1
2 ej.

UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

CLASIFICADOR DE COLOR UTILIZANDO PROCESAMIENTO DE IMAGENES

T E S I S

QUE PARA OBTENER EL TITULO DE
INGENIERO EN COMPUTACION
P R E S E N T A N :

JUAN GERARDO ACOSTA MANCILLA
CLAUDIA TORRES RODRIGUEZ

Director de Tesis: Ing. Román Osorio Comparan

México, D. F. *Octubre 1992*

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Indice

Prefacio

El Procesamiento de Imágenes es una herramienta hoy en día de gran importancia en diversas aplicaciones de la vida diaria. Debido a que esta es la clave en el análisis de muchos procesos automatizados que anteriormente eran electro-mecánicos o incluso manuales; la mayoría de estos procesos se han automatizado apoyándose en el avance de la tecnnología.

El proceso de *visión* que realiza un individuo al reconocer objetos con sus respectivos colores se encuentra dentro de los sistemas de procesamiento de imágenes. Sin embargo este procesamiento es realizado por el cerebro auxiliándose del ojo humano con una etapa de aprendizaje empírico realizada como antecedente por el individuo.

El trabajo que a continuación se expone presenta el análisis y diseño de un *Clasificador de Color* a partir de la obtención de una imagen con elementos puramente monocromáticos, el objetivo de este sistema consiste en crear una imagen en color aplicando técnicas de Procesamiento de Imágenes.

El desarrollo del sistema *Clasificador de Color* surgió como una necesidad de automatizar muchos procesos que involucran al sistema de visión humano, es decir, el análisis de imágenes para su uso posterior.

Este sistema *Clasificador de Color* utiliza técnicas de procesamiento de imágenes, debido a que este tipo de procesamiento se encarga de manipular una imagen inicial, para después de ciertas operaciones obtener una imagen final, así mismo se presenta un sistema administrador muy amigable desarrollado en base a *iconos* que representan la operación que ha de realizarse.

Capítulo 1

Procesamiento de Imágenes

1 Elementos de la Percepción Visual

El fenómeno de *visión* es indudablemente el más importante en la vida diaria del ser humano. Existen muchos fenómenos asociados con la percepción visual. Debido a la importancia y forma en que influye en el tema en cuestión, se procederá a explicar los elementos que componen esta percepción visual; los cuales son :

- Estructura del ojo humano.
- Formación de una imagen dentro del ojo.

1.1 Estructura del Ojo Humano

La visión de color esta ligada estrictamente al comportamiento del ojo humano, a lo que ocurre dentro del ojo, en el cerebro el cual es el elemento clave de esta visión del color.

Algunas características físicas como son la luz que entra en el ojo, el resultado de procesos neurales-fotoquímicos y respuestas psicológicas, son elementos que producen sensaciones de respuestas en el ser humano.

La forma del ojo asemeja mucho a la forma de una esfera, con un diámetro aproximado de 20 mm. Tres membranas (Cornea y Esclerótica, Coroide y Retina) cubren al ojo. La cornea es un tejido transparente y duro que cubre al ojo en su parte externa. Contigua a esta se encuentra una membrana opaca llamada sclera, la cual cubre el resto del globo óptico. La coroide esta directamente bajo la esclerótica. Esta membrana contiene una red de sangre que sirve como la principal nutriente de el ojo. La capa de la coroide esta fuertemente pigmentada por lo que ayuda a reducir la cantidad de luz externa que entra la ojo dentro del globo óptico.

La retina no es absolutamente uniforme tiene un lugar y un punto el cual usa el individuo cuando trata de ver cosas con mucho cuidado (enfocar objetos), este punto es llamado *'macula'* o *'fovea'*.

La figura muestra de una forma más clara una sección horizontal del ojo humano.

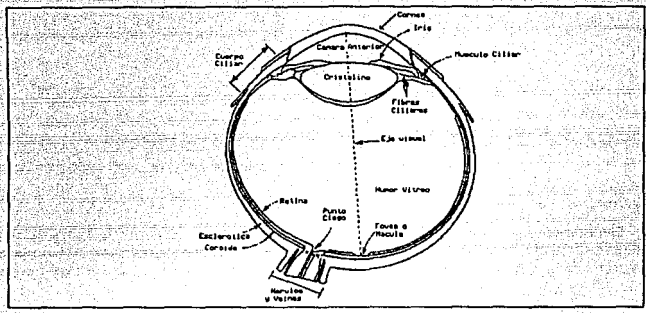


Fig. 1.1 Estructura del Ojo Humano

1.2 Formación de la Imagen en el ojo

Las imágenes que se observan, se forman de la siguiente manera :
La luz entra por el ojo a través de la cornea, por lo que diferentes partes de la retina reciben luz de diferentes partes del campo visual externo.

La principal diferencia entre el lente del ojo y un lente óptico ordinario, es su flexibilidad. La figura siguiente muestra la estructura del ojo, en la cual podemos notar que el radio de la curvatura de la superficie anterior del lente es mayor al radio de su superficie posterior. La forma del lente esta en función de la tensión de las fibras del cuerpo ciliar. Para enfocar objetos distantes, los músculos del ojo provocan que el lente se aplane relativamente. De manera contraria cuando el ojo enfoca un objeto muy cercano el lente se condensa. La imagen es primeramente reflejada en el área de la fovea, la percepción toma lugar por la excitación relativa de los receptores de luz, los cuales transforman la energía radiante en impulsos eléctricos decodificados por el cerebro.

En la siguiente figura se muestra la representación óptica del ojo observando un árbol a una distancia de 10 metros. El punto C es el centro óptico del lente del ojo.

1.3 Fundamentos del Procesamiento de Imágenes

El procesamiento de imágenes es una herramienta hoy en día de gran importancia en muchas aplicaciones de la vida diaria de cada ser humano. Debido a que las imágenes son una herramienta clave en el análisis de muchos procesos automatizados que anteriormente eran electro-mecánicos ó inclusive hasta manuales, la mayoría de estos

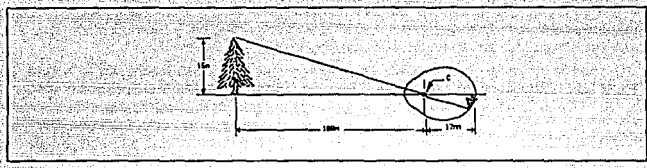


Fig. 1.2 Representación óptica del Ojo

procesos se han automatizado con el análisis de las mismas, apoyándonos en el avance de la tecnología.

Con el avance de la tecnología, se cuenta con microprocesadores sofisticados y circuitos auxiliares, que disminuyen el tiempo de procesamiento manejando 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, etc). Todo lo anterior ha sido en beneficio del procesamiento de imágenes.

1.3.1 Antecedentes Históricos

El interés en los métodos de procesamiento digital de imágenes se remonta a los años 60's y viene de dos principales áreas de aplicación : mejorar la información de imágenes para su interpretación humana, y el procesamiento de los datos de una escena para la percepción autónoma de una máquina.

Una de las primeras aplicaciones de las técnicas del procesamiento de imágenes fue, mejorar imágenes digitalizadas para el periódico, las cuales eran enviadas por un cable submarino entre Londres y New York. La introducción de este cable para el sistema de transmisión de imágenes redujo el tiempo requerido para transportar las imágenes a través del Atlántico, de un tiempo de más de una semana a menos de tres horas. Con equipo especializado se realizaba una codificación de la imagen para su envío, y por otra parte el receptor se encargaba de reconstruirla. Esta parte receptora consistía en un equipo de impresión, el cual fue sustituido por una técnica de reproducción fotográfica.

Fue en ese tiempo cuando la NASA, en uno de sus programas lunares, buscaba 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 retransmitía a los científicos para su evaluación. Previamente a las misiones RANGER

existieron otras que presentaban diversas fallas en la creación de imágenes. Estas imágenes de TV fueron tomadas en su forma analógica original y fueron convertidas a su correspondiente forma digital para ser procesadas.

El trabajo inicial en procesamiento digital de imágenes fue hecho en el NASA's Jet Propulsion Laboratory en Pasadena California. El proyecto Mariner fotografió a Marte, Venus y Mercurio; el proyecto SURVEYOR colocó cámaras en la superficie lunar; el PIONEER diez y once mando imágenes de Júpiter y Saturno; la nave VIKING equipada con cámaras, coloco algunas en la superficie de Marte. Fue así como la investigación espacial inicio el procesamiento gráfico. Hoy en día esta técnica no solo la podemos encontrar en la aplicación anterior, también es encontrada en ramas como la medicina, la física, la robótica, etc.

1.3.2 Qué es Procesamiento de Imágenes ?

El término *Procesamiento de Imágenes* ha venido siendo hoy en día una palabra clave en el mundo de la computación en donde se ha convertido en una herramienta invaluable en una gran variedad de aplicaciones.

En términos generales *Procesamiento de Imágenes* es la alteración y análisis de información de una Imagen.

Encontramos que el procesamiento de imágenes ocurre todo el tiempo en nuestra vida diaria; posiblemente el más común es el uso de "anteojos". Los anteojos correctivos, como su nombre lo indica, nos sirven para alterar las escenas que el ojo percibe antes de que lleguen a él.Modificando así todo tipo de aberraciones que se presentan en un ojo enfermo.

Otro caso común de *Procesamiento de Imágenes* es el ajuste de brillo y contraste en los controles de un televisor, realizando este ajuste hasta que la imagen sea lo más agradable posible al ojo humano.

Aún el agua en un recipiente sirve para alterar la forma de una imagen contenida en este. La imagen no es solamente reversa, sino que ofrece una imagen distorsionada. Probablemente el Sistema de Procesamiento de Imágenes más poderoso lo encontramos en nuestra vida diaria, y es :

EL OJO HUMANO y EL CEREBRO

Este sistema biológico recibe, realza, congela, analiza y almacena imágenes en enormes valores de velocidad. Todos estos son ejemplos de procesamiento de imágenes, que son tan comúnmente aceptados, que rara vez pensamos en ellos como sistemas únicos.

Fundamentalmente existen tres técnicas para implementar procesamiento de imágenes. Una técnica es óptica y las otras dos son electrónicas (analógica y digital). A pesar de que las dos últimas son electrónicas, estas difieren considerablemente, ya que

en una de ellas se representa en puntos continuos (analógica) y la otra en puntos discretos (digital).

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, telescopio, cámaras, fotografías, etc., en su forma más simple, utilizan como dispositivos elementales y necesarios lentes de aumento. Otra importante aplicación de esta técnica se encuentra en la fotografía 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 esta técnica. Las señales de TV son niveles de voltaje que varían en amplitud y representan la brillantez a través de 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.

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.

1.3.3 Representación de una Imagen Digital

En general, el procesamiento digital de imágenes es acarreado de formatos estándar de imágenes de televisión. Además casi cualquier imagen puede ser fácilmente convertida

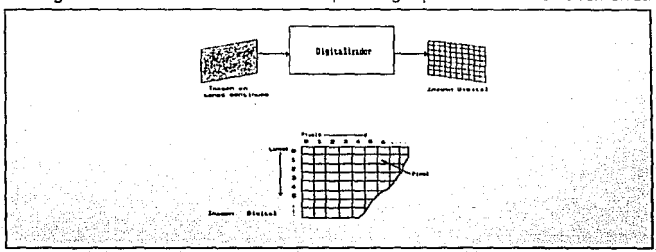


Fig. 1.3.3 Representación de una Imagen Digital

a este formato.Desde este punto asumiremos que todo nuestro procesamiento es consecuencia de imágenes estándar de televisión blanco y negro.

Dentro del dominio digital una imagen puede ser considerada como una matriz, en donde los índices de sus renglones y columnas identifican un punto de la imagen, y su correspondiente valor de elemento en la matriz identifica su correspondiente nivel de gris. A los elementos de este arreglo digital se les denomina: *'elementos de la imagen'*, *'elementos de picture'*, *'pixels'* ó simplemente *'pels'*, siendo los dos últimos las abreviaturas usadas de "*picture elements*".

La figura 1.3.3 ilustra mejor el concepto de imagen digital, la cual no es más que una representación discreta de una imagen en tonos continuos.

1.3.4 Bases para el Procesamiento Digital de Imágenes

El campo del procesamiento de imágenes es muy extenso, abarcando un rango amplio de capacidades. Cualquier acción que opera sobre o usando información de una imagen, cae dentro de la disciplina de procesamiento de imágenes. Dos términos que ayudaran a la comprensión de dicho campo, son :

- Operación en una imagen.
- Proceso en una imagen.

La operación en una imagen es cualquier acción sobre esta, definida como una aplicación. Por otra parte, el proceso en una imagen define como una operación dada es implementada sobre la imagen.

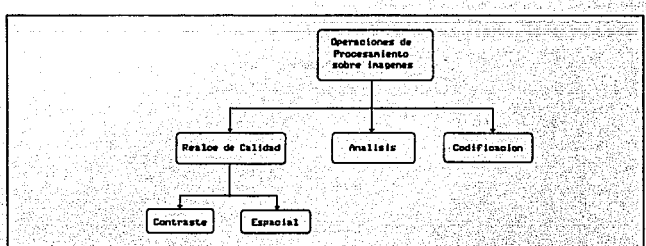


Fig. 1.3.5 Clasificación de Operaciones sobre Imágenes

1.3.5 Clasificación de las Operaciones sobre Imágenes

Tres clases de operaciones sobre imágenes pueden ser usadas para subdividir el campo de procesamiento de imágenes en sub-partes más comprensibles. Esta clasificación puede verse en la siguiente figura.

Estas operaciones son:

- Realce de calidad en la imagen.
- Análisis de imagen.
- Codificación de imagen.

1.3.5.1 Realce en la Calidad de la Imagen

El realce de la calidad en una imagen sirve para mejorar ó en alguna forma alterar la calidad de la misma, es decir su apariencia es modificada. Esta operación es en algunos casos subjetiva, ya que para una aplicación una imagen será la mejor, mientras que para otra aplicación distinta será defectuosa.

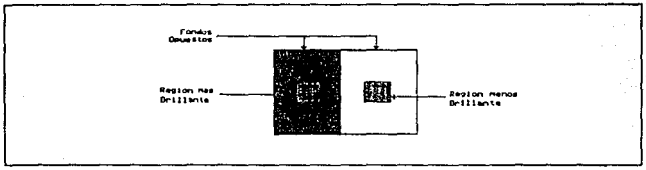


Fig. 1.3.5.1 Efecto de Contraste Simultaneo

Un efecto, referido como CONTRASTE SIMULTANEO, es una ilusión donde la brillantez percibida de una región esta en función de la intensidad del área circundante.Por ejemplo si tenemos dos cuadros de la misma intensidad pero con fondos totalmente opuestos, podemos notar que el cuadro que brilla más es el que tiene un fondo oscuro y el que aparece más oscuro es el que tiene un fondo claro.De esta forma podemos creer que los cuadros tienen diferentes intensidades. Como se muestra en la figura 1.3.5.1

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

La operación para realzar la calidad de la imagen puede ser subdividida en dos partes:

CONTRASTE y ESPACIAL.

El realce de contraste trata la alteración de brillo dentro de una imagen. Blancos, negros y grises son intensificados o suprimidos resaltando fases que difícilmente se aprecian en la imagen original.

El realce espacial modifica el detalle del contenido de una imagen. La variedad de estas dos subclases permiten al observador una gran flexibilidad en el realce espacial y de contraste para una modificación ó corrección en una imagen.

1.3.5.2 Análisis de Imagen

Esta operación produce resultados en información numérica o gráfica basada en características de la imagen original con el objetivo de describir algunos aspectos de la misma y presentar el resultado al observador. Existen varias operaciones de análisis comúnmente usadas, como son :

- Describir calidad.
- Asistiendo en realzado de operaciones.
- Descripción de características de la escena.
- Reconocimiento de patrones.

La operación más común en el análisis de una imagen es el HISTOGRAMA. Este presenta en forma de gráfica de barras la distribución de brillo que muestra una imagen siendo esta información de mucha ayuda para corregir las degradaciones de brillantes y contraste que se presenten.

1.3.5.3 Codificacion de Imagen

La operación final en el procesamiento de imágenes es la codificación, la cual sirve para reducir la cantidad de información necesaria para describir una imagen.Existen dos formas de codificación; la primera es codificar la imagen de tal forma que no se pierda la información. De esta forma la reconstrucción del esquema original se obtiene únicamente a partir de la versión codificada.

La segunda es codificarla de una forma condensada, un ejemplo de esto es cuando se particiona una imagen en estructuras primitivas, codificando únicamente el lugar y orientación de cada una de ellas.

1.3.6 Tipos de Resolución

Las imágenes gráficas sobre la pantalla están siempre dibujadas con pixels. Para una correcta interpretación de una imagen digital, es importante entender el concepto de RESOLUCION, este concepto se define como la limitante del proceso de digitalización. En otras palabras, la resolución en una imagen es lo que nos permitirá una mejor visualización de esta a medida que la resolución se vaya incrementando. Esto se debe a que el termino Resolución en este caso se refiere al número de pixels horizontales sobre una línea.

Tres resoluciones son básicamente soportadas : la baja, media y alta. El número de pixels por línea de acuerdo a las resoluciones antes mencionadas son los que se muestran a continuación :

Resolucion	Pixels/Linea	Caracteres (col x ren)
baja	160	20 x 25
media	320	40 x 25
alta	640	80 x 25

Una buena resolución requiere al menos de 350 pixels verticalmente.

En procesamiento de Imágenes la resolución esta definida por dos parámetros : ESPACIAL y de BRILLANTES unido a un tercero que no juega un papel importante en la visualización de una imagen, el FRAME RATE.

1.3.6.1 Resolución Espacial

El termino ESPACIAL se refiere al concepto de espacio, en nuestro caso a un espacio de dos dimensiones. Nuestro objeto de dos dimensiones es una imagen reparada en anchura y altura. Cuando hablamos de 'resolución espacial', estamos describiendo en cuantos pixels esta dividida nuestra imagen. Simplemente, el objetivo es tener una resolución lo más cercana posible a nuestra imagen original.

De una manera idónea, deseamos digitalizar una imagen, la cual no pierda información en la transferencia de la imagen original a la digitalizada, significando esto, que si desplegamos una imagen digitalizada deberá ser idéntica a la original para el observador. Para entender mejor el criterio de como establecer el número necesario de pixels requeridos en una imagen digital, debemos introducir el concepto de FRECUENCIA ESPACIAL, también determinada como la razón en la cual la brillantez de una imagen cambia de obscuro a claro.

La resolución espacial toma una estimación en la variación del cambio de brillo en una imagen haciendo un barrido de izquierda a derecha, así como también de arriba hacia abajo.

Para tomar una decisión acerca de la razón de muestreo necesaria para una apropiada resolución en una imagen, se siguió el Criterio de Nyquist, también conocido como el Teorema de Muestreo. Esta teoría en términos matemáticos, nos dice que para representar completamente el valor

La imagen con resolución de 32 líneas X 32 pixels muestra con tosquedad que algunos detalles se pierden por el efecto de bloque en los pixels. A medida que la resolución espacial aumenta, la imagen aparece más y más natural.Una televisión, de hecho, tiene

485 líneas visibles y 380 pixels por línea. Viendo la imagen desde una distancia razonable, resulta difícil que el observador note estas líneas, pero a medida que se acerca las líneas son claramente visibles.

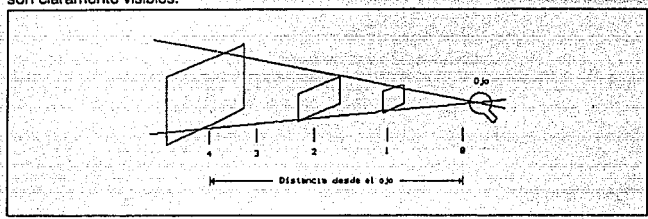


Fig. 1.3.6.1 Factores de la Resolución Espacial

En la figura 1.3.6.1 se ilustra como la resolución espacial escogida en una imagen debe tomar en cuenta tres factores : el detalle de una imagen original deberá ser visto en la imagen digitalizada, el tamaño desplegado de la imagen digital y la distancia del observador hacia está.

1.3.6.2 Resolución de Brillantez

El segundo tipo de resolución es la de BRILLANTEZ. Cada pixel representa en el punto de su muestra el brillo de la imagen original.

El concepto de resolución de brillo se refiere a que tan certera es la brillantez en un pixel comparada con la misma localidad en la imagen original. El proceso de digitalización muestrea la imagen original a locaciones predeterminadas. Cada brillantez muestreada es convertida a un valor numérico entero, a esta operación se le conoce como CUANTIZACION.

La operación de cuantización convierte en un punto de muestreo 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.

1.3.7 Histograma

El HISTOGRAMA es una representación en forma de gráfica de barras de la distribución de brillo que muestra una imagen, siendo esta información de mucha ayuda para corregir las degradaciones de brillantez y contraste que se presenten. Así mismo el histograma nos proporciona una medida concisa y fácil de leer de los atributos de contraste en una

imagen. Dándonos una representación de la concentración de pixels contra el nivel de brillo en una imagen.

Usando esta gráfica nosotros somos capaces de ver inmediatamente si la imagen es básicamente oscura o clara y si el contraste es alto o bajo. Además de que nos proporciona la primera pista de que mejoras debemos hacer al contraste para que la imagen se vuelva subjetivamente más agradable al observador.

Por lo tanto, podemos decir que los HISTOGRAMAS son una herramienta muy útil en el trabajo de procesamiento digital de imágenes, tanto cualitativa como cuantitativamente. En la fig. 1.3.7 se muestra la gráfica de un histograma típico de una imagen.

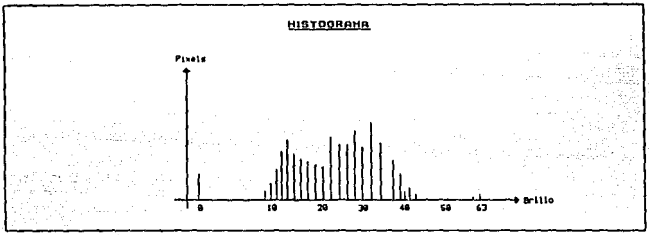


Fig. 1.3.7 Histograma Típico de una Imagen

1.3.8 Pasos Fundamentales en un Sistema de Procesamiento Digital de Imágenes

El procesamiento digital de imágenes encierra un amplio rango de elementos en software y hardware. Para realizar la tarea de procesamiento de imágenes, se sugieren los siguientes pasos fundamentales en este proceso.

En primera instancia se debe entender el dominio del problema, el objetivo al que se desea llegar y la salida deseada. Con todo esto, podemos decir que el primer paso es la "Adquisición de la Imagen", es decir adquirir una imagen digital. Para hacer esto se debe contar con un sensor de imágenes y tener la capacidad de digitalizar la señal producida por el sensor. Este sensor puede ser una cámara de TV monocromática o a colores (este tipo de cámara produce una imagen completa en 1/30 de seg.) ó bien una cámara de rastreo de línea (que produce una sola línea de la imagen a la vez).

Después de que una imagen digital ha sido obtenida, el siguiente paso es un "Preprocesamiento de la Imagen". La función de este preprocesamiento es mejorar la imagen, para incrementar las posibilidades de exito de la imagen en otros procesos. Este preprocesamiento típicamente trata con técnicas de realce de contraste, eliminación de

ruido, y aislamiento de regiones cuya textura indiquen una posibilidad de información alfanumérica.

La siguiente etapa trata con la "Segmentación". Esta segmentación esta definida como la partición de la imagen de entrada en sus constituyentes partes u objetos. En general esta etapa es una de las más difíciles tareas en el procesamiento digital de imágenes. En términos de reconocimiento de caracteres, la función de la segmentación es extraer los caracteres individuales y palabras del fondo de la imagen (background).

La salida de esta etapa es usualmente renglones de datos de pixels, la cual debe de ser representada como un límite o una región completa. El escoger esta REPRESENTACIÓN es solamente parte de la solución para la transformación de los renglones de datos a una forma mas conveniente para los procesos computacionales subsecuentes. Un método debe ser especificado para describir los datos junto con sus características. La DESCRIPCIÓN, también llamada "Selección de Características", obtiene información cuantitativa sobre las características básicas para diferenciar una clase de objetos de otros.

En la última etapa, se encuentra el "Reconocimiento e Interpretación". Este reconocimiento es el proceso que asigna una etiqueta a un objeto, basado en la información provista por sus descriptores. Por su parte la interpretación asigna un significado a un conjunto de objetos reconocidos.

La figura 1.3.8 muestra de una forma gráfica los pasos fundamentales del procesamiento de imágenes.

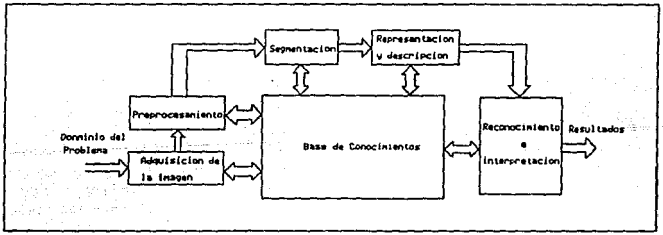


Fig. 1.3.8 Pasos del Procesamiento de Imágenes

1.3.9 Elementos de un Sistema de Procesamiento Digital de Imágenes

Un sistema de procesamiento digital de imágenes es una colección de dispositivos de Hardware con módulos de: Digitalización, Almacenamiento, Display y Procesamiento digital de Imágenes.

Los elementos que componen un sistema de procesamiento de imágenes de propósito general, son aquellos que permiten ejecutar las siguientes operaciones sobre una imagen:

1.- Adquisición
2.- Almacenamiento
3.- Procesamiento
4.- Comunicación y
5.- Despliegue.

1.3.9.1 Adquisición de una Imagen

Existen dos elementos que son requeridos para la adquisición de imágenes digitales. El primero es un dispositivo físico sensitivo a una banda en el espectro de energía electromagnética, (banda infrarroja, rayos x, ultravioleta) que produce una señal eléctrica de salida proporcional al nivel de energía censado. El segundo, es un dispositivo llamado digitalizador, el cual convierte la señal eléctrica de salida a su correspondiente forma digital.

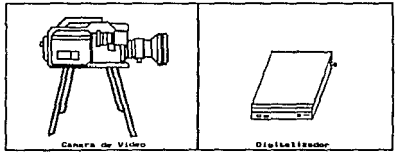


Fig. 1.3.9.1 Elementos para la Adquisición de Imágenes

Una imagen proporcionada en forma de transparencia, diapositiva, fotografía o diagrama es primero digitalizada y almacenada como una matriz de dígitos binarios en la memoria de la computadora. Esta imagen digitalizada puede ser entonces procesada y/o desplegada en un monitor de alta-resolución.

1.3.9.2 Almacenamiento de una Imagen

Para una imagen de 8-bits de 1024 x 1024 pixels de tamaño, se requieren un millón de bytes para su almacenamiento. Debido a esto el proveer a un sistema de procesamiento de imágenes de un adecuado método de almacenamiento, resulta indispensable.

Las técnicas de almacenamiento caen dentro de tres principales categorías :

a) Almacenamiento corto para usar durante el procesamiento.

b) Almacenamiento en línea para una llamada relativamente rápida.

c) Almacenamiento en archivos, caracterizado por acceso no muy frecuentes.

Un método para el almacenamiento corto consiste en utilizar la memoria de la computadora, ó bien usar tarjetas especiales (bancos de memoria), que almacenan una o más imágenes, las cuales pueden ser accesadas rápidamente. Sin embargo el rango de almacenamiento se encuentra limitado por el tamaño físico de la memoria o bien de la tarjeta especial.

El almacenamiento en línea generalmente se realiza en discos magnéticos. Los discos Winchester son muy comunes, pero una tecnología más reciente utiliza discos ópticos de 5 1/4 pulgadas que almacenan cerca de un GByte. El factor clave que caracteriza al almacenamiento en línea es su frecuente acceso a los datos.

Finalmente el almacenamiento archivado es caracterizado por sus requerimientos de almacenamiento masivo y su no-frecuente necesidad de acceso a los datos. Las cintas magnéticas y los discos ópticos son los dispositivos más comúnmente usados en esta aplicación.

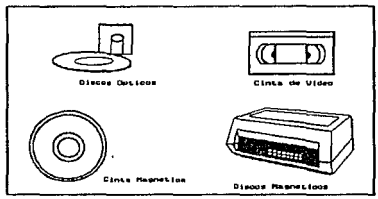


Fig. 1.3.9.2 Elementos para el Almacenamiento de Imágenes

1.3.9.3 Procesamiento de una Imagen

El procesamiento de imágenes digitales involucra procedimientos que son usualmente expresados en forma algorítmica. Por lo que, con excepción de la adquisición y despliegue, la mayoría de las funciones de procesamiento de imágenes se pueden implementar en software. La única razón por la que algunos procesos se implementan en hardware, se debe a la necesidad de velocidad para algunas aplicaciones ó para librar algunas limitaciones computacionales.

A pesar de que los sistemas de procesamiento de imágenes de larga escala continúan vendiéndose para aplicaciones masivas, tales como el procesamiento de imágenes de satélite, la tendencia a minimizar y mezclar pequeñas computadoras de propósito general equipadas con hardware para procesamiento de imágenes continua.

El procesamiento de imágenes se caracteriza por tener soluciones especificas.

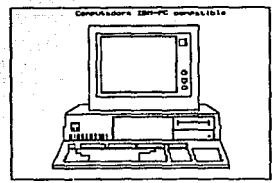


Fig. 1.3.9.3 Elemento para el procesamiento de Imágenes

1.3.9.4 Comunicación

Las comunicaciones en los sistemas de procesamiento de imágenes digitales, principalmente involucran aquellas que son locales entre sistemas de procesamiento y las remotas desde un punto a otro, típicamente en conexión con la transmisión de datos de una imagen. El HARDWARE y SOFTWARE para comunicaciones locales se encuentran disponibles para la mayoría de las computadoras, utilizando los protocolos de comunicación estandares.

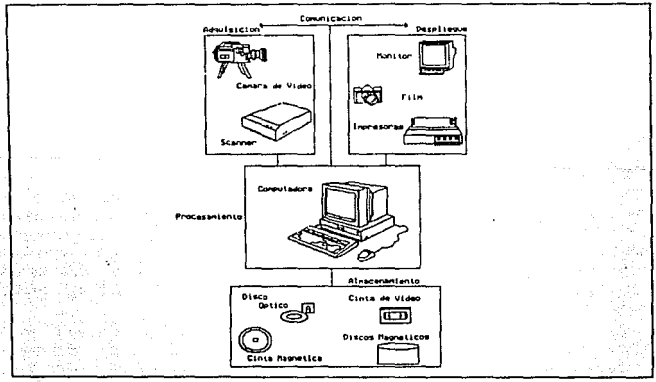


Fig. 1.3.9.4 Elementos para la Comunicación de Imágenes

1.3.9.5 Desplegado de una Imagen

En los modernos sistemas de procesamiento, los monitores de TV monocromáticos y de color, son los principales dispositivos utilizados para el desplegado de imágenes. En los sistemas CRT de acceso-aleatorio, la posición vertical y horizontal del rayo de electrones es controlado por la computadora, generando de esta manera la imagen de salida. En esta imagen, la intensidad de cada punto esta modulada por un voltaje que es proporcional al valor correspondiente del punto (pixel) en el arreglo numérico, el cual varia desde cero (para puntos 'negros') hasta el valor de 1, que es la máxima intensidad para puntos blancos.

Para desplegarse la imagen es almacenada en un buffer de memoria de acceso-rápido, el cual refresca el monitor 30 frames/s, produciendo de esta manera una imagen de visibilidad continua.

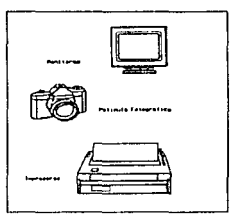


Fig. 1.3.9.5 Elementos para el Despliegue de Imágenes

1.4 Procesamiento de imágenes a color

1.4.1 Introducción

El uso del procesamiento de imágenes a color es motivado por dos principales factores. En primer lugar el análisis automatizado de una imagen a color es una poderosa descripción que ofrece una simplificada identificación de objetos y extracción completa de una escena. En segundo lugar, cuando un humano realiza un análisis es mucho más motivador para él si se muestra color, ya que el ojo humano puede distinguir miles de tono de color e intensidades, en comparación con las pocas docenas de tono de gris que nos ofrece una imagen en tonos continuos.

El procesamiento de imágenes a color esta dividido en dos principales aéreas: "Procesamiento de Imágenes en Colores Reales y Procesamiento de Imágenes en Pseudo-color". Dentro de la primera categoría, las imágenes son adquiridas típicamente con sensores de color, tales como una cámara de TV a color o scanner a color. Dentro de la segunda categoría, el problema consiste en asignar un tono de color a una intensidad

monocromática en particular o a un rango de intensidades. Hasta hace poco relativamente, mucho procesamiento de imágenes a color era realizado en niveles de pseudo-color. El significativo progreso realizado en los años 1980's ha hecho posible que los sensores de color y el hardware para el procesamiento de imágenes a color este disponible a un razonable precio. Como resultado de estos avances, las técnicas de procesamiento de imágenes en colores reales esta tomando un valor significativo en un amplio rango de aplicaciones.

1.4.2 La Percepcion del Color

Antes de iniciar el estudio de los principios fundamentales del color hay que buscar la compresión del proceso que implica la percepción de las cosas; cómo se forma en nuestro cerebro las ideas de los objetos y la influencia que existe en ello de aspectos físicos y psicológicos.

El hombre se ha encontrado siempre rodeado de colores, pero sólo hace poco tiempo que éstos se han podido producir y emplear con amplitud. Los colores de la naturaleza tienen una gran influencia sobre el hombre, quien la recibe voluntaria o involuntariamente.

La primera influencia que el hombre recibe de los colores viene señalada por la diferencia entre el día y la noche. Los colores que representan al día, amarillos, se relacionan con la actividad y la esperanza; y los que representan la noche, azules oscuros, con el descanso y la pasividad. En la actividad, el color rojo se relaciona con las acciones de ataque. La autodefensa y conservación vienen relacionadas con el verde. El rojo es un excitante que aumenta en el observador la presión sanguínea y el ritmo cardíaco. Por el contrario, el azul es sedante. A continuación se proporciona los significados de los colores básicos:

AZUL.

Representa "profundidad de sentimiento" y produce tranquilidad, ternura, satisfacción y sensación de amor perfecto.

VERDE.

Representa "constancia de voluntad" y produce sensaciones de persistencia, autoafirmación, obstinación y autoestima.

ROJO.

Representa "fuerza de voluntad" y produce sensaciones de apetencia, excitabilidad, autoridad y sexualidad.

AMARILLO.

Representa la "espontaneidad" y produce sensaciones de variabilidad, expectación, originalidad y regocijo.

NARANJA.

Es la mezcla entre rojo y amarillo. Resulta estimulante y acogedor.

BLANCO y NEGRO.

Se hallan en los extremos del espectro, tienen un valor límite y son neutros, refuerzan los colores que se combinan con ellos.

GRIS.

Ocupa un lugar intermedio entre el blanco y el negro. Supone un centro pasivo, y por ello, un factor de equilibrio.

1.4.3 Fundamentos de Color

Lo que se percibe como luz, o diferentes colores, es una banda de frecuencia estrecha dentro del espectro electromagnético. Algunas otras bandas de frecuencias contenidas en este espectro se denominan ondas de radio, microondas, ondas infrarrojas y rayos X.

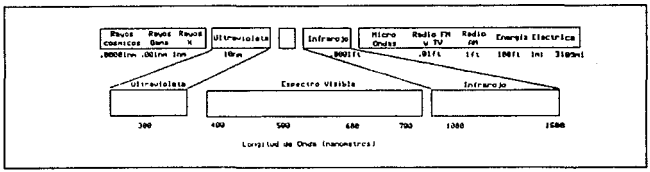


Fig. 1.4.3 Espectro Electromagnetico

Cada valor de frecuencia contenido en la banda visible corresponde a un color distinto. En el extremo de baja frecuencia hay un color rojo (en los 4.3x1014 hertz) y la más alta que se puede percibir es un color violeta (en los 7.5x1014 hertz). Entre estos dos límites, el ojo humano puede distinguir cerca de 400 000 colores diferentes. Estos colores van de los rojos a través del anaranjado y el amarillo en el extremo de baja frecuencia a los verdes, azules y violetas en el extremo opuesto.

Ya que la luz es una onda electromagnética, podemos describir los diversos colores en términos de la frecuencia o bien de la longitud de onda. La luz en el extremo rojo del espectro tiene una longitud de onda de aproximadamente 700 nanómetros (nm), mientras que la de la luz violeta en el otro extremo del espectro vale cerca de 400 nm.

Una fuente de luz como el sol o un foco emiten todas las frecuencias contenidas en el intervalo visible para producir la luz blanca, cuando la luz blanca incide sobre un objeto, algunas se reflejan y algunas son absorbidas por el objeto. La combinación de frecuencias presentes en la luz reflejada determina lo que percibimos como el color del objeto, si predominan las más bajas en la luz reflejada, el objeto se describe como rojo. En este caso, se dice que la luz percibida tiene una FRECUENCIA DOMINANTE (o bien LONGITUD DE ONDA DOMINANTE) en el extremo rojo del espectro. La frecuencia dominante se llama asimismo COLOR o MATIZ de la luz.

Otras propiedades además de la frecuencia son útiles para describir las características de la luz. Cuando se observa una fuente de luz, nuestros ojos responden al color (o frecuencia dominante) y a otras dos sensaciones básicas. A una de éstas la llamamos

Luminancia o Brillantes de la luz. La brillantes se relaciona con la intensidad de la luz: cuanto mayor sea la intensidad (o energía), más brillante se ve la fuente. La otra característica es la PUREZA o SATURACIÓN de la luz. La pureza describe cuán puro se ve el color de la luz. Los colores pasteles y pálidos se describen como menos puros. Estas tres características, frecuencia dominante, brillantez y pureza, se utilizan comúnmente para describir las diferentes propiedades que se perciben en una fuente de luz. El término Cromaticidad se utiliza para hacer referencia colectiva de las dos propiedades que describen las características del color, la pureza y la frecuencia dominante.

Por la naturaleza de la estructura del ojo humano, todos los colores son vistos como una combinación variable de tres colores llamados COLORES PRIMARIOS rojo (Red), verde (Green) y azul (Blue). Para propósitos de estandarización, el CIE (Comisión Internacional sobre Iluminación) designó en 1931 los siguientes valores específicos de la longitud de onda a los tres colores primarios: azul = 435.8 nm, verde = 564.1 nm y rojo = 700 nm. Sin embargo ningún color como tal puede ser llamado rojo, verde o azul, de esta forma el tener tres especificas longitudes de onda para propósitos de estandarización no significa que estos tres fijos componentes RGB puedan generar todo el espectro de colores. Esto es importante, porque el uso de la palabra PRIMARIOS ha sido comúnmente mal interpretada, dándole el significado que cuando se mezclan los tres colores primarios estándar en una variada proporción de intensidades puede producir todos los colores visibles. Esta interpretación no es correcta a menos que la longitud de onda también sea modificada.

En definitiva, en el sistema primario aditivo al mezclarse por partes iguales la luz roja, verde y azul, se obtiene la luz blanca. Existe también un sistema diferente de colores primarios: rojo, azul y amarillo; que es usado cuando se desea crear colores por mezcla de los mismos pintándolos sobre un papel. Este proceso es conocido como mezcla sustractiva de colores, porque se comienza con un papel blanco el cual refleja todos los colores por igual y se añade el filtro de pigmentos que refleja la luz blanca para sustraer ciertos colores, por ejemplo al mezclar los tres colores primarios se obtiene el negro; ya que se sustrae todos el reflejo de la luz (el papel se ve negro cuando ninguna luz puede ser reflejada).

Cabe hacer notar que dos de los nombres de los colores aditivos primarios son los mismos que dos de los sustractivos primarios : ROJO y AZUL. Lo cual no es lo mismo. El rojo y el azul son correctos para los primarios aditivos, pero para los primarios sustractivos el rojo y el azul deberán técnicamente ser nombrados MAGENTA, el cual es una mezcla de rojo-azul, y CYAN es cual es una mezcla de los colores azul-verde, respectivamente.

La relación de estos dos sistemas de colores primarios puede causar confusión. La figura 1.4.3' muestra el translape de los círculos de colores para ambos sistemas aditivo y sustractivo. La parte izquierda de la figura muestra como la luz blanca consiste de una mezcla por igual de los tres colores aditivos -rojo, verde y azul - y la parte derecha de la figura muestra que una mezcla por igual de los colores sustractivos da el negro.

Es preciso señalar que el color como tal no existe. Esto quiere decir que el color es una experiencia perceptiva que tenemos gracias a nuestro sistema visual; es un proceso

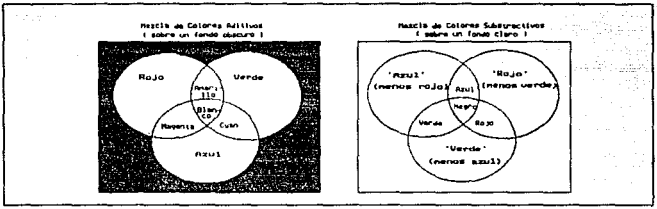


Fig. 1.4.3' Mezcla de Colores Aditivos y Substractivos

en el que intervienen a un tiempo dicho sistema visual, constituido esencialmente por la retina de nuestros ojos, una fuente de luz y una superficie determinada que es la que percibimos como color; evidentemente si una de estas condiciones se altera, todo el proceso también, y de hecho, algunas investigaciones señalan que determinados animales que tienen otros órganos distintos para la recepción de color, lo perciben también de forma diferente.

Por tanto, nunca deberemos referirnos a un objeto diciendo que es de tal color, sino que lo vemos de tal color. Así lo que sucede, cuando percibimos un objeto como de color rojo, es que la superficie de ese objeto refleja una parte del espectro de luz blanca que recibe y absorbe las demás.

1.4.4 Modelos de Color

El propósito de estos modelos es facilitar las especificaciones de colores en algún formato estándar. En esencia un modelo de color es una especificación de un sistema de coordenadas 3-D y un subespacio dentro del sistema donde cada color es representado por un solo punto.

Muchos modelos en la actualidad están orientados ya sea hacia hardware (tales como monitoresa color e impresoras) o hacia aplicaciones donde la manipulación del color es la meta (tales como la creación de gráficas a color para animación). De los modelos orientados a hardware los más comúnmente usados en la practica son el RGB(red, green, blue) que es el modelo para los monitores y para la gran variedad de cámaras de video a color, el CMY(cyan, magenta, yellow) que es el modelo para las impresoras a color y el modelo YIQ (Y es la luminancia y I y Q son dos componentes cromáticos llamados en fase y cuadratura respectivamente) el cual es el standard para la TV a color. De los modelos frecuentemente usados para manipulación de imágenes a color son el modelo HSI (hue

[matiz], saturation [saturación], intensity [intensidad]) y el modelo HVS (hue [matiz], saturation [saturación], value [valor]).

Los modelos de color más frecuentemente usados en el procesamiento de imágenes son el RGB, YIQ, CMY, y el HSI.

1.4.4.1 Modelo de Color RGB

Este modelo esta basado sobre un sistema de coordenadas cartesianas. El subespacio cartesiano de interés es el cubo mostrado en la figura 1.4.4.1, en la cual los valores RGB son tres esquinas y el cyan, magenta y amarillo son otras tres esquinas; el blanco es el origen y el negro es la esquina más lejana del origen. En este modelo, la escala de gris se extiende desde el punto que forma al color blanco hasta el que es formado por el negro, y los colores son puntos sobre o dentro del cubo, definidos por vectores que se extienden desde el origen. Por conveniencia, se asume que todos los valores han sido normalizados, de tal forma que el cubo mostrado es unitario. Esto es que todos los valores RGB se encuentran dentro del rango [0,1].

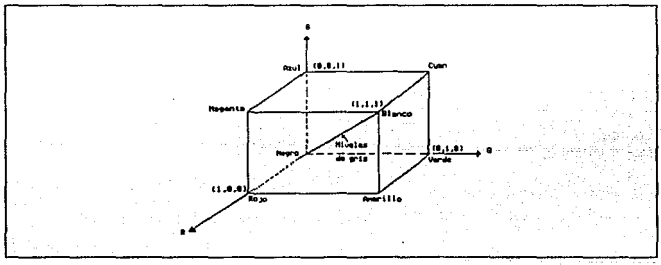


Fig. 1.4.4.1 Cubo Unitario de Colores RGB

Las imágenes en este modelo están formadas por tres planos independientes, uno para cada color primario. Cuando en un monitor RGB son suministradas estas tres imágenes, el fósforo de la pantalla es combinado para producir una imagen a color. De esta manera el uso del modelo RGB para el procesamiento de imágenes es entendido cuando estas por si mismas son expresadas en términos de sus tres planos de color. Alternativamente muchas cámaras a color usadas para adquirir imágenes digitalizadas utilizan el formato RGB, lo cual hace que este modelo sea importante en el procesamiento de imágenes.

1.4.4.2 Modelo de Color YIQ

El modelo de color YIQ es usado por los canales comerciales de televisión. Básicamente el modelo YIQ es una recodificación del modelo RGB para transmitir eficientemente y mantener compatibilidad con la televisión monocromática standard. De hecho la componente Y del sistema YIQ provee toda la información requerida por la televisión monocromática.

El modelo YIQ fue diseñado para tomar ventaja del sistema visual humano el cual es más sensitivo a los cambios en la luminancia que a los cambios en los matices o en la saturación. De esta forma el estándar YIQ utiliza un ancho de banda más grande (o bits en el caso del color digital) para representar Y, y un ancho de banda menor (o menos bits) para representar I y Q.

La principal ventaja del modelo YIQ en el procesamiento de imágenes es que la luminancia (Y) y la información de color (I y Q) están separadas. Teniendo en mente que la primera es proporcional a la cantidad de luz que es percibida por el ojo humano. De esta manera la importancia de esta separación es que la componente de luminancia de una imagen puede ser procesada sin afectar el contenido de su color.

1.4.4.3 Modelo de Color CMY

Los tres colores primarios sustractivos: cyan, magenta y amarillos son usados en el modelo de color CMY.

Como se ha notado, el cyan puede formarse agregando luz verde y azul. Por tanto, cuando se refleja luz blanca en tinta de color cyan, la luz reflejada no debe tener componente roja.

Esto es, la luz roja es absorbida, o substraída, por la tinta. Análogamente, la tinta magenta resta la componente verde de la luz incidente y el amarillo sustrae la componente azul. En la figura 1.4.4.3 se ilustra una representación en cubo del modelo CMY.

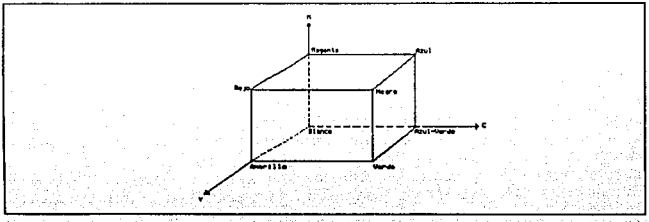


Fig. 1.4.4.3 Modelo de Color CMY

El modelo CMY, el punto (1,1,1) representa el negro, ya que todas las componentes de la luz incidente se sustrae. El origen representa la luz blanca. Cantidades iguales de cada uno de los colores primarios producen grises, a lo largo de la diagonal principal del cubo. Una combinación cyan y magenta produce luz azul, ya que las componentes roja y verde de la luz incidente son absorbidas. Otras combinaciones de color se obtienen por medio de un proceso sustractivo análogo.

A diferencia de los monitores de video, que producen un modelo de color combinando luz de los fósforos de la pantalla, los dispositivos de copia dura como las graficadoras producen una imagen de color cubriendo un papel con pigmento de color.Se observa los colores por luz reflejada, lo cual es un proceso sustractivo.

El proceso de impresión que a menudo se utiliza con el modelo CMY genera un punto de color con un conjunto de cuatro tintas, un poco parecido a un monitor RGB que utiliza un conjunto de tres fósforos. Se usa un punto para cada uno de los colores primarios (cyan, magenta y amarillo) y un adicional para el negro; se incluye este ultimo, ya que la combinación de azul-verde, magenta y amarillo comúnmente produce gris oscuro en vez de negro. Algunas graficadoras producen diferentes combinaciones de color esparciendo la tinta de los tres colores primarios uno sobre otro y permitiendo que se mezclen antes de secarse.

1.4.4.4 Modelo de Color HSI

El matiz es un atributo, que describe un color puro (amarillo puro, naranja o rojo) , donde la saturación da la medida del grado en el cual un color puro es disuelto por la luz blanca. Este modelo HSI es muy usado por dos principales factores. Primero, la componente de intensidad I esta separada de la información de color en la imagen. Segundo, los componentes de matiz y saturación están íntimamente relacionados con la forma en la cual el humano percibe el color. Estas características hacen del modelo HSI una herramienta ideal para el desarrollo de algoritmos de procesamiento de imágenes basados sobre algunas de las propiedades de sensación del sistema visual humano.

Los ejemplos de uso de este modelo van desde el diseño de sistemas de imágenes que determinan automáticamente la madurez de frutas y vegetales, a los sistemas de muestreo de reconocimiento de color o inspección de la calidad final de los mismos. En estas y en similares aplicaciones, el problema consiste en sistemas basados en operaciones sobre las propiedades de color de la misma forma que una persona usa estas propiedades para realizar la misma tarea.

1.4.5 Filtros

En fotografía se utilizan dos tipos de filtros: ópticos y de partículas o de solución. Cada filtro actúa como un tamiz que permite el paso de ciertas porciones de energía o de soluciones e impide el paso de otras porciones.

Los filtros ópticos se utilizan principalmente para controlar la energía de exposición (longitudes de onda visibles, ultravioletas e infrarrojas) y la composición de la iluminación de trabajo en las zonas en las cuales se manejan materiales sensibles.

1.4.5.1 Materiales de los Filtros

Los filtros ópticos para la mayoría de los equipos de fotografía se fabrican con películas de gelatina, acetato, plástico o de cristal. Algunos filtros para aplicaciones científicas o técnicas pueden ser líquidos o gaseosos. Los materiales que constituyen los filtros contienen una distribución equilibrada de colorantes o de otras sustancias, los cuales absorben, reflejan o dispersan ciertas longitudes de onda. Los filtros de cristal pueden ser de cristal de color, o bien estar formados por una capa de gelatina que cubre trozos de cristal incoloro o que se hallan comprendida entre ellos.

La mayoría de los filtros se colocan frente al objetivo de la cámara o de la ampliadora. Pero a veces se disponen detrás del objetivo, en el haz formador de imagen (por ejemplo, dentro de una cámara de gran formato), para protegerlos o para que no infieran con otros accesorios del objetivo. Existe una excepción importante; los filtros de acetato no deben situarse delante del objetivo ni en el haz formador de la imagen, pues el grosor y las características ópticas del acetato pueden afectar negativamente a la calidad óptica de la misma.

Los filtros de acetato se utilizan para ajustar la potencia de las fuentes luminosas. Suelen colocarse en el cabezal de la ampliadora, para controlar el color de la luz utilizada en la obtención de copias en color o en blanco y negro con contraste selectivo. Los filtros de acetato y los de plástico se utilizan también para controlar el color de fuentes luminosas, por ejemplo del flash electrónico. En este caso el filtro se coloca sobre la luz.

Generalmente, los filtros de plástico y los de cristal tienen forma de discos circulares y van enmascarados por una anilla metálica. Ciertos filtros pueden enroscarse en el objetivo; otros no poseen paso de rosca y deben montarse mediante un adaptador y una anilla de retención. Muchos filtros sin paso de rosca existen en diámetros estandarizados en series de tamaños. Esto permite utilizar varios objetivos de diferentes diámetros con adaptadores que admitan un único tamaño de filtros.

1.4.5.2 Manejo y Almacenamiento de los Filtros

Los filtros utilizados en el recorrido de la imagen deben manejarse con cuidado para evitar la degradación de la misma.

Los filtros de gelatina están cubiertos por una capa fina de laca, que los protege muy poco de la manipulación descuidada. Sólo deberían ser sostenidos por los bordes o por los ángulos. Cuando no se utilicen, resulta práctico y seguro guardarlos en su sobre original o en papel limpio entre las páginas de un libro. En todo caso los filtros de gelatina han de guardarse planos y seguros en un lugar oscuro y seco. Esto se debe a que una

presión indebida sobre los filtros de gelatina los deformará permanentemente y la humedad les restará trasparencia.

Para eliminar las partículas de polvo de los filtros de gelatina hay que cepillarlos ligeramente con un pincel de pelo de camello limpio y seco. También se elimina el polvo dirigiendo sobre el filtro aire limpio y seco. Para cortar un filtro de gelatina, debe colocarse el mismo entre dos trozos de papel rígido y limpio, emplear unas tijeras afiladas. La línea de corte puede marcarse sobre el papel.

Si se montan los filtros de gelatina en el portafiltros mencionado anteriormente o entre marcos de cartón rígido cuyos ángulos han sido cerrados con cinta adhesiva, se tendra una protección adicional durante su manipulación.

Los filtros de cristal y de gelatina que han sido montados entre cristales deben ser tratados con el mismo cuidado que los objetivos. Hay que guardarlos en estuches protectores e impedir que se deposite en ellos polvo o humedad. Un filtro montado en cristal no debe lavarse con agua, aunque sus bordes hayan recibido un recubrimiento protector de la humedad. Si el agua entra en contacto con la gelatina en los bordes de un filtro montado en cristal, ésta se hinchará y separará los cristales, permitiendo la entrada de aire. Incluso aunque no entre aire, el filtro se deformará y su superficie óptica variará.

1.4.5.3 Combinacion de Filtros

Es posible emplear dos o más filtros conjuntamente para obtener un efecto combinado. Por ejemplo, un filtro amarillo (que reduce la luz transmitida azul) puede utilizarse con un filtro de densidad neutra para reducir la cantidad de luz transmitida o con un filtro polarizador para controlar los brillos. Cuando un filtro de color se combina con uno de densidad neutra o con uno polarizador, pueden multiplicarse sus factores para determinar el factor de la combinación: un filtro de color con un factor dos y uno de densidad neutra con un factor cuatro tienen un factor combinado de ocho, lo cual requiere tres diafragmas más de exposición. Cuando se combinan dos filtros de color, debe hacerse una prueba para determinar el factor combinado.

Cuanto más grueso sea un filtro, mayor será su efecto sobre la calidad óptica de la imagen. Debido a que los filtros de gelatina son muy delgados, resultan muy convenientes para las combinaciones de varios filtros.

1.4.5.4 Principio de los Filtros

La mayoría de las fotografías se toman mediante luz blanca que incide sobre objetos de color; muy pocos elementos de la naturaleza son verdaderamente neutros. Como se ha ya mencionado, la luz blanca es la suma de todos los colores del espectro visible, mientras que el negro es la ausencia de todos los colores. Para muchas aplicaciones puede considerarse que la luz blanca se compone de tres colores primarios: rojo, verde y azul. Cuando se absorben uno o dos de estos colores, aparece la coloración restante.

Un trozo de papel rojo es de este color debido a que refleja la luz roja. Considerando que la luz blanca que incide en él está formada por una mezcla de azul, verde y rojo, el papel absorbe las luces azul y verde, pero refleja la luz roja. Por tanto, cualquier objeto que absorba (substraiga) las luces azul y verde aparecerá rojo. Así, un filtro rojo aparece de este color, ya que absorbe (sustrae) el verde y el azul, y transmite únicamente el rojo.

La sustracción es la clave que permite comprender los principios de los filtros fotográficos, los cuales siempre substraen parte de la luz reflejada por un paisaje, antes que ésta llegue a la película de la cámara.

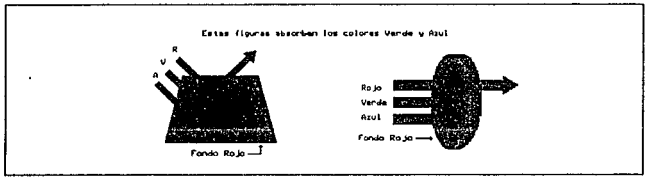


Fig. 1.4.5.4 Absorción de Colores Primarios

En la selección de filtros para efectos fotográficos lo más importante no es el color del filtro, sino los colores que absorbe, la figura 1.4.5.4 ilustra mejor este concepto. Pocos objetos son rojos, azules o verdes puros; en la luz que refleja los objetos suele haber proporciones variables de dos o tres colores primarios. Por ejemplo, un limón y una naranja reflejan los mismos colores: rojo y verde. El limón refleja casi idénticas proporciones de ellos, que aparece como amarillo al mezclarlas el ojo; en cambio, la naranja refleja mayor proporción de rojo; ambos absorben el azul. El follaje verde refleja en realidad verde, algo de azul y una pequeña cantidad de rojo.

Capítulo 2

Recursos de Cómputo

2.1 Hardware

Para la mayoría de los usuarios, las áreas más importantes en un sistema de computadoras son: el video (monitor) y teclado. La razón es que estos dispositivos son el método primario mediante el cual el usuario establece una comunicación con el equipo de computo, proporcionando y obteniendo información.

Para nuestro caso en particular, los componentes de hardware que se requieren en un sistema de procesamiento de imágenes, son:

- Cámara de video
- Monitor de TV
- Tarjeta digitalizadora
- Controlador para cambios de filtros

2.1.1 Cámara de Video.

La cámara de video (por lo general es el dispositivo de entrada más común) es el dispositivo de entrada encargado de capturar las imágenes y transferirlas por medio de una señal de video a la fase de digitalización.

La cámara utilizada para la adquisición de las imágenes utilizadas en el sistema, tiene las siguientes características:

- Video Cámara CCD monocromática.

2.1.2 Monitor.

El monitor es un dispositivo analógico, utilizado para desplegar la imagen que proporciona la cámara de video. La finalidad de este monitor externo, además de visualizar la imagen que se desea capturar, es darle un mejor enfoque, que abarque un amplio rango de tonos de gris y presente niveles mínimos de ruido para que posteriormente sea digitalizada.

El monitor de TV es un JAVELIN, que tiene las siguientes características:

- Monocromático

- Perillas de ajuste de brillo, contraste, etc.
- Entrada/Salida de Video

2.1.3 Tarjeta Digitalizadora.

La tarjeta digitalizadora es una tarjeta de circuito impreso, denominada PDS/01 desarrollada en el Departamento de Electrónica y Automatización del IIMAS, para la adquisición y despliegue de una imagen capturada con una cámara de video.

La tarjeta PDS/01 proporciona la señal de sincronía compuesta y acepta la señal de video conforme a la norma NTSC. La resolución espacial de adquisición y despliegue es de 480 líneas por 512 pixels o elementos de la imagen, la resolución cromática es de 8 bits, lo cual permite trabajar hasta 256 tonalidades de gris. La imagen es digitalizada y almacenada en el banco de memoria en 1/30 de segundo, tiempo que tarda una cámara en generar una imagen.

Para el almacenamiento de la imagen, la tarjeta cuenta con un banco de memoria RAM de 256 KBytes en una configuración que simula un doble puerto. Por un lado, el banco de memoria esta conectado al digitalizador para adquirir y desplegar imágenes, y por el otro, la PC tiene acceso directo hacia el banco como si fuera parte de su memoria. La extensión de memoria esta diseñada como memoria expandida y puede ser utilizada por la PC, con ciertas restricciones, cuando no esta siendo empleada por el digitalizador.

El digitalizador puede ser operado en tres modos: Continúo, Adquisición y Despliegue.

a) Modo Continúo.- La imagen que observa la cámara es desplegada en un monitor externo. La señal de video pasa por el convertidor analógico/digital y digital/analógico antes de ser desplegada. En este modo de operación la PC tiene acceso directo al banco de memoria, el cual se selecciona automáticamente al encender la computadora.

b) Modo de Adquisición.- Realiza la captura de la imagen enviada por la cámara de video al momento de la ejecución del comando. La información de video digitalizada es almacenada en memoria y el monitor de video continua desplegando la información en modo continuo.

c) Modo de Despliegue.- La información que se encuentra almacenada en memoria es desplegada en el monitor de video externo.

El modo de operación se selecciona por medio de un comando de control que se envía a la tarjeta digitalizadora a través de un puerto de E/S.

Las especificaciones de la Tarjeta Digitalizadora se muestran en la fig. 2.1.3. Esta tarjeta modelo PDS/01 se conecta en una de las ranuras de expansión (slots) de una computadora personal (PC). Antes de realizar la instalación se debe determinar la dirección del bloque de memoria y de los puertos de E/S.

Sistema de barrido	2:1 entrelazado
Frecuencia de barrido	15.734 KHz Horizontal
Tiempo de adquisicion	1/30 segundo
Capacidad de memoria	256 KBytes RAM estatica
Interfase	Banco de memoria 2 puertos de E/S
Sincronia	Interna
Resolucion espacial Horizontal Vertical	 512 pixels 480 lineas
Resolucion cromatica	256 tonos de gris
ENTRADA	
Senal de video de entrada	NTSC estandar (B/N) 1.0 Vpp Video: 0.7 Vpp Sync : 0.3 Vpp
Impedancia de entrada	75 ohms
SALIDA	
Senal de video de salida	NTSC estandar (B/N) 1.0 Vpp Video: 0.7 Vpp Sync : 0.3 Vpp
Impedancia de Salida	75 ohms

Fig. 2.1.3 Especificaciones de la Tarjeta PDS/01

2.1.3.1 Programación del Bloque de Memoria de la Tarjeta PDS/01

Las computadoras IBM PC/XT/AT y compatibles tienen la capacidad de direccionar hasta 1MByte de memoria RAM. Dentro de este espacio, la computadora utiliza varios bloques para su operación, dejando algunos espacios libres que están disponibles para la inserción de dispositivos periféricos. En general las computadoras personales tienen al menos un espacio de 64 KBytes libres para la instalación de extensiones de este tipo.

La dirección del banco de memoria del digitalizador dentro de la PC es completamente programable y se accesa a través de una ventana de 16 KBytes de longitud. En esta ventana existen 16 bloques en profundidad para completar los 256 KBytes requeridos.

La fig. 2.1.3.1 muestra gráficamente la forma en que la PC accesa este espacio de memoria. La programación de la tarjeta DIV/01 consiste en localizar un espacio de 16 KBytes libre dentro del mapa de memoria de la computadora y programar en la tarjeta la dirección base correspondiente. La programación se lleva a cabo mediante seis de los ocho micro-switches, localizados en la parte inferior de la tarjeta.

Para realizar el direccionamiento individual de cada localidad de memoria, se escribe primero a un puerto de control el número de bloque que corresponde a la localidad que se desea direccionar, seguido de un comando de lectura/escritura a memoria para accesar la localidad deseada dentro del bloque seleccionado. Solo es necesario escribir al puerto de control cada vez que se desea cambiar de bloque de memoria.

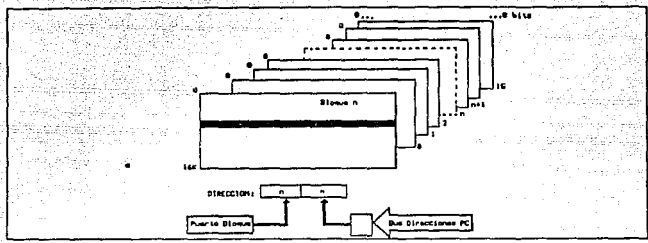


Fig. 2.1.3.1 Arquitectura de la Memoria de Video desde la PC

El manejo de la memoria desde la PC solo es posible cuando el digitalizador se encuentre trabajando en modo continuo.

La figura 2.1.3.2 muestra como una cámara adquiere una imagen y como esta es reconstruida en un monitor de video. La imagen es capturada en dos partes (campo par e impar), según la norma NTSC. Internamente en el digitalizador, se asigna a cada campo una mitad del espacio de memoria de 128 KBytes. Cada campo esta formado por 241.5 líneas y solo las primeras 240 líneas completas son digitalizadas.

Por esta razón el espacio de 8K localidades al final del campo, no contiene información valida. La imagen digitalizada completa tiene una longitud de 512 x 480, o sea, 245760 pixels. En la figura 2.1.3.3 se presenta una distribución secuencial de la memoria, cuando esta es accesada por el digitalizador de imágenes.

Fig 2.1.3.2 Reconstrucción de una Imagen de Video (par,non)

2.1.3.2 Programacion de Puertos de E/S

La comunicación de control de la PC hacia la tarjeta digitalizadora se realiza a través de dos puertos de E/S. La dirección de acceso de estos puertos debe ser programada en la tarjeta digitalizadora y debe corresponder a puertos libres dentro del mapa de puertos de

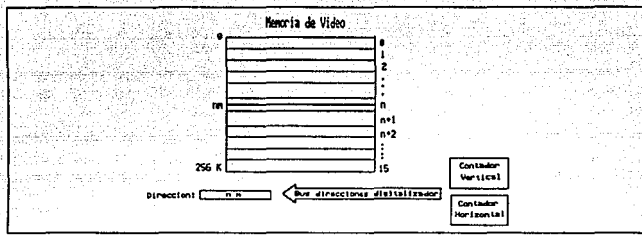


Fig. 2.1.3.3 Arquitectura de la Mem. de Video desde el Dig.

la computadora. Para efectuar dicha programación es necesario conocer la utilización de puertos de la computadora y encontrar 8 puertos libres consecutivos.

Los bits A0 y A1, son los encargados de seleccionar los dos puertos de comunicación de la PC, como se muestra en la siguiente tabla :

BITS		PUERTO
A1	A0	
0	0	Seleccion de Bloque de memoria
0	1	Seleccion de bloque de control
1	0	No se usa
1	1	No se usa

2.1.3.3 Descripcion de Puertos

Puerto de selección de bloque de memoria.- Este puerto se emplea para direccionar 1 de los 16 bloques de memoria dentro del espacio de 256 KBytes en que esta contenida la imagen. La tabla que se muestra a continuación muestra la palabra de control que se

D3	D2	D1	D0	DATO	ACCESO
0	0	0	0	X0	Bloque 0
0	0	0	1	X1	Bloque 1
0	0	1	0	X2	Bloque 2
0	0	1	1	X3	Bloque 3
0	1	0	0	X4	Bloque 4
0	1	0	1	X5	Bloque 5
0	1	1	0	X6	Bloque 6
0	1	1	1	X7	Bloque 7
1	0	0	0	X8	Bloque 8
1	0	0	1	X9	Bloque 9
1	0	1	0	XA	Bloque A
1	0	1	1	XB	Bloque B
1	1	0	0	XC	Bloque C
1	1	0	1	XD	Bloque D
1	1	1	0	XE	Bloque E
1	1	1	1	XF	Bloque F

requiere enviar a l puerto de selección de bloque de memoria y el bloque seleccionado correspondiente.

Puerto de control.- A través de este puerto se envían los comandos de control que manejan los modos de operación. Los comandos están codificados por medio de dos líneas del bus de datos como se especifica en la siguiente tabla :

BIT DATO		
B1	B2	MODO
1	1	Desplegar imagen
1	0	Continuo (manejo memoria PC)
0	1	Capturar Imagen
0	0	No permitido

2.1.3.4 Descripcion del Conector

Las señales de entrada y salida del digitalizador se transmiten a través de un conector del tipo DB-9S (hembra), el cual esta disponible en el riel posterior de la PC. La asignación de las señales en este conector se detalla en la siguiente tabla:

NUMERO DE PIN	SENAL
1	Video de entrada
2	GND (tierra)
3	Video de salida
4	Sincronia compuesta
5	+12 VDC
6	GND
7	GND
8	GND
9	GND

2.1.4 Controlador para Cambio de Filtros.

El controlador para cambio de filtros, fue diseñado e implementado para trabajar conjuntamente con el puerto paralelo, utilizando una rutina desarrollada en lenguaje 'C', para realizar la interfase con un motor de 4 pasos, que es el que se encarga de colocar el filtro apropiado frente al lente de la cámara de video.

Los elementos que conforman este controlador son:

- Un motor de pasos con las siguientes características:
  - 4 fases
  - 12 Volts
  - 36 Ohms
  - 7.5 grados/paso

- 4 optoacopladores NTE-3044
- 4 transistores 2N3053
- resistencias

2.1.4.1 Puerto Paralelo.

CONECTOR DB-25

PIN	SENAL	PIN	SENAL
1	-Habilitador	10	-Reconocimiento
2	+Bit daot 0	11	+Ocupado
3	+Bit dato 1	12	+Sin papel
4	+Bit dato 2	13	+Seleccion
5	+Bit dato 3	14	+Auto Feed
6	+Bit dato 4	15	-Error
7	+Bit dato 5	16	-Inicializa
8	+Bit dato 6	17	-Seleccion
9	+Bit dato 7	18-25	Tierra

El puerto paralelo es un conector al cual se puede unir una impresora compatible paralela ó cualquier otro dispositivo paralelo.

Este puerto utiliza un conector DB-25 el cual se especifica a continuación :

El sistema operativo DOS reserva tres nombres para los puertos paralelos, los cuales son : LPT1, LPT2 y LPT3. Estos puertos están ubicados en las direcciones de E/S 3BC, 378 y 278 respectivamente para cada uno de estos puertos. Se puede tener hasta tres puertos paralelos en la computadora solamente si el LPT1 esta sobre el adaptador de display monocromático o su equivalente. De otra manera, DOS limita a la computadora a tener dos puertos paralelos (LPT1 y LPT2).

La discusión del puerto es de importancia, porque es este puerto el que realiza la interfase de hardware entre la PC y el controlador de filtros de la cámara de video.

El uso de la conexión del puerto de impresión y la PC fue escogida principalmente por razones de rapidez y facilidad. Esto es debido a su natural paralelismo, el puerto de impresión puede transferir datos más rápidos que el puesrto serial.En la conexión paralela, muchos bits de datos son transmitidos de una manera simultanea. En cambio en una comunicasión serial, los bits de datos son transmitidos de forma secuencial y además de manera más lenta.

2.1.5 Adaptadores Gráficos

2.1.5.1 Introducción

Un *adaptador gráfico* (graphics adapter) es el hadware interno de una PC que le proporciona la capacidad para desplegar imágenes gráficas como una adición al texto. Las imágenes gráficas están compuestas por una serie de puntos (pixels) que pueden ser colocados en cualquier lugar en el área de display del monitor. Cuando el termino adaptador gráfico es usado, implica la disponibilidad tanto del modo texto como del modo gráfico en la operación de la PC.

Las capacidades del display gráfico de la IBM-PC y compatibles sean incrementado a través de los años. El primer adaptador gráfico de display, el adaptador gráfico de color el CGA (del inglés : Color Graphics Adapter), podría desplegar una pantalla de 320x200 pixel en cuatro colores o una pantalla de 640x200 en dos colores, contrastando con el actual standard gráfico de video el VGA (de Video Graphics Array), el cual puede soportar una resolución de pantalla de 640x480 en 16 colores o de 320x200 en 256 colores.

En el mundo de la IBM-PC y compatibles, existen ocho adaptadores de display standard disponibles :

a) MDA (Monochrome Display Adapter).

b) HGA (Hercules Graphics Adapter).

c) CGA (Color Display Adapter).

d) El adaptador de display en el IBM PC Jr.

e) EGA (Enhanced Graphics Adapter).

f) MCGA (Multi Color Graphics Array), el cual esta disponible en los modelos 25 y 30 de las computadoras IBM PS/2.

g) VGA (Video Graphics Array), el cual esta disponible en los modelos 50, 55, 60, 70 y 80 de las computadoras PS/2

h) La tarjeta adaptadora 8514/A y el monitor IBM 8514

Cabe hacer notar que todos los display standares en la lista anterior con excepción del adaptador Hercules, fueron desarrollados y promovidos por IBM corporation. Hay sin embargo muchas otras tarjetas de video no IBM disponibles en el mercado, algunas de los cuales tienen especificaciones más grandes que el 8514/A, pero posiblemente programas de aplicación realizados en el VGA standard no sean compatibles con estas y la aplicación no funcione.

En todo sistema de procesamiento de imágenes, una buena selección del adaptador gráfico 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 de información contenida en la imagen, además de una mejor presentación.

Capacidades Gráficas de varios Adaptadores de Video

Resolucion	Colores	CGA	PCJr	EGA	MCGA	VGA
1. 320 x 200	4	*	*	*	*	*
2. 640 x 200	2	*	*	*	*	*
3. 160 x 200	16		*			
4. 320 x 200	16		*	*		*
5. 320 x 200	256				*	*
6. 640 x 200	4		*			
7. 640 x 200	16			*		*
8. 640 x 350	2			*		*
9. 640 x 350	4			*		
10. 640 x 350	16			*		*
11. 640 x 480	2				*	*
12. 640 x 480	16					*
13. Paleta Configurable			*	*	*	*
14. Registros de color configurables					*	*

La pregunta en cuestión es: ¿Cual es la mínima funcionalidad de un adaptador gráfico requerida como aceptable para el despliegue de una imagen ? La figura 2.1.4.1 detalla las capacidades gráficas de muchos adaptadores de display, para ayudar a realizar esta determinación (Nota: El modo texto disponible en varios adaptadores de display no es importante como el modo gráfico para la aplicación de imágenes, por lo cual no es mostrado en la tabla).

Para contestar la pregunta anterior es necesario analizar el objetivo de lo que se desea desplegar. Por una parte para visualizar imágenes con múltiples niveles de escala de gris o a color, se requiere de paletas configurables y de registros de color, por otra para proveer el balance apropiado de color para el despliegue de imágenes realistas, los componentes RGB de los registros de color deberán ser colocados directamente.

Con los requerimientos anteriores y en base a la figura 2.1.4.1 podemos observar que aunque el EGA provee 16 colores simultáneos de una paleta de 64, este carece de los registros de color configurables lo cual hace de este de utilidad limitada para aplicaciones de imágenes. De la misma tabla podemos observar que el VGA cumple con todo los requisitos inicialmente expuestos por lo cual a continuación se presenta un estudio detallado de este adaptador gráfico.

2.1.5.2 El Adaptador Grafico VGA

El adaptador gráfico VGA (Video Graphics Array) fue introducido por IBM en 1987, y se construyo internamente en la motherboard de los modelos 50, 55, 60, 70 y 80 de la familia de computadoras Personal System/2 (PS/2); conservando la compatibilidad con el EGA pero añadiéndole importantes características entre las que destacan la lectura de los registros del adaptador, el incremento en la resolución y un modo en el cual se pueden tener 256 colores simultáneos.

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

El VGA cuenta con su propio BIOS, escrito en memoria ROM, lo cual le proporciona mayor compatibilidad con procesadores de textos, manejadores de base de datos, hojas de calculo, diseño asistido por computadora, etc. El VGA BIOS ROM contiene cinco distintos tamaños de caracteres: 8x8, 8x14, 8x16, 9x14 y 9x16.

2.1.5.2.1 Memoria de Display

Esta memoria es mapeada directamente dentro de la dirección de memoria de la PC, permitiendo con esto una rápida transferencia de datos entre la memoria de la PC y la memoria del display o entre la memoria del display y la memoria de la PC.

La memoria del display realiza la comunicación con el procesador de la PC a través de un bus de datos de 8 bits, independientemente del ancho del bus de datos de la PC. Esta tiene una capacidad de hasta 256 Kbytes, además puede ser organizada en una gran variedad de resoluciones dependiendo del modo en que se encuentre el display. La figura 2.1.5.2.1 ilustra un mapa de memoria para una computadora personal y compatibles.

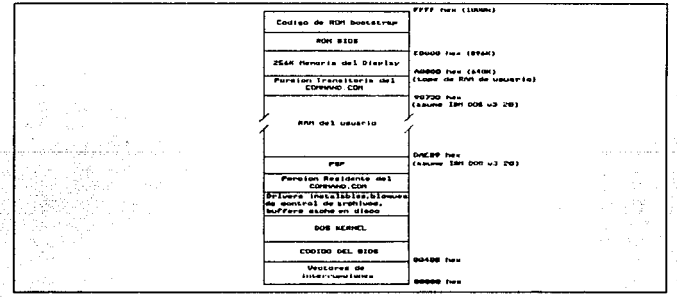


Fig. 2.1.5.2.1 Mapa de Memoria para PC IBM y Compatibles

2.1.5.2.2 Registros de Control

Ciertamente el aspecto más complicado en el adaptador VGA es comprender el control de sus registros que son divididos en cinco grupos (fig. 2.1.5.2.2). El tamaño de cada

REGISTRO		DIRECCION	
		ESCRITURA	LECTURA
GENERAL			
Miscelánea de Salida		3C2	3CC
Características de Control		3DA/3BA	3CA
Entrada de Estado 0			3C2
Entrada de Estado 1			3DA/3BA
SECUENCIADOR			
Dirección		3C4	3C4
Reset	0	3C5	3C5
Modo Reloj	1	3C5	3C5
Mapa de Máscaras	2	3C5	3C5
Selección de Mapa de Caracter	3	3C5	3C5
Modo de Memoria	4	3C5	3C5
CRTC			
Dirección		3D4/384	3D4/384
Total Horizontal	0	3D5/385	3D5/385
Fin de Display Horizontal	1	3D5/385	3D5/385
Principio de Blanco Horizontal	2	3D5/385	3D5/385
Fin de Blanco Horizontal	3	3D5/385	3D5/385
Principio de Retraso Horizontal	4	3D5/385	3D5/385
Fin de Retraso Horizontal	5	3D5/385	3D5/385
Total Vertical	6	3D5/385	3D5/385
Qverflow	7		3D5/385
Renglon Rastreado	8	3D5/385	3D5/385
Maxima Linea Rastreada	9	3D5/385	3D5/385
Principio de Cursor	A	3D5/385	3D5/385
Fin de Cursor	B		3D5/385
Principio de Dirección Alta	C	3D5/385	3D5/385
Principio de Dirección Baja	D	3D5/385	3D5/385
Ubicacion Alta del Cursor	E	3D4/384	3D4/384

REGISTRO		DIRECCION	
		ESCRITURA	LECTURA
CRTC (continuación ...)			
Ubicación baja del Cursor	F	3D5/385	3D5/385
Principio de Retraso Vertical	10	3D5/385	3D5/385
Retraso Vertical Bajo	11	3D5/385	3D5/385
Fin de Display Vertical	12	3D5/385	3D5/385
Desplazamiento	13	3D5/385	3D5/385
Localidad de Underline	14	3D5/385	3D5/385
Principio de Blanco Vertical	15	3D5/385	3D5/385
Fin de Blanco Vertical	16	3D5/385	3D5/385
Modo de Control	17	3D5/385	3D5/385
Compara Lineas	18	3D5/385	3D5/385
CONTROLADOR GRAFICO			
Dirección Gráfica		3CE	3CE
Set/Reset	0	3CF	3CF
Habilita Set/Reset	1	3CF	3CF
Compara Color	2	3CF	3CF
Rotación de Dato	3	3CF	3CF
Selección de Mapa de Lectura	4	3CF	3CE
Modo	5	3CF	3CF
Miscelánea	6	3CF	3CF
No Importa Color	7	3CF	3CF
Máscara de Bit	8	3CF	3CF
ATRIBUTOS			
Dirección		3C0	3C1
Paleta	0F	3C0	3C1
Modo Control	10	3C0	3C1
Color Sobre Rastreado	11	3C0	3C1
Habilita Plano de Color	12	3C0	3C1
Pixel Horizontal	13	3C0	3C1
Selección de Color	14	3C0	3C1

NOTA: Cuando no es dada una dirección de lectura, el registro es escrito solamente. Así mismo para la dirección de escritura. Cuando las dos direcciones son proporcionadas, la primera es para una operación monocromática y la segunda para una operación a color.

registro es de un byte y son accesados a través de direcciones de puertos de Entrada/Salida.

2.1.5.2.3 Modos de Display

El adaptador VGA puede ser configurado 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 por default y la dirección de inicio de la memoria del display. Todos estos parámetros se ilustran en la figura 2.1.5.2.3.

MODO BIOS	MODO/TIPO	RESOLUCION	DIMENSION/COLOR	DIR. INICIO	TAMANO FONT
0h	color/alfanum	320 x 200	40 x 25/ BW	b800 : 0h	8 x 8
1h	color/alfanum	320 x 200	40 x 25/ 16	b800 : 0h	8 x 8
2h	color/alfanum	640 x 200	80 x 25/ BW	b800 : 0h	8 x 8
3h	color/alfanum	640 x 200	80 x 25/ 16	b800 : 0h	8 x 8
0*	color/alfanum	320 x 350	40 x 25/ BW	b800 : 0h	8 x 14
1*	color/alfanum	320 x 350	40 x 25/ 16	b800 : 0h	8 x 14
2*	color/alfanum	640 x 350	80 x 25/ BW	b800 : 0h	8 x 14
3*	color/alfanum	640 x 350	80 x 25/ 16	b800 : 0h	8 x 16
0*	color/alfanum	360 x 400	40 x 25/ BW	b000 : 0h	9 x 16
1*	color/alfanum	360 x 400	40 x 25/ 16	b800 : 0h	9 x 16
2*	color/alfanum	720 x 400	80 x 25/ BW	b000 : 0h	9 x 16
3*	color/alfanum	720 x 400	80 x 25/ 16	b800 : 0h	9 x 16
4h	color/grafico	320 x 200	40 x 25/ 4	b800 : 0h	grafico
5h	color/grafico	320 x 200	40 x 25/ BW	b800 : 0h	grafico
6h	color/grafico	640 x 200	80 x 25/ BW	b800 : 0h	grafico
7h	mono/alfanum	720 x 350	80 x 25/ BW	b000 : 0h	9 x 14
0dh	color/grafico	320 x 200	40 x 25/ 16	a000 : 0h	grafico
0eh	color/grafico	640 x 200	80 x 25/ 16	a000 : 0h	grafico
0fh	mono/grafico	640 x 350	80 x 25/ BW	a000 : 0h	grafico
10h	color/grafico	640 x 350	80 x 25/ 16	a000 : 0h	grafico
11h	color/grafico	640 x 480	80 x 30/ BW	a000 : 0h	grafico
12h	color/grafico	640 x 480	80 x 30/ 16	a000 : 0h	grafico
13h	color/grafico	320 x 200	80 x 25/ 256	a000 : 0h	grafico

Fig. 2.1.5.2.3 Modos de Video

Cada uno de estos formatos de display necesita una cantidad de memoria en bytes diferente, dependiendo del tipo de resolución y el número de bits por pixel que requiera.Una tabla que ilustra mejor lo anterior se muestra en la figura 2.1.5.2.3'.

2.1.5.2.4 Resolución de Color

Actualmente dentro del adaptador VGA standard se cuenta con un único modo que permite desplegar 256 colores de 262,144 paletas en un monitor a color, este es numerado con el modo 13 hex. Los 256 registros de color son accesados directamente y cada pixel

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

Fig. 2.1.5.2.3' Cantidad de Memoria del Display

esta representado por 8 bits, lo cual permite 256 colores simultáneos. Este byte puede residir en uno de los cuatro planos de bits. El conjunto de caracteres por default en este modo es el conjunto de un carácter de 8x8. Esto permite 40 caracteres horizontales por 25 renglones.

El modo 13 hex. permite una resolución de display de 320 pixels por 200 renglones. Esto es equivalente a 64,000 pixels por página, puesto que hay un pixel por byte, son necesarios 64,000 bytes por página. Cuatro planos de bits son usados para almacenar los datos, así solamente 16,000 bytes son requeridos por página. Los bytes consecutivos de los datos desde la memoria de la PC son cargados dentro de los planos de bit 0,1,2 y 3 como muestra la figura 2.1.5.2.4. Cada byte de datos contiene un pixel. El primer byte, contiene el pixels 0 que reside en plano de bit 0, el segundo en el plano de bit 1, el tercero

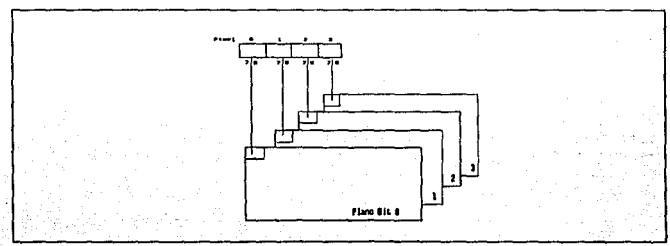


Fig. 2.1.5.2.4a Mapeo de Datos dentro de la Mem. del Display

en el plano de bits 2 y el cuarto en el plano de bits 3. Un quinto byte residirá siguiendo al pixel 0 en el plano de bit 0.

El termino *paleta* es definido en el diccionario como el conjunto de colores sobre una determinada tabla. En términos gráficos de computación, la paleta es la colección de colores disponibles para ser desplegados simultáneamente sobre un monitor a color. Los distintos adaptadores gráficos usan diferentes mecanismos para colocar colores sobre un monitor.

La figura 2.1.5.2.4b muestra la configuración de la paleta para un EGA. el cual es discutido aquí, para que las capacidades del VGA puedan ser contrastadas con este. En el EGA, los cuatro bits de almacenamiento de datos en la memoria de video forman un índice dentro de la estructura de datos de la paleta. Los cuatro bits de datos de video dan como resultado 16 valores posibles de índices de color, dentro del rango de 0 a 15. Todas las 16 entradas en la estructura de la paleta son numeros de color de 6 bits. El color especificado por el numero de color esta fijado y no puede ser alterado. Sin embargo, cada entrada en la paleta "puede" ser alterada por cualquiera de uno de los 64 posibles numeros de color. Esto da como resultado la capacidad de desplegar 16 colores simultáneos de una paleta de 64 colores posibles. Cabe hacer notar que el monitor usado por el display EGA es digital. Los seis bits de información digital de color : r'g'b'RGB se maneja directamente fuera de el adaptador del display para las equivalentes entradas

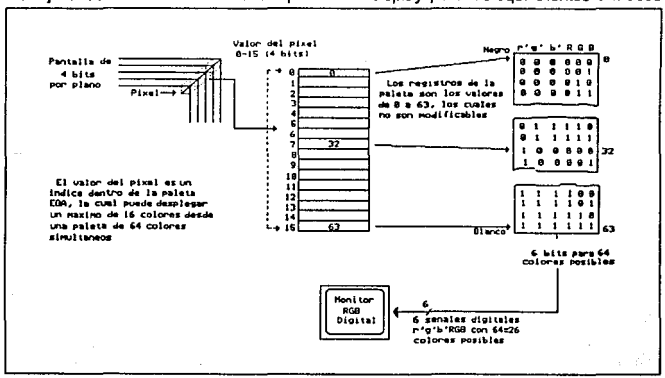


Fig. 2.1.5.2.4b Configuración de la Paleta EGA

sobre el monitor. De la teoría básica digital, los 6 bits de color digital representan 64 o 26 combinaciones totales o colores disponibles.

En la inicialización del EGA, la paleta es llenada con colores que son compatibles con el CGA. Esto fue hecho para asegurar compatibilidad con el software escrito para el CGA.

Para colocar un pixel en la luz magenta sobre la pantalla del EGA, el valor decimal 13 deberá ser metido en la apropiada dirección dentro de la memoria de video. En la actualidad la función gráfica de Turbo "C" putpixel deberá ser llamada y pasarle el valor 13 como respuesta al color.

Como los cambios son realizados en el contenido de la paleta, todos los pixels mapeados al índice que ha sido cambiado, automáticamente asumirán el nuevo color. Una vez que la pantalla es pintada con una imagen, efectos extraños pueden ser producidos por alterar los numeros de color almacenados en la paleta. Alguno de estos efectos son completamente notables.

El adaptador gráfico VGA usa dos métodos para el manejo de paletas. El método depende del modo gráfico que se este utilizando. Todos los modos gráficos con excepción del modo 13H (13 hex), modo de 320x200 a 256 colores usan básicamente el mismo

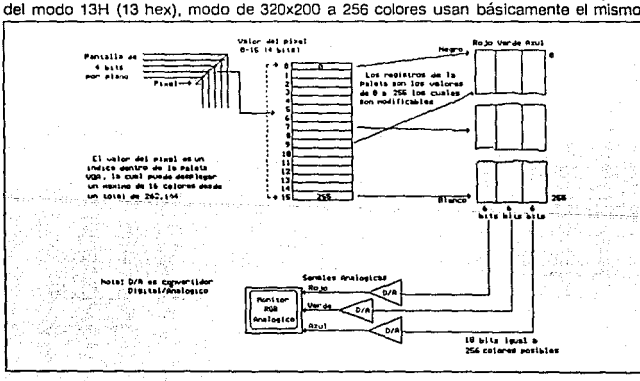


Fig. 2.1.5.2.4c Configuración de la Paleta VGA

mecanismo como el del EGA. El mecanismo de paleta VGA compatible con EGA es mostrado en la figura 2.1.5.2.4c.

La diferencia en el mecanismo VGA es que las entradas a las paletas contienen apuntadores a uno de los 256 registros de color en lugar que el conjunto de 63 numeros de color usados por el EGA. La gran diferencia entre los dos adaptadores gráficos es que los componentes individuales (RGB) rojo (Red), verde (Green), y azul(Blue) de los registros de color pueden ser configurados en el adaptador VGA, mientras que estos están fijos en el EGA. Dado que cada uno de los componentes de color, rojo, verde y azul utilizan 6 bits de información, un total de 218 posibles colores diferentes están disponibles. Esto permite trabajar con 262,144 posibles colores diferentes, 16 de los cuales pueden ser desplegados simultáneamente.

Para desplegar estos distintos colores un monitor RGB analógico deberá ser usado. Como se muestra en la figura, tres convertidores Digital-Analógicos (D/A) son necesarios para convertir los 18 bits de información de color digital a señales analógicas requeridas para manejar el monitor.

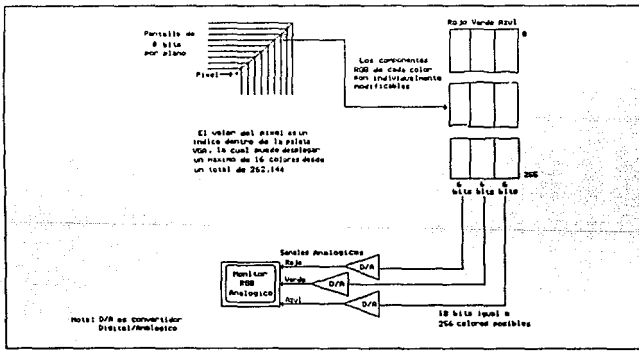


Fig. 2.1.5.2.4d Configuración de la Paleta VGA en modo 13H

El modo del VGA 13H es un modo especial de color. En este modo, una baja resolución de la imagen (320x200) con un total de 256 colores diferentes pueden ser desplegados simultáneamente. La figura 2.1.5.2.4d ilustra el mecanismo de la paleta usado para este modo especial.

Actualmente, podría ser más preciso decir la falta del mecanismo de paleta usado para este modo especial. Con el modo 13H, la memoria del display es segmentada en 8

diferentes planos de bits en comparación a los cuatro utilizados por todos los otros modos. Tener 8 planos de bit significan que el dato de un pixel de video puede tomar valores de 0 a 255. En este modo el mecanismo de la paleta es evitado completamente y el valor del dato de video directamente indexa a un registro de color. Un pixel de video de valor 10 toma el color definido por el valor RGB de registro de color 10 y así sucesivamente. Usando este modo gráfico, una imagen puede ser desplegada con 256 colores diferentes de un total de 262,144 colores posibles. Esto es cualquiera de uno de los 256 colores puede ser uno de los 262,144 colores disponibles.

En el tiempo de inicialización para el modo 13H, los registros de color son cargados como sigue:

a) Los primeros 16 registros de color son cargados con valores que corresponden a los colores del CGA y EGA mostrados en la figura 2.1.5.2.4e.

b) Los segundos 16 registros de color son cargados con tonos de gris separados uniformemente.

INDICE PALETA	COLOR	REGISTRO PALETA
0	Negro	0
1	Azul	1
2	Verde	2
3	Cyan	3
4	Rojo	4
5	Magenta	5
6	Cafe	20
7	Gris Claro	7
8	Gris Oscuro	56
9	Azul Claro	57
10	Verde Claro	58
11	Cyan Claro	59
12	Rojo Claro	60
13	Magenta Claro	61
14	Amarillo	62
15	Blanco	63

Fig. 2.1.5.2.4e Paleta Default del VGA

c) Los 216 registros de color restantes contienen valores basados sobre un patrón de tintes(hue), saturación e intensidad; adaptados para proveer un utilizable conjunto genérico de color que abarca un ancho rango de valores de color.

2.2 Software

2.2.1 Lenguaje de Programación

En todo desarrollo de un sistema, la elección de un lenguaje de programación es un factor importante; ya que de esto depende la explotación adecuada del hardware elegido así como la rapidez con que se pueda desarrollar el mismo. Para el desarrollo del presente sistema el lenguaje C fue elegido por las características que a continuación se describen, sin embargo, una rutina fue necesario implementarla en lenguaje emsamblador, ya que el compilador elegido (Turbo C + +) no tenía un BGI implementado para el manejo de los modos a 256 colores que son necesarios para la explotación de imágenes a colores reales (recientemente Borland implemento este BGI para Turbo C, Turbo Pascal y Turbo Prolog; pero desafortunadamente no se pudo conseguir para el desarrollo del presente trabajo). Cabe aclarar que esta rutina podría haberse implementado en lenguaje C, pero por la necesidad de rapidez en la misma fue necesario desarrollarla en lenguaje ensamblador.

El lenguaje C fue desarrollado para optimizar la comunicación entre un sistema operativo de computadora y su lenguaje de máquina interna, facilitando la programación de tareas y haciéndolas más directas. C es un lenguaje de programación de propósito general, relativamente estructurado, de bajo nivel (es decir, los objetos que maneja son los mismos que usa el procesador); se caracteriza por su condición y poseer una amplia gama de operadores. La generalidad del lenguaje, así como su carencia de restricciones, lo hacen más conveniente que otros lenguajes de alto nivel para ciertas tareas. Esto ha hecho que sea ampliamente usado para el desarrollo de sistemas operativos (Unix y sus derivados), compiladores y para utilerías de muy diversas características.

A pesar de la tipificación de C como "Lenguaje de Sistemas", sus usos se han extendido mucho, gracias al soporte de muchas empresas que han desarrollado un gran número de librerías de funciones para su aplicación a diferentes áreas. También su popularidad ha ido creciendo en respuesta a las facilidades ofrecidas por los compiladores actuales. Estos dos hechos han propiciado la utilización de C no sólo en el desarrollo de sistemas operativos, sino también en aplicaciones administrativas, juegos, y en general para toda clase de programas tales como dBASE, Norton Utilites, entre otros que fueron desarrollados en C.

Una característica muy importante de C es la utilización de direcciones a través de apuntadores. C no posee procedimientos tipo Pascal, sino que todo es desarrollado por medio de funciones. Sin embargo, existen funciones que no regresan ningún valor, resultando parecidas a los procedimientos en Pascal, la diferencia radica en que no es posible pasar argumentos por referencia, todo argumento es pasado por valor, excepto cuando se trata de una estructura, en cuyo caso se pasa un apuntador a esa estructura. Esto significa que cuando se requiera que sea modificado un valor pasado a una función es necesario pasarle su dirección, simulando un proceso de referencia, de manera que la función pueda actuar sobre el objeto original.

Por otra parte el nuevo estándar ANSI C aumentará la capacidad de portabilidad de los programas escritos en lenguaje C. Los escritores de compiladores prometen con esto, que si el programa de aplicación desarrollado, esta dentro de las fronteras especificadas por el estándar, el programa compilará y correrá con éxito en cualquier computadora que soporte el compilador ANSI. Desde luego si se supone que la aplicación correrá en DOS la portabilidad no será un requisito indispensable, sin embargo si se decide llevar el software a un ambiente UNIX o a una minicomputadora DEC o IBM el ANSI C deberá ser utilizado.

Turbo C++, de Borland International Inc., es uno de los compiladores de C más populares en la actualidad, no sólo por ofrecer un poderoso y sofisticado ambiente de desarrollo, sino por el poder que confiere al lenguaje el conjunto de adiciones y modificaciones realizadas por esta empresa. Turbo C++ ofrece dos versiones del compilador. La primera contiene sólo al compilador, y la segunda el ambiente de desarrollo. Se maneja a través de menús e incluye un editor, ayuda en línea, debug integrado, control de programas, así como otras opciones útiles en el proceso de desarrollo.

Puesto que C es un lenguaje modular, el compilador debe soportar compilación separada, esto es compilación de programas y rutinas que posteriormente serán integradas en un solo sistema. Turbo C++ implementa este tipo de compilación y facilidades tipo "Make-Build" (compilar solo lo necesario o todo) tanto en el modo línea como en el ambiente integrado en este ultimo utilizando los archivos *.prj* . Esto posibilita la creación de sistemas grandes, compuestos por una gran cantidad de módulos, que de otra manera serían imposibles de realizar.

El lenguaje C tiene la "mala fama" de ser un lenguaje difícil, por su brevedad en las instrucciones y la gran libertad en la programación, lo que hace que programadores inexpertos estén sujetos a cometer una serie de errores en los que no incurrirían en otros lenguajes más estrictos,como Pascal o BASIC. Sin embargo Compiladores modernos como Turbo C++ ofrecen una amplia gama de opciones y facilidades que ayudan a minimizar en gran forma la posibilidad de errores y contribuyen a facilitar la depuración de programas. Un ejemplo de esto son los prototipos de funciones, que permiten usar una sintaxis más moderna en los llamados a funciones, equivaliendo a declarativas tipo Pascal, lo que permite al compilador revisar el número y tipo de parámetros pasados a una función.

Turbo C++ que actualmente se comercializa en su versión 2.0, amplia significativamente el lenguaje estándar y ofrece facilidades y opciones que convierten a C en un lenguaje fácil de aprender y usar. Además, la velocidad de compilación, así como la optimización del código generado, tanto en tamaño como en rapidez, hacen de este compilador una opción digna de tomar en cuenta, no sólo para tener a C como lenguaje principal, sino para usarlo en combinación con otros lenguajes de más alto nivel.

Entre las características más importantes de Turbo C++ 2.0 se encuentra:

- Ambiente integrado de desarrollo.

- Compilación por separado (el programa se crea durante la liga).
- Compilación condicional (posibilidad de no compilar ciertas partes).
- Recursividad (Posibilidad de una función de invocarse a sí misma).
- Una completa librería Gráfica.
- Debug y Ayuda en línea en el ambiente de desarrollo.
- Manejo de Interrupciones.
- Facilidades para creación de programas residentes (TSR's).
- Optimización del código generado.
- Supresión de datos y código no utilizado (Liga inteligente).
- Make-Build (para control de compilación modular)

En lo que se refiere al lenguaje ensamblador a pesar de ser una herramienta muy poderosa, la mayoría de las veces se codifica en lenguaje ensamblador sólo cuando es absolutamente necesario porque tal codificación es difícil y ocasiona demasiados problemas. Sin embargo existen dos situaciones en las que la codificación en lenguaje ensamblador tiene sentido. La primera situación es cuando no hay absolutamente ningún otro camino para hacerlo la segunda es cuando se debe de reducir el tiempo de ejecución del programa.

Entre los ensambladores más populares que generan código objeto para la familia 8086/8088 se encuentran el Macro Assembler/2 de IBM, el ensamblador TASM de Borland y MASM de Microsoft, este último fue el que se utilizó para ensamblar las rutinas y poder generar su código objeto para posteriormente ligarlo con las rutinas en lenguaje C. Las características del formato que el programa en ensamblador debe tener para poder realizar la interfaz con programas en C, es discutido a continuación.

2.2.2 Modelos de Memoria

Antes de estudiar como se realiza la interfase entre programas escritos en lenguaje ensamblador y programas en lenguaje C, los modelos de memoria deberán ser discutidos porque ellos tienen un impacto significativo. Cabe recordar que el diseño del chip del CPU intel 8086 obliga que el manejo de memoria sea realizado en segmentos y cada segmento deberá direccionar un máximo de 65,536 bytes (64K). Esto es la base del concepto de los modelos de memoria, estos simplemente especifican como los segmentos son manejados.

Este concepto es poco conocido en otros lenguajes de programación. Escencialmente se refiere a cómo quedarán distribuidos los datos (segmento de datos), las instrucciones (segmento de código) en el programa compilado y al tamaño final del sistema, entre otras cosas. La figura 2.2.2, muestra la distribución de la memoria cuando un programa en C es ejecutado; al inicio de la frontera del segmento se encuentran todas las instrucciones

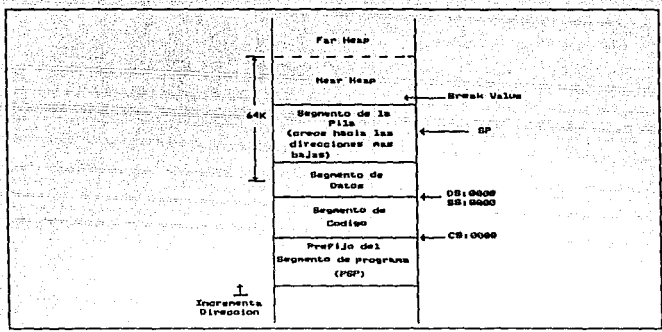


Fig. 2.2.2 Mapa de Memoria de un Programa en C para IBM-PC

del programa, el cual es llamado segmento de código (code segment), cuya dirección se encuentra el registro de segmento CS y el apuntador a la instrucción (IP) el cual dentro del microprocesador contiene el offset para obtener en combinación con el CS la instrucción a ser ejecutada.

El siguiente segmento encontrado después del código del programa, es aquel donde residen los datos del mismo. Este es llamado el segmento de datos (data segment) y su dirección esta almacenada en el registro DS. Después de los datos se encuentra un bloque fijo de memoria (4K por default, pero pude ser modificado), el cual es usado para pasar argumentos durante la llamada a funciones y para almacenar variables locales. Este bloque es conocido como segmento de stack (stack segment) y utiliza un buffer de tipo LIFO. Se utiliza un puntero al stack SP para poder accesar la actual localidad del stack, la dirección del stack se encuentra en el registro de segmento SS. En computadoras basadas en el 8086, el stack siempre crece hacia abajo. Esto significa que cuando colocamos un elemento en el stack, el puntero al stack es primero decrementado y entonces el elemento es salvado en la dirección SS:SP.

Finalmente la primera dirección arriba del segmento de datos de un programa es llamada el "valor corte" (break value) del proceso y denota el inicio del bloque de asignación de memoria para el proceso.

Cuando el compilador utiliza el offset para direccionar los datos, se dice que estos datos tienen direcciones near, de tal forma que se hace referencia a ellos como datos tipo near; todos estos datos tienen la misma dirección de segmento la cual esta almacenada en el

registro de segmento DS. Análogamente, cuando se utiliza explícitamente el segmento y el offset para direccionar un dato, estos son llamados datos tipo far. Los datos tipos far pueden estar localizados en cualquier lugar de la memoria y el compilador genera código que explícitamente carga un registro de segmento del 8086, tal como el ES para accesar estos datos.

Debido al esquema de direccionamiento segmento:offset en la PC el compilador de Turbo C++ provee a los usuarios la opción de seleccionar uno de los seis modelos diferentes de memoria cada uno ofreciendo rapidez en la ejecución contra el tamaño del programa.

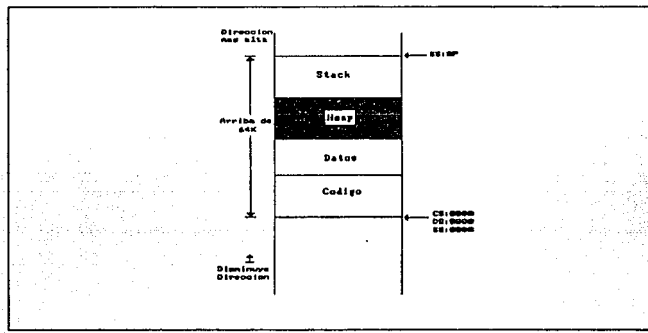


Fig. 2.2.2.1 Mapa de Memoria del Modelo Tiny

2.2.2.1 Modelo Tiny

El modelo tiny permite un único segmento de 64 kbytes tanto para datos como para código. Los cuatro segmentos de registros: CS, DS, SS y ES apuntan a la misma dirección de segmento, el mapa de memoria para este modelo se muestra en la figura 2.2.2.1. Este modelo es aconsejable para programas pequeños. Turbo C no permite llamadas a funciones gráficas en este modelo.

2.2.2.2 Modelo Small

Este modelo permite un solo segmento de 64 kbytes para datos y un solo segmento de 64 kbytes para código. Es el modelo que Turbo C adopta por defecto, debido a que para la mayoría de las aplicaciones es suficiente. Los punteros se codifican en 16 bits, el

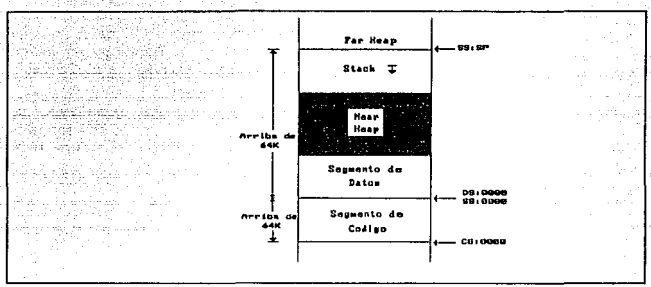


Fig. 2.2.2.2 Mapa de Memoria del Modelo Small

registro CS contiene la dirección del segmento de código del programa, mientras que DS, ES y SS están inicializadas al mismo valor y contienen la dirección de inicio del segmento de datos.En la figura 2.2.2.2 se muestra el mapa de memoria del modelo 'small'.

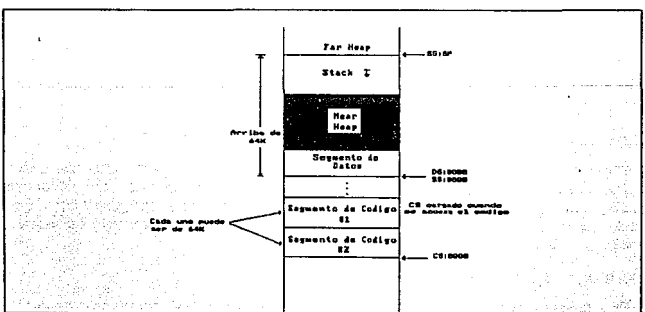


Fig. 2.2.2.3 Mapa de Memoria del Modelo Medio

2.2.2.3 Modelo Medium

En este modelo el código puede alcanzar hasta 1MB en múltiples segmentos de 64KB, pero el segmento de datos queda limitado a un bloque de 64KB; los datos fuera de este segmento pueden ser accesados por una llamada explícita mediante un puntero tipo far, por lo cual una dirección de 32 bits deberá ser especificada. De esta manera los punteros de código son de tipo far, mientras que los punteros de los datos son de tipo near. Este modelo es usual cuando se espera que la cantidad de datos sea limitada y el programa tienda ser grande. Ver figura 2.2.2.3 en donde se muestra el mapa de memoria para el modelo 'medio'.

2.2.2.4 Modelo Compact y Large

En estos modelos, el compilador genera direcciones explícitas de segmento y offset para todos los datos. De esta forma estos modelos permiten que los datos puedan extenderse hasta 1MB y los punteros se codifican en 32 bits, con la única restricción que cada item de dato no puede exceder de 64KB. El modelo compact permite un solo

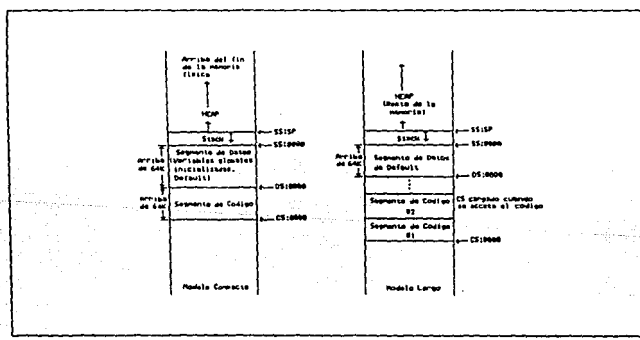


Fig. 2.2.2.4 Mapa de Memoria de Modelos Compacto y Largo

segmento de 64KB para el código, mientras que el modelo largo permite hasta un 1MB tanto en datos como en código sujeto a la condición de que cada item de dato se encuentre dentro de un solo segmento de 64KB.

Puesto que estos dos modelos permiten múltiples segmentos de datos, toda inicialización de variables globales están colocada dentro de un segmento cuya dirección

este en DS. Normalmente, el stack es también colocado en un segmento separado.Fig. 2.2.2.4.

2.2.2.5 Modelo Huge

Este modelo elimina la limitante de que cada item de dato se encuentre dentro de un bloque de 64KB, de esta forma un array puede exceder los 64KB. El modelo huge es

idéntico al modelo large excepto en el cálculo de las direcciones el cual es realizado considerando cada dirección como un valor total de 32 bits almacenado en un entero largo sin signo, con el inicio del segmento en los bits de más alto orden. La dirección es mantenida en una forma *normalizada* convirtiendo el par segmento:offset a una dirección física de 20 bits tomando los 16 bits de más alto orden como el segmento y los cuatro bits más bajos como el offset. Por ejemplo, para convertir un apuntador no normalizado de valor 3412:DF03 (que es la forma normalizada de Intel segmento:offset) a la forma normalizada :

a.- Realizar un corrimiento a la izquierda (introduciendo un cero) del valor hexadecimal del segmento 3412, obteniendo 34120.

b.- Sumar el valor del offset DF03 a 34120, obteniendo 42023 hex.

c.- Los últimos 4 bits son el valor del offset, el cual es 3.

d.- Los 16 bits más significativos son el valor del segmento 4202.

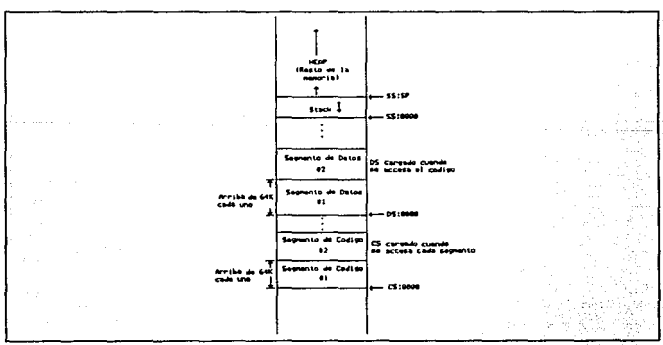


Fig. 2.2.2.5 Mapa de Memoria del Modelo Huge

De esta forma el valor del apuntador normalizado de 3412:DF03 es 4202:0003. Esta normalización garantiza que cada apuntador huge hace referencia a una única localización física de memoria, pero a cambio se tiene un overhead al normalizar los apuntadores durante la aritmética de los mismos. En la fig. 2.2.2.5 se muestra el mapa de memoria para el modelo huge.

Los seis modelos de memoria discutidos son suficientes para muchas aplicaciones, pero existen ocasiones en que se desea direccionar un dato de tipo far (por ejemplo accesar la memoria del video) en un modelo que no tiene esta capacidad. Para lo cual el compilador de Turbo C++ permite igualar datos a un tipo no disponible en el modelo seleccionado. De esta forma en los modelos compact, large y huge se puede utilizar la palabra reservada *near* para significar que ciertos datos residen en el segmento de datos por default (uno cuya dirección de segmento este en el registro DS). Similarmente, en los modelos de memoria small y medium se puede usar la palabra reservada *far* para indicar al compilador el uso explícito de una dirección segmento:offset al genera código para accesar un especifico dato (el segmento ES es usado para colocar la dirección del segmento de tal dato). Un tercera palabra reservada *huge* permite utilizar un dato de tipo huge (dato cuyo tamaño excede de 64KB) en cualquier modelo de memoria.

2.2.3 Interfase entre Lenguaje Ensamblador y Lenguaje 'C'

Existen dos técnicas básicas para compartir información.

a) Referenciar a los datos como datos globales

b) Paso de parámetros utilizando el stack.

Los datos globales es el camino más directo para compartir datos entre código de lenguaje C y ensamblador. La variable deberá ser declarada en cualquiera de los dos programas (en el de C o en el de ensamblador) y deberá ser referenciada como una variable externa en cualquier modulo de código que necesite acceder a esta. El proceso de ligado es el encargado de combinar las instrucciones de lenguaje ensamblador y del lenguaje C dentro de un programa ejecutable, y es el que resuelve la referencia a la variable externamente declarada.

Por ejemplo si una variable de tipo integer, llamada "varexterna", es declarada en el código del lenguaje ensamblador y más tarde requerida en el código C, la declaración en el segmento de datos del código ensamblador podría ser como sigue:

```
PUBLIC _varexterna

_varexterna   DW  0  ;Una variable entera requiere una palabra.
                     ;El valor inicial de varexterna es cero.
```

Para utilizar "varexterna" en C, se deberá utilizar una declaración externa tal como :

```
extern  int varexterna;
```

Recíprocamente, "varexterna" podría ser declarada en C :

```
int varexterna = 0;
```

Y referenciada en lenguaje ensamblador como :

```
EXTERN _varexterna:WORD
```

Este método de intercambio de datos no es utilizado por la desventaja que ofrecen las variables globales en un sistema modular, el paso de parámetros, discutido a continuación, fue usado en su lugar.

El paso de parámetros es el segundo método de intercambio de datos entre código ensamblador y código C. Los parámetros pueden ser pasados desde código C a una función (subrutina) escrita en lenguaje ensamblador. Recíprocamente, las funciones de lenguaje ensamblador pueden pasar un resultado al código C. Casi todas las implementaciones de C usan el stack para el paso de parámetros a las funciones. Una vez que el formato del stack es entendido, es fácil recuperar cualquier parámetro pasado desde código C a lenguaje ensamblador.

Para accesar información pasada a través del satck, las rutinas utilizan el registro BP(el puntero base). Este registro es usado como un segundo apuntador al stack principalmente para el acceso de datos sobre el stack relativo a una localización dada.

El camino especifico para usar el registro BP en gran parte depende del tipo de subrutina que esta siendo llamada. Analicemos el contenido del stack cuando la rutina en lenguaje ensamblador es un procedimiento NEAR.

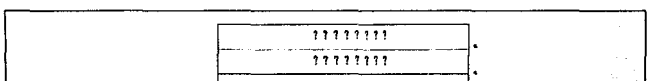


Fig. 2.2.3a Stack al Inicio de Procedimiento NEAR

Cuando la subrutina es invocada, cada parámetro individual es metido en el satck, seguido por la dirección de retorno del programa de llamada. En un procedimiento NEAR, esta dirección de retorno es técnicamente el offset del segmento, esto es el registro IP (Instruction Pointer) incrementado. Asi de esta manera en el inicio de la subrutina en lenguaje ensamblador a la cual 3 parámetros de 16-bits le están siendo pasados, el satck podría verse como lo muestra la figura 2.2.3a

Si la rutina esta llamando a un procedimiento FAR, el contenido del stack lo muestra la figura 2.2.3b

Se puede notar que el invocar un procedimiento FAR involucra meter una palabra adicional de dato. Esta palabra es el segmento del programa de llamada, técnicamente el contenido del registro de segmento de código (CS).

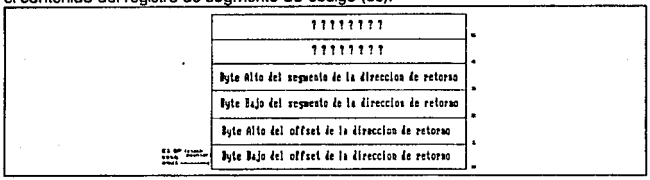


Fig. 2.2.3b Stack al Inicio de Procedimiento FAR

Para accesar la información, simplemente se coloca el registro BP y se utilizan los desplazamientos para obtener valores desde el satck. El siguiente segmento de código muestra el apropiado camino para hacer esto en un procedimiento NEAR, asumiendo que los parámetros han sido colocados sobre el satck por un compilador C y que cada uno ocupa 16-bits. (Nota: El segmento de código podría ser diferente desde otros lenguajes de programación) :

```
PARM_A          EQU   4
PARM_B          EQU   6
PARM_C          EQU   8
PRUEBA      PROC  NEAR
            PUSH  BP
            MOV   BP,SP
            MOV   AX,[BP]+PARM_A
            MOV   BX,[BP]+PARM_B
            MOV   CX,[BP]+PARM_C
; El resto del programa va aqui.
            POP   BP
            RET
TEST        ENDP
```

Note que los tres parámetros son llamados PARM_A, PARM_B y PARM_C en esta subrutina. Estos símbolos están igualados con un específico valor de desplazamiento al inicio del programa. Cuando la subrutina se ejecuta, el primer comando encontrado coloca el apuntador base (BP), asi que este puede ser usado con los valores de desplazamiento para accesar los parámetros.

Analizando el fragmento del programa anterior recordemos que al inicio el satck tiene la forma como se muestra en la figura 2.2.3a. En este punto la dirección de retorno ocupa los 2 bytes comenzando en el desplazamiento cero del stack, y los parámetros empiezan en el byte de desplazamiento 2. Sin embargo para salvar el contenido del registro BP, se

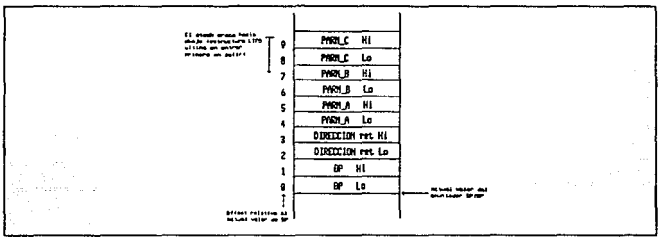


Fig. 2.2.3c Stack Completo de un Procedimiento NEAR

deberá de meter su contenido en el satck. Después de que se ha metido BP en el stack, el valor de BP ocupa los 2 bytes comenzando en el desplazamiento cero del stack, seguido por los 2 bytes de la dirección de retorno, y los parámetros entonces empiezan en el desplazamiento de valor 4.

La siguiente instrucción coloca BP igual al actual puntero del stack (SP). Puesto que se conoce que los parámetros empiezan en el desplazamiento 4, se puede accesar a los mismos a través de un direccionamiento indirecto, utilizando BP y el desplazamiento (el offset), la figura 2.2.3c muestra el stack en este punto.

Al terminar la subrutina, se deberá extraer BP del final del stack. Al hacer esto se deja limpio el stack y se restaura BP (BP deberá ser restaurado porque este es siempre usado en el programa que hizo la llamada).

Puesto que una llamada a un procedimiento FAR da como resultado una adicional palabra sobre el satck, el único cambio necesario para el anterior segmento de código, es el cambio en las igualdades (EQU) :

```
PARM_A      EQU     6
PARM_B      EQU     8
PARM_C      EQU     10
```

Este cambio compensa la adicional palabra colocada en el satck cuando el procedimiento FAR es llamado.

Una vez explicado como es realizado el paso de parámetros a través del stack, la siguiente guía es general para la interfaz de subrutinas en lenguaje ensamblador con el lenguaje C.

Primero, se deberá de dar un nombre específico al segmento de código de la subrutina en lenguaje ensamblador. Este nombre varía, dependiendo de cual compilador se este utilizando. Por ejemplo para Microsoft C, QuickC y Turbo C se requiere que el nombre del segmento sea _TEXT (para programas con modelos de memoria tiny, small o compact) o que el nombre del segmento lleve el sufijo (después del nombre) _TEXT (para los otros modelos de memoria).

Segundo, el compilador C podría requerir nombres específicos para los segmentos de datos (si los datos están siendo referenciado fuera del segmento de código). Microsoft C, QuickC y Turbo C requieren que el segmento sea llamado _DATA. Se deberán de utilizar estos nombres de segmentos, los cuales son específicos a C, para que sean apropiadamente ligados y poder ejecutar los programas.

Tercero, se deberá de tener en cuenta como las variables son pasadas a las subrutinas en lenguaje ensamblador a través del stack como se explico anteriormente. Por ejemplo en la sintaxis de la llamada a la función :

función_ejemplo(arg1,arg2,arg3,...,argn);

Los valores de cada argumento son introducidos en el stack en orden inverso. Asi el argumento n (argn) es introducido primero y el argumento 1 es introducido al final. Un valor actual o un apuntador a una variable, puede ser pasada. Aunque muchos valores y apuntadores son pasados como elementos del stack con una longitud de una palabra, los elementos de tipo largo de datos tal como long o unsigned long-tipo_variable requieren 32 bits (2 palabras) de espacio.

Si el modelo de memoria que esta siendo usado es compact, large o huge o si el item del dato exceden el segmento, los apuntadores de los datos también requieren 32 bits de espacio. Las variables de tipo float usan 64 bits (4 palabras) de espacio del stack. Recordar esta diferencia de uso del espacio del stack de acuerdo al tipo de variables; fallar al hacer esto puede producir resultados impredecibles.

Cuarto, en el archivo fuente de lenguaje ensambaldor, las rutinas en lenguaje ensamblador a ser llamadas desde C deberán empezar (para Microsoft C, QuickC o Turbo C) con un underline. Sin embargo, el underline no es incluido cuando las rutinas son invocadas en C. La opción de turbo C para esto puede ser colocada a su condición de default ON para obtener la máxima compatibilidad en el código mientras se prueba, pero puede ser colocada a OFF si se desea eliminar la necesidad para el prefijo underline.

Quinto, recordar salvar cualquier registro de propósito especial (tal como CS, DS, SS, BP, SI y DI) que la subrutina en lenguaje ensamblador prodria cambiar. Fallar al salvar las mismas puede tener consecuencias impredecibles cuando el control es regresado al programa en C. Esto es especialmente importante con Turbo C porque este (intrinsecamente) hace uso automático de las variables de registro . No es necesario salvar el

contenido de AX, BX, CX o DX porque estos registros son considerados de tipo volátil por C; esto es el lenguaje asume que se podrían cambiar los mismos y aun cuando se requieran estos para regresar algún valor.

Capítulo 3

Formatos Gráficos

3.1 Introducción

Las imágenes digitalizadas que son almacenadas en una forma que puedan ser manipuladas por otras aplicaciones, son llamadas imágenes con *formato gráfico*. Para que esto suceda es necesario que las mismas se almacenen en una forma 'estándar' que otras aplicaciones puedan interpretar y usar.

La creación de estandares gráficos constituyen una excelente idea, de tal manera que proveen un camino uniforme en la llamada de funciones (desarrollo de protocolos estandares) y en la respuesta a comandos. Así mismo estos estandares permiten al programador escribir aplicaciones sin tener que preocuparse en que tipo de hardware tiene el usuario. Otros estandares están dirigidos al almacenamiento de imágenes para que otros programas puedan hacer uso de los gráficos creados por otras aplicaciones. También permiten la transferencia de imágenes entre PC's con diferentes configuraciones o diferentes sistemas operativos, tales como MS-DOS y OS/2.

3.2 Clasificacion de Formatos Graficos

Los programas gráficos pueden ser clasificados de acuerdo a como sus datos son almacenados y desplegados. A esto se le llama FORMATOS DE ARCHIVOS GRAFICOS, los cuales podemos dividir en dos categorías :

a) Raster format y

b) Vector format.

3.2.1 Formato de Rastreo

El formato 'raster' esta compuesto de una serie de elementos de picture, o pixels, que cubren una área completa de display. Los Displays raster son usualmente generados por un barrido periódico de un rayo de electrones sobre la superficie de imagen con un patrón predeterminado (por ejemplo una video-cámara). Los pixels que forman una imagen rastreada no necesariamente se relacionan entre ellos. Estas imágenes son frecuentemente usadas para presentaciones gráficas, donde las consideraciones artísticas y la calidad de la imagen son importantes.

Las principales ventajas del formato de rastreo, son :

a. Es fácil la salida de datos desde un dispositivo de entrada de rastreo (como un video digitalizador o scanner) sobre un dispositivo de salida de rastreo como un monitor de computadora o una impresora gráfica.

b. El display de rastreo de datos es usualmente mas rápido que el display de datos vectorizados, porque los dispositivos de rastreo hacen el display y por lo tanto la conversión del vector-rastreo no tiene que ser ejecutada.

3.2.2 Formato de Vector

El formato de vector, por otra parte, involucra el uso de segmentos de línea directos en vez de pixels que construyen una imagen. Una imagen de vector esta hecha de formas que se componen de segmentos de líneas.

La conexión y jerarquía son la base principal de las imágenes de vector. Con datos de imágenes de formato-vector, es fácil determinar que componentes de segmentos de línea maquillan cualquier objeto. Las imágenes de vector son usadas principalmente en aplicaciones de CAD donde la precisión de escala y la relación de elemento son importantes. AUTOCAD es un programa ejemplo que utiliza 'vector graphics'.

3.3 Tipos de Archivos

Los programas de manipulación de imágenes, programas de control de scanner, y paquetes de desktop publishing, son todos programas de aplicación cada uno de los cuales utiliza un conjunto limitado de tipos de archivos.

Los archivos gráficos son generalmente almacenados en archivos de TIFF (Tagged Image File Format) estándar ó en archivos de formato PCX. Otros formatos como el EPS (Encapsulated PostScript) e IMG (utilizados en Ventura y algunos otros programas) son también muy comúnmente usados. Generalmente, estos tipos de archivos caen dentro de dos diferentes categorías, Mapeo de Bits ó Archivos Rastreados (raster files), como TIFF o PCX y los fuera de línea (outline) o archivos de vector (vector files), como EPS y los formatos usados por los programas como Illustrator, Corel Draw y Arts & Letters.

Cada tipo de archivo es manipulado por software que es específicamente diseñado para ese formato. De igual manera los programas de pintura (paint programs) trabajan con archivos de mapeo de bits, los cuales pueden manipular pixel por pixel. Los programas de dibujo (draw programs), en contraste, usa un vector gráfico para definir imágenes como líneas y sombras (llamadas objetos). Los paint programs usualmente no pueden manipular los archivos de tipo vector (sin conversión alguna), y draw programas no pueden trabajar con imágenes basadas en pixels.

La mayoría del software para scanner y algunos paquetes de manipulación de imágenes (particularmente paquetes de dibujo) pueden salvar archivos en más de un tipo de formato y de este modo convertirlos entre ellos.

En algunos casos los paquetes de manipulación de imágenes pueden cargar archivos en un tipo de formato y salvarlos en otro, de esta forma estamos siendo capaces de realizar una conversión de un formato a otro, si es necesario.

Para entender mejor esto, a continuación se procederá a explicar brevemente cada uno de los siguientes formatos gráficos:

* EPS	* CGM
MACPAINT	PIC
CUT	TIFF
IMG Y GEM	PCX

El formato PCX, se explicara con más detalle, ya que fue el formato implementado para el sistema.

3.3.1 Formato EPS (Encapsulated PostScript)

Este formato es el más común de los tipos de archivos de vector, permite una gran flexibilidad en el tamaño de imágenes, por lo que en este tipo de formato no se ahorra espacio en disco. Los archivos EPS pueden ser impresos por impresoras PostScript e importadas por la mayoría de los paquetes de publicaciones, tales como Ventura y Page Maker. EPS fue desarrollado por Adobe Systems.

3.3.2 Formato MacPaint

Este tipo de formato fué introducido y es usado por equipos Apple Macintosh,teniendo como principal característica que son archivos binarios, relativamente bajos en resolución.

A pesar de que los sistemas y discos de IBM y Mac son diferentes, se pueden intercambiar archivos salvados en varios formatos estandares.

3.3.3 Formato CUT

Este tipo de formato da como resultado archivos binarios, por lo que no pueden ser usados por imágenes en escalas de gris. Este tipo de formato al igual que el de PCX utiliza un algoritmo de compresión de datos RLL (run length limited).

El archivo CUT comienza con una cabecera que indica las dimensiones de la imagen, como se muestra a continuación :

3.3.4 Formato IMG y GEM

El formato IMG es un formato de mapeo de bits similar al PCX y TIFF, a diferencia del formato GEM que es usado para imágenes tipo vector. Para el caso del formato IMG, los datos de la imagen consisten de una serie de elementos de líneas rastreadas (el número de elementos esta en la palabra 7 de la cabecera), cada elemento de línea incluye un contador de replicas verticales (el número de líneas rastreadas definidas por el mismo

dato), y el dato codificado. Debido a que varias líneas pueden ser idénticas, este código comprime los datos requeridos para almacenar una imagen en pantalla.

BYTE	NOMBRE	COMENTARIO
1-2	Ancho	Ancho de la imagen (en pixels)
3-4	Alto	Alto de la imagen (en pixels)
5-6	0	No usado

Fig. 3.3.3 Cabecera de Archivo CUT

Sin importar el número de planos por bit que se usen, este formato provee datos para cuatro planos: rojo, verde, azul y gris. El dato para cada uno es codificado con 8 pixels almacenados en un byte. Indicando uno de los tres modos :

- solid run
- pattern run
- bit string

El modo *solid run* contiene un byte que indica el estado del pixel (on/off) y el número de paquetes afectado. El modo *pattern run* describe un patrón de datos y el número de veces que se repite. El modo *bit string* describe una cadena de pixels, y comienza con un contador de bytes.

PALABRA	CONTENIDO
0	Numero de version de imagen
1	Longitud de cabecera (en palabras)
2	Numero de planos (bits/pixel)
3	Longitud de definicion de patrones
4	Ancho de pixel (microns)
5	Alto de pixel (microns)
6	Ancho de linea rastreada (pixels)
7	Numero de linea rastreada

Fig. 3.3.4 Cabecera de Archivo IMG

3.3.5 Formato CGM

El formato CGM (Computer Graphics Metafile) es un estándar adoptado por ANSI en 1986 para transportar gráficos a través de diferentes displays y dispositivos de salida. Este formato es del tipo orientado a vector, el cual es usado en programas de aplicación, tales como: Arts & Letters, Lotus Freelance, Harvard Graphics y Micrografx Designer.

El formato CGM ofrece tres alternativas para codificación, cada una de las cuales con un propósito diferente :

- Carácter (ASCII)
- Binario
- Texto limpio

3.3.6 Formato PIC

El formato PIC (Picture File), también llamado GWRITE por Media Cybernetics, es una representación comprimida de memoria gráfica. Si el dispositivo gráfico tiene una paleta programable, cada uno de los archivos PIC tendrá un archivo de paleta (con el mismo nombre, pero con la extensión PAL). Para comprimir imágenes, el formato PIC utiliza el algoritmo de RLL (el mismo de PCX y CUT).

Este formato es usado en programas como HALO, Story Board,etc.

Los archivos PIC son escritos en bloques múltiples de 512-bytes, o registros. El primero contiene una cabecera de 10 bytes como se muestra a continuación :

3.3.7 Formato TIFF

TIFF es la abreviación de Tagged Image Format File. Este tipo de archivos, los cuales tienen por lo general una extensión .TIF, son los más comunes de los archivos de formato estándar. Hay que tener en mente que hay varios tipos de archivos TIFF, incluyendo TIFF

BYTE	ETIQUETA	COMENTARIO
0-1	AH	Indica un archivo HALO
2-3		Numero de version de libreria HALO
4-5	0	No usado
6	2	Indica un archivo de imagen
7	id	Codigo de dispositivo
8-9	0	No usado

Fig. 3.3.6 Cabecera de Archivo PIC

comprimidos, descomprimidos y TIFF PackBits. No todos los programas de software soportan todos los tipos de formatos TIFF.

El formato TIFF puede almacenar tanto archivos binarios (one-bit) y archivos con 16, 64 ó 256 tonos diferentes. Solo el tipo binario puede ser generalmente comprimido, por lo que los archivos TIFF pueden salvar completamente un bit de espacio en disco duro.

Las especificaciones del formato TIFF fueron desarrolladas en 1986 y es llamado así porque cada archivo incluye una colección de información, llamadas marcas (tags), que describen el tipo de archivo. Una marca puede proveer información de la resolución, número de bits usados por pixel y muchos otros descriptores. Los datos básicos necesarios para manipular un archivo son incluidas en un conjunto de marcas estandarizadas que pueden ser interpretadas por cualquier aplicación.

Sin embargo, las aplicaciones pueden crear sus propias marcas con la información que se desee almacenar con el archivo a guardar. Un ejemplo simple de esto puede ser el nombre descriptivo que es desplegado cuando el archivo es cargado. Estas marcas especiales son ignoradas por las aplicaciones que no saben como deben ser leídas, lo cual significa que se puede cambiar archivos TIFF de los más viejos a los mas nuevos, mejorando las versiones de el mismo software. Programas totalmente diferentes pueden leer archivos TIFF creados por otras aplicaciones.

Los problemas aumentan cuando nuevos tipos de marcas incluyen información importante que debe ser entendida por la aplicación para reconstruir la imagen. Es por esa razón que algunas veces se encuentran archivos TIFF que no pueden ser leídos fácilmente por otro software.

Cuatro formatos TIFF estandarizados, usados por muchas aplicaciones son clasificados como B (blanco y negro o información binaria solamente), G (escala de gris), P (paleta - un número de diferentes colores) y R (colores rojo, verde y azul RGB).

Estas clases pueden ser comprimidas o descomprimidas. Page Marker y Ventura pueden leer tanto el tipo de clase B como manipular la versión descomprimida de la clase G.

Si la aplicación da la opción de elegir, se debe usar formato TIFF descomprimido, cuando se sabe que se importaran dentro de otro programa, particularmente un paquete de publicaciones ("desktop publishing"). Más espacio en disco puede ser requerido, pero se evitara teniendo que cargar el archivo dentro del programa que se creo o capturo, y entonces salvarlo de nuevo en un formato compatible.

El formato TIFF para archivos gráficos fue diseñado para solucionar los problemas asociados con los formatos de archivos construidos. Fue creado para estandarizar la industria en los archivos de imágenes. TIFF es un superconjunto de todos los formatos de archivos gráficos existentes. TIFF incorpora la suficiente flexibilidad para eliminar la necesidad de un propietario del archivo de imagen. Fue diseñado viendo hacia el futuro no simplemente el presente, teniendo tres importantes puntos en mente :

a. Extendabilidad. Esta es la habilidad de añadir nuevos tipos de imágenes sin invalidar los tipos anteriores, y añadir nuevos campos informacionales a el formato sin afectar la habilidad de leer archivos de imágenes de viejas aplicaciones.

b. Portabilidad. Ser independiente del hardware y del sistema operativo en el que se trabaje. Tiene poca demanda sobre el medio ambiente en el que se desarrolle. Se ejecuta igual en los medios ambientes de IBM PC y APPLE Macintosh.

c. Revisabilidad. TIFF fue diseñado no simplemente para estar en un medio eficiente de cambio de información de imágenes, si no también para ser útil como formato de datos interno en aplicaciones de edición de imágenes.

El dato rastreado contenido en un archivo TIFF esta organizado en grupos de líneas barridas (o renglones) de los datos de la imagen llamados 'listas' ("strips"). Esta organización ayuda a reducir los requerimientos de la memoria para los lectores de TIFF, ya que la imagen completa no tiene que estar residente en memoria al mismo tiempo.

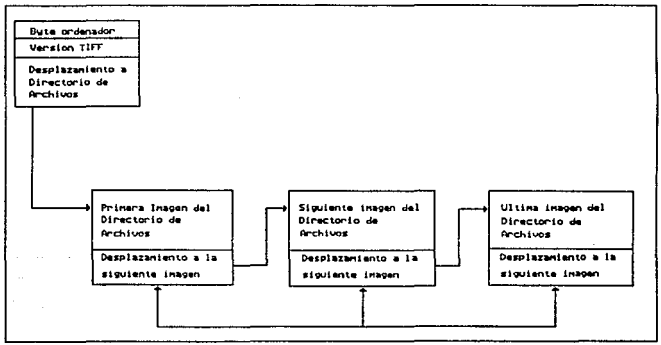


Fig. 3.3.7 Estructura de Archivo TIFF

Solamente debe haber memoria suficiente para almacenar una simple lista de la imagen en un tiempo. Las listas "strips" están limitadas a un valor de 8KB (8,192 bytes) o menos, longitud recomendable. Este código soporta strips de aproximadamente 60,000 bytes de longitud.

NOMBRE	TIPO DE DATO
Byte	1-byte enteros sin signo
Short	2-byte enteros sin signo
Long	4-byte enteros sin signo
ASCII	2-byte datos ASCII
Rational	dos tipos de datos 'long' representando una fraccion, los primeros 4 bytes divididos por los segundos

Fig. 3.3.7' Tipo de Datos TIFF

3.3.8 Formato PCX

Este archivo de formato fue uno de los primeros ensayos en el mundo de las PC's, para habilitar el almacenamiento y estandarizacion de imágenes gráficas. Un archivo de formato estándar era necesario, para permitir el movimiento de imágenes entre aplicaciones y para proveer compresión de archivos para ahorrar espacio de almacenamiento en disco. El archivo de formato gráfico PCX/PCC es un ejemplo del método usado en las industrias que manejan un estándar por default, el cual es probablemente soportado por la mayoría de los programas de aplicación gráfica.

Este código no soporta la existencia de los adaptadores gráficos de CGA y EGA, por lo que no se asegura el correcto despliegue de las imágenes en monitores que no soporten el adaptador gráfico VGA.

El formato de archivo gráfico PCX/PCC no es muy flexible con lo relacionado a la información que contiene. El formato de archivo es rígido, con un archivo de cabecera de longitud construida seguida por la imagen de datos rastreada, opcionalmente seguida por una estructura de paleta extendida. Un archivo PCC es simplemente un subconjunto de un archivo PCX.

El PCX fue originalmente un archivo de formato binario. Las capacidades para escalas de gris fueron añadidas más tarde. Como resultado de esto, mientras que muchos programas pueden trabajar con archivos PCX de 1-bit, las versiones que soportan escalas de gris son pocos y más lejanos. Por ejemplo Ventura Publisher y PageMaker, pueden importar solamente archivos de formato PCX line art. Por lo que se debe convertir archivos PCX en escala de gris a otros formatos antes de ser usados por estos programas.

Los archivos PCX empiezan con una cabecera de 128 bytes, seguida de los datos de la imagen codificada (comprimida) con un algoritmo de compresión de longitud limitada.

La información de la paleta es almacenada en varios formatos. Para datos RGB estandares (para EGA y VGA), la paleta consiste de 16 niveles posibles de 3-bytes de rojo,

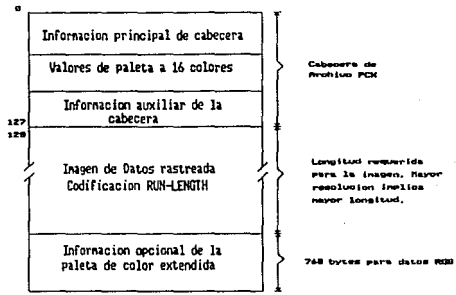


Fig. 3.3.8 Estructura de Archivo PCX

verde y azul respectivamente para mezclar y producir los 16 colores deseados. Si se trata de una paleta extendida de 256 colores 3-bytes, ésta se colocara al final del archivo PCX.

Los archivos PCX generalmente utilizan la extensión .PCX . La figura 3.3.8 muestra una capa típica de un archivo PCX/PCC.

La simplicidad de este formato de archivo hace que el código requerido para soportar PCX sea fácil de entender, desarrollar y usar, en la figura 3.3.8' ilustra la cabecera típica del archivo PCX.

3.3.8.1 Paletas de Archivo PCX

La información de la paleta es almacenada en un archivo PCX, lo que permite desplegar la imagen en los colores con que originalmente se almaceno. Sin la información de la paleta, una imagen tiene que ser desplegada con una paleta default provista por un programa de aplicación. Los colores de la paleta de default del programa de aplicación pueden no corresponder a los colores de la imagen original. De hecho el hacer esto puede arruinar la belleza de la imagen original. Salvando la información de la paleta con el archivo PCX permite que otras aplicaciones desplieguen imágenes correctamente y precisas.

Es importante entender la diferencia entre la información de la paleta almacenada en el archivo PCX y el mecanismo de la paleta usado por el adaptador de display VGA.

BYTE	DETALLE	TAMANO (bytes)	COMENTARIOS
0	Manufactura	1	10 : PC paintbrush PCX
1	Version	1	0 : Version 2.5 2 : Version 2.8 3 : Version 2.8 sin paleta 5 : Version 3.0
2	Codificacion	1	1 : PCX 'run length'
3	Bits por Pixel	1	Numero de bits por pixel
4	Ventana	8	Dimensiones de la imagen (x,y)
12	HRes	2	Resolucion horizontal en video
14	VRes	2	Resolucion vertical en video
16	Mapa de Color	48	Paletas de colores
64	Reservado	1	
65	NPlanos	1	Numero de planos de color
66	Bytes por Linea	2	Numero de bytes por linea rastreada por mapa de color
68	Informacion Paleta	2	1 : Color o Blanco y Negro
70	Restantes	58	No usados

Fig. 3.3.8' Cabecera de Archivo PCX

Una paleta es una colección de colores, que pueden ser desplegados simultáneamente y son usados para desplegar una imagen. La información de la paleta en un archivo PCX nos dice los componentes de color correctos (componentes de color rojo, verde y azul que proporcionan el color deseado) para desplegar una imagen. Debe haber un registro de la paleta en el archivo PCX para cada color usado en la imagen rastreada. El orden de las entradas de la paleta en el archivo PCX determinan el valor del pixel requerido para desplegar el color descrito. Por ejemplo un pixel de valor cero debe ser desplegado en el color especificado por el registro cero de la paleta. Así como un pixel de valor 15 debe ser desplegado por el color especificado en el registro 15 de la paleta, y así sucesivamente.

Para el correcto despliegue de una imagen PCX, es necesario construir una paleta para el adaptador gráfico y cargar los registros de color de VGA con los componentes RGB de los colores requeridos. El camino mas fácil para efectuar esto es cargar los registros de color empezando en el registro cero con los componentes de RGB leídos del archivo PCX, y entonces crear una paleta de 16 entradas conteniendo los valores de 0 a 15. Esto proporcionara el mapeo requerido de los valores de pixeles a color. Hay que notar que este es solo un mapeo de muchos que podrían proporcionar el mismo resultado. Solamente cuando la paleta de colores que esta siendo usada por el adaptador gráfico VGA semejante a la paleta contenida en el archivo PCX puede desplegar una imagen en sus propios colores.

En los archivos PCX son utilizados dos mecanismos de paletas diferentes. El mecanismo de la paleta original es construido dentro de la cabecera del archivo. Esta paleta es usada por imágenes que contienen arriba de 16 colores para el display. Esta estructura de paleta tiene espacio para 16 valores de registros de color compuestos de 1 byte de información de color rojo,verde y azul. El espacio total asignado para esta estructura de paleta en el archivo cabecera PCX es de 48 bytes (16 registro de color • 3 bytes de cada color). El como se interpreta la información de la paleta depende del adaptador gráfico que se este utilizando.

Para imágenes de 16 colores VGA normal, los primeros 16 registros de color son puestos a los colores descritos por los componentes RGB en la estructura de la paleta del archivo PCX. La paleta actual es actualizada en orden secuencial empezando en cero y terminando en 15. Un valor cero en el pixel de la imagen seleccionara el registro de color cero, que en la mayoría de las veces es 'negro'. Un valor de 15, seleccionara el color correspondiente al registro de color 15 y asi sucesivamente.

Un adaptador gráfico EGA es capaz de desplegar 16 colores simultáneamente en la pantalla con una paleta de 64 colores. El formato de un descriptor de color esta dado por:

r'g'b' R G B
donde:

R, G y B son llamados los bits de color base y
r', g' y b' son los bits de color alternado.

Si un byte de componente de color tiene cualquiera de los bits pares puestos del conjunto de bits, entonces el bit de color base en considerado como encendido. Si cualquiera de los bits impares esta puesto, el bit de color alternado esta encendido. Por otra parte si ambos pares y nones están puestos, los bits de base y color alternado están encendidos. Para determinar cual de los 64 posibles colores se refiere la entrada de la paleta, es necesario aplicar la siguiente regla para cada uno de los bytes RGB del archivo PCX. Por ejemplo, si el valor en hexadecimal de los componentes de RGB es igual a 55, AA y FF respectivamente, el descriptor de color será igual a:

Descriptor	r'	g'	b'	R	G	B	
Red = 55 = bits pares	1			0			conjunto de bit alterno
Green = AA = ebits nones		0			1		conjunto de bit base
Blue = FF = bits pares y nones			1			1	ambos conjuntos
valor del color	1	0	1	0	1	1	= Color 43

La paleta EGA para esta paleta PCX será puesta en color 43. (Hay que notar que el registro de la paleta RGB valida para este mismo color será 01, 02, 03 hexadecimal. Esto satisface el mismo criterio con los bits pares que con los bits nones).

Para imágenes CGA de 4-colores, los componentes de rojo para el primer registro de la paleta dividido entre 16 es el color de fondo 'background' (0 a 15), y el componente rojo del segundo registro de la paleta dividido entre 32 es el identificador de paleta CGA (0 a 7, aunque el estándar CGA soporta solamente paletas de 0 a 3). El resto de los datos de la paleta son ignorados.

3.4 Compresion de Imagenes

La gran cantidad de información obtenida de las imágenes gráficas digitalizadas, han desarrollado la necesidad de la creación de muchas técnicas de codificación.

Es común que a las imágenes de tamaño grande se les quiera comprimir para su mejor manipulación, transmisión y almacenamiento, por lo tanto se codifican los datos de una forma que consuman menos espacio que en su forma original, y así poder ser más rápida su transmisión y menos denso su almacenamiento. Existen numerosas técnicas para solucionar este problema, algunas más usadas que otras, dependiendo de la aplicación. La compresión de datos en una imagen se encuntran en la clase importante de operaciones de codificación de imágenes.

En una vista natural de una imagen, se puede encontrar información redundante. Cuando esta información se procesa, se puede reducir a una forma más simple, produciendo una imagen que requiera menos datos para describirla.

Los siguientes métodos de compresión de imágenes, son comúnmente usados en los formatos gráficos explicados anteriormente. El algoritmo de compresión RLL es el utilizado por el sistema *Clasificador de Color*, implícitamente usado en el formato PCX.

3.4.1 Algoritmo de Compresion de Longitud Limitada (Run Length Limited Compression)

La necesidad para la compresión de datos se vuelve cada vez mas obvia, debido a la cantidad de espacio requerido para el almacenamiento de una imagen. Como ejemplo, supongamos 307,200 bytes de almacenamiento para una imagen de 640 x 480 en tonos continuos producida por un digitalizador. Una página de scanner con una resolucion de 300 DPI (dots per inch. puntos por pulgada) en ambas direcciones, vertical y horizontal, utilizando 8 bits por pixel se produce un archivo de 8 megabytes de tamaño. En la figura que se muestra a continuación se describen los requerimientos para almacenar imágenes de mapeo de bits.

A. Se asume que todos los datos son de imágenes de 8.5 x pulgadas.

B. Las resoluciones vertical y horizontal son las mismas.

C. La siguiente tabla indica el numero total de bytes requeridos para almacenar imágenes sin comprimir, en la resolucion especificada.

La cantidad de compresión realizada prácticamente depende del tamaño del archivo, sin embargo esta es aproximadamente de la mitad a tres cuartas partes del tamaño original. Esta es una reducción significativa de un método simple de compresión.

RESOLUCION EN PUNTOS/PULGADA (DPI)	BITS POR MUESTRA			
	1	4	8	24
72	60,588	242,352	484,704	1,454,112
150	262,969	1,051,875	2,103,750	6,311,250
300	1,051,875	4,207,500	8,415,000	25,245,000

En la compresión RLL cada renglón de la imagen es comprimido por separado. Si hay múltiples-planos por bit en una imagen, la compresión de datos consiste en tomar un renglón por cada úno de los planos. El formato PCX soporta arriba de 4 planos por bit. El orden de los planos es de BGRI, que es, renglón de dato azul (B), verde (G), rojo (R) y el renglón del dato de intensidad (I). La compresión RLL no puede ligar una línea rastreada, pero si puede ligar los componentes de color de una línea rastreada. Si la imagen no usa múltiples planos por bit, el dato de la imagen es el índice dentro de la paleta para cada valor de pixel de la línea rastreada. El renglón de dato es comprimido reemplazando una secuencia repetitiva de bits por un byte contador de repeticiones y un byte de dato.

Un byte contador de repeticiones es identificado por tener sus 2 bits mas significativos en 1, sus 6 bits restantes menos significativos especifican un valor de 1 a 63 para el byte de dato que sigue. Si el byte de dato tiene puestos ambos de sus bits de mayor orden a 1, también es considerado como un byte contador de repeticiones y un byte de dato. El pseudocodigo de este método de compresión RLL es como sigue :

Pseudocodigo.

```
Se calcula el numero total de bytes a comprimir
Repite
Toma el byte de entrada de dato
 Incrementa el apuntador de entrada
 Pone el contador de repeticiones a 1
Mientras (el siguiente byte de entrada de dato sea semejante
        y el contador de repeticiones no sea mayor a 63 y haya mas
        bytes de datos a comprimir)
 Incrementa el contador de repeticiones
FindeMientras
SI (el contador de repeticiones es mayor que 1 o el
```

```
carácter tiene 2 MSBs puestos) entonces
    Pone 2 MSBs en contador de repeticiones
    Escribe el contador de repeticiones al archivo de salida
FindeSi
Escribe el byte de dato al archivo
Hasta que todos los bytes se compriman
Fin
```

Debido a que la función PCX debe ser capaz de leer y escribir archivos de imágenes PCX, es necesaria una función de expansión para la codificación RLL. El pseudocodigo para este proceso de expansión es:

Pseudocodigo.

```
Calcula el numero de bytes expandidos
Repite
Lee el byte de dato de el archivo
SI (hay 2 MSBs puestos a 1,) entonces es una marca repetida
    Enmascara los MSBs para obtener el contador de repeticiones
    Toma el siguiente byte de dato del archivo
     Almacena este byte de contador el numero de veces en un buffer de salida
SiNo
     Es byte de dato actual
    Almacena byte de dato actual en un buffer de salida
FindeSi
Hasta que todos los bytes sean expandidos
```

Las imágenes de planos de bit simples tienen un proceso diferente que las imágenes multiplano. Actualmente las imágenes de plano de bit simples tienen asociado mucho menos proceso, por lo que pueden ser comprimidas y expandidas mucho mas rápido. Las imágenes que utilizan VGA en modo 13 hex, el modo de 256 colores, son imágenes de plano simple.

3.4.2 Compresion de Datos TIFF

Como se menciono anteriormente los archivos de imágenes de mapeo de bit requieren de gran cantidad e espacio (bytes) de almacenamiento, por lo que es necesaria la implementación de algoritmos de compresión de datos, para reducir el tamaño de estos archivos. La especificación TIFF soporta 4 diferentes tipos de compresión de datos . Un generador de archivos TIFF debe soportar al menos uno de estos métodos. Un lector de archivos TIFF debe ser capaz de entender cada uno de estos métodos.

La recomendación TIFF no recomienda almacenar imágenes en forma no compresa debido al tamaño del archivo resultante.

Capítulo 4

Análisis y Desarrollo para la Adquisición de la Imagen a Color

4.1 Introducción

En términos generales el procesamiento de imágenes consiste en la manipulación de una imagen inicial, alterándola de alguna manera, para obtener una imagen final. Se puede decir que una imagen es un registro de información bidimensional de algún objeto del mundo real puesto que para que tenga sentido para el observador, siempre tendrá que tener un largo y un ancho, ya que el ser humano posee una capacidad inherente de procesamiento de imágenes, que se manifiesta con el reconocimiento de rostros, objetos, etc. De mediciones experimentales se ha obtenido que el sistema visual humano (el sistema ojo-cerebro) es capaz de distinguir aproximadamente 350,000 colores. Este sistema visual utiliza los colores dominantes para transmitir la información de la imagen a nuestro cerebro, aquí donde la imagen es recibida, los aspectos más sutiles del color dentro de la imagen son utilizados para engrandece los detalles de la misma, pero no son absolutamente necesarios para reconocerla, ya que el cerebro se encarga de llenar con detalles la imagen, para que esta tenga un sentido.

Este procesamiento natural es el que se pretende emular con un sistema computacional de procesamiento de imágenes. Las imágenes a color, en teoría pueden ser producidas por la combinación de imágenes obtenidas por medio de filtros rojo, verde y azul. La técnica en la practica para un resultado óptimo, depende de muchas variables incluyendo la calidad y tipo de los filtros, la respuesta espectral de la cámara de video, así como las características de iluminación en el objetivo, por citar algunas. Dejando como parte principal del proceso el algoritmo de clasificación de color.

4.2 La Imagen Original

Los datos de entrada son la separación de rojo, verde y azul de una imagen a color digitalizada tres veces con cada uno de los filtros rojo, verde y azul respectivamente. La estructura para cada imagen de entrada es un arreglo rectangular de pixels cada uno compuesto a su vez por 8 bits, por lo cual cada componentes de color es representado por números en el rango de [0,255] ($2^8 = 256$ colores posibles).

Para adquirir una imagen este sistema digitalizador lleva a cabo cuatro funciones específicas para una correcta colocación de datos en cada arreglo de pixels para su posterior manipulación por el algoritmo de color.

4.2.1 Digitalización de la Imagen.

Aquí se generan las señales necesarias para la adquisición de una imagen. La imagen que se observa en el monitor es la imagen que se adquiere y almacena en el banco de memoria de la tarjeta digitalizadora, mediante el PUERTO_CONTROL cuya dirección es la 319 hex, el cual debe ser puesto en MODO_ADQUISICION para que la señal de video pueda ser digitalizada y colocada en el banco de memoria cuya dirección de inicio es la D0000000 hex. El código en lenguaje C que realiza esta función es :

- outportb(PUERTO_CONTROL,MODO_ADQUISICION)

4.2.2 Obtención de imagen (320 x 200)

Para los fines de este sistema digitalizador se debe de obtener una imagen de 320 x240 de la imagen de 512 x 512 que el digitalizador tiene almacenada en su banco de memoria. Esto es realizado por las limitantes de equipo, ya que un monitor estandar VGA cuenta con un solo modo a 256 colores simultáneos el modo 13 hex de 320 x 200 a 256 colores, el cual será utilizado para visualizar la imagen resultante.

La imagen de 320 x 240 se obtiene partiendo del centro de la imagen de 512 x 512, en el pixel con coordenadas (256,256), tomando 120 pixels hacia cada extremo del eje X y 160 pixels hacia cada lado del eje Y; tal y como lo muestra la figura 4.2.2a

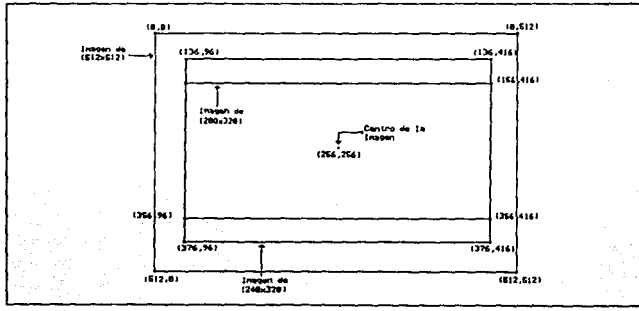


Fig. 4.2.2a Obtención de una imagen de 320 x 240

Aunque la figura 4.1 muestra la imagen en una forma matricial, esta se encuentra realmente almacenada en la memoria del digitalizador como un arreglo unidimensional. Como se explico en la descripción de la tarjeta digitalizadora en el capitulo dos,esta hace uso del concepto de memoria expandida, utilizando para ello un bloque de memoria de la PC de 16Kb en la dirección D0000000 hex. Esto implica que se tiene acceso a solo 16Kb de toda la imagen, por lo que uno se debe de mover bloque a bloque para accesar la imagen completa, esto es posible a través del PUERTO_BLOQUE y mediante la instrucción de lenguaje C:

- outportb(PUERTO_BLOQUE,numero_de_bloque)

De tal forma que la imagen completa se encuentra en un total de 16 bloques, cada bloque conteniendo 62 líneas de la imagen. Los bloques del 0 al 7 contienen las líneas pares de la imagen mientras que en los bloques del 8 al 15 están las líneas impares. Esto

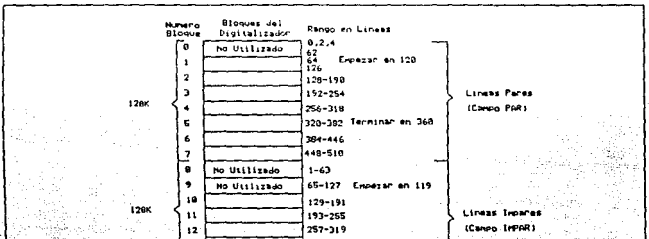


Fig. 4.2.2.b Distribución de la Memoria en la Tarjeta Dig.

puede ser visualizado en la fig. 4.2.2b donde además se indica que bloques no son utilizados para formar la imagen de 320 x 240.

4.2.3 Problemas de Aspect Ratio

Para los fines que se persiguen el "aspect ratio " puede ser definido como la correcta relación entre la altura y el ancho de una imagen desplegada sobre un monitor de video. Un circulo desplegado con un correcto aspect ratio se visualizara redondo, mientras que

un aspect ratio imperfecto deformará el círculo y hará que éste se observe como una elipse, de la misma forma un cuadrado con un aspect ratio incorrecto se vera como un rectángulo. Distorsiones en las imágenes causadas por el aspect ratio son usualmente el resultado de intentar desplegar una imagen utilizando pixels "no cuadrados".

El aspect ratio de los típicos dispositivos de video (televisión, monitores de computadora y cámaras) es de 4:3. Esto significa que el ancho del desplegado en una pantalla de monitor es aproximadamente 1.333 veces la altura de la pantalla.

En lo que respecta a las resoluciones en los monitores de computadoras el modo de 640x480 del VGA tiene pixels cuadrados puesto que la relación entre el ancho y el alto del pixel es igual al del video standard (640/480 es exactamente igual a 4/3). El ancho del display en el monitor IBM 8512 por ejemplo, es aproximadamente 9.5 pulgadas mientras que el alto del display es aproximadamente 7.125 pulgadas lo que da una perfecta relación 4:3. Partiendo de estas mediciones se puede encontrar la resolución del display en pulgadas mediante los siguientes cálculos :

PPH = Pixels por pulgada horizontalmente = 640 pixels / 9.5 pulgadas = 67.37 pixels/pulgada
PPV = Pixels por pulgada verticalmente = 480 pixels / 7.125 pulgadas = 67.37 pixels/pulgada

Lo cual sirve para obtener las dimensiones del pixel :

Pulgadas por pixel horizontalmente = 1/PPH = 0.0148 pulgadas

Pulgadas por pixel verticalmente = 1/PPV = 0.0148 pulgadas

En esta forma cada pixel sobre una pantalla de 640x480 es aproximadamente un cuadrado de 0.0148 x 0.0148 pulgadas. De la misma manera se procede para el modo 320x200 :

PPH = 320 pixels / 9.5 pulgadas = 33.68 pixels/pulgada
PPV = 200 pixels / 7.125 pulgadas = 28.07 pixels/pulgada
Pulgadas por pixel horizontalmente = 1/PPH = 0.0297 pulgadas
Pulgadas por pixel verticalmente = 1/PPV = 0.0356 pulgadas

Así cada pixel en una resolución de 320x200 es aproximadamente 0.0297 pulgadas de ancho por 0.0356 de alto, lo que definitivamente no es cuadrado.

Para corregir este problema se hace uso de la interpolación lineal, sin embargo cabe aclarar que añadir el código de la corrección del aspect ratio a la adquisición de la imagen dará como resultado mejores imágenes, pero a cambio se requerirá tiempo extra en la adquisición de la mismas. Para corregir el aspect ratio de una imagen con resolución de 320x200 es necesario:

a) Adquirir una imagen de 320x240, ya que tal imagen presenta una aspect ratio de exactamente 4:3.

b) Con una imagen de 320x240 es necesario comprimir las 240 líneas de la imagen digitalizada en 200 del modo de resolución del monitor, para lo cual se emplea la interpolación lineal que permitirá calcular exactamente los valores de intensidad de los 200 pixels verticales, consiguiendo de esta forma pixels cuadrados. El factor de conversión que permite realizar tal compresión se obtiene dividiendo las 240 líneas de la imagen a

ser comprimidas entre los 200 pixels verticales de la resolución del monitor lo cual arroja como resultado un FACTOR DE CONVERSION de 1.2.

La interpolación lineal es usada para calcular el valor de un nuevo punto situado entre otros puntos de valores y distancia conocidos. Los puntos de valor que rodean al punto a ser calculado son denominados 'vecindario'. La interpolación lineal asume que la contribución de un pixel en el vecindario varia directamente con su distancia. La figura 4.2.3 ilustra como la interpolación es usada, la intensidad del pixel calculado es derivado desde los pixeles A,B,C y D de acuerdo a su relativa distancia desde la dirección calculada desde el pixel transformado Px.

Para obtener el valor interpolado del pixel de interés Px, la distancia desde cada uno de los pixels en el vecindario deberá ser conocida para poder interpolar correctamente. En la práctica solamente la distancia del pixel de la esquina superior izquierda del vecindario (el pixel A en el diagrama) al pixel de interés deberá ser conocido.Puesto que la compresión de la imagen es en sentido vertical, el proceso de interpolación lineal se aplica solo en este sentido, de tal forma que solo es necesario conocer la línea delta la cual es calculada como :

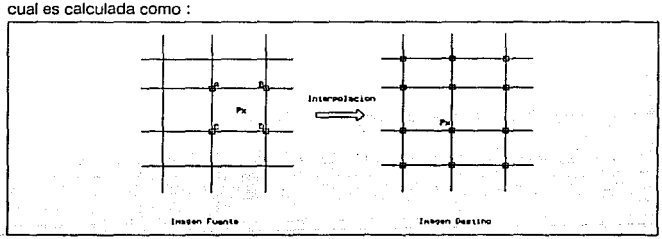


Fig. 4.2.3 Proceso de Interpolación

Línea_Delta = dirección fraccional del renglón de Px - Dirección entera del pixel A.

Y las intensidades solo son aportadas por los pixels A y C, cuya contribución esta dada por :

Intensidad de Px = Línea_delta * (Intensidad del pixel A - Intensidad del pixel C) + Intensidad del pixel C

El valor calculado para Px y colocado en la imagen destino es de esta manera proporcional a las intensidades y a la distancia de Px de cada pixel a sus alrededores.

Los efectos de no correguir el aspect ratio pueden ser visualizados en las imagenes que se encuentran al final de este capitulo, en donde se puede comparar una imagen sin pixels cuadrados y una imagen con el aspect ratio correcto.

4.2.4 Normalización de la imagen

Una vez que se tiene una imagen de 320x200, se da paso a un proceso de normalización la cual simplemente nos permite que el cubo RGB trabaje con memoria mínima. Este proceso obliga a que cada dato en la imagen se encuentre en el rango de 0 a 31 en lugar de su original rango de 0 a 255, en otras palabras solo se utilizaran 5 bit de cada pixel en lugar de los originales 8 ($2^5 = 32$ y $2^8 = 256$). Así la capacidad del cubo RGB será 32•32•32 en lugar de 256•256•256 colores posibles.

4.3 Algoritmo para la Cuantización de Imágenes a Color

LA CUANTIZACION DE IMAGENES A COLORES es el proceso de seleccionar un conjunto de colores para representar toda la gama de colores que conforman una imagen determinada, y calcular el mapeo desde el espacio de colores (colores de la imagen original) a los representativos colores (colores en el mapa de color).

El algoritmo de color empleado utiliza esta técnica de cuantización para desplegar un "frame buffer".

La técnica de cuantización de imágenes a color es dividida en cuatro fases :

- 1) Muestrear la imagen original para determinar la distribución de color.
- 2) Seleccionar el mapa de color basándose para ello en la distribución anterior.
- 3) Mapear los colores originales al más cercano representativo en el mapa color obtenido en el paso dos.
- 4) Cuantizar y redibujar la imagen original (con opcional DITHER y colores óptimos).

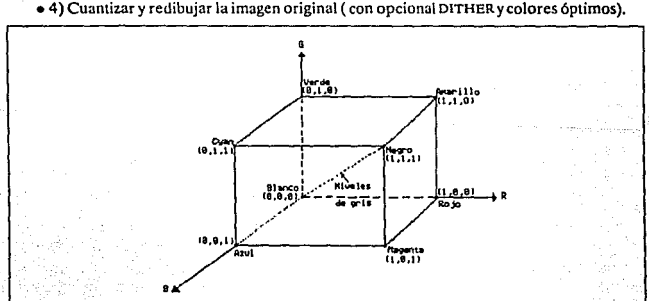


Fig. 4.3.1a Modelo de Color RGB dentro de un Cubo Unitario

4.3.1 Fase 1 : Muestreo de la Imagen Original

La información requerida por el algoritmo de selección del mapa de color de la fase dos es un 'histograma' de los colores de la imagen original.

En esta fase de muestreo el cubo RGB es llenado con las frecuencias de color obtenidas de la combinación de las imágenes roja, verde y azul.

Para lo cual se hace uso del "modelo de color RGB" (ver capitulo uno para más detalles), como se recordara este modelo puede ser representado como un cubo unitario definido en los ejes R (rojo), G (verde) y B (azul), en donde cada punto de color contenido en el cubo puede representarse como la tríada (R,G,B) y los valores de cada componente estarán en el rango de cero a uno como lo muestra la figura 4.3.1a

Ahora bien, para los fines que se persiguen y en base a la normalización que se realizó a la imagen en la etapa de adquisición explicada anteriormente; el cubo unitario se multiplica por 32, y de esta manera el cubo estará formado por 32 planos y cada plano contendrá una matriz de 32x32 elementos, partiendo desde la coordenada columna cero, renglón cero hasta la coordenada columna 31, renglón 31. Así cada elemento estará en el rango de cero a 31, por lo cual la tríada menor será (0,0,0) y la mayor será (31,31,31). La figura 4.3.1b muestra este cubo el cual a partir de este momento será denominado "Cubo RGB".

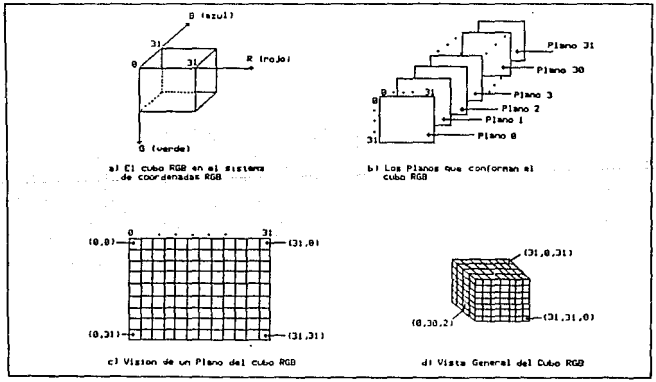


Fig. 4.3.1b Cubo RGB

Inicialmente todo el cuboRGB es colocado a ceros, para posteriormente accesar sobre la misma coordenada una matriz de la imagen roja, verde y azul y obtener en base a cada valor del elemento que se accesa de cada una de las tonalidades presentes, la localidad que le corresponde dentro del cuboRGB; incrementando en uno el valor de dicha localidad. Este proceso es realizado por cada uno de los pixels que conforman la imagen original, esto es 64,000 veces. Para saber cuantos colores están presentes de los 32,768 colores posibles, basta con recorrer todo el cuboRGB y contar todas las localidades diferentes de cero. El pseudocódigo de este proceso es el que a continuación se muestra y la visualización de dicho proceso la proporciona la figura 4.3.1c :

```
Función para colocar las frecuencias de color dentro del cuboRGB
    INICIO
/* inicializamos todo el cuboRGB a ceros */
i=0;
 REPITE
 j=0;
  REPITE
  k=0;
   REPITE
   cuboRGB[k,j,i] = 0;
    k=k+1;
   HASTA  k >32
  j=j+1
 HASTA  j>32
 i=i+1
HASTA  i>32
/* colocamos los colores presentes en la imagen dentro del cuboRGB*/
```

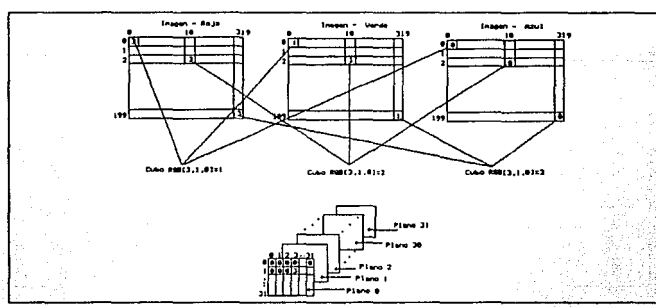


Fig. 4.3.1c Llenado del Cubo RGB con Frecuencias de Color

```
i=0;
REPITE
   cuboRGB[rojo[i],verde[i],azul[i]] =cuboRGB[rojo[i],verde[i],azul[i]] + 1;
   i=i+1;
HASTA i=TOTAL_DE_PIXELS_EN_IMAGEN
FIN
```

Esta agrupación de los colores tiene el efecto de reducir el número de diferentes colores e incrementar la frecuencia de cada color. Estas propiedades son importantes para el algoritmo que a continuación se describe.

4.3.2 Fase 2: Eleccion de un Mapa de Color

Diferentes algoritmos y técnicas son utilizadas para la selección de un mapa de color, la calidad de muchos de ellos dependen de la clase de imágenes que se este procesando. En esta sección se discute un algoritmo para elegir un conjunto representativo de colores (mapa de color) basándosepara ello sobre la distribución de entrada.

4.3.3 Algoritmo de Corte Medio

El concepto general detrás del algoritmo de corte medio es usar cada uno de los colores presentes en el mapa sintetizado de colores (el cubo RGB) para representar un igual número de pixels en la imagen original. Este algoritmo repetidamente subdivide el espacio de color dentro de pequeñas y más pequeñas cajas rectangulares. Por lo que se deberán

Columna Inicial	Renglon Inicial	Plano Inicial
Columna Final	Renglon Final	Plano Final
Numero de Elementos en esta Caja		

conocer las coordenadas que ocupa una caja dentro del cuboRGB esto es la columna y renglón inicial y el plano donde se ubican dichas coordenadas, así como la columna y renglón final con su correspondiente plano y el número de elementos o frecuencias de color que guarda dicha caja; de forma tal que una caja puede ser representada mediante la siguiente estructura:

0	0	0
31	31	31
64,000		

El algoritmo comienza con una caja la cual representa los colores de todos los 64,000 pixeles de la imagen original (imagen de 320x200),los cuales fueron distribuidos en el cuboRGB por la fase uno. Por lo tanto la primera caja deberá siempre abarcar todo el cuboRGB y contendrá todos los 64,000 pixels de la imagen, esto se ilustra en la página siguiente. De entrada esta caja deberá ser "reducida" para eliminar los planos cero sobre los lados de la caja. Este proceso de "reducción" en general consiste en buscar sobre los tres ejes del cuboRGB el primer elemento distinto de cero, y obtener en base a estas coordenadas la ubicación de la nueva caja dentro del cuboRGB.

4.3.3.1 Proceso de reducción del cuboRGB sobre el eje R

(R permanece más tiempo constante mientras que G y B varían más rápido)

Se busca en el plano cero desde la coordenada (0,0) hasta la coordenada (0,31) un elemento diferente de cero. En caso de no encontrar alguno, se continua la búsqueda sobre el plano uno desde la coordenada (0,0) hasta la coordenada (0,31). Si nuevamente

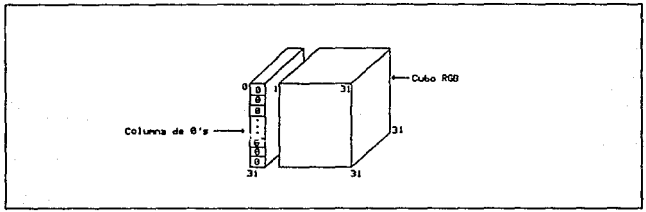


Fig. 4.3.3.1a Eliminación de Columna Cero en Cubo RGB

no se encuentra algún elemento la búsqueda continua pero ahora sobre el plano dos desde la coordenada (0,0) hasta la coordenada (0,31). Así sucesivamente hasta el plano 31 desde la coordenada (0,0) hasta la (0,31); si hasta este momento no se ha encontrado un elemento distinto de cero, la columna número cero será eliminada de todo el cuboRGB como lo muestra la figura 4.3.3.1a

El proceso continúa y la búsqueda se reanuda pero ahora partiendo desde la columna uno, renglón cero del plano cero, esto es desde la coordenada (1,0,0) del cuboRGB, se busca el elemento que es distinto de cero en profundidad hasta el plano 31. En caso de no encontrar ningún elemento distinto de cero esta nueva columna será eliminada y comenzamos nuevamente pero ahora desde la columna dos, renglón cero del plano cero o sea la coordenada (2,0,0) del cuboRGB. Estos pasos son aplicados columna a columna hasta encontrar un elemento distinto de cero o hasta llegar a la columna 31, renglón 31 del plano 31 esto es, hasta agotar todo el cuboRGB. Cuando un elemento distinto de cero

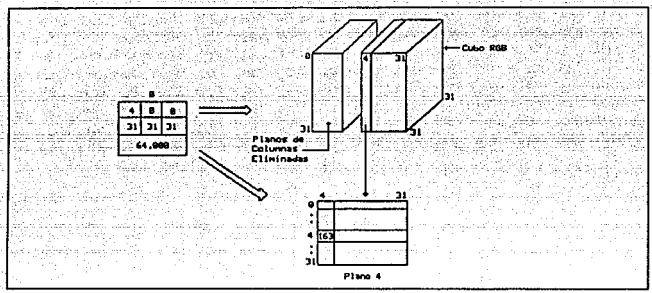


Fig. 4.3.3.1b Reducción del Cubo RGB sobre el Eje R

dentro del cuboRGB es encontrado, el renglón de este elemento indica la coordenada (x,x,x) de inicio de la nueva caja del cuboRGB. Por ejemplo si el primer elemento distinto de cero se encuentra en la coordenada (4,4,3) nuestra caja y el cubo tendrán hasta este momento la forma que se muestra en la fig. 4.3.3.1b

El siguiente paso dentro del proceso de reducción del cuboRGB es parecido al anterior, pero ahora se parte desde las coordenadas finales de la caja. De esta forma la búsqueda de un elemento distinto de cero comienza en el plano 31 desde la coordenada (31,31) hasta la coordenada (4,0); como se puede notar el proceso se va retroalimentando y en base al ejemplo que se sigue, la nueva columna de inicio es la encontrada en la primera parte (fig. 4.3.3.1b). En caso de no encontrar ningún elemento se continua la búsqueda pero ahora en el plano 30, desde la coordenada (31,31) hasta la coordenada (4,0).

Y como en la primera parte de este proceso esto es aplicado constantemente hasta lograr el objetivo que es encontrar la columna donde aparece por primera vez un elemento distinto de cero, con lo cual se tiene ahora la coordenada (x,x,x) final de la nueva caja rectangular. Por ejemplo si un elemento distinto de cero se encuentra en la coordenada (31,27,23) la caja y el cubo mantendrán la misma estructura de la figura 4.3.3.1b, puesto que la columna 31 del plano 23 contiene el primer elemento distinto de cero, lo cual implica no eliminar ninguna columna.

4.3.3.2 Proceso de reducción sobre el eje G.

(G permanece más tiempo constante mientras que R y B varían más rápidamente)

Se busca en el plano cero desde las coordenadas indicadas en la estructura de la caja, la cual se encuentra ya actualizada por el proceso de reducción en el eje R, así la caja de

entrada para el ejemplo que se sigue es la que se muestra en la figura 4.3.3.1b; de esta forma se buscará un elemento distinto de cero desde la coordenada (4,0) hasta la coordenada (31,31) del plano cero, manteniendo nuestro renglón fijo debemos eliminar ahora los planos de los renglones que sean cero a través de los 31 planos que contiene la caja. Y como en el proceso de reducción en el plano R, en caso de no encontrar un elemento distinto de cero, este renglón es eliminado y se regresa nuevamente al plano cero pero incrementando el renglón en uno, repitiendo la búsqueda pero ahora desde la

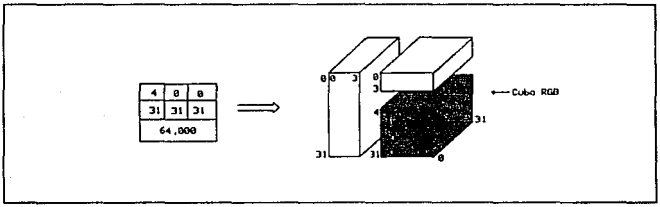


Fig. 4.3.3.2 Representación Gráfica de los Planos

coordenada (4,1) hasta la (4,31).

Así sucesivamente se continua con este proceso hasta encontrar un elemento distinto de cero. Si dicho elemento por ejemplo se ubicará en las coordenadas (4,4,3) la caja y el cuboRGB tendrán la estructura mostrada en la fig. 4.3.3.2

El siguiente paso es repetir la primera parte de la reducción sobre el eje G como fué descrito anteriormente, pero ahora partiendo desde la columna 31, renglón 31 sobre el plano 31, hasta encontrar un elemento distinto de cero. Esto es partiendo ahora desde las coordenadas finales de la caja y decrementando en uno cada una de ellas como se vaya requiriendo para accesar el cuboRGB. Si dicho elemento por ejemplo se encontrára en las coordenadas (13,31,15) la caja y el cuboRGB permanecerían inalteradas como lo indica la figura 4.3.3.2, ya que el primer elemento distinto de cero está ubicado en el renglón 31 del plano 15.

4.3.3.3 Proceso de reducción sobre el eje B

(B permanece más tiempo constante mientras que R y G varían más rápidamente)

En este proceso se eliminan los planos que contengan elementos cero para lo cual partimos de nuestra caja actualizada mostrada en la figura 4.3.3.2 y tomamos las coordenadas de inicio de esta. Así buscaremos sobre el plano cero si éste contiene algún elemento distinto de cero, partiendo desde la columna 4, renglón 4 hasta la coordenada (31,31), en caso de no contener alguno éste será eliminado, continuandose la búsqueda

sobre el plano uno en las mismas columnas y renglones, y si nuevamente este plano no contiene elementos diferentes de cero se eliminan para proseguir la búsqueda incrementado la coordenada correspondiente al plano, hasta encontrar alguno que contenga un elemento distinto de cero.

Por ejemplo si este elemento se encuentra en la coordenada (4,4,3) se eliminan los planos cero, uno y dos.

A continuación se realiza similar proceso, partiendo desde las coordenadas finales de

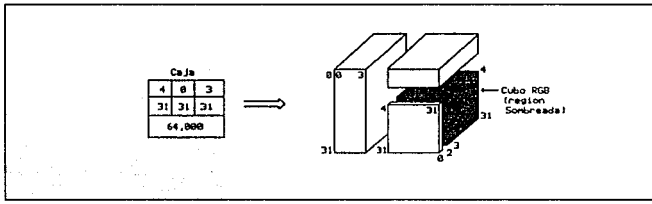


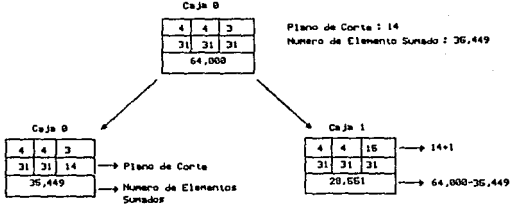
Fig. 4.3.3.3 Reducción Total del Cubo RGB

la caja, esto es desde el plano 31, columna 31 y renglón 31 y decrementando en uno el plano hasta encontrar un elemento distinto de cero; si dicho elemento se encontrará en la coordenada (8,22,31), la caja y el cuboRGB al salir del proceso de reducción total se visualizará como la estructura final que se muestra en la figura 4.3.3.3

Posteriormente arrancando de esta caja se realiza una "partición adaptativa" para decidir como se dividirá. Para lo cual se sumarán los elementos que se encuentran en cada plano comenzando desde las coordenadas iniciales de la caja de entrada, mostrada en la figura 4.3.3.3, de esta manera se comenzará desde el plano tres, renglón 4 columna 4 hasta la coordenada (31,31) de dicho plano. Al terminar de sumar cada plano se debe verificar si esta suma es menor que la mitad de los elementos contenidos en esta caja, en otras palabras si la suma no es mayor a 32,000. En caso de ser menor se continúa con el plano 4 acumulando los elementos que se encuentran en el, este proceso se realiza sucesivamente hasta sobrepasar la mitad de los elementos en la caja.

En el plano donde se exceda esta mitad será en el que se cortará la caja, como resultado se construirá otra caja que contendrá el renglón inicial y final de la caja vieja, así como la columna inicial y final, el plano final de la vieja caja y como plano inicial contendrá un número arriba del valor del plano donde justamente se ha dividido y su número de elementos será la diferencia del número de elementos de la vieja caja menos el número de elementos encontrados hasta el plano donde se ha dividido.

Por otra parte en la vieja caja se cambiará el plano final que será el valor del plano donde se ha dividido la caja y su número de elementos será la suma de elementos



encontrados hasta el plano donde es dividido.

Continuando con el ejemplo, si en el plano 13 la suma de elementos obtenida es 31,897 y al sumar los elementos del plano 14 obtenemos un total de 35,449 esta suma habrá sobrepasado la mitad de los elementos contenidos en la caja, por lo que el plano 14 será el plano donde se hará el corte. Y las dos nuevas cajas obtenidas serán por lo tanto:

Si al estar sumando los elementos no se excede la mitad del total hasta que se añade el último plano (como en el caso donde el último plano contiene la parte principal de los datos) se debe tomar este último como el plano de corte. Por lo tanto este último plano formará una nueva caja.

Estas dos nuevas cajas vuelven a pasar por el proceso de reducción para eliminar los planos cero en cada una de ellas para posteriormente seleccionar aquella que contenga el mayor número de elementos para realizar sobre ella una "partición adaptativa" y poder obtener una nueva caja a partir de ésta, ésto es aplicado iterativamente hasta que "256" cajas son generadas o hasta que las cajas seleccionadas no contengan ningún color. Si en algún punto de la subdivisión, se intenta fraccionar una caja que contiene solamente un color (quizás muchas veces), la caja sobrante la cual probablemente no se ha usado puede ser reasignada para ser dividida y de esta manera dividir siempre la caja más grande que se pueda encontrar.

Después de que las cajas son generadas, se procede a calcular el color representativo para cada una de ellas y como resultante obtener el mapa de color de la imagen digitalizada.

El primer paso a seguir es recorrer cada una de las cajas y calcular la suma de los valores contenidos en ellas para cada componente de color (el componente rojo, el verde y el azul). El proceso será explicado mediante un ejemplo para esto se selecciona una caja con iteraciones mínimas pero que permita describir la secuencia del proceso claramente, esta caja se ilustra en la fig. 4.3.3.3

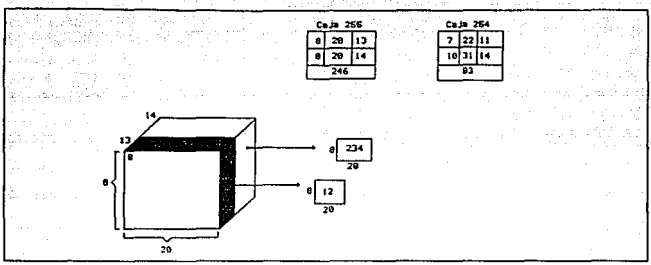


Fig. 4.3.3.3' Caja con Iteraciones Mínimas

Como se puede apreciar esta caja esta formada por un renglón y columna únicos cuyas coordenadas son (8,20), por los planos 13 y 14 y abarcando 246 elementos de la imagen de un total de 64,000. Con esta información se puede accesar directamente al cuboRGB para una coordenada en particular y obtener el número de veces que este color esta presente en la imagen, así suponiendo que al accesar el cuboRGB en la coordenada (8,20,13) tiene como valor un doce en dicha coordenada. Esta información indica que el color (8,20,13) esta presente doce veces en la imagen y con ello permite calcular el peso que dicho color esta ejerciendo sobre la misma, en cada uno de sus componentes cada una de las coordenadas forma la tríada (R,G,B), debido a que el objetivo es obtener el peso de los valores de color en esta caja, se debe ir acumulando la suma que cada coordenada de la caja proporcionará.

A manera que la coordenada (8,20,13) con un valor doce arroje los siguientes resultados:

suma roja = 12 * 8 = 96
suma verde = 12 * 20 = 240
suma azul = 12 * 13 = 156

De igual forma se accesa la siguiente coordenada de la caja, esto es la (8,20,14) con un valor de 234 (que es la diferencia de los 246 elementos en la caja menos los doce elementos encontrados en la coordenada [8,20,13]) :

suma roja = 234 * 8 = 1872
suma verde = 234 * 20 = 4680
suma azul = 234 * 13 = 3276

Más el acumulado del plano anterior :

suma roja = 1872 + 96 = 1968
suma verde = 4680 + 240 = 4920
suma azul = 3276 + 156 = 3432

Cabe aclarar que en este ejemplo sólo se recorrió la caja en el eje B para fines ilustrativos, pero el algoritmo siempre hace el recorrido sobre los tres ejes. Una vez que se han recorrido todas las coordenadas de la caja se procede a dividir cada una de las sumas de los componentes de color entre el número de elementos que contiene esta caja, para obtener de esta forma el promedio de cada componente que permitirá a su vez obtener el color representativo de esta caja. Esto es:

```
valor del componente    = suma del componente / número de elementos en la caja
valor componente rojo   = 1968 / 246   = 8
valor componente azul   = 4920 / 246   = 20
valor componente verde  = 3432 / 246   = 13
```

Para obtener finalmente el color representativo de esta caja se procede a realizar los cálculos descritos en el "manual del VGA", multiplicando por 30 el componente rojo, por 11 el componente azul y por 59 el componente verde, la suma de estos tres resultados indicarán el color seleccionado de los 262,144 colores posibles del VGA, así:

```
color seleccionado = (8*30) + (20*59) + (13*11) = 240 + 180 + 143 = 1563
```

Hasta este punto se ha conseguido obtener el color proporcionado por esta caja pero se debe recordar que para la fase siguiente en donde se debe mapear los colores originales a su más cercano representativo, se necesita el porcentaje de cada componente de color y no el color seleccionado, esto debido a que los colores originales de la imagen pueden únicamente ser obtenidos a partir de la imagen roja, verde y azul; las cuales nos proporcionan el porcentaje de cada componente y no el número de color (que finalmente es el objetivo de todo el proceso). Por otra parte para cargar los registros de color en el modo 13 hex, del VGA se necesita contar con el porcentaje de cada componente de color.

Así es necesario almacenar cada componente de color que cada caja proporciona en lugar del color representativo de la misma, sin embargo este ultimo servirá para ordenar ascendentemente las cajas encontradas así como los componentes de color de cada una de ellas. Estos dos arreglos ordenados son los que ocupara la siguiente fase. El pseudocódigo del proceso es el que se muestra a continuación:

```
PseudoCódigo.
      INICIO
indice = 0;
REPITE
 suma_rojo = 0;
 suma_azul = 0;
 suma_verde = 0;
 indice_rojo = cajas[indice].valor_bajo[0]
 REPITE
  indice_verde = cajas[indice].valor_bajo[1]
  REPITE
   indice_azul = cajas[indice].valor_bajo[2]
   REPITE
    num_color = cuboRGB[indice_rojo,indice_verde,indice_azul]
    suma_rojo = suma_rojo + indice_rojo * num_color
    suma_verde = suma_verde + indice_verde * num_color
    suma_azul = suma_azul + indice_azul * num_color
```

```
      índice_azul = índice_azul + 1
     HASTA índice_azul > cajas[índice].valor_alto[2]
     índice_verde = índice_verde + 1
    HASTA índice_verde > cajas[índice].valor_alto[1]
   índice_rojo = índice_rojo + 1
   HASTA índice_rojo > cajas[índice].valor_alto[0]
   valor_componente[índice].rojo = suma_rojo/cajas[índice].num_elementos
   valor_componente[índice].verde = suma_verde/cajas[índice].num_elementos
   valor_componente[índice].azul = suma_azul/cajas[índice].num_elementos
   índice = índice + 1
  HASTA índice = número_de_cajas_generadas
  i=0
  REPITE
   índices[i] = i
   peso_color[i] = valor_componente[i].rojo * 30 +valor_componente[i].verde * 59 +valor_com-
ponente[i].azul * 11
  HASTA i = número_de_cajas_generadas
  Ordenar_Indices_Ascendentemente_Por Peso_Color(índice,peso_color);
  /* Se obtienen los arreglos "cajas_ordenadas" y "registros_color"
    ordenados partiendo del arreglo "índices" el cual se encuentra ordenado */
  i = 0
  REPITE
 registro_color[i].rojo = valor_componente[índices[i]].rojo
  registro_color[i].verde = valor_componente[índices[i]].verde
  registro_color[i].azul = valor_componente[índices[i]].azul
  cajas_ordenadas[i].num_elementos = cajas[índices[i]].num_elementos
   k=0
   REPITE
    cajas_ordenadas[i].valor_alto[índices[k]].valor_alto[k]
    cajas_ordenadas[i].valor_bajo[índices[k]].valor_bajo[k]
    k=k+1
   HASTA K =3
  i=i+1
  HASTA i = número_cajas_generadas
 FIN
```

4.3.4 Fase 3: Mapear los colores originales al más cercano representativo en el mapa de color

Una vez que el mapa de color es generado, la siguiente parte consiste en determinar cual de los colores representativos corresponde a cada uno de los colores presentes en la imagen original, lo cual es una tarea relativamente sencilla, ya que basta llenar el cuboRGB con el número de índice de la caja que le corresponde; para realizar lo anterior se procede a recorrer cada una de las cajas ordenadas sobre los tres ejes de la misma, obteniendo una coordenada del cuboRGB, la que se colocara en el número de caja que se este accesando. Por otra parte, previamente cada localidad del cuboRGB deberá colocarse a un valor tal que no pueda ser un índice de una caja, de esta manera se podrá detectar cuando se ha alcanzado una parte del cuboRGB que no este incluida en una caja de color.

Este mapeo realizado permitirá a la siguiente fase redibujar la imagen original por un simple acceso a el cuboRGB. El pseudocódigo de este proceso es el que se muestra a continuación:

```
Función para mapear los colores originales a su más cercano representativo.
        INICIO
/* Se llena el cuboRGB a un valor tal que no sea un índice de caja. Esto es un número mayor a 256. */
  i = 0
  REPITE
   j = 0
   REPITE
    k = 0
    REPITE
      cuboRGb[i,j,k] = 512
      k = k + 1
    HASTA k > número_cajas_generadas
    j = j + 1
   HASTA j > número_cajas_generadas
   i = i + 1
  HASTA i > número_cajas_generadas
 /* Se llena el cuboRGB con el número de índice de la caja */
 i = 0
 REPITE
 rojo = cajas_ordenadas[i].valor_bajo[0]
  REPITE
   verde = cajas_ordenadas[i].valor_bajo[1]
   REPITE
    azul = cajas_ordenadas[i].valor_bajo[2]
    REPITE
      cuboRGb[rojo, verde, azul] = i
      azul = azul + 1
    HASTA azul > cajas_ordenadas[i].valor_alto[2]
    verde = verde + 1
   HASTA verde > cajas_ordenadas[i].valor_alto[1]
   rojo = rojo + 1
  HASTA rojo > cajas_ordenadas[i].valor_alto[0]
  i = i + 1
 HASTA i > número_cajas_generadas
 FIN
```

4.3.5 Fase 4: Cuantizar y redibujar la imagen.

Para cuantizar la imagen, simplemente se pasa cada pixel de la imagen original por el cuboRGB, creado durante la fase 3 y se escriben los valores del pixel dentro del frame buffer. Esto redibujará la imagen (cuantizada por supuesto) usando solamente los "n" colores que componen el mapa de color.

Dependiendo de la imagen, los errores de cuantización podrían ser obvios o invisibles. Las imágenes con altas frecuencias espaciales (tales como pelo o hierba) mostrarán

errores de cuantización mucho menores que imágenes con grandes areas lisas sombreadas (tales como rostros). El pseudocódigo de este proceso es el que a continuación se muestra :

```
Función par redibujar la imagen final
      INICIO
i = 0
REPITE
 valor_pixel = cuboRGB[imagen_rojo[i],imagen_verde[i],imagen_azul[i]]
 Imagen_Final[i] = valor_pixel
 i = i + 1
HASTA i > TOTAL_DE_PIXELS
FIN
```

4.3.6 Colores Optimos

Una mejora al redibujar la imagen consiste en medir la diferencia entre la imagen original y la imagen cuantizada (esto es el error total de cuantización). Aquí se utiliza una simple medida de color, la distancia cuadrada en el espacio RGB, la cual es elegida por su rapidez computacional y simplicidad. Esta es la función de distorsión o medida de color que mide la "diferencia" entre los correspondientes colores en la imagen original y la imagen final, definiendo la "óptima" cuantización (para una imagen y un mapa de color conocidos) como la única la cual minimiza la diferencia entre las dos imágenes.

El pseudocódigo que permite obtener los colores óptimos para una imagen dada es :

```
Función para redibujar la Imagen final con colores óptimos
      INICIO
i = 0
REPITE
 índice_correcto = 0
 mínimo_error = (imagen_rojo[i]-registros_color_ordenados[j].rojo) + (imagen_verde[i]-
registros_color_ordenados[j].verde) + (imagen_azul[i]-registros_color_ordenados[j].azul) +
/* Se busca en todos los registros de color para encontrar cual tiene el más pequeño error cuando este es
usado, para este pixel */
 j = 0
 REPITE
  error_registro = (imagen_rojo[i]-registros_color_ordenados[j].rojo) + (imagen_verde[i]-
registros_color_ordenados[j].verde) + (imagen_azul[i]-registros_color_ordenados[j].azul) +
    IF error_mínimo  mínimo_error
      mínimo_error = error_registro
      índice_correcto = k
    ENDIF
    j = j + 1
  HASTA j > número_cajas_generadas
  Imagen_Final[i] = índice_correcto
  i = i + 1
HASTA i  > TOTAL_DE_PIXELS
FIN
```

4.3.7 "Dithering"

La estrategia básica del "dithering" es negociar o cambiar la resolución de brillantez por la resolución espacial. Por el calculo de la media de brillantes de los distintos pixels vecinos uno puede obtener colores no representados por el mapa de color. Si la resolución del frame buffer es bastante alta, el ojo humano hará la mezcla espacial para el observador.

La técnica de "dithering" utilizada es debida a 'Floyd and Steinberg'. Su algoritmo compensa el error de cuantización introducido en cada uno de los pixels antes de la propagación de este a su vecino. La propagación es dirigida solamente a los pixels abajo, a la derecha y abajo a la derecha del "actual pixel".

El pseudocódigo para cuantizar y realizar el "dithering" en una imagen a color usando este algoritmo es el siguiente :

```
Función para realizar dithering sobre una imagen
    INICIO
i = 0
REPITE
/* se lee un color para encontrar el más cercano y representativo*/
   índice = cuboRGB[imagen_rojo[i],imagen_verde[i],imagen_azul[i]]
/* se dibuja la imagen cuantizada */
   Imagen_Final[i] = índice
/* se calcula el error de cuantización */
   error_rojo = registros_color_ordenados[índice_correcto].Rojo - Rojo[índice]
   error_verde = registros_color_ordenados[índice_correcto].Verde - Verde[índice]
   error_azul = registros_color_ordenados[índice_correcto].Azul - Azul[índice]
/* Se distribuye el error en tres direcciones de la imagen */
/*Si no se esta en la ultima columna de la imagen y no se ha abarcado todas las líneas se difunde 3/8 de error
a el pixel a nuestra derecha */
   IF ((i+200)/TOTAL_LINEAS) TO  \L_COLUMNAS
     Rojo[i+TOTAL_LINEAS] = Rojo[i+TOTAL_LINEAS] + (error_rojo*3)/8;
     Azul[i+TOTAL_LINEAS] = Azul[i+TOTAL_LINEAS] + (error_azul*3)/8;
     Verde[i+TOTAL_LINEAS] = Verde[i+TOTAL_LINEAS] + (error_verde*3)/8;
   ENDIF
/* si no se esta en la ultima línea, se difunde 3/8 de error al el pixel bajo */
   IF (i modulo TOTAL_LINEAS  TOTAL_LINEAS-1)
     Rojo[i+1] = Rojo[i+1] + (error_rojo*3)/8;
     Azul[i+1] = Azul[i+1] + (error_azul*3)/8;
     Verde[i+1] = Verde[i+1] + (error_verde*3)/8;
   ENDIF
/* Si no estamos en el ultimo renglón y no estamos en la ultima columna de la imagen, se difunde 1/4 de error
a el pixel abajo a la derecha */
   IF(((i+TOTAL_LINEAS) / TOTAL_LINEAS)  NCOLS) AND (i % TOTAL_LINEAS  TOTAL_LINEAS-1))
     Rojo[i+1+TOTAL_LINEAS] = Rojo[i+1+TOTAL_LINEAS] + error_rojo/4;
     Azul[i+1+TOTAL_LINEAS] = Azul[i+1+TOTAL_LINEAS] + error_azul/4;
     Verde[i+1+TOTAL_LINEAS] = Verde[i+1+TOTAL_LINEAS] + error_verde/4;
   ENDIF
   i = i+1
```

```
 HASTA i > TOTAL_DE_PIXELS
FIN
```

Imagen con Aspect Ratio Incorrecto

Imagen con Aspect Ratio Corregido

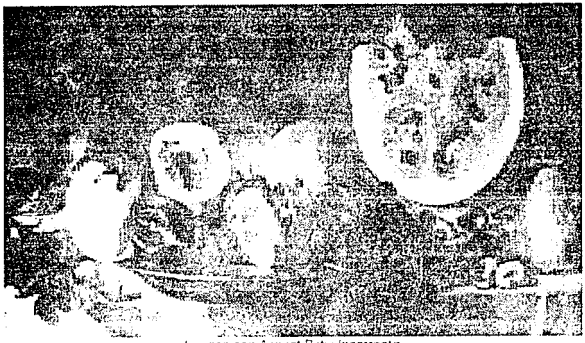


Imagen con Aspect Ratio Incorrecto

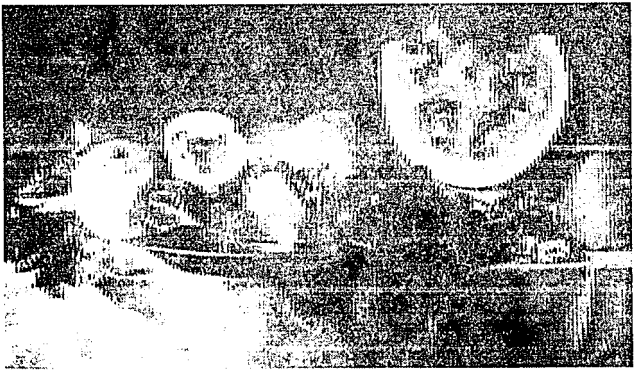


Imagen con Aspect Ratio Corregido

Imagen con Aspect Ratio Incorrecto

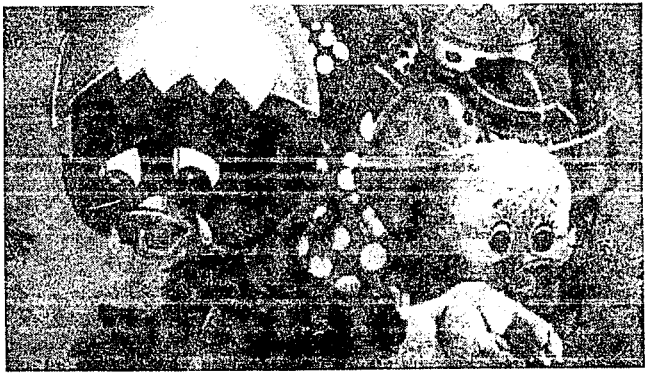
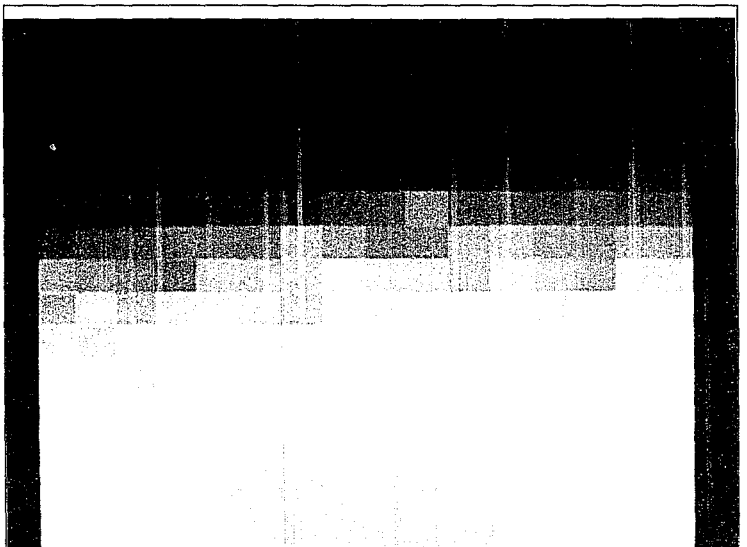


Imagen con Aspect Ratio Corregido

Imagen sin corrección de Aspect Ratio y sin Dithering

Aspect Ratio corregido, Dithering y Colores Optimos

Aspect Ratio corregido, Dithering y Colores Optimos



Paleta representativa de la Imagen

Capítulo 5

Sistema Administrador

Introducción

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 capitulo se presenta una descripción del administrador general.

Hoy en día en los sistemas de aplicación de computadoras tienden claramente a la utilización de graficos dejando atras, las interaciones con el usuario en modo texto, ya que el humano tiende a pensar en terminos de imagenes y no en terminos de texto. Una interface grafica de usuario (GUI .- Graphical user interface) hace que fácilmente un usuario final interactue con el sistema operativo y con las aplicaciones del programa; especialmente cuando, la esencia de las funciones de un programa estan orientadas a graficos.

5.1 Parámetros en el diseño de la presentación gráfica

En la actualidad el abaratamiento y adelantos en al campo del hardware ha traido como una consecuencia el cambio de los parametros para evaluar la calidad del software: La definición de un producto como utilizable, aún en la etapa de diseño y prueba, es que el producto permita al usuario concentrase en la tarea que está haciendo o en la obtención de un objetivo, y no en la forma de utilización del paquete de computación para llevar a cabo la tarea requerida. (Mary Deieli de "Usability Laboratory of the user interface Architecture")

Los expertos en el diseño de interfases gráficas suguieren que los siguientes puntos sean tomados en cuenta al diseñar las mismas :

-Con la introducción del ambiente Windows la utilización de las interfaces gráficas ha recibido gran impulso (a pesar de que este conceptó no es nuevo, pues Apple lo ha explotado por años), presentándose con esto un nuevo fenómeno: los programadores conocen de lenguajes de programación de computación, de utilerías, de análisis y diseño de sistemas, etc., pero no saben nada de percepción visual, percepción auditiva, puntos de atención, teoría del color, comunicación visual y diseño gráfico, por lo que al crear un nuevo producto cometen errores, grandes o pequeños, al establecer las pantallas de interacción entre el usuario y el paquete. En esta era de interfaces amigables es

conveniente tener un diseñador industrial o gráfico que dé su valorización del producto en la etapa de prototipo.

-La utilización de los diferentes tonos de gris en el diseño de la interfaz gráfica, para dar la impresión de tres dimensiones y de movimiento en los botones. Esto obedece a la idea de hacer que los elementos en la pantalla, aún en la primera ocasión en que un usuario trabaja con ella, no le sean del todo extraños o quizás que sean cercanos a su realidad. Cuando se presenta en la pantalla una imagen tridimensional de un botón (utilizando diferentes tonos de gris con lo que el botón en la pantalla realmente da la impresión de sobresalir), al dar un "clic" el botón se sume y vuelve a salir, esto da una impresión muy cercana al mundo real y la persona está segura de que su elección sí será ejecutada.

En las pantallas actuales solamente tenemos 16 colores y el único que podemos utilizar en el efecto de tres dimensiones es el gris, aunque en realidad se utiliza el blanco, gris claro, gris oscuro y negro.

-Al elegir los colores de las interfases debemos ser meticulosos, pues las personas son emocionales ante los colores, casi todo el mundo acepta el azul y el rojo, pero fuera de estos colores, se tienen diferentes opiniones, incluso a causa de diferencias culturales. Se debe buscar colores para los fondos de las pantallas que permitan ver perfectamente los textos, así como pensar en colores que no fatiguen al usuario, es mejor lo sencillo y directo que dé al área de trabajo comodidad de movimiento. Si una pantalla tiene muchos colores la vista estará en movimiento todo el tiempo y hará que al usuario le sea más difícil concentrarse y reaccionar.

-Para los iconos se busca acercarse a la realidad y los objetos representados deben tener colores naturales, una pluma azúl, un cuaderno de notas gris, etc.

Con estos puntos en mente, para el desarrollo del administrador del sistema se planteo desde el inicio un software de utilización sencilla, intuitiva, de una apariencia agradable a la vista y con la llegada del ambiente de windows, la presentación de iconos y ventanas faciles de operar.

5.2 Presentación

La apariencia del sistema administrador es sencilla y clara, este divide al monitor de la computadora en solo dos secciones:

- AREA DE BOTONES.- Aquí se presentan todas las opciones con las que cuenta el sistema, el usuario solo debe colocar el cursor del mouse sobre cualquiera de estos y presionar el boton izquierdo del mismo para ejecutar la función que tal boton tiene asociada. El boton seleccionado se oprime y permanece en este estado hasta que la función seleccionada termine.

- AREA DE DESPLIEGUE .- En esta área se presenta las partes que forman el proceso de color cuando una imagen esta siendo digitalizada, así como el avance que dicho proceso lleva.

Esta área ademas, esta diseñada para que en un futuro la imagen a color pueda ser visualizada sobre la misma. Cabe recordar que en este momento el sistema esta diseñado para trabajar con un monitor VGA estandar, que cuenta con un solo modo a 256 colores,

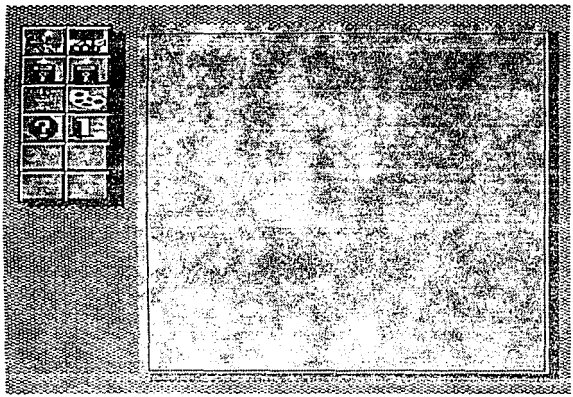


Fig. 5.1 Sistema Administrador

el cual es necesario para desplegar una imagen con colores reales, y como una mejora al sistema este será ampliado para trabajar con los nuevos modos a 256 colores que proporcionan actualmente los monitores superVGA o monitores no estandares en el mercado. La figura 5.1 muestra la distribución de la pantalla.

5.3 Módulo Iconos

Los iconos son simbolos gráficos usados sobre menus de sistemas computacionales, ventanas y pantallas. Estos representan las capacidades del sistema que pueden ser usadas por el operador. En la industria de la computación el termino icono es frecuente-

mente usado como sinonimo de un simbolo visual. Un icono es utilizado por las siguientes razones :

- Por una busqueda rapida .- Cada simbolo visual tiene una distinta forma, que puede ser reconocida instantaneamente.
- Por una reconocimiento inmediato .- La unicidad de la forma de un simbolo asegura que una vez que este es aprendido, este es recordado fácilmente y reconocido inmediatamente.
- Por una mejor forma de recordar .- Los simbolos visuales son faciles de aprender, estos son más fáciles de retener que las palabras, porque ellos son visualmente más distinguibles uno de otro.
- Por conceptos gráficos .- Describir conceptos gráficos u operaciones fisicas en palabras es una manera indirecta y poco natural. Ideas graficas necesitan etiquetas graficas.
- Por un atractivo visual.

La programación orientada a objetos (OOP) es un concepto muy simple pero poderoso. Apesar de que las especificaciones de una aplicación orientada a objetos puede variar grandemente, existen distintos principios generales que todos los sistemas de OOP tienen en común.

El principio fundamental en la OOP es que la información esta agrupada en entidades llamadas objetos. Cada objeto usualmente tiene una función especial asociada con este. Además, los objetos se comunican entre si a través de mensajes. El factor que distingue a la OOP, es que las acciones son accesadas a través de los objetos. Esto contrasta con los tradicionales metodos de programación donde los procedimientos y funciones son usadas para accesar datos y realizar acciones.

Existen distintas ventajas en la OOP. Primero, esta puede ser usada para ocultar cualquier procesamiento especial que un conjunto de datos requiere, de esta manera todos los objetos efectivamente se ven iguales desde un nivel alto. En adición la OOP puede hacer que los programas seán más entendibles y fáciles de mantener. Estos son las dos principales razones por la que un estilo de OOP es usado en un ambiente grafico.

En un ambiente gráfico interactivo, los iconos, texto, menus y figuras gráficas pueden ser todas manipuladas como objetos. Estos objetos usualmente tienen una función asociada con la misma que es ejecutada cuando el objeto es seleccionado. Los objetos son seleccionados mediante un dispositivo tal como el mouse. (un objeto es seleccionado por un "click" del boton izquierdo del mouse sobre el mismo).

Internamente, los objetos son mantenidos en una lista, cuando una región de la pantalla es seleccionada con el mouse, la lista es checada para ver si un objeto esta localizado en la posición seleccionada. Si es así la función asociada al objeto es ejecutada. Por ejemplo, si se realiza un "click" del boton izquierdo del mouse sobre el icono escritura, la función salva archivo es invocada. La figura 5.2 provee una vista grafica de este proceso.

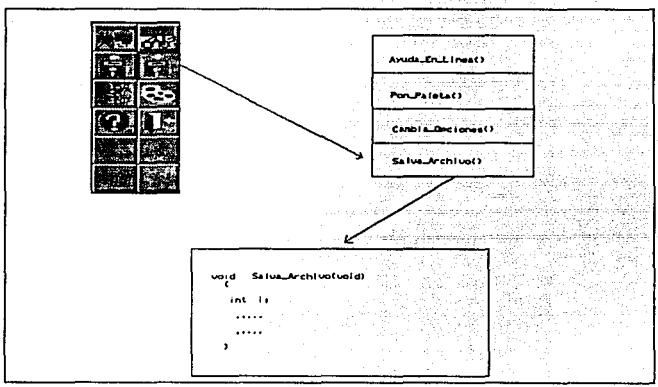


Fig. 5.2 Selección de un Objeto

5.4 Módulo ventanas

Este módulo permite al sistema la utilización de ventanas pop-up, las cuales pueden ser usadas para mandar mensajes al usuario, crear menus pull-down y permitir otras formas de entrada y salida a un programa. El proceso para crear una ventana pop-up es mostrado en la figura 5.3. Como lo muestra la figura, la idea basica es salvar la región de la pantalla donde será colocada la ventana pop-up asi como cualquier parametro del dibujo que podria ser modificado, de tal forma que esta información pueda ser usada para restaurar la pantalla a su estado que tenia antes que la ventana pop-up fuera creada. Para salvar esta información, un stack es utilizado lo cual permite construir capas de ventanas pop-up. La desventaja de utilizar un stack,es que las ventanas deberán ser removidas en orden inverso a como fueron creadas. Para remover una ventana pop-up desde la pantalla es necesario realizar los siguientes dos pasos:

1.- Redibujar la ventana pop-up con la imagen almacenada de la pantalla, realizando un pop al stack.

2.- Restaurar los parametros de la pantalla que fueron salvados en el stack. Despues de este proceso, la pantalla mostrara su estado original como si la ventana pop-up nunca hubiera existido.

La ventaja de utilizar ventanas pop-up es que no se necesita una región estática de la pantalla para mensajes enviados por el sistema al usuario, así como para la entrada y

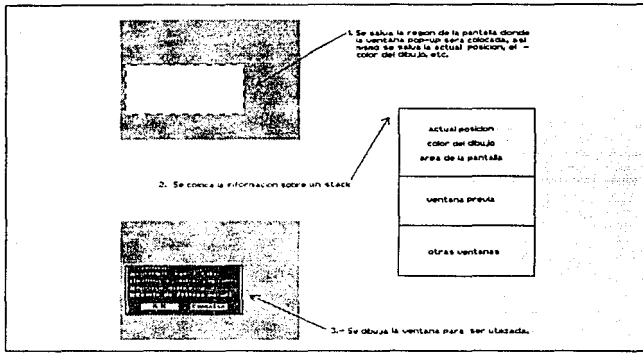


Fig. 5.3 Pasos en la Creación de una Ventana Pop-Up

salida de datos. La desventaja es obvia y es la cantidad de memoria requerida para salvar las regiones de la pantalla, pero la espectacularidad que da la apariencia de una ventana pop-up, bien vale la pena.

5.5 Módulo mouse.

Apesar de que el teclado es un invaluable dispositivo de entrada, este no es lo más aconsejable para programas gráficos interactivos. Las características de posicionamiento del cursor y la habilidad para seleccionar localidades aleatorias sobre la pantalla, las cuales son operaciones típicamente requeridas en un ambiente gráfico interactivo y hacen del mouse un mejor dispositivo de entrada Un sistema de mouse actualmente consiste de dos esenciales elementos: los mecanismos del mouse y un programa residente en memoria llamado el mouse driver. El mouse driver provee de todos los soportes de bajo nivel necesitados para comunicarse con el mouse, este es responsable de mantener automaticamente el cursor del mouse en cualquier posición de la pantalla y detectar cuando un boton es presionado.

Para accesar las distintas caracteristicas del mouse y del mouse driver se utiliza la interrupción 33h, los servicios del mouse driver llaman a esta interrupción la cual

redirecciona a su vez las funciones de bajo nivel del mouse. La función especifica es seleccionada a través del valor en el registro AX cuando la interrupción es llamada. Otros tres registros BX, CX y DX son usados para pasar parametros a las rutinas del mouse.

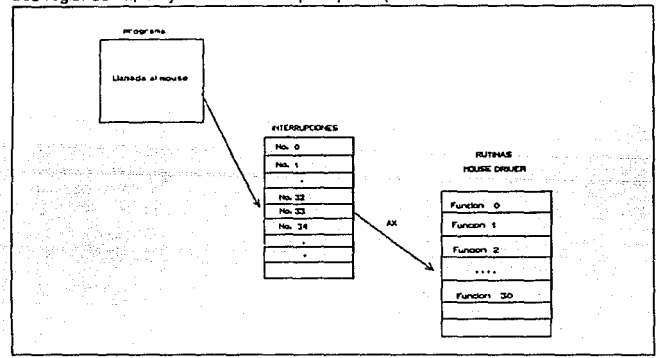


Fig. 5.4 Secuencia al Invocar una Función del Mouse

Similarmente, las funciones del mouse utilizan estos registros para retornar valores a las funciones de llamada, como la posición del mouse y el status de los botones del mouse entre otros. La figura 5.4 muestra como un programa usa la interrupción 33h para invocar una función del mouse.

5.6 Módulo texto

Este módulo contiene algunas rutinas para el manejo de texto en modo gráfico, principalmente para la entrada de caracteres con eco en la pantalla. Un punto extremadamente importante es que en el modo gráfico, el caracter backspace y distintos caracteres de control no conservan su funcionamiento. De tal forma que se debe de implementar su funcionamiento, así cuando el caracter backspace es introducido este no se toma como caracter, en su lugar, provoca que el ultimo caracter en la cadena sea reescrito con el color del fondo con lo cual efectivamente se borra este y despues de esto la actual posición es movida una caracter a la izquierda. En adición en el modo gráfico no existe cursor, el cual es implementado mediante el caracter underline, el cual aparece mientras se esta introduciendo texto hasta que el caracter return es introducido.

5.7 El icono digitalizador

Este icono permite digitalizar una imagen, para lo cual es necesario contar con el siguiente equipo:

-Camara de video monocromatica.

-Tarjeta digitalizadora DVI/1.

-Filtros rojo, verde y azul.

-Opcionalmente la tarjeta controladora de filtros.

El tiempo de proceso para obtener una imagen en colores reales depende de las opciones marcadas en el ICONO OPCIONES.

Cuando este icono es seleccionado el sistema administrador presenta la pantalla de la figura 5.5. Esencialmente tres procesos se distinguen al adquirir la imagen con cada uno de los filtros : capturar la imagen, filtrar la imagen y corregir el aspect ratio de cada una de ellas, obteniendo de esta manera tres imagenes que seran la entrada del proceso que

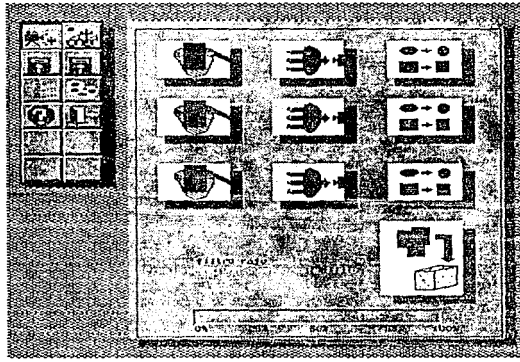


Fig. 5.5 Proceso de Adquisición de la Imagen

obtiene la imagen resultante a color. Conforme cada proceso es realizado, la imagen representativa asociada se desvanece de la pantalla como una indicación hacia el usuario. Por otra parte se tiene un indice del porcentaje de avance del proceso en general.

Cuando la imagen a color se encuentra lista para ser visualizada el icono digitalizador regresa a su posición inicial y el curso "mano" del mouse se coloca en el icono vista.

5.8 Icono vista

Se utiliza este icono para visualizar la imagen,que se obtuvo con el ICONO DIGITALIZADOR o bien con la lectura de alguna imagen en disco mediante el ICONO LECTURA. La imagen presentada se encuentra en el modo 320x200 a 256 colores, por lo cual al momento de presionar este icono la pantalla del administrador desaparecera, para que la imagen pueda visualizarse sobre la pantalla.

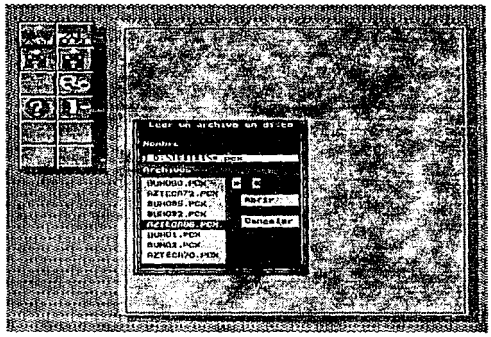


Fig. 5.6 Pantalla que Genera el Icono Lectura

Para regresar a la pantalla principal basta con presionar ESC o cualquier boton del mouse.

5.9 Icono lectura

Mediante este icono podemos seleccionar una imagen almacenada ya sea en disco duro o en alguna unidad de disco flexible. Al entrar se presentan todas las imagenes en formato PCX que se encuentren en la unidad desde donde se corre el programa. En caso de querer cambiar de unidad o directorio presionar el boton izquierdo del mouse donde se esta visualizando la unidad por default, al llevar a cabo esta acción se pide un nombre de archivo o una ruta que indique algun directorio de donde se desea se obtenga la lista

de archivos a seleccionar. La lista de archivos visualiza solamente ocho archivos a la vez para obtener los siguientes archivos presionar el boton izquierdo del mouse sobre ">" para ir hacia adelante de la lista o presionar el boton izquierdo del mouse sobre "<" para ir hacia atras. Para seleccionar algun archivo colocarse sobre el nombre de este y presionar el boton izquierdo del mouse, esta acción iluminara o invertira la región donde se encuentra el nombre del archivo.

Para cargar la imagen una vez seleccionado el archivo deseado, presionar el boton izquierdo del mouse sobre la región ABRIR, en este momento aparecera un reloj sobre la pantalla indicando que el archivo se esta leyendo de disco. Una vez completada la lectura y si esta tiene exito, la región de lectura de archivos desaparecera y el icono de la mano para la selección es visualizada. En caso de haber error en la lectura el problema encontrado es indicado y se debera presionar el boton izquierdo del mouse sobre la región OK de la pantalla.

Para salir sin seleccionar ningun archivo presionar la tecla ESC o el boton izquierdo del mouse sobre la región CANCELAR. La figura 5.6 muestra la pantalla generada por este icono.

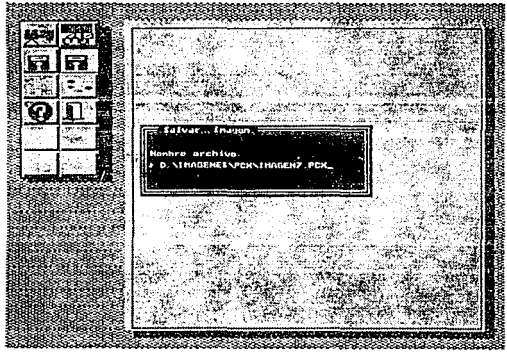


Fig. 5.7 Pantalla del Icono Escritura

5.10 Icono escritura

Este icono permite salvar la imagen que se encuentra activa esto es la imagen que podemos visualizar con el ICONO VISTA. Al entrar se pide un nombre de archivo, este puede tener extensión o no tenerla. En caso de no proporcionar una extensión el archivo es salvado como .PCX, ademas este nombre puede incluir una ruta completa de donde se desee se almacene el archivo, por ejemplo:

C:/DIGITAL/IMAGENES/IMAGEN1.PCX

En caso de solo proporcionar el nombre, el archivo es almacenado en el directorio por defecto. Para salir sin salvar el archivo presionar la tecla ESC. La figura 5.7 muestra una típica pantalla de interacción con el usuario de este icono.

5.11 Icono opciones

Se deberá de utilizar este icono para indicar como se desea la imagen resultante. Para marcar o desmarcar cada una de las opciones colocarse en la opcion deseada y presionar el boton izquierdo del mouse. El significado de cada una de las opciones es el siguiente :

- Corregir aspect ratio .- Si esta opción no se encuentra marcada la imagen resultante se verá alargada. Esto es un circulo aparecera como una elipse y un cuadrado como un rectangulo.

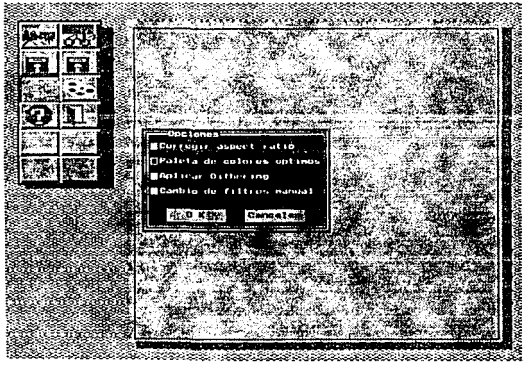


Fig. 5.8 Pantalla Generada por Icono Opciones

- Paleta de colores optimos .- Si esta opción se encuentra marcada la imagen resultante tendra una calidad excelente en cuanto al color, pero el tiempo para obtener la misma oscilara entre 20 y 30 minutos.
- Aplicar Dithering .- Con esta opción marcada la imagen presentara un mejor colorido y brillantes.
- Cambio manual de filtros .- Esta opción esta presente para el caso en que no se cuente con el controlador de cambio de filtros, si esta opción se encuentra marcada el sistema administrador solicitará los filtros como los vaya requiriendo, al adquirir la imagen.

La figura 5.8 muestra la pantalla generada por el icono opciones.

5.12 Icono Paleta

Mediante este icono podemos visualizar la paleta generada, que es representativa de la imagen digitalizada, esto es la paleta es dinamica y varia dependiendo de los colores encontrados sobre la imagen digitalizada. Esta paleta es visualizada en el modo 320x200, por lo que al presentar este icono la pantalla del administrador desaparecera. Para salir presionar ESC o cualquier boton del mouse.

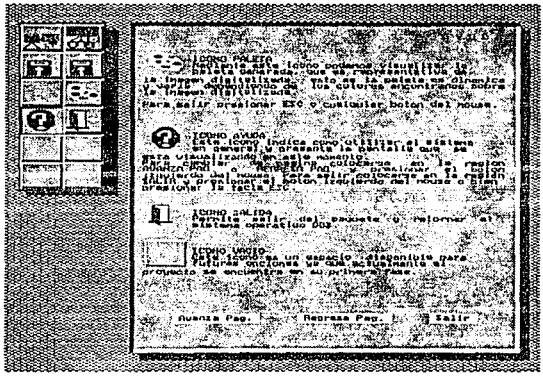


Fig. 5.9 Icono Ayuda en Línea

5.13 Icono ayuda

Este icono permite accesar información de ayuda para cada uno de los comandos o iconos que forman el sistema. Esta ayuda tambien puede ser accesada através de la tecla F1, que aunque no es un estandard, es comúnmente aceptada como la tecla de "ayuda".

Para cambiar de pagina colocarse en la región AVANZA PAG. o REGRESA PAG. y presionar el boton izquierdo del mouse. Para salir colocarse en la región SALIR y presionar el boton izquierdo del mouse o bien presionar la tecla ESC. Una pagina de la pantalla generada por el icono ayuda es mostrada en la figura 5.9.

5.14 Icono salida

Permite salir del paquete y retornar al sistema operativo DOS.

5.15 Icono vacio

Este icono es un espacio disponible para futuras opciones ya que actualmente el proyecto se encuentra en su primera fase. En la parte final del proyecto será relativamente sencillo el adicionar nuevos módulos de código que contendran algoritmos de procesamiento o algoritmos de prueba 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. La figura 5.10 muestra un diagrama a bloques del actual sistema en donde se visualiza como cada uno de los módulos estan interactuando entre sí.

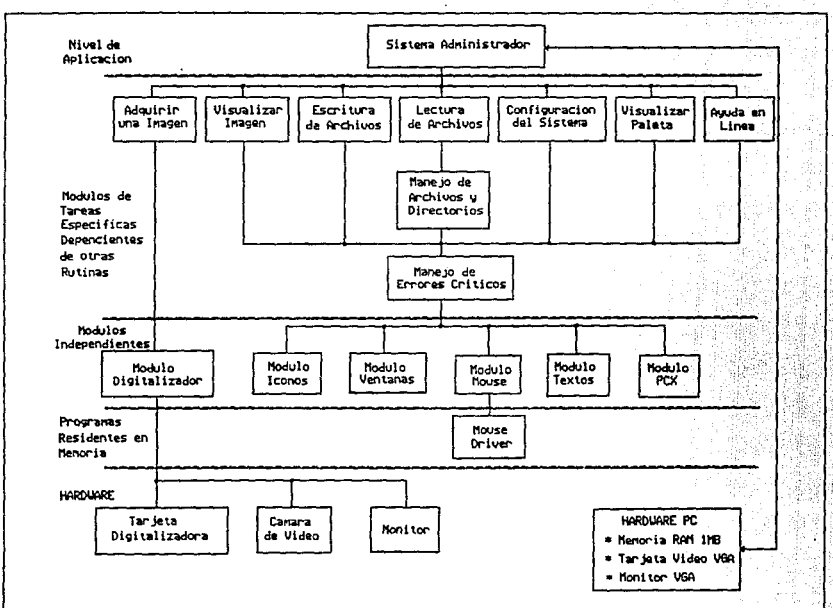


Fig. 5.10 Diagrama a Bloques del Clasificador de Color

Conclusiones

El presente trabajo en primera instancia servirá primordialmente en el área de la investigación científica, como una herramienta capaz de obtener y fijar de manera eficiente y "económica" parámetros cuantitativos de color para diagnósticos microbiológicos de la rama farmacéutica.

El desarrollo del mismo involucró diferentes áreas fundamentales de trabajo, por un lado conceptos relacionados con procesamiento de imágenes, fotografía y modelos de color. Por el otro, los de computación enfocados principalmente a aquellos que permiten la elaboración de algoritmos, sin importar si se trata de hardware y/o software, capaces de ayudar a obtener un clasificador de color.

El clasificador de color utilizando técnicas de procesamiento de imagénes, se llevo a cabo con el fin de automatizar procesos que involucren el análisis de imagénes para su posterior uso, tal es el caso de la automatización de pruebas bioquímicas aplicables en el laboratorio de análisis clínicos bacteriológicos, donde los resultados son obtenidos en base a los colores que determinada sustancia presenta a través del tiempo. Los distintos colores que la sustancia va tomando se van registrando para posteriormente comparando con tablas ya definidas poder determinar que bacteria o virus se encuentra presente.

Estos registros son tomados por una persona en distintos intervalos de tiempo y de acuerdo a su criterio determina el color que presenta la sustancia.

Así el sistema desarrollado es útil por :

1) Permite desplegar la imagen resultante sobre el monitor de la computadora, con lo cual se puede determinar que tan cercana es la imagen obtenida con la imagen original, ya que finalmente el ojo humano será el juez final para determinar la calidad de las imágenes obtenidas. Esto a su vez permitirá modificar el medio ambiente que rodea al objeto que se esta digitalizando, tal como la intensidad de la luz, la apertura del diafragma de la cámara, el enfoque sobre el objeto de interés entre otros, con el fin de mejorar la calidad de la misma.

2)Hace posible el almacenamiento de la imagen en un instante dado en formato PCX para su posterior consulta o manipulación en una forma compacta y transportable a otros sistemas.

3)Presenta grandes perpectivas como una herramienta que permita facilitar la clasificación de color para analizar determinados patrones, suministrados como una base de datos que formarán la "inteligencia" o conocimientos previos de la computadora.

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

1)Los algoritmos aquí presentados fueron desarrollados tomando en cuenta los métodos y disciplinas de la ingenieria de software.

2)Considerando los recursos de hardware con los que el sistema cuenta, el clasificador de color resulta rápido, eficiente y bastante económico. El índice de error dependerá en gran medida del ambiente que rodea la imagen a digitalizar, su importancia dependera de la presición que una determinada aplicación requiera.

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.

4)Por la gran cantidad de memoria requerida para la manipulación de imágenes, el sistema operativo MS-DOS hasta hace poco no era aconsejable para el procesamiento digital de imágenes, principalmente por la limitante de manejar hasta 1MB de memoria en modo real, pero utilizando los conceptos de memoria expandida y extendida y dispositivos que permiten manejar ésta en el modo real del MS-DOS y con la llegada del ambiente windows; el sistema MS-DOS por su popularidad y facilidad de manejo, comparado con los grandes requerimientos y complejidades que sistemas como OS/2 o UNIX necesitan, es nuevamente digno de tomarse en cuenta.

5)La utilización del compilador Turbo C + + de Borland no es lo más recomendable, principalmente por no accesar más alla de 1MB de memoria RAM. En su lugar la utilización de compiladores de lenguajes de programación bajo el ambiente de windows tales como Microsoft C o Borland C + + es lo más recomendable, ya que estos al utilizar el ambiente windows permiten la libertad de manejar más alla de 1MB en RAM. Por otro lado proporcionan rutinas para una interfaz gráfica más estandar, lo cual evita la programación de las mismas.

6)El sistema facilita su uso al contar con un drive para mouse y HELP integrado, además cuenta con manejo de errores contra posibles malos usos, haciendolo accesible a usuarios sin grandes conocimientos en computación.

Por otra parte para mejorar el rendimiento del sistema se aconseja:

1)Utilizar un monitor y tarjeta gráfica con una mayor resolución en los modos a 256 colores, tales como el adaptador gráfico SuperVGA, VGAWonder o ProDesigner VGA Plus que ofrecen una resolución de 800x600 a 256 colores. Entre los beneficios de utilizar tal hardware se encuentra:

- Un mejor aspecto .- Actualmente las imagénes desplegadas presentan el problema de la pixelización inherente al modo de resolución de 320x200, al contar con una resolución mayor como la de 800x600 a 256 colores de los adaptadores antes

mencionados, se puede desplegar además la imagen completa almacenada en el banco de memoria de la tarjeta digitalizadora; esto es visualizaremos una imagen de 512x480, lo cual proporcionara mayor nitidez en la misma.

- Velocidad al adquirir la imagen .- Al contar con un monitor de mayor resolución, la correción del aspect ratio es eliminada, ya que se puede tomar ventaja de que la imagen en la tarjeta digitalizadora tiene pixels de 1:1 y el aspect ratio de los adaptadores antes mecionados es de 1.33, lo cual nos proporciona pixels cuadrados.

2) Puesto que una imagen internamente esta representada mediante números, la utilización de un coprocesador matemático incrementara la rapidez de las operaciones realizadas sobre las mismas.

Glosario

ADAPTADOR GRAFICO. Es la circuiteria electrónica diseñada para lograr la interfase de el monitor con el sistema de computo. Usualmente localizada en una tarjeta de circuito impreso.

ADC. Ver convertidor Analógico/Digital.

ARCHIVO. Un conjunto de información almacenada en disco. Existen diferentes tipos de archivos, los cuales utilizan diferentes extensiones.

ARGUMENTO. Es el dato pasado a una función o a un programa.

ASPECT RATIO. Es el radio de ancho a alto de un solo pixel sobre el display. Si el display tiene un 'aspect ratio' de 1, se dice que tiene pixels cuadrados.

BIOS.(Basic Input/Output System) Son rutinas codificadas a nivel 'firmware' contenídas dentro de la memoria ROM localizada en la tarjeta del adaptador de display que contiene la información necesaria para decodificar las instrucciones del CPU.

BUFFER DE IMAGEN. Este termino también es conocido como buffer de página, "frame buffer" o memoria de video. Es la RAM usada por el adaptador gráfico para almacenar una pantalla de datos. El tamaño de este buffer determina la resolución máxima desplegable.

CAMARA CCD. Una camara de video que contiene un dispositivo de pareja cargado ('Charged-Coupled Device'), el cual es un censor transistorizado lumínico sobre un circuito integrado. Los CCD's son más pequeños y más duros que las antiguas cámaras de video que utilizaban un tubo de vidicon como censor luminoso.

CCITT. El CCITT (International Telegraph an Telephone Consultative Committe) es una organización internacional que desarrolla estandares en la industria de las comunicaciones.

CGA (Color Graphics Adapter). Estandar gráfico digital basado en lógica TTL desarrollado por IBM, baja resolución y falsos colores.El CGA tiene un máximo de 16 colores disponibles.

CMYK (Cyan - Magenta - Yellow - Black). Los colores primitivos usados para crear la separación de 4 colores. El modelo de color CMYK ha sido propuesto para PC's como una opción del formato RGB más comunmente usado.

COLORES FALSOS. Ver falsos colores.

COLORES VERDADEROS .Este concepto se refiere al despliegue de colores lo más natural posible, proporcionando el archivo de la imagen original que fue capturada con el equipo adecuado.Esencialmente, los colores primarios R, G y B individualmente tienen un mínimo de 256 sombras cada uno.

COMPRESION. Son diversas técnicas usadas por dispositivos y software de imagenes, para reducir los requerimientos de almacenamiento sin afectar drasticamente la apariencia de la imagen.

CONVERTIDOR ANALOGICO/DIGITAL. Es el dispositivo que censa una señal analógica y la convierte a su correspondiente representación digital.

CONVERTIDOR DIGITAL /ANALOGICO. Es el dispositivo que recibe una señal digital y la convierte a su correspondiente señal analógica.

CRT. Tubo de Rayos Catódicos; dispositivo de despliegue para una computadora o TV.

CUANTIZACION. Es el proceso de mapear los valores de brillo y/o color a numeros enteros. Los convertidores A/D ejecutan este proceso de cuantización.

DAC. Ver convertidor analógico digital.

DEFAULT. Es el valor que se asume si no se especifica otra cosa.

DIGITALIZADOR. Es un dispositivo capaz de capturar los datos de una imagen, convirtiendo una imagen en tonos-continuos a su correspondiente representación discreta.

DITHERING. Es el proceso de combinar pixels adyacentes en grupos para incrementar la resolución de color. Los algoritmos de "dithering" son muchos y variados; se utilizan para expander la resolución de color sobre sistemas con capacidades de color limitadas. Donde solo los colores blanco y negro estan disponibles, el "dithering" puede producir tonos de gris conocidos también como escala de grises.

EGA (Enhanced Graphics Adapter). Es un adaptador gráfico estandar de media resolución desarrollado por IBM, el cual puede desplegar hasta 16 colores simultaneos partiendo de una paleta de 64 colores.

ENTRELAZADO. Debido a que un display de computadora es conceptualizado como una pila de líneas horizontales, el termino de 'entrelazado' se refiere a una imagen generada por un adaptador gráfico que despliega primero todas las líneas pares y luego las impares. Esto significa que una mitad de datos de la imagen se despliega en la primera pasada y la otra mitad en la segunda pasada. A cada grupo de líneas se les denomina "campo".

ESCALA. Es lo que nos permite incrementar o decrementar el tamaño de una imagen.

ESCALA DE GRIS. Ver Niveles de gris.

ESCALAR. Es la translación de las coordenadas de pixels para manipular el tamaño de de una imagen. Se ejecuta mediante multiplicaciones de las coordenadas del pixel por un factor de escalamiento, el cual puede ser diferente para 'x' y 'y'.

FALSOS COLORES. Es un metodo para desplegar gráficas de computadora, utilizando pixels de menos de 8 bits (usualmente de dos o cuatro). El termino 'falsos' se refiere a pseudo-colores, los adaptadores CGA y EGA caen en esta categoria.

FILTRO. Es un dispositivo, método o programa que separa datos, señales o algun material basado en un criterio especificado.

FILTRO DE COLOR. Es un elemento de vidrio o plastico coloreado usado en fotografía para absorber ciertos colores y permitir el paso de otros. Permitiendo su mayor rendimiento, los filtros de color primarios (rojo, verde y ázul) pueden ser usadas con imágenes monocromáticas para crear imagenes a color.

FORMATO. Es todo un conjunto de especificaciones sobre la información contenida en archivo.

FRAME BUFFER. Es la memoria especializada de la computadora, que almacena temporalmente un número de 'frame' de video para manipulación o despliegue continuo.

FRECUENCIA ESPACIAL. Es la velocidad de cambio de la intensidad de un pixel dentro de la imagen.

ICONO. Es una representación gráfica funcional de una herramienta, utileria, archivo o comando desplegado en la pantalla.

IMAGEN. Es la representación bi-dimensional de un documento (fotografía, dibujo, texto, etc.); una función bi-dimensional obtenida con un dispositivo censor que graba los valores de las características de una imagen en todos sus puntos. Los valores pueden ser binarios para imágenes en blanco y negro, en niveles de gris para imágenes en tonos medios. Estas imágenenes son convertidas a su forma digital para procesarse con una computadora.

IMAGEN BINARIA. Es una imagen que usa solamente un bit por pixel para su despliegue. Este tipo de imágenes son desplegadas en blanco y negro. Si un pixel es un '1' usualmente se despliega como blanco y si por el contrario es un '0', es desplegada como negro.

IMAGEN EN TONOS CONTINUOS. Es una imagen que contiene tonos que van desde el negro pasando gradualmente por tonos grises hasta llegar al blanco.

LENGUAJE ENSAMBLADOR. Es el codigo en mnemónico que es convertido a lenguaje de máquina por un programa ensamblador. El lenguaje ensamblador es el único lenguaje que una computadora puede ejecutar directamente.

MAPA DE BIT. Una imagen en memoria de una porción en un área de display. El tamaño de un mapa de bit depende de la resolución en la que se esta desplegando la imagen y el número de colores que son soportados.

MAPA DE BYTE. Un tipo de imagen en la cual cada pixel esta definido como un byte de almacenamiento (ocho bits). El byte es capaz de almacenar hasta 256 niveles de gris, teniendo al '0' como negro y al '255' como blanco.

MAPEO. Es la conversión matemática de un conjunto de números a un conjunto diferente basado en alguna trasformación específica.

MASCARA. Una técnica de eliminar ciertos bits dentro de un número, dejando solamente los bits de interes.

MEMORIA. Es la unidad de almacenamiento temporal de información en una computadora (llamada RAM, Memoria de Acceso Aleatorio).

MODELO DE MEMORIA. Es la designación que dice al compilador donde se encuentran localizados los datos y el código y como van a ser referenciados.

MODO GRAFICO. Un modo del adaptador de display en el cual todos los pixels son independientemente controlables. Este es opuesto al modo texto en donde solamente caracteres y simbolos predefinidos pueden ser desplegados.

MONITOR. Es otro termino para referenciar el display o CRT.

MUESTREO. Es la selección de un conjunto de puntos sobre un campo de observación. Solamente los valores en esos puntos pueden ser en posteriores procesamientos.

NIVELES DE GRIS. Cuando una señal de video analógica es digitalizada (muestreada), los valores numéricos obtenidos deben ser mapeados dentro de un rango de números enteros (los cuales representan los niveles de gris) para desplegarse. Debido a que las imágenes en niveles de gris contienen información acerca de tonos de gris en la imagen, más de un bit por pixel es requerido.

NTSC. El estandar de video usado en Norte America, America Central y Japón. El "National Television Standard Committee", especifica una imagen entrelazada de 525 líneas rastreadas cada 1/30 segundo.

PALETA. Conjunto o colección de colores disponibles simultaneamente para el despliegue de una imagen.

PANTALLA. Un patron de puntos usados para imprimir sombras de gris en una impresora en B y N.

PEL. "Picture Element", elemento de una imagen. Es el área más pequeña de una imagen, capaz de ser direccionada y switcheada entre un estado visible e invisible.

PICTURE. Es una colección de 'pels' que al ser representados visualmente juntos forman una imagen.

PIXEL. Es un elemento de una imagen en un nivel lógico y conceptual.

PROCESAMIENTO DIGITAL DE IMAGENES. El procesamiento digital de imágenes trata con arreglos de números (muestras) que representan una imagen. Varios algoritmos se utilizan para la manipulación de muestras de imágenes en la obtención de resultados deseados. Despues de que el procesamiento es terminado las muestras que fuerón modificadas se utilizan para reconstruir la imagen en tonos-continuos.

PSEUDO COLOR. El termino "pseudo" se refiere a la aproximación más cercana de los colores reales.

PUERTO. Es el punto de conección en la computadora para dispositivos perifericos, tales como mouse o impresora.

PUNTO. Es el elemento controlable más pequeño, generalmente llamado 'pixel' o 'pel'

RAM. Ver Memoria.

RASTER. Un patron de predeterminado de líneas que provee una covertura uniforme en un espacio de despliegue.

RESOLUCION. El número de puntos discernibles en un campo de vista dado. La densidad del muestreo de puntos afecta el rango de detalles visibles en una imagen. Para mejorar la visión de una imagen la resolución debe ser más grande.

RESOLUCION ESPACIAL. En cuantización, la densidad de puntos de muestra.

RETRAZO HORIZONTAL. Es el intervalo de tiempo que ocurre cuando el rayo de electrones regresa del lado derecho al izquierdo de la pantalla de despliegue. El rayo de electrones se apaga durante este retraso.

RETRAZO VERTICAL. El intervalo de tiempo entre el despliegue de campos de video. Durante este tiempo, el rayo de electrones corre de la esquina superior izquierda a la esquina inferior derecha. Durante este lapso el rayo de electrones se encuentra apagado. Este retraso vertical ocurre entre 50 a 70 veces por segundo, dependiendo del CRT.

RGB. Modelo de color aditivo que forma colores mezclando los colores primarios rojo, verde y ázul.

SCANNER. Es el dispositivo que convierte imagenes en papel a una imagen en computadora.

SEPARACION DE COLOR. Es el proceso de separar imagenes a color en sus tres componentes de color primario: rojo, verde y ázul.

VECTOR. Un segmento de línea directo.

VERDADEROS COLORES. Es el metodo para desplegar gráficas en computadora con una reproducción más natural y aproximada posible.

VGA (Video Graphics Array). Es un estandar gráfico de pseudo-colores, desarrollado por IBM para su línea de computadoras personales PS/2. VGA utiliza un método analógico para generar imágenes a color, con 256 colores desplegables como máximo desde una paleta de 262,144 colores posibles.

VIDEO ANALOGICO. Es metodo usado para generar las señales obtenidas en un monitor de video. La información de una imagen digital es transferida a una tarjeta gráfica que contiene un convertidor digital/analógico (DAC), el cual convierte datos binarios en señales de voltaje continuos.

ZOOMING. Es la ampliación de un área visible dentro de una imagen.

Bibliografía.

- Digital Image Processing a Practical Primer
  Gregory A Baxes
  Edit. Prentice Hall
  1984
- Digital Image Processing
  Rafael G. González
  Richard E. Woods
  Edit. addisson Wesley
  1992
- Digital Video in the PC Enviroment
  Arch C. Luther
  Edit. Mac Graw Hill
  1989
- Programmer's guide to the EGA and VGA cards
  Richard F. Ferraro
  Edit. Addisson Wesley
  1988
- Avanced Graphics in C: Programming and Techniques
  Nelson Johnson
  Edit. Osborne/Mac Graw-Hill
  1987
- The Wait Group's Turbo C + + Bible
  Naba Barkakati
  edit. SAMS
  1990
- Graphics Programming in Turbo Pascal 6.0
  Ben Ezzell
  Edit. Addison-Wesley
  1991

- Practical Image Processing in C
  Craig A. Lindley
  Edit. John Wiley & Sons, Inc.
  1991
- Computer Graphics with Pascal
  Marc Berger
  Edit. Benjamin/Cummimgs Publishing
  1989
- Power Graphics Programming
  Michael Abrash
  Edit. Programming Series QUE
  1989
- Gráficas por Computadora
  Donald Hearn and M. Pauline Baker
  Edit. Prentice-Hall Hispanoamericana S.A.
  1989
- Power Graphics using Turbo C
  Keith Weiskamp
  Loren Heiny
  Namir Shammas
  Edit. John Wiley & Sons, Inc.
  1989
- Microsoft Mouse Programmer's Reference
  Edit. Microsoft Press
  1991
- Designing Screen Interfaces in C
  James Pinson
  Edit. Yourdon Press Computing Series, Prentice Hall
  1991
- Illustrating Computer Documentation
  William Horton
  Edit. Wiley
  1991

Hemerografía

- Graphics Format
  Gerald L. Graef
  BYTE
  septiembre 1989
  pág. 305-310
- Mastering the PCX Format
  Bert Tyler
  BYTE
  septiembre 1989
  págs. 183-187
- Nearest Neighbor Algoritm for Color Matching
  Geoffrey Probert
  The C User Journal : Algorithms
  febrero 1992
  págs. 31-38
- Color Image Quantization for Frame Buffer Display
  Paul Heckbert
  Computer Graphics
  Julio 1982 vol. 16 No. 3
  págs. 297-303
- In Depth: Sound an Image Processing
  Gene Smarte & Walt Penney
  BYTE
  diciembre 1989
  págs. 240-258
- VGA Video Modes
  Richard Wilton
  BYTE
  Edición Especial 1988
  págs. 187-198

- Is it Really Super ?
  Bill Nicholls
  BYTE
  Edición Especial 1988
  págs. 159-164