UNIVERSIDAD NACIONAL DE INGENIERIA
FACULTAD DE INGENIERIA ELECTRICA Y ELECTRONICA

"DISEÑO DE UN SISTEMA DE MONITOREO PARA IDENTIFICACION DE PERSONAS POR CARACTERISTICAS FACIALES"

TESIS

PARA OPTAR EL TITULO PROFESIONAL DE:

INGENIERO ELECTRONICO

PRESENTADO POR:

CARLOS ABEL GALVAN MALDONADO

PROMOCION 1984-II

LIMA-PERU
1999

Dedico este trabajo a mis queridos padres **Cipriano** y **Marcelina** asimismo a mis hermanos, que me apoyaron para culminar el presente proyecto.

“DISEÑO DE UN SISTEMA DE MONITOREO PARA IDENTIFICACION DE PERSONAS POR CARACTERISTICAS FACIALES”

S U M A R I O

En las operaciones de procesamiento digital de imágenes orientados al monitoreo de imágenes para la identificación de personas por caracteristicas faciales, hasta hace pocos años era casi imposible efectuar la restauración, debido a la degradación en la imagen original, compresión de imagenes para la transmisión de señales. Si bien muchas ecuaciones matemáticas que realizan éstas operaciones no podían ser implementados debido al alto grado de operaciones a realizar.

En el presente proyecto de tesis se desarrolla un sistema de hardware que adaptado a un computador de generación actual, sea capaz de realizar adquisiciones de señales de video en banda base (generados por una cámara de T.V.) para su almacenamiento en un formato estandar de gráfico y luego efectuar su posterior proceso digital.

Por lo que empleando la tecnología nacional se desarrolló un prototipo de un sistema para la identificación automática de personas, el mismo que presenta ventajas de reducción de costos, una confiabilidad apreciable, asimismo podrá sentar las bases para el tratamiento de señales de video e imágenes en el marco de una tecnología propia, lo cual permitirá una mejor aplicación en los requerimientos de nuestro país.

I N D I C E

Página

PROLOGO 01

CAPITULO I

DESCRIPCION Y FUNDAMENTOS DE PROCESAMIENTO DIGITAL DE SEÑALES DE IMÁGENES. 03

1.1. Descripción del proceso digital de señales. 03

1.2. Aplicaciones y perspectivas del proceso digital de señales. 06

1.3. Proceso digital de imágenes. 07

1.4. Técnicas de proceso digital de señales de imágenes. 11

1.5. Requerimientos funcionales para el proceso digital de señales. 18

CAPITULO II

PROCESAMIENTO DE SEÑALES DE VIDEO 20

2.1. Fundamentos de la exploración normalizada. 20

2.1.1 Principios de entrelazado. 21

2.1.2. Entrelazado de líneas impares. 23

2.2. Señales de sincronización. 27

2.3. Señales de video compuesto. 31

2.4. Definición de términos en procesamiento digital de imágenes. 34

2.5. Procesamiento de señales de video compuesto. 38

CAPITULO III

DISEÑO DEL SISTEMA DE ADQUISICION DE SEÑALES DE VIDEO. 49

3.1. Soporte de memorias de adquisición de datos y comunicación a computador. 49

3.1.1. Sistema de memorias ram. 52

3.1.2. Soporte decodificador de puertos de entrada/salida de adquisición de datos. 59

3.1.3. Software de soporte de acceso de datos a computador. 66

3.2. Diseño de la etapa de conversión analógico /digital. 71

3.3. Diseño de la etapa de conversión digital/analógico. 78

CAPITULO IV

PROCESAMIENTO DE SEÑALES DE VIDEO (IMÁGENES). 81

4.1. Codificación de archivos. 82

4.2. Personalización de la imagen. 83

4.3. Archivo gráfico del mapa de bits. 84

4.4. Software de conversión de archivo de imagen digital. 87

CAPITULO V

DETECCION DE LAS CARACTERISTICAS FACIALES PERSONALES. 99

5.1. Sistema de procesamiento de datos adquiridos por digitalización. 99

5.2. Detección de perfiles de imágenes. 105

CAPITULO VI

PROCESAMIENTO DE COMPRESION DE IMAGEN 112

6.1. Objetivo de Uso de compresión de datos de imágenes. 112

6.2. Criterio de diseño. 115

6.3. Métodos de compresión de datos. 117

6.4. Reconocimiento de patrón de datos en dos dimensiones. 118

6.5. Software de desarrollo. 132

CONCLUSIONES 152

BIBLIOGRAFIA 154

P R O L O G O

El requerimiento de los Sistemas Programables orientados al campo del procesamiento digital de señales es cada día mas imprescindible en diversas aplicaciones. Para realizar un sistema de adquisición de señales analógicas de baja frecuencia no es muy complejo, ya que se pueden encontrar en el mercado, los conversores ADC de 8 bits que fácilmente pueden muestrear todo este rango de frecuencias hasta aproximadamente 100 KHz. Y la adquisición de esta información, puede pasar directamente a un computador y efectuar el procesamiento digital casi en tiempo real. Pero no es el mismo caso con las señales de alta frecuencia; es decir con las señales de video donde los dispositivos a utilizar deben ser de tiempo de respuesta rápidas y los computadores no pueden llegar a realizar el almacenamiento de esta información en tiempo real, mucho menos llegar a realizar algún procesamiento. En la actualidad se han desarrollado dispositivos muy especializados de aplicación específica (ASIC) que realizan el procesamiento digital de imágenes por hardware. El objetivo del presente Proyecto de Tesis es desarrollar un sistema que permita realizar la adquisición de señales de video y poder almacenar esta información en un computador, y para efectuar el procesamiento de esta señal se implementan rutinas o

algoritmos utilizando alguna herramienta de software. Una aplicación al desarrollo de este sistema es la del reconocimiento de personas, en el cual se digitaliza la imagen de la persona y se le aplica a ello algoritmos que permiten determinar las características propias de cada persona y de este modo efectuar su reconocimiento.

El presente Proyecto se desarrolló en los laboratorios de Investigación y Proyectos de la Universidad Nacional del Centro con el valioso auspicio de CONCYTEC, por lo cual estamos profundamente agradecidos; asimismo mi agradecimiento profundo y sincero a mi asesor de la presente tesis Ing. Luis Romero Goytendia, por su apoyo muy valioso y constante el cual me permitió completar el proyecto.

CAPITULO I
DESCRIPCION Y FUNDAMENTOS DE PROCESAMIENTO DIGITAL DE SEÑALES DE IMÁGENES.

1.1 Descripción del proceso digital de señales

En nuestra actualidad, el objetivo de los sistemas programables modernos es realizar una gran variedad de operaciones en el mínimo tiempo necesario, por lo que disponen de dispositivos (microprocesadores, memorias, interfaces, dispositivos lógicos programables, arreglos lógicos programables) que funcionan en altas velocidades y márgenes de precisión óptimos, los cuales representan los requerimientos mínimos en la mayoría de problemas científicos y de ingeniería, en el cual se emplea la simulación y desarrollo empleando algoritmos matemáticos.

Singularmente el campo de procesamiento digital de imágenes también presenta éste requerimiento en donde se realizan operaciones con imágenes para resolver el problema de la restauración, debido a la degradación de la imagen original, el cual es superado gracias a los procesadores especializados que efectúan éstas y otras operaciones más complejas.

El proyecto presente contempla el desarrollo de un sistema digital de monitoreo para la identificación de personas por características

faciales, el cual adaptado a un computador, permite realizar adquisiciones de señal de video en banda base generadas por una cámara de televisión para su almacenamiento en memorias masivas según formato estandar de gráficos, para luego realizar el procesamiento digital, para lo cual se emplean algoritmos matemáticos que realizan el procesamiento de la señal de video, asimismo el sistema permite generar señales de video para ser visualizados en un monitor de televisión.

El sistema en mención esta constituido por una etapa del sub-sistema de adquisición de datos para generar el cuadro de video, el mismo que tratará de la imagen de la persona. Para implementar esta etapa se empleará de una cámara de televisión, y su circuito de proceso de señales para digitalizar dicha imagen; dicho sistema cuenta con una etapa de adaptación y amplificación para asegurar que la señal de video estandar debe estar de acuerdo con la recomendación EIA. Para que la señal de video sea adquirida adecuadamente se emplea un conversor analógico digital de alta velocidad, asimismo es necesario una lógica de control que permite que el computador maneje de manera optima las adquisiciones de imágenes, de esta forma se obtiene información en forma de señal de video sin movimiento. A su vez en ésta fase se plantea la necesidad de diseñar un hardware adecuado para el almacenamiento del registro de video, en base al uso de bancos de memorias SRAM. Se debe tener en cuenta que la velocidad de las memorias (Tiempo de acceso muy rápido y de gran densidad) en ésta aplicación se prefiere las SRAM porque son de mayor velocidad y su

bajo costo, además por que la lógica de control permite un correcto funcionamiento en esta fase, a la vez que se obtiene un funcionamiento independiente respecto del computador, y luego de digitalizado se almacenará dicha imagen en el disco duro del computador utilizando un determinado formato, para luego pasar a una etapa del procesamiento digital, el cual tendrá como función principal de efectuar la detección de las características faciales en tercera dimensión, mediante algoritmos matemáticos. En forma adicional se puede mencionar el desarrollo de una etapa interface de comunicación de la información procesada, centralizando en una estación principal donde se podrá almacenar una base de datos con características faciales de personas para su posterior identificación y reproducción en un monitor de televisión.

En la segunda fase, de modo similar que en todo diseño de sistemas de esta naturaleza, se considera el desarrollo el diseño del software de procesamiento, presentación y control de la imagen digitalizada y procesada, asimismo la reproducción de esta. El software diseñado maneja la información de imagen bajo un formato de archivo con las mayores facilidades de uso por parte del usuario, el cual será de gran utilidad en las aplicaciones requeridas, asimismo es importante el uso de algoritmos para la compresión digital de la imagen de video y para determinar las características faciales de la persona, éstas técnicas serán explicadas en los próximos capítulos.

1.2 Aplicaciones y perspectivas del proceso digital de señales

La importancia del proyecto en mención es cada vez más interesante, debido a que las aplicaciones en las que se pueden proyectar se hacen cada vez más amplias, como se puede apreciar en la descripción de los campos más importantes en las que se pueden emplear, entre ellos:

a) Sistema de seguridad nacional, para evitar suplantación de personalidades singulares, en el ámbito político, militar, científicos y hombres de ciencia.
b) Sistemas de seguridad de cajas y bóvedas de seguridad de los bancos y financieras .
c) En la industria, para su automatización, donde el sistema de supervisión visual mejora la calidad en la inspección y procesamiento del monitoreo.
d) En el sistema institucional para evitar suplantación de personas en eventos especiales.
e) En las operaciones de documentos de identificación de personas de una nación y su almacenamiento de datos en banco de datos.
f) Aplicación en la composición de imágenes, añadiendo y sustraendo objetos en una escena que no existieron originalmente.

El desarrollo del presente proyecto en el marco de la ingeniería nacional muestra grandes posibilidades de desarrollo de sistemas de video sofisticados, así como el estudio y desarrollo de técnicas de procesamiento de imágenes. El campo inicial para orientar la aplicación

del presente proyecto son las instituciones de investigación y desarrollo nacional como las universidades, instituciones militares y policiales.

El sistema permite grandes posibilidades de desarrollo de la ingeniería en proceso de imágenes en nuestro país, permitiendo el desarrollo de la ingeniería nacional en el campo de aplicación de los sistemas automatizados, dado que la implementación se puede realizar con el uso de dispositivos comerciales en nuestro mercado nacional, asimismo el proyecto es totalmente abierto en donde la señal de video es capturada y transferida al computador para realizar cualquier tipo de procesamiento digital: conversión análoga/digital, filtros digitales, compresión de imágenes, restauración de imágenes, alteración y modificación de imágenes, reconocimiento de imágenes en general, el cual esta abierto a la posibilidad de creatividad del programador. En la Fig. No 1.1 se muestra la aplicación del procesamiento digital para el reconocimiento de personas.

1.3 Proceso digital de imágenes.

El procesamiento digital de imágenes y la manipulación de imágenes procesada por los sistemas programables, es un desarrollo muy reciente estimulado por la inquietud de la raza humana a los estímulos visuales, en su poco recorrido se han realizado aplicaciones con diversos grados de éxito. En nuestra década actual de los 90, son múltiples los factores que se combinan para motivar cada vez mas el proceso digital de imágenes, uno de los factores importantes es la reducción progresiva de

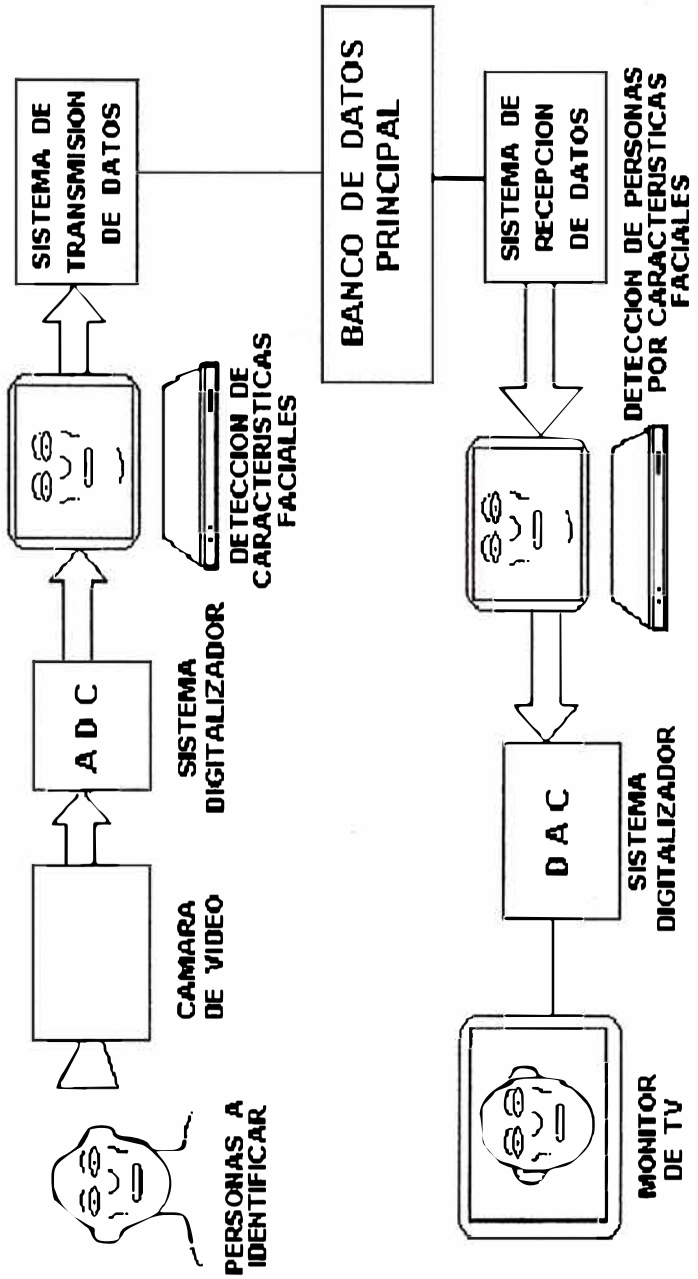


Fig. N° 1.1 Sistema de adquisición de datos de video pa reconocimiento de personas.

los componentes digitales, principalmente los procesadores son cada vez mas veloces y económicos así como los dispositivos de memoria, otro factor que incide es la creciente disponibilidad de equipos para la digitalización de las imágenes así como los dispositivos de presentación como son los monitores, paneles, etc.

El proceso digital de imágenes básicamente requiere que un sistema computador procese imágenes, así como equipos de entrada/salida de señal de video, un digitalizador de imagen y un dispositivo presentador de imagen. Como se hizo mención, las imágenes no son directamente receptivas al análisis del computador, previamente debe ser procesada mediante un proceso denominado Digitalización de la imagen y una forma común es lo representado en la Fig. No.1.2

La imagen se divide en zonas pequeñas llamadas elementos del cuadro (PIXEL), el cual es realizado bajo un plan de subdivisión obteniéndose las rejillas rectangulares de muestreo mostrado en la Fig. No.1.2. La imagen se divide en líneas horizontales constituidas por los pixel´s adyacentes. A cada pixel le corresponde un nivel de oscuridad o brillo de la imagen en ese punto. Cuando esto se ha realizado para todos los pixel´s , la imagen es representado por una matriz rectangular de enteros.

Cada pixel tiene una ubicación o dirección (el número de líneas o filas y número de muestras o columnas) y un valor entero llamado "Nivel de gris"; este conjunto de datos digitales es ahora una información para que pueda ser procesada por el computador.

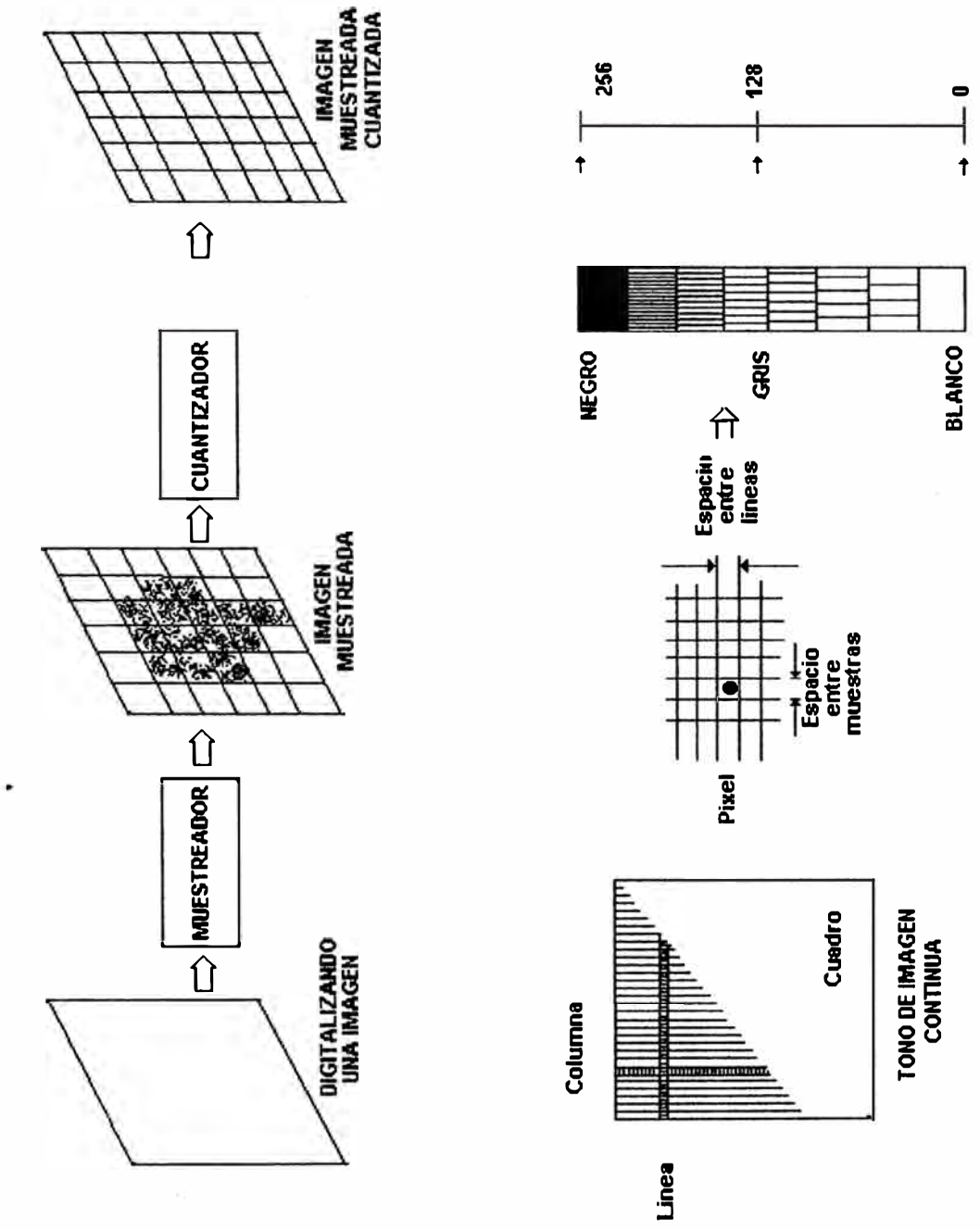


Fig. Nº 1.2 Proceso de digitalización de una imagen.

En la Fig. No. 1.3, se muestra un sistema completo para el procesamiento de imágenes, el cual luego del proceso por el equipo digitalizador va a un dispositivo de almacenamiento de información digital temporal. En respuesta a esta entrada, el sistema computador llama y ejecuta programas de procesamiento de imágenes desde su librería. Durante la ejecución, la imagen leida es transferida hacia el computador línea por línea y al operar sobre una o varias líneas, el computador genera una imagen de salida estructurada por Pixel´s .

Durante el procesamiento , los pixel´s pueden ser modificados por el programador estando únicamente limitado por su imaginación, y el tipo de procesador que se emplea. Después del proceso, el producto final es mostrado por un proceso antagónico a la digitalización. La imagen procesada es transformada a una imagen visible y por tanto detectada por la visión humana.

1.4 Técnicas de proceso digital de señales de imágenes.

Las técnicas más empleadas en el proceso digital de imágenes y los que pueden ser desarrolladas con la ayuda de este sistema son:

a) **Análisis de imágenes:**

Los procesos de análisis aplicados en imágenes, generalmente no producen resultados gráficos, sino mas bien producen información gráfica o numérica basada en las características de la imagen original. Las operaciones de análisis rompen una imagen dentro de objetos

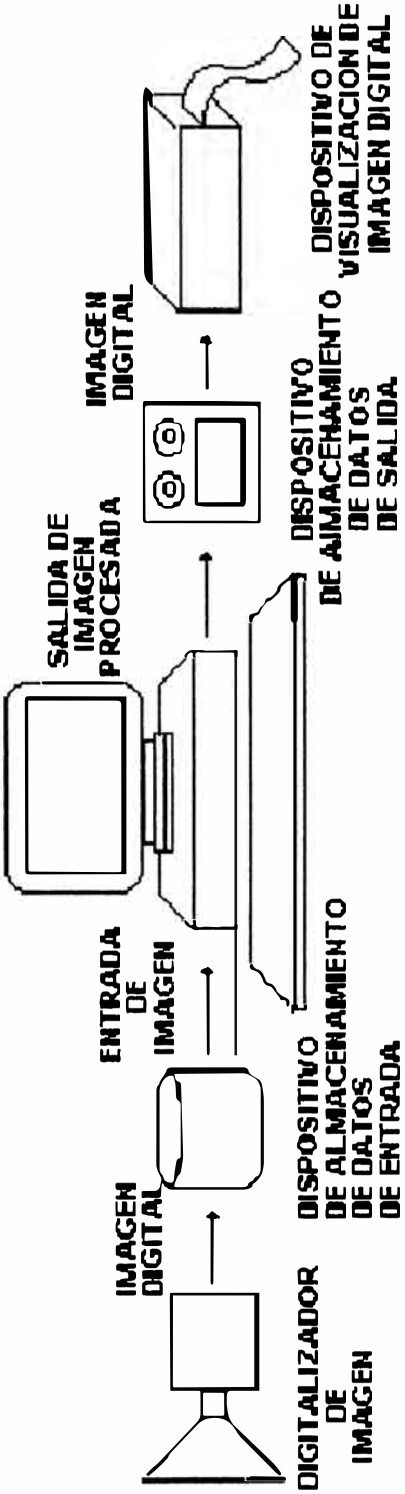


Fig. Nº 1.3 Sistema de Procesamiento digital de una imagen.

discretos y luego clasifican estos objetos usando algunos procesos de medidas. Asimismo una operación de análisis puede producir imágenes estadísticas. Asimismo estas operaciones de análisis de imagen incluyen extracciones y descripciones de escenas y sobre todo características, medidas automáticas y clasificación de objetos.

Generalmente los métodos de análisis de imágenes son principalmente usados en aplicaciones de visión para maquinas, en donde la medición de parámetros de los objetos es muy importante en su análisis; dentro de estas mediciones se pueden incluir la forma, el tamaño, las posiciones relativas, texturas, tonos de gris, colores y otros parámetros de los objetos. El aspecto geométrico por ejemplo puede tomar una cierta cualidad como pueden ser cuadrados o redondos, largos ó cortos, grandes o pequeños, etc. Las técnicas usadas para clasificar a un objeto pueden variar, dependiendo de las características de aplicación. Estas pueden ser tan simples como las características de formas ásperas ó las más complejas como la medida exacta de las dimensiones y formas geométricas. Otra aplicación del análisis pueden producir variados tipos de imágenes estadísticas, uno muy importante es el histograma de brillantez, que representa la distribución de los niveles de gris de una imagen. El histograma describe todo o parte del contraste de la imagen y pueden asimismo ser usado para determinar los parámetros de mejoramiento de contraste.

b) **Realce de imágenes:**

Las operaciones de realce mejoran la calidad de una imagen digitalizada (definido por características de brillo y contraste), las degradaciones de contraste en una imagen están asociadas con la eficiente distribución del brillo, ésto significa que la imagen pueda parecer borrosa ó muy saturada.

En una imagen muy saturada se exhibe los bajos contrastes lo cual es indeseable, los tonos de gris no abarcan todo el rango de blanco y negro: Una imagen bien balanceada de buen contraste esta compuesta por un alargado tono de gris desde el negro, pasando por los tonos de gris medios hasta el más blanco, las operaciones para realzar una imagen de pobre contraste con relativamente simples, el mejoramiento del contraste hace que la imagen exhiba una distribución más natural de los niveles de gris.

Las degradaciones espaciales en una imagen están generalmente asociadas con la presentación de detalles en la escena. Es preciso aclarar que el termino "espacial" se relaciona a una imagen natural de dos dimensiones si las áreas en detalle son borrosas ó sobresaturadas. A menudo los detalles de márgenes tales como las transiciones de negro a blanco pueden ser borrosas, generalmente la falta de sutileza esta asociada con óptimos detalles.

c) Restauración de imágenes:

La operaciones de restauración de imágenes, requieren de operaciones de mejoramiento, también mejora las cualidades de imagen

todas las operaciones de restauración, de cualquier modo son estrictamente objetivas, es decir están basadas en conocidas, supuestas ó moderadas correcciones de degradaciones de la imagen original. Usualmente las imágenes candidatas a la restauración tienden a sufrir algunas formas de degradación; las técnicas de restauración de imágenes se pueden usar para restaurar imágenes con problemas tales como distorsión geométrica, imágenes desenfocadas, ruido periódico y por movimiento de la cámara de detección de imagen.

También se pueden considerar en los sistemas de captación de imágenes las múltiples formas de distorsión, por ejemplo en las cámaras se utilizan usualmente un sistema de iluminación, las lentes, fotodetectores y funciones de digitalización, los cuales influyen en una degradación total de la imagen final.

La restauración fotométrica corrige una imagen por deficiencias a la respuesta a la intensidad, asimismo la iluminación o la respuesta de los fotodetectores podrían causar que partes de una imagen sufran de anomalías de brillo. Algunas distorsiones geométricas pueden ser causadas por la geometría física de los sistemas de adquisición de datos e imagen, tales como en las naves aeroespaciales. En este caso hay que tener en cuenta los parámetros de la órbita ó altitud para ser usados en la restauración de la imagen, porque la forma física de las lentes frecuentemente generan distorsiones en la imagen que aparezca mas ancho en el centro de la figura.

d) Compresión de imágenes:

Las operaciones de compresión y descompresión de imágenes reducen el contenido de datos necesarios para describir una imagen. La compresión de imágenes es posible porque el mayor número de imágenes contiene inherentemente largas cantidades de información redundante, la eliminación de esta redundancia es el fin de todas las operaciones de compresión de imágenes.

Generalmente las imágenes son comprimidas a formas mas pequeñas, en forma de archivos por ejemplo, los que pueden ser almacenadas en dispositivos de memorias digitales para ser fácilmente transmitidas y/o transportadas. Y Luego para ser representado la imagen deberá ser descomprimido para su proceso complementario . Este proceso de compresión de imágenes llega a ser lo mas preciso, cuando se tiene secuencias de grandes cantidades de imágenes que deberán ser tratadas como es el caso de los servicios de procesamiento o transmisión de imágenes.

Existen dos formas principales de compresión de imágenes, las técnicas de compresión sin perdida que preserva el dato exacto hallado en la imagen original y las técnicas de compresión con perdida que no representan exactamente el dato de la imagen original pero mantienen un particular nivel de la calidad de la imagen. En estas técnicas de compresión y de descompresión sin perdida son usadas cuando los datos

de la imagen deben ser exactamente preservados. Por ejemplo las imágenes médicas o científicas.

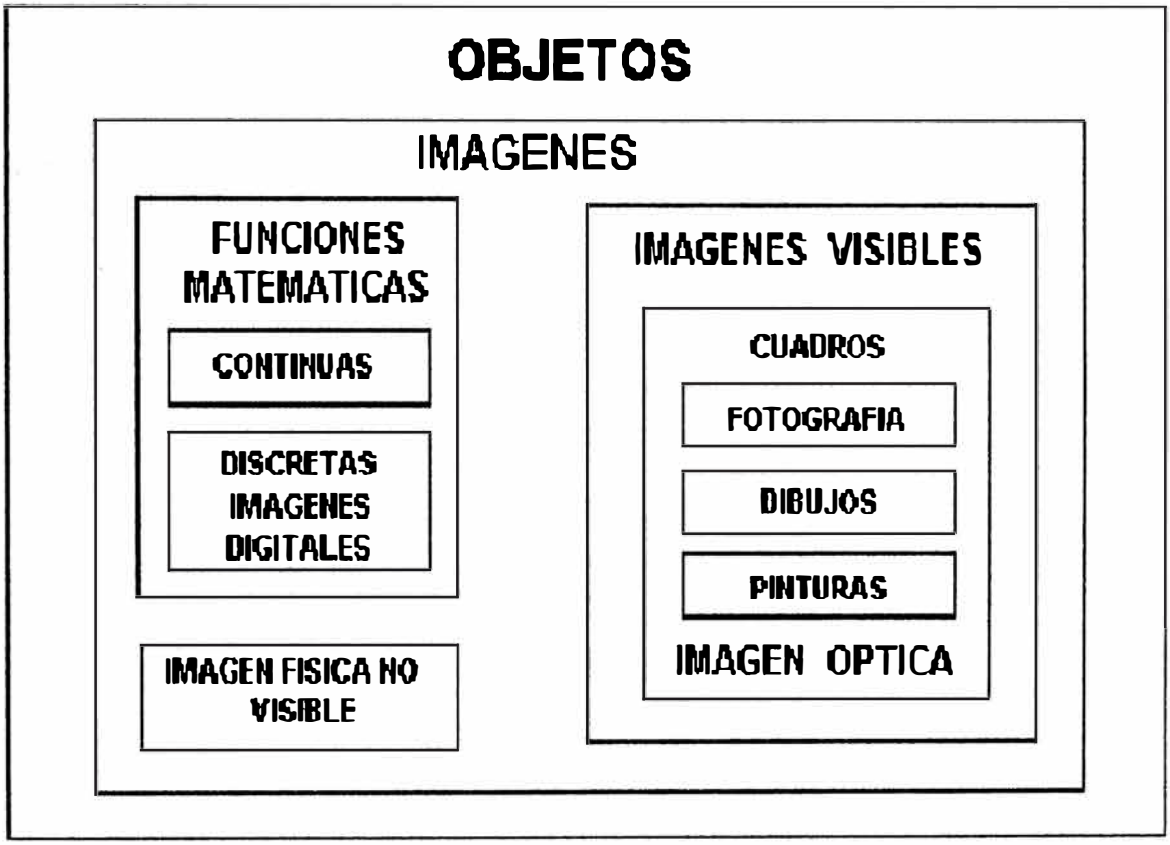


Fig. N° 1.4 Tipos de imágenes.

Las técnicas de compresión proveen un factor de compresión de uno a dos dependiendo del contenido de la imagen y de la operación usada.

Las técnicas de compresión y descompresión con perdida son usadas cuando la calidad de la imagen reconstruida debe mantener un poco nivel, pero es precisamente idéntico a la imagen original. La imágenes apropiadas para la compresión con perdida podrían ser las video conferencias, videotelefonía y técnicas de impresiones . Esta técnica provee un factor de compresión de 10:1 hasta de 100:1 y hasta mucho mas., esto depende del contenido de la imagen, la operación usada y la calidad aceptable de la imagen.

1.5 Requerimientos funcionales para el proceso digital de señales.

Para efectuar el procesamiento de imágenes se requiere de lo siguiente:

a) El hardware del sistema programable debe ser adecuado para el proceso de la digitalización de imágenes, cuyo requerimiento es muy exigente en el procesador, las memorias , los circuitos de soporte y sus unidades de entrada/salida. Por ejemplo el inadecuado muestreo en el dominio del tiempo y la inadecuada cuantización de los niveles de gris pueden degradar la imagen a digitalizar, generando fallas inconclusas. Actualmente los algoritmos de procesamiento asumen que la imagen es una función continua, si el muestreo y la

cuantización usada no justifica esta suposición, el proceso puede resultar con múltiples errores.

b) Cuando los niveles de ruido del sistema degradan la imagen, se tiene el riesgo de un proceso inadecuado. Por ello en este campo de aplicación de análisis de imágenes se requiere una alta calidad del digitalizador, así como del dispositivo de salida ó visualizador de pantalla.

c) En aplicaciones generales de trabajo, es también importante el software del sistema que debe permitir una interacción a una simple librería y ejecución del programa de análisis y procesamiento. Asimismo las librerías de los programas deberán tener como requerimiento un sistema de almacenamiento adecuados de las señales de imágenes en discos o cintas magnéticas ú ópticas.

d) Es necesario también en estas aplicaciones disponer de un banco de librerías orientados a la versatilidad de las aplicaciones, porque la potencia del sistema es grandemente mejorada si existen programas que pueden ser usadas para solucionar problemas de diversos tipos de enfoque.

e) El banco de librería deben ser modulares y extensibles para incluir nuevos programas como otros que serán desarrollados, de este modo el sistema tendrá mejoras en nivel de proceso cualitativo y cuantitativo.

CAPITULO II
PROCESAMIENTO DE SEÑALES DE VIDEO

2.1 Fundamentos de la exploración normalizada

Si deseamos obtener de manera adecuada una imagen de video, de debe considerar el fundamento del origen de este tipo de señal y analizar sus características eléctricas, porque de este modo podemos obtener del sistema diseñado sus cualidades que sean compatibles totalmente con las señales de video estandar empleadas comúnmente en los sistemas de televisión convencionales.

Analizando los sistemas de televisión, vemos que es esencialmente un sistema que reproduce una imagen en estado estático, de modo similar a una fotografía instantánea; el mismo que al presentar una tras otra a una cierta frecuencia adecuada, permite visualizar la imagen en movimiento, de donde podemos iniciar del análisis del principio básico de que el sistema de televisión debe ser capaz de recibir y transmitir una imagen simple en estado estático.

Respecto a la imagen, está compuesta fundamentalmente por una composición de muchas superficies pequeñas oscuras y claras; y cada superficie pequeña oscura ó clara es denotado como ¨elemento de imágen¨ y todos los componentes contienen información visual de la escena. Si son transmitidos y reproducidos con el mismo grado de luz y

sombra que la imagen original y la posición correcta, tendremos una óptima reproducción de la imagen transmitida. Y para producir la señal de video todos estos elementos de imagen son explorados por un haz electrónico, un elemento tras otro, en una secuencia pre-establecida, realizándose esta exploración del mismo modo en que se realiza la lectura de una página a fin de que sean consideradas toda la información de la línea y también todas las líneas de la pagina. Por lo que se emplea comunmente el método llamado "Exploración lineal horizontal", en el tubo de cámara del transmisor para disectar la imagen en los elementos de imagen y en el cinescopio del receptor para reagrupar la imagen reproducida luego del proceso concluido del proceso.

El proceso de exploración que es adoptado universalmente, se emplea la exploración línea horizontal en una figura entrelazada de líneas pares e impares. Donde las especificaciones de la CF (Federal Communications) de exploración para la teledifusión describen una característica de exploración que incluye un total de 525 líneas horizontales para un cuadro rectangular cuya relación de aspecto es de 4 : 3, asimismo los cuadros deben repetirse a una velocidad de 30 cuadros por segundo con dos campos entrelazados para cada cuadro respectivamente.

2.1.1 Principios de entrelazado.

El proceso de exploración entrelazada se puede comparar a las líneas interpuestas de la Fig. No 2.1, donde la información es continua si se leen todas las líneas impares de arriba abajo y luego se retorna a la parte

de arriba para leer las líneas impares. Considerando toda una pagina escrita y se realiza el proceso de lectura de éste modo entrelazado se tendrá la misma información en la que se proporcionase la lectura actual en la que todas las líneas están escritas unas a continuación de otras.

En el sistema de televisión la líneas de exploración
Las líneas impares omitiendo las líneas pares y luego se
Horizontal están entrelazadas de modo que aparezcan
Exploran las líneas pares para completar el cuadro sin
Dos campos de la imagen en cada cuadro. Se exploran
Que se pierda información de la imagen.

Fig No 2.1 Fundamento de Exploración Entrelazada.

En la exploración entrelazada son consideradas inicialmente de arriba a abajo todas las líneas impares, omitiendo las líneas pares del cuadro. Luego de este ciclo de exploración vertical, un retorno vertical rápido desplaza el haz electrónico de exploración a la parte superior del cuadro, luego todas las líneas pares que anteriormente han sido omitidas son exploradas de arriba abajo, completando de este modo la exploración total del cuadro con el cual se conformará la imagen del cuadro.

Como consecuencia vemos que, cada cuadro se divide en dos campos, el primero y todos los campos impares contienen las líneas impares del cuadro, mientras que el segundo y todos los campos pares incluyen las líneas pares de exploración, por lo que con los dos campos por cuadro y

30 cuadros completos explorados por segundo y la velocidad de repetición de campo es 60 por segundo y la frecuencia de exploración vertical es de 60 ciclos por segundo, representando la composición de imagen para su transmisión-recepción.

2.1.2 Entrelazado de líneas impares

La característica de exploración entrelazada de las líneas impares es representada en la Fig. No 2.2, en el cual podemos observar que el tubo de rayos catódicos emite el haz electrónico dirigiendo al haz al centro, en donde empieza la exploración, pero sin embargo, podemos iniciar la secuencia iniciando la exploración a partir del ángulo superior izquierdo del cuadro (Punto A), en la línea 1, el haz electrónico barre el cuadro de izquierda a derecha con velocidad uniforme para cubrir todos los elementos de imagen en la línea horizontal; en el extremo de esta traza el haz retrocede rápidamente al lado izquierdo del cuadro como indica la línea de trazos de la Fig. No.2.2, para luego iniciar la siguiente línea horizontal.

También se puede observar que las líneas horizontales están inclinadas hacia abajo, en sentido de la exploración debido a que las señal de deplexión vertical produce en forma simultánea un movimiento de exploración vertical pero relativamente lento en comparación con la exploración horizontal. Se observa también que la pendiente de la traza horizontal de izquierda a derecha es mayor que durante el retorno de derecha a izquierda. La razón es que el retorno es mucho mas rápido que

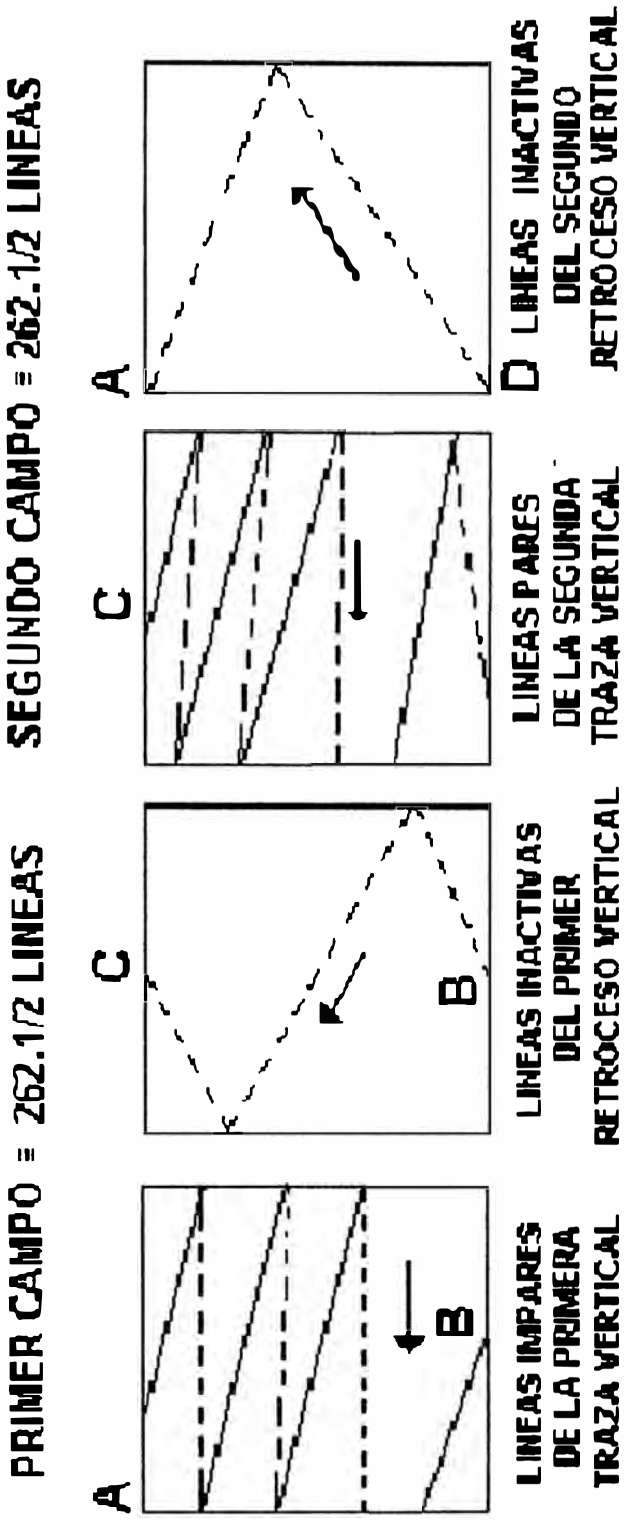


Fig. Nº 2.2 Proceso de exploración de lineas entrelazadas.

no permite que el haz invierta mucho tiempo en desviarse verticalmente. Analizando brevemente la línea 1, donde el haz esta en el lado izquierdo dispuesto para explorar la línea 3 omitiendo la segunda. Este salto de línea se realiza por la duplicación de la frecuencia de exploración vertical desde la velocidad de repetición del cuadro de 30 hasta la frecuencia de campo de 60 ciclos por segundo.

Respecto al haz vertical, el barrido debe ser realizado a una velocidad doble de la necesaria para explorar las 525 líneas para producir un periodo completo de la exploración vertical para las 262 ½ líneas solamente, con líneas alternas sin explorar. El haz electrónico explora todas la líneas impares, luego llega finalmente a una posición tal como la parte B de la Fig. No. 2.2, en la parte inferior del cuadro. En el instante B se inicia el retorno vertical a causa del retroceso de la señal de desviación vertical de diente de sierra. El haz vuelve a la parte superior del cuadro para comenzar la exploración del segundo campo, el par. Como muestra la Fig. No 2.2 el haz se desplaza desde el punto B hasta el punto C, cruzando cierto numero de líneas horizontales.

El tiempo de retorno vertical es suficientemente largo para que el haz pueda explorar varias líneas horizontales. Llamamos a esta líneas de retorno vertical, comprendiendo que son las líneas horizontales completas exploradas durante el retorno vertical. Se puede observar que la línea de retorno vertical están inclinadas hacia arriba, cuando el haz se mueve mientras las explora horizontalmente. La pendiente hacia debajo de la línea de retorno vertical es mayor que la pendiente hacia arriba de

las líneas exploradas durante la traza vertical, a causa e que el retorno ascendente es mucho más rápido que la traza descendente. Las líneas exploradas durante el retorno vertical no son visibles a causa de que el haz electrónico queda suprimido por la tensión de borrado durante el tiempo de retorno vertical.

La exploración vertical durante el segundo campo comienza cuando el haz se encuentra en el punto C de la Fig. No 2.2. este punto está en el centro (punto medio) de la línea horizontal a causa de que el primer campo contiene 262 líneas más una media línea. Después de la exploración de la media línea desde el punto C el haz explora la línea 2 del segundo campo. Luego el haz explora el espacio comprendido entre las líneas impares para producir las líneas pares que fueron omitidas durante el proceso de exploración del primer campo.

El movimiento de la exploración vertical es exactamente el mismo que en el campo anterior, dando a todas las líneas horizontales la misma pendiente descendente en el sentido de la exploración. Como consecuencia todas las líneas pares del segundo campo son explorados hacia el punto D. Los puntos D y B están distanciados media línea entre si a causa de que el segundo campo se ha iniciado con una media línea, según se observa en el gráfico.

El retorno vertical del segundo campo se inicia en el punto D de la figura No. 2.2, desde este punto el retorno vertical lleva al haz a la parte superior. Con un numero entero de líneas de retorno vertical, el haz termina el segundo retorno vertical en A. El haz terminara siempre el

segundo retorno vertical allí donde ha empezado la primera traza a causa de que el punto A, el haz explorador ha terminado precisamente 2 campos es decir el equivalente a un cuadro y esta listo para iniciar el tercer campo en que se repiten las características de exploración.

En síntesis todos los campos impares comienzan en el punto A y son los mismos, todos los campos pares comienzan en el punto C y son los mismos, puesto que el comienzo de exploración de campo par se efectúa en el mismo nivel horizontal que A con una separación de media línea y la pendiente de todas las líneas es la misma, las líneas pares de los campos pares caen exactamente entre las líneas impares del campo par. Las condiciones para el entrelazado de líneas impares es que los puntos iniciales en la parte superior del campo estén separados exactamente media línea entre los campos par e impar.

2.2 Señales de sincronización

En el proceso de la exploración del cinescopio del receptor del haz se debe efectuar la unión de los elementos de imagen en cada línea horizontal con la misma posición relativa de izquierda a derecha de los elementos de imagen en el tubo de cámara. Además, cuando el haz explora verticalmente las líneas sucesivas en la pantalla del cinescopio, debe presentar los mismos elementos de imagen que los de las líneas correspondientes en el tubo de la cámara. Por ello se transmite un impulso de sincronismo horizontal para cada línea a fin de conservar sincronizada la exploración horizontal y el impulso de sincronismo

vertical se transmite en cada campo para sincronizar el movimiento de exploración vertical. Los impulsos de sincronismo horizontal tienen una frecuencia de 15.750 cps y la frecuencia de los impulsos de sincronismo vertical es de 60 cps.

Los impulsos de sincronización se transmiten como parte integrante de la señal de imagen, pero son enviados durante los periodos de borrado cuando no se transmiten ninguna información. Esto se realiza debido a que el impulso de sincronización inicia el retorno, ya sea horizontal o vertical, y por consiguiente se produce durante el tiempo de retroceso. Las señales de sincronismo se combinan con la señal de imagen de manera que parte de la amplitud de la señal de imagen modulada se utiliza para los impulsos de sincronización y el resto para la señal de cámara.

La forma en que los impulsos de sincronización esta representado en la Fig. No. 2.3, en donde se puede observar que todos los que tienen la misma amplitud, pero el ancho o su forma de onda es diferente. Se representan de izquierda a derecha tres impulsos horizontales, un grupo de seis igualadores, un impulso vertical fraccionado y seis impulsos igualadores adicionales que van seguidos de otros tres impulsos horizontales. Hay muchos impulsos horizontales adicionales después del ultimo representado, siguiendo sucesivamente uno tras otro en la frecuencia de línea horizontal hasta que se produzcan los impulsos igualadores nuevamente para iniciar el siguiente campo. Por cada campo

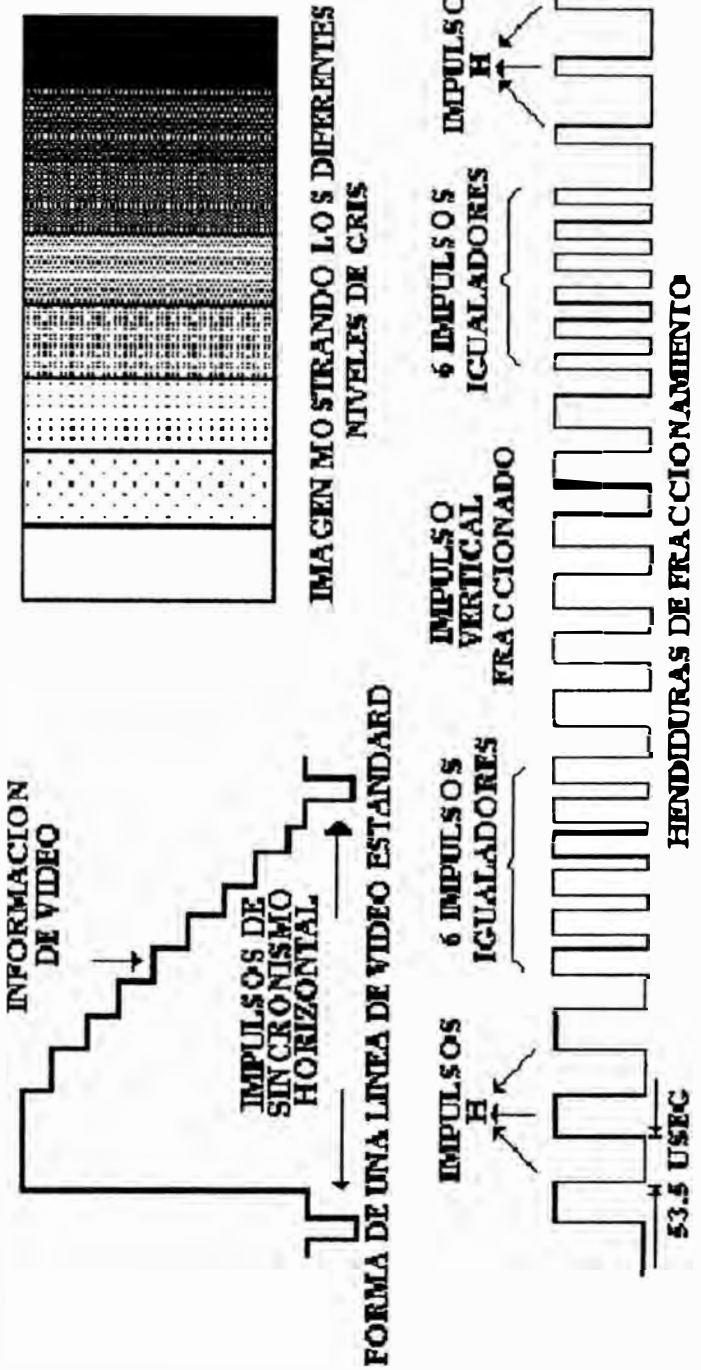


Fig. Nº 2.3 Composición de señales de video utilizados en sistemas de televisión.

se debe disponer de un impulso vertical ancho, que realmente se compone de seis impulsos individuales separados por los fraccionados.

Cada impulso de sincronismo vertical se extiende en un periodo igual a seis medias líneas o tres líneas horizontales completas, por lo que es mas ancho que el impulso horizontal; la razón de este fundamento es para dar a los impulsos verticales una forma completamente distinta de la de los impulsos horizontales. Luego pueden ser completados en forma separada en el extremo receptor, proporcionando las señales de sincronismo horizontal mientras los demás proporcionen solo la sincronización vertical.

Los cinco impulsos, dientes o almenas están insertados en el impulso vertical a intervalos de media línea. Los impulsos igualadores están también espaciados a intervalos de media línea. Estos impulsos de media línea pueden servir para la sincronización horizontal, empleándose impulsos alternados para los campos pares e impares. La razón de emplear impulsos igualadores es facilitar la sincronización vertical. Su efecto es proveer formas de onda idénticas en la señal de sincronización vertical separados para los campos pares e impares a fin de que pueda ser obtenida una sincronización constante y de este modo conseguir un buen entrelazado.

Como podemos observar los impulsos igualadores presentan una característica repetitiva a intervalos de media línea, su velocidad de recepción es dos veces 15.750 cps es decir 31.500 cps; por lo que la frecuencia del impulso horizontal es la mitad que la del impulso de

igualación. Además la frecuencia del impulso vertical es 60/31.500 lo cual es equivalente a 1/525 de la frecuencia del impulso igualador. Asimismo como estas magnitudes son submúltiplos exactos, los impulsos de sincronismo horizontal y vertical se pueden obtener por la división de frecuencia de los impulsos igualadores, de esta forma todos los impulsos de sincronismo se derivan de un generador de señal se sincronismo común en el transmisor y sus frecuencias guardan automáticamente relaciones correctas. Para obtener estas ventajas es recomendable utilizar los divisores de frecuencias para aplicar en los impulsos de sincronismo en vez de utilizar multiplicadores de frecuencia para ondas sinusoidales.

Es preciso aclarar que las señales de sincronización no realizan el proceso de exploración, son importantes los circuitos generador de diente de sierra para proveer la reflexión del haz electrónico que produce la trampa de exploración. Sin embargo los impulsos de sincronismo hacen posible que la información de imagen reproducida en la trama se mantenga estable en la posición correcta; sin el sincronismo vertical, la imagen reproducida aparece desplazándose verticalmente arriba y abajo en la trama; sin el sincronismo horizontal las líneas de los elementos de imagen son reproducidas en segmentos diagonales .

2.3 Señales de video compuesto

La señal de video compuesta es definido como un tipo de señal de video , que esta compuesto por varias partes, entre ellas:

a) La señal de la cámara correspondiente a la información de la imagen deseada.
b) Los impulsos de sincronización para la exploración del transmisor y receptor.
c) Los impulsos de borrado para que los retornos del haz electrónico sean invisibles y no afecten la visión de la imagen.

En la Fig. No 2.4 se muestra la composición de los tres componentes para producir la señal de video compuesta. La señal de cámara es representado en la parte 1, el cual es combinado con el impulso de borrado de 2 y finalmente se superpone el impulso de sincronismo sobre el pedestal del impulso de borrado para producir la señal de video compuesta en 3. El resultado que se describe es una señal de video compuesta para una línea de exploración horizontal. Con este tipo de señal para todas las líneas, la señal de video compuesta tiene la información necesaria para reproducir la totalidad da la imagen presentada.

En la Figura No. 2.5 están representados los valores sucesivos de las amplitudes de tensión o corriente para la exploración de tres líneas horizontales en la imagen.

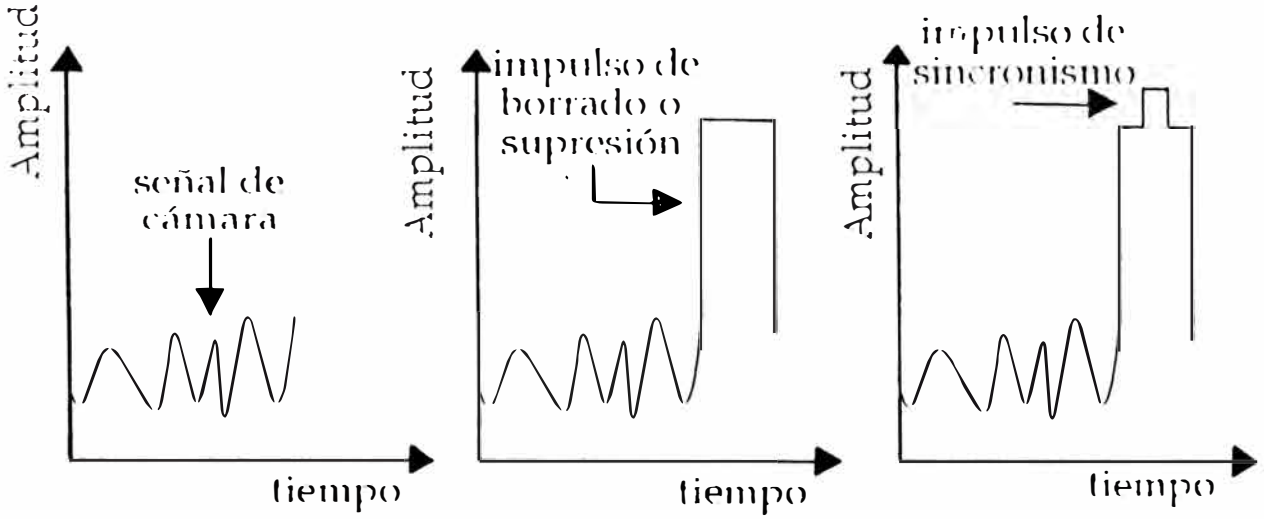


Fig. Nº 2.4 Los tres componentes de la señal de video compuesto.

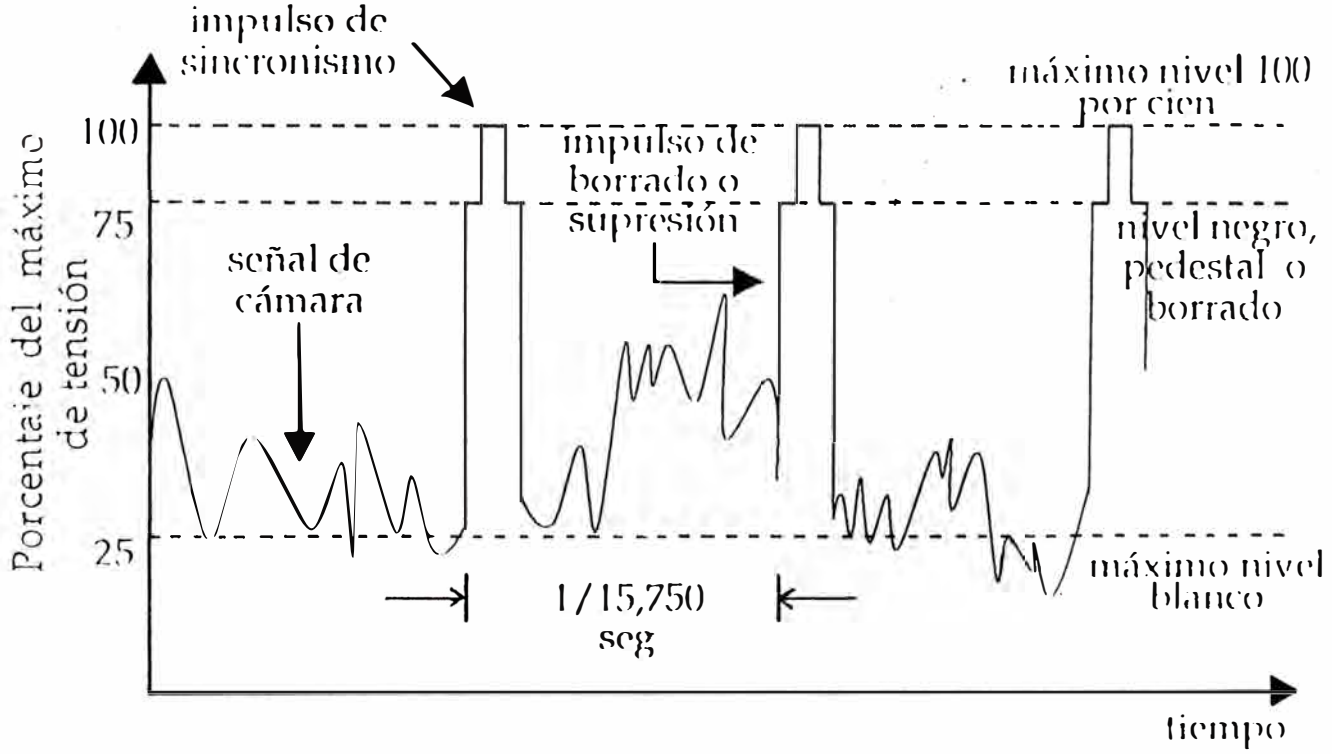


Fig. Nº 2.5 Señal de video compuesto para tres líneas horizontales consecutivos.

2.4 Definición de términos en procesamiento digital de imágenes.

Para realizar el análisis de la señal de video compuesta, empleada en el procesamiento de los sistemas de televisión a color, es necesario definir algunos términos importantes, entre ellos:

a) **Blanco:**

Por aspectos prácticos, la luz blanca puede ser considerada como una mezcla de los colores primario rojo, verde y azul en las proporciones adecuadas. El blanco de referencia para la televisión en color es una combinación que contiene el 30 % de rojo, 59 % de verde y el 11 % de azul, los cuales se mezclan y producen un color blanco azulado como la luz del día.

b) **Matiz:**

El propio color es su matiz, el color de cualquier objeto se distingue principalmente por su matiz, por ejemplo las hojas verdes tienen el matiz verde característico por naturaleza . Los distintos matices son el resultado de las diferentes longitudes de onda de la luz que produce la sensación visual en el ojo.

c) **Luminancia:**

Indica la cantidad especifica de la intensidad de luz, que es percibida por el ojo como nivel de brillo. Luego de las variaciones mas conocidas de luminancia en monocromía en blanco y negro, los colores diferentes se perciben con diferentes valores de brillo por la vista.

d) **Saturación:**

Es el nivel de disolución del color por la luz blanca, en el cual se discriminan los tintes o sombreados vivos y fuertes del mismo matiz. Por ejemplo el azul pastel, tiene poca saturación, mientras que el azul vivo esta altamente saturado. Cuanto mas difiere un color respecto del blanco mayor es su saturación. Asimismo la saturación se indica en términos de pureza y croma, o sea alta pureza y croma corresponden a alta saturación y color vivo.

e) **Crominancia:**

Se define con este termino para indicar el nivel de matiz y saturación de un color que representa la información en los colores rojo, verde y azul. Algunas veces se emplea su sinónimo de cromaticidad.

f) **Registro:**

Es referido a la distribución o posición de las imágenes de color individuales para que la imagen resultante de las mezclas de color presenten el color correcto. En el caso de imágenes superpuestas, si cada una no está colocada exactamente sobre las otras, se producirán mezclas incorrectas de color a causa de que los colores primarios están en posición indebida con respecto a los colores originales de la imagen.

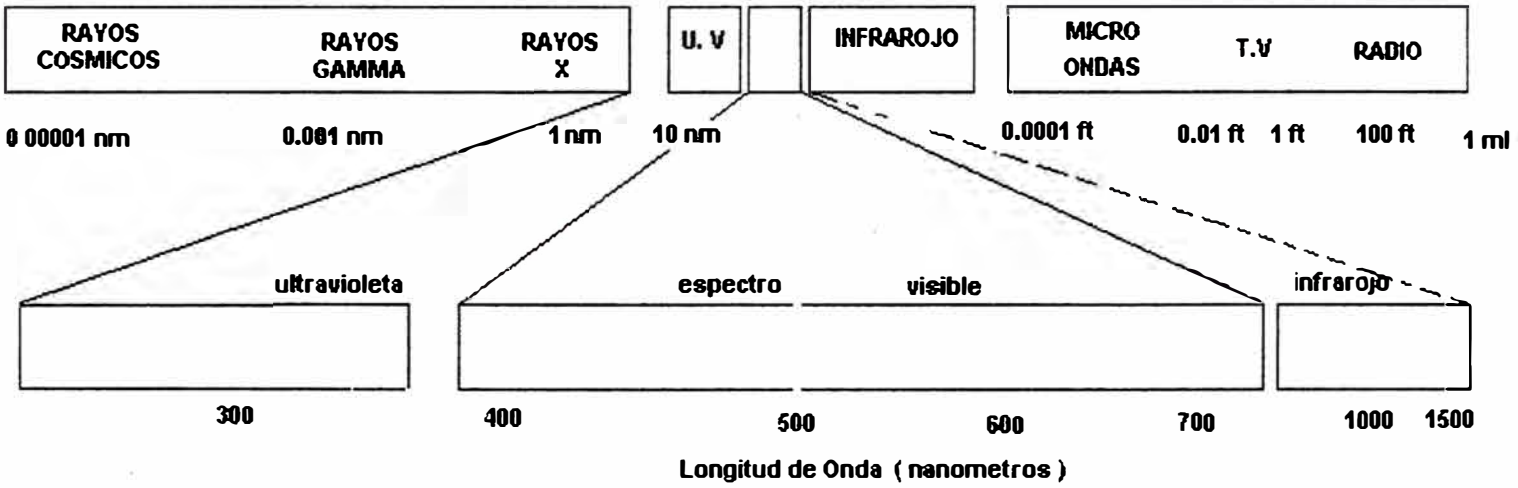
g) **Compatibilidad:**:

El procesamiento de señales de la televisión a color es compatible con la televisión monocromática, debido a que esencialmente ambos utilizan las mismas normas de exploración y la señal de luminancia hace posible que un receptor monocromático reproduzca en blanco y negro la imagen

televisada en color. Además, los receptores de televisión en color pueden utilizar una señal monocromática para reproducir la imagen en blanco y negro. La televisión en color utiliza los mismos canales de difusión de 6 MHz que la transmisión monocromática; asimismo utiliza la misma frecuencia portadora de imagen.

Las características principales para distinguir a cualquier color son : El matiz, la saturación y el brillo. Esencialmente el matiz corresponde a la longitud de onda de la luz, recuérdese que la energía luminosa se puede considerar como una forma de radiación electromagnética pero de frecuencias mucho mas altas que las ondas de radio. Las frecuencias de la luz visible son superiores a 430 x 10 E12 cps. Debido a que estos valores son altos, las ondas de luz se suelen considerar en función de la longitud de onda, utilizando la unidades milimétricas.

Como se muestra en la Fig No.2.6 , el color azul tiene la longitud de onda mas corta, 400 milimicras, el rojo tiene la longitud de onda mas larga, 700 milimicras. Cada color tiene una longitud de onda dominante, que determina su matiz. El color rojo vivo cuando se mezcla con la luz blanca aparece como color rosa, es porque el matiz de los dos colores es el mismo a causa que la longitud de onda dominante no ha cambiado. Sin embargo el color rosa esta menos saturado . El rojo vivo es considerado como ejemplo cien por cien de saturación, no tiene luz blanca , el porcentaje de saturación disminuye y el color se debilita ó desatura.



Sección del espectro de energía electromagnética mostrando el rango de longitud de onda visibles.

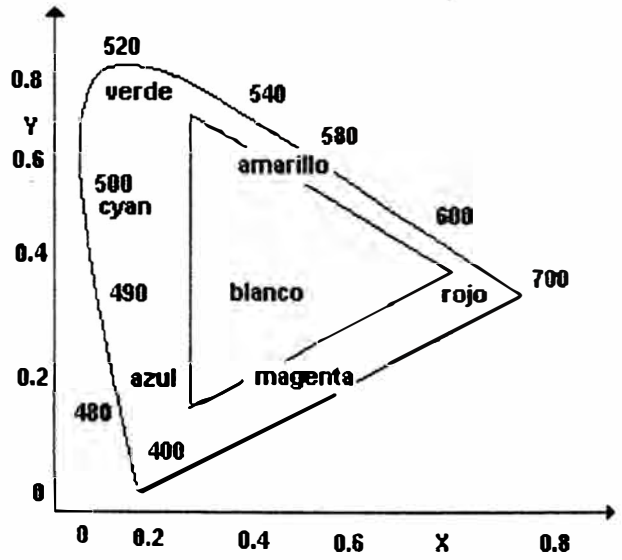


DIAGRAMA DE CROMATICIDAD DE LA CIE

- La linea oscura indica los colores visibles.
- Los números dados son las longitudes de onda y estan expresados en nanometros.

Fig. N° 2.6 Diagrama de cromaticidad de la CIE.

Finalmente la principal características de cualquier color es su brillo, esto indica realmente como aparecerá el color en la reproducción en blanco y negro. Considerando que se toma una fotografía en la representación o se televisa en sistema monocromático.

2.5 Procesamiento de señales de video compuesto.

En la Fig. No. 2.5, están representados los valores sucesivos de las amplitudes de tensión o corriente para la exploración de tres líneas horizontales en la imagen. Se puede observar que la amplitud de la señal de video está dividida en dos secciones, utilizándose la inferior al 75 % para la señal de cámara y la superior al 25 % para los impulsos de sincronización. En la señal de cámara, las amplitudes mas bajas corresponden a las artes mas blancas de la imagen mientras las partes mas oscuras de ésta tienen amplitudes mas altas. Esta es el modo de transmisión de la señal , utilizando un estandar de polarización negativa de transmisión.

La transmisión negativa significa que las partes blancas de la imagen están representadas por amplitudes pequeñas en la señal portadora de ımagen transmitida. Las amplitudes mas altas corresponden a información de imagen progresivamente mas oscura hasta que se alcanza el nivel de negro, el cual está fijado en el 75% de la máxima amplitud de la señal.

Las especificaciones mas importantes de la señal de video compuesta son los siguientes:

a) **Nivel de referencia de negro**:

La característica de señal del color negro es constante en el 75% de su amplitud e independientemente de la información de imagen a fin de obtener una referencia de brillo en el sistema de televisión. En el proceso de reproducción de la imagen, el 75% de la señal de video corresponde a la señal e video el cual corresponde a la tensión de corte de la rejilla del tubo de imagen y la ausencia de luz, ,estableciéndose así un nivel de negro. El 75% de amplitud corresponde también al nivel de pedestal o nivel de ennegrecimiento, a causa de que éste represente sobre los cuales están colocados los impulsos de sincronización. El borrado se realiza en el nivel del negro.

Cualquier amplitud de señal mayor que el nivel de negro se llama infranegro o también ¨**más que el negro**¨, a causa de que ésta tensión hace que la tensión de rejilla del tubo de imagen sea mas negativa que la de corte, por lo tanto los impulsos de sincronización son de infranegro.

b) **Impulsos de borrado**:

Es muy importante disponer de impulsos de borrado en la señal de video compuesta, para hacer que las líneas de retorno sean invisible elevando la amplitud de la señal hasta el nivel de negro durante el tiempo en que los circuitos de exploración produzcan el retroceso. Esto se producen normalmente durante el tiempo de borrado.

En la Fig. No. 2.7, se muestran los impulsos de borrado horizontal y vertical en la señal de video compuesta y los impulsos de borrado horizontal están incluidos para realizar el retorno de derecha a izquierda

en cada línea de exploración horizontal. La velocidad de repetición de los impulsos de borrado es por consiguiente la frecuencia de exploración de línea de 15.750 cps. Los impulsos de borrado vertical tienen la función de disipar las líneas de exploración producidas cuando el haz electrónico retorna verticalmente desde la parte inferior a la parte superior de cada campo. Por consiguiente, la frecuencia de los impulsos de borrado vertical es 60 cps.

c) **Tiempo de borrado horizontal**:

Para analizar los detalles del periodo del borrado horizontal, se muestra la Fig. No.2.7, en donde el intervalo entre las líneas de exploración horizontal esta designado por H, donde este tiempo de exploración de una línea completa incluye la traza y el retorno y es igual a 1/15.750 segundos (equivalente a 63.5 useg) sin embargo, el impulso de borrado horizontal tiene solo una anchura de 0.14 H a 0.18 H. Podemos considerar un valor medio del 16% del periodo de línea como valor típico. El tiempo de borrado horizontal es 0,16 x 63,5 microsegundos para H, que es igual a 10,2 microsegundos: Ahora restando de 63,5 microsegundos obtenemos una diferencia de 53,3 microsegundos con tiempo disponible para la exploración visible, sin borrado en cada línea. Los 10,2 microsegundos correspondientes al borrado brindan el tiempo necesario para el retorno.

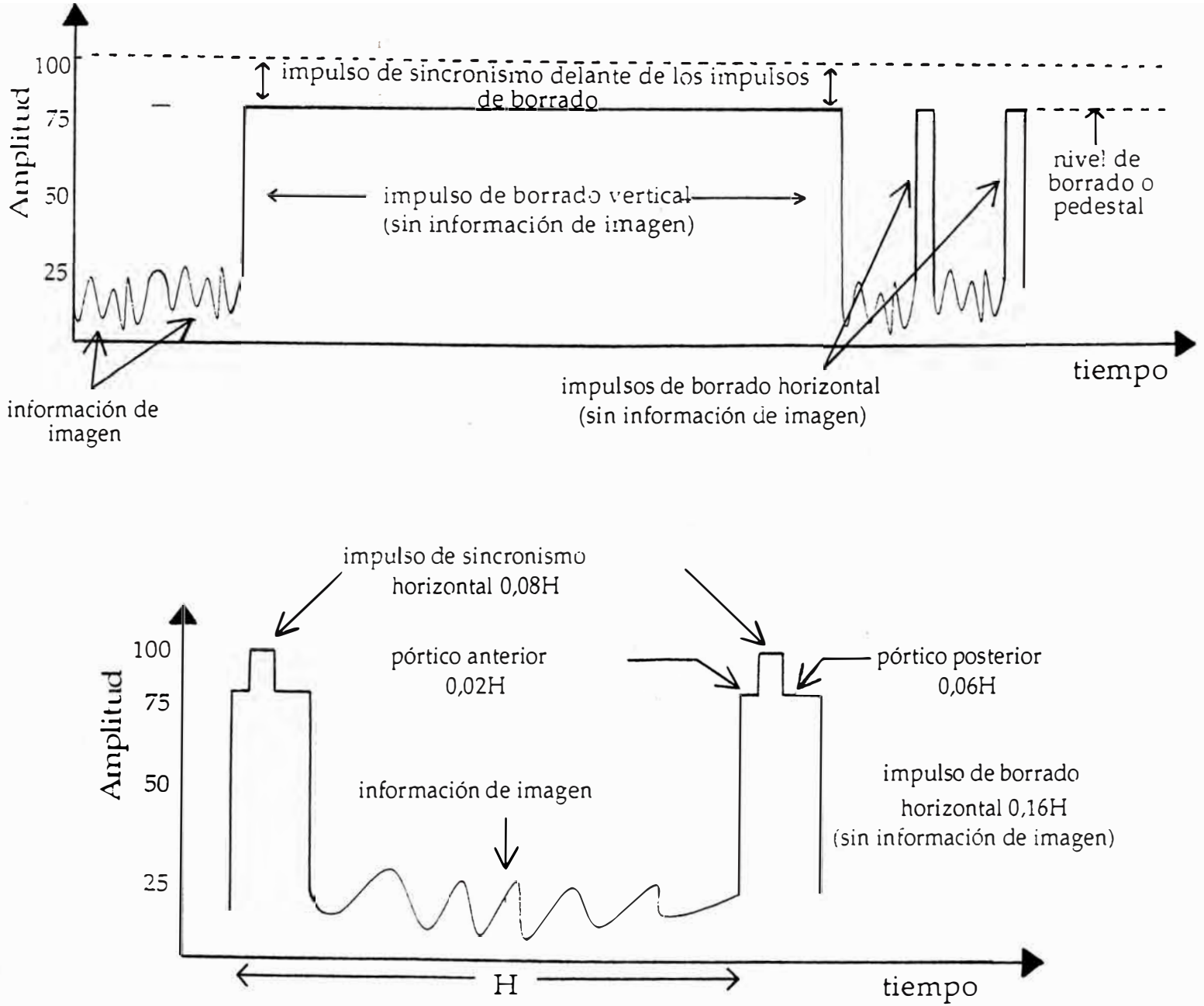


Fig. N° 2.7 Señales de sincronización horizontal.

Superpuestos sobre los pedestales provistos por los máximos de los impulsos de borrado en el nivel del negro están los impulsos de sincronismo, que son mas estrechos. Como se puede observar en la Fig No. 2.7, donde cada impulso de sincronismo horizontal tiene 0,08 H o la mitad de la anchura del impulso de borrado. Este tiempo es igual a 10,2/2 o 5,1 microsegundos.

Para la mitad restante del tiempo de borrado que es también 5,1 microsegundos, la señal está en el nivel pedestal. La parte de pedestal inmediatamente anterior a los impulsos de sincronismo se llama umbral frontal o anterior y el umbral posterior sigue el impulso de sincronismo. EL pórtico anterior es 0,02H o 1, 27 microsegundos y el pórtico posterior 0,06 H o 3,81 microsegundos. Se puede observar que pórtico posterior es tres veces mas largo que el pórtico anterior. En síntesis con 10,2 microsegundos de borrado total, 1,27 microsegundos corresponden al pórtico y 5,1 microsegundos al impulso de sincronismo y 3,81 microsegundos al pórtico posterior.

El objetivo de los impulsos de borrado es obtener que los retornos sean invisibles, por otro lado, el tiempo de borrado es ligeramente mas extenso que los valores típicos del tiempo de retorno, que dependen de los circuitos de deplexión horizontal del receptor. Como consecuencia una pequeña parte de la traza queda borrada ordinariamente al principio y al final de cada línea de exploración. Este efecto de borrado horizontal se representa por las barras negras de los lados de la izquierda y de la

derecha de la traza de la Fig No. 2.7. El borde negro e la derecha corresponde al pórtico frontal de borrado horizontal, antes de que empiece el retorno. Generalmente el borrado horizontal empieza en el borde anterior o de ataque del impulso de sincronismo. Inmediatamente antes del retroceso, cuando el haz explorador esta completando su traza en la derecha, el nivel de borrado del umbral anterior hace que en el borde de la derecha haya una pequeña parte de cada línea ennegrecida de esta manera esta barra negra puede ser considerada como reproducción de la parte del umbral anterior correspondiente al borrado horizontal.

Luego del umbral anterior de borrado, se produce el retorno horizontal cuando comienzan los impulsos de sincronismo. El retroceso queda definitivamente borrado a causa de que el nivel de sincronismo es infranegro, es decir mas negro que el negro; aunque el retroceso empiece con el pulso de sincronismo, el tiempo necesario para completar el retorno depende de los circuitos de exploración. Un tiempo de retroceso horizontal típico es 7 microsegundos. El tiempo de borrado después del umbral anterior es mas largo, igual a 9 microsegundos aproximadamente. Por lo tanto continúan 2 microsegundos de borrado después de completarse el retorno hasta el borde izquierdo. Aunque continua el ennegrecimiento, la forma de onda de la desviación en diente de sierra hace que el haz de exploración empiece su traza después el retorno. En consecuencia, la primera parte de la traza de la izquierda queda ennegrecida, o sea no es visible. Al cabo de dos segundos dl

tiempo de traza correspondiente al ennegrecimiento en el borde de la izquierda, se suprime el impulso de borrado. Entonces la señal de video reproduce la información de imagen cuando el haz de exploración continua su traza durante 53,3 microsegundos del tiempo de traza visible. Sin embargo la pequeña parte de cada línea borrada al principio de la traza forma la barra negra en el borde izquierdo del cuadro o trama. Este borde negro de la izquierda representa parte de cada umbral posterior después del impulso de sincronismo horizontal.

d) **Tiempo de borrado vertical**:

Estos impulsos de barrido vertical se incrementan en amplitud de la señal de video hasta el nivel de negro de modo que el haz electrónico de exploración queda borrado durante los retornos verticales. El ancho del impulso de borrado vertical es 0,05 – 0,08 V = 1/60 seg. Si consideramos el promedio del 6 %, el tiempo de borrado vertical es 0,06 x 1/60 seg, que es igual a 0.001 seg o sea 1,001 microsegundos. Este tiempo es suficiente para incluir muchas líneas de exploración completas. Dividiendo 1,001 microsegundos, el cual representa el tiempo de borrado vertical, por el periodo total de la línea completa de 63,5 microsegundos, se obtiene 15,7 o sea 16 líneas aproximadamente en cada campo, es decir 32 líneas por cuadro. Durante este tiempo relativamente largo quedan borradas no solamente las líneas de retroceso vertical sino también una pequeña parte de la traza vertical en las partes superior e inferior.

Los impulsos de sincronismo incluidos en la señal de video compuesta durante el ancho del impulso de borrado vertical se representan en la Fig. No. 2.8. En ellos están incluidos los impulsos igualadores, los impulsos de sincronismo vertical y algunos impulsos de sincronismo horizontal. Las señales están representadas. durante los intervalos de tiempo comprendidos entre el final de un campo y el principio del siguiente, para mostrar lo que ocurre durante el tiempo de borrado vertical. Las dos señales representadas una encima de la otra son iguales, excepto que existe un desplazamiento de media línea entre los campos. sucesivos, desplazamiento que es necesario para el entrelazado de líneas impares.

En la Fig. No. 2.8, observamos que en la parte izquierda, las cuatro últimas líneas de exploración horizontal en la parte inferior de la imagen están representadas con los impulsos necesarios de sincronismo y de borrado horizontal. Inmediatamente después de la última línea visible, la señal de video como preparación para el retroceso vertical. EL periodo de borrado vertical comienza con un grupo de seis impulsos igualadores que están espaciados a intervalos de media línea. Luego sigue el impulso de sincronismo vertical fraccionado que realmente produce el retroceso vertical en los circuitos de exploración. Estos fraccionamientos o dientes se producen a intervalos de media línea. Por lo que, el impulso de sincronismo vertical completo tiene una anchura de tres líneas. Luego del impulso de sincronismo vertical existe otro grupo de seis impulsos igualadores y un grupo de tres impulsos horizontales. Durante todo el

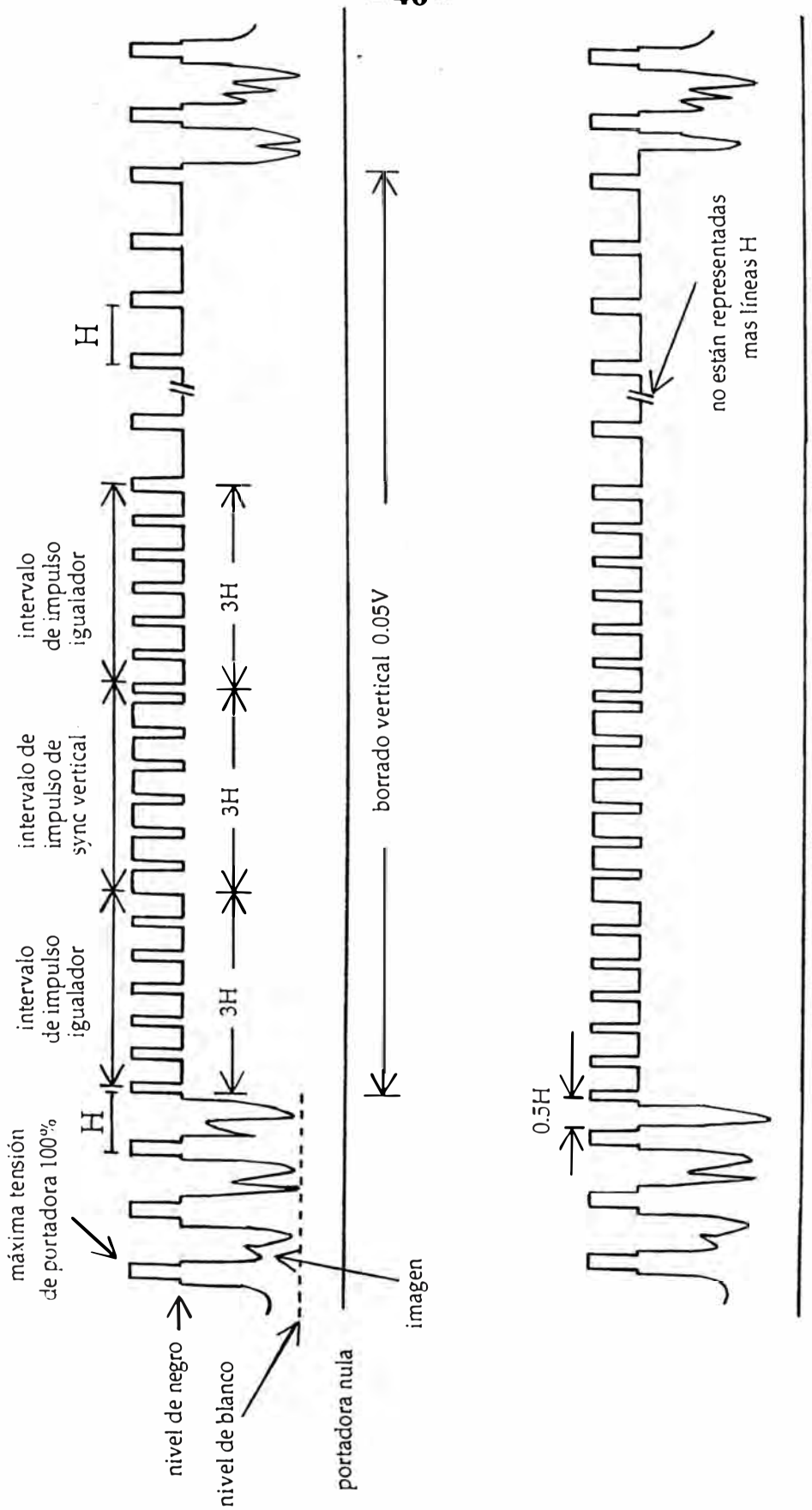


Fig. N°2.8 Diagrama de tiempos del sincronismo vertical.

periodo de borrado vertical no se produce ninguna información de imagen ya que el nivel de señal corresponde al negro y al infranegro de modo que quede borrado el retorno vertical. Se puede observar que la posición del primer impulso igualador al principio del borrado vertical en la Fig. No. 2.8. En la señal de la parte posterior, el primer impulso está separado una línea completa del impulso de sincronismo horizontal anterior en la señal y abajo el correspondiente al campo siguiente, el primer impulso esta separado media línea. Esta diferencia de media línea en el tiempo transcurrido entre los campos pares e impares continua en todos los impulsos siguientes, de modo que los impulsos de sincronismo vertical de los campos sucesivos están temporizados debidamente para el entrelazado de las líneas impares.

El impulsos de sincronismo vertical fraccionado hacen que los circuitos de desviación vertical inicien el retroceso. Sin embargo generalmente éste no comienza con el principio del impulso de sincronismo vertical a causa de que dicho impulso debe completar la carga del condensador para disparar los circuitos de exploración. Si consideramos que el borrado vertical comienza con el borde de ataque o anterior del tercer impulso fraccionado, transcurre el tiempo correspondiente a una línea durante el sincronismo vertical antes de que se inicie el retroceso vertical. Además de los seis impulsos igualadores equivalentes a las tres líneas que tienen lugar antes del sincronismo vertical; entonces son borrados 3 + 1, es decir cuatro líneas en la parte

inferior de la imagen inmediatamente antes de que comienza el retorno vertical.

El tiempo que tarda el retroceso depende de los circuitos de exploración, pero es un tiempo de retorno vertical típico correspondiente a cinco líneas. Cuando el haz de exploración retrocede desde la parte inferior hasta la parte superior del cuadro, se han reproducido cinco líneas horizontales completas. Este retroceso vertical es suficientemente rápido para ser completado dentro del tiempo de borrado vertical.

CAPITULO III
DISEÑO DEL SISTEMA DE ADQUISICION DE SEÑALES DE VIDEO

3.1 Soporte de Memorias de adquisición de datos y comunicación a Computador.

En los capítulos previos, se han analizado las características de la señal de video con el cual estaremos familiarizados en el proyecto, con lo cual podemos definir los procedimientos que debemos efectuar en el diseño del sistema. En el presente proyecto se tiene como objetivo explicar las características del sistema de adquisición de señales de video así como su funcionamiento del mismo, en cada etapa del procesamiento de las señales desde su adquisición. En la Fig No. 3.1 se muestra un esquema de bloques del sistema desarrollado, en donde se podrá definir varias etapas en el proceso de adquisición de un cuadro de video. Definiendo en el diagrama las etapas en mención como:

a) Subsistema de conversión de la señal de video analógica a digital.
b) Subsistema de almacenamiento de un cuadro de video.
c) Subsistema de conversión de la señal de video digital a analógica.

Estos subsistemas conforman el sistema de adquisición de datos de video, pero a la vez guardan su independencia relativa, de tal forma que

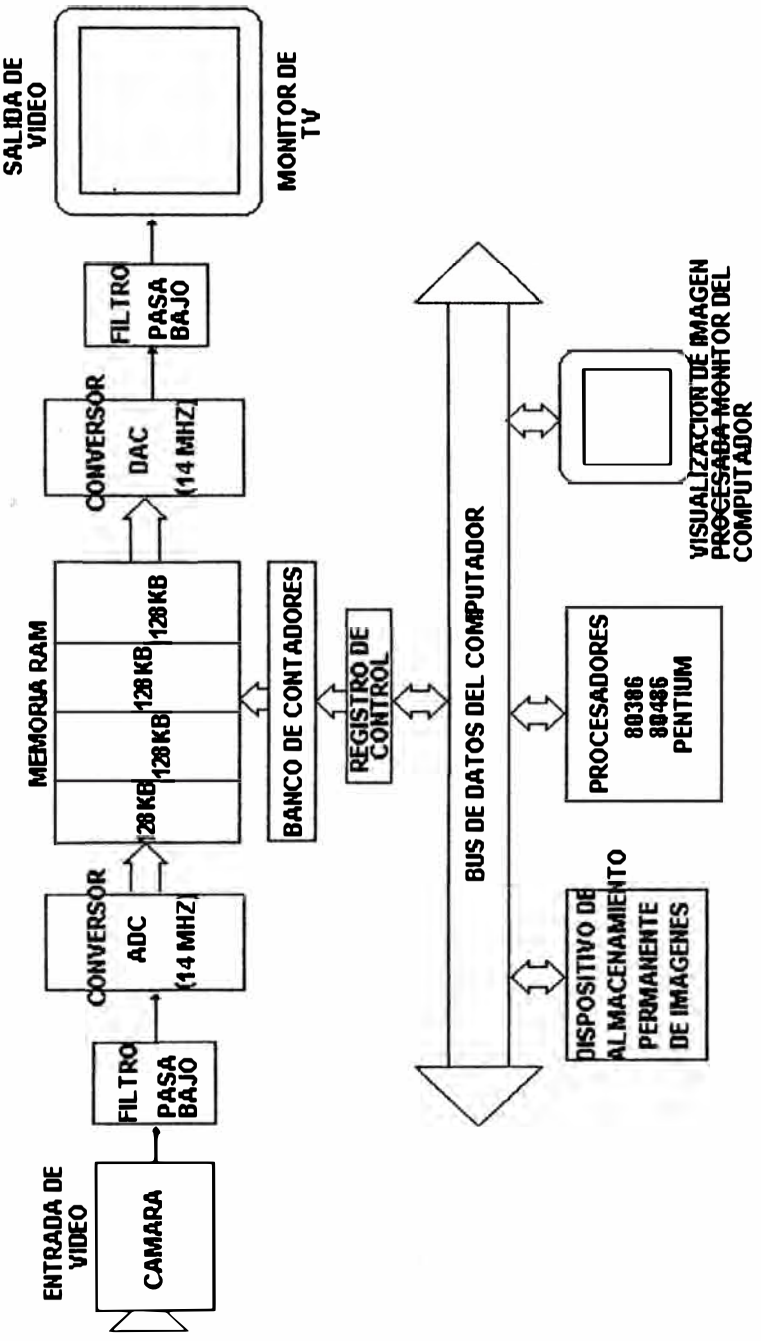


Fig. N° 3.1 Diagrama de bloques del sistema de adquisición de señales de video.

1. Para realizar que la señal de video esta siendo convertida en forma eficiente tendremos que habilitar la etapa ADC (Analógica/Digital) y la conversión DAC (Digital/Analógica)

2. En el supuesto caso de querer la reproducción de alguna imagen de video, previamente digitalizada.

3. En otro caso si se desea almacenar un nuevo cuadro de video en el disco duro del computador, se habilita la etapa de conversión Analógica/Digital y luego la etapa de almacenamiento, de este modo se podrá leer el cuadro capturado luego grabarlo en el disco duro del sistema computador.

Las etapas comprendidas en el diseño del sistema de adquisición de señales de video, son complementadas por un Software desarrollado en el lenguaje Borland C++. Con lo cual se logra controlar a través del port de salida, en donde se guardara una palabra de control para habilitar a cada una de las etapas.

La etapa de conversión de la señal de video analógica a digital, esta diseñado en función a un circuito conversor ADC de alta velocidad (Circuito Integrado MC10319), el cual tiene la frecuencia de funcionamiento máximo en la conversión de 25 MHz.; cuyo valor para nuestro proyecto cumple con las exigencias requeridas.

La etapa de almacenamiento esta compuesto por un banco de memorias SRAM (Ram Estáticas), cuyo tiempo de acceso es de 55 nanosegundos y una capacidad de almacenamiento es de 512 Kbytes.

Por ultimo la etapa del conversor de la señal digital a analógico de video (Conversor DAC) esta implementado en base al dispositivo TDC1016.

Para iniciar nuestro diseño del sistema se desarrolla el análisis de las diversas etapas que conforman el sistema de adquisición de datos de un cuadro de video, los cuales son

Etapa de acceso del computador al banco de memoria del sistema de adquisición

En el proceso de diseño del sistema de adquisición de datos, éste fue el primer subsistema que se desarrolló, y el funcionamiento aceptable de ésta etapa, permite el acceso correcto de la señal de video capturado por parte de sistema computador en los procedimientos de lectura o escritura de la información.

Esta etapa esta conformada por un banco de memoria SRAM (Ram Estáticas) de capacidad de 512 Kbytes, un juego de contadores conectados en forma de cascada, un puerto de entrada y salida de la información, un puerto de salida que almacenará la palabra de control de las diferentes etapas y luego la habilitación de dos puertos para tener control sobre los contadores. En la Fig. No 3.2 se muestra el diagrama de Bloques de este subsistema.

3.1.1 Sistema de memorias ram.

El sistema de adquisición de datos es diseñado de modo que es capaz de almacenar la mínima cantidad de información de una imagen;

es decir, un cuadro de video. Un cuadro de video esta conformado por 2 campos (campo impar y campo par), ambos campos contienen la información de un cuadro de video pero en forma entrelazada, debido a la forma en que los equipos generadores de la señal de video realizan el proceso de barrido. Como se ha analizado, un cuadro de video esta conformado por 525 líneas los cuales son enviados al receptor de video (monitor de televisión) para su visualización.

Un cuadro de video según los análisis anteriores tiene una duración de 33.3 mseg. Y la frecuencia a muestrear por parte de la etapa de conversión ADC es de 14.31818 MHz., esto nos genera una cantidad de 476,142 muestras que se obtendrán cuando se digitalice un cuadro de video. Si cada muestra representa un byte de información entonces se tendrá que tener una capacidad de almacenamiento de un mínimo de 480 Kbytes. Por lo que fue necesario que el tamaño de la RAM sea de 512 Kbytes para evitar que el diseño sea de una lógica de decodificación de la cantidad de memoria exacta requerida para almacenar un cuadro de video, esto hubiese incrementado el hardware del sistema.

La ventaja de utilizar las memorias SRAM se justifica debido a que el tiempo de acceso muy pequeño de este tipo de memorias respecto a otras memorias RAM cuyos tiempos de acceso son muy altos, en la actualidad vemos que es muy pequeño (Del orden de 8, 10, 15 nsegundos) respecto a las DRAM (Memorias RAM Dinámicas) cuyos tiempo de acceso son del orden de 60 y 70 nanosegundos.

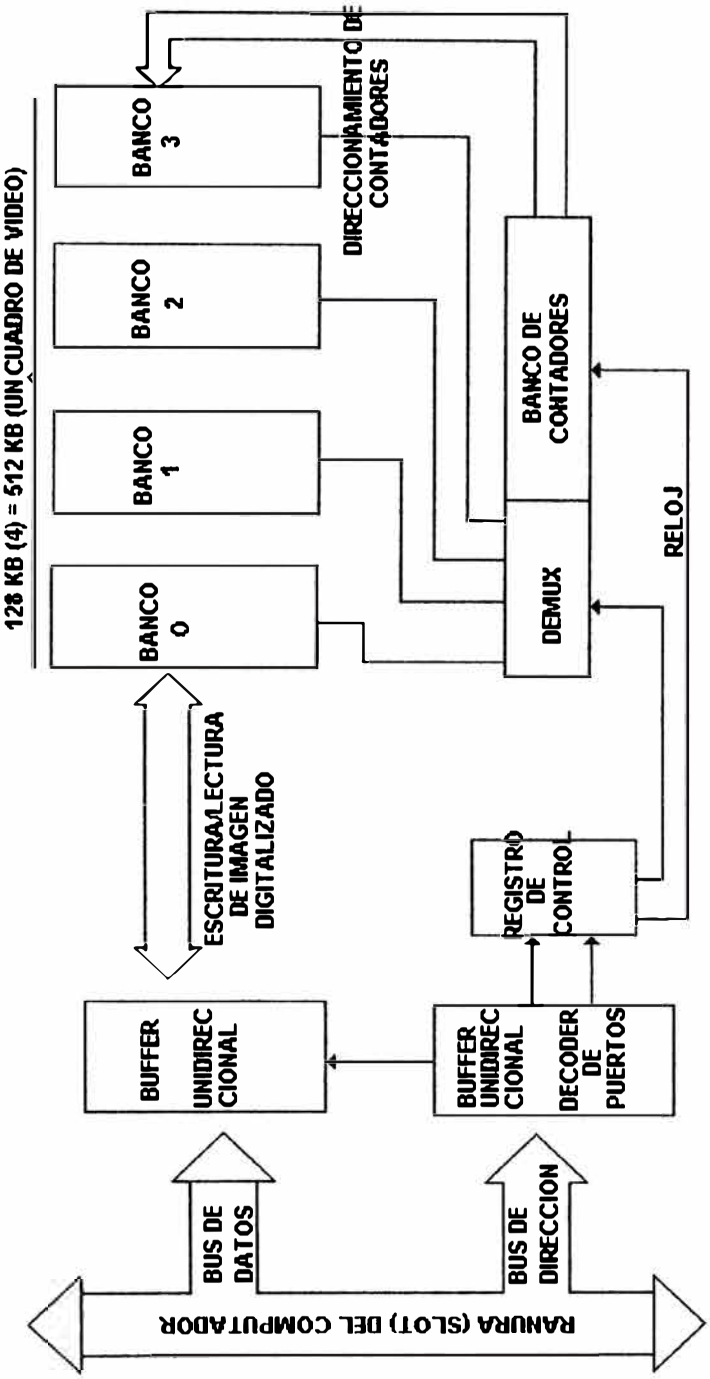


Fig. N° 3.2 Primera etapa del sistema digitalizador de video.

Además de que las SRAM no necesitan soporte de refresco de memorias que si se requieren en las Ram dinámicas, esto evita la pérdida de las muestras que se están adquiriendo por el Conversor ADC por realizar el refresco de memoria y también se tiene la ventaja de la SRAM en la organización de la memoria por lo tanto su fácil direccionamiento.

Para efectuar el acceso al banco de memorias SRAM se realiza la operación secuencial; es decir se almacena la información digital de video en una secuencia sucesiva, en todo el banco de memorias (512 Kbytes) desde la dirección 00000H hasta la dirección 80000H.

La frecuencia de acceso al banco de memorias dependerá de la forma en que se esta trabajando a la tarjeta. Si deseamos realizar la captura de un cuadro de video en el banco de memoria de 512 Kbytes entonces la velocidad de acceso será la frecuencia de muestreo de la señal de video, es decir de 14.31818 MHz; pero si deseamos almacenar el cuadro de video capturado en el disco duro del computador entonces la velocidad de acceso dependerá de la velocidad del microprocesador que se este utilizando.

Las memorias consideradas en la implementación son : S128K-55/9345 cuya capacidad es de 128 Kbytes cada una , por lo que se eplean 4 memorias para obtener un total de 512 Kbytes.

En la Figura No. 3.3 se muestra el diagrama esquemático correspondiente al banco de memoria. La estructura de la memoria S128k-55/9345, posee dos pines de habilitación, los cuales se activan con el nivel logico ¨1¨y el otro con ¨0¨ . Este chip tiene un tiempo de

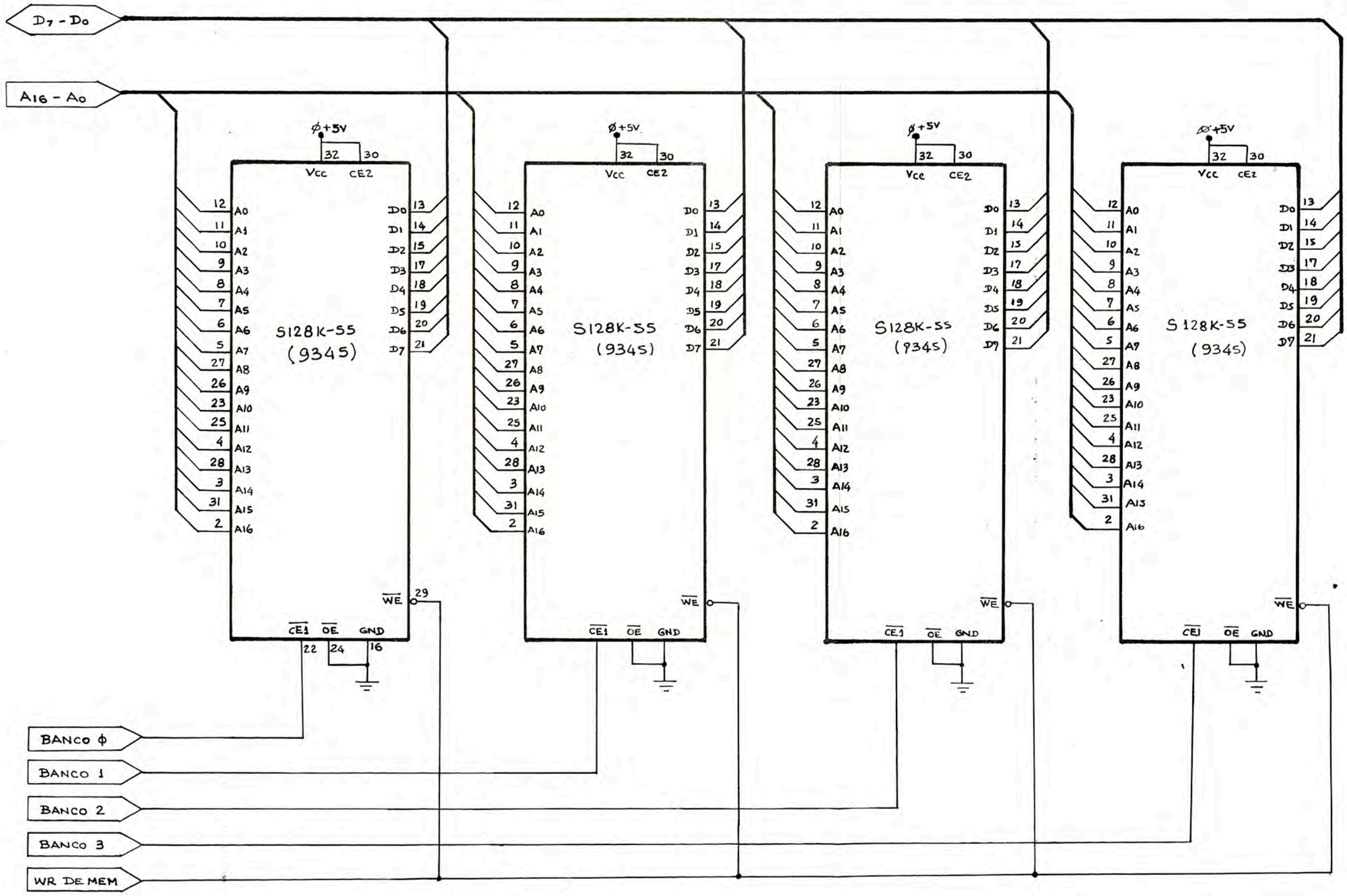

D7 - D0
A16 - A0
+5V
32
30
Vcc
CE2
S128K-55
(9345)
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
D0
D1
D2
D3
D4
D5
D6
D7
WE
CE1
OE
GND
12
11
10
9
8
7
6
5
27
26
23
25
4
28
3
31
2
13
14
15
17
18
19
20
21
29
22
24
16
BANCO ϕ
BANCO 1
BANCO 2
BANCO 3
WR DE MEM

FIG. Nº 3.3

acceso de 55 nanosegundos y esta encapsulado en un circuito integrado de 32 pines los cuales se muestran en el gráfico siguiente:.

Para realizar el direccionamiento del banco de memorias del sistema digitalizador, se realiza por medio de un juego de contadores, que generan las direcciones que se requiere por medio de un juego de contadores, que generan las direcciones que se requiere tanto para leer un dato de memoria o como para escribir un dato. La forma que se genera estas direcciones es de modo secuencial; es decir se comenzara direccionando a la dirección mas baja de memoria : 00000H (Banco 0) hasta la dirección de memoria mas alta 7FFFFH (Banco 3). Esta forma de acceso a la memoria es la mas conveniente, debido a que los datos que se van a almacenar provienen de la etapa conversora ADC, el cual es efectuado en tiempo real.

En el diseño del sistema de adquisición se ha considerado el contador 74LS161, cuyas características lo han realizado de un modo mas adecuado para su aplicación, debido a que es un contador binario y por lo tanto es ideal para el direccionamiento en forma secuencial al banco de memorias , asimismo presenta la facilidad de carga paralela que permite iniciar el acceso a la memoria no necesariamente en la dirección inicial de la memoria 00000H, asimismo tiene una señal de entrada para limpiar los contadores y de esta manera asegurar que el acceso a los datos del banco de memoria se inicie en la primera posición de memoria (dirección 00000H), tiene también una patilla de salida que indica final del conteo, que permite conectar varios contadores en

cascada y así de esta manera generar todo el rango de direcciones del banco de memoria (512 Kbytes) y la respuesta máxima de frecuencia de reloj que permite este contador es de 25 MHz, lo cual es suficientemente rápido para direccionar las memorias

La capacidad real del banco de memorias de 512 Kbytes es de 524,288 direcciones, los cuales para direccionar utilizando los contadores 74LS161 que permite disponer de 4 bits y para direccionar el total de memoria se requieren 19 Bits por lo que es necesario utilizar un banco de contadores compuesto por 5 Contadores 74LS161 conectados todos en cascada, tal como se muestra en la figura 3.4, en donde se muestra con detalle la interconexión realizada el circuito de los contadores. Para activar el banco de contadores se realiza mediante una señal proveniente del multiplexor 74LS151, la cual selecciona la velocidad con que serán activados los datos al banco de memoria. Por lo tanto si deseamos digitalizar un cuadro de video, seleccionaremos al multiplexor para que deje pasar la señal de reloj proveniente del cristal de 14.31818 MHz. A los contadores; y si queremos leer los datos del banco de memoria para almacenarlos en el disco duro del computador, seleccionaremos al multiplexor para que deje pasar la señal de reloj proveniente de la decodificación del puerto de entrada/salida de datos; es decir dependerá de la velocidad del microprocesador empleado del sistema computador.

3.1.2 Soporte decodificador de puertos de entrada/salida de Adquisición de datos.

Para efectuar el diseño del sistema de adquisición de señales de video, se requiere el uso de un puerto de tipo bidireccional , para tener la facilidad de operar los procesos de lectura y/o escritura de los datos del sistema digitalizador cuando interactúa con el sistema computador y deberá tener puertos adicionales para el control de las diferentes tareas (procesos de adquisición, reproducción, lectura y escritura de la información digitalizada). En síntesis se pueden resumir que necesitamos decodificar los puertos siguientes:

a) Un puerto de entrada/salida (I/O) de la información digital de video, localizada en la dirección 0300H.
b) Un puerto que contenga la palabra de control, para el gobierno del sistema de adquisición , localizada en la dirección 0301H.
c) Un puerto de carga paralela de la dirección de carga paralela de la dirección de memoria 0BC11H, localizado en la dirección 0302H.
d) Un puerto de salida para limpiar los contadores, para una nueva cuenta, localizada en la dirección 0303H.

En la decodificación de los puertos considerados, se han empleado las líneas de direcciones del microprocesador que están disponibles en cualquier slot (Ranura de expansión) del sistema computador Pentium MMX-266. El rango de direcciones que se consideró es la que esta disponible para los usuarios, esto es, desde la dirección 0300H hasta la dirección 031FH (32 direcciones), de todas ellas se utilizo solamente 4

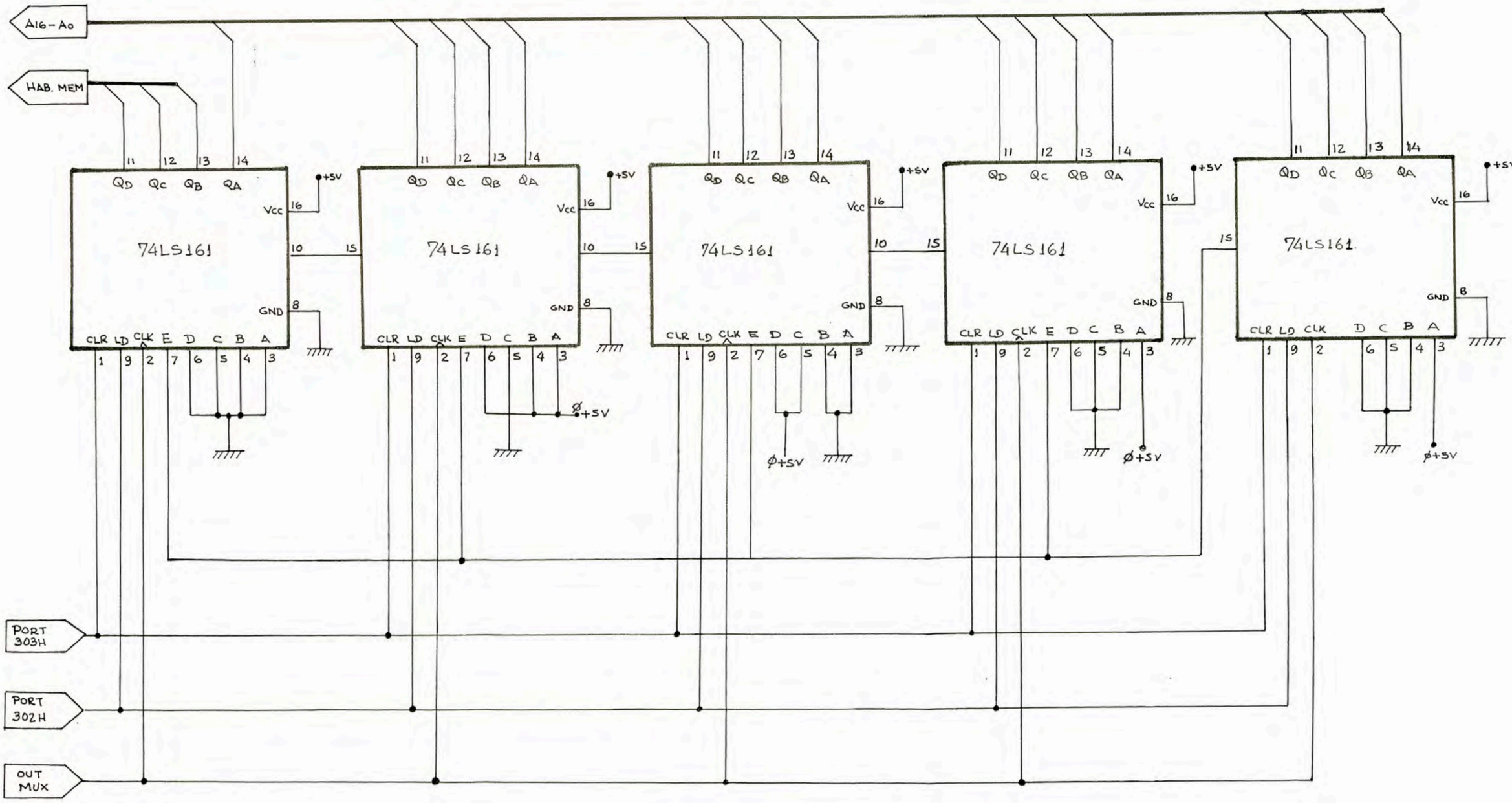

A16 – A0
HAB. MEM
74LS161
QD QC QB QA
Vcc
+5V
GND
CLR LD CLK E D C B A
PORT 303H
PORT 302H
OUT MUX

FIG. Nº 3.4

direcciones (0300H, 0301H, 0302H, 0303H). En la Fig No. 3.5 se muestra el esquema circuital de la decodificación de los puertos a utilizar.

Desarrollando la secuencia del diseño se detalla la función de cada uno e los puertos utilizados en el sistema digitalizador de las señales de video.

Puerto de salida de la dirección 0301h

Este Puerto de salida se emplea para efectuar el control de las diferentes etapas de la tarjeta de adquisición de datos de video (etapa de conversión ADC, etapa de conversión DAC y etapa de acceso al banco de Memoria). El cual esta Constituido por un latch 74LS374 de 8 bits el cual es habilitado en la dirección del puerto 0301H.

Cada uno de los 8 bits tiene una función especifica y esta se indica a continuación:

PALABRA DE CONTROL:

D0	:	0	HABILITA EL BANCO DE MEMORIA
		1	DESHABILITA EL BANCO DE MEMORIA
D1	:	X	NO IMPORTA
D2	:	X	NO IMPORTA
D3	:	0	RELOJ DEL PC
		1	RELOJ DE 14.31818 MHz.
D4	:	0	HABILITA EL MUX DE RELOJ
		1	DESHABILITA EL MUX
D5	:	0	ESCRITURA EN BANCO DE MEMORIAS
		1	LECTURA EN BANCO DE MEMORIAS
D6	:	0	HABILITA CONVERSOR DAC
		1	DESHABILITA CONVERSOR DAC
D7	:	0	HABILITA CONVERSOR ADC
		1	DESHABILITA CONVERSOR ADC.

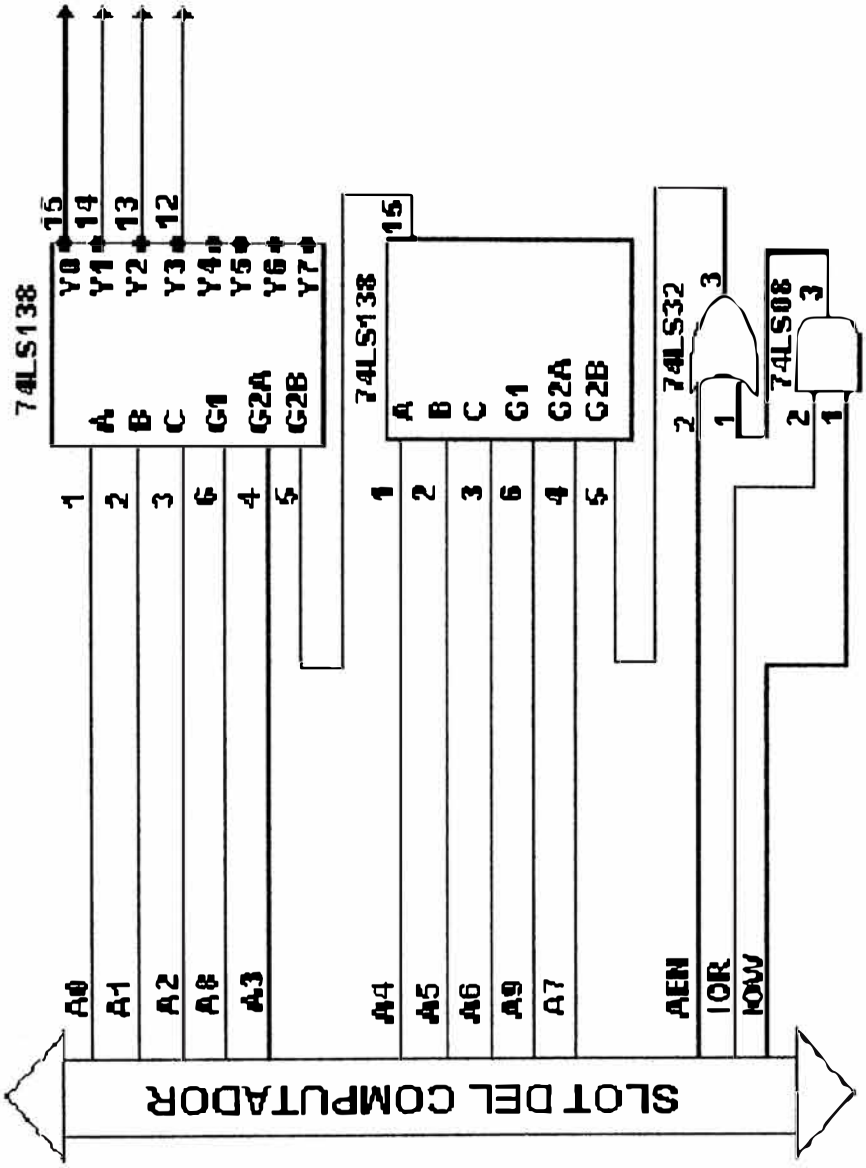


Fig. Nº 3.5 Decodificación de los puertos de Entrada/salida.

Para efectuar las diferentes formas en que se puede operar el sistema digitalizador, solo es necesario enviar los bits correspondientes del registro de control a la dirección 0301H, y de este modo queda establecido la forma de programación del registro de control y la forma en que opera el sistema digitalizador de las señales de video..

A continuación se detallan las palabras de control mas importantes, que se podrán utilizar para gobernar las diversas etapas del sistema.

PALABRA DE CONTROL	DESCRIPCION
29H	Lazo cerrado entre la etapa ADC y la etapa DAC.
08H	Conexión entre la etapa de conversión ADC y la etapa de conversión DAC almacenando la información en el banco de memoria.
A8H	Proceso de lectura del banco de memoria hacia la etapa del DAC.
E0H	Proceso de lectura del banco de memoria y Almacenar la informacion en el disco duro del sistema computador.

2. Puerto de entrada/salida de datos del computador (0300h).

Utilizando este puerto se efectúa el proceso de lectura de los datos que se encuentran en los bancos de memoria hacia el microprocesador para que luego sean almacenados en un banco de fichero de tipo binario en el disco duro, del mismo modo, si se desea efectuar el proceso de lectura de un fichero de tipo binario que contiene información

lectura de un fichero de tipo binario que contiene información digitalizada de un cuadro de video, para este caso se escribirán esta información en el banco de memoria utilizando el mismo puerto (300H). El puerto esta conformado por un circuito integrado buffer bidireccional, el cual es implementado por el IC TTL 74LS245 cuyo arreglo se muestra en la Fig. No. 3.6. Aquí se puede apreciar que este buffer bidireccional se habilita con la dirección 300H y la patilla de selección de la dirección lo determina en sistema microprocesador mediante la señal de IOR (Lectura del puerto de entrada/salida).

3. PUERTO DE CARGA DE DATOS A LOS CONTADORES (302H)

La función de este puerto, es la de cargar un dato prefijado a los contadores. Este dato que se cargará en los contadores es 0BC11H. Este número representa la diferencia que existe entre los 512 kBytes de memoria disponible en el banco y el tamaño exacto de un cuadro de video, que según se ha analizado en el capitulo anterior es de 480 Kbytes.

Para lograr cargar a los contadores con este valor prefijado se deberá habilitar el puerto 0302H, sin necesidad de enviar una palabra de control. Esto se debe a que los contadores (74LS161) tiene sus entradas paralelas colocadas fijamente con el valor 0BC11H. Las direcciones del banco de memoria que se encuentran antes de la posicion 0BC11H no serán consideradas cuando se digitalice un cuadro de video.

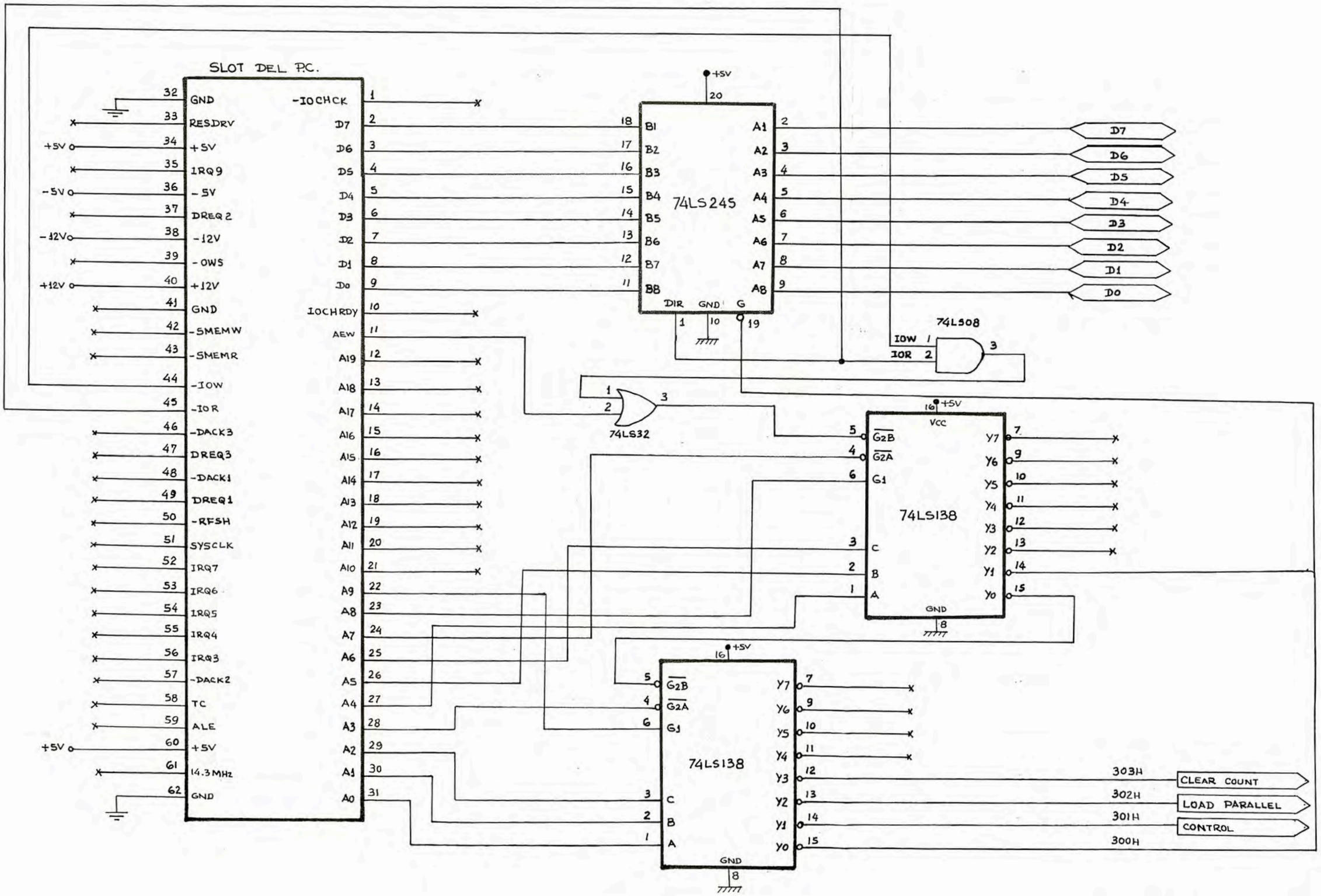

SLOT DEL P.C.
GND
RESDRV
+5V
IRQ9
-5V
DREQ2
-12V
-0WS
+12V
GND
-SMEMW
-SMEMR
-IOW
-IOR
-DACK3
DREQ3
-DACK1
DREQ1
-RFSH
SYSCLK
IRQ7
IRQ6
IRQ5
IRQ4
IRQ3
-DACK2
TC
ALE
+5V
14.3MHz
GND
-IOCHCK
D7
D6
D5
D4
D3
D2
D1
D0
IOCHRDY
AEW
A19
A18
A17
A16
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
74LS245
B1
B2
B3
B4
B5
B6
B7
B8
A1
A2
A3
A4
A5
A6
A7
A8
DIR
GND
G
74LS08
IOW
IOR
74LS32
74LS138
G2B
G2A
G1
C
B
A
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0
Vcc
303H
CLEAR COUNT
302H
LOAD PARALLEL
301H
CONTROL
300H

FIG. Nº 3.6

4. Puerto para limpiar a los contadores (0303h)

La función de este puerto es la de limpiar a los contadores después de un determinado conteo. El uso de este puerto esta orientado al acceso por parte del computador al banco de memoria, ya sea para leer datos como para escribirlos. Al igual que el puerto anterior no necesita ninguna palabra de control para limpiar los contadores.

3.1.3 Software de soporte de acceso de datos a computador.

El diagrama circuital de la Fig. No 3.7 muestra el desarrollo completo del soporte de la etapa de acceso por parte del computador al banco de memorias de la tarjeta digitalizadora. En ella se muestra la lógica de decodificación de los diferentes puertos, así como también el registro de control de la tarjeta y el buffer bidireccional a través el cual el computador leerá o escribirá un dato en las memorias. En el diagrama también se muestra la interconexión del banco de contadores con el sistema de memorias.

Una vez completado el diseño de esta etapa del sistema de adquisición de datos se procedió a efectuar la implementación, para su posterior prueba en el computador. Para lo cual se desarrollo un programa de prueba utilizando el lenguaje de Programación Borland C++ , con el cual se desarrolló el archivo "PRUEBA.CPP".

En la Fig. No. 3.8 se muestra el desarrollo del Diagrama de Flujo del programa. Este programa escribe un dato patrón en todo el banco de memorias (512 Kbytes), este dato patrón representa la cuenta numérica

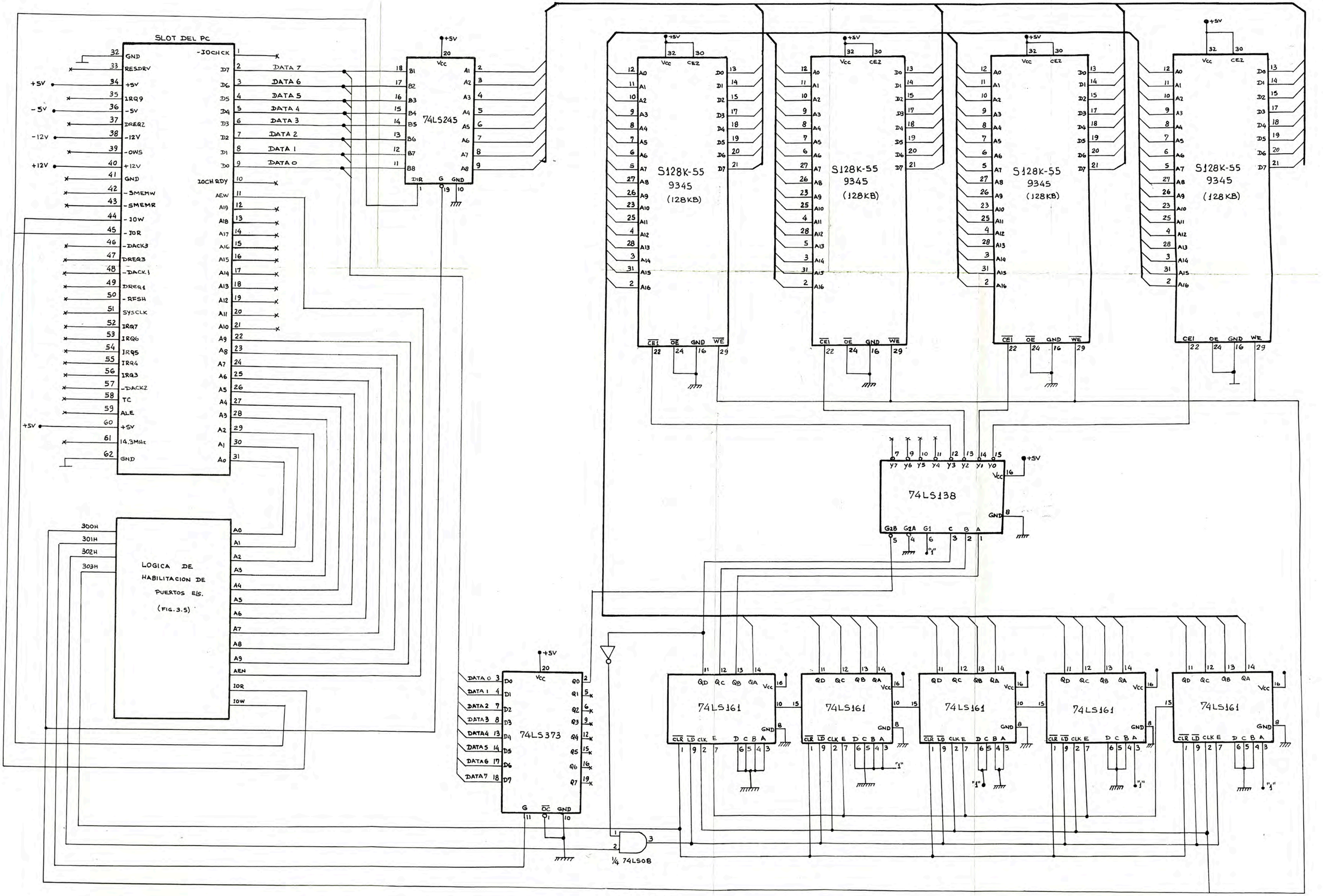

SLOT DEL PC
GND
RESDRV
+5V
IRQ9
-5V
DREQ2
-12V
-OWS
+12V
GND
-SMEMW
-SMEMR
-IOW
-IOR
-DACK3
DREQ3
-DACK1
DREQ1
-RFSH
SYSCLK
IRQ7
IRQ6
IRQ5
IRQ4
IRQ3
-DACK2
TC
ALE
+5V
14.3MHz
GND
-IOCHCK
D7
D6
D5
D4
D3
D2
D1
D0
IOCHRDY
AEN
A19
A18
A17
A16
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
DATA 7
DATA 6
DATA 5
DATA 4
DATA 3
DATA 2
DATA 1
DATA 0
74LS245
Vcc
DIR
G
GND
300H
301H
302H
303H
LOGICA DE HABILITACION DE PUERTOS E/S. (FIG. 3.5)
S128K-55 9345 (128KB)
CE1
OE
WE
CE2
74LS373
74LS138
74LS161
CLR
LD
CLK
E
OC
1/4 74LS08

FIG. Nº 3.7

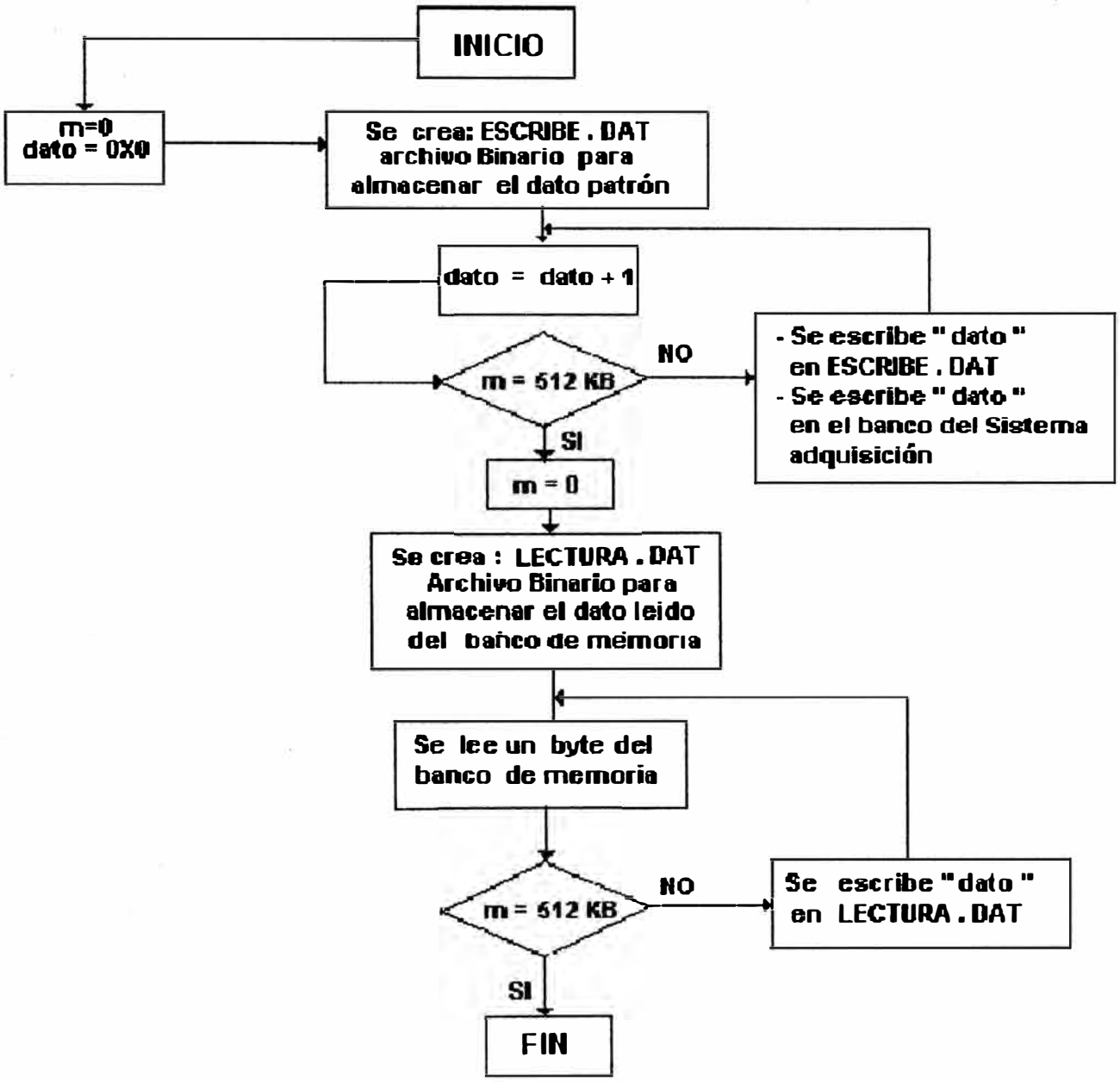


Fig. No. 3.8 Diagrama de flujo del programa Prueba.CPP.

de 00H a FFH el cual se escribe en el banco de memoria, como en el disco duro del computador en un archivo llamado ¨ESCRIBE. BAT¨. Luego se procede a leer del banco de memoria el dato patrón escrito previamente, guardando la información leída en otro archivo denominado¨LECTURA. DAT".

Finalmente se comparan los dos archivos creados utilizando el comando externo del DOS FC.EXE el cual debe entregar el mensaje de que ambos archivos no se encuentran ninguna diferencia, caso contrario la etapa de los contadores se encuentra en mal estado de implementación, o también puede ser por mal estado de contadores

El programa es el siguiente:

PROGRAMA PRUEBA.CPP

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>

/*      ; Puerto de control de la tarjeta de adquisicion.
        ; Direccion del puerto 300H: Lectura y Escritura de datos de
        ; la memoria al computador (PC).
        ; Direccion del puerto 301H
        ; D0:        ¨1¨ Habilita a las memorias estaticas MT5C1008
        ;           ¨0¨ Deshabilita a las memorias.
        ; D2-D1:     No intervienen en este proceso
        ; D3 :       ¨0¨ Deja pasar la señal de reloj del decoder de
        ;            direcciones del PC:
        ;           ¨1¨ Deja pasar la señal de reloj del cristal
        ;           (14.31818 MHz).
        ; D4 :       ¨0¨ Habilita el Mux de los relojes (74LS151).
        ;           ¨1¨ Deshabilita al Mux.
        ; D5 :       ¨1¨ Habilita a las memorias para la lectura.
        ;           ¨0¨ Habilita a las memorias para la escritura.
```

```
; D6 :        "0" Habilita el puerto para salida de datos al DAC.
; D7 :        "0" Habilita el puerto para entrada de datos del
ADC.
; Direccion del puerto 302H : Se utiliza para setear a los
contadores
;                                   con el valor 0CB11H.
; Direccion del puerto 303H : Se utiliza para borrar los
contadores.*/

main ()
(       /* DEFINICION DE LAS VARIABLES A UTILIZAR */
        int write_mem           =0xCO;
        int read_mem            =0xE0.
        Int clear_count         =0x0;
        Int port0               =0x300;
        Int port1               =0x301;
        int port3               =0x303;
        FILE in,*ou;
         Unsigned char dato     =0x0;
        /* Inicio del programa */
        printf ("\nPulse una tecla para escribir los datos en memoria\n");
        printf ("y escribirlos en el archivo ESCRIBE.DAT ....\n");
        getch ();
        in=fopen (ËSCRIBE.DAT", "wb"); /* crea el archivo
ESCRIBE:DAT */
        outport (port3,clear_count);
        outport(port1,write_mem);
        /* lazo que escribe el dato en el archivo creado y en el banco de
        memoria de la tarjeta de adquisicion */
        for (int j=0; j < =7; j++))
                for (unsigned long I=0;I<=65535;I++){
                        outportb(port0,dato);
                        fprintf(in,"%c",dato);
                        dato++;
                        }
        }
        printf("\nPulse una tecla para leer los datos de memoria\n");
        printf(" y escribirlos en el archivo LECTURA.DAT...\n);
        nosound();
        getch();
```

```
        ou=fopen("lectura.dat","wb");      /*      crea      el      archivo
LECTURA.DAT */
        outport(port3,clear_count);
        outport(port1,read_mem);
        /* lazo que escribe y lee el dato de la memoria del sistema de
                adquisicion de datos y almacena en el archivo creado */
        for(int  k=0;k<=7;k++){
                for(unsigned long m=0;m<=65535;m++){
                        dato=inportb(port0);
                        fprintf(ou,"%c",dato);
                        }
                }
        fclose(ou);
        sound(1000);
        printf("Final del programa\n");
        nosound();
        return 0;
}
```

3.2 Diseño de la etapa de conversion analogo/digital.

Este subsistema es el encargado de realizar la conversión de la señal analógica a señal digital de la imagen de video, el cual es proveniente de cualquier equipo que genere dicha señal (cámara de video, generador de barras, videograbadoras, etc.). Como se ha descrito un conversor ADC toma un voltaje de entrada analógica y luego de un tiempo se produce un código de salida que representa la entrada analógica.

El diseño de esta etapa se fundamenta en el conversor ADC MC10319 de resolución de 8 bits de la tecnología MOTOROLA. Este conversor es del tipo FLASH paralelo y está diseñado especialmente para trabajar en procesamiento de señales de video. Este tipo de conversor es el más veloz disponible en la actualidad, pero requiere de

mucha mas circuiteria que otros conversores ADC, internamente esta implementado de 255 comparadores internos.

La señal de video aplicado a este conversor es de 1 Vpp y es muestreada a una frecuencia de 14.318 MHz el cual es proveniente de un oscilador. Esta señal es aplicada simultáneamente a cada uno de los 255 comparadores a través de la patilla 14 como se muestra en la Fig 3.9. las otras entradas de los comparadores están conectadas a los voltajes de referencia que se obtiene con los resistores conectados a los voltajes de referencia que se obtiene con los resistores colocados en red de escalera. Los comparadores de este integrado tienen un ancho de banda mayor a 50MHz lo cual es mas que suficiente para cumplir con el teorema de Niquist, ya que nuestra señal de entrada tiene un ancho de banda de 4.2 MHz.

La polarización de este circuito conversor en con un voltaje de +5.0 Volt con +/- 10 % de tolerancia para los comparadores : VCC(A) y para la porción digital VDD(A) y entre los dos debe haber como máximo 100 mv de diferencia uno a otro. Existe indirectamente un acoplamiento interno entre VCC(A) y VCC(D). La corriente ICC(A) es nominalmente de 17 mA, y esto ésto no varia con la frecuencia de reloj ó con el voltaje de entrada (Vin), pero varia linealmente con VCC(A), ICC(D) es nominalmente de 90 mA y es totalmente independiente de la frecuencia de reloj. Varia linealmente con VC(D). También tiene una alimentación negativa de VEE y esta dentro del rango de '3.0 V a –6.0 V. Adicionalmente VEE debe estar 1.3 V mas negativo que VRB. La IEE

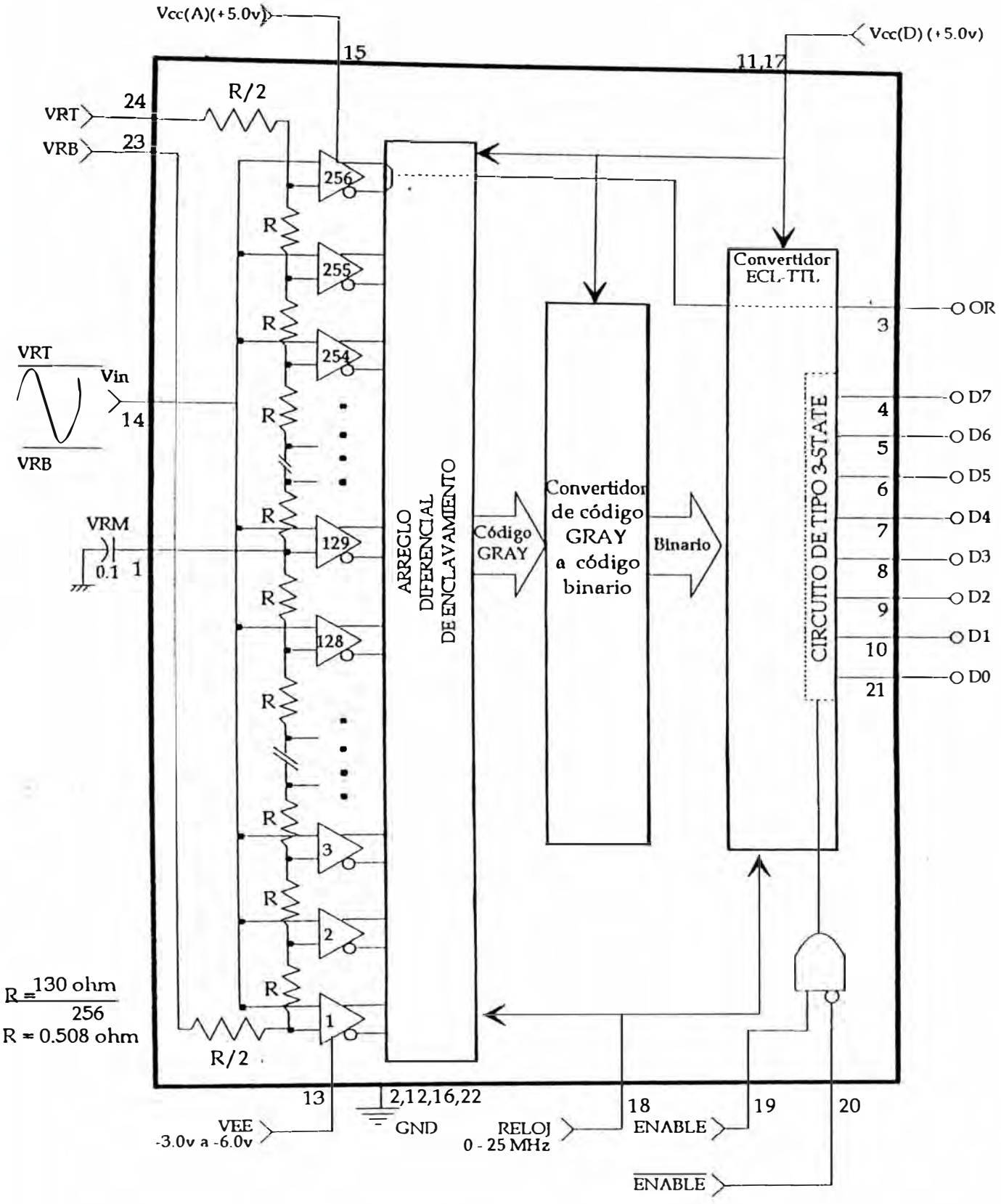


Fig. N°3.9 Estructura del ADC-MC10319

es nominalmente de -10mA y es independiente de la frecuencia de reloj, Vin y VEE. habra que indicar que en el diseño de esta etapa, las fuentes de voltaje provienen del computador (+5v, -5v y Gnd).

La señal de reloj es compatible con niveles TTL, con u n rango de frecuencia de 0 a 30MHz. El ciclo de trabajo (Duty Cicle) no tiene limitaciones. Este dispositivo además cuenta con dos habilitadores y son utilizados para el cambio de los datos de salida (D/-D0) activos a 3-state). Los habilitadores solo afectan a los estados de salidas, pero no inhiben la conversión.

La señal máxima analógica que se puede aplicar a este dispositivo conversor es de 2 Vpp y se determina aplicando un voltaje continuo, en los pines VRM (patilla1) y VRB (patilla 23).

Este dispositivo conversor convertirá cada muestra en aproximadamente 70 nseg., produciendo un numero proporcional al brillo de la imagen que se esta adquiriendo, éste valor estará en el rango de 0 a 255. Este numero es inmediatamente almacenado en el banco de memoria. Hay que tener en cuenta que en el momento de implementar este circuito, la placa debe tener un buen circuito a tierra.

Sistema de filtros

La aplicación de los filtros en este proceso, es para separar las señales útiles de otras que no se deseen procesar, para realizar el diseño de un filtro es necesario conocer el espectro de frecuencias que se requiere separar.

Un filtro ideal sería aquel circuito que trasmitiera todas las componentes útiles sin ninguna atenuación y sin desfasaje, eliminando a la vez todas las señales indeseables. La importancia de realizar un buen diseño en la implementación de un filtro esta dada por ser la primera etapa en cualquier proceso digital de señales o en una etapa en la que la selección de señales e vital para el tratamiento de la señal analógica.

La señal de video es filtrada antes de ser muestreada y cuantizada por el conversor ADC, la razón del filtro anti-aliasing es eliminar todas las componentes de frecuencias de la señal de video que exceden a la mitad de la frecuencia de muestreo (fin = 14.318 MHz), nuestra velocidad de muestreo para una señal con especificación RS-170 es 1/69.84 ns = 14.318 MHz; esto significa que las frecuencias que están sobre la frecuencia de 7.159 MHz pueden ser eliminadas. Las frecuencia por sobre este valor, pueden degradar a la imagen digital. Se puede implementar un filtro anti-aliasing como un filtro pasa-bajo cuya frecuencia de corte es la frecuencia de Nyquist (la mitad de la frecuencia de muestreo).

En el diseño del filtro, se selecciono a un filtro pasabajo de primer orden. Estos filtros tienen una respuesta en frecuencia plana en la banda de paso, eliminando los componentes de altas frecuencias. Sobre la frecuencia de corte la atenuación del filtro se incrementa a 20 db/decada o 6 db/octava por cada polo de la función de transferencia. En la fig No. 3.10 se muestra el esquema del filtro pasa bajo y su respuesta en frecuencia.

En la Fig. No. 3.11 se muestra el desarrollo de la etapa de conversión ADC de señales de video desarrollados en el presente proyecto.

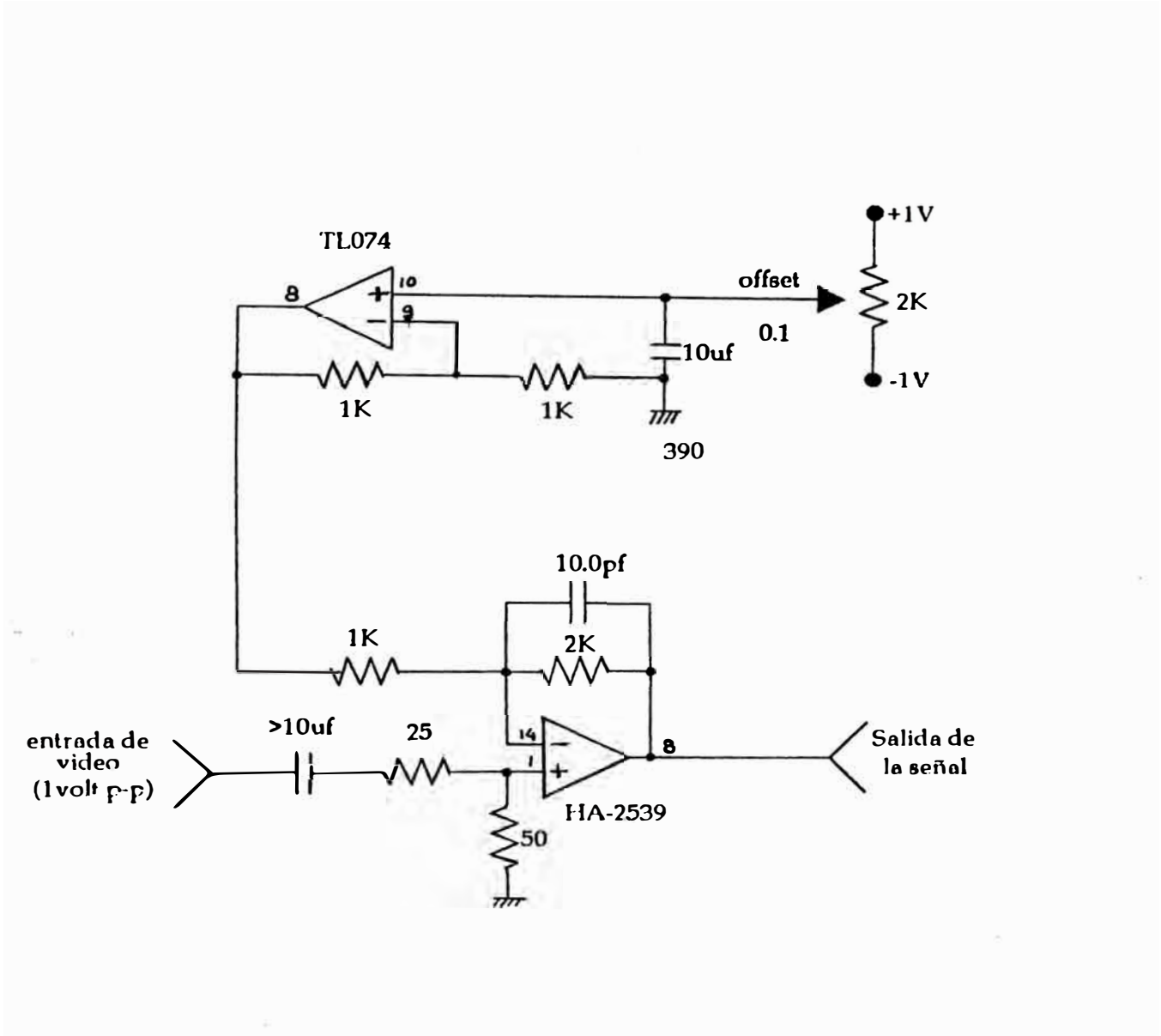


Fig. N° 3.10 Diagrama circuital del filtro Pasa Bajo de video.

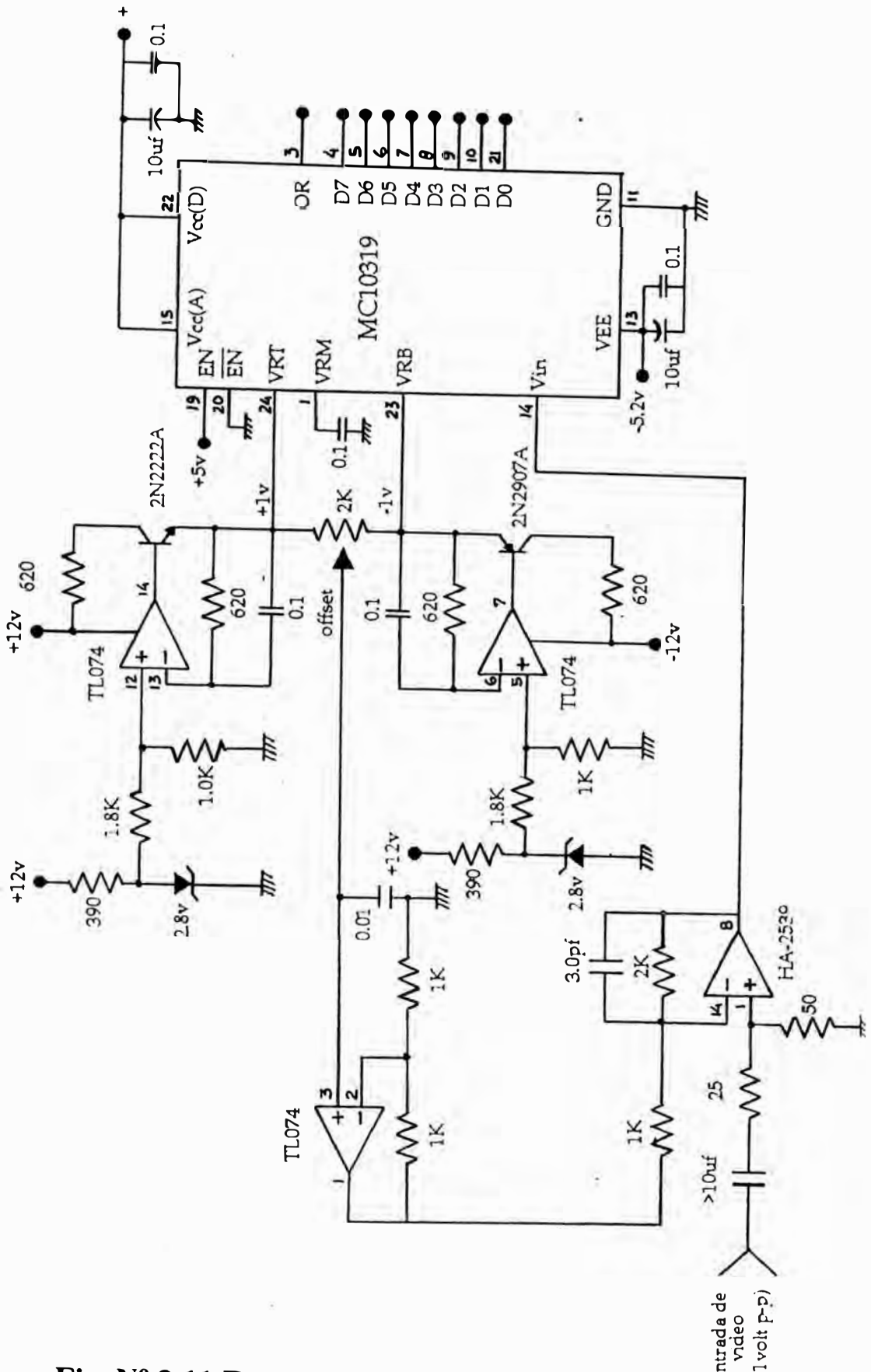


Fig. Nº 3.11 Esquema circuital del conversor Análogo/Digital.

3.3 Diseño de la etapa de conversión digital/analógica.

En la última etapa de nuestro sistema tenemos a un conversor DAC el cual es el encargado de efectuar la conversión de la imagen digital proveniente del banco de memoria de la etapa digitalizadora en su equivalente señal de video analógica para su visualización en el monitor de televisión. Se puede leer un archivo binario del disco duro del computador que tenga señal de video digitalizado (un cuadro) y almacenarlo en el banco de memoria para su conversión en señal analógica. La señal de video digital proveniente del banco de memoria es leído a la frecuencia del oscilador (14.318 Mhz) y esta misma señal del oscilador es aplicada a la patilla del reloj del conversor DAC para obtener de esta forma la señal analógica de video.

Esta etapa es implementado con el circuito integrado DAC TDC1016, de fabricación TRW. Este conversor puede efectuar conversiones digitales hasta un máximo de 20 millones de muestras por segundo. Este dispositivo incluye una entrada a un registro de datos. La alimentación es de –5.2 VDC, adaptando su entrada para niveles digitales de tipo ECL; de otra forma puede operar con voltajes +/- 5VDC, de este modo se adapta para soportar entradas digitales de tipo TTL.

Todas las versiones del conversor TDC1016 son de 10 bits, pero se puede disponer para manejar desde 8 y 9 bits. En la Figura 3.12 se muestra el diagrama de bloques interno de este conversor.

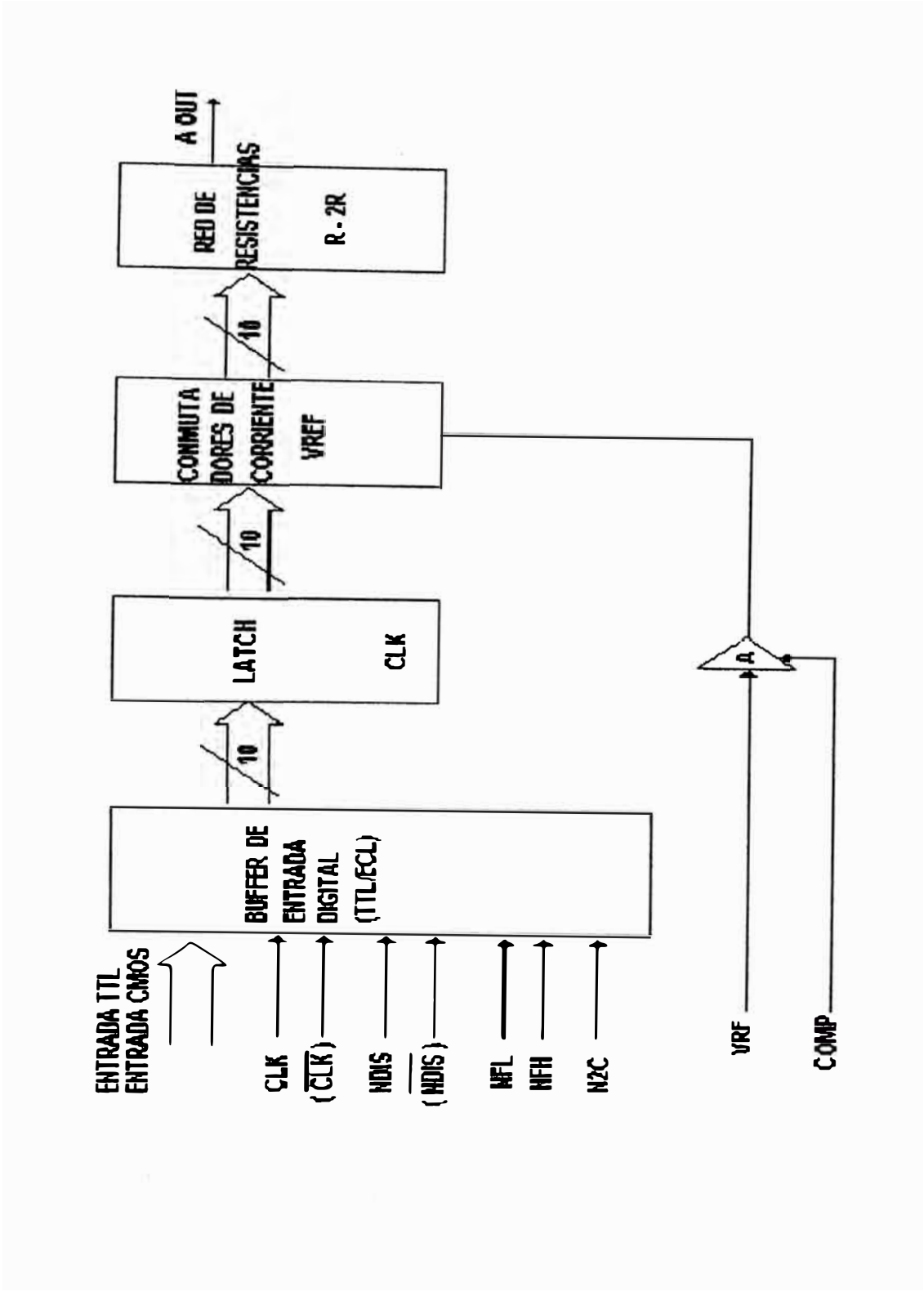


Fig. Nº 3.12 Diagrama de bloques del conversor DAC.

A la salida de este conversor DAC se encuentra un amplificador inversor de la señal de video, para lograr obtener así la señal de video de 1 Vp-p . Este amplificador esta diseñado con un dispositivo activo (amplificador operacional) HA2539, diseñado especialmente para operar con señales de video. En la Fig. 3.13 se muestra el diagrama esquemático de esta etapa.

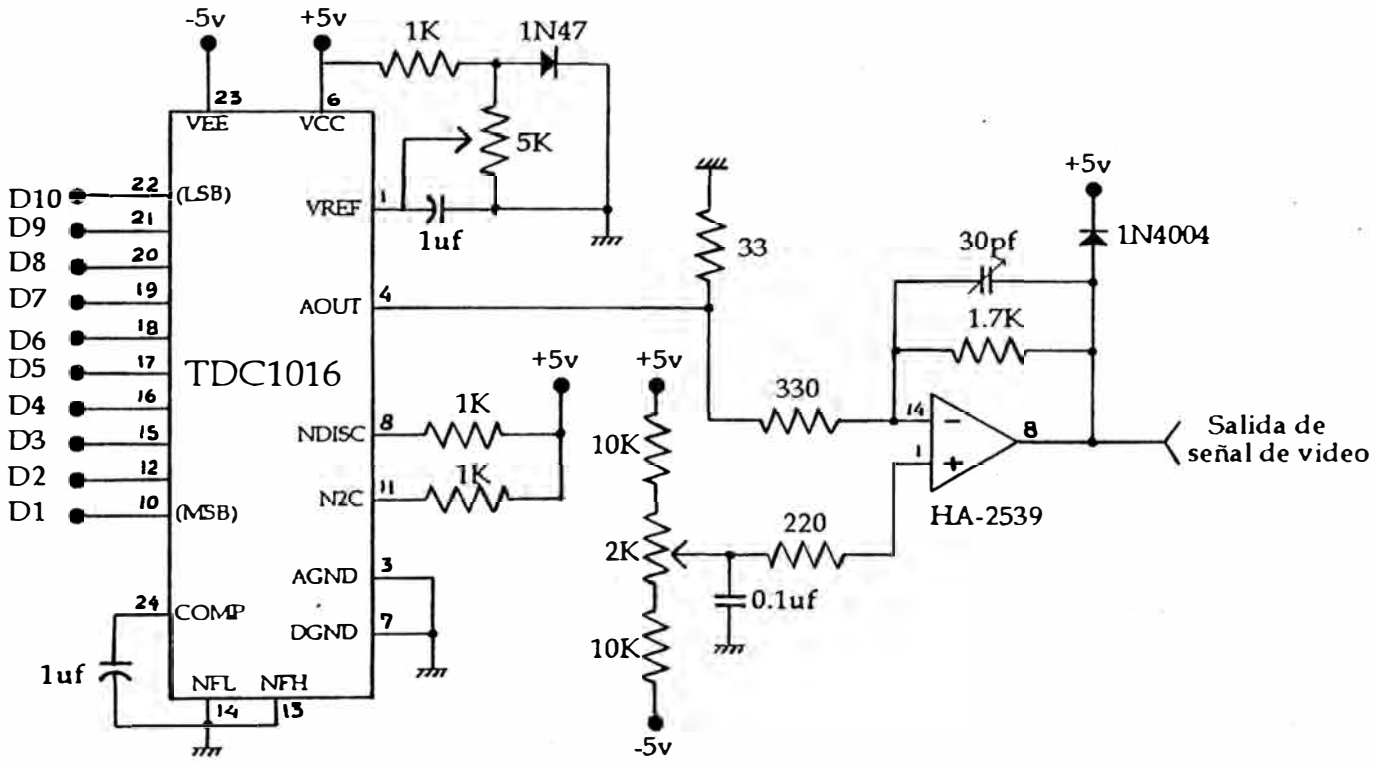


Fig. No 3.13 Esquema circuital de la etapa de conversion DAC.

CAPITULO IV
PROCESAMIENTO DE SEÑALES DE VIDEO (IMÁGENES)

4.1 Codificación de archivos

Un archivo del gráfico no es mas que un archivo que contiene una imagen, cualquier programa que muestra la imagen almacenada necesita disponer la opción de leer un archivo de gráfico y cualquier programa que muestre una imagen necesita disponer de imágenes almacenadas para un uso posterior.

Aunque solo existen unos cuantos diferentes y fundamentales caminos para codificar una imagen, hay muchos archivos incompatibles , cada uno de ellos con sus peculiaridades (BMP, PCX, GIF, JPG, WPG, etc).

Comenzaremos a estudiar su estructura general para poder manejar formatos de archivos y así pasar a observar en detalle una gran variedad de formatos específicos, para que de esta manera se pueda crear archivos de gráficos con las imágenes capturadas por la tarjeta digitalizadora de señales de video. En este aspecto es preciso aclarar, que la imagen capturada (un cuadro de video) contiene información de los niveles de brillo de la imagen, y además esta contenida en ella, los pulsos de sincronismo tanto vertical como los pulsos de sincronismo horizontal.

La información de los datos en un archivo de gráfico pueden ser codificados de diferentes maneras; en algunos casos, esta información puede ser almacenada utilizando el código ASCII, el cual es muy practico porque se relaciona al código utilizado en las computadoras personales, así como puede ser leída fácilmente por las personas, también puede ser editado fácilmente, asimismo leído y/o escrito por los programas. Pero se tiene la desventaja de que es muy voluminoso y lento para ser leído por los programas . En otros casos un archivo de gráficos, puede ser comprimido en formato binario, el cual es muy compacto pero incomprensible por los programas y por lo tanto requiere de rutinas de lectura y escritura complejas.

Para estos casos también se tienen los formatos que varían grandemente en la forma en que ofrecen flexibilidad en términos de tamaño de imagen, forma, colores y otras características de la imagen. Entre los formatos de archivos de imágenes, el mas común es el formato de Mapa de Bits (Bitmaps), el cual representa un cuadro como una matriz rectangular de puntos, como en una imagen de televisión, esto hace que sea el método ideal para relacionarlo en la forma en que la imagen de televisión es digitalizada . Este formato también es manejado por la mayoría de los graficadores tanto para el Sistema Operativo DOS así como el Sistema Operativo Windows. También se tiene el formato de Vectores, el cual representa un cuadro como una serie de líneas y arcos, en el cual la combinación de funciones permite representar los caracteres.

Con cada categoría de formato, hay un conjunto de técnicas aplicables a todos ellos. En el capitulo presente desarrollaremos la teoría que se requiere para generar archivos de gráficos utilizando el formato

de mapa de Bits, para que de este modo, el proceso de conversión utiliza el formato Mapa de bits, y de esta manera realizar la conversión de la imagen digitalizada a un archivo de gráfico en el sistema.

4.2 Personalización de la imagen.

Todas las computadoras tienen un principio de tratamiento de la geometría interna usada para posicionar elementos de un cuadro, afortunadamente el número de geometrías diferentes en la computadoras es pequeño. En el cual todos usan un modelo de papel gráfico, es decir con posiciones en el cuadro especificado por las coordenadas X e Y relativas al origen. La coordenada X es la distancia horizontal desde el origen y la coordenada Y es la distancia vertical, también relativa al origen.

coordenadas de pantalla

Estas coordenadas de pantalla como se muestra en la Fig.No. 4.1 es la mas utilizada comúnmente, en propuestas para visualizarse en la pantalla de video. En una notación convencional las coordenadas son expresadas en paréntesis como sigue (x,y). En donde el punto (0,0) esta ubicado en la esquina superior izquierda de la imagen. Asimismo la coordenada X se incrementa hacia la derecha del origen y la coordenada Y hacia abajo.

Usualmente una unidad de la distancia X es la misma que una unidad de distancia Y, en algunos casos las dos escalas son diferentes, esto ocurre muy a menudo debido a que la imagen fue salvada desde una pantalla que distorsiona la imagen , tal como ocurre en una pantalla IBM CGA o VGA.

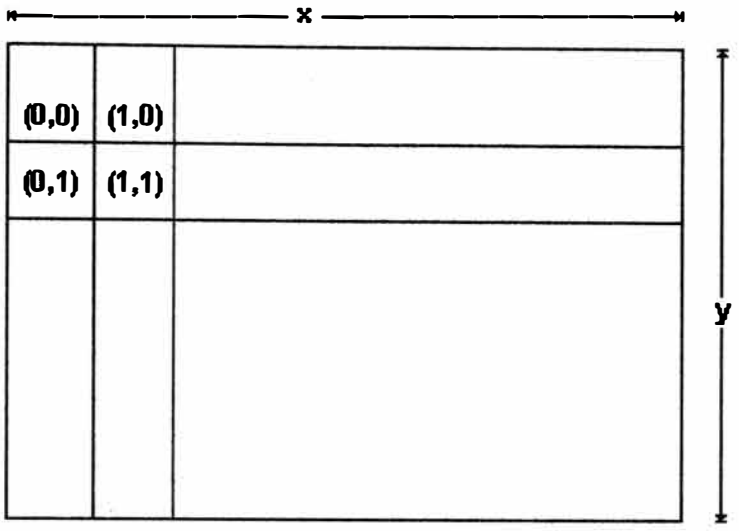


Fig. Nº 4.1 Esquema de coordenadas gráficas.

4.3 Archivo gráfico del mapa de bits

Todos los archivos de gráficos están dentro de la categoría de un pequeño conjunto de categorías, el mas común de ellos es el archivo de gráfico de mapa de bits. Como podemos apreciar los archivos BIPMAP (Mapa de Bits) son los comúnmente encontrados. Asimismo una imagen BIPMAP es una matriz rectangular de puntos, como elementos de cuadros o pixels. Cada punto de la imagen es almacenado como un numero el cual representa el color en ese punto y el valor del pixel para este punto. En los cuadros mas simples, cada punto es negro o blanco, así el valor del pixel es ¨0¨ ó ¨1¨, en un solo bit. En el caso mas general el cuadro, está en niveles de gris o de color, en cuyo caso ha de tener un rango mayor de valores de pixel. Para una imagen de niveles de gris, cada valor de pixel puede ser de 8 bits, así los valores estarán en el rango de 0 para el negro a 255 para el blanco. En la actualidad existen

diferentes vías para asignar valores de pixel a colores entre los que tenemos:

Ordenamiento de pixel

A partir del principio de que la matriz de pixel son de 2 dimensiones, cada formato de archivo necesita una conversión del orden en la cual los pixel son almacenados. Desafortunadamente hay muchas formas en la cual los pixels pueden ser almacenados:

Exploración de líneas.

Es el orden más sencillo en que se almacena los pixels, el cual se realiza en una unidad de tiempo. Cada línea es con frecuencia referido como una línea explorada, análoga a las líneas de televisión o pantallas de TRC (Tubo de rayos catódicos).

Generalmente , los pixels son almacenados, en una fila de izquierda a derecha, con las filas de arriba abajo. Pero no siempre, algunas veces los pixels van de derecha a izquierda y en algunos caos las filas van de abajo hacia arriba.

En el caso de que cada pixel es mas pequeño que un byte, surge la pregunta de cual de los pixel que están mas a la izquierda en un byte es el orden mas alto o los bits de orden mas bajos. Esta decisión es muy frecuente realizado con el formato de datos usados por el adaptador de video, desde que una imagen puede ser copiada al adaptador de video, mucho mas rápidamente no existe necesidad de reordenar los bits en un byte.

Bits planos

Cuando los pixels es una imagen contienen mas de 1 bit por cada pixel, algunos formatos almacenan el dato como bit plano. Es decir primero ellos almacenan el bit de orden mas alto de todos los pixels, el

primero es el bit plano. Luego ellos almacenan el segundo bit de los pixels, el segundo bit plano y así sucesivamente, con un bit plano por cada bit de los pixels. Una imagen con 6 bits por pixel estaría almacenando como 6 bits planos separados. Algunas veces una imagen es almacenada por plano por cada fila, antes que por plano para una imagen.

Filas entrelazadas

Generalmente las filas de pixel de una imagen son almacenadas en orden desde arriba hacia abajo. En algunos casos, sin embargo, las filas son entrelazadas. El mas simple método de entrelazado es almacenar las filas de números pares, luego las filas de números impares, así las filas pueden ser almacenados en el orden 0, 2, 4, 6 , ... 1, 3, 5, ... En algunos casos las filas de números impares están en el oren inverso, así si fuera un total de 100 filas, el oren seria 0,2,4, ... 96, 98, 99, 97, 95,... 5, 3, 1.

El punto inicial original del entrelazado fue tomado de algún modo de los monitores de video. Otra ventaja de archivos entrelazados es que uno puede construir una versión aproximada de la imagen sin tener que leer el archivo completo, (porque se tiene que el campo par se encuentra abajo y el impar arriba, que es el orden en la exploración de líneas usadas en TV), algo que puede ser usado particularmente cuando un archivo esta siendo mostrado como es recibido sobre una red lenta o línea telefónica.

Estructura de un archivo BITMAP

La razón de utilizar los archivos del mapa de bits (BITMAP) es sencillamente para almacenar una imagen en un formato que sea independiente de los esquemas de especificación de color usados en cualquier dispositivo de hardware. Los usuarios de este tipo de archivo

de gráfico se encuentran dentro de Microsoft Windows 95 (o versión superior). El tipo de dato que se utiliza es en 2 dimensiones, la representación puede ser en monocromático, niveles de gris o a color (RGB Rojo, Verde Azul). La organización de los datos es en forma secuencial y la codificación de los datos es en forma binaria. Por lo general no es comprimido pero puede ser comprimido por el método de longitud fija. El tamaño de la imagen esta dado por el numero de pixels por línea.

En la Fig No. 4.2 se muestra la forma en que son organizados los datos en el archivo BITMAP. También se explica la forma que debe de tener la inicialización del archivo BITMAP así como el dignificado de cada uno de los Bytes que conforman. Del mismo modo se explica los bytes de información del archivo de gráfico. Todo esto se dedujo analizando como es que la herramienta de software llamada PaintBrush del Windows 3.11 creaba los archivos de mapa de bits.

4.4 Software de conversión de archivo de imagen digital

El software de conversión de la imagen digitalizada para obtener un archivo de gráfico BITMAP, debe iniciar bajo la consideración de generar un archivo de gráfico bitmap, capturando una imagen de video con el sistema digitalizador, como se sabe este archivo es de tipo binario (según el software desarrollado anteriormente, se asigno a estos tipos de archivos la extensión *.DAT); es decir, son las muestras de los niveles de gris de la imagen.

Luego se debe crear los archivos que contengan la información de la cabecera del archivo BITMAP a crear, para ser utilizados en el momento en que se genera un archivo de gráficos. A estos archivos lo

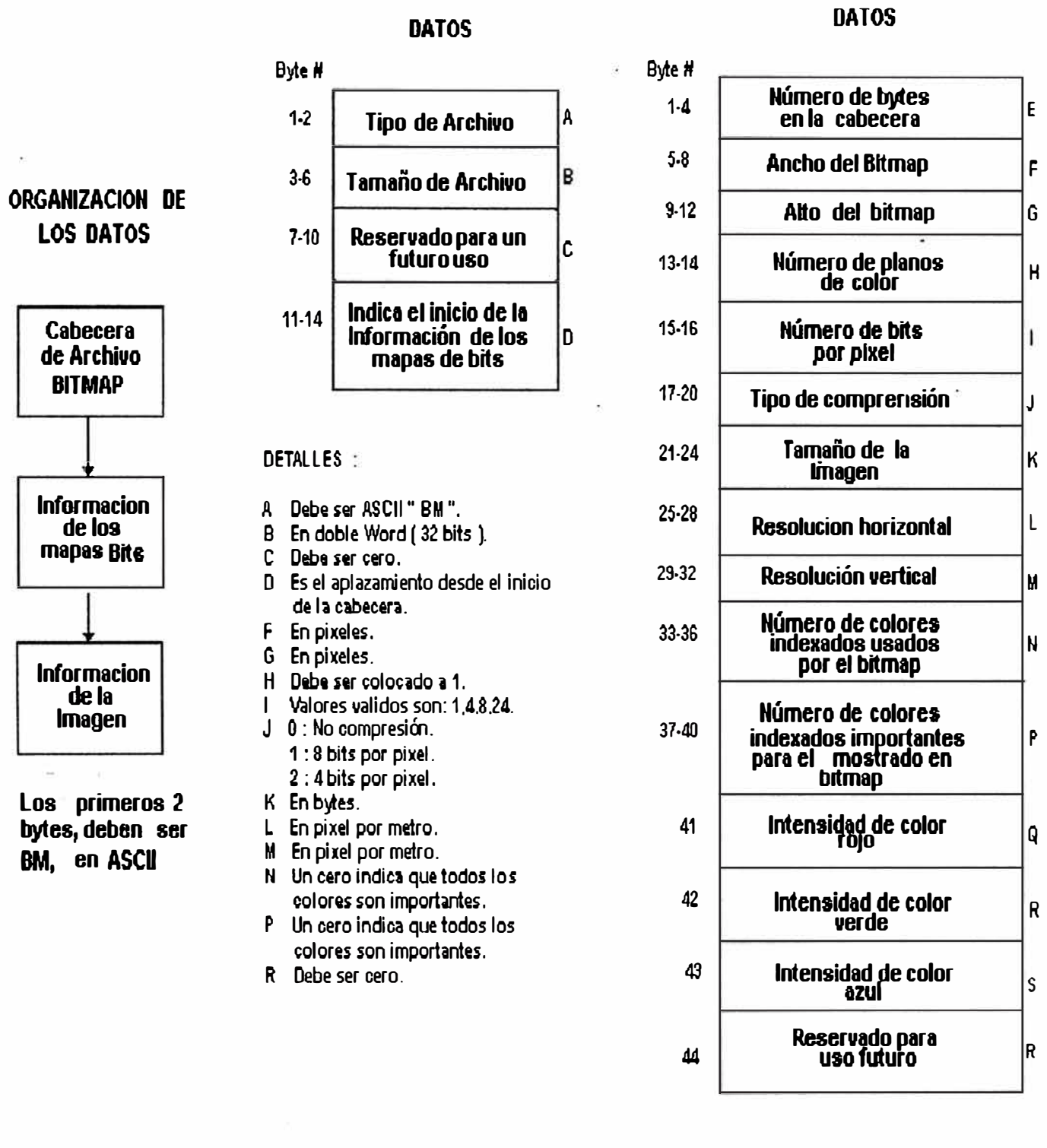

ORGANIZACION DE LOS DATOS
Cabecera de Archivo BITMAP
Informacion de los mapas Bite
Informacion de la Imagen
Los primeros 2 bytes, deben ser BM, en ASCII
DATOS
Byte #
1-2
Tipo de Archivo
A
3-6
Tamaño de Archivo
B
7-10
Reservado para un futuro uso
C
11-14
Indica el inicio de la Información de los mapas de bits
D
DETALLES :
A Debe ser ASCII " BM ".
B En doble Word (32 bits).
C Debe ser cero.
D Es el aplazamiento desde el inicio de la cabecera.
F En pixeles.
G En pixeles.
H Debe ser colocado a 1.
I Valores validos son: 1,4,8,24.
J 0 : No compresión.
1 : 8 bits por pixel.
2 : 4 bits por pixel.
K En bytes.
L En pixel por metro.
M En pixel por metro.
N Un cero indica que todos los colores son importantes.
P Un cero indica que todos los colores son importantes.
R Debe ser cero.
DATOS
Byte #
1-4
Número de bytes en la cabecera
E
5-8
Ancho del Bitmap
F
9-12
Alto del bitmap
G
13-14
Número de planos de color
H
15-16
Número de bits por pixel
I
17-20
Tipo de comprensión
J
21-24
Tamaño de la Imagen
K
25-28
Resolucion horizontal
L
29-32
Resolución vertical
M
33-36
Número de colores indexados usados por el bitmap
N
37-40
Número de colores indexados importantes para el mostrado en bitmap
P
41
Intensidad de color rojo
Q
42
Intensidad de color verde
R
43
Intensidad de color azul
S
44
Reservado para uso futuro
R

Fig. N° 4.2

denominaremos MASCARAS porque contendrán las informaciones siguientes:

a) Tamaño del archivo BITMAP a crear (Por ejemplo 640 x 480)
b) Intensidad de brillo del pixel, así como los valores de los colores primarios que se cargaran en la paleta gráfica del computador : (RGB Rojo Verde Azul).
c) Información adicional, que se detalla en la Fig. No. 4.2

En el cuadro No. 4.1 se muestra la forma que tiene el archivo llamado Mascara1 y Mascara2, los cuales han sido desarrollados para generar un archivo de gráfico de tipo BITMAP. A cualquiera de estos dos archivos se le añadirá la información correspondiente a las muestras de un archivo de tipo binario (*.DAT). Hay que tener en cuenta que no todas las muestras son consideradas en el archivo de gráfico; es decir, si tomamos que un pixel del archivo BITMAP es una muestra del cuadro de video digitalizado, entonces solamente podremos almacenar la cantidad de 640 x 480= 307200 pixel o muestras. El restante de muestras se son eliminados, una variación a este problema, es generar archivos de gráfico BITMAP cuyo numero pixel por línea sea el numero de muestras por línea; es decir una línea de exploración de TV tarda 63.5 seg si muestreamos a 14.318 MHz y con lo cual tenemos un total de 909 muestras por línea. En conclusión nuestros nuevos archivos de gráfico BITMAP a generar seria de 909 pixel por 525 líneas.

En el siguiente listado se muestra el programa llamado BINBMP1.C que convierte un archivo binario (imagen digital) a un archivo de gráfico de tipo BITMAP. Este programa produce archivos de gráfico de la misma forma como la imagen de video es generada por una cámara e TV; es decir, el ordenamiento de los pixels es en forma de exploración

```
42 4D 76 58 02 00 00 00 00 00 76 00 00 00 28 00
00 00 80 02 00 00 E0 01 00 00 01 00 04 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 23 23 23 00 23 23 23 00 23 23
23 00 23 23 23 00 23 23 23 00 23 23 23 00 41 41
41 00 72 72 72 00 A2 A2 A2 00 D3 D3 D3 00 FF FF
FF 00 FF FF FF 00 FF FF FF 00 FF FF FF 00 FF FF
FF 00 FF FF FF 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
42 4D 76 58 02 00 00 00 00 00 76 00 00 00 28 00
00 00 80 02 00 00 E0 01 00 00 01 00 04 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 02 02 02 00 02 02 02 00 00 02 02
02 00 02 02 02 00 02 02 02 00 02 02 02 00 22 22
22 00 52 52 52 00 81 81 81 00 B1 B1 B1 00 E0 E0
E0 00 FF FF FF 00 FF FF FF 00 FF FF FF 00 FF FF
FF 00 FF FF FF 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Cuadro No. 4.1

Máscaras que contienen la información correspondiente a la cabecera de los archivos BITMAP (640 x 480). La mascara hace que el archivo de gráfico presente a la imagen con mayor intensidad luminosa, en comparación a la máscara 2.

de líneas de izquierda a derecha y e arriba abajo. En la Fig No. 4.3 se muestra un archivo BITMAP generado por este programa.

En la Fig 4.4 se muestra el diagrama de flujo del programa BINBMP1.C.

Listado del programa : BINBMP1.C

```
#include <stdio.h>
int main(void)
{
        /*Definimos las variables */
        int                     i, columna;
        unsigned int            avanza;
        unsigned long l,m;
        char                    nombre1[13],nombre2[13],nombre3[13];
        unsigned char dato,dato2;
        FILE                    dat,bmp,*mascara;
        /*inicio del programa  */
        printf ("\nConvertidor de archivo de video binario (*.DAT) a formato
BITMAP");
        printf ("\nCopyright (c ) Junio – 1998 by AGL");
        printf ("\nIngrese el nombre del Archivo (*.DAT) => ");
        scanf ("%",&nombre1);
        printf ("Ingrese el numero de muestras a adelantar =>");
        scanf ("%d", &avanza);
        printf ("Espere un momento...");
        dat=fopen(nombre 1, "rb");

        /* numero de muestras a avanzar del archivo binario = 48145 */
        avanza=avanza+48145;

        for(I=0;I<=avanza;I++)
                        fscan(dat,"%c",&dato);

        printf("\ningrese el nombre del archivo (*.BMP) a crear =>");
        scanf(%s",&nombre2);
        bmp=fopen(nombre2,"wb");

        printf("Ingrese la plantilla de niveles de gris:");
        scanf("%s",&nombre3);
        mascara=fopen(nombre3,"rb");
        printf(Espere un momento...");
        for(i=0;i<=0x75;i++){
                        fscan(mascara,"mascara,"%c",&dato);
```

```
                fprintf(bmp,"%c",dato);}
fclose(mascara);
columna=0;I=0;
while(I!=307200){
                fscanf(dat,"%c",&dato);
                dato=dato/16;
                columna ++; I++;
                fscan(dat,"%c",&dato2);
                dato2=dato2/16;
                dato<<=4;
                dato=dato/dato2;
                fprintf(bmp,"%c",dato);
                I++;
                If (columna=320){
                        for(m=0;m<=269;m++)
                        fscanf(dat."%c",&dato
                        columna=0;
                }
}
fclose(bmp);
fclose(dat);
return 0;
}               /* fin del programa   */
```

En el siguiente programa se muestra otro programa llamado BINBMP2.c realiza la transformación de un archivo binario (imagen digital) a un archivo de gráfico de tipo BITMAP. Este programa produce archivos de gráfico de la imagen de video, pero en forma entrelazada; es decir, el ordenamiento de los pixels es realizado en forma de filas entrelazadas. En la Fig. No. 4.5 se muestra el archivo BITMAP generado por este programa.

En la Fig. No. 4.6 se muestra el diagrama de flujo del programa BINBMP2.c.

Listado del programa : BINBMP2.c

Fig. Nº 4.3 Archivo de grafico BITMAP

Generado por programa BINBMP1.C

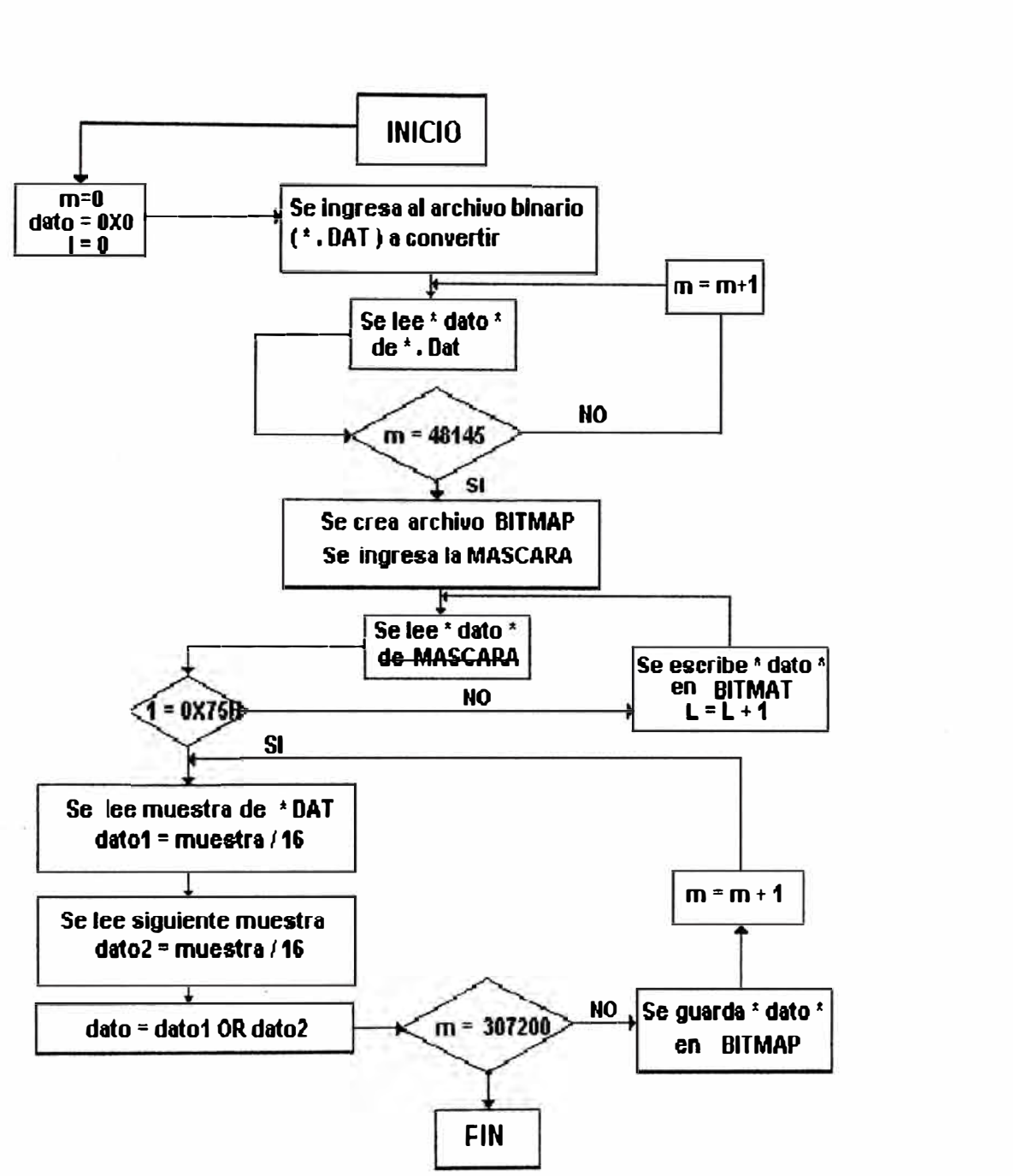


Fig. N°4.4 Diagrama de flujo del programa BINBMP1.C

```
#include <graphics.h>
#include <stlib.h>
#include <stdio.h>
#include <conio.h>

int main(void)
{
        /*Definicion de variables  */
        struct palettetype pal;
        int i,columna,par;
        unsigned int avanza;
        unsigned long I,m,cuenta;
        char nombre1[13],nombre2[13],nombre3[13];
        unsigned char dato,dato2;
        FILE   *dat1,*dat2,*bmp,*mascara;
        /*Inicio del programa  */
        printf("\nConvertidor de archivo de video binario (*.DAT) a formato
BITMAP");
        prinft("\nCopyright (c ) Junio –1996 by AGL");
        printf("\nIngrese el nombre del archivo (*.DAT) => ");
        scanf("%s",&nombre1);
        printf("Ingrese el numero de muestras a adelantar => ");
        scanf("%d",&avanza);
        printf("Espere un momento ... ");
        dat1=fopen(nombre1, "rb");
        dat2=fopen(nombre1, "rb");
        avanza=avanza+48145;
        for(I=0;I<=avanza;I++)
                        fscanf(dat1, "%c",&dato);
        for(I=0;I<=avanza;I++)
                        fscanf(dat2, "%c"&dato);

        printf("\nIngrese el nombre del archivo (*.BMP) a crear => ");
        scanf("%s",&nombre2);
        bmp=fopen(nombre2, "wb");

        print("Ingrese la plantilla de niveles de gris:");
        scanf("%s",&nombre3);
        mascara=fopen(nombre3, "rb");
        printf("Espere un momento... ");
        for(i=0;i<=0x75;i++){
                                fscanf(mascara, "%c",&dato);
                                fprintf(bmp, "%c",dato);}
        fclose(mascara);
        cuenta=0;
        do{
                        fscanf(dat2, "%c",&dato);
```

```
                    cuenta++;
}
while(cuenta!=238415);
        column=0;
        I=0;
        par=1;
        while(I!=307200){
                if(par=1){
                    fscanf(dat1,"%c,&dato);          /*leo  la  primera
muestra.dat1 */
                    dato=(dato-50)/16;
                    fscan(dat1,"%c",&dato2);   /*leo la segunda muestra.dat1 */
                    dato2=dato2/16;
                    dato<=4;
                    dato=datoIdato2;
                    fprintf(bmp,"%c",dato);          /*  escribo  el  primer
dato.bmp */
                    columna++;
                    I=2;
                        If(columna=320){
                            For(m=0;m<=269;m++)
                                Fscanf(dat1,"%c",&dato);
                            Columna=0;
                            Par=0;}
                    }
        else{
                    fscanf(dat2,"%c",&dato);   /*leo la primera muestra.dat2 */
                    dato=(dato-50)/16;
                    fscanf(dat2,"%c",&daato2); /*leo la segunda muestra.dat2 */
                    dato2=dato2/16;
                    dato<=4;
                    dato=datoIdato2;
                    fprintf(bmp,"%c",dato);          */escribo  el  primer
dato.bmp */
                    columna++;
                    I=2;
                        If(columna=320){
                            For(m=0;m<=269;m++)
                                Fscanf(dat2,"%c",&dato);
                            Columna=0;
                            Par=1}
                        }
                            //fin del while
                }
                fclose(bmp);
                fclose(dat1);
                fclose(dat2);
                return 0;
```

```
                    Fscanf(dat2,"%c",&dato);
                Columna=0;
                Par=1}
            }
}                   //fin del while
fclose(bmp);
fclose(dat1);
fclose(dat2);
return 0;
}           /* fin del programa  */
```

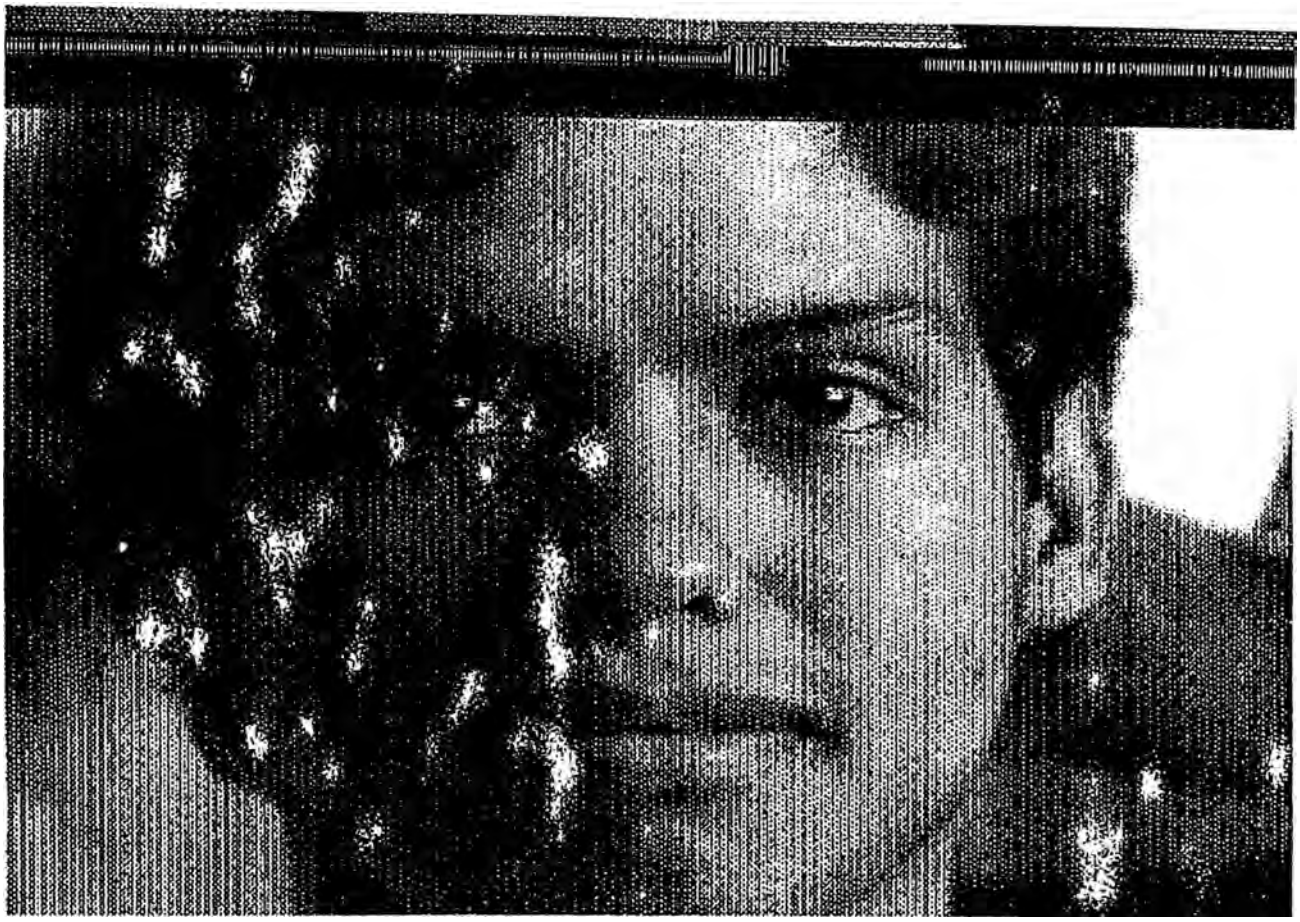


Fig. Nº 4.5 Archivo de gráfico BITMAP generado por Programa BINBMP2.C.

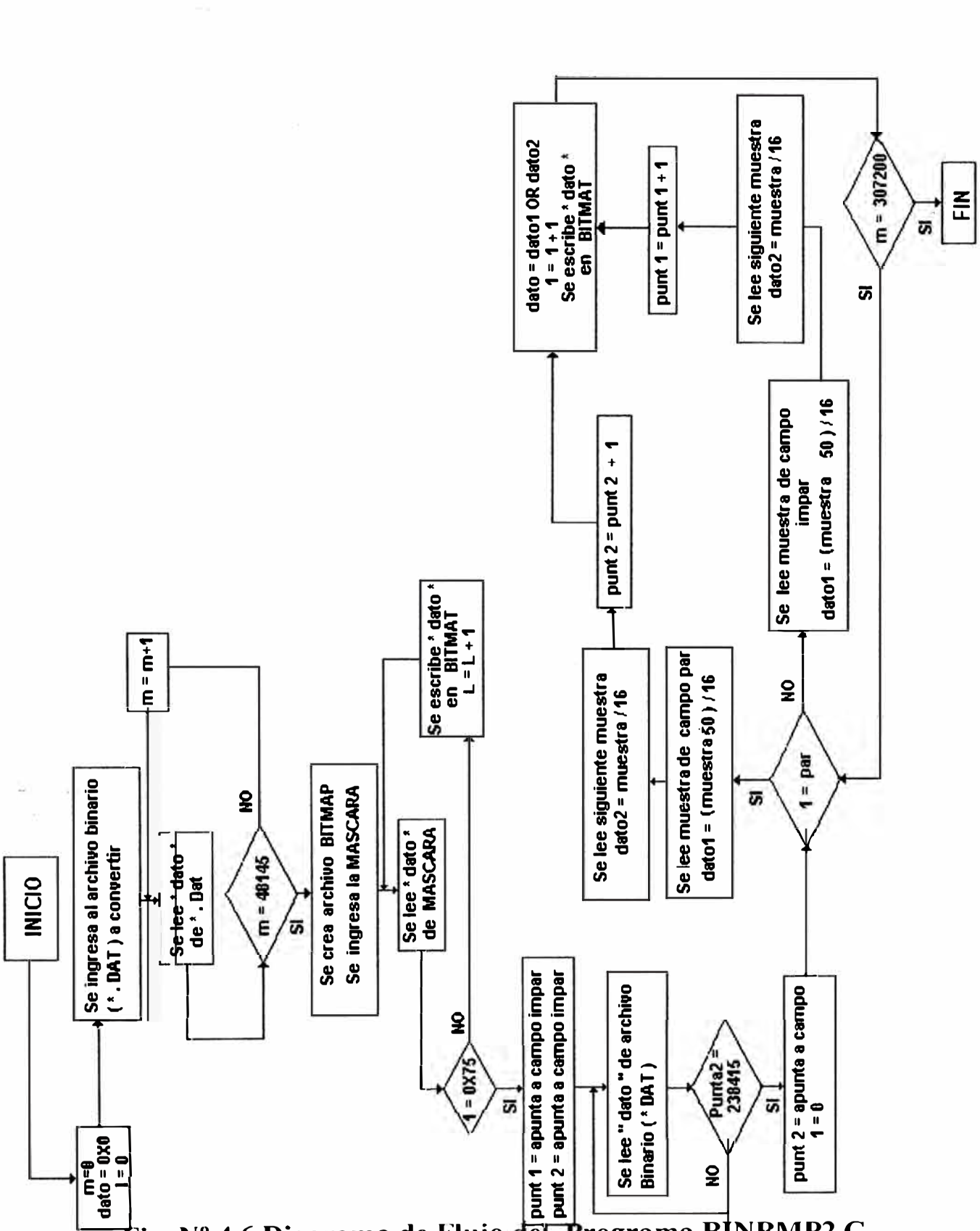


Fig. Nº 4.6 Diagrama de Flujo del Programa BINBMP2.C.

CAPITULO V
DETECCION DE LAS CARACTERISTICAS FACIALES PERSONALES

5.1 Sistema de procesamiento de datos adquiridos por digitalización

Un aspecto muy importante, en el proceso de reconocimiento de las personas, es determinar las características faciales más resaltantes del rostro. Esto se puede obtener acentuando las zonas donde se detecte cambios de niveles de gris, ésto es en partes más resaltantes como son los ojos, la nariz, la boca y el contorno del rostro. En el momento de la adquisición del rostro de la persona, se debe tener en cuenta muchas consideraciones, entre ellas fundamentalmente la posición relativa de la persona ya que éste determinará la geometría del rostro, y el contorno de la persona. La iluminación es un factor muy importante, ya que sin ella el realzamiento de los bordes como técnica de procesamiento digital no seria adecuado para obtener las características faciales. En el presente capítulo se desarrollan algoritmos empleados en el procesamiento digital de imágenes para el realzamiento de los bordes del rostro.

Después de realizado la adquisición de un cuadro de video y también el procesamiento de la conversión de la imagen a un archivo de gráfico en forma de mapa de bits, estamos en condiciones de realizar cualquier tipo de procesamiento. El procesamiento de los pixels provee combinaciones, correcciones y alteraciones en los niveles de gris de la

imagen, todos estos procesos son fundamentales como herramientas del procesamiento digital de imágenes. Como quiera que las operaciones por pixels no provee la habilidad para alterar los detalles en la escena de forma espacial dentro de una imagen, debido a que este tipo de procesamiento, actúa sobre pixel por pixel; por lo que éste proceso no considera el procesamiento entre pixels colindantes.

El procesamiento en grupos de pixels opera sobre un grupo de pixels rodeando a un pixel central. Los pixels colindantes proveen información valiosa acerca de la tendencia del brillo en el área ha ser procesada. La imagen está compuesta por componentes básicos de frecuencias , que van desde las más bajas frecuencias hasta las más altas frecuencias. Cuando las transiciones rápidas de brillo prevalecen, hay altas frecuencias espaciales. Las frecuencias más altas en una imagen son encontradas en cualquier contorno agudo ó puntos que presentan una transición de blanco a negro en 1 o 2 pixels de distancia.

Una imagen puede ser filtrada para acentuar ó remover una banda de frecuencias espaciales, tales como las altas frecuencias ó bajas frecuencias. Otras operaciones de filtros espaciales hacen posible sobreiluminar solo las transiciones agudas en la imagen, tales como los bordes de los objetos. Estas operaciones son un subconjunto de filtros espaciales como son las operaciones de realce de bordes.

Los filtros espaciales son implementados a través de procesos llamados "convolucion espacial", el cual es un método matemático usado en el procesamiento y análisis de señales. Este proceso de

convolucion espacial es también referido como un filtro de respuesta de impulso finito (FIR).

El proceso de convolucion espacial se mueve a través de la imagen, pixel por pixel, dando como resultado nuevos pixels en la imagen de salida. El brillo de cada pixel es dependiente de un grupo de pixels de entrada situados alrededor del pixel que ha ser procesada. Utilizando la información de los pixels vecinos al pixel central, la convolucion espacial calcula las frecuencias aspaciales activas en el área y es por lo tanto capáz de filtrar contenidos de frecuencias espaciales sobre áreas de imágenes.

El proceso de convolucion espacial usa un "peso promedio" de un pixel de entrada y sus vecinos inmediatos para calcular el valor de brillo del pixel de salida. El grupo de pixels usados en el calculo del "peso promedio" es llamado nucleo (kernel). Las dimensiones del núcleo son generalmente de un cuadrado con un numero impar de valores de máscara en cada dimensión. El núcleo puede tener la dimension de 1 x 1, el cual es un caso trivial de simple proceso de un punto, 3x3, 5x5 y así sucesivamente. El tamaño mas grande de núcleos de pixels usados en el calculo, aumentan los "grados de libertad " del filtro espacial. Esto significa que la flexibilidad y precisión de los filtros espaciales son incrementados cuando mas pixels vecinos son tomados dentro del calculo.

En la práctica los núcleos de 3x3 y 5x5 son usados en muchas operaciones de filtros espaciales.

En proceso de calculo de "peso promedio" es llamado un "proceso lineal" debido a que implica la adición de "elementos" multiplicado por unas "constantes". Los elementos son el brillo de los pixeles en el núcleo y las constantes son los pesos o coeficientes de convolucion. En el caso más simple donde los pesos son iguales a [1/numero de elementos del núcleo], tenemos un proceso convencional de promediacion. Si alteramos los pesos de ciertos pixels en el núcleo podríamos tener mas o menos influencia sobre todo el promedio. En efecto, ia selección de estos promedios directamente determina la acción del filtro espacial, tales como son los filtros pasa-bajos o realzamientos de bordes.

En el mecanismo de convolucion espacial es simple, en el núcleo de convolucion de 3x3, 9 coeficientes de convolucion son definidos como se indican:

A	B	C
D	E	F
G	H	I

Esta matriz de coeficientes es llamado "Mascara de convolucion". Todos los pixels en la imagen de entrada son evaluados con sus 8 vecinos, usando esta mascara para producir un valor de pixel de salida. Este proceso es ilustrado en la Figura No 5.1. En donde se puede visualizar que la mascara es colocada sobre un pixel de entrada. El pixel y sus 8 vecinos son multiplicados por sus respectivos coeficientes de

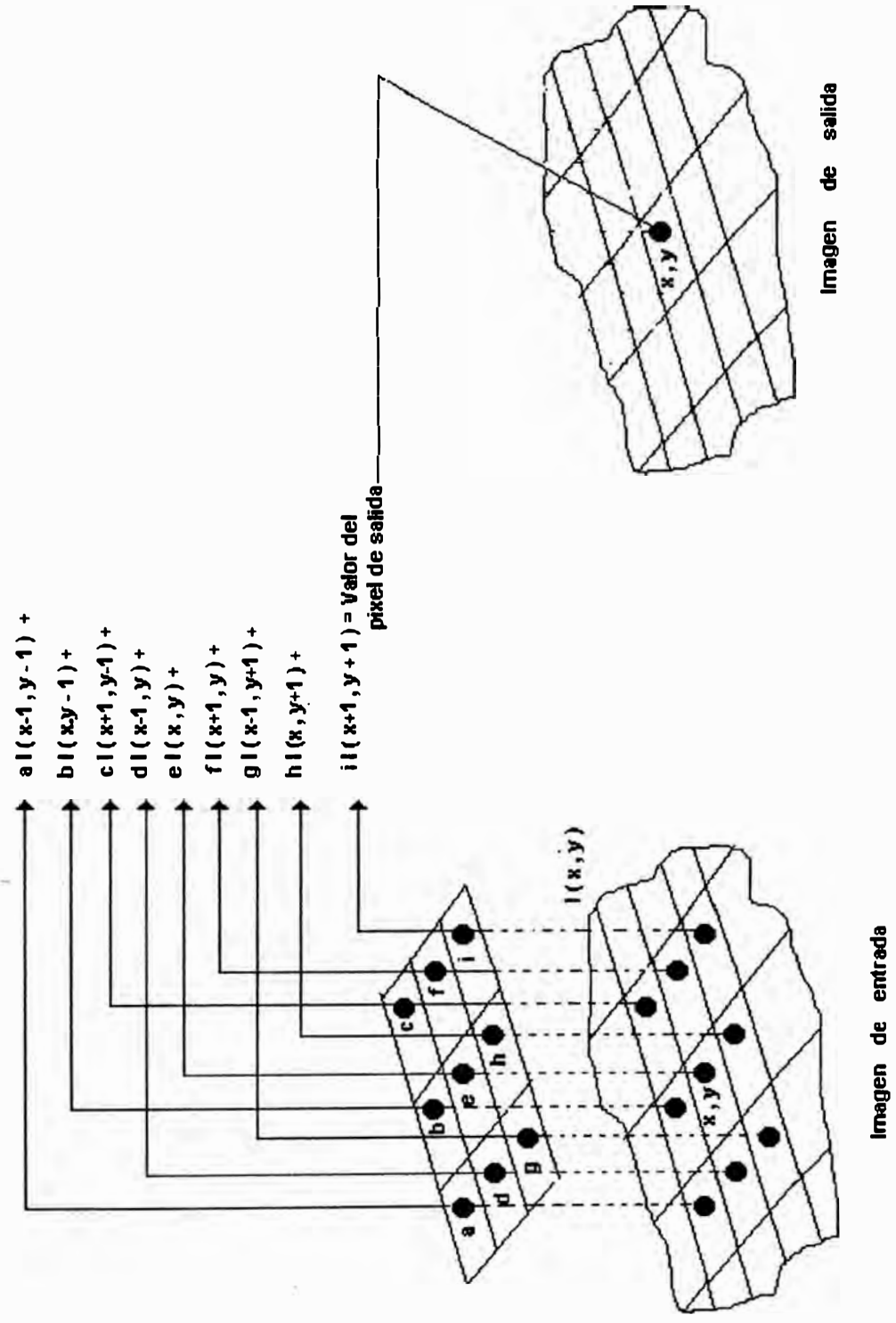


Fig. Nº 5.1 Máscara de Convolución

convolucion y las multiplicaciones son sumadas. El resultado es colocado en la imagen de salida en la misma localización central del pixel.

Este proceso ocurre pixel por pixel, para cada pixel de la imagen de entrada. La ecuación para este proceso de convolucion espacial es:

$$O(x,y) = al(x-1,y-1) + bl(x,y-1) + cl(x+1,y-1) + dl(x-1,y) + El(x,y) + fl(x+1,y) + gl(x-1,y+1) + hl(x,y+1) + Il(x+1,y+1)$$

Donde, esto implica que todos los pixels de entrada son procesados sobre esta ecuación, creando un correspondiente pixel de salida.

Los valores de la mascara de convolucion pueden generalmente tomar cualquier valor numérico. Esto es importante, como quiera que cuando el proceso de convolucion es ejecutado, el resultado final es un valor dentro del rango de 0 a 255 (para una salida de imagen de 8 bits).

En el caso de una imagen compuesta por 640 pixels x 480 lineas, la operación del filtro espacial requiere que el proceso de "peso promedio" sea pixel por pixel, esto producirá 640 x 480 = 307,200 operaciones. Cada "peso promedio" requiere de 9 multiplicaciones y 9 adiciones por pixel de la imagen de entrada. Esto significa que una operación de filtrado espacial a una imagen de 640 x 480 requiere cerca de 3 millones de multiplicaciones y adiciones para completar el proceso.

5.2 Detección de perfiles de imágenes.

El realzamiento del borde de la imagen reduce una imagen para mostrar solo sus contornos pero con mas detalle, los bordes aparecen como los contornos de objetos dentro de la imagen. Los contornos bordeados pueden ser usados en operaciones de análisis de imágenes subsecuentes o reconocimiento de objetos.

Para implementar el realzamiento de bordes se pueden hacer a través de filtros espaciales; particularmente, 3 filtros son los mas usados en la tarea de procesamiento de imágenes. Estos son los siguientes:

Desplazamiento y Diferencia.

Gradiente de Prewitt

Filtros de Bordes Laplacianos.

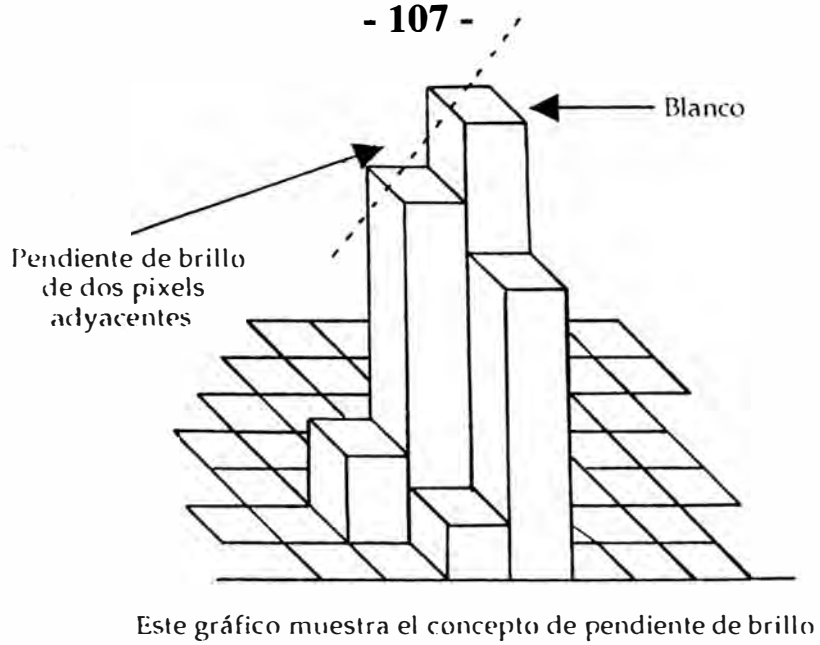
Todos ellos, están basados en los pixels de brillo con inclinación o pendiente, que sucede dentro de un grupo de pixels. Adicionalmente, otras técnicas de realzamiento de borde han sido desarrollados como: Sobel, Kirsch y los filtros Robinson. Cada uno de ellos, dependen de filtros espaciales para su implementación.

Para poder definir el termino "pendiente" en un contexto de imagen se piensa que es la intensidad de brillo de cada pixel, que es representado por la elevación que se aprecia de un plano hacia un observador; como es mostrado en la Fig 5.2. Se observa en esta figura una imagen voluminosa, en lugar de la representación estandar de niveles de gris, de intensidad de brillo de pixels y lo elevado que este aparece.

De acuerdo a la medida de la pendiente o inclinación que existe en el volumen que se encuentra en un grupo de pixel, tenemos un valor dado para poder medir el exceso de inclinación. Un valor grande corresponde a una elevada pendiente y significa un largo cambio en los niveles de gris. Un pequeño valor indica una pequeña inclinación, lo cual es equivalente a un pequeño cambio en los niveles de gris. Porque los bordes son por definición cambios bruscos de nivel de brillo, una larga inclinación indica la presencia de un borde.

La mas sencilla operación de realzamiento de borde es la de "Desplazamiento y Diferencia", este procedimiento puede realzar el borde horizontal o vertical de la información. Desplazar una imagen hacia la izquierda en un pixel y luego sustraer este valor de la imagen cambiara los bordes verticales aparentes. Esto es porque el valor de brillo del pixel de entrada es sustraido de su vecino horizontal , generando un valor de diferencia de su brillo o inclinación. Si dos pixels adyacentes tiene grandes diferencias de brillo (un borde) esto da como resultado una gran diferencia de brillo. Si dos pixels tienen similares brillos (no existen bordes) esto da como resultado, una pequeña diferencia de brillos.

El realzamiento del borde horizontal análogo es implementado por el desplazamiento de la imagen hacia arriba un pixel y realizando la sustracción.



Este gráfico muestra el concepto de pendiente de brillo

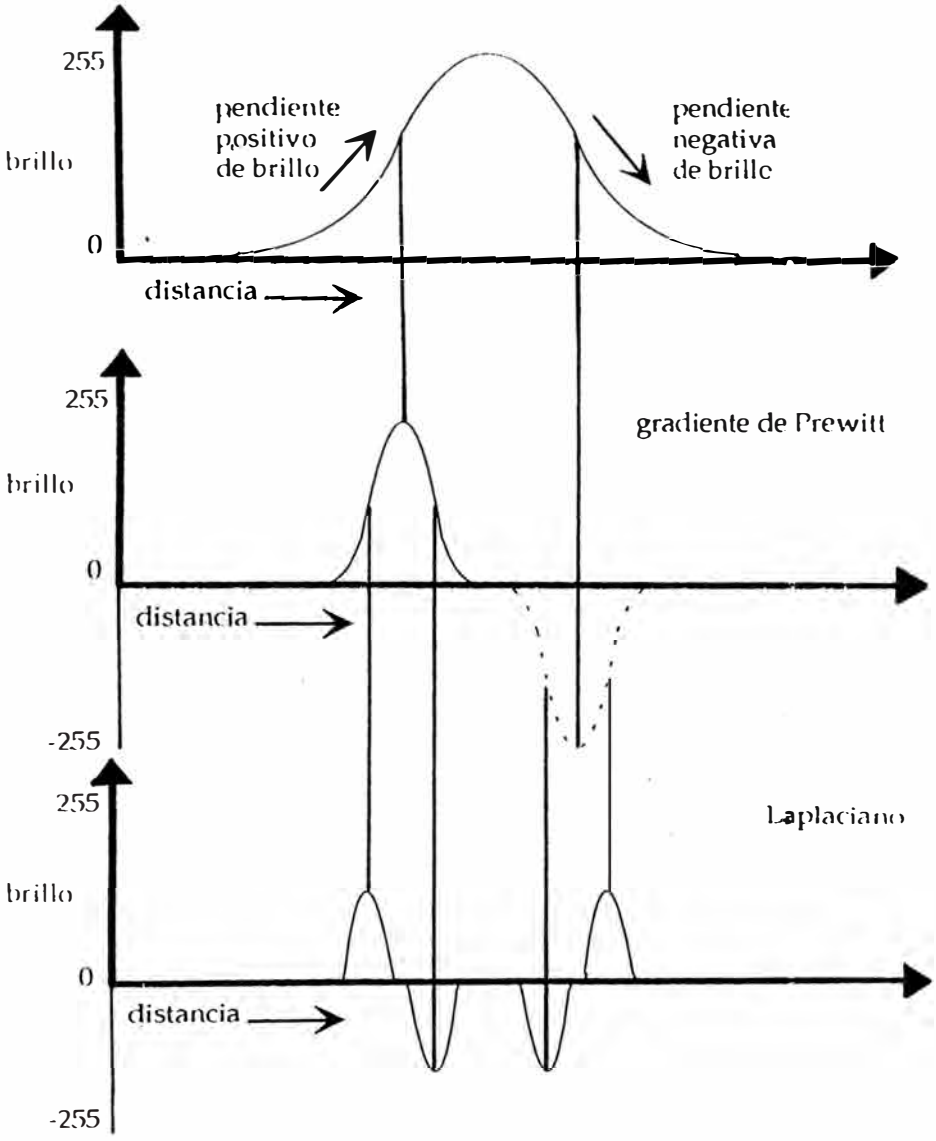


Fig. Nº 5.2 Respuestas en una dimensión de la gradiente de Prewitt y Laplaciano para las operaciones de realce de bordes.

El desplazamiento y la operación de diferencia pueden ser realizados usando un proceso de sustracción de puntos en una doble imagen o un proceso en grupos de mascaras:

0	0	0
-1	0	0
0	0	0

Vertical

0	-1	0
0	1	0
0	0	0

Horizontal

En este proceso de debe notar que los coeficientes que son solo 0 no afecta al resultado final. Esto significa que como la mascara pasa por alto una región de la imagen teniendo un brillo constante (no existen bordes) arroja un resultado de 0. Esto representa una pendiente de brillo de 0, lo cual es exactamente lo que una región de brillo constante tiene.

La operación "Gradiente de Prewitt" forma un realzamiento de borde direccional. Usando un núcleo de 3 x 3, 9 imágenes en declive pueden ser generadas desde una imagen original. Cada borde esta orientado en uno de las 8 direcciones N, NE, E, SE, S, SO, O y NO.

Si se aplica la máscara sobre una región que tiene un constante de brillo, los coeficientes añaden un 0 al pixel de salida, ésto genera como resultado un 0, representando así una elevación de brillo de 0.

Empleando la máscara Este, un cambio de oscuro a claro viniendo de izquierda a derecha será acentuando. Esto es debido a una pendiente de brillo positivo del Este existe. Pendientes con brillo en otras direcciones

suman a un valor negativo, el cual es puesto a 0, o negro. Si al aplicar la Gradiente de Prewitt, la respuesta a esta operación para bordes genera resultados negativos, el valor de salida es puesto a 0, porque los brillo negativos son indefinidos. La gradiente de la imagen aparece como negro dondequiera que los brillos de la imagen original son constantes.

El realzamiento del borde laplaciano es una operación omnidireccional que aclara a todos los bordes en una imagen, sin hacer caso de su orientación. Esta operación esta basada sobre la razón de los cambios de la pendiente de brillo dentro de un núcleo de 3x3. La mascara Laplaciana comúnmente utilizada esta compuesta de un 8 en el centro y con –1 en los alrededores.

-1	-1	-1
-1	8	-1
-1	-1	-1

Los coeficientes aumentan a 0 y como en el filtro pasa-alto, los coeficientes de valores negativos se posicionan alrededor del coeficiente positivo mas alto. El realzamiento con el operador Laplaciano genera picos mas agudos en los bordes que es lo que hace la operación gradiente. Cualquier pendiente de brillo ya sea positivo o negativo, es acentuado dando el Laplaciano una calidad omnidireccional.

En el sistema visual humano: ojo-cerebro aplica un realzamiento Laplaciano semejante a cada cosa que es vista. Podemos simular el

sistema visual humano utilizando el operador Laplaciano para realzar una imagen original usando el proceso de doble imagen punto. Esta operación con frecuencia resulta en una imagen que aparece naturalmente mas agudo con una calidad optima.

La Fig. No. 5.2 muestra la respuesta del operador Laplaciano en borde de una sola dimensión. La imagen Laplaciano es negro dondequiera que los brillos originales son constantes o contienen cambios lineales. Los bordes son hechos por transiciones no lineales de brillo, entonces son sobreiluminados como blancos.

Otro tipo de operación es el realzamiento por segmento de línea, que es con frecuencia es utilizada para limpiar los bordes de una imagen siguiendo una operación de realzamiento de bordes. Esta operaciones esfatizan los segmentos de línea dentro de una imagen. La mascara para la operación de realzamiento de segmento de línea horizontal es:

-1	-1	-1
2	2	2
-1	-1	-1

Nuevamente, los coeficientes aumentan a 0, esto significa que las regiones de brillo constantes de la imagen original podrían llegar ha ser negros cuando son procesados. Solo los segmentos de líneas podrían permanecer sobreiluminados.

Los métodos mencionados para el realzamiento de bordes se pueden aplicar a la imagen capturada para determinar las caracteristicas mas resaltantes de un rostro. Otra aplicación es en las maquinas de visión, ya sea en aplicaciones de automatización, en inspección de líneas de ensamblado o reconocimiento de objetos, estos procesos son usualmente usados, primero para acondicionar la imagen antes de aplicar una subsiguiente operación de análisis de imagen procesada.

CAPITULO VI
PROCESAMIENTO DE COMPRESION DE IMAGEN

6.1 Objetivos de uso de compresión de datos de imágenes.

Las operaciones de compresión y/o descompresión de datos son generalmente utilizados para minimizar el volumen de los datos de una imagen digital. El principio de estas operaciones es para representar una imagen, con la misma calidad de nivel requerida, en una forma más compacta. Las operaciones de compresión extraen la informacion esencial de una imagen, así ésta puede ser reconstruida posteriormente Y la información que no es esencial es descartada.

En nuestro proyecto se han empleado una compresión de imagenes como un paso previo al almacenamiento y comunicación de datos y/o transporte electrónico. Esto se debe a que ambas operaciones son sensitivas a la cantidad de datos en una imagen y como sabemos, el proceso de adquisición de un cuadro de video ocupa la cantidad de 480 Kbytes. Tamaño extremadamente grande si se requiere realizar cualquiera de éstas operaciones. El almacenamiento de imagen se refiere al almacenamiento magnético y el transporte se refiere a la transferencia de una imagen sobre un enlace de datos.

Cuando se obtiene la cualidad de comprimir la cantidad de datos necesarios que representa una imagen, la cantidad de tiempo para transportarla también puede ser reducida, de igual forma, la cantidad de espacio requerido para el almacenamiento de datos es reducido.

En síntesis la compresión de datos reduce el número de bytes requeridos para representar un conjunto de datos Los conjuntos de datos gráficos son con frecuencia muy grandes en su forma original. La compresión reduce el tamaño de memoria (espacio en disco o cinta) requerido para almacenar un conjunto de datos. Consecuentemente esto tambien reduce la cantidad de tiempo requerido para transmitir un conjunto de datos sobre un enlace de comunicaciones a una determinada velocidad.

Para efectuar compresión de datos, son utilizados la terminología que a continuación se mencionan para dsecribir los esquemas de compresión.

a. **BITS/PIXEL :** El número promedio de bits requeridos para representar el valor del dato para un solo pixel de una imagen. Esto representa otro método de especificación de la razón de compresión y es comunmente usado cuando se especifica de compresion de imágenes a color.
b. **COMPRESION SIN PERDIDA (LOSSLESS):** Es la compresión que resulta sin la pérdida de datos ó información.
c. **COMPRESION CON PERDIDA (LOSSY):** Es la compresión que puede perder porciones de datos originales. Los esquemas de compresión con pérdida pueden obtener la más alta razon de

compresión cuando son comparados con el esquema de compresión sin perdida, al precio de perdida de datos.

d. **RAZON DE COMPRESION:** Es la razon del tamaño original del conjunto de datos al tamaño del conjunto de datos comprimidos.

$$\text{Razon de compresion} = \frac{\text{Numero de bytes en dato original}}{\text{Numero de bytes en dato comprimido}}$$

La razón de compresión generalmente es expresado como un valor numérico de un un solo dígito ó tambien en dos digitos, por lo general el segundo digito es el 1. Por ejemplo la razon de 12:1 significa 12 bytes del dato original son representados por un solo byte en el dato comprimido.

e. **ENCODIFICADOR:** Es el proceso que se realiza con la toma de un conjunto de datos originales y se produce en la salida el conjunto de datos comprimidos.

e. .**DECODIFICADOR:** Es el proceso que toma como entrada un conjunto de datos comprimidos y produce como salida el conjunto de datos originales.

f. **ENCODIFICACION A TIEMPO REAL:** Es un proceso que comprime un conjunto de datos suficientemente rapiodos para satisfacer las necesidades de aplicación especifica. Este tipo sojn utilizados en los sistemas de video telefonico.

g. **DECODIFICACION A TIEMPO REAL:** Es un proceso mediante el cual puede descomprimir un conjunto de datos suficientemente rápidos para satisfacer las necesidades de la aplicación destinada.
h. **CODEC:** Abreviación de Codificador/Decodificador.
i. **ENTROPIA:** Medida de aleatoriedad en un conjunto de datos.

6.2 Criterio de diseño.

La compresión de datos gráficos se emplea con mucha frecuencia en una gran variedad de aplicaciones y cada aplicación requiere de diversos criterios de diseño y construcción de los esquemas de compresión. Como son casi cualquier tarea de diseno, el intercambio debe hacerse entre las limitaciones de diseño para encontrar criterios de desempeño y costo. A continuación se menciona un analisis efectuado en el proceso, donde se menciona las limitaciones en el diseño de compresión y posibles opciones que permitan subsanar el proceso.

a. Limitaciones de diseño por costo:
   - Requiere solo software (bajo costo)
   - Requiere un procesador matemático de punto flotante (costo adicional)
   - Requiere Hardware para propósito especial (Mas costoso)
b. Limitaciones de diseño por tiempo:
   - La codificación y decodificación requieren su proceso en tiempo real.
   - Solo la decodificación requiere de tiempo real.

- Mas lento que el procesamiento de tiempo real.

c. Limitaciones por perdida de informacion:

- Sin pérdida de información; todos los datos son recuperados con precisión del original.
- Algunos datos se pierden, pero la pérdida no es detectable por el ojo humano.
- Gran cantidad de pérdida de información de los datos originales, pero la informacion requerida por la aplicación es retenida.

d. Razon de compresión:

- Compresión de baja satisfaccion (2:1 o 3:1)
- Compresión Media satisfactoria (5:1 o 10:1)
- Alta compresión requerida (20:1 a 100:1 o mas alto).

En diversas aplicaciones se requieren informacion gráfica comprimida. Algunos ejemplos son los siguientes:

1. Videodisc:

   Emplea dispositivo de almacenamiento de solo lectura.

   Presenta limitaciones de tiempo en la codificación, pero la decodificacion debe ser en tiempo real.

2. Video telefono:

   Emplea canal de comunicación de ancho de banda angosto.

   Presenta razon de compresión debido al ancho de banda angosto.

   La Codificación y Decodificación cercana al tiempo real que requiere igual cantidad de tiempo.

3. Teleconferencia:

   Emplea canal de comunicación de gran ancho de banda.

   Presenta una alta razón de compresión para alta resolución de ımagen.

   La codificación y Decodificación en tiempo real.

4. Consultas médicas remotas:

   Emplea canales de comunicación de ancho de banda angosta y grande.

   Presenta mínima perdida de informacion para evitar malos diagnosticos.

5. Medicion de satelites:

   Emplea enlaces satelitales.

   Presenta pérdidas de datos aceptables pocas veces.

6.3 Métodos de compresión de datos.

En lo que respecta la compresión de datos, existen tres métodos involucrados, estos son:

1. Por transformación.
2. Reduccion en la precision.
3. Minimizaron del numero de bits.

Por Transformacion:

Permite realizar la transformación de un conjunto de datos en otro conjunto de datos equivalente que de alguna manera es más pequeña que

la original. Algunas transformaciones reducen el numero de datos sueltos del conjunto.

Por reducción en la precisión:

Reduce la precision de los valores de datos individuales dentro de un conjunto de datos la cual reduce el numero de digitos binarios requeridos para representar cada valor. Este método solo puede reducir aleatoriamente una cadena de datos.

Por minimizacion del número de bits:

Representa codificando cada tipo de dato para minimizar el numero total de digitos binarios requeridos para representar cadenas enteras de datos.

6.4 Reconocimiento de patrón de datos en dos dimensiones.

En esta sección se analiza uno de los esquemas más complejos que se aplican a un conjunto de datos de una matriz de dos dimensiones. Los subconjuntos de los modelos que existen en dos dimensiones que son utilizados se muestran en la Fig. No. 6.1. El cual resume éstos esquemas.

La codificación DCT (Discrete Cosine Transform: Transformada discreta del coseno) es uno de los esquemas más complejos que pueden producir la más alta razón de compresión, pero tiene la inconveniente de las pérdidas producidas en el procesamiento de las señales. La transformación DCT puede reducir la magnitud de muchos valores de datos en un mismo conjunto de datos. Esto también reduce la

significancia de muchos valores de datos, que permite menos valores importantes para ser bajados desde la cadena de datos.

TRANSFORMADA DISECRETA DEL COSENO

Muchas transformaciones matemáticas existentes, en general, transforman un conjunto de valores de datos desde un sistema de mediciones dentro de otro. Algunas veces los datos representados en el nuevo sistema tiene propiedades que facilitan la compresion de datos. Algunas transformaciones matematicas tienen que ser inventadas con el solo proposito de la compresion de datos. Otras tienen que ser tomados desde varias aplicaciones y aplicados a la compresion de datos.

Una lista parcial se muestra a continuacion:

1. DFT : Transformada Discreta de Fourier.
2. DCT : Transformada Discreta del coseno.
3. HHT : Transformada de Hadamard-Haar.
4. KLT : Transformada de Karhunen-Loeve.
5. SHT : Transformada de Slant-Haar.

De todas las transformadas que se mencionan, la predominante y la mas empleada en la compresion de imágenes digitales es la Transformada Discreta del Coseno. Esto por las razones siguientes:

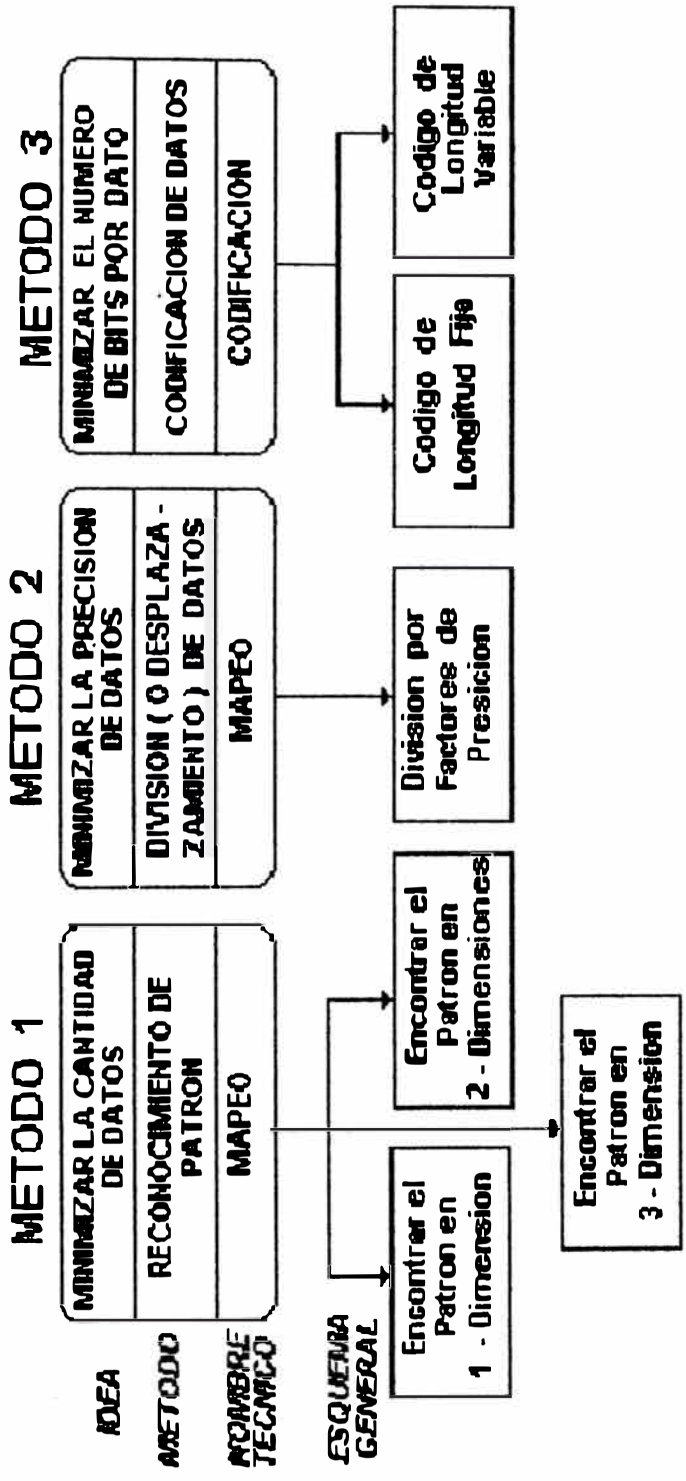


Fig. N° 6.1 Esquemas generales de las tres mayores Técnicas de compresión.

a. La DCT (Transformada Discreta del Coseno) tiene buenas cualidades computacionales, principalmente un algoritmo rápido para la implementación de dicha transformada.
b. Ha demostrado que la DCT produce visualmente mejor calidad de imagen a la más alta razón de compresión comparados con otros esquemas de transformadas.

La FDCT (transformada Discreta del coseno directa) trannsforma un bloque de datos originales a un nuevo conjunto de valores.

La IDCT (Transformada Discreta del coseno Inversa) invierte este proceso y restaura los valores de datos originales.

Sin embargo en la práctica, hay alguna información que se pierde debido a dos factores:

1. El valor del coseno no puede ser calculado exactamente porque tiene números trascendentales.
2. Los calculos repetitivos usando números precisos limitados introducen un error por aproximación y redondeo en los resultados finales.

Las variaciones entre los valores de los datos restaurados y los valores de los datos originales son típicamente pequeños, pero la cantidad de errores dependen del método usado para calcular la DCT.

La DCT tiene la ventaja de que puede ser aplicado a cualquier tamaño de bloque de datos, pero ha sido comprobado que seleccionando un bloque de 8x8 produce una buena razon de compresion mientras mantiene una buena calidad en las resoluciones de la imagen. También al

mismo tiempo los algoritmos fueron implementados comercialmente en bloques de 8x8 lo cual puede ser implementado en un solo chip con tecnologia LSI, mientras que un bloque de 16x16 no podria hacerse. Lo anteriormente mencionado implica que la DCT sea aplicada en un bloque de datos de 8x8.

ESQUEMAS DE COMPRESION QUE USAN LA TRAANSFORMADA DISCRETA DEL COSENO (DCT).

FORMATO	**IMPLEMENTACION**	**COMENTARIOS**
JPEG	DCT	Para imágenes estaticas.
MPEG	DCT	Para imágenes en movim.
CCITT H.261	DCT	Para telefonia visual.

La forma en que ha sido definida la DCT JPEG es de la siguiente manera:

a) La Transformada Discreta del coseno Directa (FDCT)

FDCT : Svu(u,v) = --- (Cu * Cv) Sxy **Cos((2x+1)**u /16)*Cos((2y+1)*

b) La Transformada discreta del Coseno Inverso (IDCT)

IDCT : Sxy(x,y)= --- (Cu*Cv)*Suv* cos((2x+1)*u /16)*cos((2y+1)*

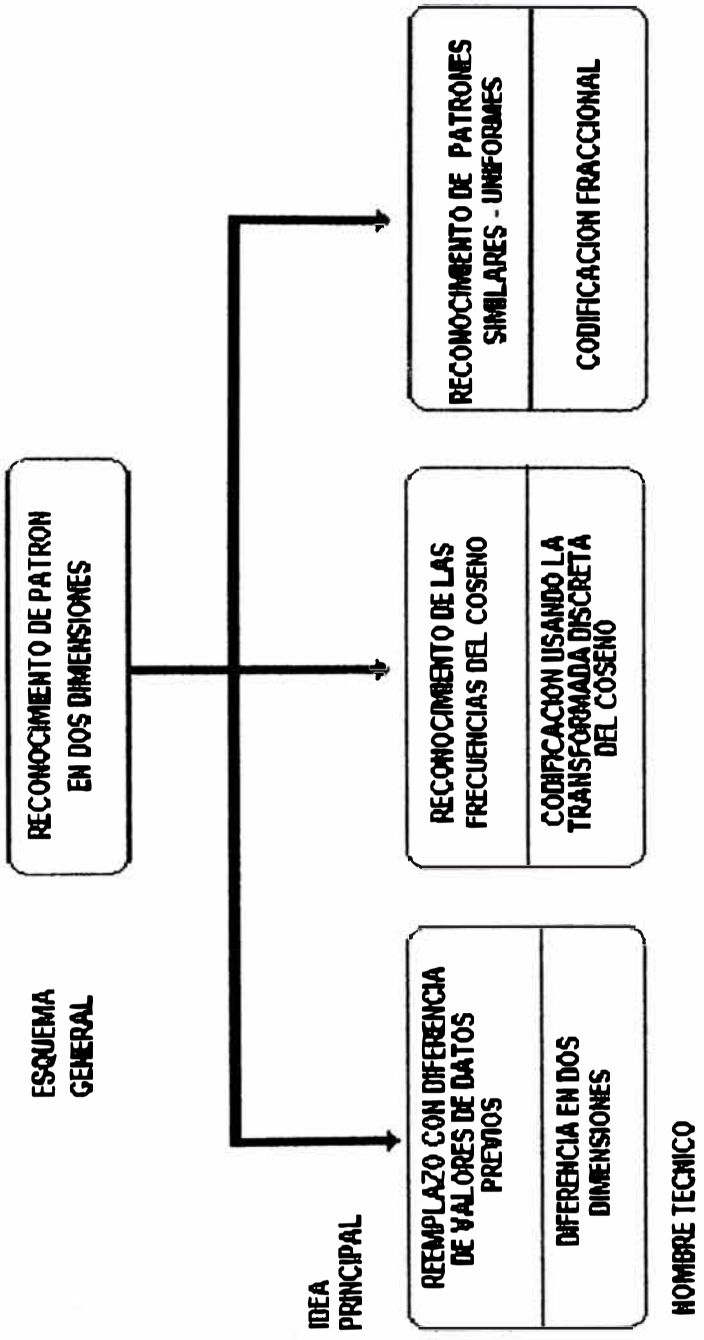


Fig. N° 6.2 Clasificación dee los esquemas de reconocimiento de patrones en dos dimensiones.

La DCT puede ser facilmente implementada usando las formulas anteriores, pero existen algoritmos que utilizan una serie de multiplicaciones y sumas, que llegan al resultado mucho más rápido.

Con las fórmulas escritas anteriormente, cada valor calculado en el proceso de transformación involucra a toda una matriz de 8x8. Cada matriz de transformacion representa una combinación de frecuencias del seno en dos dimensiones. La idea es determinar cuanto de cada patrón de frecuencia existe en un bloque de datos de 8x8. Una medicion de cada frecuencia es calculada en dos pasos básicos.

Primero: cada termino de una matriz de transformacion especifica es multiplicada por sus correspondientes terminos en el bloque de datos de 8x8.

Segundo: Todos estos productos son sumados en un solo valor. Este nuevo valor es una medida de la cantidad de estas frecuencias que están contenidas en el bloque de datos. En esencia, la transformada discreta del coseno varia los valores originales que tipicamente representan la intensidad de color en valores que representan la frecuencia del coseno sobre un area de 8x8.

Veamos algunos ejemplos que nos ayudaran a aclarar estas ideas, en un bloque de datos de 8x8, son transformadas en el tiempo en cosenos discretos usando la DCT directa y restauramos el bloque original usando la DCT inversa. Ambos procedimientos de la DCT fueron calculados con la implementacion de las ecuaciones mostradas previamente. En todas las operaciones aritmeticas se utilizaron números de doble precisión.

En los ejemplos siguientes; los valores de datos originales representan la intensidad de color en el rango de 0 a 255. La DCT produce mejores valores para la compresion si los datos originales estan en un rango que estan centrados alrededor del origen. En algun lugar, los valores usados en estos ejemplos son mostrados en sus formas cambiadas, donde se resto 128 a cada valor del original produciendo valores en el rango –128 a 127.

Consideremos el ejemplo de la Fig 6.3 donde el bloque de datos representa una región de color solida en estos datos no hay frecuencia y consecuentemente todos los términos excepto el superior de la izquierda son ceros.

DATOS ORIGINALES	**VALORES DESPUES DEL FDCT**
127 127 127 127 127 127 127 127	1016 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0

Los valores restaurados despues del IDCT	Diferencias (error)
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0
127 127 127 127 127 127 127 127	0 0 0 0 0 0 0 0

Fig No 6.3

Los datos de la Figura anterior que resultan de la FDCT pueden ser comprimidos por almacenamiento de un solo valor solo (1016) y algun codigo especial que indique que todos los valores restantes son ceros.

Considere el ejemplo de la Fig No. 6.4, donde los datos representan una superficie donde los valores de color cambian ligeramente en todo el bloque de 8x8. Si aplicamos la FDCT obtenemos que en el resultado no todos los valores son ceros. Estos terminos que no son seros corresponden a las frecuencias mas bajas de los datos originales. Esto datos pueden ser comprimidos por almacenamiento solo los terminos que no son ceros.

127	127	127	127	127	127	127	127	0	0	0	0	0	0	0	0
60	59	58	57	56	55	54	53	424	18	0	2	0	1	0	0
59	58	57	56	55	54	53	52	18	0	0	0	0	0	0	0
58	57	56	55	54	53	52	51	0	0	0	0	0	0	0	0
57	56	55	54	53	52	51	50	2	0	0	0	0	0	0	0
56	55	54	53	52	51	50	49	0	0	0	0	0	0	0	0
55	54	53	52	51	50	49	48	0	0	0	0	0	0	0	0
54	53	52	51	50	49	48	47	0	0	0	0	0	0	0	0
53	52	51	50	49	48	47	46	0	0	0	0	0	0	0	0

Los valores restaurados despues del IDCT se muestran a continuacion:

VALORES RESTAURADOS								**DIFERENCIAS (ERROR)**							
60	59	58	57	56	55	54	53	0	0	0	0	0	0	0	0
59	58	57	56	55	54	53	52	0	0	0	0	0	0	0	0
58	57	56	55	54	53	52	51	0	0	0	0	0	0	0	0
57	56	55	54	53	52	51	50	0	0	0	0	0	0	0	0
56	55	54	53	52	51	50	49	0	0	0	0	0	0	0	0
55	54	53	52	51	50	49	48	0	0	0	0	0	0	0	0
54	53	52	51	50	49	48	47	0	0	0	0	0	0	0	0
53	52	51	50	49	48	47	46	0	0	0	0	0	0	0	0

Fig No 6.4

La cantidad de ruido (en terminos de altas frecuencias)que tenemos que omitir en un bloque de datos comprimidos determina el resultado de la calidad de imagen y es el factor determinate en la cantidad de

compresion disponible. Lamentablemente la cantidad de ruido que nosotros podemos omitir mientras todavia se tenga una buen acalidad de imagen es dependioente del bloque de datos. En resumen la DCT potencialmente puede para ciertos tipos de datos, sacar muchos terminos ceros en el bloque de datos de 8x8 y la compresion puede ser archivada por simple omision de todos los terminos cero.

A continuacion se muestra el Programa DCT.C que implementa la Tranformada Discreta del Coseno Directo, asi como tambien la Transformada Discreta del coseno inverso. Este programa ha sido efectuado y verificado su funcionamiento. Estas rutinas pueden ser implementadas dentro de un software de compresion de imágenes.

Programa: DCT.C

```
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>

#define FILE 8
#define COLUMNA 8
#define PI 3.14159265

main()

        /* Definimos la matriz de valores de entrada. */
        int DATA_INI[8][8];
        /* Definimos la matriz de valores de salida despues de aplicar la FDCT. */
        int DATA_fin[8][8];
        /* Definimos la matriz de valores de salida despues de aplicar la IDCT. */
        int DATA_RES[8][8];
        /*Variables generales. */
        int x=0, y=0;
        int u,v;
        float Cuv;
        double tempo, tempo1;
```

```
double i;
double f;
/* Llenamos la matriz de 8x8 de datos. */

cirscr();
for (y=0;y<FILA;y++){
        printf("Fila No. %d \n",y);
                scanf("\t %d",&DATA_IN[y][x]);
        }
sound(1000);
delay(100);
nosound();
}

/* Impresión de la matriz de datos */
cirscr();
printf(bloque de datos originales\n");
for (y=0;y<FILA;y++){
        printf("Fila No. %d",y);
        for (x=0;x<COLUMNA;x++){
                printf("\t %i",DATA_IN[y][x]);
        }
        printf("\n");

}
getch();

/* Implementamos la DCT directa. */
for(u=0; u<8;u++)
{
        for(v=0;v<8;v++)
        {
                tempo=0;
                for(x=0;x<FILA;x++){
                        for(y=0; y<COLUMNA;y++)

tempo=tempo+(DATA_IN[y][x])*cos((2*x+1)*u*PI/16)*cos((2*y+1)*v*PI/
16);
                }
                tempo=tempo/4;
                if(u==0;v==0)
                        {
                        if(u==0 & v==0)
                                Cuv=0.5;
                        else
                                Cuv=0.707;
                        tempo=tempo*Cuv;
```

```
                }
        else{
                Cuv=1;
                tempo=tempo*Cuv;
                }
        f=modf(tempo,&i);
        if(i>=0){
                if(f>0.5)
                        i=i+1;
                        }
        if(i<0){
                f=f*(-1);
                if(f>0.5)
                        i=i-1;
        DATA_FIN[u][v] = i; /* Suv(u,v). */
    }
}
printf("\nValores del Bloque despues de aplicar la FDCT\n");
for (y=0;y<FILA;y++){
    printf("S%d0=> ",y);
    for (x=0:x<COLUMNA;x++){
        printf("\t %i",DATA_FIN[x][y]);
    }
    printf("\n");
}

getch();

/*

implementamos la DTC inversa.
*/
for(y=0;y<FILA:y++)
{
    for (X=0;x<COLUMNA;x++)
    {
        tempo=0;
        for(u=0;u<8;u++){
            for(v=0;v<8;v++)
            {
tempo=DATA_FIN[u][v])*cos((2*x+1)*u*PI/16)*cos((2*y+1)/v/PI/16);
                if(u==0 / v==0){
                    if (u==0 & v==0)
                        Cuv=0.5:
                    else
                        Cuv=0.707;
                    tempo=tempo*Cuv;}
```

```
                        else {
                                Cuv=1;
                                Tempo=tempo*Cuv;}
                        Tempo1=tempo1+tempo;
                }/* fin del for v */
        }/* fin del for u */
        tempo1=tempo1/4;
        f=modf(tempo1,&i);
        if(i>0){
                if(f>0.5)
                        i=i+1;
                }
        if(i<0){
                f=f*(-1);
                if(f>0.5)
                        i=i-1;
                }
        DATA_RES[x][y]= i; /* Suv(u,v).  */
        Tempo1=0;
    }/* fin del for y */
} /* fin del for x */
printf("\nValores del Bloque despues de aplicar la IDCT\n);
for (y=0;y<FILA;y++){
        printf("T%d0 =>",y);
        for (x=0;x<COLUMNA;x++){
                printf("\t %i", DATA_RES[x][y]);
        }
        printf('\n");
}
getch();

/*
// Se imprimen las diferencias y errores entre el bloque de datos
// original y el bloque de datos de la DCT inversa.
*/

printf("\nDiferencias y errores\n");
for (y=0;y<FILA; y++){
    printf("E%d0=>",y);
    for (x=0;x<COLUMNA; x++){{
        printf("\ %i",DATA_INI[y][x]-DATA_RES[x][y]);
    }
    printf("\n");
}
return 0;
```

6.5 Software de desarrollo.

En el desarrollo del sistema de adquisición de señales de video se consideró también el diseño y desarrollo del software, de tal modo que el manejo del sistema por cualquier usuario se realce en forma práctica. Este programa tiene la ventaja de presentar ventanas de presentación, que le muestren al usuario los diversos pasos a realizar en el proceso de adquisición de un cuadro de video, asi como también en la manipulación de algoritmos matemáticos que se apliquen en la imagen. Este software se desarrolloen el lenguaje de Programacion Borland C++ y se ha estructurado del modo siguiente:

a) Software de presentación y llamadas a las diferentes subrutinas.
b) Software de visualización en poantalla del monitor del cuadro de video.

Todo este software, han sido compilados en un solo proyecto, generando de este modo un programa ejecutable denominado VIDEO.EXE. y sus partes son los siguientes:

1. VIDEO.C
2. GRAFICA.C
3. MOUSE.C

El listado de los programas se muestran a continuacion:

SOFTWARE 1 : VIDEO.C

```
/*************************************************/
```

```
/*       SOFTWARE DE PROCESAMIENTO PRINCIPAL                    */
/*                                                               */
/* ************************************************************/
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
#include <alloc.h>
#include <stdarg.h>
#include <math.h>

/* Definimos las funciones a emplear en los programas */
void mensajes(void);
void boton_peq(void);
int leer_x_mouse(void);
int leer_y_mouse(void);
int leer tecla_mouse(void);
int leer_opcion(int,int);
int opcion_menu(int,int);
int opcion_adq(int,int);
int opcion_analisis(int,int);
void no_ver_mouse(void);
void ver_mouse(void);
void stop_mouse(void);
void limite_mouse(int,int,int,int);
void univ(void);
void limpiar_pantalla (void);
void menu_principal(void);

void primera_ventana(void);
void segunda_ventana(void);
void tercera_ventana(void);
void cuarta_ventana(void);
void botones(int, int, int, int);
void boton_pulsado(int, int, int, int, int);
void dibuja_cuadrado(int, int, int, int);


void gprintf(int, int, char *formato,...);
void adquisicion(void);
void analisis(void);
void grafica_2(void);
void analisis_tarjeta(void);

/* Definimos las variables del programa  */
int salir=0;
```

```
int tiempo=   ;
unsigned long int cuenta_i;
unsigned long int_far *p = 0x0040006C;

/* Definicion de las palabras de control;
int write_mem = 0xC0;
int read_mem  = 0xE0;
int clear_count = 0x00;
int parar  = 0xFF;
int grabar  = 0x48;
int adquirir  =  0x08;

/* Definicion  de los puertos.  */
int port0 = 0x300;
int port1 = 0x301;
int port2 = 0x302;
int port3 = 0x303;


/*************PROGRAMA PRINCIPAL******************/
void main()

        int gdriver = DETECT, gmode, errorcode;
        char nombre[12];
        initgrap(&gdriver, &gmode, "c\\borlandc\\bgi\\");
        errorcode = graphresult();
        if (errorcode! = grOk)
        {
                printf("Graphics error: %s\n", grapherrormsg(errorcode));
                printf("Press any key to halt:");
                getch();
                exit(1);
        }
        cuenta_i= *p;
        floodfill(1,1,15);
        cleardevice();
        mensajes();
        return ();
        }

        void mensajes(void)
        {
                        int xm,ymmenu;
                        int gdriver = DETECT, gmode;
                        settextstyle(0, HORIZ_DIR,1);

                        setcolor(15);
```

```
rectangle(5,45,635,435);
setfillstyle(1,1);
floodfill(6,46,15);

dibuja_cuadrado(145,105,495,375);

primera_ventana();
delay(1000);

setcolor(15);
rectangle(5,5,635,40);
rectangle(5,45,635,435);
rectangle(5,440,635,475);
setcolor(12);
setcolor(14);
rectangle(144,104,496,376);
setfillstyle(1,1);
floodfill(146,105,14);
setcolor(1);
rectangle(144,104,496,376);
setcolor(15);
rectangle(160,120,480,360);
setfillstyle(1,0);
floodfill(161,121,15);

segunda _ventana ();
boton_peq();
tercera_ventana();
boton_peq();
cuarta_ventana();
boton_peq();

for(;;){
                no_ver_mouse();
                salir=0;
                limpiar_pantalla();
                setcolor(15);
                rectangle(5,45,635,435);
                setfillstyle(1,1);
                floodfill(6,50,15);

                botones(250,100,400,130,10);
                outtextxy(270,113,"MENU PRINCIPAL");
```

```
                botones(200,170,220,190,4); //opcion 1
                botones(200,205,220,225,4); //opcion 2
                botones(200,240,220,260,4); //opcion 3

                botones(200,275,229,295,4); //opcion 4
                botones(200,310,220,330,4); //opcion 5
                botones(500,350,560,370,7); // salir

                setcolor(0);
                outtextxy(207,177,"1");
                outtextxy(207,212,"2");
                outtextxy(207,247,"3");
                outtextxy(207,282,"4");
                outtextxy(207,317,"5");
                outtextxy(511,357,"SALIR");

                ver_mouse();
                limita_mouse(5,635,45,435);
                ver_mouse();

                for(;salir==0)
{

        xm=leer_x_mouse();
        ym=leer_y_mouse();
        setcolor(14);
        if((xm>200 & xm<220)&(ym>170 & ym<190)){
                        no_ver_mouse();
                        outtextxy(280,177,"ADQUISICION");
                        ver_mouse();}
else if ((xm>200& xm<220)&(ym>205 & ym<225)){
                no_ver_mouse();
                outtextxy(280,212,"REPRODUCCION");
                ver_mouse();}
else if ((xm>200& xm<220)&(ym>240 & ym<260)){
                no_ver_mouse();
                outtextxy(280,247,"ANALISIS");
                ver_mouse();}
else if ((xm>200& xm<220)&(ym>275 & ym<295)){
                no_ver_mouse();
                outtextxy(280,282,"TRANSMISION");
                ver_mouse();}
else if ((xm>200& xm<220)&(ym>310 & ym<330)){
                no_ver_mouse();
                outtextxy(280,317,"IMPRESION");
                ver_mouse();}
```

```
else{
        if((*p – cuenta_i)/18.2>=0.5){
                no_ver_mouse();
                setcolor(0);
                setfillstyle(1,0);
                floodfill(26,453,15);
                outtextxy(280,177,"ADQUISICION");
                outtextxy(280,212,"REPRODUCCION");
                outtextxy(280,247,"ANALISIS");
                outtextxy(280,317,"IMPRESION");
                cuenta_i = *p;
                ver_mouse();}

if(1==leer_tecla_mouse()){
        menu=opcion_menu(xm,ym);
        if(menu==6){
                        boton_pulsado(500,350,560,370,7);
                        stop_mouse();
                        botones(500,350,560,370,7);
                        closegraph();
                        break;          //Fin del Programa.
                        }
switch (menu){
case 1: //opcion 1
                boton_pulsado(200,170,220,190,4);
                stop_mouse();
                botones(200,170,220,190,4);
                adquisicion();
                salir=1;
                break;
case 2: //opcion 2
                boton_pulsado(200,205,220,225,4);
                stop_mouse();
                botones(200,205,220,225,4);
                no_ver_mouse();
                grafica_2();
                initgraph(&gdriver,&gmode,
                "c:\\borlandc\\bgi\\");
                setcolor(15);
                rectangle(5,5,635,40);
                rectangle(5,45,635,435);
                rectangle(5,440,635,475);
                setcolor(12);
                salir=1;
                break;

case 3: //opcion 3
```

```
                    boton_pulsado(200,240,220,260,4);
                    stop_mouse();
                    botones(200,240,220,260,4);
                    analisis ();
                    salir=1;
                    break;
        case 4: //opcion 4
                    boton_pulsado(200,275,220,295,4);
                    stop_mouse();
                    botones(200,275,220,295,4);
                    break;

        case 5: //opcion 5
                    boton_pulsado(200,310,220,330,4);
                    stop_mouse();
                    botones(200,310,220,330,4);
                    break;
              }             /* fin de la sentencia switch */
        stop_mouse();
        }             /* fin del if  */
}       /* fin del for infinito  */

}

}

int leer_opcion(int x, inty)
        {
        int opcion;
        opcion=0;
        if((x>390 & x<465)&(y>320 & y<345)) opcion=1;
        return(opcion);
}

int opcion_menu(int x, int y)
        {
        int opcion;
        opcion=0;
        if ((x>200 & x<220)&(y>170 & y<190)) opcion = 1;
        if ((x>200 & x<220)&(y>205 & y<225)) opcion = 2;
        if ((x>200 & x<220)&(y>240 & y<260)) opcion = 3;
        if ((x>200 & x<220)&(y>275 & y<295)) opcion = 4;
     if ((x>200 & x<220)&(y>310 & y<330)) opcion = 5;
        if ((x>500 & x<560)&(y>350 & y<370)) opcion = 6;
       return(opcion);
}
```

```
/* Rutina que realiza la adquisicion de un cuadro de video  */
void adquisicion (void)

        int xm,ym,opcion,color;
int valor;
float tiempo2;
void *buffer;
unsigned tamanio;
FILE *ou;
unsigned char dato=0x0;
 char nombre[12];

stop_mouse();
no_ver_mouse();

setcolor(0);
rectangle(5,45,635,435);
setfillstyle(1,0);
flodfill(26,453,15);
limpiar_pantalla();
setcolor(15);
rectangle(5,45,635,435);
setfillstyle(1,1);
floodfill(6,50,15);

setcolor(15);
rectangle(270,230330,250);
rectangle(269,229,331,251);
rectangle(340,120,600,350);
circle(70,150,6); outtextxy(95,147,"LIMPIAR CONTADORES");
circle(70,180,6); outtextxy(95,177,"SETEAR CONTADORES");
circle(70,210,6); outtextxy(95,207,"COMENZAR ADQUISICION");
circle(70,240,6); outtextxy(95,237,"TIEMPO DE ADQUISICION");
circle(70,270,6); outtextxy(95,267,"ALMACENAR EN DISCO");
circle(70,150,8);
circle(70,180,8);
circle(70,210,8);
circle(70,240,8);
circle(70,270,8);

setfillstyle(1,0);
floodfill(70,150,15);
floodfill(70,180,15);
floodfill(70,210,15);
floodfill(70,240,15);
floodfill(70,270,15);
floodfill(341,121,15);
```

```
botones(70,320,120,340,6);
botones(200,320,250,340,6);
setcolor(0);
outtextxy(87,327,"OK");
outtextxy(212,327,"ESC");

setfillstyle(1,0);
floodfill(271,231,15);

setcolor(8);
gprint(280,237,"%d ms, tiempo);
ver_ mouse();
for(;;)
{
        xm=leer_x_mouse();
        ym=leer_y_mouse();
        if(1==leer_tecla_mouse())
        {
                opcion=opcion_adq(xm,ym);
                no_ver_mouse();
                if (opcion==1){
                        setfillstyle(1,12);
                        floodfill(70,150,15);
                        valor=1:}
                if (opcion==2){
                        setfillstyle(1,12);
                        floodfill(70,180,15);
                        valor=2:}
                if (opcion==3){
                        setfillstyle(1,12);
                        floodfill(70,210,15);
                        valor=3:}
                if (opcion==4){
                        setfillstyle(1,12);
                        floodfill(70,240,15);
                        valor=4:}
                if (opcion==5){
                        setfillstyle(1,12);
                        floodfill(70,270,15);
                        valor=5:}
                ver  mouse();
                if(opcion)==6){
        boton_pulsado(70320,120,340,6);
        stop_mouse();
        botones(70,320,120,340,6);
        setfillstyle(1,0);
```

```
switch(valor){

        case 1:
                valor=getcolor():
                floodfill(70,150,15);
                //Limpiar los contadores
                outport(port3,clear_count);
                //
                setcolor(14);
                outtextxy(360,150,"Los contadores han sido");
                outtextxy(360,170,"borrados...");
                sound(2000);
                delay(500);
                nosound();
                delay(900);
                setcolor(0);
                setfillstyle(1,0);
                no_ver_mouse();
                flodfill(360,150,15);
                ver_mouse();
                setcolor(valor);
                valor=0;
                break;
        case 2:
                valor=getcolor():
                floodfill(70,180,15);
                //Limpiar los contadores
                outport(port2,clear_count);
                //
                setcolor(14);
                outtextxy(360,150,"Los contadores han sido");
                outtextxy(360,170,"puestos para que
almacenen");
                outtextxy(360,190,"1 cuadro completo de
video");

        outtextxy(360,210,".....");
                outtextxy(360,270,"capture cuando esta listo");
                sound(2000);
                delay(500);
                nosound();
                delay(2000);
                setcolor(0);
```

```
                    setfillstyle(1,0);
                    no_ver_mouse();
                    flodfill(360,150,15);
                    ver_mouse();
                    setcolor(valor);
                    valor=0;
                    break;
            case 3:
                    valor=getcolor():
                    floodfill(70,210,15);
                    //Comenzar adquisicion
                    outportb(port1,adquirir);
                    delay(tiempo);
                    outportb(port1,parar);
                    //
                    setcolor(14);
                    outtextxy(360,150,"Se ha adquirido un
cuadro");
                    outtextxy(360,170,"de video y esta
almacenado");
                    outtextxy(360,190,"en memoria del sistema");
                    outtextxy(360,210,"de video ...");
                    outtextxy(360,270,"Si desea almacenar en un");
                    outtextxy(360,270,"fichero.");
                    sound(2000);
                    delay(500);
                    nosound();
                    delay(2000);
                    setcolor(0);
                    setfillstyle(1,0);
                    no_ver_mouse();
                    flodfill(360,150,15);
                    ver  mouse();
                    setcolor(valor);
                    valor=0;
                    break;
            case 4:
                    valor=getcolor():
                    floodfill(70,240,15);
                    setcolor(14);
                    outtextxy(360,150,"Ingrese el tiempo que se");
                    outtextxy(360,170,"utilizar para almacenar
un"):
                    outtextxy(360,190,"cuadro de video");
                    outtextxy(360,250,"El tiempo esta en miliseg");
                    outtextxy(360,270,"Y por defecto es 33 mseg");
                    outtextxy(360,290,"fichero=>.");
```

```
            setcolor (14);
            gprint(280,237,"%d ms", tiempo);
            leerf(440,290,4,0, & tiempo2,14);
            setcolor(0);
            gprintf280,237,"%d ms, tiempo):
            tiempo=tiempo2;
            tiempo2=0;
            sound(2000);
            delay(500);
            nosound();
            delay(2000);
            setcolor(0);
            setfillstyle(1,0);
            no_ver_mouse();
            flodfill(360,150,15);
            ver_mouse();
            setcolor(8);
            gprintf(280,237,"%d ms",tiempo);
            setcolor(valor);
            valor=0;
            break;
case 5:/* Almacenar en disco  */
            valor=getcolor();
            floodfill(70,270,15);
            setcolor(14);
outtextxy(360,150,"Escriba el nombre del fichero");
outtextxy(360,190,"de video...=> ");
tamanio=imagesize(180,380,460,420);
if(tamanio=-1){
buffer=malloc(tamanio);
            if (buffer){
            getimage(180,380,460,420,buffer);}
            }
setcolor(12);
for(xm=0;xm<=11,xm++){
            nombre[xm]=getch();
            if (nombre[xm]!= V')
            gprintf(480+xm*9,190,"%c", nombre[xm]);
            else{
                        nombre[xm]='\x0';
                        break;}
                        }
            setcolor(0);
            setfillstyle(1,0);
            floodfill(360,150,15);
            dibuja_cuadrado (180,380,460.420);
            setcolor(14);
```

```
                    gprintf(155,460,"almacenando en disco,
                    espere un momento...";
                    ou=fopen(nombre,wb");
                    outportb(port1,parar);
                    outportb(port2,clear_count);
                    outportb(port1,read_mem);
                    for(xm=0;xm<=254;xm++){
                            setcolor(0);
                            gprint(305,397,"%2.0f%%",xm*0.30);
                            for(ym=0;ym<=2064;ym++){
                                    dato=inportb(0x300);
                                    fprintf(ou,"%c",dato);}
                            setcolor(4);
                            line(xm+193,390,xm+193,410);
                            if(xm>=115)
                                    setcolor(4);
                            else
                                    setcolor(7);
                                    gprint(305,397,"%2.0f
                                    %%,xm*0.39);
                            }

                            setcolor (0);
                            sound(1000);
                            gprint(305,397,"100 %");
                            gprint(155,460,"Almacenando en
                            disco, espere un momento..."
                            fclose(ou);
                            setcolor(14);
                            gprint(155,460,"proceso Finalizado");
                            delay(500);
                            setcolor(500);
                            setcolor (0);
                            gprint(155,460,"Proceso Finalizado");
                            setcolor(valor);
                            valor=0;
                            nosound();
                            break;

                    }
            }
if(opcion==7){
            boton_pulsado(200,320,250,340,6);
            stop_mouse();
            botones(200,320,250,340,6);
            break;}
            }
```

```
}       /* fin del for infinito  */
}

int opcion_adq(int x, inty)
        {
        int opcion;
        opcion=0;
       if ((x>60 & x<80) & (y>140 & y<160)) opcion=1;
        if ((x>60 & x<80) & (y>170 & y<190)) opcion=2;
         if ((x>60 & x<80) & (y>200 & y<220)) opcion=3;
        if ((x>60 & x<80) & (y>230 & y<250)) opcion=4;
         if ((x>60 & x<80) & (y>260 & y<280)) opcion=5;
         if ((x>70 & x<120) & (y>320 & y340)) opcion=6;
        if ((x>200 & x<250) & (y>320 & y<340)) opcion=7;
       return(opcion);
}

void limpiar_pantalla(void)
{
        int color,izq=5,der=635,i;
        color= getcolor();
        setcolor(0);
        for(i-1<=316;i++,izq++,der--)
                                {
                                line(izq,46,izq,434);
                                line(der ,46, der,434);}
                setcolor (color);
        }

/* Subrutina que muestra en la pantalla del monitor, las muestras adquiridas de la
senal de video  */

voidanalisis(void)
{
        unsigned char dato:
        FILE *ou;
int menu;
char nombre[12];
int columna,muestra,tecla,xm1,ym1;
unsigned int i;
float cuenta;

stop_mouse();
no_ver_mouse();

setcolor(0);
rectangle(5,45,635,435);
```

```
setfillstyle(1,0);
floodfill(26,453,15);
limpiar_pantalla();

setcolor(15);
rectangle(10,50,630350);
rectangle(5,45,635,435);
setfillstyle(1,1);
floodfill(6,50,15);

setcolor(0);
botones(40,320,120,340,4); //Avanzar
botones(505,320,600,340,4); //Retroceder
botones(275,320,355,340,4); //Salir
gprintf(53,327,"AVANZAR");
gprintf(515,327,"RETROCEDER");
gprintf(294,327,"SALIR");
setcolor(14);
gprintf(50,360,"Ingrese el nombre del archivo:");
setcolor(12);
for(xm1=0;xm1<=11;xm1++){
        nombre[xm1]=getch();
        if (nombre[xm1]!='\r')
                gprintf(300+xm1*9,360,"%c",nombre[xm1]);
        else{
                nombre[xm1]='\x0';
                break;}

}
setcolor(13);
gprint(480,450,"Muestra No.");
gprint(480,460,"Amplitud =>");
ver_mouse();
ou_fopen(nombre,"rb");
cuenta=0;
xm1=0;
columna=11;
salir=0;
moveto(11,150);
while(!feof(ou) && salir==0){
                        fscanf(ou,"%c",&dato);
                        muestra=dato;
                        muestra=255-muestra;
                        setcolor(14);
                        lineto(columna,muestra+52);
                        if(columna==629){
        ver_mouse();
```

```
        for(;salir!=2;){
        xm1=leer_x_mouse();
        ym1=leer_y_mouse();
        if(1==leer_tecla_mouse()){
                        menu=opcion_analisis(xm1,ym1);
                        if(menu==2){
                                boton_pulsado(275,320,355,340,4);
                                stop_mouse();
                                botones(275,320,355,340,4); //salir
                                salir=1;
                                break;          //Fin de Analisis.

                        If(menu==1){
                                Boton_pulsado(40,320,120,340,4);
                                Stop  mouse();
                        botones(40,320,120,340,4); //
                         salir=2;}
                    if(menu==3){
                                boton_pulsado(505,320,600,340,4);
                                stop_mouse();
                                botones(505,320,600,340,4); //
                                salir=2;}
                    } // fin del if
                setcolor(14);
                gprintf(580,450,"%.0f",cuenta+xm1-10);
                gprintf(580,460,"%d",ym1-50);
                setcolor(0);
                gprintf(580,450,"%.0f",cuenta+xm1-10);
                gprintf(580,460,"%d",ym1-50);
                } // fin del for
                moveto(11,150);
                if (salir==2)
                        salir=0;
                menu=0;
                cuenta+=629;
                columna=11;
                no_ver_mouse();
                setfillstyle (1,0);
                floodfill(50,60,15);
                } //findel if
        columna+=1;
        } //fin del while
        fclose(ou);
        floodfill(500,470,15);
}

int opcion_analisis(int x,int y)
```

```
        {
        int opcion;
        opcion=0;
        if ((x>40 & x<120) & (y>310 & y <330)) opcion=1;  // Avanzar
         if ((x>275 & x<355) & (y>310 & y <330)) opcion=2;  // Salir
        if ((x>505 & x<600) & (y>310 & y <330)) opcion=3;  // Retroceder
        return(opcion);
}
void gprintf(int x, int y,char "formato,...)
{
va_list argu;
static char buffer[100];

        va_start(argu,formato);
        vsprintf(buffer,formato,argu);
        outtextxy(x,y,buffer);
        va_end(argu);
}
```

PROGRAMA: GRAFICA.C

```
#include <graphics.h>
#include <stdio.h>

/* Rutina que se ejecutara desde el programa VIDEO.C */
void grafica_2(void)
{
        /*Definimos las variables a utilizar  */
        char nombre2[13];
        FILE *ou;
struct palettetype pal;
struct palettetype gal;

int fila,columna,i,j;
float cuenta=0;

unsigned char dato;
char nombre[13];

int muestra,tecla;
unsigned int avanza;
unsigned int avanza2;
unsigned int[3,m;
int col;

/* Inicio de la rutina  */
```

```
cleardevice();
printf("\nIngrese el nombre del archivo (*.DAT)a graficar en pantalla\n");
printf("\nARCHIVO ==> ");
scanf("%s",&&nombre);
printf("\nIngrese el nombre del archivo No1 (*.MEM) a almacenar en
pantalla\n");
printf('\nARCHIVO==> ");
scanf("%s",&nombre2);
printf("\nIngrese el numero de la muestra con que comienza:");
scanf("%d",&avanza2);
printf("\ningrese el numero de la paleta de color:");
scanf("%d",&col);
cleardevice();

getpalette(&gal);
getpalette(&pal);

for (i=0; i<pal.size; i++)
        setrgbpalette(pal.colors[i], i*4, i*4, i*4);

avanza=avanza2;
cuenta=0;
ou=fopen(nombre,"rb");
avanza=avanza+48145;
for(I3=0;I3<=avanza;I3++)
                fscan(ou,"%c, &dato);
columna=1;
fila=1;
while(!feof(ou)){
                        fscanf(ou,"%c", &dato);
                        muestra=dato-60;
                        i=16-(muestra/16);
                        putpixel(columna,fila,i);
                        columna +=1;
                        if (columna==704){
                                for(m=0;m<=206;m++)
                                        fscanf(ou,"%c", &dato);
                                columna=1;
                                fila +=2;
                        }
                        cuenta++;
                        if (cuenta==185495)
                                fila=2;
        }
        fclose(ou);
        setcolor(13);
        gprintf(250,400,"Final del Archivo");
```

```
        ver_mouse();

        setallpalete(&gal);
        closegraph();
}
```

PROGRAMA: MOUSE,C

```
#include<dos.h>

union REGS regs;

int leer_tecla_mouse()
        {
        int tecla;
        regs.x.ax = 3;
        int86(0x33, &regs, &regs);
        tecla = (regs.x.bx);
        return (tecla);
        }

int leer  x  mouse()
        {
        int xm;
        regs.x.ax =3;
        int86(0x33, &regs, &regs);
        xm = (regs.x.cx);
        return (xm);
        }

int leer_y_mouse()
        {
        int ym;
        regs.x.ax = 3;
        int86(0x33, &regs, *regs);
        ym = (regs.x.dx);
        return (ym);
        }

void no_ver_mouse(void)
        {
        regs.x.ax = 02;
        int86(0x33,&regs, &regs);
        }
        void ver  mouse(void)
        {
```

```
                regs.x.ax = 01;
                int86(0x33, &regs, &regs);
}
void limita_mouse(int x, int y, int z, int w)
{
        regs.x.ax = 7;
        regs.x.cx = x;
        regs.x.dx = y;
        int86(0x33, &regs, &regs);
        regs.x.ax = 8;
        regs.x.cx = z;
        regs.x.dx = w;
        int86(0x33, &regs, &regs);
}
void stop_mouse(void)
{
        int boton;
        for (;;){
        regs.x.ax = 6;
        regs.x.bx = 0;
        int86(0x33, &regs, &regs);
        boton = (regs.x.ax);
        if (boton == 0x0)
                break;}
}
```

CONCLUSIONES

1. El proyecto desarrollado en el presente tema de tesis, muestra que el procesamiento digital de señales de imágenes es un campo de aplicaciones muy importantes en nuestra sociedad actual, el mismo que puede perfeccionarse en base a la ingeniería desarrollada en nuestro medio.
2. Los sistemas existentes en otros países son sumamente costosos el cual limita la mayor aplicación en nuestro medio, aun cuando es imprescindible su aplicación.
3. El aporte del presente proyecto es para desarrollar éstos sistemas con una eficiencia y confiabilidad elevados y a menor costo, porque están desarrollados con dispositivos y componentes existentes en el mercado.
4. La aplicación en que se orienta el presente tema es para la identificación de personas por características faciales, el cual se hace cada vez mas necesario en nuestro medio debido a situaciones de seguridad, crecimiento poblacional y otros factores incidentes.
5. El desarrollo del presente tema no se restringe a la identificación de personas; se orientar asimismo en aplicaciones industriales para el reconocimiento de objetos según características geométricas del mismo.

6. El desarrollo del presente proyecto es un aporte a la aplicación del procesamiento digital de señales de imagen y reconocimiento de personas por características faciales, el cual podrá ser mejorado con nuevas tecnologías para obtener cada vez resultados óptimos..
7. Las pruebas realizadas en el prototipo muestran la eficiencia considerable obtenidas en la implementación del proyecto.
8. El costo de desarrollo e implementación del proyecto es relativamente mucho menor de otros sistemas ya existentes desarrollados con tecnología extranjeras.

BIBLIOGRAFIA

1. Digital Image Processing.
   Rafael, Gonzales y Paul Wintz.
2. Digital Image Processing.
   Kenneth R. Casrieman.
3. Digital Image Processing Principies and Applications.
   Gregory a. Baxes-Jhon Wiley.
4. The image Processing Handbook.
   Jhon C. Russ.
5. Television Practica. Fundamentos y Reparacion.
   Bernard Grob.
6. Programming for Graphics Files in C and C++.
   Jhon Levine.
7. Borland c/c++ Manual de referencia.
   Herbert Schild.