UNIVERSIDAD VERACRUZANA

INSTITUTO DE INGENIERIA

" Diseño e Implantación de Algoritmos para la Transformación Controlada de Imágenes Digitalizadas en Mapas de Bits "

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A :

Abelardo Rodríguez León

ASESOR: JOSE EVARISTO PACHECO VELASCO

H. VERACRUZ, VER. JUNIO DE 1996

UNIVERSIDAD VERACRUZANA
INSTITUTO DE INGENIERIA

H. Veracruz, Ver., Mayo 21 de 1996.

Al candidato al Grado señor:
ING. ABELARDO DOMINGUEZ LEON
P R E S E N T E:

En atención a su solicitud relativa, me es grato transcribir a Usted a continuación el tema que aprobado por esta Dirección propuso el C. M.C. EVARISTO PACHECO VELASCO, para que lo desarrolle como tesis, para obtener el Grado de Maestro en Ciencias de la Computación:

T E M A:

"DISEÑO E IMPLANTACION DE ALGORITMOS PARA LA TRANSFORMACION CONTROLADA DE IMAGENES DIGITALIZADAS EN MAPAS DE BITS"

I .- FUNDAMENTOS
II .- OBJETIVOS
III .- HIPOTESIS
IV .- REVISION BIBLIOGRAFICA
V .- ESCALACION DE IMAGENES
VI .- ROTACION DE IMAGENES
VII .- EFECTOS ONDULATORIOS
VIII.- TRANSFORMACION USANDO MALLA DE ESLABONAMIENTO
CONCLUSIONES Y RECOMENDACIONES

Sin otro particular, me es grato reiterarle la seguridad de mi más atenta y distinguida consideración.

A T E N T A M E N T E
"LIS DE VERACRUZ: ARTE, CIENCIA, LUZ"

DR. ENRIQUE A. MORALES GONZALEZ
DIRECTOR

Dedicatorias:

A la memoria de mi padre Abelardo Rodríguez Hernandez,
un ejemplo de lucha incansable ante toda adversidad.

A mi madre Virginia León Vda. de Rodriguez,
por mostrarme desde siempre, el camino adecuado en la vida.

A mis dos principales fuentes de autorealización,
amor y cariño: Loren y Abe.

A mis hermanas Claudia y Lizbeth y sobrinos Ely y Beto
que son tambien una parte importante en mi vida.

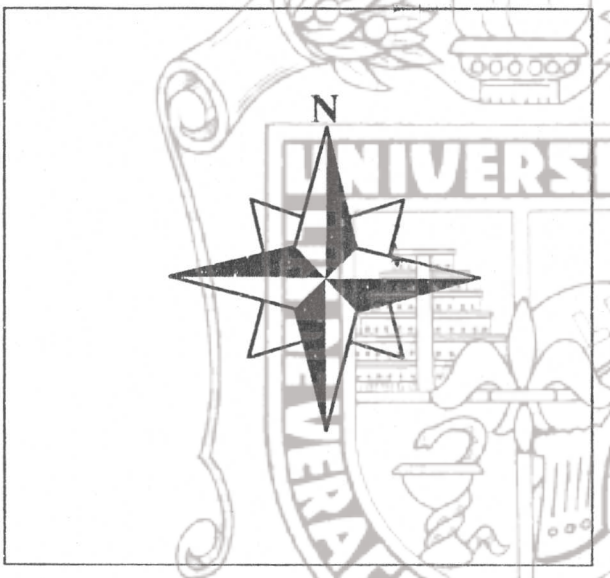
Al Dr. David Riestra Diaz, quien afino mi vocación de investigación, y de quien aprendi buena parte de lo que se.

A mi Asesor, M.C. Evaristo Pacheco Velasco, ya que sin su constante insistencia este trabajo todavia no estaria terminado.

A las personas que revisaron y aportaron a este trabajo:
M.C. Alberto Lorandi
M.C. Felipe Verdalet Guzman
Dr. Enrique Morales G.

INDICE

INDICE DE FIGURAS



Capítulo I
Fundamentos

"Siempre hay una primera vez"

I. Fundamentos

1.1. Introducción.

Las evolución que ha sufrido la información actual, es un aspecto importante a considerar, para hacer un adecuado manejo de la misma, a través de una computadora. La diversificación que se ha dado en la información es notable; pasando de la simple representación en texto, a incluir también sonidos e imágenes en aplicaciones actuales.

La inclusión de nuevos medios de transmisión se ha dado recientemente por el desarrollo del hardware computacional, que ahora incluye poderosas tarjetas de sonido que puede manejar amplios espectros sonoros, con una gran calidad. Además, los actuales sistemas incluyen tarjetas de vídeo que permiten manejar millones de colores y soportan resoluciones que antes solo tenían estaciones de trabajo para el uso de CAD

En general, las aplicaciones que usan mas de un medio para transmitir, información; reciben el nombre de aplicaciones multimedia.

A pesar que la diversidad de medios usados actualmente no es muy grande, las posibilidades de aplicar en un futuro no lejano otros medio; ha hecho este campo prometedor. En la actualidad las aplicaciones multimedia se concretan a manejar los siguientes medios: escrito, sonoro y el gráfico para transmitir información.

De estos medios, uno de los mas importantes (en general en cualquier aplicación moderna) son los gráficos. Los gráficos son en esencia imágenes, las cuales se clasífican en dos tipos: vectorizadas y mapeadas en bits.

Las imágenes vectorizadas son aquellas que se producen como una descripción numérica de los elementos básicos que forman un dibujo. Dichos elementos se encuentran interconectados de diversas formas y son llamados comúnmente primitivas de dibujo. Estas primitivas varían en complejidad, según la aplicación que genere la imagen.

Existen diversos libros sobre imágenes vectorizadas, que explican detalladamente como hacer transformaciones sobre estas imágenes, usando sus atributos matemáticos para: trasladar, rotar, escalar, colorear, iluminar, sombrear, etc. Dándole a la imagen dibujada una apariencia mas real. Este tipo de imágenes son las usadas por los paquetes de CAD.

A pesar de representar aspectos de la vida real, el origen de las imágenes vectorizadas es de naturaleza computacional, ya que se crean y transforman totalmente en la computadora.

Otro tipo de imágenes, las mapeadas en bits son producto del mundo real o de un programa de pintura. Las imágenes del mundo real son llevadas a la computadora por medio de dispositivos de digitalización como scaner, que mediante rastreo convierten cada punto de color en un valor

numérico que se almacena en una cuadricula o malla llamada mapa. Dependiendo de la cantidad de colores que se deseen, un punto de color puede ocupar 8, 16 o 24 bits, de ahí su nombre de mapa de bits.

Las imágenes generadas en programas de pintura, son creadas con una metáforas de herramientas similares a las que usa un pintor, al crear un cuadro, como son pinceles, paletas, borrador, brocha, sprite, etc. El formato en que se almacenan estas imágenes es igual al de las imágenes obtenidas mediante un scaner, aunque su procedencia sea diferente, es decir son imágenes netamente computacionales.

1.2. Problemática.

Pocos trabajos tratan el tema de la transformación de imágenes fotográficas (mapeadas en bits).

La mayor parte de imágenes usadas en revistas, libros, boletines, etc. son imágenes procedentes del mundo real. Estas imágenes, como ya se mencionó, son generalmente pasadas a la computadora a través de un scaner, y son usadas con algunos cambios en su apariencia (aunque pocas veces en su forma). Esto quiere decir que siguen usando el formato clásico de fotografía rectangular. En algunos casos, las imágenes son "recortadas" para cambiar la forma rectangular, pero el interior de las mismas siguen siendo igual.

Para lograr estos cambios existen actualmente una gran diversidad de programas de edición de imágenes en bits; para lograr una amplia variedad de efectos (cambios en su apariencia) sobre una imagen, como son: control cromático, nitidez, contraste o retoque de la imagen.

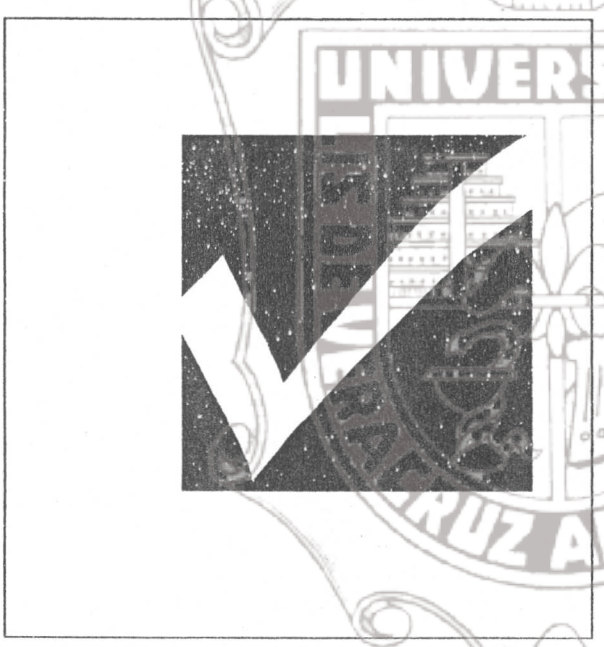
Sin embargo, ningúno pocos efectos se abocan a permitir la transformación de la imagen. El control de esta transformación se puede dar por medio de vectores o ecuaciones; que permitan cambiar la apariencia de una imagen en forma controlada y planeada.

A pesar de la amplia difusión y uso de imágenes vectorizadas, pocos libros de graficación tratan sobre el manejo de imágenes fotográficas. Los que lo hacen, mencionan esto como una posibilidad teórica, pero no concretan gran cosa.

Estas transformaciones implicarían un manejo semivectorizado de una imagen fotográfica, el cual hasta ahora no ha sido muy explotado. Sin embargo estas transformaciones representan una amplia gama de posibilidades en la obtención de nuevas efectos en una fotografía.

Algunos métodos que permiten trazar curvas como son la función senoidal o el algoritmo de Bezier, pueden ser ampliados para aplicarse al procesamiento de imágenes fotográficas y lograr efectos espectaculares, controlados en las imágenes.

Instituto de Ingeniería
Universidad Veracruzana



Capitulo II
Objetivos

"No hay frontera entre la realidad y la fantasia"

II. OBJETIVOS.

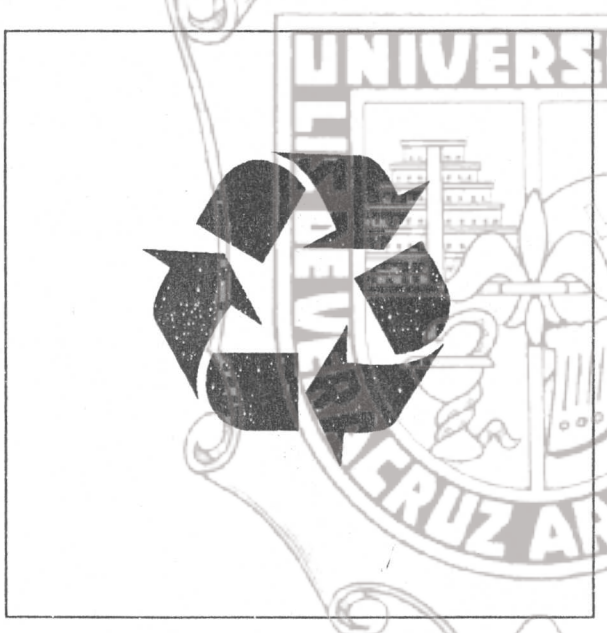
Objetivo General.

Desarrollar e implantar un conjunto de algoritmos que permitan realizar transformaciones controladas en imágenes fotográficas (mapas de bits); teniendo como resultado final un algoritmo de transformación al cual llamaremos malla de eslabonamiento.

Objetivos Específicos.

- Diseñar un algoritmo que permitan reducir el tamaño de una imagen.
- Diseñar un algoritmo que permita ampliar el tamaño de una imagen, procurando conservar en la calidad de la imagen.
- Diseñar un algoritmo que permita realizar la rotación de una imagen, en cualquier ángulo.
- Diseñar un algoritmo que le de a una imagen un efecto ondulatorio, basado en la función senoidal.
- Diseñar un algoritmo que le de a una imagen un efecto ondulatorio, basado en una curva de Bezier.
- Diseñar un algoritmo que permita aplicar el efecto de eslabonamiento a una imagen, en base a la rotación en X y Y con diferentes ángulos.
- Diseñar un algoritmo que permita alterar una imagen, usando una malla de eslabonamiento con como modelo de transformación.
- Aplicar en una función, cada algoritmo diseñado, para mostrar su utilidad potencial.

Instituto de Ingeniería
Universidad Veracruzana



Capitulo III
Hipotesis

"Si Justificas cada paso que des, tu camino será firme"

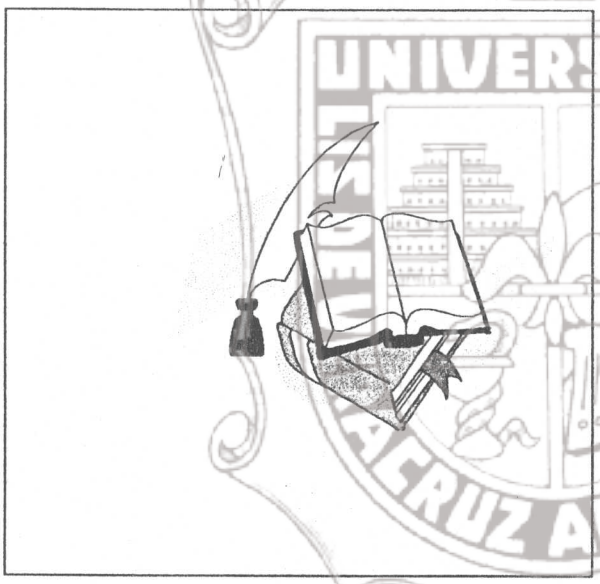
Proyecto de Digitalización de Tesis
Responsable M.I. Alberto Pedro Lorandi Medina
Colaboradores: Estanislao Ferman García
M.I. Enrique Rodríguez Magaña

III. HIPÓTESIS.

"La transformación de imágenes fotográficas, resultará útil para la obtención de efectos especiales en fotografías; estos algoritmos serán comunes en un futuro no muy lejano, y se encontrarán implantados comúnmente en programas de edición de imágenes fotográfica".

" El tiempo computacional que implica el uso de estos algoritmos, resulta muy elevado, por la cantidad de cálculos que se realizan, por lo que su funciónamiento se debe dar preferentemente en computadoras o estaciones de trabajo con amplia capacidad de procesar gráficos."

" Al tener como premisa principal (en el desarrollo de los algoritmos), *la conservación de la calidad original de la fotografía* (en la medida de los posible), el tiempo de procesamiento de cada uno de los algoritmos es considerablemente mayor al que requieren los efectos de cambio de apariencia, que implementan los programas actuales de edición de imágenes, mapeadas."



Capitulo IV
Revisión Bibliográfica

"Quinientos canales en la TV, y nada que ver....."

IV. REVISIÓN BIBLIOGRÁFICA.

4.1. Organizacion de la información.

La mayor parte de la información necesaria para realizar el presente trbajo de investigación es basíca, es decir, es un conocimiento que cualquier profesionista del area de las ciencias exactas tiene, como el funciónamiento de identidades trigonometricas y algebra.

Sin embargo hubo alguna información mas especializada que se obtuvo de algunos libros de matematicas y graficación que se listan en la bibliografia. Dicha información es la siguiente.

4.2. El inicio de la graficación

Una de los campos de las ciencias computacionales que mas se han desarrollado en los últimos años es precisamente la que tiene que ver con la representación de imagenes a traves de la computadora.

El desarrollo de dicho campo empezo con el trabajo pionero de Ivan Shutherland en 1963, cuando este desarrollo (para su tesis doctoral) un sistema trazador de lineas rectas llamado Sketchpad, el cual permitia trazar lineas rectas interconectadas entre si, auxiliado por una dispositivo similar a ráton (el cual fue invetado por el mismo).

Para recalcar la importancia y el avance que produjo el trabajo de Shuterland en 1963, cabe mencionar que los dispositivos de despliege visual de esa epoca eran en su mayoria terminales de teletipo, y no fue sino hasta !965 que IBM creo un TRC (Tubo de Rayos Catodicos) con refresco calculado de la imagen a un costo de 10,000 dlls.

Tres años despues, en 1968 Textronic creo un TRC que retenia permanentemente un dibujo hasta que este se borraba manualmente, a un precio de 15,000 dlls.

No fue sino hasta los 70´s, en que la reduccion en precio y tamaño de dispositivos de despliege, con retencion de imagen por memoria, en un TRC permitio que este dispositivo se volviera mas comun en un equipo de computo.

El trabajo de Shuterland (sin duda relevante en su tiempo) dio pie a la creación de toda la industria del hardware, que aprovecho este descubrimiento para crear sistemas de computo accesibles al ser humano comun y corriente.

En cuanto al software, tambien el trabajo de Ivan Shuterland dejo una honda huella, y se le considera el iniciador de este campo al desarrollar algunos de los mejores algortimos y estructuras de datos en que se basa la graficacion actual.

Otros trabajos que contribuyeron grandemente con al desarrollo de la graficación actual, son los de algunos reconocidos matematicos como el caso de Coons, Bezier y Manderbrot; asi como de otros tantos.

4.3. Tipos de graficas

Aunque al inicio de la graficación se consectualizaba una grafica como una serie de instrucciones que permitian pintar "algo" en la pantalla; con el paso del tiempo y el desarrollo de los dispositivos que se pueden conectar a la computadora, se tuvo que ampliar este concepto, para considerar como una grafica, a la imagen capturada proveniente del mundo real.

Es asi como en la actualidad se habla esencialmente de dos tipos de graficas: las vectotizadas y las mapeadas en bits.

Las graficas vectorizadas son imagenes que estan compuesta por primitivas de dibujo. Una primitiva es la descripción de un elemento basico de dibujo, como lo es una linea, un circulo, una curva, etc.. Estas primitivas interconectadas entre si, forman imagenes complejas que son las que nos muestran los dispositivos de salida como un monitor o una impresora.

Las graficas de mapas de bits son imagenes capturadas del mundo real y almacenadas en funcion de los colores que tiene cada punto que forma la imagen. En este tipo de imagenes no importan las formas que tiene las

figuras contenidas en la imagen, lo que interesa es las descripción de colores en cada punto.

A continuación se describen sus principales caracteristicas.

Vectorizadas	Mapa de Bits
• Usa archivos pequeños	• Es espacio es generalmente grande.
• Ideal para uso ingenieril	• Ideal para uso artistico o publicitario.
• No suelen verse naturales	• Representa casi siempre imagenes reales.

4.4. Usos de la graficación en la actualidad

Las graficas en la actualidad de usan en una amplia variedad de aplicaciones computacionales. En muchas de ellas solo se usa el resultado de ellas como parte del sistema, otras aplicaciones se encargan modificar las imagenes. A continuación mencionaremos algunas de las aplicaciones que usan comunmente graficos.

Instruccion asistida por computadora	Este tipo de aplicaciones usa la vistosidad de las graficas por computadora, para atraer la atención de un educando y asi facilitar el proceso de enseñanza. Existen programas que enseñan temas de todo tipo, desde el nivel elemental, hasta enciclopedias completas.

Presentaciones	Se usan para realizar alguna demostración ante una audiencia, por lo general este tipo de aplicaciones permiten crear efectos animados y tienen formas variadas de despliege de graficas.
Multimedia	Se le llama asi a toda aplicación que usa mas de dos medios de transmisión de información . Un medio tradicional de transmisión es el texto, un segundo tipo que se agrego fueron precisamente las gráficas, y recientemente se ha agregado sonido y video.
Simuladores	Son aplicaciones que tiene como objetivo preparar a un especialista en el manejo de un equipo sofistica. Los tipo de simulares se usan capacitar operadores de centrales nucleares y aviones. El uso de las graficas en estas aplicaciones se da para recrear un ambiente muy aproximado al que se enfrentará el operador que se esta capacitando.

En cuanto a aplicaciones que se encargan de crear y/o mejorar graficas, existen un tipo para imagenes vectorizada y otro para las imagenes en mapas de bits. Dichas aplicaciones se describen a continuación.

Sistemas de CAD	Permiten crear y modificar imagenes basadas en vectores. Estos sistemas dan facilidad para el manejo por grupos de primitivas dando la posibilidad de darle algunos efectos como: colorización, iluminación, texturizado, etc.

Sistemas de dibujo	Aplicaciones tambien llamadas de retoque que permiten mejorar una imagen proveniente del mundo real, capturada a traves de un Scaner. Dichos programas permiten (en su mayoria) solo cambiar atributos cromaticos, pero no siempre permiten modificar la forma de la imagen.

4.5. Transformaciones en graficas de vectores

Estas transformaciones son descritas ampliamente en muchos libros sobre graficación. En esencia son tres los tipos de transformaciones elementales. A partir de ellas se crean otras transformaciones que aplicadas a dos y tres dimensiones permiten modificar y/o animar una imagen vectorizada.

Las tres transformaciones basicas son las siguientes:

Traslación

Es el movimiento en línea recta de un objeto, de una posición a otra. Se traslada un punto de la posición coordenada (x, y) a una nueva posición (x', y') agregando distancias de traslación, Tx y Ty, a las coordenadas originales:

$$x' = x + Tx, \qquad y' = y + Ty$$

El par de distancia de traslación (Tx, Ty) se denomina también vector de traslación o bien vector de cambio.

Escalación

Permite alterar el tamaño de un objeto. Puede efectuarse con polígonos multiplicando los valores coordenados (x, y) de cada vértice de frontera por los factores de escalación Sx y Sy para producir las coordenadas transformadas (x', y').

$$x' = x \bullet Sx \qquad y' = y \bullet Sy$$

Los valores menores que 1 reducen el tamaño de los objetos; los valores mayores que 1 producen un agrandamiento. Cuando a Sx y Sy se les asigna el mismo valor, se produce una escalación uniforme.

Cuando un objeto se vuelve a trazar con las ecuaciones de escalación, la longitud de cada línea de la figura se hace a escala de acuerdo con valores asignados a Sx y Sy. Además, la distancia de cada vértice al origen del sistema de coordenadas también se hace a escala. Los objetos agrandados se retiran del origen coordenado.

Se puede controlar la localidad de un objeto a escala eligiendo una posición, llamada punto fijo. Las coordenadas del punto fijo, (xF, yF), pueden elegirse como uno de los vértices, el centro del objeto o bien cualquiera otra posición. Para un vértice con coordenadas (x, y), las coordenadas a escala (x' y') se calculan como

$$x' = xF + (x - xF)Sx, \qquad y' = yF + (y - yF)Sy$$

Se pueden reacomodar los términos de estas ecuaciones para obtener transformaciones de escalación relativas a un punto fijo seleccionado como

$$x' = x \bullet Sx + (1 - Sx) \bullet Xf$$

$$y' = y \bullet Sy + (1 - Sy) \bullet Yf$$

Rotación

La transformación de puntos de un objeto situados en trayectoria circulares se llama rotación. Se especifica con un ángulo de rotación.

$$x' = r\cos(\phi + \theta) = r\cos\phi\cos\theta - r\,\mathrm{sen}\phi\,\mathrm{sen}\theta$$

$$y' = r\,\mathrm{sen}(\phi + \theta) = r\,\mathrm{sen}\phi\cos\theta + r\cos\phi\,\mathrm{sen}\theta$$

$$x = r\cos\phi, \qquad y = r\,\mathrm{sen}\phi$$

$$r = x / \cos\phi \qquad r = y / \mathrm{sen}\phi$$

4.6. Transformaciones en graficas de mapas de bits

Este tipo de transformaciones se refieren a cambios en los colores usando algunos procesos que mejoran la imagen. A estos procesos se les llama "filtros" y estos permiten entre otras cosas:

- Modificar uno o varios colores.

- Cambiar la brillantes de los colores.

- Cambiar el contraste entre los colores fuertes y debiles.

- Aplicar gradientes a algunos colores para "desgastarlos" o "realzarlos".

La mayoria de estas caracteristicas viene descritas en libros de optica, pero no en libros de computación.

Sin embargo lo importante es que solo algunos paquetes manejan cambios en la forma de la imagen, de forma similar a como se hace en imagenes vectorizadas.

4.7. Curvas de Bézier.

Este tema se aplica al algoritmo del Efecto Ondulatorio con curvas de Bezier (punto 7.2).

Para trazar una curva de Bezier, se deben de tener un grupo de n+1 puntos de control, los cuales se designan:

$$p_k = (x_k, y_k, z_k),$$

Para k de 0 a n. A partir de estos puntos coordenados, se aplica la función vectorial de Bézier de aproximación P(u). Dicha función esta representada por tres ecuaciones paramétricas, que ajustan la curva a los puntos de control de entrada p_k.. Dichas ecuaciones son las siguientes:

$$P(u) = \sum_{k=0}^{n} p_k \cdot B_{k,n}(u)$$

Cada función $B_{k,n}$ (u) es una función polinomial que se define como:

$$B_{k,n}(u) = C(n,k) \cdot u^k \cdot (1-u)^{n-k}$$

y la función C(n, k) representa a los coeficientes binomiales

$$C(n,k) = \frac{n!}{k! \cdot (n-k)!}$$

En forma explícita, las ecuaciones paramétricas anteriores se aplican a las coordenadas individuales de la curva, de la siguiente forma:

$$x(u) = \sum_{k=0}^{n} X_k \cdot B_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} Y_k \cdot B_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} Z_k \cdot B_{k,n}(u)$$

A los polinomios $B_{k,n}$ (u), se les denominan funciónes de combinación, ya que combinan los puntos de control para formar una función compuesta que describe la curva.

Está función compuesta es un polinomio de un grado menor que el número de puntos de control que se utiliza. Tres puntos generan una parábola, cuatro puntos una curva cúbica y así sucesivamente.

Una propiedad importante de la función de Bézier es que crea una curva de cavidad convexa a la frontera del polígono, de los puntos de control. Esto garantiza que la curva siga fácilmente los puntos de control sin oscilaciones erráticas.

Las curvas cerradas se generan especificando el primero y último puntos de control en la misma posición. La especificación de múltiples puntos de control en una sola posición le otorga más valor o peso.

En general, podría especificarse cualquier número de puntos de control para una curva de Bézier, pero este requiere el cálculo de funciónes polinomiales de mayor grado. Cuando es necesario generar curvas más complicadas, estas pueden formarse dividiendo varias secciones de Bézier de grado menor. Ya que las curvas de Bézier atraviesan puntos extremos, es fácil ajustar secciones de curva (con continuidad de orden cero).

Otra propiedad importante de las curvas de Bézier es que la tangente a la curva en un punto extremo está a lo largo de la línea que une ese extremo al punto de control adyacente. Para obtener la continuidad de primer orden entre dos secciones de una curva, el usuario puede elegir puntos de control de manera que las posiciones p_{n-1} y p_n de una sección estén a largo de la misma recta como posiciones p_0, p_1. Las curvas de Bézier, en términos generales, no poseen continuidad de segúndo orden.

4.8. Otras técnicas de transformación.

Morphing

Una técnica que permiten obtener resultados similares a los algoritmos aquí presentados, son las técnicas para realizar Morphing. Este

proceso consiste esencialmente en cambiar porciones de una imagen inicial (subdividida en triángulos) por la misma porción que se encuentra en una imagen final.

El Morph substituye las áreas triangulares establecidas dentro del dibujo, que delimitan el orden en que va siendo substituida la porción de la imagen, cambiando gradualmente el color de un punto a otro.

El inconveniente del Morph radica precisamente en que requiere una imagen destino hacia la cual lograr la deformación de la imagen inicial.

Tecnicas de programas comerciales

Otra alternativa la representan los efectos que proporcionan algunos programas muy sofisticados de edición y retoque fotográfico que permiten "ampliar", "reducir" y "ondular" parcial o totalmente una fotografía.

Estos programas proporcionan un buen control sobre la imagen, sin embargo es claro que no difunden las técnicas que utilizan, para realizar sus transformaciones.

4.9. Lenguaje usado en la implementación

El lenguaje usado para llevar a cabo la implementación de los algoritmos fue lenguaje C, usando el compilador de la compañia Borland en su versión 3.1.

Se eligio este lenguaje por ser uno de los mas usados en el desarrollo de sistemas, asi como por el dominio que se tiene de el.

Se eligio en particular el compilador de la compañia Borland porque esta proporciona desde hace varios años en sus compiladores, un nucleo grafico configurable llamado BGI (Borland Graphics Interface), el cual es muy usado y flexible, ademas de aceptar drivers frabricados por terceros que permiten manejar mejores resoluciones en nuevo equipo.

4.10. Libreria comercial para el manejo de graficos.

Para facilitar el tratamiento de las imagenes se decidio usar una libreria comercial para manejar dichas imagenes en formatos de mapa de bits PCX. Dicha libreria permite ademas el manejo de alta resolución (Super VGA) usando el nucleo grafico que proporciona Borland C para DOS.

El nombre comercial de dicha libreria es SVGA de la compañia Ryle Desing y sus caracteristicas principales son:

- Lectura y escritura de imagenes PCX, BMP y GIF
- Manejo de modos de alta resolución clasificados como SVGA comenzando con resolución 640 x 480 con 256 colores, hasta 1280 x 1024 con 256 colores.
- Compatibilidad total con el nucleo grafico de Borland C

4.11. Caracteristicas de las imagenes usadas en implementación.

Para todos los algoritmos se usaron imagenes de 300 x 200 pixeles con 256 colores. Se escogio el formato PCX por ser mas pequeñas las imagenes que en un BMP.

En las imagenes probadas se busco tomar, tanto fotografias como imagenes con texturas, para probar que el algoritmo conserva (en la medida de lo posible) la calidad de la imagen.

4.12. Hardware usado en el desarrollo

El equipo de computo donde se desarrollaron los algoritmos fue :

Computadora 486 DX2 de 66 Mhz con:

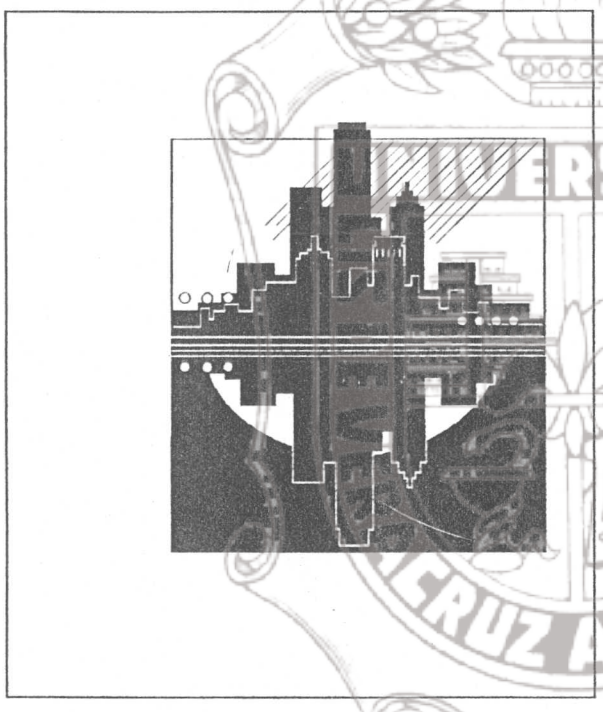
8 Mb de RAM

Disco Duros de 220 Mb

Tarjeta de Video Cyrrus con 1 Mb

La resolución que se uso en desarrollo del mismo fue 800x600 con 256 colores.

Instituto de Ingeniería
Universidad Veracruzana



Capitulo V
Escalación de Imagenes

"La altura de un hombre se mide de la cabeza al cielo "

Proyecto de Digitalización de Tesis
Responsable M.I. Alberto Pedro Lorandi Medina
Colaboradores: Estanislao Ferman García
M.I. Enrique Rodriguez Magaña

V. ESCALACION DE IMÁGENES.

5.1. Reducción.

5.1.1. Descripción general.

El algoritmo se basa en la información inicial sobre el tamaño de la imagen original y el tamaño al que se desea reducir. La diferencia entre ambos tamaños es la cantidad de pixel a perder.Estos pixeles a eliminar se distribuiran simétricamente a lo largo de la imagen para hacer que el efecto de reducción no deforme la imagen. Si la cantidad de pixel a reducir se eliminaran en una sola parte de la imagen (Figura 5.1,b), esta se adelgazaría de algunas partes pero perdería su proporción.

Con este algoritmo se calcula cada cuantos pixel (columnas) se debe ignorar de la imagen original y como ignorarlos simétricamente. Esto hace que los pixel que se pierdan no deformen la imagen significativamente (Figura 5.1.C).

Este proceso se debe realizar para cada columna de pixel de la imagen fuente y también a cada renglón con la idea de reducción simétrica.

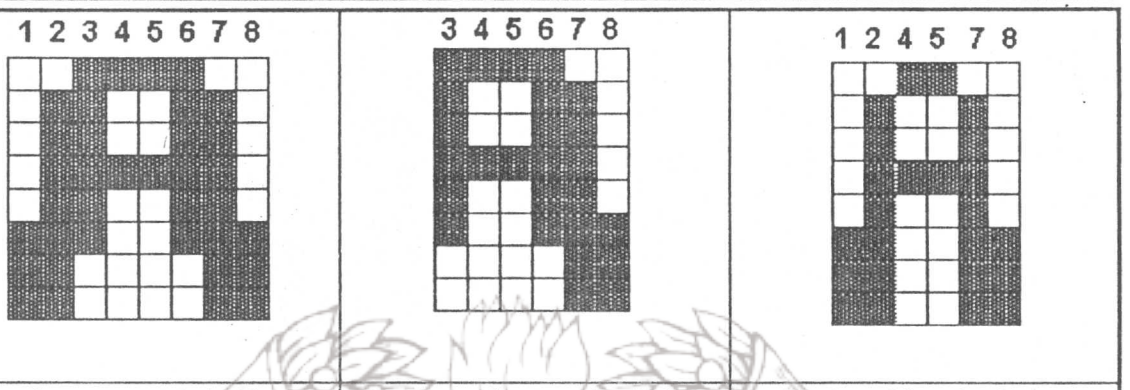


Imagen original	Reducción no simétrica	Reducción simétrica
a	b	c

Figura 5.1.- Comparación entre reducion no simetrica y simetrica

5.1.2. Parámetros de entrada.

Los parámetros de entrada son los siguientes:

xoi, xof	Columna inicial y final en que se encuentra la imagen fuente
yo	Renglón al que se aplicara la reducción
xdi,ydi	Coordenada destino donde se colocara el renglón reducido
xdf	Columna donde terminara el renglón reducido.

5.1.3. Algoritmo.

1.-Calcula cada cuantos pixel se deben ignorar un pixel que seria el pixel a perder.

2.-Incializa acumulador de pixel con valor anterior

3.-Recorre el renglón de la imagen a reducir.

4.-¿Es la cantidad de pixel recorridos diferente del acumulador de pixel a eliminar?

Si: Copia el pixel de la imagen destino

No: Ignora pixel

5.-¿Es el último punto del renglón ?

Si: Ir a 6

No: Ir a 3.

6.- Fin

5.1.4. Implementación.

En la practica la reducción se debe de poder aplicar en el plano X y Y de la figura. Por ello en realidad de cuentan con dos rutinas, una para reducir en X y otra para reducir en Y. Ambas rutinas son llamadas por otra que une a ambas.

CODIGO	COMENTARIOS
if xdf-xdi>xof-xoi xdf=xdi+(xof-xoi);	Si tamaño destino es mayor que fuente, ajusta a tamaño fuente
dx=(xdf-xdi+1)/((xof-xoi)-(xdf-xdi)+1);	Calcula el factor que determina cada cuantos pixeles ignora la columna
vdx=dx;	Inicializa contador de columna a eliminar
for(x=xoi;x<=xof;x++,xdi++)	Recorre renglón *yo* de imagen fuente

if (x-xoi!=round(vdx))
putpixel(xdi,ydi,getpixel(x,yo));
else { vdx+=dx+1; xdi--; };
	Si distancia (x-xo) no es múltiplo de vdx, la columna es ignorada
	• Entonces "copia" el pixel de la imagen fuente.
	• Sino incrementa contador de columnas y decrementa la columna destino

5.1.5. Comentarios.

Este proceso es eficiente para poder realizar reducciones con una perdida aceptable de calidad de la imagen fuente.

No es posible realizar el proceso de ampliación con la misma lógica, ya que como se vera a continuación, la ampliación es substancialmente diferente de la reducción.

5.1.6. Muestra del algoritmo

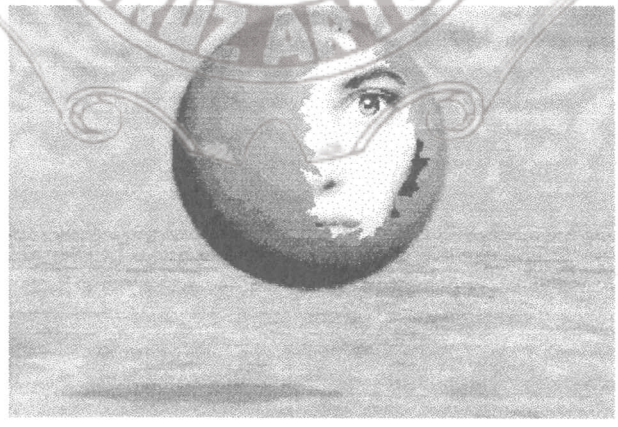
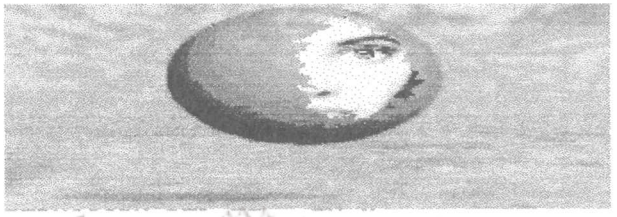


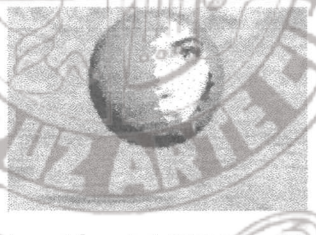
Imagen Original

Figura 5.2 (1 de 2). Muestra de imagenes reducidas.



Reducción del 50 % en Y

Reducción del 50% en X



Reducción del 50% en X y Y

Figura 5.2.(2 de 2).

5.2. Ampliación.

5.2.1. Descripción general.

Esencialmente el proceso consta de dos pasos que se dan selectivamente dentro de un proceso repetitivo.

Estos pasos son:

1.- Verificar si se necesita crear una columna.

2.- Si no entonces copia la columna de imagen fuente.

Creación de una columna

Este proceso es contrario a la reducción. Aunque la forma de calcular simétricamente el lugar donde se insertara la columna es la misma; el proceso de inserción requiere un análisis de tendencia de colores.

Este análisis involucra a los puntos vecinos del cada pixel de la columna a insertar. Para facilitar la explicación se darán los siguientes nombres de colores a los pixel vecinos

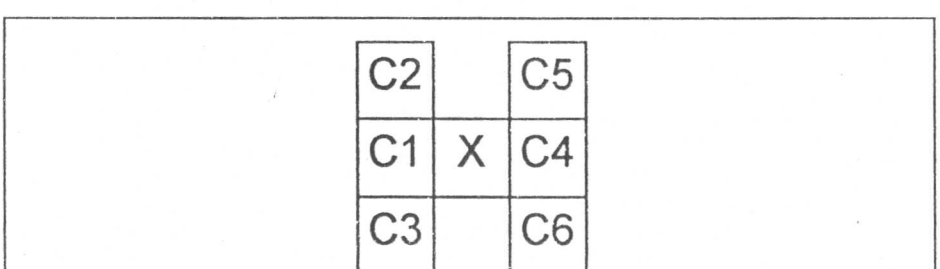


Figura 5.3. Identificador de puntos vecinos en analisis de color

En la Figura 5.3, "X" representa el pixel de la columna que se está insertando. Los colores de los puntos vecinos esta representados por "C1" a "C6".

Las tendencias observadas mas frecuentemente son las siguientes:

TENDENCIA	COLOR ELEGIDO
C1=C4	C1;
C2=C6	C2;
C3=C5	C3;
C2=C1 Y C1=C3	C2;
C5=C4 Y C4=C6	C5;
C2=C5 Y (C5=C4 O C2=C1)	C2;
C3=C6 Y (C3=C1 O C6=C4)	C3;
(C1=C5 Y C5=C4) O (C1=C6 Y C6=C4)	C1;
(C4=C2 Y C2=C1) O (C4=C3 Y C3=C1)	C4;

En caso de no seguirse ninguna de las tendencias arriba citadas, se procede a contar los colores que mas se repite entre los puntos vecinos. Este criterio se tomo, ya que una imagen que no muestra algunas de las tendencia de colores arriba citadas, es una imagen difusa en colores, por lo que entonces lo que importa es la cantidad y no la tendencia. El color mas frecuente es el asígnado al pixel a crear.

Como se puede observar, el pixel superior a "X" no se considera en el análisis, a pesar de haber sido el inmediato calculado. Esto se hizo así, ya

que al implementarlo, se observó una diferencia notable en la elección del color a crear; además de aumentar la cantidad de casos a analizar, lo cual consumía mas ciclos de procesos sin mejorar significativamente la calidad de la imagen.

Este proceso se repite para cada pixel de la columna que se está creando, lo cual demanda una cantidad considerable de cálculos. La ventaja que presenta con respecto a los métodos de ampliación que contiene otros paquetes, es que la calidad de imagen se conserva en la ampliación evitando el efecto de "cuadriculado" que se presenta en estos paquetes.

Copiado una columna

Esta parte del proceso consiste solo en copiar la columna de la imagen fuente en la imagen destino. Este proceso se da en las columnas intermedias a las columnas que insertan. En la Figura 5.4 se puede observar este proceso:

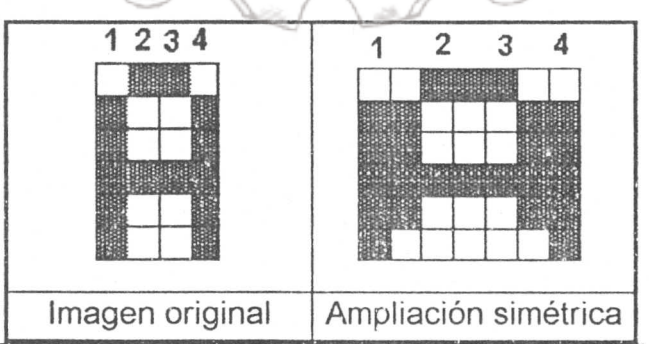


Figura 5.4. Ampliación simetrica

5.2.2. Parámetros de entrada.

Los parámetros de entrada son los siguientes:

xoi, xof	Columna inicial y final en que se encuentra la imagen fuente
yo	Renglón al que se aplicara la reducción
xdi,ydi	Coordenada destino donde se colocara el renglón ampliado
xdf	Columna donde terminara el renglón ampliado.

5.2.3. Algoritmo.

1.- Calcular cada cuantas columnas se crea una nueva

2.- Recorre cada elemento del renglón

3.- ¿El número de pixel recorridos es igual al del pixel a insertar?

Si: Lee los colores de puntos vecinos y crea pixel según análisis

No: Copia el pixel de la imagen fuente

4.- ¿Es el último pixel del renglón?

Si: Ir a 5

No: Ir a 2

5.- Fin

5.2.4. Implementación.

CODIGO	COMENTARIOS
float dx,vdx;	dx desplazamiento en X vdx desplamiento acumulado
register int x; int c1,c2,c3,c4,c5,c6; dx=(xof-xoi+1)/(float)((xdf-xdi)-(xof-xoi)+1);	x contador recorre columna c1 a c6 colores de puntos vecinos Calcula cada cuantas columnas se debe crear una columna
vdx=dx;	Inicialización de acumulador de columnas
for(x=xoi;x<=xof;x++,xdi++)	Recorrido del renglón *yo*
if ((x-xoi)==round(vdx))	¿Se inserta columna?

c1=getpixel(x-1,yo);
c2=getpixel(x-1,yo-1);
c3=getpixel(x-1,yo+1);
c4=getpixel(x+1,yo);
c5=getpixel(x+1,yo-1);
c6=getpixel(x+1,yo+1);
color=col_rep(c1,c2,c3,c4,c5,c6)
putpixel(xdi,ydi,color);
vdx+=dx;
x--;
	Si: se toman colores ptos. vecinos
	Pixel izquierdo
	Pixel izquierdo superior
	Pixel izquierdo inferior
	Pixel derecho
	Pixel derecho superior
	Pixel derecho inferior
	Determina color por tendencia
	Pinta el pixel del color elegido
	Incremento acumulador
	Decrementa *x* ya que se creo,.

else putpixel(xdi,ydi,getpixel(x,yo));	Sino: copia pixel de la imagen original

La implementacion mostrada arriba, amplia solo un renglón. Si se desea aplicar a toda la imagen, esta función debe estar dentro de un ciclo que recorra todos los renglónes de la imagen.También se diseño otra función similar a la anterior, que amplia una columna especifica, para poder tener ampliación sobre el eje Y también.

Por último, existe una versión de la función de ampliación que hace el trabajo en ambos sentidos, usando las funciónes arriba citadas.

5.2.5. Comentarios.

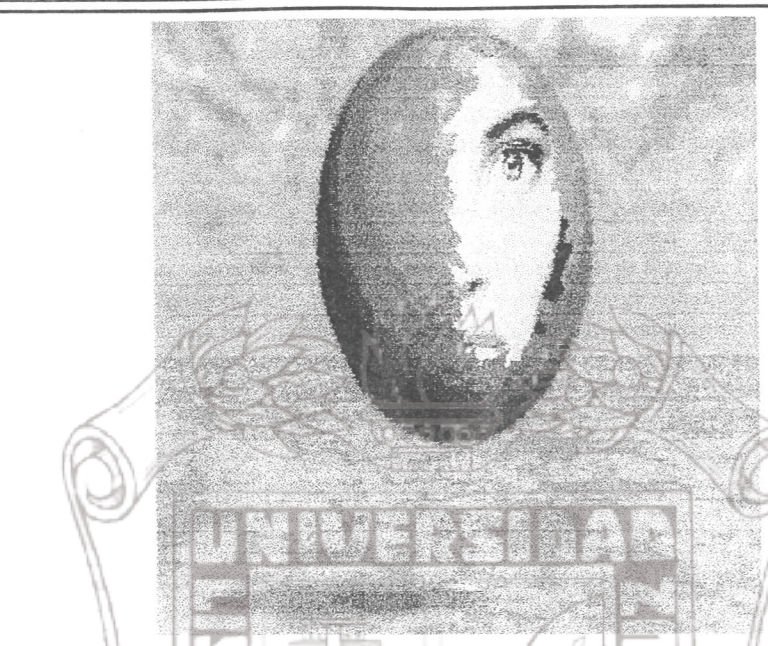
La implementación de este algoritmo se concreto a "probar" la eficiencia en la calidad de la imagen. Si se implementa de manera formal en alguna aplicación es posible optimizar algunos aspectos para hacerlo mas rápido.

Otra mejora (a considerare en una aplicación final) es el unir en una sola subrutina el llamado a las funciónes de reducción o ampliación, para que esto quede transparente para el usuario.

5.2.6. Muestra del Algoritmo.

Imagen Original

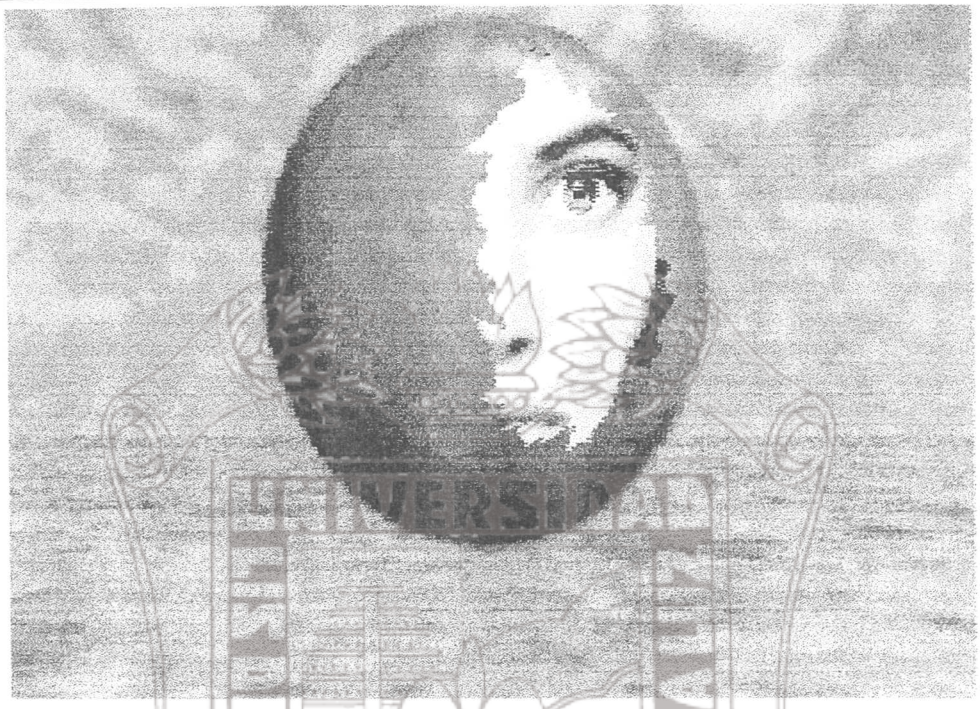
Figura 5.5(1 de 3). Muestra de imagenes ampliadas



Ampliación 50% en Y

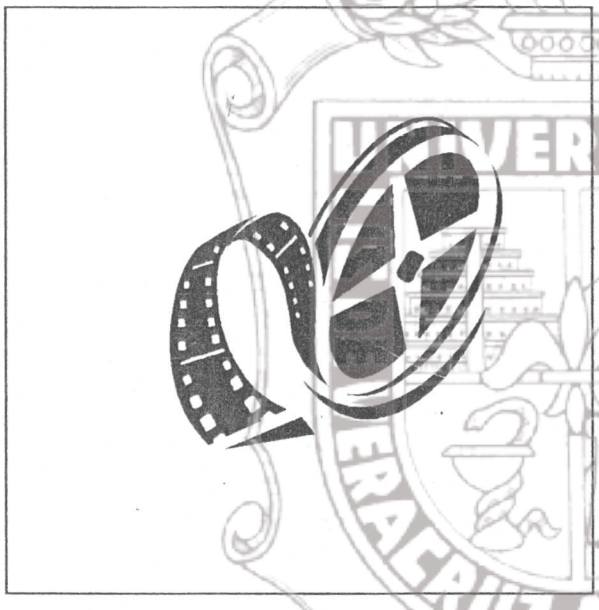
Ampliación 50% en X

Figura 5.5.(2 de 3)



Ampliación 60% en X y 90% Y

Figura 5.5.(3 de 3)



Capítulo VI
Rotación de Imagenes

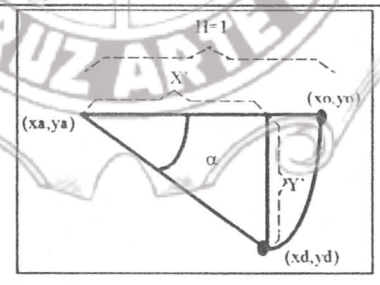
"La diferencia entre humanos y animales quizas solo sea la risa"

VI. ROTACIÓN DE IMAGENES.

6.1. Rotación simple.

6.1.1. Descripción general.

El proceso de rotación consiste en girar α grados, cada renglón de la imagen fuente. A cada giro de un renglón completo, hay que calcular también el lugar donde comenzará cada nuevo renglón. Este proceso se realiza punto por punto, en realidad se traduce a un simple incremento del cosα (redondeado al entero mas proximo) para cada nuevo valor de X y un incremento de senα (también redondeado a entero) para cada nuevo punto de Y. La siguiente Figura 6.1 ilustra dicha situación:



$$\cos\alpha = \frac{X'}{H} \qquad \operatorname{sen}\alpha = \frac{Y'}{H}$$

$$X' = H * \cos\alpha \qquad Y' = H * \operatorname{sen}\alpha$$

Figura 6.1. Calculo de incrementos en la rotación

Como H=1, ya que la pantalla trabaja a nivel de pixel y es la distancia minima entre punto y punto.

Un proceso similar a este se presenta en el cambio de posición que tiene cada pixel que inicia el renglón; ya que como gira el renglón la posición de inicio del mismo debe "ajustarse" con el ángulo complementario. Por esto la posición del primer punto de cada renglón se calcula con valores acumulativos de senα para X y cosα para Y.

Esto en si no presenta ningún problema, sin embargo hay una consideración, al "girar", cada renglón las líneas ya no "coinciden" punto a punto presentando algunos "puntos blancos" que hacen que la imagen no tome un aspecto solido. La Figura 6.2 ilustra este caso:

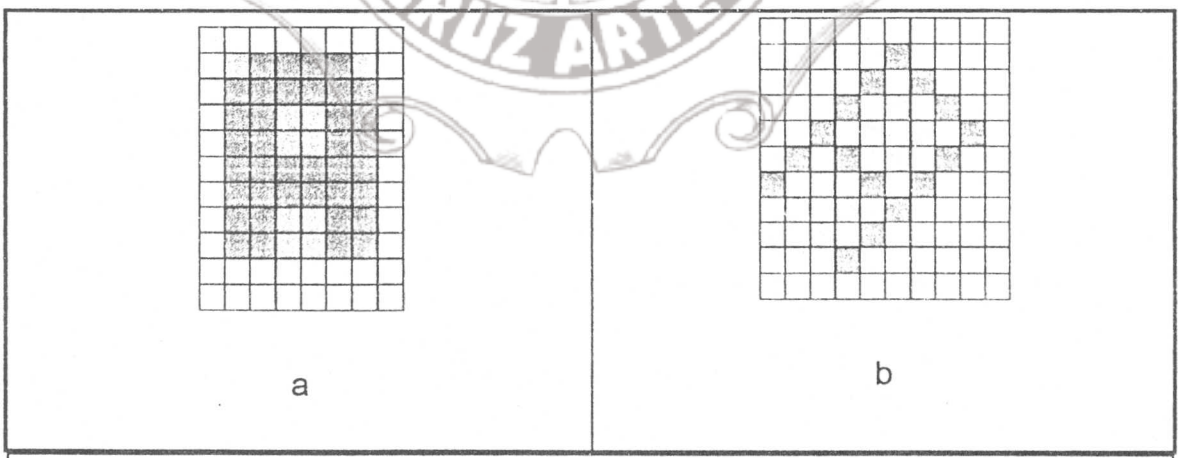


Figura 6.2. Muestra de "puntos blancos" en figura rotada

La imagen original (Figura 6.2a) se transforma en la imagen de la Figura 6.2*b,* al aplicarle una rotación de - 45° . La figura resultante presenta "puntos blancos" en algunas partes de la misma. Este proceso al aplicarse a una fotografía produce que una imagen sin nitides, borrosa. Para corregir este problema, se decidió copiar el mismo renglón una posición adyacente al punto actual. La selección del punto adyacente depende del cuadrante donde se haga el giro. Por ejemplo en la Figura 6.2b el punto adyacente a llenar esta un pixel a la derecha y abajo del punto.

Los incrementos que se deben aplicar a los puntos normales, para ocultar los puntos blancos, en función del ángulo se muestran a continuación:

Rango angular	Incremento en X	Incremento en Y
0 a 90	1	- 1
91 a 180	- 1	- 1
181 a 270	-1	1
271 a 360	1	1

El proceso descrito anteriormente se aplica renglón por renglón a toda la figura fuente y coloca la figura rotada en la coordenada destino.

6.1.2. Parámetros de entrada.

Los parámetros de entrada son los siguientes:

x1, y1	Coordenada inicial de la imagen fuente.
x2,y2	Coordenada final de la imagen fuente.
xd,yd	Coordenada destino, que representa el centro sobre el cual gira la imagen.
ang	Ángulo de rotación que se aplicara a la imagen.

6.1.3. Algoritmo.

1.-Determinar el cuadrante destino para la rotación, así como parámetros de ajuste para puntos blancos

2.-Recorre la figura por renglón

3.-¿El punto de inicio del renglón es diferente al anterior?

Si: Ir a 4

No: Ir a 9

4.-Recorre cada punto del renglón elegido

5.-¿El punto a copiar de imagen fuente es diferente al punto anterior?

Si: Ir a 6

No: Ir a 7

6.-Copia pixeles (calculado y ajustado) a coordenada destino

7.-Incrementa coordenada del pixel a pintar según el cuadrante

8.-¿Terminó de recorrer el renglón?

Si: Ir a 9

No: Ir a 4

9.-Incrementa coordenada del inicio del renglón según cuadrante

10.-¿Terminó de recorrer todos los renglones?

Si: Ir a 11

No: Ir a 2

11.-Fin

6.1.4. Implementación.

CODIGO	COMENTARIOS
int xt1=0, yt1=0; int x,y; int xt,yt;	• Coordenada destino inicio, de cada renglón a rotar; se controla por el ciclo exterior • Control de ciclos • Coordenada destino de cada elemento de un renglón rotado; se controla con el ciclo mas interior.
int xant,yant; int xant1,yant1;	• Valor de la columna y renglón pintados anteriormente, si este valor es igual al renglón o columna a pintar, entonces evitar el pintado
float sa,ca; float dc,ds; float ds1,dc1;	• Seno y coseno respectivamente del ángulo a rotar • Incremento acumulado de seno y coseno para coordenada de cada elemento de renglón • Incremento acumulado de seno y coseno para coordenada de inicio de cada renglón
char flag=0; char a,b;	Cuadrante en que se empezará a dibujar la figura a rotar Ajustan pixel a pintar para evitar

	"puntos blancos".
if 0<ang<90 { flag=1; a=1; b=-1; } if 90<ang<=180 { flag=2; a=-1; b=-1;} if 180<ang<=270 { flag=3; a=-1; b=1; } if 270<ang<=360 { flag=4; a=1; b=1; }	Verifica el ángulo de rotación para establecer incrementos de puntos vecinos para sobreescribir puntos blancos.
ca=cos(ang); sa=sin(ang); ds1=sa; dc1=ca;	• Calcula seno y coseno de ángulo • Inicializa acumulador de sen y cos
xd+=sa*((y2-y1)/2)-ca*((x2-x1)/2); yd+=-ca*((y2-y1)/2)-sa*((x2-x1)/2);	• Calcula el lugar de la coordenada inicial en función del ángulo.
xt1=round(ds1); yt1=round(dc1); xant1=0; yant1=0;	Inicializa acumuladores de control de columnas
for (y=y1; y<=y2; y++)	Ciclo que recorre las columna
if(xant1!=xt1||yant1!=yt1)	¿Cambio coord. anterior de columna?

dc=ca; ds=sa; xt=round(dc);
yt=round(ds); xant=0; yant=0;
for (x=x1; x<=x2; x++)
	Si: inicializa acumuladores de control de renglónes
	Ciclo que recorre los renglónes

if ((xant!=xt||yant!=yt))	¿Cambio coord. anterior de renglón?

putpixel(xd+xt+xt1+1,yd+yt+yt1,
getpixel(x+a,y+b));
putpixel(xd+xt+xt1 ,yd+yt+yt1,
getpixel(x,y));
	Copia pixel vecino en probable punto blanco
	Copia pixel en punto rotado.

xant=xt; yant=yt;
dc+=ca; ds+=sa;
switch(flag)
	Guarda coord. anterior del renglón
	Incrementa acumulador de ángulo
	Evaluacion de flag en renglón

case 1:
if (xt<round(dc)) xt++;
if (yt<round(ds)) yt++;
case 2:
if (xt>round(dc)) xt--;
if (yt<round(ds)) yt++;
case 3:
if (xt>round(dc)) xt--;
if (yt>round(ds)) yt--;
case 4:
if (xt<round(dc)) xt++;
if (yt>round(ds)) yt--;
	Cuadrante 1
	Si coordenada no rebasa acumulador, la incrementa
	Cuadrante 2
	ident.
	Cuadrante 3
	ident
	Cuadrante 4
	ident.

xant1=xt1; yant1=yt1; switch(flag)	Guarda coord. anterior del renglón Evalua flag en columna
case 1: ds1-=sa; dc1+=ca; if (xt1>round(ds1)) xt1--;	Cuadrante 2 Incrementa o decrementa según

```
            if (yt1<round(dc1)) yt1++;      cuadrante
case 2: dc1+=ca; ds1-=sa;                   Cuadrante 3
            if (xt1>round(ds1)) xt1--;      Ident.
            if (yt1>round(dc1)) yt1--;
case 3: dc1+=ca; ds1-=sa;                   Cuadrante 4
            if (xt1<round(ds1)) xt1++;      Ident.
            if (yt1>round(dc1)) yt1--;
case 4: ds1-=sa;  dc1+=ca;                  Cuadrante 1
             if (xt1<round(ds1)) xt1++;     Ident.
             if (yt1<round(dc1)) yt1++;
```

6.1.5. Muestra del algoritmo

Imagen Original

Figura 6.3 (1 de 4). Muestra de rotacion simple de una imagen

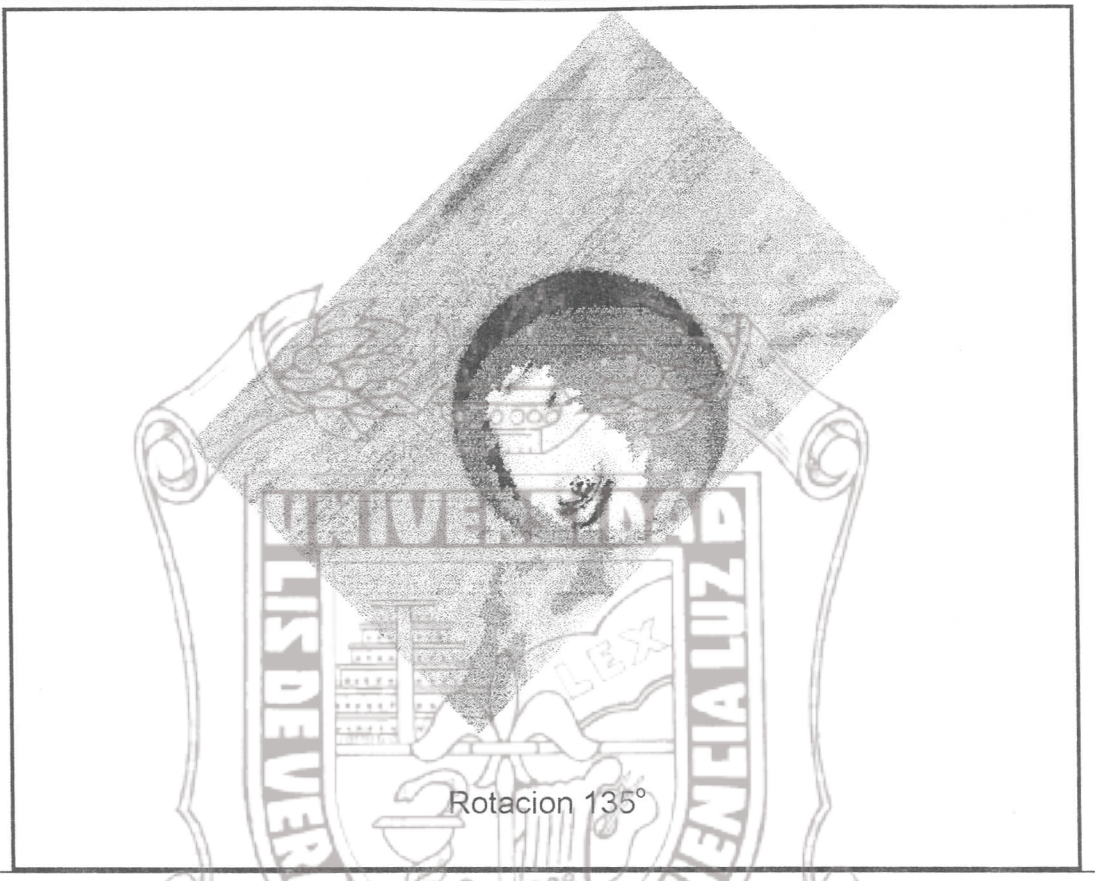


Figura 6.3.(2 de 4)

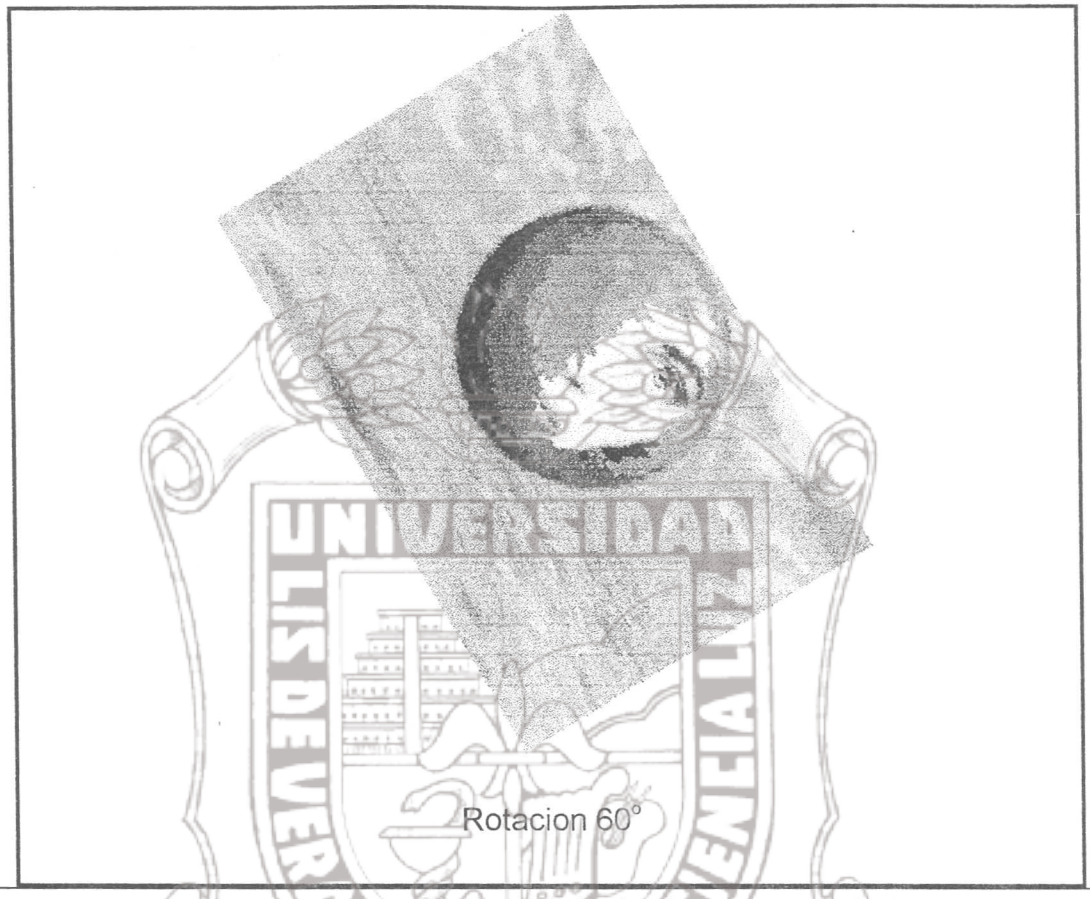


Figura 6.3 (3 de 4)

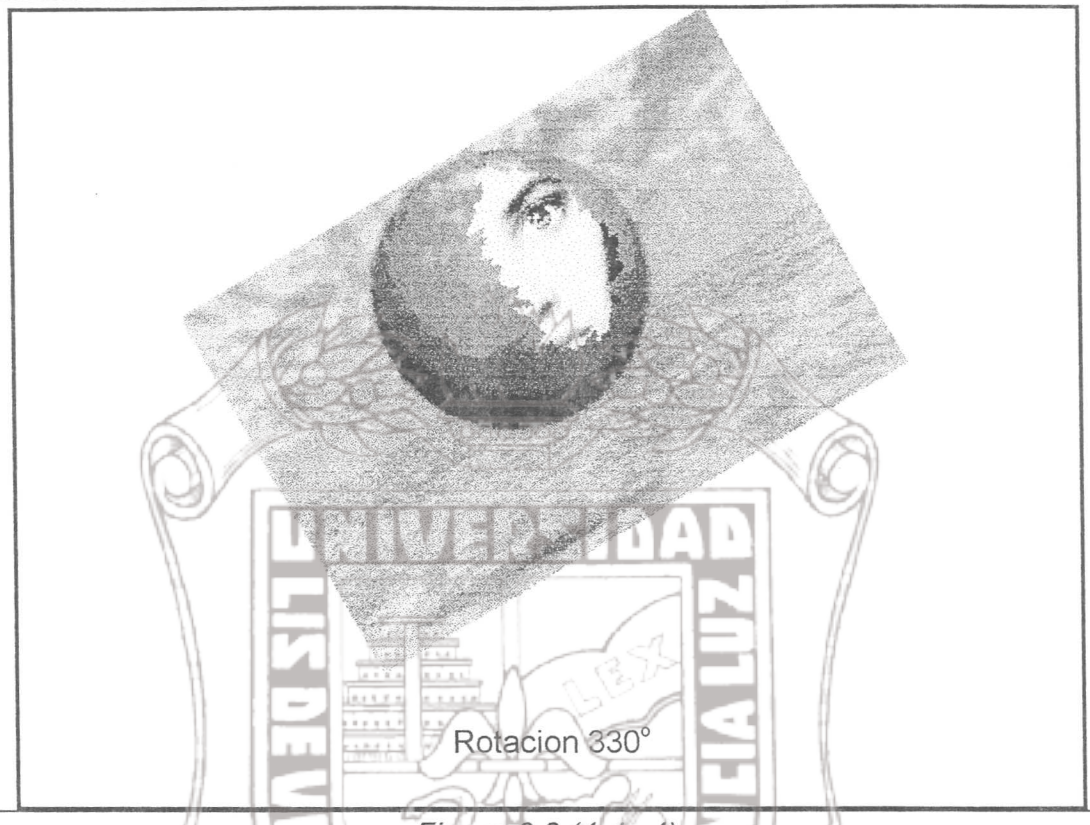


Figura 6.3.(4 de 4)

6.2. Rotacion con dos ángulos (eslabonamiento).

6.2.1. Descripción general.

Proceso similar al de rotación de una figura rectangular. La diferencia es la relación entre renglónes y columnas de pixels no es de 90° , ahora la relación se da por ángulos en α para las columnas y β para los renglónes, como se puede ver en la siguiente Figura 6.4.

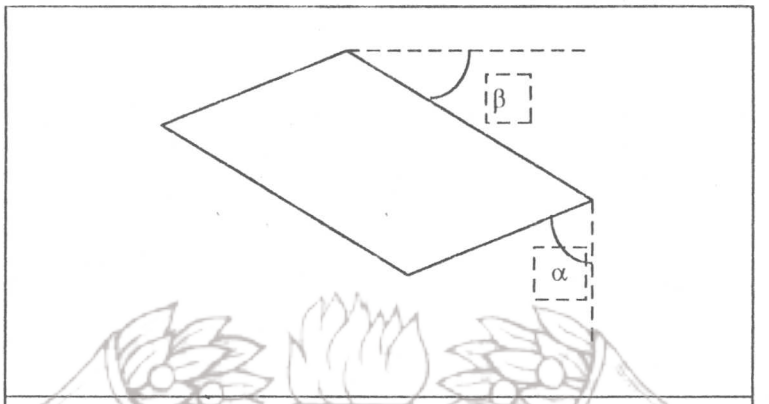


Figura 6.4. Ángulos aplicables al eslabonamiento

Esta característica es importante, ya que en esencia la imagen conserva los mismos renglónes de pixels, solo que estos estan "desfazados" de tal forma que cada inicio de renglón se ajusta por el ángulo de la columna.

El inconveniente "puntos blancos" que presentó la rotación simple, es el mismo que se tiene para este algoritmo, por lo que se resuelve de igual forma, es decir repitiendo algunos renglónes para evitar puntos blancos.

6.2.2. Parámetros de entrada.

x1, y1	Coordenada inicial de imagen fuente
x2, y2	Coordenada final de imagen fuente
xd, yd	Coordenada destino
angx	Ángulo de rotación para renglónes
angy	Ángulo de rotación para columnas

6.2.3. Algoritmo.

1.-Determinar el cuadrante destino para la rotación, así como parámetros de ajuste para puntos blancos

2.-Recorre la figura por renglón

3.-¿El punto de inicio del renglón es diferente al anterior?

Si: Ir a 4

No: Ir a 9

4.-Recorre cada punto del renglón elegido

5.-¿El punto a copiar de imagen fuente es diferente al punto anterior?

Si: Ir a 6

No: Ir a 7

6.-Copia pixeles (calculado y ajustado) a coordenada destino

7.-Incrementa coordenada del pixel a pintar según el cuadrante

8.-¿Terminó de recorrer el renglón?

Si: Ir a 9

No: Ir a 4

9.-Incrementa coordenada del inicio del renglón según cuadrante

10.-¿Terminó de recorrer todos los renglones?

Si: Ir a 11

No: Ir a 2

11.-Fin.

6.2.4. Implementación.

CODIGO	COMENTARIOS

int xt1=0,yt1=0,xt2=0,yt2=0;
 	
int x,y,cx,cy;
int xant,yant,xant2,yant2;
 	
float sax,cax,say,cay;
 	
float dcx,dsx,dsy,dcy;
 	
char flagx=0,flagy=0;
 	
char a,b,a2,b2;
	• Coordenada destino inicio, de cada renglon a rotar se controla por el ciclo exterior e interior
	• Control de ciclos
	• Se usan para guardar el valor de la columna y renglon pintados anteriormente, si este valor es igual al renglon o columna a pintar, entonces por if se evita el pintado
	• Tendran el seno y coseno respectivamente del angulo de la columna y renglon a rotar
	• Lleva el incremento acumulado de seno y coseno para cada elemento de renglon, asi como el inicio de la columna.
	• Indica el cuadrante de columna y renglon en que dibuja la fig. a rotar
	• Establece los valores de ajuste de pixels a pintar para evitar "puntos blancos", en renglon como en col.

if(angx<0) angx=360+angx; if(angy<0) angy=360+angy;	Si el ángulo en columnas o renglónes es negativo suma 360°
if 0<angx<=90 {flagx=1; a=1;b=-1;} if 90<angx<=180 {flagx=2; a=-1;b=-1;} if 180<angx<=270 {flagx=3; a=-1; b=1;} if 270<angx<=360 {flagx=4; a=1; b=1;}	Determina el cuadrante del ángulo para renglónes y los valores de ajuste de a (para columna) y b (para renglónes).

angx=angx*3.1416/180;
cax=cos(angx); sax=sin(angx);
dsx=sax; dcx=cax;
	Convierte ángulo X en radianes
	Calcula seno y coseno de este ángulo
	Inicia acumuladores de seno y cos.

if 0<angy<=90 {flagy=1; a=1; b=-1;} if 90<angy<=180 {flagy=2; a=-1; b=-1;} if 180<angy<=270 {flagy=3; a=-1; b=1;}	Determina el cuadrante del ángulo para columnas y los valores de ajuste de a (para columna) y b (para

if 270<angy<=360 {flagy=4; a=1; b=1;}	renglónes).
angy=angy*3.1416/180;	Convierte ángulo Y en radianes
cay=cos(angy); say=sin(angy);	Calcula seno y coseno de este ángulo
dsy=say; dcy=cay;	Inicia acumuladores de seno y cos.
xt1=round(dsx); yt1=round(dcx);	Inicia acumuladores de control de
xant2=0; yant2=0;	columnas y renglónes
xt2=round(dcy); yt2=round(dsy);	
for (y=y1; y<=y2; y++)	Ciclo que recorre las columna
if(xant2!=xt2||yant2!=yt2)	¿Cambio coord. anterior de columna?
dcx=cax;	Si: inicializa acumuladores de control
dsx=sax;	de renglónes
xt1=round(dcx);	
yt1=round(dsx);	
xant=0; yant=0;	
for (x=x1; x<=x2; x++)	Ciclo que recorre los renglónes
if ((xant!=xt1||yant!=yt1))	¿Cambio coord. anterior de renglón?
color= getpixel(x+a,y+b)	Copia pixel vecino en probable punto
putpixel(xd+xt2+xt1+1,	blanco
yd+yt2+yt1,color);	Copia pixel en punto rotado.
color= getpixel(x,y);	
putpixel(xd+xt2+xt1,yd+yt2+yt1,	
color);	
xant=xt1;	Guarda coord. anterior del renglón
yant=yt1;	
dcx+=cax; dsx+=sax;	Incrementa acumulador de ángulo
switch(flagx)	Evaluacion de flagx en renglón
case 1:if xt1<round(dcx) xt1++;	Cuadrante 1
case 2: if xt1>round(dcx) xt1--;	Cuadrante 2
case 3: if xt1>round(dcx) xt1--;	Cuadrante 3
case 4:if xt1<round(dcx) xt1++;	Cuadrante 4
xant2=xt2; yant2=yt2;	Guarda coord. anterior del renglón
switch(flagy)	Evaluacion de flagy en columna
case 1:dsy-=say;	Cuadrante 2
dcy+=cay;	Incrementa o decrementa según
if (xt2>round(dsy)) xt2--;	cuadrante
if (yt2<round(dcy)) yt2++;	
case 2: dcy+=cay;	Cuadrante 3
dsy-=say;	Ident.
if (xt2>round(dsy)) xt2--;	
if (yt2>round(dcy)) yt2--;	

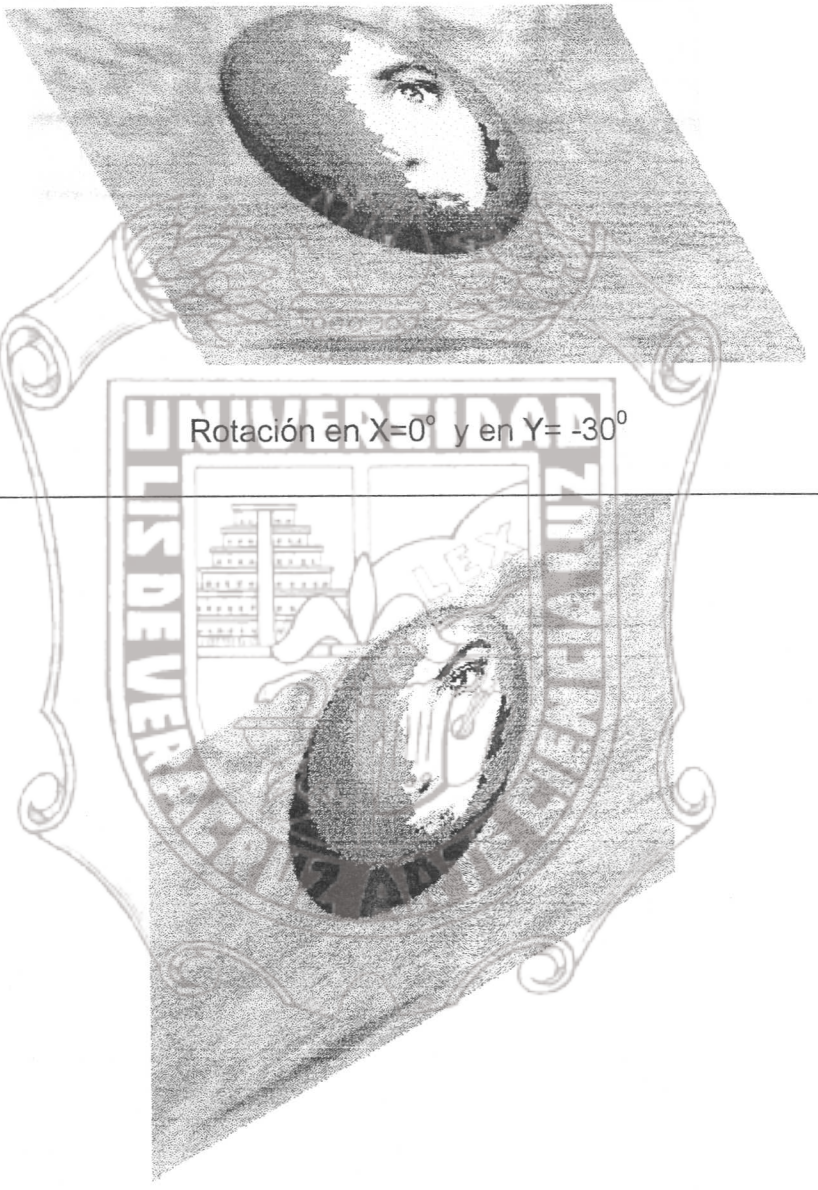
case 3: dcy+=cay;
dsy-=say;
if (xt2<round(dsy)) xt2++;
if (yt2>round(dcy)) yt2--;
	Cuadrante 4
	Ident.

case 4: dsy-=say;
dcy+=cay;
if (xt2<round(dsy)) xt2++;
if (yt2<round(dcy)) yt2++;
	Cuadrante 1
	Ident.

6.2.5. Muestra del algoritmo

Imagen Original

Figura 6.5.(1 de 3) Muestra de rotación con 2 angulos en una imagen



Rotación en X=0° y en Y= -30°

Rotación en X=-30° y en Y= 0°

Figura 6.5. (2 de 3)

Figura 6.5 (3 de 3)

Capítulo VII
Efectos Ondulatorios

"Caminante no hay camino, se hace camino al andar"

VII. EFECTOS ONDULATORIOS.

7.1. Función seno.

7.1.1. Descripción general.

El algoritmo consiste en aplicar la función seno a cada una de las líneas que componen el mapa de bits. El aplicar la función seno, da como resultado un movimiento ondulatorio continuo y uniforme. Para comenzar el trazado de una onda, es necesaria la siguiente información : ángulo inicial y altura de monte (Figura 7.1).

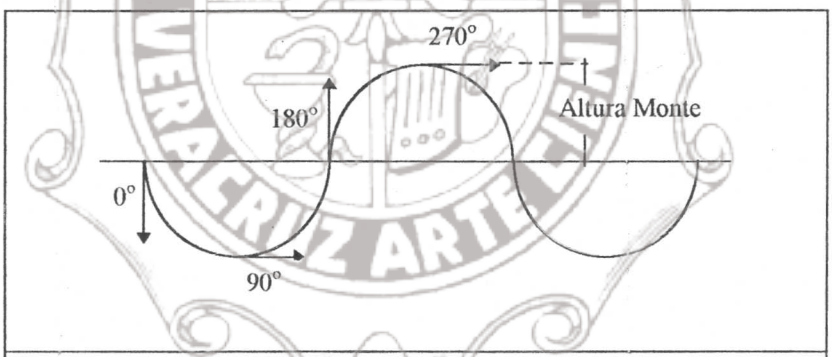


Figura 7.1. Datos iniciales para una función senoidal

Si a los parámetros iniciales en lugar de darles valores fijos continuos en todas las ondas, se usan valores incrementales, tendremos una onda senoidal no uniforme (Figura 7.2).

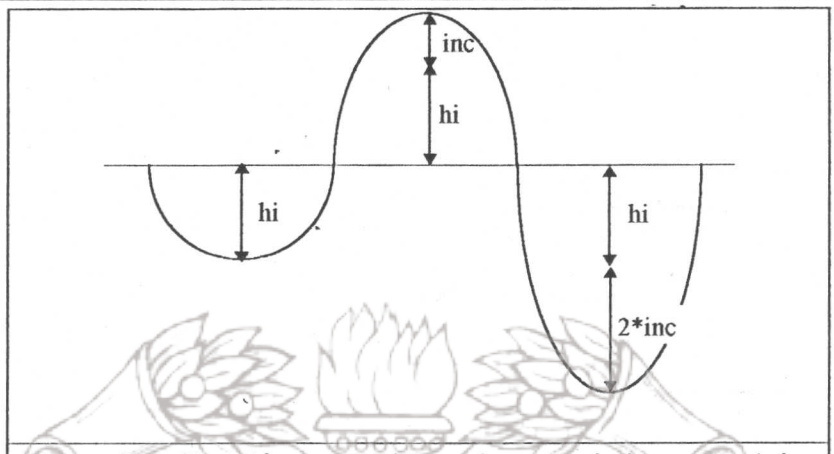


Figura 7.2. Función con valores de entrada incrementales

Observe que la altura inicial de la onda es aumentada a partir de la segunda, en función de un incremento establecido. Con esto obtendremos una onda continua no uniforme.

7.1.2. Parámetros de entrada.

Los parámetros de entrada son los siguientes:

x1,y1	Coordenada inicial de la imagen fuente
x2,y2	Coordenada final de la imagen fuente
xd,yd	Coordenada destino, punto del cual empieza el despliege.
angi	Ángulo inicial con que comenzará la curva
incang	Incremento angular, incrementos mayores producen mas montes
alti	Altura inicial del monte
incalt	Incremento de altura de monte

7.1.3. Algoritmo.

1.- Recorre renglon N de la figura.

2.- Calcula seno ángulo

3.- Determina signo del seno del ángulo

4.- Recorre M columna del renglón N

5.- Calcula variante aplicable a coordenada Y

6.- Pinta el pixel leido de la imagen fuente en coordenada modificada

7.- ¿ Hubo cambio de signo ?

Si: Ir a 8.

No Ir a 9.

8.- Incrementa altura

9.- ¿Terminaron las columnas del renglón N ?

Si : Ir a 12

No: Ir a 10

10.- Incrementa M

11.- Ir a 4

12.-¿ Terminaron todos los renglónes ?

Si: Ir a 15.

No:Ir a 13

13.- Incrementa N

14.- Ir a 1

15.- Fin.

7.1.4. Implementación.

CODIGO	COMENTARIOS
int x,y,t,xt,a,a1; float sen,signo; for(y=y1; y<=y2; y++)	Variables de control de ciclos y acumuladoras Ciclo para recorrido de renglónes

xt=xd; a=angi; a1=alti;
sen=sin(a*3.1416/180);
if(sen==0)signo=0;
else signo=sen/fabs(sen);
 	
for(x=x1; x<=x2; x++, a+=incang)
	Calcula seno del ángulo
	Si seno es 0 inicializa signo con 0
	Sino con el signo del seno en el cuadrante
	Ciclo para el recorrido en el renglón

t=yd+round(a1*sen);
if(sen==0) a1+=incalt;
else if(signo!=sen/fabs(sen))
{ signo=sen/fabs(sen);
a1+=incalt; };
putpixel(xt++,y+t,getpixel(x,y));
sen=sin(a*3.1416/180);
	Calcula variacion de Y
	Si seno es 0 es cambio de onda e incrementa altura
	Si no verifica cambio signo de seno
	Si : reasigna valor de signo
	incrementa altura
	Pinta pixel en coordenada calculada
	Calcula seno para nuevo ángulo

7.1.5. Muestra del algoritmo

Angulo inicial	=0	Incremento angular	=1
Altura inicial	=15	Incremento de altura	=0

Figura 7.3.(1 de 2) Muestra de algoritmo de función seno en una imagen.

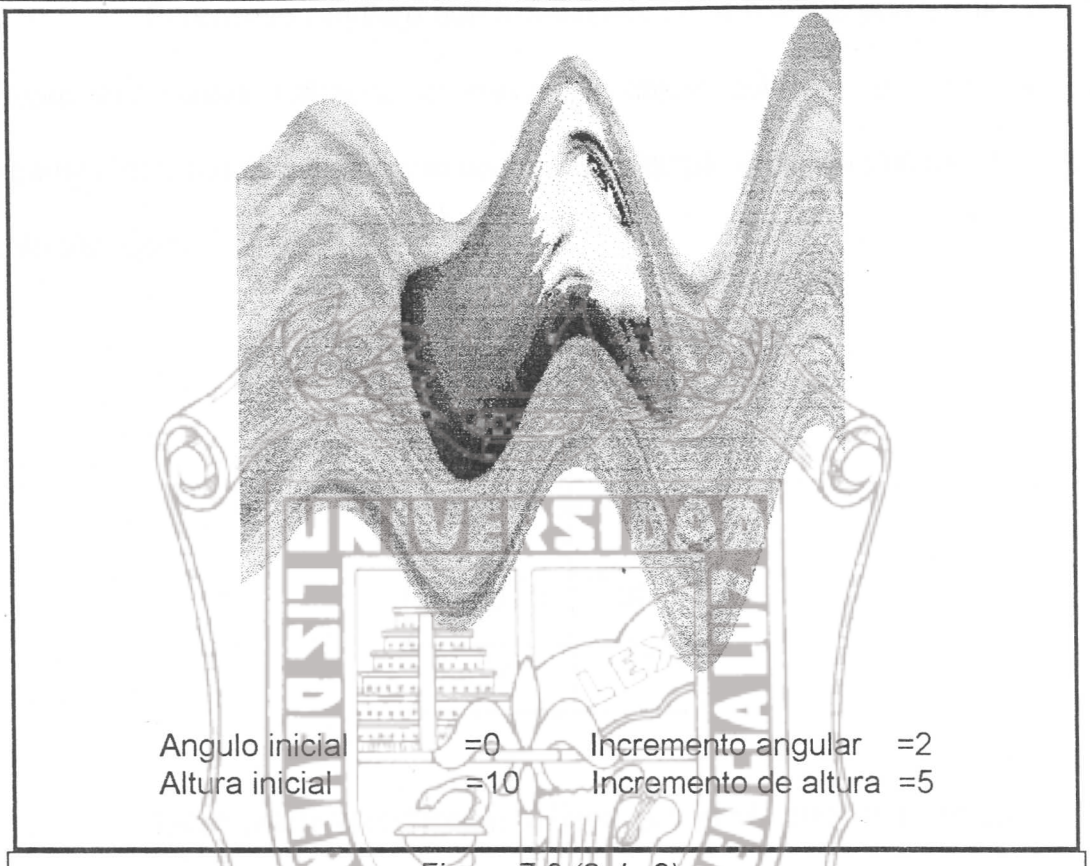


Figura 7.3.(2 de 2)

7.2. Curva de Bezier.

7.2.1. Descripción general.

Para esta transformación se creo un objeto, por la complejidad que implica la aplicación de las curvas de Bezier a un mapa de bits. Como el proceso que requiere esta transformación necesita realizar algunas operaciones antes de aplicarse a la imagen, se necesitaron banderas para asegurar que se elaboraron todos los pasos previos necesarios.

El proceso requiere que incialmente se le den los puntos de control sobre los cuales trabajará la curva. El objeto contiene algunos valores predefinidos, modificables por el usuario.Un ejemplo de estos puntos de control iniciales seria:

X	Y
30	50
40	30
50	90
60	30
90	50

Estos valores producirian la línea que se observa en el inciso Figura 7.4a Después de ajustar esta línea a una curva de bezier quedaría como la curva ilustrada en la Figura 7.4b.

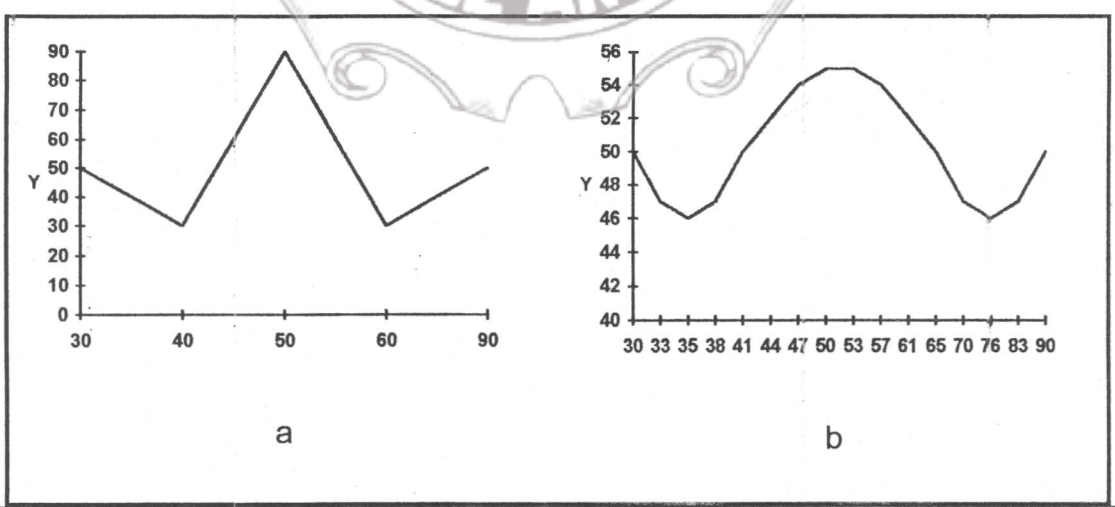


Figura 7.4. Puntos de control aproximados por curva Bezier

La característica principal de la curva de Bezier es "suavizar" la línea haciendo que los extremos de la curva coincidan con el primer y último punto de control, cosa que es importante para este método, dado que los bordes del mapa de bits esta bien delimitado y no puede ser cambiado.

Con esta idea inicial, se debe subdividir una imagen en columnas de bits y después aplicar un proceso de deformación que en graficación vectoria se le llama "*corte*".

El proceso consiste en afilar dos vertices opuestos de un rectángulo, haciendo que giren α grados, sin dejar de ser paralelas dos líneas opuestas. En la implementación de este algoritmo, el giro se da solo en las líneas horizontales de la imagen, tal como lo muestran las Figura 7.5.

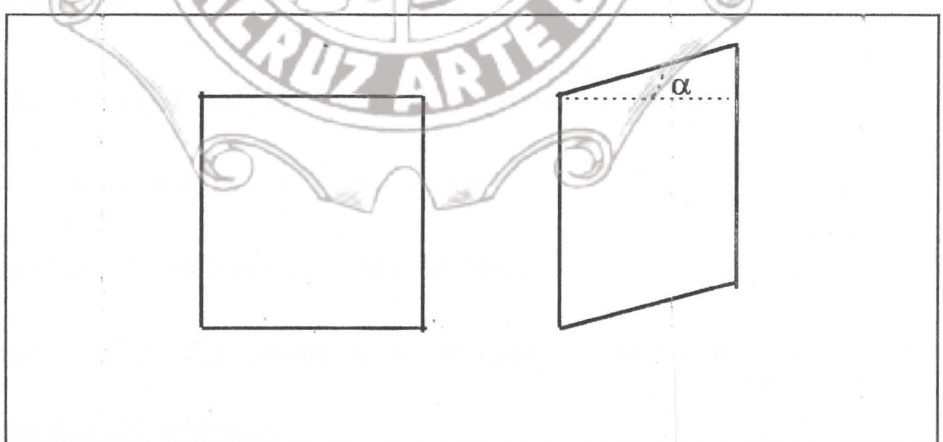


Figura 7.5. Afilamiento de vertices en una sección de una imagen

El ángulo del giro que debe realizarce en cada segmento del mapa de bits, se calcula en función de los segmentos que componen la curva de Bezier obtenida a partir de los puntos de control.

El resultado del proceso anterior se puede observar en la Figura 7.6.

7.2.2. Parámetros de entrada.

xoi, yoi	Coordenada inicial de la imagen fuente.
xof, yof	Coordenada final de la imagen fuente.
xdi, ydi	Coordenada inicial de la imagen destino.
n	Numero de puntos de control
m	Numero de puntos calculados
x[n],y[n]	Coordenadas de puntos de control

7.2.3. Algoritmo.

1.- Construir el objeto definiendo cuantos puntos se calcularán.

2.- Determinar los puntos de control a usar.

3.- Calcular puntos en base a puntos de Bezier.

4.- Aplicar puntos calculados a la imagen en base proceso de corte a segmentos de la imagen.

7.2.4. Implementación.

CODIGO	COMENTARIOS
Datos Privados	
int x[maxcontrol],y[maxcontrol];	x[] , y[] arrays de puntos de control
int control[maxcontrol][2];	estructura que tiene puntos calculados
int n;	n es num. de puntos de control
int m;	m es num. de puntos en la curva
char ok_pctrl,ok_curva;	Banderas de procesos previos
Métodos Privados	
void inicializa(int n,...);	Recibe puntos de control para x[] , y[]
void calcula_curva(void);	Calcula puntos en estructura "control"
void corte(int xoi,int yoi,int xof,int yof, int xdi, int ydi, int ydf);	Aplica el efecto "corte" a una rebanada de imagen.
Métodos Publicos	
trans_bez(int p);	Constructor que inicializa la cantidad de puntos calculados a usar.
void aplica(int xoi,int yoi, int xof, int yof, int xdi, int ydi);	Función que aplica la transformación a la imagen. Usa calcula_curva y corte
friend void muestra_bez(trans_bez &, char *);	Función que muestra la transformación y permite al usuario cambiar los puntos de control

7.2.5. Muestra del algoritmo

Imagen Original

X	Y
20	50
30	80
40	20
50	70
60	30
70	80
80	50

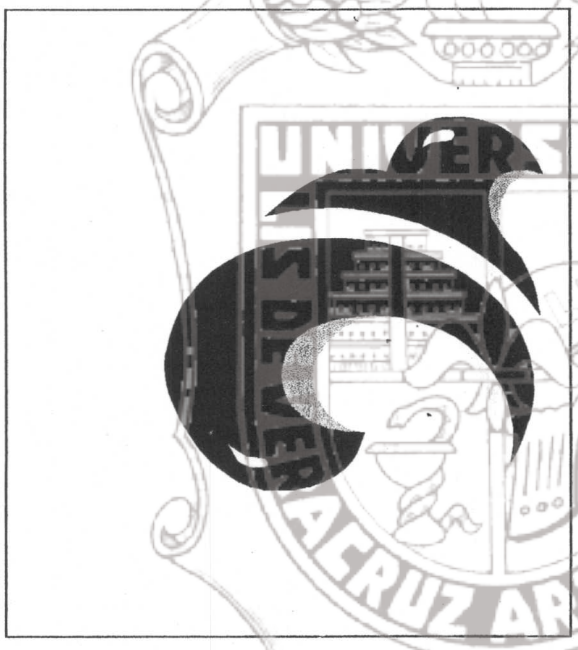
Curva Bezier de puntos anteriores

Figura 7.6 (1 de 2)Imuestra de ajuste de imagen a curva Bezier

Imagen ajustada

Figura 7.6.(2 de 2)

Instituto de Ingeniería
Universidad Veracruzana



Capítulo VIII
Malla de Eslabonamiento

"El show debe continuar"

Proyecto de Digitalización de Tesis
Responsable M.I. Alberto Pedro Lorandi Medina
Colaboradores: Estanislao Ferman García
M.I. Enrique Rodríguez Magaña

VIII. TRANSFORMACION USANDO MALLA DE ESLABONAMIENTO.

8.1. Descripción general.

La transformación consiste en crear una malla imaginaria que permita ajustar la segmentos de la imagen fuente, mediante eslabonamiento a la malla definida. Para lograr esto es necesario determinar la longitud y ángulo que tendran los segmentos de la primera columna y el primer renglón. Los segmentos posteriores se obtiene por medio de proyección paralela de los segmentos base. La Figura 8.1 muestra claramente los segmentos base.

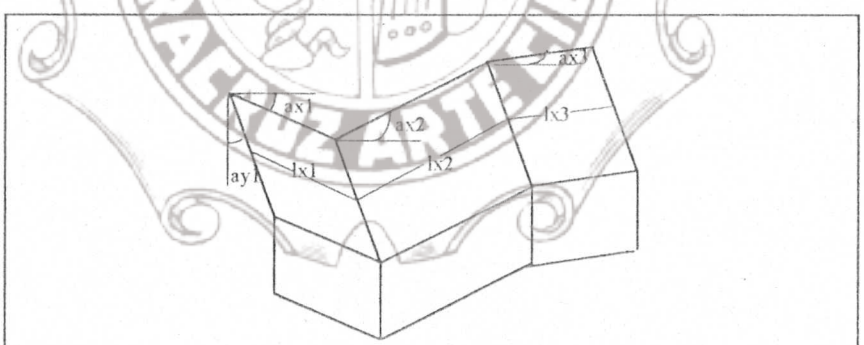


Figura 8.1. Establecimiento de Segmentos Base

El primer renglón y la primera columna de la malla lo constituyen los segmentos base. Cada segmento base cuenta con una longitud (o altura) y un ángulo con respecto a la horizontal (vertical para segmentos base en columna).

Con los datos iniciales se cálcula por paralelismo los demas puntos de la malla. Dichos puntos sirven de base para saber como se segmentará la imagen original.

8.2. Parámetros de entrada.

desx[REN]	Descripción (ángulo y longitud) de lo segmentos base del primer renglón de la malla.
desy[COL]	Descripción (ángulo y longitud) de los segmentos base de la primera columna de la malla.
col, ren	Cantidad de columnas y renglónes en la malla
pix,piy	Coordenada de imagen fuente
pdx,pdy	Coordenada destino de la malla

8.3. Algoritmo.

1.- Mostrar descripción de columnas y renglónes al usuario.

2.- Verificar si desea usar estos valores o dar otros nuevos.

3.- Si cambiaron puntos recalcula la malla

4.- Pinta la malla para referencia.

5.- Aplica la malla calculada a la imagen fuente, haciendo que cada segmento de la malla se "eslabone" (capitulo 6).

8.4. Implementación.

CODIGO	COMENTARIOS
Datos Privados	
int col,ren;	Establece el número de renglónes y columnas
char initOK;	Bandera que checa la inicialización de malla
char mallOK;	Bandera que checa la malla
struct l_a { int lon; float ang;} desx[REN], desy[COL];	Estructura que tiene la longitud y el ángulo de cada segmento. desx y desy segmento en el primer renglón y la primera columna
float float senx[COL],cosx[COL], float seny[REN], cosy[REN];	Guarda el seno y conseno para cada renglón
struct punto mall[REN+1][COL+1];	Define la malla de puntos que divide la imagen fuente.
int pix,piy,pdx,pdy;	Coordenadas de imagen fuente y destino de malla.
Métodos Publicos	
malla(int c,int r);	Constructor que inicializa la malla.
void calcula();	Función que calcula los puntos en la matriz
void pinta();	Función que pinta la malla calculada
void aplica();	Función que aplica la figura a la malla
friend void muestra_malla(malla &, char *);	Función externa que permite determinar la longitud y ángulo de cada elemento base de la malla.

8.5. Muestra del algoritmo

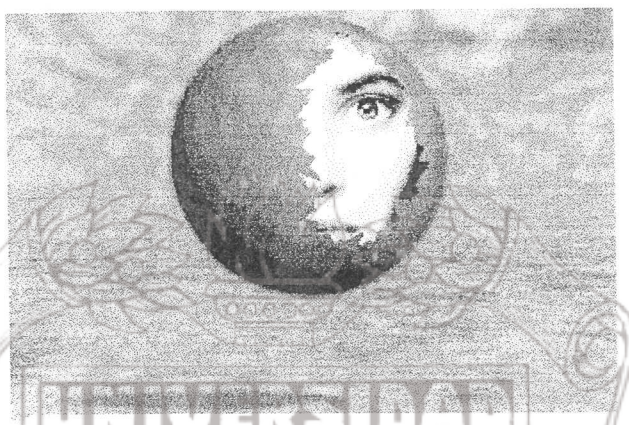
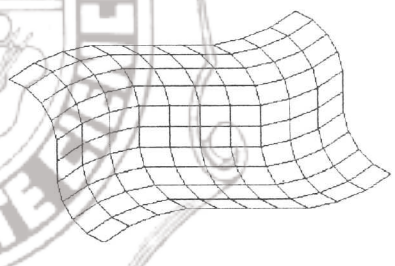


Imagen Original

Columnas		Renglones	
Long	Ang	Long.	Ang.
30	-30	20	-60
30	-20	20	-45
30	-10	20	-30
30	0	20	-15
30	0	20	0
30	0	20	0
30	0	20	-15
30	-10	20	-30
30	-20	20	-45
30	-30	20	-60



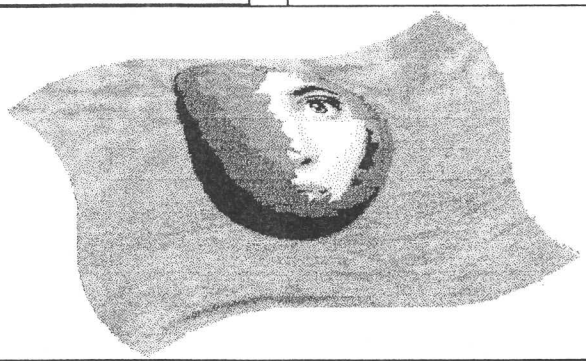
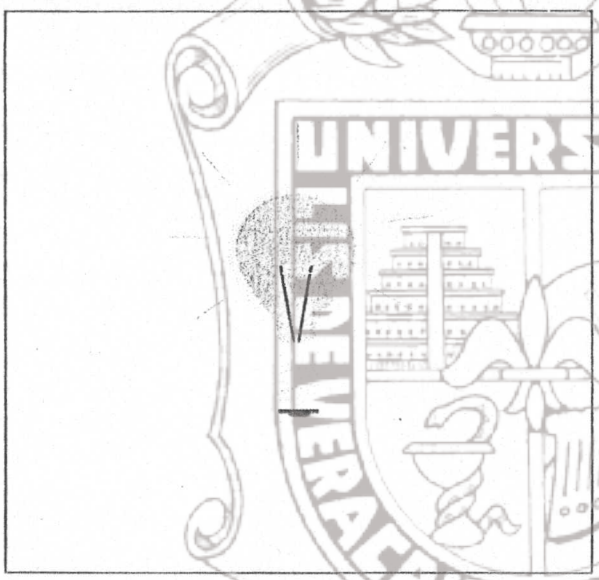


Figura 8.2. Muestra de imagen ajustada a una malla de elabonamiento.



V Conclusiones y Recomendaciones

"Todo lo que sube debe de bajar,

todo lo que inicia debe terminar"

CONCLUSIONES Y RECOMENDACIONES.

Conclusiones

En la actualidad son cada vez mas los programas de edición de imagenes mapeadas, que estan incursionando en nuevos campos como la transformación física de las imagenes, ademas de las ya conocidas opciones de manejo cromatico. Estas nuevas características seran una parte comun en los programas de edición en breve tiempo, sin embargo actualmente solo algunos de ellos pueden dar características amplias de deformación.

En cuanto a la premisa establecida sobre la conservación de la calidad de la imagen, se logro cumplir esta hipotesis, ya que la calidad proporcionada por estos algoritmos, en las imagenes finales no varia del original. A pesar de aplicarse algunas transformaciones que "rompen" la imagen en secciones, los algoritmos siempre ajustan de la mejor manera posible las secciones con lo que la imagen pierde muy poca nitidez.

Obviamente el cumplimiento de la hipotesis mencionada anteriormente tiene su precio en rendimiento, aunque no es tanto como para prohibir su uso en PC. Teniendo la ventaja de que aun se puede aplicar optimizaciones o trucos que permitan ahorrar algo de tiempo en los algoritmos.

Recomendaciones.

Existen aun mucho campo de desarrollo en este sentido, de hecho se piensa trabajar en la ampliación de algunos conceptos, como la transformación basada en curvas de Bezier para lograr implementar una malla que permita la deformación (no solo el eslabonamiento) total de cada segmento de la malla.

Otros caminos estan en la aplicación (o creación) de una malla de Bezier tridimensional, en la cual la tercera dimensión estaría representada por los colores, con lo que se podrian lograr efectos de profundidad en una imagen fotográfica.

Las dos ideas anteriores son algunos ejemplos de los caminos que se pueden seguir en este sentido.

La inovación en la mayoria de las ocasiones, no consiste mas que en tomar algunos elementos que ya se tenian a la mano y tratar de unirlos de forma no comun. De estas uniones podemos encontrar cosas interesantes las cuales pueden resolver problemas .

Por último, el impacto que puede tener una inovación radica simplemente en su aplicación a la solución de un problema.

Como en el "Principito" este es un camino tan cercano y tan lejano a la vez.

BIBLIOGRAFIA

Berger,M. -*"Graficación por Computadora con Pascal"*. Addison Wesley Iberoamerica.1991.

Demetel,J.T./Miller,M.J.-*"Graficas por computadora"*.Mc Graw Hill.1984.

Hearn,D./Baker,M.P. - *"Graficas por Computadora"* . Prentice Hall Hispanoamerica. 1993.

Hill,F.S. - *"Computer Graphics"*. Ed. Maxwell Macmillan International Editions.1990.

Lehmann, C.H. - *"Geometria Analítica"*. Ed. Limusa

Ortiz,K.R. -*"Desarrollo e implelemtación de una herramienta de software: Interfaz Grafica Avanzada para Microcoputadora"* . Tesis Licenciatura I.S.C., I.T.V.. 1991.

Plastock,R.A./Kalley,G.- *"Graficas por computadora"*. Schaum MacGraw Hill.1986.

Vijay Nagassamy/ Noshir A, Langrana.- *"Engineering Drawing Processing and Vectorization System"*. Computer Vision, Graphics and image Processing, No. 49, 1990.