6

Universidad Nacional Autónoma de México

FACULTAD DE INGENIERIA

PROCESAMIENTO GRAFICO EN ANGIOLOGIA RETINIANA (S.P.G.A.R.)

T E S I S

Que para obtener el Título de
INGENIERO EN COMPUTACION
P r e s e n t a n

Sergio Avendaño Bonilla
Aldo Granados Pérez

Director de tesis: Ing. Luís Beltrán del Río.

México, D.F.

1991

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

PREFACIO 1

CAPITULO UNO INTRODUCCION Y DESCRIPCION DEL PROBLEMA ... 3

1.1 Procesamiento Digital 4
1.1.1. Qué es el procesamiento de imágenes ?..... 4
1.1.2. Historia 6
1.1.3. Bases del procesamiento digital de imágenes . 7
1.1.3.1. Clasificación de las operaciones sobre imágenes 8
1.1.4. Análisis de imagen 10
1.1.5. Codificación de imágenes 11
1.1.6. Clasificación de procesamiento de imágenes .. 11
1.1.7. El sistema de procesamiento 13
1.1.8. Formato de una imagen digital 15
1.1.9. Tipos de resolución 16
1.1.10. Resolución espacial 17
1.1.11. Resolución de brillantez 18

1.2 La Fluoresceingrafía Antecedentes y Conceptos 20
1.2.1. Principios fundamentales 21
1.2.2. Técnica 21

CAPITULO DOS TENDENCIAS ACTUALES DEL PROCESAMIENTO DIGITAL DE IMAGENES 26

2.1. Formatos 27
2.1.1. Formato Inovión 27
2.1.2. Formato PCX 28
2.1.2.1. Método de compresión Run Length 29
2.1.3. Formato GIF 31
2.1.4. Formato TIFF 35
2.1.4.1. Método de compresión LZW 39

2.2. Hardware 41
2.2.1. Tarjetas gráficas 42
2.2.2. Dispositivos de obtención de imágenes 47

CAPITULO TRES ELECCION Y DESCRIPCION DE RECURSOS DE COMPUTO 51

3.1. Introducción 51

3.2. Módulo de obtención y digitalización de imágenes . 52
3.2.1. Cámara de video 53
3.2.2. Monitor de T.V. 53
3.2.3. Sistema Electrónico de digitalización 54
3.2.4. Microcomputadora PC-AT 55

3.3. Software externo 55

3.4. Equipo de cómputo para almacenamiento y procesamiento 56

3.5. Tarjeta gráfica para desplegar imágenes (VGA) 57

3.6. Lenguaje de programación 65

CAPITULO CUATRO SISTEMA ADMINISTRADOR 69

4.1. Descripción 69

4.2. Presentación 71

4.3. Modulos Help y Mouse 73

4.4. Lectura, Escritura y Almacenamiento de Imágenes .. 73
4.4.1. Rutinas de Lectura y Desplegado de Imágenes . 74
4.4.2. Rutinas de Almacenamiento de Imágenes a Disco 75

4.5. Falsos Colores 78

4.6. Tonos de gris 79

4.7. Editor 80

4.8. Histograma 83

4.9 Perspectivas 83

CAPITULO CINCO ANALISIS Y DESARROLLO DE ALGORITMOS 86

5.1. Algoritmos de eliminación de ruido 86
5.1.1. Algoritmo de selección de venas y arterias .. 87
5.1.2. Algoritmo de eliminación de grises 91

5.2. Algoritmo de cálculo de áreas 91

5.3. Algoritmo de longitud de vasos sanguíneos 98

CONCLUSIONES 104

APENDICE A..... 109

BIBLIOGRAFIA 123

PREFACIO

El procesamiento digital de imágenes es un campo fascinante de la computación, es una herramienta importante en la investigación científica, y dada su versatilidad y su gran aceptación se ha hecho popular en disciplinas humanísticas como en arte, restauración de documentos y otras más; en actividades comerciales: publicidad y presentaciones.

Anteriormente esta técnica era utilizada solamente para el estudio de imágenes espaciales, en aplicaciones de factores de control automático, robótica, seguridad, etc.

Actualmente se ha dado una aplicación de la técnica a la rama de la medicina, principalmente en radiología, citología, en la medicina nuclear, y en los últimos años en el estudio del cerebro y las enfermedades mentales así como en la oftalmología.

En la oftalmología existe una técnica para la obtención de fotografías del fondo ocular llamada *fluorescingrafía* (técnica que consiste en fotografiar fondo ocular irrigado con un colorante fluorescente llamado "flouresceina" que permite resaltar venas y arterias) -se explicará más a detalle posteriormente-. La técnica será utilizada para la obtención de las imágenes que servirán de entrada al sistema que desarrollaremos, el cual proporcionará parámetros geométricos útiles para el diagnóstico.

La tesis estará contenida en cinco capítulos. El primero de ellos sirve de introducción a los conceptos médicos y de procesamiento gráfico. Contendrá además la descripción del problema. El segundo cápitulo hablará del equipo, formatos y tendencias del procesamiento digital. En el tercer capítulo se proponen alternativas de software y hardware para la realización del sistema. En el cuarto capítulo desarrollaremos el sistema administrador de imágenes. En el último capítulo se implementarán los algoritmos capaces de obtener los parámetros geométricos útiles en el diagnóstico. Terminaremos el trabajo con una conclusión general del sistema.

CAPITULO UNO

INTRODUCCION Y DESCRIPCION DEL PROBLEMA

En la actualidad ninguna disciplina puede desempeñar sus actividades sin relacionarse con otras, por ejemplo: para que una obra civil sea realizada adecuadamente, es necesario, que interactuen diferentes profesionistas, Ingenieros, Arquitectos, Abogados, Economistas, etc..

La medicina no podía ser la excepción a la afirmación anterior. Actualmente existen sistemas expertos realizados por ingenieros y doctores cuya finalidad es sustituir en casos de emergencia al doctor, siendo el sistema el encargargado de diagnosticar y dar un tratamiento.

En Angiología, para el estudio de los vasos del fondo de ojo humano, se auxilia de la fluorescengrafía -que será explicada a detalle posteriormente-. La técnica proporciona fotografías, realzando en ellas las venas, arterias y patologías para un análisis gráfico. Sin un sistema automático el análisis es lento e impreciso. Por lo cual se pretende elaborar un sistema de cómputo, cuyo propósito es obtener un conjuto de parámetros geométricos, que permita al oftalmólogo realizar el análisis de manera rápida y confiable.

El trabajo que se desarrollará, realiza primero la lectura y despliegue en pantalla de computadora de una imagen digitalizada que muestre la distribución de vasos del fondo ocular. Mediante algoritmos de filtrado se eliminará un porcentaje alto de ruido contenido en ella, como un paso anterior al procesamiento. Contendrá además un editor que funcionará como un manejador de expediente clínico. Manejo de falsos colores y tonos de gris, que permitan una visión más clara de la imagen. Y por último realizará la parte más importante de el sistema, un análisis geométrico que permita detectar y cuantificar patologías y morfologías presentadas en la imagen.

1.1 PROCESAMIENTO DIGITAL

1.1.1 QUE ES EL PROCESAMIENTO DE IMAGENES

El procesamiento de imágenes es un término muy manejado hoy en día. Con el avance de la tecnología, se cuenta con microprocesadores sofisticados y circuitos auxiliares, que disminuyen el tiempo de procesamiento, manejan más información por palabra (hasta 32 bits en microcomputadoras) y son más económicos. Además existen dispositivos de memoria que pueden almacenar varias imágenes que contengan gran cantidad de pixels y el tiempo de acceso a la información se reduce considerablemente. Las tarjetas gráficas también han evolucionado, manejando una gran cantidad de colores, alta resolución que permite una mejor definición de las imágenes en video. Otros dispositivos que han sido beneficiados con el desarrollo de la tecnología, son los de captura de imágenes (cámaras, digitalizadores, scanners, microdensitómetros, etc). Todo lo anterior ha sido en beneficio del procesamiento de imágenes.

En la vida cotidiana encontramos procesamiento de imágenes, un caso común es el proceso de ajustar la brillantez y contraste en un aparato de televisión. Un ejemplo biológico al cual no le prestamos mucha atención, es el realizado por el ojo humano y el cerebro en conjunto, el sistema recibe, aumenta, analiza y almacena imágenes rápidamente. Los ejemplos anteriores muestran lo que es el procesamiento de imágenes.

Durante la confirmación como disciplina del procesamiento de imágenes, uno de los objetivos es aumentar la visibilidad de una imagen; otra importante finalidad es evaluar estadísticamente algunos parámetros de una imagen que no son fácilmente cuantificables. Los objetivos son llevados a cabo con el desarrollo e implementación de procesos necesarios para operar sobre imágenes.

Fundamentalmente existen tres técnicas para implementar el procesamiento de imágenes. Una técnica es óptica y las otras dos son electrónicas (analógica y digital). Si bien las dos últimas son electrónicas, entre ellas existe una gran diferencia -una de ellas es representada en puntos continuos mientras que la otra se representa en puntos discretos-. Cada uno de los tres métodos esta fundado en rutinas definiendo la aplicación particular del aprovechamiento de la práctica en implementar el proceso en necesidad.

El procesamiento óptico implica el uso de fundamentos y conceptos de óptica para llevar a cabo el proceso. Los cristales de aumento o lentes son ejemplo de dicho procesamiento. Los instrumentos ópticos tales como el Microscopio Simple (lupa), el Microscopio Compuesto, Anteojo Astronómico, Telescópios, Cámaras Fotográficas etc.. en su forma más simple, utilizan como dispositivos elementales y necesarios lentes de aumento. Otra importante aplicación de ésta técnica se encuentra en la fotografia de cuartos o recintos obscuros.

El procesamiento analógico de imágenes se basa en la alteración de las mismas a través de medidas eléctricas; el ejemplo de la TV encaja en dicha técnica. Las señales de TV son niveles de voltaje que varian en amplitud y representan la brillantez a través de la imagen. Por alteraciones eléctricas de esta señal podemos alterar la apariencia de la imagen en el desplegado final. Con los controles de brillantes y contraste ajustamos la amplitud y la referencia de la señal de video.

El procesamiento digital es una técnica que causa la llegada de las computadoras digitales, con ellas se permite la implementación precisa de procesos, a su vez que proporciona gran flexibilidad y fuerza en general a las aplicaciones de la técnica.

Dentro del dominio digital una imagen es representada por puntos discretos definidos por su intensidad. Cada punto cuenta con una posición y un valor.

Para manipular los valores de intensidad la computadora es capaz de realizar operaciones complejas con relativa facilidad. La flexibilidad en la programación de las computadoras permiten operaciones que modifican la imagen fácilmente, característica que no se presenta en el procesamiento óptico y analógico.

1.1.2 HISTORIA

Los orígenes del procesamiento digital se remontan a los años 60's. En ese tiempo la N.A.S.A. en uno de sus programas lunares buscaban caracterizar la superficie lunar que serviría más tarde para el programa APOLLO. El programa RANGER establecía en parte,

la superficie lunar por medio de imágenes que retransmitia a los científicos para su evaluación. Previamente a las misiones RANGER existieron otras que presentaban diversas fallas en la creación de imágenes, fue hasta RANGER 7 que fueron transmitidas a la tierra miles de imágenes. Esta imágenes de TV fueron tomadas en su forma analógica original y fueron convertidas a forma digital y así fueron procesadas.

El trabajo inicial en procesamiento digital fue hecho en el NASA's Jet Propulsion Laboratory en Pasadena California. El proyecto MARINER fotografió a Marte, Venus y Mercurio; el proyecto SUVERYOR colocó cámaras en la superficie lunar; el PIONEER 10 y 11 mandó imágenes de Júpiter y Saturno; la nave VIKING equipada con cámaras colocó algunas en la superficie de Marte. Fué así como la investigación espacial inició el procesamiento gráfico. Hoy en día la técnica no solo la encontramos en la aplicación anterior, también es encontrada en ramas de la medicina, de la Física, en aplicaciones de robótica etc..

1.1.3 BASES DEL PROCESAMIENTO DIGITAL DE IMAGENES.

El procesamiento de imágenes es un campo que abarca un rango amplio de capacidades. Dos términos que pueden ayudar a la comprensión del campo son :

-OPERACION SOBRE UN IMAGEN: es una acción cualquiera distinta para cada aplicacion, llevada a cabo siempre sobre una imagen.

-PROCESO SOBRE UNA IMAGEN: define como se debe de implementar la operación sobre la imagen.

1.1.3.1 CLASIFICACION DE LAS OPERACIONES SOBRE IMAGENES

Tres clases de operaciones pueden ser usadas en las imágenes, a saber :

(1) Realce de calidad en la imagen.- Operación que modifica la apariencia de una imagen,

(2) Análisis de imagen .- Operación que produce información numérica basada en la imagen,

(3) Codificación de imágenes.- Operación que cambia una imagen en una nueva forma.

El realce de la calidad de la imagen es una operación que sirve para resaltar o en algunos casos alterar la calidad de una imagen. Esta operación en ocaciones es subjetiva, para una aplicación una imagen será la mejor, mientras que para otra distinta aplicación la misma imagen será defectuosa.

Por eso el resultado de una operación de realce es una imagen distinta a la versión original.

El realce puede ser subjetivo u objetivo. Es subjetivo cuando la operación se usa para hacer que una imagen sea más visible y esto se realiza cuando una imagen cuenta con esa característica. Para realce objetivo se modifica la imagen para reconocer degradaciones en ella. Un ejemplo de ella es la fotometría correctiva.

La operación para realzar la calidad de la imagen puede ser subdividida en dos subclases. *contraste y espacial.*

El realce por contraste trata con alteraciones de brillantez en la imagen. Blancos, negros y grises son intensificados o suprimidos resaltando fases que difícilmente se aprecian en la imagen original.

El realce espacial es una operación que modifica el detalle del contenido de una imagen. Así con un manejo de ambas subclases, se puede tener una flexibilidad tal para corregir o modificar una imagen.

La figura 1.1.3.1 muestra las clases de operaciones sobre una imagen.

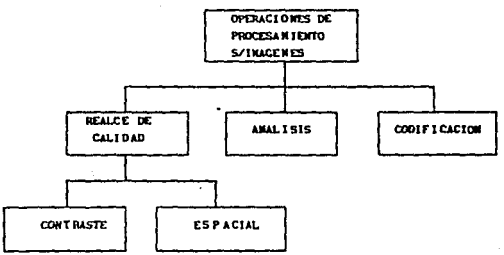


FIGURA 1.1.3.1 Operaciones sobre imágenes.

Degradación/Realce de Contraste: La degradación de contraste es un problema asociado con características de brillantes pobre. El término *CONTRASTE* trata con distribución de brillantez en una imagen.

Una imagen es de contraste deficiente cuando tiene atributos altos o bajos de brillantez. Si tratamos con imágenes en blanco y negro, existe contraste alto si se presenta tonos en negro obscuro y blanco claro o brillante. Detalles de escena son distintivos al contar con tonalidades de grises suaves. Una imagen con bajo contraste cuenta con un aspecto descolorido, únicamente se cuenta con pocos tonos de gris, no existen virtualmente ni negros ni blancos intensos y las escenas son vistas con dificultad. para obtener una imagen buena basta con jugar con los tonos de gris bajando o subiendolos. El realce de contraste es una técnica muy usada para hacer visibles algunos aspectos que antes estaban ocultos.

Degradación/Realce Espacial: la degradación espacial en una imagen es un problema asociado con la presentación de detalles de una escena en la misma. El término espacial trata con dos dimensiones en la imagen. Una imagen cuenta con una deficiencia espacial y existen áreas borrosas, puede ser borrosa sin nitidez, También son problemas de este tipo la nieve en imágenes de televisión y la distorsión de escenas geométricas dentro de una imagen. Esta técnica es usada para alterar la calidad de detalles espaciales de una imagen.

1.1.4 ANALISIS DE IMAGEN

La operación de análisis produce resultados no pictóricos, en cambio la salida es numérica, o información gráfica basada en caracteristicas de la imagen original, con el objetivo de describir algunos aspectos de la imagen y presentando los resultados al observador. La operación de análisis sirve para describir calidades de imagen, apoyado en la operacion de realce.

Además de la descripción de características de escenas en la imagen también el análisis se encarga de la medición de escenas y en el reconocimiento de patrones. La operación más común en el procesamiento general de una imagen es obtener el histograma.

El histograma asocia en barras gráficas la distribución de intensidades que presenta una imagen. Información de contraste es facilmente obtenida de dicha gráfica, permitiendo escoger la operación de realce más adecuada. las mediciones de contraste e intensidad dadas por el histograma son invaluables cuando se intenta corregir una imagen.

1.1.5 CODIFICACION DE IMAGENES

Esta operación reduce la cantidad de información necesaria para describir una imagen. Existen 2 tipos de codificación. El primero en el cual de alguna forma no se pierda información. La reconstrucción de la imagen original se obtiene unicamente a partir de la versión codificada. El segundo tipo de codificación está dentro de una forma abreviada. Un ejemplo lo encontramos cuando se particiona una imagen en estructuras primitivas, codificando unicamente el lugar y orientación de cada una de ellas.

1.1.6 CLASIFICACION DE PROCESAMIENTO DIGITAL DE IMAGENES

El acto de llevar a cabo una operación computacional es llamado *un proceso*. El término *frame process* se refiere a una operación aplicada sobre una imagen.

El en el caso más general un *frame process* puede ser desarrollado en software y ejecutado por una computadora teniendo acceso a los datos de la imagen a procesar.

En general el campo del procesamiento de imágenes se subdivide en 3 categorias de operación.

- Realce de Calidad,
- Análisis y
- Codificación.

En donde cada una de ellas puede ser manejada como una área de estudio diferente. Todas ellas son importante, pero la mas sobresaliente es la de realce de calidad ya que tiene gran prioridad para las otras aplicaciones del procesamiento. El sistema de procesamiento de imágenes tiende a soportar las operaciones de realce de calidad para que se tenga una alto nivel para las otras categorias.

También en procesamiento se seleccionan y/o se procesan una secuencia larga de imágenes, por lo que la rápidez con la que se obtengan las operaciones es de gran interés. Por esta razón, el sistema de procesamiento de imágenes también incorpora una especial atención al hardware que proporcione una gran velocidad de ejecución de ciertas operaciones de realce.

En procesamiento digital de imágenes, dos procesos son clasificados fuera de la clase general de procesamiento, para ser manejado por hardware de alta velocidad, para mejorar los resultados. Estos son conocidos como *procesos de punto y procesos de grupo*. Estos sirven para implementar *realce de contraste y realce espacial* respectivamente. Además los procesos de punto constan de dos partes "*imagenes simples e imagenes duales*". Los procesos de imágenes simples permite estandizar el contraste. mientras que los procesos de imágenes duales agregan la capacidad

de combinar múltiples imágenes. Esta subdivisión se muestra en la figura 1.1.6

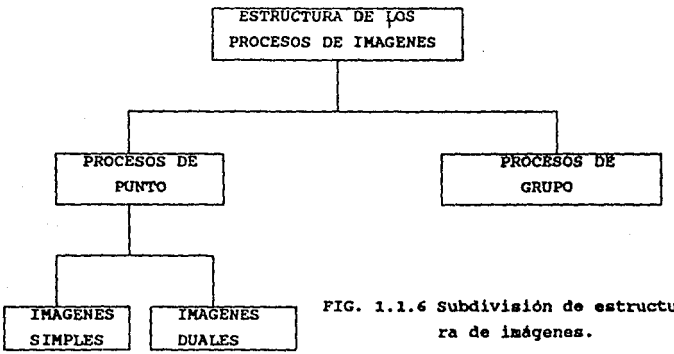


FIG. 1.1.6 Subdivisión de estructura de imágenes.

Para las operaciones de procesos de grupos que implican realce espacial, junto con el análisis y codificación de imágenes son manejados a través de los *frame process*, para ser ejecutados mediante programas de software.

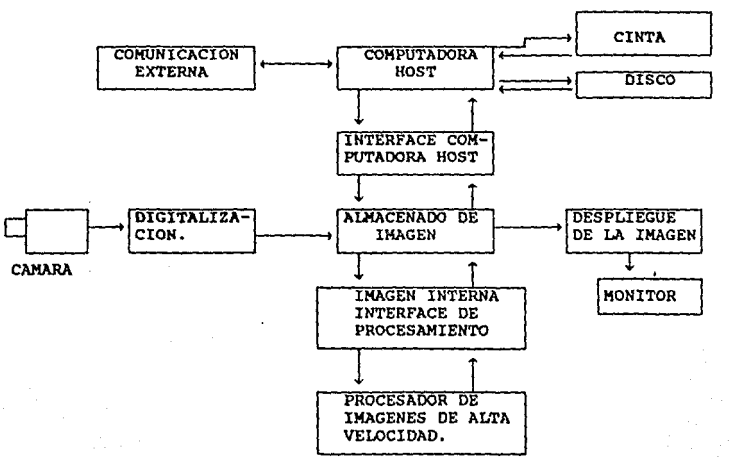
1.1.7 EL SISTEMA DE PROCESAMIENTO.

Un sistema de procesamiento digital de imágenes es una colección de dispositivos de hardware y módulos en software que proporcionan la digitalización, almacenado, despliegue y procesamiento.

La figura 1.1.7 ilustra la estructura común de un sistema de propósito general.

El primer paso en un sistema de procesamiento de imágenes es la digitalización de una de ellas. Un sistema de propósito más general debe de aceptar el standar de señal de video de televisión como entrada. El dispositivo de entrada es normalmente una cámara de video que enfoca o toma la escena de interés. El acto de digitalización consiste en convertir la señal analógica de video a una señal digital para ser almacenada en un dispositivo de memoria. Aunque la imagen existe en memoria es necesario tener disponible un despliegue (display) para verla antes y después de procesar.

El desplegado de una imagen residente en memoria es acompañado por una lectura repetitiva de una imagen digital colocando los datos de salida en un subsistema de despliegue. Aquí los datos de la imagen son reconvertidos al standar del formato de televisión.



El director del sistema de procesamiento es una computadora maestra. Esta computadora controla el sistema, decide cuando digitaliza, despliega y procesa la imagen. La computadora o host también sirve como interface entre el usuario y el sistema, tiene acceso directo a la imagen en memoria. Adicionalmente, el host puede transferir los datos a cinta o disco magnético. Para sistemas de procesamiento de imágenes pequeñas el host es una microcomputadora.

El elemento final de un sistema de procesamiento digital es el procesador de imágenes de alta velocidad. Aunque la computadora tiene la habilidad de obtener y definir procesos sobre la imagen almacenada, la velocidad de ejecución puede ser relativamente lenta. Por lo que es necesaria la implementación en hardware de procesos de imágenes que reduzcan al mínimo el tiempo de ejecución. Como mencionamos anteriormente los procesos de punto y grupo son manejados frecuentemente en hardware. Estos procesos permiten al usuario la habilidad de modificar el contraste de imagen y los atributos espaciales.

1.1.8 FORMATO DE UNA IMAGEN DIGITAL

Por definición el procesamiento de imágenes opera sobre información gráfica en forma digital. Las imágenes de interés pueden ser derivadas desde una gran variedad de fuentes: fotografía, televisión, radar, detectores de barrido infrarojo o rayos X.

En general el procesamiento digital de imágenes es obtenido desde formatos de T.V. porque son standares ampliamente difundidos y aceptados en ramas relacionadas con la industria.

Una fotografía típica en blanco y negro está compuesta de niveles de gris expandidos desde el negro hasta el blanco y son llamados *tonos continuos de la imagen*. La mezcla de sombras obscuras y de gris, juntamente con una continuidad fiel reproduce los elementos de la escena original.

Para convertir una imagen continua a una imagen digital debemos cortar la imagen en puntos individuales. Esto es la digitalización o más propiamente el *muestreo* porque se toman muestras de intensidades de la imagen en posiciones específicas.

A cada muestra se le da un valor numérico basado en la brillantes desde un rango de negro pasando por grises y llegando a blancos. Adicionalmente a cada muestra se le asignan coordenadas (X,Y) que describan su localización dentro de la imagen.

Una muestra es referida a un elemento de la pintura o *pixel* porque es esta la representación de un elemento discreto de una imagen digital.

Una imagen es digitalizada dentro de una parrilla (matriz) cuadrada de pixels, cada uno de los cuales es etiquetado con un par de coordenadas, una que define la columna y otra el renglón.

1.1.9 TIPOS DE RESOLUCION

RESOLUCION: Se define como la limitación del proceso de digitalización. En procesamiento de imágenes la resolución esta definido por dos parametros: *ESPACIAL* y de *BRILLANTES* con un tercer tipo *FRAME RATE* que no juega un papel relacionado con la apariencia visual de la imagen.

1.1.10 RESOLUCION ESPACIAL

El término espacial se refiere al concepto de espacio. En nuestro caso un espacio de 2 dimensiones.

Para sistemas de procesamiento digital de imágenes un objeto de dos dimensiones es una imagen con altura y ancho fijo. Cuando nosotros hablamos de resolución espacial, estamos describiendo cantidad de pixels divididos dentro de nuestra imagen digital.

Optimamente, deseamos digitalizar una imagen en donde la información no se pierda en la traslación de la imagen original a la imagen digital. Esto significa que una imagen digital apropiada sea idéntica a la original para el observador, para entender mejor el criterio para establecer el número necesario de muestras requeridas en una imagen digital, debemos de introducir el concepto de *FRECUENCIA ESPACIAL*. Para cuantificar este detalle visual hablamos de frecuencia espacial o de la razón en la cual la brillantez de una imagen cambia de obscuro a claro.

Para tomar una decisión acerca de la razón de muestreo necesaria para una apropiada resolución de la imagen, usamos el clásico criterio de Nyquist, también conocido como el Teorema de Muestreo. El teorema nos dice en términos matemáticos y relacionados con el procesamiento digital de imágenes: que para representar completamente la razón de cambios de brillantes o de detalles de una imagen original debemos muestrear, al menos, en una tasa de 2 veces más que la frecuencia más alta del detalle.

En otras palabras, si un detalle particular en una imagen original varia desde obscuro hasta blanco luminoso en cierta distancia, nuestras muestras o pixels, deben ser lo suficientemente finos para que dos de ellos cubran el detalle. Es verdad también que llega a ser redundante e inútil, muestrear una imagen a una tasa más rápida que dos veces su máxima frecuencia espacial contenida.

Algunas aplicaciones no requieren que todos los detalles se presenten en una imagen digitalizada. Manteniendo en mente que una vez que una imagen es digitalizada con un límite de frecuencia de muestreo la pérdida en los detalles es por siempre.

Otro de los tópicos que deben de considerarse para la elección de la resolución espacial, es el concepto de *ALIASING*. El fenómeno de aliasing tiene que ver con la representación erronea de la imagen original. El fenómeno aparece cuando el criterio de Nyquist es violado por una frecuencia espacial presentada en la imagen original. Esto ocurre cuando un detalle dentro de una escena tiene una frecuencia espacial más grande que la media de la frecuencia de muestreo. Cuando el fenómeno de aliasing se presenta, más patrones ocurren en la imagen digitalizada. A menos, que una imagen de muchas frecuencias altas sean repetidas y tenga un bajo muestreo, el efecto de aliasing puede no tomarse en cuenta.

Para escoger la resolución espacial de una imagen digital se deben de considerar 3 factores.

1.- El detalle de la imagen digital,
2.- El tamaño del display (video) para la imagen digital y
3.- La distancia a la que se encuentre el observador.

1.1.11 RESOLUCION DE BRILLANTEZ

La segunda resolución que concierne a las imágenes digitales es la brillantez. Como se cubrió en la sección previa, cada pixel representa la brillantez de la imagen original en el punto de su muestreo.

El proceso de digitalización muestréa la imagen original a locaciones predeterminadas. Cada brillantez muestreada es entonces convertida a un valor numérico entero, esto es conocido como *cuantizacion*.

La operación de cuantización convierte en un punto de muestréo un nivel analógico de brillantez a un valor numérico dentro de una cierta tolerancia ó resolución de brillantez. Este proceso es llevado a cabo por un convertidor analógico digital.

Para cuantizar la brillantez de un pixel, primero debemos de definir la exactitud con la que la conversión será hecha. Por ejemplo, la conversión a un número binario de 3 bits permite que cada pixel se represente por uno de ocho niveles de brillantez. Los 8 niveles de brillantez comprenden lo que es llamado escala de gris. Los fabricantes de equipo procesador de imágenes han optado generalmente por escalas de gris logarítmicas de 8 bits para representar imágenes digitalizadas.

La cuantización de 8 bits asegura un cambio no detectable de un nivel de gris a otro adyacente, y la cuantización logarítmica toma ventaja de las características de la respuesta del ojo humano.

1.2 LA FLUORESCEINGRAFIA: ANTECEDENTES Y CONCEPTOS.

Por el año de 1975 se publicaron los primeros resultados de un examen angiográfico del fondo ocular. Para realizar las pruebas se inyectaba una solución acuosa de azul de tripano en la arteria carótida de gatos y mediante oftalmoscopía directa median el tiempo de la circulación retiniana del colorante. Posteriormente la sustancia azul fué sustituida por la fluoresceína. El proceso en el fondo ocular fué fotografiado en color a la velocidad de 20 imágenes por segundo, iluminando el fondo ocular por medio de luz incandescente filtrada a través de azul de cobalto.

En 1961 Novotny y Alvis con su método fotografiaron selectivamente la fluorescencia del colorante interponiendo una serie de filtros en el sistema de iluminación y en el de la fotografía del retinógrafo. Usaron películas de 35 mm, monocromaticas y de alta sensibilidad y con ello demostraron que detalles diminutos difíciles de observar por medio de la oftalmoscopía podian ser estudiados con este método.

El método de la angiografía fluorescente del fondo ocular sigue aplicandose sin modificaciones de su idea básica, el perfeccionamiento del material fotográfico y la introducción de una colección de filtros más eficientes y de diversas modificaciones en el retinógrafo, han permitido el registro de detalles aún más finos de la imagen, así como más rapidez en el proceso de fotografía seriada de la fluorescencia, la estereofotografía y la fotografía instantánea, que facilita un diagnóstico inmediato.

En Japón su uso ha sido importante en el diagnóstico de enfermedades del fondo ocular y de retinopatía diabética . Hoy en día parece que la angiografía fluorescente es indispensable para la investigación y diagnóstico de enfermedades del fondo ocular.

1.2.1 PRINCIPIOS FUNDAMENTALES

El principio de la luminiscencia sucede cuando la luz es absorbida por una sustancia, que convierte a la energía fótica en calor o en energía química o en luz de otra longitud de onda. Cuando la fotoluminiscencia es menor de 10^{-4} segundos, se llama fluorescencia. El estímulo fótico que causa la fluorescencia se llama luz estimulante o activadora. La fluoresceína soluble tiene la propiedad de fluorescencia muy intensa por lo cual es utilizada en la angiografía fluorescente del fondo ocular.

La fluoresceína sódica, es soluble en agua con un color rojo amarillento y muestra una intensa fluorescencia verde amarillenta cuando la solución es neutra o alcalina. Así es posible fotografiar la fluorescencia de la fluoresceína contenida en los vasos del fondo, iluminando éste con luz azul y mediante una filtración selectiva de la fluorescencia a través de un sistema adecuado de filtros que se interpone delante de la película.

La fluorescencia sódica es estable contra el calor y no tiene nunguna acción farmacológica de importancia, exceptuando las náuseas y vómitos que, algunas veces, pueden seguir a su administración rápida por vía intravenosa, cuando se administra por esta vía, aproximadamente una sexta parte es excretada por la orina y el resto con la bilis, sin pasar por una desintregración metabólica.

1.2.2 TECNICA

El procedimiento preliminar consiste en fotografiar el fondo de ojo usando película de color o monocromática, por que las necesidades de la clínica no son satisfechas con las fotorgrafías

del fondo con fluoresceína al proporcionar sólo datos complementarios para la interpretación de las observaciones oftalmológicas, por tal razón es conveniente contar con dos cámaras, una para fotografiar en color y otra para fotografía fluorescente.

Una vez que el paciente ha sido inyectado con fluoresceína, si la iluminación es la apropiada y existe el enfoque exacto del fondo se comienza la fotografía seriada con los filtros en posición a los 5 segundos de la aplicación de la inyección. Las fotografías sucesivas cuentan con un intervalo de 2 segundos para cada una. A los 10 ó 12 segundos, la sangre arterial que contiene fluoresceína alcanza la coroides y posteriormente las arterias retinianas. El polo posterior toma un aspecto nebulosos, seguido por una delimitación nítida de las arterias retinianas. Después de otro segundo, el retorno venoso de la fluoresceína se inicia en forma estratificada para que llene en toda su anchura a la vena retiniana. Al cabo de unos segundos comienza a declinar el contenido de la fluoresceína, primero en la sangre arterial y luego en la sangre venosa. La fotografía seriada dura los primeros 30 segundos para continuar a intervalos mayores. Ahora si se desea puede repetirse el proceso a los 5-10 minutos de la primera inyección. Como se dijo con anterioridad puede existir una sensación de náuseas e incluso vómitos un minuto después de terminada la inyección, la sensación dura aproximadamente unos 60 segundos. Otro efecto secundario es el tono de piel que adquiere el paciente, será un tono ictérico y durará hasta el otro día.

La fotografía en color del fondo ocular fluorescente tiene un valor singular como complemento para las observaciones registradas en películas en blanco y negro. Existen varias ventajas de este método. Al documentar las observaciones flourescentes en forma de diapositivas listas para la proyección sobre una pantalla, pueden ser estudiadas sin recurrir a procedimientos de reproducción o de

inversión. El aspecto más notable de este método es la gran cantidad de información en comparación con fotografías en blanco y negro.

Después de la inyección intravenosa de fluoresceína en la vena cubital deben de pasar de 7 a 12 segundos para que el colorante aparezca en la arteria central de la retina, este período de tiempo representa el tiempo de *circulacion brazo-retina*. Las arterias de la retina son las primeras en llenarse de fluoresceína, mientras que las venas siguen siendo obscuras, a este se le denomina *fase arterial*. Dos segundos después ocurre el retorno venoso de la sangre con fluoresceína. Las venas presentan al principio un aspecto laminado, por que el intercambio arteriovenoso es completado antes en las cercanías del disco óptico que en la periferia retiniana, si una vena que contiene fluoresceína se une a otra no fluorescente la porción de la vena que sigue a esta confluencia, presenta una imagen estratificada o *corriente laminada*. La corriente laminada continua durante aproximadamente de 7 a 10 segundos y esta etapa reciba el nombre de *fase venosa precoz*. Posteriormente las venas toman una fluorescencia más uniforme al disminuir el colorante en la arteria, el estado de repleción uniforme de la vena con la fluorescencia es la *fase venosa tardia*. Cuando la intensidad de la fluorescencia es igual en la vena como en la arteria, los capilares se vuelven visibles y el conjunto del fondo ocular ofrece una fluorescencia nebulosa difusa debido a que se alcanzo una máxima concentración de la fluoresceína en los capilares retinianos, esto indica el final de la fase venosa precoz conocida también como *fase capilar*. Y es en la fase venosa tardia o *fase posterior* cuando la concentración de la fluoresceína desciende en las arterias y en los vasos haciendo difícil identificar a los mismos vasos retinianos.

Mediante la fluoresceingrafía del fondo ocular se observa la distribución del árbol vascular retiniano y en ocaciones hasta la red capilar en todo detalle. El fondo ocular comienza un segundo antes de que la fluoresceina aparezca en las arterias retinianas. Este estado se mantiene durante la fase arterial y venosa y por definición se llama *fluorescencia de base*.

Los párrafos anteriores han explicado de manera breve la técnica de la fluorescengrafía, para más detalles consultar la referencia 1.

CAPITULO DOS

TENDENCIAS ACTUALES DEL PROCESAMIENTO DIGITAL DE IMAGENES

En los inicios del procesamiento de imágenes, el equipo con el que se contaba limitaba muchas veces las aplicaciones (SOFTWARE) por la escasez de recursos, de manera que se necesitaba disponer de equipo más sofisticado; por tal razón los laboratorios, centros de investigación e industrias se dedicaron a desarrollar y fabricar nuevos dispositivos de procesamiento. Actualmente la investigacion y fabricación de equipo sigue su marcha, haciendo posible que el procesamiento gráfico sea una herramienta accesible y eficaz en cualquier disciplina.

El presente capítulo mostrará una perspectiva de las tendencias de HARDWARE y SOFTWARE del procesamiento de imágenes, en microcomputadoras, con el fin de mostrar un panorama de los principales avances que se presentan en esta disciplina. Cabe hacer notar que describimos solo productos nuevos de diversas compañias, pero sin olvidar que existen un sin número de equipo y dispositivos no mencionados, sin embargo, existe entre ellos una similitud en cuanto a tecnología y rendimiento que resultaría redundante describir a todos.

Para poder llevar a cabo esta función nosotros hemos clasificados los avances del procesamiento digital de imágenes de la siguiente forma:

2.1.- Formatos.
tipos y estructuras, Técnicas de compresión de datos

2.2.- Hardware
2.2.1 tarjetas gráficas
2.2.2 dispositivos de obtención de imágenes

2.1 FORMATOS

La información es un conjunto de datos, si estos son estructurados en una forma especifica se dicen que tiene un FORMATO.

A continuación describimos algunos de los principales formatos que utilizan en la actualidad algunos paquetes gráficos y de investigación. Comenzando por describir el formato utilizado para la realización del Sistema Gráfico en Angiología Retiniana. Hablaremos también de un formato muy conocido, el PCX y su algoritmo de compresión. Se incluye la descripción de dos formatos (GIF & TIFF) que pretenden ser un estándar en imagenes.

2.1.1 FORMATO INOVION

El formato INOVION presenta un tipo de almacenamiento de datos sumamente sencillo de codificar y decodificar. Aunque es un formato relativamente antiguo, sigue siendo muy utilizado en centros de investigación dentro de la U.N.A.M.

Las imágenes almacenadas en este tipo de formato comienzan con un encabezado de 6 bytes, el encabezado se muestra en la figura 2.1.1. Donde las dimensiones de la imagen se señalan en los bytes 1-4, dos para el tamaño en "X" y dos para "Y" respectivamente. Cuando se conoce las dimensiones de la imagen el encabezado se puede ignorar.

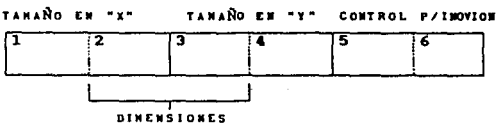


FIGURA 2.1.1 ENCABEZADO DEL FORMATO INOVION.

La información del archivo es secuencial, el primer valor obtenido (intensidad) corresponde al pixel en la posición (1,1), el siguiente al (1,2) y así sucesivamente. Los valores de cada uno de los pixels está dentro del rango de 0-127 en tonos de gris cuando la imagen es monocromática. Si esta incluye colores las intensidades de los tonos RGB se almacenan por separado, cada uno en un byte.

EL formato inovión presenta la desventaja de no comprimir datos, generando archivos grandes y siempre de tamaño constante para imágenes que presenten diferentes características.

2.1.2 FORMATO PCX

Un tipo de formato muy utilizado en la actualidad en paquetes gráficos como PC Paintbrush, Ventura Publisher y PageMaker es el formato PCX. Comienza con un encabezado de 128 bytes que en muchos

casos se puede ignorar cuando las imágenes a manejar tengan la misma resolución, pero si se quiere manejar imágenes con diferente resolución o colores, es necesario darle una interpretación correcta al encabezado. La figura 2.1.2 muestra el contenido de los 128 bytes. El resto de la imagen consiste en datos codificados. El método de codificación es orientado a la técnica RUN-LENGTH.

La información de las paletas (bytes 16-63) es almacenada en el archivo en formato standar RGB (IBM EGA, AT&T DEB, Sigma Color 400, Color 350,...), los datos son almacenados en 16 arreglos de 3 bytes cada uno correspondientes a los valores de Rojo, Verde y Azul.

Para un standar IBM CGA, únicamente el primer byte de la tripleta es utilizado, el cual representa el color del background (fondo) para encontrar el valor se toma el contenido sin signo y se divide entre 16 obteniendose un resultado entre 0 y 15. El primer byte de la segunda tripleta representa el color del foreground (frente).

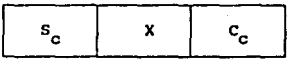
2.1.2.1 METODO DE COMPRESION RUN LENGTH

RUN-LENGTH es un método de compresión de datos utilizado por PCX para reducir físicamente un archivo que contenga secuencias de datos repetidos.

La técnica utiliza un caracter especial que denota que una compresión ha ocurrido, normalmente seguido por el caracter a repetir y por último un contador que indica el numero de veces que se ha repetido en la secuencia, almacenandolo en una cadena de compresión. La figura 2.1.2.1 muestra el formato general del string.

INFORMACION DEL ENCABEZADO		
DETALLE	TAMAÑO BYTES	DESCRIPCION/COMENTARIO
MANUFACTURA	1 BYTE	BANDERA CONSTANTE 10 = PC PAINTBRUSH PCX
VERSION	1 BYTE	INFORMACION DE VERSION 0 = VER 2.5 2 = VER 2.8 3 = VER 2.0 5 = VER 3.0
CODIFICACION	1 BYTE	CODIFICACION USADA 1 = PCX RUN-LENGTH
BITS POR PIXEL	1 BYTE	NUMERO DE BITS POR PIXEL B/NUMERO
VENTANA	8 INT[4]	DIMENSIONES DE LA IMAGEN (Xmin Ymin)-(Xmax Ymax) TODAS LAS COORDENADAS SON EN FORMATO ENTERO
HRES	2 INT	RESOLUCION HORIZONTAL EN EL DISPOSITIVO DE VIDEO
VRES	2 INT	RESOLUCION VERTICAL EN EL DISPOSITIVO DE VIDEO
COLORES	48 BYTE 16 X 3	ES UNA COMBINACION DE VARIAS PALETAS DE COLORES. VECTOR CON VALORES RGB (0..255)
RESERVADO	1	USO P/PCX
NPLANOS	1 BYTE	NUMERO DE PLANOS DE COLORES EN LA IMAGEN
BYTE POR LINEA	2 INT	NUMERO DE BYTES POR CADA LINEA EN LA IMAGEN
RESTANTES	120	NO USADOS

FIGURA 2.1.2 ENCABEZADO DEL FORMATO PCX



S_c = Caracter especial que indica compresión
X = Cualquier dato a repetir
C_c= Contador de caracteres, número de veces que ocurrió el caracter X.

FIGURA 2.1.2.1 FORMATO GENERAL DEL STRING DE COMPRESION

PROCESO DE CODIFICACIÓN

El proceso utilizado por RUN-LENGTH para la codificación es mostrado por el diagrama de flujo 2.2.1.

PROCESO DE DECODIFICACIÓN

La función necesaria para decodificar la información obtenida por el proceso de codificación se muestra en el diagrama 2.2.2.

2.1.3 FORMATO GIF

GIF (Graphics Interchange Format) fué desarrollado por Compu-Serve en 1987 para satisfacer la necesidad de contar con un protocolo de transferencia para imágenes de color. Fue diseñado para soportar hasta 64000 pixels, 256 colores de 16 millones de paletas, imágenes múltiples en un solo archivo, rápida decodificación, eficiente compresión e independencia de hardware. El formato hace uso de campos etiquetados, si bien gran parte de la información es almacenada en un encabezado posicional, de aquí en adelante el formato intercambia a estructuras etiquetadas.

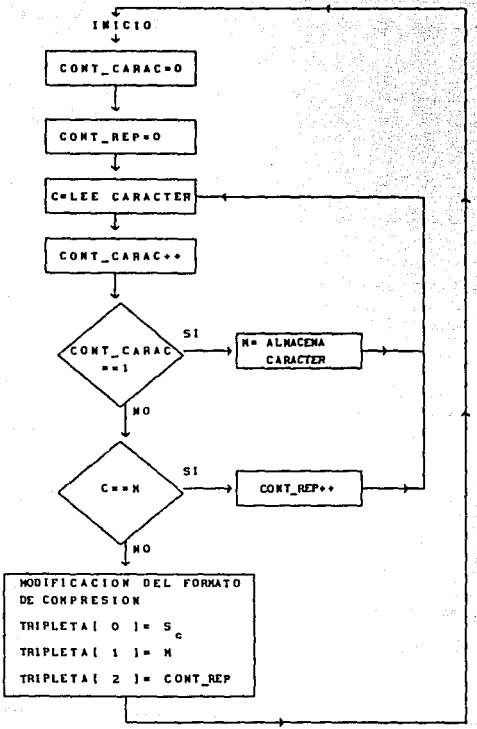


DIAGRAMA 2.2.1: PROCESO BASICO DE CODIFICACION DE RUN-LENGTH

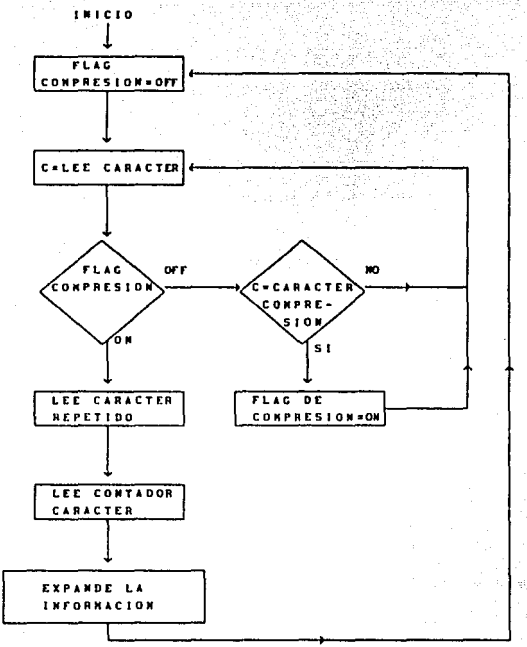


DIAGRAMA 2.2.2 PROCESO DE DECODIFICACION DEL RUN-LENGTH

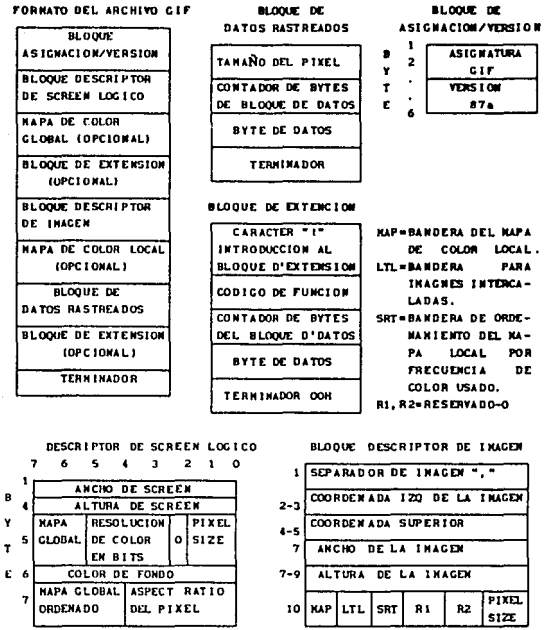


FIGURA 2.1.3. FORMATO DE IMAGEN GIF

En GIF, los bloques marcados o etiquetados son llamados de extensión. Actualmente, se soportan dos tipos. El primero es un bloque de comentarios de información del scanner, digitalizador o software creador de la imagen. El segundo contiene comandos de control de la imagen, de ese modo define funciones de control adicional relativas a varios aspectos del despliegue de la imagen.

La figura 2.1.3 muestra el formato GIF. Note que los bloques de extension pueden venir antes o después de los datos de la imagen. El números de colores o tonos de escala de gris son almacenados como un números de 3 bits. Por lo tanto para disponer de 2 a 256 colores o tonos de gris se necesitan de 1 a 8 bits por pixel. Los colores "rastreados" son almacenados como código, de esa manera son referenciados en una tabla activa de colores. Un dato de la tabla tiene un byte por cada plano de color (Red, Verde, Azul).

Los archivos GIF pueden tener diversos mapas de colores; muchos usan simplemente un mapa global, o en su lugar podemos hacer uso de un mapa local que es usado únicamente por el bloque de datos que aparecen en los datos rastreados. No se pueden utilizar los dos tipos simultáneamente.

Otra característica de GIF es poder intercambiar imágenes en diversos equipos IBM PC y compatibles, Atari ST, Commodore Amiga y familia Macintosh.

Dependiendo del número de colores presentados, el tamaño de la imagen y la cantidad de detalles presentes; GIF reduce el archivo mediante el algoritmo LZW entre un medio a un octavo de la imagen original.

2.1.4 FORMATO TIFF

TIFF (Tag Image File Format) es basado totalmente en el concepto de campos etiquetados, implementados en el formato GIF, evitando la obsolecencia. TIFF fue desarrollado conjuntamente por Aldus y Microsoft como un formato común para scanners y software de publicidad.

la figura 2.1.4 muestra el formato de un archivo TIFF. Como se observa, el encabezado inicial contiene únicamente 8 bytes. Toda la información y parámetros relacionados con la imagen son almacenados en campos etiquetados. La versión 5.0 de TIFF incluye 45 de estos campos; el número es engañoso, porque hay dos campos etiquetados separados que indican las dimensiones de la imagen, además existen campos para identificar el dispositivo origen, marca, modelo, descripción, software y fecha. Diversos campos necesarios tienen un valor por default y no necesitan ser especificados. Muchos de los campos etiquetados no son necesarios para la producción de la imagen, aunque una imagen puede llegar a ser distorsionada fuera de ellos. Por ejemplo, TIFF provee campos que habilitan imágenes de "aspect ratio" inusual para ser propiamente desplegada. Fuera de los campos de datos apropiados la imagen puede aparecer estirada.

TIFF soporta combinaciones de compresión, funciones especiales de control de imágenes y muchas otras características. Porque TIFF es grande, requiere código extensivo para desarrollar implementaciones completas. La versión 5.0 define cuatro clases de TIFF: TIFF-B para imágenes bi-nivel (1 bit), TIFF-G para imágenes en escala de gris, TIFF-P para imágenes en paletas de color y TIFF-R para imágenes en RGB. TIFF-X se refiere a programas que soportan las cuatro clases. Las clases habilitan programas de aplicación para usar únicamente las características que ellos necesitan para su propia ejecución. Cada clase tiene un número mínimo de campos para asegurar compatibilidad entre ellas.

En los archivos TIFF debe de aparecer un encabezado inicial de 8 bytes. El directorio de la imagen (IFD) contiene una lista de los campos presentes en el archivo. Los puntos de offset del campo de directorio nos llevan a la localización en el archivo donde la información es almacenada, permitiendo los datos en cualquier lugar dentro del archivo. Los datos de la imagen actual

son almacenados en "strips" (tiras) que son encontradas por medio de un registro en el archivo de directorio, los strip pueden tener un tamaño. El default es un strip conteniendo el registro de archivo, pero el formato especifica recomendable un tamaño de 8K bytes de longitud. Además TIFF es un formato basado en apuntadores siendo más complejo que GIF, pero permite gran flexibilidad porque los campos pueden ser escritos en cualquier orden.

TIFF como GIF soportan múltiples imágenes en un solo archivo (haciendo referencia como subarchivos), si bien no se necesita decodificadores para procesos. El último registro en un IFD para uno o para otro es 0000 para el final de un archivo o un offset para el IFD del siguiente subarchivo.

TIFF soporta dos métodos de almacenamiento de datos en color. TIFF-P que es similar a GIF. Un simple campo define un mapa de color para la imagen. Los datos de la imagen misma están almacenados como codigos relativos al mapa de color. Este método permite un eficiente modo de almacenado, aunque está limitado a 256 colores. El mapa de color dibuja sus entradas a partir de una paleta de 48 bits (la unidad básica de TIFF, es una palabra de dos bytes, por lo tanto con los 16 bits se forman los planos Red, Green, Blue). TIFF-R es usado para definir imagenes totalmente en RGB. Un pixel es representado por tres valores de 8 bits de RGB que proporcionan más de 16 millones de colores.

Para facilitar la reprodución fiel de imágenes en distintos equipos, TIFF soporta diversos campos extras. Estos campos hacen poco uso de formatos especifico de hardware y generalmente son independientes de ellos. La habilidad para redefinir un punto blanco y la coloración primaria es importante cuando la imagen se despliega en equipo no estándar. Otra importante habilidad son las especificaciones del aspect ratio. Las habilidades anteriores y otras se dan gracias al uso de los campos extras que maneja TIFF. Si se desea contar con un campo para su propia aplicación se requiere su registro con Aldus y Microsoft.

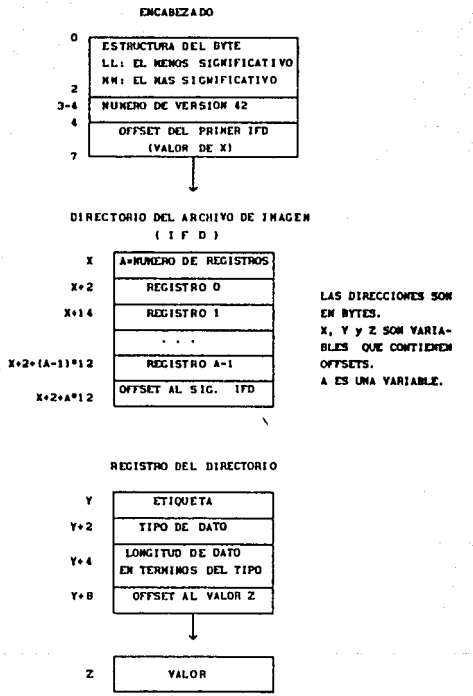


FIGURA 2.1.4 FORMATO DE UNA IMAGEN TIFF.

Cabe hacer mención que los dos formatos anteriores usan el algoritmo de compresión de datos LZW (Lempel-Ziv and Welch). Que se explica a continuación.

2.1.4.1 METODO DE COMPRESION LZW

Es uno de los mejores algoritmos de compresión de datos; con una rapidez en el orden de 50 Kb por segundo en un sistema 386. El método reduce en un 50% y en ocasiones hasta en un 90% el archivo de una imagen. LZW usa una "tabla de String" (TS) para almacenar cadenas de caracteres que representan datos de entrada.

METODO DE CODIFICACIÓN

El archivo es examinado secuencialmente caracter a caracter, obteniendo los elementos distintos que pasaran a inicializar la TS, cada uno con valor de código secuencial.

El archivo de entrada es leido de izquierda a derecha, comenzando con el primer caracter, si este existe en la TS se concatena con el siguiente caracter, comparandose nuevamente con la TS, y así sucesivamente hasta encontrar un string que no se encuentre en la tabla, si este es el caso se agrega a ella, con un nuevo valor de código. El ciclo se repite, comenzando con el último elemento del string anterior como pivote, hasta el fin del archivo. Se presenta a continuación el pseudocódigo del algoritmo.

```
Inicializar la tabla de  String  (TS)
z=NULL
FOR cada caracter en el archivo
   x=siguiente caracter en el archivo
   IF z+x estan en la TS
      z=z+x
   ELSE
      escribir z al archivo de salida
      agregar z+x a la TS
      z=x
   ENDIF
```

Las tablas 2.1.5 muestran un ejemplo de compresión utilizando LZW.

SIMBOLOS ENTRADA	*a*	*b*	*a*	*b*	*c*	*b*	*a*	*b*	*a*	*b*	*a*	*a*	*a*	*a*	*a*
CODIGO SALIDA	1	2	4	3	5		8		1		10		11		
NUEVO STRING A LA TS		4		6		8				10		12			
			5		7			9			11				

TABLA STRING TS		TABLA ALTERNATIVA	
a	1	*a*	1
b	2	*b*	2
c	3	*c*	3
ab	4	1	4
ba	5	2*a*	5
abc	6	4*c*	6
cb	7	3*b*	7
bab	8	5*b*	8
baba	9	8*a*	9
aa	10	1*a*	10
aaa	11	10*a*	11
aaaa	12	11*a*	12

TABLAS 2.1.5. UN EJEMPLO DE COMPRESION UTILIZANDO LZW

METODO DE DECODIFICACION

Utiliza la misma Tabla de String (TS), y en forma similar se reconstruye el mensaje. Cada valor de código recibido es referido a la TS, y sustituido por su correspondiente código y caracter de extensión, la operación es recursiva hasta que la cadena es un simple caracter. La tabla 2.1.6 muestra el método, usando el archivo codificado anteriormente.

CODIGO DE ENTRADA	1	2	4	3	5	8	1	10	11
	a	*b*	1*b*	*c*	2*a*	5*b*	*a*	1*a*	10*a*
			a		*b*	2*a*		*a*	1*a*
						b			*a*
DATOS DE SALIDA	*a*	*b*	*ab*	*c*	*ba*	*bab*	*a*	*aa*	*aaa*

TABLA 2.1.6 EJEMPLO DE UNA DECODIFICACION CON LZW.

2.2 HARDWARE

El hardware encierra todo lo referente a CPU, coprocesadores, buses de datos y direcciones, memorias, convertidores y otros componentes físicos que forman parte de los dispositivos de proceso. Estos elementos han sufrido grandes modificaciones en los últimos años, su evolución ha sido acelerada. Toca hablar de aquellos elementos que componen un sistema digitalizador de imágenes y que es la parte escencial del mismo.

2.2.1 TARJETAS GRAFICAS

VIDEO FRAME GRABBER MODELO MV1 (METRABYTE/ASYST/DAC)

Es un tarjeta de 512 x 512 x 8 bits digitalizadora en tiempo real para PC AT/XT y compatibles. La tarjeta toma una señal de video analógica, almacenando la información digital en un banco de memoria, permitiendo la edición de la imagen y agregar texto relacionado. Puede desplegar la información en display RGB o monocromáticos. Acepta señales RS-170 o CCIR, de cámaras interlazadas y con salida digital. Cuenta con dos bancos de memoria de video de 256 K cada uno, capaces de almacenar hasta dos imágenes usando una técnica de overlay. Mediante software se puede configurar el tamaño de la estructura de la imagen, siendo común las de 512 x 512 y 1024 x 256, pero se puede manipular dimensiones arbitrarias. Manejo de LUT (Look-Up Tables) para transformaciones de punto. Las LUT de salida permiten a los usuarios cambiar hasta en 256 niveles de intensidad cada pixel para cada uno de 256 colores o tonos de gris, tomados de una paleta de más de 16 millones de colores.

4MEG VIDEO MODELO 10 (EPIX)

4MEG VIDEO facilita la aceleración de funciones de procesamiento, gracias a que cuenta con un TMS320C25 (procesador de señales digitales de Texas Instruments) que funciona como microprocesador de propósito general, facilitando la implementación de algoritmos de inspección y procesamiento de imágenes elaborados por el usuario en lenguaje ensamblador. se tienen 8 k word de memoria de programación.

Los datos de los pixels son almacenados en una memoria para

imágenes (1 Mb o 4 opcional) como valores secuenciales de 8 bits. Usando un controlador de memoria integrado en la terjeta, la memoria puede ser organizada para una sola imagen o para contener a varias imágenes pequeñas; además es utilizada para almacenar cálculos intermedios, overlays o menues. La memoria de imagen es accesada en bloques de 64 k.

La tarjeta maneja algunas librerias en lenguaje "C" para simplificar su uso. Opcionalmente puede contar con un menú que permite el acceso inmediato de la 4MEG.

Otras características son: velocidad de muestreo y de despliegue de 19 MHz. Hasta 1984 pixels por linea. Señales de entrada/salida RS-170, RS-330, y CCIR. Entradas analógicas o digitales. Compatible con Bus PC o AT.

OCULUS-500 (CORECO INC.)

Esta tarjeta es compatible con el bus AT. Dependiendo el uso y su aplicación exacta, puede ocupar uno o dos slots de expansión. A continuación se describen las principales características de la tarjeta.

MOTHER-BOARD

La mother-board (500FB-ALU8) es una tarjeta que contiene 2MB de memoria de video, un procesador gráfico el TI 34010 equipado con un ALU DE 8 bits y un procesador de despliegue de alta resolución, el 34010 puede accesar 512Kb de memoria RAM en tarjeta y 128Kb de ROM. Contiene librerias de software para expander imágenes para el VGA standar; es compatible con el formato VGA de 640 X 480 pixels. De cualquier manera la resolución de desplegado es programable, permitiendo con esto desplegar imágenes de diferentes tamaños desde 256 X 256 hasta 1280 X 1024 pixels.

El total color RGB es llevado a cabo en tres salidas Look-Up Tables (LUT) (una por cada color).

El desplegado de la imagen puede girar y hacer el scroll a través de simples incrementos de pixels. En un monitor regular de 40 lineas de resolución puede ser usado para desplegar en ventanas móviles una imagen de 2280 X 1024 pixels.

Inversamente el procesador gráfico Oculus-500 hace posible el desplegado de la imagen en una porción del screen, mientras las caracteristicas aparecen en otra área. Una cualidad particularmente atractiva de la tarjeta es la compatibilidad con dispositivos de rayos X como son el NRM y el CAT-scan, utilizados en campos de la medicina.

ESTRUCTURA DEL BUFFER AUXILIAR

La expansión de la estructura de memoria es llevada a cabo a través de una tarjeta de tamaño medio (500FB-ALU16 o la 500FB-ALU16-GO). Esta tarjeta contiene 2Mb de memoria de expansión junto con un circuito de expanción para el ALU permitiendo 16 bits por pixel. La tarjeta también puede tener 512 Kb de memoria RAM adicional.

MODULO DIGITALIZADOR

La digitalización de la imagen puede ser llevada a cabo (OPCIONAL) por el Procesador de entrada Multiple Barrido (500MS-AD8 o el 500MS-AD12). La tarjeta puede ser usada como un digitalizador Stand-Alone.

La comunicación con el resto de las tarjetas es a través de un bus de alta velocidad.

El módulo digitalizador contiene un convertidor Analógico Digital (A/D) de 8 o 12 bits. El formato de entrada es enteramente programable, para mayor flexibilidad acepta por separado las entradas Vertical, Horizontal y de reloj.

El convertidor A/D puede ser programado por software para muestrear a 10, 12 o a 15 MHz. La parte analógica del circuito es equipada con un filtro anti-Aliasing de 4.2 MHz. La entrada digital puede aceptar 8 o 12 bits por pixel en un rango máximo de 20 MHz.

La sincronización tiene que ser llevada a cabo con los standares de video RS-170, RS-330 y CCIR. La tarjeta puede también sincronizar las siguientes cámaras de alta resolución. Videk, EG&G Reticon modelo 9128, Fairchild modelo 100R.

Cuando se usa el modulo digitalizador, una de las más notables características de Oculus-500 es la separación total de canales de entrada y salida. Esta cualidad hace posible digitalizar imágenes en un rango de velocidad y desplegarlas y procesarlas en otros rangos distintos.

SOFTWARE

La tarjeta es soportada por el manejador OD5, el manejador permite la interface con el software del sistema central (OKS). Además permite la compatibilidad de software de la tarjeta Oculus-300.

Para aplicaciones orientadas a gráficas Coreco propone soportar el manejador TIGA (Texas Instrument Graphic Architecture) que es el standar para sistemas gráficos de media y alta resolución. El TIGA es soportado por compañias como Hewlett Packard, Microsoft, Media Cybernetics y Autodesk. Fue liberada al mercado a principios de 1990.

OCULUS-64 Y OCULUS RLE

Es importante notar que existen varias tarjetas, no digitalizadoras de imágenes, pero cumplen una función complementaria. Por ejemplo Coreco Inc. desarrolló una tarjeta de bajo costo para realizar transferencias rápidas de imagenes. OCULUS-64 maneja imágenes monocromáticas con resolución 512 H x 256 V o 256 x 256 pixels y transferirlas por un bus AT a razón de 10 por segundo. La tarjeta puede ser incluida en un sistema que realice "Análisis de Movimiento" (pisadas de un corredor, identificar contenedores en movimiento, etc). Oculus-64 es perfecta para aplicaciones de visión en computadora, que requieran capturar y analizar diversas imágenes tomadas en pequeños intervalos de tiempo. La tarjeta permite la transferencia de información hacia el Host durante el 90 % del ciclo de video, mientras otras solo usan el 15 o 20 % de dicho ciclo. Equipada con puerto dual de video RAM, recoge imágenes en 1/60 segundos a través del puerto serie, mientras coloca la información para su utilización en el puerto paralelo, se realizan dos operaciones simultaneamente . Cuenta con una resolución de 8 bits permitiendo 256 niveles de gris.

Existe otra tarjeta lanzada por Coreco Inc. La tarjeta RLE (Run Length Encoder) preprocesadora de imágenes capturadas y digitalizadas por una tarjeta Oculus 300. Dado que la codificación implica reducir una imagen en segmentos de linea, sin destruir información, se usa el método de compresión RLE para implementarla. Dentro de sus componentes físicos se encuentra un ALU que trabaja en tiempo real, además de implementar por hardware el proceso de la codificación. El objetivo de la tarjeta es optimizar una imagen para su transferencia, llegando a mandar 3 Mb por segundo.

2.2.2 DISPOSITIVOS DE OBTENCION DE IMAGENES

- STAR 1 SYSTEM (PHOTOMETRICS)

El sistema STAR 1 puede ser usado como un sistema digital completo, o puede ser controlado por una computadora host, el STAR1 incluye una cámara pequeña de bajo peso y un subsistema de control digital.

CAMARA STAR 1

El corazón de la cámara de sistema STAR 1 es un CCD de grado científico. Las imagenes pueden ser capturadas y eficientemente transferidas, en intervalos de tiempo pequeños y en tamaños variables dentro de una matriz de 384 x 576 pixels, manejando 12 bits para 4096 niveles de gris. El CCD es enfriado por un dispositivo termoeléctrico de tres etapas. El STAR 1 también incluye un obturador electrónico y un lente standar de 50mm; f/1.8.

SUBSISTEMA DE CONTROL DIGITAL

El controlador de la cámara opera desde una distancia mayor a 3 metros, es equipado con un mouse, un monitor de video y un puerto de comunicaciones IEEE-488.

El controlador de la camara STAR 1 contiene un scan electrónico y memoria RAM usados para digitalizar y almacenar una imagen respectivamente. El sistema de control implementado en FIRMWARE incluye todos los comandos de adquisición y despliegue, además de contar con comandos de transferencia y almacenamiento de datos en una computadora host o en una cinta de cartucho.

Opcionalmente el sistema puede contar con software habilitado en IBM PC AT/XT o MACINTOSH II. El software opcional pueda manipular los datos de la imagen digital obtenidas por el STAR 1. La imagen puede ser almacenada en el host en 12 bits o en formato TIFF.

- PHOTOMETRICS 286/386 DIGITAL CCD WORKSTATION

Es un sistema programable de alto rendimiento en sistemas digitales, equipado con CCD de tecnología de enfriamiento, para bus AT de computadoras 286/386. La tecnología de enfriado proporciona alto rendimiento comparado con los CCD convencionales. El sistema es diseñado para aplicaciones de investigación, ofreciendo bajo nivel de ruido, respuesta lineal y capacidad para bajos niveles de luz.

La mayoría de las cámaras cuentan con un rango dinámico de niveles de gris de 8 bits por pixel, el DIGITAL CCD WORKSTATION es configurado con 4096 nivels de gris 12 bits, o para 16384 niveles de gris 14 bits o para 65536 niveles de gris 16 bits por pixel.

La cámara es equipada con una tarjeta controladora programable conectada directamente al bus AT para almacenar la imagen en memoria del host (se requiere memoria extendida para imagenes mayores de 512 x 512).

La cámara cuenta con un lente de 50mm; f/1.8.

El sistema incluye librerías compiladas en lenguaje "C" para control de la cámara, de manera tal que el usuario pueda escribrir sus propias aplicaciones. Comandos del host pueden ser usados para programar el microprocesador (Motorola 56001) de la tarjeta.

-SCANMASTER 3 (HOWTEK)

Es un scan digital de color de alto rendimiento para medio ambiente profesional. El SCANMASTER 3 presenta un avance en digitalización en color combinando calidad de imagen, fiabilidad y mayor exactitud en el realce del barrido. El SCANMASTER 3 utiliza dispositivos CCD, utiliza iluminación fluorescente sencilla. La arquitectura incluye un procesador DSP (Digital Signal Processor) encargado de procesar los algoritmos que requieren de velocidad y calidad. Los rangos de resolución de barrido varian de 75 a 400 dpi con respuesta espectral exacta. La conversión analógica utiliza 24 bits; 8 bits por cada color Rojo, Verde y Azul (RGB) y para barrido blanco y negro (256 niveles de gris por color).

Las características del SCANMASTER permiten exactitud de barrido, estabilidad, detalles de sombra y descriminación de color. El SCANMASTER utiliza una interface IEEE-488.

El software incluido en el SCANMASTER puede ser usado en IBM PC o PS/2, MACINTOSH y WORKSTATION. Incluye un manejador de mouse.

La imagen final puede ser salvada en formato TIFF, PICT, PICT32, TGA, RIFF, ARTISAN y SUNRASTER.

CAPITULO TRES

ELECCION Y DESCRIPCION DE RECURSOS DE COMPUTO

3.1 INTRODUCCION

En el desarrollo de todo sistema de cómputo es indispensable definir los requerimientos de software, hardware y de equipos y sistemas externos para que en base a ellos se pueda contar con un diseño que sea fácil de elaborar, entendible, óptimo y de bajo costo, tomando en cuenta los recursos con los que se dispone.

De acuerdo al planteamiento del problema y a la formulación de objetivos requerimos un sistema con las siguientes herramientas:

1.- Modulo de obtención y digitalización de imágenes
2.- Software externo para la manipulación del modulo digitalizador
3.- Equipo de cómputo para almacenamiento y procesamiento
4.- Tarjeta gráfica para desplegar imágenes (VGA)
5.- Lenguaje de Programación.

Si se cuenta con las imágenes ya digitalizadas y almacenadas en dispositivos externos de memoria, nuestro sistema gráfico excluye los dos primeros puntos.

En el presente capítulo definiremos y describiremos todas las herramientas con las que se implementará el sistema.

3.2 MODULO DE OBTENCION Y DIGITALIZACION DE IMAGENES

El módulo de obtención y digitalización de imágenes consta de lo sigiente (figura 3.2.1):

1.- Cámara de video
2.- Monitor de T. V.
3.- Sistema electrónico de digitalización
(Tarjeta digitalizadora, monitor RGB y mouse óptico)
4.- Microcomputadora PC-AT.

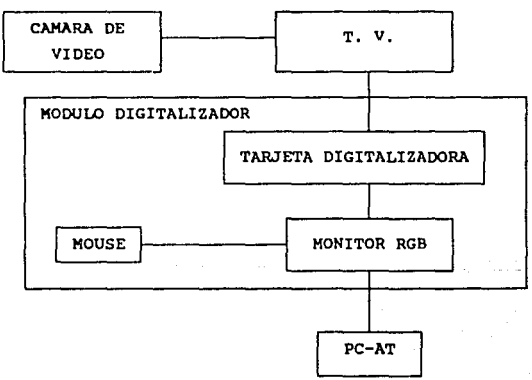


FIGURA 3.2.1 MODULO DE OBTENCION Y DIGITALIZACION DE IMAGENES.

El sistema anterior excepto el primer punto es conocido como INOVION.

3.2.1 CAMARA DE VIDEO

La cámara de video es el dispositivo encargado de capturar las imágenes y transferirlas por medio de una señal de video a la fase de digitalización.

La cámara utilizada en la digitalización de las imágenes procesadas por el sistema, tiene las siguientes características:

- Video-Cámara/Magnetófono 8 mm Handycam Pro Sony
- Video captor: CCD (Coupled Charge Device)
- Visor: Electrónico 7.5 in b/n
- Lente de 1.6 con macro
- Objetivo: Combinación de zoom de 6 aumentos
  Foco=12-72 mm
- Videoseñal: Color NTSC (National Television System Comitte).

NOTA: El sistema NTSC en lugar de transmitir señales RGB en secuencia, separa la información en señales de luminancia y crominancia que se transmiten simultáneamente con multiplexión de frecuencia.

3.2.2 MONITOR DE T.V.

El monitor es un dispositivo analógico, usado para desplegar la imagen que proporciona la cámara de video. La finalidad es seleccionar de una misma imagen, la que tenga mejor enfoque, abarque un amplio rango de tonos de gris y presente niveles mínimos de ruido para que posteriormente sea digitalizada.

El monitor de T.V. es un Zenith Space Command Receiver/Monitor tiene las siguientes caracteristicas:

- Color 19"
- Modificado por INOVION Co.
- Perillas de ajuste (color level,tint,sharpness,picture)

3.2.3 SISTEMA ELECTRONICO DE DIGITALIZACION

Sistema central del módulo de obtención y digitalización de imágenes, compuesto por una tarjeta digitalizadora, monitor RGB y mouse óptico.

Es el encargado de convertir la imagen capturada mediante la cámara y el monitor de T.V. a imagen digital. Discretizando la señal analógica en puntos que contendran información de intensidad o color.

Cabe hacer mención que el monitor RGB es capaz de contener una imagen digitalizada de 512 x 512 pixels, manejando 127 tonos de gris.

La tarjeta digitalizadora consta de:

- Un procesador 6502 con 8k de RAM exclusivo para procesamiento y 32 de EPROM, con dos puertos para interface.

- Ocho módulos de memoria cada uno de 64 kb.

- Interface para monitor RGB, mouse y microcomputadora.

- Convertidor D/A

- Señal NTSC

3.2.4 MICROCOMPUTADORA PC-AT

La microcomputadora PC-AT basada en un procesador 8086, con 640 Kb de RAM, tarjeta gráfica PGC (Personal Graphics Card) monitor de alta resolución, 256 colores y disco duro de 20 Mb es la encargada de manipular las imágenes dentro del sistema INOVION y almacenarlas en sus dispositivos de memoria, por medio del software externo. Además cuenta con programas -realizados por investigadores del Instituto de Física de la U.N.A.M- capaces de desplegar y preprocesar las imágenes para futuras aplicaciones.

3.3 SOFTWARE EXTERNO

Una vez que la imagen ya esté desplegada en el monitor del sistema digitalizador, es necesario almacenar la información de la tarjeta en un dispositivo de memoria, por lo que se requiere de un programa capaz de manipular y transferir los datos al mismo, para que sea procesada posteriormente.

El software externo para leer la imagen de la memoria de la tarjeta digitalizadora es proporcionado por INOVION Co. Consta de diversas opciones, como son:

- Lectura de zonas de tamaño variable contenidos en la memoria de la tarjeta, para que sean transferidas a una microcomputadora.

- Modificar información contenida en la tarjeta.

- Ampliación del histograma.

- Manipulación de los tonos R G B. Etc.

El programa almacena los datos en tres etapas. Primero, lectura

de un bloque de información de la tarjeta, segundo: transferencia de los datos secuencialmente a través del puerto serie hacia una microcomputadora. Por último: escritura del bloque al dispositivo de almacenamiento de la computadora.

El programa transfiere seis bytes de encabezado para control, que proporciona información del tamaño de la imagen,. Este formato es propio, conocido como INOVION, la extención del archivo es .INO.

El factor más importante para decidirnos a utilizar el sistema INOVION es que almacena las imágenes en un formato que es muy sencillo de manipular, al contar únicamente con las intensidades de cada punto en código ASCII. Además cuando se quiera procesar imágenes con otro formato será fácil modificar, dentro del sistema que se desarrolló, el módulo de entrada para que acepte la imagen, sin necesidad de afectar la filosofía del mismo.

3.4 EQUIPO DE COMPUTO PARA ALMACENAMIENTO Y PROCESAMIENTO

Para la realización del Sistema de Procesamiento Gráfico en Angiología Retiniana se cuenta con el siguiente equipo de cómputo:

- Microcomputadora PS:

Características

- Procesador 80286
- Coprocesador 80287
- Disco duro de 40 Mb
- 1 Mb de memoria RAM
- Unidad de disco de 3.2" de 1.44 Mb
- Tarjeta gráfica VGA
- Monitor de color VGA
- Puerto de entrada para mouse

Ventajas

- Capacidad de memoria para manejar dos imágenes de 64 k.
- Amplio rango de colores y niveles de gris
- Velocidad de operación 10 Mhz
- Facilidad para programación de la tarjeta gráfica
- En modo óptimo resolución de 640 X 480 pixels

Desventaja

- El sistema solo es compatible con modo gráfico VGA

3.5 TARJETA GRAFICA PARA DESPLEGAR IMAGENES (VGA)

En todo sistema de procesamiento de imágenes, una buena elección de la tarjeta gráfica es fundamental. Un sistema gráfico adecuado permite una mejor nitidez, una mayor resolución, una amplia gama de colores y con todo esto se podrá procesar la mayor cantidad posible de información contenida en la imagen, además de una mejor presentación.

Video Graphics Array (VGA) es un standar gráfico para computadora desarrollado por IBM para la familia Personal System/2 (PS/2). Anteriormente se contaban con otros sistemas gráficos que exhibian pobre resolución y colores limitados. El Monocromatic Display Adapter (MDA) y Color Graphics Adapter (CGA) fueron los primeros standares para las microcomputadoras, basados en el chip 6845 CRT con resolución de 40 u 80 columnas y 25 renglones en texto y 320 por 200 pixels en modo gráfico con hasta cuatro colores en CGA, manejados por cuatro bits (Intensidad, R, G, B). Posteriormente, surgió el MCGA (Multicolor Graphics Array) que es similar en función con el CGA pero son escasamente compatible en BIOS, en registros de control y en niveles de memoria de display.

El MGA es equipado con un conjuto de 256 registros de color idénticos a los de VGA, almacenando 6 bits de resolución por cada uno de los tres colores primarios RGB, proporcionando hasta 256 K colores.

La tarjeta Monocromática Hércules continuó con los standares, basada en el Motorola 6845, con 80 columnas por 25 renglones en modo alfanumérico y una relativa alta resolución de 720 por 348 pixels en modo gráfico. Con siete bits maneja video normal, blink, subrayado y video inverso.

Posteriormente apareció el Professional Graphics Adapter (PGA), capaz de operar en resolución gráfica con 640 por 480 y 256 colores simultaneos; basada en el 6845 y el 8088.

El Enhanced Graphics Adapter (EGA) desarrollado para la familia de PC es el predecesor del VGA, parecidos en características pero con menor resolución y manejo de colores en ciertos casos. (tabla 3.5.1 resolución y colores de VGA y EGA).

TABLA 3.5.1

MODO	ADAPTADOR	RESOLUCION (pixels)	NUMERO DE COLORES
MONO	EGA/VGA	640 POR 200	2
MONO	EGA/VGA	640 POR 350	4
MONO	VGA	640 POR 480	2
COLOR	VGA	320 POR 200	256
COLOR	EGA/VGA	640 POR 200	16
COLOR	EGA/VGA	640 POR 350	16
COLOR	VGA	640 POR 480	16
COLOR	EGA/VGA	320 POR 200	16

VGA es implementada con circuitos VLSI (Very Large Scale Integrated) compatibles con varias tarjetas gráficas de PC, por consiguiente son de bajo costo, alta velocidad y densidad de memoria y bajo costo de mantenimiento.

La tarjeta VGA cuenta con su propio BIOS, escrito en memoria ROM, dando con esto mayor compatibilidad en manejadores de bases de datos, hojas de cálculo, utilerias, etc. VGA tiene un bus interno de hasta 32 bits. Su memoria RAM es usada para almacenar texto y/o datos gráficos. Sus dos tipos de memoria usan 16 bits de direccionamiento.

El VGA BIOS ROM contiene cinco distintos tamaños de caracteres: 8 x 8, 8 x 14, 9 x 14, 8 x 16 y 9 x 16.

Ciertamente el aspecto más complicado en VGA es comprender el control de sus registros que son divididos en cinco grupos. (tabla 3.5.2). El tamaño de cada registro es de un byte y algunos de ellos son subdivididos en campos. Son accesados a través de direcciones de puertos de Entrada/Salida.

NOMBRE	NUMERO DE REGISTROS
- Registros Generales	4
- Registros Secuenciales	5
- Registros de control CRT	28
- Registros de Control de Gráficos	9
- Registros de Atributo	22
- Registros de Color	5

TABLA 3.5.2 REGISTROS DE V.G.A.

Las tarjetas VGA pueden ser configuradas en una gran variedad de formatos de display. Los formatos controlan la configuración de la memoria de display, la resolución, el número de bits por pixel, el tipo de Font de default y la dirección de inicio de la memoria de display. Tabla 3.5.3 y tabla 3.5.4.

RESOLUCION	BITS/PIXEL	CANTIDAD DE MEMORIA (BYTES)
320 x 200	1	8000
320 x 200	2	16000
320 x 200	8	64000
640 x 200	2	32000
640 x 350	1	28000
640 x 350	2	56000
640 x 350	4	112000
640 x 480	1	38400
640 x 480	4	153600

TABLA 3.5.3 CANTIDAD DE MEMORIA DE DISPLAY

MODO (HEX)	ADAPTADOR	COL/FILAS	DIRECCION	TAMAÑO FONT
0,1	EGA/VGA	40 x 25	B8000	8 x 8
0,1	EGA/VGA	40 x 25	B8000	8 x 14
0,1	VGA	40 x 25	B8000	9 x 16
2,3	EGA/VGA	80 x 25	B8000	8 x 8
2,3	EGA/VGA	80 x 25	B8000	8 x 14
2,3	VGA	80 x 25	B8000	9 x 16
7	EGA/VGA	80 x 25	B0000	9 x 14
7	EGA/VGA	80 x 25	B0000	9 x 16

TABLA 3.5.4 TAMAÑO DE FONT: MODO ALFANUMERICO

En un modo apropiado VGA puede simultáneamente desplegar hasta 256 colores de 262,144 paletas en un monitor a color. Los 16 colores con los que cuenta la primera paleta estan predefinidos por el driver .BGI, esta paleta es la de default. Las primeras 256 paletas (numeradas de 0.. 255) pueden ser individualmente definidas por tres diferentes argumentos: rojo, verde y azul (R,G,B) utilizando únicamente los 6 bits más significativos del byte (LSB). (Tabla 3.5.5).

REGISTRO DE PALETAS	VALOR COLOR (DEFAULT) HEX	6-BIT	BINARY	COMPONENTE COLOR	NOMBRE COLOR
0	0	000	000	- - - - - -	NEGRO
1	1	000	001	- - - - - B	AZUL
2	2	000	010	- - - - G -	VERDE
3	3	000	011	- - - - G B	CYAN
4	4	000	100	- - - R - -	ROJO
5	5	000	101	- - - R - B	MAGENTA
6	14	010	100	- g - R - -	CAFE
7	7	000	111	- - - R G B	GRIS-INTENSO
8	38	111	000	r g b - - -	GRIS-CLARO
9	39	111	001	r g b - - B	AZUL-INTENSO
A	3A	111	010	r g b - G -	VERDE-INTENSO
B	3B	111	011	r g b - G B	CYAN-INTENSO
C	3C	111	100	r g b R - -	ROJO-INTENSO
D	3D	111	101	r g b R - B	MAGENTA-INT.
E	3E	111	110	r g b R G -	AMARILLO
F	3F	111	111	r g b R G B	BLANCO

TABLA 3.5.5 PALETA DE DEFAULT V.G.A.

Existen varias tarjetas VGA que presentan mejoras en funcionamiento, pero que posiblemente en programas de aplicación realizados en VGA standar estas no sean compatibles y la aplicación no funcione. La tabla 3.5.6 muestra las características de algunas tarjetas VGA.

CARACTERISTICAS / TARJETAS	VGAWONDER	SUPERVGA 5300/5400	PRODESIGNER VGA PLUS
COMPAÑIA	ATI Technologies	Genoa Systems Corp.	Orchid Technology
MAXIMA RESOLUCION* DE TEXTO	132 X 44	132 X 60; 100 X 75	132 X 42
GRAFICOS (pixel x pixel x color)	800 X 600 X 256 1024 X 768 X 16	800 X 600 X 256 1024 X 768 X 16	800 X 600 X 256 1024 X 768 X 16
DRIVER	AutoCAD/Shade; GEM 2 Y 3 Lotus Symphony; Windows 1.X,2.X y Windows/286 2.X	AutoCad/Shade; GEM 2 y 3; VenturaPublisher 1.1 y 2.0; Framework II 1.1 Lotus y Symphony Windows 286 1.03 y 2.X; Windows/386 2.03	AutoCAD/Shade; GEM Desktop 1.0; Lotus y symphony Ventura Publisher 1.X; Windows 1.x y 2.03; WordPerfec 5.0

* PARA TODOS LOS ADAPTADORES LA MAXIMA RESOLUCION GRAFICA REQUIEREN DE 512 Kb DE MEMORIA.

TABLA 3.5.6 TARJETAS V.G.A.

CARACTERISTICAS / TARJETA	VGAWONDER	SUPERVGA 5300/5400	PRODESIGNER VGA PLUS
CARACTE- RISTICAS	Incluye mouse Microsoft; niveles de registros adicinales compatibles con MDA, CGA, EGA y Hércules; soporta monitores analógicos y digitales	Soporta: monitores Analógicos o Digitales; Nivel de registros adicional compatibles con MDA, CGA, EGA y Hercules.	Programa editor-Font; Nivel de registros adicional compatibles con MDA, CGA, EGA y Hércules.
TAMAÑO DE LA TARJETA EN PULGADAS	$8\frac{1}{2}$ X $4\frac{1}{5}$	$8\frac{1}{2}$ X $4\frac{1}{5}$	$13\frac{1}{4}$ x $3\frac{1}{5}$
DOCUMENTACION	Manual de Usuarios	Manual de Usuarios	Manual de Hw y Manual de Sw
PRECIOS	Versión 256 Kb $ 499 Versión 512 Kb $ 699	Modelo 5300 (256 Kb) $ 449 Modelo 5400 (512 Kb) $ 699	Versión 512 Kb $ 599

TABLA 3.5.6 TARJETAS V.G.A. (CONTINUACION)

3.6 LENGUAJE DE PROGRAMACION

ALTERNATIVAS

- Turbo Pascal

  Características

  - Lenguaje de alto nivel,
  - Lenguaje estructurado,
  - Amplio manejo de gráficos,
  - Popular en medio ambiente de PC, PS/2 y compatibles,
  - Alta rapidez en código de ejecución,
  - Fácilmente combinado con lenguaje ensamblador,
  - Alta velocidad en compilación,
  - Fácil de escribir y de entender,
  - Diversos tipos de datos (bit, byte, word, etc.),
  - Fácil manejo de string (cadenas),
  - Poco transportable.

- Turbo Basic

  Características

  - Lenguaje de alto nivel,
  - Amplio manejo de gráficos,
  - Popular en medio ambiente de PC y compatibles,
  - Fácil de escribir y de entender,
  - Lento en código de ejecución,
  - Poca transportabilidad.

- Fortran

Características

- Lenguaje de alto nivel,
- Lenguaje orientado a programas científicos,
- Fácil manejo de fórmulas,
- Poco manejo de gráficos,
- Poco popular en medio ambiente de PC y PS/2,
- Restringido en tipo de datos,
- Es complicado el manejo de string,
- No es transportable.

- Turbo "C" Versión 2.0 Borland

Características

- Lenguaje de alto nivel,
- Lenguaje estructurado,
- Amplio manejo de gráficos,
- Optimo uso de funciones gráficas en EGA/VGA,
- Popular en medio ambiente de PC, PS/2 y compatibles,
- Alta rapidez en código de ejecución,
- Fácilmente combinado con lenguaje ensamblador,
- Fácil de escribir y de entender,
- Variedad en tipos de datos,
- Fácil manejo de string (cadenas),
- Transportable,
- Lento en el proceso de compilación y ligado (link).

Tomando en cuenta las características de los distintos lenguajes de programación que se tenían como alternativa, los lenguajes que facilitan la realización del sistema son : PASCAL y "C".

Los puntos a considerar para la elección del mejor lenguaje entre las dos opciones finales fueron las siguientes:

- Tiempo en compilación y/o ligado
  Mejor opción: PASCAL

- Uso de strings (cadenas)
  Mejor opción: PASCAL

- Manejo de funciones gráficas
  Mejor opción: "C"

- Tipos de datos
  Mejor opción: "C"

- Transportabilidad
  Mejor opción: "C"

- Generación de código ejecutable
  Mejor opción: "C"

Tomando en cuenta los puntos considerados, el lenguaje elegido para el sistema: Procesamiento Gráfico en Angiología Retiniana es "Turbo C Versión 2.0 Borland".

Por lo tanto el sistema se realizará con las siguiente herramientas:

-Digitalizador: Sistema Inovion.

-Microcomputadora: PS-2.

-Tarjeta Gráfica: VGA.

-Lenguaje de programación: Turbo "C" Versión 2.0 de Borland.

CAPITULO CUATRO

SISTEMA ADMINISTRADOR

En todo sistema existe una sección encargada de llevar el control de todos los subsistemas que forman parte de él . Por lo que en este capítulo presentamos una descripción del administrador general, al que hemos denominado Sistema Administrador de Imágenes Digitales (SAID).

SAID tiene como función principal la de controlar y ejecutar las opciones elegidas por el usuario, además, controla los recursos del equipo como son memoria principal (RAM): asignación de memoria para vaciar y manipular la información de las imágenes digitales; memoria secundaria (discos): para el almacenamiento de las imágenes procesadas, información asociada a ellas, espacio libre en la misma, así como la información de ayuda correspondiente a cada comando.

4.1 DESCRIPCION

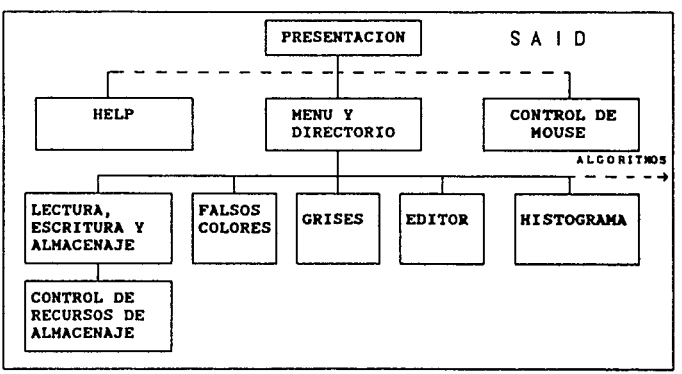
El Sistema Administrador de Imágenes Digitales (SAID), maneja hasta dos imágenes de 256 X 256 pixels que se encuentran en memoria principal. Las imágenes serán leídas de disco, previamente digitalizadas en formato inovión con extención ".INO".

Se podrá realizar operaciones en una o en ambas imágenes, presentando el resultado de esta en la sección contraria a la imagen habilitada. Las imágenes podrán ser salvadas y leídas en dos tipos de formatos, el formato INO mencionado anteriormente y el formato IMG, que presenta las características de ser corto, que permite una lectura y/o escritura en forma rápida. IMG es válido únicamente para presentaciones.

SAID permitirá la elaboración de un expediente que describa las características de cada una de las imágenes. El expediente es almacenado en un archivo en disco con el mismo nombre de la imagen leída pero con extensión ".HIS". Si este ya existe, el sistema permitirá editarlo y modificarlo.

Además cuenta con comandos que permiten cambiar las imágenes a tonos de gris y falsos colores.

SAID facilita la labor del usuario, cuenta con ayuda para cada uno de los comandos que maneja, presionando la tecla F1 antes de ejecutar el comando elegido. Presenta mensajes de error previniendo el mal uso del paquete, permitiendo reintentar la operación mal ejecutada. La tecla ESC (escape) ha sido activada para abortar comandos u operaciones escogidos erroneamente o cuando ya no se desea proseguir. El siguiente esquema presenta un sistema de bloques de los cuales se compones SAID.



4.2 PRESENTACION

El sistema divide al monitor de la computadora en nueve secciones. La figura 4.2 muestra la distribución de la pantalla.

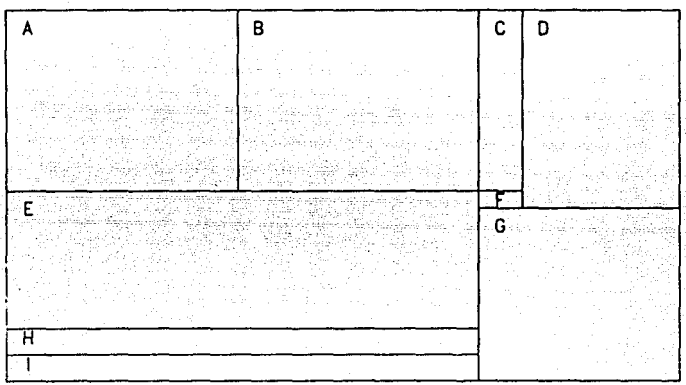


FIGURA 4.2 DISTRIBUCION DEL MONITOR

A continuación describimos cada una de las nueve subdivisiones que realiza SAID para presentación del sistema.

SECCION A: Lugar donde se coloca la imagen original, leida de disco con formato INOVION. El tamaño de la imagen es de 256 x 256 pixels con hasta 128 niveles de gris distintos. También se colocará el resultado de una operación realizada sobre la imagen colocada en la sección "B".

SECCION B: Posición donde se coloca una imagen afectada por una operación realizada sobre la imagen de la sección A.

SECCION C: Indicador de avance de los procesos en ejecución.

SECCION D: Area del menú, en donde se puede seleccionar los comandos u operaciones que se pueden realizar sobre las imágenes. Con las teclas "+" y "-" del teclado número o presionando la letra de cada comando que se encuentre en mayúsculas.

SECCION E: Lugar destinado a propósito general: Expediente clinico (editor gráfico), histograma, gráficas etc.

SECCION F: Area que indica que imagen se encuentra activa, colocando la letra "A" si se trata de la imagen de la sección 'A' o colocando la letra "B" si es la de la sección 'B'.

SECCION G: Estatus del sistema, contiene los parámetros de: intensidad de un punto de la imagen, localización coordenadas (x,y) del mismo, distancia y angulo entre dos puntos, nombre de la imagen de la seccion "A". Los puntos en cuestión estarán representados por un cursor. Se mostrará una amplificación de la zona alrededor del cursor.

SECCION H: Región de mensajes proporcionados por el sistema (errores, teclas a utilizar etc..).

SECCION I: Zona de entrada de datos, proporcionados por el usuario para la ejecución de los comandos y algoritmos.

4.3 MODULOS HELP Y MOUSE

Se desarrollaron dos rutinas para facilitar el uso del sistema, módulos Help y Mouse. El primero permite al usuario, mediante la pulsación de la tecla F1, accesar información de ayuda para la ejecución de cada uno de los comandos, mostrandose en la sección "E" de la pantalla de presentación. El llamado de la rutina puede .er en cualquier momento sin alterar de ningún modo el desarrollo del comando.

La rutina de Mouse cuenta con un cursor de movimiento rápido capaz de desplazarse a través de la seccion de imágenes, el cursor proporciona información del pixel señalado, actualizando los parámetros en el estatus. Además, permite seleccionar puntos necesarios para la ejecución de algunos comandos. Así como para escoger una opción de una lista (Directorio de archivos, Menú de comandos, respuestas, etc..).

La rutina unicamente usa el mouse Microsoft (2 teclas), La habilitación se realiza mediante la llamada a la interrupción 33H, que es una interrupción de software al procesador.

4.4 LECTURA, ESCRITURA Y ALMACENAMIENTO DE IMAGENES

Para procesar imágenes es necesario contar con mecanismos de lectura, escritura y almacenamiento de las mismas. El ciclo anterior permite gran versatilidad a los sistemas y es necesario manejarlo adecuadamente para un mejor rendimiento y de alguna o de otra forma que permita la interacción con otros sistemas y equipos.

SAID cuenta con cuatro rutinas para el manejo de imágenes, dos exclusivamente para lectura y desplegado en pantalla de las mismas, y las otras dos para almacenamiento en dispositivos externos como son disco duro y flexible. Se presenta a continuación una descripción donde se enfatizan las diferencias entre las rutinas anteriores.

4.4.1 RUTINAS DE LECTURA Y DESPLEGADO DE IMAGENES

SAID cuenta con dos rutinas de lectura y desplegado de imágenes. La primera lee la imagen almacenada en disco en formato INOVION, el formato consta de un encabezado de seis bytes. La rutina verifica esta información, si la imagen es de 256 X 256 pixels continua con la operación, en caso contrario mandando un mensaje de incompatibilidad de archivo.

El bucle de lectura itera 65536 veces colocando los pixels en pantalla de la sección "A",en hileras de 256.

La segunda rutina de lectura es más sencilla en su implementación, al ejecutar comandos propios del lenguaje que permiten la lectura de manera rápida de imágenes con formato IMG. El formato IMG es un bloque de información compactada que hace el tiempo de acceso a disco más óptimo tanto para lectura como escritura, colocando la imagen completa en un solo instante. La colocación de la imagen será en el lado opuesto a la imagen activada.

4.4.2 RUTINAS DE ALMACENAMIENTO DE IMAGENES A DISCO

La primera rutina almacena en memoria secundaria la imagen seleccionada, leyendo la sección de la memoria que la contiene, agregando al inicio los seis bytes de encabezado que son parte del formato INOVION, para que dispositivos compatibles con este formato puedan hacer uso de las imágenes y que además SAID pueda procesarlas nuevamente.

El otro modo de almacenamiento consiste en utilizar el formato IMG con la desventaja de no poder usarlo directamente para procesamiento.

Para almacenar una imagen sea por cualquier método, SAID verifica el espacio disponible del dispositivo de memoria externo. Desplegando un mensaje de error en caso de faltar de espacio.

EL pseudocódigo que realiza la lectura, desplegado y almacenamiento de las imágenes se presenta a continuación:

LECTURA DE IMAGENES.INO:

```
INICIO
   Elegir imagen
   verificar existencia del archivo
   IF (existe)
     lectura del encabezado
     i=0
     j=0
     IF (compatible)
       REPITE
         lectura de byte (pixel)
         matriz1[i,j]=byte
         escalar byte a tonos de 0-127
```

```
        (*)colocar pixel en posicion i+2,j+2
           IF i>=255
             i=0
             j=j+1
           ENDIF
         HASTA EOF(archivo)
       ELSE
         mensaje error
       ENDIF
     ELSE
       mensaje error
     ENDIF
   FIN (Procedimiento)
```

(*) El incremento de dos se debe a que la imagen empieza a colocarse en la posición 2 absoluta sobre la pantalla; 0 y 1 son parte del marco de presentación.

ESCRITURA DE ARCHIVOS .INO

```
   INICIO
      Proporcionar NOMBRE_ARCHIVO a salvar
      IF (espacio_disco)
         escribir 6 byte de encabezado
         i=0, j=0
         REPITE
            byte=matriz1[i,j]
            escribir al archivo (byte)
            IF (i>=255)
               i=0
               j=j+1
            ENDIF
         HASTA j>255
      ELSE
```

```
        mensaje error
      ENDIF
    FIN (Procedimiento)
```

LECTURA DE IMAGENES.IMG

```
     INICIO
       proporcionar archivo a leer
       IF (existe)
         size= tamaño archivo
         buffer= alloc_memoria[size] (asignación de memoria)
         buffer=archivo
     (*) desplegar imagen en pantalla(x,y)
       ELSE
         mensaje error
       ENDIF
       libera memoria
    FIN (Procedimiento)
```

(*) x,y: coordenadas donde se colocará el inicio de la imagen.

ESCRITURA DE IMAGENES.IMG

```
    INICIO
       nombre de la imagen a salvar
       posición de la imagen(left,top,right,bottom)
       size=tamaño imagen
       buffer=alloc_memoria[size] (asignación de memoria)
       IF (espacio_disco)
         WRITE(archivo,buffer)
       ELSE
         error
       libera_memoria
    FIN (Procedimiento)
```

4.5 FALSOS COLORES

Dentro del procesamiento de imágenes existen técnicas de realce de detalles que a simple vista no son tan fáciles de identificar. Existen algoritmos complejos que realizan esta función, para diferente tipos de imágenes, pero en algunos de los casos no es necesario usarlos. Una forma de resaltar detalles de manera sencilla es la de utilizar falsos colores, que aunque algunas veces enmascara rasgos de la imagen, en otros permite identificar detalles necesarios para el análisis o procesamiento de la misma.

Lo anterior es posible gracias a que los equipos cuentan con un display que permiten observar una diversidad de colores, entra aquí el concepto de Paleta (Palette en Inglés), que es el máximo número de colores que pueden ser desplegados en un mismo tiempo.

La selección de paletas que permiten el cambio de colores para realzar detalles se hizo de manera secuencial usando 15 paletas diferentes en modo VGA, porque al realizar pruebas con distintas paletas en las diversas imágenes se observó que al menos una de ellas resaltaba los detalles no visibles a simple vista y esta paleta no es la misma para todas.

El código siguiente es el encargado de realizar esta función:

```
/*-------------------------------------------------------------*/
/* FUNCION PARA CAMBIAR LAS PALETAS                            */
/*  num=  número  de  paleta para el cambio a falsos colores */

void paleta(num)
int num;
{ int colors;
```

CONTINUACION

```
if (num!=1){
for (colors=0; colors<15; colors++){
 if (colors > 7 ) setrgbpalette(colors+48,colors*num*1,
                                colors*num*10,color*num*20);
 else
  setrgbpalette(colors,colors*num*1,colors*num*10,colors*num*20);}
 colors--;
 setrgbpalette(20,colors*num*1,colors*num*10,colors*num*20);}
else{

/* PALETA ORIGINAL (VALORES DEL REGISTRO DE COLOR  (VGA) */

  setrgbpalette(0,0,0,0);              setrgbpalette(38,21,21,21);
  setrgbpalette(1,0,0,42);             setrgbpalette(57,21,21,63);
  setrgbpalette(2,0,42,0);             setrgbpalette(58,21,63,21);
  setrgbpalette(3,0,42,42);            setrgbpalette(59,21,63,63);
  setrgbpalette(4,42,0,0);             setrgbpalette(60,63,21,21);
  setrgbpalette(5,42,0,42);            setrgbpalette(61,63,21,63);
  setrgbpalette(20,42,21,0);           setrgbpalette(62,63,63,21);
  setrgbpalette(7,42,42,42);           setrgbpalette(63,63,63,63);}
   }
```

4.6 TONOS DE GRIS

En el área de procesamiento gráfico contar con software capaz de manejar tonos de gris es indispensable, debido a la gran información que proporciona esta coloración de la imagen. "SAID" cuenta con una opción de grises.

Esta función transforma la imagen que se encuentra en pseudocoloración a tonos de gris, el cambio de tonos se hace mediante la llamada de una función propia de Turbo C en donde son

manejados los tonos RGB. Dado que utilizamos tarjeta VGA en modo de 16 colores la escala de grises es de 0-15.

4.7 EDITOR

En algunas ocasiones el contar con gran cantidad de información de un objeto provoca confusión o perdida de ella. Es necesario contar con un sistema que permita almacenar la información que describa o identifique las características del mismo.

SAID tiene un editor, que tiene como objetivo principal la de capturar un expediente que defina las características de la imagen, permitiendo manejar dinámicamente las imágenes como los expedientes de todos los pacientes, reduciendose con esto gran parte del espacio y trabajo que se necesita para tener archivado en papel esta misma información.

Dentro de las características que presenta el editor se encuentran las siguientes:

- Editor de 15 renglones por 70 columnas, lo que permite almacenar en disco un historial completo y facilmente modificable de la imagen cargada en memoria principal.

- Permite las funciones básicas de un editor como son borrado de caracteres, borrado de lineas, modo inserción, salto manual y automático de renglón, acceso a cualquier parte de la sección de la pantalla destinada al editor por medio del uso de las teclas de dirección. Cabe destacar que la implementación del editor antes mencionado se uso el modo gráfico en el ambiente de Turbo C, lo que permite estar observando y describiendo la imagen al mismo tiempo.

Pseudocódigo del editor:

```
INICIO
 proporcionar nombre del archivo a editar
 IF (existe)
   REPITE
     linea[i]=linea(archivo)
     desplegar en pantalla renglón[i]
     i=i+1
   HASTA EOF(archivo)
   renglón=0
   columna=0
   REPITE
     ch=read(tecla)
     IF (ch) es
        flecha_arriba: IF (renglón > 0) renglón=renglón-1
                       ELSE  beep
        flecha_abajo:  IF (renglón < tot_renglones)
                           renglón=renglón+1
                       ELSE  beep
        flecha_izq:    IF (columna > 0) columna=columna-1)
                       ELSE  beep
        flecha_der:    IF (columna < tot_columnas)
                           columnas=columnas+1
                       ELSE  beep
        CR :  IF (renglón < tot_renglones)
                renglón=renglón+1
                columna=0

        BS :  IF (columna > 0)
                borra caracter a la izquierda
              ElSE  beep
        DEL:  IF (columnas > 0)
                borra caracter a la derecha
```

```
                ELSE  beep
        CTRL Y:  IF (existen lineas)
                    borrar linea[renglón]
                    actualiza posición de lineas
                 ELSE  beep
      CARACTER
   ALFANUMERICO
O DE PUNTUACION:  IF (columna < tot_columnas)
                    linea[renglón,columna]=caracter
                    desplegar linea actualizada

           INS:
            REPITE
             ch=read(tecla)
             IF (ch) es
                CARACTER:
                        IF (linea_no_llena)
                          abrir un espacio en linea
                          linea[columna,renglón]=caracter
                          desplegar linea actualizada
                          columna=columna+1
                        ELSE   beep
                     CR:
                        IF (espacio p/nuevas lineas)
                              abrir nueva linea
                              actualizar linea
                              renglón=renglón+1
             ENDIF
            HASTA (ch=ESC) o (renglón > tot_renglón)
          ENDIF
      HASTA (ch=ESC) o (renglón > tot_renglón)
   FIN (procedimiento)
```

4.8 HISTOGRAMA

Una de las herramientas más simples y usadas en el procesamiento digital de imágenes es el histograma. Esta función resume el contenido de niveles de gris de una imagen. El histograma contiene considerable información y cada una de ellas solo cuenta con uno.

Como sabemos el histograma es una función que muestra, para cada nivel de gris, el número de pixels que contiene la imagen, pero no nos da información de la posición del mismo. La abcisa representa las intensidades y la ordenada, la frecuencia o número de pixels.

SAID puede obtener el histograma de 128 niveles de gris de cualquiera de las dos imágenes que pueden ser visualizadas en pantalla, presentando en la sección "E" las gráficas correspondientes.

SAID utiliza esta herramienta para que el usuario pueda identificar los tonos de mayor frecuencia y con esto se facilite el algoritmo de umbral del que se hablará posteriormente.

4.9 PERSPECTIVAS

El sistema SAID fue diseñado y desarrollado como parte inicial de un proyecto ambicioso a mediano plazo de procesamiento y administración de imágenes dentro del Instituto de Física de la U.N.A.M., capaz de coordinar y manipular operaciones sobre imágenes obtenidas en este centro de investigación.

La parte inicial solamente procesará imágenes médicas digitalizadas con características ya establecidas y como parte primordial del trabajo de tesis. El enfoque de SAID es crecer sistemáticamete de acuerdo a las nuevas necesidades de software y hardware, a la creación de tendencias de procesamiento de imágenes.

En la parte final del proyecto será relativamente sencillo el adicionar nuevos módulos de código que contengan algoritmos de procesamiento o algoritmos de pruebas sin alterar la estructura original del sistema inicial, es decir, bastará que el usuario elabore por su lado el programa e indique al sistema su existencia, el nuevo módulo será un proceso hijo y deberá de tener como variables generales dos matrices de tipo caracter semejantes a las contenidas en el sistema general.

CAPITULO CINCO

ANALISIS Y DESARROLLO DE ALGORITMOS

5.1 ALGORITMOS DE ELIMINACION DE RUIDO

En términos generales la digitalización de una imagen nunca es pura, es decir, la imagen obtenida no es exactamente la original o en otro caso, se agrega ruido que se encuentra fuera de las fronteras de la sección a analizar. Este ruido puede ser ocasionado por fallas en el equipo digitalizador, por digitalizar una mala fotografía, por deficiencia en la iluminación, etc. Por lo que es necesario que el sistema SAID realice un procesamiento previo al de obtención de parámetros, que permita la eliminación del ruido, así como la de selección de las áreas de interés.

El sistema presenta la facilidad de eliminar ruido contenido en las imágenes, reduciendolo al mínimo y resaltando venas, arterias y patologías.

SAID maneja dos tipos de algoritmos para la eliminación del ruido, y selección de áreas de interés.

Para poder desarrollar los algoritmos más eficientes fué necesario diseñar y programar distintas metodologías para disminuir el ruido, después de diversas pruebas que se realizaron sobre las imágenes disponibles se obtuvieron los siguientes algoritmos:

5.1.1 ALGORITMO DE SELECCION DE VENAS Y ARTERIAS

Analizando las imágenes disponibles con los efectos de la fluoresceina sobre las venas y arterias, así como de tumores y derrames, observamos lo siguiente:

- La imagen presenta gran diversidad de tonalidades abarcando todo el rango de VGA.

- Las venas, arterias y patologías presentadas en la imagen debido a la fluoresceina comprenden un rango de grises intensos (55-127).

- Y el resto del ojo presenta un rango de grises entre 0 y 120.

Como podemos observar de las afirmaciones anteriores la separación del ruido de los vasos sanguíneos y de las zonas de interés en la imagen es complicado debido a que en un rango de colores podemos encontrar fondo de ojo mezclado con zonas de estudio. La figura 5.1.1.1 muestra una imagen típica de fondo de ojo.

En base al histograma típico de fondo de ojo presentado en la figura 5.1.1.2. Si cortamos una banda de frecuencias dejando pasar el pico de mayor intensidad se dibujarían los vasos, sin embargo los grises de los vasos están mezclados con tonalidades del fondo de la imagen. Para nuestro algoritmo el concepto anterior fue retomado pero no en la totalidad de la imagen si no en segmentaciones constantes de tamaño variable. El número de segmentos de la imagen es múltiplo de una potencia de dos, proporcionado por el usuario. Es decir, se obtiene un histograma de intensidad de cada segmentación, partiendo de que podemos tener intensidades de 0 a 127, se calcula el promedio de intensidades así como el máximo valor dentro de la sección. Partiendo del valor

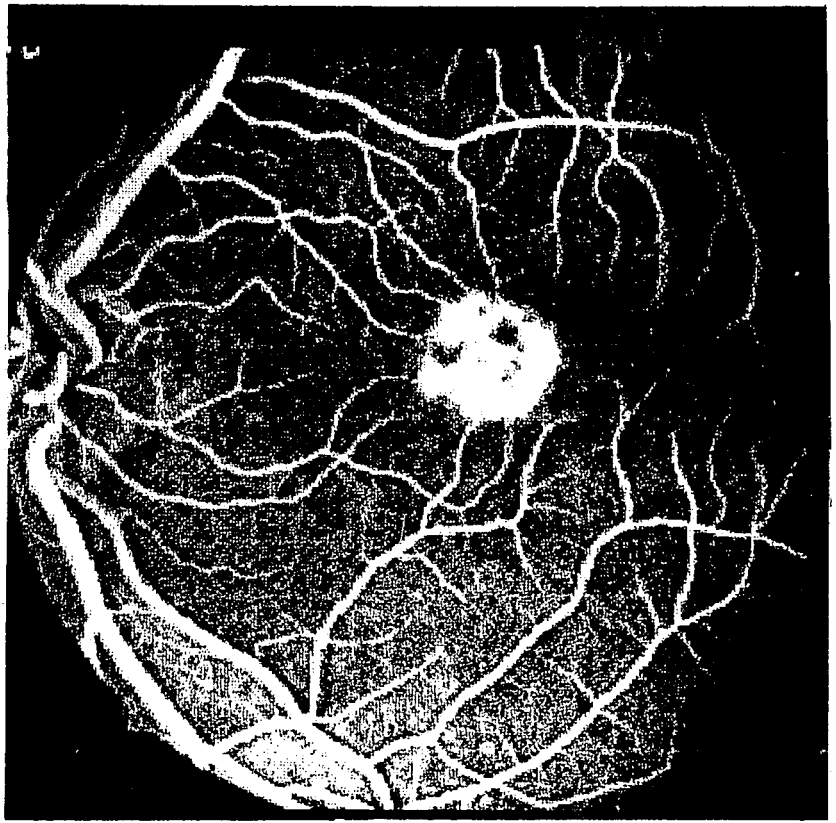


FIGURA 5.1.1.1 FONDO DE OJO CON FLUORESCEINA

anterior (pico de las mayores intensidades) se lleva a cabo un "barrido" , de tal forma, que se encuentra el punto en donde se forma un valle en la curva, ese punto es determinante en el proceso puesto que representa el valor en donde se cortará el histograma.

Pero es necesario, que el usuario del sistema proporcione como entrada un valor que indique aproximadamente la intensidad promedio máxima del ruido, generalmente el valor oscila entre 50 y 80.

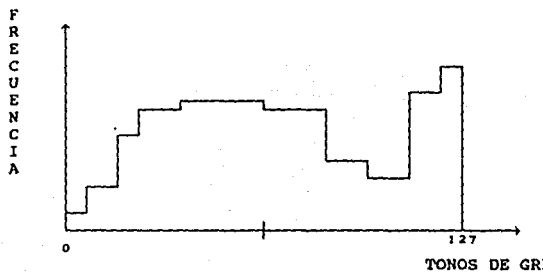


FIGURA 5.1.1.2 HISTOGRAMA TIPICO DE UNA IMAGEN DE FONDO DE OJO CON FLUORESCEINA.

Cada uno de los pixels de cada parte segmentada es comparado con los valores obtenidos hasta el momento (Promedio del segmento, punto de corte del histograma, máxima intensidad, punto proporcionado por el usuario) si los pixels comparados cumplen con la condición de tener un valor de intensidad mayor a los datos anteriores, se coloca en su posición, en caso contrario es desplegado un pixel negro (no forma parte del área de interés).

la figura 5.1.1.3 presenta un diagrama de bloques del algoritmo:

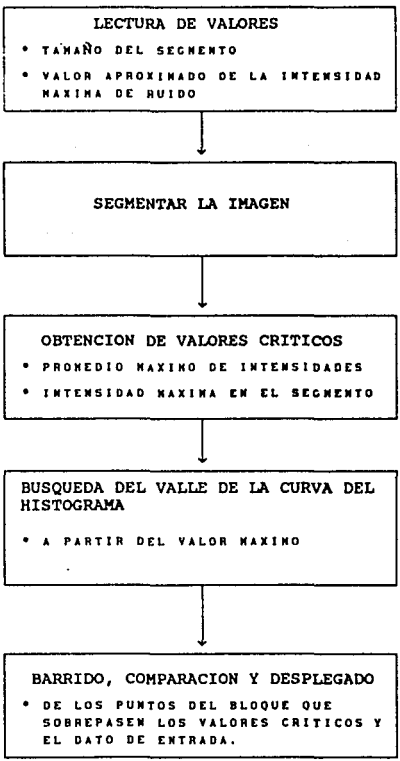


FIGURA 5.1.1.3 DIAGRAMA DE BLOQUES DEL ALGORITMO DE SELECCION DE VASOS SANGUINEOS.

5.1.2 ALGORITMO DE ELIMINACION DE GRISES

Tomando nuevamente el histograma de intensidades de una imagen de fondo de ojo (ver figuara 5.1.1.2) es muy fácil implementar una rutina que coloque los pixels con intensidades mayores a un valor determinado por el usuario, que es la función del algoritmo aqui implementado, es necesario recordar los rangos en los que se encuentra las zonas de interés y las zonas de fondo de imagen para deducir que el algoritmo por si sólo no es muy poderoso; por lo tanto, su razon de ser se debe a que es un algoritmo complementarios del algoritmo de SELECCION DE VENAS, que realiza un barrido total a la imagen para eliminar residuos de ruido dejados por el primer proceso.

Al contar con imágenes terminadas se podrán aplicar sobre ellas los algoritmos especializados, en la busqueda y cuantificación de parámetros que sean útiles.

La figura 5.1.2.1 muestra la imagen de la figura 5.1.1.1 pero procesada con los algoritmos de eliminación de ruido.

5.2 ALGORITMO DE CALCULO DE AREAS

Un sistema de procesamiento de imágenes debe de contar entre su opciones el poder limitar y cuantificar de forma sencilla zonas o áreas especificas.

En Angiologia Retininana es importante medir las áreas de venas, tumores, derrames y patologías encontradas en el fondo ocular, para determinar existencia de enfermedad, avance de patologias y de resultados de tratamientos aplicados al enfermo.

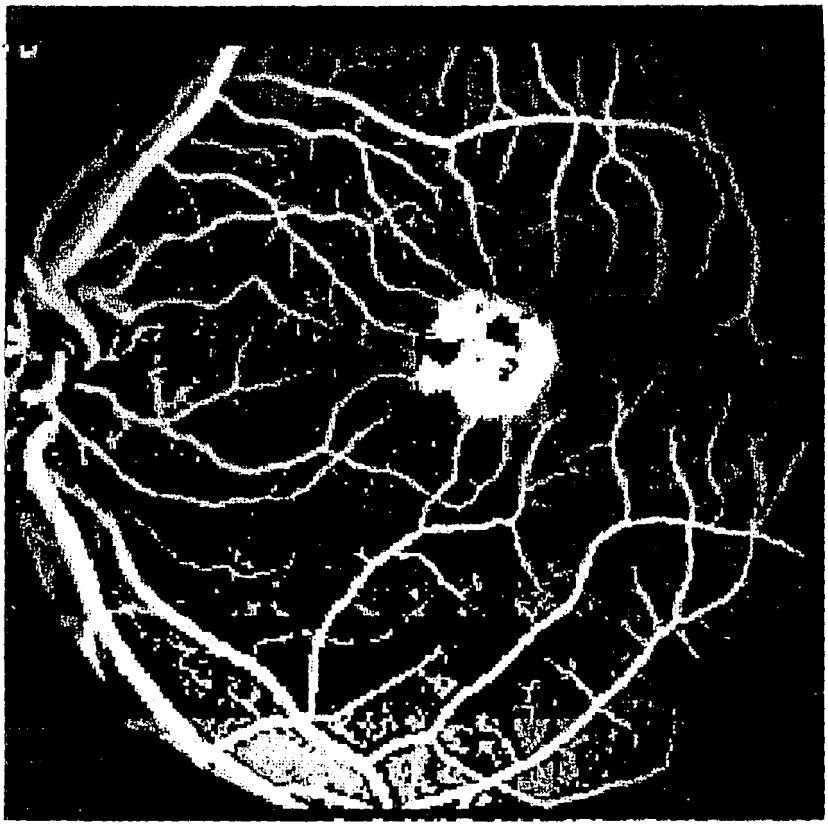


FIGURA 5.1.2.1 IMAGEN DE FONDO DE OJO PROCESADA CON ALGORITMOS DE FILTRADO

Nuestro sistema cuenta con un bloque encargado del cálculo de áreas de zonas escogidas por el usuario. Dada la irregularidad en la forma de los componentes -venas, derrames, patologías, etc- de una imagen, se cuenta con cuatro distintas alternativas; que dependen de la geometría que presente el objeto.

En primer lugar se tiene la opción del cálculo de áreas en forma aproximadamente circular; el algoritmo es sencillo porque requiere únicamente como datos de entrada dos puntos: el centro y un punto del perímetro de la zona. Con ayuda de la ecuación general de la circunferencia se determinan los puntos que pertenecen a la zona, al mismo tiempo los que formen parte del cuerpo son acumulados en la totalidad del área.

Una segunda alternativa es aplicada cuando la imagen presenta forma rectangular. El usaurio proporciona cuatro puntos que formarán parte de un polígono irregular de cuatro lados que delimitará el área a obtener.

Qué hacer en caso de que el cuerpo presente una forma totalmente irregular ?.

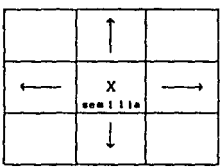
El sistema cuenta con un algoritmo poderoso capaz de calcular el área, de la siguiente forma:

-Limitar "n" fronteras trazando líneas de corte con dos puntos por línea, proporcionados del exterior con ayuda del mouse o de las flechas de dirección.

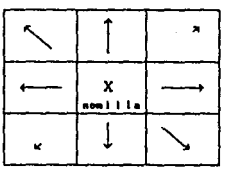
-Aplicación de un algoritmo de llenado que a partir de un punto de la región se expanda en todas las direcciones incluyendo diagonales, acumulando los pixels pintados para la obtención del área.

Es importante señalar, que para dicho algoritmo de llenado, no se utilizó la función propia del lenguaje, debido a que su secuencia de llenado no era capaz de llenar líneas diagonales de un solo pixel y necesareamente la semilla de llenado debería de ser colocada en una parte central. Por lo que fué necesario la realización de una rutina propia de llenado, en la que se cubrirá cualquier región sin importar el tamaño y tipo de la misma.

La figura 5.2.1 (a) muestra las direcciones que sigue la rutina de llenado (floodfill) propia del lenguaje "C" y la fig 5.2.1 (b) la implementada para el sistema.



(a)



(b)

FIGURA 5.2.1 (a). Direcciones de llenado de la función "floodfill" del lenguaje "C". (b). Direcciones de llenado de la función "llenado" diseñada para el sistema.

Finalmente el sistema dispone de un algoritmo sencillo de cálculo de área global, que consiste en barrer totalmente la imagen, acumulando los pixels que tengan el color de llenado obteniendose así el área.

La unidad en la que se desplegarán los resultados será para todos los casos PIXELS. Pero si existe posteriormente la necesidad de representar estas medidas en unidades absolutas será indispensable contar con información que indique la escala en la imagen para realizar conversiones y/o escalamientos.

La figura 5.2.2 muestra el diagrama de bloques de los algoritmos anteriores.

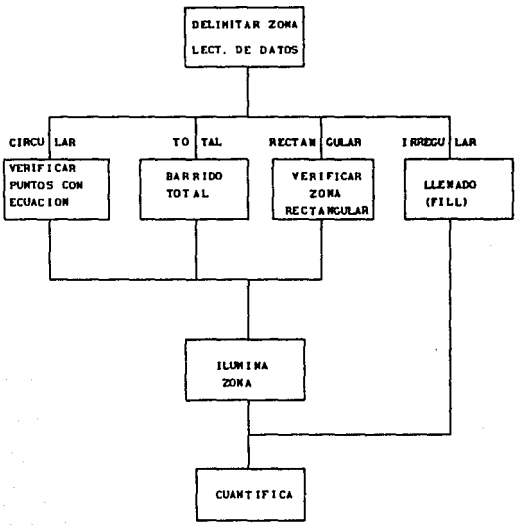


FIGURA 5.2.2 DIAGRAMA DE BLOQUES ALGORITMOS DE CALCULO DE AREA.

El siguiente pseudocódigo muestra la estructura del algoritmo de llenado utilizado para el cálculo de áreas irregulares.

PSEUDOCODIGO DEL ALGORITMO DE LLENADO

```
INICIO
   Limitar área irregular
   colocarse en punto dentro del área de interés (x,y)
   pintar con color_llenado
   cont=0
   nuevo_punto[cont]=punto_interés
   REPITE
   * derecha *
   punto_derecha= punto a la derecha del punto_interés
     IF (punto_derecha<>borde)and(punto_derecha<>color_llenado))
        IF (punto_derecha NO rebasa zona)
           pinta(punto_derecha,color_llenado)
           cont=cont+1
           nuevo_punto[cont]=punto_derecha
        ENDIF
     ENDIF
     * izquierda *
     punto_izquierda= punto a la izquierda del punto_interés
     IF (punto_izq<>borde)and(punto_izq<>color_llenado))
        IF (punto_izq NO rebasa zona)
           pinta(punto_derecha,color_llenado)
           cont=cont+1
           nuevo_punto[cont]=punto_izq
        ENDIF
     ENDIF
     * abajo *
     punto_abajo= punto abajo del punto_interés
     IF (punto_abajo<>borde)and(punto_abajo<>color_llenado))
        IF (punto_abajo NO rebasa zona)
           pinta(punto_derecha,color_llenado)
           cont=cont+1
           nuevo_punto[cont]=punto_abajo
        ENDIF
     ENDIF
```

```
* arriba *                                        CONTINUACION
punto_arriba= punto arriba del punto_interés
IF (punto_arriba<>borde)and(punto_arriba<>color_llenado))
   IF (punto_arriba NO rebasa zona)
      pinta(punto_arriba,color_llenado)
      cont=cont+1
      nuevo_punto[cont]=punto_arriba
   ENDIF
ENDIF
* diagonal derecha arriba *
punto_dere_arriba= punto diagonal a la derecha arriba del
                   punto_interés
IF (punto_dere_arriba<>borde)and
   (punto_dere_arriba<>color_llenado))
   IF (punto_dere_arriba NO rebasa zona)
      pinta(punto_dere_arriba,color_llenado)
      cont=cont+1
      nuevo_punto[cont]=punto_dere_arriba
   ENDIF
ENDIF
* diagonal izquierda arriba *
IF (punto_izq_arriba<>borde)and
   (punto_izq_arriba<>color_llenado))
   IF (punto_izq_arriba NO rebasa zona)
      pinta(punto_izq_arriba,color_llenado)
      cont=cont+1
      nuevo_punto[cont]=punto_izq_arriba
   ENDIF
ENDIF
* diagonal derecha abajo *
IF (punto_dere_abajo<>borde)and
   (punto_dere_abajo<>color_llenado))
   IF (punto_dere_abajo NO rebasa zona)
      pinta(punto_dere_abajo,color_llenado)
      cont=cont+1
      nuevo_punto[cont]=punto_dere_abajo
   ENDIF
```

```
      ENDIF                                          CONTINUACION
      * diagonal izquierda abajo *
      IF (punto_izq_abajo<>borde)and
         (punto_izq_abajo<>color_llenado))
         IF (punto_izq_abajo NO rebasa zona)
            pinta(punto_izq_abajo,color_llenado)
            cont=cont+1
            nuevo_punto[cont]=punto_izq_abajo
         ENDIF
      ENDIF
      punto_interés=nuevo_punto[cont]
      cont=cont-1
   HASTA cont<=0
FIN (Procedimiento)
```

NOTA: El algoritmo es óptimo para zonas pequeñas, pero se degrada el tiempo de procesamiento conforme las áreas se incrementan en tamaño.

5.3 ALGORITMO DE LONGITUD DE VASOS SANGUINEOS

El sistema cuenta con una herramienta capaz de obtener longitudes de todo conducto sanguíneo, parámetro importante para el diagnóstico y determinación de enfermedades.

En principio el algoritmo utiliza la primera parte del algoritmo de cálculo de áreas irregulares, que consiste en la limitación de la zona con líneas y realizar un llenado (fill) de la zona. Posteriormente pasará a realizar la parte fundamental que consiste en trazar una línea por la parte central del conducto marcado. Método llevado a cabo a través de cuatro barridos encargados de identificar la ruta de trayectoria.

A continuación se presenta la parte complementaria del algoritmo.

El primer barrido horizontal calcula y pinta de color (1) las partes centrales de las líneas de barrido que comprendan conductos. Formandose así un eje vertical sobre los vasos.

El primer barrido vertical calcula y pinta de color (2) las partes centrales de las líneas de barrido comprendidas en los conductos, además pinta de un color (3) el punto de coincidencia de color (1) y (2). Tomando forma un eje horizontal sobre los vasos.

Los segundos barridos, tienen la función de marcar la orientación de la línea central del conducto sanguíneo.

El segundo barrido horizontal busca colores (1) y/o (3) tomándolos como pivotes, se obtiene la totalidad de pixels que contenga el conducto sobre el eje que pasa por la coordenada X del pivote; realizandose la misma operación sobre el eje Y. Por último se comparan las dos sumas de pixels marcandose el pivote con color (4) si la suma horizontal fué menor que la vertical, en caso contrario no se afecta el color del punto de referencia.

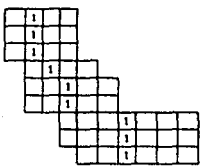
En forma análoga funciona el segundo barrido vertical, aunque busca colores (2) y (3) en su proceso de barrido.

A continuación se presenta el pseudocódigo del algoritmo.

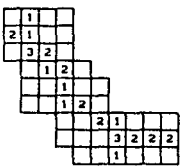
```
* -----PRIMER BARRIDO HORIZONTAL -----*
longitud=0
obtención de las coordenadas mínimas y máximas
REPITE
 REPITE
   MIENTRAS (pixel(x,y) <> color_llenado) y ( x < max_x)
     x=x+1
   extremo_izq_vaso=x
   MIENTRAS (pixel(x,y) = color_llenado) y (x < max_x)
     x=x+1
   extremo_der_vaso=x
   centro=(extremo_der_vaso + extremo_izq_vaso )/2
   PINTA centro con color (1)
 HASTA (x > max_x)
 y=y+1
HASTA (y > max_y)

* -----PRIMER BARRIDO VERTICAL -----*
obtención de las coordenadas mínimas y máximas
REPITE
 REPITE
   MIENTRAS (pixel(x,y) <> color_llenado) y ( y < max_y)
     y=y+1
   extremo_sup_vaso=y
   MIENTRAS (pixel(x,y) = color_llenado) y (y < max_y)
     y=y+1
   extremo_inf_vaso=y
   centro=(extremo_inf_vaso + extremo_sup_vaso )/2
   SI (pixel(centro) = (1))
        PINTA centro con color (3)
   EN CASO CONTRARIO
        PINTA centro con color (2)
 HASTA (y > max_y)
 x=x+1
```

```
HASTA (x > max_x)

* -----SEGUNDO BARRIDO HORIZONTAL -----*

obtención de las coordenadas mínimas y máximas
REPITE
 REPITE
   MIENTRAS (pixel(x,y) <> (1) y (3)) y ( x < max_x)
     x=x+1
   pivote_x=x
   obtención de la totalidad de pixels sobre x
   obtención de la totalidad de pixels sobre y
   SI (total_pixels_x < total_pixels_y)
      PINTA pivote con color (4)
      INC(longitud)
 HASTA (x > max_x)
 y=y+1
HASTA (y > max_y)
* -----SEGUNDO BARRIDO VERTICAL -----*
  obtención de las coordenadas mínimas y máximas
REPITE
 REPITE
   MIENTRAS (pixel(x,y) <> (2) y (3)) y ( y < max_y)
     y=y+1
   pivote_y=y
   obtención de la totalidad de pixels sobre x
   obtención de la totalidad de pixels sobre y
   SI (total_pixels_y < total_pixels_x)
      PINTA pivote con color (4)
      INC(longitud)
 HASTA (y > max_y)
 x=x+1
HASTA (x > max_x)
```

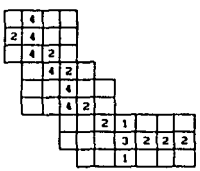
La figura 5.3 muestra esquemáticamente el proceso del algoritmo de longitud.



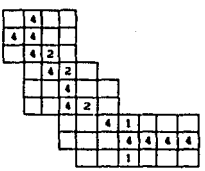
(a) 1^{er} barrido horizontal



(b) 1^{er} barrido vertical



(c) 2^{do} barrido horizontal



(d) 2^{do} barrido vertical

LONGITUD 12 PIXELES

FIGURA: 5.3 Representación del proceso de algoritmo de distancias.

El apéndice A muestra el código fuente de los algoritmos desarrollados.

CONCLUSIONES

El presente trabajo involucró dos áreas de trabajo diferentes, por un lado conceptos biomédicos como morfologías, patologías y tratamientos referentes al campo de estudio de los vasos en el ojo humano conocida como Angiología Retiniana. Por el otro, los de computación, enfocandonos a aquellos que permitan la elaboración de algoritmos -sin importar si es hardware y/o software- capaces de ayudar a resolver problemas que se presentan en el área de Angiología. Es la razón por lo que hemos decidio obtener y describir conclusiones de interés para ambas.

Para dar mayor verasidad y utilidad al Sistema de Procesamiento Gráfico en Angiología Retiniana (SPGAR), se realizó una demostración física, contando con la presencia de personal médico de la Asociación Para Evitar la Ceguera en México, permitiendonos escuchar comentarios que dieron pauta para la elaboración de las conclusiones; además de sugerencias para futuras mejoras y acciones complementarias al sistema.

En primer lugar, se mencionó que SPGAR servirá primordialmente en el área de la investigación científica, como una herramienta capaz de poder obtener y fijar -de manera más rápida , eficiente y económica - parámetros geométricos cuantitativos de morfologías del fondo ocular, que en la actualidad resultan difíciles de obtener mediante el uso de técnicas rudimentarias y empíricas

presentando mayor índice de error o mediante dispositivos sofisticados no comercializados, encontrandose estos únicamente en centros de investigación avanzada.

SPGAR es un sistema útil para los investigadores en el área de Angiología Retiniana por:

1.-Permite desplegar imágenes en una computadora con una resolución tal que es posible identificar perfectamente intensidades subsecuentes que el médico oftalmólogo no puede percibir a simple vista, permitiendo con esto una mejor identificación de las formas y resaltando zonas de interés.

2.-Hace posible que el especialista logre obtener superficie y longitud de zonas en estudio, importante en el análisis e identificación de morfologías y topologías de fondo ocular, con una precisión excelente, a diferencia del obtenido mediante los métodos empíricos convencionales. Cabe hacer mención que el sistema será utilizado para cuantificar áreas específicas de imágenes pertenecientes a pacientes sin anomalías, para posteriormente poder obtener patrones de comparación, útiles en la identificación de patologías y deformaciones en otras imágenes.

3.-Presenta grandes perspectivas como una nueva herramienta que permita facilitar la búsqueda y análisis de patrones geométricos representativos en la estructura del ojo humano. Primero para contar con un conocimento mayor de la misma y segundo para lograr un mejor diagnóstico y tratamiento de patologías.

De acuerdo al punto de vista de un sistema computacional concluimos lo siguiente:

1.-Los algoritmos aqui presentados, fueron desarrollados tomando en cuenta los métodos y disciplinas de la ingeniería de software aprendidos a lo largo de la preparación profesional.

2.-Tomando en cuenta los recursos de hardware con que el sistema cuenta, los algoritmos resultan rápidos, eficientes y seguros. Presentando un indice de error minimo y en algunos casos totalmente nulo.

3.-El sistema presenta la característica de modularidad permitiendo con esto facilidad de crecimiento, así es posible agregar nuevos bloques de código útiles para necesidades propias o de otras disciplinas. Una prueba de ello es que diferentes rutinas del sistema han sido utilizadas en el Instituto de Física de la UNAM para otro tipo de aplicaciones, como el Sistema Administrador de Imagenes Digitales "SAID" utilizado para presentaciones de distintos tipos de imágenes agregándosele pequeños modulos para procesar las mismas. Es decir el sistema ha hecho uso facilmente de otras rutinas que no fueron desarrolladas para él y viceversa, rutinas de SPGAR han sido utilizadas por otros sistemas desarrollados en el mismo Instituto.

4.-El sistema facilita su uso al contar con un driver para MOUSE y HELP integrado, además cuenta con filtros contra posibles malos usos, haciendolo accesible a usuarios sin grandes conocimientos en computación.

5.-Por otra parte el sistema es único en el país, no existe software que realice un análisis semejante al del trabajo aquí desarrollado. Con algunas mejoras y módulos complementarios SPGAR podría ser comercializado para ser utilizado en otros centros de investigación relacionados a la Angiología Retiniana.

COMENTARIOS

Con la experiencia obtenida en la elaboración del sistema, por su característica de ser interdisciplinario y haber trabajado con gente de áreas diferentes a la nuestra, nos permite señalar que la Ingeniería en Computación tiene un amplio campo de desarrollo, por lo que es satisfactorio haber realizado un sistema de tal magnitud.

Los autores.

NOTA: Las personas interesadas en el funcionamiento de SPCAR, así como de las distintas aplicaciones del mismo, pedir informes con:

Sergio Avendaño Bonilla

Aldo Granados Pérez

Luis Beltrán Del Río Caballero

Instituto de Física U.N.A.M

Departamento de Materia Condensada

Circuito de la Investigación Científica

Ciudad Universitaria México D. F.

APENDICE A

CODIGO DEL ALGORITMO PARA ELIMINAR GRISES

```
/*-------------------------------------------------------------*/
/* FUNCION PARA ELIMINAR GRISES                                */
/*  initx,inity,finx,finy  coordenadas  de  lectura; op_umbral
valor superior de los grises a  eliminar; matriz1  matriz fuente
matriz2 matriz destino                                         */

 umbral(int initx,int inity,int finx,int finy,int op_umbral,
       char far *matriz1[],char far *matriz2[])
                              (
  int renglon,escalado,incre;
  int coordx,coordy,verti,horiz;
  char ancho,escala[8];
  unsigned tono[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
  unsigned ton_aux;

  for (verti=inity; verti<finy; verti++)
    for  (horiz=initx; horiz<finx; horiz++){
                        ton_aux=*(col u m[verti]+horiz)/8;
         tono[(int)(ton_aux)]++;    )
    incre=(pos_image=='A')?262:2;
  for (renglon=0; renglon<MAXRENG; renglon++){
    limpia_pagina(incre,renglon+2,255+incre,renglon+2);
    for (verti=0; verti<MAXRENG; verti++){
       escalado=*(matriz1[renglon]+verti);
       if (escalado>op_umbral){
         putpixel(verti+incre,renglon+2,escalado/8);
                * (matriz 2  [renglon]+vert i   ) = escalado;}
       else *(matriz2[renglon]+verti)=0;}}}
 /* Fin del Procedimiento */
```

CODIGO DEL ALGORITMO PARA ELIMINAR GRISES EN FORMA FLEXIBLE

```
/*-------------------------------------------------------------*/
/*   FUNCION PARA FILTRADO DE GRISES EN FORMA FLEXIBLE RESALTAN */
/*   DO VASOS SANGUINEOS, TUMORES, DERRAMES ETC.               */
/*   op_umbral: Dato proporcionado por el usuario. Valor       */
/*   mínimo promedio de intensidades de areas de interés       */
/*-------------------------------------------------------------*/
umbral_flexible (op_umbral)
   int op_umbral;
  {
    int renglon,columna,lado=16,nure;
    long int suma=0,suma_parc[16][16];
    char *flecha;
    float prom,umb_pes,escalado,umb_parc;
    nure=MAXRENG/lado;
    for (renglon=0; renglon<nure; renglon++)
        for (columna=0; columna<nure; columna++)
              suma_parc[columna][renglon]=0;
        for (renglon=0; renglon<nure; renglon++)
           {
             for (columna=0; columna<nure; columna++)
                {
                   for (x=0;x<lado;x++)
                      {
                      for (y=0;y<lado;y++)
                         {
                          flecha=colum[renglon*lado+y]+
                                 (columna*lado+x);
                          suma_parc[columna][renglon]+=*flecha;
                          suma+=*flecha;}}}}
        prom=suma/65536;
        umb_pes=op_umbral/prom;
        for (renglon=0; renglon<nure; renglon++)
          {
            for (columna=0; columna<nure; columna++)
              {
                umb_parc=umb_pes*suma_parc[columna][renglon]/256;
                for (x=0;x<lado;x++)
                    {
                       for (y=0;y<lado;y++)
                          {
                            flecha=colum[renglon*lado+y]+
                                   (columna*lado+x);
                            escalado= *flecha/8;
                            if (*flecha > umb_parc)
                              putpixel(columna*lado+x+260,
                                       renglon*lado+y+2,escalado);
                            else putpixel(columna*lado+x+260,
                                          renglon*lado+y+2,0);
                          }}}}
  }   /* FIN DE FUNCION UMBRAL2 */
```

CODIGO DEL ALGORITMO PARA CALCULO DE AREA DE DIVERSAS ZONAS

(vasos, tumores, derrames etc.)

```
#include<graphics.h>
#include<alloc.h>
#include<math.h>
#include<stdlib.h>
#define MAXRENG 256
#define ESC 27

extern int raton;        /* INDICA SI EXISTE MOUSE  */
extern int dedo[32];     /* VARIBLE PARA DIBUJAR UN CURSOR DE*/
                         /* PARA MOUSE                       */
extern int reloj[32];    /* VARIBLE PARA DIBUJAR UN CURSOR DE*/
                         /* PARA MOUSE                       */
extern float area_tot;   /* ALMACENA EL AREA TOTAL  */
double rad2,eq=0.0;
float area=0.0;
struct coordenadas {
   int tamano[4];
   int x,y; } sq;
typedef char STRING18[12];
STRING18 confirma[2]={"ACEPTAS", "RECHAZAS"};
/*-------------------------------------------------------------*/
/* FUNCION REGLA: INDICA EL AVANCE DEL PROCESO EN EJECUCIÓN    */
/*-------------------------------------------------------------*/
regla(lineas,y) /* y: coordenada para pintar la linea */
int lineas,*y;
{  int veces;
   setcolor(15);
   for (veces=1;veces<=lineas; veces++){
   line(519,*y,532,*y);
   *y=*y+1;}

/*-------------------------------------------------------------*/
/* FUNCION CALAREACIRC: CALCULA EL AREA DE UNA ZONA CIRCULAR    */
/*  cx: centro en X; cy: centro en Y; radio: radio del área    */
/*-------------------------------------------------------------*/
calareacirc(int cx, int cy, double radio)
{ int x,y;
  char strarea[20];
  rad2=radio*radio;
  for (y=cy-radio; y<=cy+radio; y++)
    for (x=cx-radio; x<=cx+radio; x++){
        eq=(x-cx)*(x-cx)+(y-cy)*(y-cy);
        if (eq<rad2)
           if (getpixel(x,y)>0){
               area+=1.0;
               putpixel(x,y,10);}}
}
```

CONTINUA

```
/*-------------------------------------------------------------*/
/* FUNCION MARCACIRCULO:  ENCARGADA DE HACER LAS OPERACIONES   */
/*  PARA EL CALCULO DE AREAS DE ZONAS DE DIFERENTES FORMAS     */
/* ptos_x y ptos_y : contienen las coordenadas en X y Y de las */
/* distintas zonas                                             */
/*-------------------------------------------------------------*/
marcacirculo(int tipo, int ptos_x[], int ptos_y[], int *band)
{
  unsigned long size1;
  int maxix=0,minix=256,maxiy=0,miniy=256,repite=0,x,y,
      yr=0,resp=0;
  char *cuadro1, strarea[20], valor;
  double radio;
  *band=0;
  area=0.0;
  sq.tamano[0]=240; sq.tamano[1]=310;
  sq.tamano[2]=350; sq.tamano[3]=345;
  sq.x= 245;  sq.y=315;
  setcolor(15);
  if (raton) mshow(2);  /* función asociada al manejo de mouse */
  switch(tipo){ /* Area circular */
   case 0:limpia_pagina(2,451,516,476);/* borra sección
                                          del screen */
         radio=ceil(sqrt((ptos_x[1]-ptos_x[0])*(ptos_x[1]-
         ptos_x[0]+(ptos_y[1]-ptos_y[0])*(ptos_y[1]-ptos_y[0])));
         size1=imagesize(ptos_x[0]-(radio+1),ptos_y[0]-(radio+1),
         ptos_x[0]+(radio+1),ptos_y[0]+(radio+1));
         cuadro1=(char *)malloc(size1);
         if (cuadro1!=NULL){
           getimage(ptos_x[0]-(radio+1),ptos_y[0]-(radio+1),
           ptos_x[0]+(radio+1),ptos_y[0]+(radio+1),cuadro1);
           setviewport(0,0,258,258,0);
           circle(ptos_x[0],ptos_y[0],(radio+1));
     /* función para obtener si se acepta el área o no */
           repite=ponvideo2(sq,confirma,2,&resp);
           if (resp==1)
             *band=1;
           else{
             if (repite!=ESC){
               if (raton) {
                     mput(258, 263); /* funciones asociadas al
                                        manejo de mouse */
                mshape(5,0,&reloj);
                mshow(1);}
                setviewport(0,0,639,479,1);
                calareacirc(ptos_x[0],ptos_y[0],radio);
         }}}
         else ventana_errores("  Memoria Insuficiente ");
         if (raton) mshape(5,0,&dedo); /* función asociada al
                                          manejo de mouse */
         break;
```

CONTINUA

```
case 1: repite=ponvideo2(sq,confirma,2,&resp); /* area
        if (resp==1)
           *band=1;
        else{
        if (repite!=ESC){
           x=ptos_x[0];
           y=ptos_x[2];
           while (y<=ptos_x[3]){
              while (x<=ptos_x[1]){
                 if (getpixel(x,y)==10)
                    area+=1.0;
                 x++;}
              x=ptos_x[0]; y++;}}}
           break;
case 2: size1=imagesize(ptos_x[0],ptos_y[0],ptos_x[1],258);
        cuadro1=(char *)malloc(size1);
        if (cuadro1==NULL)
           ventana_errores("  Memoria Insuficiente ");
        else{
           getimage(ptos_x[0],ptos_y[0],ptos_x[1],258,cuadro1);
           if (raton) {
              mput(258, 263);  /* funciones asociadas al
                                  manejo de mouse */
              mshape(5,0,&reloj);
              mshow(1);}
           x=ptos_x[0];
           y=ptos_y[0];
           yr=2;
          while (y<258){
            regla(1,&yr);
            while (x<ptos_x[1]){
            if (getpixel(x,y)>0){
               area+=1.0;
               putpixel(x,y,10);}
               x++;}
            x=ptos_x[0]; y++;}}
          if (raton) mshape(5,0,&dedo); /* función asociada
                                           al manejo de mouse */
         break;
case 3: for (repite=0; repite<8; repite+=2){
        if (ptos_x[repite]>=maxix)
           maxix=ptos_x[repite];
        if (ptos_x[repite]<=minix)
        minix=ptos_x[repite];
        if (ptos_x[repite+1]>=maxiy)
           maxiy=ptos_x[repite+1];
        if (ptos_x[repite+1]<=miniy)
           miniy=ptos_x[repite+1];}
        size1=imagesize(minix,miniy,maxix,maxiy);
```

CONTINUA

```
          cuadro1=(char *)malloc(size1);
          if (cuadro1!=NULL){
             getimage(minix,miniy,maxix,maxiy,cuadro1);
             setcolor(5);
             setfillstyle(EMPTY_FILL,5);
             fillpoly(4,ptos_x);
             repite=ponvideo2(sq,confirma,2,&resp);
             if (resp==1)
                *band=1;
             else{
               if (repite!=ESC){
                  if (raton) {
                    mput(258, 263);  /* funciones asociadas al
                                        manejo de mouse */
                    mshape(5,0,&reloj);
                    mshow(1);}
                  y=miniy+1; x=minix-1;
                  while (y<maxiy-1){
                    while ((getpixel(x,y)!=5)&&(x<maxix+1))
                      x++;
                    while ((getpixel(x,y)==5)&&(x<maxix+1))
                      x++;
                    while ((getpixel(x,y)!=5)&&(x<maxix+1)){
                      if (getpixel(x,y)>0){
                         area+=1.0;
                         putpixel(x,y,10);}
                      else
                         putpixel(x,y,8);
                      x++;}
                    y++;x=minix-1;}}}
               else ventana_errores("  Memoria Insuficiente ");
             /* función asociada al manejo de mouse */
               if (raton) mshape(5,0,&dedo);
          break;}
    if ((!*band)&&(repite!=ESC)){
      if (area!=0.0){
        gcvt(area,5,strarea);
        limpia_pagina(2,259,517,418);/* borra parte del screen */
        outtextxy(20,300,"El Area de la Zona de Interes es: ");
        outtextxy(400,300,strarea);
        area_tot=(100.0 * area)/area_tot;
        gcvt(area_tot,3,strarea);
        outtextxy(20,315,"Del Area Total Representa El ");
        outtextxy(270,315,strarea);
        outtextxy(325,315,"%");
        ventana_errores("Presiona una Tecla");}
      else{
        gcvt(area,5,strarea);
        limpia_pagina(2,259,517,418);/* borra parte del screen */
        outtextxy(20,300,"El Area es: 0 o Puntos fuera de control,.
```

CONTINUA

```
                intente de nuevo");
          ventana_errores("Presiona una Tecla");))
        if (cuadro1!=NULL)
          switch (tipo){
          case 0: putimage(ptos_x[0]-(radio+1),ptos_y[0]-(radio+1),
                  cuadro1,0);
                  free(cuadro1);break;
          case 3: putimage(minix,miniy,cuadro1,0);free(cuadro1);
                  break;
          case 2: putimage(ptos_x[0],ptos_y[0],
                  cuadro1,0);free(cuadro1);break;}
          limpia_pagina(2,259,517,418);  /* borra sección del
                                            screen*/
          limpia_pagina(2,451,516,476);
          limpia_pagina(519,2,531,256);
}         /* Fin de Función */
```

CODIGO DEL ALGORITMO QUE OBTIENE LONGITUD DE VASOS

```
/*-------------------------------------------------------------*
/* FUNCION DIAMETRO : CALCULA LA LONGITUD Y PROPORCIONA UNA    *
/*   GRAFICA DE LOS DIAMETROS DE LOS CONDUCTOS SELECCIONADOS   *
/*-------------------------------------------------------------*
long       diam[150];/*    ARREGLO DE DIAMETROS  */
int        longitud; /*  LONGITUD DE UN CONDUCTO */
extern int minmax[5];/*  DELIMITADORES DE UNA ZONA SELECCIONADA  /
diametrovenas()
{
 int x=0,y=0,inst_number=0;
 double maxi;
 long cont;
 char ch,numero[10],strlong[20];
 setcolor(15);
 longitud=0;
 for (x=0; x<50; x++)
     diam[x]=0;
 barrido(1);          /* llamadas a las funciones encargadas */
 barridov(2);         /* de los cuatro barridos              */
 barridohor(minmax);
 barridover(minmax);
 itoa(longitud,strlong,10);
 setcolor(15);
 limpia_pagina(2,259,516,418);/*limpia una sección del screen*/
 outtextxy(20,280,"La longitud de la Zona de Interes es: ");
 outtextxy(400,280,strlong);
 ventana_errores("Presiona una Tecla");
 limpia_pagina(2,259,517,418);/*limpia una sección del screen*/
 limpia_pagina(2,259,516,418);/*limpia una sección del screen*/
 settextstyle(3,HORIZ_DIR,2);
 outtextxy(440,407,"DIAMETRO");                    CONTINUA
```

```
  outtextxy(20,407,"1 2 3  4 5 6 7 8 9 10 1 2 3 4 5 6 7 8
  9 20 1 2 3 4 5");
  settextstyle(3,VERT_DIR,2);
  outtextxy(13,285," FRECUENCIA ");
  settextstyle(0,HORIZ_DIR,1);
  line(20,278,20,400);
  line(20,400,438,400);
  maxi=0.0;
  setcolor(15);
  for (x=0; x<50; x++)
   if (diam[x])
    if (log10(diam[x])>maxi)  /* escala el maximo valor de diam */
      maxi=log10(diam[x]);
   if (!maxi) maxi=0.1;
    for (x=0; x<25; x++){
     if (diam[x]){
       cont=(log10(diam[x])*120.0)/maxi;
       if (cont>0)
         for (y=0; y < cont; y++)
          line(20+x*5+(12*(x)),400-y,20+x*5+(12*(x))+10,400-y);}}
    maxi=pow(10,maxi);
    gcvt(maxi,3,strlong);
    outtextxy(4,270,strlong);
    ventana_errores(" Presiona una tecla: ");
    limpia_pagina(2,259,516,418);
}
/*-------------------------------------------------------------*/
/* FUNCION BARRIDO:  RELIZA EL PRIMER BARRIDO HORIZONTAL DE LA */
/*                   ZONA ESCOGIDA                             */
/*-------------------------------------------------------------*/
barrido(int color){
   int distx,disty,d1=0,d2=0,x,y=minmax[2];

   do{
     x=minmax[0];
     do{
     while((getpixel(x,y)!=LIGHTGREEN)&&(x<=minmax[1]))
   x++;
   if (x<=minmax[1]){
   d1=x;
   while((getpixel(x,y)==LIGHTGREEN)&&(x<minmax[1]))
     x++;
   d2=x;
   distx=ceil((d2-d1)/2);
   putpixel(d1+distx,y,color);}
   }while(x<minmax[1]);
   y++;
   }while (y<=minmax[3]);
  }
```

CONTINUA

```
/*-------------------------------------------------------------*/
/* FUNCION BARRIDOV: REALIZA EL PRIMER BARRIDO VERTICAL        */
/*-------------------------------------------------------------*/
barridov(int color)
{
  int distx,disty,d1=0,d2=0,x=minmax[0],y=0;
   do{
     y=minmax[2];
     do{
     while((getpixel(x,y)!=LIGHTGREEN)&&
      (y<=minmax[3])&&(getpixel(x,y)!=1))
           y++;
     if (y<=minmax[3]){
       d1=y;
       while((getpixel(x,y)!=0)&&(y<minmax[3]))
          y++;
        d2=y;
        disty=ceil((d2-d1)/2);
        if ((getpixel(x,d1+disty))==1)
           putpixel(x,d1+disty,3);
        else
           putpixel(x,d1+disty,color);}
     }while(y<minmax[3]);
     x++;
     }while (x<=minmax[1]);
   }
/*-------------------------------------------------------------*/
/* FUNCION BARRIDOHOR: REALIZA EL SEGUNDO BARRIDO HORIZONTAL*/
/*-------------------------------------------------------------*/
barridohor(int minmax[])
{
  unsigned distx,disty,posx,posy,conx, cony;
  int xh=0,yh=0;
  cony=posy=0;
  posx=conx=0;
  yh=minmax[2];
   do{
     xh=(minmax[0]*1);
     do{
     while((getpixel(xh,yh)!=1)&&(getpixel(xh,yh)!=3)&&
       (xh<=minmax[1]))
     xh++;
     if (xh<=minmax[1]){
        posx=xh;
        posy=yh;
        cony=0;
        while((posy>=0) && (getpixel(posx,posy)!=0)) {
           posy--;
           cony++;}
```

CONTINUA

```
        cony--;
        posy=yh;
        while ((posy<=minmax[3]) && (getpixel(posx,posy)!=0)){
           posy++;
           cony++;}
        cony--;
        conx=0;
        posy=yh;
        while  ((posx>=0) && (getpixel(posx,posy)!=0)){
           posx--;
           conx++;}
        conx--;
        posx=xh;
        while ((posx<=minmax[1]) && (getpixel(posx,posy)!=0)) {
           posx++;
           conx++;}
        conx--;
        if (conx<=cony){
            putpixel(xh,yh,4);
            longitud++;
            diam[ceil(conx)]++;}
     }
    xh++;
     }while(xh<minmax[1]);
   yh++;
   }while (yh<=minmax[3]);
}
/*-------------------------------------------------------*/
/* FUNCION BARRIDOVER: REALIZA EL SEGUNDO BARRIDO VERTICAL */
/*-------------------------------------------------------*/
barridover(int minmax[])
{
  unsigned distx,disty,posx=0,posy=0,conx=0,cony=0;
  int x,y;
  x=minmax[0];
   do{
     y=minmax[2];
     do{
       while(((getpixel(x,y)!=2)&&(getpixel(x,y)!=3))&&
             (y<=minmax[3]))
           y++;
     if (y<=minmax[3]){
        posx=x;
        posy=y;
        cony=0;
        while ((posy>=0) && (getpixel(posx,posy)!=0)){
           posy--;
           cony++;}                                    CONTINUA
```

```
        cony--;
        posy=y;
        while ((posy<=minmax[3]) && (getpixel(posx,posy)!=0)){
           posy++;
           cony++;}
        cony--;
        conx=0;
        posy=y;
        while ((posx>=minmax[0]) && (getpixel(posx,posy)!=0)){
           posx--;
           conx++;}
        conx--;
        posx=x;
        while ((posx<=minmax[1]) && (getpixel(posx,posy)!=0)){
           posx++;
           conx++;}
        conx--;
        if (cony<=conx){
            putpixel(x,y,4);
            longitud++;
        diam[ceil(cony)]++;}
     y+=cony;
     }while(y<minmax[3]);
   x++;
   }while (x<=minmax[1]);
}  /* FIN DEL ALGORITMO DE LONGITUD */
```

CODIGO DEL ALGORITMO QUE REALIZA UN FLOOD DE UNA ZONA CUALQUIERA

```
/*-------------------------------------------------------------*/
/*FUNCIONLLENADO:REALIZAUNFILLDEUNAZONASELECCIONADA*/
/*-------------------------------------------------------------*/
#include<graphics.h>
externintminmax[5];/*DELIMITADORESDELAZONA*/
externinti ni1,ini2;/*INICIODELASIMAGENES*/
externintraton;/*EXISTENCIADERATON*/

intcx[7000],cy[7000],centrox,centroy,maximo=0;
llenado(intx,inty,intcolor)
{
  int contarre=0;

  setcolor(WHITE);
  if(raton){
     centrox=x;centroy=y;}
  else{
     centrox=x+ini1;centroy=y+ini2;}
  cx[contarre]=centrox;cy[contarre]=centroy;
  putpixel(centrox,centroy,LIGHTGREEN);
  minmax[0]=minmax[1]=centrox;
  minmax[2]=minmax[3]=centroy;
  do{
     /*derecha*/
     if((getpixel(centrox+1,centroy)!=color)&&
        (getpixel(centrox+1,centroy)!=LIGHTGREEN)){
        putpixel(centrox+1,centroy,LIGHTGREEN);
        if(centrox+1>minmax[1]) minmax[1]=centrox+1;
        contarre++;
        cx[contarre]=centrox+1;cy[contarre]=centroy;}
     /*izq*/
     if((getpixel(centrox-1,centroy)!=color)&&
        (getpixel(centrox-1,centroy)!=LIGHTGREEN)){
        if(centrox-1<minmax[0])minmax[0]=centrox-1;
        putpixel(centrox-1,centroy,LIGHTGREEN);
        contarre++;
        cx[contarre]=centrox-1;cy[contarre]=centroy;}
     /*abajo*/
     if((getpixel(centrox,centroy+1)!=color)&&
        (getpixel(centrox,centroy+1)!=LIGHTGREEN)){
        putpixel(centrox,centroy+1,LIGHTGREEN);
        if(centroy+1>minmax[3]) minmax[3]=centroy+1;
        contarre++;
        cx[contarre]=centrox;cy[contarre]=centroy+1;}
     /*arriba*/
     if((getpixel(centrox,centroy-1)!=color)&&
         (getpixel(centrox,centroy-1)!=LIGHTGREEN)){
         putpixel(centrox,centroy-1,LIGHTGREEN);
         contarre++;
         if(centroy-1<minmax[2]) minmax[2]=centroy-1;
         cx[contarre]=centrox;cy[contarre]=centroy-1;}
                                                    CONTINUA
```

```
    /*diagonalizqarr*/
    if((getpixel(centrox-1,centroy-1)!=color)&&
       (getpixel(centrox-1,centroy-1)!=LIGHTGREEN)){
       putpixel(centrox-1,centroy-1,LIGHTGREEN);
       if(centroy-1<minmax[2]) minmax[2]=centroy-1;
       if(centrox-1<minmax[0])minmax[0]=centrox-1;
          contarre++;
       cx[contarre]=centrox-1;cy[contarre]=centroy-1;}
    /*diagderarr*/
    if((getpixel(centrox+1,centroy-1)!=color)&&
       (getpixel(centrox+1,centroy-1)!=LIGHTGREEN)){
       putpixel(centrox+1,centroy-1,LIGHTGREEN);
       if(centroy-1<minmax[2]) minmax[2]=centroy-1;
       if(centrox+1>minmax[1]) minmax[1]=centrox+1;
       contarre++;
       cx[contarre]=centrox+1;cy[contarre]=centroy-1;}
    /*diagizqabajo*/
    if((getpixel(centrox-1,centroy+1)!=color)&&
       (getpixel(centrox-1,centroy+1)!=LIGHTGREEN)){
       putpixel(centrox-1,centroy+1,LIGHTGREEN);
       if(centroy+1>minmax[3]) minmax[3]=centroy+1;
       if(centrox-1<minmax[0]) minmax[0]=centrox-1;
       contarre++;
       cx[contarre]=centrox-1;cy[contarre]=centroy+1;}
    /*diagderabajo*/
    if((getpixel(centrox+1,centroy+1)!=color)&&
       (getpixel(centrox+1,centroy+1)!=LIGHTGREEN)){
       putpixel(centrox+1,centroy+1,LIGHTGREEN);
       if(centroy+1>minmax[3]) minmax[3]=centroy+1;
       if(centrox+1>minmax[1]) minmax[1]=centrox+1;
       contarre++;
       cx[contarre]=centrox+1;cy[contarre]=centroy+1;}
       centrox=cx[contarre];centroy=cy[contarre];
       if(contarre>=maximo)
          maximo=contarre;
       contarre--;
}while(contarre>=0);
if(maximo>=7000)
  ventana_errores("ERROR:FUERADEMEMORIA");
}/* Fin del Algoritmo */
```

BIBLIOGRAFIA

1. ATLAS DE FLOURESCEINGRAFIA DEL FONDO OCULAR
   Shinichi Shikano, M.D. y Koichi Shimizu, M.D.
   Traducido por: Dr. F. Palomar-Petit.
   Salvat Editores.

2. DIGITAL IMAGE PROCESSING
   Gregory A. Baxes.
   Editorial: Prentice-Hall.
   Año: 1984.

3. DIGITAL IMAGE PROCESSING
   Kenneth R. Castlman.
   Editorial: Prentice-Hall Signal Processing Series.
   Año: 1979.

4. MEMORIAS DEL SIMPOSIUM
   "PROCESAMIENTO DIGITAL DE IMAGENES EN BIOMEDICINA"
   Editorial: Facultad de Medicina e Instituto de Fisiologia
   Celular U.N.A.M.
   Febrero 1989.

5. DATA COMPRESSION
   Gilbert Held.
   Editorial: John Wiley and Sons.

6. MATRIX STRUCTURED IMAGE PROCESSING
   Edward R. Dougherti & Charles R. Giardina.
   Editorial: Prentice-Hall.
   Año: 1987.

7. GRAFICAS POR COMPUTADORA
Donald Hearn and M. Pauline Baker.
Traducción: Juan Carlos Fagoaga.
Editorial: Prentice-Hall Hispanoamericana S.A.
Enero 1989.

8. PROGRAMMERS'S GUIDE TO THE EGA AND VGA CARDS
Richard F. Ferraro.
Editorial: Addison-Wesley Publishing Company, Inc.
Año 1988.

9. INOVION
Manual de Operación.
Inovion Co. Layton Utah.

10. "C" GUIA PARA USUARIOS EXPERTOS
Herbert Schildt.
Traducción: José A. Moreno R.
Fernando Bienvenido B.
Julio Baron M.
Editorial: Osborne/McGraw-Hill.
Septiembre 1989.

11. GRAPHICS PROGRAMMING IN TURBO C 2.0
Ben Ezzell.
Editorial: Addison-Wesley.
Julio 1989.

12. ADVANCED GRAPHICS IN C: PROGRAMMING AND TECHNIQUES
Nelson Johnson.
Editorial: Osborne/McGraw-Hill.
Año: 1987.

13. MANUALES DE LENGUAJE TURBO C VERSION 2.0
User Guide and Reference Guide.
Por Borland Internatonal Inc.
Año: 1988.

14. SISTEMAS DE COMUNICACION
Ferrel G. Stremler.
Traducción: Ruy Renau Ballester.
Editorial: Fondo Educativo Interamericano.
Año: 1985

15. INSIDE THE IBM PC AND PS/2
Peter Norton.
Editorial: Brady.
Año: 1990.

HEMEROGRAFIA

1. GRAPHICS FORMATS
   Gerald 1. Graef.
   BYTE.
   Septiembre de 1989.
   pp. 305-310.

2. IN DEPTH: SOUND AND IMAGE PROCESSING
   Gene Smarte & Walt Penney.
   BYTE.
   Diciembre 1989.
   pp. 240-258.

3. TURBO PASCAL DRIVES THE MOUSE
   John Figueras.
   BYTE.
   Septiembre de 1985.
   pp. 161-168.

4. VGA VIDEO MODES
   Richard Wilton.
   BYTE.
   Edicion Especial 1988.
   pp. 187-198.

5. IS IT REALLY SUPER ?
Bill Nicholls.
BYTE.
Edicion Especial 1988.
pp. 159-164.

6. DEBUNKING 16-BIT VGA
Bradley Dyck Kliewer.
BYTE.
Junio de 1989.
pp. 195-199.

7. A VGA ON EVERY DESK
Stanford Diehl and Howard Eglowstein.
BYTE.
Marzo de 1990.
pp. 126-138.

8. MASTERING THE PCX FORMAT
Bert Tyler.
BYTE.
Septiembre 1989.
pp. 183-187.

9. A TECHNIQUE FOR HIGH-PERFORMANCE DATA COMPRESSION
Terry A. Welch.
IEEE COMPUTER.
Junio 1984.
pp. 8-19.

10. BIDIRECTIONAL HUFFMAN CODING
A.S.Fraenkel & S.T.Klein.
The Computer Journal.
Abril 1990.
pp. 296-307.

11. PIPELINING DATA COMPRESSION ALGORITHMS
R.L. Bailey & R. Mukkamala.
The Computer Journal.
April 1990.
pp. 308-313.

CATALOGOS

1. CATALOGO DE " EPIX, INC. "
   4MEG Video model 10
   Dirección: 310 Anthony Trail, Northbrook, Il 60062
   Teléfono: (312) 498-4002

2. CATALOGO DE " KEITHLEY METRABYTE "
   Accesorios, Software, Dispositivos de Adquisición de imágenes (Hardware) etc..
   Dirección: 440 Myles Standish Blvd; Tauton, MA 02780
   Teléfono: (580) 880-3000.

3. CATALOGO DE "CORECO INC."
   Aplicaciones y tips en procesamiento de imágenes.
   Dirección: 6969 Trans-Canada Highway Suite 113, Saint-Laurent, Quebec, Canada.
   Teléfono: (514) 333-1301.

4. CATALOGO DE "TRUEVISION INC."
   Targa+. (Tarjeta Digitalizadora)
   Dirección: 7340 Shadeland Station,Indianapolis, IN 46256
   teléfono: (317) 841-0332.

5. CATALOGO DE "PHOTOMETRICS LTD."
   Digital CCD Workstation & Product Summary.
   Dirección: 3440 East Britannia Drive, Tucson
   Arizona 85706.
   Teléfono: (602) 889-9933.

6. CATALOGO DE "HOWTEK INC."
   Color Scanners.
   Dirección: 21 Park Avenue, Hudson
   New Hampshire 03051.
   Teléfono: (603) 882-5200.

7. CATALOGO DE "LORAL FAIRCHILD IMAGING SENSORS"
   CCD Sensors, Systems & Developmental Technology.
   Dirección: 1801 McCarthy Blvd, Milpitas, CA 95035.
   Teléfono: (408) 433-2500.