UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO Y DESARROLLO DE UN MEDIO AMBIENTE GRAFICO PARA EL PROCESAMIENTO DIGITAL DE IMAGENES

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

GIL TAPIA GUERRERO

Dir. Dr. Rogelio Alcántara Silva

MEXICO, D. F. 1991

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Contenido

CAPITULO I

1. **Introduccion**.....1
   1.1. **Procesamiento de Imagenes**.....2
      1.1.1. **Representación de Imagenes y Modelado**.....4
      1.1.2. **Realce de Imágenes**.....5
      1.1.3. **Recuperacion de Imágenes**.....5
      1.1.4. **Análisis de Imágenes**.....6
      1.1.5. **Reconstruccion de Imagenes a partir de Proyecciones**.....6
      1.1.6. **Compresion de Imágenes**.....6
   1.2. **Conceptos para el Diseño de Software para Procesamiento de Imágenes**.....7
      1.2.1. **Modulos individuales para procesamiento de imagenes**.....7
      1.2.2. **Descripción de Componentes de un Sistema para procesamiento de imagenes**.....9
         1.2.2.1. **Software Ejecutivo**.....9
         1.2.2.2. **Preprocesamiento**.....9
         1.2.2.3. **Procesamiento**.....10
         1.2.2.4. **Módulos de Salida**.....10
      1.2.3. **Filosofia para el Diseño de Sistemas para procesamiento de imagenes**.....10

CAPITULO II

2. **CONCEPTOS DE SIMULACION, MEDIOS AMBIENTES DE VISUALIZACION POR PROTOTIPOS Y DIAGRAMAS DE FLUJO**.....11
   2.1. **EL PROCESO DE SIMULACION Y ANALISIS DE RESULTADOS**.....11
   2.2. **El Proceso de Analisis: El Ciclo de Visualizacion**.....14
   2.3. **Medios Ambientes para Visualización por Prototipos**.....15
   2.4. **Lenguajes de Flujo de Datos**.....17
   2.5. **Graficas Sincronas de Flujo de Datos**.....19
   2.6. **Interfas con el usuario.- Interfaces de Manipulacion directa**.....22

CAPITULO III

3. **DESARROLLO**.....24
   3.1. **Especificacion del sistema**.....24
   3.2. **Arquitectura del Sistema**.....24
      3.2.1. **Editor de Estrategias**.....26
      3.2.2. **Compilador de Estrategias generadas**.....26
      3.2.3. **Simulador/Ejecutor de Estrategias**.....26
      3.2.4. **Conjunto de rutinas para procesamiento de imagenes**.....27
      3.2.5. **Herramientas(Visualización, impresion, al-**

macenamiento 27
3.2.6. Conjunto de rutinas para acceso a estrategias y filosofia general 27
3.3. Criterios para la implantacion 28
3.4. Interfas del sistema 30
3.5. Uso del Mouse como medio de comunicacion con el sistema 32

CAPITULO IV

4. MODULOS DEL SISTEMA : DESCRIPCION Y OPERACION 37
4.1. Editor de Estrategias 37
4.1.1. Programas Monoliticos 37
4.1.2. Uso de librerias 37
4.1.3. Estructuras de Datos del sistema 39
4.1.4 Alta de un Bloque Funcional 43
4.1.5. Modificacion de Parametros del Bloque Funcional 44
4.1.6. Visualizacion de Bloques Funcionales 45
4.1.7. Borrado de un Bloque Funcional 46
4.2. Compilador para analisis de las estrategias generadas 47
4.2.1. Algoritmo de Sorting 48
4.3. Simulador/Ejecutor de estrategias 52
4.3.1. Scheduling 53
4.3.2. Administracion de Memoria 54
4.4. Conjunto de rutinas para Procesamiento de Imagenes 55
4.4.1. Descripcion de componentes con que debe de contar una rutina 56
4.5. Herramientas para la visualizacion, impresion y almacenamiento de resultados 62
4.5.1. Visualizacion de resultados 62
4.5.2. Impresion 64
4.5.3. Almacenamiento 64
4.6. Conjunto de rutinas para acceso a estrategias 66
4.6.1. Archivos con informacion para la interfas 66
4.6.2. Llamadas de Informacion 68
4.6.3. COMPILACION Y LIGADO 70

CAPITULO V

5. PROCESAMIENTO DE IMAGENES 73
5.1. Representacion de una Imagen Digital 73
5.2. Convenciones para la Representacion de una Imagen digital 74
5.3. Tecnicas de Realce aplicadas a Imagenes 75
5.3.1. Acotamiento de una Imagen 75
5.3.2. Modificacion del Histograma 77
5.3.3. Manejo del histograma 79
5.3.4. Metodos de Preprocesamiento 81

5.3.4.1. Correccion por Media local 81
5.3.4.2. Modificación por Histograma Local ... 81
5.3.4.3. Laplaciano 81
5.3.5. Estudio de filtros Digitales.- Caso Unidimensional 82
5.3.5.1. Introducción 82
5.3.5.2. Antecedentes 82
5.3.5.3. Diseño de filtros digitales 83
5.3.5.4. Clasificacion de filtros: Filtros FIR 84
5.3.5.5. Clasificacion de filtros : Filtros IIR 86
5.3.5.6. Metodos de Diseño para filtros FIR 87
5.3.5.7. Diseño por Muestreo en Frecuencia 88
5.3.5.8. Diseño por error minimo cuadratico en la frecuencia 88
5.3.5.9. Diseño de filtros con una region de transicion y 91
5.3.5.10. Uso de una banda de transicion 91
5.3.5.11. Ventanas 91
5.3.6. Técnica para transformacion de filtros unidimensionales a Bidimensionales 94
5.3.6.1. Transformacion de Huang 94
5.3.7. Estudio de filtros digitales: Caso Bidimensional 96
5.3.7.1 METODOLOGIA PARA DISEÑO DE FILTROS 96
5.3.7.2 ALGEBRA DE FILTROS 97
5.3.7.3 Implantacion de un filtro Paso-Alta 97
5.3.7.4 Implantacion de un filtro Paso Banda 99

CAPITULO VI

6. BLOQUES DE PROCESAMIENTO DE IMAGENES IMPLANTADOS EN EL SISTEMA 100
6.1. Familias de procesamiento 100
6.2. Bloques de Procesamiento 101
6.2.1. Bloque de Procesamiento : FILL 101
6.2.2. Bloque de Procesamiento : HEQU 102
6.2.3. Bloque de Procesamiento : HACC 102
6.2.4. Bloque de Procesamiento : HSPE 103
6.2.5. Bloque de Procesamiento : LAPL 103
6.2.6. Bloque de Procesamiento : MLOC 104
6.2.7. Bloque de Procesamiento : HLOC 105
6.2.8. Bloque de Procesamiento : FILT 105
6.2.9. Bloque de Procesamiento : IMPL 106
6.2.10. Bloque de Procesamiento : GANF 106
6.2.11. Bloque de Procesamiento : SUMF 107
6.2.12. Bloque de Procesamiento : LSWH 107
6.2.13. Bloque de Procesamiento : LSTH 108
6.2.14. Bloque de Procesamiento : SMTH 109
6.2.15. Bloque de Procesamiento : INIM 110

CAPITULO VII

7. EJEMPLOS DE OPERACION DEL SISTEMA 112
7.1. Introduccion 112
7.2. Interfas Hombre-maquina 112
7.3. Ejemplo 1.- Lectura de una imagen 114
7.4. Ejemplo 2.- Tecnicas Clasicas de Procesamiento 127
Ejemplo 3.- Diseño de un filtro Bidimensional Paso-Baja 137
Ejemplo 4.- Diseño de un filtro Paso-alta Bidimensional 142
Ejemplo 5.- Utilizacion de Procesamiento Clasico y filtrado bidimensional 148

CAPITULO VIII

Conclusiones 155

APENDICE

A. BIBLIOGRAFIA 158

1. Introducción

A lo largo de los ultimos años, la computación ha alcanzado una mayor importancia en nuestra sociedad debido a la inmensa cantidad de campos en que a hecho incursion. El esquema original que durante mucho tiempo fue el usado para el manejo de datos se restringia primordialmente a su uso como medios de almacenamiento masivo de información, junto a su uso en áreas operativas de las empresas(Nóminas, inventarios, etc.)

Esta situacion ha cambiado a grado tal que en la actualidad el uso de computadores no se restringe sólo a su uso como medios masivos de información o de apoyte de procesos rutinarios. A la fecha su uso se ha diversificado a departamentos tales como proyectos, diseño, etc. donde el computador se utiliza como herramienta que apoya el diseño de nuevo equipo así como a su prueba mediante simulaciones de los nuevos diseños, ahorrándose de esta manera importantes cantidades de dinero y recursos en general. A todas las técnicas relacionadas a esta área de la computacion se les conoce con las siglas de Diseño Ayudado por Computadora (CAD-Computer Aided Design).

Por otro lado es importante hacer notar que otra de las áreas en las que la computación ha empezado a ser usado de una manera importante es como medio de comunicación de información de diversos tipos tales como voz, imagenes, texto. Esta nueva rama de la computacion se le ha dado en llamar Multimedia. Esta rama comprende el manejo de cualquier tipo de información por medio de medios electronicos.El nombre generalizado de esta area es : **Computer-based multimedia communications** y se refiere a la representacion y diseminacion de información procesable por maquinas. esta información puede estar expresada en multiples medios (Voz, imagenes, texto escrito).

Ahora bien, una de las ramas que mayor desarrollo ha tenido y sigue teniendo en la actualidad es Multimedia, es la referente al manejo de información que comprende dos dimensiones como son las imagenes.

De la situación anterior, se presenta como una necesidad el diseño y desarrollo de herramientas que permitan el desarrollo de nuevos productos bajo un enfoque estructurado y haciendo uso de los recursos de computo que ha la fecha han reducido sus costos.

Es por todo esto que el propósito del presente trabajo es presentar el diseno y desarrollo de un ambiente de trabajo que sirva de herramienta para entender las tecnicas de procesamiento de imagenes de una forma sencilla y clara mediante la manipulación de bloques que representen algoritmos de procesamiento de imagenes. Una vez logrado esto primer proposito, el segundo ob-

jetivo que debe cumplir este sistema será el de utilizarlo como una herramienta de desarrollo que facilite el diseño de sistemas basados en las técnicas de procesamiento de imágenes.

Debido a que el área a que pertenece este sistema es la del procesamiento de imágenes, es importante dar una descripción de esta área, así como de las subáreas en las que se divide su estudio de tal forma que nos sirva como marco de referencia para las discusiones posteriores.

Una vez hecho lo anterior se procederá por objetos a definir las características que deberá de tener este ambiente de trabajo sistema, así como también una descripción de las técnicas existentes para implantar este esquema. En específico, área de simulación, lenguajes de flujo de datos e interfaces hombre-máquina.

1.1. Procesamiento de Imágenes

El Procesamiento de Imágenes (Image Processing) es el conjunto de algoritmos para analizar y modificar imágenes. Los algoritmos para procesamiento de imágenes son procedimientos mediante los cuales se llevan a cabo en una forma estructurada las operaciones que modificarán las imágenes originales, facilitando su análisis y la extracción de información deseada. La gente a menudo implanta tales algoritmos utilizando computadoras, las cuales son flexibles, además de que los costos de procesamiento y memoria son relativamente bajos. Los algoritmos se expresan por medio de programas de computadora que son sinónimos de estos.

Para entender claramente el procesamiento de imágenes, a continuación se da una descripción de como se lleva a cabo este proceso.

Un sistema típico para el procesamiento de imágenes, consiste de un dispositivo para adquisición de imágenes, un banco de memoria para almacenar las imágenes, una computadora que pueda accesar esta memoria, y un dispositivo que pueda desplegar el contenido de la memoria. Este sistema adquiere, procesa en forma única y despliega imágenes.

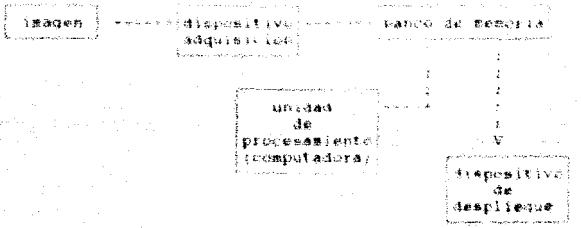
El dispositivo de adquisición pone una imagen en el banco de memoria. Esta operación envuelve la operación de digitalización de la imagen - explorar una imagen continua (tal como una fotografía y muestrla, separándola en un arreglo de valores de intensidad llamados pixeles(picture element). La mayoría de los sistemas de procesamiento de imágenes tienen por ejemplo un convertidor Analógico Digital que transforma la señal de video en un arreglo de pixeles.

El dispositivo de adquisición puede escribir al banco de memoria,

el cual a su vez puede ser leído y escrito por el CPU del sistema. Si el banco de memoria asociado con imagen completa es llamado Frame Buffer.

La computadora procesa los pixeles localizados en el banco de memoria. Una vez hecho esto por medio del dispositivo de despliegue se pueden observar los pixeles ya procesados como una imagen. El dispositivo de despliegue es usualmente un convertidor digital analógico (DAC) que dirige la señal a un monitor monocromático o de color.

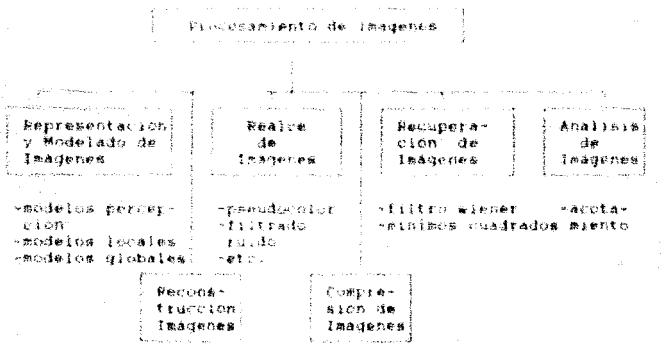
Este proceso es ilustrado gráficamente en el siguiente diagrama:



Sistema Mínimo para Procesamiento de Imágenes

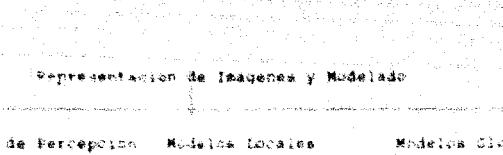
Como se puede ver en el diagrama anterior, el elemento que en un momento dado puede realizar cambios sobre la información capturada(imagen) es la computadora(unidad de procesamiento) a través de un algoritmo particular. Existe toda una familia de técnicas (algoritmos) de procesamiento de imágenes que varían en propósito. A continuación se describen.

Jain [Jain 1989] establece un conjunto de áreas en las que pueden agruparse a los algoritmos de procesamiento de imágenes: (Para mayor información sobre esta clasificación consultar [Jain 1989]).



1.1.1. Representacion de Imagenes y Modelado

La representacion de una imagen esta asociada con la caracterizacion o funcion que caracteriza a cada pixel (picture-element).En general, cualquier funcion bidimensional que contenga informacion puede ser considerada una imagen. Un modelo de una imagen da una descripcion logica o cuantitativa de las propiedades de esta tal es. A continuacion se presentan las ramas y problemas asociados a esta area:



Esta area se refiere a la representacion de modelos para imagenes asi como el establecimiento de algoritmos de procesamiento de acuerdo al modelo escogido.

1.1.2. Realce de Imagenes

En el realce de imagenes la meta es acentuar ciertos rasgos de la imagen para un subsecuente analisis o para despliegue de la imagen.Tecnicas que ejemplifican lo anterior son realce de contraste y bordes, pseudocolor, filtrado de ruido, acentuamiento/afilamiento y magnificacion. El realce de imagenes es util en la extraccion de rasgos, analisis de imagenes y despliegue de informacion visual. El proceso de realce en si mismo no incrementa el contenido de la informacion contenido en los datos. Simplemente enfatiza ciertas caracteristicas especificas de la imagen. Los algoritmos de realce son generalmente interactivos y dependientes de la aplicacion.

Las tecnicas de realce varian en cuanto a los rasgos que realzan o modifican. Por ejemplo, el metodo de ecualizacion del histograma, donde los niveles de gris de entrada son mapeados tal que los niveles de gris de salida tienen una distribucion uniforme. Otras tecnicas de realce realizan operaciones en la vecindad local. Ejemplos son la convolucion pseudocolor.

1.1.3. Recuperacion de Imagenes

La recuperacion de imagenes se refiere a la remocion o minimizacion de degradaciones conocidas sobre una imagen. Esto incluye la descorrelacion de imagenes degradadas por limitaciones en el sensor de adquisicion o por condiciones ambientales; tambien intervienen el ruido del filtrado.

La remocion fundamental en la teoria de filtrado usada corrientemente en esta area de recuperacion de imagenes es el llamado filtro inverso. Existen otras tecnicas para recuperacion de imagenes tales como minimos cuadrados, minimos cuadrados restringidos y

metodos de interpolacion por "splines".

1.1.4. Analisis de Imagenes

El analisis de imagenes esta relacionado con la realizacion de medidas cuantitativas de una imagen para producir una descripcion de esta. Las Tecnicas para analisis de imagenes requieren la extraccion de ciertos rasgos que ayuden a la identificacion del objeto. Esta area tambien es llamada por algunas personas como de clasificacion. Para clasificar elementos dentro de una imagen se necesita proveer a la computadora con conocimiento adicional sobre lo que constituye un elemento-imagen. Para simplificar la computacion de estas imagenes se procede a 'binarizar' la imagen de tal manera que los valores de pixeles abajo de un umbral son considerados cero, y a los que los sobrepasan se les asigna el maximo nivel con que se cuente (por ejemplo 255).
Estas tecnicas pueden ser muy utiles en ambientes industriales donde existan brazos mecanicos.

1.1.5. Reconstruccion de Imagenes a partir de Proyecciones

La reconstruccion de imagenes a partir de proyecciones es una clase especial de problemas de recuperacion de imagenes donde un objeto tridimensional es reconstruido a partir de proyecciones bidimensionales. Cada proyeccion es obtenida mediante la emision de un rayo penetrante a traves del objeto. Proyecciones planares son asi obtenidas mediante la visualizacion del objeto desde diferentes angulos. Los algoritmos de reconstruccion derivan una imagen a partir de rebanadas axiales del objeto.

Estas tecnicas son muy importantes dentro del area de la medicina, astronomia y exploracion geologica por nombrar algunas areas.

1.1.6. Compresion de Imagenes

La compresion de imagenes es relativa a tecnicas y metodos para la reduccion del numero de bits requerido para almacenar o transmitir imagenes sin una perdida apreciable de informacion.
La cantidad de datos asociados con informacion visual es tan grande que su almacenamiento requeriria enormes cantidades de almacenamiento. Aunque las capacidades de muchos medios de almacenamiento son grandes, sus velocidades de acceso son inversamente proporcionales a su capacidad.

Debido a su amplio uso, la compresion de datos es de gran importancia en el procesamiento digital de imagenes.

Como se ha podido ver existen diferentes aplicaciones del procesamiento de imagenes, asi como diferentes tecnicas con diferente proposito. Ahora bien, la meta del sistema a plantearse consiste en desarrollar un sistema que permita el analisis de diferentes tecnicas de procesamiento de imagenes dandole una estructura tal que permita en un momento posterior agregar sin un gran esfuerzo nuevos bloques de procesamiento.

Las tecnicas de procesamiento a realizar en este trabajo son de las areas de realce de imagenes y analisis de imagenes.

1.2. Conceptos para el Diseno de Software para Procesamiento de Imagenes

A continuacion se describen diferentes conceptos y la filosofia bajo la cual debe ser disenado un sistema de software para procesamiento de imagenes. Los conceptos aqui expuestos son tomados de [Green [illegible]], pienso que constituyen un buen punto de partida para el trabajo posterior.

Existen diferentes categorias de programas de las que se constituye un sistema. A continuacion se explican estas categorias.

1.2.1. Modulos individuales para procesamiento de imagenes

Estos modulos ejecutan las operaciones matematicas sobre las imagenes digitales, por la implantacion de diversos algoritmos para el procesamiento de imagenes (como los explicados anteriormente). Las funciones tipicas de estos modulos (Image Processing Modules- IPM) incluyen la manipulacion de contraste, filtrado y transformacion geometrica. Estos modulos operan dentro de un sistema de computo particular y son soportados por las capacidades provistas por el sistema operativo residente. Estos modulos a su vez pueden ser divididos como sigue:

Subrutinas de Proposito general.- Un conjunto de subrutinas puede soportar una variedad de operaciones requeridas por la mayoria de los modulos de procesamiento de imagenes. Estas funciones de soporte pueden incluir la entrada-salida, transferencia de solicitudes del usuario a IPMs especificos, y paquetes de despliegue que provean una interfaz entre un modulo de procesamiento y el hardware display. Estas subrutinas son usadas por modulos de procesamiento individual conforme son requeridas.

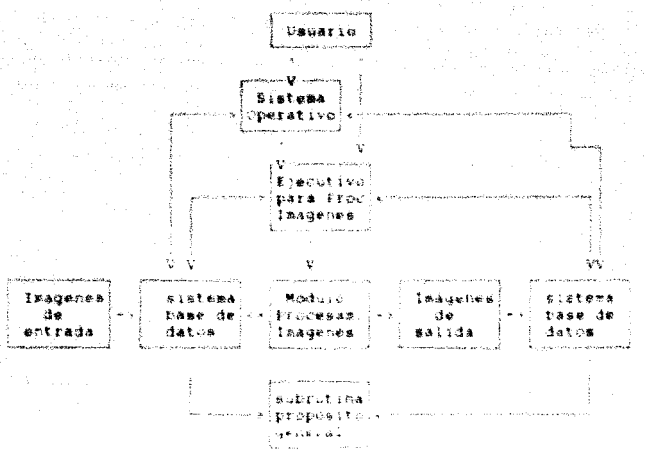
Ejecutivo para Procesamiento de Imagenes - (Image Processing Executive) Usualmente los modulos de procesamiento de imagenes (IPMs) son operados directamente a nivel del sistema operativo de la computadora. El usuario debe entender el

sistema operativo y debe aprender el lenguaje de tareas del sistema operativo para ejecutar el procesamiento de imágenes deseado. Muchos usuarios que desean ejecutar procesamiento de imágenes y análisis de imágenes puede que no estén familiarizados con el sistema operativo o puede que no tengan experiencia de ningún tipo con computadoras. Es a menudo deseable crear un programa ejecutivo que soporte la interacción del usuario con el sistema de procesamiento de imágenes. El ejecutivo puede ser escrito de tal forma que sea adaptable con el usuario. La comunicación con el usuario puede ser a un alto nivel, habilitando al usuario a describir funcionalmente las operaciones de procesamiento de imágenes que quieren ser ejecutadas. El ejecutivo entonces traslada los requerimientos del usuario en un conjunto de comandos y órdenes para el control del tiempo y la asignación de memoria. De esta forma el usuario no requiere aprender los complejos detalles de funciones del sistema operativo y el ejecutivo traslada las solicitudes del usuario en órdenes al sistema operativo.

Sistema de Administración de Base de Datos de Imágenes. - Este sistema provee el mecanismo para catalogar, almacenar y recuperar el conjunto de imágenes residentes en el sistema.

Sistema Operativo. - El sistema operativo incluye las funciones para control básico y administración de archivos de datos. El sistema de procesamiento de imágenes descrito aquí en una computadora dada será implantado y operado dentro de restricciones establecidas.

Los diferentes categorías arriba mencionadas interactuan entre ellas de una manera compleja para lograr el procesamiento de imágenes. El siguiente diagrama intenta mostrar la interfaz que puede existir al ejecutarse el procesamiento en base a un requerimiento del usuario. En sistemas de software altamente exitosos el usuario no está consciente de las complejas interacciones que ocurren entre los diferentes componentes del sistema.



Interfaces de un sistema de Procesamiento de Imagenes

1.2.2. Descripcion de Componentes de un Sistema para Procesamiento de Imagenes

Esta seccion describe los tipos de modulos de que consiste un sistema de procesamiento de imagenes. La intencion es agrupar software para procesamiento de imagenes dentro de categorias y proveer un ejemplo de los tipos de software que pueden existir en un sistema.

1.2.2.1. Software Ejecutivo

El software ejecutivo controla la ejecucion de los trabajos de procesamiento de imagenes, maneja el almacenamiento de datos y administra las funciones durante la ejecucion de una tarea. Este puede consistir del sistema operativo de la computadora o puede ser incluido como un software diseñado especificamente para la interaccion del usuario con el sistema.

1.2.2.2. Preprocesamiento

Existen dos categorias de software de preprocesamiento. La primera categoria es el software de adquisición que convierten

imagenes en su forma original al formato estandar usado por el sistema. La tercera categoria es el software de rectificacion y recalibracion que se aplica a imagenes que han sido adquiridas de fuentes especificas tales como satelite, en este caso se deben de remover distorciones y efectos introducidos por el sensor. Ambas clases de software son diseñadas para preparar imagenes digitales para procesamiento.

1.2.2.3. Procesamiento

Los modulos de procesamiento son la representacion de algoritmos de procesamiento que ejecutan el procesamiento y manipulacion de la imagen.Dentro de las familias de modulos de procesamiento que existen se encuentran: modulos para realce, modulos para transformacion geometrica como por ejemplo proyecciones, modulos para procesamiento radiometrico, modulos para procesamiento en el dominio de la frecuencia y modulos para la extraccion de informacion.

1.2.2.4. Modulos de Salida

Este tipo de modulos se utilizan cuando se cuenta con mas de un dispositivo de despliegue y o se cuenta con un sistema de grabado de tal forma que un despliegue pueda ser las imagenes generadas

1.2.3. Filosofia para el Diseño de Sistemas para Procesamiento de Imagenes

Los sistemas de procesamiento de imagenes pueden variar de un sistema minimo a un sistema altamente complejo. Un sistema minimo utiliza las capacidades del sistema operativo para la interfaz con el usuario, procesa imagenes adquiridas y provee la salida de los resultados. Un sistema complejo posee la capacidad de almacenar imagenes y modulos de procesamiento, el control es manejado a traves de un programa ejecutivo capaz de soportar multiples usuarios. Las rejas basicas de diseño son : modularidad del sistema, un formato estandar para imagenes, incorporacion de opciones manejadas por parametros dentro de los modulos de aplicacion y el uso extensivo de subrutinas de uso general deben gobernar el diseño del sistema. Cuando estas reglas basicas son seguidas, es facil expander o modificar el software del sistema.

2. CONCEPTOS DE SIMULACION, MEDIOS AMBIENTES DE VISUALIZACION POR PROTOTIPOS Y DIAGRAMAS DE FLUJO

2.1. EL PROCESO DE SIMULACION Y ANALISIS DE RESULTADOS

Como ha continuacion se explica, el proceso de generar un prototipo, simularlo y analizar sus resultados es un ciclo iterativo que se repite tantas veces como se necesario hasta lograr un modelo que satisfaga las necesidades del usuario.

Los siguientes parrafos vienen a mostrar como se genero el ciclo anterior y como es [illegible] por medio de una tecnica de creciente popularidad y en la que se basa el presente sistema. Esta tecnica recibe el nombre de **Visualizacion Cientifica**.

Este inciso y los dos siguientes pretenden dar fundamento al uso de conceptos tales como la visualizacion cientifica y los medios ambientes de visualizacion por prototipos. Con [illegible] conocimiento de estos conceptos extiende a nuevos horizontes el uso que se puede dar a las computadoras y a la computacion en general.

Debido a la importancia que encierran estos conceptos, se ha optado por presentarlos tal y como son definidos en un articulo de Upson [Upson 1990] y [Chang 1992].

En años recientes, investigadores y cientificos relacionados al area de la computacion han sido beneficiados del desarrollo y uso de la Visualizacion Cientifica: la aplicacion de graficas de computadora y tecnologias de procesamiento de imagenes para el estudio de procesos mecanicos y naturales. Para estudiar tales procesos, los investigadores han tradicionalmente generado simulaciones numericas complejas que producen tremendas cantidades de datos. Debido a la importante labor que estas simulaciones desempeñan, estas han requerido super computadoras muy cercanas a las supercomputadoras para su simulacion. Previamente, los datos obtenidos de estas simulaciones eran representados por copiosas cantidades de numeros en papel. Esta situacion ha cambiado grandemente, debido a la disponibilidad de maquinas mas poderosas y menos costosas, y al desarrollo de la visualizacion cientifica, mediante la cual se han mejorado los medios de interpretacion de los resultados de simulaciones a traves del uso de despliegues graficos.

El proceso de Visualizacion. En el curso de la simulacion de un proceso natural, un ingeniero debe avanzar a lo largo de distintos pasos. Estos pasos pueden ser divididos en tres categorias, caracterizadas por la funcion del cientifico o ingeniero realizador: el teorico, el experimentalista y el cientifico en computacion. Hasta hace poco tiempo que esta distincion de roles ha sido reconocida como el metodo de trabajo fundamental o basico en la ciencia moderna. Hace solo 20 años, teoricos y experimen-

talistas eran consideradas las únicas elementos esenciales del método científico; ellos han sido complementados en la actualidad por su contraparte en el área computacional.

Los primeros pasos son tomados por el experimentalista. El observa un fenómeno para desarrollar un modelo del fenómeno bajo estudio. El experimentalista debe correlacionar numerosos conjuntos de datos para formar una hipótesis. La correlación de estos datos sobre varios eventos brinda al experimentalista una visión del mecanismo físico que se está desarrollando.

El modelo formal relaciona todas la variables como se muestra en la figura. Este modelo es tomado por el teórico, quien debe derivar un conjunto de ecuaciones matemáticas de él. El teórico trata de representar los volúmenes de datos obtenidos por el experimentalista y en el proceso, crear iteraciones progresivas de conjuntos de ecuaciones y parámetros. En este punto, asumiendo que el experimentalista y el teórico son personas distintas, las ecuaciones obtenidas por el experimentalista, quien intenta obtener nuevos datos críticos para la verificación de la formulación matemática. Este proceso de toma y daca ocurre varias veces antes que un modelo adecuado sea desarrollado.

Las relaciones matemáticas que al final resultan, no es posible resolverlas usualmente en forma directa y deben de ser aproximadas.El trasladar estos modelos exactos, a una forma numérica está dentro de los dominios del científico en computación. Este proceso involucra 3 aproximaciones distintas, el primero de los cuales envuelve la derivación de ecuaciones parciales ordinarias del conjunto de ecuaciones original. Estas ecuaciones pueden ser resueltas numéricamente, dado un adecuado dominio físico.

El dominio contiene tres componentes: una representación discreta en espacio y tiempo; condiciones iniciales, definidas en cada localidad en el dominio para el cual las ecuaciones son validas; y condiciones de límite que detallan como este dominio se interrelaciona con el resto del mundo. El dominio discreto o computacional generalmente consiste de una malla de nodos en un espacio-N con conectividad simple- esto es, una matriz de n-dimensiones.

No todos los problemas son lo suficientemente simples para ser aproximados por esta representación: en tales casos, muchas de estas mallas deben de ser conectadas juntas, cada cual con sus propias áreas de refinamiento local, o un mapeo separado debe ser hecho de este espacio computacional simple al espacio físico.Este mapeo relaciona el índice computacional de cada nodo a sus coordenadas físicas verdaderas. En general,este mapeo del espacio computacional al espacio físico es muy complicado, y puede requerir la inversión mayor de tiempo por parte del científico.

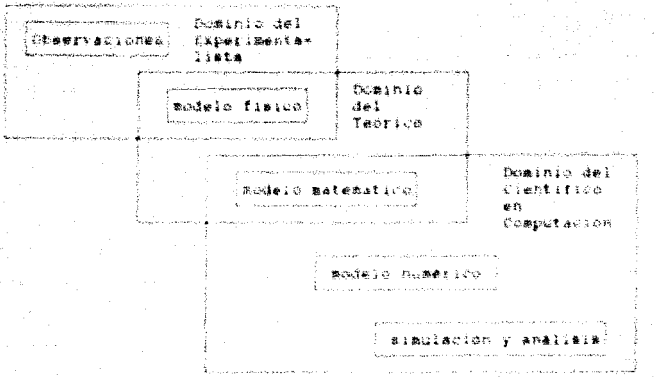
Un valor inicial para cada variable en cada nodo es requerido a fin de inicializar las ecuaciones numericas. En adicion, debido a que el dominio discreto solo aproxima una pequeña porcion del mundo real, la diferenciacion entre lo que es modelado y lo que no, juega un papel en la exactitud de la simulacion numerica. Estas condiciones de frontera permanecen validas sobre la secuencia de tiempo en la que se lleva a cabo la simulacion. Estas tres componentes (la malla discreta, las condiciones iniciales y las condiciones de frontera) forman el escenario de la simulacion.

En la segunda parte de la aproximacion, una solucion es [illegible] e iterada a lo largo del tiempo, con cada solucion obtenida usada como punto inicial para la proxima solucion. La salida de la simulacion es entonces colectada para un analisis posterior. Un posible paso para este analisis es que la solucion obtenida pueda ser inexacta, en cuyo caso una nueva malla es creada o la anterior es refinada en regiones criticas. Inestabilidades numericas podrian ser reveladas, en cuyo caso la aproximacion numerica es reformulada; esto podria ser tan sencillo como reducir el tiempo de paso o tan complicado como cambiar el metodo.

Un tercer posible resultado negativo es la determinacion de que aunque las ecuaciones han sido simuladas correctamente, la solucion no representa el comportamiento del fenomeno bajo estudio. En este punto el teorico y el experimentalista tienen que reexaminar la validez de la representacion matematica y del modelo fisico.

Claro que el analisis podria llevar a la confirmacion del proceso entero; esto es, la simulacion refleja lo que se observa en la naturaleza. El cientifico puede estudiar los resultados a un mayor detalle para obtener un mejor entendimiento del sistema en general. Esto es logrado por medio del estudio de numerosos parametros para determinar la sensibilidad del sistema a perturbaciones

En general, ninguno de estos resultados es probable; usualmente la cantidad de datos generada es tal que el entendimiento de todos ellos es muy dificil. La simulacion de un sistema tridimensional variable con el tiempo puede generar tantos datos que el cientifico queda empantanado. Es aqui donde las tecnicas de visualizacion pueden ser aplicadas con gran ventaja.



El proceso numérico de simulación

2.3. El Proceso de Análisis: El Ciclo de Visualización

El proceso de análisis de resultados de una simulación puede ser dividido en varias fases. En cada fase el científico intenta reducir o transformar los datos obtenidos en un conjunto más relevante de datos o aplicar técnicas de representación visual.

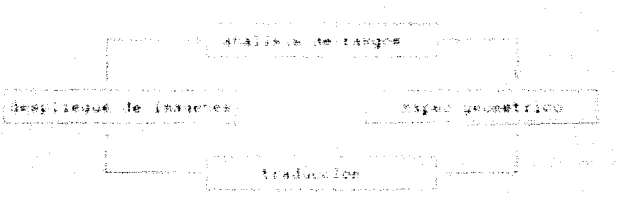
El primer paso en este proceso es el análisis o extracción de campos desde figura 1. El objeto es reducir el número de datos a un tamaño más comprensible. Esta clase de operaciones también incluye la derivación de otras variables físicas a partir de las variables primitivas. Así, un analista del clima por ejemplo un científico podría estar interesado en analizar el campo de presión y derivar el gradiente, el cual tiene implicaciones particulares en la simulación de sistemas climáticos. Operaciones de procesamiento de imágenes frecuentes y transformaciones geométricas son también componentes de esta fase, así de rangos.

El segundo paso en el proceso de análisis es la transformación de este conjunto reducido de datos en primitivas geométricas. Un conjunto dado de datos puede ser transformado en un arreglo de primitivas geométricas. Por ejemplo, un campo escalar tridimensional puede ser visualizado usando desde primitivas de dimensión cero, contornos lineales unidimensionales, polígonos bidimensionales o celdas volumétricas tridimensionales. Cada forma de representación revela diferentes aspectos de la información con-

tenida.

Durante el siguiente paso que es el de traducción, las primitivas geométricas son convertidas en imágenes. Aquí atributos tales como composición del cuadro, colores, transparencia y textura son determinados.

El último paso es el despliegue actual de la imagen y su interacción en el medio de trabajo del científico.



Flujo de Control del Ciclo de Visualización

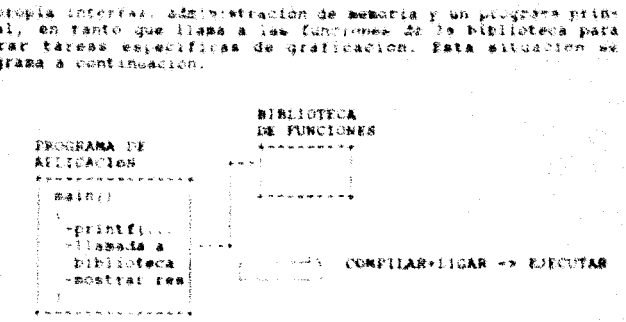
Una vez que se ha analizado el problema al que se enfrenta una persona cuando quiere aislar un problema en específico de todos los pasos y componentes que se requieren para visualizar, surge la necesidad de contar con algún medio que permita al usuario dedicarse al problema que tiene en específico dentro de que este medio ambiente debe de ser lo suficientemente flexible para que pueda servirle. Una respuesta al presente problema es usar medios ambientes que incluyan el Medio Ambiente de Visualización por Prototipos llamado **Visualization-Prototyping Environments** [ ]. A continuación se explica cual es la función de este tipo de medios ambientes así como las características que debe de cumplir para que el científico pueda trabajar [ ].

2.3. Medios Ambientes para Visualización por Prototipos

La meta de este tipo de medios ambientes es proveer flexibilidad a formas de trabajo ya existentes.

La forma en que se generan aplicaciones es nueva forma de trabajo. Típicamente el usuario se enfrenta a la necesidad de conocer y aprender el sistema gestativo de la computadora que utiliza. El usuario provee el programa principal de la aplicación y este interfaz al usuario junto con componentes para la entrada salida (véase imagen **MEDIO AMBIENTE DE TRABAJO TÍPICO**). Este enfoque difiere de medios ambientes para programación por visualización y gráficos en que cada nuevo programa debe proveer

su propia interfaz, administracion de memoria y un programa principal, en tanto que llama a las funciones de la biblioteca para lograr tareas especificas de graficacion. Esta situacion se diagrama a continuacion.



MEDIO AMBIENTE DE TRABAJO TIPICO

La diferencia principal entre el enfoque tipico y el de medios ambientes por prototipos es que estos ultimos cuentan con una superestructura que permite un control y una mayor flexibilidad en cuanto al uso de las funciones.

Las caracteristicas basicas de estos sistemas son:

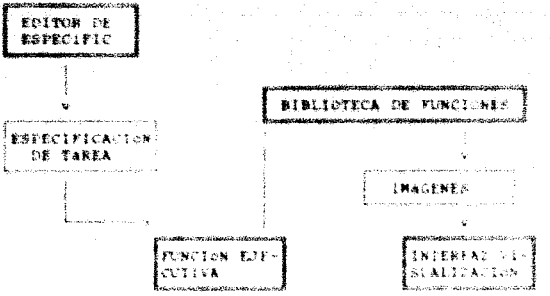
- consisten de una funcion ejecutiva o proceso que controla el orden de ejecucion de computaciones de bajo nivel

- contienen numerosos modulos funcionales que ejecutan una funcion en especifico (tal como por ejemplo extraccion de trabajos, mapeo geometrico, traduccion y despliegue de imagenes)

- estos modulos son reusables y el grado de reusabilidad es inversamente proporcional a la riqueza de su tipificacion

- el mecanismo de conexion usado para permitir la transferencia de datos entre modulos es manejado tipicamente de una manera visual

- la forma para usar este tipo de sistemas consiste de dos pasos: configurar una aplicacion mediante la construccion de una grafica directa o dirigida y la ejecucion de esa grafica



MEDIO AMBIENTE DE TRABAJO POR PROTOTIPOS

Hasta este punto se ha hecho una descripción de la situación a la que normalmente se enfrenta un conjunto de personas que intenta estudiar un fenómeno específico y a los problemas que se enfrentan para obtener la información que necesitan. Así mismo se menciona que una solución para este tipo de problemas es usar sistemas como los arriba mencionados. Estos sistemas tener de cumplir con las características señaladas para que puedan ser de uso adecuado.

En la siguiente sección se plantea que herramientas y técnicas es posible usar para implantar sistemas de este tipo.

2.4. Lenguajes de flujo de Datos

Por lenguaje de flujo de datos se entiende cualquier lenguaje basado enteramente en la noción del flujo de datos de una entidad función a otra, o cualquier lenguaje que soporte tal flujo de información. Este concepto de flujo da a los lenguajes de flujo la ventaja de permitir definiciones de programas por medio de gráficas (Davis 1982).

Estos lenguajes se basan en un modelo que tiene las siguientes características.

El modelo de flujo de datos trabaja solo con valores y no con los nombres de los bloques que los contienen, esto es, un

operador en estos lenguajes produce un valor el cual es usado por otros operadores.

El modelo de flujo de datos no tiene nada parecido a un contador de programa: una instrucción o función es habilitada si y solo si todos los valores de entrada requeridos están disponibles (han sido obtenidos). Las instrucciones habilitadas consumen los valores de entrada, ejecutan y producen conjuntos de valores de salida los cuales son mandados a otras instrucciones que necesitan esos valores. Una instrucción en lenguajes de flujo de datos no tiene efectos colaterales y una lenguaje basado en el concepto de flujos de datos no introduce restricciones de secuencialización más allá de las impuestas por las propias dependencias del algoritmo que se este ejecutando.

Un programa en un lenguaje de flujos de datos es directamente transferible a una gráfica cuyos nodos representan funciones y cuyos arcos representan dependencias de datos entre funciones.

Los procesadores de flujo de datos son el medio por el que los programas de flujo de datos son ejecutados. Pueden ser implantados en hardware o en software. El procesador en si mismo es diseñado para reconocer cuales de las instrucciones en el programa están habilitadas. Estas instrucciones son despachadas para su ejecución tan pronto como los recursos estén disponibles. No existe la noción de una sola línea de ejecución. Las o mas instrucciones pueden ser ejecutadas concurrentemente. Para mayor información consultar (Agerwala [82]).

Como se dijo anteriormente, en los lenguajes de flujo de datos, un programa es dividido en piezas llamadas nodos o bloques , los cuales pueden ser ejecutados (disparados) siempre y cuando sus datos de entrada estén disponibles. Un algoritmo es tenerlo como una gráfica de flujo de datos una gráfica dirigida donde los nodos representan funciones y los arcos representan rutas de datos como se muestra en la figura.

grafica de flujo de datos

los bloques para representar modelos tambien son llamados diagramas de bloques, estos son muy utilizados en las ramas de la ingeniería. A los diagramas de bloques se les denomina gráficas

de flujo de datos de grano grande, su denominación en inglés es (LGDF-Large Grain Data Flow), y en estos, los nodos o bloques pueden ser atómicos tal como sumadores, multiplicadores, etc., o no-atómicos (de grano grande -large grain), tales como filtros digitales, moduladores, etc. El grano o granularidad del lenguaje se determina por la complejidad de las funciones que desempeña cada uno de los bloques. La programación dentro de los bloques puede ser llevada a cabo en lenguajes convencionales tales como C o pascal por ejemplo. En este punto, es importante considerar la granularidad ya que si esta es pequeña por ejemplo, los bloques funcionales pueden ser sencillos y combinados de tal forma que se tenga una representación bastante modular del sistema que esta siendo modelado. Por otro lado, si la granularidad es muy grande, entonces el sistema puede quedar definido por bloques que dan una representación del sistema pero mas general. En todo lo anterior, antes de desarrollar algún lenguaje de este tipo, es importante considerar la granularidad con que se va a contar, así como si se va a contar con bloques con diferente grado de granularidad.

Los LGDF son como ya se dijo para hacer representaciones en el área de la ingeniería, tomando por ejemplo el caso del área de procesamiento de señales y el área de simulación. Han existido diversos lenguajes que se basan en este esquema de manejo por datos. A estos lenguajes se les ha dado en llamar lenguajes de diagramas de bloques y su mayor ventaja radica en que permiten expresar de una manera mas natural al usuario la representación del sistema que quieren modelar. Algunos ejemplos de estos lenguajes son : BLODI, FAISI, BLOCKB, LOTUS. Tambien es importante notar que en la actualidad con esta misma filosofía han sido desarrollados para las áreas de control de Procesos Industriales, sistemas como : PARAGON y GENESIS.

El uso de este enfoque ha sido utilizado en áreas como procesamiento digital de señales (Lee y Messerschmitt 1987).

2.5. Gráficas Síncronas de Flujo de Datos

Al momento de trabajar con algún lenguaje en computación, usualmente no solo basta con que este defina claramente la tarea o trabajo a ser realizado, sino que tambien debe de ser capaz de ejecutar este modelo. De la exposición que se hizo anteriormente, mediante el uso de lenguajes de flujo de datos, es claro que esta característica tambien puede lograrse. Aquí, el diseñador del sistema se enfrenta con la necesidad de habilitar un procedimiento interno en el sistema que maneja las gráficas de flujo de datos, de tal forma que las organiza y ejecuta como anteriormente se mencionó al momento en que el bloque cuenta con los datos de entrada necesarios para su ejecución. Este mecanismo que permite la ejecución de cada uno de los bloques recibe el nombre de scheduling. El scheduling es una parte muy importante

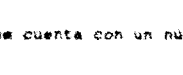
dentro de los lenguajes de flujo de datos, ya que permite la ejecucion de los bloques. El scheduling, aplicado a una grafica general de flujo de datos puede llegar a ser una funcion bastante compleja dentro del sistema, si es que el numero de entradas a los bloques, es conocido solo al momento de la ejecucion. La complejidad radica en que el programa no puede determinar cuando puede iniciar un bloque. Una posible solucion a este problema ha sido mediante el manejo de un concepto llamado umbral (threshold), este concepto consiste, en que el bloque cuenta con un valor que indica el numero minimo de entradas con que debe contar el bloque para poder iniciar. Una vez que se el bloque cuenta con las muestras minimas, entonces puede iniciar.

Una desventaja del esquema anterior, y en general de las graficas de flujo de datos, es que la seleccion del bloque y de su ejecucion (el proceso de scheduling) es durante el tiempo de ejecucion, por lo que bastantes recursos de computo pueden ser consumidos en esta labor, reduciendo el desempeño general del sistema.

Ante esta situacion, se busco algun esquema que resolviera los problemas anteriores para el sistema en desarrollo, y que sirviera de base para realizar una aplicacion mas acorde y dentro de los recursos con los que se cuenta.

A continuacion se define un subconjunto dentro de las graficas de flujo de datos, este subconjunto recibe el nombre de sincrono.

Un bloque es una funcion que es invocada cuando hay suficientes entradas disponibles para ejecutar una computación de este bloque(bloques que no cuentan con entradas definidas pueden ser llamados en cualquier instante). Cuando un bloque es invocado, este consumira un numero fijo de nuevas muestras de entrada. Estas muestras pueden permanecer en el sistema por algun tiempo, para ser usadas como muestras anteriores, pero nunca volveran a ser consideradas como muestras nuevas. Un bloque se dice que es sincrono si podemos especificar a priori el numero de muestras de entrada consumidas y el numero de muestras de salida producidas en cada salida cada vez que el bloque es invocado. Asi, un bloque sincrono se muestra a continuación.



Un nodo sincrono

Este bloque cuenta con un numero asociado a cada entrada y a cada

salida especificando el numero de entradas consumidas y el numero de salidas producidas. Estos numeros son parte de la definicion del bloque. Por ejemplo, un filtro digital tendria una entrada y una salida, y el numero de muestras consumidas o muestras de salida producidas seria uno.

Una grafica sincrona de flujo de datos

Una grafica sincrona de flujo de datos es una red de bloques sincronos como se muestra en la figura.

Como se dijo anteriormente la funcion de scheduling de bloques en linea podria ser realizada durante la ejecucion del sistema, aunque este enfoque dinamico representaria un gran consumo de recursos dentro del desempeño del sistema. Esta funcion de supervisar puede ser [illegible] y verificada, pero es costosa.

Para solucionar este problema podemos restringirnos al uso de graficas sincronas de flujo de datos. Este enfoque nos reduce considerablemente el trabajo, y no afecta los propositos generales para el desarrollo en cuestion, es decir el desarrollo de un sistema que permita facilidad y naturalidad para realizar el analisis y procesamiento de imagenes, ya que la aplicacion es muy similar al de procesamiento digital de senales

Lee y Messerschmitt (Lee y Messerschmitt 1987) proponen que con ayuda de los SDF(Synchronous Data Flow), la ejecucion y secuenciamiento de las graficas a ejecutar (scheduling), puede ser realizada estaticamente, (es decir en tiempo de compilacion), independientemente del hardware o software que se tenga, logrando ademas una reduccion en el overhead o carga asociada con el proceso de tipo dinamico. Especificamente, un compilador de granularidad grande (Large Grain Compiler) determina el orden en el cual los nodos, pueden ser ejecutados y construye una secuencia de codigo para cada procesador, en nuestro caso solo se cuenta con un procesador. La comunicacion entre los nodos es habilitada por el compilador, de tal forma que ningun control en linea es necesario, mas alla de un control secuencial tradicional.

los LGDF(Large Grain Data Flow) son una solucion atractiva para el usuario, ya que proporciona una interfaz muy natural para construir facilmente diagramas de flujo de datos que representan modelos.

A partir de aqui se define a una estrategia como una grafica de flujo de datos que representa un modelo a ser ejecutado mediante una simulación por computadora.

2.6. Interfas con el usuario.- Interfaces de Manipulacion Directa

Hasta aqui se ha planteado el uso de lenguajes de flujo de datos como una herramienta que permite al usuario un uso mas natural de la computadora.

Ahora bien, la presentacion de las opciones con que cuenta el sistema y ademas de la representacion de la información tambien es una parte muy importante dentro del desarrollo del sistema.

La construccion de sistemas es una tarea compleja, aun cuando el sistema sólo sea intensivo en la computacion desarrollada, su enfatize en el diseño de la interfaz. Incluir al usuario dentro de toda la ejecucion que se lleva a cabo, necesita que se cuente con una interfaz hombre-maquina de calidad. lo cual incrementa la dificultad en el desarrollo del sistema. Un sistema interactivo - uno que cuente con una interfaz hombre-maquina - no es juzgado solo por su habilidad para realizar los calculos y simulaciones requeridos, sino tambien por la información que proporciona al usuario. De hecho, si los usuarios no pueden comunicarse efectivamente con el sistema interactivo, toda su habilidad computacional no podrá ser accesible.

El software para desarrollar una interfaz es grande, complejo y dificil de monitorear y modificar. La interfaz de una aplicacion puede ser gran parte del codigo desarrollado. Como se menciona en [Myers 1989], encuestas realizadas sobre aplicaciones, el codigo necesario para implantar una interfaz puede ser del 40 o 50 por ciento dentro de la totalidad del codigo.

A lo large del tiempo se han desarrollado diferentes tecnicas para el desarrollo de interfaces. Conforme las interfaces se han hecho mas faciles de utilizar, ellas a su vez se hacen mas dificiles en crear.

Una tecnica de interaccion es una forma de usar un dispositivo de entrada fisico (tal como un mouse, teclado, tablet, joystick, etc.) para accesar un valor (tal como un comando, numero,localizacion o nombre por ejemplo), junto a la cual aparece una retroalimentación en la pantalla [Myers 1989].

Ejemplos de tecnicas de interaccion son los menus, scroll bars y

botones que aparecen en la pantalla cuando se hace uso de un mouse.

Existe un tipo de manipulacion que ha cobrado mucha fuerza, pues por medio de esta la interaccion del usuario con la computadora es muy sencilla. A este tipo de manipulacion se le denomina de manipulacion directa. Este tipo de manipulacion es dificil de crear, porque provee graficas elaboradas, manejo de diferentes dispositivos de entrada asincronos , un modo al que se le llama libre ; el usuario puede dar un comando en cualquier instante de tiempo ; y una rapida retroalimentacion semantica ; la respuesta apropiada a solicitudes del usuario que se basa en informacion especial de los objetos manejados).

Las interfaces de manipulacion directa son populares en muchos sistemas. Estas interfaces permiten al usuario operar directamente sobre objetos que son visibles en la pantalla, permitiendo diversas acciones a ejecutar sobre estos (alta, borrado y modificacion, asi como solicitud de informacion relativa a esos objetos).

A los objetos que se manejan en las pantallas anteriores se les conoce como ICONS. Un ICON es una representacion grafica de un objeto como representante de tal objeto , se pueden hacer todas las operaciones permitidas a ese tipo de objeto, como por ejemplo borrarlo, modificarlo, etc. Un ejemplo muy conocido de icons se encuentra en la forma como se manejan los archivos en la Macintosh de Apple. Por ejemplo, cuando se desea borrar algun archivo, se procede a seleccionar el icon que representa a ese archivo. Esta seleccion se hace por medio de un mouse. Una vez que se ha seleccionado, se arrastra hasta otro icon que simboliza un bote de basura, se encima en este al archivo y se presiona el mouse. Con la anterior accion se logra que el archivo sea borrado del sistema.

Este concepto de icon, permite un manejo mas natural de los objetos para el usuario. Se ha seleccionado el uso de icons y mouse como medios de interfaz con el sistema.

3. DESARROLLO

3.1. Especificacion del sistema

Como fue mencionado en la introduccion de este trabajo, el sistema desarrollado debe de ser una herramienta que facilite la operacion y manejo de tecnicas de procesamiento de imagenes.

Para lograr este objetivo, el sistema debe de cumplir con las siguientes funciones.

- El sistema debe de permitir el manejo de tecnicas de una manera sencilla, de preferencia orientado a su manejo por medio de diagramas de flujo (DFDS).

- Debe de contar con mecanismos de edicion de las estrategias generadas.

- Una vez que se halla definido la estrategia a realizar debe de existir una funcion que realice la validacion de esta estrategia, indicando si no llega a presentar la presencia de errores.

- Ejecucion de las estrategias proporcionadas, de tal forma que el usuario pueda ver los resultados.

- Debe contar con medios para la visualizacion de los resultados, entendiendose por visualizacion en este caso al despliegue de las imagenes resultantes asi como resultados impresos.

- Debe de existir la posibilidad de poder almacenar la estrategia para que en cualquier otro momento posterior pueda ser consultada y utilizada.

Partiendo de estos requerimientos se plantea a continuacion una arquitectura del sistema que define sus elementos.

3.2. Arquitectura del Sistema

De acuerdo a los requerimientos anteriormente planteados, se propone la siguiente arquitectura del sistema.

- Editor de estrategias (Data-Flow Diagram Editor)

- Compilador para analisis de las estrategias generadas

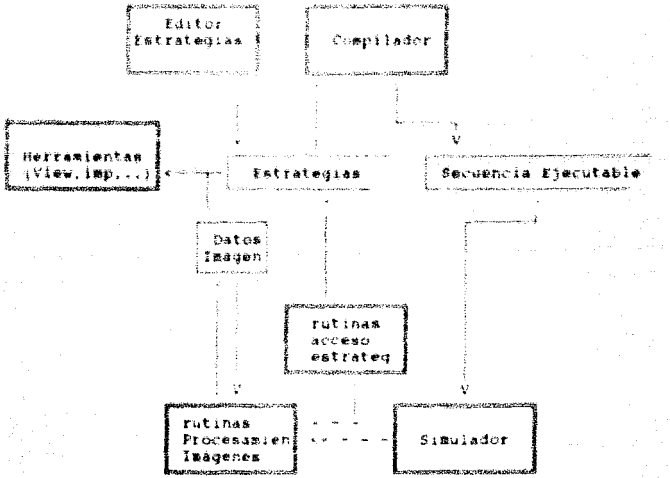
- Simulador Ejecutor de estrategias

- Conjunto de rutinas basicas que implantan tecnicas de procesamiento

- Herramientas para la visualización, impresión y almacenamiento de resultados

- Conjunto de rutinas para acceso a estrategias y estructura general del programa que permitan la inclusión posteriormente de nuevas rutinas

Los elementos constituyentes y su relación se pueden visualizar por medio del siguiente diagrama.



———> representa flujo de datos

- - -> representa relaciones de control entre modulos

representa modulos del sistema

representa base de datos

representa datos

A continuación se proporciona una descripción de cada uno de estos módulos.

3.2.1. Editor de Estrategias

Aquí se hace un uso indistinto de los términos **Estrategia** y **Diagrama de Flujo de Datos** aunque exista una diferencia entre ellos, en el sentido de que la estrategia involucra una capa de control sobre las acciones realizadas, a diferencia del diagrama de flujo de datos que es sólo la representación del sistema modelado.

Los diagramas de flujo de datos representan al sistema gráficamente mostrando el flujo de datos que existe entre los procesos. Por medio de este módulo se pueden incluir nuevos procesos, así como la eliminación de otros ya existentes en la estrategia. También se puede alimentar a cada uno de los bloques de proceso con datos necesarios para su correcta ejecución.

Además, una vez que se tienen todos los bloques necesarios, se puede proceder a establecer relaciones de operación entre ellos. También se pueden eliminar estas relaciones. Los bloques de proceso pueden ser seleccionados de familias de procesamiento ya previamente establecidas. Para realizar toda esta edición de bloques se cuenta con el teclado y con el mouse para permitir una mejor interacción y facilidad de uso para el usuario.

3.2.2. Compilador de Estrategias Generadas

Una vez que se tiene la estrategia tentativa, se cuenta con una opción que permite la compilación de ésta. El proceso de compilación se entiende como el trasladar programas escritos en un lenguaje fuente (Source Language) a un programa equivalente en un lenguaje objetivo (Target Language) ([illegible]). En este sentido cabe destacar que nuestro lenguaje fuente con el que expresamos nuestros requerimientos es por el de la propia especificación de la estrategia. El lenguaje objetivo por otro lado, es el conjunto de rutinas que implantan a los bloques de procesamiento de imágenes. El compilador tiene como otro objetivo primordial, el asegurar la correcta especificación de la estrategia, es decir, indicar la existencia de errores cuando los haya.

3.2.3. Simulador/Ejecutor de Estrategias

Al momento en que se ha validado la especificación de la estrategia planteada, se cuenta con una representación ejecutable del proceso. Esta representación es un conjunto de llamadas a rutinas cuya principal característica es que el orden en que están ordenadas es dependiente de la disponibilidad de los recursos de datos, esto es nunca se ejecutará un bloque que no cuente con

los datos completos para su operación.

Partiendo de lo anterior, la tarea del simulador es la de ejecutar los bloques que definen la estrategia, de acuerdo al orden impuesto por el compilador.

3.2.4. Conjunto de rutinas para Procesamiento de Imagenes

Este conjunto de rutinas son la implantación de diversas técnicas utilizadas en el área de procesamiento de imágenes. Para su correcta ejecución dentro del sistema, existe una interfaz de software por medio de la cual estas se pueden comunicar con el sistema, para solicitar información del entorno en que se están desarrollando, así como información de la propia estrategia; también se hacen en esta capa de llamadas que proporcionan información de la entrada y/o

Estas rutinas a las cuales nos hemos referido pueden ser ejecutadas independientemente por otras personas, con el requerimiento de insertar y extraer la información y de ser posible,

3.2.5. Herramientas(Visualisacion, impresion, almacenamiento)

El propósito de las herramientas es el de permitir acceso a la información generada. Como información no puede entenderse como la descripción propia de la estrategia así, los resultados propios de los bloques de procesamiento, también se hay que considerar la visualización de las imágenes después de un determinado procesamiento.

También existe la opción de poder grabar en disco a la estrategia que se tenga en ese momento, para poder recuperarla posteriormente.

3.2.6. Conjunto de rutinas para acceso a estrategias y filosofia general

Este conjunto de rutinas tiene como propósito lograr el fácil acceso a los recursos del sistema por los bloques de procesamiento. Ambos niveles permitirán en un momento dado una mayor independencia del sistema, liberando a la persona de tener que entender el funcionamiento interno del sistema.

Por otro lado, la filosofía general del sistema esta propuesta de tal forma que entre los modulos exista la minima interrelación posible. Las rutinas que implantan a los bloques de procesamiento no tienen nada que ver con la operación interna del sistema. El sistema ve a estos bloques de procesamiento como unidades indivisibles que son ejecutadas de acuerdo al mandato del usuario.

3.3. Criterios para la Implantación

Para el desarrollo del sistema se parte tomando en cuenta los siguientes criterios:

- El diseño del sistema tiene que ser modular respetando en lo mas que se pueda la arquitectura propuesta

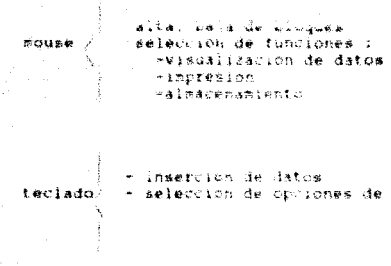
- Debe de ser relativamente sencillo el incluir nuevas rutinas para el procesamiento de imágenes, logrando de esta manera un sistema abierto

- El desarrollo se lleva en una computadora personal AT utilizando como medios de comunicación con el sistema monitor VGA teclado y mouse. Además el sistema operativo utilizado es el sistema operativo MsDos. La razón por la que se escogió esta opción es por que en la actualidad existe un gran número de este sistemas en operación con lo que no habrá problemas en la instalación del sistema. Asimismo se eligió esta opción porque existe una gran cantidad disponible de información de este tipo de máquinas. Al final en el capítulo de pruebas, les se comenta cuales fueron los resultados al utilizar este tipo de máquina.

- La implantación del sistema se hará en el lenguaje C ya que ha sido escogido este lenguaje debido a que por medio de el se puede programar a diferentes niveles, esto es tiene las características de un lenguaje de alto nivel como la definición de estructuras, estructuras de control, tales como while, switch, if, for etc. Además de que su interfaz con dispositivos de entrada salida es independiente del hardware que se este usando. El otro caso también tiene mayor a nivel de bits de datos, además de que dependiendo del paquete de lenguaje C usado, es posible utilizar rutinas de bajo nivel. En este caso específico, se hará uso del paquete Turbo-C versión 2.0 de Borland. Al final, entre las conclusiones se presentara cuales fueron los resultados al utilizar este paquete

- Como dispositivos de entrada para la interaccion con el sistema se han elegido al teclado y el mouse. El uso del mouse tiene como proposito permitir que se interactue con los bloques de ejecucion y con las opciones del sistema de una manera sencilla y más natural. A continuacion se presenta un diagrama de como los dispositivos interactuan con el sistema, asi como el tipo de informacion que es manejada

Configuracion de dispositivos en el Sistema



Funciones manejadas por cada dispositivo

En los capitulos que siguen se describira cada uno de los modulos con que cuenta el sistema.

A continuacion se describe una parte muy importante del sistema referente a la interfaz hombre-maquina del sistema. Se presenta de una manera separada de los demas modulos pues es el medio por el cual el usuario tiene comunicacion con el sistema. En la literatura se refiere a que el exito de un sistema tiene que ver mucho con la interfaz hombre-maquina generada (Hix 1987). La forma como el usuario se siente con el sistema dependera primordialmente de esta.

3.4. Interfaz del sistema

La interfaz del sistema se basa en un ambiente gráfico, basado en la manipulación de bloques funcionales, así como su interconexión para formar una estrategia.

La interfaz se realiza de la siguiente forma :

El usuario invoca al sistema desde el sistema operativo. Una vez que la presentación se ha dado, el sistema procede a presentar en la pantalla una imagen como la que se anexa.

En esta imagen se pueden apreciar cuatro áreas principales que se describen a continuación:

-área I.- En esta área a la que se denominara ACTIVA es donde se pueden definir las estrategias consistentes en bloques funcionales que pueden ser interconectados. Esta área también podría ser llamada PROGRAMACION, pues es aquí en donde se diseñan las estrategias que deben ser representadas.

-AREA II.- En esta área se encuentran todos los posibles comandos o herramientas con que cuenta el sistema. A esta área se le denominara AREA DE COMANDOS. Como se puede ver, el sistema cuenta con los siguientes comandos.

*SALIDA.- Al seleccionar esta opción el usuario sale del sistema

*FILE.- Se puede recuperar alguna estrategia ya archivada o archivar la actual para posteriormente accesarla

*IMPR.- Mediante la selección de esta opción el usuario puede solicitar que la información asociada a un bloque funcional pueda ser impresa.

*VIEW.- Mediante la selección de esta opción el usuario puede visualizar en pantalla información de cualquier bloque funcional. La diferencia de esta opción con la anterior radica primeramente en que la información no es mandada a impresora sino a pantalla; la segunda diferencia radica en que la información se presenta en forma gráfica debido al gran volumen de esta , en este caso se despliegan las imágenes, así como el procesamiento que han sufrido al pasar por los diferentes bloques de proceso, también se proporciona el histograma y la representación gráfica de los filtros bidimensionales.

*BORRADO.- Al seleccionar esta opción podemos borrar bloques de procesamiento que ya aparecen en el área ACTIVA

*CONEXION.- Por medio de este comando podemos conectar los diferentes bloques de procesamiento que hemos definido.

EJECUCION.- Al seleccionar esta opción solicitamos que el sistema valide la estrategia que hemos generado, así como su ejecución mediante una simulación. Al terminar la simulación se puede visualizar la información generada por cada bloque a través del comando VIEW, también se puede imprimir la información a través del comando PRINT.

El sistema cuenta con cuatro modos de operación:

MODO ACTIVO.- Este modo se activa cuando se selecciona un bloque de procesamiento dentro del área de BLOQUES DE PROCESAMIENTO.

MODO BORRAR.- Para borrar bloques funcionales.

MODO VIEW.- Para visualizar la información de los bloques.

MODO PRINT.- Impresión de la información del bloque en la impresora.

MODO CONEXION.- El usuario puede interconectar bloques funcionales.

Area III.- En esta área se encuentran todos los bloques de procesamiento con que cuenta el sistema. Los bloques están divididos en familias de acuerdo a área de procesamiento de imágenes de que se trate. Esta área incluye el nombre de AREA DE BLOQUES DE PROCESAMIENTO con las familias existentes pudiendo activar todos los bloques de procesamiento con que cuenta esa familia, así como hacer un cambio de la familia de bloques de procesamiento. Una vez que se selecciona el bloque de procesamiento, podemos colocarlo en la posición que mejor nos parezca dentro del AREA ACTIVA.

Area IV.- En esta área se presenta la posición de cuenta cursor, así como el modo en que se encuentra el sistema (NORMAL, ACTIVO, CONEX, VIEW, PRINT ... Esta área incluye el nombre de AREA DE STATUS.

Como se puede ver el usuario puede escoger la opción o bloque funcional que mejor le parezca. La forma en que el usuario selecciona cualquiera de estas opciones es por medio del mouse. Con este dispositivo el puede situarse a lo largo de todas las opciones y seleccionar la que desee con solo posicionarse en esa opción y oprimir el botón derecho del mouse. A partir de ese momento la opción está activa hasta que se seleccione otra opción.

Antes de pasar a esta sección cabe que vale la pena que se establecen que dentro de este grupo de comandos se puede establecer

que en realidad existen dos tipos diferentes de comandos:

* El primer grupo es el concerniente a la operación y administración de los bloques de proceso. Estos comandos lo único que hacen es seleccionar el modo en que se va a operar sobre los bloques (es decir, si se va a borrar, conectar o accionar la información que contienen). Los comandos que quedan dentro de este grupo son: BORRADO, CONEXION, VIEW e INSPECCION. A este grupo de comandos se le llamará GRUPO DE COMANDOS DE MODO.

* El segundo se refiere a la operación general del sistema. Los comandos pertenecientes a este grupo son: SALIDA, FILE y EJECUCION. Este grupo de comandos se denominará GRUPO DE COMANDOS DEL SISTEMA.

Por medio de esta interfaz es [illegible] realizar todas las tareas que requiere un usuario para definir y ejecutar cualquier estrategia.

3.5. Uso del Mouse como medio de comunicación con el sistema

Como se mencionó anteriormente el dispositivo con el que mayor simplicidad se interactúa con el sistema es el mouse. En esta sección se indicará la forma de operación del mouse así como una introducción de su programación para usarlo en sistemas. Después se definirá cómo el sistema interpreta el posicionamiento y selección de sus teclas para [illegible] comandos.

Para mayor información sobre el uso y operación del mouse se recomienda:

*Duttmann, Jerry H. [illegible] Programmer's Reference. Que Corporation 1988

*Microsoft Mouse Programmer's Reference. Microsoft Press 1989

El mouse es un dispositivo electrónico que envía señales a la computadora. Para el software que lo maneja, estas señales representan movimientos del cursor y botones que se presionan. Sin embargo, el caudal de datos enviados a la computadora es difícil de manejar en su forma original. Debido a esto, y para dar una mayor facilidad de uso para los programadores, [illegible] una interfaz consistente, Microsoft y la mayoría de los fabricantes proveen un driver para el mouse.

Un driver de un dispositivo es un software que permite al sistema operativo interpretar adecuadamente los datos enviados por el dispositivo. El driver de mouse de Microsoft hace esto proporcionando 35 funciones para que programadores de aplicación realicen tareas específicas tales como checar el estado del mouse.

El driver para mouse de Microsoft se encuentra en los archivos

MOUSE.SYS y MOUSE.COM. Debe ser instalado antes de llamar cualquier aplicacion que haga uso de el.

En este trabajo el mouse ha sido usado y manipulado dentro del sistema a traves de llamadas directas a la interrupcion 33H de MSDOS. La razon de escoger este metodo radica en que las llamadas a MSDOS por interrupcion puede ser hecho en cualquier lenguaje. Por otro lado si se quisiera trabajar por medio de bibliotecas de funciones de mouse seria necesario contar con la biblioteca de Microsoft, ademas con el lenguaje que [illegible] producido por Microsoft.

Al programar el mouse es necesario tener en cuenta ciertos aspectos particulares [illegible]:

- Como afecta el tipo de pantalla utilizada

- Como manipular el cursor

- Como puede el programa de aplicacion obtener datos acerca de las actividades del mouse

- Concepto de pantalla virtual (virtual screen)

Debido a que en el mercado existe una gran cantidad de monitores, cada uno de estos tiene caracteristicas diferentes, siendo algunos a colores y otros a blanco y negro por ejemplo. Dependiendo de las caracteristicas con que cuente cada monitor se podra utilizar el mouse en mayor o menor grado.

Los modos en los que puede operar un monitor son dos, modo texto y modo grafico.

En el modo TEXTO solo se muestran caracteres agrupados en celdas y de tamaño fijo. Monitores con 80 x 25 o 40 x 25 corresponden a este tipo de modo. Monitores de Modo TEXTO usan menos memoria y son generalmente mas rapidos que los graficos. La desventaja es que la combinacion de colores aplica a celdas enteras de pixeles.

El modo grafico tiene la ventaja de que se puede tener una mayor resolucion en las imagenes generadas. Aunque por otro lado consume mayor memoria y es mas lento.

A continuacion se describe el concepto de pantalla virtual.

Como ya se dijo existen diferentes monitores en el mercado, debido a esto el establecer un manejador de mouse para cada uno de estos seria imposible. El driver de mouse maneja el concepto de pantalla virtual para resolver este problema.

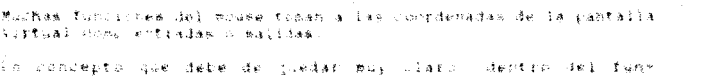
Una pantalla virtual simplifica la programacion del mouse. Una pantalla virtual puede ser conceptualizada como una malla que se

trabaja a la pantalla física. El driver traduce las coordenadas de la pantalla virtual en coordenadas físicas de la pantalla de acuerdo al modo de pantalla usado.

El tamaño de esta malla puede ser como mínimo de 640 x 200 pixeles sin importar si esta en modo texto o modo gráfico.

Muchas funciones del mouse toman a las coordenadas de la pantalla virtual como entradas o salidas.

Un concepto que debe de quedar muy claro dentro del funcionamiento del mouse es que el software que lo maneja (driver), es una tarea manejada por interrupción. Se tiene el siguiente esquema:



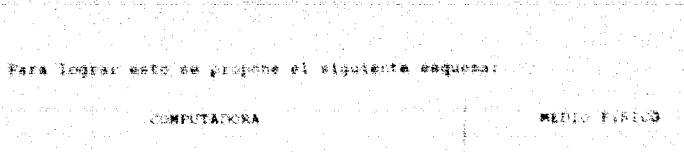
Esquema de Operacion del Mouse

Como se puede ver el driver es el que atiende al mouse cada vez que lo solicita. El driver se encarga de determinar la posición del mouse y proyectarlo en el monitor, también mantiene un estado de las características del mouse en una variable llamada INTERNAL CURSOR FLAG.

El programa de aplicación se encarga de habilitar el uso o no uso del mouse así como inicializar sus características. También desde el programa de aplicación se puede especificar si se desea trabajar en modo texto o modo gráfico. Por otro lado el programa de aplicación puede estar constantemente preguntando al driver sobre el estado del mouse. Este esquema es poco efectivo pues consumiría al programa de aplicación 3 preguntas y por lo tanto asignar recursos para mantener vista del mouse.

Un esquema alterno al anterior consiste en dejar que el driver se encargue de indicarnos cuál es la posición y el estado del mouse.

Para lograr esto se propone el siguiente esquema:



Esquema de Operacion de mouse con rutina de Atencion

Como se puede ver en este nuevo esquema se cuenta con una rutina asociada al driver que se activa cuando hay una interrupcion especifica por parte del mouse. Esta rutina lo unico que hace es almacenar el INTERNAL STAT'S MASK en variables visibles al programa de aplicacion de tal forma que la aplicacion no se tiene que preocupar de preguntar si ha habido algun cambio solo pregunta por las variables.

Para implantar el esquema anterior se usa la funcion de mouse numero 20. Esta funcion recibe el nombre de SWAP INTERRUPT SUBROUTINES y realiza la instalacion de rutinas generales por el usuario. Cuando se ejecuta, la rutina de interrupcion controlada por el driver del mouse checa la tabla de llamada una tabla intermedia de bits que corresponden a cada tipo de accion que cuenta el mouse. Si el bit correspondiente a la accion que se acaba de realizar esta activado, entonces llama a la rutina de atencion. En caso contrario el manejador de interrupcion tambien ejecuta la rutina escrita por el usuario para tal evento.

La funcion numero 21 activa dentro de la tabla de llamada y habilita la direccion donde se localiza la rutina de atencion.

Como se indico anteriormente si el driver detecta que una condicion tiene una rutina de atencion asociada entonces pasa el control a esta. Aqui es importante indicar que el driver antes de pasar el control a la rutina de atencion carga la siguiente informacion en los registros de CPU:

AX = mascara de condicion, similar a la mascara de llamada excepto que solo un bit es habilitado indicando la condicion de

error)

BX Estado de los botones

CX Coordenada horizontal del cursor

DX Coordenada vertical del cursor

ES El registro ES que contiene el segmento de datos del driver del mouse

Este funcionamiento fue aprovechado para generar una rutina de atención que ferara precisamente los contenidos de estos registros y los depositara en variables accesibles al programa de aplicación que en este caso es el programa ejecutivo.

4. MODULOS DEL SISTEMA : DESCRIPCION Y OPERACION

4.1. Editor de Estrategias

La idea original de contar con un sistema que fuera lo suficientemente flexible para aplicar bloques de procesamiento de muy diversa índole surgió como una necesidad, después de revisar varios esquemas que inicialmente cubrían esta necesidad se concluyó que el enfoque orientado a diagramas de flujo de datos era el más conveniente.

Antes de continuar, se describirán tres esquemas que fueron revisados antes de tomar la decisión.

4.1.1. Programas Monoliticos

La idea consistiría de desarrollar programas con una estructura específica de ejecución. La permanente fue el largo tiempo. Básicamente lo que se hace es ir desarrollando el programa, correrlo y obtener los resultados. Si se requiere un cambio en la lógica del programa, entonces se debe de proceder a fuente, editar al programa, recompilarlo y volverlo a correr. Este esquema es muy ineficiente si se cuenta con el requerimiento de continuos cambios al programa.

Algunas de las desventajas que tiene este esquema son :

- La persona debe de conocer algun lenguaje de programación y tener conocimientos de computación.

- Si se requieren cambios continuos, se debe de cambiar el código, recompilarlo y volverlo a correr.

- La persona debe de implantar el algoritmo propuesto en el lenguaje que este utilizando.

- Otra desventaja no muy evidente de este esquema, es que si no se mantiene un seguimiento estricto de los cambios realizados la documentación puede ser muy vaga e incompleta.

La ventaja que tiene este esquema, radica en que si la aplicación no varía muy frecuentemente, entonces los cambios se pueden hacer directamente al codigo.

4.1.2. Uso de librerias

El uso de librerías de alguna manera libera al usuario de implantar el algoritmo de procesamiento en que esta trabajando, pues el solo genera el programa que invoca a estas librerías, compila el

programa y lo ejecuta.

Desventajas

- Se sigue teniendo el problema de la secuencia fija de procesos, debiendo de entrar y modificar el codigo cuando hay algun cambio en el orden de ejecucion.

- Se sigue teniendo el problema con el manejo de la documentacion.

Ventajas

-Aqui se tiene la ventaja de que los bloques de procesamiento se encuentran en un conjunto de bibliotecas listas a ser utilizadas.

Las ventajas que posee el uso de diagramas de flujo fueron abordadas en el capitulo dos. Como se menciono, el uso de medios ambientes para visualizacion por prototipos es un nuevo enfoque para el diseño de sistemas que requieren gran flexibilidad de uso y manejo mas natural.

Bien, una vez que se han planteado las opciones existentes, podemos proseguir en la descripcion del editor de estrategias.

Como ya ha quedado constatado en los capitulos anteriores, el uso de lenguajes de flujos de datos proporciona una adecuada solucion al problema que se tiene que es el de contar con un medio que permita definir de una manera clara los modelos que deseamos sean ejecutados posteriormente.

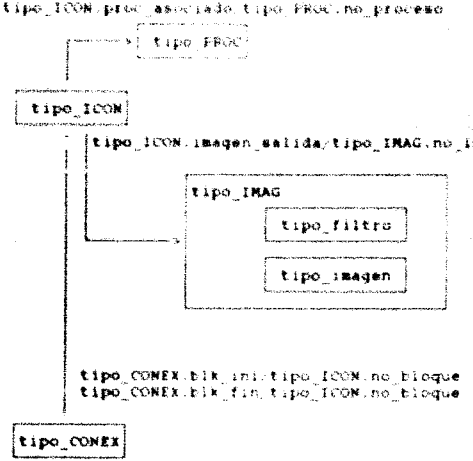
El editor de estrategias, se basa en el uso de un medio ambiente grafico por medio del cual se realiza la alta, baja y modificacion de los icons que representan bloques funcionales de operaciones sobre imagenes.

Antes que se inicie con la descripcion de las funciones del editor, es necesario que se introduzca la estructura de datos interna del sistema. Al entender esta estructura, sera mas facil entender las funciones no solo del editor, sino tambien de todos los elementos del sistema.

4.1.3. Estructuras de Datos del sistema

Gráficamente la estructura de datos del sistema se visualiza de la siguiente manera:

(Se definen las entidades que intervienen, así como la relación que existe entre ellas)



ESTRUCTURA DE DATOS DEL SISTEMA

La simbología Entidad.atributo Entidad.atributo denota los campos por los cuales se establece la relación entre las entidades

los atributos que definen a cada una de las entidades del sistema son :

(la descripcion se da en formato de lenguaje C)

```
typedef struct {
          int ocupado;              /* casillero ocupado ? */
          int no_bloque;            /* numero de bloque    */
          int imagen_salida;        /* imagen salida asociada */
          int temporal;             /* la salida es temporal? */
          int proc_asociado;        /* bloque de proceso asociado */
          localiza localizacion;    /* coordenadas en la pantalla */
          int bloque_in[5];         /* numero de bloque a entrada */
          INFO args[4];             /* argumentos asociados       */
               } tipo_ICON ;


typedef struct {
          int no_proceso;    /* numero de bloque de proceso */
          char leyenda[5];   /* nombre del proceso o funcion*/
          int (*ptr_func)(); /* apuntador a la funcion      */
          int token;
          int tipo_dato;     /* tipo dato que maneja la funcion
          int nargs;         /* numero de argumentos */
          int largs[4];      /* leyendas de argumentos */
          int tipo[4];       /* tipo de cada argumento:int,float*/
               } tipo_PROC;

typedef struct {
          int blk_in;               /* bloque fin     */
              blk_fin;              /* bloque inicio */
          natural    xini,          /* localizacion de conexion*/
                     xfin,
                     yini,
                     yfin;
          int ocupado;              /* conexion ocupada ? */
          int casilla;              /* casilla ocupada */
               } tipo_CONEX;

typedef struct {
          unsigned char huge *ptr_imagen;     /*apuntador a imagen
          float  huge *ptr_hist;              /* apuntador a histograma
          char arch_asoc[8];        /* archivo para almacenar imag
          int loc_fisica;           /* DISCO/MEMORIA para lectura */
               } tipo_imagen;
```

```
typedef struct {
          int N_D2;          /* longitud del filtro 2-dimensional */
          double far (*); MAX_RC]; /* apuntador a filtro */
          char arch_asoc 8];    * nombre de archivo asoc a est */
          int loc_fisica;       * DISCO MEMORIA para lectura */
               } tipo_filtro;

typedef union  {
          tipo_imagen imagen;
          tipo_filtro filtro;
               } imagen_filtro;

typedef struct {
          int no_imagen;
          int tipo_dato    * 0 = TIM ;  1 = TFI    */
          imagen_filtro i_f;
          int apuntando; *esta apuntando a algun area?(SI,NO) */
          int ocupado;
               } tipo_IMAG;

typedef union {
          tipo_ICON  *icono;
          tipo_CONEX *conex;
               } ptr_icon_or_conex;

struct limites
{     int xmax,
      xmin,
      ymax,
      ymin,
      leyenda[5];
};

struct stabla_proc {
          int definido;
          int no_bloque;          /* en tabla procesos */
          char leyenda[10];
            };
```

El esquema arriba especificado se interpreta de la siguiente manera:

- El programador del sistema cuenta con un conjunto de funciones ya predefinidas y que cuentan con una interfaz para la comunicacion con el sistema. Estas funciones estan en una biblioteca de funciones, pero si el programador lo desea puede generar nuevas funciones que implementen nuevos bloques funcionales. Estas rutinas debe de cumplir con una serie de requisitos para poder comunicarse con el sistema y posteriormente se proporcionan cuales son las caracteristicas que deben de cumplir. Una vez que las funciones son generadas, son dadas de alta en

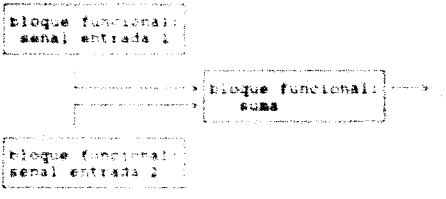
el sistema incorporandolas a la tabla tipo_PROC, para que el sistema sepa que estan disponibles, especificandose además de su presencia, atributos tales como el tipo de datos que maneja esta función, los datos que necesita para operar, así como el tipo especifico de cada uno de estos (flotante,entero,cadena, etc.)

- La entidad tipo IMAG, define los datos que estan siendo procesados por el sistema, en nuestro caso especifico, se refiere a imagenes y filtros bidimensionales. Se definen al sistema de tal forma que los maneje como elementos atomicos(es decir indivisibles). Se puede contar con 2 tipos de datos : **Tipo_imagen**, se refiere a la imagen a procesar y a todos los datos relativos a ella, tales como donde se encuentra localizada, si puede ser eliminada en caso de necesitarse espacio, etc. Tipo_filtro, contiene información referente al filtro como su longitud, frecuencia de corte, etc.
Tipo_filtro, define un ordenamiento especial de los datos pues son de tipo real, tiene atributos que no son de uso para el tipo_imagen.

- La entidad tipo_ICON es de caracter muy especial, ya que precisamente en la que define cuando el usuario a encargado un tipo especifico de bloque funcional a ser ejecutado. Dentro de la información que maneja es la referente a los datos de entrada tales como parametros que el bloque funcional necesita para ser ejecutado (Por ejemplo, si el bloque funcional es un filtro bidimensional, permite especificar la frecuencia de corte del filtro). Cada vez que se crea un registro en este tipo de entidad, se le asigna (porque se sabe por la peticion del usuario) un indice que nos indica que bloque de procesamiento esta asociado a este, además se asigna un registro dentro de la entidad tipo_IMAG, para que el bloque tenga acceso a recursos de memoria (la memoria se asigna de acuerdo al tipo de dato que se maneje-imagen o filtro). Una caracteristica muy importante con que cuenta esta entidad, es que posee campos en los que se indica cual es la relacion de este bloque con el resto de los bloques, es decir se registra cuales son los bloques funcionales origen de donde el bloque puede obtener los datos que necesita.

- La entidad tipo_CONEX especifica las relaciones que existen entre los bloques. Puede interpretarse la función de esta entidad como redundante pues esta función es ya realizada por la entidad tipo_ICON, en el sentido de que esta tiene conocimiento de los bloques funcionales que estan alimentando datos a este, pero la función principal de la entidad tipo_CONEX, es solo para mantener la operación del sistema en un ambiente grafico, ya que lleva registro de las coordenadas origen-destino pra dibujar las lineas de comunicacion de datos del sistema

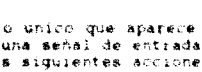
A continuación se presenta un ejemplo del uso de estas entidades para representar una estrategia, así como los procedimientos necesarios para ejecutar cada una de las funciones con que debe de contar nuestro editor de estrategias. Supóngase que se quiere representar la siguiente estrategia:



Como se ve, la estrategia consiste en sumar 2 señales de entrada. Aunque en el sistema se manejen imágenes en lugar de señales, el hacer uso de éstas últimas en este ejemplo no repercute en el sistema. Es solo para fines didácticos.

6.1.4. Alta de un Bloque Funcional

El primer paso consiste en dar de alta cada uno de los bloques funcionales. Para hacerlo supóngase que se tiene la capacidad de indicarlo mediante la selección del bloque que se desea e insertandolo en la pantalla. El primer bloque es la señal de entrada 1. El sistema queda de la siguiente manera:



Como se ve lo único que aparece en la pantalla es el bloque funcional para una señal de entrada, pero internamente el sistema ha realizado las siguientes acciones:

- Crear un registro en la entidad tipo_ICON que contiene la siguiente información:

- El proceso asociado es el de la lectura de una señal de entrada (no se sabe como trabaja esta función, solo se sabe que existe).

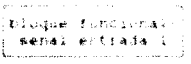
- El sistema asigna a este registro de tipo_ICON otro registro en tipo_IMAG, para que el sistema pueda asignar recursos de memoria al bloque fucnional, de tal forma que pueda almacenar sus resultados.

* Asignar información extra tal como el tipo de dato que maneja el bloque funcional, así como algunos valores de inicialización.

Como se ve, a pesar de que solo se ha definido un solo bloque, se han realizado diversas acciones.

4.1.5. Modificacion de Parametros del Bloque Funcional

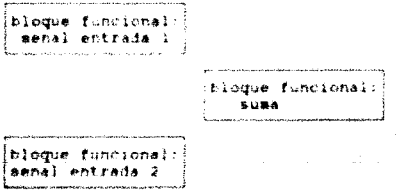
El segundo paso consiste en asignar valores a posibles parametros del bloque funcional. Por ejemplo, los datos extras consistirían en dar nombre al bloque. Supongase que de alguna manera se asignan datos al bloque posicionandose encima de el y tecleando los datos necesarios. En este caso especifico se indica que el bloque se trata de la señal 1.



Internamente los datos que se proporcionan al sistema se almacenan en los campos que han sido diseñados para tal fin, en la entidad tipo IDDM

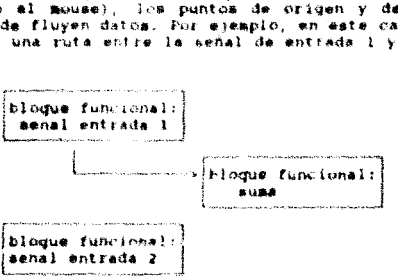
En bloques funcionales tales como filtros o procesamiento de imágenes como por ejemplo la tecnica de acotamiento, se agregarian mas datos al anterior como el umbral.

Una vez que se han definido los parametros del bloque funcional, los dos pasos anteriores pueden ser repetidos para los otros dos bloques funcionales de tal forma que el sistema quede de la siguiente forma:



6.1.6. Conexión de Bloques Funcionales

La siguiente tarea consiste en establecer la relación que existe entre los bloques funcionales de tal forma que el flujo de datos sea establecido. Se puede indicar por medio de algún dispositivo (por ejemplo el mouse), los puntos de origen y destino de una ruta por donde fluyen datos. Por ejemplo, en este caso específico se establece una ruta entre la señal de entrada 1 y el bloque de suma:

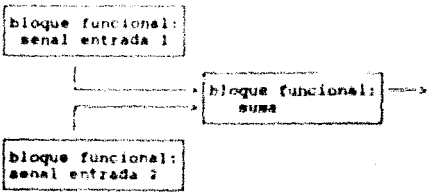


Internamente dentro del sistema,la operación realizada fue la siguiente:

+ La relación establecida es registrada dentro de la entidad tipo_CONEX, de tal forma que las coordenadas origen-destino son asignadas.

+ Por otro lado el bloque funcional que recibe la información registra la fuente de la información; en este caso el bloque de suma registra a la señal entrada 1 como su fuente. los datos del bloque funcional estan contenidos en la entidad tipo ICON.

Este mismo proceso se puede realizar para conectar a los bloques señal de entrada 2 y bloque suma:



De esta manera ha quedado definida la estrategia deseada por medio de un diagrama de flujo de datos.

Precisamente todo este manejo de estructuras es realizado por el editor de estrategias, de tal forma que las estructuras siempre deben de proyectar la estrategia correctamente.

El editor cuenta con herramientas para dar de alta, borrar o modificar el contenido de un bloque funcional ya definido, así como la conexión entre estos bloques.

Hasta este punto se han descrito tres de las funciones con que debe de contar el editor, a continuación se describe la última función.

4.1.7. Borrado de un Bloque funcional

Utilizando el mismo ejemplo, supongase que se desea eliminar la señal 2, para lograrlo se debe de utilizar la opción de borrado en el sistema. Una vez que se seleccione la opción se debe de posicionar el mouse encima del bloque que se desea borrar e indicar al sistema que este es el que queremos borrar. El bloque funcional desaparecerá junto con todas aquellas conexiones que tenga asociadas a otros bloques.

Internamente el sistema realizara las siguientes acciones :

- Buscara dentro de la entidad tipo_CONEX todos aquellos registros que contengan como elemento participante de la relación origen-destino al bloque funcional que se desea eliminar.

- Una vez que encuentre este grupo, entonces procedera a ir eliminando cada uno de estos registros, no sin antes hacer las siguientes acciones:

+ Eliminación de la relación existente en los campos de cada bloque que mantenia una relación de flujo de datos con el bloque que se desea borrar. Esta eliminación se hace en la entidad tipo_ICON.

- Eliminación del bloque funcional de la estructura de datos del sistema, eliminación de este registro de la entidad tipo_ICON

- Eliminación gráfica de las líneas que establecian las relaciones de flujo de datos.

- Eliminación del gráfico (icon) que representaba al bloque funcional.

4.2. Compilador para analisis de las estrategias generadas

Una vez que se ha definido la estrategia o modelo del sistema a simular por medio del editor el paso que sigue es el la validar que esta estrategia halla sido correctamente estructurada asi como que este completa, no faltando bloques funcionales o conexiones entre ellos.

Como se menciono en capitulos anteriores el uso de graficos de flujos de datos facilita grandemente la expresion de estrategias que involucran diversos bloques de procesamiento. Tambien se menciono que existe una subfamilia dentro de estas gráficas que tienen la caracteristica principal de conocer para cada uno de sus bloques el numero de entradas maximo con que puede contar de tal manera que el scheduling u orden de ejecucion de los procesos puede ser establecido de una forma estatica y no dinamica por medio de un compilador que genera una secuencia de los bloques de ejecucion de acuerdo a la disponibilidad con que se tenga, logrando con esto que la ejecucion sea dirigida de acuerdo a la disponibilidad de datos de entrada. Estas gráficas reciben el nombre de graficas sincronas de flujo de datos. Es precisamente sobre este tipo de graficas que se ha especificado la definicion de estrategias dentro del sistema, vale la pena recordar las caracteristicas de este tipo de gráficas:

- Son graficas dirigidas, es decir los arcos que conectan a los diversos bloques tienen un sentido en el cual fluye la información.

- Cada bloque cuenta con un numero maximo definido y conocido de entradas desde su definicion.

De todo esto se desprende como conclusion que los objetivos de este compilador son :

1-obtener una secuencia de ejecucion de la estrategia de tal forma que pueda ser ejecutada.

2-informar sobre errores encontrados.

El compilador estara compuesto por dos fases, la primera de las cuales se encargara de la correcta secuenciacion de los bloques: la segunda fase consiste en verificar que las estrategias no posean errores en su definicion.

Para lograr el primer objetivo anterior se hara uso de una tecnica conocida como **sorting**, que tiene como proposito el obtener el scheduling de ejecucion para la estrategia.

4.2.1. Algoritmo de Sorting

Dentro de la literatura se encuentran diversas descripciones de el algoritmo de sorting [Kelly 1961] [Stein 1960], pero son incompletas o poco claras. Dertouzos y Kaliski [Dertouzos 1969] proporcionan una descripcion del algoritmo que es bastante clara. Es esta la que se describe a continuacion, presentandose posteriormente el algoritmo que resulto de esta descripcion.

Existen algunas definiciones que tienen que ser establecidas primeramente.

Lista de interconexion - La lista de interconexion de un sistema S es una lista no necesariamente ordenada de 4-tuplas , , todos los elementos interconectados a . Esto se puede traducir en nuestro sistema como los bloques de procesamiento existentes asi como la lista de bloques de entrada asociados a cada bloque.

Sistemas explicitos - Un sistema es explicito si su lista de interconexion puede ser reordenada u ordenada tal que todas los bloques de entrada aparezcan antes que el elemento que los esta usando. Si esto no es posible, entonces el sistema es llamado implicito.

A continuacion se tienen que tomar como sinonimos a elementos y bloques de procesamiento.

Los sistemas explicitos son sistemas cuya lista de interconexion puede ser ordenada, ademas de que tienen la propiedad de que la lista de bloques resultante puede ser evaluada en el orden en que se genere.

Dada la definicion anterior, el sistema contempla el manejo de sistemas explicitos.

La descripcion del algoritmo se da a continuación:

Dentro del proceso de sorting, la lista de interconexion es parcialmente ordenada en todo estado en que se encuentre el proceso. Algunos elementos obtenidos de la lista no ordenada han sido ordenados para formar una lista ordenada, mientras que el remanente de los elementos tienen aun que ser ordenados.

Iniciando de un elemento no ordenado como la base de un arbol, se hacen ramificaciones hacia las entradas de ese elemento. Cada entrada que es la salida de un elemento aun no ordenado es colocada como la base de un nuevo arbol, el arbol viejo es salvado y el proceso de formacion de arboles es repetido para el nuevo arbol. Este proceso continua hasta que 1) todos los elementos en las ramas de un arbol se encuentren en la lista ordenada, en cuyo caso el elemento en la base de ese arbol puede ser

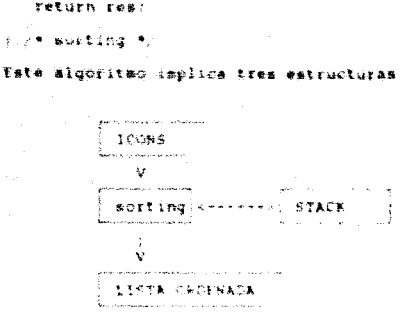
colocado en la lista ordenada; el proceso es continuado en arboles salvados previamente; 2) el proceso de formacion de arboles no puede ser continuado en algun punto porque el elemento encontrado en la rama de un arbol (por ejemplo el elemento x) es el mismo que el elemento en la base de un arbol previamente salvado. La ultima alternativa significa la existencia de un loop.

De la descripcion anterior se genera el siguiente algoritmo:

```
explota_arbol( icon_actual )
int head;
{

    if ( icon actual ya pertenece a lista ordenada ) regresa OK;

    encuentra conjunto de bloque de entrada asociados a este icon

    push( icon actual )

    para cada bloque de entrada asociado realiza:
    {
                    if ( este bloque de entrada esta en el stack (
                    )
                        return NO OK; /* LOOP IMPLICITO! error */
                    );

                    busca bloques asociados a este bloque de entrada
                    hasta que encuentres un error(loop implicito) o
                    no halla mas elementos por buscar
                    (explota_arbol(a partir de este bloque de proc))

                    retorna cual fue el ultimo estado (exito/fracaso)
    );

    pop( icon_actual )

    como este icon no contiene errores entonces agregalo a la
    lista de los elementos ya ordenados

    return OK;

} /* explota_arbol */
```

```
sorting()
{

   inicializacion de estructuras y variables

   para cada icon ocupado realiza lo siguiente :
   {

       toma_informacion del icon actual

       if ( icon_ocupado == TRUE )
                          expleta_arbol a partir de icon actual

   }

   return res;

} /* sorting */
```

Este algoritmo implica tres estructuras de datos :

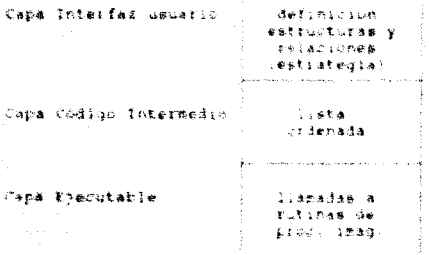


En el STACK se almacenan toda la serie de bloques de procesamiento que en algun momento participan en una secuencia encadenada. **Aqui el algoritmo verifica que ninguno de los bloques participantes en la cadena sea utilizado sin haber sido ejecutado previamente**. Para que el algoritmo lleve lo anterior se necesita que sea recursivo hasta encontrar que no hay mas bloques o que hay un error debido a un loop implicito.

Por otro lado, al verificar que el bloque en cuestion este en la lista ordenada tambien esta verificando la condicion arriba indicada, ya que todos los elementos que aparezcan en la lista ya han sido ejecutados.

El algoritmo recorre todas las ramas involucradas , definidas en la estrategia. Este algoritmo es una herramienta muy poderosa debido a su naturaleza recursiva.

Para visualizar que capas involucra el proceso anterior y su relación con las demas capas, a continuación se muestra un diagrama que explica como se desarrolla la ejecución dentro del sistema:



Secuencia para Ejecución de una estrategia a partir de su definición por diagramas de flujos

4.3. Simulador/Ejecutor de estrategias

Una vez que se ha obtenido la secuencia de ejecución por medio del módulo compilador, lo que resta es ejecutar esta secuencia con el fin de obtener los resultados de la simulación.

El módulo SIMULADOR EJECUTOR DE ESTRATEGIAS es el encargado de cubrir el objetivo anterior. Para lograrlo cumple con las siguientes funciones:

- Ejecutar cada uno de los bloques de proceso de acuerdo al orden establecido por la lista generada por el compilador

- Manejar estados anómalos generados al momento de la ejecución e informarlos al usuario abortando la ejecución.

- Asegurarse que existe el mínimo de recursos necesarios para la ejecución de bloques de proceso, de no ser así suspender la ejecución.

Los recursos que se mencionan en el punto anterior se refieren a la memoria y al CPU del sistema. Típicamente las técnicas de procesamiento de imágenes demandan grandes cantidades de memoria así como de tiempo de procesador. Para tener una idea del orden de recursos necesarios se da el siguiente ejemplo: si tenemos una imagen de 256 por 256 puntos cada punto ocupando un byte (representación entera con niveles de 0 a 255), el espacio requerido por toda la imagen es de 65536 bytes. Pero hay que tomar en cuenta que algunas técnicas de procesamiento de imágenes llegan a requerir por razones de precisión el uso de números reales. En el lenguaje C un número real puede estar representado por 4 u 8 bytes significando que algunas representaciones intermedias de la imagen pueden requerir espacios de memoria entre 262144 hasta 524288 bytes. Por otro lado los tiempo de procesamiento pueden llegar a durar hasta varias horas dependiendo de la cantidad de operaciones que implica el algoritmo de procesamiento.

Debido a las razones anteriores es necesario que este módulo sea altamente automatizado ya que en la simulación de una estrategia compleja se puede requerir el manejo de mucha memoria y el de varias horas de proceso antes de obtener un resultado.

Este módulo siempre es llamado por el comando de ejecución como paso siguiente a la compilación. Este módulo es solo llamado si el resultado de la compilación ha sido exitosa

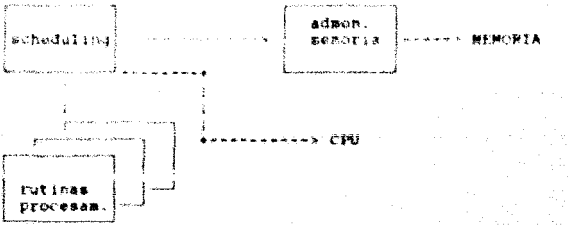
La administración de los recursos de CPU y memoria se logra a través de dos procedimientos :

- Scheduling .- Este proceso se encarga de ejecutar los bloques de procesamiento de acuerdo al orden que fue especificado por el

compilador. El proceso se realiza en forma secuencial.

- Manejo de memoria.- Son un conjunto de llamadas para la obtencion de memoria. Dentro del sistema se deben de utilizar estas llamadas debido a que el sistema siempre tendrá pista de a que procesos asigno recursos y de esta manera poder liberarlos. Estas llamadas son invocadas por las rutinas de procesamiento asi como tambien por el proceso de scheduling.

La situacion anterior se visualiza de la siguiente manera:



4.3.1. Scheduling

El proceso de scheduling se describe a continuación:

```
-para cada elemento en la lista ordenada realizar:
{
   -leer datos del icon asociado
   si existe error abortar ejecución !

   -asignar parametros para ejecución

   -ejecutar el bloque asociado
   si existe error abortar ejecucion !

   -liberar recursos utilizados por rutina
   si existe error abortar ejecucion !
```

Como se puede ver el algoritmo no es muy complicado y esto se debe a todo el trabajo anterior ya realizado. El simulador/ejecutor es solo una rutina que va ejecutando cada uno de los bloques de proceso de acuerdo al orden especificado por la lista ordenada.

Por último, también es importante apuntar que esta rutina es sencilla debido al establecimiento de la interfaz del sistema con los bloques de procesamiento. Si esta interfaz no hubiera sido establecida el esquema anterior podría haber sido más complejo.

4.3.2. Administracion de Memoria

Las rutinas de administración de memoria consisten principalmente en el registro en estructuras del sistema de los procesos que están ocupando memoria, la razón por la que se lleva un registro de estos procesos es para en un momento dado liberar la memoria asignada a ellos. La memoria puede ser liberada justo después de que termina la ejecución de cada proceso, el resultado final es almacenado en archivos.

Estas rutinas de administración son utilizadas por el sistema tanto por las rutinas de procesamiento como por el módulo de visualización de resultados.

4.4. Conjunto de rutinas para Procesamiento de Imagenes

Este modulo se compone de diversos algoritmos de procesamiento de imagenes implantados y adecuados para funcionar dentro del sistema.

Este Conjunto de rutinas se considera como un modulo, pero no quiere decir que entre ellas exista una relacion funcional las rutinas estan organizadas dentro del sistema como un conjunto de librerias de funciones. Todas estas rutinas tienen como caracteristicas principales las siguientes.

- Cada rutina es la implantacion de algun algoritmo de procesamiento de imagenes como se vio en la parte de descripcion de tecnicas de procesamiento de imagenes, estas rutinas o funciones varian grandemente en cuanto a su aplicacion y complejidad.

- Todas las rutinas para poder operar dentro del sistema tienen una interfaz ya establecida de tal manera que puedan ser invocadas por el simulador del sistema. Esta interfaz tambien sirve para enviar informacion a la rutina que se esta llamando.

- Aparte de la interfaz con que deben de contar las rutinas cualquier acceso a recursos del sistema tales como memoria, se realiza por funciones especiales definidas dentro del sistema para ese proposito. El fin de contar y utilizar con estas rutinas es el de asegurar que una vez que se termine esa rutina el sistema tenga manera de liberar recursos utilizados.

- El sistema llama a estas rutinas siempre a traves de la informacion contenida en tipo IMAGE.

El punto mas importante dentro de este modulo es el referente a la descripcion de la interfaz de comunicacion establecida de tal forma que el sistema tenga acceso a estas rutinas.

El proposito de explicar esta interfaz es la de permitir que en un momento dado se puedan desarrollar rutinas de procesamiento de imagenes e incluirlas dentro del sistema como nuevas opciones de bloques de procesamiento. En el presente capitulo solo se describiran los requerimientos y caracteristicas que deben de cumplir las rutinas desarrolladas de tal forma que puedan ser utilizadas por el sistema.

Como comentario cabe mencionar que todas las funciones que fueron incluidas en el sistema, estan descritas en la parte de Procesamiento de Imagenes de este trabajo.

4.6.1. Descripcion de componentes con que debe de contar una rutina

Toda rutina a ser llamada por el sistema debe cubrir el siguiente formato:

```
int nombre_rutina( int count, INFO argelo[] )
{

 Declaracion variables locales

 analisis de parametros
  son correctos y corresponden al numero esperado?
  si :

    -llamar a rutinas necesarias para obtener informacion
     del procesamiento actual ( segun las rutinas de acceso )

     si se obtiene algun error retornar codigo de error

     extraer datos de los llamados anteriores

    -solicitar recursos de memoria al sistema para la operacion

     si se obtiene algun error retornar codigo de error

    -invocar rutina(s) de procesamiento
     cuidar que estas rutinas reflejen cualquier error por medio
      de codigos !

    -retornar estado resultante de la ejecucion de rutinas

  no :
    -informar mediante un codigo de error que hubo un error

}
```

En el capitulo referente al procesamiento de imagenes se cubre la técnica de acotamiento.

La tecnica de Acotamiento tiene como proposito principal el detectar formas en imagenes. Tambien se utiliza para convertir una imagen multinivel en una imagen binaria.

Se tomara como ejemplo esta tecnica para mostrar como se aplica lo anteriormente descrito.

A continuación se muestra el código fuente que compone la interfaz con el sistema:

```
 ****************************
*
* programa: ejemplo de interfaz
*
* autor   : gil tapia
*
* fecha   : julio 1990
*
*

 * pueden ir otros includes */

#include "info.h"


int h_acc( int count, INFO arreglo[] )

{

tipo_IMAGEN imag;
int no_actual, no_imag;
float huge *h_ant, *h_act, *haux;
unsigned char huge *r_ant, *r_act, *raux;
int res, tipos;
int lista[10];
int num, umbral;
tipo_ICON icon;

            /* Este modulo es una entrada-una salida  */

            if ( count == 2 )
            {
             tipos      = arreglo[ 0 ].tipo;
             no_actual  = arreglo[ 0 ].valores.s_val;
             no_imag    = arreglo[ 1 ].valores.s_val;
             /* asignacion de recursos */

             res = genera_lista( ON, no_imag, no_actual );
             if( (res=info_imag_x_icon( ACTUAL, no_actual, &imag
) != TRUE ) return res;
             h_act = imag.i_f.imagen.ptr_hist;
             r_act = imag.i_f.imagen.ptr_imagen;
             if( (res=info_icon_x_icon( ACTUAL, no_actual, &icon
) != TRUE ) return res;
             umbral = icon.area.U.valores.s_val;
```

```
            /* obtencion de argumento de entrada, se sabe que
solo hay uno */

            tmem_contenido( no_actual, &num, lista );
            if ( num != 1 ) return FALSE;
            if( (res=info_icon_a_icon( ACTUAL, lista[0], &icon
))!= TRUE ) return res;
            no_imag = icon_imagen_salida;
            res = allocate_image( ON, no_imag, lista[0] );
            if( (res=info_imag_a_icon( ACTUAL, lista[0], &imag
))!= TRUE ) return res;
            h_ant = imag.i_f_imagen[ ptr_hist];
            t_ant = imag.i_f_imagen[ ptr_imagen];

            res = accion( ventana, t_ant, f_act );

            return res;
           }
           else
                return no_ok;
}  /* h_accion */
```

Al observar el código es importante indicar que el sistema siempre proporciona como los dos primeros argumentos el identificador del icon y el identificador de la imagen.

El identificador del icon es un indice dentro de la tabla de icons. Mediante este identificador podemos accesar mayor información del sistema, por medio de llamadas.

El identificador de la imagen cumple la misma función que el identificador de icon.

La función h_accion será identificada por el sistema como la implantación de la rutina de acotamiento. Esta función siempre deberá de ser de tipo entero(int) de tal manera que pueda informar el estado con que termino la rutina(código de falla exito). Por otro lado los argumentos siempre deben de ser dos como se especifica. Los argumentos son datos enviados por el sistema. El primer campo indica cuantos argumentos son enviados y el segundo son los argumentos con su tipo asociado.

Sistemas Manejadores de Bases de datos como ORACLE y UNIFY cuentan con un esquema parecido al anterior. Este esquema da libertad al sistema de proporcionar cuantos argumentos considere necesarios pero bajo una interfaz bien definida.

A continuacion se muestra la definicion del archivo info.h

```
 ********************************
*
* programa : info.h
*
* autor    : gil tapia
*
* fecha    : julio 1990
*
*

union vals {
          float  f_val;
          double d_val;
          short  s_val;
          long   l_val;
          char far *c_val;
          };

#define FLOAT  1
#define DOUBLE 2
#define SHORT  3
#define LONG   4
#define CHAR   5

#define VALS   union vals

struct tipo_info {
          int tipo;
          union vals valores;
          };

#define INFO  struct tipo_info
```

Como se puede ver INFO es una estructura consistente basicamente de dos campos. El primer campo indica el tipo de dato que se encuentra localizado en el segundo campo. El segundo campo es un union, esto quiere decir que podemos alojar cualquier tipo de dato definido en el union. Aqui se pueden tener datos tipo float, double, short long y apuntador a cadenas.

Para entender lo anterior claramente supongase que se tiene definida una estructura semejante a la anterior (pero menos compleja, solo para fines de claridad).

```
#define BYTE 0
#define INT  1

union vals {
        unsigned char c;
        short  s_val   ;
        };

struct tipo_info {
        int tipo;
        union vals valores;
        } A;
```

Supongase que A contiene la siguiente informacion:

	TIPO	VALORES
A	00	00 01

El funcionamiento del algoritmo es tomar el valor de acuerdo al tipo que se tenga. En este caso como tipo=0=BYTE entonces se obtendra en VALORES = 00.

Ahora supongase que A contiene la siguiente informacion:

	TIPO	VALORES
A	01	00 01

En este caso dado que TIPO=1=INT entonces VALORES = 1. Como se puede ver este esquema es bastante flexible cuando se desea manejar datos no solo con valores dinamicos sino tambien con tipos.

Como esta planteado en la rutina INFO arreglo[ ] es un apuntador a un arreglo cuya dimension se determina por el valor del argumento count.

Antes de pasar con otras caracteristicas es importante que se use la declaracion como se indica:

```
tipo nombre_funcion ( tipo arg1, tipo arg2 )
```

Esta declaración recibe el nombre de prototipo y es propio usarla cuando diferentes modulos hacen uso de rutinas no incluidas en su codigo fuente.

4.5. Herramientas para la visualisacion, impresion y almacenamiento de resultados

Las herramientas con las cuales se pueden visualizar imagenes procesadas o en general ver los resultados obtenidos pertenecen al menu de COMANDOS DE MENU.

Basicamente se han seleccionado tres opciones estas son:

-Visualizacion de resultados

-Impresion

-Almacenamiento

Para seleccionar a cualquiera de estas opciones basta con que el usuario posicione el mouse dentro del AREA DE COMANDOS y oprima la tecla derecha del mouse. Una vez hecho esto puede posicionarse en cualquiera de los bloques de procesamiento y al oprimir la tecla derecha aparecera un menu de opciones. El usuario puede elegir por medio del teclado la opcion que desee.

A continuacion se explican cada una de estas opciones.

4.5.1. Visualisacion de resultados

Una vez que se ha ejecutado la estrategia es logico que se cuente con la posibilidad de ver los resultados. Dado que este sistema esta orientado al area de imagenes se ha elegido la opcion de ver los resultados graficamente debido a que en la mayoria de las veces el volumen de informacion generada es muy grande. Dentro de estas opciones se ha elegido la presentacion de la siguiente informacion:

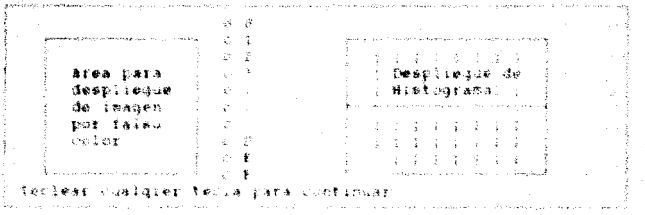
Para imagenes :

- Desplegar la imagen resultante con la niveles de intensidad mapeando a los 256 con que cuenta la imagen.

- Despliegue del respectivo histograma. Se eligio el histograma ya que es una representacion muy generalizada de las graficas.

Mediante solicitud del usuario se genera una pantalla que contiene estas dos informaciones.

la información se visualiza de la siguiente manera:

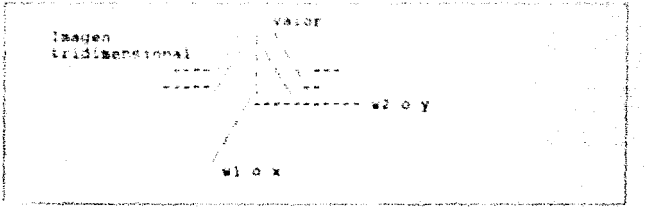


Visualización de una Imagen

Para filtros :

- Despliegue tridimensional en espacio o en frecuencia del filtro generado. Se eligieron estos despliegues ya que cuando se diseña un filtro es útil visualizar como se ve en ambos espacios.

El usuario tiene que escoger que representación desea del filtro. La presentación de pantallas es como se muestra:



Visualización de un Filtro

El algoritmo para presentar imagenes bidimensionales en tercera dimension se obtuvo de [Szabo 1988].

Por otro lado la obtencion de transformadas (representaciones en la frecuencia) a partir de representaciones espaciales bidimensionales se obtuvo de un articulo que presenta un metodo muy sencillo y eficiente. Este algoritmo se encuentra en [Bernabeu 1989].

4.5.2. Impresión

Se cuenta con la opción de obtener información impresa de tal forma que el usuario pueda usarla como referencia.

La información se da en forma de listados.

La información que puede ser impresa es :

Para imágenes:

- Descripción del bloque de procesamiento

- Parámetros del bloque de procesamiento

- Histograma de la imagen generada

Para filtros:

- Características del filtro (frecuencia de corte, longitud, orden, información adicional)

- Descripción del método de diseño

- Filtro bidimensional (coeficientes)

Para estrategia:

- Impresión de los bloques de proceso junto con sus características y parámetros, así como sus interconexiones

4.5.3. Almacenamiento

Debido a cuestiones de espacio principalmente se decidió que una vez que se terminara el procesamiento de cualquiera de los bloques con que contara una estrategia, el resultado fuera filtro o imagen se almacenaría en disco y se liberaría el área utilizada para que el siguiente proceso pudiera usarlo nuevamente.

Ahora bien, dentro de los parámetros con que cuenta cada bloque de proceso se especifica si el resultado será temporal o si permanecerá en un archivo de tal manera que después se pueda usar. Los campos que especifican lo anterior son :

```
¿ Es temporal ?

Nombre Archivo asociado:
```

la respuesta que asigna el sistema al oír el bloque es SI.

El usuario puede indicar que no es temporal y proporcionar un nombre de archivo para almacenarlo.

La estrategia también puede ser almacenada de tal forma que posteriormente pueda ser invocada. El sistema cuenta con tres opciones para manejo de estrategias y son :

-Salvar - para almacenar la estrategia. Posteriormente puede ser recobrada.

-Recobrar - se lee la estrategia de archivos almacenados en disco. La vez que se oye la ejecuta toda la estrategia en la pizarra.

-Nueva - esta opción elimina cualquier estrategia que se encuentre en ese momento en pizarra, también libera toda la memoria utilizada.

4.6. Conjunto de rutinas para acceso a estrategias y estructura general del programa que permitan la inclusion posteriormente de nuevas rutinas

Una de las metas principales que se persigue con el desarrollo de este sistema es el de poder incrementar la cantidad de bloques de procesamiento de una manera sencilla. Para lograr esto el sistema cuenta con una serie de llamadas que pueden ser usadas para obtener información tal como:

- Información acerca del icon asignado a la rutina de procesamiento

- Asignación de memoria

- Información acerca del ratón apuntado

Aparte de la anterior se cuenta con llamadas por medio de las cuales es posible incluir nuevas rutinas de procesamiento.

En el capítulo referente a **Conjunto de rutinas de Procesamiento de Imagenes**, se explica cuales son las características que debe de cumplir el código que se desee realizar. En esta sección se explicaran el conjunto de rutinas con que cuenta el sistema, de tal forma que pueda accesarse información del sistema así como recursos.

Se cuenta con dos grupos de llamadas:

- llamadas de información

- llamadas para acceso de recursos

4.6.1. Archivos con información para la interfaz

Existen tres archivos que deben ser incluidos por medio de includes en la rutina a ejecutar:

defin.h

```
/**********************
*
* programa : defin.h
*
* autor    : Gil Tapia Guerrero
*
* fecha    : Julio 1990
*/

#define OK 0
#define NO_OK 1
```

Define codigos de regreso de tal manera que sea interpretado por el sistema. (Hubo error o no lo Hubo)
En general cualquier valor diferente de cero sera interpretado como un error.

info.h

Este archivo ya fue explicado en capitulos anteriores.

icon.h

Este archivo contiene los prototipos de las funciones del sistema que pueden ser llamadas por el programador.

Una de mayores fortalezas con que cuenta C es que sus funciones son faciles de definir y de usar. La declaracion de los tipos de parametros de una funcion y el valor de retorno es llamado un **prototipo de una funcion.**

Una tarea para los prototipos de funcion es que permiten al compilador checar si se esta llamando a las funciones con el numero correcto de parametros y si estos parametros son del tipo correcto. Pero este no es su unico uso ellos ayudan tambien al compilador a ejecutar conversiones automaticas de tipo entre los parametros de las funciones.

En este caso especifico los prototipos ayudan a que el programador tenga claro los tipos que debe de intercambiar.

```
/*****************************************************
 *
 * programa : icon.h
 *
 * autor    : gil tapia
 *
 * fecha    : agosto 1990
 *
 * Conjunto de llamadas y definiciones necesarias para la comu
 * nicacion con el sistema
 *
 */


#defineMAX_PC 100/*longitud maxima para filtros 2-dimensionales*/

typedef struct {
                        unsigned char huge *ptr_image;
                        int no_col, no_ren, no_niv[65];
                        float huge *ptr_hist;
                        char arch_asoc[8];
                        char tag[8];
```

```
                    int loc_fisica; /* DISCO o MEMORIA */
                } tipo_imagen1;

typedef struct {
                    int N_D2;/*longitud filtro 2-dimensional */
                    char arch_asoc[8]; /* nombre archivo asociado
                    char tag[8]; /* nombre */
                    int loc_fisica; /* en disco o en memoria */
                    double far (*ri)[MAX_RC]; apuntador a filtro
                } tipo_filtro1;

typedef union  {
                    tipo_imagen1 imagen1;
                    tipo_filtro1 filtro1;
                } imagen_filtro1;

typedef struct {
                    int no_imagen;
                    int tipo_dato; /* de imagen,  o filtro   */
                    imagen_filtro1 i_f;
                    int apuntando;/*esta apuntando a algun área ?
(SI/NO) */
                    int ocupado; /* este espacio esta imagen ?
                    int nargs;   /* numero de argumentos
                    int largs[8];/* leyendas argumentos */
                    INFO args[8];/*
                } tipo_IMAG1;

int trae_icon( int direccion, int *no_icon );

int info_icon_x_icon(int direccion,int no_icon,tipo_ICON *icon );

int info_imag_x_icon(int direccion,int no_icon,tipo_IMAG1 *imag);

int info_proc_x_icon(int direccion,int no_icon,tipo_PROC *proc );
```

4.6.2. Llamadas de Informacion

Las llamadas de informacion son :

int trae_icon(int direccion, int *no_icon);

Proporciona el numero de icon dentro de la lista de ejecución.

ENTRADA:

direccion=
ACTUAL -se desea valores del no icon especificado
PROXIMO.-se desea valores del icon proximo a no_icon

ANTERIOR.se desea vlores del icon anterior a no_icon

SALIDA:

no_icon - Identificador del icon dentro del sistema. Este valor lo proporciona el sistema en el primer valor dentro de la lista de argumentos.

int info_icon_x_icon(int direccion,int no_icon,tipo_ICON *icon);

Mediante esta llamada se puede accesar la informacion del icon actual. Este icon esta identificado por el valor que proporciona el sistema al ejecutar el bloque.

ENTRADA:

direccion=
ACTUAL.-se desea valores del no_icon especificado
PROXIMO.-se desea valores del icon proximo a no_icon
ANTERIOR.se desea vlores del icon anterior a no_icon

Nota : el proximo y anterior se refieren a, orden con el que fueron dados de alta por medio del editor.

no_icon - Identificador del icon dentro del sistema. Este valor lo proporciona el sistema en el primer valor dentro de la lista de argumentos.

SALIDA:

icon - Es la estructura que contiene toda la informacion referente al icon solicitado.

int info_imag_x_icon(int direccion,int no_icon,tipo_IMAGI *imag);

Mediante esta llamada se puede accesar informacion de la imagen de salida asociada al icon actual. Este icon esta identificado por el valor que proporciona el sistema al ejecutar el bloque.

ENTRADA:

direccion=
ACTUAL.-se desea valores de la imagen asociada al no_icon especificado
PROXIMO.-se desea valores de la imagen asociada al proximo icon
ANTERIOR.-se desea valores de la imagen asociada al icon anterior

Este y el proximo y anterior se refieren al orden con el que fueron dados de alta por medio del editor.

no_icon - Identificador del icon dentro del sistema. Este valor lo proporciona el sistema en el primer valor dentro de la lista de argumentos.

SALIDA:

imag - Es la estructura que contiene toda la información referente a la imagen asociado al icon.

int info_proc_icon(int direccion,int no_icon,tipo_PROC *proc);

Mediante esta llamada se puede accesar información del proceso asociado al icon actual. Este icon esta identificado por el valor que proporciona el sistema al ejecutar el bloque.

ENTRADA:

direccion=
ACTUAL.-se desea valores del proceso asociado al no_icon especificado
PROXIMO.-se desea valores del proceso asociado al proximo icon
ANTERIOR.-se desea valores del proceso asociado al icon anterior

Nota : el proximo y anterior se refieren al orden con el que fueron dados de alta por medio del editor

no_icon - Identificador del icon dentro del sistema. Este valor lo proporciona el sistesa en el primer valor dentro de la lista de argumentos.

SALIDA:

proc - Es la estructura que contiene toda la información referente al proceso asociado al icon.

4.6.3. COMPILACION Y LIGADO

Una vez que se ha desarrollado la rutina, esta debe de ser compilada con el modelo LARGE de turbo-C. Es necesario que se use este modelo de memoria debido a que algunas rutinas como por ejemplo el manejo del mouse requieren que el apuntador con sus llamadas cuente con segmento y offsett, tambien el tamaño de los datos manejados (imagenes) requieren apuntadores con segmento y offsett. Existen 6 modelos de memoria. Se muestra a continuacion una tabla que muestra las diferencias entre ellos:

	small code	large code
small data	TINY SMALL	MEDIUM
large data	COMPACT	LARGE HUGE

Tabla de Modelos de Memoria existentes en Turbo-C

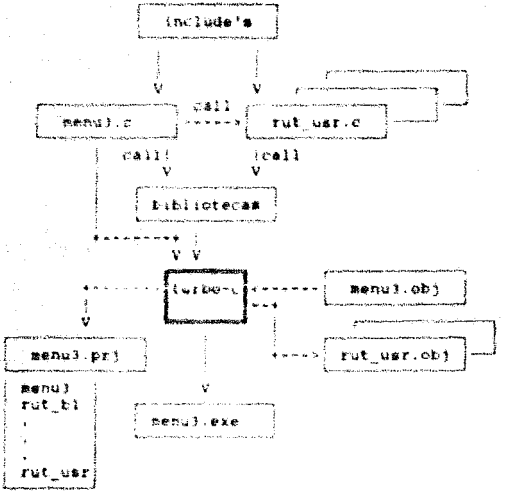
small o large se refiere al modo en que se trata el direccionamiento en memoria. Con el direccionamiento small, el programa no necesita especificar un segmento ya que todo el código y datos del programa están dentro del mismo segmento.

Para mayor información sobre modelos de memoria se recomienda "Smith (1988)" y "Hergert (1989)". Estos libros hablan sobre el manejo de diferentes tópicos referentes a Turbo-C.

Ya que se ha generado el código objeto entonces es necesario incluir el nombre de este archivo dentro del archivo MENU.PRJ. Los archivos .PRJ definen el conjunto de programas objeto o fuente que constituyen un programa. Este tipo de archivos instruyen a Turbo-C de realizar un programa ejecutable tomando en cuenta todos los archivos que se definen en el archivo .PRJ. Este tipo de archivos son de utilidad cuando la aplicación es muy grande y se requiere dividir el sistema en módulos funcionales. Para solicitar que se genere un programa ejecutable se debe de hacer uso de la opción MAKE.

Cabe destacar que debido a la facilidad de contar con estos archivos *.PRJ el programador de nuevas funciones solo debe de incluir los nombres de los nuevos modulos y pedir que se genere la version ejecutable.

El esquema de operación se presenta en el siguiente diagrama:



Proceso para generacion de programas en Turbo-C

3. PROCESAMIENTO DE IMAGENES

El propósito de este capítulo es presentar primeramente una introducción a la representación de imágenes, presentación del concepto de realce y técnicas aplicadas hacia este fin dentro del área de pre-procesamiento de imágenes.

Existen dos enfoques desde los cuales han sido desarrolladas un gran número de técnicas de realce.

El primer enfoque cubre a técnicas que surgieron como resultado de la práctica en el procesamiento de imágenes. Estas técnicas tienen la característica que realizan a aventuras ciegas específicas, se les ha dado en llamar técnicas clásicas o Ad-Hoc. Ejemplos de estas técnicas son el manejo del histograma y el laplaciano.

El segundo grupo se fundamenta en el área de procesamiento digital de señales extendido a dos dimensiones. Esto nos permitirá el diseño de filtros bidimensionales y considera a las imágenes propensas a ser filtradas por medio de los filtros obtenidos de tal manera que pueden existir filtros pasa-bajas, pasa-altas, etc. que pueden ser aplicados a las imágenes (señales en dos dimensiones).

Como se verá más tarde todas estas técnicas pueden ser combinadas dentro del sistema para formular estrategias de solución a problemas de realce.

Las fuentes consultadas para el desarrollo de este capítulo y en general para la implantación de estas técnicas son [Ang 1982], [Gonzalez 1986], [Pavlon 1982], [Parks 1987], [Jain 1989] y [Rummel 1977].

3.1. Representacion de una Imagen Digital

Como fue expuesto en la introducción, una vez que una imagen ha sido adquirida se almacena en un bloque de memoria. Su representación interna usualmente es por medio de una tabla In de enteros la cual puede ser accesada proporcionando 2 entradas (renglon y columna) y obteniendo 1 salida. Cada salida contiene los valores de una medida asociada a la imagen. Lo anterior se puede escribir de la siguiente manera:

$$In(i,j) = \begin{cases} In(1,j) \\ \vdots \\ In(n,j) \end{cases}$$

-Los valores de las entradas o de las salidas son discretos

-Los indices i,j toman sus valores en el dominio espacial discreto de la imagen

(i,j) ∈ [1,M] x [1,N] ; M,N dependientes del tamaño de la imagen

-Los valores de cada campo Imp(i,j) son positivos y enteros

Imp(i,j) ∈ [0,Ip-1]

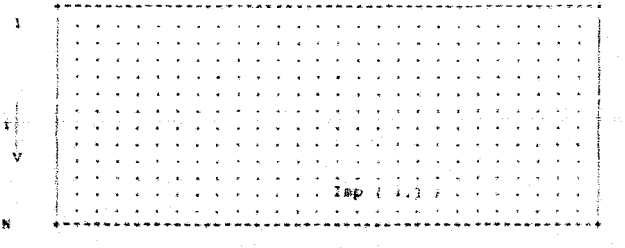
con Ip dependiente del numero de niveles usados para la cuantificacion del campo Imp.

El caso mas simple es cuando p=1, esto indica que solo se tiene una medida de la imagen. Las imagenes monocromaticas son un ejemplo de imagenes con p = 1 e Im(i,j) representa la intensidad de la señal de la imagen (en blanco y negro). Los I valores discretos se llaman niveles de gris. Cada punto de la imagen, definido por los indices i,j reciben el nombre de pixel (picture element).

En el caso de imagenes naturales en color, p=3 y los tres campos Imp(i,j) representan los colores primarios utilizados para grabar las imagenes originales de las imagenes. En algunas ocasiones, los valores de cada uno de los campos reciben el nombre de niveles de gris.

5.2. Convenciones para la Representacion de una imagen digital

Cada campo Im(0,1,...,p) de la imagen digital Imp se representa con una matriz cuyos renglones se especifican con el indice i y cuyas columnas con el indice j con respecto a un origen, ubicado en el elemento superior izquierdo de la matriz. Cada elemento de la matriz indica el valor del pixel asociado de este campo.



5.3. Tecnicas de Realce aplicadas a imagenes

El realce se basa en tratar de realzar ciertos rasgos de la imagen para enfatizar ciertas caracteristicas .

El realce consiste en encontrar una propiedad o rasgo de la imagen de tal forma que al realzarla sobre los demas rasgos, se obtenga una mayor información . Una vez que se ha definido este rasgo se define un algoritmo que haga mas notorio este rasgo.

En general, los metodos disponibles ya establecidos, corresponden a procedimientos ad-hoc que solucionan problemas muy especificos.

Algunos de estos procedimientos son :

-Acotamiento

-Laplaciano

-Operaciones al histograma

-Filtrado lineal

La mayor parte de estos procedimientos son de caracter no-lineal.

El ultimo metodo (filtrado lineal) corresponde a operaciones de filtrado sobre la imagen utilizando la tecnica de los filtros digitales. A continuación se presenta la descripcion de algunos de estos metodos.

5.3.1. Acotamiento de una imagen

La tecnica de Acotamiento tiene como propósito principal el detectar formas en imagenes. Tambien se utiliza para convertir una imagen multinivel en una imagen binaria.

La tecnica de acotamiento se rige por la siguiente regla que debe de ser aplicada a cada uno de los punto de la imagen .

$$nf = f(n0) = \begin{cases} 0 & si\ 0 <= n0 < umbral \end{cases}$$

L-1 si umbral <= nf <= L-1

donde:

umbral.-Es el valor contra el que se comparan todos los puntos

n0.- Cada uno de los puntos de la imagen

nf.- Nuevo valor que toma el punto

L.- Numero de niveles que tiene la imagen

Deteccion de Formas

En las imágenes digitales a veces es necesario localizar un patrón. Esto se puede lograr con la ayuda de la técnica de umbralamiento. El procedimiento consiste en aplicar a la imagen este método para utilizando diferentes valores de umbral hasta que aparezca el patrón deseado.

Obtencion de Imágenes Binarias

Por medio del método de umbralamiento se pueden obtener imágenes binarias asignando un valor A a todos aquellos pixeles que tengan un valor menor al umbral y asignando un valor B a todos aquellos pixeles que tengan valores mayores o iguales a este umbral.

Al aplicar este método a imágenes multinivel se pueden obtener imágenes binarias para todos el rango de niveles con que cuenta la imagen.

5.1.2. Modificacion del Histograma

Modificacion de Contraste de una Imagen

La modificacion de contraste de una imagen se puede hacer por medio de la modificacion de los valores del histograma.

Los metodos para la modificacion del histograma son : Ecualacion de histograma y Especificacion de histograma.

Esta tecnica puede llegar a estar constituida por 3 partes:

1.- Obtencion del histograma

2.- Modificacion del histograma por medio de ecualacion de histograma

3.- Modificacion del histograma por medio de especificacion de histograma

Obtencion del Histograma

El histograma de intensidades de una imagen da una descripcion global de la apariencia general de una imagen. Basicamente lo que nos proporciona es la cantidad de pixeles utilizados para cada nivel utilizado en la imagen, de esta forma se puede decir que una imagen oscura estara dominada por niveles bajos.

Ahora bien, dado que el histograma nos proporciona una descripcion global de la imagen si se modifica de alguna forma este histograma y se proyectan estos cambios a la imagen entonces se estara afectando el contraste de la imagen.

El histograma se obtiene por medio de la acumulacion de cada una de las apariciones de los niveles de esa imagen y calculandolos de acuerdo a la formula (1) (Ver formulacion matematica). Estos valores son almacenados para uso posterior.

Modificacion de histograma por medio de igualacion de histograma

Una vez que los valores del histograma se obtuvieron, se procede a manipular esta información de acuerdo a la formula de cierta formulacion matematica. Los valores obtenidos son la nueva representacion de histograma con que cuenta la imagen. Estos nuevos valores se deben de mapear a la imagen de acuerdo a la siguiente regla:

Para cada pixel :

nuevo_nivel_intensidad = (suma nuevo_histo[anterior_nivel_intensidad] / N) * MAX

Una vez hecho esto se cuenta con una nueva imagen modificada. Es importante indicar aqui que la transformacion se efectua con cierta especificacion de forma requerida para el histograma. Este tipo de igualacion de histograma produce basicamente imagenes cuyas regiones regiones en las que la mayor concentracion de niveles a lo largo de la escala de niveles de la imagen (intensidad).

Modificacion de histograma por medio de especificación de histograma

Este metodo es una extension del metodo anterior ya que permite especificar la forma deseada que debe de tener el histograma.El procedimiento consiste basicamente en :

1.- Obtener el histograma de la imagen

2.- Obtener la igualacion del histograma

3.- Obtener la funcion inversa de la funcion que se desea tenga el histograma

4.- Aplicar a cada uno de los valores obtenidos por igualacion de histograma la funcion inversa obteniendo de esta forma el nuevo histograma

5.- Obtener las nuevas intensidades de cada pixel con el histograma anterior

Los puntos 1,2 y 5 se generan como ya se explico anteriormente. El punto 3 (funcion inversa) se obtiene primero especificando la funcion que se desea tenga el histograma. En este caso se especifico la siguiente funcion :

$$y = -x + 1$$

Esta es una funcion que decrece desde 1 a 0 en el dominio [0 1].

Nota: Antes de continuar es importante indicar que la funcion que se desea debe de generar valores en el rango y = [0..1].

La funcion inversa de esta funcion es :

x = 1 - y

El punto a considerar en aplicar esta funcion a cada uno de los valores de, histograma ; es necesario para usarlos en el punto 5.

5.3.3. Manejo del histograma

Formulacion Matematica

El histograma es una arreglo (NIV_MIN..NIM_MAX) que contiene la cantidad de apariciones de niveles en una imagen . Se obtiene de la siguiente manera:

$$p(r_j) = n_j / n \qquad (1)$$

donde :

p = es el arreglo

r_j = es el nivel

n_j = es el numero de apariciones del nivel r_j

n = es el numero total de pixeles en la imagen

Igualacion de histograma

La igualacion de histograma se hace por medio de la siguiente ecuacion:

[illegible]

Como se puede apreciar esta es una funcion cumulativa [illegible] distribuir [illegible] a lo largo de los niveles de intensidad. Una vez obtenido el nuevo histograma se procede a mapear [illegible] los valores de la imagen original a los nuevos valores.

Especificacion de histograma

1.- Obtener el histograma [illegible] de la imagen

2.- Hacer una igualacion a los niveles del histograma obtenido

3.- Obtener una funcion de transformacion de la forma

[illegible]

usando el histograma especificado

4.- Obtener la funcion de transformacion inversa y aplicarla al histograma del punto 2

[illegible]

5.- Mapear estos resultados a la imagen original

A continuacion se presentan algunas tecnicas que pueden ser usadas en conjuncion con las tecnicas de manejo de histograma. Estas tecnicas tienen la caracteristica de acentuar rasgos finos de la imagen [illegible] se basan principalmente [illegible] haciendo uso de [illegible] local para cada punto en la imagen [illegible] la informacion visual existente en una imagen digitalizada. Estas tecnicas pueden ser utilizadas [illegible] igualacion de histograma para producir imagenes [illegible] distribucion [illegible] mas plana.

5.3.4. Metodos de Preprocesamiento

5.3.4.1. Correccion por Media local

Este metodo ajusta el nivel de gris local en cada punto hacia la media local. Supongase que μ denota la media global y que el nivel de gris promedio en la vecindad n por n de un pixel es $\bar{g}$. El algoritmo iluminara el valor del nivel de gris g del pixel si $\bar{g}$ es mas obscuro que el promedio general μ, y obscurecera g si $\bar{g}$ es mas brillante. Especificamente se dara al pixel un nuevo nivel de gris:

[illegible]

Los parametros para este esquema son el tamaño de la vecindad n, y el coeficiente de [illegible] β.

Valores grandes de n incrementan tanto el realce de lineas y curvas como los valores de los puntos en el interior de extensas regiones.

La computacion de medias locales es simple y barata, ya que una vez que se ha obtenido la suma de la vecindad alrededor del punto (supongase n×n), la suma para cada vecindad sucesiva puede ser obtenida con solo n adiciones y n substracciones.

5.3.4.2. Modificacion por Histograma Local

Transforma cada punto acorde al histograma local definido en la vecindad del punto. De un modo similar a la anterior tecnica, regiones obscuras y sombreadas tienden a ser iluminadas por este proceso. La Modificacion por Histograma Local retiene la ventaja de igualacion de histograma en regiones pequeñas, pero esencialmente genera una serie de iluminaciones de estas regiones en una sola region mediante el computo separado para cada punto.

A diferencia de la correccion por media local, la modificacion por histograma local no necesita de parametros para ajustar el grado de correccion.

5.3.4.3. Laplaciano

Una tecnica que puede preceder una igualacion global del histograma mejorara despegacion de imagenes. Esta tecnica recibe el nombre de Laplaciano. Es bien conocido que si un multiplo pequeño del Laplaciano es substraido en cada punto de una imagen, el resultado tendera a realzar bordes, lineas y puntos (Hummel 1977).

Una imagen procesada por el laplaciano puede ser computada en cada punto de acuerdo a la siguiente formula :

$$L_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{n} g_{ij} * \phi_{ij}$$

una vez que se tiene L_{ij} para obtener el pixel procesado por el laplaciano se debe tener la siguiente operación :

$$y = x - L_{ij}$$

la imagen Y es el resultado de la aplicación del laplaciano.

La tarea principal del laplaciano es la de descubrir bordes. Esto trabaja mejor cuando los bordes y líneas que son el objeto del trabajo delinean regiones bien definidas correspondientes a tanto diferentes en la imagen original. Cuando las líneas y bordes son en el mismo los fuegos más prominentes, la descripción no trabaja muy bien.

Una dificultad importante con la teoría de los laplacianos consiste en que pulsos de ruido son a menudo realzados que los bordes.

5.3.5. Estudio de filtros Digitales.- Caso Unidimensional

5.3.5.1. Introduccion

El estudio de filtros digitales corresponde a la aplicación que se desea realizar referente al realce de imágenes. A continuación se citan algunas definiciones y explicación de las tres familias de filtros que existen. A saber : filtros fir, filtros iir

Tambien se plantean varios metodos para el diseño de filtros fir que tienen la caracteristica de poseer una fase lineal. Por ultimo se explica la metodologia utilizada para la obtencion de filtros bidimensionales a partir de filtros unidimensionales.

El material para el caso de filtros unidimensionales fue tomado de [vera 90] .

5.3.5.2. Antecedentes

Antes de iniciar una explicación referente a filtros digitales, se dan algunas definiciones.

La respuesta impulso h[n] de un sistema se define como la salida obtenida al aplicar un pulso unitario δ[n] en el instante cero.

$$y[n] = h[n] * \delta[n]$$

$$y[n] = h[n] \qquad n = 0$$

La función de transferencia de un sistema se define como la transformada Z de su respuesta impulso:

$$H(z) = \sum_{n=0}^{\infty} h[n] z^{-n}$$

La respuesta en frecuencia de un sistema corresponde a evaluar su transformada Z en el punto $z = e^{j\omega}$, lo que físicamente equivale a aplicar una entrada que contiene todo el rango de frecuencias. Esto nos da el comportamiento del sistema para todo el rango de frecuencias.

$$H(e^{j\omega}) = \sum_{n=0}^{\infty} h[n] e^{-j\omega n}$$

El corrimiento de fase (phase delay) se define como

$$\tau_p = \frac{-\phi(\omega)}{\omega}$$

En especial esta última definición es de gran utilidad para los filtros FIR de fase lineal.

5.3.5.3. Diseño de filtros digitales

Parks y Burrus proponen una metodología para el diseño de filtros (Parks 1987). A continuación se explica brevemente.

El diseño de filtros digitales involucra la ejecución iterativa de los siguientes pasos:

1 - Aproximación

Es el proceso de generar una función de transferencia que refleje un conjunto de especificaciones deseadas (como frecuencia, tamaño del filtro, etc.).

2.- Realización

Consiste en la conversión de la función de transferencia deseada en una red de filtrado.

3.- Estudio de errores aritmeticos

4.- Implantación

Este método es iterativo porque en cada uno de estos pasos se establece si nuestro modelo sigue cumpliendo con las especificaciones deseadas, sino es así, entonces se procede a buscar otra opción que de los resultados esperados para obtener el filtro correcto.

Dentro de los filtros digitales lineales existen 2 familias de filtros que son los FIR e IIR, entre los cuales la principal diferencia radica en la duración de la respuesta impulso de los mismos siendo para el primero finita y para el segundo infinita.

5.3.5.4. Clasificación de filtros: Filtros FIR

los filtros FIR se expresan de la siguiente manera :

$$y(n) = \sum_{m=0}^{N-1} h(m)\, x(n-m)$$

Estos filtros cuentan con las siguientes caracteristicas :

- Duración finita de la secuencia
- Función de transferencia polinomial (solo ceros)
- Por el método de implantación se dice que es no recursivo
- Respuesta pulso unitario simetrica alrededor de (N-1)/2
- Pueden tener fase lineal (τg constante)
- Estable
- Para operaciones en que importa la forma de la onda
- Metodos de diseño generalmente lineales
- Calculo sencillo

- Puede llegar a requerir grandes longitudes para altas respuestas de frecuencia, esto implica mayor cantidad de memoria como de tiempo de procesamiento.

- Si hay fase lineal el retraso de tiempo puede llegar a ser de hasta p/2

5.3.5.5. Clasificación de filtros : Filtros IIR

Los filtros IIR(Respuesta Impulso Infinita) se representan por medio de la siguiente expresion :

$$y(n) = -\sum_{k=1}^{N} a(k)\, y(n-k) + \sum_{k=0}^{M} b(k)\, x(n-k)$$

Parte recursiva o retroalimentacion — Parte No-recursiva

Claramente se ve que existe una parte de retroalimentacion que es la que puede reducir la cantidad de coeficientes, pero tambien si no es bien manejada puede traer problemas principalmente de inestabilidad.

Sus principales caracteristicas son las siguientes:

- Region de transicion muy corta entre banda de paso y banda supresora
- Corte mas agudo que un filtro FIR debido a que posee polos y ceros
- rp y rq no tan buenos como en los FIR
- Menos memoria que FIR
- Inestabilidad y gran ruido de cuantizacion
- Para aplicaciones en que la fase no es importante
- La retroalimentacion proporciona un filtro con menos coeficientes y menos aritmetica
- Funcion de transferencia racional (polos y ceros)
- No hay posibilidad de fase lineal
- Mayor sensibilidad a efectos de cuantizacion

En este resumen se proporciona informacion acerca de los metodos de diseño de filtros FIR. Tambien es importante notar que debido a que los filtros FIR poseen una fase lineal seran estos filtros los unicos que se utilicen en las aplicaciones con imagenes.

Ya vez descritos los filtros existentes, se procederá a describir en detalle algunas características importantes para los filtros FIR.

Cuatro de los filtros FIR tenemos una característica que los hace destacar de un manera importante respecto a los IIR. Esta característica se refiere a la linealidad de la fase del filtro.

Existen 4 tipos de filtros FIR. A saber:

Filtro de longitud impar y simetría par

Filtro de longitud par y simetría par

Filtro de longitud impar y simetría impar

Filtro de longitud par y simetría impar

Esta categorización es importante ya que así podemos elegir que tipo de filtro es mas propio para un aplicación dada.

5.3.5.6. Metodos de Diseño para filtros FIR

Los metodos de diseño para filtros FIR tienen como objetivo obtener los coeficientes de la respuesta impulso del filtro, esto se logra por metodos de aproximacion a la función deseada .

Existen basicamente 5 procedimientos que son:

1.- Diseño por muestreo en frecuencia

2.- Diseño por error mínimo cuadratico en la frecuencia

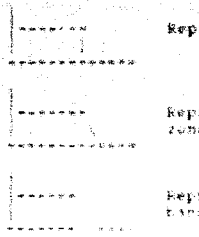
3.- Diseño de filtros con una region de transicion y error mínimo cuadrático

4.- Diseño de filtros con banda de tramsicion

5.- ventanas

Todos estos procedimientos aproximan una función a la función deseada(en el caso de los filtros FIR, una función polinomial) .

Estos metodos se basan en las siguientes representaciones de un filtro:



Representacion 1. Filtro Ideal

Representacion 2. Filtro con zona de transicion

Representacion 3. Filtro con banda de transicion

REPRESENTACIONES DE FILTROS

Al considerar cualquiera de las representaciones anteriores se refleja en la realizacion fisica, error obtenido, confiabilidad; criterios que son importantes.

5.3.5.7. Diseño por Muestreo en Frecuencia

Este metodo consiste en obtener muestras de la expresion que representa el comportamiento en frecuencia del filtro. Una vez hecho esto se obtiene la IDFT y asi se obtienen los coeficientes . Es importante indicar aqui que la funcion de transferencia se muestrea en N muestras alrededor del circulo unitario ; N es la longitud del filtro que se desea).

Este metodo es directo y rapido. Su desventaja radica en que este metodo de ninguna manera nos da un comportamiento adecuado entre las muestras que obtengamos, esto puede traer problemas con señales que posean frecuencias diferentes de las muestras obtenidas.

Este algoritmo se tiene implementado.

5.3.5.8. Diseño por error minimo cuadratico en la frecuencia

El anterior metodo consiste basicamente en obtener N muestras de la funcion de transferencia del filtro no tomando en cuenta que pasa cuando se obtienen señales que tienen una frecuencia entre muestras.

Para resolver este problema lo que se hace es obtener muchas más muestras en el dominio de la frecuencia (L muestras). Una vez que se tienen estas muestras se deben truncar y seleccionar solo aquellos valores situados simetricamente alrededor del origen , se seleccionan M = (N-1) / 2 muestras a la izquierda y otras M muestras a la derecha de tal forma que se obtengan los N valores deseados. Al truncar solo los N valores obtenemos una aproximacion que tiene un error llamado minimo cuadratico definido de la siguiente manera:

$$E = \sum_{k=0}^{(L-1)} Ak = Ada \int = \sum_{n=0}^{n(L-1)} hn = hna$$

Este criterio es de bastante utilidad pues nos permite visualizar el error como la energía de diferencia existente entre dos señales.

Existe otro criterio de error que se llama criterio de error por aproximación integral. Sustituya la expresión que a definido en semejante al criterio de error anterior, la diferencia radica en que en lugar de considerarse en cuenta solo las muestras, se considera todo el espectro de frecuencias. Este criterio es utilizado solo cuando el numero de muestras N es muy grande. La solución es analítica.

El método con uso de el criterio de error mínimo cuadrático ha sido implantado.

3.3.5.5. Diseño de filtros con una region de transicion y LS(error minimo cuadratico)

Este diseño se basa en obtener un filtro que tenga la representacion numero 2 de los tipos de filtros definidos. Existe una funcion que encadena o liga a la banda de paso y a la de supresion. Esta funcion suaviza la brusca caida que una representacion ideal de un filtro produce. Esta discontinuidad si no es suavizada produce el fenomeno de Gibbs que es una especie de oscilacion en la zona de transicion. Para mayor informacion sobre el fenomeno de gibbs consulte (Parks [illegible]).

Este metodo es muy flexible ya que al implantarlo en un programa podemos apreciar diferentes respuestas del filtro de una manera interactiva y con esto podemos elegir el mejor filtro que se adecue a nuestras especificaciones. Su principal desventaja radica en que es un metodo de prueba y error.

Las dos principales funciones de transicion utilizadas son:

$$h(n) = \begin{cases} \left(\dfrac{\sin(\pi(f2-f1)(n-M)/P)}{\pi(f2-f1)(n-M)/P} \right)^{P} \dfrac{\sin(\pi(f2+f1)(n-M))}{\pi(n-M)} & ; 0 <= n <= N-1 \\ 0 & ; \text{otro numero} \end{cases}$$

SPLINE DE ORDEN P

$$h(n) = \begin{cases} \left(\dfrac{\cos(\pi(f2-f1)(n-M))}{1 - 4(f2-f1)(n-M)} \right)^{P} \dfrac{\sin(\pi(f2+f1)(n-M))}{\pi(n-M)} & ; 0 < n < M \\ 0 & ; \text{otro numero} \end{cases}$$

RAISED-COSINE DE ORDEN M

Este metodo ha sido implantado

5.3.5.10. Uso de una banda de transicion

Este metodo usa la representacion numero 1 de los tipos de filtros. Por tanto no considera en el error una parte del dominio de la frecuencia. Esto trae como resultado una disminucion en el error estimado. Su desventaja radica en que si se presenta una senal con una frecuencia dentro de esta zona muerta, los resultados no se pueden predecir.

5.3.5.11. Ventanas

Una ventana se define como una funcion que multiplica a otra, alterando la respuesta de esta ultima :

$$h_d(n) * w(n) = h_d(n)$$

$h_d(n)$ = prototipo ideal

$w(n)$ = ventana

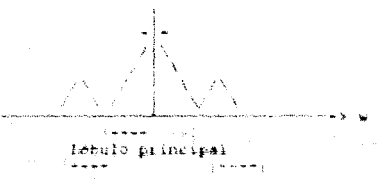
Existen muchos tipos de ventanas ya definidas como se indicara abajo, pero en general se recomienda utilizar y seleccionar una ventana bajo el siguiente principio de seleccion (para mayor informacion consulte [Harris 197?]):

Definicion :

1 bin (FT = cualquier multiplo de (2*pi)/N

donde N es la longitud de la ventana

En general la representacion espectral de una ventana es de la siguiente forma:



Principio : el ancho del lobulo principal (central) no puede ser menor que (2**r)/N, que es el ancho de banda del kernel de Dirichlet o ventana rectangular. Examinando varias ventanas vemos que cada 20 dB de

caida de los lóbulos laterales costara aproximadamente un bin DFT de ancho. Las ventanas de muy buen rendimiento exhibiran anchos de banda entre 3 y 4 Bins para una supresion de lóbulos laterales de mas de 60 dB. Tambien se debe tomar en cuenta la forma que toman estos lóbulos laterales (examinado graficas de frecuencia Vs. 20 log(magnitud)), pues en un momento dado y por las señales aplicadas puede introducir un ruido.

Diferentes tipos de ventanas:

Basicamente se pueden distinguir dos grupos de ventanas que son las fijas y las que estan en funcion de parametros que pueden tener o no alguna relacion con las estadisticas de la imagen o señal.

Ventanas Fijas:

Ventana Rectangular

$W(n) = 1.0 \; ; \; n = -N/2, \ldots, -1, 0, 1, \ldots, N/2$

Ventana Triangular (Fejer/Parzen) :

$$W(n) = 1.0 - \left| \frac{2n}{N} \right| \; ; \; n = -N/2, \ldots, -1, 0, 1, \ldots, N/2$$

$$0.42 + 0.5\cos(2\pi n/(N-1)) + 0.08\cos((4\pi n)/(N-1)) \; ; \; |n| <= (N-1)/2$$

$$0 \quad \text{cualquier otro numero}$$

Ventana triangular Bartlett

$$W(n) = \begin{cases} 2(n-1)/(N+1) & : \; n = 0, 1, 2, \ldots, (N-1)/2 \\ 2 - 2(N-1)/(N+1) & ; \; n = (N-1)/2, \ldots, N-1 \\ 0 & : \quad \text{cualquier otro numero} \end{cases}$$

Ventana $\cos^2$: (Hanning)

$$W(n) = 0.5 + 0.5 * \cos(n*(2*pi)/N) \; ; \; n = -N/2, \ldots, -1, 0, 1, \ldots, N/2$$

Ventana Raised cosine : (Hamming)

$$W(n) = 0.54 + 0.46\cos(n*(2*pi)/N) \qquad ; n = -N/2,\ldots,-1,0,1,\ldots,N/2$$

Ventanas parametricas

Ventana Gaussiana

$$W(n) = \exp(-1/2*(n/N*beta)^2) \qquad ; n = -N/2,\ldots,-1,0,1,\ldots,N/2$$

Ventana Dolph-Tchebyschev

Se define por medio de su espectro en la frecuencia:

$$W(n) = FFT\left[\frac{\cos(N*\cos^{-1}(\alpha*\cos(\pi*n/N)))}{\cosh(N*\cosh^{-1}(\alpha))}\right]$$

donde

$$\alpha = \cosh\left(\frac{1}{N}\cosh^{-1}\left(10^{beta}+\sqrt{10^{(2*beta)}-1}\right)\right)$$

Ventana Kaiser

$$W(n) = \begin{cases} \dfrac{Io\left(\beta\sqrt{1-(2n-1)/(N+1)}\right)}{Io(\beta)} & ; n = 0,1,2,\ldots,N-1 \\ 0 & ; \text{cualquier otro numero} \end{cases}$$

Observaciones respecto a estas ventanas:

La ventana rectangular genera el problema de Gibbs.

La ventana rectangular de Barttlet reduce el overshoot pero amplia la region de transicion.

Las ventanas de Hanning,Hamming , Blackman usan funciones cosenoidales mas complicadas para obtener un truncamiento suave. Aqui hay que agregar que el termino cosenoidal adicional en la ventana de Blackman genera una reduccion extra en la amplitud de las oscilaciones de Gibbs.

Las ventanas parametricas son particularmente importantes ya que con estas podemos obtener una funcion ajustandola a lo que deseemos por medio de sus parametros. Estos parametros son :

- Para la gausiana el parametro beta que es la desviacion estandar

reciproca

- Para la Dolph-Tchebyschev es beta que es el numero de decadas que los lóbulos laterales caen relativos al lóbulo principal

- Para la Kaiser es beta que es el producto del ancho de banda. Este parámetro beta permite un ajuste de el compromiso entre la reducción del overshoot y el ancho de la region de transicion

- La ventana kaiser es un tipo de funcion que es muy cercana a lo optimo en el sentido de tener la mayor cantidad de energia en el lóbulo principal

Estas ventanas han sido implantadas.

Hasta este punto se han planteado técnicas y procedimientos para la obtención de filtros paso-bajas de una sola dimensión. En la parte que sigue se explicara como es posible obtener filtros bidimensionales a partir de estos. Tambien se introducen algunas relaciones de utilidad para la obtención de filtros paso-altas y paso-bandas a partir de filtros pasobajas. A esta ultima parte se le denomina algebra de filtros, pues como se verá involucra operaciones de tipo aritmetico entre los bloques de procesamiento (coeficientes del filtro). Dentro de esta parte se ejemplifica como debe de ser desarrollado este tipo de algebra dentro del sistema y se plantea el uso del filtro obtenido para el filtrado de imagenes proporcionadas al sistema (medal en dos dimensiones).

A continuación se presenta una transformación mediante la cual es posible obtener filtros bidimensionales a artir de filtros unidimensionales. Mediante esta transformación ha sido posible obtener las representaciones bidimensionales de los filtros dentro del sistema.

5.3.6. Técnica para transformación de filtros unidimensionales a bidimensionales

La siguientes técnica de transformacion que se presenta tienen como proposito obtener filtros bidimensionales a partir de filtros unidimensionales. La ventaja que se obtiene al utilizar esta estrategia consiste en hacer el mayor uso posible de técnicas y procedimientos ya bastante aceptados para diseño de filtros unidimensionales aplicados al filtrado bidimensional.

La tecnica de transformacion utilizada es la transformación de Huang.

5.3.6.1. Transformacion de Huang

Thomas S. Huang demostró [Huang 1972] una transformación que ha sido de mucha utilidad para el diseño de filtros bidimensionales a partir de filtros unidimensionales.

A continuación se analiza la transformación de Huang.

Huang demostró que si una ventana unidimensional simetrica w1(x) y una ventana bidimensional w2(x,y) estan relacionadas en la forma:

$$w2(x,y) = w1(\sqrt{x^2 + y^2})$$

entonces las transformadas de Fourier W(u) y W2(u,v) satisfacen la relación :

$$\frac{1}{2\pi} W2(u,v) = H2(u,v) * W(u) * H(u)$$

donde

$$H(u) = \begin{cases} 1 & \text{para } u \geq 0 \\ 0 & \text{para } u < 0 \end{cases}$$

$$H2(u,v) = \begin{cases} 1 & \text{para } u \geq 0 \text{ y todas las } v \\ 0 & \text{para } u < 0 \text{ y todas las } v \end{cases}$$

* = convolución

Mediante este resultado lo que se obtiene como conclusion es que la respuesta en frecuencia w2 es igual en forma a la obtenida del filtro w1 pero con una escalación de un factor $1/(2\pi)$, esto es, que si la respuesta del filtro unidimensional es satisfactoria, tambien lo sera para el filtro bidimensional.

Ademas, el filtro w2 obtenido tiene la propiedad de que es circularmente simetrico en el dominio de la frecuencia.

5.3.7. Estudio de filtros digitales: Caso Bidimensional

5.3.7.1. METODOLOGIA PARA DISEÑO DE FILTROS

El uso de bloques funcionales para especificar un filtro es útil en el sentido de que se pueden agregar o quitar bloques y probar cual es el resultado. En este sentido a continuación se proporciona una metodología para el diseño de filtros dentro del sistema.

Los estudios de diseño de filtros utilizados son dirigidos hacia la obtención de filtros paso-bajas unidimensionales. Es sabido que a partir del diseño de filtros paso-baja es posible obtener filtros de otros tipos como por ejemplo filtros paso-altas y pasa banda por ejemplo. A continuación se proporcionan las relaciones por las que es posible llegar a estos filtros. Estas relaciones son de gran utilidad pues permiten que se haga un desarrollo por capas como se muestra:

OBTENCION DE FILTROS :

	FILTROS PASO-ALTAS, PASO-BAJAS BIDIMENSIONALES
A PARTIR DE :	FILTROS PASO-BAJAS BIDIMENSIONALES
A PARTIR DE :	FUNCION DE TRANSFORMACION (MEANS)
A PARTIR DE :	FILTROS PASO-BAJAS UNIDIMENSIONALES
A PARTIR DE :	METODOS PARA DISEÑO FILTROS PASO-BAJAS UNIDIMENSIONALES
USO DE :	ESPECIFICACIONES Y CARACTERISTICAS DE FILTRO A SISTEMA

DISEÑO DE FILTROS BIDIMENSIONALES

Por medio de esta gráfica queda explicado que pasos se realizan para llegar a la obtención de filtros bidimensionales.

5.3.7.2. ALGEBRA DE FILTROS.- Obtencion de Filtros PASO-ALTA y PASO-BANDA a partir de filtros PASO-BAJAS

A partir de la obtencion de filtros paso-bajas, es posible obtener filtros paso-altas y Paso-Bandas mediante las siguientes relaciones

Si h-LP(m,n) denota a un filtro paso-bajas FIR, entonces un filtro paso altas FIR h-HP puede ser definido como:

$$h\text{-}HP(m,n) = \delta(m,n) - h\text{-}LP(m,n)$$

Tal filtro puede ser implantado mediante la substraccion de la salida del filtro pas-bajas de la entrada original. Como se puede ver no hay ninguna restriccion en cuanto al metodo de diseño del filtro paso-baja empleado.

Un filtro pas-banda puede ser obtenido mediante la siguiente expresion:

$$h\text{-}BP(m,n) = h\text{-}L1(m,n) - h\text{-}L2(m,n)$$

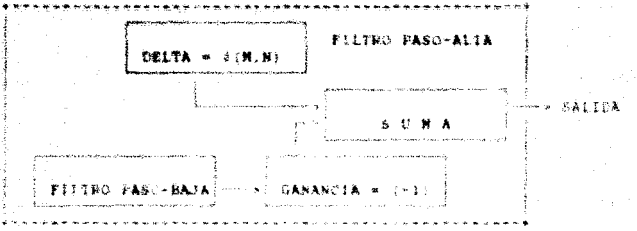
donde h-L1 (m,n) y h-L2 (m,n) denotan dos FIRs de filtros paso-bajas.

Ahora en cuanto al uso que se puede dar a estos filtros Jain (Jain 1989) aconseja los siguientes usos :

los filtros paso-bajas son utiles para suavizacion de ruido , filtros Paso-altas son utiles para la extraccion de bordes y para la agudizacion de imagenes. los filtros paso-bandas son utiles para el realce de bordes y otras caracteristicas de alta frecuencia en imagenes bajo la presencia de ruido.

5.3.7.3. Implantacion de un filtro Paso-Alta dentro del Sistema

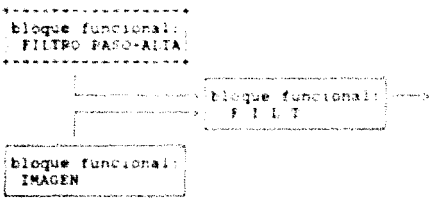
La implantacion de un filtro paso-altas en el sistema consiste basicamente en declarar y corectar los siguientes bloques:



Bloque Funcional FILTRO PASO-ALTA

Como se puede ver la implantación consiste en aplicar la fórmula arriba especificada para obtener filtros paso-alta a partir de filtros paso-baja.

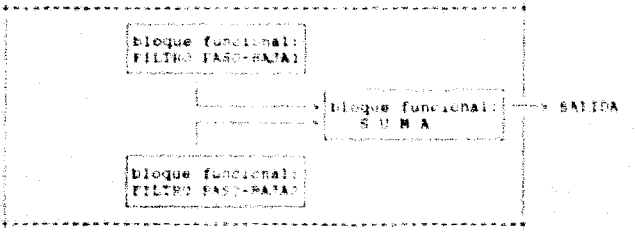
La SALIDA de suma es adonde se encuentra el filtro paso-altas y esta salida es la que se debe de conectar como se indica para lograr el filtrado de imágenes dentro del sistema:



Configuración para Filtrado de una Imagen

5.3.7.4. Implantación de un filtro Paso-Banda dentro del Sistema

La implantacion tambien se lleva de acuerdo a la formula arriba especificada :



Bloque Funcional FILTRO PASO-BANDA

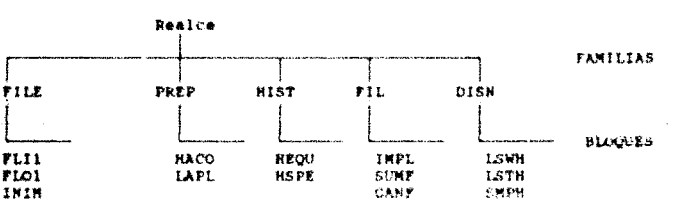
La SALIDA obtenida se multiplica con una imagen de la misma forma que para los filtros paso-alta y paso-baja (Ver **Configuracion para Filtrado de una Imagen**).

6. BLOQUES DE PROCESAMIENTO DE IMÁGENES IMPLANTADOS EN EL SISTEMA

Como mencione al principio de este trabajo, el sistema esta orientado hacia la implantación de algunas técnicas para realce de imágenes. Principalmente se ha hecho distinción de dos áreas en las que las técnicas de realce han tenido un gran desarrollo. La primera consiste en la implantación de técnicas que son denominadas clásicas, ya que tienen una aplicación o resultado muy específico. Ejemplos de estas técnicas son la manipulación del histograma de una imagen, el preprocesamiento de imágenes mediante el análisis de características tales como la media local, histograma local y laplaciano.

Dentro del segundo grupo de aplicaciones se encuentran aquellas que tienen una relación directa con el procesamiento digital de señales, pero aplicado en dos dimensiones. Ejemplos de esto son el uso de técnicas bien conocidas y probadas para el desarrollo de filtros unidimensionales, las cuales mediante transformaciones pueden ser mapeados a dos dimensiones. Dentro de esta misma parte pero no siendo implantados como bloques funcionales, se han implantado a nivel de herramientas la visualización en tres dimensiones de los filtros generados tanto en el dominio espacial como en el dominio de la frecuencia; en especial en esta ultima parte se hizo uso de un algoritmo novedoso para obtener la representación frecuencial a partir de la representación espacial (articulo de IEEE Tansactions [Benmahammed 1989]).

Ahora bien, las diferentes técnicas implantadas han sido agrupadas como sigue:



6.1. Familias de procesamiento

FILE .- Se refiere a todas las operaciones para leer o almacenar una señal (imagen) dentro o fuera del sistema (es la interfaz con el mundo exterior).

PREP .- Es el conjunto de funciones que son recomendables antes de algun procesamiento en especial de funciones para manejo del his-

tograma.

HIST .- Son funciones para manejo del histograma. Entre las funciones implantadas se encuentran la igualacion de histograma y la especificacion de histograma.

FIL .- Se refiere a las operaciones que involucran filtros. Entre estas operaciones se encuentran las mask de filtros, ganancia de un filtro y el operador FILT que permite obtener el filtrado de una imagen a partir de filtros generados por el sistema (familia FISN o como resultado de algebra de filtros).

FISN .- Esta familia contiene varias tecnicas mediante las cuales se pueden obtener filtros isotropicos de filtros. Es importante indicar que los filtros obtenidos mediante esta familia son filtros pasa-bajas siendo esta restriccion de los metodos de diseño empleados. También como nota importante en lo que se refiere a que partir de estos filtros y mediante el algebra de filtros es posible obtener filtros pasa-altas y pasa-banda.

6.2. Bloques de Procesamiento

A continuacion se presenta una explicacion de los bloques de procesamiento implantados en el sistema.

6.2.1. Bloque de Procesamiento : FLI1

Nombre Completo : lectura de Imagenes

Por medio del bloque FLI1 se realiza la lectura de imagenes. Las imagenes que accesa el sistema estan almacenadas en archivos. Este bloque de procesamiento accesa imagenes de 256 por 256 pixeles, cada pixel ocupa un caracter y almacena la intensidad de la imagen con valores de 0 a 255.

Parametros:

Archivo a leer : Cadena de caracteres para especificar el archivo que contiene la imagen.

Tipo de Salida : IMAGEN

6.2.2. Bloque de Procesamiento : HEQU

Nombre Completo : ECUALIZACION DEL HISTOGRAMA

Este bloque realiza la ecualizacion de histograma de la imagen que se le proporcione como entrada.

Parametros:

NINGUNO

Tipo de Salida : IMAGEN

6.2.3. Bloque de Procesamiento : HACO

Nombre Completo : ACOTAMIENTO DE IMAGENES

El acotamiento de imagenes es una tecnica que permite obtener imagenes binarias a partir de imagenes con varios niveles de intensidad. Para obtener la imagenes binarias se compara cada pixel de la imagen contra un valor de comparacion que se llama umbral. Si el valor del pixel es mayor que el umbral se asigna un valor 1 al pixel , si es menor entonces se asigna un valor de 0.

Parametros:

Umbral : Valor entero de 0 a 255

Tipo de Salida : IMAGEN

4.2.4. Bloque de Procesamiento : HEPE

Nombre Completo : ESPECIFICACION DE HISTOGRAMA

Este bloque realiza la especificacion de histograma como fue explicado en capitulos anteriores. La forma de histograma manejada es una linea recta $x = 1 - y$.

Parametros:

NINGUNO

Tipo de Salida : IMAGEN

4.2.5. Bloque de Procesamiento : LAPL

Nombre Completo : OPERADOR LAPLACIANO

Este bloque aplica el operador laplaciano a la imagen de entrada. Este bloque se recomienda que sea utilizado en conjunción con la igualación del histograma para obtener imagenes con un histograma mas plano. La configuración que se recomienda para lo anterior es imagen_entrada -> laplaciano -> igualacion histograma.

Parámetros:

Beta : Valor real de preferencia en el rango de [0.1 a 5]

Tipo de Salida : IMAGEN

6.3.6. Bloque de Procesamiento : MLOC

Nombre Completo : CORRECCION POR MEDIA LOCAL

Este bloque realiza la corrección de imágenes a partir de una comparación entre la media de toda la imagen y la media de la región en la que se encuentra el pixel. El método es llamado Corrección por media local y esta explicado en un capitulo anterior. Este método también es recomendado a operar en manera semejante a como el Laplaciano opera con la igualación de histograma.

Parametros:

Beta : Valor real dentro del rango [0 .. 1]

Tipo de Salida : IMAGEN

6.2.7. Bloque de Procesamiento : HLOC

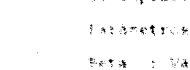
Nombre Completo : MODIFICACION POR HISTOGRAMA LOCAL

El método de modificación por histograma local funciona de una manera parecida al de corrección por media local. Aquí las medidas de comparación son el histograma global y el local dentro de la vecindad del pixel. También se recomienda si se desea una operación semejante a ecualizar en conjunción con la igualación de histograma.

Parametros:

Beta : Valor real dentro del rango (0 .. 1)

Tipo de Salida : IMAGEN



6.2.8. Bloque de Procesamiento : FILT

Nombre Completo : FILTRADO DE UNA IMAGEN A PARTIR DE UN FILTRO

Este filtro realiza la multiplicacion de un filtro por una imagen. El usuario debe de conectar una imagen y un filtro como entradas a este bloque, de lo contrario obtendrá un error por parte del sistema. La secuencia de ejecución se abortará.

Parametros:

NINGUNO

Tipo de Salida : IMAGEN

6.2.9. Bloque de Procesamiento : IMPL

Nombre Completo : FUNCION IMPULSO BIDIMENSIONAL

Este bloque se proporciona para poder obtener filtros pasa-bandas y pasa-altas a partir de combinaciones con filtros pasa-bajas obtenidos por cualquiera de los metodos con que cuenta el sistema.

Parametros:

N : Orden del filtro

Tipo de Salida : FILTRO

6.2.10. Bloque de Procesamiento : GANF

Nombre Completo : GANANCIA PARA FILTROS

Este bloque permite la multiplicación de un filtro por una ganancia especificada en el parametro Ganancia, los valores pueden ser positivos o negativos. Este bloque es de gran utilidad cuando se desea hacer algebra de filtros.

Parametros:

GANANCIA : Factor de ganacia del filtro (valor real)

Tipo de Salida : FILTRO

6.2.11. Bloque de Procesamiento : SUMF

Nombre Completo : SUMA DE FILTROS

Este bloque permite la suma de filtros. Los filtros deben de ser del mismo orden. Este bloque es de gran utilidad cuando se desea hacer algún tipo de filtros.

Parametros:

NINGUNO

Tipo de Salida : FILTRO

6.2.12. Bloque de Procesamiento : LSWH

Nombre Completo : DISEÑO DE FILTRO PASO BAJAS EN BASE A VENTANAS Y TRANSFORMACION DE HUANG

Por medio de este bloque es posible obtener un filtro bidimensional pasobajas, debido a que el diseño se basa en un método unidimensional se aplica la transformacion de Huang para obtener un filtro bidimensional. El usuario cuenta con diferentes tipos de ventanas a utilizar.

Parametros:

N : Orden del filtro : se especifica el numero de renglones. Los filtros obtenidos son cuadrados, es decir de dimension N x N ;

Tipo : Especifica el tipo de ventana que se quiere utilizar. Los tipos con que cuenta el sistema son :

* TIPOS DE VENTANAS *

RECTANGULO	0
BARTLETT	1
HANNING	2

HAMMING	3
BLACKMAN	4
KAISER	5

FP : Un valor real en Hertz (Hz) en el rango [0.. 0.5]. Se proporcionan estos valores pues los metodos de diseño trabajan sobre valores normalizados.

Tipo de Salida : FILTRO

Nota :

El parametro FP es un parametro de diseño que especifica la frecuencia de corte del filtro paso-baja. En general, dentro de este sistema y debido a que los metodos de diseño estudiados asi lo utilizaban, se hace uso de una notacion normalizada para la especificacion de las frecuencias involucradas. Un ejemplo es un filtro paso-baja con una banda de paso que se puede extender hasta la frecuencia de Nyquist (que es la mitad de la frecuencia de muestreo). Con notacion normalizada, la frecuencia de muestreo es W = 2 * rad/seg = 1 * 1 Hz (un ciclo por segundo). Por tanto, la frecuencia maxima que se puede especificar dentro del diseño de un filtro varia de 0 a 0.5 Hz (W* 5.

6.2.11. Bloque de Procesamiento : LSTH

Nombre Completo : DISEÑO DE FILTRO PASO BAJAS USANDO FUNCION DE TRANSICION Y TRANSFORMACION DE HUANG

Por medio de este bloque es posible obtener un filtro bidimensional paso-bajas.
El método de diseño utilizado implica el uso de una región de transicion. La forma que puede tener esta region de transicion es uno de los parametros de la funcion.Para obtener el filtro bidimensional se hace uso de la transformacion de Huang.

Parametros:

n : Orden del filtro (se especifica el numero de regiones. los filtros obtenidos son cuadrados. es decir de dimension N x N)

Tipo : Especifica la forma de la region de transicion a utilizar. los tipos con que cuenta el sistema son :

```
* Forma de Region de Transicion */

        RAISED_COSINE 0
        LINEAR        1
        SND_ORDER     2
        TRD_ORDER     3
        FTH_ORDER     4
```

FP : Un valor real en Hertz (Hz) en el rango [0.0 , 0.5]. FP es el límite hasta donde existe la banda de paso (Pass Band). Se proporcionan estos valores pues los metodos de diseño trabajan sobre valores normalizados.

FS : Un valor real en Hertz (Hz) en el rango [... , 0.5]. FS especifica el punto a partir del cual empieza la region de supresion o de Paro (Stop Band). Se proporcionan estos valores pues los metodos de diseño trabajan sobre valores normalizados.

Tipo de Salida : FILTRO

6.2.14. Bloque de Procesamiento : SMPH

Nombre Completo : DISEÑO DE FILTRO PASO-BAJAS MEDIANTE EL METODO DE MUESTREO EN FRECUENCIA Y TRANSFORMACION DE HUANG

Por medio de este bloque es posible obtener un filtro bidimensional paso-bajas, de tal forma que puede aplicarse con ayuda del bloque de multiplicacion a imagenes.
El método de diseño utilizado es por muestreo en frecuencia. Una vez que se tiene el filtro se aplica la transformacion de Huang para obtener una representacion bidimensional.

Parámetros:

N : Orden del filtro (se especifica el numero de regiones, los filtros obtenidos son cuadrados, es decir de dimension N x N)

FP : Un valor real en Hertz (Hz) en el rango [0.0 , 0.5]. FP es el límite hasta donde existe la banda de paso (Pass Band). Se proporcionan estos valores pues los metodos de diseño trabajan sobre valores normalizados.

CDC : Especifica si el filtro a obtener tiene componente en directa.

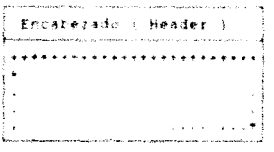
```
/* EXISTE COMPONENTE DE DIRECTA ? */
        DC              0
        NO_DC           1
```

Tipo de Salida : FILTRO

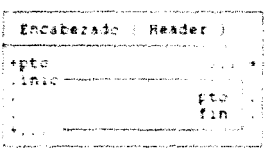
6.2.15. Bloque de Procesamiento : INIM

Nombre Completo : LECTURA DE IMaGENES CON FORMATO DIFERENTE

A partir de este modulo es posible realizar la lectura de imagenes provenientes de otros sistemas. Las imagenes provenientes de otro sistema y que se deseen leer por medio de esta opcion deben de contar con la siguiente estructura logica :



Debido a que m y n pueden ser valores grandes (por ejemplo 1024), se ha agregado un procedimiento de tal forma que solo se accese una parte de la imagen a modo de ventana. Esta situacion se ilustra con el siguiente diagrama:



El usuario debe de proporcionar los parametros para que el sistema tome la informacion adecuada.

Parametros:

ENTRADA : nombre del archivo que contiene la imagen (incluyendo la extension)

HEADER : longitud en bytes de la cabezera (header) que posee la imagen externa

N_RENG : numero de renglones con que esta compuesta la imagen

N_COL : numero de columnas que posee cada renglon

X_INIC : renglon a partir del cual se deben de leer datos

Y_INIC : columna inicial para lectura de datos

X_FINAL : renglon limite para lectura de datos

Y_FINAL : columna limite para lectura de datos

7. EJEMPLOS DE OPERACION DEL SISTEMA

7.1. Introduccion

El presente capitulo tiene como proposito la explicacion de la operacion del sistema. A traves de ejemplos se explica como modelar y ejecutar las estrategias generadas por el usuario.

El capitulo inicia con una explicacion de la interfaz del sistema con el usuario. Se hace la indicacion de las areas en las que se divide a la pantalla. Todas las areas indicadas son explicadas en detalle en el capitulo de especificacion del sistema.

Una vez hecha la anterior se procede a mostrar ejemplos de configuracion de diferentes estrategias. Se inicia con un ejemplo sencillo que permite el acceso de una imagen por el sistema. El siguiente ejemplo presenta una estrategia que involucra tecnicas clasicas para procesamiento de imagenes. Despues se introduce un ejemplo para configurar un filtro bidimensional. El siguiente ejemplo discute el filtrado de una imagen haciendo uso de filtros bidimensionales como el del ejemplo anterior. El ultimo ejemplo consiste en la combinacion de tecnicas de procesamiento clasico con tecnicas de filtrado bidimensional.

7.2. Interfaz Hombre-maquina

Como quedo establecido en la especificacion del sistema, la interfaz hombre-maquina utilizada se fundamenta en el uso de un ambiente grafico en el que se modelan las estrategias que se desea probar. La especificacion de las estrategias se realiza por medio de la tecnica de diagramas de flujo, esto es, se colocan los bloques de procesamiento que se desea utilizar y se procede a conectarlos definiendo el orden de ejecucion.

En la siguiente pagina se presenta la interfaz hombre-maquina utilizada. Como se puede ver al centro se tiene un icon que simula al mouse. La eleccion de cualquiera de las opciones del sistema se realiza por medio del mouse. En la parte izquierda de la pantalla se localiza el conjunto de comandos que pueden ser realizados por el sistema. Para seleccionar cualquiera de los comandos basta con posicionar el mouse sobre el area en que aparece la leyenda y presionar la tecla derecha del mouse.

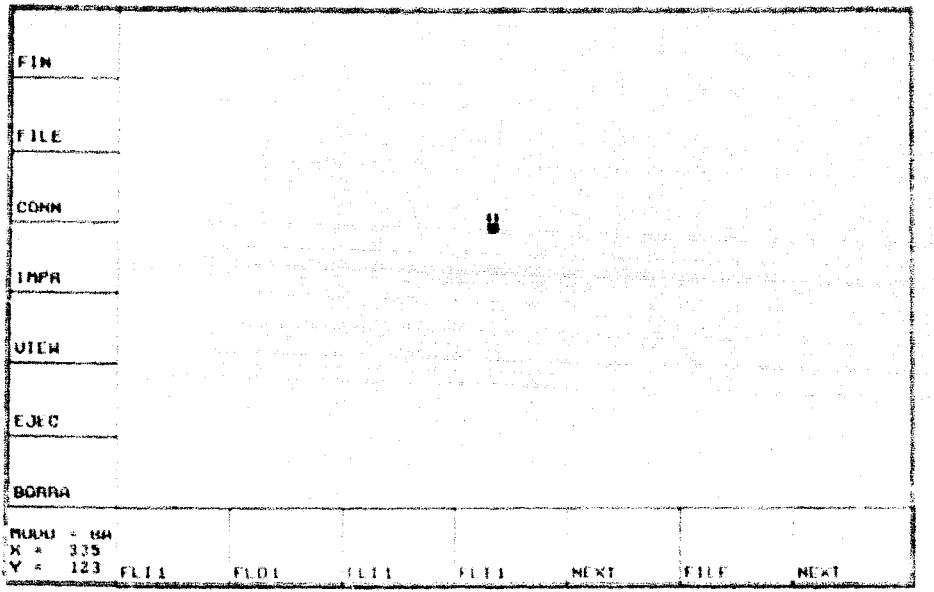


fig 1 Interfas Grafica del Sistema SIMAU

El ultimo recuadro que se encuentra hasta abajo en la parte izquierda presenta el estado de operacion del sistema. Al lado derecho de la palabra modo se puede encontrar una palabra de dos caracteres. Esta palabra me indica bajo que modo esta operando el sistema. La explicacion de los diferentes modos con que cuenta el sistema se encuentra en el capitulo de especificacion. Aqui se mencionan los diferentes modos con que cuenta el sistema:

BA - Bloque Activo
CN - Conexion
VS - Vista
BO - Borra

El modo determina que tipo de accion se desea realizar sobre cualquiera de los bloques definidos en la estrategia. Por ejemplo si el modo BA esta activo, esto quiere decir que puedo dar de alta bloques de proceso o modificar sus parametros. Si el modo es CN esto indica que puedo conectar a los bloques de proceso. Si modo es BO entonces puedo borrar bloques de procesamiento.

Abajo de la palabra modo aparecen las coordenadas en las que esta posicionado el mouse (conforme se mueve el mouse estos valores van cambiando).

La parte inferior de la pantalla presenta los bloques de procesamiento que tengo disponibles. Los bloques de procesamiento estan agrupados en familias de procesamiento. Para cambiar de familia de procesamiento se debe posicionar el mouse sobre el bloque mas a la derecha y oprimir el boton derecho (donde aparece la leyenda NEXT) del mouse varias veces hasta que aparezca la familia que se desea utilizar (las familias y bloques de procesamiento disponibles se encuentran explicadas en capitulos anteriores). Una vez que se tiene la familia se procede a seleccionar el bloque que se desea utilizar.

7.1. Ejemplo 1.- Lectura de una imagen

Este ejemplo tiene el proposito de mostrar los pasos que se deben realizar para definir, ejecutar y finalmente desplegar los resultados de una estrategia.

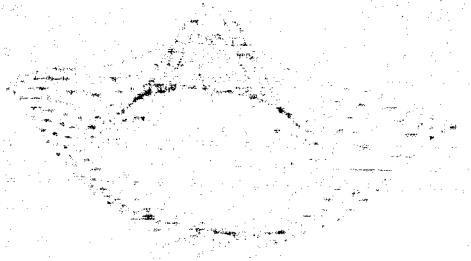
El primer paso consiste en entrar al sistema. Para lograrlo desde el sistema operativo es necesario teclear SIMAG (que significa sistema de imagenes):

C:>simag <RTN>

Aparece la pantalla de presentacion del sistema (figura no. 2).

Se teclea RTN y se pasa a la interfaz hombre-maquina del sistema (es la mostrada en la figura no. 1). En este primer ejemplo se desea

realizar la lectura de una imagen y visualizarla. Para lograrlo se selecciona el bloque de procesamiento Fill y se pone en el área de trabajo presionando la tecla derecha del mouse. El resultado se muestra en la figura no. 1 .



SISTEMA DE IMAGENES
Desarrollado por gil tapia

Teclee cualquier tecla para continuar ()

fig 2. Presentación del Sistema SIMAG

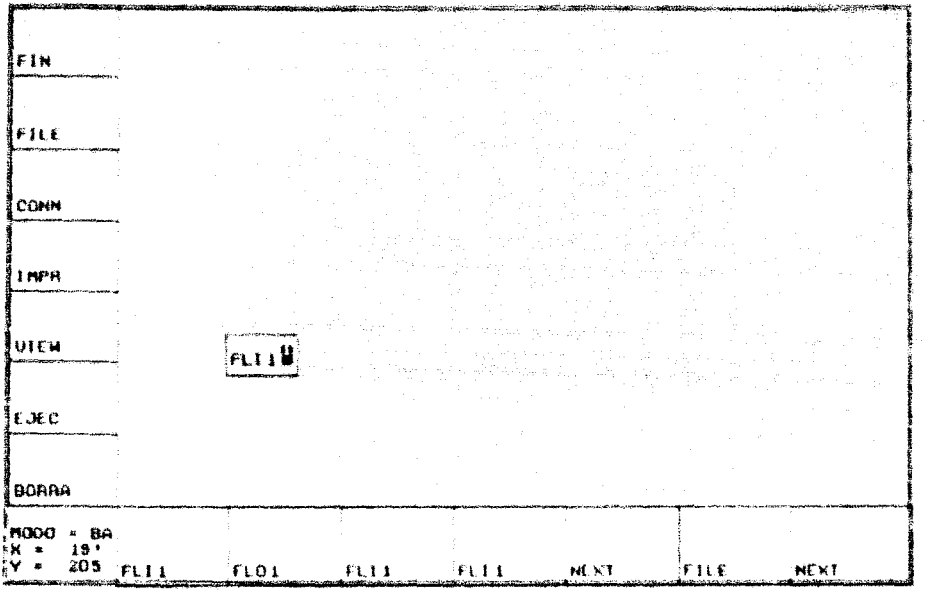


fig. 3 Inserción de un Bloque de Procesamiento

Una vez que se ha hecho lo anterior se procede a proporcionar los parametros al bloque de procesamiento. Para lograrlo se posiciona el mouse sobre el bloque de procesamiento y se presiona el boton derecho. Se presenta la figura no. 4. Como se puede ver aparece un menu para introducir los parametros.

En este momento mediante el teclado se indica que se desea una imagen temporal(es decir, al finalizar el sistema no se almacena la lectura de la imagen). El sistema solicita los parametros. En este caso el nombre del archivo donde se encuentra la imagen. La situacion se presenta en la figura no. 5.

El paso que sigue es solicitar que la estrategia sea ejecutada. Se posiciona el mouse sobre el comando EJEC y se presiona la tecla derecha. En ese momento aparece la figura no. 6 . Para continuar se presiona la tecla RTN. A partir de ese momento el sistema inicia la ejecucion de la secuencia. Una vez que concluye aparece la imagen de la figura no. 7 . Se presiona RTN para continuar.

Para visualizar el resultado se selecciona el comando VER . Una vez que se ha seleccionado todo debe indicar que es OK . Se posiciona el mouse sobre el bloque de procesamiento PIII y se presiona la tecla derecha. Aparece la figura no. 8. Este es un menu de opciones. La opcion se hace con el teclado marcando con una X la opcion que se desea hacer. La opcion deseada es imagen salida. Despues de cierto tiempo aparece la figura no. 9. Aqui se presenta la imagen leida y el histograma asociado. Se presiona cualquier tecla para continuar y se regresa al sistema.

Ahora bien supongase que se desea almacenar la estrategia generada. En este caso se selecciona el comando FIN. Aparece la figura no. 10. Este es un menu de opciones. Dado que se quiere almacenar entonces se oprime la tecla A de archivar.

Para salir del sistema (hacia el sistema operativo) se selecciona el comando FIN, si se han introducido nuevos bloques funcionales o se ha hecho algun cambio a alguna estrategia ya existente(como es el caso), aparece la figura no. 11. En este caso se selecciona la tecla S de SI. Se retorna al sistema operativo.

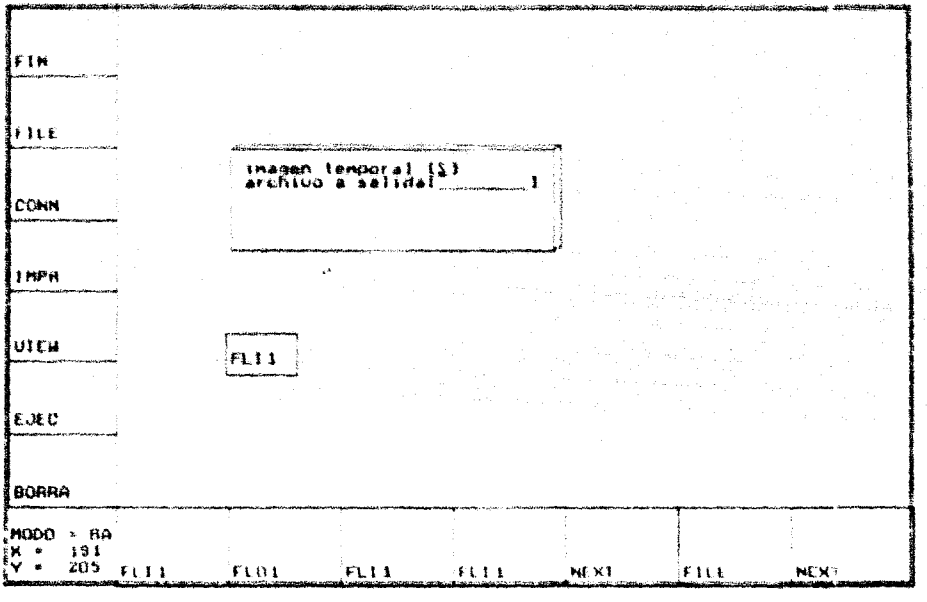


fig 4 Menú para introducción de parámetros

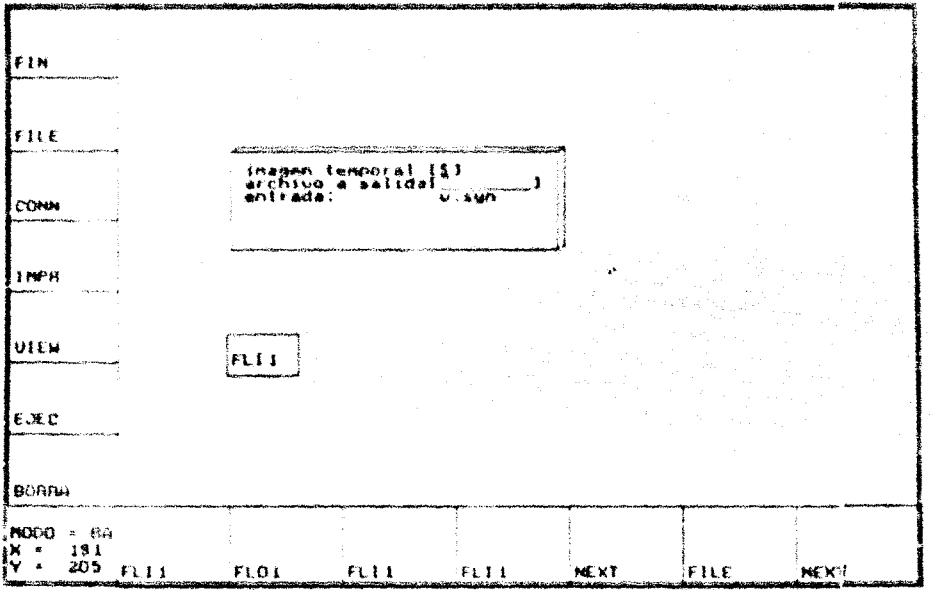


fig 5 Introducción de Parámetros del Bloque de Procesamiento

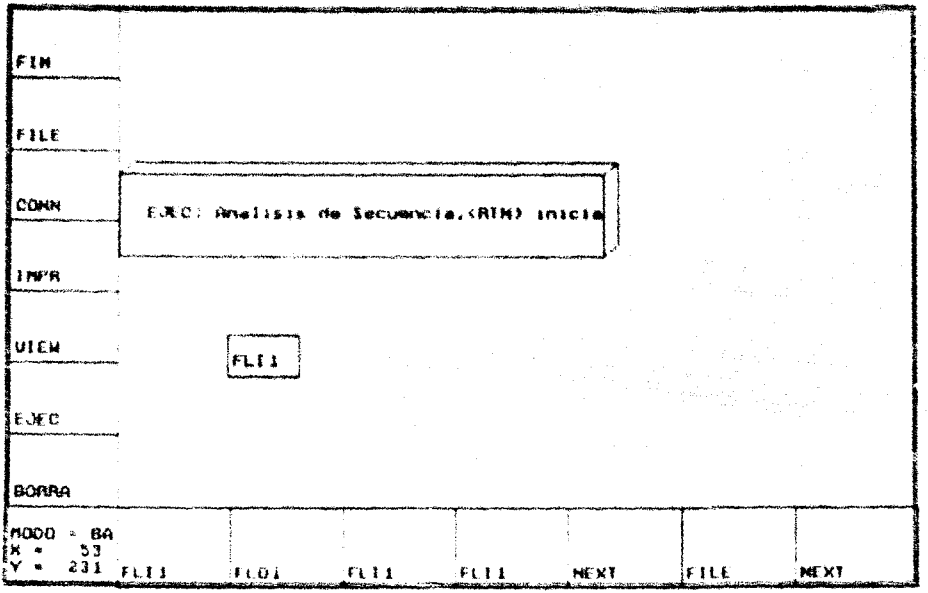


fig. 6 Solicitud para Compilación y Ejecución de la Estrategia

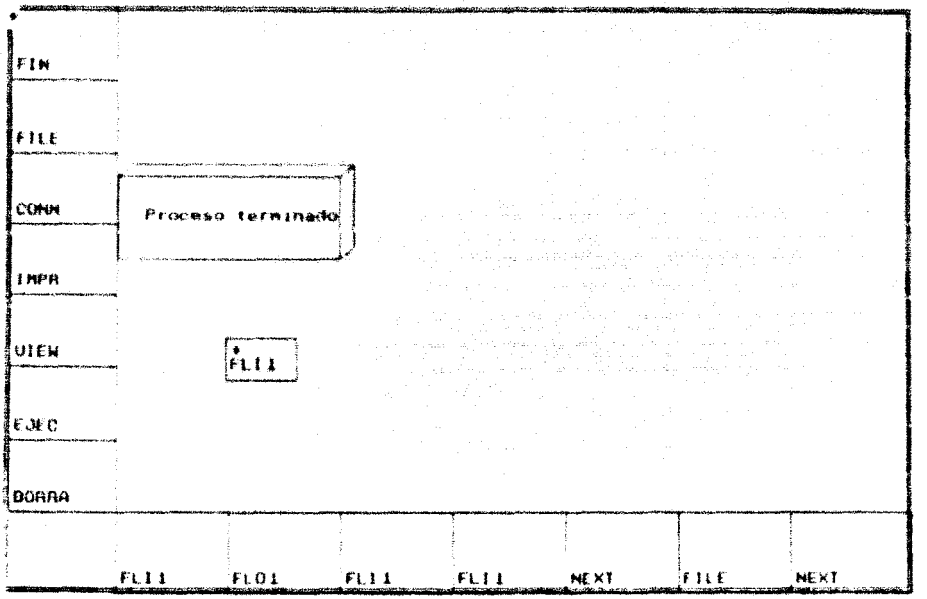


fig 7 Aviso de Finalización de Ejecución de una Estrategia

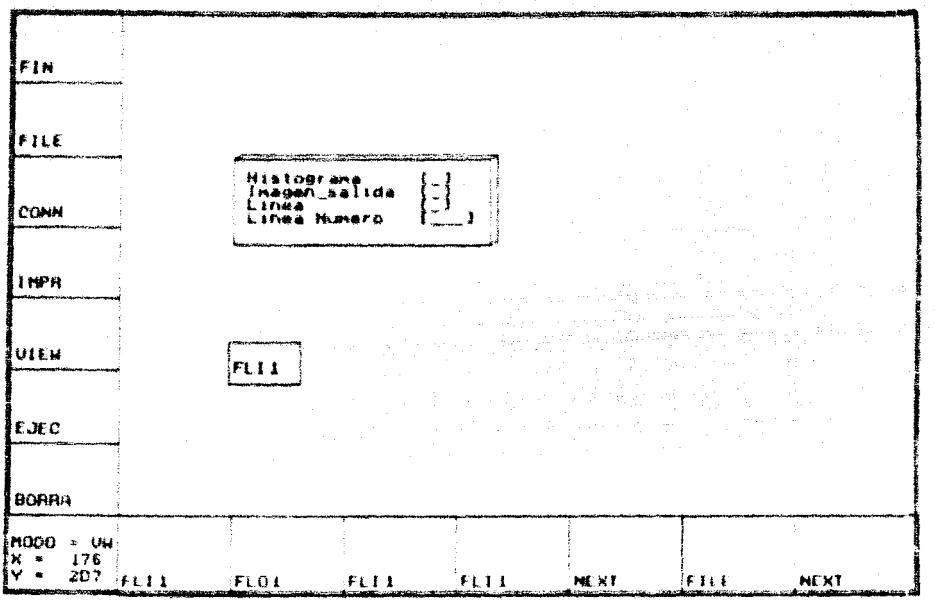
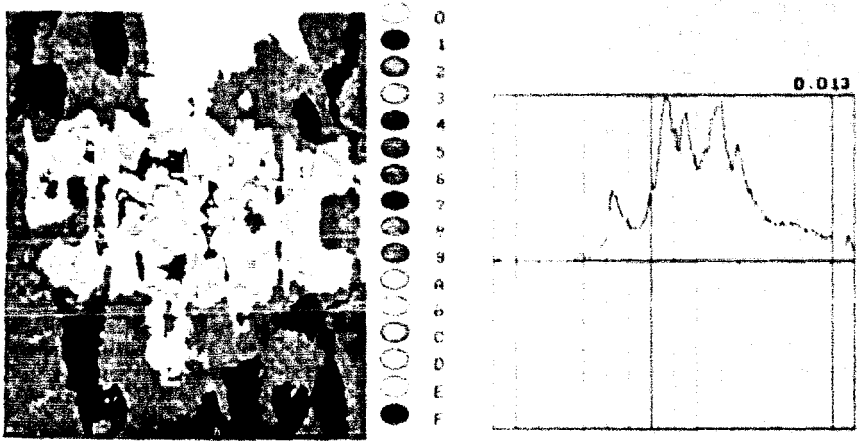


fig 8 Menú para Visualización de una Imagen



Teclee cualquier tecla para continuar ()

fig 9 Visualisación de una Imagen y su Histograma

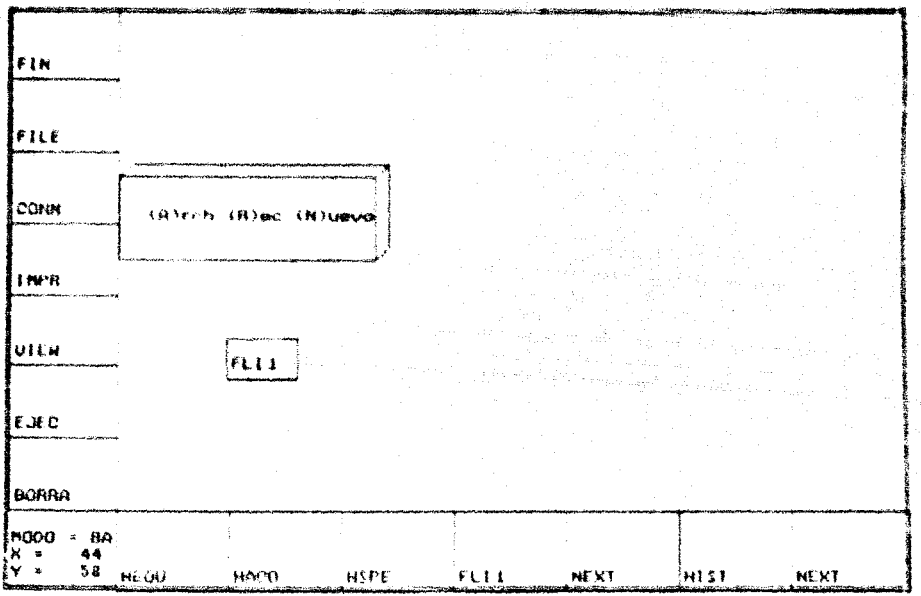


fig. 10 Menu para Archivar y Recobrar Estrategias

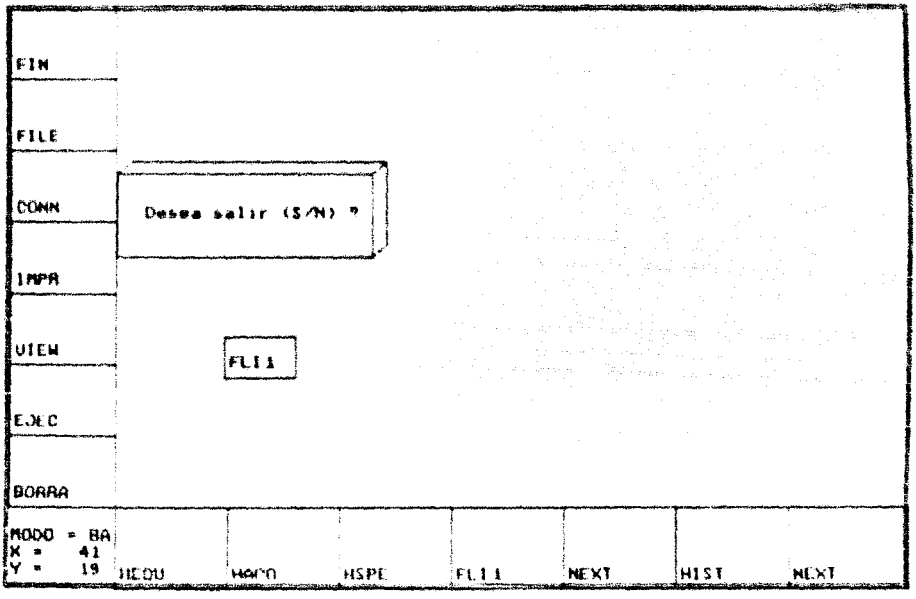


fig 11 Menú para salir del sistema SIMAG

7.6. Ejemplo 2.- Técnicas Clásicas de Procesamiento de Imágenes

El propósito de este ejemplo es mostrar los resultados que se pueden obtener por medio del sistema. En este caso específico se utilizan técnicas de procesamiento que involucran el manejo del histograma. Para observar los resultados se agrega a cada fase de procesamiento un bloque de acotamiento con el fin de que los cambios sean más aparentes.

La estrategia radica en aplicar a una imagen de entrada (Fig) diferentes técnicas con el fin de realzar rasgos de la misma. En este caso específico se ha decidido utilizar el operador Laplaciano (LAP) para realizar cambios en la tonalidad. Después se aplica la igualación de histograma (HEQ) con el fin aplanar el histograma de la imagen. Después se aplica la técnica de especificación de histograma (HSP), la forma de histograma deseado es para obtener el negativo de la imagen observada.

La estrategia arriba planteada se muestra en la figura No. 10, las figuras que siguen muestran los resultados de la evaluación de la estrategia.

Como se puede apreciar, el uso de la técnica de acotamiento permite observar como realzan la técnicas de procesamiento a la imagen original (Nota: el valor de umbral utilizado es 200).

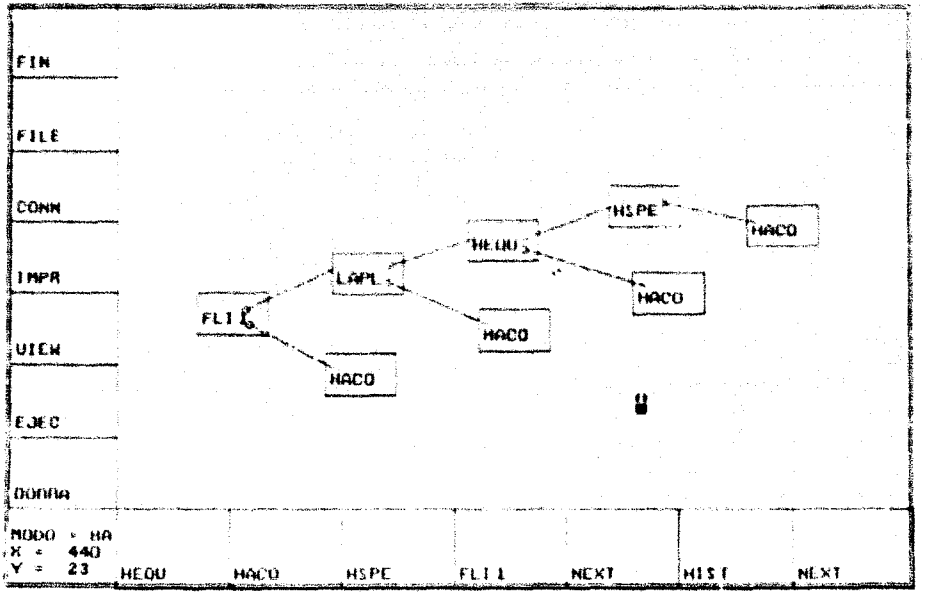


fig 12 Estrategia de Ejemplo No 2

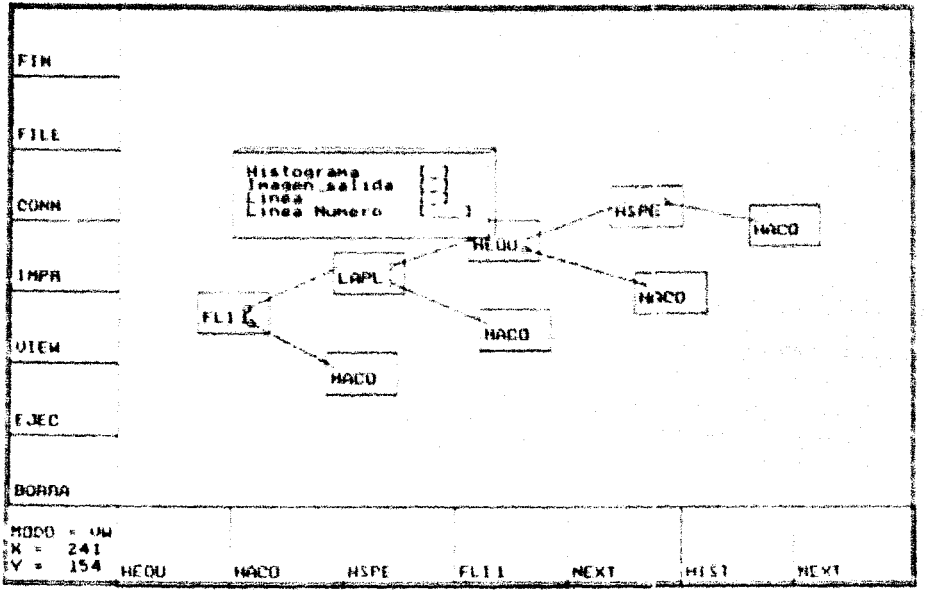
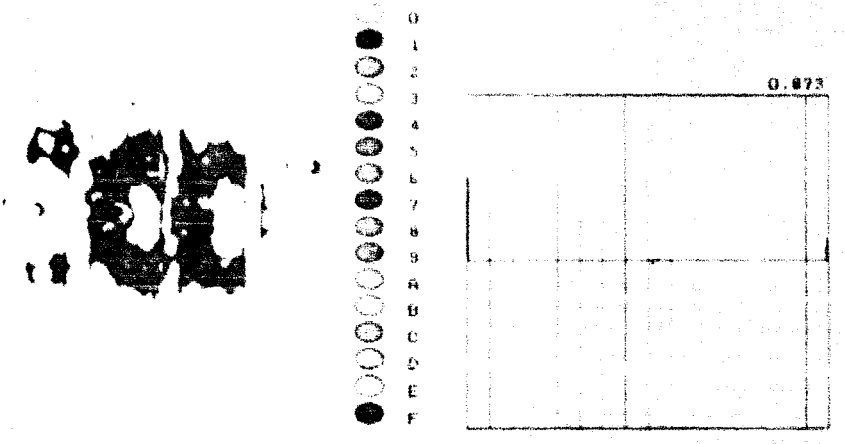
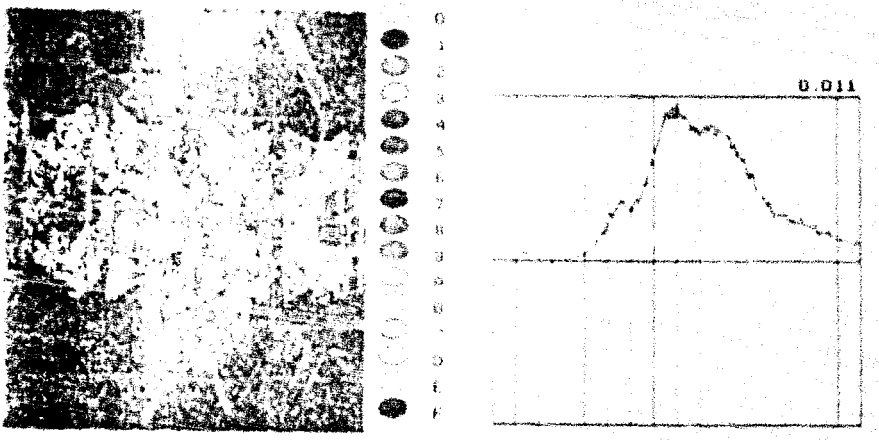


fig 13 Solicitud para Visualizacion de Imagen Resultante



Teclee cualquier tecla para continuar ()

fig. 14. Imagen Resultante de Acotamiento a Imagen Original

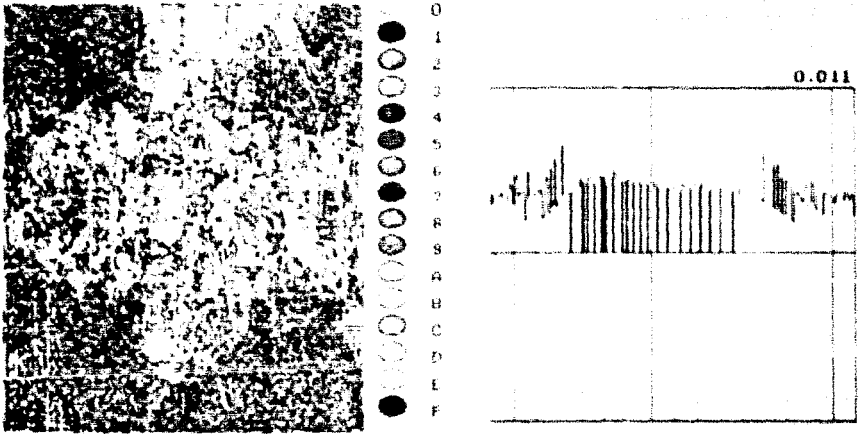


teclee cualquier tecla para continuar (.)

fig 15 Imagen Resultante de Laplaciano (0.1.5) a Imagen Original

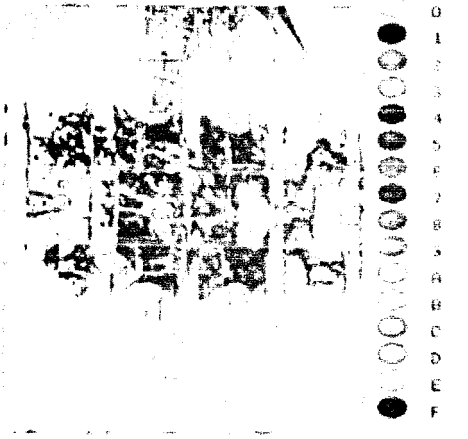
Teclee cualquier tecla para continuar ()

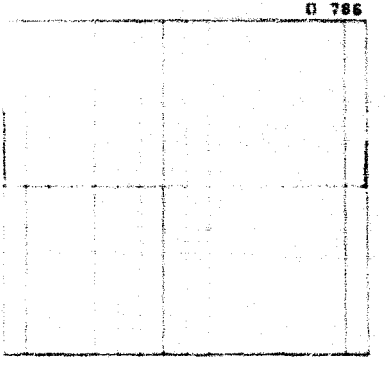
Fig. 19 [illegible] a salida de impresora



Teclee cualquier tecla para continuar ()

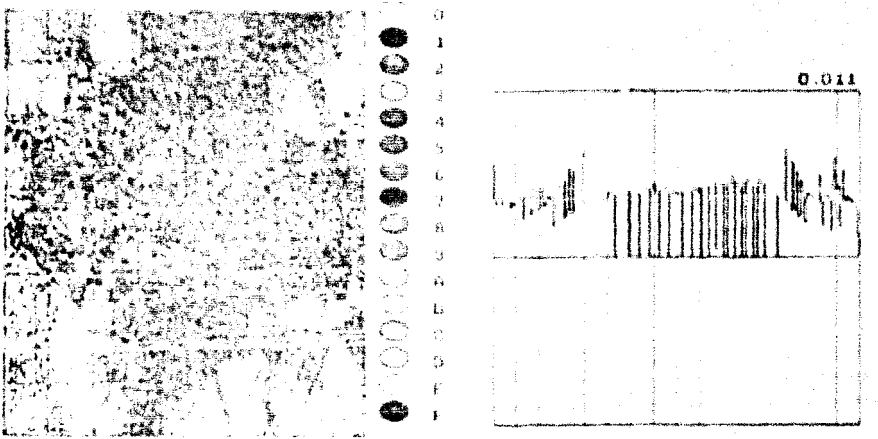
fig 17 Imagen Resultante de Igualación de Histograma a salida de Laplaciano





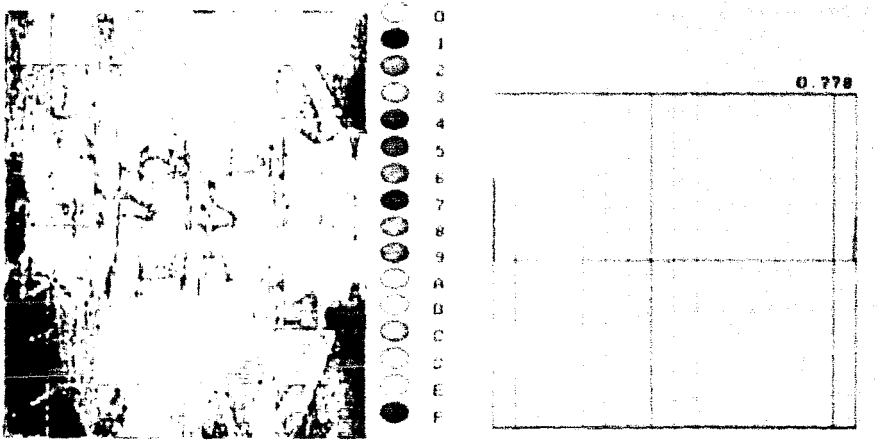
Teclee cualquier tecla para continuar ()

Fig 18 Imagen Resultante de Acotamiento a Salida de igualación de Histograma



Teclee cualquier tecla para continuar ...

Fig. 14 Imagen Resultante de Especificación de Histograma a Salida de Igualación de Histograma



Teclee cualquier tecla para continuar (>

fig. 20 Imagen Resultante de Acotamiento a salida de Especificación de Histograma

3.5. Ejemplo 3.- Diseño de un filtro bidimensional pasa-baja

Este ejemplo muestra el diseño de un filtro pasa-bajas bidimensional. El método de diseño utilizado es por ventana circular. Se hace uso de la transformación de Huang (bloque twHU). La longitud del filtro es 11 y la frecuencia de paro es 0.35 Hz(normalizada).

La figura de la siguiente página ,ilustra la ventana (figura no. 21).

Una vez que se ha especificado el filtro y sus parametros se procede a su ejecución. Dentro de la ejecución ha sido posible proveer a la visualización de los resultados. Se selecciona el bloque VIEW y se posiciona el mouse sobre el bloque twHM. Aparece la figura no. 22. Se escoge por medio del teclado que representación se desea. Se oca una x. Los resultados obtenidos se muestran en las figuras no. 23 y no. 24.

La primera figura corresponde a la representación espacial del filtro. La segunda figura es la representación en el dominio de la frecuencia (w1 y w2). Se hizo esta representación en proyecciones pues ayuda mucho al usuario a visualizar como es el filtro.

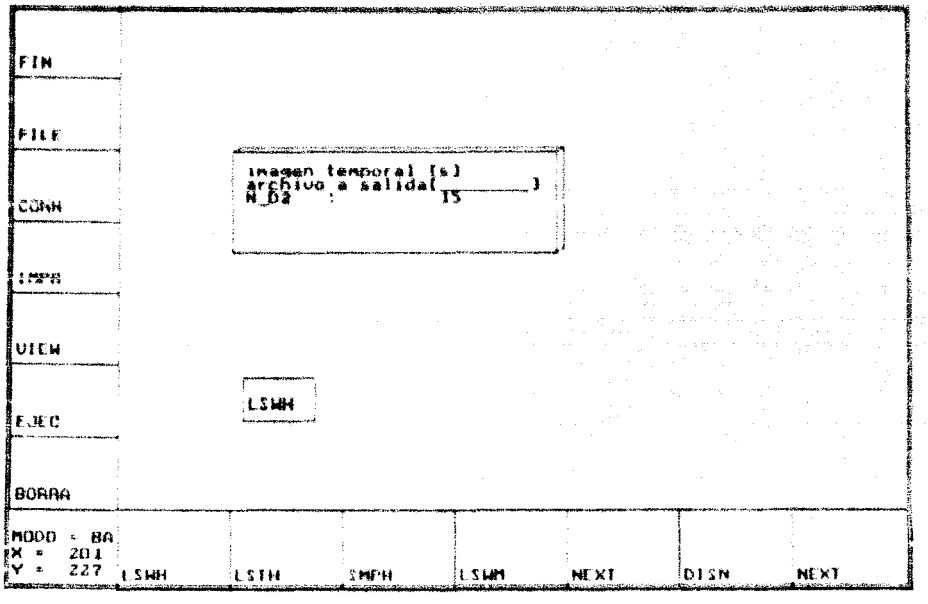


fig 21 Bloque para diseño de un Filtro Paso baja Bidimensional

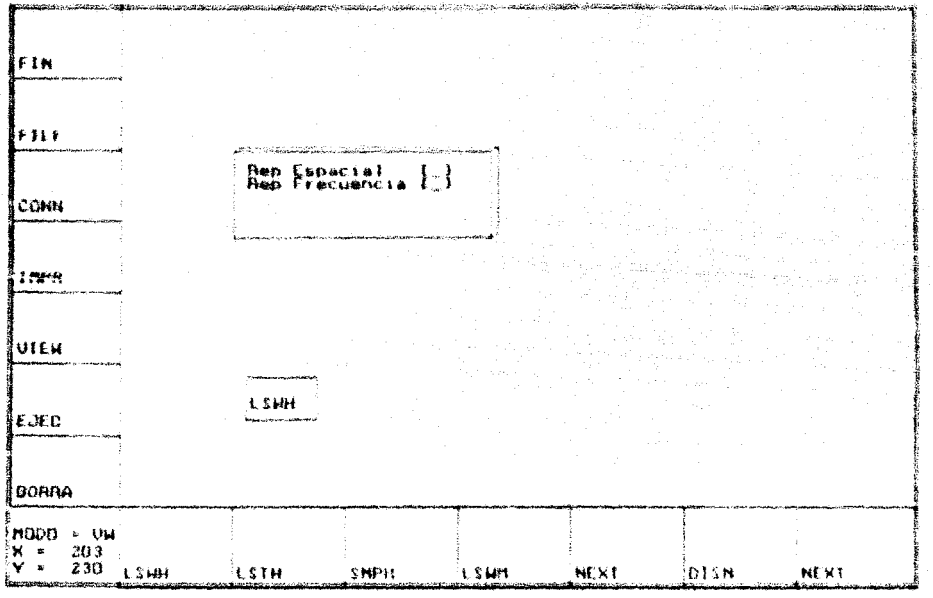


fig 22 Solicitud para Visualización de títulos

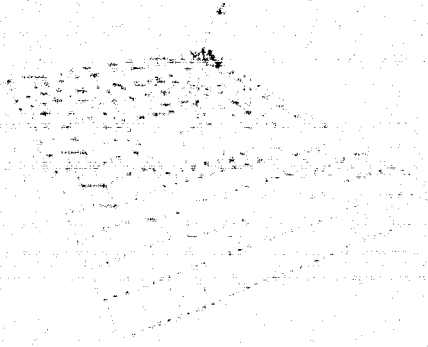


fig. 23 Representación Espacial del filtro Paso bajas del Ejemplo 3.

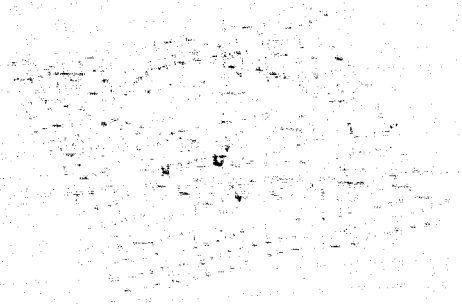


Fig. 24. Representación Frecuencial del filtro Paso baja del Ejemplo 3

7.6. Ejemplo 4.- Diseño de un filtro Paso-alta Bidimensional

En el ejemplo anterior se mostro como diseñar un filtro paso-bajas bidimensional. En este ejemplo se muestra como diseñar un filtro paso-altas.

Este ejemplo es de gran importancia pues muestra la aplicacion de la metodologia para usar el algebra de filtros propuesta en capitulos anteriores. En esta metodologia se propone que a partir de filtros paso-bajas es posible obtener filtros de diferentes tipos (en este caso un filtro paso-altas). Esta idea no es nueva pues es un resultado del area de procesamiento de señales, lo que se desea resaltar es que el sistema cuenta con esta habilidad , ya que mediante el algebra de filtros implantada se puede aplicar este enfoque.

Para realizar este filtro (paso-altas) es necesario aplicar el modelo propuesto en la parte de filtros bidimensionales. Este modelo propone que a partir de la relacion matematica:

$$h_{hi}(m,n) = \delta(m,n) - h_{lo}(m,n)$$

es posible obtener un filtro paso-alta.

Este modelo se presenta a modo de estrategia dentro del sistema SIMAS. El modelo se presenta en la figura no. .

Verificando el significado de cada bloque se puede ver que el resultado es una diferencia de filtros. En este caso es cuando es util usar el bloque IMPI que es la funcion impulso en 2 dimensiones.

Como se puede ver, el bloque LPWH corresponde a un metodo para la obtencion de un filtro paso-baja.

El operador GANF multiplica el filtro de entrada por una ganacia proporcionada por el usuario. En este caso la ganancia es -1 con lo que se obtiene el negado del filtro.

El bloque SUMF obtiene la suma de los filtros de entrada (en este caso una resta por el uso de GANF).

Después de definir parametros (el filtro LPWH es especificado con los siguientes valores :

N = 15
Tipo = RECTANGULAR
Frecuencia de corte = 0.3

) y ejecutar la estrategia se obtienen los siguientes resultados.

Se muestran en las siguientes figuras los filtros resultantes tanto en el dominio espacial (2 dimensiones) como en el frecuencial .

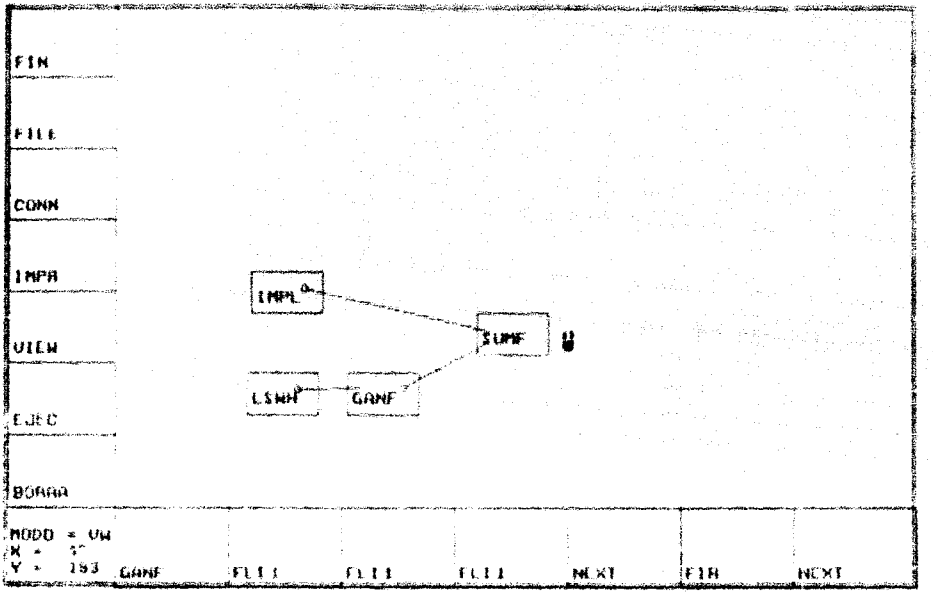


Fig 25 Configuración para obtención de filtro paso alto

fig 26. Representación espacial de filtro LSWK(pasa-baja)

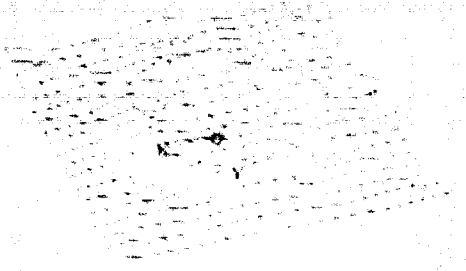


Fig. 27. Représentation fréquentielle du filtre Laplacien

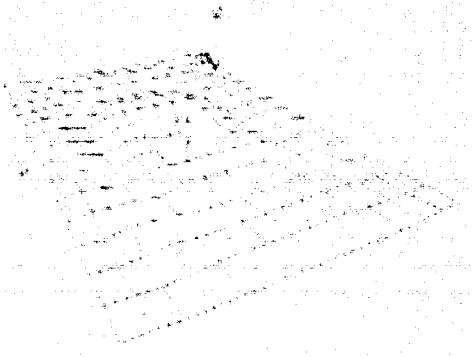


fig 28 Representación espacial de filtro SOBEL(paso-alta)

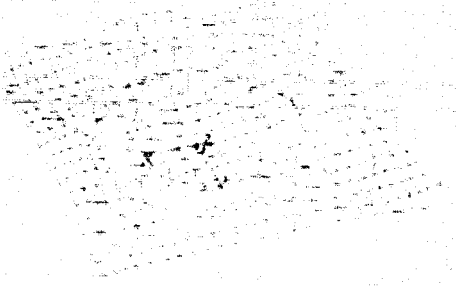


fig 29 Representación frecuencial de filtro [illegible]

7.7. Ejemplo 5.- Utilizacion de Procesamiento Clasico y Filtrado bidimensional

En este ejemplo se presentan la combinacion de técnicas clásicas de procesamiento de imagenes combinadas con técnicas de filtrado bidimensional.

Como se puede ver en la figura, se tiene una imagen de entrada a la cual se le aplican diversas técnicas de procesamiento de imagenes. El objeto es realzar cambios bruscos en las intensidades de la imagen.

A partir de la utilizacion de las técnicas de filtrado usadas y de las técnicas clasicas es posible obtener resultados interesantes pues estas técnicas se complementan unas con otras. Los resultados se muestran en las siguientes figuras.

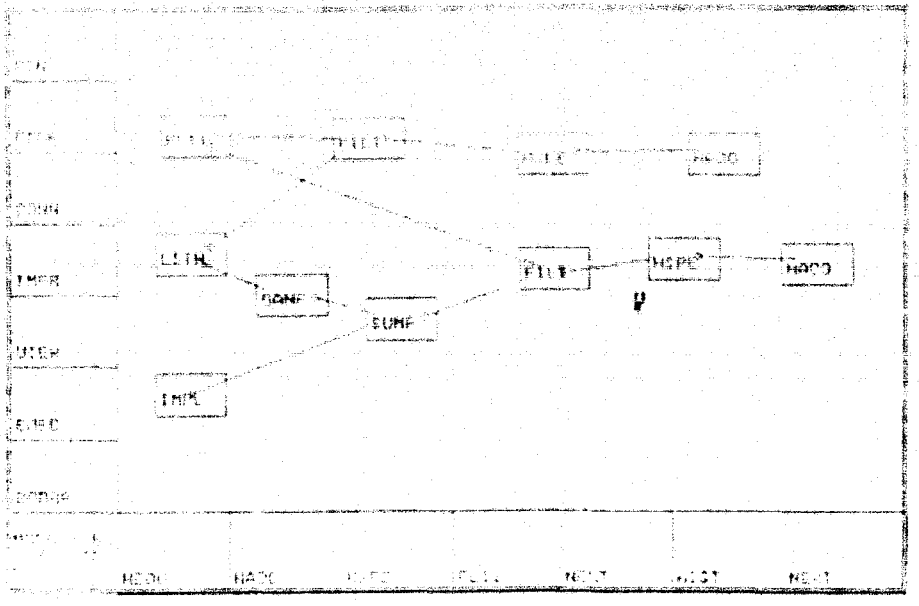


fig. 30 Estrategia del Ejemplo 5

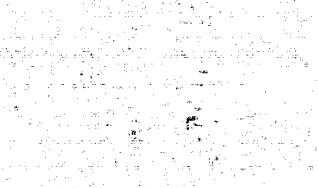
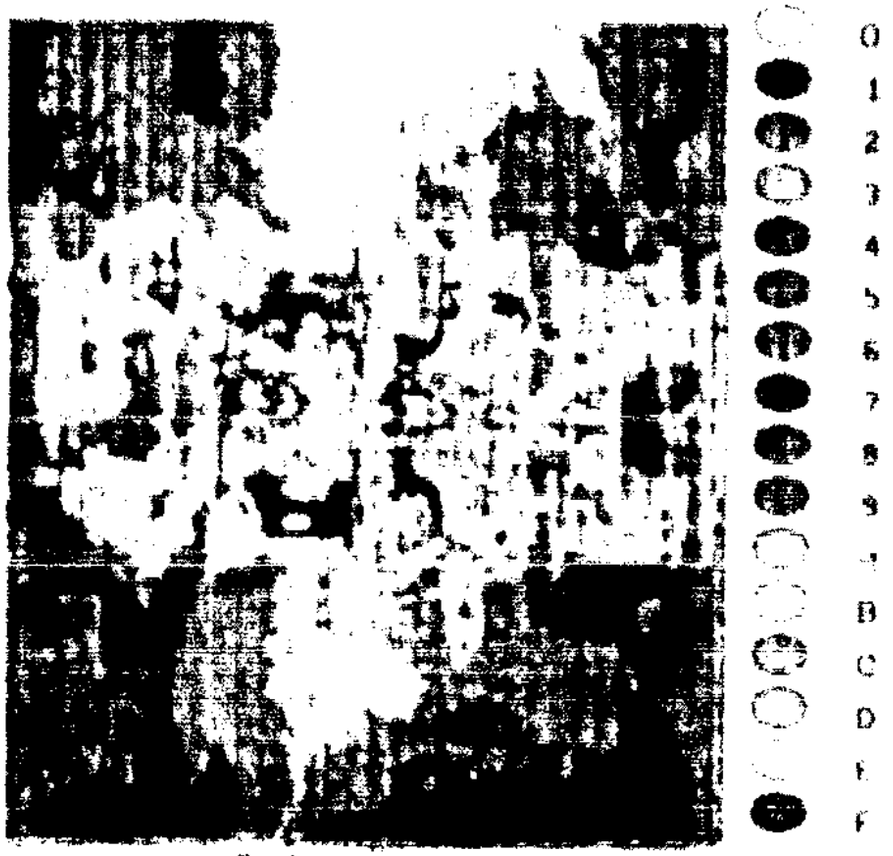
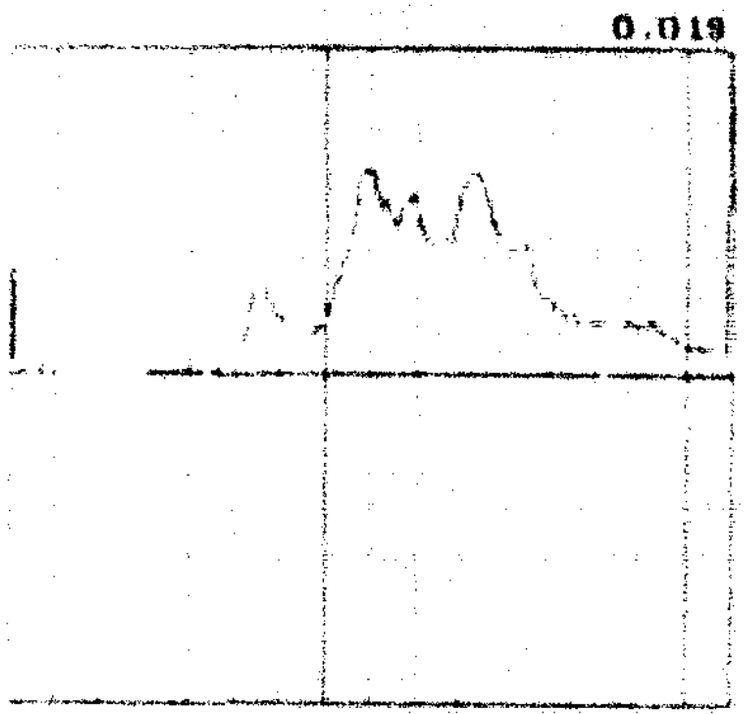


fig. 11 Filtro Paso Bajas aplicado (Representación en frecuencia)




0.019

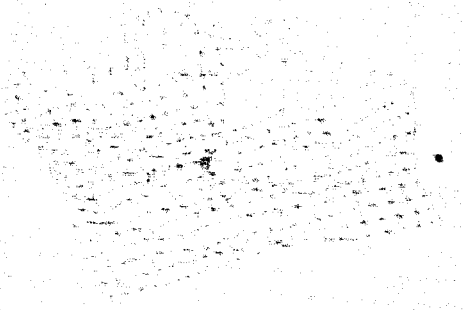
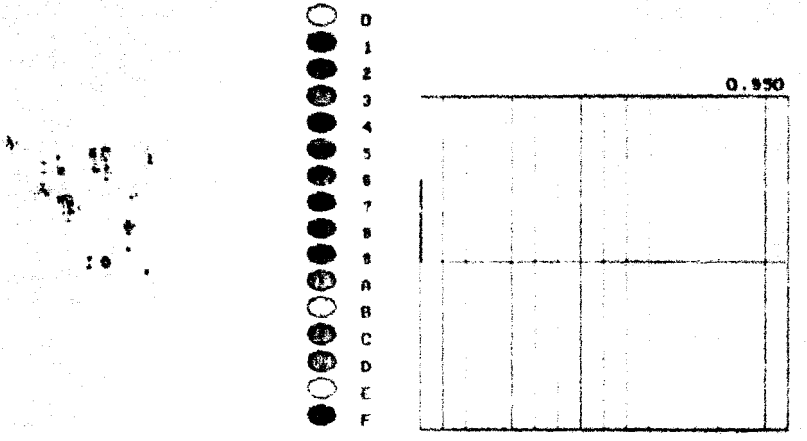


fig. 33 Filtro Paso Altas aplicado (Representación en frecuencia)



Pulse cualquier tecla para continuar ()

Fig. 34 Imagen original filtrada por Filtro Paso Altas anterior

8. Conclusiones

A continuación se proporcionan las conclusiones sobre el trabajo desarrollado, así como mejoras que podrían ser adicionadas al sistema.

El uso de técnicas de diseño ayudado por computadora, así como el de otras áreas de computación contínuadas permiten utilizar al computador como un medio que apoye el desarrollo de nuevos productos.

El uso de un editor de estrategias para plasmar un modelo propuesto, permite al usuario especificar claramente la interrelación que existe entre los diversos procesos que se desarrollan. Este editor de estrategias podría ser usado para otras áreas, que no solo pertenecen al procesamiento de imagenes. Dentro del mercado existen diversos paquetes que explotan este tipo de conceptos, y que permiten al usuario expresar sus requerimientos. Ejemplos de estos paquetes son por ejemplo FARAGON y GENESIS, sistemas orientados hacia el control de procesos industriales. Por otro lado existen otros paquetes que están orientados ha explotar la ingeniería de software ayudada por computadora (CASE-Computer Aided Software Ingineering). Ejemplos de estos sistemas son Anatool y PowerTools. Estos sistemas se encargan de apoyar el diseño y análisis de sistemas utilizando los conceptos de Yourdon y DeMarco.

Una de las ventajas principales de usar este tipo de esquemas, es que se puede contar en todo momento con la documentación del sistema planteado, pues la especificación del sistema lo conforma la estrategia en si misma.

El uso del concepto de medios ambientes para visualización por prototipos constituye una herramienta poderosa para el desarrollo de sistemas orientados al diseño ayudado por computadora.

Ahora bien, el contar con la posibilidad de ejecutar la estrategia planteada viene a completar el ciclo de simulación y prueba de un modelo.

Con estos elementos y con las herramientas para consultar los resultados obtenidos , el usuario cuenta con todo un ambiente que le proporciona la capacidad de idear y probar diferentes configuraciones.

Otra aplicación importante para este sistema es de tipo didáctico, ya que el usuario puede plantear estrategias básicas consistentes de un solo bloque de procesamiento y una entrada, correr la simulación y visualizar los resultados. Una vez que se ha entendido los conceptos básicos puede avanzar para experimentar con estrategias mas complicadas.

Al comparar los resultados de las diferentes tecnicas de procesamiento de imagenes que fueron implantadas es claro que un buen paradigma de desarrollo consiste en combinar estas diferentes tecnicas. En especial, el enfoque de hacer uso de tecnicas de procesamiento digital de señales me parecio muy atractivo pues aparte de que goza de una amplia base teorica que la soporta, se puede pensar en extensiones en su aplicacion como lo muestra el uso del Algebra de filtros para detectar fallas de otros tipos como las parrillas.

Por otro lado, es evidente que el uso de diversas tecnicas y metodologias para el desarrollo de este trabajo implico una gran atencion a articulos relacionados a estos temas.

Pienso que una parte muy importante dentro de este desarrollo consistio en dar organizacion y orden a las ideas y material recabado. La integracion de un sistema (aparte de la implantacion) constituye una parte muy importante dentro del desarrollo de [illegible], pues es donde se da forma y orientacion al trabajo realizado.

A continuacion se indican posibles extensiones que podrian mejorar el desempeño del sistema.

-El uso de memoria expandida permitiria el manejo de imagenes de mayores dimensiones, aunque existen tecnicas que permiten un procesamiento semejante y en las que no es necesario que se tenga a toda la imagen presente (manejo de imagenes por bloques).

- El uso de una biblioteca de simbolos para representar a cada proceso seria de gran ayuda pues permitiria al usuario entender mas rapidamente una estrategia dada. Una biblioteca de simbolos es un conjunto de simbolos graficos que representan a cada uno de los procesos, por ejemplo para representar el proceso de suma de dos señales(imagenes) bastaria que se representara con un simbolo '+' dentro del diagrama.

- Debido a que el sistema fue desarrollado en turbo-C, conforme fueron creciendo las capacidades del sistema, esto requirio una mayor cantidad de memoria utilizada, es por esto que al requerirse el uso de tecnicas como manejo de codigo por overlays, se averiguo que turbo-C no posee esta facilidad. Se sabe que existe una nueva version de turbo-C que es llamada C++ Professional, esta nueva version posee la caracteristica de que si puede manejar overlays.

El enfoque dado al sistema para manejo de tecnicas de tesis no impide en ningun momento que se puedan desarrollar nuevos bloques de procesamiento que permitan explotar otras areas del procesamiento de imagenes, de hecho el sistema fue conceptualizado con esa finalidad.

En general debido a la cantidad de recursos de computo y de memoria que requiere el sistema, si se deseara implantar tecnicas que

requirirán mayores recursos. la recomendación sería la de portar el código a ese sistema. Debido a que el código esta escrito en 'C', un sistema operativo que posee gran potencia y con el que se cuenta con varias herramientas para ambientes gráficos (Xwindows) es el sistema operativo UNIX.

Una conclusión personal dentro del proceso de desarrollo de este sistema consiste en buscar y apoyarse en la documentación que se especifica al final de cada trabajo para poder adquirir un mayor grado de profundidad. Es indudable que el uso de medios bibliográficos es muy necesario y constituye en si una parte muy fuerte del desarrollo pues hay que dedicar bastante tiempo en la búsqueda de artículos de interés y asimilar la información en ellos contenida para posteriormente utilizarla.

Appendix A. BIBLIOGRAFIA

[Agerwala 1982] Data Flow Systems, Tilak Agerwala y Arvind, Computer February 1982

[Benmahammed 1989] Computation of the 2-D System Frequency Response, Khier Benmahammed IEEE Transactions on Circuits and Systems, January 1989 Vol. 36 No. 1 pp. 126 - 128

[Chang 1990] Visual Languages and Visual Programming , Shy-Kuo Chang. Plenum Press

[Davis 1982] Data Flow Program Graphs, Alan L. Davis Robert M. Keller, IEEE Computer, February 1982

[Dawson 1987] Introduction to Image Processing Algorithms, Benjamin M. Dawson, BYTE March 1987

[Dertouzos 1969] On-line Simulation of Block-Diagram Systems, Michael L. Dertouzos, Martin E. Hellman, Kenneth P. Polzen , IEEE Transactions on Computers VOL C-18, No. 4, April 1969

[Dettman 1988] DOS Programmer's Reference, Terry R. Dettmann QUE Publications

[Gonzalez 1986] Image Enhancement and Restoration , R. C. Gonzalez, Handbook of Pattern Recognition and Image Processing , 1986

[Green 1989] Digital Image Processing : A Systems Approach, William B. Green, Van Nostrand Reinhold Press, 1989

[Harris 197-], Windows, Frederic J. Harris, Trigonometric Transforms : A unique introduction to FFT, Spectral Dynamics Corporation, 1977

[Hix 1989] User Interfaces : Opening a window on the computer, Deborah Hix, IEEE Software, January 1989

[Huang 1972] Two-Dimensional Windows , Thomas S. Huang, IEEE Transactions on Audio and Electroacoustics, March 1972

[Hummel 1977] Image Enhancement by Histogram Transformation , Robert Hummel, Computer Graphics and Image Processing 6 , 184-195 (1977)

[Jain 198x] Fundamentals of Digital Image Processing, Jain, Anil K., Prentice-Hall 1989

[Kelly 1961] A Block Diagram Compiler, John L. Kelly Carol Lochbaum, The Bell System Technical Journal, May 1961

[King 19xx] Digital Filtering in One and Two Dimensions:- Design and Applications, Robert King, Majid Ahsadi, Plenum Press 1989

[Lee 198?] Pipelined Interleaved Programmable DSP's: Synchronous Data Flow Programming, Edward A. Lee and David G. Messerschmitt, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol ASSP-35 No. 9 September 198?

[Lorho 1984] Methods and Tools for Compiler Construction ,- B. Lorho, Cambridge University Press.

[Mersereau 197x] Mcclellan Transformations for Two-Dimensional Digital Filtering : I - Design, Russell M. Mersereau , IEEE Transactions on Circuits and Systems, Vol CAS-23 NO 7, JULY 1976

[Miller 1989] The Official Borland Turbo C Survival Guide, Laurence H Miller y Alexander F. Quilici, Addison-Wesley

[Murphy 1985] Software Diagramming John S. Murphy y Karl G. Baike, McGraw-Hill

[Myers 1988] User-Interface Tools: Introduction and Survey, Brad A. Myers, IEEE Software January 1988

[Parks 1987] Digital Filter Design, T.W. Parks and C.S. Burrus, Wiley-InterScience, 1987

[Smith 1989] Advanced Turbo C .- James T. Smith , McGraw-Hill 1989

[Szabo 1-68] The Perspective Representation of Functions of Two Variables, B. Hubert, J. Szabo & S. Giulieri, Journal of the Association of Computing Machinery (ACM), Vol. 15, No. 2, April 1968, pp. 193-204

[Upson 1990] Tools for Creating Visions, Craig Upson, UNIX REVIEW September 1990