

UNIVERSIDAD DON BOSCO

**Programa de Reconocimiento y Clasificación de
Imágenes**

Trabajo de graduación

**Preparado para la
Facultad de Estudios Tecnológicos**

**Para optar al Grado de
Técnico en Ingeniería en Computación
Opción Analista Programador**

**Presentado por
Carlos Alberto Montalvo Lardé**



Ciudadela Don Bosco, septiembre de 2001

UNIVERSIDAD DON BOSCO

Rector

Ing. Federico Huguet

Vice-rector Académico

Lic. Baltasar Díaz

Secretario General

Lic. Mario Olmos

Decano de la Facultad de Estudios Tecnológicos

Ing. Víctor Cornejo

Asesor del trabajo de graduación

Ing. Óscar G. Durán

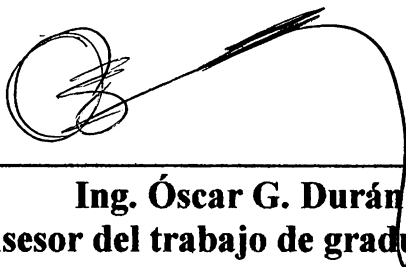
Jurado evaluador

Ing. Héctor Carías

Jurado evaluador

Téc. César Melgar

UNIVERSIDAD DON BOSCO



Ing. Óscar G. Durán
Asesor del trabajo de graduación



Ing. Héctor Carías
Jurado evaluador



Téc. César Melgar
Jurado evaluador

Dedicatoria

A mis padres:

Héctor y Lorena

A mi hermana:

Patricia Lorena

A mi sobrina:

Andrea Patricia

**A todas las personas que detestan la mediocridad y buscan la
superación y la excelencia en su diario vivir**

Índice general

Introducción.....	3
Antecedentes.....	4
Descripción del proyecto	6
Situación actual.....	6
Situación propuesta.....	7
Justificación del proyecto	9
Objetivos.....	10
Objetivo general.....	10
Objetivos específicos.....	10
Alcances y limitaciones	11
Alcances.....	11
Limitaciones.....	12
Metodología general.....	13
Desarrollo del proyecto.....	14
Descripción técnica del entorno de desarrollo.....	14
Comparación entre el sistema de coordenadas gráficas y el plano cartesiano.....	15
Fundamentos teóricos del algoritmo de reconocimiento.....	17
Otros métodos para el procesamiento y análisis de imágenes.....	21
Fundamentos teóricos del algoritmo de clasificación y almacenamiento.....	23
Implementación del proyecto	25
Descripción del algoritmo de reconocimiento.....	25
Peculiaridades del algoritmo de reconocimiento.....	27
Descripción del algoritmo de clasificación y almacenamiento	30
Glosario técnico.....	32
Bibliografía.....	35
Anexos	36



Índice de figuras

Figura 1: Cronograma del proyecto	7
Figura 2: Diagrama de flujo general del proyecto	8
Figura 3: Sistemas de coordenadas: a) Plano cartesiano; b) Pantalla gráfica.....	16
Figura 4: Imagen de un círculo: a) Tamaño original; b) Aumentada 10 veces	17
Figura 5: Polígono representado por las coordenadas cartesianas de sus vértices	18
Figura 6: Polígono representado por las coordenadas polares de sus vértices	19
Figura 7: Gráfica de la función resultante de un cuadrado.....	19
Figura 8: Jerarquías: a) De elementos textuales; b) De polígonos (triángulos).....	24
Figura 9: Imágenes para analizar: a) Cuadrado; b) Triángulo isósceles.....	28
Figura 10: "Delta" con el vértice mayor a 180° señalado.....	29
Figura 11: Polígono segmentado	30



Introducción

El presente proyecto consiste en elaborar y poner a prueba un programa capaz de reconocer y clasificar imágenes sencillas, específicamente figuras geométricas en blanco y negro.

Este documento se divide en seis apartados. Primero se presentan los antecedentes acerca de la disciplina de reconocimiento de imágenes en computación. Luego se detalla la situación actual observada en relación al software existente de reconocimiento de imágenes; también se plantea la situación propuesta por medio de la elaboración de este proyecto. A continuación, y de cara al desarrollo de programas de este tipo en nuestro medio, se exponen los motivos detrás del desarrollo del proyecto. Después se describen los objetivos generales y específicos del programa. Además se define lo que el programa será capaz de realizar y lo que se considera fuera de sus posibilidades. Finalmente, se describe el funcionamiento general del programa y el análisis propuesto de su eficacia y eficiencia.

Al final de este documento se encuentra un glosario de términos técnicos relacionados con el reconocimiento de imágenes y este mismo proyecto.

Antecedentes

En el campo de las ciencias de la computación existen diversas ramas, una de las cuales es la computación cognitiva, también conocida como inteligencia artificial. Esta rama se divide en diversas áreas de estudio, como por ejemplo la computación neural y el reconocimiento de patrones.

Las primeras investigaciones sobre el reconocimiento de patrones se efectuaron al empezar a utilizarse las computadoras para análisis estadísticos. En los años 60 y 70 aparecieron numerosos estudios y aplicaciones sobre el reconocimiento de tendencias o patrones estadísticos para áreas como economía y demografía. A partir de la década de 1980, gracias en parte a la gran difusión de computadoras cada vez más poderosas y asequibles y al surgimiento de los primeros sistemas operativos gráficos, la idea de una interacción natural entre los usuarios y las computadoras comenzó a despertar un interés generalizado. Dentro del área de reconocimiento de patrones aparecieron nuevos objetos de estudio, entre ellos el reconocimiento de voz e imágenes.

Una aplicación práctica del reconocimiento de imágenes ha sido el reconocimiento óptico de caracteres, u OCR por sus siglas en inglés. Los primeros programas OCR comerciales aparecieron a finales de los años 80, impulsados por la disponibilidad de scanners y otros tipos de hardware para la digitalización de imágenes. Estas primeras aplicaciones OCR eran poco eficaces, pero las versiones actuales de estos programas realizan el reconocimiento de caracteres con gran exactitud, pudiendo diferenciar entre diferentes tipos de letra e incluso varios idiomas.

Dentro del ámbito de reconocimiento de imágenes, el OCR es una de las ramas más conocidas. Sin embargo, en la actualidad existen una gran cantidad de aplicaciones diferentes del reconocimiento de imágenes. Un ejemplo es el uso de cámaras de video para el control de robots industriales. Otro lo constituye el análisis computarizado de imágenes en medicina, o el procesamiento de fotografías del espacio en astronomía.

Descripción del proyecto

Situación actual

El software de reconocimiento de imágenes puede clasificarse en dos grandes categorías: programas OCR comerciales y programas especializados de reconocimiento de imágenes.

Actualmente, el software OCR comercial es capaz no sólo de transformar texto impreso en digital, sino que en muchos casos también conserva el formato del documento original. Sin embargo, estos programas no están diseñados para manipular imágenes o símbolos no textuales de la misma forma que el texto.

Por otra parte, los programas especializados trabajan con imágenes sin contenido textual en formas muy específicas: desde la lectura de códigos de barra impresos hasta la manipulación digital de video. Muchos de estos programas tienen como objetivo la interpretación y tratamiento de imágenes a nivel numérico, es decir, alterar las imágenes mismas o extraer de ellas algún tipo de datos simples.

Ambos tipos de software, al menos en sus versiones comerciales, generalmente no poseen la capacidad de realizar un análisis semántico, o sea, la asociación de imágenes con conceptos y atributos. Sin embargo, existen programas experimentales y educativos que sí tienen alguna capacidad de análisis y aprendizaje.

En la Universidad Don Bosco se han desarrollado algunos trabajos de graduación a nivel de ingeniería electrónica, relacionados con el reconocimiento de imágenes. Uno de ellos consiste en una aplicación práctica del reconocimiento de imágenes provenientes de una cámara de video para el control de un brazo robot; otro es un sistema combinado de software y hardware de video para la captura y procesamiento de imágenes, sin capacidades de reconocimiento.

Situación propuesta

El programa descrito en este documento pretende precisamente vincular imágenes con nombres o términos mediante la creación de una base de datos que será capaz de almacenar dichas imágenes junto con sus atributos en forma ordenada. Esto tiene como objetivo crear un sistema de reconocimiento supervisado por el usuario y basado en la comparación de imágenes y la asociación de ciertos atributos. Dicho sistema podría servir de base para programas más complejos, como procesadores de imágenes, juegos educativos y otros.

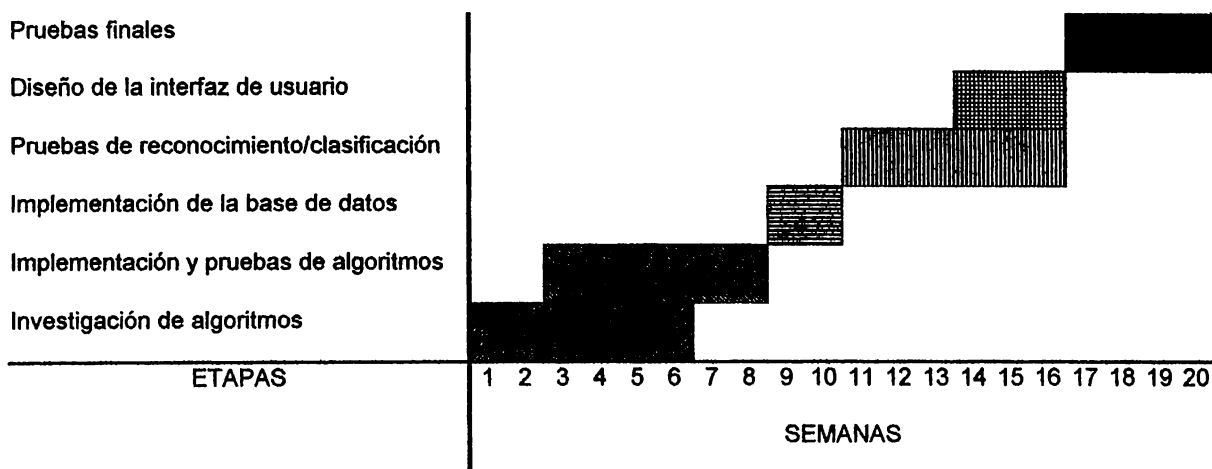


Figura 1: Cronograma del proyecto

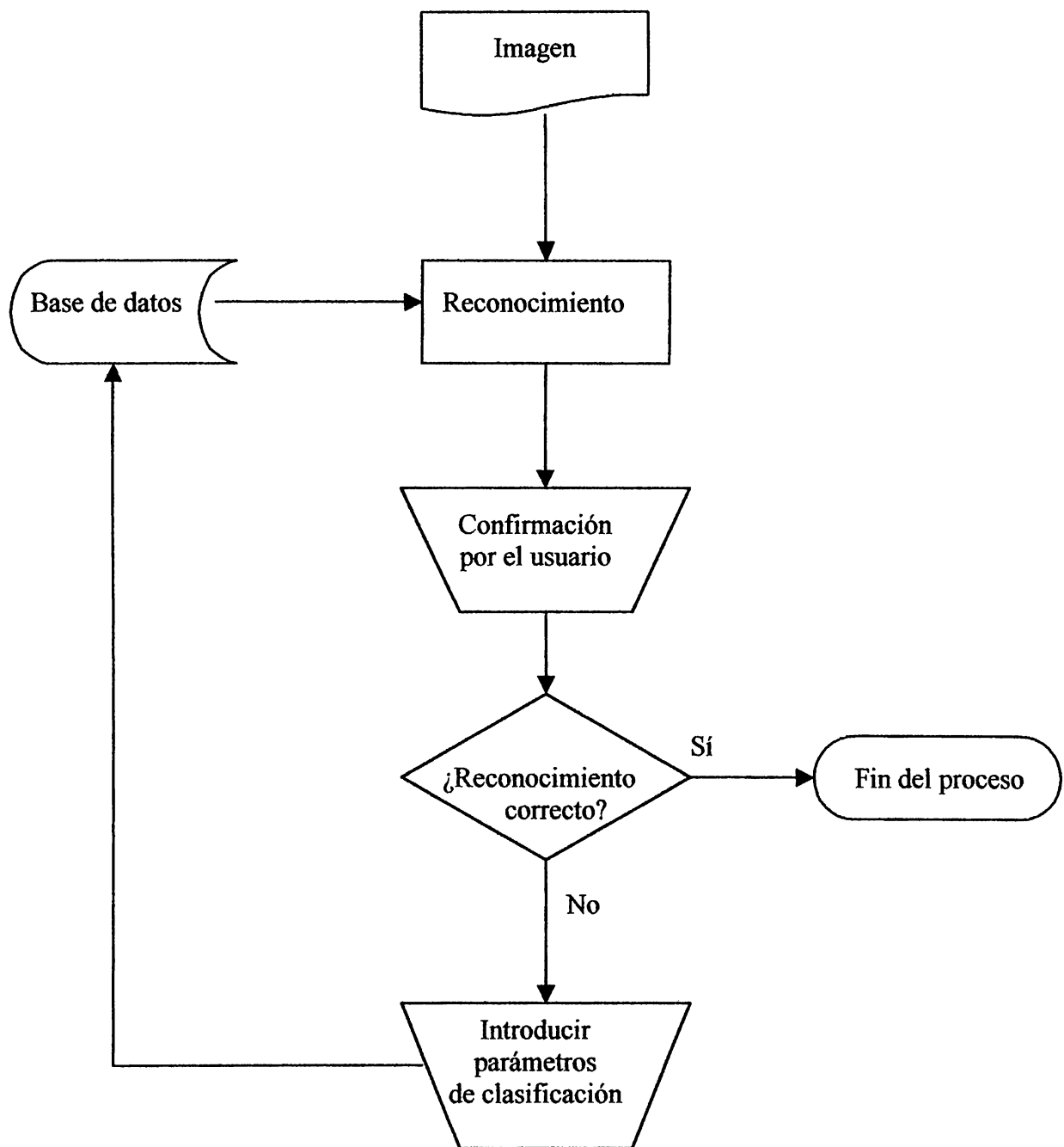


Figura 2: Diagrama de flujo general del proyecto

Justificación del proyecto

En nuestro medio es escasa la labor de investigación y desarrollo de aplicaciones relacionadas con inteligencia artificial en general y, por lo tanto, con el reconocimiento de imágenes. Además, la adopción de tecnologías relacionadas, como el OCR, ha sido lenta y poco difundida. En realidad son pocas las instituciones que utilizan en forma extensiva información digitalizada, pues todavía se hace uso de documentos impresos para procesos que muchas veces se beneficiarían con la digitalización de la información.

El software disponible comercialmente para este fin, por lo general tiene un precio considerable y está pensado para procesamiento de texto de uso general. Existen programas especializados para ciertas tareas de reconocimiento de imágenes, como se ha mencionado anteriormente, pero la mayoría son aplicaciones no comerciales que son de uso restringido o muy costosas. Tanto las aplicaciones comerciales como las especializadas son desarrolladas en el extranjero, y son distribuidas por un número limitado de empresas, algunas de las cuales son dueñas de las patentes.

Por las razones antes mencionadas, el programa propuesto en este proyecto se basa en una arquitectura de código abierto, utilizando componentes de distribución libre. La estructura del programa se ha pensado como un punto de partida para programas más sofisticados en diversas áreas, entre las cuales se podría mencionar el procesamiento de imágenes, procesamiento de formularios, análisis de gráficos, juegos, tutores y otros.

Objetivos

Objetivo general

Elaborar y poner a prueba un programa capaz de reconocer y clasificar imágenes sencillas, específicamente figuras geométricas en blanco y negro. El tipo de figuras geométricas que será procesado comprende polígonos regulares e irregulares con un número de lados no mayor a 10.

Objetivos específicos

- Elaborar el módulo de reconocimiento de imágenes.
- Elaborar el módulo de clasificación y almacenamiento de imágenes.
- Ejecutar pruebas para comprobar y/o mejorar la exactitud del reconocimiento de las imágenes.

Alcances y limitaciones

Alcances

- El programa podrá manejar imágenes compuestas cada una de un polígono regular o irregular, con un máximo teórico de 10 lados.
- Será capaz de almacenar en su base de datos el arquetipo o paradigma de cada tipo de figura y sus nombre o término descriptivo, estos últimos proporcionados por el usuario.
- El proceso de identificación involucrará el conteo de los vértices del polígono, la longitud de sus lados y las distancias entre los vértices y el centro geométrico; los datos mencionados son suficientes para determinar el tipo de polígono analizado.
- Tendrá la capacidad de distinguir entre polígonos diferentes y también variantes de un mismo polígono, identificándolos por su tipo y atributos.
- Capacidad de aprendizaje: a medida que el programa procese cada vez más imágenes, éste incrementará su capacidad de reconocimiento.

Los tipos de polígonos que se espera poder clasificar e identificar son: triángulos (equiláteros, isósceles, escalenos), cuadriláteros (cuadrados, rectángulos, rombos, etc.) y demás polígonos regulares e irregulares con 5 a 10 lados.

Limitaciones

- Las imágenes procesables por el programa se limitarían a polígonos sólidos en blanco y negro. Esto excluye el uso de imágenes complejas, como fotografías o dibujos elaborados.
- Las imágenes deberán provenir de archivos almacenados en disco, es decir, no se utilizará ningún método para capturar imágenes reales.
- Cada imagen no podrá contener más de una figura (polígono) y debe ser del tamaño suficiente como para poder distinguir su forma.¹
- Debido a que el reconocimiento será realizado en base a los parámetros geométricos de polígonos, esto excluye del análisis a las figuras formadas por curvas cerradas como círculos y elipses.

¹ Para las imágenes de prueba, se han escogido dimensiones de 250 x 250 píxeles en promedio, área suficiente para que un polígono sea distinguible claramente, tanto por un observador humano como por este programa. Esto no quiere decir que no se pueda procesar imágenes en tamaños menores.

Metodología general

El programa funcionará de la siguiente manera: Partiendo de archivos de imágenes previamente creados, tratará de identificar los polígonos presentes en ellos. Si se reconoce, dará su descripción; para el caso de figuras no reconocidas, el programa pedirá su descripción al usuario, que será almacenada para poder identificar figuras similares posteriormente. De esta forma, el programa “aprenderá” a reconocer imágenes.

La exactitud del programa se medirá a través del número de imágenes correctamente reconocidas. En la base de datos serán almacenados solamente los arquetipos o paradigmas de cada tipo básico de polígono reconocible. Dado que es imposible almacenar en una base de datos cada variante posible de una imagen, ni siquiera de polígonos simples, el programa tendrá un algoritmo para analizar la forma (número y longitud de sus lados, etc.) de cada imagen procesada, de manera que sea posible obtener un nivel alto de exactitud al tiempo que se almacena un mínimo de imágenes en la base de datos.

Otro factor a considerar para la evaluación del programa será la velocidad del procesamiento de las imágenes. Para esto se utilizarán polígonos de diferentes tamaños y formas, dentro de los límites expresados anteriormente. El objetivo es poder completar el proceso en el mínimo tiempo posible. Sin embargo, debido a la naturaleza experimental de este proyecto y a su alcance restringido y específico, el desempeño de los algoritmos usados en el programa no será comparado con ningún otro programa existente. Tampoco está previsto que personas ajenas al desarrollo de este proyecto lo evalúen o efectúen pruebas.

Desarrollo del proyecto

Descripción técnica del entorno de desarrollo

El programa ha sido escrito en su totalidad en lenguaje C y se ejecuta bajo el sistema operativo GNU/Linux. Utiliza la librería C de GNU (glibc), versión 2.2.2, y la librería DAFS para administración de documentos e imágenes, versión 1.0. La compilación del programa se ha efectuado con el compilador C de GNU (gcc), versión 2.96. La versión del sistema operativo GNU/Linux empleado es la 8.0 de Linux Mandrake.

Adicionalmente se ha utilizado software adicional como auxiliar para el desarrollo de este programa, el cual también se ejecuta bajo Linux y es detallado a continuación:

- Nedit, versión 5.1.1. Editor de texto para programadores, con el cual fue escrito el código fuente del programa.
- Xpaint, versión 2.6. Programa básico de dibujo, usado para crear las imágenes de prueba para el programa.
- Illuminator, versión 0.10b. Visualizador de documentos e imágenes con formato TIFF monocromático, basado en la librería DAFS. Fue utilizado fundamentalmente para verificar la compatibilidad de los archivos de imágenes con dicha librería y para examinar la base de datos de imágenes.
- Gnuplot, versión 3.7.1. Generador de gráficas en múltiples formatos, empleado como herramienta de diagnóstico para graficar y verificar la exactitud de los datos procesados por el algoritmo reconocedor de imágenes.

Algunas características relevantes del hardware utilizado para elaborar y probar el programa son:

- PC genérica (clon), compatible con IBM
- Procesador Intel Celeron, 500 Mhz
- 128 MB de memoria RAM
- Disco duro de 20 GB, con 4.4 GB de espacio asignado al sistema operativo Linux

Comparación entre el sistema de coordenadas gráficas y el plano cartesiano

Las computadoras con capacidades gráficas, como las PCs de IBM y compatibles, representan las imágenes en pantalla como una matriz de píxeles. Dicha matriz posee un sistema de coordenadas similar a un plano cartesiano, cuyo origen está ubicado en la esquina superior izquierda de la pantalla.

Sin embargo, existen 3 diferencias relevantes entre el el sistema de coordenadas cartesiano y el utilizado para graficar en una PC. La primera es que el plano cartesiano en su totalidad se compone de 4 cuadrantes, mientras que la matriz gráfica sólo posee uno (en sentido estricto), el cual comprende toda el área disponible para mostrar la información gráfica, es decir, el área de la pantalla. La segunda diferencia consiste en la dirección del eje vertical o eje "y", el cual en un plano cartesiano es positiva hacia arriba del origen y negativa hacia abajo; en la matriz gráfica, este eje es positivo hacia abajo, lo cual implica que el valor en "y" es igual a cero en la parte superior de la pantalla y va aumentando su valor hacia abajo.

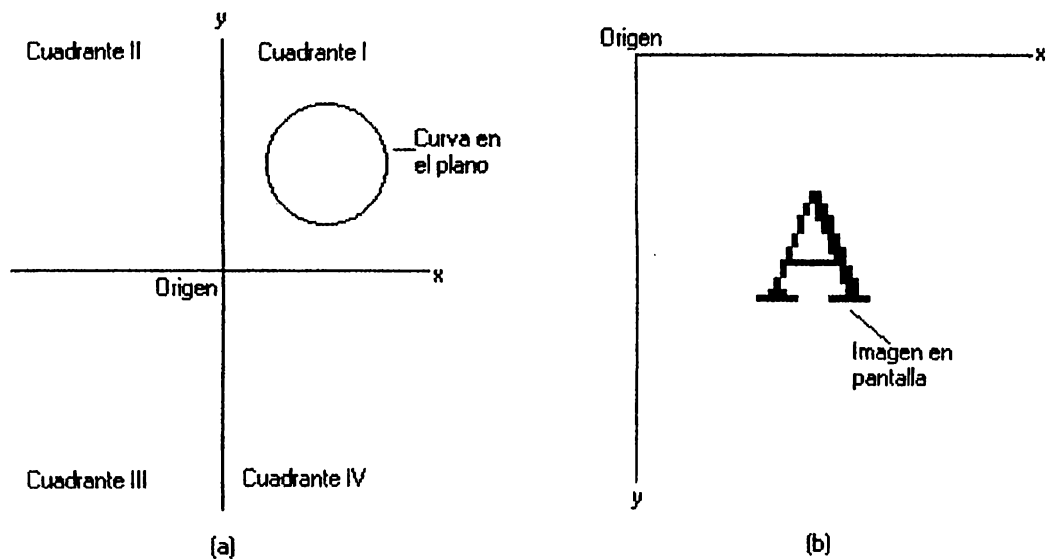


Figura 3: Sistemas de coordenadas: a) Plano cartesiano; b) Pantalla gráfica

La tercera diferencia es determinante para el análisis y la solución del problema planteado, como se verá más adelante. Cada punto en el plano cartesiano posee una ubicación dada por un par de coordenadas "xy", pero el punto en sí no tiene área, por lo que sólo es una referencia en el plano. Por otra parte, la unidad más pequeña para representar imágenes en la pantalla es el pixel. Cada pixel también tiene un par de coordenadas asignadas; sin embargo, ocupan un espacio físico, cuyas dimensiones están determinadas por la resolución de la pantalla. Por esto, la pantalla gráfica es en realidad una matriz de pixeles que tienen una forma más o menos cuadrada. Esto da lugar al conocido "efecto de escalera" o "borde de sierra", el cual se observa claramente en imágenes que han sido agrandadas. Los bordes de las imágenes en una PC no son uniformes y suaves, sino que presentan discontinuidades. Este fenómeno se ilustra con la siguiente figura:

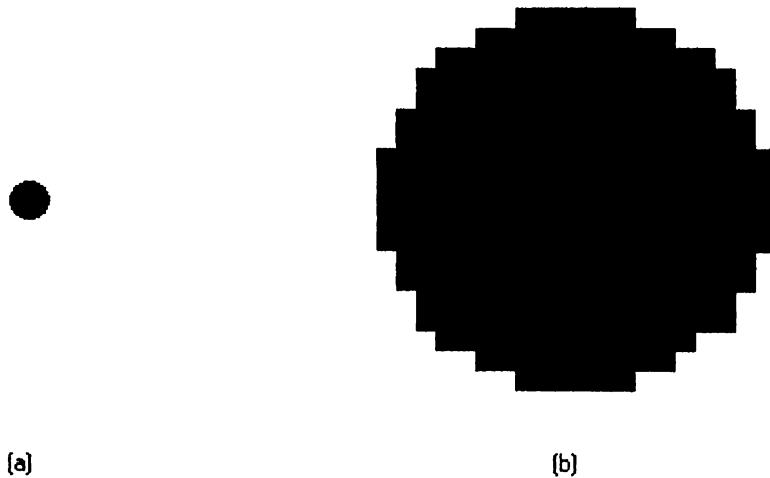


Figura 4: Imagen de un círculo: a) Tamaño original; b) Aumentada 10 veces

Fundamentos teóricos del algoritmo de reconocimiento

Existe una variedad de métodos para describir figuras en un sistema de coordenadas², específicamente los polígonos, que son el tipo de figuras a analizar en este trabajo. En general, los polígonos pueden representarse como una serie de puntos en el plano, conectados por segmentos de rectas cuyo número es igual a la cantidad de puntos. Dichos segmentos forman una curva cerrada que no se corta a sí misma. Los segmentos forman los lados del polígono, y los puntos constituyen sus vértices. En esencia, cualquier polígono puede ser representado en base a estos parámetros.

² En lo sucesivo, se usará como referencia el sistema de coordenadas de la pantalla gráfica.

Una ventaja de este método consiste en ofrecer una representación intuitiva desde un enfoque de percepción espacial. En consecuencia, una persona puede percibir la figura representada fundamentalmente en base a la ubicación de sus vértices. Un ejemplo de este tipo de representación se puede observar en la siguiente figura:

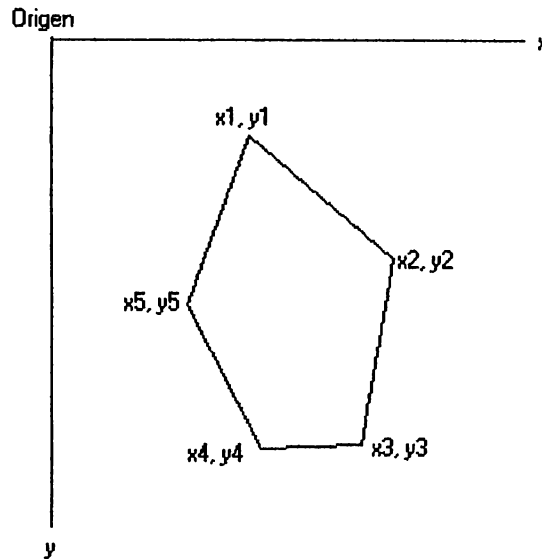


Figura 5: Polígono representado por las coordenadas cartesianas de sus vértices

Otro método consiste en representar un polígono como una función periódica, cuyos valores son las coordenadas polares de los puntos que conforman los lados y vértices. En teoría, el origen de las coordenadas polares puede ser elegido arbitrariamente en el plano. Sin embargo, y en el caso particular de este programa, con el fin de simplificar el proceso de análisis, el origen ideal de las coordenadas polares es el centro geométrico del polígono. Este método no es intuitivo, pero simplifica el análisis computacional con respecto al método anterior. A continuación se presenta un ejemplo ilustrado:

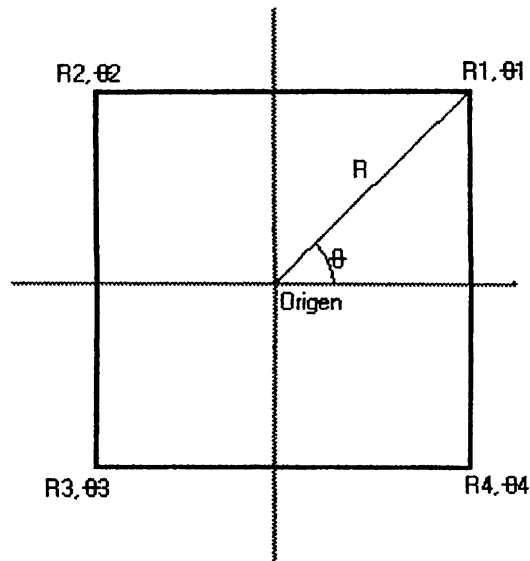


Figura 6: Polígono representado por las coordenadas polares de sus vértices

Al conocer las coordenadas polares de los puntos que conforman el polígono, es posible obtener la función correspondiente, que es periódica en 2π radianes (o 360 grados). La gráfica de dicha función correspondiente a un cuadrado similar al de la figura 6 se muestra aquí:

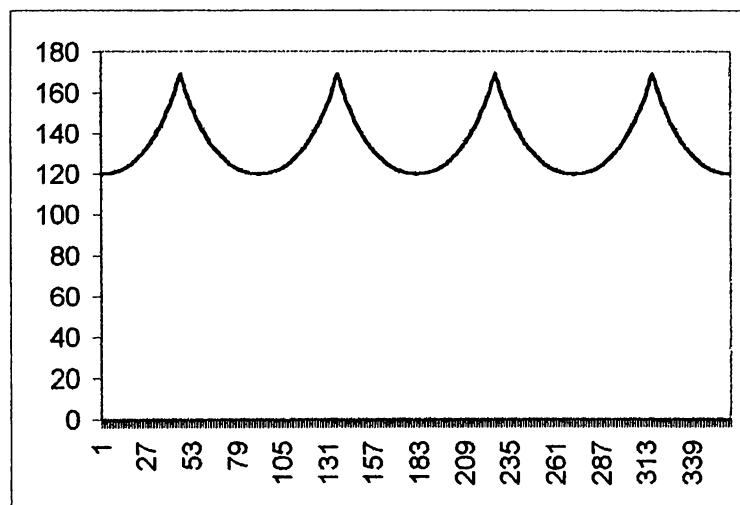


Figura 7: Gráfica de la función resultante de un cuadrado

En la gráfica anterior, el eje horizontal representa el ángulo θ , y el eje vertical significa la distancia (o radio) R . Los cuatro picos indican la presencia de 4 vértices equidistantes del origen o centro geométrico de la figura. La distancia entre picos indica la longitud de los lados del cuadrado, que son iguales.

Para gráficas de R y θ de cualquier polígono, las siguientes características son de importancia para su análisis:

- El número de picos en la gráfica indica el número de vértices del polígono.
- La distancia entre picos determina la longitud de los lados del polígono.
- El valor de R indica la distancia entre un punto dado y el centro geométrico.
- Debido a que la función es periódica, rotar el polígono en relación a su centro solamente desplaza la gráfica, sin afectar sus valores.

Este método para el análisis de las figuras ha sido escogido principalmente por su simplicidad en términos matemáticos y su idoneidad para el limitado número de características que se necesita identificar de la imagen. Además no es necesario ningún procesamiento previo que modifique la imagen que se va a examinar.

A pesar de todo, este método presenta ciertos problemas en cuanto a su implementación práctica, que serán estudiados más adelante.

Otros métodos para el procesamiento y análisis de imágenes

A través de los años se han desarrollado una enorme variedad de algoritmos para procesar y analizar imágenes computarizadas de los más diversos tipos. La gran mayoría de estos algoritmos están elaborados para su aplicación en imágenes digitalizadas, provenientes de fuentes externas. En otras palabras, se utilizan con imágenes de objetos reales obtenidas por medio de cámaras de video o scanners. Algunos algoritmos son mejores que otros para analizar ciertos tipos de imágenes o para extraer determinadas características, por ello se dice que estos algoritmos son especializados. En términos estrictos, no existe ningún método o algoritmo universal que sea aplicable en cualquier tipo de imagen o situación.

Existen varias categorías en las cuales se pueden agrupar dichos algoritmos. Una distinción fundamental podría hacerse entre los algoritmos que sólo realizan algún tipo de preprocesamiento de la imagen, otros que se limitan al análisis de rasgos o características, y un tercer tipo que combina ambas funciones.

Otro tipo de categorización se basa en la función específica que realizan los algoritmos. A continuación se muestra una lista parcial de aspectos o funciones de análisis, mencionando ejemplos de los algoritmos respectivos:

- **Resaltado de bordes por filtrado:** Grupo de algoritmos que filtran una imagen utilizando una máscara (llamada también ventana o núcleo) cuyo tamaño usual es de 3 x 3 píxeles. Esta máscara "barre" la imagen, pixel por pixel, resaltando los bordes presentes. Estos algoritmos están pensados para imágenes en escalas de gris; como ejemplos pueden citarse el filtro de Sobel, de Prewitt, de Roberts, etc.

- **Detección de bordes:** Algoritmos utilizados conjuntamente con alguno de los anteriores (junto con un proceso de umbralizado) para determinar la ubicación exacta de los bordes de la figura. En esta categoría se destaca la transformada de Hough, un algoritmo que convierte la imagen binaria (blanco y negro) a su representación paramétrica, utilizando una matriz acumuladora. Después del proceso, la matriz contiene puntos con valores considerados como máximos, que indican una línea presente en las correspondientes coordenadas cartesianas. Luego se emplea la transformada inversa de Hough para crear una imagen de las rectas que representan los bordes. Esta imagen resultante usualmente se superpone a la original.
- **Seguimiento de bordes:** Algoritmos que utilizan un método diferente para detectar bordes, el cual consiste en "recorrer" la figura, registrando la ubicación del contorno. Estos algoritmos difieren entre sí en cuanto a los cálculos realizados para tal fin. En esta categoría se mencionan el seguidor diferencial de bordes, el seguidor heurístico de búsqueda gráfica y el seguidor de programación dinámica; ellos trabajan con imágenes en escalas de grises sin procesamiento previo. Otro algoritmo de esta categoría es el seguidor basado en el código cadena, el cual se utiliza en imágenes binarias. Una de sus variantes, llamada "tortuga de Papert modificada", se ha empleado en este proyecto.
- **Análisis de formas:** Esto comprende los algoritmos que describen la forma de una imagen en base a aproximaciones geométricas o a ciertos parámetros. Algunos de ellos son algoritmos de aproximación polinomial y coeficientes descriptores de Fourier.

Fundamentos teóricos del algoritmo de clasificación y almacenamiento

Para facilitar la creación y el manejo de una base de datos de imágenes y atributos, se ha empleado una librería de programación en C llamada DAFS. Su nombre proviene de las siglas de *Document Attribute Format Specification* (especificación de formato para atributos de documentos). DAFS hace referencia tanto a un formato específico de documento como a la librería que permite utilizar dicho formato en programas de C.

El formato DAFS fue diseñado originalmente como un estándar para la representación y almacenamiento de documentos digitalizados. La respectiva librería de C fue puesta a disposición de los programadores en el año 1995, con el fin de promover el desarrollo de aplicaciones compatibles con el estándar DAFS. Sin embargo, dicho estándar no alcanzó popularidad y fue relegado por otros, como el formato Acrobat (PDF) de la empresa Adobe Systems Inc.

DAFS permite la creación de una base de datos jerárquica, compuesta de elementos de imagen que pueden subdividirse en otros, cada uno asociado con un término, atributo o etiqueta que lo describe.

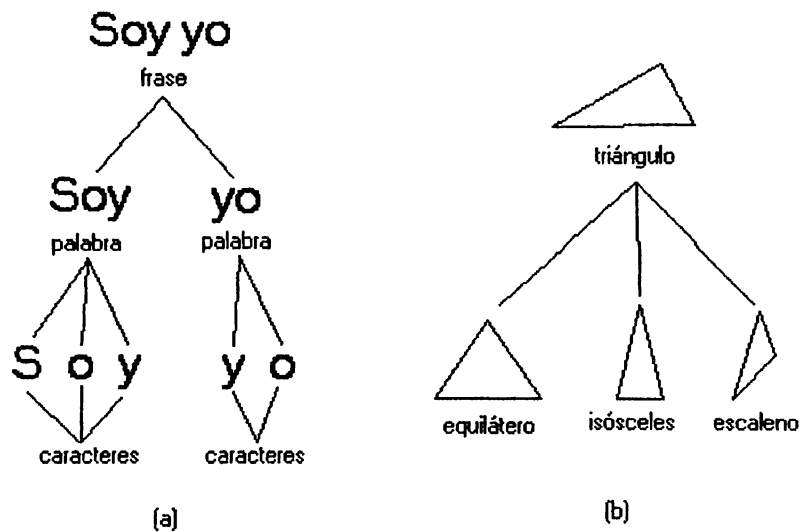


Figura 8: Jerarquías: a) De elementos textuales; b) De polígonos (triángulos)

Las razones por las que se ha elegido a DAFS como soporte de este programa, en lugar de otras alternativas, son fundamentalmente las siguientes:

- La capacidad de representar y relacionar "familias" de elementos mediante una estructura jerárquica de árbol. En el caso de este proyecto, los elementos equivalen a los diferentes tipos de polígonos analizables.³
- La característica de asociar imágenes con uno o más términos o conceptos.
- La librería DAFS está disponible bajo una licencia de código abierto (open-source), que permite su libre uso y distribución.

³ Ver el apartado de "Limitaciones".

Implementación del proyecto

Descripción del algoritmo de reconocimiento

1. El programa solicita el nombre de un archivo de imagen en formato TIFF monocromático.⁴ Se utilizan funciones de la librería DAFS para decodificar el archivo, obtener su tamaño en píxeles y transformar la imagen en un mapa de bits (bitmap) almacenado en la memoria RAM.
2. La información contenida en el mapa de bits, a su vez, es copiada en una matriz de memoria con las mismas dimensiones que la imagen original; cada celda de la matriz corresponde a un píxel de la imagen. La única razón para realizar este paso consiste en simplificar los cálculos trigonométricos posteriores, puesto que en el mapa de bits original hay 8 píxeles "empaquetados" en cada byte de datos.
3. Durante el proceso de creación de la matriz, la imagen es "barrida" fila por fila (de píxeles), hasta llegar al primer píxel del polígono, es decir, el primer píxel negro que se encuentre. Sus coordenadas cartesianas quedan registradas como el punto de partida del seguidor de bordes, llamado "tortuga de Papert modificada".⁵ Se recorre las orillas del polígono hasta llegar al píxel inicial. En el trayecto se utilizan las fórmulas para calcular los momentos de orden 0 y 1, valores que luego sirven para estimar el centro geométrico del polígono.⁶

⁴ El uso de polígonos sólidos mejora dramáticamente la eficacia del seguidor de bordes, el cual suele fallar al tratar de seguir bordes con alguna pendiente y sólo un píxel de ancho

⁵ Cf. Pitas, I., "Digital Image Processing Algorithms", pp. 301-303

⁶ Cf. Gómez-Allende, D., "Reconocimiento de Formas y Visión Artificial", pp. 353-365

4. Después de haber encontrado el centro geométrico, se procede a hacer un nuevo "barrido" de la imagen; esta vez partiendo desde dicho centro hacia el exterior de la imagen. El sentido del barrido es circular, completando una vuelta completa (360°). Durante el barrido se recorre nuevamente el contorno del polígono, registrando las coordenadas polares de cada pixel del contorno, a intervalos de 1° . La función que se encarga de esto se denomina "función radar".⁷
5. Lo siguiente es determinar cuáles de los pixeles del contorno representan vértices potenciales. En teoría, esta tarea consistiría simplemente en buscar una sola vez los "picos"⁸ de la función radar. En la práctica, sin embargo, el efecto escalera que está presente en todos los lados del polígono que no son paralelos con ninguno de los ejes "x" o "y", es el causante de la aparición de varios picos falsos.⁹ Esto hace necesario una serie de búsquedas adicionales, a intervalos angulares crecientes. En cada búsqueda se realiza un conteo de cuántos picos fueron encontrados.
6. Lógicamente, al aumentar el intervalo angular de cada búsqueda, se pierden algunos picos, los cuales pueden ser vértices reales o no. Por esta razón, al final de las búsquedas se calcula la media aritmética "n" de los picos detectados. Los primeros "n" picos cuyos valores de R sean los más altos, son considerados los vértices reales del polígono.
7. En general, si la distancia angular entre vértices es la misma, entonces se considera al polígono como regular; si esto no se cumple, entonces el polígono es irregular.

⁷ Ver fundamentos teóricos

⁸ Un pixel es considerado vértice (potencial) cuando su valor de R es mayor que el de sus vecinos inmediatos

⁹ Ver el siguiente tema: Peculiaridades del algoritmo...

8. Finalmente, si los resultados no concuerdan con los parámetros reales del polígono, por ejemplo, si el número de vértices resulta incorrecto, se da la opción al usuario de introducir manualmente dicho valor.

Peculiaridades del algoritmo de reconocimiento

Las peculiaridades que afectan al algoritmo de reconocimiento son básicamente éstas:

- El "efecto escalera" que presentan todas las imágenes digitales
- La presencia de vértices "cóncavos" (con un ángulo mayor a 180°)
- Figuras que poseen segmentos discontinuos

Como ya se mencionó anteriormente, el sistema de coordenadas gráfico posee la particularidad de que es una matriz de tamaño fijo (para una resolución determinada); por lo tanto, los puntos en el plano (es decir, los píxeles) ocupan efectivamente espacio en la pantalla. Los bordes de las imágenes no son uniformes, presentando un "efecto de escalera" o "bordes de sierra", muchas veces visible sin necesidad de agrandar la imagen. Este fenómeno afecta adversamente la efectividad del algoritmo de reconocimiento propuesto. Para demostrar esto, se puede comparar dos polígonos diferentes, tales como un cuadrado y un triángulo, analizándolos mediante este algoritmo.



Figura 9: Imágenes para analizar: a) Cuadrado; b) Triángulo isósceles

El cuadrado posee lados que son paralelos con los ejes "x" y "y", por lo que son continuos y perfectamente "lisos"; la gráfica de la función radar presenta una serie de parábolas regulares (ver Gráfica 1 en los anexos).

En el caso del triángulo isósceles, sólo la base es paralela con el eje "x" y sus otros lados poseen pendientes muy inclinadas. Dichos lados muestran un marcado efecto de escalera; como consecuencia, la gráfica de la función radar aparece con varios picos falsos que son pequeños y se hallan distribuidos en las partes bajas de la curva (ver Gráfica 2 en los anexos).¹⁰

Al utilizar intervalos mayores para θ , se eliminan muchos de estos picos falsos, como puede verse en la Gráfica 3 en los anexos, que es la misma figura de la Gráfica 2, pero esta vez utilizando un intervalo de 5° . En cuanto al funcionamiento del algoritmo de

¹⁰ Las gráficas 1 y 2 han sido generadas a intervalos de 1° para θ

reconocimiento, el promedio de vértices probables se obtiene luego de varias "pasadas" a intervalos crecientes que van "suavizando" cada vez más la curva. Sin embargo, intervalos muy grandes tienden a generar gráficas lineales en lugar de parabólicas, con la consiguiente eliminación de vértices reales debido a que hay valores de R que son "saltados".¹¹

Una segunda peculiaridad importante consiste en que el algoritmo ignora deliberadamente los "picos negativos" que pueden ser producidos por vértices cuyos lados forman ángulos mayores que 180° . A continuación se muestra la figura de un polígono de 4 lados con un vértice de este tipo:

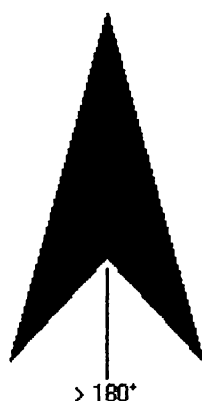


Figura 10: "Delta" con el vértice mayor a 180° señalado

Existen ciertas razones para ignorar este tipo de vértices:

- En términos de la función radar, los picos negativos son muy difíciles de distinguir de entre los valores más bajos de la curva, sobre todo en intervalos grandes (ver Gráfica 4 en los anexos).

¹¹ Como un límite obvio, ningún polígono puede tener menos de 3 vértices

- Las posibilidades de ocurrencia de estos vértices es muy inferior, comparada a la de los vértices normales: como muestra de esto, ningún polígono regular ni triángulo posee este tipo de vértices.

Polígonos que poseen 2 o más de estos vértices constituyen un problema mayor, ya que algunos de ellos son figuras compuestas de varios segmentos que quedan fuera del alcance de la función "radar", como se ve a continuación:

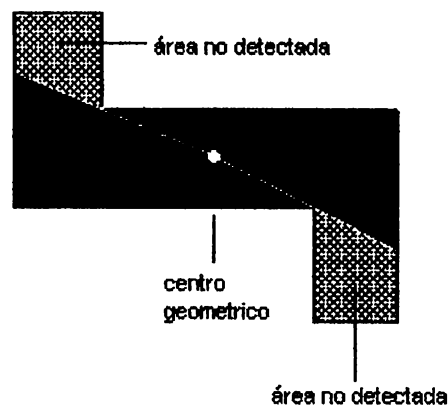


Figura 11: Polígono segmentado

En conclusión, los polígonos que presentan vértices cuyos ángulos son mayores que 180° están fuera de la capacidad de análisis de este algoritmo.

Descripción del algoritmo de clasificación y almacenamiento

1. Después de realizar el proceso de reconocimiento (avalado por el usuario), se realiza lo siguiente:

- a. Si la base de datos de las imágenes está vacía o el polígono identificado es el primero en su "familia" (en cuanto al número de vértices), queda registrado en la base de datos como el primero de su jerarquía.
 - b. Si ya hay registrado algún(os) polígono(s) en su familia correspondiente, se efectúa una comparación de sus atributos (intervalo angular y distancia R de los vértices). En caso de coincidir, la comparación se detiene; si no coinciden, el polígono queda registrado como "miembro" de la familia correspondiente.
2. Todo lo anterior se efectúa bajo la supervisión del usuario, quien se encarga de "etiquetar" (asignar términos) a cada polígono que vaya a ser registrado en la base de datos, por ejemplo: "triángulo equilátero". De la misma forma, confirma o rechaza las operaciones de registro correspondientes.

Glosario técnico

Algoritmo: En general, forma de resolver un determinado problema por medio de una secuencia establecida de pasos. En el contexto de este documento, este término se refiere al proceso computacional para el reconocimiento de imágenes.

Análisis semántico: Estudio acerca de la relación entre un objeto y sus posibles significados. En el campo de la inteligencia artificial, implica el uso de algoritmos para dotar a una computadora de cierta capacidad para reconocer objetos.

Arquetipo: En el contexto de este proyecto, imagen que sirve como base para comparar e identificar imágenes con figuras del mismo tipo.

Código abierto: Forma de distribuir o comercializar software, en la cual se incluye el código fuente del programa. Normalmente se concede el permiso a otras personas ajenas al productor del software para modificar y redistribuir el programa. Este proyecto se basa en programas de código abierto.

Computación cognitiva: Simulación computarizada de los procesos del pensamiento humano.

Computación neural: Simulación computarizada de la estructura y funcionamiento del sistema nervioso.

Digitalización: Proceso de conversión de datos obtenidos del mundo real, normalmente en forma de señales analógicas, a información binaria que puede ser procesada y almacenada por una computadora.

Inteligencia artificial: Véase "Computación cognitiva".

OCR: Siglas en inglés de "reconocimiento óptico de caracteres". Proceso por el cual se extrae texto en un formato inteligible para la computadora, a partir de la imagen de un documento obtenida previamente.

Paradigma: Véase "Arquetipo".

Scanner: Dispositivo capaz de digitalizar diversos tipos de documentos impresos. Se usa principalmente para el procesamiento de imágenes y el reconocimiento óptico de caracteres (OCR).

Reconocimiento de imágenes: Proceso por el cual una computadora es capaz de distinguir y clasificar imágenes diversas, imitando, hasta cierto punto, la capacidad humana de distinguir entre objetos por sus formas. El reconocimiento de imágenes forma parte de la disciplina del reconocimiento de patrones.

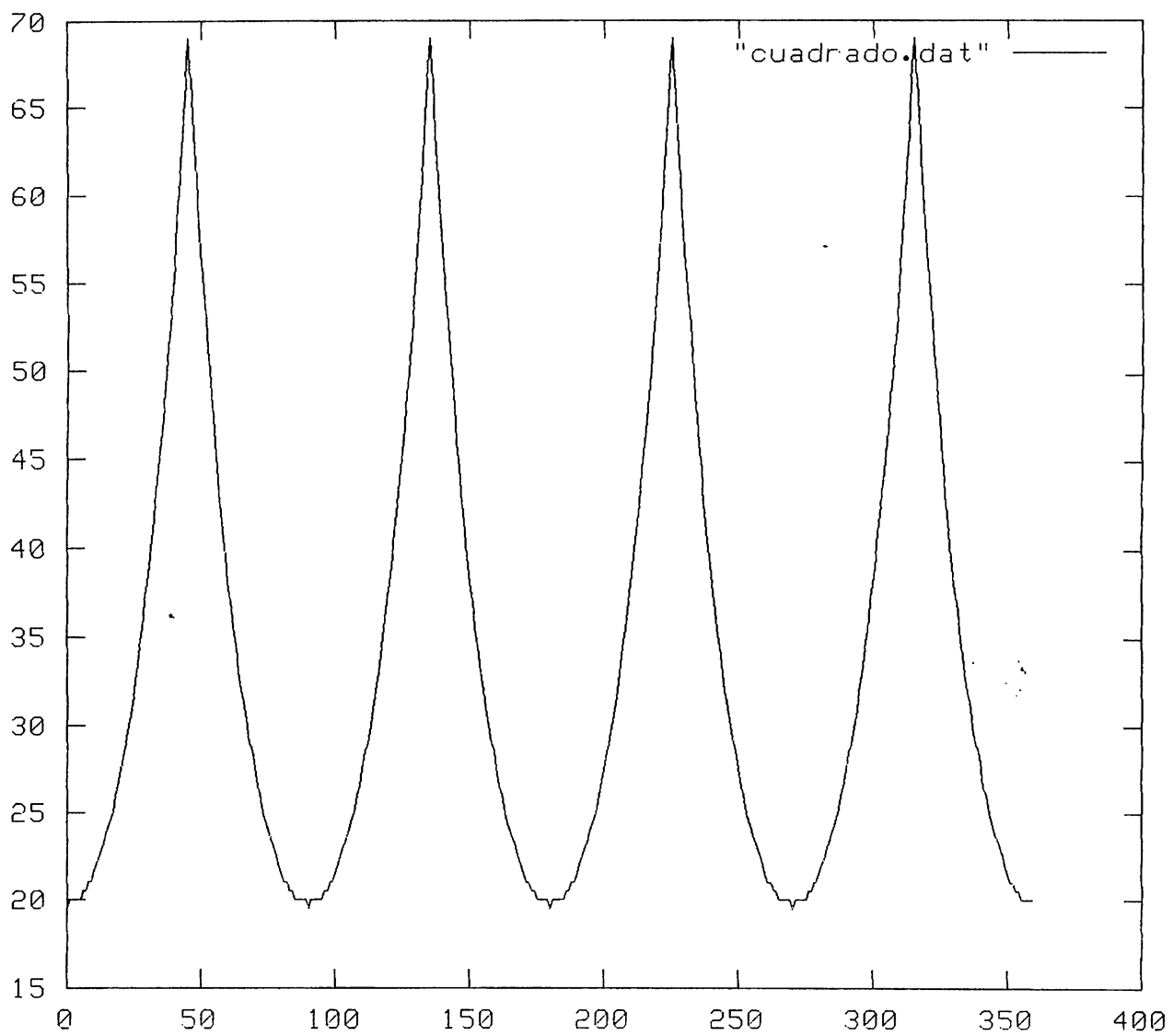
Reconocimiento de patrones: En general, análisis de tendencias o patrones presentes en datos numéricos por medio de computadoras. Esta disciplina abarca un gran número de campos, como pueden ser el reconocimiento de voz e imágenes, etc.

Bibliografía

- Cruz, W., Maccagno, N., Marín, L., *Diseño y Construcción de un Sistema de Captación y Procesamiento de Imágenes Digitales*, trabajo de graduación, Universidad Don Bosco, 1998
- Gamero, E., Alas, J., *Aplicación del Reconocimiento Óptico de Objetos en un Brazo Robot dentro de un Entorno Controlado*, trabajo de graduación, Universidad Don Bosco, 1999
- Gómez-Allende, D., *Reconocimiento de Formas y Visión Artificial*, 1ª. Ed., RA-MA Editorial, Madrid, 1993
- Pitas, I., *Digital Image Processing Algorithms*, 1ª. Ed., Prentice Hall International, Cambridge, 1993
- Varios, *Enciclopedia Microsoft Encarta 97*, CD-ROM, Microsoft Corporation, Redmond, 1997
- *Online Computer Dictionary for Internet Terms and Technical Support*, <http://www.webopedia.com>

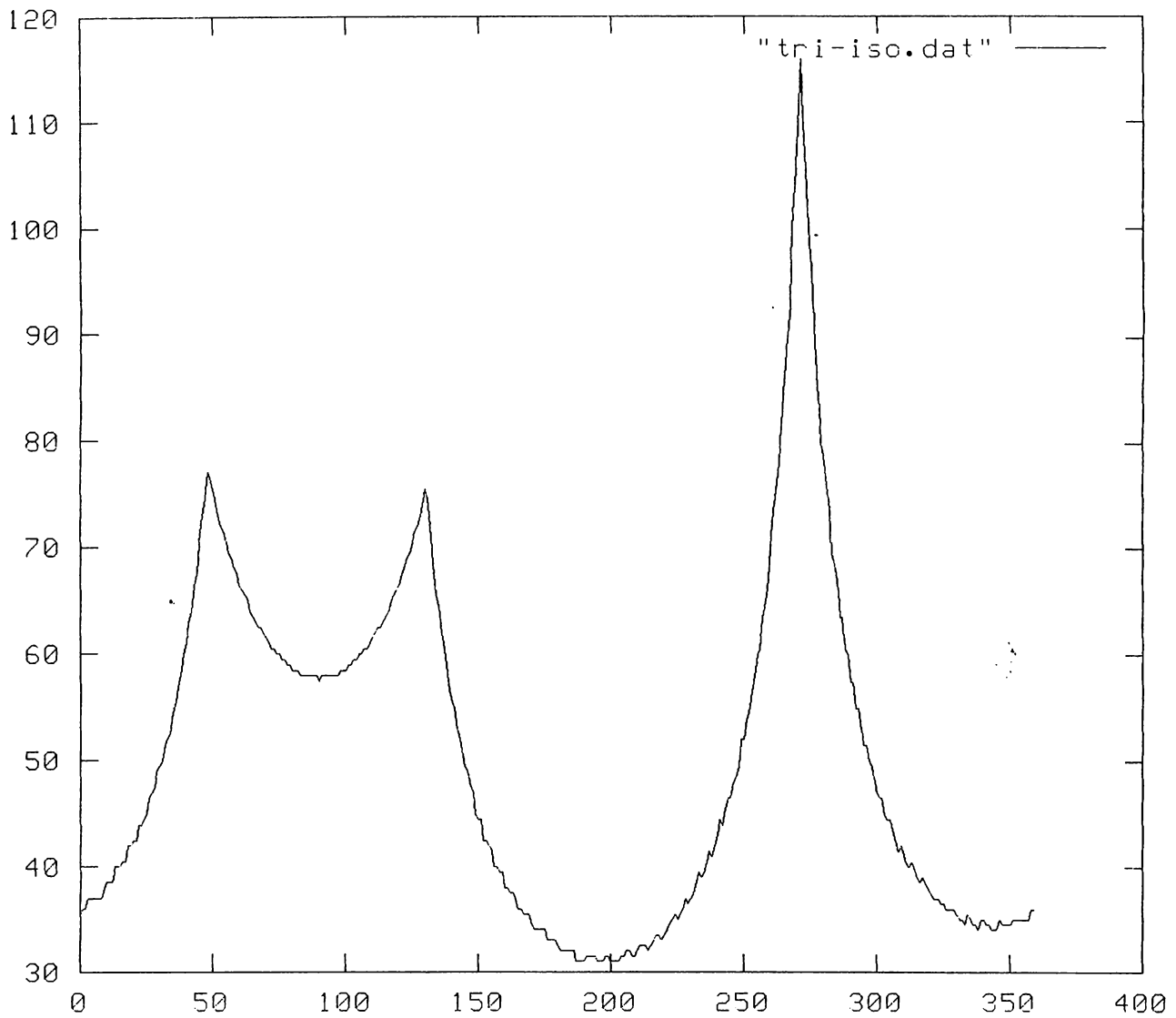
Anexos

- Gráfica 1: Curva de la función radar para un cuadrado, a intervalos de 1°
- Gráfica 2: Curva de la función radar para un triángulo isósceles, a intervalos de 1°
- Gráfica 3: Curva de la función radar para un triángulo isósceles, a intervalos de 5°
- Gráfica 4: Curva de la función radar para un polígono "delta", a intervalos de 10°
- Nota de copyright de la librería DAFS (traducción al español)
- Nota de copyright de la librería DAFS (original en inglés)
- Licencia pública general de GNU (traducción no oficial al español)
- Licencia pública general de GNU (original en inglés)
- Código fuente del programa

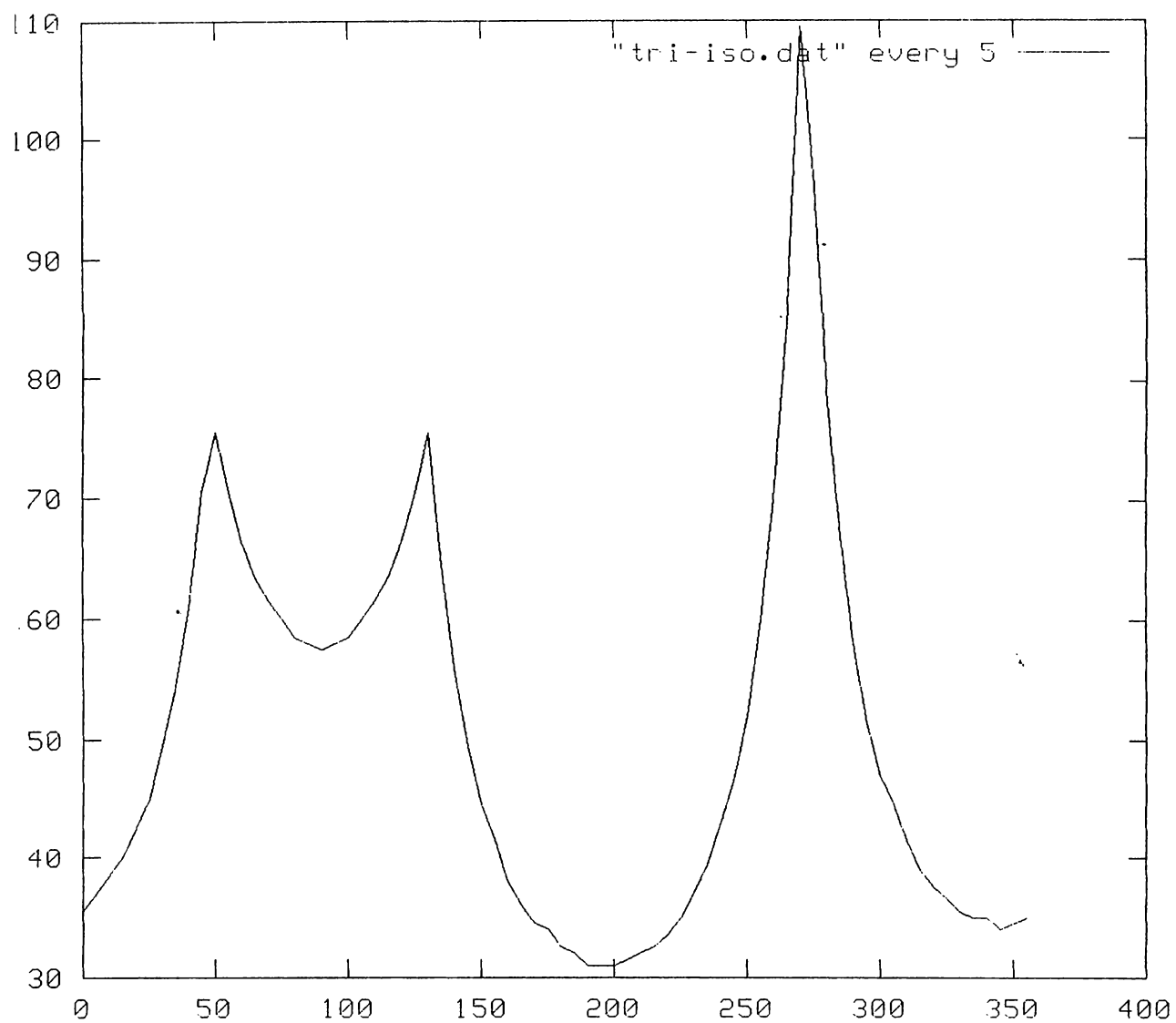


Gráfica 1: Curva de la función radar para un cuadrado, a intervalos de 1°

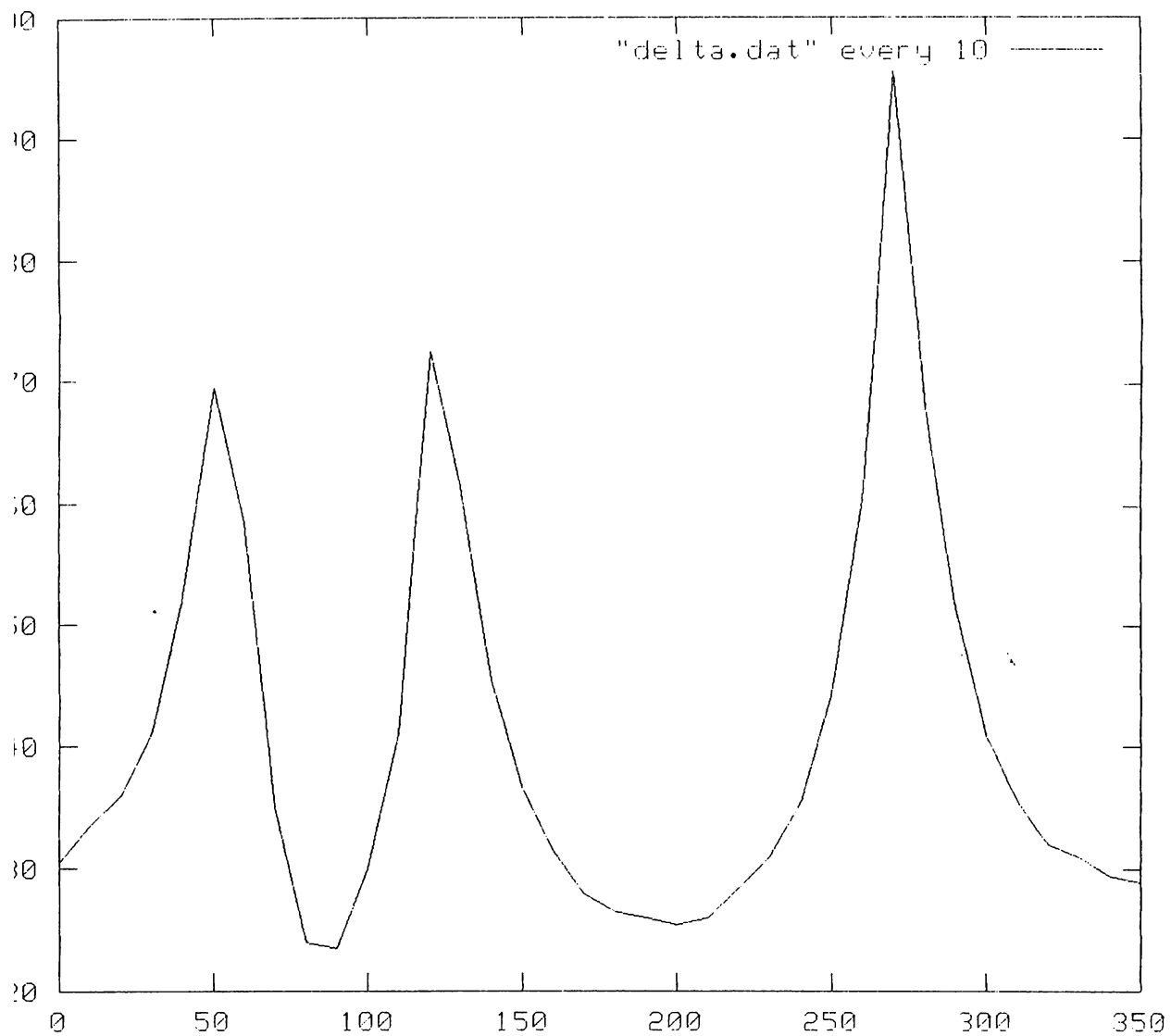
Gráfica 2. Curva de la función radar para un triángulo isósceles, a intervalos de 1



Gráfica 2: Curva de la función radar para un triángulo isósceles, a intervalos de 1°



Gráfica 3: Curva de la función radar para un triángulo isósceles, a intervalos de 5°



Gráfica 4: Curva de la función radar para un polígono "delta", a intervalos de 10°

Nota de copyright de la librería DAFS (traducción al español)

Copyright © 1993, 1994, 1995 RAF Technology, Derechos reservados

Desarrollado para ARPA bajo el contrato MDA 904-93-C-7219

Se concede permiso para utilizar, copiar, modificar, distribuir y vender este software y su documentación, sin cargo y para cualquier propósito, si se cumple (i) que la nota de copyright anterior y esta nota de permiso aparezcan en todas las copias del software y documentación relacionada, y (ii) que los nombres de RAF Technology o ARPA no sean utilizados en publicidad o anuncios relacionados con dicho software, sin el consentimiento previo y por escrito de RAF Technology y ARPA.

ESTE SOFTWARE SE PROVEE "COMO ES" Y SIN NINGÚN TIPO DE GARANTÍAS: EXPLÍCITAS, IMPLÍCITAS O DE CUALQUIER OTRA CLASE, INCLUYENDO SIN LIMITACIONES CUALQUIER GARANTÍA DE COMERCIALIZACIÓN O DE IDONEIDAD PARA UN PROPÓSITO PARTICULAR.

NI RAF TECHNOLOGY NI ARPA, BAJO NINGUNA CIRCUNSTANCIA, SE HACEN RESPONSABLES POR DAÑOS ESPECIALES, INCIDENTALES, INDIRECTOS O DIRECTOS DE NINGÚN TIPO, O CUALQUIER PERJUICIO RESULTANTE DE LA PÉRDIDA DE USUFRUCTO, DATOS O GANANCIAS, INDEPENDIENTEMENTE DEL CONOCIMIENTO DE LA POSIBILIDAD DE DAÑOS, RECHAZANDO CUALQUIER TEORÍA DE DEMANDA QUE SURJA DE O EN RELACIÓN A EL USO O DESEMPEÑO DE ESTE SOFTWARE.

Al momento de redactar este documento, la librería DAFS completa, la documentación y el software correspondiente se encuentran en el siguiente sitio web:

<http://documents.cfar.umd.edu/resources/source/illuminator.html>

Nota de copyright de la librería DAFS (original en inglés)

Copyright (c) 1993, 1994, 1995 RAF Technology

developed for ARPA under contract MDA 904-93-C-7219

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of RAF Technology or ARPA may not be used in any advertising or publicity relating to the software without the specific, prior written permission of RAF Technology and ARPA.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL RAF TECHNOLOGY OR ARPA BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Licencia Pública General de GNU (traducción no oficial al español)

Esta es la conocida GNU Public License (GPL), versión 2 (de junio de 1.991), que cubre la mayor parte del software de la Free Software Foundation, y muchos más programas.

NOTA IMPORTANTE:

Esta es una traducción no oficial al español de la GNU General Public License. No ha sido publicada por la Free Software Foundation, y no establece legalmente las condiciones de distribución para el software que usa la GNU GPL. Estas condiciones se establecen solamente por el texto original, en inglés, de la GNU GPL. Sin embargo, esperamos que esta traducción ayude a los hispanohablantes a entender mejor la GNU GPL.

IMPORTANT NOTICE:

This is an unofficial translation of the GNU General Public License into Spanish. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL--only the original English text of the GNU GPL does that. However, we hope that this translation will help Spanish speakers understand the GNU GPL better.

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

675 Mass Ave, Cambridge, MA 02139, EEUU

Se permite la copia y distribución de copias literales de este documento, pero no se permite su modificación.

Preámbulo

Las licencias que cubren la mayor parte del software están diseñadas para quitarle a usted la libertad de compartirlo y modificarlo. Por el contrario, la Licencia Pública General de GNU pretende garantizarle la libertad de compartir y modificar software libre, para asegurar que el software es libre para todos sus usuarios. Esta Licencia Pública General se aplica a la mayor parte del software de la Free Software Foundation y a cualquier otro programa si sus autores se comprometen a utilizarla. (Existe otro software de la Free Software Foundation que está cubierto por la Licencia Pública General de GNU para Bibliotecas). Si quiere, también puede aplicarla a sus propios

programas.

Cuando hablamos de software libre, estamos refiriéndonos a libertad, no a precio. Nuestras Licencias Públicas Generales están diseñadas para asegurarnos de que tenga la libertad de distribuir copias de software libre (y cobrar por ese servicio si quiere), de que reciba el código fuente o que pueda conseguirlo si lo quiere, de que pueda modificar el software o usar fragmentos de él en nuevos programas libres, y de que sepa que puede hacer todas estas cosas.

Para proteger sus derechos necesitamos algunas restricciones que prohíban a cualquiera negarle a usted estos derechos o pedirle que renuncie a ellos. Estas restricciones se traducen en ciertas obligaciones que le afectan si distribuye copias del software, o si lo modifica.

Por ejemplo, si distribuye copias de uno de estos programas, sea gratuitamente, o a cambio de una contraprestación, debe dar a los receptores todos los derechos que tiene. Debe asegurarse de que ellos también reciben, o pueden conseguir, el código fuente. Y debe mostrarles estas condiciones de forma que conozcan sus derechos.

Protegemos sus derechos con la combinación de dos medidas:

1. Ponemos el software bajo copyright y
2. le ofrecemos esta licencia, que le da permiso legal para copiar, distribuir y/o modificar el software.

También, para la protección de cada autor y la nuestra propia, queremos asegurarnos de que todo el mundo comprende que no se proporciona ninguna garantía para este software libre. Si el software se modifica por cualquiera y éste a su vez lo distribuye, queremos que sus receptores sepan que lo que tienen no es el original, de forma que cualquier problema introducido por otros no afecte a la reputación de los autores originales.

Por último, cualquier programa libre está constantemente amenazado por patentes sobre el software. Queremos evitar el peligro de que los redistribuidores de un programa libre obtengan patentes por su cuenta, convirtiendo de facto el programa en propietario. Para evitar esto, hemos dejado claro que cualquier patente debe ser pedida para el uso libre de cualquiera, o no ser pedida.

Los términos exactos y las condiciones para la copia, distribución y modificación se exponen a continuación.

Términos y condiciones para la copia, distribución y modificación

1. Esta Licencia se aplica a cualquier programa u otro tipo de trabajo que contenga una nota colocada por el tenedor del copyright diciendo que puede ser distribuido bajo los términos de

esta Licencia Pública General. En adelante, «Programa» se referirá a cualquier programa o trabajo que cumpla esa condición y «trabajo basado en el Programa» se referirá bien al Programa o a cualquier trabajo derivado de él según la ley de copyright. Esto es, un trabajo que contenga el programa o una porción de él, bien en forma literal o con modificaciones y/o traducido en otro lenguaje. Por lo tanto, la traducción está incluida sin limitaciones en el término «modificación». Cada concesionario (licenciatario) será denominado «usted».

Cualquier otra actividad que no sea la copia, distribución o modificación no está cubierta por esta Licencia, está fuera de su ámbito. El acto de ejecutar el Programa no está restringido, y los resultados del Programa están cubiertos únicamente si sus contenidos constituyen un trabajo basado en el Programa, independientemente de haberlo producido mediante la ejecución del programa. El que esto se cumpla, depende de lo que haga el programa.

2. Usted puede copiar y distribuir copias literales del código fuente del Programa, según lo has recibido, en cualquier medio, supuesto que de forma adecuada y bien visible publique en cada copia un anuncio de copyright adecuado y un repudio de garantía, mantenga intactos todos los anuncios que se refieran a esta Licencia y a la ausencia de garantía, y proporcione a cualquier otro receptor del programa una copia de esta Licencia junto con el Programa. Puede cobrar un precio por el acto físico de transferir una copia, y puede, según su libre albedrío, ofrecer garantía a cambio de unos honorarios.
3. Puede modificar su copia o copias del Programa o de cualquier porción de él, formando de esta manera un trabajo basado en el Programa, y copiar y distribuir esa modificación o trabajo bajo los términos del apartado 1, antedicho, supuesto que además cumpla las siguientes condiciones:
 - a. Debe hacer que los ficheros modificados lleven anuncios prominentes indicando que los ha cambiado y la fecha de cualquier cambio.
 - b. Debe hacer que cualquier trabajo que distribuya o publique y que en todo o en parte contenga o sea derivado del Programa o de cualquier parte de él sea licenciada como un todo, sin carga alguna, a todas las terceras partes y bajo los términos de esta Licencia.
 - c. Si el programa modificado lee normalmente órdenes interactivamente cuando es ejecutado, debe hacer que, cuando comience su ejecución para ese uso interactivo de la forma más habitual, muestre o escriba un mensaje que incluya un anuncio de copyright y un anuncio de que no se ofrece ninguna garantía (o por el contrario que sí se ofrece garantía) y que los usuarios pueden redistribuir el programa bajo estas condiciones, e indicando al usuario cómo ver una copia de esta licencia. (Excepción: si el propio programa es interactivo pero normalmente no muestra ese anuncio, no se

requiere que su trabajo basado en el Programa muestre ningún anuncio). Estos requisitos se aplican al trabajo modificado como un todo. Si partes identificables de ese trabajo no son derivadas del Programa, y pueden, razonablemente, ser consideradas trabajos independientes y separados por ellos mismos, entonces esta Licencia y sus términos no se aplican a esas partes cuando sean distribuidas como trabajos separados. Pero cuando distribuya esas mismas secciones como partes de un todo que es un trabajo basado en el Programa, la distribución del todo debe ser según los términos de esta licencia, cuyos permisos para otros licenciarios se extienden al todo completo, y por lo tanto a todas y cada una de sus partes, con independencia de quién la escribió.

Por lo tanto, no es la intención de este apartado reclamar derechos o desafiar sus derechos sobre trabajos escritos totalmente por usted mismo. El intento es ejercer el derecho a controlar la distribución de trabajos derivados o colectivos basados en el Programa.

Además, el simple hecho de reunir un trabajo no basado en el Programa con el Programa (o con un trabajo basado en el Programa) en un volumen de almacenamiento o en un medio de distribución no hace que dicho trabajo entre dentro del ámbito cubierto por esta Licencia.

4. Puede copiar y distribuir el Programa (o un trabajo basado en él, según se especifica en el apartado 2, como código objeto o en formato ejecutable según los términos de los apartados 1 y 2, supuesto que además cumpla una de las siguientes condiciones:
 - a. Acompañarlo con el código fuente completo correspondiente, en formato electrónico, que debe ser distribuido según se especifica en los apartados 1 y 2 de esta Licencia en un medio habitualmente utilizado para el intercambio de programas, o
 - b. Acompañarlo con una oferta por escrito, válida durante al menos tres años, de proporcionar a cualquier tercera parte una copia completa en formato electrónico del código fuente correspondiente, a un coste no mayor que el de realizar físicamente la distribución del fuente, que será distribuido bajo las condiciones descritas en los apartados 1 y 2 anteriores, en un medio habitualmente utilizado para el intercambio de programas, o
 - c. Acompañarlo con la información que recibiste ofreciendo distribuir el código fuente correspondiente. (Esta opción se permite sólo para distribución no comercial y sólo si usted recibió el programa como código objeto o en formato ejecutable con tal oferta, de acuerdo con el apartado b anterior).

Por código fuente de un trabajo se entiende la forma preferida del trabajo cuando se le hacen modificaciones. Para un trabajo ejecutable, se entiende por código fuente completo todo el código fuente para todos los módulos que contiene, más cualquier fichero

asociado de definición de interfaces, más los guiones utilizados para controlar la compilación e instalación del ejecutable. Como excepción especial el código fuente distribuido no necesita incluir nada que sea distribuido normalmente (bien como fuente, bien en forma binaria) con los componentes principales (compilador, kernel y similares) del sistema operativo en el cual funciona el ejecutable, a no ser que el propio componente acompañe al ejecutable.

Si la distribución del ejecutable o del código objeto se hace mediante la oferta acceso para copiarlo de un cierto lugar, entonces se considera la oferta de acceso para copiar el código fuente del mismo lugar como distribución del código fuente, incluso aunque terceras partes no estén forzadas a copiar el fuente junto con el código objeto.

5. No puede copiar, modificar, sublicenciar o distribuir el Programa excepto como prevé expresamente esta Licencia. Cualquier intento de copiar, modificar sublicenciar o distribuir el Programa de otra forma es inválida, y hará que cesen automáticamente los derechos que te proporciona esta Licencia. En cualquier caso, las partes que hayan recibido copias o derechos de usted bajo esta Licencia no cesarán en sus derechos mientras esas partes continúen cumpliéndola.
6. No está obligado a aceptar esta licencia, ya que no la ha firmado. Sin embargo, no hay nada más que le proporcione permiso para modificar o distribuir el Programa o sus trabajos derivados. Estas acciones están prohibidas por la ley si no acepta esta Licencia. Por lo tanto, si modifica o distribuye el Programa (o cualquier trabajo basado en el Programa), está indicando que acepta esta Licencia para poder hacerlo, y todos sus términos y condiciones para copiar, distribuir o modificar el Programa o trabajos basados en él.
7. Cada vez que redistribuya el Programa (o cualquier trabajo basado en el Programa), el receptor recibe automáticamente una licencia del licenciataria original para copiar, distribuir o modificar el Programa, de forma sujeta a estos términos y condiciones. No puede imponer al receptor ninguna restricción más sobre el ejercicio de los derechos aquí garantizados. No es usted responsable de hacer cumplir esta licencia por terceras partes.
8. Si como consecuencia de una resolución judicial o de una alegación de infracción de patente o por cualquier otra razón (no limitada a asuntos relacionados con patentes) se le imponen condiciones (ya sea por mandato judicial, por acuerdo o por cualquier otra causa) que contradigan las condiciones de esta Licencia, ello no le exime de cumplir las condiciones de esta Licencia. Si no puede realizar distribuciones de forma que se satisfagan simultáneamente sus obligaciones bajo esta licencia y cualquier otra obligación pertinente entonces, como consecuencia, no puede distribuir el Programa de ninguna forma. Por ejemplo, si una patente no permite la redistribución libre de derechos de autor del Programa por parte de todos aquellos que reciban copias directa o

indirectamente a través de usted, entonces la única forma en que podría satisfacer tanto esa condición como esta Licencia sería evitar completamente la distribución del Programa.

Si cualquier porción de este apartado se considera inválida o imposible de cumplir bajo cualquier circunstancia particular ha de cumplirse el resto y la sección por entero ha de cumplirse en cualquier otra circunstancia.

No es el propósito de este apartado inducirle a infringir ninguna reivindicación de patente ni de ningún otro derecho de propiedad o impugnar la validez de ninguna de dichas reivindicaciones. Este apartado tiene el único propósito de proteger la integridad del sistema de distribución de software libre, que se realiza mediante prácticas de licencia pública. Mucha gente ha hecho contribuciones generosas a la gran variedad de software distribuido mediante ese sistema con la confianza de que el sistema se aplicará consistentemente. Será el autor/donante quien decida si quiere distribuir software mediante cualquier otro sistema y una licencia no puede imponer esa elección.

Este apartado pretende dejar completamente claro lo que se cree que es una consecuencia del resto de esta Licencia.

9. Si la distribución y/o uso de el Programa está restringida en ciertos países, bien por patentes o por interfaces bajo copyright, el tenedor del copyright que coloca este Programa bajo esta Licencia puede añadir una limitación explícita de distribución geográfica excluyendo esos países, de forma que la distribución se permita sólo en o entre los países no excluidos de esta manera. En ese caso, esta Licencia incorporará la limitación como si estuviese escrita en el cuerpo de esta Licencia.
10. La Free Software Foundation puede publicar versiones revisadas y/o nuevas de la Licencia Pública General de tiempo en tiempo. Dichas nuevas versiones serán similares en espíritu a la presente versión, pero pueden ser diferentes en detalles para considerar nuevos problemas o situaciones.
Cada versión recibe un número de versión que la distingue de otras. Si el Programa especifica un número de versión de esta Licencia que se refiere a ella y a «cualquier versión posterior», tienes la opción de seguir los términos y condiciones, bien de esa versión, bien de cualquier versión posterior publicada por la Free Software Foundation. Si el Programa no especifica un número de versión de esta Licencia, puedes escoger cualquier versión publicada por la Free Software Foundation.
11. Si quiere incorporar partes del Programa en otros programas libres cuyas condiciones de distribución son diferentes, escribe al autor para pedirle permiso. Si el software tiene copyright de la Free Software Foundation, escribe a la Free Software Foundation: algunas veces hacemos excepciones en estos casos. Nuestra decisión estará guiada por el doble objetivo de preservar la libertad de todos los derivados de nuestro software libre y promover el que se comparta y reutilice el software en general.

AUSENCIA DE GARANTÍA

12. Como el programa se licencia libre de cargas, no se ofrece ninguna garantía sobre el programa, en todas la extensión permitida por la legislación aplicable. Excepto cuando se indique de otra forma por escrito, los tenedores del copyright y/u otras partes proporcionan el programa «tal cual», sin garantía de ninguna clase, bien expresa o implícita, con inclusión, pero sin limitación a las garantías mercantiles implícitas o a la conveniencia para un propósito particular. Cualquier riesgo referente a la calidad y prestaciones del programa es asumido por usted. Si se probase que el Programa es defectuoso, asume el coste de cualquier servicio, reparación o corrección.
13. En ningún caso, salvo que lo requiera la legislación aplicable o haya sido acordado por escrito, ningún tenedor del copyright ni ninguna otra parte que modifique y/o redistribuya el Programa según se permite en esta Licencia será responsable ante usted por daños, incluyendo cualquier daño general, especial, incidental o resultante producido por el uso o la imposibilidad de uso del Programa (con inclusión, pero sin limitación a la pérdida de datos o a la generación incorrecta de datos o a pérdidas sufridas por usted o por terceras partes o a un fallo del Programa al funcionar en combinación con cualquier otro programa), incluso si dicho tenedor u otra parte ha sido advertido de la posibilidad de dichos daños.

FIN DE TÉRMINOS Y CONDICIONES

Apéndice: Cómo aplicar estos términos a sus nuevos programas.

Si usted desarrolla un nuevo Programa, y quiere que sea del mayor uso posible para el público en general, la mejor forma de conseguirlo es convirtiéndolo en software libre que cualquiera pueda redistribuir y cambiar bajo estos términos.

Para hacerlo, añada los siguientes anuncios al programa. Lo más seguro es añadirlos al principio de cada fichero fuente para transmitir lo más efectivamente posible la ausencia de garantía. Además cada fichero debería tener al menos la línea de «copyright» y un indicador a dónde puede encontrarse el anuncio completo.

<una línea para indicar el nombre del programa y una rápida idea de qué hace.>

Copyright (C) 19aa <nombre del autor>

Este programa es software libre. Puede redistribuirlo y/o modificarlo bajo los términos de la Licencia Pública General de GNU según es publicada por la Free Software Foundation, bien de la versión 2 de dicha Licencia o bien (según su elección) de cualquier versión posterior.

Este programa se distribuye con la esperanza de que sea útil, pero SIN NINGUNA GARANTÍA, incluso sin la garantía MERCANTIL implícita o sin garantizar la CONVENIENCIA PARA UN PROPÓSITO PARTICULAR. Véase la Licencia Pública General de GNU para más detalles.

Debería haber recibido una copia de la Licencia Pública General junto con este programa. Si no ha sido así, escriba a la Free Software Foundation, Inc., en 675 Mass Ave, Cambridge, MA 02139, EEUU. Añada también información sobre cómo contactar con usted mediante correo electrónico y postal.

Licencia pública general de GNU (original en inglés)

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of

this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

/*
Programa de Reconocimiento y Clasificación de imágenes

Archivo: auxiliar.h

Copyright (C) 2001 Carlos Montalvo

Este programa es software libre. Puede redistribuirlo y/o modificarlo bajo los términos de la Licencia Pública General de GNU según es publicada por la Free Software Foundation, bien de la versión 2 de dicha Licencia o bien (según su elección) de cualquier versión posterior.

Este programa se distribuye con la esperanza de que sea útil, pero SIN NINGUNA GARANTÍA, incluso sin la garantía MERCANTIL implícita o sin garantizar la CONVENIENCIA PARA UN PROPÓSITO PARTICULAR. Véase la Licencia Pública General de GNU para más detalles.

Debería haber recibido una copia de la Licencia Pública General junto con este programa. Si no ha sido así, escriba a la Free Software Foundation, Inc., en 675 Mass Ave, Cambridge, MA 02139, EEUU. Añada también información sobre cómo contactar con usted mediante correo electrónico y postal.

*/

/* Estructuras y variables globales */

```
typedef struct{  
    int angulo;  
    float radio;  
} PUNTO;
```

/* Prototipos de las funciones auxiliares */

```
char **CrearMatriz(int x, int y);  
void QuitarMatriz(char **matriz, int indice);  
int Redondear(double valor);  
int OrdenarRadio(const void *a, const void *b);  
int OrdenarAngulo(const void *a, const void *b);  
void ErrorFatal(iDAFSError error);
```

/* Crear la matriz para almacenar temporalmente la imagen */

```
char **CrearMatriz(int x, int y){  
    char **mat;  
    int i;  
  
    mat = (char **) calloc(x, sizeof(char *));  
    if(mat == NULL)  
        return(NULL);  
    for(i = 0; i < x; i++){  
        mat[i] = (char *) calloc(y, sizeof(char));  
        if(mat[i] == NULL){  
            QuitarMatriz(mat, i);  
            return(NULL);  
        }  
    }  
    return(mat);  
}
```

}

```

/* Si se produce un error, liberar la memoria asignada a la matriz */
void QuitarMatriz(char **matriz, int indice){
    int i;

    for(i = 0; i < indice; i++)
        free(matriz[i]);
    free(matriz);
}

/* Redondear valores decimales a enteros */
int Redondear(double valor){
    int i;
    double e;

    if(modf(valor, &e) >= 0.5)
        i = (int) ceil(valor);
    else if(modf(valor, &e) <= -0.5)
        i = (int) floor(valor);
    else
        i = (int) valor;
    return(i);
}

/* 2 funciones comparadoras, auxiliares de qsort() */
int OrdenarRadio(const void *a, const void *b){
    PUNTO *da = (PUNTO *) a;
    PUNTO *db = (PUNTO *) b;

    return (da->radio < db->radio) - (da->radio > db->radio);
}

int OrdenarAngulo(const void *a, const void *b){
    PUNTO *da = (PUNTO *) a;
    PUNTO *db = (PUNTO *) b;

    return (da->angulo > db->angulo) - (da->angulo < db->angulo);
}

/* Terminar inmediatamente el programa en caso de error grave */
void ErrorFatal(iDAFSError error){
    i_ExitLib();
    fprintf(stderr, "ERROR: %s\n", i_GetError(error));
    exit(1);
}

```

```

/*****
Programa de Reconocimiento y Clasificación de imágenes

Archivo: reconocedor.h
*****/

Copyright (C) 2001 Carlos Montalvo

Este programa es software libre. Puede redistribuirlo y/o modificarlo
bajo los términos de la Licencia Pública General de GNU según es
publicada por la Free Software Foundation, bien de la versión 2 de
dicha Licencia o bien (según su elección) de cualquier versión
posterior.

Este programa se distribuye con la esperanza de que sea útil, pero SIN
NINGUNA GARANTÍA, incluso sin la garantía MERCANTIL implícita o sin
garantizar la CONVENIENCIA PARA UN PROPÓSITO PARTICULAR. Véase la
Licencia Pública General de GNU para más detalles.

Debería haber recibido una copia de la Licencia Pública General junto
con este programa. Si no ha sido así, escriba a la Free Software
Foundation, Inc., en 675 Mass Ave, Cambridge, MA 02139, EEUU.
Añada también información sobre cómo contactar con usted mediante
correo electrónico y postal.
*/

/* Librerías estándar de C */
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <math.h>

/* Librerías auxiliares */
#include "dafs.h"
#include "auxiliar.h"

/* Valores constantes */
#define RADIANTES M_PI / 180
#define HAY_ERROR if(e != DAFSOK) return(e);
#define HAY_ERROR_FATAL if(e != DAFSOK) ErrorFatal(e);

/* Parámetros modificables generales */
#define ARCHIVO_DATOS "/home/carlos/tesis/datos/poligonos.bd"
#define LISTAR_FIGURAS "ls /home/carlos/tesis/figuras"
#define GRAFICADOR "gnuplot -persist"
#define COMANDO_GRAF "plot '-' with lines\n"

/* Parámetros modificables del algoritmo de reconocimiento */
#define MAX_VERTICES 60
#define MIN_VERTICES 3
#define PASOS 180
#define INTERVALO_R 0.5f
#define MARGEN_ERROR_R 2.0f
#define MARGEN_ERROR_TH 5

/* Estructuras y variables globales */
iEntityPtr basedatos;

/* Prototipos de las funciones */
iDAFSError ProcesarImagen(char archivo[50]);
void CodigoCadena(char **matriz, int x0, int y0, int *cx, int *cy);
void Radar(char **matriz, int cx, int cy, float f[360]);
int Vertices(float f[360], int intervalo, PUNTO *vertices);
void Deltas(int nv, PUNTO *vr, int *vi, int *li);
iEntityPtr CompararImagen(int num_vertices, int vertices_ig, int lados_ig);
iDAFSError AlmacenarProto(iImagePtr figura, int num_vertices, int vertices_ig, int
lados_ig);
iDAFSError Almacenar(iEntityPtr padre, iImagePtr figura, int vertices_ig, int lados_ig);

```

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

Programa de Reconocimiento y Clasificación de imágenes

Archivo: reconocedor.c

```
*****
```

Copyright (C) 2001 Carlos Montalvo

Este programa es software libre. Puede redistribuirlo y/o modificarlo bajo los términos de la Licencia Pública General de GNU según es publicada por la Free Software Foundation, bien de la versión 2 de dicha Licencia o bien (según su elección) de cualquier versión posterior.

Este programa se distribuye con la esperanza de que sea útil, pero SIN NINGUNA GARANTÍA, incluso sin la garantía MERCANTIL implícita o sin garantizar la CONVENIENCIA PARA UN PROPÓSITO PARTICULAR. Véase la Licencia Pública General de GNU para más detalles.

Debería haber recibido una copia de la Licencia Pública General junto con este programa. Si no ha sido así, escriba a la Free Software Foundation, Inc., en 675 Mass Ave, Cambridge, MA 02139, EEUU. Añada también información sobre cómo contactar con usted mediante correo electrónico y postal.

```
*/
```

```
/* Librería principal */  
#include "reconocedor.h"
```

```
/* Procesar y efectuar el reconocimiento de la imagen */
```

```
iDAFSError ProcesarImagen(char archivo[50]){
```

```
    iImagePtr imagen;  
    iBox area;  
    int2 bytes_fila;  
    char *bitmap = NULL;  
    char **matriz = NULL;  
    int i, j, k, l;  
    bool origen = false;  
    int xorigen, yorigen;  
    int xcentro, ycentro;  
    float func[360];  
    PUNTO vert[MAX_VERTICES], *vert_reales;  
    int svert = 0, nvert;  
    char respuesta[2];  
    int nvert_ig, lados_ig;  
    iEntityPtr figura;  
    iDAFSError e;
```

```
    /* Leer archivo de imagen y parámetros de área */
```

```
    e = i_ReadImage(archivo, &imagen);  
    HAY_ERROR  
    area = i_GetImageBounds(imagen);
```

```
    /* Transformar imagen a bitmap */
```

```
    e = i_Image2BitMap(imagen, area, iFP_ONE, &bytes_fila, &bitmap);  
    HAY_ERROR
```

```
    /* Copiar info. del bitmap a una matriz temporal para simplificar cálculos */
```

```
    matriz = CrearMatriz(bytes_fila * 8, area.h);  
    if(matriz == NULL){  
        perror("Falló la creación de la matriz\n");  
        i_ExitLib();  
        exit(1);  
    }  
    for(i = 0; i < area.h; i++){  
        for(j = 0; j < bytes_fila; j++){  
            for(k = 0; k < 8; k++){
```



```

        l = (8 * j) + k;
        matriz[l][i] = ((l << (7 - k)) & bitmap[i * bytes_fila + j]) != 0;
        if(matriz[l][i] == 1 && origen == false){
            xorigen = l;
            yorigen = i;
            origen = true;
        }
    }
}

/* Extraer el centro geométrico de la imagen */
CodigoCadena(matriz, xorigen, yorigen, &xcentro, &ycentro);

/* Extraer la cantidad de vértices y sus distancias */
Radar(matriz, xcentro, ycentro, func);
for(i = 1; i <= PASOS; i++){
    nvert = Vertices(func, i, vert);
    if(nvert < MIN_VERTICES){
        i--;
        break;
    }
    svert += nvert;
}
nvert = Redondear((float) svert / (float) i);
if(nvert < 3)
    nvert = 3;

/* Ordenar vértices según sus máximos */
qsort(vert, MAX_VERTICES, sizeof(PUNTO), OrdenarRadio);

/* Confirmar número de vértices */
printf("La figura tiene %d vértices.\n¿Correcto (s/n)? ", nvert);
scanf("%s", respuesta);
if(strcmp(respuesta, "s") != 0){
    do{
        printf("Escriba el número correcto de vértices: ");
        scanf("%d", &nvert);
    } while(nvert < 3 || nvert > 10);
}

/* Extraer los vértices reales y ordenarlos según el ángulo */
vert_reales = (PUNTO *) malloc(nvert * sizeof(PUNTO));
memcpy(vert_reales, vert, nvert * sizeof(PUNTO));
qsort(vert_reales, nvert, sizeof(PUNTO), OrdenarAngulo);
Deltas(nvert, vert_reales, &nvert_ig, &lados_ig);

/* Guardar imagen y resultados, si no existe ya una similar */
figura = CompararImagen(nvert, nvert_ig, lados_ig);
if(figura == NULL){
    e = AlmacenarProto(imagen, nvert, nvert_ig, lados_ig);
    HAY_ERROR
}
else{
    if(strcmp(i_GetText(i_GetParentEntity(figura), NULL), "Poligonos") == 0){
        printf("Esta figura es un nuevo tipo de %s.\n", i_GetText(figura, NULL));
        e = Almacenar(figura, imagen, nvert_ig, lados_ig);
        HAY_ERROR
    }
    else
        printf("Esta figura es un %s (%s).\n",
            i_GetText(i_GetParentEntity(figura), NULL), i_GetText(figura, NULL));
}

/* Eliminar referencias */
free(vert_reales);
QuitarMatriz(matriz, area.w);
i_DisposeImage(imagen);

```

```

        return(DAFSOK);
    }

/* Calcular el código cadena, los momentos de orden 0 y 1 y el centro de la imagen */
void CodigoCadena(char **matriz, int x0, int y0, int *cx, int *cy){
    unsigned int direccion = 3;
    int x, y, dx, dy;
    double m00 = 0.0, m10 = 0.0, m01 = 0.0;

    /* Seguimiento de bordes y extracción del código cadena */
    x = x0;
    y = y0;
    do{
        if(matriz[x][y] == 1)
            direccion = (direccion < 3) ? direccion + 1 : 0;
        else
            direccion = (direccion > 0) ? direccion - 1 : 3;
        switch(direccion){
            case 3:
                y++;
                dx = 0;
                dy = 1;
                break;
            case 2:
                x--;
                dx = -1;
                dy = 0;
                break;
            case 1:
                y--;
                dx = 0;
                dy = -1;
                break;
            case 0:
                x++;
                dx = 1;
                dy = 0;
                break;
        }
        m00 += (x * dy) - (y * dx);
        m10 += ((x * dy) - (y * dx)) * (x - (dx / 2));
        m01 += ((x * dy) - (y * dx)) * (y - (dy / 2));
    } while(x != x0 || y != y0);

    /* Cálculo del centro geométrico a partir de los momentos de orden 0 y 1 */
    m00 /= 2;
    m10 /= 3;
    m01 /= 3;
    *cx = Redondear(m10 / m00);
    *cy = Redondear(m01 / m00);
}

/* Cálculo de la función "radar": r = f(theta), donde 0 <= theta < 360 */
void Radar(char **matriz, int cx, int cy, float f[360]){
    char respuesta[2];
    FILE *salida;
    int i;
    int x, y;
    double theta;
    float r;

    /* Graficar opcionalmente la función "radar" por medio de un programa externo */
    printf("\n¿Desea graficar la salida de la función radar (s/n)? ");
    scanf("%s", respuesta);
    if(strcmp(respuesta, "s") == 0){
        salida = popen(GRAFIADOR, "w");
        if(ferror(salida))
            perror("Error no definido, no se pudo graficar la función\n");
        else

```

```

        fprintf(salida, COMANDO_GRAF);
    }

    /* Valores de la función */
    for(i = 0; i < 360; i++){
        theta = i * RADIANTES;
        r = 0.0;
        do{
            r += INTERVALO_R;
            x = Redondear(cos(theta) * r);
            y = Redondear(sin(theta) * r);
        } while(matriz[cx + x][cy + y] == 1);
        f[i] = r;
        if(strcmp(respuesta, "s") == 0)
            fprintf(salida, "%d %f\n", i, f[i]);
    }
    if(strcmp(respuesta, "s") == 0)
        fclose(salida);
}

/* Cálculo de la cantidad e intervalos en theta de los vértices */
int Vertices(float f[360], int intervalo, PUNTO *vertices){
    int i, j, k, l = 0;
    const int max_th = (int) ceilf((float) 360 / (float) intervalo) - 1;

    for(j = 0; j < 360; j += intervalo){
        if(j == 0){
            i = max_th * intervalo;
            k = j + intervalo;
        }
        else if(j == max_th * intervalo){
            i = j - intervalo;
            k = 0;
        }
        else{
            i = j - intervalo;
            k = j + intervalo;
        }
        if((f[j] > f[i]) && (f[j] > f[k]) && (l < MAX_VERTICES)){
            if(intervalo == 1){
                vertices[l].angulo = j;
                vertices[l].radio = f[j];
            }
            l++;
        }
    }
    return(l);
}

/* Estimación del número de vértices y lados iguales */
void Deltas(int nv, PUNTO *vr, int *vi, int *li){
    int i, j;
    int v = 0, l = 0;

    for(i = 1; i < nv; i++){
        /* Comparación y conteo de vértices iguales */
        if(fabsf(vr[0].radio - vr[i].radio) <= MARGEN_ERROR_R){
            if(v == 0)
                v = 2;
            else
                v++;
        }

        /* Comparación y conteo de lados iguales */
        if(abs((vr[i].angulo - vr[i - 1].angulo) - (vr[1].angulo - vr[0].angulo))
            <= MARGEN_ERROR_TH)
            l++;
    }
}

```

```

    }
    if(abs((vr[1].angulo - vr[0].angulo) - ((360 - vr[i - 1].angulo) + vr[0].angulo))
        <= MARGEN_ERROR_TH)
        l++;

    /* Corrección de valores según el número de vértices */
    switch(nv){
        case 3:
            if(l == 2 || v == 2){
                l = 2;
                v = 2;
            }
            else if(l < 2 || v < 2){
                l = 0;
                v = 0;
            }
            break;
        case 4:
            if(fmodf((float) l, 2.0) != 0 || fmodf((float) v, 2.0) != 0){
                l = 0;
                v = 0;
            }
            break;
        case 5:
        case 6:
        case 7:
        case 8:
        case 9:
        case 10:
            if(l != v){
                l = 0;
                v = 0;
            }
    }
    *vi = v;
    *li = l;
}

/* Comparar la imagen actual con otras en la BD y analizar vértices y lados */
iEntityPtr CompararImagen(int num_vertices, int vertices_ig, int lados_ig){
    iEntityPtr familia, tipo;
    long nv, vi, li;
    iDAFSError e;

    familia = i_GetChildEntity(basedatos);

    /* Si la BD está vacía, no hay nada que comparar */
    if(familia == NULL)
        return(NULL);

    /* Recorrer iterativamente las familias y los tipos de polígonos */
    do{
        e = i_GetLongProp(familia, "num_vert", &nv);
        HAY_ERROR_FATAL
        if((int) nv == num_vertices){
            tipo = i_GetChildEntity(familia);
            do{
                e = i_GetLongProp(tipo, "vert_ig", &vi);
                HAY_ERROR_FATAL
                e = i_GetLongProp(tipo, "lados_ig", &li);
                HAY_ERROR_FATAL
                if(((int) vi == vertices_ig) && ((int) li == lados_ig))
                    return(tipo);
                tipo = i_GetNextEntity(tipo);
            } while(tipo != NULL);
            return(familia);
        }
        familia = i_GetNextEntity(familia);
    }
}

```

```

    } while(familia != NULL);

    /* Si no hay un polígono similar, no retornar nada */
    return(NULL);
}

/* Guardar imagen prototípica en la BD como una nueva entidad */
iDAFSError AlmacenarProto(iImagePtr figura, int num_vertices, int vertices_ig, int lados_ig){
    iEntityPtr nueva_fig;
    char nombre[30];
    iDAFSError e;

    /* Almacenar la imagen y su área completa */
    nueva_fig = i_NewEntity(iRegionT, basedatos);
    i_SetImage(nueva_fig, figura);
    i_SetBox(nueva_fig, i_GetImageBounds(figura));

    /* Pedir la familia del polígono (triángulo, cuadrilátero, pentágono, etc.) */
    printf("Familia del polígono: ");
    scanf("%s", nombre);

    /* Almacenar la familia y cantidad total de vértices */
    e = i_SetText(nueva_fig, nombre, 100);
    HAY_ERROR
    e = i_SetLongProp(nueva_fig, "num_vert", (int4) num_vertices);
    HAY_ERROR

    /* Almacenar una copia del prototipo como un subtipo de polígono */
    e = Almacenar(nueva_fig, figura, vertices_ig, lados_ig);
    HAY_ERROR
    return(DAFSOK);
}

/* Guardar imagen específica en la BD como una nueva entidad */
iDAFSError Almacenar(iEntityPtr padre, iImagePtr figura, int vertices_ig, int lados_ig){
    iEntityPtr nueva_fig;
    char nombre[30];
    int i;
    PUNTO *vertices_reales;
    uint2 tamano;
    iDAFSError e;

    /* Almacenar la imagen y su área completa */
    nueva_fig = i_NewEntity(iRegionT, padre);
    i_SetImage(nueva_fig, figura);
    i_SetBox(nueva_fig, i_GetImageBounds(figura));

    /* Pedir el tipo de polígono (regular, irregular, etc.) */
    printf("Tipo de %s: ", i_GetText(padre, NULL));
    scanf("%s", nombre);

    /* Se almacena el tipo de polígono y el número de vértices y lados iguales */
    e = i_SetText(nueva_fig, nombre, 100);
    HAY_ERROR
    e = i_SetLongProp(nueva_fig, "vert_ig", (int4) vertices_ig);
    HAY_ERROR
    e = i_SetLongProp(nueva_fig, "lados_ig", (int4) lados_ig);
    HAY_ERROR
    return(DAFSOK);
}

int main(void){
    char archivo[50];
    iDAFSError e;

    /* Inicializar librería DAFS */
    e = i_InitLib();
    HAY_ERROR_FATAL

```

```

/* Leer base de datos, o crearla si no existe */
e = i_ReadEntity(ARCHIVO_DATOS, &basedatos);
if(e != DAFSOK){
    basedatos = i_NewEntity(iNullT, 0);
    e = i_SetText(basedatos, "Poligonos", 100);
}

/* Bucle principal del programa */
do{
    printf("\nNombre de la imagen ('l' para mostrar lista, 'q' para salir): ");
    scanf("%s", archivo);
    if(strcmp(archivo, "l") == 0)
        system(LISTAR_FIGURAS);
    else if(strcmp(archivo, "q") == 0)
        break;
    else
        e = ProcesarImagen(archivo);
} while(true);

/* Guardar la base de datos */
printf("Guardando base de datos...\n");
e = i_WriteEntity(ARCHIVO_DATOS, basedatos, DAFSB);
HAY_ERROR_FATAL

/* Salir del programa en forma normal */
i_ExitLib();
return(0);
}

```