar el resultado de la técnica de iluminación en contraluz. Con esta figura se demuestra un aspecto de suma importancia en procesamiento de imágenes, ya que hasta ese momento no se tiene una imagen binaria en su totalidad así como se desearía para ciertos procesos relacionados a la técnica de iluminación antes mencionada. Para superar este inconveniente se recurre a la umbralización entre otras técnicas. Tal como se presenta en la figura 4.3.24, se ha elegido un valor de umbral de 127, si se aplica el algoritmo a la imagen de la figura 4.3.1(a) se obtendrá el siguiente resultado.

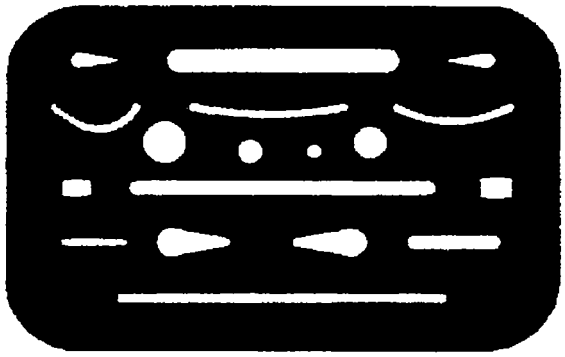


Figura 4.3.26. Imagen de la figura 4.3.1(a) umbralizada a un nivel "T" de 127.

4.3.4 MENÚ PROCESOS

Se le ha llamado así, ya que en él se encuentran agrupadas las opciones para el procesamiento de imágenes propiamente dicho. Los algoritmos disponibles han sido diseñados con el objeto de trabajar directamente con una imagen, y que posteriormente se tenga la posibilidad de realizar un análisis en particular, considerando que los algoritmos se encargan de realizar el filtrado, efectuar transformaciones morfológicas, etc.

La mayoría de los elementos de este menú, son considerados los más importantes debido a que, por medio de ellos es posible interpretar los resultados de las técnicas

aplicadas a una imagen, lo cual significa que es posible verificar los parámetros involucrados en un procesamiento y de esa manera encontrar el mejor de los resultados por medio de la experimentación. En la siguiente figura se presenta una imagen de las opciones contenidas en el menú "Procesos" cuando el usuario lo despliega.

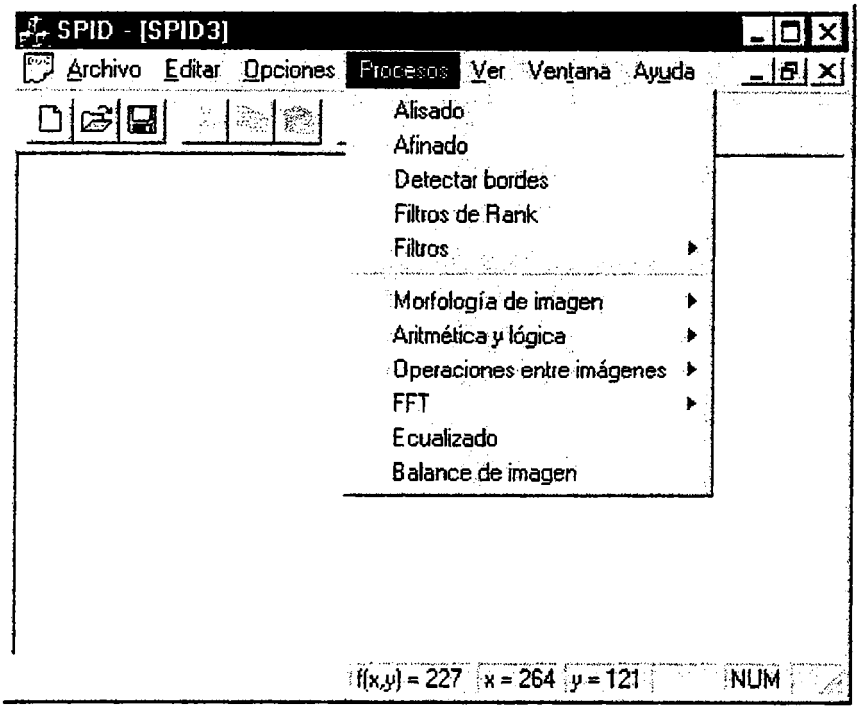


Figura 4.3.27. Despliegue del menú procesos del marco de la aplicación.

4.3.4.1 FILTRADO ESPACIAL

Es una técnica muy importante en el procesamiento de imágenes, ya que es el medio que permite llevar a cabo las tareas de filtrado de ruido en imágenes. Hay filtros espaciales lineales y no lineales. Los lineales hacen uso de las máscaras y la convolución espacial para llevar a cabo el filtrado. Por otro lado los no lineales se basan en las características propias de la vecindad de pixeles de una imagen, como por ejemplo el orden estadístico de cada pixel en la vecindad que se evalúa en cada punto.

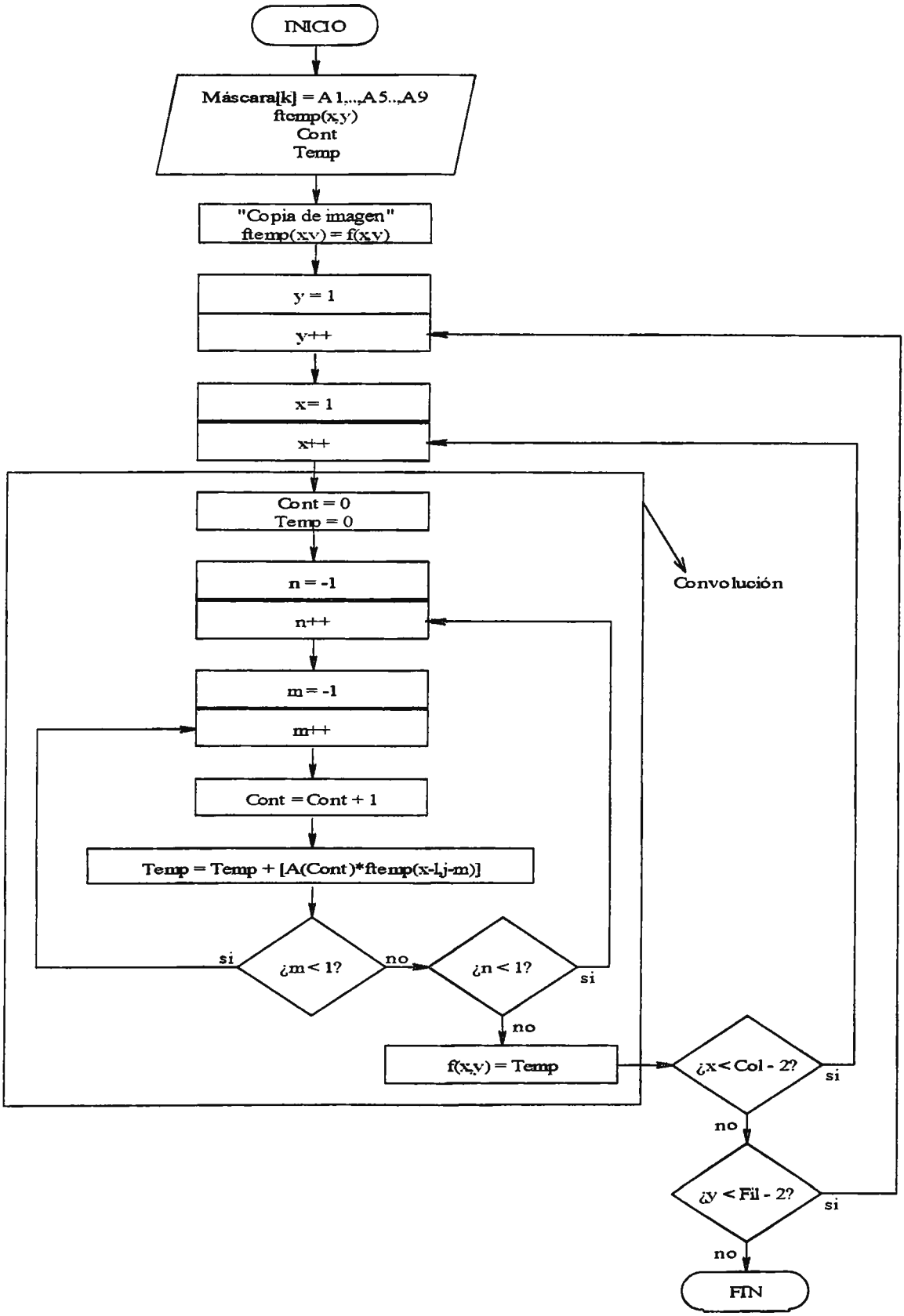


Figura 4.3.28. Diagrama de flujo para obtener el filtrado espacial.

La mayoría de los programas de procesamiento de imágenes incluyen opciones para el filtrado lineal de pasa-bajos y pasa-altos con un tipo de máscara específica para cada caso y con dimensión fija o variable. En nuestro caso se incluye un filtro de promedio como filtro pasa-bajos (alisado) y un filtro pasa-altos (afinado).

El algoritmo presentado para el filtrado lineal, considera que la convolución efectuada no altera los pixeles de la primera y última fila/columna de la imagen, con el objeto de simplificar el proceso de filtrado. Además, se usa un tamaño de máscara de filtro de 3x3 pixeles, pudiendo ampliarse mediante un pequeño ajuste en las variables internas en el lazo de convolución. El diagrama de flujo simplificado se presenta en la figura 4.3.28. Es de notar la creación de una copia temporal de la imagen con el objeto de preservar los datos del original mientras se obtiene la nueva imagen ya filtrada.

Alisado

Para el filtrado pasa-bajos se ha utilizado la respuesta impulso paramétrico de 3x3 representado por (2.3-68) en la sección 2.3. El filtrado pasa-bajos se encuentra bajo el nombre de "Alisado" en el menú "Procesos". Al seleccionar esta opción, el programa le pide al usuario que defina el valor del parámetro entre 0 y 10 (con el valor de 1 por omisión), el cual indica el grado de tolerancia al dejar pasar frecuencias altas o discontinuidades en la imagen. Así por ejemplo, el valor de 0 hace que el filtro sea más eficiente en el rechazo de altas frecuencias, en cambio el valor de 10 es un filtro menos eficiente al detectar y eliminar frecuencias altas. En la figura 4.3.29 se muestran dos imágenes filtradas con distinto valor del parámetro b.

a) b)

Figura 4.3.29. Filtrado espacial pasa-bajos con parámetro a) b=0 y b) b=5.

Afinado

Este puede ser considerado como un filtro pasa-altos, ya que permite acentuar los detalles más finos dentro de una imagen. A diferencia del pasa-bajos este tipo de filtro no es conveniente para remover el ruido de una imagen. El proceso para su ejecución es similar al pasa-bajos, discutido anteriormente y su única diferencia es el tipo de máscara aplicada. La máscara es de tipo Laplaciano, por que se desea detectar y realzar discontinuidades mediante la suma de la imagen original y el resultado obtenido de la convolución. El diagrama es el mismo que el de la figura 4.3.28 y los parámetros de la máscara son:

$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 7 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad (4.3\text{-}12)$$

Filtros de Rank

Estos pertenecen a una familia de filtros no lineales denominados de orden estadístico. Entre los filtros más conocidos de este tipo están los de Mediana, Máximo y Mínimo. El filtro de Mediana es ideal para eliminar ruido impulsivo del tipo sal y pimienta, ya que trabaja con valores de pixel en la mitad de la escala, es decir, entre el nivel de pimienta (usualmente 0) y sal (usualmente 255), y el filtro Máximo/Mínimo es útil para eliminar pimienta y sal respectivamente.

Para el cálculo de estos filtros se requiere disponer de un algoritmo que ordene los pixeles miembros de la vecindad de menor a mayor y de esta manera poder seleccionar el pixel en la ubicación que corresponde a cada filtro en particular: $x(0)$ para el filtro Mínimo, $x(4)$ para la Mediana y $x(8)$ para el filtro Máximo, para una vecindad de 3x3 pixeles. Entre los algoritmos más conocidos están el QUICKSORT, SHELL-METZNER y BUBBLESORT (método de la burbuja), en donde la velocidad de ejecución de cada uno de ellos depende de la complejidad a la hora de su construcción. El más rápido de estos es el

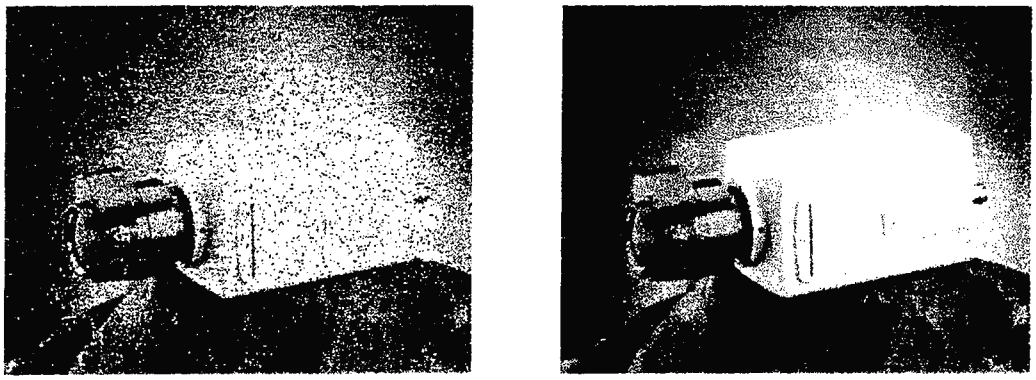
QUICKSORT, pero es el más complejo a nivel de desarrollo. Una alternativa consiste en usar el nuevo método COMB que es más eficiente que el SHELL-METZNER [23]. El método de ordenamiento COMB, en un arreglo *A* con *N* elementos, ejecuta los siguientes pasos:

1- Inicializa la variable *Nivel* con *N*, usada en la comparación de elementos.
2- Ajusta *Nivel* a 8**Nivel*/11 ó a 1; lo que sea mayor.
3- Ajusta el indicador *EnOrden* como verdadero.
4- Crea un ciclo para los valores de 0 a *N-Nivel*, mediante la variable de control de ciclo *I*:

   Asigna (*I* + *Nivel*) a *J*

   Si *A*[*I*] es mayor que *A*[*J*], intercambia *A*[*I*] con *A*[*J*] y ajusta el indicador *EnOrden* a falso
5- Continúa con el paso 2 si *Nivel* es 1 y *EnOrden* es falso.

En la figura 2.3.30(a) se presenta la imagen de la cámara afectada con ruido impulsivo a un 10% y en (b) el resultado de aplicar a esa imagen un filtro de mediana. En la figura 2.3.31 se muestra el diagrama de flujo para llevar a cabo el proceso del filtrado de orden estadístico.



a) b)

Figura 2.3.30. Aplicación del filtrado de Mediana. a) Imagen con ruido sal y pimienta al 10% y b) Imagen Filtrada.

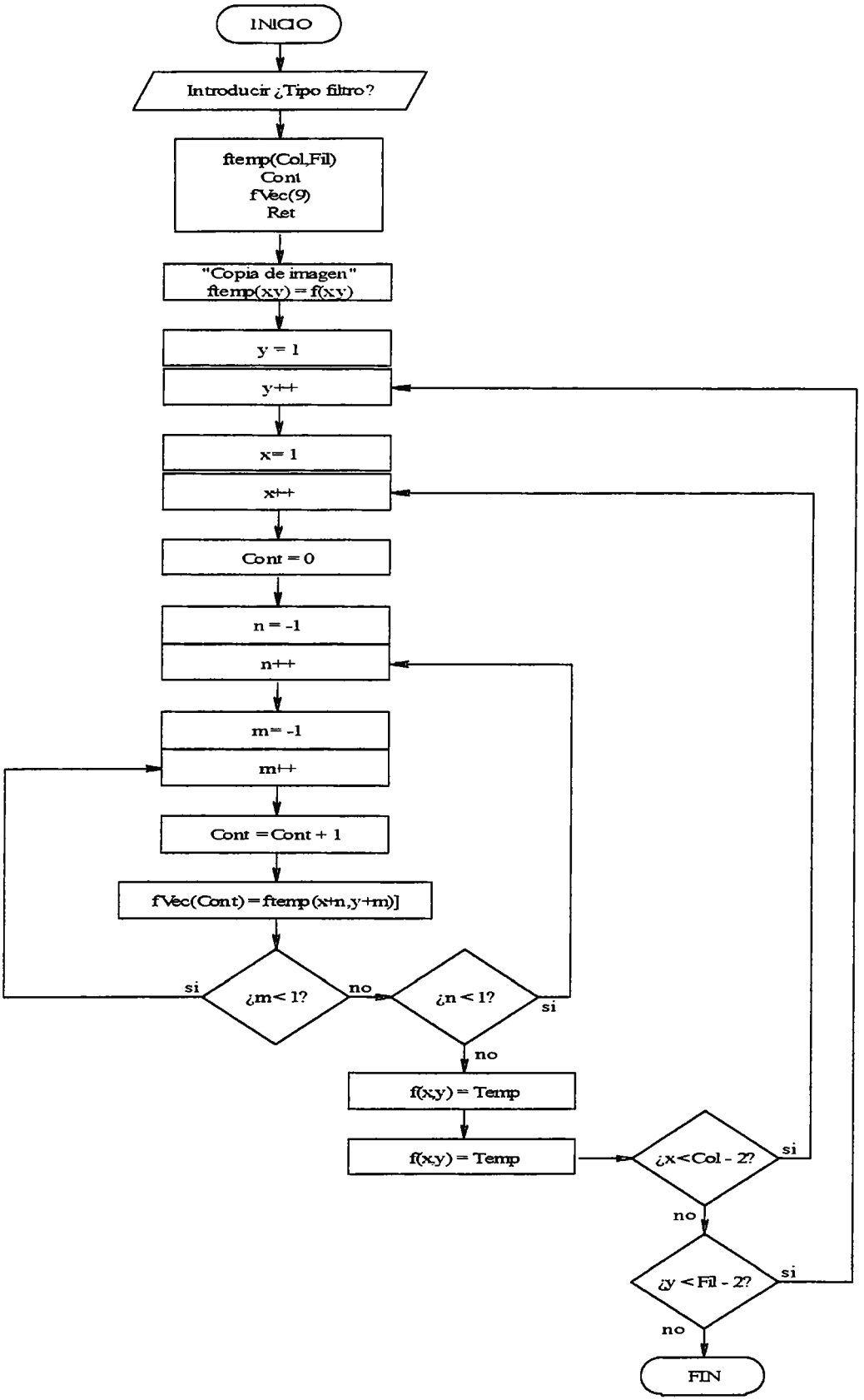


Figura 4.3.31. Diagrama de flujo para el filtrado de orden estadístico.

4.3.4.2 FILTRADO PERSONALIZADO

Es una opción incluida en el programa, para que el usuario pueda comprobar los resultados de aplicar una determinada máscara a una imagen. Para implementar esta opción se ha diseñado el cuadro de diálogo que se muestra a continuación.

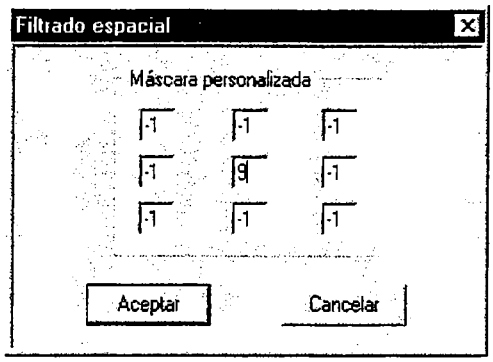


Figura 4.3.32. Cuadro de diálogo para efectuar filtrado personalizado.

En éste aparecen 9 celdas para introducir los parámetros de la máscara de 3x3 pixeles, o si se desea, se puede aceptar los valores por omisión que corresponden al detector de bordes de tipo Laplaciano. La opción del filtrado personalizado es una opción del programa que viene a reforzar la característica que hace que el programa sea considerado como un banco de trabajo, y no solamente un programa de procesamiento de imágenes, ya que permite interactuar con los resultados y evaluar las distintas variantes en cuanto al procesamiento se refiere. En la figura 4.3.33 se muestra el resultado de aplicar el filtro cuya máscara se muestra en la figura 4.3.32.

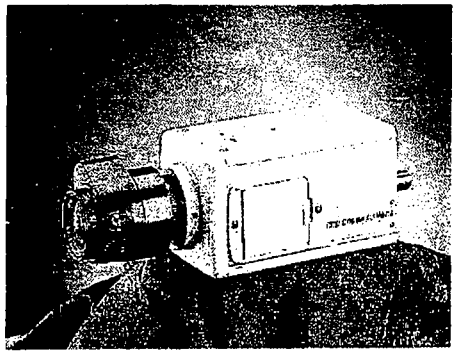


Figura 4.3.33. Imagen resultante al aplicar el filtrado personalizado.

4.3.4.3 FILTRADO EN LA FRECUENCIA

Como se explica en la sección 2.3.6 referente al filtrado de las imágenes, el filtrado en la frecuencia es una opción importante para eliminar el ruido. Además de eliminar el ruido, los filtros en frecuencia se usan en la detección de bordes (filtros pasa-altos) y en la restauración de imágenes. Para usar un filtro en frecuencia es necesario contar con el algoritmo FFT, ya que todos los cálculos se realizan sobre la representación de la imagen en frecuencia (espectro de magnitud). A los filtros que trabajan sobre el espectro de magnitud se les conoce como "de Fase Cero o Invariable" [26, 17]. Para llevar a cabo un filtrado en frecuencia usando filtros de Fase Cero, debe seguirse la secuencia mostrada en la figura 2.3.29.

Primero debe obtenerse la transformada de Fourier de la imagen y para ello debe referirse a la sección que trata la FFT en este capítulo. Una vez se tiene el espectro de magnitud de la imagen es necesario seleccionar la respuesta del filtro que se desea aplicar. En el programa se han incluido las respuestas impulso de los filtros Butterworth pasa-bajos y pasa-altos, y que se muestran a continuación

$$H(f) = \frac{1}{\sqrt{1+(\frac{f}{\omega_c})^{2N}}} \qquad (4.3\text{-}13)$$

Pasa-bajos

$$H(f) = \frac{1}{\sqrt{1+(\frac{\omega_c}{f})^{2N}}} \qquad (4.3\text{-}14)$$

Pasa-altos

donde f es igual a la raíz cuadrada de $(n^2 + m^2)$, o sea las coordenadas en la frecuencia; ω_c es la frecuencia de corte del filtro y N el orden del mismo. Es importante señalar que estos filtros poseen simetría circular, lo cual permite tomar ventaja de la distribución del espectro en frecuencia, en donde su centro (frecuencia cero o nivel DC) coincide con el centro geométrico del rectángulo de la imagen.

Para seleccionar uno de estos filtros, el programa presenta un cuadro de dialogo cuando se elige la opción. En este cuadro aparecen también las celdas, para seleccionar la

frecuencia de corte y el orden del filtro, con los que se pueden obtener diversos resultados al variar cada uno de estos valores para una misma imagen.

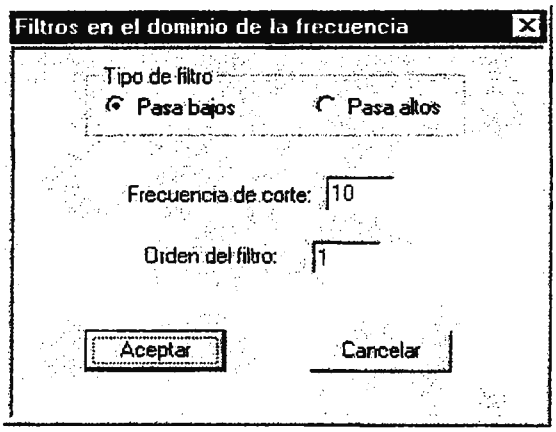


Figura 4.3.34. Cuadro de diálogo de la opción de filtrado en la frecuencia.

El proceso de filtrado consiste en realizar el producto de la respuesta del filtro con los valores de la imagen transformada. De aquí la utilidad e importancia de este tipo de filtro, en cuanto al menor tiempo de ejecución, y la posibilidad de obtener resultados con la precisión que se desee, teniendo como limitante la calidad de la respuesta impulso del filtro utilizado.

Después del filtrado es importante evaluar los resultados y para ello es necesario retornar al dominio espacial, por medio de la FFT inversa. En la figura que se presenta a continuación se puede observar el efecto de filtrado en la frecuencia sobre una imagen que ha sido adaptada a las dimensiones que exige la ejecución de la FFT y a la cual se le ha aplicado la expresión 4.3.13 [5].

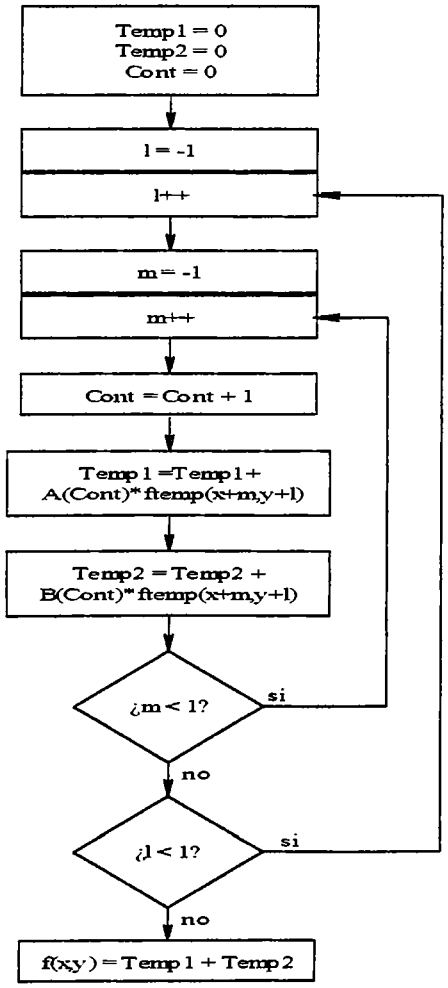
a) b)

Figura 4.3.35. a)Imagen original, b)Imagen filtrada en el dominio de la frecuencia.

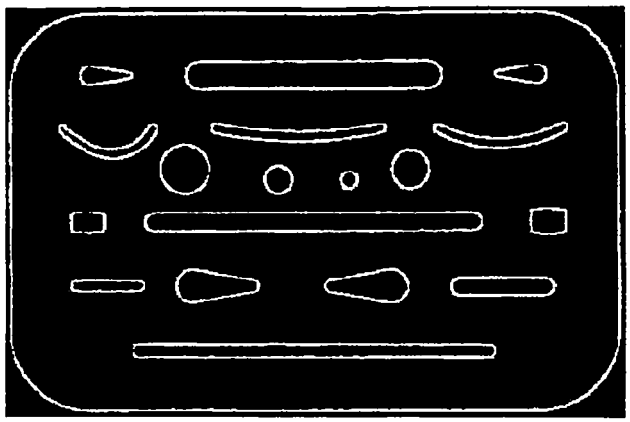
4.3.4.4 DETECTAR BORDES

El algoritmo para la detección de bordes por medio del gradiente, es muy parecido al presentado anteriormente para el filtrado espacial, ya que ambos recurren a la convolución espacial para realizar el procesamiento. La única diferencia está en el uso de más de una máscara usada en la convolución, lo que se define como una doble convolución para el caso de algunos operadores como el de Sobel o Robert (gradiente ortogonal).

Para llevar a cabo la detección de bordes, se necesita sustituir el bloque de la convolución de la figura 4.3.28 por el diagrama mostrado a continuación, donde *A*(*Cont*) y *B*(*Cont*) representan los coeficientes de las dos máscaras del detector de bordes en las direcciones de *x* y *y* respectivamente y, además, *Cont* toma valores desde 1 hasta 9.



a)



b)

Figura 4.3.36. Convolución para la detección de bordes por gradiente ortogonal. a) Convolución aplicada para la detención de bordes, b) Un ejemplo de aplicar a).

La figura 4.3.36(b) muestra el resultado de su aplicación a la imagen de la figura 4.3.1(a), haciendo uso de las máscaras de los detectores de Sobel. El algoritmo permite, además, usar otras máscaras como la Prewitt, Frei y Chen entre otras cuyas máscaras en la dirección x y y se muestran a continuación:

$$f_x(x,y)=\begin{bmatrix}-1 & 0 & 1\\ -1 & {}^{\bullet}0 & 1\\ -1 & 0 & 1\end{bmatrix} \qquad f_y(x,y)=\begin{bmatrix}-1 & -1 & -1\\ 0 & {}^{\bullet}0 & 0\\ 1 & 1 & 1\end{bmatrix} \tag{4.3-13}$$

Prewitt

$$f_x(x,y)=\begin{bmatrix}-1 & 0 & 1\\ -\sqrt{2} & {}^{\bullet}0 & \sqrt{2}\\ -1 & 0 & 1\end{bmatrix} \qquad f_y(x,y)=\begin{bmatrix}-1 & -\sqrt{2} & -1\\ 0 & {}^{\bullet}0 & 0\\ 1 & \sqrt{2} & 1\end{bmatrix} \tag{4.3-14}$$

Frei y Chen

4.3.4.5 MORFOLOGÍA DE IMAGEN

Morfología Binaria

Para obtener un resultado lógico y satisfactorio haciendo uso de la morfología de imagen binaria, es necesario que la imagen sometida sea binaria, ya que para el algoritmo desarrollado, el objeto estructural se compone de una matriz binaria de 3×3 con la que se efectuará el procedimiento de convolución. El elemento estructural que en general se utiliza para la erosión y dilatación, se compone de una arreglo en el cual los niveles de 255 de la escala, forman aproximadamente un circulo así como se presenta en la figura 2.3.14 de la sección 2.3.5.

Si el usuario desea realizar un proceso morfológico activando la opción, se encontrará con el cuadro de diálogo que se presenta en la siguiente figura. Obsérvese la posibilidad de iterar un máximo de cinco veces para cualquiera de los procedimientos.

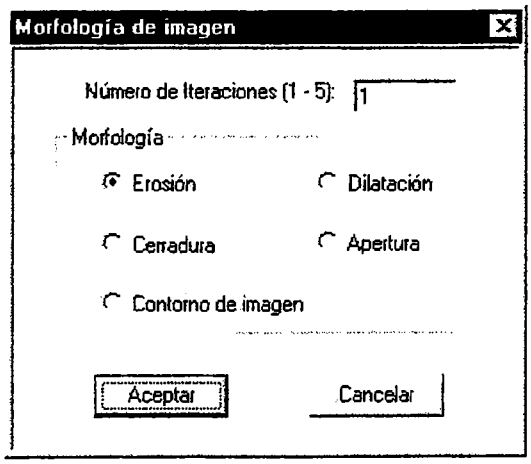


Figura 4.3.37. Cuadro de diálogo que permite elegir un proceso de morfología de imagen.

La forma en la que se ha organizado cada uno de los procesos, ha sido partiendo desde las operaciones más básicas como lo es la erosión y la dilatación, el resto de ellas resultan de una combinación de las primeras. El contorno de imagen que aparece como una

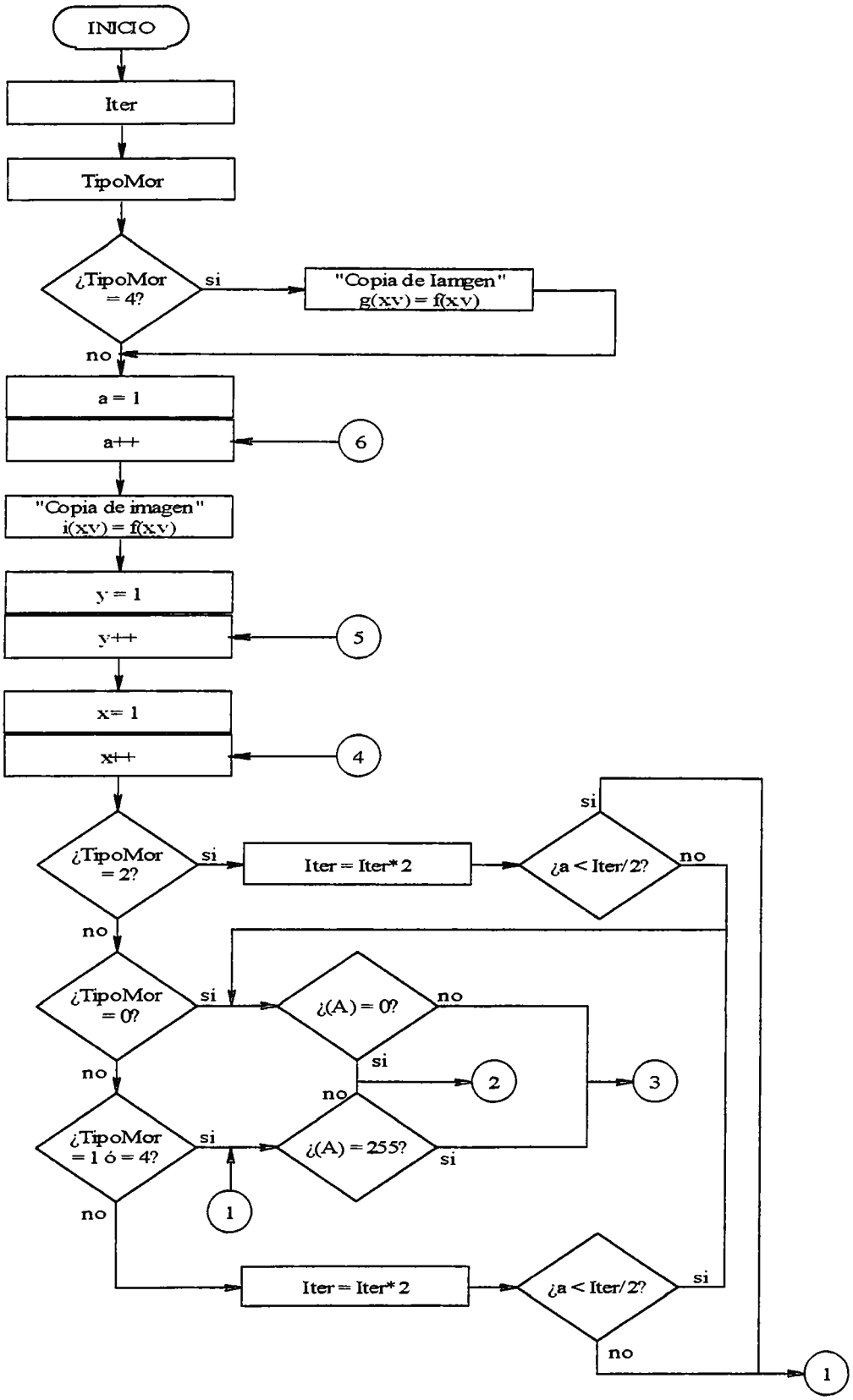
opción basada en la morfología, representa una alternativa para evaluar los bordes existentes en una imagen binaria. Las operaciones que se encuentran detrás de este resultado, se ejecutan así: en primer lugar se aplica un dilatado de la imagen inicial, como segundo y último paso se precisa que la imagen dilatada se reste con la imagen base, lo cual se consigue a través de la operación X-OR o directamente con la resta en el ámbito de pixeles.

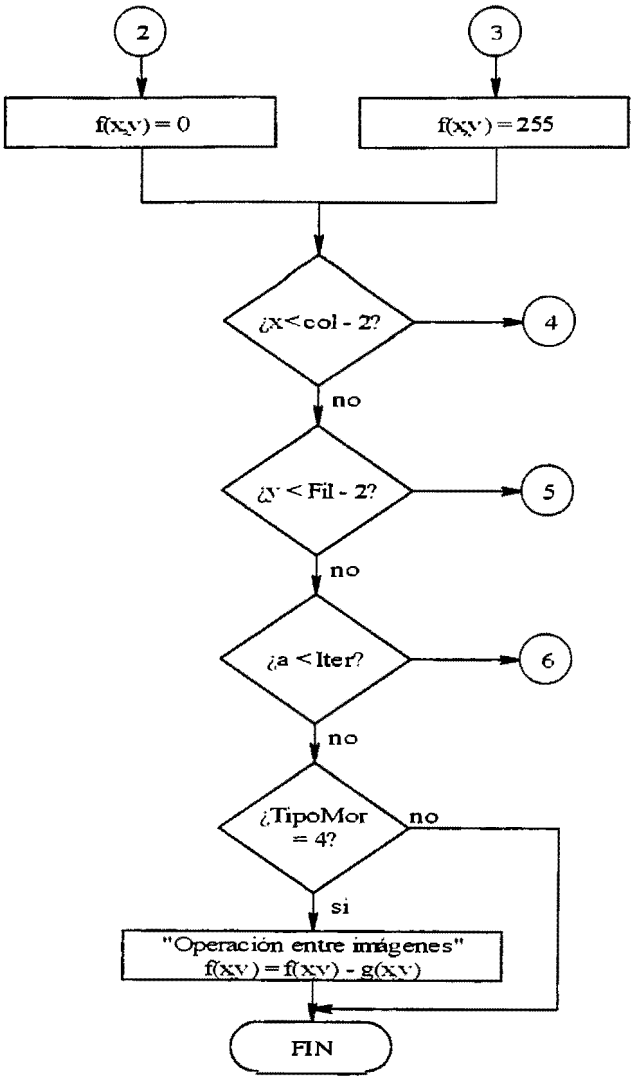
La diferencia principal entre el procedimiento "Contorno de imagen" y la opción "Detectar bordes", consiste en que la primera es realizable de manera satisfactoria para imágenes totalmente binarias, mientras que en la segunda, la intensidad del borde depende de la variación de la intensidad en la dirección de las coordenadas espaciales.

Cada uno de las operaciones que contempla el cuadro de diálogo, se han logrado haciendo uso de la teoría de conjuntos, lo cual se materializa en las decisiones que se presentan en el algoritmo de la figura 4.3.38(a) como ¿(A) = 0? ó ¿(A) = 255?. Se efectúa una copia temporal de las imágenes a ser modificadas denotadas por la función $i(x,y)$, con el objeto de mantener las propiedades de la misma y poder trabajar con sus elementos, mientras se modifica la matriz que la contiene. Se hace la aclaración que se ha utilizado la simbología (A) para hacer denotar el siguiente conjunto en el diagrama de flujo: $i(x\text{-}1,y)$ ó $i(x,y)$ ó $i(x+1,y)$ ó $i(x,y+1)$ ó $i(x,y\text{-}1)$, la magnitud que almacenan representa el resultado de una convolución entre la imagen y la máscara utilizada. Se han tomado cinco elementos porque es posible que en ellos no se produzcan multiplicaciones nulas, y a partir de esos resultados poder concluir que la operación lógica OR es cierta cuando cualquier elemento de (A) es igual a 255; caso contrario para la operación AND, que es cierta cuando algún elemento de (A) se le atribuye el valor 0 en la escala de grises.

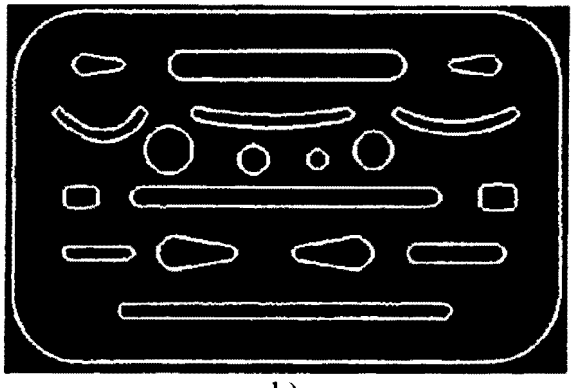
TipoMorfo	Tipo de morfología
0	Erosión
1	Dilatación
2	Cerradura
3	Apertura
4	Contorno de imagen

Tabla 4.3.3. Tipos de morfología de imagen implementadas en el algoritmo.


INICIO
Iter
TipoMor
¿TipoMor = 4?
si
"Copia de Iamgen" g(xy) = f(xy)
no
a = 1
a++
6
"Copia de imagen" i(xy) = f(xy)
y = 1
y++
5
x= 1
x++
4
¿TipoMor = 2?
si
Iter = Iter* 2
¿a < Iter/2?
si
no
no
¿TipoMor = 0?
si
¿(A) = 0?
no
si
no
2
3
no
¿TipoMor = 1 ó = 4?
si
¿(A) = 255?
si
1
no
Iter = Iter* 2
¿a < Iter/2?
si
no
1



a)



b)

Figura 4.3.38. Contorno de imagen haciendo uso de la morfología. a) Diagrama de flujo, b) Resultado al aplicar "Contorno de imagen" a la imagen de la figura 4.3.18 con una iteración igual a cuatro.

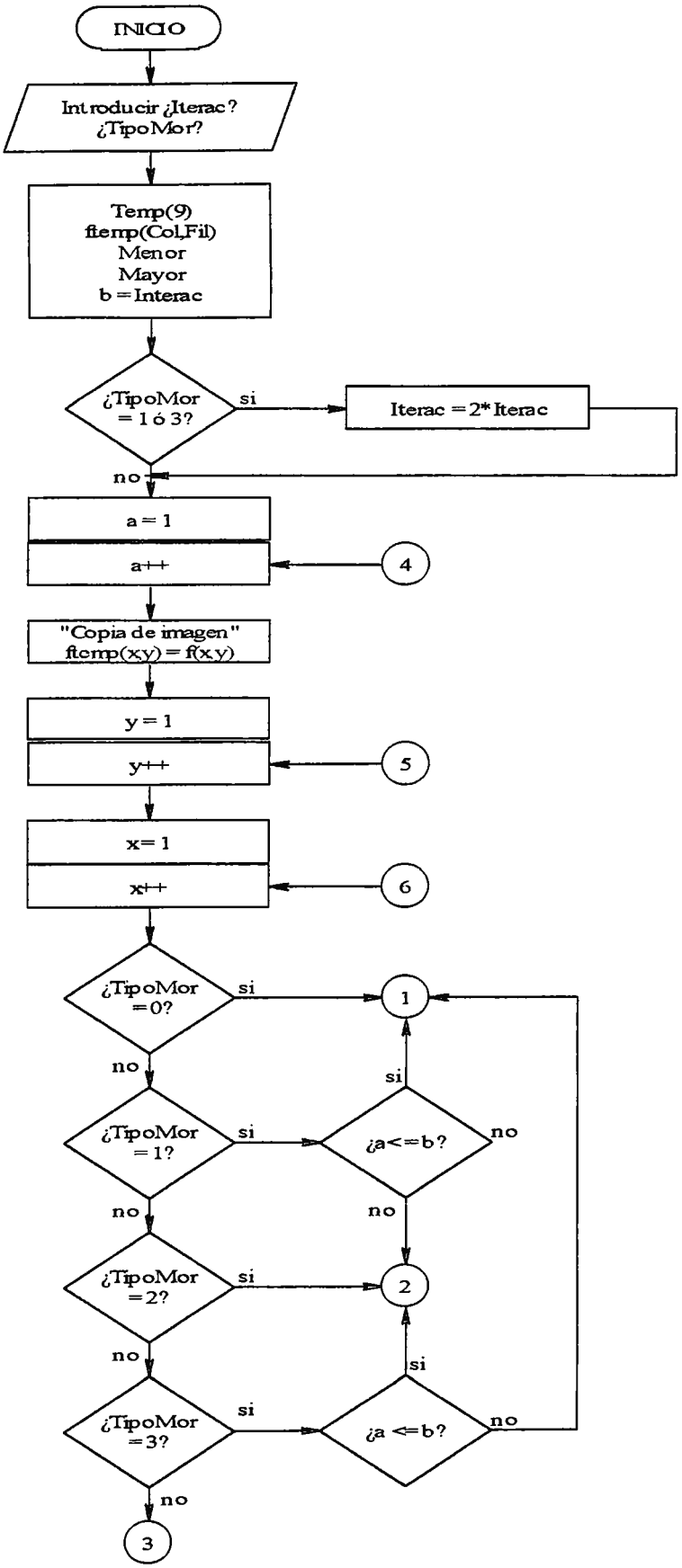
Morfología en Niveles de Gris

A diferencia de la morfología binaria tratada anteriormente, este tipo de morfología se usa en imágenes que no son binarias. Los conceptos y operaciones realizadas son similares a los usados en la morfología binaria, es decir, es posible erosionar, dilatar, y desde luego la apertura y cerradura como producto de la combinación de las dos primeras. En morfología binaria el concepto fundamental consiste en la modificación del área y la forma geométrica de un objeto. Para describir un objeto en niveles de gris, es necesario tener una representación en tres dimensiones, dos para la posición y una para caracterizar los niveles de gris de cada pixel dentro de la imagen. Por lo tanto la morfología en niveles de gris trabaja con volúmenes.

El elemento estructural para llevar a cabo las operaciones básicas de erosión y dilatación es una máscara que representa una semiesfera en tres dimensiones, la que a continuación se presenta

$$s(x,y) = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad (4.3\text{-}15)$$

Esta máscara se usa de la misma forma que el elemento estructural de la erosión y dilatación binaria, es decir, se basa en el mismo concepto del "círculo rodante" presentado en la figura 2.3.10(b) y 2.3.13 del capítulo 2.3. En una imagen binaria los niveles de gris forman una superficie compleja, en la cual la semiesfera representada por la máscara, se traslada sobre toda la imagen para generar una nueva superficie erosionada o dilatada dependiendo, si los coeficientes de la máscara se suman o se restan a los correspondientes valores de la imagen. La figura 4.3.39 muestra el diagrama de flujo de la morfología en niveles de gris. Los dos valores de entrada son el número de iteraciones de la operación y el tipo de morfología aplicada. Esta última variable puede tomar cualquier valor de cuatro posibles que se presentan en la tabla 4.3.4.


INICIO
Introducir ¿Iterac?
¿TipoMor?
Temp(9)
ftemp(Col,Fil)
Menor
Mayor
b = Interac
¿TipoMor = 1 ó 3?
si
Iterac = 2* Iterac
no
a = 1
a++
4
"Copia de imagen"
ftemp(x,y) = f(x,y)
y = 1
y++
5
x= 1
x++
6
¿TipoMor =0?
si
1
no
¿TipoMor =1?
si
¿a<=b?
si
no
no
no
¿TipoMor =2?
si
2
no
¿TipoMor =3?
si
¿a <=b?
si
no
no
3

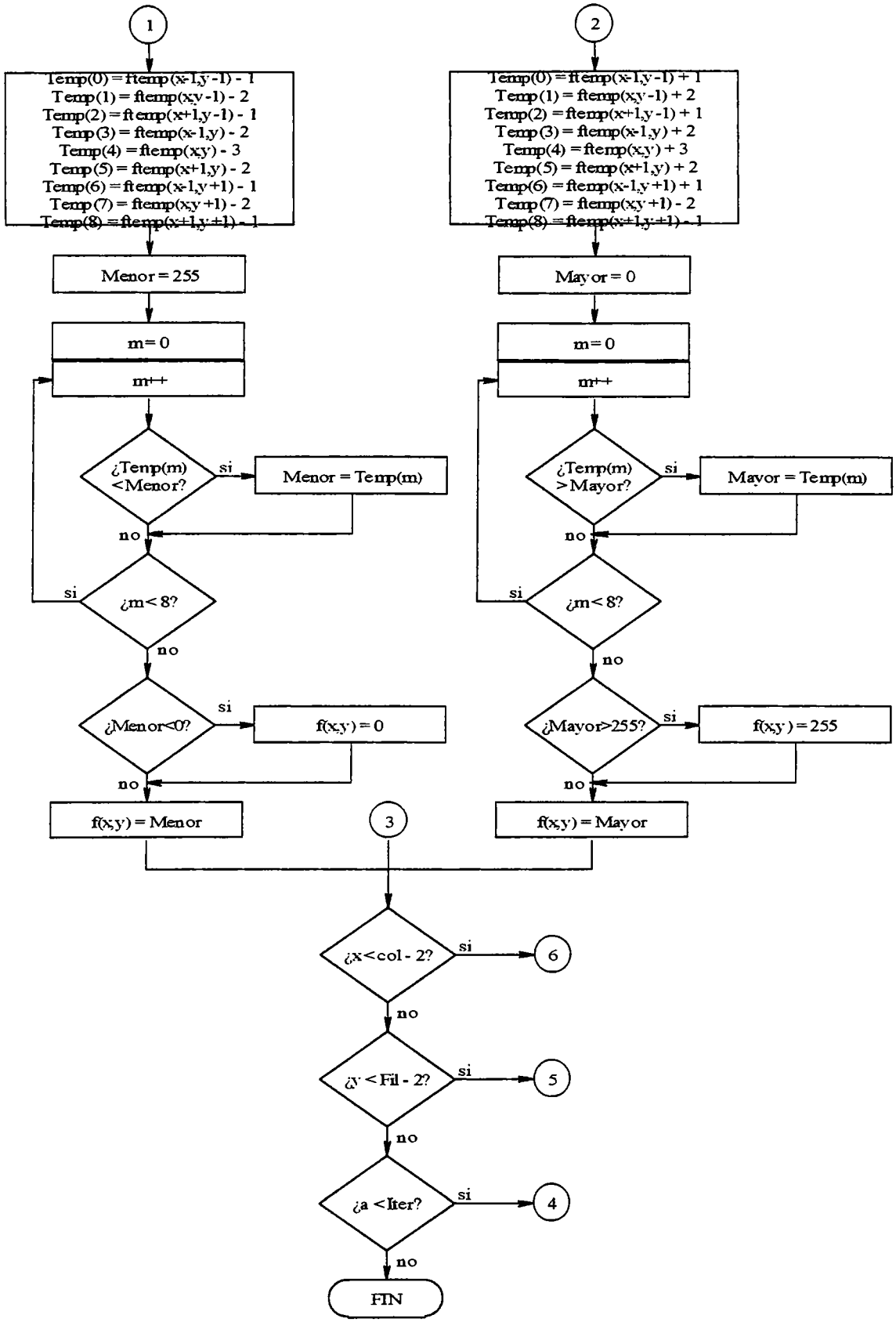


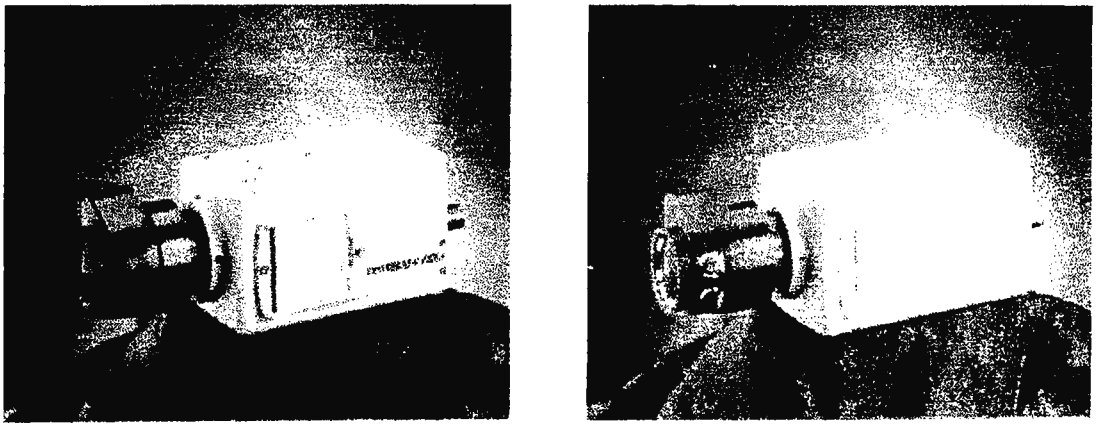
Figura 4.3.39. Diagrama de flujo para morfología en niveles de gris.

TipoMor	Tipo de morfología
0	Erosión
1	Apertura
2	Dilatación
3	Cerradura

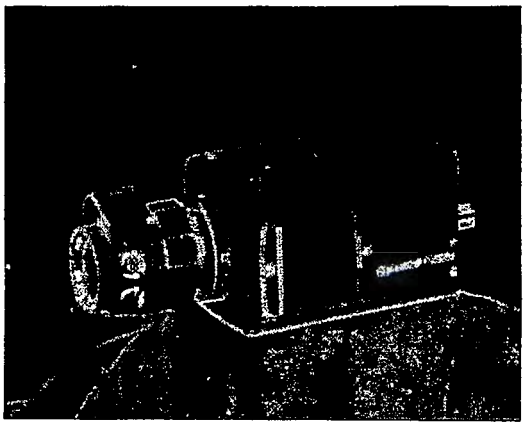
Tabla 4.3.4. Tipos de morfología en niveles de gris implementadas en el algoritmo.

Al seleccionar *TipoMor* = 1 ó 3, el número de iteraciones se duplica para tener igual cantidad de iteraciones para la erosión y la dilatación. Para la operación de erosión, se realiza primero la resta de cada pixel de la vecindad de 3x3 con su correspondiente parámetro en la máscara. Después se ordena la vecindad y obtiene el menor de ellos, sustituyendo el pixel que se evalúa por este valor. Y este proceso se ejecuta recurrentemente en toda la imagen y para cada iteración. Tratándose de la dilatación, los parámetros de la máscara se suman con la vecindad y se obtiene el mayor de los resultados, y después de esto se prosigue como en la erosión.

En la figura 4.3.40 se demuestra la aplicación de la morfología en niveles de gris, para la detección de bordes. El método consiste en obtener una imagen erosionada y otra dilatada de la misma imagen que se desea detectar sus bordes. Ambas imágenes se restan pixel a pixel y el resultado es la imagen de bordes detectados.



a) b)



c)

Figura 4.3.40. Aplicación de la morfología en niveles de gris como detector de bordes. Imagen a) erosionada, b) dilatada y c) resultado de la resta de (a) y (b) con bordes detectados.

4.3.4.6 ARITMÉTICA Y LÓGICA

Estas son las operaciones básicas de procesamiento. Casi todos los programas comerciales de procesamiento incluyen la posibilidad de modificar las imágenes de esta manera. Al seleccionar una de estas opciones en el programa, se presenta un cuadro de dialogo para cada uno de los casos y el tipo de operación, a saber la suma, resta, multiplicación y división.

Se cuenta con ciertos espacios interactivos en los que el usuario puede introducir el valor del operador. Este operador toma valores comprendidos desde 0.5 hasta 2.5 para las operaciones aritméticas y así mismo desde 20 hasta 200 para las lógicas. Este tipo de operación se efectúa en el ámbito de pixel y son ampliamente utilizadas para efectuar el realce de imágenes que poseen una definición pobre, así se puede modificar el contraste de las imágenes por medio de la multiplicación o bien incrementar el brillo a través de la suma.

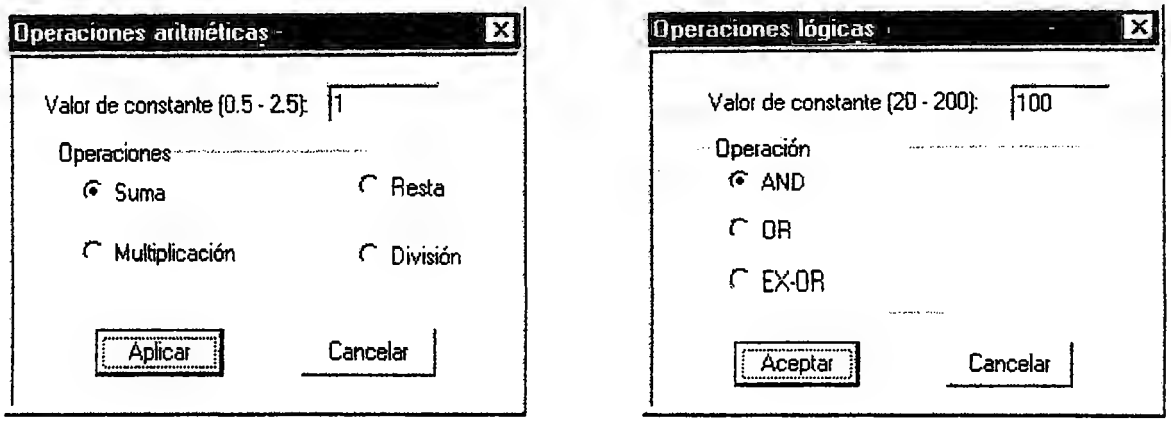
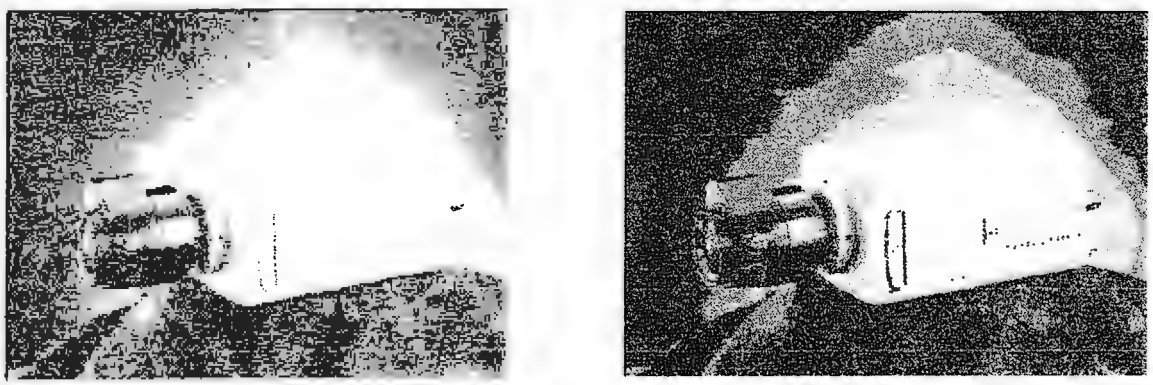


Figura 4.3.41. Cuadros de diálogo para las operaciones aritméticas y lógicas.

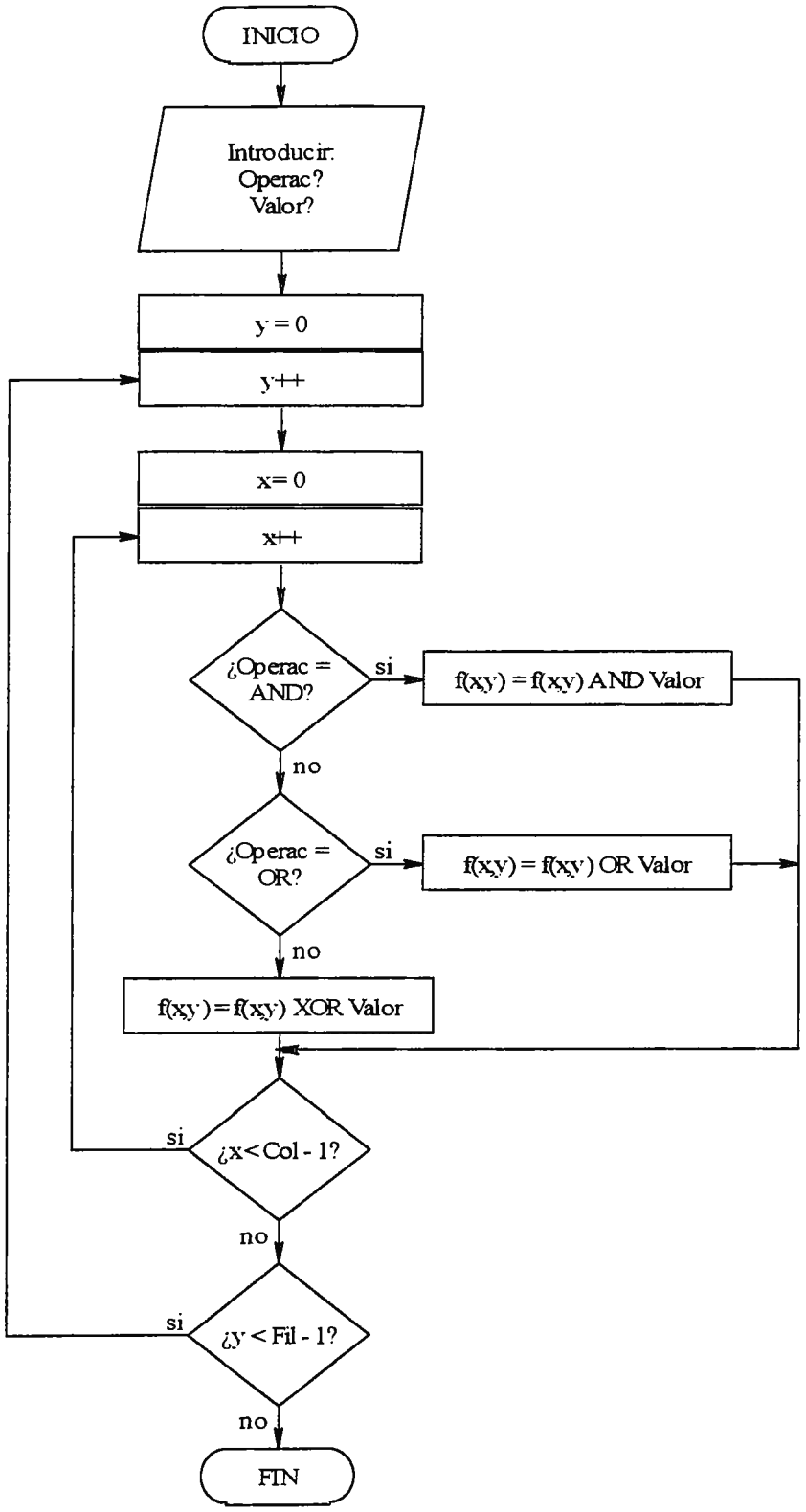
Las operaciones lógicas entre una constante y cada uno de los pixeles que componen la imagen, son usados ampliamente para producir resultados análogos a las operaciones aritméticas, entre esos resultados podemos decir que la operación aritmética de la resta produce el mismo resultado que la operación lógica X-OR.

La utilización de estas operaciones se muestra en la figura 4.3.42, donde se ha seleccionado la imagen de la figura 4.3.1(b) y se le ha sumado el valor de 1.5 para generar la imagen en (a) y en otra imagen (b) se ha utilizado un operador OR con un valor constante de 50. Los algoritmos para estos procesos se muestran en la figura 4.3.43(a) y (b). Si el valor resultante luego de la operación es mayor que 255 el programa fija este valor a 255.

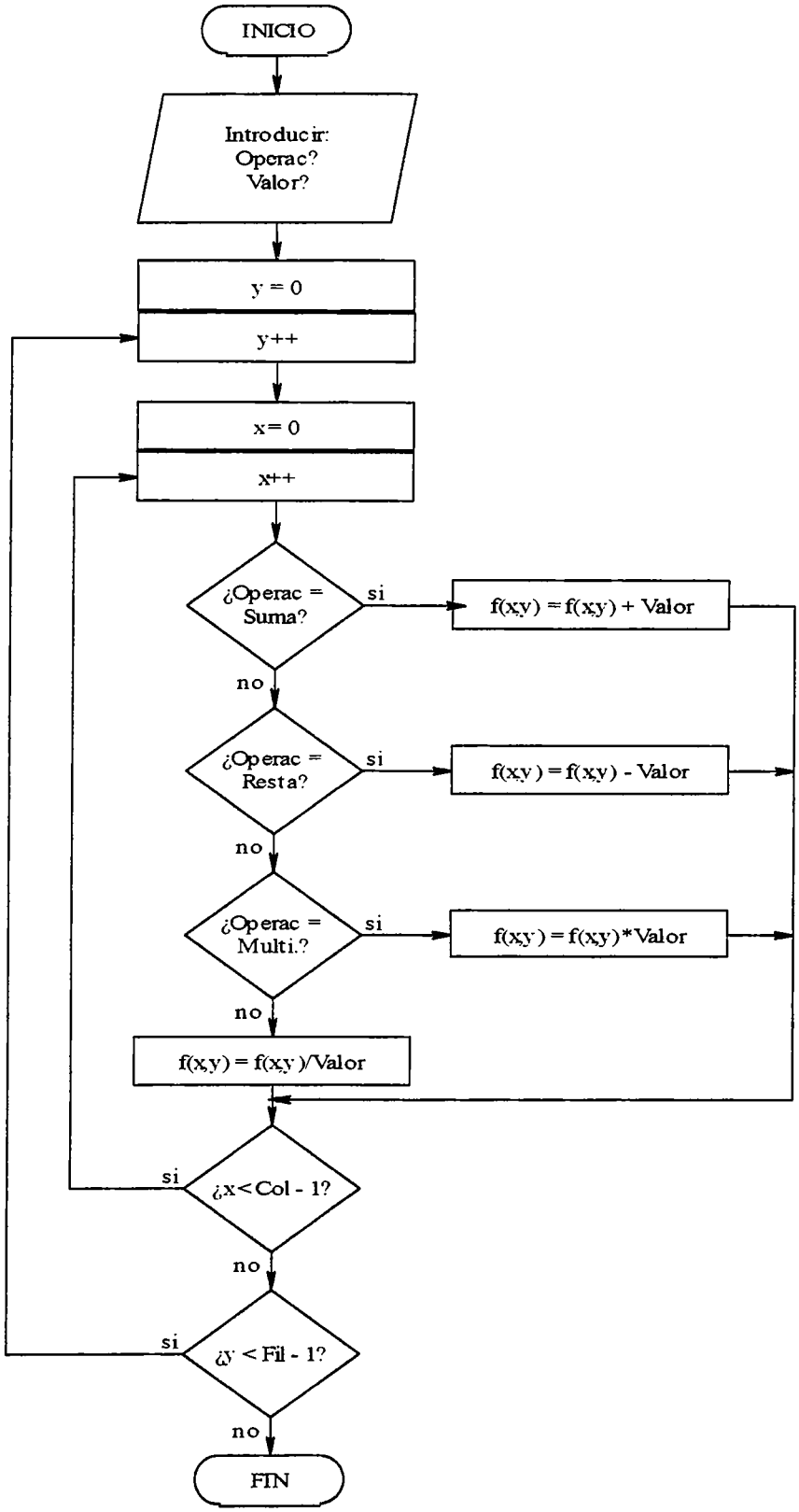


a) b)

Figura 4.3.42. Operación con imagen de la figura 4.3.1(b). a) suma con 1.5 y b) OR con 50.


INICIO
Introducir:
Operac?
Valor?
y = 0
y++
x= 0
x++
¿Operac = AND?
si
f(x,y) = f(x,y) AND Valor
no
¿Operac = OR?
si
f(x,y) = f(x,y) OR Valor
no
f(x,y) = f(x,y) XOR Valor
si
¿x<Col - 1?
no
si
¿y < Fil - 1?
no
FIN

a)



b)

Figura 4.3.43. Diagrama de flujo para las operaciones a) Lógicas y (b) Aritméticas.

4.3.4.7 OPERACIONES ENTRE IMÁGENES

Las operaciones entre imágenes siguen siendo operaciones básicas que están contempladas en el procesamiento de imágenes, por lo que la implementación de su algoritmo es relativamente sencilla. Todas las operaciones incluidas hoy en día y que en términos generales se refiere a las operaciones aritméticas y lógicas, se llevan a cabo en el ámbito de pixel, es decir, el elemento $f(x,y)$ de una imagen se opera con el elemento $g(x,y)$ de otra imagen.

Para poder usar correctamente cada una de las opciones implementadas en el algoritmo, es necesario con anterioridad definir una imagen patrón, la cual se guarda temporalmente en un bloque de memoria y que es reutilizable posteriormente. La imagen patrón es manejada con un puntero del tipo estático, el cual es el responsable de preservar las características de la misma cuando se alterna de una imagen hacia otra dentro del marco principal de la aplicación [9]. El lector puede remitirse al resultado obtenido en la imagen de la figura 4.3.36(c) para verificar el empleo y la versatilidad que esta opción proporciona.

4.3.4.8 ECUALIZADO

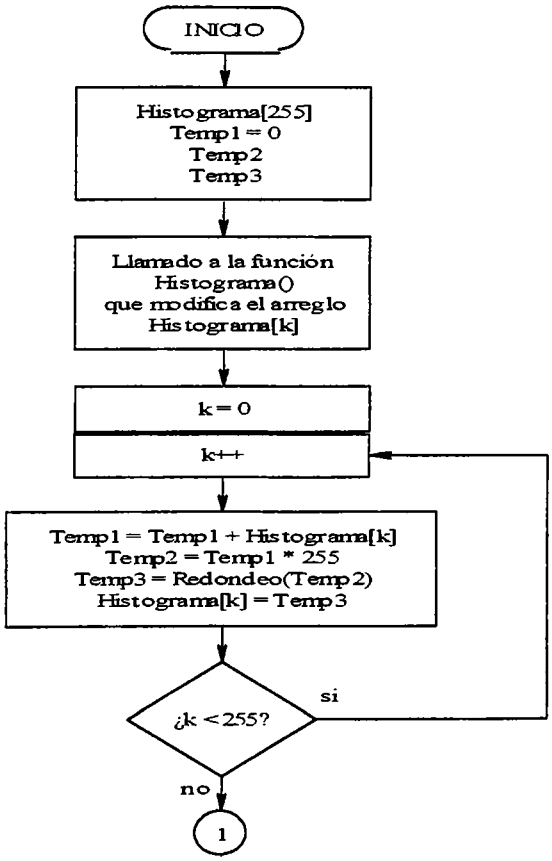
Es la aplicación en el uso del histograma, con el objeto de mejorar las características estructurales de una imagen (contraste y brillo). Es una forma fácil de realzar una imagen que será sometida a un procesamiento posterior. El método utilizado en este caso, es el que se define como la ecuación de histograma y que se discute en la sección 2.3.1. La función de transferencia usada es la ecuación (2.3-14) y el algoritmo para su aplicación es el que se muestra en la figura 4.3.44.

El programa se encarga de aproximar un valor real al entero próximo superior o inferior, ya que con las imágenes, solo se trabaja con valores enteros y se hace de esa

manera siguiendo los pasos de una suma acumulada en la que el valor actual depende del valor anterior. Es importante notar que la función *Histograma*() es la solicitud de servicio proporcionado por el algoritmo que se ejemplifica en la figura 4.3.22, el cual permite modificar y actualizar los valores de histograma para la imagen activa.

Para mostrar un ejemplo de los resultados obtenidos al usar esta técnica de realce de imagen, se dispone de la figura 4.3.45, en la que se presenta la imagen resultante y el histograma uniformemente distribuido de la imagen de la figura 4.3.1(b), cuyo histograma original se aprecia en la figura 4.3.21. Es importante notar que al comparar ambos histogramas e imágenes se aprecia una clara diferencia y que esta imagen que originalmente era más oscura, ahora ha tomado un aspecto claro y con una mejor definición.

Después de aplicado el ecualizado, la imagen luce con más brillo y con un mejor contraste, lo que nos permite poder apreciar detalles finos de la imagen que en la original estaban ocultos, como por ejemplo la textura del manto debajo de la cámara.



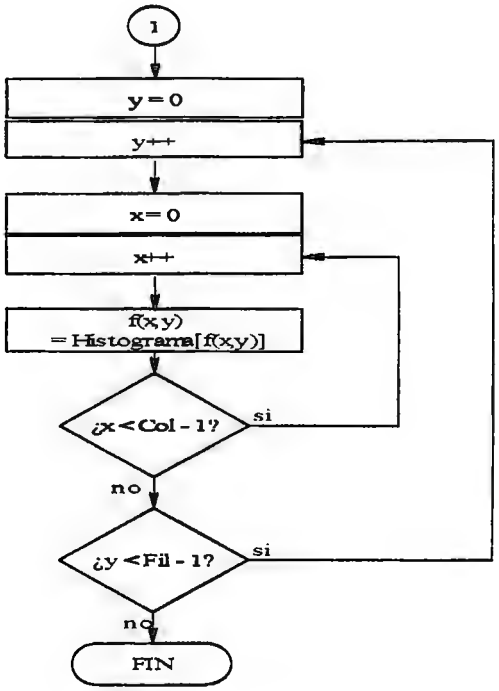
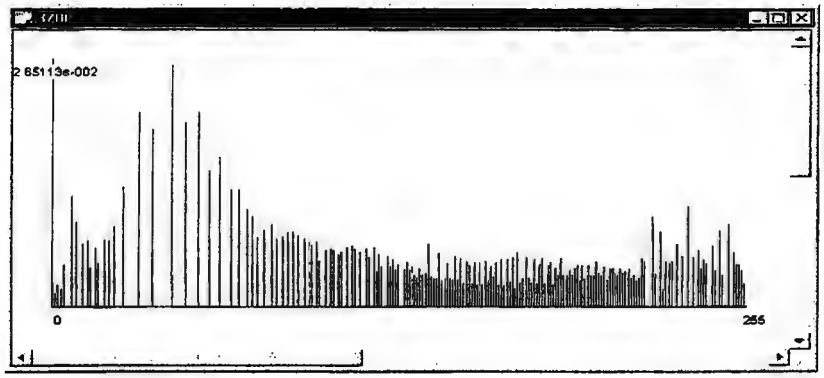
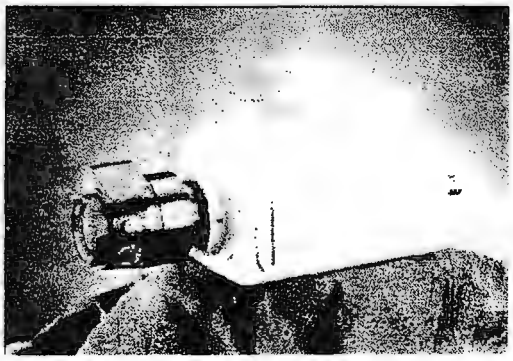


Figura 4.3.44. Algoritmo del ecualizado de imagen.



a)



b)

Figura 4.3.45. Aplicación del ecualizado sobre la imagen de la figura 4.3.1(b), a) Histograma y b) Imagen ecualizada.

Un campo de aplicación del ecualizado se encuentra en imágenes estelares que en la mayoría de los casos son captadas por medio de telescopios electrónicos y en las que la iluminación de la escena obviamente no puede ser manipulada.

4.3.4.9 BALANCE DE IMAGEN

Se le llama así al procedimiento que se encarga de modificar el contraste y el brillo a imágenes con características oscuras o bajas en definición. Se puede tener una imagen con una definición pobre por diversas razones que a lo largo del desarrollo se han discutido, por lo tanto aquí se tiene una forma de obtener el realce de aquellos detalles de interés contenidos en ciertas imágenes y que resulta imposible visualizar en primera instancia. El balance en la imagen es otra de las técnicas que permite hacerlo y que por medio del cual se persigue el resultado antes mencionado. Así como otros procesos, éste llama a un cuadro de diálogo que sirve como una interface para que el usuario defina los parámetros que serán reutilizados para el cálculo del brillo y del contraste en la imagen.

El algoritmo que se encarga de dar servicio a las llamadas de balance de imagen se ejemplifica en el diagrama de flujo que se presenta en la figura 4.3.49. Para la función manual y lineal se ha hecho uso de la función de mapeo que se define a continuación.

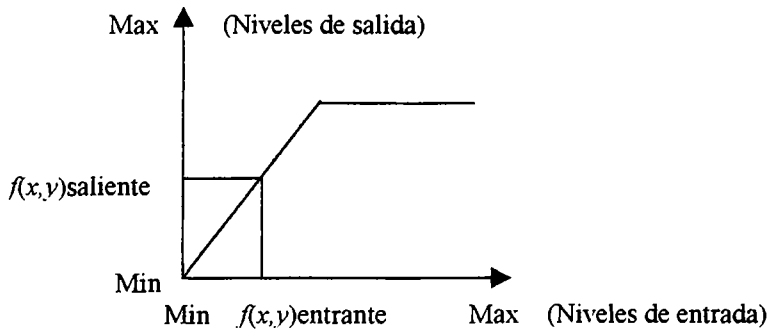


Figura 4.3.46. Una función básica de mapeo para encontrar el balance de imagen.

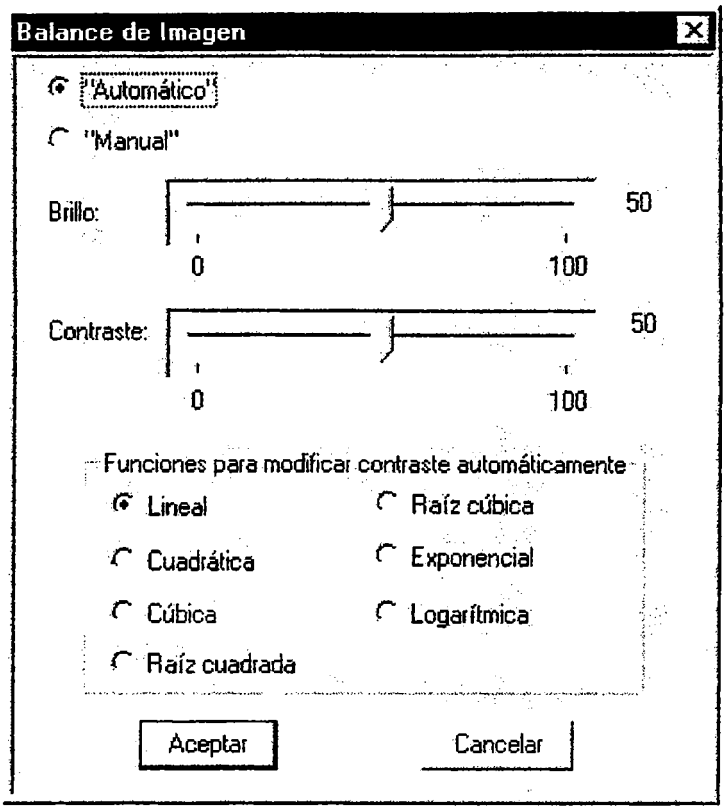


Figura 4.3.47. Cuadro de diálogo para definir los parámetros del balance de imagen.

Supóngase que el usuario prefiere un balance de imagen del tipo "Automático lineal". La rutina de servicio se encarga inicialmente de calcular los niveles máximo y mínimo de la imagen, los cuales se ven representados por las variables *Temp*1 y *Temp*2 del diagrama de flujo. Además de asignarle un nuevo valor a $f(x,y)$, en virtud de la función de transferencia que se presenta en la figura 4.3.46 definida por las variables antes mencionadas.

Ahora bien, si se desea efectuar un balance de imagen del tipo "Manual"; el algoritmo simplemente asocia un nuevo valor a $f(x,y)$, de acuerdo a la función de transferencia anteriormente presentada y que en esta ocasión se define de acuerdo a los parámetros de contraste y brillo elegidos por el usuario. La variable *nB* del diagrama de flujo guarda el promedio de los niveles de gris de la imagen, ésta se incluye en el procedimiento manual con el objeto de que las variables *Contraste* y *Brillo* produzcan un efecto totalmente independiente sobre el resultado global.

A continuación se presenta , la aplicación de balance de imagen manual. Nótese como cambia el fondo de la imagen y la forma en la que se agrega sombra a la misma, al aumentar el contraste y disminuir el brillo.

a)

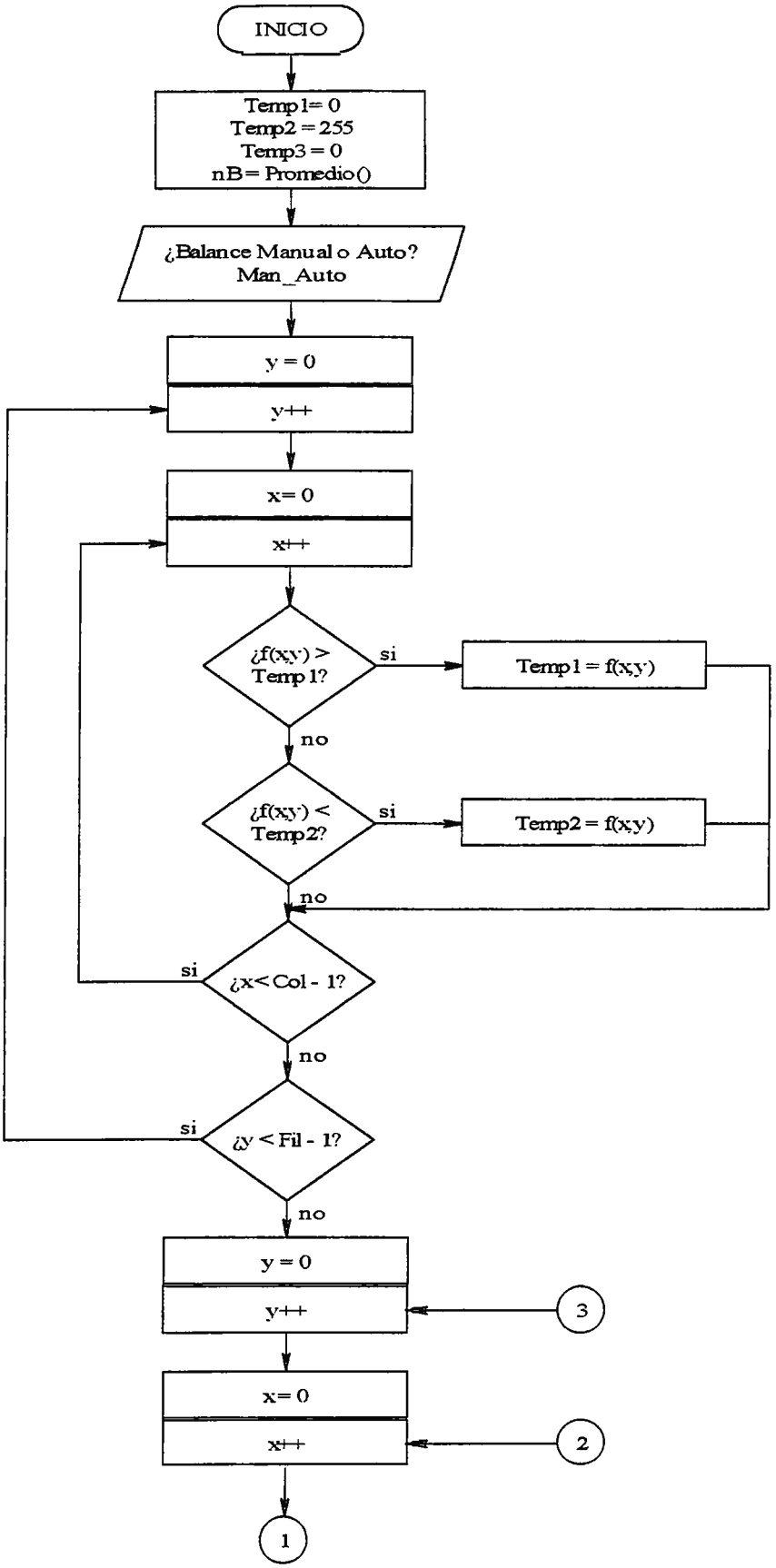
b)

Figura 4.3.48. Balance de imagen. a) Imagen original. b) Resultado de modificar el contraste y brillo de a).

En la siguiente tabla se presentan los valores que toma la variable *TipoBal* y de acuerdo a esa función, se aplica a la imagen la función de transferencia respectiva.

TipoBal	Función de transferencia
0	Lineal
1	Cuadrada
2	Cúbica
3	Raíz cuadrada
4	Raíz cúbica
5	Exponencial
6	Logarítmica

Tabla 4.3.5. Funciones de transferencia implementadas en el balance de imagen.


INICIO
Temp1= 0
Temp2 = 255
Temp3 = 0
nB = Promedio()
¿Balance Manual o Auto?
Man_Auto
y = 0
y++
x= 0
x++
¿f(x,y) > Temp1?
si
Temp1 = f(x,y)
no
¿f(x,y) < Temp2?
si
Temp2 = f(x,y)
no
si
¿x< Col - 1?
no
si
¿y < Fil - 1?
no
y = 0
y++
3
x= 0
x++
2
1

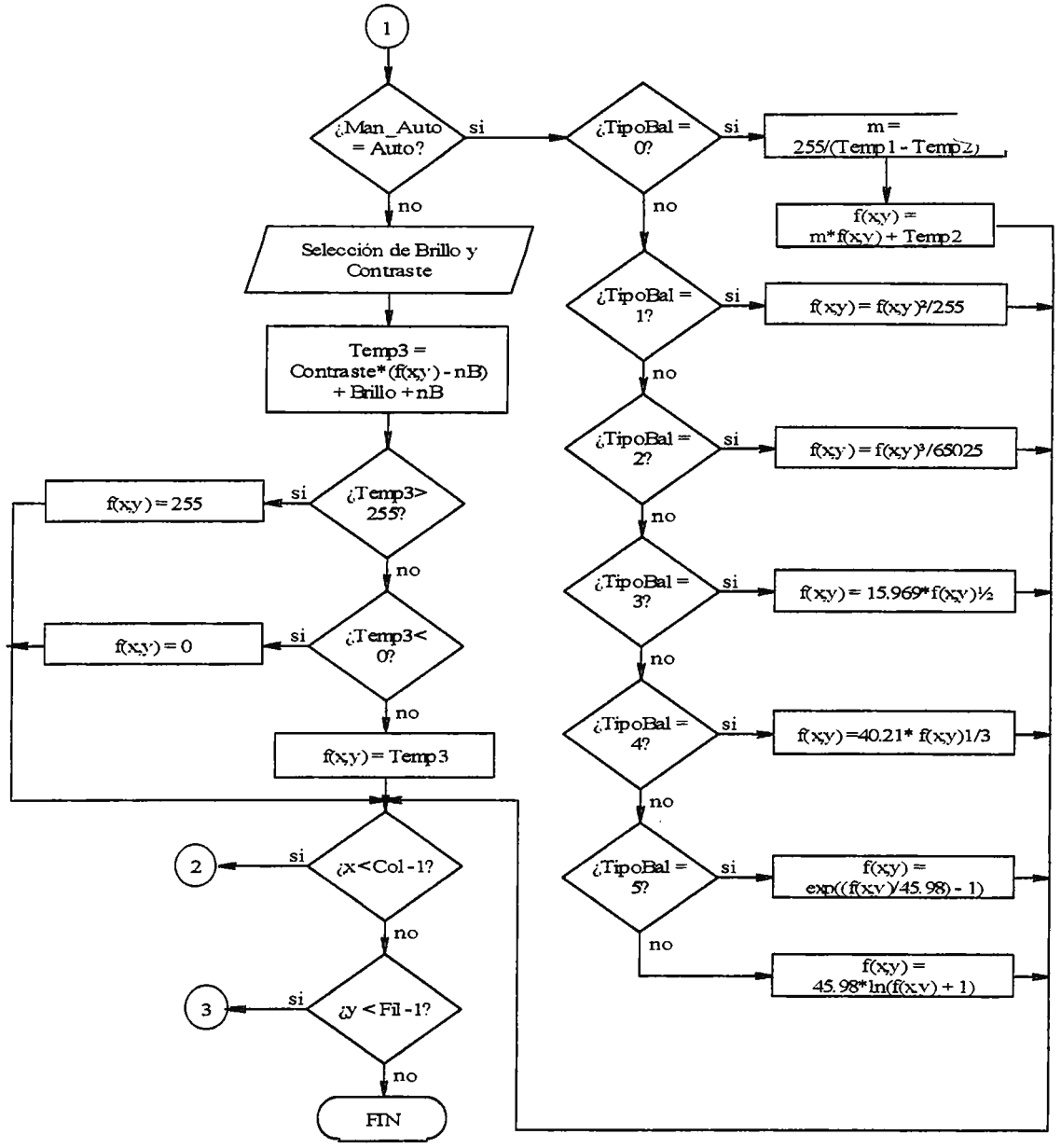


Figura 4.3.49. Diagrama de flujo para obtener el balance de imagen.

4.3.4.10 TRANSFORMADA DE FOURIER RÁPIDA

Las transformadas de imagen, juegan un papel muy importante en el procesamiento digital de imágenes como una herramienta teórica y de implementación en numerosas tareas,

notablemente en el filtrado, restauración, codificación y análisis de imágenes. Las transformadas de la imagen son a menudo de orden lineal debido a su simplicidad de implementación, algunas de las transformadas más comunes y utilizadas son las denominadas transformadas de Fourier.

La serie discreta de Fourier en dos dimensiones, se puede usar para caracterizar señales periódicas en dos dimensiones lo que resulta de una extensión de la definición en una dimensión [17].

La transformada discreta de Fourier es otra herramienta importante en el procesamiento y análisis de señales discretas en dos dimensiones, dado que permite un cálculo directo de cada uno de los elementos que la componen.

A continuación se presenta el par de transformadas que dan como resultado una función en el dominio de la frecuencia y espacio respectivamente

$$F(u,v) = \frac{1}{MN} \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)} \tag{4.3-16}$$

$$f(x,y) = \sum_{v=0}^{M-1} \sum_{u=0}^{N-1} F(u,v) e^{j2\pi(ux/M + vy/N)} \tag{4.3-17}$$

donde N y M son las dimensiones horizontal y vertical de una matriz bidimensional.

Una de las limitantes que posee este par de transformadas, es la cantidad excesiva de operaciones necesarias para encontrar la imagen de salida, siendo esa cantidad de la forma N^2 para cada fila y columna. Esa es la razón principal que justifica el diseño de un algoritmo que reduzca la cantidad de sumas y multiplicaciones complejas a un valor igual a $N \log_2 N$ para cada fila y columna. Esa es la tarea que se le ha encomendado al algoritmo denominado FFT.

Uno de los procedimientos usados para obtener la transformada de Fourier directa o inversa consiste en emplear una propiedad denominada separabilidad [5]. Su implementación se centra inicialmente en la transformación de las filas y luego en la transformación de todas las columnas o bien puede emplearse un proceso contrario al anterior. Cualquiera de los procedimientos anteriores son válidos ya que ambos producen el mismo resultado. A continuación se presenta la figura 4.3.50 en la que se describe el proceso que se indicó anteriormente.

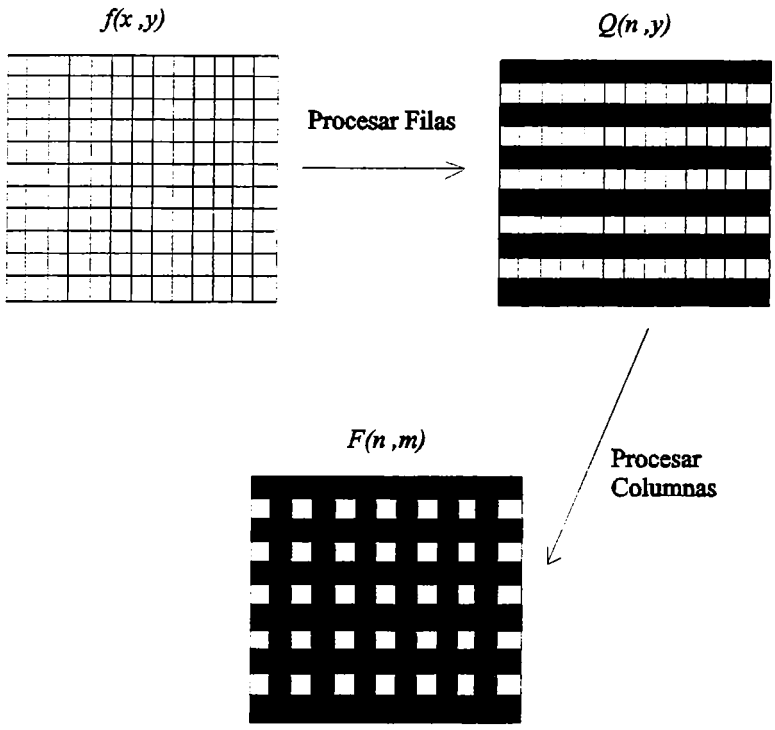
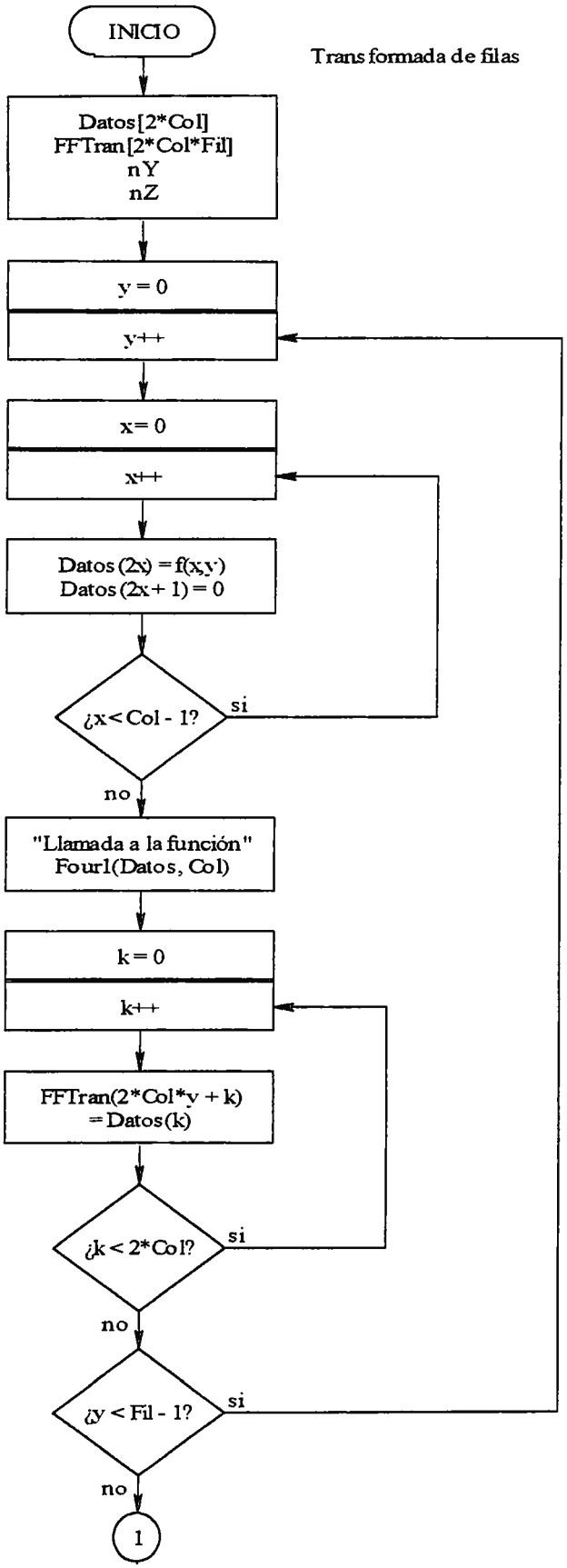


Figura 4.3.50. Empleo de la propiedad de separabilidad para obtener la FFT.

El método que se utilizado para la realización de la FFT como algoritmo es el mismo que se ha descrito anteriormente. El diagrama de flujo de la figura 4.3.51 consiste principalmente en la exploración de la imagen para obtener en primer lugar cada una de las filas y a posterior cada una de las columnas. Luego se pasa cada uno de los parámetros a la función que se encarga de ejecutar la FFT en una dimensión cuyo diagrama de flujo se presenta en la figura 4.3.52.


INICIO
Trans formada de filas
Datos[2*Col]
FFTran[2*Col*Fil]
nY
nZ
y = 0
y++
x= 0
x++
Datos (2x) = f(x,y)
Datos (2x+ 1) = 0
¿x< Col - 1?
si
no
"Llamada a la función"
Four1(Datos, Col)
k = 0
k++
FFTran(2*Col*y + k)
= Datos(k)
¿k < 2*Col?
si
no
¿y < Fil - 1?
si
no
1

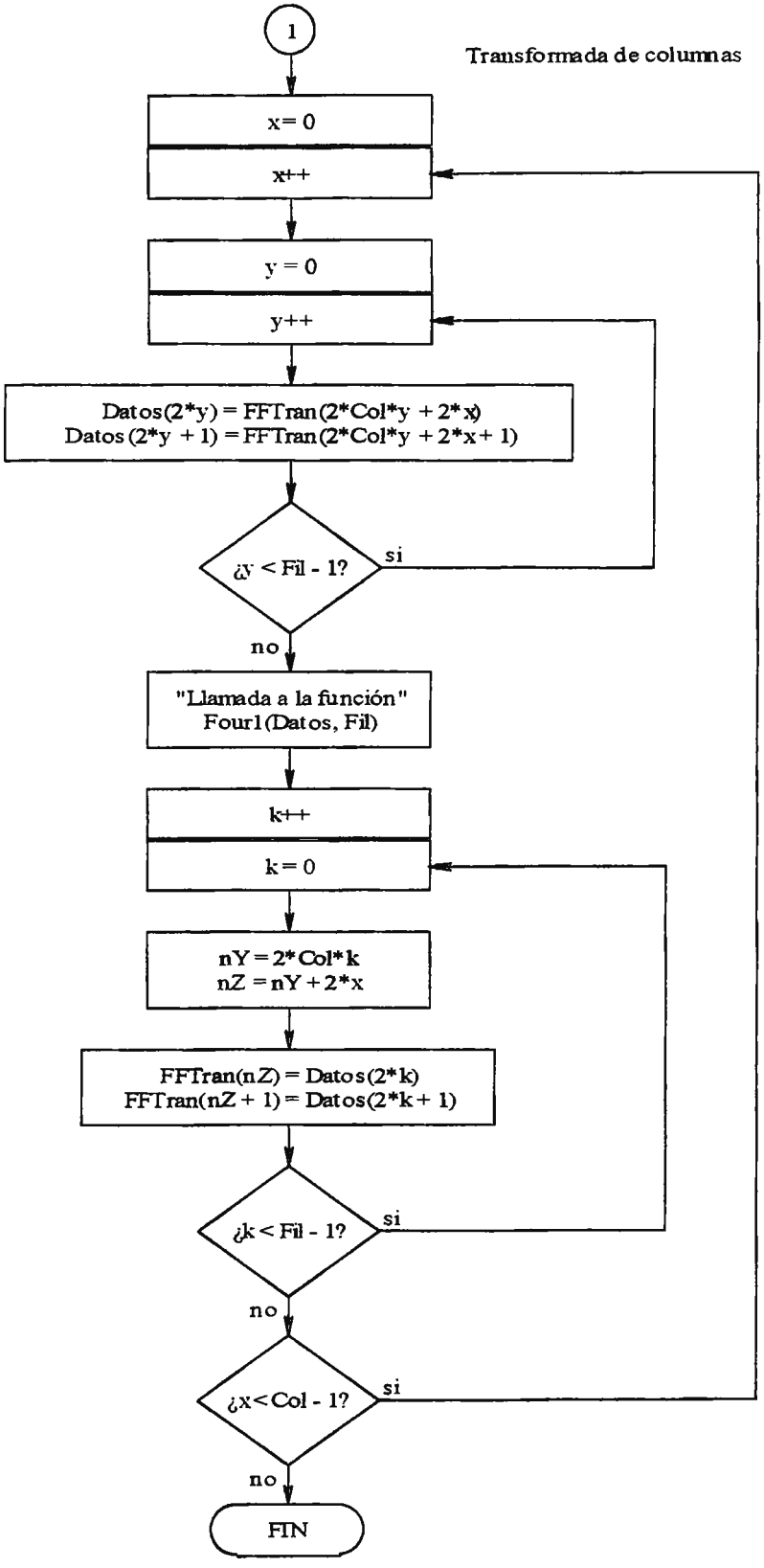
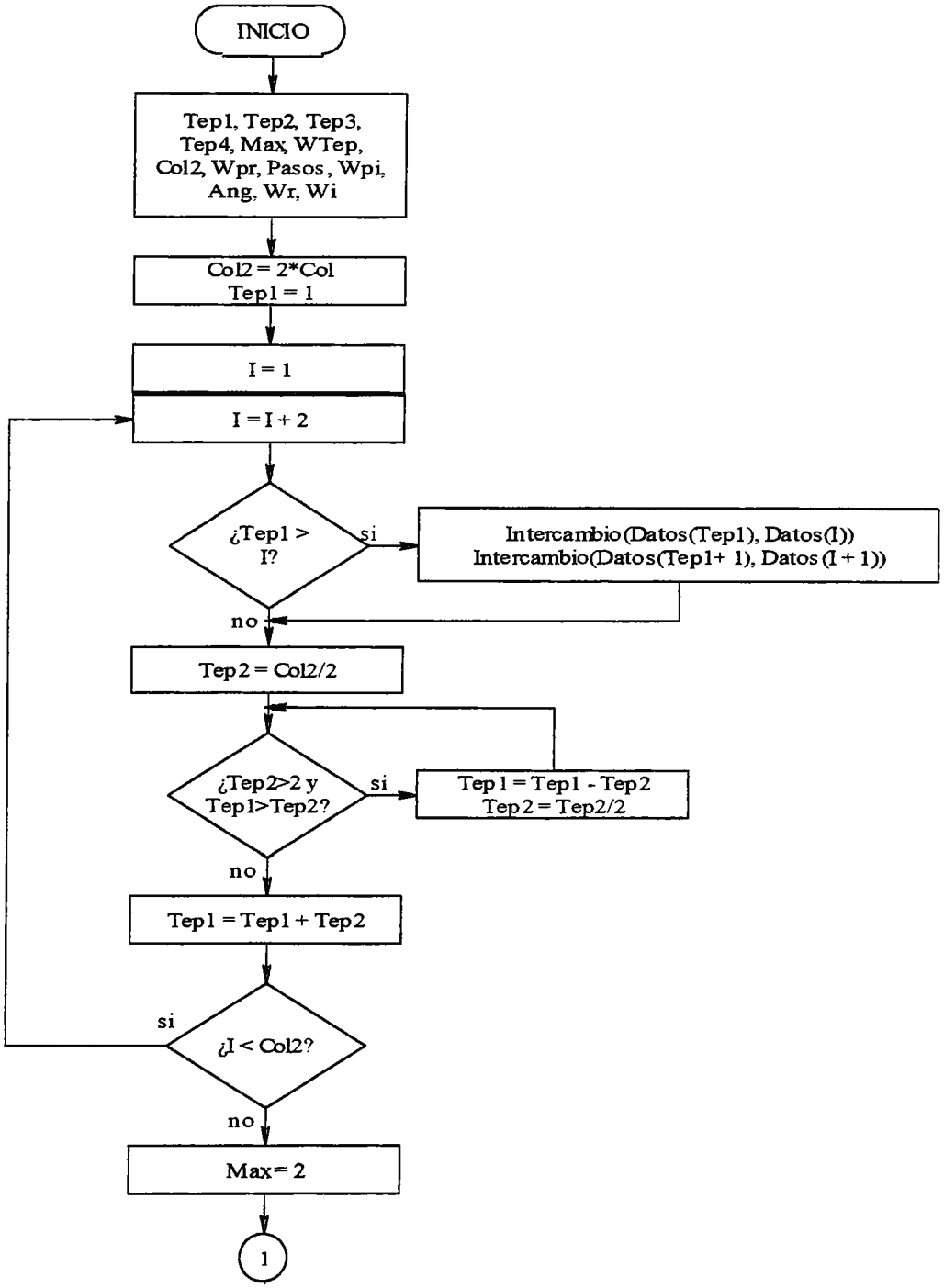


Figura 4.3.51. Diagrama de flujo genérico para la obtención de la FFT.

Es de hacer notar que en este diagrama de flujo, cada una de las variables mostradas bajo la nomenclatura *TepX* representa cada una de las variables temporales que se usan para la ejecución de la transformada en una dimensión.



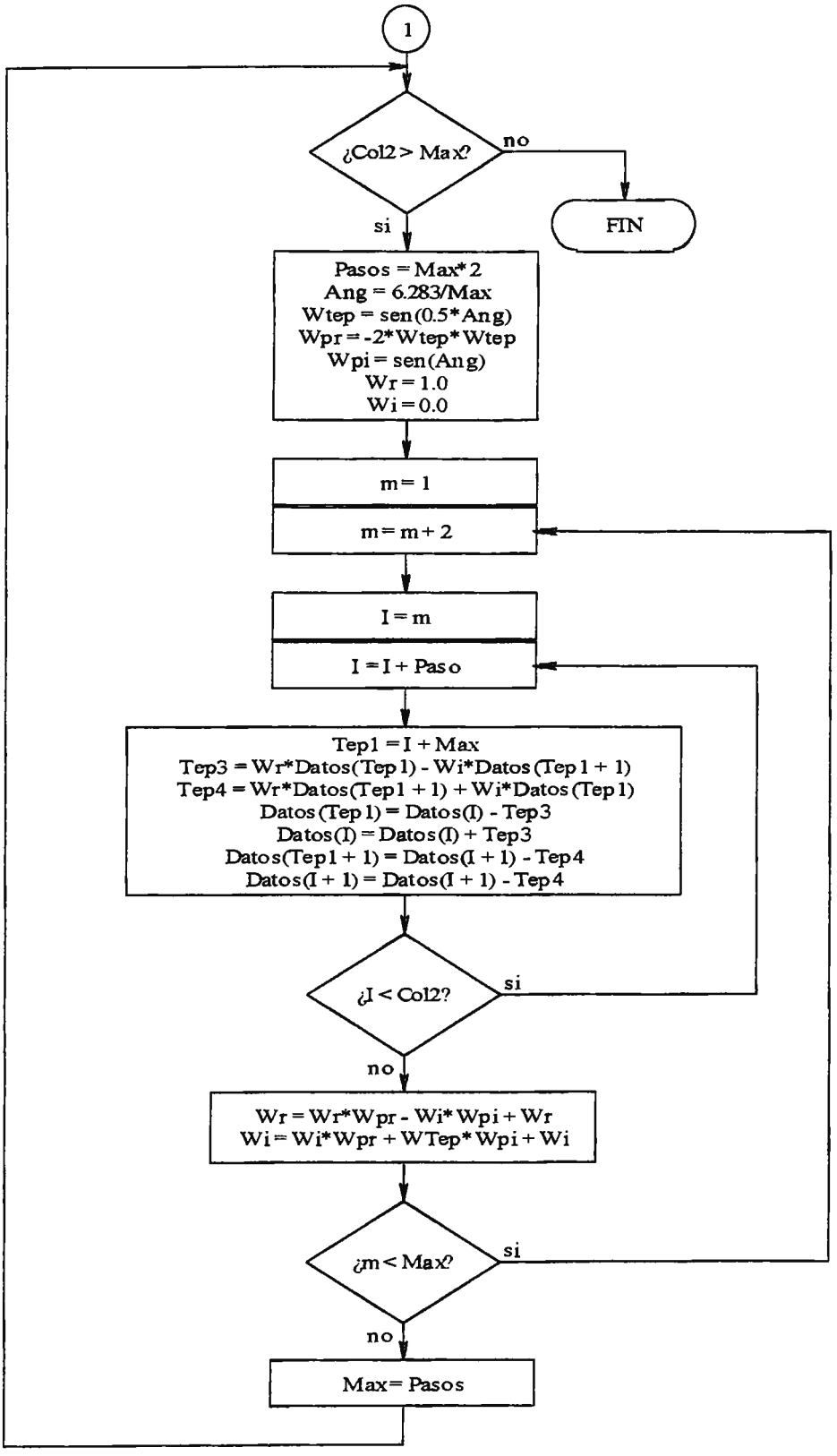


Figura 4.3.52. Diagrama de flujo para obtener la FFT en una dimensión.

CONCLUSIONES

- El desarrollo de un sistema como este requiere del análisis y estudio por separado de las partes que lo constituyen. Sólo así se puede garantizar el éxito del diseño.

- Considérase importante realizar un estudio de factibilidad económica para no caer en el derroche de esfuerzos y recursos que no podrán recuperarse de ninguna manera.

- La iluminación y el sistema óptico se encuentran íntimamente relacionados y son tan flexibles que permiten una gran cantidad de combinaciones, para lograr obtener una imagen que contenga las características relevantes que se desean estudiar.

- Se ha comprobado que las técnicas de iluminación que se han desarrollado en este trabajo no revisten un elevado grado de complejidad, con lo que se reafirma la poca existencia de bibliografía relacionada con dichos temas. Más aún siendo que programas de aplicación en procesamiento de imágenes digitales como este vienen a compensar gran parte de las deficiencias de una iluminación pobre o deficiente.

- La existencia de herramientas de programación orientadas a objetos como el Visual C++, hacen que se reduzcan los tiempos de implementación de las aplicaciones, lográndose resultados atractivos a muy corto plazo.

- Es de relevante importancia tomar en cuenta las demoras existentes en la adquisición de bibliografía y equipamiento, por el hecho de que en el país no existe ni comercio, ni productos relacionados a esta rama de la ingeniería.

- Este tema muestra el campo multidisciplinario contenido en esta rama de la ingeniería, por lo que pudo haber sido desarrollado por cualquier grupo de estudiantes pertenecientes a alguna de las ramas.

- La definición vertical y horizontal deben tenerse en cuenta dependiendo de la redundancia de información. Es decir, si la definición es mayor en el sentido horizontal, un código de barras debería orientarse en el sentido vertical.

RECOMENDACIONES

- Para operar adecuadamente el sistema es necesario contar con el conocimiento básico de los conceptos de adquisición y procesamiento de imágenes digitales.

- Es fundamental la elección de la herramienta de programación adecuada a la plataforma en que se fuere a aplicar dicho desarrollo. Esto disminuye enormemente los esfuerzos y el tiempo en la obtención de un resultado.

- Este trabajo viene a sentar el inicio de una rama de especialización en ingeniería, por lo que requiere de impulso, interés, seguimiento, divulgación, y fundamentalmente de personas que participen y se involucren en el proceso de expansión de estos conocimientos, en busca de aplicaciones sencillas e inmediatas. Un ejemplo de una aplicación sencilla es el mejoramiento de imágenes radiográficas.

- Utilizar un modelo de memoria que permita trabajar asignando memoria en forma dinámica, y que sea compatible con la plataforma Windows 95, ya que con el modelo actual limita los bloques de memoria a un tamaño máximo predefinido.

- Las exposiciones deben de ser un complemento del trabajo escrito, e ir más allá de lo que se expresa, con la finalidad de aclarar el objetivo que se persigue y sin olvidar el fundamento que lo realiza.

- El desarrollo de productos tecnológicos actuales es un beneficio que se reparte en toda la sociedad. En primera medida nos pone en contacto con productos de primera calidad realizados en otros países, lo que nos exige obtener resultados similares. Por otra parte, debemos lidiar con nuestras barreras culturales, y demostrar que podemos vencerlas obteniendo productos de calidad académica y comercial. Además, debe contarse con que lo que hacemos no es para nuestro beneficio, sino para las generaciones venideras.

- Para futuras aplicaciones se podría adecuar el formato utilizado en el manejo de las imágenes a formatos más actuales y flexibles, que además permitan la compresión de datos.

- La Universidad debe seguir aportando a través de toda su estructura ideas que permitan implementar trabajos de relevancia tecnológica y que éstos se adecuen a la realidad nacional.

- En la aplicación de la FFT es recomendable incluir una escala logarítmica en ambos ejes, de manera tal que pueda obtenerse información relevante al momento de aplicar filtrado o determinar características en frecuencia de una transformada.

- En algunas aplicaciones es necesario contar con la posibilidad de obtener una mejor apreciación de zonas específicas, por lo que incluir en un trabajo futuro la ampliación (zoom) de la imagen original se considera de relevante importancia.

BIBLIOGRAFÍA

[1] Acevedo Moran, William Edgardo; *"Sistema de Adquisición de Datos de Presión, Temperatura y Nivel para el Control Automático Manejado por Computadora"*; Tesis 88 de UDB.

[2] Andreen, Gerry; *"Robot Dessing Handbook"*; Mc Graw Hill; 1988.

[3] De La Escalera, Arturo y Armigol Moreno, José María; *"Introducción a la Visión por Computador"*; Universidad Carlos III de Madrid; 1994.

[4] Franklin, Gene y Powell, David; *"Digital Control of Dynamic Systems"*; 1980; Addison Wesley.

[5] Gonzales, Lee y Fu; Robótica: *"Control, Visión e Inteligencia"*; McGraw Hill; 1988.

[6] Gonzales, Rafael C. y Woods, Richard E.; *"Digital Image Processing"*; Addison Wesley; 1985.

[7] Gonzáles Núñez, Juan; *"El Control Numérico en las Máquinas Herramientas"*; CECSA; 1990.

[8] Hecht, Zajac; *"Óptica"*; Addison Wesley; 1986.

[9] Kruglinski, David J.; *"Programación Avanzada con Visual C++"*; McGraw Hill; 1996.

[10] Kuni, Christophe; *"Introduction to Computer and Digital Processing in Medical Imagin"*; Year Book Medical; 1988.

[11] Kuo, Benjamin C.; *"Digital Control in Systems"*; HRW; 1980.

[12] McCloy, D.; *"Robótica"*; Limusa; 1993.

[13] Millerson, Gerald; *"La Iluminación en Televisión; Instituto Oficial de Radio y Televisión"*.

[14] Mompin, José; *"Inteligencia Artificial: Conceptos, Técnicas y Aplicaciones"*; Marcombo; 1987.

[15] Muñoz, Cuauhtémoc; *"Métodos Ópticos"*; Limusa; 1981.

[16] Pitas, I.; Venetsanopoulos, A. N.; *"Nonlinear Digital Filters: Principles and Applications"*; Kluwer Academic Publishers; 1990.

[17] Pitas, Ioannis; *"Digital Image Processing Algorithms"*; Prentice Hall; 1993.

[18] Pratt, William K.; *"Digital Image Processing"*; Wiley Interscience; 1989.

[19] Raymond, Often; *"VLSI Image Processing"*; McGraw Hill; 1986.

[20] Russ, Jonh C.; *"The Image Processing Handbook"*; CRC & IEEE Press; 1995.

[21] Science Reference Series; *"Optics Source Book"*; McGraw Hill.

[22] Seely, Samuel y Poularikas, Alexander; *"Signals and Systems"*; PWS; 1985.

[23] Shammas, Namir Clement; *"Aprendiendo Visual C++ en 21 Días"*; Prentice Hall; 1994.

[24] Silva, Manuel y Roy, Armando; *"Inteligencia Artificial y Robótica Industrial"*.

[25] Usategui, Angulo; *"Robótica Práctica"*; Paraninfo; 1986.

[26] Weeks, Arthur R.; *"Fundamentals of Electronics Image Processing"*; SPIE/IEEE Image Science & Engineering; 1996.

[27] Zapata, Carlos Roberto; *"Prototipo Experimental de un Sistema de Adquisición de Datos"*; Tesis 11 de UDB.

MANUAL DE OPERACIÓN

SPID

Revisión 1.0

1998

San Salvador, El Salvador, septiembre de 1998.

ÍNDICE

Introducción

El campo del procesamiento de imágenes se ha desarrollado considerablemente en las últimas décadas con un elevado grado de utilización en innumerables aplicaciones, acompañado con los avances en tamaño, velocidad y menor costo en las computadoras y la tecnología de procesamiento de señales. El procesamiento de imágenes digitales es un amplio tema que abarca estudios de física, psicología, ingeniería electrónica, ciencias de la

computación y matemáticas. Así podemos ver que ha tomado un rol significativo en aplicaciones científicas, industriales, biomédicas, espaciales y gubernamentales.

Dentro del sistema de procesamiento de imágenes digitales es importante entender la física de la formación de la imagen capturada por los sensores y sistemas ópticos, incluyendo la percepción visual humana. Además es preciso establecer las características espaciales y temporales de campos continuos de imágenes que proveen las bases para la interrelación de las muestras en imágenes digitales. El mejoramiento de imágenes incluye tres tipos de procesos de manipulación: el realce, la restauración y la modificación geométrica de la imagen. Este banco de trabajo trata todos estos aspectos y provee una herramienta para aplicar una gran cantidad de conceptos en procesamiento de imágenes digitales y luego poder concluir en un proceso a ser incluido por ejemplo, en un ambiente productivo.

El sistema completo que se ha desarrollado está compuesto de cinco partes: una lente manual estándar de 1/3", una cámara de CCD de 1/3", encargados de realizar la operación de sensado de las imágenes, una tarjeta digitalizadora que puede operar en tiempo real, cuya finalidad es digitalizar la imagen captada y proveer un cierto grado de flexibilidad en las configuraciones, y finalmente un programa para realizar procesamiento de imágenes digitales que hace uso de una librería con funciones de control de la tarjeta, y que provee una interfaz de usuario sumamente amigable.

En este manual de operación del sistema se encontrarán todas las especificaciones técnicas de los diferentes componentes del mismo y una explicación de la forma de utilización del sistema entero con una descripción de las opciones más importantes contenidas dentro del programa de aplicación. Además se incluye un capítulo donde se detallan los proveedores con los datos completos de manera tal que pueda conseguirse un soporte a través de correo convencional, correo electrónico o telefónicamente.

1 EL SISTEMA COMPLETO: Componentes y Características

1.1 LA LENTE

La lente que se proporciona en el sistema es una lente estándar de 1/3 de pulgada de diámetro con un montaje tipo CS y longitud focal de 8 mm. En la siguiente página (cuadro y figura 1) se muestran todas las características técnicas y dimensionamiento como una figura que la muestra en su totalidad.

LENTE T0812FICS-3

para cámaras con formato 1/3 pulgada y montaje CS

Distancia focal	8mm
Máxima relación de apertura	1:1.2
Máx. formato de imagen HxVxD	4.8 x 3.6 x 6mm
Rango de operación Iris Foco	F1.2 - 16C 0.2m - Infinito
Dimensión del objeto a M.D.O.	12.5 x 9.2cm
Ángulo de visión D H V	43.5 34.7 25.9
Rango de temperatura	- 20°C ~ + 50°C
Abertura efectiva de lente Frente Atrás	25.0mm (diámetro) 8.8mm (diámetro)
Longitud focal hacia atrás	8.1mm
Longitud entre bordes	12.5mm
Montaje	Tipo CS
Tamaño de filtro	M30.5 x 0.5
Dimensiones	34.5(diámetro) x 33.0mm
Peso	37gr

M.D.O.= Mínima distancia al objeto

Cuadro 1: Características de la lente T0812FICS-3

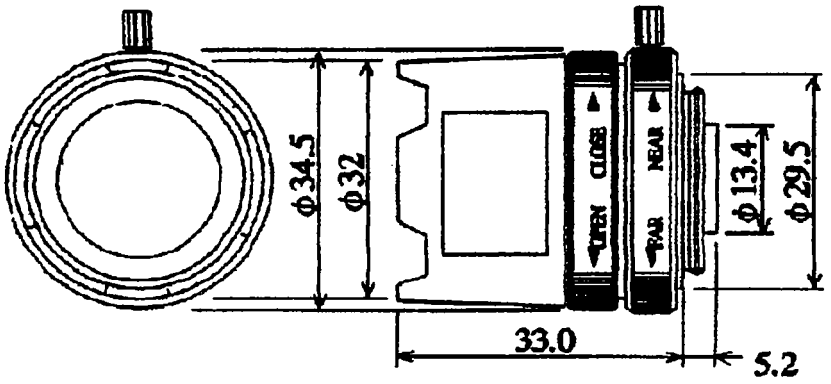


Figura 1: Dimensiones de la lente

1.2 LA CÁMARA DE INSPECCIÓN

La cámara junto con la lente son los elementos responsables de la adquisición de las imágenes, por ende su correcto funcionamiento es fundamental. La cámara de CCD T-370 fabricada por CRESCENT INTERNATIONAL de Japón, que se incluye en el sistema es suficientemente robusta y tiene características suficientes para adaptarse al procesamiento de imágenes.

1.2.1 DESCRIPCIÓN Y CARACTERÍSTICAS

Esta cámara de inspección de CCD monocromática ha sido diseñada para sistemas CCIR/EIA. Provee imágenes con una demora reducida, baja distorsión geométrica y una buena resistencia a las vibraciones. Seguidamente se detallan las características más relevantes de la cámara y que la hacen adecuada para aplicaciones en sistemas de procesamiento de imágenes.

1. Alta resolución: La cámara puede producir imágenes con una resolución de hasta 400 líneas horizontales. Posee una cantidad de elementos sensitivos que compatibiliza con ambos estándares: CCIR, 512(H) por 582(V) y EIA, 512(H) por 492(V).

2. Alta sensibilidad: con un mínimo de iluminación de 0.1 lux (F1.2 3000°K) se asegura que la unidad producirá una buena imagen.
3. Selección de GAMMA: Estas cámaras contienen un selector para GAMMA de 1 o 0.45, así se puede optar para obtener la mejor imagen.
4. Obturador electrónico automático: Este modelo ajusta automáticamente la velocidad del obturador de acuerdo con la brillantez de la escena garantizando una exposición estable.
5. Modo de sincronía de línea (line-lock): La imagen permanecerá estable en el monitor de TV operando este interruptor, aplicable en sistemas de inspección u observación.
6. Aumento de la sensibilidad: Diseñado para circunstancias especiales y proveer imágenes de buena calidad.

1.2.2 PRECAUCIONES

1. En relación a proteger la cámara, evite colocarla o utilizarla a la luz del sol directa, lluvia o polvo.
2. No toque el sensor de CCD directamente con los dedos. Si fuera necesario, utilice un paño suave humedecido en alcohol para quitar el polvo.
3. Cuando la cámara está fuera de uso, mantenga la lente debidamente sujetada en su lugar para proteger el sensor de CCD.
4. Para modelos de DC, utilice una fuente regulada de +12 volts, para los de AC, conéctelos directamente a la fuente de AC.
5. No golpee, o someta a fuertes vibraciones o deje caer la cámara, ya que dejaría de funcionar correctamente.

1.2.3 ESPECIFICACIONES

Área de imagen	5.24mm x 6.4mm
Elementos de captura	EIA: 512(H) X 492(V) CCIR: 512(H) X 582(V)
Sistema de barrido	2:1 Entrelazado EIA, V: 60Hz, H: 15750Hz CCIR, V: 50Hz, H: 15625Hz
Sistema de sincronismo	Interno/externo (VBS) DC Interno/enganchado AC

Salida de vídeo	Salida compuesta de 1 Vpp, 75 Ohms
Montaje de la lente	C o CS
Corrección de Gamma	Seleccionable 0.45 o 1
Iluminación mínima	0.1 lux (F1.2 3000°K)
Resolución	400 líneas de TV (horizontales)
Fuente de alimentación	AC 230V/110V/24V DC 12V
Consumo de energía	AC 6.0W (máx) DC 3.0W (máx)
Ambiente	-10 ~ 50°C / 30 ~ 90%RH
Dimensiones	120(Largo)x65(Ancho)x60(Alto)mm
Peso	AC 700gr (máx) DC 400gr (máx)
Control de obturador	Automático: C:1/50 o E:1/60, hasta 1/100000 seg., MES (Modo 8)
Control de Iris	Salida de vídeo/Interfaz incorporada para manejo IRIS
Control de ganancia	AGC promedio, Elevada sensibilidad
Corrección de apertura	Horizontal

La cámara posee una interfaz incorporada para manejar un iris en forma automática. Existen dos posibilidades de operar esta funcionalidad y que se describen a continuación.

1. Cuando de opera con el amplificador: seleccione EE en los interruptores del interior de la cámara a la posición AI, luego enchufe debidamente el conector del iris automático en la parte trasera del panel de la cámara.
2. Cuando se opera sin el amplificador: seleccione en los interruptores en el interior de la cámara el interruptor EE a la posición AI AMP, y a continuación conecte adecuadamente el conector del iris automático en la parte trasera de la cámara.

Definición de los pines para el IRIS

	AI	AI AMP
1	+12V	DAMP - (y)
2	NC	DAMP - (r)
3	VÍDEO	DRIVER +(wh)
4	GND	DRIVER - (g)

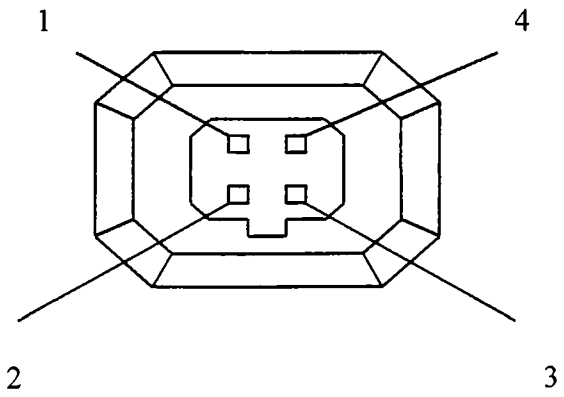


Figura 2: Definición de pines para el IRIS

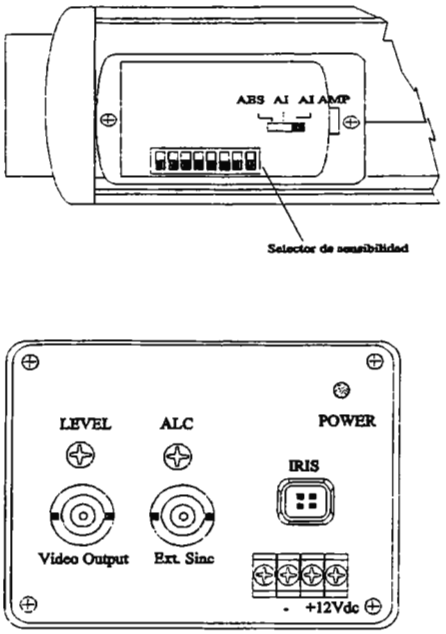


Figura 3: El panel de control y los conectores de la cámara

1.3 LA INTERFAZ DIGITALIZADORA

1.3.1 ARQUITECTURA

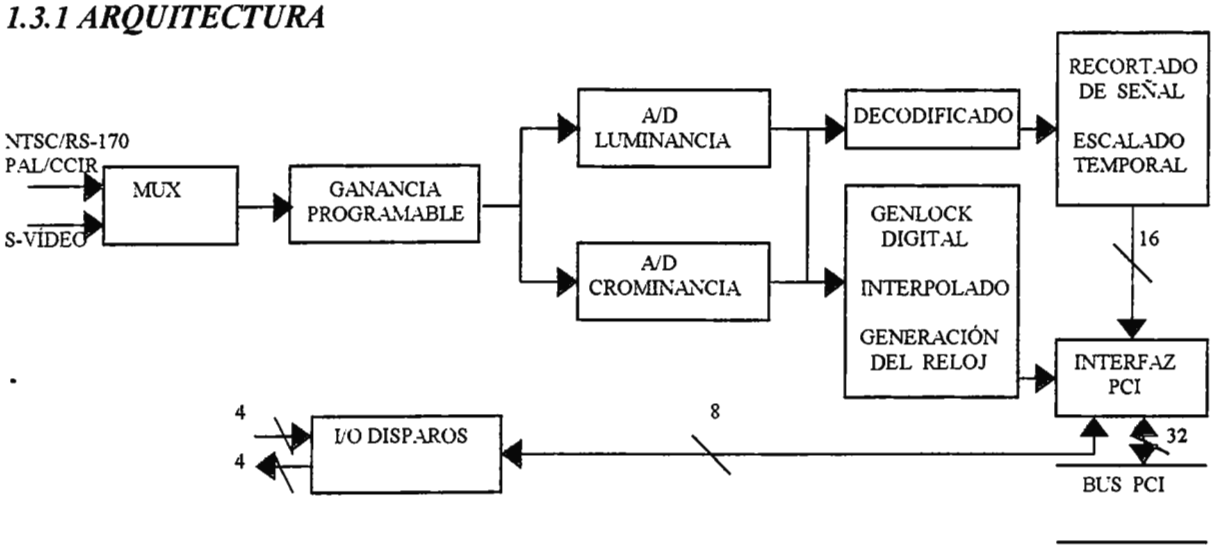


Figura 4: Arquitectura de la tarjeta de imágenes PIXCI-SV4

El multiplexor de entrada selecciona la fuente de vídeo durante el apagado del barrido vertical, y dependiendo de la fuente que provenga, de la entrada de vídeo S (mux 0), del BNC1 (mux 1), o del BNC 2 (mux2) lo pasa a la sección de Ganancia automática y fija.

Esta tarjeta está preparada para ajustar automáticamente la ganancia cuando la señal análoga de entrada es reducida, ajustando el voltaje de referencia de los conversores A/D. Los ajustes de ganancia, brillantez, saturación y contraste pueden realizarse en un rango desde 0% hasta 200%.

El conversor de luminancia A/D provee la conversión de NTSC, RS-170, CCIR, PAL y de la componente Y de Vídeo-S. Posee un prefiltro programable que reduce los efectos de aliasing. Ambos conversores A/D, de crominancia y luminancia, funcionan a una frecuencia de 28.6363 Mhz para formato de vídeo NTSC y a 35.46895 Mhz para PAL.

El decodificador separa las componentes Y/C y genera las señales diferencia de color U/V. Los formatos de vídeo de entrada estandarizados son convertidos a YCrCb en el formato 4:2:2.

El circuito digital de genlock interpola las líneas de menor y mayor longitud que las programadas a un valor fijo de pixeles por línea. Además genera el reloj de pixel para la transferencia de datos de la imagen para la interface del bus PCI. También utiliza el método de interpolación para reducir el tamaño de las imágenes hasta 1/14 del original.

Posee cuatro entradas de disparo TTL que pueden utilizarse para sincronizar eventos externos. Las señales de disparo se controlan desde el CPU de la computadora. La interfaz está preparada para operar a una velocidad de transferencia de 132 Mbytes/segundo en el modo de transferencia burst. Como bus maestro, la tarjeta PIXCI-SV4 no espera que la CPU lea la memoria de la tarjeta sino que transfiere directamente los datos al bus PCI. EN el modo burst se garantiza la transferencia de dos bytes por ciclo, por lo que si no se utiliza, la transferencia puede tomar varios ciclos del bus PCI. Se puede capturar una secuencia de imágenes a velocidad plena o reducida hacia el bus PCI, para almacenarlas en la memoria de la computadora, o pasarla a otro dispositivo en el bus PCI.

La tarjeta PIXCI-SV4 puede operar imágenes en color RGB de 24 bits, en Vídeo-S con 16 bits, o monocromáticas en 8 bits, dependiendo del adaptador de vídeo VGA. Los datos

de las imágenes pueden pasar directamente al adaptador S/VGA para reproducir vídeo en vivo.

1.3.2 ESPECIFICACIONES

Formato de vídeo

- Adquisición de vídeo color o monocromático: Vídeo-S, NTSC, PAL, RS-170, CCIR
- Mínimo voltaje de entrada: 0.5 Volts
- Pixeles de resolución: 754 x 480: Vídeo-S, NTSC, RS-170
  922 x 580: Vídeo-S, PAL, CCIR
- Profundidad de resolución: 8 bits: RS-170, CCIR
  YUV [4:2:2]: NTSC, PAL
  YCrCb: Vídeo-S
- Frecuencia de Captura/Display: 30 cps: Vídeo-S, NTSC, RS-170
  25 cps: PAL, CCIR

Requerimientos de bus

- Ranura de 32 bits, 33 Mhz
- 0.55 Amps @ 5.0 Volts
- 4.913 x 3.35 pulgadas (ranura pequeña)
- Requiere de una tarjeta con bus PCI que opere en modo burst para operar a máxima resolución en capturas de imagen en la memoria DRAM de la tarjeta madre.

Monitor DOS

- Vía VGA estándar: limitado a 4 bits (16 niveles de gris), sin captura en tiempo real
- Vía Super VGA: 8 bits, 256 niveles de gris. Monitores en color soporta 24 bits RGB o Y/C. Los adaptadores S/VGA deben ser compatibles con VESA 1.0

Monitor Windows

- La resolución del monitor va de acuerdo con el manejador VGA instalado

- Se requiere de un adaptador DCI compatible con S/VGA para mostrar vídeo en vivo

Conectores

- DIN de 4 pines hembra: entrada de Vídeo-S
- 2 conectores BNC: entradas de vídeo compuesto
- Receptáculo DB15: I/O de disparo tipo TTL

1.4 LA LIBRERÍA DE FUNCIONES DE LA INTERFAZ

Para la instalación y uso completo de estas librerías refiérase al manual de las mismas, denominado Manual de Referencia Librería XCOBJ "C". Esta librería versión 1.5 soporta todas las versiones anteriores de tarjetas de imágenes PIXCI, permitiendo un acceso y control conveniente de las facilidades de dichas tarjetas.

Dentro del paquete viene incorporado un ejemplo llamado **xcobjex1.c** que muestra muchas de las funciones de la librería. Este programa fuente puede ser compilado y ejecutarse para familiarizarse con el sistema, además de ir complementando con la lectura del libro.

Esta librería es compatible con los siguientes compiladores, versiones, modelos de memoria y ambientes:

Compilador	Modelo	Librería	Opciones en compilado	Ambiente	Módulos en tiempo corrida
Watcom C/C++ V10.0	f, 32 bits	XCOBW0DF	-4r -mf -w3 -ox	DOS4GW Profesional	ninguno
Watcom C/C++ V11.0	f, 32 bits	XCOBW1DF	-4r -mf -w3 -ox	DOS4GW Profesional	ninguno
Çualquiera de 32 bits compatible con Microsoft		XC150W95	-GB -LD	Windows 95	XC150W95.DLL EPIXXC32.VXD EPIXXC.SYS (opc.)
Cualquiera de 32 bits compatible con Microsoft		XC150WNT	-GB -LD	Windows NT	XC150WNT.DLL EPIXXCNT.SYS

Es muy conveniente que en el momento de correr la aplicación se encuentre en el mismo directorio el archivo XC150W95.DLL y las extensiones que fueren necesario. El archivo EPIXXC32.VXD debe ser copiado al directorio C:\WINDOWS\SYSTEM.

Básicamente esta librería contiene un sin número de funciones que pueden agruparse de la siguiente manera:

- Librería de interfaz de Funciones "C" Simples (SCF en inglés): provee servicios suficientes para las aplicaciones más comunes haciendo uso de una simple tarjeta de imágenes. Estas funciones han sido diseñadas para programadores casuales en "C", y no requieren estar familiarizado con estructuras y punteros. Estas funciones están distinguidas por el prefijo **pxd_**.
- Funciones no-SCF: estas funciones proveen un control más sofisticado de la tarjeta PIXCI para aplicaciones menos usuales, tal como proveer una interfaz más estructurada y orientada a objetos en todas las aplicaciones. Además estas pueden agruparse en: Funciones de soporte e inicialización, Control de Vídeo, Acceso a la memoria de imágenes y Servicios adicionales.

1.5 EL PROGRAMA DE APLICACIÓN

Este tipo de aplicación se basa en la creación de un proyecto que genera un archivo ejecutable a partir de las MFC de Windows desarrollado a partir de Visual C++. En la aplicación pueden distinguirse activos dos menús: uno que aparece por defecto dentro del marco de la aplicación, el otro se vuelve activo durante la apertura de una imagen.

El menú sin documento contiene las conocidas opciones siguientes: Archivo, Editar, Ver , Ventana y Ayuda, además se agrega la barra de estado y la de herramientas. Al abrir un documento se activa un nuevo menú, que contiene todas las opciones necesarias para completar un banco de trabajo en procesamiento de imágenes digitales.

Esta herramienta permite realizar operaciones como: generación de patrones de ruido, filtrado espacial, FFT, manipulación de contraste, modificación de histograma, ecualización, modificaciones geométricas, morfología de imágenes (dilatación, erosión), detección de bordes y texturas.

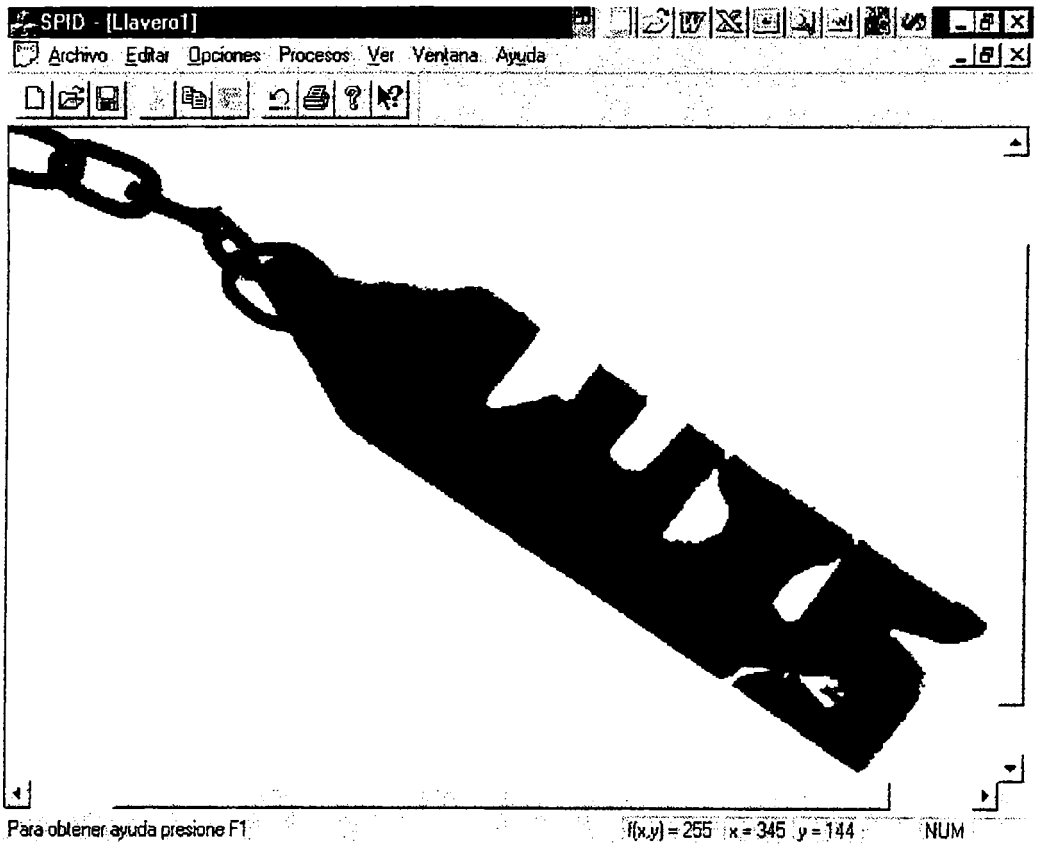


Figura 5. El marco de la aplicación SPID y un documento activo

2. USO DEL SISTEMA

2.1 LA ALIMENTACIÓN DEL SISTEMA

2.1.1 DE LA CÁMARA DE INSPECCIÓN

1. Verifique la polaridad adecuada de la alimentación y el valor del voltaje a aplicarle.
2. Reconozca los terminales en la cámara en el panel trasero.
3. Estando apagada la fuente de alimentación conecte los pines adecuados al panel trasero de la cámara.
4. Enchufe la fuente de alimentación a la red de 110 VAC.

5. En general el foco viene calibrado de fábrica. Pero dependiendo de la aplicación, podría ser necesario un reajuste del foco. En ese caso, gire el foco en la lente a ∞ y tome una imagen a 20m de distancia o más. Libere el montaje de la lente desde el tornillo que lo sujeta y gírelos hasta obtener la imagen en foco. Ahora, ajuste el tornillo con la herramienta hexagonal.
6. Si la aplicación contemplara el movimiento de objetos a una considerable velocidad, debe utilizarse una mayor velocidad de obturación. Refiérase al interior del panel rebatible en el lateral de la cámara para seleccionar la velocidad más adecuada.

2.1.2 DE LA INTERFAZ PIXCI-SV4

1. La interfaz utiliza alimentación que la toma directamente desde el bus PCI.
2. Verifique que la alimentación de 110 VAC del computador se encuentre debidamente aterrizada.
3. Proceda a interconectar los diferentes elementos del sistema como lo muestra la siguiente figura.

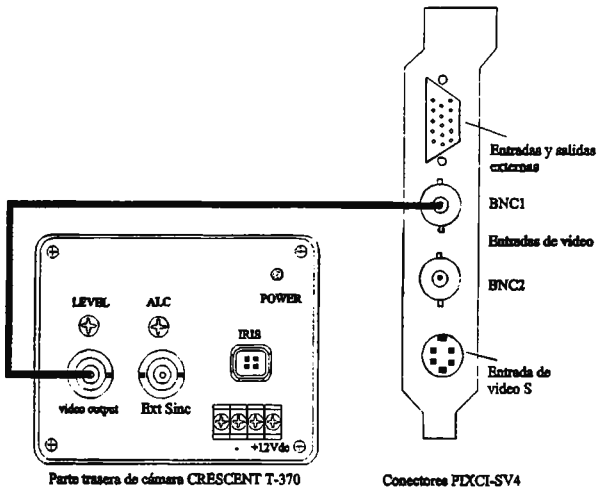


Figura 6: Interconexión del sistema

2.2 ENCENDIDO INICIAL

1. Verifique que el lente se encuentre instalado en la cámara.

2. Verifique que la computadora se encuentre enchufada a la red de 110VAC.
3. Encienda la cámara y la computadora, y espere a que inicie el equipo.
4. Busque en la lista de programas a SPID y selecciónelo para arrancarlo.

2.3 UTILIDADES DEL PROGRAMA DE APLICACIÓN

2.3.1 MENÚ ARCHIVO

La sección archivo del menú principal contiene en su interior funciones que son comunes para la mayoría de los programas de basados en Windows. Estas funciones permiten abrir, cerrar, guardar un archivo, preparar página, configurar impresora, presentación preliminar, imprimir y obtener información.

En la opción obtener información se presenta el promedio, la desviación estándar, la cantidad de bitas por pixel, el ancho y alto en pixeles y el tamaño de la imagen. Por último puede verse el camino donde se encuentra guardada dicha imagen.

2.3.2 MENÚ EDICIÓN

En este menú, se encuentran opciones que se encargan de modificar la imagen para que posteriormente se compruebe un proceso en particular. A partir de éste se tiene una alternativa para agregar un tipo de ruido a la imagen en cuestión y de esa manera poder comprobar el resultado del filtrado espacial que se ubica en el menú procesos.

La función invertir convierte los niveles oscuros de la imagen de entrada en niveles claros y viceversa. Algunas veces a la inversa de la imagen se le conoce como *negativo de la imagen*. En la opción de agregar ruido aparece un cuadro de diálogo que permite seleccionar el tipo de ruido a agregar a la imagen y para el caso del gausiano y exponencial negativo se puede seleccionar el promedio y la desviación estándar. La rotación de imagen permite realizar operaciones con la geometría de la misma, obteniéndose resultados hacia derecha, izquierda a un ángulo específico menor que 90 grados. Además se pueden hacer reflejos verticales y horizontales.

2.3.3 MENÚ OPCIONES

Las opciones que se encuentran ubicadas dentro de este menú, son opciones que permiten la creación de una nueva imagen mediante la captura producida por la tarjeta o en todos los casos la preparación de una imagen activa en el banco de trabajo, para que posteriormente se le aplique un proceso en particular.

El control de interfaz permite realizar capturas de imágenes, mostrar vídeo en vivo, modificar la resolución dela imagen y seleccionar la ruta donde se guardará dicha imagen. El falso color nos brinda la opción de poder colorear con tres colores, uno por vez, un determinado nivel de gris de la escala o un rango de niveles de gris. El histograma es una de las principales fuente de información, respecto a la distribución de los distintos niveles de gris que componen la imagen. Por último la opción umbralizar permite elegir entre hacerlo a un nivel o dos, y para ello simplemente se coloca el nivel de gris al cual se desea hacerlo.

2.3.4 MENÚ PROCESOS

Se le ha dado este nombre ya que en él se han nucleado toda las funciones que están relacionadas con el procesamiento de imágenes propiamente dicho. Vamos a encontrar opciones que permiten alisar y afinar imágenes, filtrar en el dominio de la frecuencia o del espacio, y en forma personalizada seleccionando los valores de la matriz de convolución que se utilizará, detectar bordes por Sobel, Roberts, Prewitt, Frei y Chen, y otros. Además realizar morfología de imágenes binarias o en escala de grises, realizar operaciones aritméticas y lógicas sobre una imagen o entre dos imágenes, obtener la transformada rápida de Fourier directa o inversa, ecualizar y balancear imágenes.

.

2.3.5 MENÚ VER

Este permite activar la barra de estado, la de herramientas y realizar una réplica exacta de la imagen activa, para utilizarla posteriormente.

2.3.6 MENÚ VENTANA

Podrá seleccionar tareas que son comunes en la mayoría de las aplicaciones de Windows, organizar iconos y mosaicos, abrir una ventana nueva, y visualizar las ventana s que se encuentran abiertas y cual está activa.

2.3.7 MENÚ AYUDA

Presenta un acceso a una ayuda inmediata que le permitirá conocer el procedimiento, proceso o función que desea llevar a cabo.

3. MANTENIMIENTO

El equipo no requiere de un mantenimiento riguroso, simplemente es importante que se tengan en cuenta los puntos mencionados en el apartado 1.2.2 Precauciones de la cámara de inspección. En relación a la lente, esta puede ser retirada para limpiarla con un paño humedecido en alcohol por ambos lados garantizando una limpieza adecuada, retirando el polvo que pudiere haber recogido.

Siempre mantenga los niveles de temperatura y humedad ambiente entre los valores permitidos mostrados en las especificaciones de los dispositivos. No los moje abundantemente en sus carcazas, pero puede humedecerlas para limpiarlas, sin olvidar antes desconectarlos de la alimentación eléctrica.

4. DONDE BUSCAR AYUDA

Fabricantes de la cámara de inspección
CRESCENT INTERNATIONAL LTD.
1460-52 NAGAE, HAYAMA, KANAGAWA 240-01 JAPÓN
TEL: +81-468-77-1155

FAX: +81-468-76-3509

E-mail: sales@crescentint.com

Web site: www.crescentint.com

Fabricantes de la tarjeta de adquisición de imágenes

EPIX, Incorporated

381 Lexington Drive

Buffalo Grove, IL 60089 USA

Tel: +1-847-465-1818

Fax: +1-847-465-1919

E-mail: epix@epixinc.com

Web site: www.epixinc.com

Distribuidor de la lente COMPUTAR

Rock House Products International

P.O. Box 4242

Middletown, New York 10941 USA

Tel: +1-914-692-4077

Fax: +1-914-692-4299

E-mail: sales@Rock2000.com

Web site: www.Rock2000.com

Proyecto de graduación: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE PROCESAMIENTO DE IMÁGENES DIGITALES.

Disponible en la biblioteca de la Universidad Don Bosco (El Salvador).

PIXCI-SV4

Description

EPIX develops, manufactures, and sells image acquisition and processing hardware and software products PC bus compatible computers. The products are unique in their ability to acquire images from a wide variety of nonstandard video sources, in addition to standard formats such as RS-170 and CCIR. Devices that produce nonstandard video formats include line scan cameras, high-resolution cameras, or high frame-rate area scan cameras, and medical imaging equipment

Keywords

Images, Frames, Videos, Captures, Grabbers, Imaging, Acquisition, Processing, Hardware, Software, EPIX, Epix, NTSC, PAL, CCIR, RS-170, Digital, Camera, Interfaces, Line, Area, Scan, PCI, ISA, Cards

Imaging Board for the PCI Bus
PIXCI-SV4 Card

INNOVATIONS

Color and Monochrome Video
Formats Supported: S-Video, NTSC, RS-170, CCIR, PAL
PCI Bus Master
Real-Time Transfer to PCI Bus
Crop, Scale, and View in a Window
Automatic Format Detection
4 Input Triggers and 4 Output Triggers
Programmable Hue, Brightness, Saturation, and Contrast
Plug 'N Play Operation
Extensive Software

APPLICATION

Automated Inspection, Motion Analysis, Microscopy, Medical Imaging, Robotics, Laser Beam Analysis, Object Tracking, Multimedia, Print Quality Inspection

The PIXCI-SV4 imaging board, for the PCI bus, is designed to take advantage of the power of the host computer. Applications which were once restricted by limited memory or processing power can now be easily accomplished with the PIXCI-SV4 board and a compatible PCI computer.

ACQUISITION

A unique digital genlock circuit ensures precise synchronization of every image. The PIXCI-SV4 board automatically recognizes unstable signals (for example, from a VCR) and adapts its locking mechanism to accommodate the source.

A multiplexer allows software selection of either a composite or an S-Video source. Programmable automatic gain, hue, brightness, saturation, and contrast adjustments condition the video signal.

Image sequences may be captured at full or reduced frame rates, onto the PCI bus, for storage in the host computer's memory, or passed to other devices on the PCI bus such as disk controllers or VGA adapters.

SCALING AND CROPPING

The window of video to be captured may be cropped in single pixel increments, then scaled in ratios from 1:1 to 1:14, down to as few as 4 pixels by 1 line of image data. Horizontal and vertical scaling is performed in real-time by interpolation, providing an accurate representation of the original image.

DISPLAY

Depending on the VGA adapter, 24 bit RGB color images or 8 bit monochrome images may be displayed. The full, scaled, or cropped image may be placed anywhere on the VGA screen.

Luminance or monochrome image data can be passed directly to the VGA for live video-in-a-window display. Color images are normally stored in the host computer's memory, converted into RGB data, then displayed on the VGA adapter. With a fast processor, fast PCI bus, and fast VGA adapter, live color image data may be displayed. If the VGA adapter can accept Y/C color pixels, then images can be sent directly to the VGA adapter across the PCI bus.

TRANSFER RATES

The PIXCI-SV4 board is designed to use the 132 Mbytes/sec burst mode transfer rate of the PCI bus. As a bus master, the PIXCI-SV4 board sends image data to the PCI bus; it does not wait for the computer's CPU to read images from the board into PC memory.

I/O CONTROL

Four input and four output TTL trigger signals are available or synchronization with external events.

SOFTWARE

Extensive software is available as a menu driven package for DOS or Windows 95, Windows NT, or as programmer libraries. This software provides capturing of images and image sequences, triggered capture, scaling, cropping, image processing, measurement, analysis, display, archiving, and graphing.

MUX

The multiplexer selects the video source for the Programmable Gain from either the S-Video input connector or from the composite video source on the BNC connector. The multiplexer may be switched during vertical blanking.

Programmable Gain - Compensates for reduced amplitude in the analog signal input.
Gain can be programmed from 0% to more than 200%.
Luminance A/D - Provides analog to digital conversion of NTSC, RS-170, CCIR, PAL, and the luminance (Y) component of S-Video sources.
Chrominance (Color) A/D - Provides analog to digital conversion of the color (C) component of S-Video.
Decoder - Separates the Y/C components. Generates the U/V color difference signals.
Digital Genlock - Automatic synchronization circuitry for precise digitization. Accommodates video sources which have variable periods, such as video tape recorders.
Generates the pixel clock for transferring image data to the PCI bus interface.
Scaling, Cropping - Interpolation is used to scale images to 1/14 of their original size.
Temporal Scaling - Image sequences may be captured at full or reduced frame rates.
Trigger I/O - Four input and four output TTL triggers can be used for synchronization with external events. The trigger signals are controlled by the host CPU.

SOFTWARE

XCIPLITE - A menu driven program for DOS or Windows, providing capture, display, examination and graphing of image data, and saving of images and graphs to file.

XCIP - A powerful, interactive, menu driven package for DOS, Win-95, or Win-NT. XCIP has an extensive set of image processing functions, and is easily enhanced with user-developed custom menus and scripts. Features include image sequence capture, triggered capture, display, processing, printing, archiving, analysis, and calibrated measurement. Color and monochrome processing is supported. (Optional)

XCOBJ - Programmer's libraries providing easy board access and control, such as video format and resolution selection, image capture, and image memory access. Supports Watcom (32 bit) compiler with Tenberry DOS Extender, Win-95 DLL, or Win-NT DLL. (Optional)

XCOBJIPL - All of the XCOBJ features plus an image processing library. Provides a wealth of ready-to-run functions for image enhancement, analysis, graphics, measurement, load/save, and printing. Supports Watcom (32 bit) compiler with Tenberry DOS Extender, Win-95 DLL, or Win-NT DLL. (Optional)

CONNECTIONS:

4 Pin DIN: S-Video Input
Two BNC-Jacks: Composite Video Inputs
DB15: TTL I/O Triggers
Cables optional

VIDEO INPUT:

Color or Monochrome Video Acquisition: S-Video, RS-170, CCIR, NTSC, PAL
Resolution-Pixels: 754x480: RS-170, NTSC, S-Video 922x580: CCIR, PAL, S-Video
Resolution-Depth: 8 bit: RS-170, CCIR YUV [4:2:2]: NTSC, PAL YCrCb: S-Video
Capture/Display Rate: 30 fps: RS-170, NTSC, S-Video 25 fps: CCIR, PAL, S-Video

DATA FORMATS TO PCI BUS:

Monochrome 8 bit
YCrCb (UYUV, YUV4:2:2): 16 bit
RGB: 24 or 32 bit
Capture Rate:
30 fps: RS-170, NTSC, S-VIDEO
25 fps: CCIR, PAL, S-VIDEO

TRANSFER RATES:

Requires a PCI motherboard with burst mode to host memory data rates of at least 30 MB/S.
Contact EPIX or an EPIX distributor for suggested motherboards.

DISPLAY - Windows:

Display resolution as per installed VGA device driver.
A DCI compatible S/VGA adapter is required for real-time display.

DISPLAY - DOS:

Via Standard VGA: limited to 4 bits (16 gray levels), non real-time display.
Via Super VGA: 8 bit, 256 gray level display.
Color display via adapters supporting 24 bit RGB.
S/VGA adapters must be VESA 1.0 compatible.
Contact EPIX or an EPIX distributor for suggested S/VGA adapters.

BUS REQUIREMENTS:

32 bit PCI Bus Master slot
0.55 Amps @ +5 Volts
4.913 in. by 3.350 in.

Image Processing Products For Research and Industry
Specifications subject to change without notice.

EPIX products are made in the USA.

1 September 1997
Web: http://www.epixinc.com/epix Email:epix@epixinc.com

4MEG VIDEO Model 12
EPIX Inc.

Description

EPIX develops, manufactures, and sells image acquisition and processing hardware and software products PC bus compatible computers. The products are unique in their ability to acquire images from a wide variety of nonstandard video sources, in addition to standard formats such as RS-170 and CCIR. Devices that produce nonstandard video formats include line scan cameras, high-resolution cameras, or high frame-rate area scan cameras, and medical imaging equipment

Keywords

Images, Frames, Videos, Captures, Grabbers,
Imaging, Acquisition, Processing, Hardware, Software,
EPIX, Epix, NTSC, PAL, CCIR,
RS-170, Digital, Camera, Interfaces, Line,
Area, Scan, PCI, ISA, Cards

INNOVATIONS

4 thru 256 Megabytes of Configurable Image Memory
2 to 50 MHz Sample & Display Rate
4 to 8K Pixels/Line (31K optional)
1 Line/Image to 16,000 Lines/Image
9,000 Frames/Second at 32 x 32
Digital Signal Processor On-Board
Dual TMS320C40 Processor Option
Programmable Video Formats

APPLICATIONS

High Frame Rate Motion Analysis
Automated Inspection & Sorting
Calibrated Image Measurement
High Resolution Cine Loops
Scientific Image Analysis
Character Recognition
Document Processing
Medical Imaging

4MEG VIDEO Model 12 offers a flexible image processing platform for system integrators, OEMs, VARs, and imaging engineers. A programmable video timing generator allows image acquisition from many different video sources including line scan cameras, high resolution cameras, high frame rate cameras, medical imaging equipment, or from graphics CRTs. The image memory can be configured to accept one or many images (depending on resolution). A 12 MIPS TMS320C25 digital signal processor accelerates image processing

functions including user-programmed custom algorithms. The architecture allows for parallel processing in a multiple board configuration.

MULTI-MODE CAPTURE AND DISPLAY

4MEG VIDEO Model 12 is the latest evolutionary step in the 4MEG VIDEO series. The analog and digital input can be sampled at rates up to 50 MHz. The pixel clock can be selected from either the video camera (or other external source) or an on-board generator. The Model 12 can genlock to composite video, composite sync, or to horizontal and vertical drive signals from almost any video source and format. Alternatively, the Model 12 can generate a variety of capture or display video formats including RS-170, CCIR, RS-330, and RS-343-almost any video format can be generated (within the 50 MHz pixel clock limit). The generation of video formats, coupled with the unique ability to isolate image capture from image display, provides the ability to acquire nonstandard images (line-scan, high frame rate, high resolution) in one format and display in a different format on both standard video and VGA monitors.

An external TTL level trigger can be used to initiate image sequence capture or other functions. A TTL level external output can control strobes, sound an alarm, or initiate defective part removal.

Programmable video resolution is provided in 4 pixel increments. The Model 12 can capture an image as small as 1 line by 4 pixels, or as large as 16,000 lines by 31K pixels-limited only by memory and sampling constraints. An optional 4 input multiplexer allows selection of one of 4 analog video inputs and one of 4 pixel clocks for precise acquisition from multiple cameras. Interfaces to almost all digital cameras including those with up to 16 bits per pixel are available as user installable options.

ACCELERATED ON-BOARD PROCESSING

4MEG VIDEO Model 12 accelerates image processing with a 12 MIPS (50 MHz) Texas Instruments TMS320C25 Digital Signal Processor (DSP). The high-speed math-intensive processor can be downloaded with user-written custom programs. The DSP is programmed in assembly language for optimum performance. A 32K word on-board program memory is provided. The 4MEG VIDEO Model 12 allows execution of algorithms on the DSP while the PC performs other tasks. Multiple boards (up to 8) can be configured for maximum performance, parallel processing.

CONFIGURABLE IMAGE MEMORY

4 thru 256 megabyte image memory boards can be configured to accept a few large images or a multitude of small images. The image memory can also be used to store intermediate processing results, menus, or overlays. A bit-plane write protect feature allows text and overlays to be written over image data. Multiple boards (up to 8) can be configured for up to 1024 megabyte image sequences.

Image memory is accessed by the PC or the TMS320C25 in 64K byte blocks. A memory offset register permits the window to be located at any 16K byte boundary. Data paths for video acquisition, display, and processing are provided: only one of these three paths can be accessed at any one time. During image acquisition, the display is driven with live video.

Processing is performed during blanking intervals to maintain an uninterrupted display.

4MEG VIDEO Model 12 SOFTWARE

4MIPTOOL is a menu-driven program allowing basic capture and display with the ability to examine image buffers. (Included)

QUICK SET VIDEO programs provide predefined configurations for nonstandard cameras and nonstandard video applications. (Included)

4MIP is a powerful, interactive, menu-driven program that has an extensive set of image processing functions, and is easily enhanced with user-developed custom menus and scripted algorithms. Functions include sequential capture and display, calibrated measurement, processing, enhancement, archiving, and printing of imagery. (Optional)

4MOBJIPL "C" library provides the programmer with easy board access and control, under either DOS or Windows, eliminating the need for board-level programming. An image processing library provides a wealth of ready-to-run image enhancement, analysis, graphical, and load/save operations. (Optional)

Downloading the TMS320C25 with user-developed programs is supported by 4MIP, 4MIPTOOL, and 4MOBJIPL software.

Call for a third party software list.

VIDEO ACQUISITION/DISPLAY

2 to 50 MHz sample/display rate
1 to 16,000 lines/field or frame
4 to 8K pixels/line (31K optional)
9,000 frames/second (32 x 32)
RS-170, CCIR, RS-330, & RS-343 in/output
Acquire & display nonstandard formats
Programmable timing & resolution
Genlock to external video sources
Generate master video timing
Pseudo-color display on RGB monitor
Non-destructive overlay cursor
Trigger signal input and output
Analog or digital inputs
Optional 4 input video and pixel clock multiplexer

ON-BOARD PROCESSOR/ACCELERATOR

Texas Instruments TMS320C25 DSP
12 million instructions per second (50 MHz)
16-bit arithmetic with 32-bit accumulation
32K word program memory
Direct image memory access
Download custom software

DUAL/SINGLE COPROCESSOR OPTION

One or two TMS320C40 DSPs
4 or 8MB triple ported image memory on TMS320C40 global buses
1 or 2MB static ram on TMS320C40 local buses
Dual port ram for TMS320C40 to PC data transfer and control

IMAGE MEMORY

4 thru 256 megabyte add-on memory boards
Programmable image sizes
Bit-plane write protect (64 megabyte only)

CAMERA/VIDEO INTERFACE OPTIONS

Interface to Cidtec, Cohu, Dage-MTI, Dalsa, DVC, EG&G Reticon, Hamamatsu, Hitachi, Kodak, JC Labs, Loral Fairchild, Nec, Panasonic, PCO, Philips, Pulnix, Sierra Scientific, Sony, Texas Instruments, Xillix, Xybion and MORE! Interface to CT, MR, and ultrasound medical imaging equipment as well as VCRs. Some options require an interface card.

EPIX products are made in the USA

FPG-44
FPG Power Grabber
Vision Systems

Flexible frame grabbing and high speed image processing
for PCI bus PCs

FEATURES

Flexible frame grabbing from area & line scan cameras.
Analog input up to 20 MHz and digital input up to 32 Mbytes/Sec.
Powerful 50 MHz TMS320C44 Digital Signal Processor.
Supports simultaneous image acquisition and processing.
Up to 8 Mbytes zero wait state SRAM image memory.
Optional real time display with on board SVGA for resolutions up to 1280 x 1024 non-interlaced.
DSP image processing library for DOS, Windows 3.1/95/NT.
Fully software compatible with XPG-1000 Power Grabber.
Lightning fast PCI interface supporting bus master mode.
Optional ISA interface.
Attractive low price.

The FPG-44 Power Grabber product line is a family of imaging boards and software offering flexible frame grabbing and high speed image processing for PCI bus PCs. The FPG-44 offers the best of both worlds - powerful 50 MHz TMS320C44 DSP and lightning fast PCI interface at an amazingly attractive price. It is designed specifically for vision applications which require high speed and flexible performance at low recurring cost.

The FPG-44 will interface with any analog/digital camera or sensor at data rates up to 32 Mbytes/Sec. It supports up to 8 Mbytes of on board SRAM for fast zero wait state image processing. With on board TMS320C44 DSP and a unique memory architecture, the FPG-44 offers simultaneous image acquisition and processing. The FPG-44 includes the HSI bus interface to an optional real time display board with on board SVGA.

The FPG-44 has a PCI interface supporting bus master mode for fast data transfers to the host PC. The FPG-44 is fully software compatible with the popular XPG-1000 Power Grabber, and comes complete with a comprehensive software package including the XVL function library running on the C44 DSP.

The high processing performance and attractive price of the FPG-44 makes it ideal for a number of applications including:

Industrial Inspection & Machine Vision
Microscopic & Scientific Imaging
Document Imaging
Security and Night Vision Applications
Non-standard Sensor Acquisition and Control

ARCHITECTURE

The FPG-44 is a flexible frame grabber and image processing board for PCI and ISA bus Pcs. Modeled around the 50 MHZ TMS320C44 DSP from Texas Instruments, it supports up to 8 Mbytes of on board SRAM for fast zero wait state processing. The FPG-44 can easily interface to a wide variety of analog and digital cameras

at data rates up to 32 Mbytes/Sec. The on board C44 DSP provides over 275 MOPS of processing power for demanding vision and imaging applications.

The FPG-44 includes a High Speed Interface (HSI) bus running at over 50 Mbytes/Sec, which can support the Enhanced Display Board (EDB) for real time display. Spare C44 Comm Ports (maximum two) are also available to provide high speed bi-directional communication between multiple FPGs or XPGs or with third party C40/C44 processors.

CAMERA AND SENSOR INPUT

The FPG-44 provides flexible camera input to easily interface with virtually any type of analog/digital camera or sensor including RS170, CCIR, high resolution area or line scan cameras and multi-tap sensors. Input images can be as large as the available memory on the FPG with no limitations on image or line size. The camera acquisition circuitry is connected directly to C44 Comm Ports allowing fast and efficient data transfer to image memory on both the C44 global and local bus.

ANALOG INPUT FPG-44

Software selectable four analog channels (optional eight inputs available).
Each channel accepts single ended or differential video.
Square pixels from RS170, CCIR or any non-standard analog camera or sensor.
Composite sync on video or separate sync can be used.
Precision gain circuit to enhance video signal.
Software adjustable gain and offset control, with 12 bits precision.
Software selectable internal (programmable) pixel clock or external camera clock.
Genlock circuitry to synchronize camera input with internal pixel clock.
Programmable sub window acquisition.
Four general purpose inputs and outputs (selectable as RS422 or TTL), and additional four outputs (TTL only).
256 grey-level digitization with maximum pixel jitter of _ 4 nsec.
8 bit input LUT for real time thresholding or grey level adjustment.
Supports data rates up to 20 Mhz.

DIGITAL INPUT FPG-44

8 or 16 bit digital input as TTL or RS422.
Input data rates up to 32 Mbytes/Sec.
Supports any digital area or line scan cameras including multi-tap sensors.
Eight general purpose RS422 or TTL inputs and outputs for camera control.
Programmable sub window acquisition and de-interlacing on input data.
Optional 16 bit input LUT for real time thresholding or grey level adjustment.

IMAGE AND MEMORY

The FPG-44 offers up to 8 Mbytes of high speed zero wait state SRAM for image, program and data storage. The SRAMs are designed as SIMM modules which can be easily added to provide future upgrades. The memory can be populated on both the global and local bus of the TMS320C44 DSP. This provides concurrent image acquisition and processing, taking maximum advantage of the parallel processing architecture of the C44 DSP. Separate image display memory is available as VRAM on the optional Enhanced Display Board. Following are the main features of the FPG-44 memory:

Configurable from 512 Kbytes up to 8 Mbytes.
Use of SIMM modules for easy upgrades.
Addressable as 8, 16 or 32 bits without performance degradation.
Dynamically allocated for use as image, program and data storage.
Supports arbitrary image size and shape with no limitations.
SRAM with zero wait state on read and write cycle
Memory can be populated on both C44's global and local bus for concurrent memory access and more efficient processing.

PROCESSING

The FPG-44 offers superior image processing performance with the combination of the on board powerful TMS320C44 DSP and zero wait state SRAM. The processing power of the FPG-44 makes it an ideal platform for a wide variety of vision and imaging applications requiring ultra high speed processing. The design of the FPG makes optimum utilization of the C44 parallel processing architecture with Comm Ports connected directly to the camera acquisition circuitry, and providing concurrent image acquisition and processing.

TMS320C44 DSP

50 MHz TMS320C44 DSP offers 275 MOPS performance.
50 Mflops 32-bit floating point operations.
Four 20 Mbytes/Sec asynchronous Comm Ports.
Resident DMA controller allows concurrent image grab and processing.
Interruptible by external sync input or by PC.
In Circuit Emulator port for debugging.
Texas Instruments C compiler, assembler and debugger available.

IMAGE &DISPLAY

Images acquired on the FPG-44 can be displayed either by transferring data from FPG memory to the PC SVGA over the lightning fast PCI bus or using the optional Enhanced Display Board (EDB). The PCI interface supports bus master mode, and provides sustained data transfer rates of around 34 Mbytes/Sec. The optional EDB board has a combination of onboard accelerated SVGA and Digital Video Processor. This offers combined VGA and image data on a single monitor with resolutions up to 1280 x 1024 non-interlaced. The EDB also offers dual screen operation at resolutions up to 1280 x 1024 non-interlaced.

HIGH & SPEED & BUSES

The FPG-44 offers an optimum architecture for efficient image data transfer with the C44 Comm Ports and the HSI bus. The C44 offers four 20 Mbytes/Sec asynchronous Comm Ports for independent DMA transfers. The Comm Ports form an integral part of image acquisition and PC interface. Depending on the camera data rate and number of bits per pixel, up to two Comm Ports can be allocated for camera acquisition. The other two Comm Ports can be allocated for high speed data transfer over the PCI bus. Any Comm Ports not being used for camera acquisition or PC interface are available for interface with multiple FPGs, XPGs or external third party C40/C44 processors.

The asynchronous HSI bus running at 50 Mbytes/Sec supports the Enhanced Display Board and other third party boards. This bus connects directly to the C44 Local Memory Bus. It provides direct memory access by the DSP to peripherals on this bus.

PCI & INTERFACE

The FPG-44 offers very high speed data transfer to the host PC over the PCI bus. The PCI interface supports target and master mode, burst mode and DMA access. The PCI interface on the FPG is accomplished through C44 Comm Ports, where up to two Comm Ports can be dedicated to data transfers. This achieves a maximum sustained data rate of 34 Mbytes/Sec and burst rate of up to 132 Mbytes/Sec. The FPG-44 PCI interface conforms to the PCI Plug-n-play specifications and supports auto-configuration, allowing smooth integration of the board in the host

CONFIGURATION

The FPG-44 comes in two configurations - one for analog input and the other for digital input.
These are two separate boards with different front end circuitry. The FPG-44 supports two SIMM sockets for SRAM memory. The memory on the FPG-44 is configurable from 512 Kbytes up to 8 Mbytes of SRAM. The memory can be populated on both the global and local bus of the TMS320C44 DSP, providing concurrent image acquisition and processing. All boards include the HSI bus interface supporting the optional Enhanced Display Board.

SOFTWARE

The FPG-44 comes complete with a comprehensive software package including an extensive XVL vision library software package under DOS, Windows 3.1/95/NT running on the C44 DSP. It is the same software package which runs on all Power Grabber boards. The XVL includes over 200 DSP based functions for image acquisition, processing, display and data transfer. A comprehensive memory management system has been implemented to dynamically allocate memory for image data, program code, histogram, LUTs and other data buffers. The software fully supports different image size, bit depth and data types.

For easy prototyping and testing of new applications, the FPG also includes an interactive Windows based software environment called WiTlite to access the XVL functions. This gives the user the freedom to explore different imaging operators and see the results immediately with a demand-driven mode of operation.

To develop custom C44 based software on the FPG, extensive tools from Texas Instruments such as C compiler, assembler, linker, simulator and debugger are available. These are the same tools used for C40 development on the XPG. Several third party Windows based software packages are also available on the FPG.

For further information, contact:
Dipix Technologies Inc.
Vision Products Division
1051 Baxter Road,
Ottawa, Ontario
K2C 3P1, Canada

Tel: 613-596-4942
Fax: 613-596-4914
Toll Free: 800-724-5929
E-mail: mailto:sales@dipix.com

LPG-132 Power Grabber
Flexible frame grabbing to PC over PCI bus from area & line scan cameras

FEATURES

Flexible frame grabbing from area & line scan cameras.
Analog input up to 20 MHz and digital input up to 32 Mbytes/Sec.
Up to eight software selectable analog channels.
Very high accuracy analog to digital circuitry.
Support for digital multi-tap cameras.
Support for digital cameras up to 16 bits per pixel.
Up to 16 bit input LUT.
Software library under Windows 95 and NT.
Fully software compatible with XPG-1000 and FPG-44 Power Grabber.
Lightning fast PCI interface supporting bus master mode.
Attractive low price.

INTRODUCTION

The LPG-132 Power Grabber product line is a member of the Power Grabber family of products from DIPIX, covering applications requiring flexible frame grab directly to the PC over the PCI bus without on board processing. It is designed specifically for vision applications which require low recurring cost and high / flexible performance.
The LPG-132 Power Grabber product line is a family of imaging boards and software offering flexible frame grabbing and high speed transfers to the PC over the PCI bus. The LPG-132 will easily interface to virtually any analog or digital camera and provides lightning fast PCI transfers at an amazingly attractive price.
The LPG-132 will interface with any camera or sensor at data rates up to 32 Mbytes/Sec. It has a PCI interface supporting bus master mode for fast data transfers to the host PC. The LPG-132 is fully software compatible with the popular XPG-1000 and FPG-44 Power Grabber boards, and comes complete with a comprehensive software package.

CONFIGURATION

The LPG-132 comes in two configurations - one for analog input and the other for digital inputs. These are two separate boards with different front end circuitry.

CAMERA AND SENSOR INPUT

The LPG-132 provides flexible camera input to easily interface with virtually any type of analog / digital camera or sensor including RS170, CCIR, high resolution area or line scan cameras and multi-tap sensors. Input images can be as large as required with no limitations on image or line size. The camera acquisition circuitry is controlled directly by an intelligent on board controller, allowing fast and efficient data transfer to the PC memory over the PCI bus.

ANALOG INPUT LPG-132

Four software selectable analog channels (optional eight inputs available).
Each channel accepts single ended or differential video.

Square pixels from RS170, CCIR or any non-standard analog camera or sensor.
Composite sync on video or separate sync can be used.
Precision gain circuit to enhance video signal.
Software adjustable gain and offset control, with 12 bits precision.
Software selectable internal (programmable) pixel clock or external camera clock.
Genlock circuitry to synchronize camera input with internal pixel clock.
Programmable sub window acquisition and de-interlacing on input data.
Four general purpose inputs and outputs (selectable as RS422 or TTL), and additional four outputs (TTL only).
256 grey-level digitization with maximum pixel jitter of +/- 4 nsec.
8 bit input LUT for real-time thresholding or grey level adjustment.
Supports data rates up to 20 MHZ.

DIGITAL INPUT LPG-132

8 or 16 bit digital input as TTL or RS422.
Input data rates up to 32 Mbytes / Sec.
Supports any digital area or line scan cameras including multi-tap sensors.
Eight general purpose RS422 or TTL inputs and outputs for camera control.
Programmable sub window acquisition and de-interlacing on input data.
Optional 16 bit input LUT for real time thresholding or grey level adjustment.

IMAGE DISPLAY

Images acquired on the LPG-132 can be displayed by transferring data from the LPG to the host PCI VGA card over the PCI bus in real time. Comprehensive display software supporting Direct Draw is included with the LPG-132 to provide real time display to VGA screen under Windows 95 and NT. The PCI interface supports bus master mode, and provides sustained data transfer up to the maximum data rate of the input device.

HIGH ACCURACY ANALOG CIRCUIT

The LPG-132 incorporates very high accuracy analog and data conversion circuitry to ensure sharp image acquisition. The analog to digital circuitry on the LPG differs from other low-end frame grabbers in two significant ways. The digital Phase Lock Loop (PLL) and the Programmable Gain Amplifier (PGA). The digital PLL ensures a maximum pixel jitter of +/-4 ns regardless of the sampling rate, with the jitter occurring only at the horizontal sync position where video is not typically digitized. The digital PLL uses a high-stability crystal
oscillator which guarantees an accurate pixel to pixel sample time on each digitized line of video. Most other frame grabbers use an analog PLL which constantly adjusts the pixel clock every line in order to maintain genlock with the video source, resulting in slightly different sampling rates for every line in the digitized image.
The PGA on the LPG is an innovation which again is not generally found on other lower cost frame grabbers. The LPG contains in its video path a Programmable Gain Amplifier which can be adjusted with 12 bits of accuracy (from 0db to +30db). This, coupled with a 12 bit adjustable offset stage, can enhance the video before it is sampled by the A/D convertor. Other frame grabbers typically adjust the A/D convertor's digitization range by controlling the positive reference input (gain) and the negative reference input (offset). Controlling the analog conversion range in this manner results in a significant amount of increased noise, especially when dealing with low contrast images. By comparison, with the separate PGA on the LPG, the noise does not change

regardless of the gain or offset setting because the A/D convertor always digitizes a fixed range. This also allows the LPG to enhance low-contrast images, effectively increasing the dynamic range available from the A/D convertor.

PLUG-AND-PLAY CAMERA INTERFACE

The LPG-132 offers plug-and-play compatibility with most standard and non-standard cameras including area and line scan cameras. The camera interface includes pre-configured camera definition files, application notes, cables and demo software for most non-standard cameras including ones from DALSA, EG&G Reticon, Kodak, Pulnix, Loral Fairchild, Cohu, Amber and many others.

POWER GRABBER COMPATIBILITY

The LPG-132 is fully compatible with the more powerful XPG-1000 and FPG-44 range of Power Grabber imaging boards which offer on board DSP processing. All the Power Grabber boards have identical camera / sensor interface and run the same XVL software package. Any application developed on the LPG-132 will run on the XPG-1000 or FPG-44 without any change. This allows customers to easily migrate up from the LPG-132 to much higher performance imaging boards as their application gets more demanding.

PCI INTERFACE

The LPG-132 offers very high speed data transfer to the host PC over the PCI bus. The PCI interface supports target and master mode, burst mode and DMA access. The PCI interface on the LPG is accomplished through on board parallel DMA channels under the control of an intelligent controller. This achieves a maximum sustained data rate of over 25 Mbytes/Sec and burst rates of up to 132 Mbytes/Sec. The LPG-132 PCI interface conforms to the PCI Plug-n-Play specifications and supports auto-configuration, allowing smooth integration of the board in the host PC.

SOFTWARE

The LPG-132 comes complete with a comprehensive software package for flexible image grabbing, and transfer to the host PC memory. The software provides easy configuration for any analog or digital camera type. The software includes Windows 95 and NT DLLs for image grab, transfer to host memory and SVGA screen. The software package also includes several demo and sample programs in source code form.

APPLICATIONS
The LPG-132 is suitable for many applications which require flexible frame grab from any area or line scan camera and fast transfer to host PC. These include:
Machine vision & industrial inspection.
Medical & Scientific imaging.
IR and thermal imaging.
Document imaging.
Non-standard sensor acquisition and control.

Dipix Technologies Inc.
Vision Products Division
1051 Baxter Road,
Ottawa, Ontario
K2C 3P1, Canada

Tel: 613-596-4942 Fax: 613-596-4914 Toll Free: 800-724-5929
E-mail: sales@dipix.com
XPG-1000 Power Grabber

Flexible frame grabbing up to 48 Mhz powerful TMS320C40 DSP and up to 256 Mbytes image memory for PCI and ISA bus PCs

KEY FEATURES

Flexible frame grabbing at up to 48 MBytes/Sec
Multiple analog and digital camera modules
A 50 MHz TMS320C40 Digital Signal Processor
Simultaneous image acquisition and processing
Unique combination of DRAM and SRAM memory
Up to 256 MBytes DRAM & 256 Kbyte SRAM
Power Processing Modules for frame rate processing
Real time display with SVGA at 1280 X 1024 non-interlaced
Comprehensive DSP Image processing library for DOS and MS Windows
PCI & ISA bus interface

The XPG-1000 Power Grabber product line is a family of imaging boards and software offering flexible frame grabbing and high speed image processing for PCI and ISA bus PCs. The XPG-1000 will interface with any sensor or camera at data rates up to 48 MBytes/Sec, and includes the powerful TMS320C40 Digital Signal Processor for high speed image processing. The XPG-1000 supports up to 256 MBytes of image memory and an optional real-time display board with on board SVGA for display resolutions up to 1280 X 1024 non-interlaced.

The XPG-1000 incorporates a modular architecture offering a range of functionality for various applications. It includes programmable analog and digital camera modules at data rates up to 48 MBytes/Sec. In addition to standard RS170 and CCIR cameras, most non-standard cameras such as area and line scan cameras and multi-tap sensors are supported. The on-board TMS320C40 DSP along with a unique combination of DRAM and SRAM memory offers zero wait state high speed processing. Optional Power Processing Modules (PPM) provide super accelerated processing for specific imaging functions.

The high processing performance and the flexibility of the XPG-1000 makes it ideal for a number of applications including:

Industrial Inspection, Machine Vision, Microscopic Imaging, Scientific and Document Imaging, Security and Night Vision Applications, Non-standard Sensor Acquisition and Control.

XPG-1000 ARCHITECTURE

The XPG-1000 is a modular PCI & ISA bus image processing platform for vision applications. Modeled around the 50 MHz TMS320C40 DSP from Texas Instruments, it supports up to 256 MBytes of flexible image memory and 512 KBytes of zero-wait-state cache memory. The versatile XPG Powerbus based on up to three C40 Comm Ports, supports multiple analog and digital camera modules that interface to any camera or sensor at data rates up to 48 MBytes/Sec. In addition to the 275 MOPS processing power of the onboard C40 DSP, an optional Power Processing Module (PPM) provides super accelerated processing for specific imaging functions.

A High Speed Interface (HSI) bus running at over 50 MBytes/Sec supports the optional Enhanced Display Board (EDB) for real time display. The EDB includes onboard SVGA and offers non-interlaced display resolutions to 1280 X 1024 in both single and dual monitor configurations. Spare C40 Comm Ports are also available to provide high speed bi-directional communication between multiple XPGs or with third party C40 processor boards.
The XPG-1000 comes complete with a comprehensive XVL Function Library software package running on the C40 for rapid application development. In addition, support tools from Texas Instruments such as C compiler, assembler, debugger and simulator and third party Windows software packages are available.

CAMERA AND SENSOR INPUT

Camera acquisition modules connect various types of cameras and sensors to the XPG platform. These are piggy-back modules which connect directly to the XPG main board, and together occupy a single PC slot. The XPG supports multiple camera modules to simultaneously interface with many cameras or multi-tap sensors. The camera acquisition modules are designed to easily interface with virtually any type of analog / digital camera or sensor including RS170, CCIR, high resolution area or line scan cameras and multi-tap sensors. Input images can be as large as the available memory on the XPG with no limitation on image or line size. The
camera acquisition modules are connected directly to multiple C40 Comm Ports allowing fast and efficient data
transfer to image memory on both the global and local bus. New camera modules will be developed in the future to allow users to take advantage of rapidly improving sensor technology.

ANALOG CAMERA MODULE

Software selectable four analog channels
Each channel accepts single ended or differential video
Square pixels from RS170, CCIR or any non-standard analog camera or sensor
Composite sync on video or separate sync can be used
DC restoration circuit available to enhance video signal
Software adjustable gain and offset control, with 12 bits precision
Software selectable internal (programmable) pixel clock or external camera clock
Genlock circuitry to synchronize camera input with internal pixel clock
Programmable sub window acquisition
Sixteen general purpose RS422 or TTL inputs and outputs for camera control.
256 grey-level digitization with maximum genlock pixel jitter of _ 4 nsec.
8 bit input LUT for real time thresholding or grey level adjustment.
Multiple analog camera modules supported to simultaneously interface with several cameras.
Standard module supports data rates up to 20 Mhz with optional modules available for higher data rates.

DIGITAL CAMERA MODULE

8, 16 or 32 bit digital input as TTL or RS422.
Input data rates up to 48 MBytes/Sec.
Supports any digital area or line scan cameras including multi-tap sensors.

Sixteen general RS422 or TTL inputs and outputs for camera control.
Programmable sub window acquisition and de-interlacing on input data.
8 or 16 bit input LUT for real time thresholding or grey level adjustment.
Multiple modules supported for simultaneous interface with several cameras.

IMAGE MEMORY

The XPG-1000 offers a unique combination of DRAM and SRAM memory to optimize image acquisition and processing. In keeping with the modular architecture, the image memory is designed as SIMM modules which can be easily added to provide future upgrades. The optional zero-wait-state SRAM memory is an add-on module which piggybacks to the main board. All XPG memories can be populated on both the global and local bus of the TMS320C40. This provides concurrent image acquisition and processing, taking maximum advantage of the parallel processing architecture of the C40. Separate image display memory is available as VRAM on the optional Enhanced Display Board.

DRAM MEMORY

Configurable from 4 MBytes up to 256 MBytes.
Use of SIMM modules for easy upgrades.
Addressable as 8, 16 or 32 bits without performance degradation.
Dynamically allocated for use as image, program and data storage.
Supports arbitrary image size and shape with no limitations.
Access for same page requires 1 wait state on read and 0 wait state on write cycle
Can be divided equally between C40's global and local bus for concurrent memory access and more efficient processing.

SRAM MEMORY

Optional SRAM module up to 256 Kbytes
SRAM memory provides for zero-wait-state processing
Can be divided equally between C40's global and local bus for concurrent memory access and more efficient processing

PROCESSING

The XPG-1000 offers superior image processing performance with a combination of the powerful TMS320C40 DSP for flexible image processing and the optional Power Processing Module (PPM) for super accelerated processing. The processing power of the XPG makes it ideal for a wide variety of vision and imaging applications requiring ultra high speed processing. The design of the XPG makes optimum utilization of the C40 parallel processing architecture with the Comm Ports connected directly to the camera acquisition modules, and providing concurrent image acquisition and processing.

TMS320C40 DSP

50 MHz TMS320C40 DSP offers 275 MOPS performance
50 MFlops 32-bit floating point operations
Six 20 MByte/sec asynchronous Comm Ports

Resident DMA controller allows concurrent image grab and processing
Interruptible by external sync input or by PC
JTAG diagnostic support and In Circuit Emulator port for debugging
Texas Instruments C compiler, assembler, linker and debugger available

POWER PROCESSING MODULES

Add-on processing modules for super accelerated processing
Interfaces directly to the 48 MByte/sec Powerbus
Processes both incoming camera images and stored images on XPG memory
Histogram, Convolution, Arithmetic processing
Support future application specific custom processing modules

IMAGE DISPLAY

Images acquired on the XPG-1000 can be displayed either by transferring data from XPG memory to the PC SVGA over the PCI and ISA bus or by using the optional Enhanced Display Board (EDB) for real time display. The EDB has a combination of onboard accelerated SVGA and Digital Video Processor. This offers combined VGA and image data on a single monitor with resolutions of up to 1280 X 1024 non-interlaced. The EDB also offers dual screen operation at resolutions up to 1280 X 1024 non-interlaced.

ENHANCED DISPLAY BOARD

Live video display of acquired images from the XPG
Display resolutions from 640 X 480 to 1280 X 1024 non-interlaced
All resolutions supported in single or dual screen modes
Onboard accelerated Super VGA chip set
On board Digital Video Processor<li>Up to 8 MByte image / overlay memory and 2 Mbyte SVGA memory
16 to 8 bit input image LUT for cameras with up to 16 bit pixels
Hardware pan, scroll, zoom
Microsoft Windows compatible

HIGH SPEED BUSES

The XPG-1000 offers an optimum architecture for efficient image data transfer with the XPG Powerbus and the HSI bus. The XPG Powerbus allows synchronous data transfer from camera acquisition and Power Processing modules at data rates up to 48 Mbytes/sec. The asynchronous HSI bus running at over 50 MBytes/sec supports the Enhanced Display Board and other third party boards. The C40 Comm Ports form an integral part of the XPG Powerbus. Of the six C40 Comm Ports, up to three are available for camera acquisition depending on the camera data rate and number of bits per pixel, while one Comm Port is dedicated for the PC interface. All the other Comm Ports including the ones not being used for camera acquisition are available for interface with multiple XPGs or external C40 processors. Both buses are well documented to allow third party vendors to interface custom boards to the XPG.

XPG POWERBUS

Non proprietary interface running at up to 48 MBytes/sec.
Based on up to three C40 Comm Ports for high speed transfer to image memory
Supports up to three camera acquisition modules
Supports Power Processing Module

XPG HSI BUS

High speed interface bus running at over 50 MBytes/sec.
Connects to the C40 Local Memory Bus
Provides direct memory access by the DSP to peripherals on this bus
Supports the Enhanced Display Board for real time display

PC INTERFACE

The base XPG-1000 occupies a single PC slot and supports PCI & ISA bus interface. The PC interface is accomplished through dedicated C40 Comm Ports, and is entirely I/O based thus eliminating any conflicts with other boards in the PC address space.

The PCI XPG-1000 utilizes a PCI local bus master/slave controller. This PCI controller allows the PCI XPG-1000 to become a PCI bus master, taking control of the PCI bus in burst mode, while freeing the host processor to perform useful processing while data is being transferred to/from the PCI XPG-1000.

To enhance the PCI burst mode performance, the PCI XPG-1000 incorporates a 1 Kbyte PCI transfer FIFO. This FIFO allows the bus mastering PCI controller to perform longer burst mode PCI transfers at the 132 Mbytes/second PCI peak data transfer rates. This results in a benchmarked sustained PCI bus throughput of over 50 Mbytes/second.

SOFTWARE

The XPG-1000 comes complete with a comprehensive software package including an extensive XVL vision library software package under DOS, Windows 3.1/95/NT running on the C40 DSP. It is the same software package which runs on the complete family of Power Grabber imaging boards. The XVL includes over 200 DSP based functions for image acquisition, processing, display and data transfer. A comprehensive memory management system has been implemented for the XPG to dynamically allocate memory for image data, program code, histogram, LUTs and other data buffers. The software fully supports different image size, bit depth and data types.

For easy prototyping and testing of new applications, the XVL also includes an interactive Windows based software environment called WiTlite to access the XVL functions. This gives the user the freedom to explore different imaging operators and see the results immediately with a demand-driven mode of operation.

To develop custom C40 based software on the XPG-1000, extensive tools from Texas Instruments such as C compiler, assembler, linker, simulator and debugger are available. Several third party Windows based software packages are also available on the XPG.

Dipix Technologies Inc.
Vision Products Division

PULNiX Products
TM-7 Series
1/2" Monochrome Mini Cameras

The TM-7 Series offers the highest resolution interline transfer 1/2" CCD imager available to date, in a very tiny package. Designed to meet a variety of application requirements, these four camera models have many standard and optional features. The most commonly needed adjustments:gain, gamma, AGC, and field/frame selection are easily accessible on the rear camera panel (excepting the super-miniature TM-7X). Shutter capability is another standard feature of this series. Each camera model is available in both EIA format (the TM-7 Series) and CCIR format (the TM-6 Series).

The "CN" and "EX" versions allow for the external selection of eight different shutter speeds (1/60 sec. to 1/10,000 sec.), by attaching the SC-745 Shutter Controller to the 6-pin connector located on the rear of the camera. Pixel clock output is a standard feature on the "CN" version; this function is optional on the TM-7X. Asynchronous shuttering is the key feature of the TM-7AS, which also offers a rotary select switch to control shutter settings.

The "AS" and "EX" versions accept external sync. Offering the tiniest package yet for location in small areas, the mini-cylindrical TM-7X features all input and output functions over a single, flexible cable. All cameras are C-mount; the "CN" and "EX" models include auto iris outputs.

The miniature size of the TM-7 Series cameras eliminates the need for a remoted imager camera in all but the most confined spaces. These cameras fit easily, both physically and functionally, into all types of machine vision, automated inspection, and related applications. Other uses include remotely piloted vehicles, miniature inspection devices, surveillance, microscopes and medical
equipment.

Specifications

Model	TM-7EX(EIA)	TM-7CN(EIA)	TM-7AS(EIA)	TM-7X(EIA)
Imager	1/2" Interline Transfer CCD			
Pixel	768(h) x 494(v)			
Cell Size	8.4µm x 9.8µm			
Scanning	525 lines 60Hz			
Sync.	Int./ext. auto switch fH+15.734 KHz 5% fV+59.94 Hz 5%	Internal crystal	Int./Ext. auto switch	Internal crystal
TV Resolution	570(h) x 350 (v)			
Min. Illumination	0.5 lux (F=1.4)			
Video output	1.0 Vp-p composite video, 75 ohms			
Gamma	1 or 0.45 back panel switch			
	1 (0.45 optional)			
AGC	ON/OFF back panel switch (16 dB std., 32 dB max.)			
	ON/OFF (OFF Std)			
	ON(OFF optional)			

Lens mount C-mount
Power Req. 9-12 V DC, 220 mA (TM-7AS is 250 mA)
Operating temp. -10 C to +50 C
Vibration & shock Vibration 7G, Shock 70G
Size (W X H x L) 45.8mm x 39.4mm x 66.3mm
1.81" x 1.54" x 2.60"
45.8mm x 39.4mm x 61.3mm
1.81" x 1.54" x 2.40"
45.8mm x 39.4mm x 66.3mm
1.81" x 1.54" x 2.60"
32mm (dia.) x 86.11mm
1.26" x 3.39"
Weight 177g(6.2 oz.) 171g(5.9 oz.) 190g(6.6 oz.) 115g(4 oz.)
Power cable 12P-02 2-meter cable attached
Power supply K25-12V, DC-12N, PD-12 or PD-12P ("CN" model)
Auto iris connector PC-6P none
Functional options

Model TM-6EX(CCIR) TM-6CN(CCIR) TM-6AS(CCIR) TM-6X(CCIR)
Imager 1/2" Interline transfer CCD
Pixel 752 (H) x 582 (V)
Cell size 8.6 nm x 8.3 nm (TM-6AS is 8.4 nm x 8.2 nm)
Scanning 625 lines CCIR
TV resolution 560 (H) x 420 (V)
AGC ON/OFF Back panel switch (16 dB std., 32 dB max.)
ON/OFF (OFF Std)
ON (OFF optional)
Gamma 1 or 0.45 back panel switch
1 (0.45 optional)
Power supply P-15-12
All other specifications same as EIA models.</B>

PULNiX America, Inc.
Web: http://www.pulnix.com Email: imaging@pulnix.com

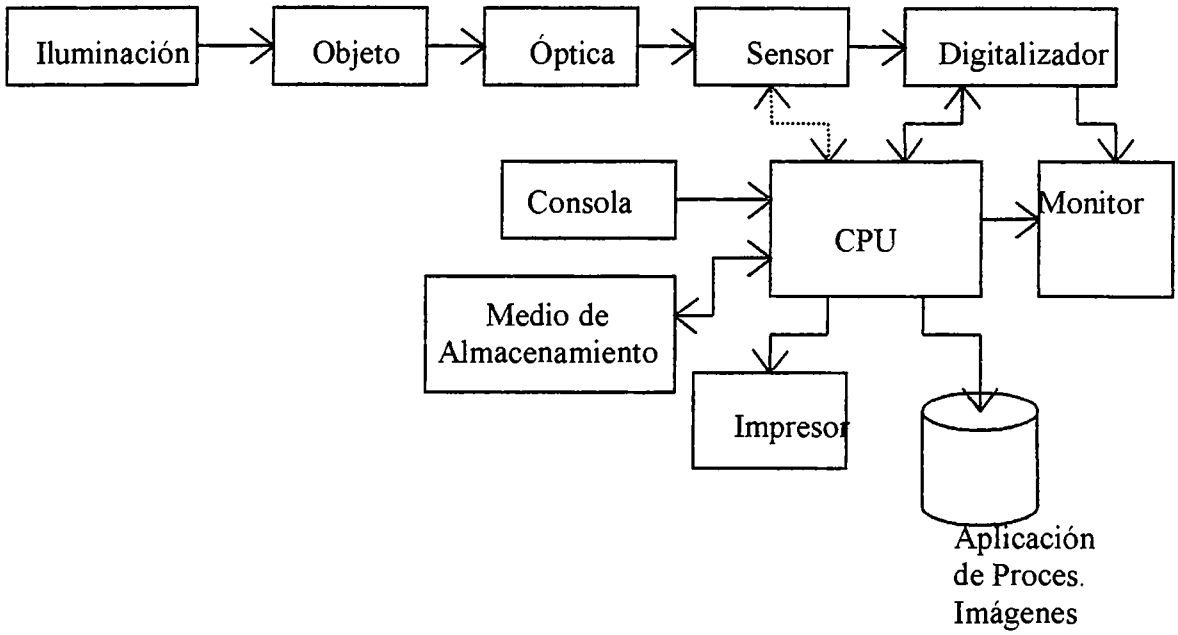
ANEXO DEFENSAS

En el desarrollo del trabajo se han diferenciado cinco etapas:

1. Recopilación de la información: La tarea de recopilar información se vio muy frustrada a nivel local, en consultas a bibliotecas de Universidades y librerías, puesto que han sido muy pocos los textos relacionados al tema del trabajo. Sin embargo, a través de Internet y de catálogos que se han recibido con las publicaciones de la IEEE en procesamiento de señales, se ha logrado juntar una serie de publicaciones y comprar libros con los que se completó el material de apoyo. Esta etapa demandó dos meses, aunque no está cerrada puesto que siempre puede aparecer alguna otra fuente que complemente lo que se ha encontrado.
2. Selección del equipamiento del sistema: Es notable como la mayoría de los libros consultados no dedican un espacio para describir o enunciar los criterios para utilizar uno u otro equipo en la adquisición de imágenes ni el procesamiento, si bien hacen mención de algunas características de ellos. También fue notable no haber encontrado un persona en ventas o área similar que manejara o conociera de este tema. Lo que manifiesta la poca preparación y divulgación que existe en el país. Esta fue una de las causas que nos impulsó a realizar este trabajo y por la que también alentamos a otros que hagan lo mismo, que busquen temas de vanguardia tecnológica para crecer en conocimientos científicos y reducir la brecha que nos separa de los países más desarrollados.
3. Experimentación con el equipamiento: Una serie de experiencias con el equipo de adquisición de imágenes e iluminación y óptica nos permitirán obtener los datos necesarios para proveer a los laboratoristas de la Universidad de una serie de experiencias que les permitan armar las guías de laboratorio. Así podrán en un futuro experimentar con los equipos a través del desarrollo de dichas prácticas.
4. Primera fase de desarrollo de programas: En esta etapa se llevará a cabo el desarrollo de los algoritmos de procesamiento de imágenes para incorporarlos luego a la aplicación principal y diseñar el ambiente de trabajo. Los programas de aplicación estarán desarrollados en Visual C++ en una plataforma Windows 95. Las etapas 3 y 4 estarían concluyéndose en dos meses y medio aproximadamente.

5. Integración del sistema y pruebas finales: Una vez definido el ambiente de trabajo, los algoritmos y experimentado con el equipamiento se realizará la integración del mismo y se redactará un manual de operaciones del sistema completo. De manera tal que cualquier persona con los conocimientos teóricos básicos en procesamiento de imágenes pueda operar el mismo y sacarle el máximo de provecho.

Se presenta a continuación un esquema de un sistema de procesamiento de imágenes genérico.



- La Iluminación: Muchas veces la iluminación ambiental no es suficiente o no cumple con los requerimientos mínimos para obtener una imagen adecuada de la escena en cuestión. Más evidente se hace esto cuando nos enfrentamos a aplicaciones que trabajan con elementos de iluminación que operan en otro espectro de energía que no es el visible. En consecuencia, el tipo de iluminación que se utilice va a estar condicionada por: (1) la fuente de luz, (2) las características del ambiente y (3) las características que se desean resaltar en el objeto.
- El Objeto: dos características de todo cuerpo deben tenerse en cuenta al formar la escena: (1) la capacidad de reflexión del objeto y (2) la capacidad de absorción de los

rayos. En relación a estos dos parámetros quedará definida la cantidad de energía que captará el sensor desde el objeto para formar una imagen de la escena formada.

- El Sistema Óptico: está constituido por una serie de lentes y básicamente actúan modificando dos parámetros: (1) la distancia de enfoque y (2) la abertura de diafragma. Con la combinación de estas dos selecciones se obtendrá: (1) Una profundidad de campo, (2) un área de enfoque y (3) una distancia de trabajo, adecuados todos al objeto específico. En algunos sistemas de visión se hace necesario operar con lentes especiales como lo son los de ojo de pescado. Una aplicación muy común de este tipo de lentes se da en los robots, ya que estas lentes permiten captar la escena de manera más completa por tener un mayor campo de visión, equivalente al ángulo visual. Las principales dificultades a vencer por las lentes son las aberraciones, sean cromáticas o geométricas, y la distorsión. Pero la diminuta dimensión de los sistemas ópticos minimizan dichas dificultades.
- El Sensor: Se pueden hacer varias clasificaciones de los sensores disponibles, siempre íntimamente relacionados con la aplicación específica. (1) En relación a la forma de exploración: pueden ser de línea o de área y (2) en relación al espectro de sensibilidad: los hay en UV, en IR, en Visible y pueden ser a la vez en blanco y negro o en colores y en Rayos X, por ejemplo. Otro aspecto importante a tener en cuenta es la resolución del mismo, es decir, la cantidad de elementos sensitivos que posee, también denominados pixeles.
- El Digitalizador: Puede ser interno o externo al sensor. Existen algunos que realizan tareas como simplemente cuantizar las muestras y codificar la información para que pueda ser incorporada luego a otro sistema digital. Como los hay otros más complejos generalmente externos y que cuentan con una gran versatilidad de programación y configuraciones, incorporando funciones de procesamiento de imágenes digitales. También tienen la capacidad de poder operar con más de un sensor en forma multiplexada.
- El resto del equipamiento se puede interpretar como aquel que proveerá los medios para poder interactuar con la primera parte, la de adquisición de imágenes. Para ello puede contarse con uno o varios monitores, brindando la posibilidad de observar las imágenes captadas y las procesadas. Una consola o teclado para ingresar datos o realizar selección

de parámetros, un impresor para reproducir los resultados obtenidos, imprimir configuraciones, uno o varios medios de almacenamiento, si fuere necesario, y fundamental la existencia de un programa de aplicación en Procesamiento de Imágenes Digitales.

Audiencia en general cuerpo evaluador, tengan todos muy buenas tardes. Mi intervención tendrá una duración de aproxiamadamente 22 minutos. En esta oportunidad se va hablar un poco del equipo de procesamiento de imágenes, que fue posible consultar. Dicho equipo comprende las funciones de captación y digitalización de la imagen, para que posteriormente se den las fases de presentación y procesamiento. Además se va a tratar una parte del espectro de las aplicaciones de los sistemas de procesamiento de imágenes.

El siguiente punto a tratar es:

CAMPO DE APLICACIÓN DE LAS CÁMARAS.

La industria de las cámaras, se extiende hacia la fabricación de cámaras de vídeo y cámaras fotográficas, más sin embargo en esta ocasión se desea destacar la aplicación que ha sido posible encontrar para las cámaras de vídeo. Como se puede observar en el esquema, las hay en vídeo de color y en B/N. Las cámaras B/N, por ser monocromáticas únicamente requieren de un dispositivo sensor que transforma la energía lumínica en energía eléctrica, las cámaras de color por lo general contienen en su interior tres sensores uno para cada color fundamental. El costo de las cámaras de color, es mayor en comparación a una cámara monocromática para una misma resolución, por lo que las de color rara vez se encuentran en una muy alta resolución.

Ahora bien, en relación al formato de la información que ellas entregan, se tienen cámaras analógicas y cámaras digitales; la resolución de una cámara analógica depende de la cantidad de líneas horizontales de TV que ellas entregan, indistintamente de la cantidad de elementos sensores. Según nuestra investigación fue posible encontrar una variedad de precios que van desde los $300 hasta $10,000 aproximadamente, el formato de salida y la

resolución necesaria, será de acuerdo a la aplicación asociada a este equipo. Entre los formatos de vídeo compuesto estándar se puede mencionar el RS-170 que también es conocido como formato EIA, NTSC, PAL.... y las aplicaciones asociadas a éstas pueden ser: ...

Por lo general las cámaras digitales presentan una alta resolución, propiedad que hace posible detectar detalles minuciosos en los objetos a ser captados. Es común que los formatos de la información proveniente de una cámara digital sea RS-422 ó TTL. De acuerdo a las características anteriores son aptas en los sistemas como ....

Después de haber obtenido una tabla concreta de comparación, se optó por adquirir una cámara CRESENT T-370 de una empresa Japonesa que se contactó vía Internet. Dicha cámara es la que presenta mejor opción de compra ya que su precio en virtud de las características es relevante y éstas se adecuan a las necesidades y aplicaciones demostrativas actuales del proyecto.

Ahora vamos a pasar al punto siguiente:

CÁMARA DE VISIÓN CONTRA CÁMARA FOTOGRÁFICA DIGITAL

Se tiene en pantalla un cuadro comparativo de la cámara de visión y la cámara fotográfica digital:

1) Como tercer punto: la cámara de visión, únicamente requiere de un disparo ya sea interno o externo para que en su salida se encuentre disponible la información necesaria para formar una imagen. Ese no es el caso de una cámara fotográfica digital, porque se requiere de un programa de aplicación para realizar el control y comunicación desde la PC hasta la cámara. Obviamente es una de las grandes desventajas para aplicarse en procesamiento de imágenes.

2) En el cuarto punto: se observa que las cámaras de visión son ideales para las aplicaciones como por ejemplo seguridad, inspección y por supuesto procesamiento de imágenes en todo

su espectro. El campo de aplicación de las cámaras fotográficas es en Multimedia, donde no se requiere mayor manipulación con las imágenes obtenidas, así como se observan en Internet.

3) Se encuentran dentro de una amplia gamma de resolución, pero como anteriormente se explicó, el factor que depende de ello es el precio y la aplicación misma. En cuanto a la contraparte, por lo general su resolución no representa ninguna diferencia apreciable para la vista humana, más sin embargo existen cámaras de vídeo de B/N, de muy alta resolución, por lo tanto las cámaras fotográficas digitales no son utilizables en aplicaciones como reproducción de piezas, medición,etc.

4) Ahora bien, las cámaras de visión no permiten almacenar imágenes en su interior y es una de las propiedades que se utilizan para realizar la transferencia de imágenes en tiempo real, por lo que luego de proporcionarle el disparo se obtiene la información de la secuencia de imágenes para observar video en vivo. Esta característica es limitada en una cámara fotográfica digital, ya que su aplicación se encuentra en otro campo, más no en aplicaciones tan específicas como por ejemplo en una línea de inspección automatizada, donde la cámaras de video son ideales.

5) Los conectores que se encuentran a la salida de una cámara de visión son del tipo RS-422 si su salida es digital, interface grandemente aplicada en la industria por tener un amplio rango de invulnerabilidad al ruido y del tipo BNC si la salida es vídeo compuesto. La razón principal de estos tipos de conectores es porque las cámaras están diseñadas de tal manera que sean interconectadas con una interface o frame grabeer. Mientras que las cámaras digitales fotográficas están diseñadas para importar archivos desde éstas hacia un puerto de la computadora como por ejemplo el RS-232 donde la aplicación en la mayoría de los casos es comercial.

6) Como anteriormente se dijo, las cámaras de visión pueden tener una salida digital o de vídeo compuesto sin otra información adicional, para que luego de haber captado y digitalizado la imagen, el usuario tenga la posibilidad de darle el formato deseado a cada

una de las imágenes. Las cámaras fotográficas, por lo general manejan un archivo, cuyo formato está definido por JPEG, listo para ser transferido a la PC, lo cual es una prueba más de la poca flexibilidad que nos permite dicho equipo.
Ahora centremos nuestra atención en el siguiente punto:

TARJETAS INTERFACES Y FRAME GRABBERS

1) Por lo general las tarjetas digitalizadoras no permiten la inserción de otra tarjeta. Y ello es una de las principales ventajas que proporcionan los Frame Grabbers, básicamente una Frame Grabbers es una tarjeta madre en la que es posible apilar más de una tarjeta hija con la capacidad de expandir el sistema en general. La potencialidad de conexión es tan elevada que permite la adición de tarjetas hijas, tanto así que la tarjeta FT-VI 24A de ALACRON permite la conexión de 4 cámaras de color ó 12 cámaras monocromáticas.
2) Las tarjetas digitalizadoras pueden o no poseer memoria de almacenamiento temporal, esa característica depende en gran medida de la potencialidad de procesamiento que tiene la tarjeta. Si las tarjetas no poseen memoria es posible transferir las imágenes en tiempo real, mediante la solicitud de un servicio DMA, aprovechando la potencialidad del bus PCI. Por otro lado el Frame Grabbers define una cantidad de memoria fisica, porque existe la posibilidad de administrar múltiples procesos.

3) Una característica que presentan las tarjetas es, que detectan el formato del vídeo compuesto de manera programada a través de las librerías de control o automáticamente. Los Frame Grabbers, solamente detectan el vídeo compuesto de manera programada. Ésta es una desventaja frente a las tarjetas.

4) Por lo general las tarjetas digitalizadoras únicamente aceptan vídeo compuesto en la mayoría de los formatos. Una gran ventaja que presentan los Frame Grabbers es que existe la posibilidad de conectar una salida digital que no necesariamente debe ser vídeo, además de que se puede conectar una salida analógica. Este un aspecto de flexibilidad en comparación a las tarjetas.

5) Las tarjetas digitalizadoras pueden o no tener Microprocesadores, tal el caso de la 4MEG Vídeo Modelo 12 de Epix, la cual posee uno o dos Microprocesadores. Se construyen con dos Microprocesadores cuando se realiza la adquisición y el procesamiento de la imagen al interior de la tarjeta. Los frame grabbers necesariamente tienen instalado por lo menos un Microprocesador para que administre el flujo de información proveniente de diversos puertos.

6) El costo de las tarjetas digitalizadoras está sujeto a la capacidad de procesamiento de la misma, así su costo varía entre $450 y $12000. El costo de los Frame Grabbers depende de la capacidad de expansión, capacidad de procesamiento, y por su puesto la cantidad de tarjetas hijas que sean necesarias para el sistema completo, de esa forma los precios oscilan entre $1600 a $6000.

Finalmente la tarjeta que se adquirió fue la PIXCI - SV4 de EPIX, la cual se eligió porque sin ninguna duda cumple con la aplicación que se desea implementar actualmente y principalmente porque se desea incorporar el equipiamiento a una maquinaria del tipo CNC así como el existente en los laboratorios del CITT, en este tipo de aplicación se hace necesaria la transferencia de imágenes en tiempo real, característica que posee la tarjeta adquirida, por otro lado se pretende continuar con la investigación por otra generación de esutudiantes, quienes se encargarán de realizar un sistema similar pero con imágenes a color, y desde el punto de vista económico fue nuestra mejor opción.

El siguiente punto a tratar es:

APLICACIÓN DE UN SISTEMA DE PROCESAMIENTO DE IMÁGENES DIGITALES

El espectro de las aplicaciones de los sistemas de procesamiento de imágenes es muy amplio por lo que se tratará una parte de ellas.

MILITAR: Reconocimiento aéreo de objetivos, y detección de movimientos militares.

OCEANOGRAFÍA: Estudio de la perforación de posos petroleros con precisión elevada, estudio de mareas en el océano.

VULCANOLOGÍA: Estudio de la posibilidad volcánica en relación al desplazamiento de masas en el interior.

SEGURIDAD: Se basa en el establecimiento de un patrón y cualquier diferencia con éste será tomado como una alteración a la normalidad así se tiene el reconocimiento de huellas digitales.

INSPECCIÓN AUTOMATIZADA: En una línea de producción como el llenado de botellas para verificar el rango permisible del volumen de líquido. Si la botella no cumple con el parámetro definido, un brazo electrónico se encarga de desplazarla.

ROBÓTICA: En un sistema de transportación inteligente que permite transportar cualquier tipo de carga a un sitio predefinido, guiándose por medio de un patrón.

MEDICIÓN: En las imágenes obtenidas se cuenta con la información de las dimensiones de los objetos captados, con una precisión de acuerdo a la resolución de la cámara.

ASTRLOGÍA: El interés se centra en el mejoramiento de las imágenes astrológicas, como por ejemplo a esta imagen original se le aplicó algoritmos para modificar el contraste y brillo, con el objeto de hacer resaltar detalles de la misma.

GEOGRAFÍA: En este tipo de aplicación la resolución es un factor crucial, según esta imagen, es posible calcular la altura de los árboles en virtud de la sombra producida de acuerdo a la posición del sol

CONTROL DE CALIDAD: Aplicación en la que se verifica que el producto terminado cumpla con los requerimientos de calidad misma. En esta imagen se ha tomado una imagen patrón que sirve de comparación con respecto al resto de las imágenes captadas dentro de la línea de producción. En la segunda imagen se observa que el dispositivo señalizado no cumple con las especificaciones.

PROCESAMIENTO GRÁFICO: En la imagen original se puede observar una degradación de la misma; en la imagen procesada se visualiza la mejoría de la textura del dibujo y el ajuste de los colores.

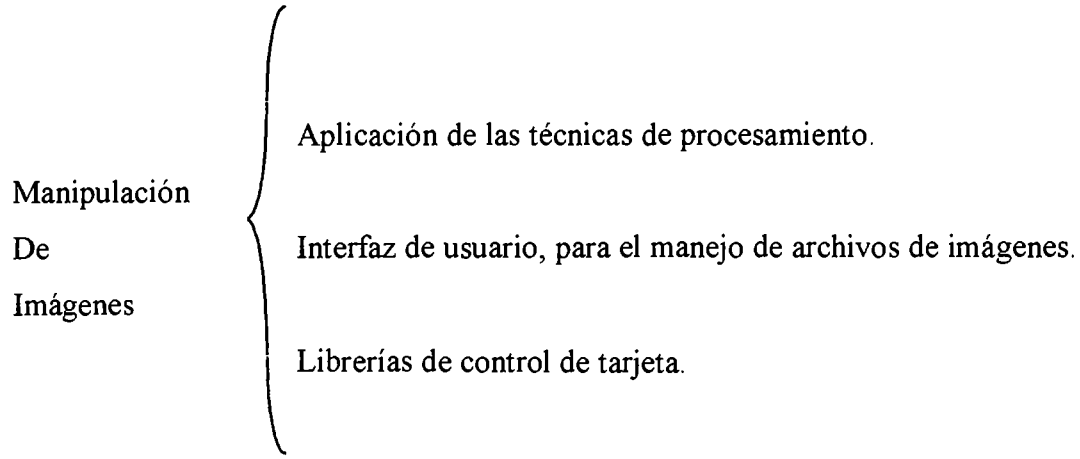
EL PROGRAMA DE PROCESAMIENTO DE IMÁGENES

Es el complemento de la tarjeta de adquisición. Esto es en virtud de las diferentes funciones que cada uno de ellos realiza y que se complementan para formar parte del sistema completo. La mayoría de fabricantes de tarjetas de adquisición a nivel comercial proveen también los programas de aplicación, para llevar a cabo el procesamiento de imágenes. Estos programas pueden estar incluidos en el precio de compra de la tarjeta o tener un precio adicional.

El precio de compra de los programas, depende en gran medida de la potencialidad y calidad de los mismos. La mayoría de veces el precio de estos programas es igual e incluso mayor que el de la misma tarjeta. Por ejemplo, el programa de aplicación de procesamiento XCAP, provisto por EPIX, la empresa fabricante de la tarjeta que se ha adquirido tiene un costo más del doble que el precio de la tarjeta.

Si no se dispone de un sistema de procesamiento dedicado, los programas de procesamiento de imágenes constituyen la mejor opción, debido que nos permiten hacer uso de los recursos de presentación y almacenamiento que dispone una computadora, sin incurrir en gastos adicionales. En términos generales los programas de procesamiento de imágenes constituyen el medio que nos permiten manipular las imágenes en forma digital.

El trabajo con las imágenes no solamente incluye la aplicación de las técnicas de procesamiento propiamente dichas, sino también tareas adicionales como por ejemplo:



La aplicación de las técnicas es la base principal del procesamiento de imágenes. Y en este aspecto uno de nuestros mayores compromisos al llevar a cabo este trabajo es el desarrollar un programa de aplicación que incluya las técnicas básicas del procesamiento de imágenes.

La interfaz de usuario constituye el modo de acceso que el usuario tiene a los datos de la aplicación, que en nuestro caso sería la información de las imágenes. Ésta permite manejar las imágenes como archivos, de esta manera se puede capturar, recuperar, presentar, almacenar y transmitir imágenes.

La librería de control de tarjeta es un tipo especial de programa que incluye funciones para controlar el uso de la tarjeta. Al igual que los programas de aplicación, estas librerías tienen un costo adicional relativamente alto, que para nuestro caso significa cerca de un 80% del costo de la tarjeta.

Ahora bien, ¿ Cuando es necesario el uso de la librería?.

Cuando se trabaja con un programa de aplicación no es necesario el uso de la librería, excepto en el caso que se desee extender la capacidad de la aplicación, siempre y cuando la aplicación lo permita, ó como en nuestro caso, cuando se quiera crear la aplicación completa. La librería que se utiliza es una colección de funciones creadas en lenguaje C. A través de la librería se puede controlar entre otras cosas:

La resolución de la imagen.

La profundidad de pixeles (N° de bits por pixel).

El formato de vídeo de entrada.

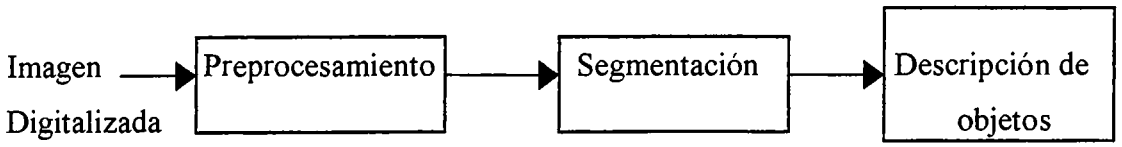
Modo de adquisición (por cuadro ó multicuadro).

El control de las entradas y salidas de disparo del conector de la tarjeta, etc.

Vale la pena mencionar también que una de las cosas que nos obligó a pensar en la necesidad de crear una aplicación haciendo uso de las librerías fue debido a las limitaciones de acceso a los programas propietarios de las cámaras digitales multimedia, como por ejemplo la PhotoPC600 que presenta muy buenas características de adquisición de imágenes, pero impide que la información proporcionada por esta cámara sea utilizada por otra aplicación, que no sea la diseñada por el fabricante.

LAS TÉCNICAS DE PROCESAMIENTO Y LAS APLICACIONES

Como se mencionó antes existen muchas aplicaciones del procesamiento de imágenes. Y la facilidad ó dificultad para obtener la información de las imágenes va a depender en gran medida del tipo de aplicación. Al momento de diseñar una aplicación es necesario conocer las etapas que comprende la aplicación y la secuencia correcta en la ejecución de cada una de ellas. A continuación se muestra un diagrama de las etapas comprendidas en una aplicación de reconocimiento de objetos, en una línea de producción.



Filtrado de ruido
Mejoramiento del Contraste

Detección de bordes
Umbralización
Segmentación por Regiones

Asignación de forma
Asignación de contorno

Preprocesamiento: Ayuda a realzar ó restaurar características perdidas durante la adquisición.

Segmentación: Separa los objetos del fondo, para facilitar la identificación de los objetos.

Descripción de objetos: Consiste en representar a los objetos a través de formas geométricas más simples.

El motivo de presentar este diagrama es para comprender la idea del procesamiento de imágenes. Es decir partir de las características generales de la imagen , hasta llegar a las específicas a través de la aplicación de las técnicas, lo cual implica su ejecución en el orden señalado.

AMBIENTE DE TRABAJO

El ambiente de trabajo es un elemento importante de un programa de aplicación, ya que determina la facilidad en el uso del programa. De hecho como parte de los alcances de nuestro trabajo, se ha incluido el desarrollar una interfaz de programa que sea lo más amigable posible. Dos aspectos que ayudan a mejorar el ambiente de trabajo son:

1) Un ambiente interactivo, que ayude a controlar y llevar un seguimiento de las operaciones de procesamiento.

2) La incorporación de una plataforma de programa que facilite la aplicación de las técnicas y la presentación de las imágenes.

Existen programas para trabajar en plataforma de PC DOS y Windows 3.1/95/NT.

Con el propósito de mostrar algunos detalles de la interfaz gráfica del programa que estaremos diseñando, hemos preparado un prototipo con las siguientes características:

Aplicación para Windows 95 (Aprovechar las características del Windows 95 + librerías de control de tarjeta para Windows 95)
Desarrollada en Visual C++
Uso de la MFC (Biblioteca fundamental de Microsoft)
Una interfaz múltiple de documentos como la base de la aplicación.

FORMATO DE LAS IMÁGENES

Es un aspecto importante a tomar en cuenta dentro del ambiente de trabajo. Si queremos almacenar imágenes necesitamos escoger uno o varios formatos para las imágenes. Tipos:

*TIFF
*PICT
*PCX
X/Y BINARIO
JPEG
BMP

* Requiere un formato especial, estandarizado por la industria.
Admiten compresión de datos.
Destinados para almacenar grandes imágenes y en color.

X/Y BINARIO: Matriz de pixeles sin formato, similar a una distribución en memoria. Usado en el análisis de imágenes, en donde no se requiere una representación gráfica, mas bien la representación de los datos.

JPEG: Es un algoritmo de almacenamiento y compresión de imágenes, generalmente fotográficas. Usado ampliamente en los escáner y cámaras digitales.

BMP: Es el formato estándar de Windows, tiene una estructura que contiene:

Una cabecera de dispositivo, que describe el tamaño del archivo de la matriz de datos.
Una paleta lógica de colores.
Y la matriz misma de datos.

ALGUNAS CONSIDERACIONES IMPORTANTES A CERCA DE LA ILUMINACIÓN DE LAS ESCENAS

- Footcandles, footlamberts y lumens:
- Apertura de diafragma y profundidad de campo:
- Los tipos de luces:
- Direccionalidad, uniformidad:
- Propuesta para mejorar la iluminación difusa:

Footcandles: Es una medida de la luz incidente, es decir, de la cantidad de luz llega a una superficie dada. Valores usuales van entre 30 y 70, el instrumento de medida que se utiliza es el fotómetro.

Footlamberts: Es una medida de la luz transmitida o reflejada. Todos vemos footlamberts. También es una medida de la máxima brillantez.

Lumens: Es una medida de la cantidad total de luz emitida por una fuente de luz.

Es muy importante lograr un contraste adecuado, por lo que será necesario trabajar con cuidado con la técnica de iluminación y la exposición. El gamma es una medida importante en los sistemas ópticos y de iluminación, y nos da una idea de la relación que existe entre el contraste real de la escena y el contraste con se reproducirá la misma. Para el caso de gamma = 1, tenemos que la escena se reproducirá con los mismos valores de la escena real, si es mayor que uno se tendrá un mayor contraste, y si es menor que uno, el

contraste será menor. Muchas cámaras traen consigo la posibilidad de cambiar este valor de gamma. En algunas aplicaciones en procesamiento de imágenes puede darse el caso en que el contraste más adecuado para trabajar no sea el mismo que el de la situación real, es decir, que el gama no sea igual a uno.

La apertura del diafragma y la profundidad de campo están íntimamente ligadas. Cuanto mayor es la apertura de diafragma, es decir, menores son los números f-stop, mayor es la profundidad de campo lograda. Por ello es importante trabajar con niveles de iluminación bajos, lo suficiente como para permitir una buena apreciación de las características de los objetos en la escena. Un claro ejemplo es la imagen que se ha adquirido del disipador con iluminación direccional, en contraste con las escenas saturadas de la iluminación difusa.

Íntimamente relacionado a estos temas está el tipo de luz que se utilice. Así podemos decir que existen básicamente tres tipos de luces que pueden ser aplicables a estas experiencias: las luces incandescentes comunes, las halógenas como lo son las de tungsteno, y por último las fluorescentes. De las incandescentes podemos decir que cuentan con el más bajo nivel de eficiencia en la conversión de energía eléctrica a lumínica, además, su vida útil también es reducida. Las lámparas halógenas producen una luz más blanca y tienen una expectativa de vida mayor que las anteriores. Cabe mencionar que la direccionalidad de luz producida por este tipo de lámparas es muy adecuado para aplicaciones en iluminación direccional. En relación a las fuentes de luz fluorescentes, estas son hasta cuatro veces más eficientes que las incandescentes. Esta es una de las razones por las que los fabricantes de lámparas se han enfocado en estos últimos años en proporcionar tantos modelos diferentes mejorando la coloración de la luz y sus aspectos estéticos. Este tipo de lámparas pueden aplicarse muy bien en iluminación direccional como en iluminación difusa de escenas haciendo uso de difusores de luz adecuados.

Una manera de poder mejorar las características de la iluminación difusa es operar con fuentes de luz que permitan regular su voltaje de alimentación. Para nuestro caso se utilizaron fuentes fluorescentes y superficies espejadas. Una buena alternativa en

iluminación difusa y direccional lo son las lámparas fluorescentes circulares. Para el caso de la iluminación direccional, la fuente de luz debe de colocarse a una distancia considerable de la escena, de manera tal que pueda considerarse que los haces luminosos forman un frente de ondas plano.

Analizando un poco el caso de la iluminación a contraluz, debe de considerarse que hay que minimizar cualquier otra fuente de luz existente en el ambiente. Esto garantiza un contraste total entre el fondo y los objetos que forman la escena, y obtener así imágenes con dos niveles de gris bien diferenciados.
de imágenes para incorporarlos luego a la aplicación

CODIFICACIÓN DE ALGORITMOS

A continuación se presenta el código fuente de cada una de las rutinas que han sido implementadas en el procesamiento de las imágenes. Vale la pena hacer la aclaración que el código pertenece al compilador de Visual C++ versión 5.0.

En las primeras páginas se encontrará la declaración de las funciones y datos miembro, en la siguiente parte de este anexo se encontrará con la codificación anfitrión de cada una de las funciones que componen esta clase.

```
//////////////////////////////// SPIDDoc.h : interface of the CSPIDDoc ////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////////////

#if
!defined(AFX_SPIDDOC_H__C4BE7854_3A74_11D2_A2B7_444553540000__INCLUD
ED_)
#define
AFX_SPIDDOC_H__C4BE7854_3A74_11D2_A2B7_444553540000__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#include "dibapi.h"

class CSPIDDoc : public CDocument
{
protected: // create from serialization only
        CSPIDDoc();
        DECLARE_DYNCREATE(CSPIDDoc)

// Attributes
public:
        HDIB GetHDIB() const
                { return m_hDIB; }
        CPalette* GetDocPalette() const
                { return m_palDIB; }
        CSize GetDocSize() const
                { return m_sizeDoc; }

//Declaración de datos miembros de clase
public:
```

```
        LPSTR lpDIB, lpbi;
        LPBYTE lpImagen;
        BYTE* lpCopia;
        LPBYTE lpTemp;
        static LPBYTE lpPatron;
        double* pHistogram; // Puntero a los valores de Histograma
        double* FFTran;

        int nAncho, nAlto, nY, nZ, nX, nW;
        int i, j, nDeshacer, nNivelGris, nFFT;
        static int nLocal;
        double Max, Coef;

        DWORD nLongiDIB, nBytesPix;

//Declaración de funciones miembros de clase
public:
        void NoImagenAct();
        int Promedio();
        int DesvEstand();
        double Histograma();
        DWORD TablaColor();
        DWORD NumColor();
        void OnEditarRotarDerecha();
        void OnEditarRotarIzquierda();
        void OnEditarDeshacer();
        void OnEditarEscalarImagen();
        void OnOpcionesControlInterface();
        void OnVerCopiaImagen();
        void OnEditarRotarAngulo();
        void OnFFT();
        void OnFFTInversa();
        void dfour1(double* data, unsigned long nn, int ising);

        BYTE SortArray(BYTE* nArray,int n,int nValue); // Ordenamiento de arreglos
        inline double Convolution1(int nZ,double x[]); //Convolución de la vecindad
                                                                    //centrada
en nZ con la mascara en x[]
        inline double Convolution2(int nZ,double x[],double y[]); //Convolución
        void Filtrado(BOOL bProc = TRUE);
        void Asig_Mas(double &p1,double &p2,double &p3,
                                        double &p4,double &p5,double &p6,
                                        double &p7,double &p8,double &p9,
                                        double d1, double d2, double d3,
                                        double d4, double d5, double d6,
                                        double d7, double d8, double d9);
private:
```

```
        void CopiaImagen();
        void CopiaImagen_Inv();
// Operations
public:
        void ReplaceHDIB(HDIB hDIB);
        void InitDIBData(BOOL FalsoColor, int nNiveles, int nColor,
                                        int nNivelUn, int nNivelMin, int nNivelMax);
// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CSPIDDoc)
        public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CSPIDDoc();
        CSize m_sizeDoc;

protected:
        virtual BOOL OnSaveDocument(LPCTSTR lpszPathName);
        virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);

protected:
        HDIB m_hDIB;
        CPalette* m_palDIB;

#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
        //{{AFX_MSG(CSPIDDoc)
        afx_msg void OnEditInvertir();
        afx_msg void OnFileInformacion();
        afx_msg void OnOpcionUmbralizar();
        afx_msg void OnAgregarRuido();
        afx_msg void OnEditarReflejarHorizontal();
        afx_msg void OnEditarReflejarVertical();
        afx_msg void OnProcesosAlisado();
        afx_msg void OnProcesosAfinado();
        afx_msg void OnProcesosDetectarBordes();
```

```
        afx_msg void OnProcesosFiltrosRank();
        afx_msg void OnProcesosMorfologiaImagen();
        afx_msg void OnProcesosAritmeticaPix();
        afx_msg void OnProcesosLogicaPix();
        afx_msg void OnProcesosImagenPatron();
        afx_msg void OnProcesosAritmeticaImagen();
        afx_msg void OnProcesosLogicaImagen();
        afx_msg void OnProcesosEcualizado();
        afx_msg void OnProcesosBalanceImagen();
        afx_msg void OnProcesosMorfNivelgris();
        afx_msg void OnProcesosFft();
        afx_msg void OnEditarEscalaGrises();
        afx_msg void OnProcesosFiltroPersonalizado();
        afx_msg void OnProcesosFftInversa();
        afx_msg void OnProcesosFiltrosFrecuencia();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()

        // Generated OLE dispatch map functions
        //{{AFX_DISPATCH(CSPIDDoc)
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_DISPATCH
        DECLARE_DISPATCH_MAP()
        DECLARE_INTERFACE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_SPIDDOC_H__C4BE7854_3A74_11D2_A2B7_444553540000__INCLUD
ED_)
```

```
////////////////////////////// SPIDDoc.cpp : implementation of the CSPIDDoc ////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "SPID.h"
#include <limits.h>
#include <math.h>
#include "SPIDDoc.h"
```

```
#include "SPIDView.h"
//Todos los archivos *.h que están a contunuación son los archivos de
//cabecera para las clases asociadas a cada cuadro de diálogo
#include "DialogInformation.h"
#include "DlgEscalaGris.h"
#include "DlgAgregaRuido.h"
#include "DlgEscalarImagen.h"
#include "DlgRotAngulo.h"
#include "DialogHumbral.h"
#include "DlgFiltrado.h"
#include "DlgBordes.h"
#include "DlgRank.h"
#include "DlgFiltradoEsp.h"
#include "DlgFiltroFrec.h"
#include "DlgFiltroPer.h"
#include "DlgMorfImagen.h"
#include "DlgMorfNiveles.h"
#include "DlgOperAritme.h"
#include "DlgOperLogic.h"
#include "DialogBalanceImagen.h"
#include "DlgInterface.h"
#include "DlgArimeImage.h"
#include "DlgLogicImage.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CSPIDDoc

IMPLEMENT_DYNCREATE(CSPIDDoc, CDocument)

BEGIN_MESSAGE_MAP(CSPIDDoc, CDocument)
        //{{AFX_MSG_MAP(CSPIDDoc)
//Éstos son los mensajes que se envían al sistema operativo para que actualice el
7/el estado de procesamiento de dicho mensaje
        ON_COMMAND(ID_EDIT_INVERT, OnEditInvertir)
        ON_COMMAND(ID_FILE_INFO, OnFileInformacion)
        ON_COMMAND(ID_OPCION_UMBRALIZAR, OnOpcionUmbralizar)
        ON_COMMAND(ID_EDITAR_AGREGARRUIDO, OnAgregarRuido)
        ON_COMMAND(ID_EDITAR_REFLEJAR_HORIZONTAL,
OnEditarReflejarHorizontal)
        ON_COMMAND(ID_EDITAR_REFLEJAR_VERTICAL,
OnEditarReflejarVertical)
```

```
        ON_COMMAND(ID_PROCESOS_ALISADO, OnProcesosAlisado)
        ON_COMMAND(ID_PROCESOS_AFINADO, OnProcesosAfinado)
        ON_COMMAND(ID_PROCESOS_DETECTARBORDES,
OnProcesosDetectarBordes)
        ON_COMMAND(ID_PROCESOS_FILTROSDERANK, OnProcesosFiltrosRank)
        ON_COMMAND(ID_PROCESOS_MORFOLOGADEIMAGEN,
OnProcesosMorfologiaImagen)
        ON_COMMAND(ID_PROCESOS_ARITMTICAYLGICA_OPERACIONESARIT
MTICAS, OnProcesosAritmeticaPix)
        ON_COMMAND(ID_PROCESOS_ARITMTICAYLGICA_OPERACIONESLGIC
AS, OnProcesosLogicaPix)
        ON_COMMAND(ID_PROCESOS_OPERACIONESENTREIMGENES_IMAGEN
PATRN, OnProcesosImagenPatron)
        ON_COMMAND(ID_PROCESOS_OPERACIONESENTREIMGENES_OPERAC
IONESARITMTICAS, OnProcesosAritmeticaImagen)
        ON_COMMAND(ID_PROCESOS_OPERACIONESENTREIMGENES_OPERAC
IONESLGICAS, OnProcesosLogicaImagen)
        ON_COMMAND(ID_PROCESOS_ECUALIZADO, OnProcesosEcualizado)
        ON_COMMAND(ID_PROCESOS_BALANCEDEIMAGEN,
OnProcesosBalanceImagen)
        ON_COMMAND(ID_PROCESOS_MORFNIVELGRIS,
OnProcesosMorfNivelgris)
        ON_COMMAND(ID_PROCESOS_FFT_FFT, OnProcesosFft)
        ON_COMMAND(ID_EDITAR_ESCALADEGRISES, OnEditarEscalaGrises)
        ON_COMMAND(ID_PROCESOS_FILTROS_PERSONALIZADO,
OnProcesosFiltroPersonalizado)
        ON_COMMAND(ID_PROCESOS_FFT_FFTINVERSA, OnProcesosFftInversa)
        ON_COMMAND(ID_PROCESOS_FILTROS_ENLAFRECUENCIA,
OnProcesosFiltrosFrecuencia)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CSPIDDoc, CDocument)
        //{{AFX_DISPATCH_MAP(CSPIDDoc)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_ISPID to support typesafe binding
// from VBA.  This IID must match the GUID that is attached to the
// dispinterface in the .ODL file.

// {C4BE7848-3A74-11D2-A2B7-444553540000}
static const IID IID_ISPID =
{ 0xc4be7848, 0x3a74, 0x11d2, { 0xa2, 0xb7, 0x44, 0x45, 0x53, 0x54, 0x0, 0x0 } };
```

```
BEGIN_INTERFACE_MAP(CSPIDDoc, CDocument)
        INTERFACE_PART(CSPIDDoc, IID_ISPID, Dispatch)
END_INTERFACE_MAP()

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CSPIDDoc construction/destruction

CSPIDDoc::CSPIDDoc()//Constructor
{
        m_hDIB = NULL;
        m_palDIB = NULL;
        m_sizeDoc = CSize(1,1); // valor correcto para CScrollView

        EnableAutomation();

        AfxOleLockApp();
}

CSPIDDoc::~CSPIDDoc()  //Destructor
{
        if (m_hDIB != NULL)
        {
                ::GlobalFree((HGLOBAL) m_hDIB);   //liberación del bloque de memoria
        }
        if (m_palDIB != NULL)
        {
                delete m_palDIB;
        }

        AfxOleUnlockApp();
}

BOOL CSPIDDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;

        // TODO: add reinitialization code here
        // (SDI documents will reuse this document)

        return TRUE;
}

void CSPIDDoc::InitDIBData(BOOL FalsoColor, int nNiveles, int nColor,
                                                int nNivelUn, int nNivelMin, int nNivelMax)
{
        if (m_palDIB != NULL)
```

```
        {
                delete m_palDIB;
                m_palDIB = NULL;
        }
        if (m_hDIB == NULL)
        {
                return;
        }
        // Definición del tamaño del documento
        LPSTR lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB);
        if (::DIBWidth(lpDIB) > INT_MAX ||::DIBHeight(lpDIB) > INT_MAX)
        {
                ::GlobalUnlock((HGLOBAL) m_hDIB);
                ::GlobalFree((HGLOBAL) m_hDIB);
                m_hDIB = NULL;
                CString strMsg;
                strMsg.LoadString(IDS_DIB_TOO_BIG);
                MessageBox(NULL, strMsg, NULL, MB_ICONINFORMATION |
MB_OK);
                return;
        }
        m_sizeDoc = CSize((int) ::DIBWidth(lpDIB), (int) ::DIBHeight(lpDIB));
        ::GlobalUnlock((HGLOBAL) m_hDIB);
        // Crea una copia de la paleta
        m_palDIB = new CPalette;
        if (m_palDIB == NULL)
        {
                //Debemos estar muy bajos en la cantidad de memoria
                ::GlobalFree((HGLOBAL) m_hDIB);
                m_hDIB = NULL;
                return;
        }
        if (::CreateDIBPalette(m_hDIB, m_palDIB, FalsoColor, nNiveles,
                nColor, nNivelUn, nNivelMin, nNivelMax) == NULL)
        {
                //Quiza el DIB no tiene una paleta
                delete m_palDIB;
                m_palDIB = NULL;
                return;
        }
}

BOOL CSPIDDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
        CFile file;
        CFileException fe;
        if (!file.Open(lpszPathName, CFile::modeRead | CFile::shareDenyWrite, &fe))
```

```
	{
		ReportSaveLoadException(lpszPathName, &fe,
			FALSE, AFX_IDP_FAILED_TO_OPEN_DOC);
		return FALSE;
	}

	DeleteContents();
	BeginWaitCursor();

	//Reemplace las llamadas para serializar con la funciónReadDIDFile()
	TRY
	{
		m_hDIB = ::ReadDIBFile(file);
	}
	CATCH (CFileException, eLoad)
	{
		file.Abort(); //Sera abortado el proceso
		EndWaitCursor();
		ReportSaveLoadException(lpszPathName, eLoad,
			FALSE, AFX_IDP_FAILED_TO_OPEN_DOC);
		m_hDIB = NULL;
		return FALSE;
	}
	END_CATCH

	InitDIBData(FALSE, 0, 0, 0, 0, 0);//Para crear nuevas paletas de colores
	EndWaitCursor();

	if (m_hDIB == NULL)
	{
		//Quiza no es un formato DIB
		CString strMsg;
		strMsg.LoadString(IDS_CANNOT_LOAD_DIB);
		MessageBox(NULL, strMsg, NULL, MB_ICONINFORMATION |
MB_OK);
		return FALSE;
	}
	SetPathName(lpszPathName);
	SetModifiedFlag(FALSE);
	return TRUE;
}


BOOL CSPIDDoc::OnSaveDocument(LPCTSTR lpszPathName)
{
	CFile file;
	CFileException fe;
```

```
        if (!file.Open(lpszPathName, CFile::modeCreate |
          CFile::modeReadWrite | CFile::shareExclusive, &fe))
        {
                ReportSaveLoadException(lpszPathName, &fe,
                        TRUE, AFX_IDP_INVALID_FILENAME);
                return FALSE;
        }

        // reemplaso de llamadas para serializar con la función SaveDIB
        BOOL bSuccess = FALSE;
        TRY
        {
                BeginWaitCursor();
                bSuccess = ::SaveDIB(m_hDIB, file);
                file.Close();
        }
        CATCH (CException, eSave)
        {
                file.Abort(); // El proceso es abortado
                EndWaitCursor();
                ReportSaveLoadException(lpszPathName, eSave,
                        TRUE, AFX_IDP_FAILED_TO_SAVE_DOC);
                return FALSE;
        }
        END_CATCH

        EndWaitCursor();
        SetModifiedFlag(FALSE);

        if (!bSuccess)
        {
                // Puede ser otro tipo de DIB (se puede cargar pero no guardar)
                //o otros problemas en SaveDIB
                CString strMsg;
                strMsg.LoadString(IDS_CANNOT_SAVE_DIB);
                MessageBox(NULL, strMsg, NULL, MB_ICONINFORMATION |
MB_OK);
        }

        return bSuccess;
}

void CSPIDDoc::ReplaceHDIB(HDIB hDIB)
{
        if (m_hDIB != NULL)
        {
                ::GlobalFree((HGLOBAL) m_hDIB);
```

```
        }
        m_hDIB = hDIB;
}

/////////////////////////////////////////////////////////////////////////////
// CSPIDDoc serialization

void CSPIDDoc::Serialize(CArchive& ar)
{
        if (ar.IsStoring())
        {
                // TODO: add storing code here
        }
        else
        {
                // TODO: add loading code here
        }
}

/////////////////////////////////////////////////////////////////////////////
// CSPIDDoc diagnostics

#ifdef _DEBUG
void CSPIDDoc::AssertValid() const
{
        CDocument::AssertValid();
}

void CSPIDDoc::Dump(CDumpContext& dc) const
{
        CDocument::Dump(dc);
}
#endif //_DEBUG

////////// Implementación de algoritmos para el procesamiento las imágenes//////////////////////

/////////////////////////////////////////////////Menú Archivo/////////////////////////////////////////////////////

void CSPIDDoc::NoImagenAct()
{
//Esta función genera un cuadro de diálogo en el que aparece la cadena indicada en la
función
        AfxMessageBox("No puede ejecutarse ninguno de los comandos si no existe una
imagen sobre el documento actual");
}
```

```
DWORD CSPIDDoc::TablaColor()//Esta función retorna el número de bits por cada pixel
de la imagen
{
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
DIB
        nBytesPix = ::DIBCount(lpDIB);//Función global que recupera la cantidad descrita
        return nBytesPix;
}

DWORD CSPIDDoc::NumColor()//Devuelve el número de colores usados en la paleta
{
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
DIB
        nBytesPix = ::DIBNumColors(lpDIB);//Función global que devuelve la cantidad
arriba descrita
        return nBytesPix;
}

int CSPIDDoc::Promedio()//Esta función evalúa el nivel de gris que es el promedio de los
niveles en imagen
{
        int nSumOfPix(0);//definición de variable acumuladora

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
DIB
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        for (i = 0; i <= nAlto -1; i++){//Estos lazos se encargan de explorar toda la imagen
                nY = (nLongiDIB*i)/nAlto;
        for (j = 0; j <= nAncho -1; j++){
                        nZ = nY + j;//nZ hace referencia a un pixel en particular asociado a
esa dirección
                        nSumOfPix += lpImagen[nZ];//Suma acumulada
                }
        }
                nSumOfPix /= (nAncho*nAlto);//Normalización de los datos
                return nSumOfPix;
}

int CSPIDDoc::DesvEstand()
{
        int nB,  nSumDesviacion(0);
        int nContraste;
        UINT nDesviacion;
```

```
        nB = Promedio();//Llamado a la función que calcula el promedio
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        for (i = 0; i <= nAlto -1; i++){
                nY = (nLongiDIB*i)/nAlto;
        for (j = 0; j <= nAncho -1; j++){
                        nZ = nY + j;
                        nDesviacion = ((lpImagen[nZ]- nB)*(lpImagen[nZ]- nB));
                        nSumDesviacion += nDesviacion;
                }//Con estos lazos se efectúa el cálculo de las distintas desviaciones
        }
                nContraste =(int) sqrt((double) nSumDesviacion/(nAncho*nAlto));
                return nContraste;//Esta variable contiene la desviación estándar
}

void CSPIDDoc::OnFileInformacion()//Se refiere a la información de la imagen o archivo
{
        if(m_hDIB != NULL){//Coprueba de que existe una imagen en el documento
        CDialogInformation dlg;//dlg es un objeto incrustado de la Clase
CDialogInformation
        int nBrillo, nContraste;

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        nBytesPix = ::DIBCount(lpDIB);

        nBrillo = Promedio(); //Valor del promedio de la imagen
        nContraste = DesvEstand();  //Valor de desviación estándar
        CString Tray = GetPathName();        //Se obtiene el path name de archivo

        sprintf(dlg.m_strNombre.GetBuffer(200),"%s", (const char* ) Tray);        //Escritura
de variables de cuadro de diálogo
        sprintf(dlg.m_strBytesPix.GetBuffer(5),"%d", nBytesPix);
        sprintf(dlg.m_strNbytes.GetBuffer(15),"%d", nLongiDIB);
        sprintf(dlg.m_strMean.GetBuffer(10),"%d", nBrillo);//Estas instrucciones se
encargan de presentar las variables en
        sprintf(dlg.m_strDeviation.GetBuffer(10),"%d", nContraste);//forma de texto
        sprintf(dlg.m_strNxpix.GetBuffer(10),"%d", nAncho);
        sprintf(dlg.m_strNypix.GetBuffer(10),"%d", nAlto);
```

```
        dlg.m_strNombre.ReleaseBuffer();  //Liberar la cantidad de memoria
        dlg.m_strNxpix.ReleaseBuffer();
        dlg.m_strNypix.ReleaseBuffer();
        dlg.m_strNbytes.ReleaseBuffer();
        dlg.m_strBytesPix.ReleaseBuffer();
        dlg.m_strMean.ReleaseBuffer();
        dlg.m_strDeviation.ReleaseBuffer();

        int ret=dlg.DoModal();          //Función que llama al cuadro de diálogo
        }
        else{NoImagenAct();}//Se manda un mensaje informando que no existe una imagen
en el documento
}

////////////////////Implementación de los algoritmos para el menú editar//////////////////////////

LPBYTE CSPIDDoc::lpPatron = NULL;    //Definición de variables globales estáticas
int CSPIDDoc::nLocal = 10;

void CSPIDDoc::CopiaImagen()  //Copia temporal de la imagen
{
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
DIB
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación de los datos de la
imagen
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);//Datos de imagen ó f(x,y)

        lpCopia = new BYTE[nLongiDIB];  //Definición de puntero temporal

        //Copia temporal de imagen
        memcpy(lpCopia, lpImagen, nLongiDIB);
}

void CSPIDDoc::CopiaImagen_Inv()
{
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
DIB
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);//f(x,y), datos de imagen

        memcpy(lpImagen, lpCopia, nLongiDIB); //Copia inversa de la imagen
        //para deshacer la última acción

}

void CSPIDDoc::OnEditarDeshacer()
{
```

```
        switch(nDeshacer){
        case 255://Significa que la imagen fue rotada a la derecha y para deshacer hay
                OnEditarRotarIzquierda();//que rotar a la izquierda
                nDeshacer = 1000;
                break;

        case 256:
                OnEditarRotarDerecha();//Este caso es similar al anterior
                nDeshacer = 1000;
                break;

        case 260://Este valor significa que cualquiera de las funciones restantes fue
ejecutada
        CopiaImagen_Inv();
        UpdateAllViews(NULL);
        }
}

void CSPIDDoc::OnEditarEscalaGrises()
{
        if(m_hDIB != NULL){//Comprueba si hay imagen activa en el documento
        UINT m = 0;

        CDlgEscalaGris dlg;//Objeto incrustado de CDlgEscalaGris que es un cuadro de
diálogo
        if(dlg.DoModal() == IDOK){//DoModal() es la función encargada de llamar a todas
las funciones
                if(nNivelGris != 1000){//relacionadas al cuadro de diálogo
                CopiaImagen();//Copia que sirve para deshacer esta acción
                nNivelGris = 1000;}

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        lpTemp = new BYTE[256];

        for(i = 0; i <= 255; i++){

                lpTemp[i] = m;//Se asigna un nivel de gris a cada uno de los elementos
//del arreglo de acuerdo a las elecciones definidas por el usuario
        if(dlg.m_nGrises == 0){
                if(i == (int) m + 8){//Para 32 valores posibles
                        m = i;}}
```

```
        if(dlg.m_nGrises == 1){
                if(i == (int) m + 16){//Para 16 valores posibles
                        m = i;}}
        if(dlg.m_nGrises == 2){
                if(i == (int) m + 32){//Para 8 valores posibles
                        m = i;}}
        if(dlg.m_nGrises == 3){//Para 4 valores posibles
                if(i == (int) m + 64){
                        m = i;}}
        if(dlg.m_nGrises == 4){//Para 64 valores posibles
                if(i == (int) m + 4){
                        m = i;}}
        if(dlg.m_nGrises == 5 && nNivelGris == 1000){
                CopiaImagen_Inv();}//Esta función se llama cuando el usuario elegido
restablecer del cuadro de diálogo

        }
        if(dlg.m_nGrises != 5){

        for (i = 0; i <= nAlto -1; i++){
                nY = (nLongiDIB*i)/nAlto;                    //i Coordenada y de la imagen
        for (j = 0; j <= nAncho -1; j++){
                nZ = nY + j;                                          //j Coordenada x de la
imagen
                        lpImagen[nZ] = lpTemp[lpImagen[nZ]];     //lpImagen[nZ] equivale
a f(x,y)
                        }
                }
        }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//Pone la bandera de verificación de guardar antes de salir
o cerrar el doc
        UpdateAllViews(NULL);//Actualiza las vistas de todas las imágenes
                }
        }
        else{NoImagenAct();}//Si no hay imagen activa manda un mensaje
}

void CSPIDDoc::OnEditInvertir()
{
        if(m_hDIB != NULL){//Igual que los anteriores y siguientes
                CopiaImagen();
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
DIB
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
```

```
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);//Datos de imagen

        for (i = 0; i <= nAlto -1; i++){
                nY = (nLongiDIB*i)/nAlto;                    //i Coordenada y de la imagen
        for (j = 0; j <= nAncho -1; j++){
                nZ = nY + j;                                          //j Coordenada x de la
imagen
                        lpImagen[nZ] = 255 - lpImagen[nZ];        //lpImagen[nZ] equivale
a f(x,y)
                }
        }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//  Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);}
        else{NoImagenAct();}
}

//Función global generadora de números aleatorios
int random(int maxVal)
{       return rand()%maxVal;}

void CSPIDDoc::OnAgregarRuido()
{
        if(m_hDIB != NULL){
        CDlgAgregaRuido dlg;
        int nX, nW, nTemp1, nTemp2;
        double Temp3;
        int nResponse = dlg.DoModal();
        if(nResponse == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        lpTemp = new BYTE[nLongiDIB];//Definición de un puntero temporal

        for (i = 0; i <= nAlto -1; i++){
                nY = (nLongiDIB*i)/nAlto;
        for (j = 0; j <= nAncho -1; j++){
                nZ = nY + j;
                lpTemp[nZ] = 0;}}//Inicialización del puntero en 0, este se usa como una
tabla
//donde se verifica si un pixel cualquiera ya ha sido modificado
```

```
        nX = nAncho*nAlto;
        nW = (dlg.m_nPercent*nX)/100;//Define la cantidad de pixeles a ser afectados con
ruido
        nTemp1 = dlg.m_nDesviacion*dlg.m_nDesviacion;

        if(dlg.m_nTipoRuido == 0){//Hace referencia al tipo de ruido sal
                i = 255;
                goto GenRuido;
                }

        else if(dlg.m_nTipoRuido == 2){//Se refiere al ruido salpimienta por eso nW/2
                nW /= 2;}
        else if(dlg.m_nTipoRuido == 3){//Se refiere al ruido uniforme por esa razón
nW/256
                nW /= 256;}

        for (i = 0; i <= 255; i++){//Lazo que se encarga de generar los niveles de gris de
ruido
         if(dlg.m_nTipoRuido == 4){//Ruido exponencial negativo tarta de presentar los
256 niveles de gris
                 Temp3 = (exp(-i/dlg.m_nDesviacion))/1000;//de acuerdo a la función de
trasferencia
                 nW = (int)(Temp3*nX)/dlg.m_nDesviacion;
          nW *= 300;}

         else if(dlg.m_nTipoRuido == 5){//Ruido Gaussiano, trata de genrerar los 256
valores de gris
                 nTemp2 = (i - dlg.m_nPromedio)*(i - dlg.m_nPromedio);//por definición
                 nW = int((exp((-nTemp2)/nTemp1)/dlg.m_nDesviacion*2.507)
                         *nX)/10;}

GenRuido: for (j = 0; j <= nW; j++){
NewPos:                 nY = (nLongiDIB/nAlto)*random(nAlto);//Fila aleatoria
                        nZ = nY + random(nAncho);//Columna aleatoria
                if(lpTemp[nZ] == 0){
                lpImagen[nZ] = i;//Substitución de f(x,y) por ruido
                lpTemp[nZ] = 1;}
                else{ goto NewPos;}
                }

                if(dlg.m_nTipoRuido==1){i=255;}//Para ruido del tipo pimienta
                if(dlg.m_nTipoRuido == 2){
                  i = 255;
                for (j = 0; j <= nW; j++){
                nY = (nLongiDIB/nAlto)*random(nAlto);
                nZ = nY + random(nAncho);
                lpImagen[nZ] = i;}
```

```
                    }
                }
        nDeshacer = 260;//Para proceder a Deshacer esta acción
        SetModifiedFlag(TRUE);//  Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        }
        else{NoImagenAct();}//Igual que las anteriores y siguientes
}

void CSPIDDoc::OnEditarEscalarImagen()
{
        if(m_hDIB != NULL){
        CDlgEscalarImagen dlg;
        DWORD TempX, TempY, Temp, Nuevo1X, Nuevo1Y, nTaman;
        double NuevoX, NuevoY;

        if(dlg.DoModal() == IDOK){//Igual que las anteriores y siguientes
                CopiaImagen();
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);         //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);
        lpTemp = new BYTE[nLongiDIB];
        memcpy(lpTemp, lpImagen, nLongiDIB);
//Las variables Tempx guardan los datos antes del escalado
        TempX = nAncho;
        TempY = nAlto;
        nAlto = (DWORD)(nAlto*dlg.m_nEscalarY);//Redimendiona el alto de imagen
        nAncho = (DWORD)(nAncho*dlg.m_nEscalarX);//Redimendiona el ancho de
imagen
        if(nAlto > 950 || nAncho > 950){
                AfxMessageBox("¡Ha sobrepasado el límite de la matriz de datos!");}
        else{
        Temp = nLongiDIB;
        nLongiDIB = ::SizeImageNew(nAncho, nAlto, lpDIB);//Redifición de la longitud
del DIB
        nTaman = ::GlobalSize((HGLOBAL) m_hDIB) + (nLongiDIB - Temp);
        m_hDIB = (HDIB)::GlobalReAlloc((HGLOBAL) m_hDIB, nTaman, 0);//Permite
redefinir el tamaño del DIB en Memoria
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB);//matriz de datos
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);//Nuevos datos de imagen

        for (i = 0; i <= (nAlto - 1); i++){//Exploración en virtud de la nueva imagen
                nY = (nLongiDIB*i)/nAlto;
```

```
                NuevoY = i/dlg.m_nEscalarY;
                Nuevo1Y = (int) NuevoY; NuevoY -= Nuevo1Y;

        for (j = 0; j <= (nAncho - 1) ; j++){
                nZ = nY + j;
                NuevoX = j/dlg.m_nEscalarX;
                Nuevo1X = (int) NuevoX;
        int nZ1 = (Nuevo1Y*Temp)/TempY + Nuevo1X;
                NuevoX -= Nuevo1X;

        if(dlg.m_nInterpolacion == 0){//Interpolador réplica de pixel
                if(NuevoX < 0.5 || NuevoY < 0.5){
                        lpImagen[nZ] = lpTemp[nZ1];}
                else{lpImagen[nZ] = lpTemp[nZ1+(Temp/TempY)+1];}
        }

        if(dlg.m_nInterpolacion == 1){//Interpolador vecino más cercano
                lpImagen[nZ] = (BYTE) ((1 - NuevoX)*(1 - NuevoY)*lpTemp[nZ1] +
                        NuevoX*(1 - NuevoY)*lpTemp[nZ1 + 1]
                        + NuevoY*(1 - NuevoX)*lpTemp[nZ1 + (Temp/TempY)] +
                        NuevoX*NuevoY*lpTemp[nZ1 + (Temp/TempY) + 1]);
        }

        if(dlg.m_nInterpolacion == 2){//Interpolador de promedio
                lpImagen[nZ] = (BYTE) ((lpTemp[nZ1] + lpTemp[nZ1 + 1] +
                        lpTemp[nZ1 + (Temp/TempY)] + lpTemp[nZ1 + (Temp/TempY) +
1])/4);
                }
        }
        }
        m_sizeDoc.cx = nAncho;//Redifinicón del tamaño de las barras de desplazamiento
horizontal
        m_sizeDoc.cy = nAlto;//y vertical
        SetModifiedFlag(TRUE);//   Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
                }
        }
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnEditarRotarAngulo()
{
        if(m_hDIB != NULL){
        CDlgRotAngulo dlg;
        if(dlg.DoModal() == IDOK){
```

```
        DWORD nTemp;
    int nAltoNuevo, nAnchoNuevo;
    int nXant, nYant;
    int nY1, nZ1, m_nFondo;
    float m_nGrado;
    m_nGrado = dlg.m_nAngulo;
    m_nFondo = dlg.m_nFondo;
    const double pi = 4*atan(1),DegARad = pi/180;
    double cosAng, senAng;
    lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición
                                                    //del puntero al encabezado de archivo

    nAlto = ::DIBHeight(lpDIB);//Alto de imagen
    nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
    nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación

    cosAng = cos(DegARad*m_nGrado);//Transformación y cálculo de los coeficientes
    senAng = sin(DegARad*m_nGrado);
    nAltoNuevo = int((nAlto*cosAng)+ (nAncho*senAng));//Nueva dimensión el alto
de la imagen
    nAnchoNuevo = int((nAlto*senAng) + (nAncho*cosAng));//Nueva dimensión en el
ancho de la imagen
    if(nAltoNuevo > 950 || nAnchoNuevo > 950){
        AfxMessageBox("¡Ha sobrepasado el límite de la matriz de datos!");}
    else{
    nTemp = nLongiDIB; //nTemp almacena el tamaño original
    lpImagen = (LPBYTE)::FindDIBBits(lpDIB);
    lpTemp = new BYTE[nLongiDIB];
    memcpy(lpTemp, lpImagen, nLongiDIB);//Copia temporal de imagen

    nLongiDIB = ::SizeImageNew(nAnchoNuevo,nAltoNuevo,lpDIB);//asignar
                                                              //nuevas
dimensiones
    DWORD dTamOrig = ::GlobalSize(m_hDIB);//Obtener tamaño
                                                              //original del DIB

    m_hDIB = (HDIB)::GlobalReAlloc((HGLOBAL)m_hDIB, dTamOrig +
nLongiDIB - nTemp,0);
    lpDIB = (LPSTR)::GlobalLock((HGLOBAL) m_hDIB);
    lpImagen = (LPBYTE)::FindDIBBits(lpDIB);
    memset(lpImagen, m_nFondo, nLongiDIB); //Llenar el bloque  de memoria
                                                    //con un nivel de gris (nFondo)

    /* Escaneo de la imagen destino y mapeo a la imagen origen
    para evitar el patrón de fondo, producido por el proceso inverso
    es decir escanear la imagen origen y mapear a la imagen destino
    NOTA: Para mayor detalle sobre este proceso consultar "Algoritmos
```

```
        de Procesamiento de Imagenes Digitales" de I.Pitas*/


        for (i = 0; i <=(nAltoNuevo - 1); i++){
                nY = (int)(nLongiDIB/nAltoNuevo)*i; // Pos.del primer dato de la Fila
                int ii = (int) (i - (nAncho*senAng)); //Coordenada con desplazamiento

                for (j = 0; j <=(nAnchoNuevo - 1); j++){
                nZ = nY + j; // Punto específico dentro del arreglo de datos

                nXant = int (j*cosAng - ii*senAng);//Cálculo de la antigua ubicación
                nYant = int(j*senAng + ii*cosAng);//de cada uno de los f(x,y)

                if (nXant>=0 && nXant<nAncho-1 && nYant>=0 && nYant<nAlto-1){
                        nY1 = (nTemp/nAlto)*nYant;
                        nZ1 = nY1 + nXant;
                        lpImagen[nZ] = lpTemp[nZ1];}
                }
        }
        m_sizeDoc.cx = nAnchoNuevo;//Actualización del tamaño de la ventana
        m_sizeDoc.cy = nAltoNuevo;//vista y las barras de desplazamiento
        SetModifiedFlag(TRUE);//Notificación de cambios en la imagen
        UpdateAllViews(NULL);//Dibujar la imagen actualizada
        }
        }
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnEditarRotarDerecha()
{
        if(m_hDIB != NULL){
        DWORD nTemp;
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        lpTemp = new BYTE[nLongiDIB];

        memcpy(lpTemp, lpImagen, nLongiDIB);   //Copia temporal de imagen

        nTemp = nLongiDIB;
        nLongiDIB = ::SizeImageNew(nAlto, nAncho, lpDIB);//esta función se encarga de
modificar las dimensiones y
```

```
        nX = nLongiDIB/nAncho;//la justificación de los datos

        for (i = 0; i <= nAlto - 1; i++){
                nY = (int)(nTemp/nAlto)*i;

        for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;
        nW = nLongiDIB - (1 + j)*nX + i;   //Cálculo de la nueva ubicación
        lpImagen[nW] = lpTemp[nZ];                  //de cada uno de los f(x,y)
                }
        }
        m_sizeDoc.cx = nAlto;
        m_sizeDoc.cy = nAncho;
        nDeshacer = 255;
        SetModifiedFlag(TRUE);//   Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnEditarRotarIzquierda()
{
        if(m_hDIB != NULL){
        DWORD nTemp;
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);
        lpTemp = new BYTE[nLongiDIB];

        memcpy(lpTemp, lpImagen, nLongiDIB);   //Copia temporal de imagen

        nTemp = nLongiDIB;
        nLongiDIB = ::SizeImageNew(nAlto, nAncho, lpDIB);

        nX = nLongiDIB/nAncho;

        for (i = 0; i <= nAlto - 1; i++){//Con estos lazos se toman dos pixeles
                nY = (int)(nTemp/nAlto)*i;//simultáneamente y se relocalizan en la matriz
de datos

        for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;
                nW = nLongiDIB - (nAncho - (1 + j))*nX - i
                 - (nX - nAlto) - 1;//nZ contiene las coordenas complementarias a nW
```

```
        lpImagen[nW] = lpTemp[nZ];
                }
        }
        m_sizeDoc.cx = nAlto;
        m_sizeDoc.cy = nAncho;
        nDeshacer = 256;
        SetModifiedFlag(TRUE);//  Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);    //Actualización de la vista
        }
        else{NoImagenAct();}
}


void CSPIDDoc::OnEditarReflejarHorizontal()
{
        if(m_hDIB != NULL){
        BYTE Temp1, Temp2;

                CopiaImagen();
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        int nAncho1 = nAncho - 1;
        int nAnchor = nAncho1/2 ;

        for (int i = 0; i <= nAlto - 1; i++){//Con estos lazos se toman dos pixeles
                nY = (nLongiDIB/nAlto)*i;//en la matriz de datos

        for (int j = 0; j <= nAnchor ; j++){
                nZ = nY + j;
                nX = nY + nAncho1 - j;

                Temp1 = lpImagen[nZ];
                Temp2 = lpImagen[nX];
                lpImagen[nZ] = Temp2;
                lpImagen[nX] = Temp1;
                }
        }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//  Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);    //Actualización de la vista
        }
        else{NoImagenAct();}
```

```
}

void CSPIDDoc::OnEditarReflejarVertical()
{
        if(m_hDIB != NULL){
        BYTE Temp1, Temp2;

                CopiaImagen();
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        int nAlto1 = nAlto/2;
        int nAltor = nAlto1 - 1, a;
        int Temp = (nLongiDIB/nAlto) - nAncho;

        for (int i = 0; i <= nAltor; i++){//Con estos lazos se toman dos pixeles
                nY = (nLongiDIB/nAlto)*i;//en la matriz de datos
                a = i + 1;
        for (int j = 0; j <= nAncho - 1 ; j++){
                nZ = nY + j;
                nX = nLongiDIB - a*Temp - a*nAncho + j; //Cálculo de la nueva posición
del pixel

                Temp1 = lpImagen[nZ];//intercambio temporal de localidades
                Temp2 = lpImagen[nX];
                lpImagen[nZ] = Temp2;
                lpImagen[nX] = Temp1;
                }
        }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//   Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        else{NoImagenAct();}
}

//////////////////Implementación de algortimos para las rutinas de opciones//////////////////////////

void CSPIDDoc::OnOpcionesControlInterface()
{
        CDlgInterface dlg;
        if(dlg.DoModal() == IDOK){
        OnOpenDocument(dlg.m_strPathName);
```

```
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen

        m_sizeDoc.cx = nAncho;//Definen las dimensiones de las barras de desplamiento de
la vista
        m_sizeDoc.cy = nAlto;
        }
}

double CSPIDDoc::Histograma()
{
        double Temp;
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);
        Temp = pow(2, ::DIBCount(lpDIB));
        double dMaxOfHistogram(0); //Valor máximo del histograma.
        pHistogram = new double[(int)Temp];

        for (int m = 0; m <= (int) Temp - 1; m++){
                pHistogram[m] = 0; //inicializar cada valor de histograma
        }

        for (i = 0; i <= nAlto - 1; i++){
                nY = (nLongiDIB*i)/nAlto;
        for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;
                pHistogram[lpImagen[nZ]] += 1;//Suma uno a cada elemento del array
siempre
                }//que un nivel de gris hace referencia a una localidad específica
        }

        for (int k = 0; k <= (int) Temp - 1; k++){
                pHistogram[k] /= (nAlto*nAncho);
                if(pHistogram[k] > dMaxOfHistogram){
                        dMaxOfHistogram = pHistogram[k];}//Normalización de los datos

        }
        return dMaxOfHistogram;//Retorna el valor máximo de los datos
}
```

```
void CSPIDDoc::OnOpcionUmbralizar()
{
        if(m_hDIB != NULL){
        CDialogHumbral dlg;
        if(dlg.DoModal() == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
DIB
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);         //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        for (i = 0; i <= nAlto - 1; i++){
                nY = (nLongiDIB/nAlto)*i;

        for (int j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;

        switch(dlg.m_nNivel){

        case 0://Umbralización a un nivel
        if(lpImagen[nZ] >= (BYTE) dlg.m_VHumbral){
                        lpImagen[nZ] = 255;}
        else{lpImagen[nZ] = 0;}
                break;

        case 1://Umbralización a doble nivel
                if(lpImagen[nZ]>=dlg.m_nNivel1 && lpImagen[nZ]<=dlg.m_nNivel2){
//Definición del rango
                        lpImagen[nZ] = 255;}
        else{lpImagen[nZ] = 0;}
                break;

        case 2://Umbralización a doble nivel con pendiente igual a uno
        if(lpImagen[nZ]>=dlg.m_nNivel1 && lpImagen[nZ]<=dlg.m_nNivel2){
                        lpImagen[nZ] = 255;}          //Se afectan los f(x,y) que se encuentran
dentro del rango
                break;
                                }
                        }
                }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//   Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
```

```
        }
        else{NoImagenAct();}
}

////////////////////////////////Implementación de algoritmos para menú
procesos////////////////////////////

void CSPIDDoc :: Filtrado(BOOL bProc)
{
        CDlgFiltrado dlg;
        dlg.m_nb = 1;
        int nRetorno = dlg.DoModal();
        double b = dlg.m_nb; // b es el grado en la ecuación generadora
        double Den = (b + 2)*(b + 2);
        double param1, param2, param3;

        if (nRetorno == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

                if (bProc == TRUE) {
                        param1 = 1/Den;
                        param2 = b/Den;
                        param3 = b*b/Den;}
                else {
                        param1 = -1/Den;
                        param2 = -b/Den;
                        param3 = 4*(b+1)/Den;}
        double x[] =
{param1,param2,param1,param2,param3,param2,param1,param2,param1};

        int* pImageTemp = new int[nLongiDIB - 1];

        memcpy(pImageTemp, lpImagen, nLongiDIB);   //Copia de la imagen

        //Cómputo sobre la imagen
        //(sin incluir primera y última fila/columna)
        for (i = 1; i <= nAlto -2; i++){
                nY = (nLongiDIB*i)/nAlto;
                for (j = 1; j <= nAncho -2; j++){
                nZ = nY + j;
```

```
                double dSumOfNeigh = Convolution1(nZ,x);
                        if (bProc == FALSE){

                                if (dSumOfNeigh < 0 ){
                                dSumOfNeigh = 0;
                                }
                                else {
                                dSumOfNeigh *= 255/(255*x[4]);
                                //dSumOfNeigh *= -1;
                                }
                        }
                        pImageTemp[nZ] = (int)dSumOfNeigh;

                }
        }
        for (int m = 0; m <= (int) nLongiDIB - 1; m++){
                lpImagen[m] = (BYTE) pImageTemp[m];
        }
        }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//  Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);    //Actualización de la vista
}

double CSPIDDoc::Convolution1(int nZ,double x[])
{
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        double dSumOfNeigh1(0);
        int nCountOfNeigh(0);
        for (int p = -1; p <= 1; p++){//Lazos para la recuperación de los datos que son
afectados
                int nYx = (nZ + (nLongiDIB/nAlto) * p);//por la máscara
        for (int q = -1; q <= 1; q++){
                        int nZx = nYx + q;
                        if ( x[nCountOfNeigh]!= 0 ){
                                dSumOfNeigh1 += lpImagen[nZx]*x[nCountOfNeigh];}
                        nCountOfNeigh++;
                }
        }
        return dSumOfNeigh1;
}
```

```
double CSPIDDoc::Convolution2(int nZ,double x[],double y[])
{
	lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
	nAlto = ::DIBHeight(lpDIB);//Alto de imagen
	nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
	lpImagen = (LPBYTE)::FindDIBBits(lpDIB); //Imagen f(x,y)

	double dSumOfNeigh1(0);
	double dSumOfNeigh2(0);
	int nCountOfNeigh(0);
	for (int p = -1; p <= 1; p++){
		int nYx = nZ + (nLongiDIB/nAlto) * p;
		for (int q = -1; q <= 1; q++){
			int nZx = nYx + q;
			if ( x[nCountOfNeigh]!= 0 ){
				dSumOfNeigh1 += lpImagen[nZx]*x[nCountOfNeigh];}
			if ( y[nCountOfNeigh]!= 0) {
				dSumOfNeigh2 += lpImagen[nZx]*y[nCountOfNeigh];}
			nCountOfNeigh++;
		}
	}
	if (dSumOfNeigh1 < 0) dSumOfNeigh1 *= -1;
	if (dSumOfNeigh2 < 0) dSumOfNeigh2 *= -1;
	return dSumOfNeigh1 + dSumOfNeigh2;
}

void CSPIDDoc::Asig_Mas(double &p1,double &p2,double &p3,
                                                double &p4,double &p5,double &p6,
                                                double &p7,double &p8,double &p9,
                                                double d1,double d2,double d3,
                                                double d4,double d5,double d6,
                                                double d7,double d8,double d9)
{
	p1 = d1;	//El objetivo de esta función es asignar
	p2 = d2;	//cada una de las máscaras dependiendo del
	p3 = d3;	//operador para detectar bordes
	p4 = d4;
	p5 = d5;
	p6 = d6;
	p7 = d7;
	p8 = d8;
	p9 = d9;
}

BYTE CSPIDDoc::SortArray(BYTE* nArray,int n,int nValue)
{
```

```
        /*Esta función se escarga de ordenar una cantidad de datos del valor
        menor hasta el valor que sea encontrado a través del arreglo que se le pasa
        a ésta como un parámetro más*/
        int offset, temp, inOrder, ret;

        offset = n;
        do {
                offset = (8 * offset)/11;
                offset = (offset == 0) ? 1 : offset;
                inOrder = TRUE;
                for (int m = 0, k = offset; m < (n - offset); m++, k++) {
                        if (nArray[m] > nArray[k]) {
                                inOrder = FALSE;
                                temp = nArray[m];
                                nArray[m] = nArray[k];
                                nArray[k] = temp;
                        }
                }
        }while (!(offset == 1 && inOrder == TRUE));

        if (nValue != 3) {
        switch (nValue) {
                case 0:
                        ret = 0;
                        break;
                case 1:
                        ret = (n-1)/2;
                        break;
                case 2:
                        ret = n-1;
                        break;
        }
        return nArray[ret];     }
        else {
                return BYTE((nArray[n-1]+nArray[0])/2);}
}

void CSPIDDoc::OnProcesosAlisado()
{
        if(m_hDIB != NULL){
        BOOL bProc = TRUE;
        Filtrado(bProc);//Llamado a la función que se enacarga de efectuar la convolución
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosAfinado()
```

```
{
        if(m_hDIB != NULL){
        int nZY1, nZY2, nZY3, nZY4, nZX1, nZX2, nZX3, nZX4;
        int nTemp; double Convo;
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
DIB
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        lpTemp = new BYTE[nLongiDIB];
        nTemp = nLongiDIB/nAlto;

        memcpy(lpTemp, lpImagen, nLongiDIB);   //Copia temporal de imagen

        for (i = 1; i <= nAlto -2; i++){
                nY = (nLongiDIB*i)/nAlto;
        for (j = 1; j <= nAncho -2; j++){
                nZ = nY + j;
                nZY1 = nZ - nTemp; nZY2 = nZ + nTemp; nZX1 = nZ - 1;//nZXX define
las coordenadas de cada uno de
                nZX2 = nZ + 1; nZY3 = nZY1 - 1; nZY4 = nZY1 + 1;//los elementos que
circundan al pixel central, es decir
                nZX3 = nZY2 - 1; nZX4 = nZY2 + 1;//se ecnuentra cada uno de los 8
vecinos

        Convo = (- lpTemp[nZY3] - lpTemp[nZY1] - lpImagen[nZY4]
                        - lpTemp[nZX1] + 7*lpTemp[nZ] - lpImagen[nZX2] //Definición de
                        - lpTemp[nZX3] - lpTemp[nZY1] - lpImagen[nZX4])/9; //Laplaciano
y multiplicaión de cada vecino por el parámetro de la máscara

        Convo = lpTemp[nZ] + Convo + 20;
        if(Convo > 255){
                lpImagen[nZ] = 255;}//Si convolución sobrepasa la escala el pixel actual
toma el máximo
        else if(Convo < 0){//si convolución es negativo f(x,y0 toma el mínimo de la escala
                lpImagen[nZ] = 0;}
        else { lpImagen[nZ] = (BYTE) Convo;}//de lo contrario toma un byte del resultado
                }
        }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//   Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
```

```
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosDetectarBordes()
{
        if(m_hDIB != NULL){
        double px1,px2,px3,px4,px5,px6,px7,px8,px9;
        double py1,py2,py3,py4,py5,py6,py7,py8,py9;

        CDlgBordes dlg;
        dlg.m_strTipoMascara = "Sobel";

        if (dlg.DoModal() == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

                if (dlg.m_strTipoMascara == "Gradiente"){
                        Asig_Mas(px1,px2,px3,px4,px5,px6,//asignanción de cada una de las
máscaras respectivas
                                px7,px8,px9,0,0,0,-1,1,0,0,0,0);
                        Asig_Mas(py1,py2,py3,py4,py5,py6,
                                py7,py8,py9,0,-1,0,0,1,0,0,0,0);}

                if (dlg.m_strTipoMascara == "Robert's"){
                        Asig_Mas(px1,px2,px3,px4,px5,px6,
                                px7,px8,px9,0,0,0,0,-1,0,0,0,1);
                        Asig_Mas(py1,py2,py3,py4,py5,py6,
                                py7,py8,py9,0,0,0,0,0,-1,0,1,0);}

                if (dlg.m_strTipoMascara == "Sobel"){
                        Asig_Mas(px1,px2,px3,px4,px5,px6,
                                px7,px8,px9,-0.25,0,0.25,-0.5,0,0.5,-0.25,0,0.25);
                        Asig_Mas(py1,py2,py3,py4,py5,py6,
                                py7,py8,py9,-0.25,-0.5,-0.25,0,0,0,0.25,0.5,0.25);}

                if (dlg.m_strTipoMascara == "Sobel (Diagonal)"){
                        Asig_Mas(px1,px2,px3,px4,px5,px6,
                                px7,px8,px9,-0.5,-0.25,0,-0.25,0,0.25,0,0.25,0.5);
                        Asig_Mas(py1,py2,py3,py4,py5,py6,
                                py7,py8,py9,0,-0.25,-0.5,0.25,0,-0.25,0.5,0.25,0);}
```

```
        if (dlg.m_strTipoMascara == "Prewitt"){
                Asig_Mas(px1,px2,px3,px4,px5,px6,
                        px7,px8,px9,-1,0,1,-1,0,1,-1,0,1);
                Asig_Mas(py1,py2,py3,py4,py5,py6,
                        py7,py8,py9,-1,-1,-1,0,0,0,1,1,1);}

        if (dlg.m_strTipoMascara == "Frei_Chen"){
                Asig_Mas(px1,px2,px3,px4,px5,px6,
                        px7,px8,px9,-1,0,1,-1.414213,0,1.414213,-1,0,1);
                Asig_Mas(py1,py2,py3,py4,py5,py6,
                        py7,py8,py9,-1,-1.414213,-1,0,0,0,1,1.414213,1);}

        if (dlg.m_strTipoMascara == "Laplaciano"){
                Asig_Mas(px1,px2,px3,px4,px5,px6,
                        px7,px8,px9,0,-1,0,-1,4,-1,0,-1,0);
                Asig_Mas(py1,py2,py3,py4,py5,py6,
                        py7,py8,py9,0,0,0,0,0,0,0,0,0);}

double x[] = {px1,px2,px3,px4,px5,px6,px7,px8,px9};
double y[] = {py1,py2,py3,py4,py5,py6,py7,py8,py9};

int* pImageTemp = new int[nLongiDIB];

memcpy(pImageTemp, lpImagen, nLongiDIB); //Copia de la imagen

//Cómputo sobre la imagen
//(sin incluir primera y última fila/columna)
double dMaxOfNeigh(0); //Almacena el máximo valor de la derivada
for (i = 1; i <= nAlto -2; i++){
        nY = nLongiDIB*i/nAlto;
        for (j = 1; j <= nAncho -2; j++){
                nZ = nY + j; //( nZ) Posición de Byte real
                double dSumOfNeigh = Convolution2(nZ,x,y);
                if (dSumOfNeigh > dMaxOfNeigh) dMaxOfNeigh = dSumOfNeigh;
                pImageTemp[nZ] = (int)dSumOfNeigh;
        }
}
for (int m = 0; m <= (int) nLongiDIB - 1; m++){
        if (pImageTemp[m] >= dMaxOfNeigh*0.25/2){
                lpImagen[m] = 255;}
        else {
                lpImagen[m] = (BYTE)pImageTemp[m];}
        }
nDeshacer = 260;
SetModifiedFlag(TRUE);//   Verifica si se desea guardar los cambios
UpdateAllViews(NULL);     //Actualización de la vista
}
```

```
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosFiltrosRank()
{
        if(m_hDIB != NULL){
        CDlgRank dlg;
        int nValue;
        int nMasDim;
        int nCor;

        dlg.m_StrTipos = "Mediana";
        dlg.m_StrMascara = "3×3";
        int nRetorno = dlg.DoModal();
        if (nRetorno == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        if (dlg.m_StrTipos == "Mínimo") nValue = 0;
        if (dlg.m_StrTipos == "Mediana") nValue = 1;
        if (dlg.m_StrTipos == "Máximo") nValue = 2;
        if (dlg.m_StrTipos == "Punto medio") nValue = 3;
        if (dlg.m_StrMascara == "3×3") {//definición de máscara de 3×3
                nMasDim = 3;
                nCor = 0;}
        if (dlg.m_StrMascara == "5×5") {//definición de máscara de 5×5
                nMasDim = 5;
                nCor = 1;}

        int nNumVec = nMasDim * nMasDim;
        LPBYTE pVec = new BYTE[nNumVec];
        LPBYTE pImageTemp = new BYTE[nLongiDIB];
        memcpy(pImageTemp, lpImagen, nLongiDIB); //Copia de la imagen

        //Cómputo sobre la imagen
        //(sin incluir primera y última fila/columna)

        for (i = 1 + nCor; i <= nAlto - 2 - nCor; i++){
                nY = (nLongiDIB*i)/nAlto;
        for (j = 1 + nCor; j <= nAncho - 2 - nCor; j++){
```

```
                nZ = nY + j; //(nZ) Posición de Byte real
        //Cálculo del vector "vecindad"
                int nCountOfNeigh(0);

                for (int p = - 1 - nCor; p <= 1 + nCor; p++){
                        int nYx = nZ + (nLongiDIB/nAlto) * p;
                for (int q = - 1 - nCor; q <= 1 + nCor; q++){
                        int nZx = nYx + q;
                        pVec[nCountOfNeigh] = pImageTemp[nZx];
                        nCountOfNeigh++;
                                }
                        }
                        BYTE nRet = SortArray(pVec, nNumVec, nValue);
                        lpImagen[nZ] = nRet;
                        }
                }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//   Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosFiltroPersonalizado()
{
        if(m_hDIB != NULL){
                CDlgFiltroPer dlg;
        float Temp;
        int nTemp, nZY1, nZY2, nZY3, nZY4,
                nZX1, nZX2, nZX3, nZX4;

        if(dlg.DoModal() == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
DIB
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        lpTemp = new BYTE[nLongiDIB];
        nTemp = nLongiDIB/nAlto;

        memcpy(lpTemp, lpImagen, nLongiDIB);   //Copia temporal de imagen
```

```
        for (i = 1; i <= nAlto -2; i++){
                nY = (nLongiDIB*i)/nAlto;
        for (j = 1; j <= nAncho -2; j++){
                nZ = nY + j;
                nZY1 = nZ - nTemp; nZY2 = nZ + nTemp; nZX1 = nZ - 1; //Igual que el
caso del afinado
                nZX2 = nZ + 1; nZY3 = nZY1 - 1; nZY4 = nZY1 + 1;//cada parámetro de la
máscara se multiplica por cada vecino correspondiente
                nZX3 = nZY2 - 1; nZX4 = nZY2 + 1;

                Temp = dlg.m_Elem1*lpTemp[nZY3] + dlg.m_Elem2*lpTemp[nZY1]
+//dlg.ElemX son datos ingresados por el usuario
                        dlg.m_Elem3*lpTemp[nZY4] + dlg.m_Elem4*lpTemp[nZX1] +
                        dlg.m_Elem5*lpTemp[nZ] + dlg.m_Elem6*lpTemp[nZX2] +
                        dlg.m_Elem7*lpTemp[nZX3] + dlg.m_Elem8*lpTemp[nZY2] +
                        dlg.m_Elem9*lpTemp[nZX4];
                if(Temp < 0){
                        lpImagen[nZ] = 0;}
                else if(Temp > 255){
                        lpImagen[nZ] = 255;}
                else{lpImagen[nZ] = (BYTE) Temp;}
                }
        }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//  Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);    //Actualización de la vista
        }
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosFiltrosFrecuencia()
{
        double Max, Coef, Denom, Hij;
        int nAncho2, nAlto2, nZ1, nZ2, nZ3, nZ4, nOffset;
        CDlgFiltroFrec dlg;
        if(dlg.DoModal()==IDOK){
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);       //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);
        double* pdTemp = new double[nLongiDIB];

        for (i = 0; i <= nAlto/2 - 1; i++){
                nY = 2*nAncho*i;
```

```
                nOffset = i + 1;

                for (j = 0; j <= nAncho/2 - 1; j++){

                        nZ1 = nY + 2*j;//Se evalúa cada pixel que se encuentra formando un
círculo con el mismo
                        nZ2 = nY - 2*j + 2*nAncho - 2;//coeficiente dado por el filtro
                        nZ3 = 2*nLongiDIB - 2*nOffset*nAncho + 2*j;
                        nZ4 = 2*nLongiDIB - 2*j - 2*(nOffset - 1)*nAncho - 2;

                        if(dlg.m_nTipoFiltro == 0){
                        Denom = sqrt((double) (j*j + i*i))/(dlg.m_FrecCorte);//Por definición
del filtro tipo Butterworth
                        Denom = 1 + pow(Denom, 2*dlg.m_Orden);//pasa bajos
                        Hij = 1/sqrt(Denom);}

                        else{
                                if(i==0 && j==0){
                                        Denom = 10000000000;}
                                else {Denom = (dlg.m_FrecCorte)/(sqrt((double) (j*j +
i*i)));}
                        Denom = 1 + pow(Denom, 2*dlg.m_Orden);//Butterworth pasa altos
                        Hij = 1/sqrt(Denom);}

                FFTran[nZ1] *= Hij; //Cada dato real e imaginario se multiplica por
                FFTran[nZ1 + 1] *= Hij;//el coeficiente dado por el filtro para cada
                FFTran[nZ2] *= Hij;//una de las frecuencias especificadas y además en
virtud de la
                FFTran[nZ2 + 1] *= Hij;//frecuencia de corte
                FFTran[nZ3] *= Hij;
                FFTran[nZ3 + 1] *= Hij;
                FFTran[nZ4] *= Hij;
                FFTran[nZ4 + 1] *= Hij;

                        }
                }

        nAlto2 = nAlto/2; nAncho2 = nAncho/2;
        Max = 0.0;
//A partir de estos lazos se produce el almacenamiento de cada componente del
//espectro de Fourier
        for (i = 0; i <= nAlto - 1; i++){
                nY = (nLongiDIB*i)/nAlto;

        for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;
```

```
                pdTemp[nLongiDIB - (nLongiDIB*(i+1))/nAlto + j] = sqrt((double)
FFTran[nY*2 +2*j]*FFTran[nY*2 +2*j] + FFTran[nY*2 +2*j+1]*FFTran[nY*2 +2*j+1]);
                if(pdTemp[nZ] > Max){
                        Max = pdTemp[nZ];}
                }
        }
//Para la presentación de los datos se ha definido una escala logarítmica
        Coef = 255/log(1.0+Max);
//A continuacíon se presenta la rutina que se encarga de trasladar los cuadrantes
//que componen la imagen en forma diagonal de forma tal que el dato de F(0,0) se
posicione
//en el centro del cuadrado en el que aparece la imagen del espectro
        for(i = 0; i <= nAlto2 - 1; i++){
                nY = (nLongiDIB*i)/nAlto;
                for(j = 0; j <= nAncho2 - 1; j++){
                        nZ = nY + j;
                        lpImagen[nZ] =
                                (BYTE) (0.5 + log((double) (1 + pdTemp[nZ + nLongiDIB/2
+ nAncho2]))*Coef);
                }
                for(j = nAncho2; j <= nAncho - 1; j++){
                        nZ = nY + j;
                        lpImagen[nZ] =
                                (BYTE) (0.5 + log((double) (1 + pdTemp[nZ + nLongiDIB/2
- nAncho2]))*Coef);
                        }
                }

        for(i = nAlto2; i <= nAlto - 1; i++){
                        nY = (nLongiDIB*i)/nAlto;
                for(j = 0; j <= nAncho2 - 1; j++){
                        nZ = nY + j;
                        lpImagen[nZ] =
                                (BYTE) (0.5 + log((double) (1 + pdTemp[nZ - nLongiDIB/2
+ nAncho2]))*Coef);
                }
                for(j = nAncho2; j <= nAncho - 1; j++){
                        nZ = nY + j;
                        lpImagen[nZ] =
                                (BYTE) (0.5 + log((double) (1 + pdTemp[nZ - nLongiDIB/2 -
nAncho2]))*Coef);
                }
        }
        UpdateAllViews(NULL);
        }
}
```

```
void CSPIDDoc::OnProcesosMorfologiaImagen()
{
        if(m_hDIB != NULL){
        CDlgMorfImagen dlg;
        int nTemp, nZY1, nZY2, nZX1, nZX2, a, b;
        BYTE Temp1, Temp2, Temp3, Temp4, Temp5;

        int nResponse = dlg.DoModal();
        if(nResponse == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);         //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        lpTemp = new BYTE[nLongiDIB];
        LPBYTE pdTemp = new BYTE[nLongiDIB];

        nTemp = nLongiDIB/nAlto;

        if(dlg.m_nTipoMorfo==4){
        memcpy(pdTemp, lpImagen, nLongiDIB);  //Copia temporal de imagen
        }

        for(a = 1; a <= dlg.m_nIteracion; a++){
        memcpy(lpTemp, lpImagen, nLongiDIB);   //Copia temporal de imagen

        for (i = 1; i <= nAlto -2; i++){
                nY = (nLongiDIB*i)/nAlto;
        for (j = 1; j <= nAncho -2; j++){
                nZ = nY + j;
                nZY1 = nZ - nTemp;
                nZY2 = nZ + nTemp;
                nZX1 = nZ - 1;
                nZX2 = nZ + 1;

                Temp1 = lpTemp[nZY1];//Extracción de los cinco vecinos y con los
                Temp2 = lpTemp[nZY2];//que las multiplicaciones con cada elemento de la
                Temp3 = lpTemp[nZ];//máscara es probable que no sea nula
                Temp4 = lpTemp[nZX1];
                Temp5 = lpTemp[nZX2];

        if(dlg.m_nTipoMorfo == 0){//Erosión binaria
        if(Temp1==0||Temp2==0||Temp3==0||Temp4==0||Temp5==0){
```

```
            lpImagen[nZ] = 0;}
        else{lpImagen[nZ] = 255;}}

        else if(dlg.m_nTipoMorfo == 1 || dlg.m_nTipoMorfo == 4){//Dilatación o controno
de imagen
            if(Temp1==255||Temp2==255||Temp3==255||Temp4==255||Temp5==255){
            lpImagen[nZ] = 255;}
            else{lpImagen[nZ] = 0;}}

        else if(dlg.m_nTipoMorfo == 2){//Cerradura binaria
        b = dlg.m_nIteracion;
        dlg.m_nIteracion *= 2;
        if(a <= b){
            if(Temp1==255||Temp2==255||Temp3==255||Temp4==255||Temp5==255){
            lpImagen[nZ] = 255;}
            else{lpImagen[nZ] = 0;}      }
        else if(a > b){
            if(Temp1==0||Temp2==0||Temp3==0||Temp4==0||Temp5==0){
            lpImagen[nZ] = 0;}
            else{lpImagen[nZ] = 255;}
            }}

        else if(dlg.m_nTipoMorfo == 3){//Apertura binaria
        b = dlg.m_nIteracion;
        dlg.m_nIteracion *= 2;
        if(a <= b){
            if(Temp1==0||Temp2==0||Temp3==0||Temp4==0||Temp5==0){
            lpImagen[nZ] = 0;}
            else{lpImagen[nZ] = 255;}
            }
        else if(a > b){
            if(Temp1==255||Temp2==255||Temp3==255||Temp4==255||Temp5==255){
            lpImagen[nZ] = 255;}
            else{lpImagen[nZ] = 0;}
            }}
            }       }       }
        if(dlg.m_nTipoMorfo == 4){//Controno de imagen
        for (i = 0; i <= nAlto - 1; i++){
            nY = (nLongiDIB/nAlto)*i;

        for (j = 0; j <= nAncho - 1; j++){
            nZ = nY + j;
            if(lpImagen[nZ] == pdTemp[nZ]){
                lpImagen[nZ] = 0;}
            else {lpImagen[nZ] = 255;}
            }
        }
```

```
        }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//   Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosMorfNivelgris()
{
        if(m_hDIB != NULL){
        CDlgMorfNiveles dlg;
        int nMayor, nMenor, nTemp, nZY1, nZY2, nZY3, nZY4,
                nZX1, nZX2, nZX3, nZX4, a, b;

        if(dlg.DoModal() == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        lpTemp = new BYTE[nLongiDIB];
        int* pdTemp = new int[9];
        nTemp = nLongiDIB/nAlto;
        b = dlg.m_nIteracion;

        if(dlg.m_nTipoMorf == 1 || dlg.m_nTipoMorf == 3){//Para apertura o cerradura en
niveles de gris
                dlg.m_nIteracion *= 2;}

        for(a = 1; a <= dlg.m_nIteracion; a++){//Lazo para las iteraciones
        memcpy(lpTemp, lpImagen, nLongiDIB);   //Copia temporal de imagen

        for (i = 1; i <= nAlto -2; i++){
                nY = (nLongiDIB*i)/nAlto;
        for (j = 1; j <= nAncho -2;){
                nZ = nY + j;
                nZY1 = nZ - nTemp; nZY2 = nZ + nTemp; nZX1 = nZ - 1; //Para
erosionado el pixel se resta a
                nZX2 = nZ + 1; nZY3 = nZY1 - 1; nZY4 = nZY1 + 1;//la máscara
respectivamente
                nZX3 = nZY2 - 1; nZX4 = nZY2 + 1;
```

```
            if(dlg.m_nTipoMorf == 0){ //Erosionado en niveles de gris
Eros:   pdTemp[0] = lpTemp[nZY3] - 1; pdTemp[1] = lpTemp[nZY1] - 2;
            pdTemp[2] = lpTemp[nZY4] - 1; pdTemp[3] = lpTemp[nZX1] - 2;
            pdTemp[4] = lpTemp[nZ] - 3;   pdTemp[5] = lpTemp[nZX2] - 2;
            pdTemp[6] = lpTemp[nZX3] - 1; pdTemp[7] = lpTemp[nZY2] - 2;
            pdTemp[8] = lpTemp[nZX4] - 1;

            nMenor = 255;
            for(int m = 0; m <= 8; m++){
                  if(pdTemp[m] < nMenor){//Determinación del menor de los
resultados de la vecindad
                        nMenor = pdTemp[m];}
            }
            if(nMenor < 0){
                  lpImagen[nZ] = 0;}
            else{lpImagen[nZ] = (BYTE) nMenor;}

            if(dlg.m_nTipoMorf == 1 || dlg.m_nTipoMorf == 3){
                  goto Continuar;}
      }

      if(dlg.m_nTipoMorf == 2){ //Dilatado en niveles de gris
Dilat:  pdTemp[0] = lpTemp[nZY3] + 1; pdTemp[1] = lpTemp[nZY1] + 2;//Definición de
la máscara
            pdTemp[2] = lpTemp[nZY4] + 1; pdTemp[3] = lpTemp[nZX1] + 2;
            pdTemp[4] = lpTemp[nZ] + 3;   pdTemp[5] = lpTemp[nZX2] + 2;
            pdTemp[6] = lpTemp[nZX3] + 1; pdTemp[7] = lpTemp[nZY2] + 2;
            pdTemp[8] = lpTemp[nZX4] + 1;

            nMayor = 0;
            for(int m = 0; m <= 8; m++){
                  if(pdTemp[m] > nMayor){//Determinación del mayor de los
resultados de la vecinadad
                        nMayor = pdTemp[m];}
            }
            if(nMayor > 255){
                  lpImagen[nZ] = 255;}
            else{lpImagen[nZ] = (BYTE) nMayor;}

            if(dlg.m_nTipoMorf == 1 || dlg.m_nTipoMorf == 3){
                  goto Continuar;}
      }

      if(dlg.m_nTipoMorf == 1){ //Morfología apertura
            if(a <= b){ goto Eros;}
            else { goto Dilat;}
```

```
        }

        if(dlg.m_nTipoMorf == 3){ //Morfología cerradura
                if(a <= b){ goto Dilat;}
                else { goto Eros;}
        }

 Continuar:  j++;}
        }
        }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//   Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosAritmeticaPix()
{
        if(m_hDIB != NULL){
        CDlgOperAritme dlg;
        float Temp;

        if(dlg.DoModal() == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        for (i = 0; i <= nAlto - 1; i++){
                nY = (nLongiDIB/nAlto)*i;

        for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;

        switch(dlg.m_nOper){
        case 0://suma a nivel de pixel y constante
                Temp = lpImagen[nZ] + (50*dlg.m_Constante);
                if(Temp > 255){lpImagen[nZ] = 255;}
                else{lpImagen[nZ] = (BYTE)Temp;}
                break;
        case 1://Resta a nivel de pixel y constante
```

```
                Temp = lpImagen[nZ] - (50*dlg.m_Constante);
                if(Temp < 0){
                        lpImagen[nZ] = 0;}
                else{lpImagen[nZ] = (BYTE) Temp;}
                break;
        case 2://Multiplicación a nivel de pixel y constante
                Temp = lpImagen[nZ]*(dlg.m_Constante);
                if(Temp > 255){lpImagen[nZ] = 255;}
                else{lpImagen[nZ] = (BYTE)Temp;}
                break;
        case 3://División a nivel de pixel y constante
                Temp = lpImagen[nZ]/(dlg.m_Constante);
                if(Temp > 255){lpImagen[nZ] = 255;}
                else{lpImagen[nZ] = (BYTE)Temp;}
                break;
                                }
                        }
                }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);//   Verifica si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosLogicaPix()
{
        if(m_hDIB != NULL){
        CDlgOperLogic dlg;

        if(dlg.DoModal() == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        for (int i = 0; i <= nAlto - 1; i++){
                nY = (nLongiDIB/nAlto)*i;

        for (int j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;
```

```
        switch(dlg.m_nOperacion){

        case 0://AND a nivel de pixel y constante
                lpImagen[nZ] &= dlg.m_ValorOp;
                break;

        case 1://OR a nivel de pixel y constante
                lpImagen[nZ] |= dlg.m_ValorOp;
                break;
        case 2://X-OR a nivel de pixel y constante
                lpImagen[nZ] ^= dlg.m_ValorOp;
                break;
                                }
                        }
                }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);// Hace que se verifique si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosImagenPatron()
{
        if(m_hDIB != NULL){
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        lpPatron = new BYTE[nLongiDIB];
        memcpy(lpPatron, lpImagen, nLongiDIB);//lpPatron guarda los datos de la imagen
patrón
        nLocal = -25;
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosAritmeticaImagen()
{
        if(m_hDIB != NULL){
        CDlgArimeImage dlg;
        int Temp;

        if(nLocal == -25){
                if(dlg.DoModal() == IDOK){
```

```
                    CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);         //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        for (i = 0; i <= nAlto - 1; i++){
                nY = (nLongiDIB*i)/nAlto;

        for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;

                switch(dlg.m_nOper){
                case 0://Suma entre la imagen actual y la patrón
                Temp = lpPatron[nZ] + lpImagen[nZ];
                if(Temp>255){
                        lpImagen[nZ] = 255;}
                else{ lpImagen[nZ] = (BYTE)Temp;}
                break;

                case 1://Resta entre la imagen actual y la patrón
                Temp = lpImagen[nZ] - lpPatron[nZ];
                if(Temp < 0){
                        Temp *= -1;
                        lpImagen[nZ] = (BYTE) Temp;}
                else{ lpImagen[nZ] = (BYTE) Temp;}
                break;
        }               }
        } nLocal = 300;
        nDeshacer = 260;
        SetModifiedFlag(TRUE);// Hace que se verifique si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
                }
        }

        else{
                AfxMessageBox("¡Necesita una imagen patrón!");}
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosLogicaImagen()
{
        if(m_hDIB != NULL){
```

```
        CDlgLogicImage dlg;
        if(nLocal==-25){
        if(dlg.DoModal() == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        for (i = 0; i <= nAlto - 1; i++){
                nY = (nLongiDIB*i)/nAlto;

        for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;

        switch(dlg.m_nOper){
        case 0://AND entre la imagen actual y la patrón
                lpImagen[nZ] &= lpPatron[nZ];
                break;
        case 1://OR entre la imagen actual y la patrón
                lpImagen[nZ] |= lpPatron[nZ];
                break;
        case 2://X-OR entre la imagen actual y la patrón
                lpImagen[nZ] ^= lpPatron[nZ];
                break;
                                }
                        }
                }       nLocal = 300;
        nDeshacer = 260;
        SetModifiedFlag(TRUE);// Hace que se verifique si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        } else{
                AfxMessageBox("¡Necesita una imagen patrón!");}
        }
        else{NoImagenAct();}

}

/*Básicamente el algortimo utilizado en la FFT e IFFT, se encarga de
realizar la transformada primero a nivel de filas y luego a nivel de
columnas, esto es posible gracias a la propiedad de separibilidad que
presenta dicha transformada*/
```

```
void CSPIDDoc::OnProcesosFft()
{
	double mPrueba; int nPrueba;

	if(m_hDIB != NULL){

	lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
	nAlto = ::DIBHeight(lpDIB);//Alto de imagen
	nAncho = ::DIBWidth(lpDIB);	//Ancho de imagen
	nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación

	mPrueba = (log(nAlto)/log(2));
	nPrueba = (int)mPrueba;
	mPrueba -= nPrueba;

	if(nAlto == nAncho && mPrueba == 0){ //Prueba de que las dimensiones sean
potencia de 2
//y además de que sean dimensiones exactamente iguales
	if(nFFT==567){AfxMessageBox("Necesita una imagen cuya función es real");}
	else{
	OnEditarReflejarHorizontal();//Primero se refleja horizontalmete la imagen
//para que los datos sean ordenados correctamente en la transformada
	OnFFT();//Llama la función que realiza la transformada Fourier directa de la
imagen
	nFFT = 567;//Ayuda en la identificación de una imagen representa en el espacio o
en
//la frecuencia, es el mismo caso para el algortimo siguiente
	nDeshacer = 456;
	UpdateAllViews(NULL);
			}
		}
	else{AfxMessageBox("Necesita una imagen cuadrada con dimensiones potencia de
dos, por ejemplo: 64×64, 128×128, 512×512, etc.");}
	}
	else{NoImagenAct();}
}

#define SWAP(a,b) tempr = (a); (a) = (b); (b) = tempr//Función global para la aplicación
del
//bit reversion

void CSPIDDoc::dfour1(double* data, unsigned long nn, int isign)//Función que hace la
FFT e IFFT en una dimensión
{
	unsigned long n,mmax,m,j,istep,i;
	double wtemp,wr,wpr,wpi,wi,theta;
```

```
        double tempr,tempi;

        n=nn << 1;
        j=1;
        for (i=1;i<n;i+=2) {
                if (j > i) {
                        SWAP(data[j],data[i]);//Se efectúa un intercambio de datos en las
                        SWAP(data[j+1],data[i+1]);//posiciones especificadas
                }
                m=n >> 1;
                while (m >= 2 && j > m) {
                        j -= m;
                        m >>= 1;
                }
                j += m;
        }
        mmax=2;
        while (n > mmax) {
                istep=mmax << 1;
                theta=isign*(6.28318530717959/mmax);//A partir de esta
                wtemp=sin(0.5*theta);//instrucción se lleva a cabo el cálculo
                wpr = -2.0*wtemp*wtemp;//de cada coeficiente de los datos del vector
                wpi=sin(theta);
                wr=1.0;
                wi=0.0;
                for (m=1;m<mmax;m+=2) {
                        for (i=m;i<=n;i+=istep) {//Iteraciones sucesivas para obtener la
transformada undimensional
                                j=i+mmax;
                                tempr=wr*data[j]-wi*data[j+1];
                                tempi=wr*data[j+1]+wi*data[j];
                                data[j]=data[i]-tempr;
                                data[j+1]=data[i+1]-tempi;
                                data[i] += tempr;
                                data[i+1] += tempi;
                        }
                        wr=(wtemp=wr)*wpr-wi*wpi+wr;
                        wi=wi*wpr+wtemp*wpi+wi;
                }
                mmax=istep;
        }
}

#undef SWAP

void CSPIDDoc::OnFFT()//Función que ejecuta la FFT
{
```

```
        int nAlto2, nAncho2;
        double Max, Coef;

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);        //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        double* FFT = new double[2*nAncho];
        double* pdTemp = new double[nLongiDIB];
        FFTran = new double[2*nLongiDIB];

        for (i = 0; i <= nAlto - 1; i++){
                nY = nAncho*i;

                for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;
                FFT[2*j + 1] = lpImagen[nZ];
                FFT[(1+j)*2] = 0;
                }
                dfour1(FFT, nAncho, -1);//Primera transformación mediante la evaluación
de
                for(j=1; j<=2*nAncho; j++){//cada una de las filas que componen la imagen
                FFTran[j + 2*nAncho*i - 1]=FFT[j];}
        }

        for (j = 0; j <= nAncho - 1; j++){

                for (i = 0; i <= nAlto - 1; i++){
                nY = 2*nAncho*i;
                nZ = nY + 2*j;

                        FFT[2*i + 1] = FFTran[nZ];
                        FFT[(1+i)*2] = FFTran[nZ + 1];
                }
                dfour1(FFT, nAncho, -1);//Segunda transformación de acuerdo a las
columnas
//componentes de la imagen
                for (i = 0; i <= nAlto - 1; i++){
                nY = 2*nAncho*i;
                nZ = nY + 2*j;

                FFTran[nZ] = FFT[2*i + 1];
                FFTran[nZ + 1] = FFT[(1+i)*2];
                }
```

```
}

        nAlto2 = nAlto/2; nAncho2 = nAncho/2;
        Max = 0.0;

        for (i = 0; i <= nAlto - 1; i++){
                nY = (nLongiDIB*i)/nAlto;

        for (j = 0; j <= nAncho - 1; j++){

                pdTemp[nLongiDIB - (nLongiDIB*(i+1))/nAlto + j] = sqrt((double)
FFTran[nY*2 +2*j]*FFTran[nY*2 +2*j] + FFTran[nY*2 +2*j+1]*FFTran[nY*2 +2*j+1]);

                if(pdTemp[nLongiDIB - (nLongiDIB*(i+1))/nAlto + j] > Max){
                        Max = pdTemp[nLongiDIB - (nLongiDIB*(i+1))/nAlto + j];}
                }
        }

        Coef = 255/log(1.0+Max);
//A continuacíon se presenta la rutina que se encarga de trasladar los cuadrantes
//que componen la imagen en forma diagonal de forma tal que el dato de F(0,0) se
posicione
//en el centro del cuadrado en el que aparece la imagen del espectro y la adecuación de cada
//uno de los datos a la escala de 0-255 en niveles de gris
        for(i = 0; i <= nAlto2 - 1; i++){
                nY = (nLongiDIB*i)/nAlto;
                for(j = 0; j <= nAncho2 - 1; j++){
                        nZ = nY + j;
                        lpImagen[nZ] =
                                (BYTE) (0.5 + log((double) (1 + pdTemp[nZ + nLongiDIB/2
+ nAncho2]))*Coef);
                }
                for(j = nAncho2; j <= nAncho - 1; j++){
                        nZ = nY + j;
                        lpImagen[nZ] =
                                (BYTE) (0.5 + log((double) (1 + pdTemp[nZ + nLongiDIB/2
- nAncho2]))*Coef);
                        }
                }

        for(i = nAlto2; i <= nAlto - 1; i++){
                        nY = (nLongiDIB*i)/nAlto;
                for(j = 0; j <= nAncho2 - 1; j++){
                        nZ = nY + j;
                        lpImagen[nZ] =
                                (BYTE) (0.5 + log((double) (1 + pdTemp[nZ - nLongiDIB/2
+ nAncho2]))*Coef);
```

```
                }
                for(j = nAncho2; j <= nAncho - 1; j++){
                        nZ = nY + j;
                        lpImagen[nZ] =
                                (BYTE) (0.5 + log((double) (1 + pdTemp[nZ - nLongiDIB/2 -
nAncho2]))*Coef);
                }
        }
}

void CSPIDDoc::OnProcesosFftInversa()
{
        if(m_hDIB != NULL){
                if(nFFT==567){
                OnFFTInversa();//Llamado a la función que ejecuta la IFFT
                nFFT = 1525;
                OnEditarReflejarHorizontal();//Para el ordenamiento inverso de los datos
                nDeshacer = 456;
                }

                else{AfxMessageBox("Necesita una imagen cuya función es compleja");}
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnFFTInversa()//Esta función obtiene la IFFT
{
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);         //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        double* FFT = new double[2*nAncho];

        for (i = 0; i <= nAlto - 1; i++){
                nY = nAncho*i;
                for (j = 0; j <= nAncho - 1; j++){
                FFT[2*j+1] = FFTran[2*nY + 2*j];
                FFT[(1+j)*2] = FFTran[2*nY + 2*j + 1];
                }
                dfour1(FFT, nAncho, 1);
                for(j=1; j<=2*nAncho; j++){
                FFTran[j + 2*nAncho*i - 1]=FFT[j];}
        }
```

```
        for (j = 0; j <= nAncho - 1; j++){

                for (i = 0; i <= nAlto - 1; i++){
                nY = 2*nAncho*i;
                nZ = nY + 2*j;

                        FFT[2*i + 1] = FFTran[nZ];
                        FFT[(1+i)*2] = FFTran[nZ + 1];
                }
                dfour1(FFT, nAncho, 1);//Transformación de las filas
//La función anterior es la misma que se usa para la FFT, la variante
//es que se le debe pasar el parámetro de +1 para la FFT y -1 para la IFFT
                for (i = 0; i <= nAlto - 1; i++){
                nY = 2*nAncho*i;
                nZ = nY + 2*j;

                FFTran[nZ] = FFT[2*i + 1];//Retorno de datos en columnas así para
                FFTran[nZ + 1] = FFT[(1+i)*2];//todos los casos anteriores y similares
                }
        }

        for (i = 0; i <= nAlto - 1; i++){
                nY = nAncho*i;
        for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;
                FFTran[2*nY + 2*j] /= nLongiDIB;
                lpImagen[nZ] = (BYTE) FFTran[2*nY + 2*j];
                }//Esta porción del programa se encarga de normalizar los datos reales para
luego ser
        }//presentados
}

void CSPIDDoc::OnProcesosEcualizado()
{
        if(m_hDIB != NULL){
        double Temp3 = 0, Temp2;
        UINT Temp1;
        int nTemp, Temp4;

                CopiaImagen();
        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);         //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);
```

```
        Temp1 = ::DIBNumColors(lpDIB); //Retorna la escala de gris de trabajo
        pHistogram = new double[Temp1];  //TablaColor() es una función que retorna
        Temp1--;                                        //la escala de gris utilzada

        for (UINT k = 0; k <= Temp1; k++){ //Inicialización del arreglo
                pHistogram[k] = 0;    //que almacena los valores de histograma
        }

        for (i = 0; i <= nAlto -1; i++){
                nY = (nLongiDIB*i)/nAlto;

        for (j = 0; j <= nAncho -1; j++){
                nZ = nY + j;
                nTemp = lpImagen[nZ];
                pHistogram[nTemp] += 1;
                }
        }

        for (k = 0; k <= Temp1; k++){

                Temp3 += pHistogram[k]/(nAlto*nAncho); //Suma acumulada para
asignarle el
                Temp2 = Temp3*Temp1;            //nuevo valor a cada f(x,y)
                Temp4 = (int)Temp2;
                if(Temp2 - Temp4 >= 0.5){
                pHistogram[k] = Temp4 + 1;}
                else{pHistogram[k] = Temp4;}
        }

        for (i = 0; i <= nAlto -1; i++){
                nY = (nLongiDIB*i)/nAlto;
        for (j = 0; j <= nAncho -1; j++){
                nZ = nY + j;
                lpImagen[nZ] = (BYTE) pHistogram[lpImagen[nZ]];
                }
        }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);// Hace que se verifique si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        else{NoImagenAct();}
}

void CSPIDDoc::OnProcesosBalanceImagen()
{
        if(m_hDIB != NULL){
        CDialogBalanceImagen dlg;
```

```
        BYTE Temp1 = 0, Temp2 = 255;
        double Temp3;
        int B = Promedio();

        if(dlg.DoModal() == IDOK){
                CopiaImagen();

        lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB); //Definición del puntero al
encabezado de archivo
        nAlto = ::DIBHeight(lpDIB);//Alto de imagen
        nAncho = ::DIBWidth(lpDIB);         //Ancho de imagen
        nLongiDIB = ::DIBSizeImage(lpDIB);//Tamaño con justificación
        lpImagen = (LPBYTE)::FindDIBBits(lpDIB);

        if(dlg.m_Automatico == 1 && dlg.m_nFuncion==0){
        for (i = 0; i <= nAlto - 1; i++){
                nY = (nLongiDIB*i)/nAlto;

        for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;

                if(lpImagen[nZ] > Temp1){  //Cálculo de los valores
                Temp1 = lpImagen[nZ];}              //máximos y mínimos en la imagen

                else if(lpImagen[nZ] < Temp2){
                        Temp2 = lpImagen[nZ];}
                }
        }
        }
        for (i = 0; i <= nAlto - 1; i++){
                nY = (nLongiDIB*i)/nAlto;

        for (j = 0; j <= nAncho - 1; j++){
                nZ = nY + j;

        if(dlg.m_Automatico == 1 && dlg.m_nFuncion==0){
                Temp3 = (255/(Temp1 - Temp2))*(lpImagen[nZ] - Temp2); //Función de
transferencia lineal
                if(Temp3 > 255){ lpImagen[nZ] = 255;}
                else if(Temp3 < 0){lpImagen[nZ] = 0;}
                else{ lpImagen[nZ] = (BYTE) Temp3;}      }

        else if(dlg.m_Automatico==1 && dlg.m_nFuncion==1){
                Temp3 = (lpImagen[nZ]*lpImagen[nZ])/255;//Función de trasferencia
cuadrática
                lpImagen[nZ] = (BYTE) Temp3;
        }
```

```
        else if(dlg.m_Automatico==1 && dlg.m_nFuncion==2){
                Temp3 = (lpImagen[nZ]*lpImagen[nZ]*lpImagen[nZ])/65025;
                lpImagen[nZ] = (BYTE) Temp3;//Función de transferencia cúbica
        }

        else if(dlg.m_Automatico==1 && dlg.m_nFuncion==3){
                Temp3 = 15.969*sqrt(lpImagen[nZ]);//Función de transferencia raíz
cuadrada
                lpImagen[nZ] = (BYTE) Temp3;
        }

        else if(dlg.m_Automatico==1 && dlg.m_nFuncion==4){
        Temp3 = 40.2124*pow(lpImagen[nZ], 0.33333);//Función de transferencia raíz
cúbica
                lpImagen[nZ] = (BYTE) Temp3;
        }

        else if(dlg.m_Automatico==1 && dlg.m_nFuncion==5){
                Temp3 = exp(lpImagen[nZ]/45.98 - 1);//Función de transferencia
exponencial
                lpImagen[nZ] = (BYTE) Temp3;
        }

        else if(dlg.m_Automatico==1 && dlg.m_nFuncion==6){
                Temp3 = 45.98*log(lpImagen[nZ] + 1);//Función de transferencia
logarítmica
                lpImagen[nZ] = (BYTE) Temp3;
        }

        else{
        Temp3 = dlg.m_Contraste*(lpImagen[nZ] - B)/(50) + B + (2*dlg.m_Brillo) - 100;
        if (Temp3 > 255){//Balance de imagen manual
        lpImagen[nZ] = 255;}
        else if (Temp3 < 0){
        lpImagen[nZ] = 0;}
        else {
                lpImagen[nZ] = (BYTE) Temp3;}}
                        }
                }
        nDeshacer = 260;
        SetModifiedFlag(TRUE);// Hace que se verifique si se desea guardar los cambios
        UpdateAllViews(NULL);     //Actualización de la vista
        }
        }
        else{NoImagenAct();}
}
```