Vectorización de planos de ingeniería por emparejamiento de contornos

Agradecimientos

En este apartado se desea dejar constancia de la enorme gratitud hacia todas aquellas personas que han tenido relación con la realización de este proyecto, y en especial a:

- Jose María Gomis Martí, por el interés mostrado.
- Mª Carmen Juan, por toda la ayuda prestada en el aprendizaje del Visual C++ y las facilidades prestadas.
- Miguel Angel Landete y Carlos Díaz, por su gran ayuda en las primeras fases del proyecto y amenizarme las largas horas en el laboratorio.
- Vicente Escuderos por facilitarme bibliografía y buenos consejos junto a un café.
- Tomás, por sus interesantes ideas sobre el tema comentadas durante la instalación del Windows NT.
- Kiks, Averias, Paspa y Pollo por soportarme estoicamente.

1. Introducción.

1.1 Ráster y vector.

Uno de los mayores componentes en cualquier diseño industrial y actividades de producción es una adecuada documentación y en este contexto, los dibujos de ingeniería juegan un papel muy importante. Siendo de naturaleza gráfica, estos dibujos representan información compleja de una manera concisa. Los dibujos de ingeniería abarcan documentos tan diversos como planos, esquemas, diagramas, etc. Por ejemplo el conjunto de documentación de un proyecto típico de magnitud razonable como puede ser un reactor nuclear incluye aproximadamente 30.000 documentos técnicos. Considerando que el 25 % de estos documentos se consideran activos y que el tiempo de vigencia de los diseños de la mayoría de los productos oscila entre 10 y 40 años, podemos entender la tarea que supone soportar este conjunto de documentos. De echo, a finales de los 80 se calculaba que sólo en Estados Unidos había más de 2000 millones de documentos activos. La mayoría de las compañías almacenan un gran número de tales dibujos en sus archivos y aproximadamente un 20 % están activos cada año. Una enorme cantidad de horas de trabajo humano se consume en crear, actualizar y mantener esos dibujos usando técnicas de delineado convencionales.

Los sistemas de CAD y GIS proporcionan un medio eficiente para crear, almacenar y actualizar dibujos de ingeniería y mapas, pero estos beneficios todavía no se encuentran disponibles para la multitud de dibujos existentes en papel debido a la ausencia de un puente apropiado entre el mundo manual y el de los ordenadores. La introducción manual de dibujos en una base de datos computerizada es un proceso lento, caro y tedioso, por lo tanto se siente necesidad de un sistema automático que escanee un dibujo y lo convierta en un formato adecuado para su posterior procesamiento en un ordenador. No obstante, el reconocimiento y comprensión de dibujos de ingeniería es con mucho una tarea más complicada que requiere de la capacidad de percepción visual y de la interpretación inteligente.

Ráster y vectorial son las dos estructuras básicas para almacenar y manipular datos espaciales en un ordenador. Los paquetes de CAD, GIS o de Diseño Gráfico más importantes disponibles hoy en día están basados de manera primordial en una de las dos estructuras, o basados en ráster o basados en vectores, aunque pueden tener algunas funciones para soportar el otro extremo de algún modo.

Los datos ráster vienen en forma de pixeles individuales y cada posición espacial o elemento de resolución tiene un pixel asociado donde el valor del pixel indica un atributo, como color, elevación, o número de identificación.

Los datos de un ráster se adquieren normalmente mediante un escáner óptico, una cámara CCD digital u otros dispositivos de entrada. Su resolución viene determinada principalmente por la resolución del dispositivo de entrada y la fuente de datos como puede ser un mapa, un plano o un diagrama en papel. Puesto que los ficheros ráster de datos deben tener pixeles para todas las posiciones, están limitados por el tamaño del área que representan. Incrementando la resolución espacial dos veces, el tamaño total de un conjunto de datos ráster bidimensional se incrementará 4 veces porque el número de pixeles se duplica en las dimensiones X e Y. Esto mismo sucede cuando se pretende cubrir un área de mayor tamaño.

Los datos vectoriales tienen la forma de puntos y líneas que están geométricamente y matemáticamente asociados. Los puntos están almacenados usando coordenadas, por ejemplo, un punto bidimensional se almacena como (x,y). Las líneas se almacenan como series de puntos.

Mientras que la forma ráster tiende a ser más cercana a las fuentes del mundo real, la forma vectorial es una pura abstracción del mundo y se obtiene normalmente a través de un proceso de digitalización. Ya que la forma vectorial es flexible y eficiente para representar datos espaciales, especialmente mapas y dibujos CAD y necesitan menos recursos del sistema para su manipulación y almacenamiento, históricamente la mayoría de los programas de GIS y CAD trabajan basándose en información en formato vectorial.

1.2 Algo de historia.

La historia de la vectorización a partir de imágenes escaneadas comienza a finales de los 60 con una compañía (Visicom Inc.) comprometida en escanear y convertir datos ráster en información vectorial usando grandes ordenadores IBM para convertir gráficas de datos analógicos producidas por instrumentos de laboratorio. Más tarde, esa tecnología se aplicó para capturar información de mapas, por aquella época la compañía vendió la tecnología a Broomall Industries Inc., actualmente Scangraphics Inc. Scangraphics es una de las dos compañías supervivientes que siguen vendiendo escaners y sistemas con tecnología pionera en los últimos años 60.

La otra compañía, Laser Scan Ltd., la cual comenzó en Cambridge, Inglaterra, desarrolló tecnología basada en la captura de trazas de partículas en cámaras de nubes en experimentos de física nuclear. Inicialmente, Laser Scan vendió un producto llamado Fastrak, el cual posteriormente se llamó V-Trak. Ambos sistemas de Laser Scan usaron tecnología laser para escanear y seguir (seguimiento de líneas semiautomático) información sobre películas transparentes. Esto difirió de la aproximación original de Visicom, que era escanear en un ráster un dibujo o un mapa, convertir en diferido la información del ráster en segmentos de línea, y posteriormente procesar los segmentos de línea usando gráficos de ordenador interactivos. Algunas compañías más tenían productos de seguimiento semiautomático de líneas, pero se usaron en dibujos a mano alzada. Los dibujos a mano alzada normalmente eran dibujos a tamaño natural que bosquejaban una parte de un avión con fines de ensamblaje. Tridea Systems propiedad de Mc Donnell Douglas, construyó un sistema así durante los 70. También, Gerbert Scientific Instruments produjo una versión de un seguidor de líneas similar, que podía unirse a algunos de sus grandes trazadores de mesa. Aunque las dos tecnologías originariamente diferían, ahora Laser Scan se basa en el escaneado de rasters para obtener la información usada para el seguimiento semiautomático de líneas, usando técnicas de gráficos por ordenador interactivos.

La siguiente compañía en escena era Scitex-Israel cuyo fundador había trabajado para un contratista de la defensa de los Estados Unidos realizando procesamiento de imágenes. Scitex originariamente aplicó técnicas de escaneo sobre patrones dibujados a mano para programar y controlar tejedoras para la industria textil. Más tarde, Scitex se interesó por la cartografía. Estableció un cuartel general estadounidense para obtener negocios de algunas de las agencias de cartografía estadounidenses. Scitex también desarrolló algunas de las primitivas técnicas de edición de rasters, las cuales se aplicaron a mapas escaneados. La edición de rasters también fue usada para crear y cambiar diseños en operaciones de preimpresión en color en la industria de impresión. Más tarde Scitex encontró más lucrativo el mercado de preimpresión y virtualmente dirigió todos sus esfuerzos hacia este negocio.

Las agencias cartográficas del gobierno fueron el principal objetivo de las primitivas empresas de escaneado-vectorización. Scangraphics, produjo sistemas para la Agencia Cartográfica de la Defensa que fueron usados para la captura de información

de mapas de curvas de nivel para generar modelos digitales del terreno para el programa del misil Cruise.

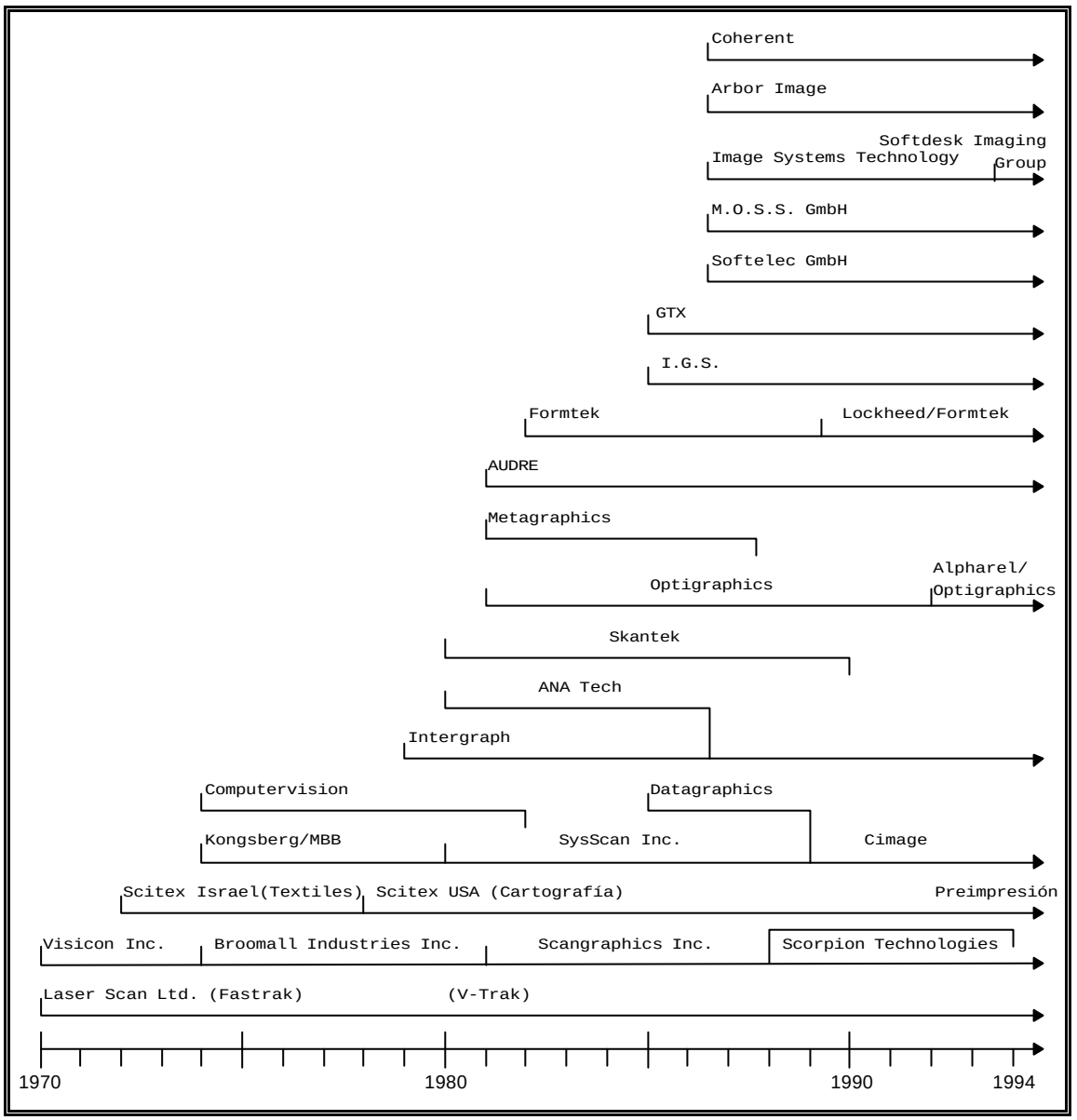


Figura 1

Cronológicamente, la siguiente compañía que siguió a Scitex fue Kongsberg de Noruega, la cual se unió con MBB de Alemania con el fin de crear una compañía para escanear y procesar datos ráster. Esta compañía tomó dos direcciones en el mercado, una usando todo el procesamiento ráster para la publicación técnica dentro de la industria aerospacial y la otra usando conversión de ráster a vectores para aplicaciones cartográficas. Después de la formación de esta alianza, la compañía se convirtió en SysScan Inc., que más tarde se unió con Datagraphics para convertirse en Cimage Corporation.

Una compañía más que entró en el mercado durante los 70 fue Computervision Corporation cuyo primer y único gran negocio de escaneado fue un contrato con el Departamento de Agricultura de los Estados Unidos para producir un gran sistema cartográfico automatizado. Adicionalmente a este sistema cartográfico, Computervision creó sistemas para escanear esquemas de circuitos para propósitos de bocetado automático. Este sistema nunca se hizo popular y fue abandonado a principios de los 80.

A principios de los 80 comenzaron varias compañías de escaneado-vectorización, incluyendo a Intergraph, la cual obtuvo un gran contrato del ejercito australiano para una aplicación dentro de sus operaciones cartográficas; ANATech, la cual comenzó como una compañía independiente y más tarde se fusionó con Intergraph; Skantek que más tarde desapareció; Optigraphics, la cual continúa como parte de ALPHAREL/Optigraphics; Metagraphics, la cual más tarde desapareció; y AUDRE, la cual todavía continúa. También en este periodo comenzó Formtek, que exploró en sus principios el uso de bocetos ráster como una alternativa al CAD, bajo la dirección del doctor Charles Eastman de la Universidad Carnegie Mellon. Formtek fue comprada por Lockheed en 1990 y continúa hoy en día como una compañía de gestión de documentos.

Durante la última mitad de los 80, varias nuevas compañías comenzaron en los Estados Unidos incluyendo a GTX Corporation, Information and Graphic Systems, Image Systems Technology (la cual ha sido adquirida recientemente por Softdesk Inc.) y Arbor Image. M.O.S.S. GmbH y Softelec GmbH comenzaron en Alemania. Una evolución cronológica de compañías de escaneado para la vectorización se muestra en la figura 1.

Mientras que muchas pequeñas compañías con paquetes basados en PCs ofrecieron capacidades de conversión de ráster a vectores, la mayoría no se convirtió en un factor significativo en el mercado. La única otra compañía en ser mencionada es Coherent Radiation, un contratista de la defensa que ha aplicado técnicas de inteligencia artificial para capturar mapas para compañías de servicios públicos. Otro contratista de la defensa, TASC, también ha aplicado su tecnología de inteligencia artificial a la captura de mapas de servicios públicos y también pudiera estar haciendo algunos negocios con compañías eléctricas.

1.3 Tecnología de escaneado.

El método más popular de convertir información en 2D a su forma electrónica es el escaneado, en el cual una "instantánea" de la información es procesada en una imagen ráster digital de mapa de bits. Esta imagen ráster esta formada por información descompuesta en diminutos puntos, o pixeles, de modo muy parecido a una imagen en una pantalla de televisión. La calidad de una imagen ráster depende de la talla de los pixeles individuales (la resolución). La resolución se mide en pixeles (o puntos) por pulgada (dpi o ppp).

Los primeros escaners de ráster fueron del tipo tambor, en los que el documento estaba montado en un rodillo giratorio y la información era registrada mediante un tubo fotoeléctrico. Más tarde, evolucionaron para incorporar el uso de fotodiodos y con el tiempo dispositivos de acoplamiento de carga (CCD). Aunque los escaners de tambor ya no son tan comunes como lo fueron, la tecnología todavía se usa en áreas como la cartografía, donde se requiere precisión y resoluciones extremadamente altas.

La mayoría de los modernos escaners de gran formato usan una técnica en la cual el documento se desplaza a través del campo focal de una cámara (o cámaras) que contiene múltiples elementos CCD. Estas matrices CCD tienen normalmente dos pulgadas de longitud y contienen 5000 elementos CCD. De este modo, si un escáner contiene una cámara con 5000 elementos que cubre una anchura de escaneado de 36 pulgadas, la resolución óptica del escáner se puede calcular dividiendo 5000 pixeles por 36 pulgadas, lo que es aproximadamente igual a 140 pixeles por pulgada, o 140 dpi. Un modelo de dos cámaras tendría una resolución óptica alrededor de 270 dpi, un modelo de tres cámaras sobre 400 dpi, y así sucesivamente. En las unidades de cámara múltiple, las cámaras deben solapar sus áreas de cobertura unos pocos pixeles con el fin de evitar perder datos. Esta área de solapamiento es conocida como punto de cosido. Los puntos de cosido de cámara en un escáner de gran formato son un ajuste crítico, y cualquier desalineamiento- de las cámaras entre sí o entre las cámaras y la ventana de escaneado puede terminar en una perdida de datos, dobles imágenes, escalado incorrecto o distorsión de formas.

La tecnología que usa matrices de CCD junto con lentes de cámara ha sido incorporada por los mayores fabricantes de escaners, incluyendo ANATech, Scangraphics, Vidar y Contex (comercializados por Ideal, Vemco, Calcomp y Océ Bruning). También está siendo usada otra técnica en la cual una única barra de CCDs

alineados de hasta 36 pulgadas reemplazan a las cámaras. El documento se mueve a través de la barra CCD iluminada en la que se detecta la información. Una de las ventajas de usar esta aproximación es que no hay puntos de cosido que puedan llegar a desalinearse. Estas barras CCD iluminadas normalmente tienen 200 o 400 elementos por pulgada, resultando una resolución óptica real de 200 o 400 dpi. Dos fabricantes que ofrecen escaners que usan esta tecnología son Xerox y Widecom.

Podemos darnos cuenta de que la mayoría de los fabricantes de escaners anuncian la capacidad de sacar una resolución que puede ser el doble de la cantidad de la resolución óptica real de la unidad. ¿Pueden hacer esto de verdad? Bueno, sí y no. Mientras que las cámaras sólo pueden ver imágenes a cierta dpi, la resolución puede ser incrementada en una dirección de la imagen simplemente con solo mover el documento a través del escáner en incrementos más pequeños de tamaño. Esto se conoce como interpolación.

Las técnicas de interpolación pueden permitir a un escáner de 200 dpi producir un fichero de 400 dpi. De cualquier modo, ya que las imágenes sólo pueden ser vistas por las cámaras a 200 dpi, en los detalles el fichero interpolado de 400 dpi nunca puede ser tan preciso como la imagen creada por un escáner con una resolución óptica real de 400 dpi. Para las aplicaciones tratadas en este artículo, una regla aproximada es usar 200 dpi para almacenar imágenes ráster y digitalizar “con las cabezas levantadas” (heads-up digitizing) y usar 400 dpi para las técnicas de vectorización automática. Las aplicaciones cartográficas pueden requerir resoluciones más altas.

Una vez que la información ha sido detectada por los CCDs, las señales analógicas alimentan un conversor analógico-digital. En este punto, los grados de variación de luz y oscuridad de la señal analógica son convertidos a niveles de gris en la salida digital o señal de escala de grises. La mayoría de escáners modernos usan escalas de grises de 8 bits, con lo que resultan 256 niveles del blanco al negro. Esta señal de escala de grises de 256 niveles de gris puede ser extraída por un ordenador a ficheros de formato estándar. Esto resulta óptimo para fotografías monocromas ya que contienen los diversos niveles de gris requeridos para construir la imagen. Un escáner en color trabaja de modo parecido, usando los tres colores primarios rojo, verde y azul. Puesto que ahora tenemos tres colores básicos en lugar de uno, la mayoría de formatos de fichero para imágenes en color soportan 24 bits. Los ficheros de escala de grises y los ficheros en color tienden a ser bastante grandes y pueden fácilmente ser de 10 veces el

tamaño de la imagen que únicamente contiene información en blanco y negro, conocido como fichero binario.

El formato de fichero binario es la salida más deseable para documentos que contienen líneas o información con alto contraste, y es el formato de fichero requerido por la mayoría de programas de vectorización automática y CAD. El proceso de escaneado descrito anteriormente solo es capaz de producir una salida en escala de grises. Para que un escáner produzca un fichero binario debe reducir sus niveles de gris a solo dos: negro (pixel encendido) y blanco (pixel apagado). Esta técnica se conoce como umbralización (thresholding), y en ella la señal de escala de grises es analizada y cualquier cosa por encima de un nivel de gris se vuelve negro y cualquier otra por debajo se vuelve blanco.

La mayoría de los escaners de gran formato ofrecen tecnologías de umbralización automática. Calculan un ajuste óptimo que es variado continuamente sobre el área del documento que está siendo escaneado. Los escaners que incorporan esta característica pueden reducir mucho el tiempo que lleva conseguir un escaneado aceptable, o establecer la diferencia entre tener una imagen con la que se puede trabajar o no. Si la mayoría de los dibujos tienen ruido, poco contraste, o están en malas condiciones, un escáner que utilice umbralización será una buena elección.

Actualmente, la mayoría de los escaners de rango medio en el mercado están muy próximos entre ellos en cuanto a funcionamiento, precio y especificaciones. Un principio a seguir es que no hay que tener en cuenta los reclamos de resolución interpolada y se debe elegir un modelo que tenga una resolución óptica real igual o superior a los requerimientos de salida deseados.

Se debe prestar mucha atención a las tolerancias de precisión especificadas por el fabricante, especialmente si las aplicaciones son de cartografía o GIS. Habrá que vigilar la garantía y políticas de asistencia del fabricante o distribuidor. Deberemos examinar cuidadosamente el software que viene con el escáner; esto puede ser más importante que el escáner en sí. ¿Es fácil de usar?, ¿tiene las características de funcionamiento que necesitamos, como borrado, corrección de desviaciones y limpieza?. Si no es así, deberemos considerar el usar software de terceros que ofrezca tales características. Finalmente, tendremos en cuenta cuánto vamos a estar utilizando la unidad. Si el escáner va a ser usado para aplicaciones de gran volumen o en turnos de 24

horas debería tenerse en consideración una máquina más sólida pensada para el trabajo intensivo que tendrá un precio sensiblemente superior.

1.4 Métodos de vectorización.

La *digitalización manual* usando una tableta digitalizadora ha sido ampliamente usada. Con este método, el operador traza manualmente todas las líneas desde su copia del mapa usando un puntero y creando un mapa digital idéntico en su ordenador. Una línea se digitaliza recogiendo una serie de puntos a lo largo de la línea.

Aunque este método es bastante directo, requiere operadores experimentados y consume mucho tiempo. Para un complejo mapa de curvas de nivel, puede llevar a una persona de 10 a 20 días conseguir el mapa completamente digitalizado.

Otra gran desventaja de este método es su baja precisión. La precisión de la digitalización manual meramente depende de lo precisa que sea la duplicación a mano de la copia del mapa en el ordenador. El nivel de precisión espacial que la mano humana es capaz de resolver está entorno a los 40 dpi (dots per inch) en el mejor de los casos, y será más bajo si el operador está cansado y aburrido después de trabajar en ello durante un periodo de tiempo. Se realizó un experimento en una universidad, se pidió a un grupo de estudiantes de geografía que digitalizarán el mismo mapa y los mapas finales fueron superpuestos uno encima del otro para crear un nuevo mapa. El resultado no fue sorprendente, el nuevo mapa estaba fuertemente distorsionado en comparación con el mapa original.

La digitalización manual es soportada por la mayoría de los paquetes de CAD y GIS mediante la conexión directa a muchas de las tabletas de digitalización populares.

Además de los anteriores métodos que parten de la información de un ráster también se pueden convertir dibujos lineales desde papel o película a una representación vectorial utilizando los siguientes métodos:

Vectorización “ciega”. En este método, el dibujo que va a ser vectorizado se pega sobre una tableta digitalizadora. El operador usa un ratón para introducir los puntos de inicio y fin de cada línea pulsando sobre ellos. Él está concentrado en el dibujo y no tiene retroalimentación de los resultados.

Vectorización “interactiva”. Se usa el anterior procedimiento, pero el operador tiene retoralimentación mediante la visualización de los puntos y líneas introducidos en una pantalla de alta resolución. La desventaja es que el operador

tiene que mover constantemente su cabeza entre la pantalla y la tableta digitalizadora. Esto podría no ser deseable desde el punto de vista de la ingeniería humana.

Hay 6 métodos básicos para convertir la información de un ráster en datos vectoriales para CAD:

Emplear seguimiento electrónico de líneas (establecido por LaserScan), ahora llamado seguimiento interactivo de líneas de datos escaneados en un ráster.

Escanear en un ráster el dibujo para crear un fichero ráster, conectar los puntos del ráster, adelgazar la línea resultante en un único vector o seguir los contornos de la línea para crear dos conjuntos de vectores. La mayoría de las aplicaciones requieren líneas finas en lugar de conjuntos de vectores duplicados.

Escanear en un ráster el dibujo, conectar los puntos del ráster, adelgazar las líneas y segmentar los vectores normalmente cortos en vectores en línea recta más largos con puntos de fin y posiblemente con anchuras de línea. El resultado es una serie de vectores en línea recta que representan el dibujo o mapa.

Escanear en un ráster el dibujo para crear un fichero ráster, conectar los puntos, adelgazar las líneas, segmentar las líneas en vectores más largos, interpretar las formas de las líneas, reconocer el texto y asociar el texto a las formas de las líneas. El resultado de estas operaciones es crear símbolos reconocidos y líneas conectadas junto con cualquier información textual que pueda describir el símbolo.

Escanear en un ráster el dibujo para crear un fichero ráster, entonces hacer corresponder el patrón de pixeles con un patrón estándar de pixeles que identifican un símbolo estándar. Este proceso no incluye la vectorización; es simplemente un método para hacer corresponder pixeles o patrones ráster con formas conocidas. Una relación numérica entre los puntos se usa para identificar el símbolo, por eso el símbolo reconocido debe estar muy próximo a los símbolos estándar conocidos. Cambios en el símbolo como la orientación o la talla pueden impedir que el símbolo sea reconocido.

Emplear el método de digitalización "con las cabezas levantadas" (heads-up digitizing), que implica escanear el dibujo para producir una imagen ráster en la pantalla, sobre la que los vectores son dibujados usando uno de los muchos

paquetes de software disponibles. Aunque AutoCAD y MicroStation son los dos paquetes de CAD más populares para usar esta aproximación, ahora virtualmente todo el software de CAD permite tipos similares de digitalización "con las cabezas levantadas". Algunos paquetes de GIS basados en vectores proporcionan algunas funciones de digitalización con la “cabeza levantada” dentro de sus programas, por ejemplo ArcInfo y ArcView de ESRI y MapInfo de MapInfo. Se implementa combinando la visualización de imágenes raster con su editor de vectores para guiar al operador al dibujar líneas y puntos justo encima de una imagen escaneada. De cualquier modo, para algunos sistemas software basados en vectores, hay algunas limitaciones importantes que les impiden manejar ciertos tipos y tallas de imágenes.

La digitalización con la cabeza levantada es similar a la digitalización manual desde el punto de vista de que las líneas tienen que ser trazadas a mano, pero trabaja directamente en la pantalla del ordenador usando la imagen raster escaneada como fondo. Cuando el programa no es capaz de tomar una “buena” decisión acerca de cómo continuar, se deja a juicio del operador. La ventaja es que parte de la vectorización se realiza automáticamente, la desventaja es que –dependiendo de lo que uno considera “bueno”– numerosos parámetros deben ser seleccionados previamente para obtener una vectorización satisfactoria. El termino "cabezas levantadas" se deriva del hecho de que los operadores trabajan en pantallas directamente enfrente de ellos, en lugar de trabajar en mesas de digitalización, que sería una aproximación con las "cabezas agachadas". A pesar de que las líneas todavía son trazadas a mano , el nivel de precisión es más alto que usando tabletas digitalizadoras porque las imágenes raster son escaneadas a alta resolución, desde 200 dpi a 1600 dpi normalmente . Con la ayuda de herramientas de visualización, tales como aumento y reducción de zoom el operador puede trabajar realmente con la resolución de los datos raster, es decir digitalizar con un nivel de precisión mayor. De cualquier modo, el nivel de precisión todavía no está garantizado ya que es altamente dependiente del operador y de su modo de digitalizar. Este método también consume mucho tiempo llevando aproximadamente lo mismo que el método de digitalización manual.

1.5 Niveles de inteligencia de los documentos escaneados.

Como se muestra en la tabla 1 hay definidos 5 niveles de inteligencia asociados con la información de dibujos escaneados.

Nivel de inteligencia	Contenido del fichero	Aplicaciones
1	Imágenes ráster.	Gestión de documentos y digitalización por capas.
2	Imagen ráster más texto ASCII y posiblemente vectores.	Gestión de documentos.
3	Vectores conectados.	Visualización CAD y posible digitalización por capas.
4	Vectores conectados con valores etiquetados.	Cartografía / GIS.
5	Vectores, símbolos, atributos asociados.	CAD / GIS.

Tabla 1

1. Imagen ráster mantenida como un ráster puro o fichero de pixeles. Normalmente, esas imágenes son almacenadas como ficheros ráster comprimidos para reducir los tamaños de los ficheros aproximadamente a un treintavo de la información originalmente escaneada.

2. Imágenes ráster escaneadas junto con información textual y vectorial asociada (híbrido). Este nivel de inteligencia puede ser creado escaneando un documento, usando la información ráster como fondo para situar vectores y/o textos asociados a los vectores. Un fichero híbrido puede contener alguna información de ráster y alguna vectorial para producir el resultado deseado.

3. Dibujos hechos de vectores conectados para formar una imagen bidimensional. Aunque este nivel de inteligencia puede proporcionar una imagen más comprimida que la imagen ráster escaneada, este tipo de fichero raramente se usa en aplicaciones CAD. Con ficheros ráster comprimidos a veces más pequeños que los ficheros vectoriales, hay poco que ganar con la conversión.

4. Vectores conectados con valores etiquetados. Ese tipo de fichero se usa por algunas aplicaciones de cartografía, particularmente en curvas de nivel con valores de elevación marcados, hidrología y clasificación de tierras con identificadores.

5. Vectores con símbolos conectados y atributos asociados. Este es el último objetivo para la mayoría de los sistemas CAD: una base de datos asociada. Este es también el nivel de base de datos más difícil de lograr a partir de datos escaneados.

De estos 5 niveles, los 4 primeros son relativamente fáciles de obtener con la actual tecnología de vectorización a partir de datos escaneados. Aunque existe tecnología para obtener el nivel cinco para algunos tipos de dibujos, este nivel de información es muy difícil de obtener en la mayoría de los casos. La mayor parte del resto de este capitulo se centra en los problemas asociados con este nivel.

1.5.1 Problemas de conversión a bases de datos inteligentes.

El mundo de los dibujos posee varios problemas para la conversión en una base de datos inteligente. Muchos dibujos tienen información desaparecida o errónea. En algunos tipos de dibujos, la información no está en la forma que se necesita para el escaneado. Por ejemplo, un detalle de un dibujo mecánico podría incluir una descripción gráfica de una parte a mayor tamaño, pero también podría incluir una tabla de valores de diferentes dimensiones relativas a diferentes partes numeradas. Esta información no puede ser manejada directamente desde un dibujo escaneado y necesitaría interpretación OCR y asociarse automáticamente al gráfico a extraer.

Un problema más difícil para la conversión automática es la asociación de información de atributos a los gráficos. Al menos una compañía fundada en los ochenta, Metagraphics, gastó considerables recursos tratando de resolver este problema en dibujos mecánicos, sin lograr una buena solución en cuanto a costes. La experiencia a mostrado que solo ciertos tipos de dibujos de una sola línea (dibujos esquemáticos) son buenos candidatos para asociar atributos y gráficos. Las tuberías y los diagramas de instrumentación son algunos de los mejores ejemplos que podrían tener una solución con esta técnica.

Además de los problemas que acabamos de indicar, el mundo de los dibujos de mala calidad también ha ralentizado el proceso de conversión. Los dibujos muy usados tienden a tener pliegues, arrugas y manchas, y a veces son cianotipos de poca calidad, reproducidos para ser distribuidos. Los colores sepia (p.e. por la acción de la luz y el

paso del tiempo sobre el papel) representan uno de los mayores desafíos, en particular aquellos que han sufrido cambios varias veces, causando amplias variaciones de sombras en el fondo en diferentes áreas del dibujo. Estas circunstancias requieren técnicas de umbralización dinámica, disponibles en la mayoría de los mejores escaners, para detectar y ajustarse a las diversas sombras del fondo.

En un estudio entre una serie de compañías privadas, el acuerdo general alcanzado en cuanto a la calidad de los dibujos fue que:

El 20% de los dibujos eran de excelente calidad para el escaneado.

El 40 % pudo ser escaneado usando las mejores técnicas de mejora disponibles en los sistemas de escaneado.

El 20% necesitó todas las capacidades de mejora de los escaners además de edición interactiva del ráster para ser limpiado. Nótese que este 20% está más allá de las capacidades de mejora de la mayoría de los escaners. Esta clase de dibujos requiere inteligencia humana para editar interactivamente información imposible de ser detectada y limpiada por los escaners disponibles.

El 20% era inútil escanearlos porque eran totalmente ilegibles, estaban rotos en varios pedazos, o simplemente muy deteriorados desde que se crearon. La mayoría de estos podrían ser mejorados fotográficamente; otros necesitarían ser redibujados.

1.5.2 Servicios de escaneado y conversión.

Las dificultades de convertir dibujos por medio de escaneado y conversión tanto para uso en CAD o GIS ha engendrado una industria de agencias que prestan este servicio. Las agencias de conversión han creado un nicho de mercado, realizando un servicio que requiere un largo entrenamiento y la adquisición de cierto equipamiento especializado. Muchas compañías con grandes cantidades de dibujos a ser convertidos para CAD o GIS han encontrado el coste de equipos, personal y formación poco rentable. Si añadimos el actual clima en la industria de reducción de tamaño y concentración en el núcleo de negocio de la compañía, probablemente no tiene sentido realizar las conversiones en la propia empresa.

Las oficinas de escaneado y conversión pueden proporcionar los servicios de conversión a un precio más competitivo y en mucho menos tiempo que el que se hubiera requerido para formar y entrenar personal interno. Y los servicios de conversión

de escaneados pueden realizar la conversión de documentos a cualquiera de los cinco niveles de inteligencia citados previamente.

1.5.3 Automatización de la conversión a bases de datos inteligentes.

Para crear automáticamente una base de datos inteligente asociada se requiere usar el cuarto método de vectorización anteriormente mencionado. Esto es, escanear en un ráster el dibujo, conectar los puntos, adelgazar las líneas, segmentarlas, interpretar formas (reconocimiento de símbolos), reconocimiento de atributos y asociación de atributos a los símbolos. El reconocimiento de símbolos requiere crear conjuntos de reglas de símbolos que definen símbolos específicos. La mayoría de la tecnología de hoy requiere de la programación de inteligencia artificial para crear las reglas de símbolos. La situación ideal, todavía no disponible comercialmente, es tener esa programación con una interface de herramientas gráficas que permitirían la localización en pantalla de un símbolo y la creación de la base de reglas del símbolo automáticamente a partir de la descripción gráfica. (Tal interface está disponible para emparejamiento de patrones - pattern matching -, pero los patrones reconocidos están limitados por las tallas y orientaciones de los símbolos que se almacenan como símbolos maestros para el emparejamiento. Las bases de reglas pueden ser más generalizadas para manejar variaciones en tamaño y orientación).

El uso de reglas basadas en la inteligencia artificial para emparejar simbolos requiere lógica difusa para incrementar la fiabilidad del reconocimiento en dibujos reales. La tecnología de redes neuronales está proveyendo de sistemas entrenables para un mayor éxito en el reconocimiento de símbolos y caracteres. Esta tecnología encabeza las soluciones para la automatización de la conversión en bases de datos inteligentes.

1.6 Aplicaciones.

La gestión de documentos se ha convertido en la partida de coste más alto dentro de las grandes compañías y es el segmento de mercado de más rápido crecimiento de la información computerizada. La gestión de documentos comprende la conversión, almacenamiento y distribución de todo tipo de información, ya sea a partir de papel y documentos en microfilm, ficheros de CAD, ficheros de GIS, procesamiento de textos, hojas de calculo, etc. Toda información necesita ser organizada en bases de datos para proporcionar acceso en segundos en lugar de días.

Los niveles de inteligencia 1 y 2 mencionados anteriormente, comprenden imágenes ráster puras. Las imágenes ráster con texto asociado y vectores pueden ser manejados por sistemas de gestión de documentos. Estos niveles de inteligencia son los menos costosos de crear a partir de papel y documentos en microfilm. Si los documentos van a integrarse en ficheros CAD, el nivel 2 puede ser fácilmente aplicado.

El simple visionado de un fichero CAD puede utilizar nivel de inteligencia 3, pero el valor práctico de convertir a un fichero totalmente vectorial simplemente con propósitos de visualización es cuestionable, puesto que una imagen ráster comprimida proporciona más información legible y posiblemente una menor talla de fichero. No existe mercado para la conversión de nivel de inteligencia tres.

Las aplicaciones cartograficas, principalmente curvas de nivel y información de uso del suelo, pueden ser manejadas con el nivel de inteligencia 4. Los vectores pueden ser marcados con valores de información de elevación o información de uso del suelo para ser usados por un GIS.

La información de nivel 5 que comprende una base de datos inteligente asociada, parece trabajar mejor con esquemas y otros diagramas de una sola línea tales como mapas, tuberías y diagramas de instrumentación.

El presente proyecto fin de carrera se encuadra dentro de una línea de investigación que trabaja en la reconstrucción geométrica tridimensional. Este campo de investigación pretende abrir nuevos caminos que reduzcan la distancia que hay entre el razonamiento gráfico humano y las interfaces de los programas de CAD. Dentro de este marco, la conversión de bocetos en papel a un formato vectorial en el que se expresen todos los elementos del dibujo como primitivas geométricas y la asociación de los símbolos normalizados que expresarán en estos dibujos información complementaria a la propia forma geométrica que será añadida a los modelos 3D, constituyen el primer paso en el camino hacia estos nuevos flujos de trabajo. Tal aplicación requiere del nivel de inteligencia 5 puesto que aparte de la información geométrica de las figuras habrá que discriminar alguna parte de ellas e interpretarla no como información geométrica, sino como información asociada al modelo 3D.

1.7 Análisis de vectorizadores.

1.7.1 Introducción.

El siguiente análisis pretende obtener una visión global del estado actual en el que se encuentra el mercado y las técnicas de vectorización. Dicho análisis se ha centrado en diez vectorizadores con distinto origen: dos de ellos (Corel OCR-Trace y MicroStation Reprographics) estaban disponibles en los ordenadores del DEGI (Departamento de Expresión Gráfica en la Ingeniería), otro más fue el trabajo realizado por Mª Carmen Juan y Vicente Escuderos en su PFC y los siete restantes (Streamline, TracTrix, I/Vector 3.5, PixEdit, VPmax, Scanvector y R2v) han sido estudiados a partir de las versiones demo obtenidas a través de Internet. Muchas otras compañías disponen de productos similares, pero debido a problemas en sus páginas web o por no disponer de versiones de evaluación, nos ha resultado imposible incluirlas en nuestro análisis.

Debido a que algunos de los vectorizadores imponen diversas restricciones a causa de su carácter de demos, resulta materialmente imposible comparar unos con otros en aspectos tales como la velocidad de vectorizado, el tamaño máximo de la imagen procesable, etc. ya que no es posible encontrar un banco de pruebas común a todos ellos en el que puedan trabajar en igualdad de condiciones (tamaños de imagen o ficheros de imagen restringidos).

Aunque los productos citados con anterioridad pueden agruparse bajo la denominación de vectorizadores ya que, a partir de un fichero raster obtienen un fichero en formato vectorial, se observa que por motivos comerciales evidentes, todas las compañías fabricantes a la hora de publicitar sus productos no restringen su uso a un campo en concreto, si no que los ofrecen como productos de uso general. Sin embargo, a través del análisis se podrá apreciar que según las prestaciones propias de cada uno su uso resulta más apropiado para un campo de trabajo en particular (CAD, GIS, diseño artístico, ...), y por tanto un vectorizador de uso general con un rendimiento satisfactorio en todas esas áreas será casi imposible de encontrar.

Aunque muchos de los vectorizadores incluyen herramientas para trabajar con imágenes en color y escala de grises y otras opciones especializadas para campos concretos (GIS, cartografía, diseño gráfico, ...), es necesario mencionar que no se ha prestado especial interés a dichas herramientas, limitándose a una simple mención de las

mismas ya que nuestro objetivo se restringe al campo del CAD en el que suponemos que trabajaremos con imágenes binarias.

1.7.2 Vectorizadores en versión demostración.

1.7.2.1 TRACTRIX

- Versión: TracTrix for CAD release 2.0 Demo Version.

- Empresa fabricante: Trix Systems Inc.

- Requerimientos:

Sistema operativo: Windows 95, 3.11, NT 3.51 ó NT 4.0
Espacio en disco duro: 3MB
No requiere de otro software para su funcionamiento.

- Publicidad en la red:

"Conversión de Raster a Vector*: Convierte dibujos escaneados (en monocromo o en color) a CAD y a formatos vectoriales NC tales como DXF, DWG e IGES.*
La conversión del TracTrix es muy precisa - puede rastrear un centro de línea en un par de milésimas de pulgada (las tabletas de digitalización manual pueden manejar solamente 25 milésimas). La conversión puede ser totalmente automática, manual (utilizando herramientas propias de vectores y herramientas que trabajan sobre el raster) o una combinación de las dos.

Conversión de Vector a Raster*: Convierta ficheros con formatos DXF y DWG a ficheros raster para transmisión electrónica y almacenamiento. Defina colores en su diseño de CAD para resaltar las líneas en la nueva imagen rasterizada. Elija la resolución que quiera para su imagen y guárdela en cualquiera de los treinta formatos de ficheros raster más utilizados.*

Vistas/Marcado de líneas/ Anotación/ Conversión de Raster a Raster*: Soporta más de 40 formatos de ficheros raster y vectoriales. Edita imágenes raster y vectoriales. Corta y pega imágenes y detalles de éstas entre diferentes ventanas o en otras aplicaciones tales como procesadores de textos.*

Gestión de la impresión*: Perfecto control de la impresión de ficheros raster o vectoriales en cualquier impresora láser. Escala, ajusta, asigna pesos a las líneas en cada trabajo de impresión. Reduce o elimina la dependencia de los plotters.*

Almacenamiento híbrido Raster/Vector*: Combina ficheros raster con extensiones DXF y DWG y los almacena en un único fichero para facilitar el archivado y la posterior edición.*

¿Cómo disponer del TracTrix?*: TracTrix R2 es una aplicación estándar para cualquier paquete CAD o NC. Versiones en inglés, chino, alemán, italiano, japonés y sueco. Las versiones en francés y polaco estarán disponibles en breve. TracTrix para AutoCAD 14 es una aplicación integrada en ObjectARX para usuarios de AutoCAD R14. Versiones en inglés y sueco. Las versiones en francés, chino, alemán, italiano, japonés y polaco estarán disponibles en breve.*

Precio*: El precio recomendado para la venta de TracTrix R2 o TracTrix para AutoCAD en todo el mundo es de 950 dólares por cada licencia.**"***

- **Diferentes formatos de ficheros admitidos**:

Entrada: TIX (el de TracTrix), TIFF, CALS, BMP, PCX, WMF, JPEG, DWG y DXF. Hay que decir que el programa sólo deja abrir los ficheros de prueba que acompañan a la versión de demostración.

Salida: Puede guardar los vectores en formato DWG, DXF, IGES, Postcript, DRW y HPGL. Las imágenes raster se pueden grabar en formato BMP, CALS, GIF, ICO, JPEG, Paint, PCX, PSD, varios tipos de TIFF, WMF, WPG (estos son los más conocidos). Sin embargo, el trabajo entero sólo se puede guardar en formato TIX, propio de TracTrix.

- **Tratamiento previo de la imagen**:

a) Edición: Hay diversas herramientas que permiten editar capas (añadir, borrar, esconder, ...), además de las clásicas de dibujo y borrado de la imagen. Además en el cuadro **propiedades** se pueden consultar un gran número de datos sobre la imagen: el nombre, medidas, tamaño en Kb, el tipo de fichero, la resolución, los colores que usa y el tamaño original del raster (importante para saber si la imagen original era en color o en blanco y negro, dado que pasar de uno a otro supone un ahorro de espacio muy significativo).

b) Preprocesado: Hay un filtro que permite **limpiar** la imagen, pudiendo especificar el tamaño de dicha limpieza (de pequeña a grande, en una escala). También se pueden **llenar huecos** en la misma, escogiendo su tamaño en una escala.

Es posible **insertar** una imagen raster o un dibujo de Autocad en el fichero de trabajo actual. El raster además se puede **invertir**, **voltearlo** en horizontal o vertical y **cambiarlo** (de formato, altura, anchura y resolución). El raster puede representarse en modo simple (más rápido), antialiasing y en escala de grises (muy recomendable).

Hay diversas anchuras de línea (cada una con un color), pudiéndose cambiar el color asignado a cada grosor. Las unidades de medida pueden ser milímetros o pulgadas.

- **Opciones de vectorización**:

a) Por línea centrada, donde se puede especificar un grosor máximo de línea de 0 a 2 milímetros (por defecto es 1.2 mm).

b) Por línea exterior.

Las entidades de dibujo que se usarán al vectorizar pueden ser de tres tipos:

- Líneas, arcos y círculos.
- Polilíneas.
- Curvas Bezier y líneas.

Según el tipo elegido el programa intentará vectorizar las entidades de la imagen aproximándolas a las entidades seleccionadas.

La opción de ajustes permite introducir diversos valores para adecuar la vectorización a las exigencias del usuario, como son la tolerancia (precisión, ruido, radio máximo de un arco y radio mínimo de una esquina) e intersecciones (distancia entre ellas para hacerlas una sola y ángulo de las mismas). Los vectores se pueden ajustar ortogonal, vertical u horizontalmente.

Se permite editar cada capa, ver la separación entre ellas, poner líneas de una determinada anchura en una capa distinta y saber qué anchura tiene las líneas de una capa concreta (se suele especificar como un rango en milímetros).

- **Valoración**:

El TracTrix es un producto con una vectorización bastante precisa, aunque los textos, símbolos y puntas de flecha no los reconozca como tales. Aunque en principio (y como puede desprenderse de su nombre) este programa estaría orientado a vectorizar imágenes para su posterior tratamiento en CAD, en nuestra opinión no estaría orientado a ningún campo específico, ya que por ejemplo en planos de ingeniería no reconoce las

entidades más importantes que se pueden encontrar en ellos (líneas rayadas, puntas de flecha, etc.). Tampoco tiene herramientas específicas para GIS y cartografía ni opciones orientadas al diseño gráfico. En líneas generales, se puede considerar al producto como satisfactorio, aunque no del todo para nuestro campo.

1.7.2.2 R2V

-Versión: R2V for Windows 95 & NT 3.0.11 (también existe una versión para Windows 3.1).

Al tratarse de una demo, no es posible exportar imágenes de dimensiones superiores a 512 x 512 pixeles.

-Empresa fabricante: Able Software Co.

-Requerimientos:

Sistema operativo: Windows 95, Windows NT ó Windows 3.1.

Sistema: 486 PC o superior.

Memoria: 8MB RAM, 16MB o más recomendados para un funcionamiento óptimo.

-Publicidad en la red:

"Able Software Company desarrolla software de conversión automática de raster a vector para GIS, mapas y aplicaciones de CAD. El R2V para Windows y NT es el más reciente paquete de software de conversión de raster a vector diseñado para vectorizar de forma automática mapas escaneados o para crear mapas digitales provenientes de fotos aéreas o de imágenes de satélite. .

El R2V está siendo usado actualmente en más de 60 países por cientos de usuarios para la digitalización automática de mapas y como aplicación para la obtención de datos para GIS.

El R2V para Windows 95 y NT es un potente software de conversión de raster a vector a un precio razonable. El R2V combina la capacidad de la tecnología de vectorización automática con una interfaz gráfica con el usuario guiada por menús y fácil de usar en el entorno de trabajo Windows 95 y NT. El software convierte mapas escaneados o imágenes, a formato vectorial para su posterior uso como mapas, sistemas de información geográfica (GIS), CAD y aplicaciones informáticas científicas.

El sistema es sencillo de utilizar y rápido de aprender incluso para aquellos usuarios que no poseen ningún tipo de experiencia técnica.

El R2V proporciona una sencilla y completa solución para digitalizar imágenes provenientes de distintas fuentes, tales como mapas escaneados o para crear mapas digitales provenientes de fotos aéreas o de imágenes de satélite. La totalidad del proceso de conversión en estructuras vectoriales es completamente automático y no necesita intervención humana. Visualice la imagen escaneada en su pantalla y seleccione el comando de vectorización. ¡Eso es todo lo que necesita!. Todas las líneas se extraen en segundos y se visualizan sobre la imagen para que puedan ser verificadas y editadas. Se proporcionan poderosas funciones de edición y procesado para editar, trabajar con geo-referencias y etiquetar sus datos. El R2V posee todas las herramientas para obtener un conjunto perfecto de datos vectoriales más rápida y más sencillamente que por cualquier otro método.

Con el R2V, ya se puede ir olvidando del lento e impreciso trazado a mano sobre tabletas digitalizadoras; simplemente escanee su mapa o dibujo y deje que el R2V lo vectorize automáticamente, con un elevado nivel de precisión. Un mapa de curvas de nivel escaneado a 200 ppi (puntos por pulgada) en blanco y negro o en escala de grises puede ser vectorizado en segundos o minutos en un PC.

Sabemos que la edición de la imagen raster y los datos vectoriales son extremadamente importantes para usted, así que hemos hecho el esfuerzo para construir editores inteligentes y fáciles de usar por usted para manejar todos los tipos de datos visualizados en una misma ventana, incluyendo líneas, puntos, polígonos, texto, etiquetas de texto, pixeles de la imagen y puntos de control. Con el R2V, puede vectorizar de manera automática mapas o dibujos, llevar a cabo rápidas digitalizaciones en pantalla y fotos aéreas de georeferencia o imágenes de satélite y actualizar sus conjuntos de datos vectoriales.

¿Tiene algunos mapas en color?. No hay problema. Escanéelos en color, el R2V clasificará los colores y vectorizará cada color por separado. Luego etiquete las líneas utilizando las herramientas de etiquetado de vectores semiautomáticas..

Esto es lo que dicen nuestros clientes del R2V:

"El R2V es el mejor y más rápido software de vectorización automático de mapas que nunca he visto", dice P. Hastings, Director del Instituto Medioambiental de

Thailandia. En el Instituto, siete estaciones de trabajo R2V NT están siendo utilizadas diariamente para la digitalización automática de un gran número de mapas topográficos e incluso fotos aéreas.

Para ver como puede utilizar el R2V para acelerar sus proyectos y ahorrar tiempo y dinero, por favor escríbanos, llámenos o envíenos un fax ahora."

- **Formatos de ficheros admitidos**:

Entrada: Opción *open*: TIFF, SPOT Images HDR y BMP que contengan imágenes binarias, en escala de grises o en color.

Opción *import*: Linefiles (ARC), SDL, Point Files PNT y XYZ Point Files (XYZ).

Salida: ARC (líneas), PNT (puntos), MIF (MapInfo), DXF, SHP (ArcView 2), SDL y XYZ para guardar las imágenes vectoriales (mediante la opción *export vector*) y TIFF para las imágenes raster.

- **Tratamiento previo de la imagen**:

a)Edición: Para la edición del raster disponemos de una opción que nos proporciona **información** sobre la imagen (tamaño, tipo de imagen y número de planos) además de las típicas opciones de **zoom** para ampliar o reducir la imagen, mostrarla completa o ampliar un rectángulo seleccionado. También existe una herramienta que nos permite conocer el **valor** de un píxel en particular, **modificarlo** y en caso de trabajar con imágenes en color o escala de grises, permite **cambiar** a un nuevo valor todos los pixeles de la imagen de un determinado color.

b)Preprocesado: Dispone de una serie de filtros para conseguir imágenes **simétricas** (vertical y horizontalmente), **girar** la imagen (90º o en el sentido y número de grados especificados), **invertir** los colores (NOT lógico a nivel de bits), **recortar** la imagen rectangularmente, cambiar su **resolución** y **deformarla** (usando puntos de control). Para trabajar con imágenes en tonos de gris disponemos de un **suavizado** de la imagen que filtra usando la media o la mediana, **borrado de fondos** oscuros, **mejora de contornos** (usando un filtro Sobel, los gradientes de la imagen, o dilatando los contornos), selección de **umbrales** de vectorización y **segmentado** de la imagen en regiones homogéneas. Para trabajar con imágenes en color tenemos una herramienta especifica para agrupar los colores de la imagen en clases. Existe una última

herramienta que sirve para **convertir** entre sí los distintos formatos de información del color y transformar imágenes en color a escala de gris.

- **Opciones de vectorización**:

Apenas disponemos de opciones de vectorización, ya que sólo podemos seleccionar el tipo del documento fuente (mapa o dibujo CAD) y el tipo de seguimiento a realizar (contornos o centro de líneas). El tipo de vectorización que se realiza dependiendo del tipo del documento fuente es muy diferente. Mientras que con los mapas los vectores generados intentan ajustarse con la mayor fidelidad a las líneas del raster, con los diseños los vectores se ven menos afectados por las irregularidades de las líneas, pero de cualquier modo no se reconocen entidades. La vectorización se puede realizar sobre la imagen completa o solo en un área rectangular previamente seleccionada.

También existe una modalidad de vectorización semiautomática que funciona de modo muy similar a como lo hace MicroStation ReproGraphics en uno de sus modos, es decir, sigue una línea hasta que llega a una intersección, momento en el cual se detiene el proceso en espera de una indicación del usuario acerca de la dirección en la que debe seguir el proceso. Existe otra herramienta semiautomática similar a la anterior pero que permite realizar el mismo proceso de seguimiento de líneas sobre varias de ellas simultáneamente sólo con seleccionarlas.

El OCR que incorpora R2V es bastante interesante aunque resulta costoso el comenzar a trabajar con él. Para poder reconocer texto primero debemos indicarle el tamaño de los caracteres de los textos que debe intentar localizar. Una vez hecho esto debemos corregir los errores que comete al detectar y agrupar los posibles bloques de texto. Una vez tenemos los bloques correctos procederemos a “entrenar” al OCR. Esto último consiste en ir marcando uno de los bloques que contienen los vectores resultantes de la conversión de una letra o símbolo y asociarle manualmente a dicho nombre esta letra o símbolo. Éste es un proceso costoso la primera vez que se realiza puesto que por cada letra distinta que aparece en el texto deberemos realizar esta operación. Todavía resulta más engorroso si tenemos en cuenta que para una misma letra deberemos repetir este proceso cada vez que nos la encontremos con distinto tamaño, tipo de letra u orientación. Disponemos de una herramienta para comprobar si un bloque determinado lo reconoce correctamente o no con solo pulsar sobre él. A continuación deberemos convertir los bloques de texto en notas para finalmente corregir los errores que pudieran existir y terminar el proceso.

- **Particularidades**:

R2V dispone de multitud de herramientas y utilidades para trabajar con sistemas de georeferencia, puntos de control, generación de polígonos, etc... que resultan de gran ayuda para trabajar en cartografía o GIS. Además permite trabajar con ficheros en formatos usados en aplicaciones de este campo para permitir el intercambio de datos tanto de entrada como de salida con estas aplicaciones (ArcInfo y ArcView de ESRI y MapInfo de MapInfo).

Otra de las características de esta versión de R2V es su capacidad para vectorizar imágenes en batch, es decir, que se puede lanzar en diferido varias imágenes para ser vectorizadas. Para ajustar las opciones de vectorización a cada una de las imágenes dispone de la capacidad de interpretar scripts de control.

R2V posee multitud de herramientas para manipular la salida vectorial generada. Estas herramientas permiten actuar sobre puntos añadiéndolos, borrándolos, moviéndolos o asignándoles un identificador. También existen herramientas para trabajar con vectores que además de las operaciones anteriores correspondientes permiten añadir, borrar y mover nodos de las Polilíneas, partirlas, unirlas, cerrarlas, copiarlas y resaltarlas.

- Valoración:

R2V es un programa orientado a la vectorización de imágenes para GIS y Cartografía. Aunque incluye opciones para vectorizar dibujos de ingeniería, los resultados que se obtienen con ellas no son satisfactorios para nuestros objetivos ya que no reconoce entidades y no se dispone de suficientes parámetros para ajustar el proceso de vectorización a las características del dibujo. Por otra parte las posibilidades que ofrece para trabajar con imágenes en tonos de gris y en color, junto con los formatos de ficheros que soporta para conectarlo con diversos programas de GIS hacen de él una buena alternativa a la vectorización “heads-up”, la cual también puede emplearse con este vectorizador aunque esencialmente se oriente hacia la vectorización automática.

1.7.2.3 I/VECTOR 3.5

-Versión: I/VECTOR 3.5 Demo Version.

Al tratarse de una demo, la empresa limita sus capacidades restringiendo el área de trabajo a una porción de la imagen. Existen otras versiones del mismo producto como I/Vector LT o I/Vector 3.0 pero con características más restringidas.

-**Empresa fabricante**: Ideal Scanners & Systems.

-Publicidad en la red:

"I/VECTOR es la forma más rápida y sencilla de conseguir que sus dibujos queden almacenados en un formato vectorial inteligente para todas las variedades de CAD, GIS, Publicaciones técnicas, Gestión de datos de productos y aplicaciones para Sistemas de Gestión de datos de Ingeniería.

I/VECTOR es adecuado para expertos pero no necesita ser utilizado por un operador de CAD experimentado para obtener excelentes resultados.

I/VECTOR es fácil de usar y rápido de aprender por que sólo existen tres pasos básicos:

1.Previsualice la imagen raster y seleccione el tipo de dibujo.

2.Ajuste los parámetros para el reconocimiento y pruébelos sobre una sección del dibujo.

3.Convierta el dibujo entero y guárdelo en un fichero con formato DXF (o DGN).

Ventajas del I/VECTOR.

Es rápido. Convierte documentos de tamaño A1 en menos de 5 minutos.

Visualiza los estructuras vectoriales resultantes en una capa sobre su imagen raster original para verificar la precisión. Trabaja bajo Windows 95, Windows NT, Windows 3.1x, SUN, HP, AIX, y versiones de SGI.

Disponga del programa I/VECTOR 3.5 de conversión de Raster a Vector por 3995 dólares.

Es la mejor solución profesional para convertir documentos. El I/VECTOR 3.5 permite el procesamiento de múltiples documentos y la preparación de los mismos para su conversión, mientras otros documentos se están convirtiendo en background. El paquete I/VECTOR 3.5 incluye:

-Módulo completo de vectorización con múltiples interfaces para documentos y capacidad para trabajar en batch.

-Editor de raster para la limpieza, actualización, rotación y alineación de imágenes.

-Editor de regiones para asignar conjuntos variables de parámetros a diferentes secciones de un mismo dibujo..

-Registro de mensajes que registra los eventos de conversión para propósitos de monitorización y facturación.

-Gestión de capas para la personalización de las salidas en ficheros DXF o DGN.

"I/VECTOR es el programa de conversión de raster a vector más competente que he visto hasta la fecha. La ayuda en línea incluye el "cómo hacerlo" más inteligente que he visto en ninguna ayuda de Windows. Usted probablemente perderá mucho menos tiempo limpiando la imagen después de la conversión de lo que lo hubiera hecho con cualquier otro programa."

-David Byrnes, CADalyst Magazine, Agosto de 1995

IDEAL SCANNERS LANZA AL MERCADO EL I/VECTOR 3.5

18 de Junio de 1996 Rockville, MD - IDEAL Scanners and Systems, Inc. lanza al mercado el I/Vector 3.5, un rápido conversor de raster a vector de 32 bits que convierte imágenes raster escaneadas en ficheros DXF y DGN vectoriales.

El I/Vector 3.5 reconoce de un modo inteligente círculos, arcos, símbolos, texto OCR en cualquier ángulo, estilos de línea, anchura de línea, rayados, puntas de flecha y contornos de área para ser utilizados en CAD, GIS y publicaciones técnicas y sistemas de gestión de datos en Ingeniería. Debido a que el I/Vector 3.5 trabaja guiado por parámetros, todas las funciones están definidas y controladas por el usuario. Una nueva caja de herramientas permite a los usuarios manipular y editar cadenas de texto reconocidas por un OCR. Los usuarios son capaces de visualizar la estructura vectorial como una capa sobre la imagen raster original con tal de verificar los resultados. El I/Vector 3.5, con calidad contrastada de reconocimiento y optimización, convierte documentos de 3 a 5 veces más rápido que las versiones anteriores.

Además del editor de raster para la limpieza, actualización, rotación y alineación de imágenes y el editor de regiones para asignar conjuntos variables de parámetros a diferentes secciones de un mismo dibujo, el I/Vector 3.5 incluye un editor de texto para ser utilizado antes de la vectorización. El I/VECTOR 3.5 permite el

procesamiento de múltiples documentos y la preparación de los mismos para su conversión, mientras otros documentos se están convirtiendo en background. Ambas imágenes, raster y vectorial, son visibles a través de los zooms, desde la imagen completa hasta vistas ampliadas con el fin de una mejor estimación del ancho de las líneas, tamaño de los textos y tolerancias. Además, el gestor de capas del I/Vector dispone las entidades repartiéndolas cada una en su capa apropiada, incluyendo líneas estándar, texto, líneas discontinuas o punteadas y personalización de los ficheros DXF de salida.

Diseñado alrededor de una interfaz gráfica con el usuario (GUI), el I/Vector 3.5 es fácil de usar y posee un registro de mensajes que almacena los eventos de conversión para propósitos de monitorización y facturación. El I/Vector 3.5 también incluye un extensísimo sistema de ayuda en línea. El sistema de ayuda contiene información de todos los menús e iconos, botones y los parámetros relacionados con la vectorización. "El I/Vector 3.5 va revolucionar el mercado de la conversión de raster a vector ya que tanto los expertos como los usuarios novatos pueden utilizar este paquete de software para convertir dibujos y documentos," afirma Sean Eikenbery, Director de Desarrollo de Software de IDEAL.

El I/Vector 3.5 ha añadido el formato CALS Grupo 4 a los formatos de ficheros raster existentes, los cuales incluyen RLC, TIFF Grupo 4, RLE, y PCX, y convierte a ficheros DXF y DGN vectoriales. El I/Vector 3.5 trabaja bajo Windows 3.1, 95, NT, y UNIX/Motif. Puede trabajar en modo sencillo o batch y su precio es de 3.995 dólares.

IDEAL Scanners and Systems, Inc. integra, fabrica y soporta un amplio rango de soluciones de escaneado, desde dibujos de formatos grandes y pequeños y dibujos técnicos hasta escaneados centrados exclusivamente en la impresión de documentos, incluyendo conversión de raster a vector con el OCR incorporado. Para más información sobre el I/Vector 3.5 contacten con Kathy Magenheim al (301) 468-0123 ext. 1230 o vía e-mail al kathy@ideal.com."

-**Formatos de ficheros admitidos**:

Entrada: TIFF, GP4, CG4, CAL, MIL (CALS), RLC, RLE, IMG, PCX, GEN (Vectorlist).

Salida: GEN (propio de Ideal), DXF y DGN.

-**Tratamiento previo de la imagen**:

a) Edición: Posee opciones de **zoom** (zoom in, zoom out, pantalla completa, tamaño natural y zoom anterior), de **medida**, **rejilla** centrable por el usuario y con posibilidad de definir su densidad (el cursor sólo se mueve por las posiciones de la rejilla cuando ésta está activa), posibilidad de limitar las líneas de dibujo a horizontales y verticales (opción *ortho*), herramientas de **edición** del raster (lápiz, líneas, polilíneas, arcos, polígonos, rectángulos, círculos, además de rectángulos círculos y polígonos rellenos) incluidas en la misma caja de herramientas, así como el *undo* y el selector de tintas (sólo puede trabajar con dos colores: blanco y negro, puesto que este programa sólo trabaja con ficheros binarios) y herramientas de **borrado** (permite borrar lo que queda fuera de un polígono o rectángulo definido por el usuario).

Para el borrado individual de pixeles, no existe una herramienta específica si no que, al trabajar sólo con imágenes binarias, éste se consigue dibujando sobre el objeto a borrar con el color de fondo.

También posee la capacidad de transformar la imagen, **reflejándola** (horizontal o verticalmente), **invirtiendo** los colores del fondo y del dibujo, **rotándola** (90º, 180º o 270º) o **alineándola** con respecto a una línea horizontal o vertical definida por el usuario, que todas las demás tomarán como referencia para su orientación.

b) Preprocesado: Posee un **filtro de limpieza** de manchas en la imagen (*speckle filter*), con la posibilidad de indicar el tamaño (en pixeles) del área que inscribe a la mancha y si ésta es cuadrada o rectangular. También permite seleccionar el color de las manchas (blancas o negras) y la superficie a limpiar (la imagen completa o una porción rectangular de ella).

Existe otro filtro (*amplify*) que permite realizar operaciones de **erosión** y **dilatación** sobre huecos en la imagen, seleccionando el color a amplificar (blanco o negro respectivamente). Permite cuatro grados de intensidad que se corresponden con el número de pixeles a modificar en los huecos. Al finalizar una sesión de edición y preprocesado se pueden aceptar o rechazar los cambios realizados sobre el raster.

- **Opciones de vectorización**:

Carece de las típicas opciones (outline, centerline, etc ...), pero ofrece multitud de **parámetros** que controlan la vectorización:

a) Reconocimiento:

a.1 .- Área rellena mínima.

a.2 .- Radio mínimo.

a.3 .- Tamaño mínimo de entidades válidas aisladas.

a.4 .- Presencia de símbolos y texto.

a.5 .- Presencia de diferentes tipos de línea.

a.6 .- Grosores de línea.

a.7 .- Áreas rayadas.

b) Alineamiento:

-Alineamiento del dibujo. Toma la línea más larga y considera su orientación horizontal o vertical para corregir pequeñas desviaciones en las demás líneas del dibujo.

-Alineamiento de rectas. Corrige pequeñas variaciones sobre la horizontal o la vertical en las líneas rectas que forman parte del dibujo.

c) Tolerancias:

c.1 .- Ajuste de líneas.

c.2 .- Ajuste de círculos y arcos.

c.3 .- Salto de pequeños espacios en blanco.

c.4 .- Borrado de pequeñas líneas con un extremo inconexo (“hairs”).

c.5 .- Ajuste respecto del vectorizado con respecto al raster.

d) Parámetros extra:

d.1 .- Conexión a través de los puntos de cruce.

d.2 .- Aislamiento de marcas de frontera.

d.3 .- Desplazamiento de los vectores con respecto al raster.

-**Gestión de parámetros**:

Existen cinco conjuntos de parámetros de vectorización preestablecidos, según las características de la imagen a vectorizar: General, Mecánica, Arquitectura, Mapas Catastrales y Mapas de Líneas de Nivel. Además, se pueden editar los parámetros de estos conjuntos aunque algunos no estén disponibles para todos ellos. Los nuevos conjuntos así generados pueden guardarse en un fichero (con extensión .PAR), cargarse

y borrarse. Los parámetros de vectorización no solo se pueden asociar a una imagen completa sino que también pueden asociarse a una región de ella. Dichas regiones pueden tener cualquier forma poligonal y pueden solaparse, aunque solo un conjunto de parámetros tendrá validez para la zona solapada.

-Parámetros en la creación de ficheros .DXF:

Comprenden la unidad de medida (mm o pulgadas), el juego de caracteres (ANSI u OEM), formato de salida de polilíneas (permite convertirlas a líneas para facilitar su edición), número de decimales en las cifras del fichero, características de las capas (nombre, color, aparición o no en el fichero de salida), capas extra y eliminación de capas. El concepto de capa en el I/Vector consiste en una agrupación de objetos vectoriales que ha reconocido como del mismo tipo (entidades como arcos, puntas de flecha, líneas del mismo grosor, círculos, texto, rayados, ...), asignándoles un identificador, un color y su estado de visualización (si están ocultas o visibles). Existe un editor de textos que permite editar las entidades texto que se ha reconocido (añadir textos nuevos, borrarlos, modificarlos, orientarlos, separar dos textos que están juntos y fijar el tipo de letra y tamaño) aunque no es muy cómodo su manejo debido a su falta de flexibilidad de uso.

Como última característica de este vectorizador señalar que posee la opción de vectorizar en batch una serie de dibujos con un conjunto de parámetros específico para cada uno. Mientras el proceso batch está en curso, puede ser interrumpido o reanudado y las imágenes pueden ser cambiadas de orden en la cola, eliminadas o pueden modificarse sus parámetros.

Antes, durante y después de la vectorización se dispone de una ventana que muestra los mensajes emitidos por la aplicación, discriminándolos según su tipo. Esta ventana actúa a modo de registro y facilita las tareas de monitorización y facturación.

- Valoración:

I/Vector resulta un producto interesante para vectorizar dibujos de ingeniería debido al gran número de parámetros y opciones que incluye. Todas estas opciones permiten al operador configurar el programa para adaptarse fácil y rápidamente a las características de cada imagen, ya que con un poco de experiencia y aprovechando que se pueden almacenar los ajustes de los parámetros para su uso posterior, esto se puede conseguir casi de modo automático.

La separación en capas de objetos de la misma clase permite controlar fácilmente los resultados de la vectorización. Por otra parte, el editor de textos reconocidos no resulta demasiado cómodo de usar aunque es bastante completo. El editor de vectores posiblemente por tratarse de una demo no dispone de herramienta alguna por lo que no cumple en absoluto la finalidad que se le supone.

1.7.2.4 PIXEDIT.

Versión: PixEdit V. 3.989-A0

Esta versión demo inhabilita los comandos *save, save as, copy* y *cut*, con lo que se imposibilita producir cualquier salida con la misma.

- **Empresa fabricante**: TechSoft.

- **Publicidad en la red**:

"PixEdit es un editor de raster profesional para escanear, realinear, modificar, vectorizar y convertir entre formatos de ficheros imágenes bitonales (con dos colores). Todo, desde documentos A4 a una cara o multipágina hasta grandes dibujos en formato A0 o más grandes, se procesan rápida y fácilmente por PixEdit bajo MS Windows. En los complicados entornos software de hoy en día, la eficiencia y la productividad son dos factores importantes al considerar la incorporación de una nueva herramienta de edición gráfica. PixEdit maneja la carga, procesamiento y almacenamiento de todas las grandes y complejas imágenes bitonales (dos colores) con increíble facilidad. Es rápido y fácil de usar, permitiendo gastar el valioso tiempo en otras tareas. PixEdit ha sido instalado en el entorno de varios negocios y puede reemplazar fácilmente a otros editores de documentos debido a su versatilidad, potentes funciones de procesamiento y detección automática de formatos de ficheros para más de 80 formatos de raster. PixEdit ha demostrado ser una y otra vez la mejor elección para procesar documentos técnicos y mapas de gran formato.

Puntos fuertes del producto:

Amplias posibilidades de edición del raster con Undo y Redo ilimitados.

Soporta un amplio conjunto de los escáners de gran formato más populares además de completo soporte TWAIN.

Potentes funciones de procesamiento para filtrado y limpieza de documentos.

Función de transformación para la corrección de mapas permitiendo la composición de mapas sin costuras.

Reorientación automática y manual de imágenes escaneadas.

Soporte de múltiples páginas y capas de raster.

Incorpora funciones de vectorizado y rasterizado.

Opciones especializadas de impresión..

Potentes macros.

Ventanas de navegación y ampliación fáciles de usar.

Rápida interfaz amistosa para el usuario con ayuda sensible al contexto.

Amplia interfaz con otras aplicaciones a través de DDE.

Soporta más de 80 formatos de raster que se detectan automáticamente.

Tecnología de compresión bajo petición para formatos de imágenes comprimidas.

¡Rapidísimos documentos TIFF A0(E) que se cargan en 1 segundo!

Contacte hoy mismo con nosotros para descubrir lo fácil que es incorporar PixEdit a su entorno software, proporcionándole una herramienta de edición de documentos de coste razonable y altamente eficiente. Techsoft está comprometida en desarrollar y distribuir software de alta calidad, proporcionándole la solución software que se adecue a sus necesidades especificas de edición.

Funciones de vectorización y conversión a raster incorporadas. *El vectorizador incorporado de PixEdit convierte áreas de un documento o el documento entero, para que pueda ser cargado en otros sistemas como AutoCAD. La vectorización se puede realizar automáticamente en grandes grupos de documentos mediante el uso de macros. Como método alternativo a la función de vectorización de PixEdit, las imágenes raster o las áreas seleccionadas también pueden salvarse en formato DXF como vectores horizontales. La función de conversión a raster de PixEdit acepta ficheros de vectores como por ejemplo el formato de fichero HPGL."*

- **Formatos de ficheros admitidos**:

Entrada: TDF (Techsoft), ACAD Cadcamera, ANA Tech G4, ANA Tech LRD, APOLLO HDRU, BMP, BROOKTROUT FAX, CALS, CIMAGE DSI, CLP (Clipboard), CMU WinManager, DATACOPY (IMG), DCX, DR. HALO PUT, Edmics (C4), ELDAK SCN, GEM (IMG), Generic Fax, Hitachi HRF, HPGL 7475A, IBM MOD:CA (A??), IFF/ILBM, Image Machines (TG4), Intergraph Bynary (BIN), Intergraph CIT, Intergraph RLE9, Intergraph TG4, Jt Fax (0??), LaserData (LDA), Lotus Manuscript (BIT), Macintosh PICT (PCT), MacPaint (MAC), MGR, Microsoft Paint (MSP), Microtek Eyestar (IMG), Notis OND, PBM, PC Paint Plus 2.0 (PIC), PCX, PFS ART, PhotoShop (PSD), Ricoh IS30 (PIG), RLC, SmartFax (0??), SUN (RAS), SysScan LSC, Talaris (TIF), Targa (TGA), uncompresed (IMG), Vidar (SCN), WordPerfect (WPG), XBM, Xionics SMP, XWD.

Salida: DXF, HPGL para la parte vectorial. Para la parte raster podemos realizar conversión entre todos los formatos de entrada.

-Tratamiento previo de la imagen:

a) Edición: PixEdit permite escanear, visualizar, editar, imprimir y convertir entre formatos imágenes raster binarias de hasta tamaño A0. Una de las características que cabe resaltar en PixEdit son los distintos **zooms** que posee porque aparte de los tradicionales (ampliar, reducir, ampliar una zona, ver toda la imagen, ajustar a la máxima anchura y ver a tamaño natural) podemos usar dos ventanas especiales que nos facilitan el trabajo. La primera de ellas llamada “Overwiew” nos ofrece una **visión panorámica** de la imagen completa en la que aparece indicada la zona de ella visualizada en ese momento en pantalla. El tamaño de esta ventana puede ser variado por el usuario y permite desplazar la zona marcada dentro de la imagen en tiempo real. Resulta muy útil en imágenes grandes donde no es difícil perderse dentro del dibujo. La otra ventana llamada “blowup”, consiste en una ampliación en tiempo real de la **zona circundante** a la posición del cursor. Se pueden cambiar las dimensiones de la zona ampliada para hacer que esta ventana amplíe más o menos la imagen.

PixEdit solo puede trabajar con una imagen, aunque los documentos con que permite trabajar pueden componerse de varias páginas y ofrece herramientas para moverse y editar todas estas páginas. Este programa puede trabajar con hasta 8 capas superpuestas para formar la imagen, pero en un momento determinado tan sólo una de ellas estará activa para permitir su edición. Cualquier entidad puede ser copiada o movida de una

capa a otra. Las capas pueden ser guardadas individualmente en cualquier formato. Existe una barra de control de capas para gestionar todas sus características: nombre, visibilidad, color, prioridad y posibilidad de edición. Para salvar todas las capas de una imagen podemos usar el formato propio de TechSoft que almacena todas ellas junto con sus atributos dentro de un mismo fichero.

También disponemos de una caja de herramientas para :

- Abrir un fichero como una nueva imagen a editar, una página a añadir a la imagen actual o como una figura a insertar en ella.

- Guardar la imagen actual.

- Imprimir.

- Copiar un área determinada de la imagen al portapapeles.

- Pegar el contenido del portapapeles en una de las capas de la imagen.

- Escanear utilizando el escáner predeterminado.

- Moverse a través de las diversas páginas de las que puede constar el documento que contiene la imagen en edición.

- Aumentar o reducir la imagen en pantalla.

- Rotar la imagen 90º, 180º ó 270º.

- Seleccionar áreas conteniéndolas en un rectángulo, en un polígono o seleccionar los pixeles iluminados conectados con uno dado.

- Duplicar las áreas seleccionadas.

- Medir distancias y ángulos para poder usarlos automáticamente dentro de otras funciones.

- Añadir cotas a la imagen.

- Borrar el interior de un rectángulo, de un polígono, o utilizar el ratón como una goma de borrar.

- Rellenar áreas cerradas con su color opuesto.

- Dibujar rectángulos, círculos, arcos y polilíneas.

- Deshacer y rehacer acciones anteriores.

También hay disponible una rejilla cuya densidad, posición y estilo pueden ser definidos por el usuario.

b) Preprocesado: Este vectorizador no posee herramientas de limpieza. PixEdit posee multitud de filtros de tratamiento de la imagen: adelgazamiento de líneas (thinning), extracción de contornos, dilatación de líneas (closing), inversión de colores, cambio de dimensiones y de orientación, reflejos horizontales y verticales. En caso de existir deformaciones en la imagen escaneada y existir una rejilla o un patrón regularmente repetido en ella, es posible corregir tales deformaciones con la información extraída de la rejilla; además, se puede eliminar dicha rejilla de la imagen. Todo esto se lleva a cabo gracias a dos filtros especiales (*Map Correction* y *Map Grid Removal* respectivamente). Existe un último filtro que permite el tratamiento de las capas de la imagen realizando operaciones lógicas a nivel de bit entre ellas. El resultado de dichas operaciones se puede almacenar en cualquiera de las capas.

- **Opciones de vectorización**:

La vectorización que lleva a cabo PixEdit es muy elemental y los vectores no aparecerán como una capa de la imagen ya que únicamente la aproxima por líneas rectas y sólo se pueden establecer los siguientes parámetros básicos: formato de salida (DXF o HPGL), precisión de la vectorización (cuanto mayor es el valor, más cortos serán los vectores generados y más se ceñirán al raster), vectorización por contorno o por esqueleto (para vectorizar las líneas siguiendo su centro o sus bordes). Hay una opción especial de vectorización que consiste en aproximar la imagen del raster mediante vectores horizontales contiguos; con esta opción el aspecto de la imagen es muy similar al del raster pero genera ficheros de gran tamaño y nula utilidad en CAD. Por último, hay una opción que se aplica en caso de vectorizar una porción de la imagen que se utiliza para que en el fichero de salida los vectores conserven la misma posición que ocupaba el área vectorizada en el raster y no tomen un nuevo origen de coordenadas.

- **Particularidades**:

PixEdit soporta diversos modelos de escáners además del estándar TWAIN, estos escáners son: Contex, Vidar truscan, Vidar 4220, Vidar 4240 (S), Vidar 4250S, Bell & Howell, Xerox 7336, HP Scanjet, Fujitsu y Xionics. También permite imprimir las imágenes con diferentes opciones para controlar el tamaño y demás características de la salida impresa (tamaño natural, ajuste a la página de impresión, uso de la resolución de la impresora, escalado de la imagen, división de la imagen en múltiples páginas,

impresión de parte de la imagen e impresión de cada capa de la misma de un modo distinto para diferenciarlas).

Una característica destacable de PixEdit es su capacidad de facilitar la construcción y la ejecución de complicadas macros que permiten realizar casi cualquier tarea. El programa dispone de un lenguaje especial de programación de macros, e incluso en caso de disponer del programa en varios ordenadores de una red, permite el procesamiento en paralelo de un conjunto de imágenes mediante la opción “Macro Server”.

Otra peculiaridad de PixEdit es que puede trabajar como aplicación servidora de otras aplicaciones clientes. La integración de PixEdit dentro de otras aplicaciones se consigue a través de la DLL de Windows DDEML o del protocolo basado en mensajes DDE.

- Valoración:

PixEdit resulta una buena herramienta de preprocesado de imágenes binarias debido al gran número de formatos que acepta y al conjunto de herramientas de edición de raster de que dispone. Por otra parte, la vectorización que realiza es bastante primitiva al no disponer de reconocimiento de entidades. Tampoco dispone de herramientas de edición vectorial ya que la vectorización se incluye como un simple puente hacia los paquetes de CAD que serían los encargados de editar los vectores.

1.7.2.5 VPSTUDIO.

- **Versión**: VPSTUDIO 6.01 Demo Version.

  Esta demo tiene dos tipos de restricciones, la primera es el **tamaño** (abre imágenes de cualquier tamaño, pero sólo trabaja con imágenes de 297 x 210 mm como máximo) y la segunda son los **formatos de salida** (aunque con el VPSTUDIO se pueden grabar ficheros con la opción *export* con multitud de formatos distintos -ver párrafo sobre formato de ficheros- ésta versión demo no permite grabar con extensiones .DWG y .DGN).

- **Empresa fabricante**: Softelec Corporation.

- **Requerimientos:**

  Sistema operativo: Windows® 95, Windows® NT.

  Sistema: (486)/Pentium.

  Memoria: 32MB.

-Publicidad en la red:

"VPStudio. *Escanee directamente de su escáner de tamaño E. Importe su selección de raster: color, escala de grises, blanco y negro con reducción de color y efectúe la clasificación y extracción de ficheros para una fácil vectorización automática o interactiva. Edición híbrida y reconocimiento de símbolos con atributos. La vectorización automática incluye un eficiente rastreo de líneas interactivo, y exporta en formato DWG, DGN, DXF o IGS.*

VPMaxPro. *Está un paso por debajo de las características del VPStudio, pero proporciona una conversión profesional de raster a vector para mapas y dibujos grandes incluyendo pequeños bocetos DTP. VPMaxPro ofrece soporte para escáner de gran formato en blanco y negro con una integración muy eficiente de la vectorización automática e interactiva (incluido el rastreo de líneas) con potentes características de edición para producir datos vectoriales preparados para una aplicación CAD.*

VPMax. *Es nuestro programa de conversión independiente para dibujos de tamaño D y mayores. Esta versión incluye una extensa gama de edición del raster, bandas de goma y calibración del raster. La caja de herramientas del editor de vectores incluye muchas entidades de CAD como líneas, arcos y círculos, además de la edición de líneas ortogonales, estilos de línea, círculos, etc. simplemente pinchando en ellas. La sección de procesado de vectores del VPMax está diseñada para el vectorizado preestablecido de varios tipos de dibujos como los mecánicos, eléctricos, mapas GIS, etc. Los parámetros preestablecidos pueden grabarse para procesar por lotes dibujos de un tipo concreto, o ajustarse a cada dibujo.***"***

- **Formatos de ficheros admitidos**:

Entrada: - opción *open*: sólo permite abrir el formato del vpstudio: .VCF.

- opción *import*: permite abrir TIF, PCX, GIF, BMP, DXF; RLC, IG4 y GP4 (de Image System), CAL y CG4 (de CALS) y RLE y CIT (de Intergraph).

Salida: - opción *save* : sólo permite salvar con el formato del vpstudio: .VCF.

- opción *export*: permite salvar TIF, PCX, GIF, BMP, DXF, DGN (de MicroStation), IGS (de IGES) y CAL, GP4 y CG4 (de CALS).

-Tratamiento previo de la imagen:

a)Edición: A parte de las típicas opciones de **zoom** (zoom in, zoom out, centrado en pantalla, ...), **inversión** y demás, permite **rotaciones** (de 90º en 90º o los grados que se desee), **reflejar** la imagen (horizontal y verticalmente) y **escalarla** (isotrópica -manteniendo la forma de la imagen- y anisotrópicamente). También posee otra herramienta que permite "trocear" la imagen y almacenar cada uno de los trozos en un fichero distinto y después tratar cada uno de dichos trozos por separado.

b)Preprocesado: Existe una **herramienta de limpieza** con la posibilidad de quitar manchas de la imagen (mediante la opción *remove speckles*). Para llevar a cabo dicha limpieza se nos pide el tamaño de la mancha en pixeles (es decir, el valor hasta el cual una agrupación de pixeles se considera una mancha y no una parte del dibujo) o bien se selecciona de la imagen una mancha en concreto para que limpie todas las manchas iguales o menores a la elegida.

- **Opciones de vectorización**:

El proceso de vectorización se divide en tres pasos:

1.- Vectorización de la imagen: Existen tres tipos de vectorización. En el primero de ellos, **centerline**, se elige el centro de las líneas para vectorizar la imagen, en la vectorización **outline** se convierten los contornos de las estructuras raster, y el último tipo, la vectorización **centerline/outline**, es un método intermedio entre el primer y el segundo. El valor que se le da -llamado *threshold*- es el valor umbral en pixeles a partir del cual se vectoriza una línea utilizando su contorno -opción outline- y no su esqueleto -opción centerline). En el cuadro de diálogo de estas tres opciones se incluyen las opciones de vectorizar toda la imagen, un área poligonal o un área rectangular de la misma. Este paso proporciona una "primera vectorización" en la que todos los elementos del raster son aproximados mediante rectas.

2.- Elección de los parámetros del post procesado: En este segundo paso se fijan a gusto del usuario los valores (altura del texto, longitud de las cabeza de flecha, diámetro de los círculos, "reparto" de las capas -existen tres capas, la azul para las líneas de contorno, la roja para las líneas de cota y ejes y la verde para texto-, ...) de los que depende el post procesado de la imagen vectorial. Existen 4 conjuntos de valores de parámetros predefinidos y se pueden definir 6 conjuntos más, existiendo dos formas de dar valores a dichos parámetros: bien desde teclado o bien situándose sobre el raster y

eligiendo con el puntero del ratón la longitud de las cabezas de flecha, el diámetro de los círculos y arcos, la altura del texto, ...

3.- Postprocesado de la imagen vectorial: En este último paso, se discriminan todas las entidades en la imagen vectorial. El postprocesado se realiza en 7 pasos, una vez finalizado el último de ellos, todas las entidades identificadas (líneas de contorno, líneas de cota, arcos, círculos, elipses, cabezas de flecha, texto y símbolos) quedan resaltadas en la imagen vectorial por colores (las líneas de contorno en azul, las líneas de cota y las cabezas de flecha en rojo, los círculos en azul claro, los arcos y el texto en verde, ...), con la posibilidad de editar y añadir las características de las entidades (coordenadas, el centro de los círculos y arcos, el texto, ...) pudiéndose añadir también nuevas entidades.

- Valoración:

EL VP Studio es un vectorizador eficiente para el tratamiento de planos de ingeniería. La flexibilidad y alto grado de personalización de los parámetros de vectorización (aquí llamados “de preprocesado”), la óptima identificación de entidades (tanto en cantidad como en calidad), la incorporación de OCR y la posibilidad de integrar fácilmente un escáner al sistema (a través del estándar TWAIN) hacen de él un programa muy recomendable para su uso.

1.7.2.6 SCANVECTOR.

- **Versión**: SV ScanVector demo version 1.2.

Esta versión demo tiene restricciones en lo referente a los ficheros de entrada (sólo permite abrir imágenes de 2000 x 1400 pixeles) y de salida (no se permite guardar ficheros vectoriales con el formato DXF).

- **Empresa fabricante**: Cybersonic Technologies.

- Requerimientos:

Sistema operativo: DOS V.5.0 y Windows 3.1, Windows 95, Windows NT.

Sistema: 386/486/586 con coprocesador matemático.

Memoria: 8Mb RAM.

Espacio en disco duro: 4 MB

- **Publicidad en la red**:

"Conversión de raster a vectores. *ScanVector es un programa para Windows que convierte de raster a vectores automáticamente. ScanVector convierte cualquier fichero en formato TIFF o BMP en un fichero DXF para poder importarlo en su aplicación de CAD/GIS.*

Aplicaciones. *ScanVector es la herramienta profesional para trasladar sus dibujos a su entorno de CAD:*

** Ilustraciones técnicas.*

** Arquitectura.*

** Ingeniería mecánica.*

** Ingeniería eléctrica.*

** Registro de propiedades.*

** Ingeniería civil.*

** Aplicaciones GIS (Geographical Information System).*

Funcionalidad. *Los algoritmos inteligentes usados por ScanVector transforman líneas, arcos y círculos automáticamente en vectores. La función de OCR de ScanVector graba los caracteres reconocidos como texto ASCII en una capa separada.*

+ <u>Edición de mapas de bits</u>

** Alineamiento/corrección de distorsiones.*

** Funciones de filtro y eliminado de manchas.*

** Closing y adelgazamiento de líneas.*

** Generación de superficies y contornos.*

** Borrado y edición de pixeles.*

** Lápiz y dibujo de líneas rectas.*

** Inversión.*

** Espejo y giros.*

** Rotación de 90 grados.*

+ <u>Vectorización automática</u>

** Conversión automática de raster a vectores.*

** Reconocimiento de líneas, arcos, círculos y polilíneas.*

** Distinción de grosores y tipos de línea (líneas discontinuas).*

** Reconocimiento de superficies.*

** OCR (Optical Character Recognizer).*

** Vectorización de toda el área o de algunas partes de la misma.*

+ *Otras funciones*

** Transformación de coordenadas.*

** Importación de capas individuales (prototipo DXF).*

** Gestión de capas.*

** Selección de vectores y asignación a capas.*

** Edición e inserción de texto.*

** Edición secuencial del texto reconocido.*

** Inserción de bloques DXF.*

+ *Interfaz gráfico de usuario*

** Presentación simultánea de mapas de bits y vectores.*

** Funciones de zoom.*

** Diferentes ventanas (aplicaciones MDI).*

** Visualización individual (activado/desactivado) de mapas de bits, texto y vectores.*

Formatos de entrada-salida. *Los dibujos o documentos pueden ser escaneados con escáners comerciales de mesa. ScanVector maneja documentos de tamaño A4 escaneados a 300 puntos por pulgada al igual que de tamaño A0 (talla E) escaneados a resoluciones mayores.*

** Importa: TIFF, BMP.*

** Exporta: DXF, BMP.*

Otros formatos hay que pedirlos "

- **Formatos de ficheros admitidos**:

Entrada: BMP y TIFF. También se puede cargar un prototipo DXF.

Salida: BMP y TIFF para el raster y DXF para los vectores.

- **Tratamiento previo de la imagen**:

a) Edición: Posee una herramienta **lápiz** y una **goma** de borrar, además del **zoom** y movimientos de **rotación** (90 grados a derecha e izquierda).

b) Preprocesado: Tiene diversas herramientas de limpieza: un **filtro**, una operación **closing** (una dilatación seguida de una erosión), un **thinning** (operación que busca un esqueleto de la imagen con líneas de grosor uniforme), una operación que borra el interior de un área rellena (cuyo diámetro mínimo se puede variar) y también resalta el **contorno** de la imagen con un grosor de línea elegido entre 4 posibles. El área de trabajo se puede modificar en la opción **área**, que posee también un **filtro** para los vectores (eliminando todos aquellos que estén por debajo de un valor umbral) y una **tolerancia** del ajuste de los vectores a las líneas de la imagen raster (ambas medidas en una escala de 0 a 100). La resolución del raster puede ajustarse (medida en DPI o puntos por pulgada).

Opciones de vectorización:

En este programa no existen opciones de vectorización propiamente dichas, pero en cambio proporciona algunos parámetros para controlar la misma: se pueden alinear las líneas (con un ángulo que se puede variar), crear una capa para cada grosor de línea (se puede cambiar el número de capas), reconocer líneas punteadas (con rayas, puntos o ambos), almacenar las polilíneas por separado (con opción de hacerlo sólo con las cerradas), saltar las posibles discontinuidades en una línea vectorizada (indicándole los pixeles que debe tener como mínimo dicha discontinuidad) y por último reconocer pequeños círculos (el diámetro de los mismos se puede fijar en milímetros).

Además hay un editor de capas donde se indica el nombre de cada una, su estado (activa o no), su color y su tipo (línea, polilínea, texto, etc.). La capa 0 es todo el dibujo, la L1 es la de las líneas, la P1 la de Polilíneas y la T1 la del texto. Se pueden ver u ocultar el raster, los vectores, el texto, las áreas y los puntos inicial y final de cada vector, así como las intersecciones de vectores (marcadas en rojo). También posee un OCR (reconocedor óptico de caracteres) al que se le puede poner el ángulo del texto.

Hay que decir que ScanVector reconoce las principales entidades (círculos, líneas, arcos, etc.) pero no es capaz de dibujar ninguna de ellas.

- **Valoración**:

Podemos concluir que ScanVector es quizás el que menor flexibilidad permite a la hora de vectorizar, dado que no hay opciones para la vectorización. Tampoco permite dibujar sobre el raster las principales entidades (círculos, curvas, etc.), lo cual lo deja muy limitado. Este producto es uno de los menos flexibles del estudio realizado.

1.7.2.7 STREAMLINE 4.0.

-**Versión**: STREAMLINE 4.0 Demo.

Al tratarse de una versión demo existen restricciones tales como la inhabilitación de los comandos *save, drag & drop* y *cut & paste.*

-**Empresa fabricante**: Adobe Systems Inc.

- Requerimientos:

Sistema operativo: Windows 95/Windows NT® 4.0

Sistema: Intel® i486(TM)

Memoria: 16 MB

Espacio en disco duro: 20 MB

Lector de CD-ROM.

Para un funcionamiento óptimo se recomienda: Pentium o superior, 32 MB o más de RAM, tarjeta de vídeo Super-VGA de 24 bits o superior y alguna aplicación que soporte la impresión de ficheros EPS.

-**Publicidad en la red**:

"*Nuevas y mejoradas características y herramientas hacen que el Streamline sea más poderoso que nunca en convertir mapas de bits y refinar los vectores resultantes al tiempo que mantiene la más alta integridad en tonos de color y en trayectorias.*

La unión de Streamline 4.0 con otros productos de Adobe como Pagemaker, Illustrator, etc. es perfecta, y permite la inserción de un documento con sólo

seleccionarlo con el ratón y llevarlo al otro programa durante el proceso de producción.

La amplia compatibilidad del producto permite elegir entre multitud de formatos para exportar a otros programas de dibujo, gráficos y edición de páginas. Deje que Streamline elimine el tedio de las tareas de conversión y edición de imágenes y le ayude a conseguir el producto final de más alta calidad que hubiera imaginado.

Convertir con control sin igual. *No más trazado manual para obtener el nivel deseado de control al convertir imágenes a vectores. Ahora, Adobe Streamline ofrece el máximo control sobre el proceso de trazado. Y las nuevas características de color en Streamline 4.0 potencian el control creativo, permitiéndole contemplar la imagen que representó, no una versión con los colores cambiados.*

** Escanee imágenes y bocetos en Streamline, o importe vectores, fotos en color u otros ficheros vectoriales.*

** Empiece por usar los parámetros predeterminados para mayor facilidad de uso; después experimente con parámetros de conversión personalizados como línea exterior, línea centrada, reconocimiento de líneas, etc... Grábelos para reutilizarlos, y así minimizar el trabajo para una posterior conversión..*

** Convierta sin huecos, trazados solapados u otras inexactitudes. Streamline ajusta el alineado incorrecto de imágenes escaneadas y endereza líneas horizontales y verticales.*

** Deje que Streamline seleccione los colores apropiados hasta un número ilimitado, o especifique una lista de colores personalizada.*

** Simule dos, tres o cuatro tonos con tintas de color personalizadas.*

** Designe parámetros específicos distintos para diferentes áreas de la misma imagen.*

** Si tiene muchas imágenes para convertir, como un conjunto de logotipos, puede procesar todos los ficheros en un lote.*

Edición con herramientas de postconversión flexibles. *Refine las imágenes convertidas para ajustarse a sus necesidades con las potentes herramientas y características de edición del software Streamline 4.0. Las herramientas de edición en Streamline están hechas especialmente para permitirle refinar rápidamente su dibujo convertido. Después de la conversión, puede exportar los vectores resultantes a su*

programa de dibujo favorito, y redimensionarlo a cualquier tamaño sin perder claridad en la imagen.

Herramientas de edición de vectores

** Use herramientas de dibujo familiar para editar y corregir rellenos, trazos y anchuras de línea.*

** Seleccione partes de una imagen para convertirlas o modificarlas.*

**Asigne colores personalizados a toda la imagen o un área seleccionada y cámbielos fácilmente a otro proceso o color personalizado.*

** Suavice los puntos de dirección para un trazado de líneas más preciso.*

** Añada, borre o mueva puntos.*

** Convierta las líneas rectas en curvas o viceversa.*

** Convierta los trazados en formas geométricas.*

** Haga los trazados más suaves.*

**Pase de la imagen raster original al gráfico vectorial resultante al tiempo que aplica los refinamientos.*

Un nuevo nivel de integración. *Nunca antes la integración entre productos de Adobe había sido tan compacta, facilitando el proceso de producción. Incremente su valía como profesional creativo con la plataforma de expertos en Streamline 4.0; se han hecho versiones idénticas en Macintosh y Windows para aprender a usar ambos fácilmente.*

** Importe y exporte elementos de y hacia Adobe Photoshop, Pagemaker, Illustrator y otros productos de Adobe.*

** Use las herramientas, menús y teclas rápidas que usted ya conoce de otras aplicaciones de Adobe.*

** Sea diestro entre plataformas con el interfaz de paridad entre las versiones Macintosh y Windows del producto.*

** Importe imágenes de mapas de bits en formato TIFF, PSD, PNT y PCX.*

** Exporte los gráficos vectoriales en formato Adobe Illustrator, DXF, EPS y EMF.***”**

-**Formatos de ficheros admitidos**:

Entrada: TIFF, PCX, PNT y PSD (Photoshop 3.0).

Salida: Adobe Illustrator 3, Adobe Illustrator EPS, DXF, PICT y WMF para imágenes ya vectorizadas y Adobe Photoshop 2.0, Adobe Photoshop, PICT (Macintosh), PCX y TIFF para imágenes rasterizadas.

-**Tratamiento previo de la imagen**:

a) Edición: Posee las típicas herramientas de **zoom** (zoom in, zoom out, ajuste de ventana y escala natural), **selección** (por colores, de una zona rectangular, una poligonal, toda la imagen o parte de ella, invertir lo seleccionado...), herramientas de **dibujo** sobre el raster (líneas rectas, trazos y borrado), de **ajuste** del dibujo (brillo, contraste y niveles gamma) y herramientas para trabajar con imágenes en color (selector de colores).

b) Preprocesado: No existe herramienta alguna de limpieza ni de preparación previa de la imagen. Las únicas herramientas que se pueden considerar de limpieza serían el *noise supressing* (ver apartado "gestión de parámetros") y el borrado a mano con la herramienta de borrado de la *toolbox* de la imagen raster).

-**Opciones de vectorización**:

1) Centerline: se elige el centro de las líneas para vectorizar la imagen. Apropiado para dibujos técnicos.

2) Outline: convierte los contornos de las estructuras raster. Apropiado para fotografías y dibujos artísticos y hechos a mano.

3) Line Recognition: sólo reconoce y convierte las líneas rectas que se desvían como mucho 5º de los ejes horizontal y vertical.

4) Centerline/outline: método intermedio entre 1 y 2. Resulta útil cuando existen en un mismo dibujo áreas rellenas y líneas de grosor homogéneo. El valor que se le da -llamado *line thinning*- es el valor umbral en pixeles a partir del cual se vectoriza una línea utilizando su contorno (vectorización *outline*) y no su esqueleto (vectorización *centerline*).

Centerline/Line Recognition: En primer lugar se convierten las líneas horizontales y verticales y en segundo lugar se convierten las curvas.

Existe también la posibilidad de convertir varias imágenes en modo batch.

- Gestión de parámetros:

Existen seis parámetros de vectorización llamados “opciones generales de conversión”:

1) Supresión de ruido: Está habilitado sólo con la opción de vectorización *outline*. El valor determina el número de pixeles a partir del cual un área se ignorará una vez vectorizada la imagen raster (se puede considerar como una herramienta de limpieza “a posteriori”).

2) Adelgazamiento de líneas: Este parámetro modifica el modo en que el Streamline calcula el centro de las líneas. A valores más pequeños, mayor precisión. Para comprobar si se ha “hilado demasiado fino”, el Streamline dibuja un círculo en los puntos de unión de las líneas vectorizadas. Si hay círculos que no se corresponden con uniones en la imagen raster, el valor de dicho parámetro se deberá aumentar. Activando el *interactive thinning*, se ofrece la posibilidad de interrumpir la vectorización, cambiar el parámetro de adelgazamiento de líneas y continuar por donde se había detenido.

3) Tolerancia: Sirve para determinar cómo se ajustarán las líneas resultantes de la vectorización a las rectas y curvas de la imagen raster. Si se le da un valor pequeño se producirá un mejor ajuste pero aparecerán más segmentos y más puntos de unión.

4) Líneas blancas: Sólo es útil en caso de que la imagen raster tenga el fondo negro.

Opciones de rastreo: Existe la posibilidad de indicarle que detecte exclusivamente líneas rectas, líneas curvas o que detecte ambos elementos. En este último caso, se indica el radio de curvatura de las líneas en la imagen vectorial (un valor bajo hará que el Streamline tienda a dibujar las líneas de la imagen raster como rectas y un valor alto hará que se dibujen como curvas). También se puede modificar el grosor de todas las líneas de la imagen vectorial con la opción *uniform line weight*.

Inversión de imagen: Parámetro para invertir la imagen antes de su conversión.

Separado de formas: Disponible sólo con la opción *centerline*. Permite trabajar y editar por separado áreas poligonales de una imagen raster. Útil para fines artísticos.

Por último significar que existe la posibilidad de grabar, modificar y borrar diferentes conjuntos de parámetros y que una misma imagen se puede dividir en distintas áreas y darle a cada área un conjunto de parámetros diferente.

- Valoración:

Este vectorizador, debido a sus características (los parámetros de vectorización, la escasez de herramientas de limpieza y la amplia gama de herramientas de edición y de dibujo en color) hacen de él un programa orientado más hacia el diseño gráfico y artístico que hacia el tratamiento de planos de ingeniería.

1.7.3 Vectorizadores en versión completa.

1.7.3.1 COREL OCR-TRACE

-Empresa fabricante: Corel Corporation.

-Versión: Corel OCR-TRACE 7.

-Requerimientos:

Sistema operativo: Windows 95 o Windows NT 4.0
Sistema: 486 DX 100 MHz.
Memoria: 16 MB RAM.
Tarjeta gráfica: VGA
Espacio en disco duro: 40 MB
Lector de CD-ROM
Recomendado: Pentium 120 MHz, 32 MB RAM, monitor SVGA.

-Formatos de ficheros admitidos:

Entrada: JPEG, BMP, SCT, TGA, GIF, ICO, CUR, IMG, PPD, PS4, PCD, MAC, CALS, CPT, TIFF, PCX y EPS.

Salida: Permite guardar la imagen raster, el texto y los vectores en ficheros separados con muy diversos formatos. En los formatos de salida de imagen tenemos prácticamente los mismos que de entrada. En cuanto al texto, permite guardarlo en ficheros con extensión DOC (de varias versiones de Word), WP4, WP5, WP6 (todas ellas de WordPerfect), WSD, XY, SAM, TXT y RTF. Por último, los vectores se pueden grabar como DXF, WMF, EMF, CMX (propio de CorelDraw), AI (Adobe Ilustrator), WPG (WordPerfect) y GEM.

-Tratamiento previo de la imagen:

a) Edición: Dispone de opciones de **zoom, dibujo** y **borrado**, **eliminar nodos**, crear **curvas Bezier**, **selección** de áreas, etc. Hay que decir que las modificaciones hechas se efectúan en una ventana distinta a la que contiene la imagen original, pudiendo ver así su aspecto después de cada cambio. También posee opciones para **invertir** la imagen, **reflejarla** horizontal o verticalmente, **convertirla** a blanco y negro y **rotarla** de las siguientes formas: 90º a la derecha, 90º a la izquierda, 180º o bien con parámetros personalizados como los grados a rotar y el sentido del giro (horario o antihorario).

b) Preprocesado: No posee ninguna herramienta de limpieza de la imagen.

- Opciones de vectorización:

Dispone de 6 opciones distintas para vectorizar, todas ellas con parámetros propios, siendo cada una adecuada para un campo de trabajo. Las opciones son:

a) Por **contorno**. Este método ofrece una imagen vectorial parecida al raster original, y es bueno si se quiere mantener el aspecto inicial de la imagen.

Los parámetros que tiene son fundamentalmente dos:

- Relleno de huecos. Le indica al programa el tamaño mínimo del hueco que se debe rellenar. Se puede seleccionar este tamaño en una escala graduada de 0 a 100.

- Tolerancia al color. Indica lo distintos que pueden ser los colores en el raster original para convertirse en el mismo color en la imagen vectorial. Cuanta menos tolerancia, más colores aparecen en la imagen final, y también más capas.

b) Por **línea centrada**. Esta opción convierte un mapa de bits en un dibujo de líneas, con la posibilidad de especificar los grosores de las mismas. Es adecuado para planos de ingeniería, por lo que sería la opción más interesante para nuestro análisis.

Los parámetros principales son dos:

- Reducción de ruido. Se mide en pixeles y elimina todas las zonas de color que son menores que este valor.
-. Iteraciones. Número de veces que reduce las líneas en la imagen para encontrar el centro de la línea.

c) Por **grabado en madera**. Se utiliza para objetos que poseen elementos de diferentes anchuras. Sólo existe un parámetro relevante, el umbral. Puede ir de claro a oscuro y permite dividir la imagen en distintas áreas según la intensidad de su color, obteniéndose una línea de contorno vectorizada.

d) Por **esbozo**. Crea el llamado efecto malla, por medio de capas distintas de líneas. Resulta apropiado para crear efectos especiales en el dibujo.

Sus parámetros son dos:

- Interlineado. Permite elegir un punto de corte del valor de la intensidad para determinar qué partes de la imagen se van a convertir.
- Umbral. Es igual al punto anterior, pero esta vez permite un valor de umbral diferente para cada capa.

e) Por **mosaico**. Convierte la imagen en una matriz de objetos simétricos, y cuantos más mosaicos haya, más se parecerá el modelo a la imagen original.

Los parámetros más relevantes son:

-Forma. Determina el tipo de objetos para el modelo de vectorización, y pueden ser rectángulos, círculos o rombos.
-Número de mosaicos horizontales. Permite escoger cuántos mosaicos horizontales se van a coger, y puede ir de 1 a 500.
-Número de mosaicos verticales. Análogo al anterior, pero con mosaicos verticales.

f) Por **mosaico 3D**. Transforma la imagen en un modelo de objetos simétricos 3D, y añade un efecto de textura a la imagen. Sus parámetros son los que la opción mosaico, pero las formas de los objetos que compondrán el mosaico pueden ser pirámides, ladrillos y abanicos.

Además de la vectorización normal, se permite hacer una **vectorización-OCR**, que separa en dos ventanas el texto y los vectores obtenidos. También se puede extraer únicamente los caracteres mediante el OCR sin realizar la vectorización del resto del dibujo. Tiene un gestor de capas que permite editar cada una de ellas y ver información de las mismas. También hay un gestor del color, cuya paleta permite usar multitud de paletas de color diferentes (Toyo, Dic, Pantone, etc ...) o bien personalizados por el usuario.

- **Valoración**:

Este programa es de los que más se aparta de los objetivos de nuestro estudio, y parecer estar más encaminado a diseño gráfico y artístico, dado el gran número de opciones de vectorización con texturas y efectos visuales. No es el más recomendable para nuestros propósitos, pero resulta bastante bueno en otros campos.

1.7.3.2 MICROSTATION REPROGRAPHICS

-**Empresa fabricante**: Bentley Systems Inc.

-**Versión**: MICROSTATION REPROGRAPHICS Academic Version.

-**Requerimientos**:

Sistema operativo: DOS 3.1, Microsoft Windows 3.1x, 95 y NT.

Software requerido: MicroStation 95 o MicroStation PowerDraft.

Sistema: 80486 o Pentium Pro o DEC Alpha.

Memoria: 8MB mínimo RAM; 16MB recomendados (DOS)

16MB mínimo RAM; 24MB recomendados (Intel – Windows)

Espacio en disco duro: 200MB (instalación típica: 60 MB incluyendo Microstation o 30 MB incluyendo PowerDraft).

Se recomienda usar una tarjeta gráfica high-color o true-color y soporta el uso de una tableta gráfica así como el uso de dos monitores.

-**Diferentes formatos de ficheros admitidos**:

Nombre del formato	Extensión del fichero	Tipo de pixel	Tipo de compresión
HMR	HMR	2COLORES	NINGUNO PACKBITS CCITT3 CCITT4
HMR	HMR	256COLORES	NINGUNO DEFLATE
HMR	HMR	GRAYSCALE	NINGUNO DEFLATE
Img (24 bit)	A	24BITCOLOR	POR DEFECTO
Img File	P	16COLORES 256COLORES GRAYSCALE	POR DEFECTO
Intergraph CIT	CIT	2COLORES	CCITT4
Intergraph COT	COT	256 Colores	NINGUNO
Intergraph COT	COT	GRAYSCALE	NINGUNO
Intergraph RGB	RGB	24BITCOLOR	POR DEFECTO
Intergraph RLE	RLE	2COLORES	RLE
JPEG	JPG	24BITCOLOR GRAYSCALE	JPEG
PCX	PCX	16COLORES 256COLORES GRAYSCALE	POR DEFECTO
SUN Raster file	RS	256COLORES GRAYSCALE	POR DEFECTO
TARGA	TGA	24BITCOLOR	POR DEFECTO
TIF	TIF	24BITCOLOR	NINGUNO DEFLATE
TIF	TIF	256COLORES	NINGUNO DEFLATE
TIF	TIF	GRAYSCALE	NINGUNO DEFLATE
TIF	TIF	2COLORES	NINGUNO PACKBITS CCITT3 CCITT4
Windows BMP	BMP	16COLORES 256COLORES 24BITCOLOR GRAYSCALE	POR DEFECTO

Tipo de fichero	Extensión
Fichero de proyecto	PRJ
Fichero de imágenes	HMR, CIT, COT, RGB, RLE, A, P, JPG, PCX, RS, TGA, TIF, BMP
Fichero de tabla de colores	TBL
Fichero de filtros de color	FTR
Fichero de tema	THM
Fichero de rayado	HAT
Fichero de patrón	PAT
Fichero de nodos	NOD
Fichero de registro	RGR
Fichero de muestra	RSP

-Tratamiento previo de la imagen:

a) Herramientas de tratamiento del color: **histograma** (estadísticas sobre el color de una imagen), **canales RGB**, **editor de la tabla de colores**, **contraste** (ajuste a través del histograma), **densidad del color** (ajuste a través del histograma), **compresión del color** (descompone la imagen en tres canales, uno para cada canal RGB), **transparencia** y **translucidez**. También se pueden usar **filtros** de color que permiten tratar una imagen en color como si fuera binaria, es decir que solo permitirá modificar los pixeles del color del filtro tratando a los demás como si fueran colores de fondo.

b) Herramientas de edición del raster: permiten **limpiar** pequeñas manchas, **borrar** áreas de la imagen con diversas herramientas, **rellenar** pequeños huecos, **copiar** regiones de la imagen, **pegar** regiones en otras imágenes o como elementos de diseño, dibujar en el raster usando las herramientas vectoriales de MicroStation, **estampar vectores** en el raster y **dibujar** en el raster con pinceles de distinta forma y tamaño.

c) Herramientas de transformación: algunas de estas herramientas tienen versiones para aplicarlas a la imagen completa o tan solo a un área de ésta y permiten **mover**, **escalar**, **rotar**, **reflejar** (horizontal y verticalmente) y **deformar** la imagen o el área elegida. Aparte de estas herramientas hay otras que únicamente se pueden aplicar a la imagen completa: **alinearla** horizontal o verticalmente, **unir** en una sola imagen rectangular varias imágenes raster creando un nuevo fichero y **unirlas en una imagen** de forma **poligonal** (el polígono de destino puede estar rotado, escalado o desplazado) pudiendo incluir elementos vectoriales.

-Opciones de vectorización:

Este producto se diferencia de los demás vectorizadores de esta comparativa en que el proceso de vectorizado no tiene un nivel de automatización tan elevado como los demás ya que requiere de mucha más interacción por parte del usuario. A este proceso de vectorización semiautomático se le conoce como “head-up vectorizing” ya que el usuario trabaja mirando a la pantalla con la cabeza levantada en lugar de estar mirando

al tablero digitalizador con la cabeza agachada. En realidad el proceso seguido por ReproGraphics consiste en seguir las líneas de las imágenes raster que se han cargado en el área de trabajo de modo similar a como lo haríamos si estuviéramos calcando estas imágenes sobre un papel transparente.

ReproGraphics proporciona herramientas que permiten seguir las líneas del raster de diferentes formas:

Dibujando manualmente sobre ellas: Reprographics proporciona la posibilidad de ajustar los puntos de definición de las entidades vectoriales al raster automáticamente (al centro de la línea o al borde más próximo). Para facilitar el proceso de indicación de puntos disponemos de una herramienta especial que amplía a modo de lupa la zona próxima al cursor y que nos permite marcar los puntos dentro de ella para obtener mayor precisión.

Dibujando automáticamente sobre ellas: Bien por su centro o bien por una frontera (recordemos que podemos trabajar con colores) aunque eso sí, el usuario debe indicar el punto de la línea a partir del cual comenzará la vectorización. Hay dos herramientas de estas características: la más básica de ellas sigue la línea en ambas direcciones a partir del punto inicial hasta encontrar una bifurcación, un nodo de control, una línea ya vectorizada o el final de dicha línea y la segunda de ellas se comporta del mismo modo, pero en caso de encontrar una bifurcación seguirá vectorizando todas las ramas que partan de la misma, hasta alcanzar el final de todas las líneas o líneas previamente vectorizadas o hasta alcanzar un nodo de control. Los nodos de control se pueden disponer manualmente sobre el raster o bien con una herramienta que los sitúa automáticamente, y su función consiste en controlar el proceso de vectorización. La primera de estas herramientas permite la posibilidad de seguir vectorizando al alcanzar una bifurcación sólo con seleccionar la rama por la que debe seguir. También podemos asignar un tema a la entidad que estamos vectorizando. Un tema es una serie de atributos que se asocian a una entidad vectorial. Estos atributos son: grosor de línea, estilo, patrón, sombreado, color y nombre. Los temas permiten agrupar entidades del mismo tipo. Los textos tienen atributos de tema específicos para ellos: fuente, anchura, altura, interlineado, longitud de línea, espaciado, sangrado, ángulo, subrayado e inclinación.

El tratamiento que hace ReproGraphics de los textos y los símbolos sigue la misma línea que el resto del programa, es decir, que disponemos de diversas herramientas que hacen más cómodo el proceso de conversión manual de textos. Para convertir un texto del raster lo que se debe hacer es seleccionarlo con la herramienta adecuada (dependiendo de si es horizontal o tiene alguna orientación) y una vez obtenido el tamaño al que se debe ajustar el texto, se puede seleccionar un tema adecuado para dicho texto para simplemente después teclearlo y que aparezca en la capa vectorial en la misma posición que ocupaba en el raster. El proceso seguido en el caso de querer introducir cualquier símbolo como una entidad dentro de la capa vectorial es muy similar.

- Particularidades:

MicroStation ReproGraphics™ de Bentley Systems, Inc. necesita para su funcionamiento ser instalado en un equipo en el que previamente se encuentre instalado MicroStation o MicroStation PowerDraft. Para acceder a las nuevas funciones que MicroStation ReproGraphics ofrece debe ser cargado desde uno de los programas anteriormente citados cada vez que se inicie una sesión de trabajo.

La interfaz de ReproGraphics sigue los convenios de MicroStation para mostrar las imágenes en las vistas y para editar archivos de diseño, es decir, esencialmente lo que añade son nuevas cajas de herramientas que el usuario puede organizar a su gusto en la pantalla de MicroStation del mismo modo que si fueran las cajas de herramientas estándar de que éste dispone.

La mecánica de trabajo de ReproGraphics consiste en fijar una o varias imágenes raster a las vistas, y en cada una de estas vistas habrá varias capas que contendrán imágenes o diseños, es decir, imágenes raster o diseños vectoriales. En una misma vista se pueden cargar varias imágenes a las cuales se les puede fijar individualmente la posición dentro de la vista. Las imágenes mostradas dentro de una vista se disponen a modo de pila que crece hacia el fondo de la misma de modo que las imágenes se pueden solapar ocultándose unas a otras. Las imágenes pueden ser añadidas a una o varias vistas al ser abiertas, además las imágenes de una de ellas pueden encenderse o apagarse mostrándose en pantalla o no.

Las imágenes tienen dos propiedades que se pueden activar y desactivar: la transparencia y la translucidez. La transparencia es una propiedad que se puede aplicar a

un conjunto de colores de la imagen por la cual dichos colores no aparecerán en pantalla y en caso de haber otra imagen debajo de los colores transparentes ésta se podría ver. La translucidez funciona de modo similar sólo que a los colores del conjunto se les aplica un porcentaje de translucidez con lo que no son totalmente transparentes.

-Valoración:

MicroStation ReproGraphics es un buen ejemplo de lo que se conoce como “heads-up vectorizing”, es decir que el usuario debe seguir las líneas del raster que aparece como fondo de la pantalla con la ayuda de todas las herramientas que le proporciona el programa.

Teniendo esto en cuenta, la calidad del resultado final de la vectorización depende en gran medida del operador (experiencia, interés por el trabajo, ambiente, etc...) por ello es de vital importancia en este tipo de vectorización elegir un buen programa que permita conseguir la mayor productividad del operador. ReproGraphics ofrece un amplio conjunto de herramientas con la misma interfaz para el usuario que MicroStation por lo que para los usuarios que estén acostumbrados a trabajar con MicroStation no resultará difícil acostumbrarse a trabajar con ReproGraphics. Por otra parte, ReproGraphics ofrece tantas herramientas y opciones que no resulta sencillo conocerlas todas con profundidad. Por suerte, la documentación que lo acompaña es de bastante buena calidad. Entre esta documentación existe un tutorial que paso a paso nos permite conocer las herramientas y opciones básicas de que dispone ReproGraphics, pero para trabajar con detalle es necesario consultar la guía del usuario.

ReproGraphics resulta adecuado para trabajar con imágenes en color procedentes de satélites o fotografías aéreas para su posterior tratamiento en sistemas de GIS y cartografía debido a que, entre otras facilidades, tiene capacidad de filtrar colores y componer mosaicos de imágenes. Para trabajar con dibujos de ingeniería del estilo de los nuestros, resultan más adecuados otros programas que utilicen métodos de vectorización automáticos, ya que permiten centrar el esfuerzo del operador en el preproceso de la imagen y la posterior corrección de fallos en lugar de en el proceso de vectorización en sí.

1.8 PROYECTOS FIN DE CARRERA.

-**Origen**: PFCs de la Facultad de Informática de la UPV de Vicente Escuderos Abella y de Mª Carmen Juan Lizandra. Cabe destacar el carácter complementario de ambos trabajos. Mientras que el proyecto de Vicente implementa una serie de técnicas de preprocesado de ficheros raster y una primera aproximación a la identificación de entidades, el proyecto de Mª Carmen se encarga de realizar una primera discriminación entre texto y gráficos pero no proporciona reconocimiento de caracteres.

-**Diferentes formatos de ficheros admitidos**:

Entrada: TIFF.

Salida: DXF.

-**Tratamiento previo de la imagen**:

a) Edición: No se ha implementado en este trabajo ningún módulo de edición.

b) Limpieza: Se dispone de las siguientes herramientas:

- Erosión: El propósito de la erosión es eliminar aquellos pixeles que no deberían encontrarse allí, como por ejemplo las pequeñas agrupaciones de pixeles formadas por las manchas del papel.

- Dilatación: Es complementaria a la erosión. Esta operación provoca que se añada una capa de pixeles periférica a todos los objetos de la imagen, con lo que se aumenta las dimensiones de los mismos.

- Closing: Consiste en una dilatación seguida de una erosión. Se llama closing en referencia a la capacidad de unir objetos que se encuentran muy próximos, y además, de rellenar pequeños huecos en los objetos.

- Opening: Es la combinación de una erosión seguida de una dilatación. El nombre tiene su origen en la capacidad de abrir espacios entre objetos que se encuentran tocándose. Es una de las operaciones más usadas en el proceso de eliminación de ruido en imágenes binarias.

- **Opciones de vectorización**:

No dispone de ninguna opción al vectorizar, siguiendo siempre la misma secuencia de pasos:

a) Preprocesado. El ruido no afecta al buen funcionamiento de los algoritmos, pero provoca que la calidad del esqueleto resultante no sea la esperada. Los errores más

comunes que se producen a consecuencia del ruido son las líneas en zig-zag, pequeñas líneas con un extremo inconexo denominadas “hairs” y algún pequeño fallo de conexión. Esta herramienta pretende corregir todos estos fallos.

b) Thinning y detección de grosores. En este módulo a partir de la imagen raster se extrae un esqueleto de la imagen binaria que se almacenará en forma de dos ficheros, uno con los nodos correspondientes a los puntos de intersección etiquetados y otro con las líneas, formado por sus etiquetas, los vectores que las aproximan y el grosor detectado durante esta operación.

c) Conversión. Consiste en generar un grafo a partir de los dos ficheros producidos en la fase anterior y una vez construido dicho grafo, refinarlo para corregir pequeños defectos. El grafo se almacenará en forma de dos tablas hash que constituirán la entrada de la siguiente fase.

d) Reconocimiento. Después de una primera clasificación de las primitivas en rectas, arcos y círculos, se someterá a cada una de ellas a una serie de sucesivos tests de identificación (test de recta, test de arco, test de circunferencia, test arco-arco, test recta-arco y test arco-recta) con el fin de comprobar la correcta clasificación de la entidad.

-**Particularidades**: Debido al carácter de PFC de este vectorizador, este trabajo no puede considerarse un vectorizador propiamente dicho y por tanto no se puede comparar con los productos comerciales desde el momento en que es sólo una primera aproximación. Por otro lado, teniendo esto en cuenta y una vez analizados los demás vectorizadores, hay que destacar que es una aproximación nada desdeñable y que merece ser tenida en cuenta.

1.9 CONCLUSIONES.

Aunque por evidentes razones comerciales la mayoría de los vectorizadores son considerados por sus fabricantes como productos de uso general, e intentan incluir opciones para permitir su uso en diversos campos para ampliar el mercado de dicho producto, una vez analizados es sencillo darse cuenta de que cada uno de ellos, debido a las características que incorpora, resulta más apropiado para una disciplina en particular (GIS y cartografía, CAD, diseño gráfico, ...).

El tipo de imágenes con que se trabaja en cada campo es muy diferente: en diseño gráfico el color resulta esencial, por lo que las imágenes con las que se deberá trabajar en la mayoría de los casos tendrán multitud de colores; en GIS y cartografía es habitual trabajar con imágenes aéreas o fotografías de satélite que suelen escanearse en color o en escala de grises para no perder información; toda la información presente en un dibujo de ingeniería puede estar contenida en una imagen binaria, teniendo el color mucha menos importancia que en los demás casos.

También existen diferencias en cuanto al modo en que está representada la información en estas imágenes: por ejemplo, en dibujos de ingeniería son importantes aspectos tales como el grosor de línea, las relaciones entre ellas (perpendicularidad, paralelismo, etc.) y otros que deberán ser tratados adecuadamente por el vectorizador para no perder información. Otro ejemplo sería la importancia que pueden tener las deformaciones producidas durante el escaneado de imágenes para GIS y cartografía, las cuales deberán ser corregidas para evitar imprecisiones al unir varias de ellas en un mosaico las cuales darían lugar a un mapa inexacto. Debido a todas estas diferencias, las herramientas y métodos que se exigen para afrontarlas también diferirán mucho.

Para darse cuenta de lo anterior basta con comparar la problemática que conlleva seguir un contorno de una entidad en una imagen en color debido a las diversas tonalidades que aparecen en dicho contorno frente a la mayor simplicidad que supone seguir un contorno en una imagen binaria. A causa de esta diversidad de herramientas es difícil que un mismo vectorizador las aúne a todas en un mismo paquete ya que debería contener tantas opciones que su manejo resultaría excesivamente complicado. Además, un producto de estas características requeriría un amplio esfuerzo para su desarrollo que no sería ventajoso para nadie porque para un mismo usuario no serían de interés las herramientas destinadas a otros campos distintos al suyo, a no ser que el usuario se dedicara a vectorizar imágenes de terceros. En este último caso podía ser más conveniente tener varios vectorizadores especializados y sencillos de usar, que un programa monolítico imposible de usar debido a su complicación.

Los distintos vectorizadores que hemos estudiado en nuestro análisis se podrían clasificar según el campo al que están enfocados en:

a) Artísticos. Este grupo permite trabajar con imágenes en color, ya que el color es una parte importante de los diseños artísticos. La información vectorizada consiste en

meras líneas que aproximan la forma que aparece en el raster. Además no incorporan soporte para la información textual. Dentro de este tipo estarían productos como Streamline (de la empresa Adobe) y OCR-Trace (de Corel).

b) Orientados a GIS (Geographical Information System). Este grupo también posibilita trabajar con gráficos en color, pero por otros motivos. Los colores que aparecen en los mapas proporcionan información adicional que no puede perderse durante el proceso de vectorización. Además, las imágenes fuente con las que estos vectorizadores se ven forzados a trabajar incluyen fotos aéreas e imágenes de satélite, en las cuales es fundamental el trabajar con colores. Además de estas capacidades pueden incluir facilidades para el manejo de información de geo-referencia y otras características para facilitar la comunicación con programas de GIS. Deben soportar la información textual de algún modo (para trabajar con topónimos, alturas de líneas de cota, etc.), bien mediante un OCR o facilitando de algún modo el manejo de esta. Los vectores que producen llevan asociada información adicional en forma de atributos de estos. A este segundo grupo pertenecerían productos como MicroStation ReproGraphics (de Bentley) y R2V (de Able Software).

c) Orientados a CAD/CAM. Estos no dan tanta importancia al color (ya que los planos suelen venir en blanco y negro), pero algunos permiten tratar con imágenes coloreadas (más que nada en colores planos). Aquí es importante el reconocimiento de entidades de dibujo, de texto y la separación entre ambos. De este último grupo se pueden citar el VPStudio (de Softelec), I/Vector (de Ideal Systems), el ScanVector (de Cybersonics) y el TracTrix (de Trixsystems).

Esta clasificación no pretende encerrar a los vectorizadores en grupos independientes entre los que no hay relación alguna, ya que la mayoría de los vectorizadores incorporan características que pueden corresponder a un grupo distinto al que ellos pertenecen, pero generalmente todos tienen una tendencia global que los hace más adecuados para una u otra tarea. Por ejemplo hay un vectorizador que por sus características no se podría incluir en ninguno de estos 3 grupos debido a lo indicado anteriormente: el PixEdit (de la empresa Techsoft). En principio parece encaminado a trabajos técnicos de CAD, pero su vectorización es muy pobre comparada con algunos de los anteriores y más que nada parece un complemento a sus herramientas de edición raster para facilitar la comunicación con programas de CAD. Debido a todo esto estaría a caballo entre el grupo a y el c.

Ya que el objeto de este análisis es centrarse en aquellos programas orientados a CAD, es conveniente establecer una clasificación comparativa entre los vectorizadores pertenecientes a este último grupo (TracTrix de Trix Systems, VPStudio de Softelec, I/Vector de Ideal y Scanvector de Cybersonic) a fin de elegir el programa que más se aproxime a nuestros objetivos. Como se puede observar en la siguiente tabla, y comparando los valores de los campos más significativos (filtros de limpieza, herramientas de edición, reconocimiento de entidades y discriminación de textos), el ScanVector y el TracTrix ofrecen menos prestaciones (no reconocen entidades de orden superior como puntas de flecha o áreas rayadas) que el I/Vector y el VPStudio. Además, las posibilidades de edición de los dos primeros son muy inferiores comparadas con los productos de Ideal y Softelec.

Por otra parte, y centrándonos ahora en los dos productos más interesantes (I/Vector y VPStudio), podemos ver que en el campo “herramientas de edición vectorial” el VPStudio es mejor, puesto que tiene la posibilidad de editar y añadir características de las entidades (coordenadas, el centro de los círculos y arcos, el texto, ...) pudiéndose añadir también nuevas entidades. Todas estas posibilidades ofrecidas por el VPStudio están disponibles, bien dibujando sobre la pantalla o para mayor exactitud a través de cuadros de diálogo. Además el VPStudio resulta más sencillo de usar ya que I/Vector está dividido en tres editores (del raster, de vectores y de textos) en los cuales hay en entrar y salir sucesivamente para procesar la imagen por lo que el trabajo resulta más incómodo. Así pues, finalmente el vectorizador que parece adaptarse a nuestros propósitos con más exactitud es el VPStudio.

Nombre y empresa	Trabajo en batch	Orientación	Precio	Filtros de limpieza[i]	Nivel de las herramientas de edición del raster[ii]	Herramientas de edición vectorial	Reconocimiento de entidades[iii]	Discriminación de textos	Múltiples imágenes[7]	Formatos de imágenes disponibles	Método de vectorización
TRACTRIX FOR CAD RELEASE 2.0 DEMO VERSION. Trix Systems Inc.	NO	CAD	950$	BAJO	ALTO	SI	MEDIO	NO	SI	binarias escala de grises color	Automática
R2V FOR WINDOWS 95 & NT 3.0.11. Able Software Co.	SI	GIS	1495$	ALTO[iv]	MEDIO	SI	NULO	SI[5]	SI	binarias escala de grises color	Automática y vectorización Heads-Up opcional.
I/VECTOR 3.5 DEMO VERSION. Ideal Scaners & Systems.	SI	CAD	3995$	ALTO	ALTO	NO	ALTO	SI[5]	NO	binarias	Automática.
PIXEDIT V. 3.989-A0. TechSoft.	SI	ARTE/CAD	A0: 1595$ A4: 595$	BAJO	ALTO	NO	NULO	NO	NO	binarias	Automática.
VPSTUDIO 6.01. Softelec Corporation.	NO	CAD	5000$	BAJO	ALTO	SI	ALTO	SI[5]	NO	binarias escala de grises color	Automática.
SV SCANVECTOR DEMO VERSION 1.2. Cybersonic Technologies.	SI	CAD	A0: 4460$ A3: 2430$ A4: 1490$	BAJO	MEDIO	NO	MEDIO	SI[5]	SI	binarias escala de grises color	Automática.
STREAMLINE 4.0 DEMO. Adobe Systems Inc.	SI	ARTE	199$	NULO	MEDIO	SI	NULO	NO	NO	binarias escala de grises color	Automática.
COREL OCR-TRACE 7. Corel Corporation.	NO	ARTE	695$	NULO	ALTO	NO	NULO	SI[v]	NO	binarias escala de grises color	Automática.
MICROSTATION REPROGRAPHICS ACADEMIC VERSION. Bentley Systems Inc.	NO	GIS	245$	ALTO	ALTO	SI	NULO	NO	SI	binarias escala de grises color	Vectorización Heads-Up.
PROYECTOS FIN DE CARRERA. Mª Carmen Juan Lizandra y Vicente Escuderos Abella	NO	CAD	---------	ALTO	NULO	NO	BAJO	SI[vi]	NO	binarias	Automática.

2 Algoritmo para la vectorización de planos de ingeniería.

2.1 Planteamiento del problema.

A pesar de los importantes avances en CAD, los diseñadores todavía prefieren el lápiz y el papel. Especialmente en las fases más conceptuales del diseño, en las que se baraja una colección incompleta de requisitos e ideas abstractas sobre lo que el producto diseñado deberá ser. Los ingenieros diseñadores están "entrenados" en el empleo del lenguaje gráfico utilizado en los Gráficos de Ingeniería en general, y los planos normalizados en particular. Este lenguaje ha mostrado ser una herramienta potente y flexible para ayudar durante todo el proceso de diseño, y constituye un lenguaje casi universal para especificar gran parte de la información asociada a los diseños de ingeniería. Por tanto, parece razonable utilizar dicho lenguaje para eliminar la falta de flexibilidad en la comunicación entre los diseñadores y los sistemas CAD.

Por otra parte, los diseños ya existentes están especificados en el soporte tradicional: los planos normalizados. Por lo que la tarea de actualización y modificación de viejos diseños requiere generar modelos sólidos de los objetos existentes.

No obstante, conseguir sistemas capaces de interpretar la información técnica contenida en un plano de ingeniería es un problema muy complejo.

La comunicación entre los diseñadores y los sistemas de diseño asistido por ordenador (CAD) está desequilibrada a favor de las necesidades de programación. En su estado actual, los sistemas CAD fuerzan al diseñador a controlar un flujo secuencial; dirigido desde las especificaciones hasta el diseño detallado. Desgraciadamente, dicha circunstancia puede incidir muy negativamente en la creatividad del diseñador.

La causa de la estructura secuencial de los sistemas CAD es la naturaleza secuencial de los lenguajes algorítmicos de programación (y se debe resaltar que las interfaces gráficas de usuario –GUI's- no son ninguna excepción a esta regla). La necesidad de que los programadores definan un "modelo del proceso" aumenta la tendencia a la secuenciación, porque, para los programadores, definir modelos "conceptuales" (lo que el sistema debe hacer), tan parecidos como sea posible a los modelos de "implementación" (la forma en la que el sistema debe hacerlo) es siempre la solución más simple. El resultado es que los sistemas CAD están constantemente pidiendo al usuario que especifique *acciones* (tareas secuenciales completamente

especificadas). Y esta es una mala estrategia para la fase de síntesis, cuando el diseñador está tratando de fijar “visiones”, es decir, ideas poco definidas y desordenadas.

Los comandos “transparentes” (esas órdenes que pueden ser invocadas en cualquier momento durante la ejecución de un programa interactivo guiado por comandos), pueden dar la falsa impresión de que el usuario puede hacer casi todo y en cualquier momento. De hecho, los sistemas CADD (los sistemas de *dibujo* asistido) son muy “interactivos”, porque imponen pocas limitaciones incluso a los usuarios más erráticos. Pero se debe recordar que esto es debido a que están basados en Geometría Descriptiva/Constructiva y en Dibujos Normalizados de Ingeniería, que son dos disciplinas no secuenciales. O, mejor dicho, dos disciplinas basadas en la utilización de dos aspectos de un *lenguaje gráfico* (no secuencial). Por tanto, lo que el usuario hace es utilizar un ordenador para construir una representación gráfica que, una vez acabada, será un documento que contendrá información interpretable sólo por el ser humano. Es decir, que el ordenador será incapaz de interpretar la información contenida en su propia base de datos como información útil para automatizar otras tareas de diseño (ni siquiera cuando la representación esté acabada).

Además, desde el punto de vista del diseñador, en los sistemas CAD de modelado tridimensional (los sistemas de *diseño* asistido) la situación es más precaria. Con los sistemas CAD 3D se pueden crear modelos virtuales tridimensionales, los cuales pueden ser posteriormente mostrados por medio de imágenes tan realistas como se desee y/o por medio de planos normalizados. Pueden ser utilizados incluso como información “de entrada” para todo tipo de análisis y evaluaciones del diseño (estudio de la compatibilidad geométrica, de la resistencia mecánica, del comportamiento térmico, etc.). Pero la construcción de dichos modelos es puramente secuencial. Se ejecuta una única acción tras cada comando, y el sistema vuelve al estado “neutro” (modo de espera para recibir y ejecutar una nueva orden). Además los modelos conceptuales utilizados para construir estos modelos están más próximos a los modelos internos del sistema CAD que a la forma de pensar del diseñador (modelos BRep, CSG, etc.). Es decir, que no se cuenta con el soporte de un lenguaje tan desarrollado como el que da la Geometría Constructiva y la Normalización que sirva para canalizar las ideas poco estructuradas que el diseñador tiene en las fases iniciales de especificación y síntesis del nuevo diseño.

Estas apreciaciones deben quedar referidas al hábito consolidado de los diseñadores de realizar la fase de síntesis mediante la utilización del Croquis, la Geometría Descriptiva y el Dibujo Normalizado. Como se ha dicho anteriormente en los tres casos se trata de lenguajes no secuenciales, siendo precisamente aquellos aspectos derivados de las posibilidades de dichos lenguajes como la sobrexposición de ideas, los solapes visuales (sobremuestras) y, en definitiva, la coexistencia de diferentes soluciones en un mismo marco de trabajo las que hacen de éstos un vehículo muy apreciable para canalizar la creatividad del diseñador.

Resumiendo, el problema estriba en que el pensamiento gráfico (en el sentido de "no verbal") no puede expresarse cómodamente a través de un lenguaje verbal. Definiendo el término verbal como sinónimo de secuencial. Es decir, entendiendo que los lenguajes verbales están basados en la variación de un conjunto de signos a lo largo del tiempo, sin importar que los signos sean símbolos gráficos o sonidos. Por el contrario, los lenguajes no verbales o visuales, y dentro de éstos los "gráficos", son aquellos que basan la transmisión de la información, no sólo en el significado de un conjunto de símbolos gráficos predefinidos, sino también en las relaciones espaciales entre dichos símbolos. Es decir, que las relaciones de semejanza, orden, proporcionalidad y vecindad entre los símbolos tienen una importancia capital que hace inviable expresar oralmente una comunicación gráfica.

Por tanto, los potentes postprocesadores gráficos disponibles en la actualidad para tareas de diseño asistido, deben complementarse con los correspondientes preprocesadores. Entonces, los postprocesadores se concentrarán en su verdadera tarea de ayudar al diseñador en la manipulación de los modelos virtuales 3D, *después* de que dichos modelos hayan sido creados con la ayuda de preprocesadores capaces de leer la información expresada por el diseñador en el lenguaje gráfico que a él le resulta cómodo, en lugar del lenguaje verbal que resulta más cómodo al sistema.

El objetivo explicado arriba es casi utópico, dada la complejidad del lenguaje gráfico. Podemos resumir el problema diciendo que necesitamos capacidad para:

Interpretar un boceto (dibujo sin instrumentos de delineación) como una figura "vectorial" plana. Es decir, una figura descrita en términos de primitivas geométricas.

Reconstruir un modelo geométrico tridimensional a partir de una o más figuras geométricas planas [33].

Añadir al modelo 3D toda la información complementaria a la propia forma geométrica. Información que estará expresada por medio de símbolos normalizados.

Cada una de las tres tareas apuntadas es suficientemente compleja, por lo que la consideración simultánea de todas ellas está de momento fuera del alcance de desarrollos prácticos. Además, lo ideal sería que la capacidad de "lectura" fuera interactiva. Es decir, que la información debería ser leída e incorporada al modelo tal como el diseñador la fuera introduciendo, de forma que se generasen modelos "provisionales" que permitiesen al diseñador ir comprobando algunos aspectos de la validez de los mismos.

El presente proyecto fin de carrera pretende desarrollar algunos aspectos encuadrados dentro de la primera de éstas tres tareas, es decir la obtención de dibujos 2D en formato vectorial a partir de dibujos de ingeniería en papel.

Los planos de ingeniería incluyen además de una descripción geométrica de los elementos que en ellos se representan, información adicional en forma textual (cifras de cota, leyendas, etc.) y simbólica (líneas de acotación, líneas auxiliares de cota, patrones de sombreado, etc.). Así pues podemos desglosar el proceso en tres nuevas tareas diferenciadas:

Obtención de entidades geométricas.

Discriminación de textos.

Discriminación de entidades simbólicas.

Estas tres tareas, aunque perfectamente diferenciadas, deberán de relacionarse entre sí para obtener una interpretación semántica de los dibujos. Probablemente el ejemplo más claro de esta relación se presente en las acotaciones, ya que en ellas debemos distinguir entre la cifra de cota (texto), la línea de cota y líneas auxiliares de cota (entidades simbólicas) y la parte del dibujo que se está acotando (entidad geométrica). Para abarcar la semántica completa de una acotación, en primer lugar debemos identificar las líneas de cota y asociarlas con sus correspondientes cifras de cota y con sus líneas auxiliares de cota para más tarde relacionar este conjunto con la

entidad geométrica que se está acotando y de este modo poder interpretar como un conjunto la forma y toda la información adicional que nos proporciona la cifra de cota.

Otras características simbólicas que se deberán tener en cuenta a la hora de analizar un dibujo serán el grosor de línea, las líneas discontinuas y los rayados (sombreados) de diferentes tipos. Para poder reconocer todo este tipo de características será necesario que los dibujos respeten algún tipo de standard de dibujo como puedan ser las normas ANSI o ISO.

Si consideramos todos los requerimientos que un sistema de estas características tiene, podemos clasificarlo como un sistema de nivel de inteligencia 5 según la escala que vimos en el capítulo anterior.

El presente trabajo se centrará principalmente en la obtención de entidades geométricas y más específicamente en sus primeras fases, aunque también se intentará proporcionar una visión global de un sistema de estas características.

2.2 Otras aproximaciones al análisis de planos de ingeniería.

En la abundante bibliografía consultada aparecen diversos sistemas que a partir del ráster obtenido al escanear un dibujo en papel pretenden reconstruir un modelo 2D de la información gráfica que en él aparece. Nosotros hemos podido encontrar hasta seis sistemas distintos propuestos por diferentes autores que proporcionan diferentes aproximaciones al tema. Tres de estos sistemas parecen estar bastante desarrollados según la bibliografía que hemos podido encontrar y son los propuestos entre otros autores por Kasturi [6, 7], Langrana [16 - 19] y Dori [8 - 10]. Los dos primeros hacen una aproximación al problema de la obtención de las componentes lineales de los dibujos basada en tradicionales algoritmos de adelgazamiento (thinning), mientras que el último desarrolla una técnica propia. Hay un cuarto sistema propuesto por Priestnall y otros [2] que por la bibliografía que hemos podido encontrar no está en un nivel de desarrollo tan avanzado como los tres anteriores. En cualquier caso, aporta interesantes características como la estructura de datos de contornos emparejados (paired outlines data structure) para realizar el proceso de adelgazamiento que nos han inclinado a basarnos en algunas de sus ideas para el desarrollo de nuestro sistema. Otros autores como Han y Fan [3] desarrollan una técnica de adelgazamiento basada en emparejamiento de contornos (contour matching) pero no llegan a describir un sistema de procesamiento de dibujos de ingeniería completo. El último de los artículos en ser

mencionado [11] tampoco llega a desarrollar un sistema completo y al igual que el anterior solo comenta una técnica de extracción y vectorización de líneas que según el autor, Toru Kaneko, permite extraer líneas que no sólo tocan, sino que incluso atraviesan caracteres y símbolos.

2.2.1 Kasturi

El algoritmo propuesto por Kasturi et al. en [6] no está dirigido específicamente hacia dibujos de ingeniería pero en [7] extiende su trabajo para extraer conjuntos de dimensión (acotaciones) en planos de ingeniería dibujados según la norma ANSI. Su sistema incluye:

Separación de cadenas de texto de los gráficos.

Generación de componentes conexas.

Filtros de área/proporción. Aparta aquellas componentes conexas cuyo tamaño está muy por encima del área media de las componentes conexas de la imagen.

Agrupación de componentes conexas en el dominio de Hough. Utilizando la transformada de Hough sobre los centros de las componentes conexas obtiene alineamientos de estas que servirán para agruparlas según su colinearidad.

Separación de componentes en palabras y frases. Se agrupan las componentes conexas dentro de una misma componente colinear en palabras y frases.

Postproceso para refinar la segmentación. Se aplican dos filtros para considerar como gráficos los caracteres repetidos y las líneas discontinuas. Además, se emplea una técnica especial para intentar resolver el problema de los caracteres en contacto con líneas.

Identificación de segmentos de línea y sus atributos.

Separación de componentes gráficas sólidas. Se somete a la imagen ya libre de símbolos a un proceso de limpieza y a una serie de erosiones seguidas del mismo número dilataciones más algunos pasos de procesamiento adicionales para separar las áreas gruesas (áreas rellenas de un color) que tantos problemas causan a los algoritmos de adelgazamiento.

Esqueletización y seguimiento de contornos. Utilizando técnicas tradicionales de adelgazamiento para la imagen de líneas y de seguimiento de contornos para la de objetos sólidos se procesarán las imágenes obtenidas en el anterior paso. Además se aplicará para las líneas finas un filtro para calcular su grosor medio.

Segmentación entre líneas rectas y curvas. Se intenta buscar los puntos críticos del esqueleto obtenido analizando la pendiente de las líneas en busca de un cambio brusco. Las líneas que están continuamente cambiando de pendiente se considerarán curvas. El autor reconoce ciertos problemas en el método especialmente cuando hay ruido en los datos.

Detección de líneas discontinuas. Aplicando un interesante algoritmo se detectan líneas discontinuas de diferentes tipos que tengan más de cuatro tramos.

Reconocimiento de primitivas de dibujo y sus atributos.

Generación de bucles de redundancia mínima. Se tratará de encontrar los bucles que son necesarios y suficientes para describir las formas cerradas de la imagen y luego se compararán con un catalogo de formas conocidas.

Reconocimiento de formas superpuestas. Se intenta clasificar los bucles no reconocidos como formas ocluidas por otras teniendo en cuenta que son polígonos convexos.

Detección de patrones de relleno y relaciones de inclusión. Se buscarán rallados dentro de los bucles identificados como formas conocidas y además se intentarán establecer relaciones de inclusión espacial entre estos mismos bucles teniendo en cuenta que son polígonos convexos. Cuando cierto número de pequeños objetos está rodeado por otro se considerarán un patrón de relleno.

Descripción estructural de las primitivas constituidas por segmentos curvos.

Estimación de radio y centro. Se calculan por un método de bipartición a lo largo de las líneas consideradas curvas en la segmentación. En él se van

tomando conjuntos de tres puntos para estimar el centro de los arcos circulares.

Reconocimiento de círculos. Se utiliza un método similar al utilizado para encontrar los bucles de redundancia mínima poligonales pero en lugar de segmentos rectos se toman segmentos de arco.

2.2.2 Langrana

El trabajo de Langrana et al. consiste en un sistema especializado para planos de ingeniería (RENDER [16]) que se complementa con un sistema de postprocesado (P-RENDER [17]) y con dos trabajos más, uno que se encarga de los aspectos de la acotación (D-RENDER) y otro que se encarga del reconocimiento de rasgos geométricos 2D y 3D en dibujos consistentes en vistas ortográficas de componentes mecánicas (I-RENDER [18, 19]). El sistema RENDER consiste en los siguientes pasos:

Preprocesado. Para eliminar ruido y otras características no deseadas mediante operaciones morfológicas tradicionales.

Segmentación de la imagen. Esta fase es casi idéntica a la fase de separación de componentes gráficas sólidas del articulo [6] aunque en [16] la explicación es ligeramente más completa. Se procede como en el primer articulo utilizando adelgazamiento para las líneas obtenidas (aunque no se hace mención alguna al filtro de detección de grosor) y seguimiento de contornos para las áreas sólidas que contendrán las cabezas de flechas que serán esenciales para extraer los conjuntos de acotación.

Vectorización de las líneas. En un proceso de preprocesado se obtienen los puntos críticos y en la vectorización propiamente dicha se etiquetan las regiones que pueden pertenecer a curvas para aproximarlas mediante cónicas y ajuste mínimo cuadrático y el resto de puntos se aproximan con rectas mediante un algoritmo de aproximación poligonal.

P-RENDER pretende corregir las imperfecciones que se introducen durante el proceso anterior reconociendo en la imagen ciertos conjuntos de primitivas y ocupándose también de procesar las entidades trazadas con líneas discontinuas utilizando exactamente el mismo algoritmo que Kasturi.

La salida de P-RENDER se utiliza para reconocer rasgos geométricos 2D de vistas ortográficas de los que se ha elaborado una librería. Estos rasgos se deberán buscar en cada una de las tres vistas por separado y después se relacionarán para intentar identificar detectar el rasgo geométrico tridimensional con el que se corresponden. Los dibujos con los que se debe alimentar este sistema no deben tener elementos de acotación. Los elementos de acotación se discriminarán e interpretarán y la información extraída se asociará con la geometría que está describiendo utilizando técnicas de geometría variacional.

2.2.3 Dori

El sistema desarrollado por Dori [8 - 10] se conoce como MDUS y su fase de detección de primitivas comprende cuatro algoritmos: OZZ (orthogonal zig-zag) para la detección de líneas rectas, PBT (perpendicular bisector tracing) para la segmentación de arcos [9], SAR (self-supervised arrowhead recognition) para reconocer cabezas de flechas y un algoritmo de detección de cajas de texto. OZZ está constituido por 4 fases:

Sparse screening (cribado esparcido). Cuando se encuentra un pixel negro se empieza a contar el número de pixeles negros que le siguen en esa dirección hasta que se encuentran suficientes pixeles blancos. Si se han encontrado suficientes pixeles negros se comienza el siguiente paso.

Zigzag. Se recorre el conjunto de pixeles negros sospechosos de ser una línea siempre en dirección horizontal o vertical y cuando se encuentra un área blanca se cambia la dirección 90º. Se mantienen estadísticas acerca de los conjuntos de series de pixeles negros recorridos en cada dirección que proporcionan información para decidir sobre la presencia o no de una línea, sus puntos extremos, su anchura y para permitir saltarse las uniones. Como resultado de este proceso se generan dos listas, una de líneas diagonales y otra de líneas paralelas a los ejes.

Unión de líneas. Se tratará de unir líneas procedentes de las dos listas.

Corrección de esquinas. La lista resultante del proceso anterior se procesará para rellenar los pixeles negros de las esquinas.

El algoritmo PBT toma como entrada la lista de líneas que produce OZZ. Se formarán cadenas de líneas de presuntos arcos y se decidirá la duración de la cadena de líneas mientras no varíe su dirección de curvatura. Se tomarán los puntos extremos de la

cadena de líneas y se trazará en la dirección de la mediatriz de la línea que une dichos puntos hasta encontrar una serie de pixeles negros. Buscaremos el centro de la serie para obtener un tercer punto que junto con los otros dos nos proporcionará un primer centro del arco calculando la intersección de las mediatrices de las dos nuevas líneas que unen los dos puntos iniciales con el nuevo punto. Se repetirá el proceso recursívamente hasta obtener tres centros y se comprobará que los centros que se han obtenido están suficientemente cerca en cuyo caso el centro de masas del triángulo que forman se considerará el centro. Este proceso se repetirá partiendo ahora de los puntos que se han obtenido situados en el arco y se continuará hasta que la distancia entre dos centros consecutivos sea menor que un determinado umbral.

La bibliografía de este autor que se ha podido consultar es más bien escasa pero se tienen referencias de numerosos artículos publicados por él sobre el tema que son citados por multitud de autores. Sobre todo hay que destacar las más que frecuentes referencias que se hacen a él en el área de la detección de los conjuntos de acotación y de la interpretación de los planos en general.

2.2.4 Priestnall

El articulo [2] presenta un método de reconocimiento de cabezas de flecha en dibujos de ingeniería. El método se aprovecha de una nueva estructura formada a partir de los vectores de contorno extraídos de la imagen binaria escaneada. Los vectores de contorno opuestos se emparejan cuando es posible creando una estructura de contornos conectados que representan rasgos lineales, áreas desemparejadas en las uniones, esquinas y puntos extremos. Esta estructura es muy descriptiva y conserva la mayor parte de la información y, además, permite que se desarrollen módulos de extracción de rasgos basados en reglas. Esta estructura de contornos emparejados y el método de extracción de un tipo estándar de cabeza de flecha se describen con detalle, además, en el artículo se cita la posibilidad de explotar la estructura para otros tipos de símbolos y cabezas de flecha.

2.2.5 Han y Fan

Los autores presentan en [3] una nueva aproximación para la esqueletización de dibujos de ingeniería totalmente distinta de las técnicas de adelgazamiento tradicionales que se basa como en el anterior articulo en el emparejamiento de contornos. Los contornos se obtienen mediante un algoritmo clásico de extracción de contornos.

Posteriormente se vectorizan y se aproximan mediante rectas, arcos de circunferencia y curvas cúbicas. Para detectar las dos primeras se utiliza la función de densidad de la pendiente (slope density function) que es el histograma o distribución de frecuencias del cambio de la pendiente recogido a lo largo del contorno. Si no se puede aproximar adecuadamente con este método se utilizarán polinomios cúbicos ajustados mediante un método iterativo de mínimos cuadrados.

A los contornos vectorizados se les aplicará un método de emparejamiento de líneas que emparejará respectivamente entre sí los elementos de cada uno de los tres grupos. Los segmentos de contorno no tienen porqué emparejar en toda su longitud, es decir que se dividirán en dos (segmento emparejado y segmento no emparejado). Se utilizará un conjunto de reglas de emparejamiento especifico para cada uno de los tres casos:

Rectas:

- Pendiente de las líneas aproximadamente igual (el ángulo entre ellas es 0º ó 180º aprox.).
- Su proyección sobre algún eje debe solaparse y el ratio entre la longitud de la parte solapada y la longitud del segmento más corto de los dos debe ser mayor que un parámetro (90%).
- Al menos dos de las cuatro distancias entre los extremos de las líneas y éstas debe estar por debajo de un umbral .

Arcos:

- Centros próximos.
- La diferencia entre sus radios debe estar por debajo de un umbral.
- El ángulo de comienzo de un arco debe ser menor que el de fin del otro arco y viceversa.

Curvas:

- Sus proyecciones sobre alguno de los dos ejes deben solaparse y el ratio de solapamiento debe superar un umbral.
- Deben estar próximas.

Una vez emparejan dos elementos se procederá a extraer la fórmula de la línea del esqueleto correspondiente:

Rectas: unir los puntos medios de los segmentos que unen los extremos de ambas rectas.

Arcos: arco con el mismo centro y radio r1 + [(r2 - r1) / 2].

Curvas: unir los puntos medios de los segmentos que unen los puntos tomados en la medición de distancias entre curvas. Esta unión se realizará mediante una curva calculada por el método de los polinomios cúbicos citados anteriormente.

Para rellenar los espacios que quedan entre las diferentes parejas se propone un método por el que se genera un grafo que es la unión del grafo de adyacencia de las figuras y el grafo que representa los emparejamientos que se han producido. El nuevo grafo tendrá los arcos etiquetados y se buscarán circuitos en los que las etiquetas sigan cierta secuencia realizando un recorrido DFS (primero en profundidad) del grafo para ir eliminando ciertas aristas. Al final del proceso los circuitos resultantes con longitud impar indican que entre sus aristas no hay colinearidades, mientras que si la longitud del circuito es par tenemos una colinearidad entre las aristas que en él intervienen. Dependiendo de sí se ha detectado colinearidad o no, se procederá a resolver la unión de un modo especifico a cada caso. El proceso se compone de las siguientes fases:

Generación del grafo de adyacencia de contornos: sus nodos son los vectores de contorno y los arcos significan la adyacencia entre ellos.

Generación del grafo de contornos emparejados: sus nodos son los vectores de contorno y los arcos son las relaciones de emparejamiento.

Generación del grafo de relación de contornos: es la unión de los dos anteriores etiquetando los arcos provenientes del primero como A (adyacencia) y los provenientes del segundo como P (emparejados).

Detección de huecos: los huecos se detectan mediante la extracción de ciclos con arcos P-A alternados con longitud mayor de tres. Para no repetir ciclos se deben borrar los arcos A que formen parte de ellos una vez detectados. La búsqueda de ciclos se lleva a cabo mediante un algoritmo DFS comenzando siempre por el nodo de índice menor. Cuando todos los vértices sean P deberemos parar.

Relleno de los huecos: Tenemos dos casos dependiendo de la longitud de los ciclos (ver articulo para la clasificación):

Ciclos pares: no hay vectores del esqueleto continuos colineares.

Ciclos impares: solo hay un par de vectores del esqueleto continuos colineares.

2.2.6 Kaneko

La principal característica del algoritmo de Kaneko consiste en la extracción de líneas que incluso atraviesan caracteres. El algoritmo consta de 4 etapas cada una con sus correspondientes subfases:

Detección de indicios de línea.

Propagación de la distancia direccional. Se realizan cuatro barridos (dos en vertical y dos en horizontal) sobre una matriz que ocupa un byte por cada bit del ráster. Cada uno de estos barridos se realiza en los dos sentidos y se van anotando en la correspondiente posición de la matriz los valores de la máxima distancia en la dirección del barrido desde cada pixel de figura hasta el contorno de esta. Mediante un complicado mecanismo de buffers se consigue realizar el proceso con un único recorrido de la matriz fila a fila hacia delante y hacia atrás

Clasificación de pixeles. Los pixeles se clasifican como pixeles indicio de línea, pixeles de indicio de línea de frontera y pixeles de fondo teniendo en cuenta los valores obtenidos en el paso anterior.

Etiquetado. Mediante algoritmos de rellenado clásicos se etiquetan los diferentes grupos de pixeles de indicio de línea que están separados entre sí por pixeles de indicio de línea de frontera.

Adaptación de líneas. Mediante un algoritmo recursivo se encuentran para cada grupo de pixeles con la misma etiqueta los puntos que corresponden a los vértices de las líneas.

Vectorización.

Eliminación de pequeñas líneas espurias. A veces se detecta parte de un carácter o símbolo que toca una línea larga como un indicio de línea. Se eliminarán todas aquellas líneas que estén por debajo de un parámetro.

Unificación de líneas solapadas. Si un carácter es atravesado por una línea aparecerán varios conjuntos de indicios de línea que derivarán en varias líneas superpuestas que deberemos representar por una sola.

Conexión de pequeños huecos por chequeo de colinearidad. Cuando los segmentos a ambos lados de un pequeño hueco son casi colineares deberemos eliminar el hueco.

Intersección de las uniones de línea. Se calculará la intersección de las líneas que confluyan en esquinas o en uniones en T.

Unión de líneas partidas no colineares. En ocasiones dos líneas que debían intersectarse en una esquina no lo hacen por el error admitido en la fase de adaptación de líneas y ahora deberemos calcular la intersección.

Rastreo de líneas perdidas. Cuando muchos caracteres interrumpen una línea esta no se detecta como tal y se ve interrumpida así que ahora se reconstruirá comprobando si al trazar una línea entre dos extremos inconexos que están dentro de un rango esta cae siempre encima de pixeles de figura del ráster.

2.3 Proyecto de sistema de interpretación de planos de ingeniería.

En este punto presentaremos una exposición de un posible sistema para la interpretación de planos de ingeniería orientado a formar parte de un proyecto más ambicioso de reconstrucción tridimensional. Nuestro sistema se dividirá en las siguientes fases:

Preprocesado del ráster. Con este proceso se tratará de compensar las deficiencias de la imagen que pueden proceder básicamente de dos fuentes:

Pequeños defectos en el proceso de escaneado y umbralización debido a factores tales como la desigual iluminación del documento, ruido debido a fluctuaciones en la iluminación, limitaciones espaciales del ancho de banda del sistema de escaneado, suciedad presente en el sistema de escaneado, etc.

Baja calidad del dibujo original principalmente si el dibujo es un dibujo usado y tiene cierta antigüedad pueden aparecer manchas, sombras, variaciones del color, arrugas o pliegues en el papel y otros tipos de ruido que aparecerán en la imagen final.

El ruido procedente de alguna de estas fuentes como las grandes manchas o los pliegues y arrugas del papel son imposibles de eliminar del ráster utilizando métodos automáticos por lo que se deberá recurrir a sistemas manuales de edición ráster para subsanarlos. Otros de los problemas antes mencionados causan en el ráster la aparición de huecos y pequeñas rupturas así como ruido en forma de pequeños puntos dispersos por la imagen. Este último tipo de ruido puede ser corregido en parte por la utilización de operaciones morfológicas clásicas [12, 16] como pueden ser:

Erosión: eliminar aquel pixel del dibujo que tenga por vecino a otro que forme parte del fondo. Elimina una capa de pixeles de los objetos.

Dilatación: añadir aquel pixel que forme parte del fondo que tenga por vecino a uno del dibujo. Añade una capa de pixeles a los objetos.

Opening: erosión seguida de una dilatación. Elimina una capa del objeto para después recuperarla. Abre huecos entre objetos que inicialmente estaban en contacto.

Closing: dilatación seguida de una erosión. Es opuesta a la erosión (aunque no estrictamente su inversa) y produce la unión de objetos muy próximos así como la eliminación de pequeños huecos que aparecen en el interior de las figuras.

Estos filtros poseen diversos parámetros que habrá que ajustar para conseguir resultados adecuados a cada tipo de circunstancias:

Tipo de vecindad: 8-conexa, 4-conexa.

Umbral de pixeles de la vecindad: número de pixeles iluminados/apagados en la vecindad de uno dado que determinan el futuro estado de este último pixel.

Profundidad: número de veces que se aplica la operación.

Los filtros morfológicos deben de ser aplicados por el usuario en áreas determinadas de la imagen para conseguir buenos resultados y no producir nuevos errores en otras zonas. La elección del filtro, los parámetros y la zona de aplicación adecuados a cada caso no es un asunto fácil que requiere de la intervención de un operador humano experimentado que tome estas decisiones.

Aunque no es el objetivo de este proyecto, podía ser interesante realizar una búsqueda bibliográfica para tratar de encontrar mejores métodos de limpieza (más automáticos), así como algún filtro especializado en la eliminación y suavizado del ruido de los contornos de las figuras que como ya veremos, podría proporcionar algunas ventajas en fases posteriores.

Hay que aclarar que las imágenes sobre las que trabajaremos serán imágenes binarias (solo 2 colores: fondo y dibujo) puesto que según se ha podido constatar en la bibliografía consultada este tipo de imagen es suficiente para los dibujos de ingeniería, ya que en ellos no se requiere el color como para otras aplicaciones como por ejemplo los mapas. Las imágenes serán previamente escaneadas con una resolución de al menos el doble del menor grosor de línea para evitar el aliasing [20]. Si la imagen original fuera obtenida en escala de grises seria necesario binarizarla (binarization) o umbralizarla (thresholding) para convertirla en una

imagen binivel. En [31] se puede ver una comparativa del comportamiento de diferentes métodos de binarización.

Discriminación y reconocimiento de textos. La discriminación de textos parece ser un problema bastante bien resuelto en el proyecto de Mª Carmen Juan [13] que emplea el algoritmo de Kasturi y Fletcher precursor del utilizado en el sistema anteriormente expuesto del primer autor [6]. En este último sistema se completa el algoritmo original al ocuparse de problemas como los caracteres repetidos, las líneas discontinuas detectadas como texto y los textos en contacto con líneas. Respecto a este último tema podía resultar de interés el artículo [32] que aunque se centra en la extracción de textos de formularios rellenados a mano o a máquina puede aportar ideas interesantes para ayudar a resolver el problema.

Una posibilidad de solución de este problema que se ha descartado, es la utilización del algoritmo de Kaneko para la extracción de líneas [11]. Este algoritmo aparte de ser muy complicado (solo hay que ver la cantidad de fases que tiene y la complejidad de estas) deja numerosos puntos oscuros en la explicación de muchas de sus fases como por ejemplo la extracción de los caracteres que estaban en contacto con las líneas. Por otra parte hay gran cantidad de parámetros en el proceso que se deberán ajustar según las características de cada dibujo para que los resultados sean óptimos y esto no es una tarea fácil. También hay dificultades para detectar líneas gruesas y para detectar líneas curvas a partir de cierta curvatura. Aparte de todo esto, el algoritmo no parece ser demasiado económico ni temporal ni espacialmente lo cual resulta esencial para procesar planos de un tamaño considerable sin depender fuertemente de la resolución de escaneado. Por todas estas razones este algoritmo no parece una solución muy adecuada a nuestros problemas.

El reconocimiento de caracteres OCR es un tema en el que se han realizado muchos trabajos y muy buenos por lo que parece más adecuado adaptar a las necesidades de del nuestro, un sistema OCR ya existente, en lugar de desarrollar uno por nuestra cuenta, tarea nada trivial.

Extracción de líneas. Como ya hemos visto, en este punto los sistemas difieren considerablemente unos de otros. Por un lado están los sistemas que utilizan técnicas tradicionales de adelgazamiento (thinning) para obtener un esqueleto del

que luego extraerán los puntos críticos (segmentación) que posteriormente aproximarán mediante rectas, arcos y curvas [26 - 28]. Por otra parte aparecen los sistemas que a partir de los contornos de las figuras pretenden conseguir el esqueleto del área que encierran haciéndolos corresponder unos con otros cuando se encuentran en lados enfrentados de una línea.

El primero de los métodos tiene dos importantes inconvenientes derivados de la naturaleza local de los operadores utilizados en el proceso de adelgazamiento. El primero de ellos se produce en la intersección de líneas de un grosor considerable ya que se producen los llamados efectos X, Y y T (ver [12] pags. 155 y 156 y [26]) que introducirán distorsiones en la imagen. El segundo problema es todavía más preocupante y consiste en la incapacidad de los métodos de adelgazamiento para procesar áreas sólidas (rellenas de un color). La importancia de estas áreas sólidas radicará en que algunos tipos de cabezas de flecha se pueden considerar como tales y como ya veremos son vitales para la detección de los conjuntos de acotación por lo que no podemos ignorarlas.

Existen multitud de algoritmos diferentes de adelgazamiento debido al profuso uso que se hace de ellos en diferentes campos: reconocimiento de caracteres, inspección de circuitos impresos, análisis de la forma de los cromosomas, análisis de glóbulos blancos, mecanografía cuantitativa, clasificación de huellas dactilares, etc. [25]. Pueden verse algunos de estos artículos en el Vol. 7 No. 5 (1993) del International Journal of Pattern Recognition and Artificial Intelligence.

Existen diferentes artículos que comparan entre sí las características de distintos algoritmos de este tipo [23, 24] así como estudios acerca de los principios en los que se basan estos algoritmos[25]. Generalmente la conclusión que se obtienen de estos artículos es que no existe un algoritmo de adelgazamiento que resulte una panacea. Además estos estudios parecen bastante orientados a la utilización de los algoritmos de adelgazamiento en el campo del reconocimiento de caracteres. El problema de las distorsiones en los cruces de líneas puede abordarse desde el punto de vista de emplear un algoritmo especialmente pensado para planos de ingeniería [28] pero por lo que hemos podido apreciar, este problema no lo resuelven completamente. Hu y Li [26] proponen un algoritmo de adelgazamiento orientado especialmente para preservar los cruces de líneas. Este algoritmo tiene entre otros, el inconveniente de que no garantiza la conectividad de las líneas de la imagen y, además, supone que

estas tienen grosores similares, siendo este último aspecto poco asumible en los dibujos de ingeniería.

El problema de las áreas rellenas se puede afrontar desde el punto de vista de Kasturi o Langrana [6] separando las áreas sólidas de las lineales mediante operaciones morfológicas. Todo lo que tiene un grosor en pixeles superior a cierto parámetro se considerará un área rellena y Langrana va mucho más allá considerando potenciales puntas de flecha a todas las entidades detectadas de este modo.

Las técnicas de adelgazamiento tradicionales, lo que obtienen es un esqueleto formado por puntos conexos de la imagen del objeto. Este esqueleto será necesario someterlo a un proceso de vectorización para obtener una aproximación más compacta y manejable. Existen diferentes métodos de vectorización, Janssen y Vossepoel [20] han realizado una revisión de ellos y los comentan del siguiente modo:

Métodos de área. El método de Wall-Danielsson [41] es un ejemplo de este tipo de métodos. Una extensión se describe en [42]. El criterio usado es la máxima desviación de área permitida por unidad de longitud desde la vectorización a la curva aproximada. El último punto de la curva que no viola este criterio se convierte en un nodo en la vectorización y este proceso se repite hasta el final de la curva.

La principal dificultad es que los nodos a veces no corresponden con esquinas en la curva porque un nuevo borde solo se empieza tan pronto como se viola el criterio[43]. Esto puede suceder solo unos pocos pixeles después de que una larga línea gire una esquina y no en la misma esquina.

Métodos de conos de intersección. Estos métodos, presentados por Williams [44, 45] y Sklansky y Gonzalez [46] se basan en definir un circulo alrededor de cada punto de la curva del dibujo. El borde aproximado tiene que intersectar todos estos círculos. El nombre de este método tiene origen en la forma de unión de todos esos círculos vistos desde el punto de comienzo, la cual es un cono. En [45] el autor mejora sus resultados obtenidos en [44] permitiendo que los bordes finalicen fuera de los círculos. De este modo, una propiedad de este método es que los bordes aproximados no son necesariamente parte de la curva original.

Métodos minimax. Estos métodos presentados por Kurozumi y Davis [47] se basan en la minimización de la máxima distancia entre la curva original y los bordes aproximados. En consecuencia, los nodos de los bordes aproximados no son necesariamente parte de la curva original como sucedía en el método anterior.

Métodos de franjas de tolerancia. Dado un número de puntos y una anchura de franja máxima y mínima, estos métodos (Dettori [48], Leung y Yang [49]) tratan de ajustar una franja alrededor de los puntos que van a ser aproximados. Se pueden tomar diversas aproximaciones: el número de puntos originales de la imagen en una franja o la longitud de la franja pueden ser maximizados, o la anchura de la franja puede ser minimizada. También son posibles las combinaciones. El borde resultante es el eje central de la franja.

En [48, 49] no está claro como se manejan los puntos de ramificación. Los algoritmos construyen una franja desde un punto de comienzo dado hasta el siguiente punto hasta que se viola el criterio. Cuando el punto de comienzo es un punto de ramificación, pueden surgir situaciones ambiguas, especialmente cuando la máxima anchura de franja es grande.

Métodos de series longitudinales. Después de calcular la codificación en series longitudinales (run-lenght) de la imagen, se construye un grafo de líneas de adyacencia (ver por ejemplo Pavlidis [50, 51]). Este grafo se usa para determinar los bordes de la vectorización. Este método puede no ser adecuado porque una línea ligeramente inclinada con una frontera dentada puede generar bastante más de un borde. Además, la vectorización resultante puede depender de la orientación.

Métodos de marcado y barrido. Usando como entrada las coordenadas del primer punto de una curva y el código de encadenamiento del contorno, el método de Sirjani y Cross [52] se basa en marcar todos los puntos que pueden ser finales de una línea recta. En el siguiente paso los puntos superfluos marcados se descartan, y los puntos restantes son los nodos de la vectorización. Este método no es adecuado porque se usa el contorno exterior: los dibujos que tienen líneas completamente encerradas por otras no se vectorizarán correctamente.

Métodos de curvatura mínima y máxima. Se pueden usar los puntos de curvatura mínima y máxima de una curva para construir la vectorización. Los métodos de

Teh y Chin [53] y Ray y Ray [54] no requieren de ningún parámetro de entrada: los puntos se seleccionan basándose en sus regiones de soporte. La idea es que esos "puntos dominantes" proporcionan información importante para el proceso de reconocimiento del sistema de visión humano. La posición de los puntos dominantes depende fuertemente del método por el que se determina la región de soporte. Si se determina incorrectamente, los puntos dominantes pueden ser descartados. En [53] se usan tres métodos diferentes pero no está claro cual se prefiere.

Métodos de distancia perpendicular máxima. El método de Douglas-Peuker [22] es un ejemplo de estos métodos. Se usa como criterio la máxima distancia perpendicular en pixeles t_{dp} desde lo vectorizado hasta la curva a aproximar. El algoritmo comienza con dos puntos dominantes de la curva como nodos de la vectorización. Cuando la máxima distancia perpendicular desde la vectorización a la curva es mayor que t_{dp}, un nuevo nodo se añade a la vectorización en el lugar de la máxima desviación. Este proceso se repite recursívamente.

Otros autores que usan este principio incluyen a Ramer [57] y a Leu y Chen [56]. Los últimos se concentran en la unicidad de la vectorización aproximadora y en su precisión.

Dori y otros describen un método para vectorizar dibujos de ingeniería el cual no requiere líneas de un píxel de grosor (no necesita adelgazamiento). Esto se realiza escaneando cada cierto número de filas y columnas, y por lo tanto evita la necesidad de direccionar cada uno de los pixeles de la imagen. Después de que ha sido encontrada una masa (con forma de línea) de pixeles de objeto, se usa un algoritmo de "zig-zag" que sigue los pixeles de objeto de la masa en el sentido de las columnas hasta que se encuentre un pixel que no sea del objeto. Entonces sigue los pixeles de la masa en el sentido de las columnas hasta que se encuentra un pixel que no es del objeto, en ese momento seguirá de nuevo en el sentido de las filas, etc. hasta que se haya alcanzado el final de la masa. Se han proporcionado facilidades especiales para escanear líneas horizontales y verticales. Para detectar uniones en el dibujo, se inspecciona la longitud media en cada recorrido de fila y columna. Si se detecta una desviación, se asume una unión o cambio en el grosor de la línea, y se emprende la acción apropiada.

Una desventaja de este método es que aproximadamente 12 umbrales deben ajustarse previamente para el algoritmo de vectorización. No se especifica para muchos parámetros como deben establecerse sus valores. También, es difícil juzgar la ejecución ya que no hay presente ninguna medida.

La última aproximación que se ha encontrado al tema del adelgazamiento de la imagen consiste en las técnicas de emparejamiento de contornos. En lugar de ir eliminando capas de pixeles del contorno del objeto basadas en operadores locales como hace el método tradicional de adelgazamiento, estas técnicas parten de una aproximación de los contornos de los objetos de la imagen. Consistirán en encontrar parejas entre las líneas que aproximan los contornos, tales que se encuentren en los lados opuestos de los contornos de las componentes lineales de las figuras. El siguiente paso consiste en encontrar la línea central de las dos que están emparejadas. Hay que destacar que en este momento es posible calcular el grosor medio de la línea de la imagen que encierra cada pareja. Las parejas dejan en las esquinas, uniones y cruces de líneas espacios en los que no existe ningún emparejamiento que deberemos rellenar prolongando las líneas que confluyen en cada uno de estos puntos hasta aproximar correctamente la imagen original. También se producen desconexiones en otros puntos de la imagen que corresponderán con las áreas sólidas a las que anteriormente se hacia referencia al hablar del emparejamiento clásico.

En este momento conviene citar el trabajo de Di Zenzo [15] en el que desarrolla algunos aspectos teóricos del emparejamiento de contornos aunque sin llegar a ninguna realización práctica.

Lo más destacable de estos métodos es que al no emplear información tan local como la que utilizan los algoritmos clásicos de adelgazamiento el resultado no se ve tan afectado por las distorsiones que sufren las líneas centrales obtenidas con el adelgazamiento clásico. Por otra parte, los contornos del objeto contienen toda la información que aparecía en la imagen original, mientras que el adelgazamiento tradicional se basa en ir descartando parte de esa información al ir eliminando capas de pixeles. La diferencia fundamental entre ambos métodos consiste en que mientras que el adelgazamiento clásico empieza trabajando en el dominio del ráster, los algoritmos de emparejamiento lo hacen en el vectorial. Adicionalmente, la salida

que producen estos algoritmos es un esqueleto que se encuentra directamente en forma vectorial, en lugar de una imagen ráster del esqueleto que posteriormente deberá ser vectorizada como hace el adelgazamiento clásico. Otra ventaja de los algoritmos emparejamiento de contornos es que resulta muy fácil el calcular el grosor medio de las líneas medido en pixeles, puesto que para calcular las parejas hay que calcular necesariamente la distancia entre las líneas de contorno que estamos intentando emparejar y esta distancia coincide con el grosor de la línea a la que corresponden.

Como ya hemos comentado en el anterior apartado de este capitulo, se han encontrado dos algoritmos que se basan en el emparejamiento de contornos. Una diferencia fundamental entre estos dos algoritmos consiste en el mecanismo de vectorización de los contornos que utilizan. Han y Fan en [3] no explican en detalle el método que utilizan para aproximar los contornos con rectas, arcos de circunferencia y curvas cúbicas paramétricas, pero por lo que comentan en el articulo, el método que utilizan tiene una fuerte componente de procesamiento matemático. En el articulo de Priestnall y otros autores [2], cada contorno se aproximará utilizando un polígono, es decir, que la imagen se aproximará solamente con rectas. En principio, esto resulta una desventaja del método de los segundos autores frente al de los primeros, ya que obtendremos una aproximación más fiel y más compacta de la imagen con el método de Han y Fan. Por otra parte, parece razonable pensar que el emparejamiento resultará más rápido cuanto más pequeño sea el conjunto en el que tenemos que buscar las parejas. Ya que el método de Han y Fan aproxima la imagen mediante un menor número de elementos y, además, estos están distribuidos en tres conjuntos diferentes (rectas, arcos y curvas) el emparejamiento debería resultar más rápido con este método. Sin embargo, debido al método que se emplea en el algoritmo de Priestnall para aproximar poligonalmente los contornos, obtenemos informaciones adicionales como el sentido de las aristas que componen los polígonos que permitirán establecer normas de emparejamiento más estrictas que las que se empleaban en el método de Han y Fan para las rectas. Esta mayor rigidez a la hora de caracterizar una pareja provocará que se produzca un menor número de errores en el emparejamiento. Además, el conjunto de búsqueda de parejas en el algoritmo de Priestnall se reduce a los

polígonos que aproximan una componente conexa particular y no a todos los de la imagen también debido al método de extracción de contornos.

El algoritmo de Han y Fan tiene un inconveniente más debido al uso de curvas para aproximar parte de los contornos, ya que no es sencillo determinar cuando emparejan dos curvas y cuando lo hacen tampoco es fácil calcular la curva que constituirá el esqueleto. Los autores proponen métodos que toman varios puntos equidistantes a lo largo de las curvas y los van cogiendo de dos en dos calculando distancias entre ellos y, además, en el caso del cálculo de la curva central se deberá hacer un ajuste por el método de los mínimos cuadrados. Además de la complejidad que añaden estos cálculos matemáticos, el coste computacional que suponen también deberá tenerse en cuenta.

En el algoritmo de Priestnall, a medida que se van generando parejas se va componiendo una potente estructura de datos dinámica que nos permite representar las componentes conexas que componen la imagen en forma de un grafo en el que los nodos serán las áreas desemparejadas y los arcos las parejas que se han encontrado. Cada nodo, o lo que es lo mismo, cada área desemparejada, estará constituido por un polígono que describirá sus contornos y además, podremos conocer que líneas (parejas) inciden en ese desemparejamiento.

Por su parte, Han y Fan para detectar los desemparejamientos utilizan el método basado en teoría de grafos que ya comentamos en su momento, consistente en la búsqueda de ciclos de determinadas características mediante recorridos DFS con el elevado coste computacional que esto supone. Además, a pesar de que los grafos con los que se trabaja manejan información equivalente a la que contiene la estructura de contornos emparejados de Priestnall, en ningún momento se llega a obtener como resultado una estructura similar. Han y Fan utilizan la longitud de los ciclos obtenidos para determinar cuando se encuentran ante una intersección de cierto número de líneas y si hay por lo menos un par de ellas que son colineares (longitud impar) o no (longitud par).

En el presente proyecto fin de carrera se implementara un vectorizador basado en el articulo de Priestnall ya que se considera que las técnicas de emparejamiento de contornos son más adecuadas que las de adelgazamiento tradicional por las numerosas razones expuestas anteriormente. Además, se cree que la sobrecarga de

procesamiento que se tiene que asumir por el hecho de utilizar varios tipos de entidades durante el proceso de vectorización de contornos no está justificada por las ventajas de ajuste que se obtienen. Esta elección esta todavía más clara si tenemos en cuenta que en fases posteriores del proceso se puede utilizar la estructura de contornos emparejados para ayudarnos a realizar una identificación y sustitución de los segmentos rectilíneos que aproximan arcos de circunferencia. Para ello se pueden utilizar otros métodos del estilo del PBT propuesto por Dori o el utilizado por Escuderos [12] inicialmente propuesto por Brusola [14]. Además, en el articulo de Priestnall se comenta la posibilidad de utilizar esta misma estructura para detectar cabezas de flecha y otras entidades como sombreados (rayados).

El programa implementado en este proyecto utiliza el algoritmo de aproximación poligonal de contornos de Capson [1], en lugar de utilizar el algoritmo que se indica en [2] que es obra de Elliman, D.G. y P.J. Connor (1990) y se titula “CAM direct from the drawing” publicado en la 4th International Conference on Computer Aided Production Engineering en Edimburgo (p. 39 - 44) al que no hemos podido tener acceso. El algoritmo de Capson tiene unas características muy similares a las del anterior algoritmo tal y como se describen en [2]. Incluso existe la posibilidad de generar los contornos vectorizados directamente desde el escaner debido a la aplicación original para la que fue concebido el algoritmo de Capson que era el procesamiento de imágenes en tiempo real. También a causa de su origen el algoritmo tiene algunas pequeñas deficiencias que se intentarán corregir a través de una serie de modificaciones que, además, intentarán mejorar el ajuste de las aproximaciones poligonales por él generadas. Hay que mencionar que se han implementado dos filtros que trabajan sobre los contornos vectorizados para reducir el ruido de estos y para simplificarlos eliminando líneas quasi-colineares.

Tanto el algoritmo de Capson y sus modificaciones como el de emparejamiento de contornos serán analizados con detalle en posteriores capítulos.

Identificación de primitivas. En esta fase se procederá a agrupar las líneas descritas por las parejas encontradas en líneas rectas más largas y en arcos de circunferencia. En una primera fase habrá que proceder a la identificación de estas primitivas entre las líneas que estén unidas por desemparejamientos con solo dos líneas confluentes. Deberemos cerciorarnos de que las áreas desemparejadas que atravesaremos son de pequeño tamaño para no cometer errores al eliminar áreas que deban permanecer sin

emparejar ya que pertenecen a un área sólida. Habrá que tener especial cuidado de no eliminar símbolos como las cabezas de flecha que se encuentran en los extremos de algunas líneas. Una vez se han detectado las primitivas formadas por líneas separadas por desemparejamientos con solo dos líneas incidentes, habrá que tratar de extenderlas más allá de las intersecciones representadas por áreas desemparejadas con más de dos líneas incidentes.

El primer paso en el procesamiento de áreas desemparejadas será determinar la complejidad del rasgo geométrico que describen. Si el área desemparejada es un simple punto final o una esquina las líneas que inciden se prolongarán dentro de ella y se calculará la intersección. En los cruces se debe comprobar si hay alguna pareja de líneas que resulten ser colineares y en ese caso se deberán prolongar a través del área desemparejada preservando la colinearidad y evitando de este modo una de las distorsiones más comunes en el adelgazamiento tradicional. El resto de las líneas participantes en el cruce se considerará que terminan en él y se proyectarán sobre la línea considerada colinear.

También se puede tratar de encontrar rasgos característicos en las áreas desemparejadas que se producen por ejemplo en los cruces entre dos líneas que forman un ángulo muy pequeño o para eliminar la línea central que a veces aparece en el vértice de algunas esquinas muy puntiagudas.

Un punto a tener en cuenta a la hora de decidir sobre la continuidad de dos primitivas a través de un área desemparejada es el grosor medio observado en estas, ya que si se produce una variación significativa entre el grosor de ambas podremos considerarlas diferentes primitivas.

Además de esquinas, puntos finales y diferentes tipos de cruces, se puede tratar de reconocer diferentes tipos de rasgos característicos de configuraciones de líneas emparejadas y áreas desemparejadas alternadas con una frecuencia fija que pueden ayudar a la detección de rayados (sombreados) y moleteados. Estas configuraciones de líneas generalmente se distorsionan al vectorizarlas y además las líneas del borde del área sombreada dejan de ser rectas. Si detectamos la presencia de una de estas configuraciones podemos eliminar las líneas que lo forman y sustituir las líneas que rodean el área sombreada por líneas etiquetadas como pertenecientes a un área sombreada.

El método de detección y aproximación de los arcos de circunferencia puede consistir en una combinación de las técnicas de Brusola y de Dori. Ambas técnicas tienen puntos comunes como la idea de la aproximación del centro de la circunferencia a partir de las mediatrices de los lados del triángulo formado por tres puntos del arco pero a partir de este punto difieren en el método seguido para refinar la posición de este punto. Tampoco coinciden en el modo de detectar las líneas que aproximan un arco, mientras que Dori controla el incremento del valor absoluto de la variación del ángulo de los vectores consecutivos, Brusola directamente intenta aproximar el centro del arco con la intersección de las mediatrices.

El método de Dori parece más sofisticado pero hace uso de información procedente del ráster en algunas ocasiones, por su parte el método de Brusola aunque más sencillo, tiene en cuenta las uniones de los arcos con las rectas y entre ellos para tratar de mejorarlas cosa que Dori no contempla. En resumen, resulta difícil decidir a priori sobre cual de los dos métodos resultará mejor para nuestro sistema.

También existe abundante bibliografía sobre la transformada de Hough y métodos derivados de ella [34 – 40] que son habitualmente usados para la extracción de objetos de imágenes binarias. La transformada de Hough se usa para la segmentación de arcos en casos con puntos aislados que potencialmente pueden situarse formando círculos o arcos de circulo. Estos métodos no se tendrán en consideración puesto que requieren del procesamiento masivo de la información del ráster por lo que resultan demasiado costosos para grandes imágenes como algunos planos.

Reconocimiento y discriminación de conjuntos de acotación. El proceso de acotación se dividirá en tres fases. En la primera fase se segmentan las puntas o cabezas de flecha, en la segunda se asocian dichas puntas a sus líneas de cota correspondientes (formando las llamadas flechas) y por último, en la tercera fase, se asocian dichas flechas a las líneas auxiliares de cota y a las cifras de cota, formando los conjuntos de acotación.

La primera fase comenzará con la detección de un desemparejamiento producido por un engrosamiento repentino de la línea que hará de suceso disparador tal y como indica Priestnall [2]. A continuación se construirá un modelo de la flecha y se ajustará de acuerdo a la posición del disparador y después se utilizará este modelo

como mascara para hacer una operación de AND lógico con los bits del bitmap. Se contará el número de pixeles que coinciden y si supera el 75% consideraremos que nos encontramos ante una flecha.

En la segunda fase comenzaremos considerando las líneas discontinuas del siguiente modo: aquellas que no presenten discontinuidades uniformes (raya-punto-raya o raya-punto-punto-raya, ...) serán catalogadas como líneas no pertenecientes a la geometría del objeto (las correspondientes a las trayectorias y a los ejes), las que sí presenten un tipo de discontinuidad más uniforme (raya-espacio-raya) se considerarán como líneas pertenecientes a la geometría del objeto (líneas ocultas).

Tanto Kasturi [6, 7] como Langrana [16 - 19] utilizan en sus sistemas el algoritmo desarrollado por el primero para la detección de líneas discontinuas. Las explicaciones que dan en los artículos citados tienen carácter complementario. El algoritmo que comentan permite distinguir entre líneas discontinuas formadas por diferentes patrones: líneas discontinuas con todos los segmentos de igual longitud o líneas discontinuas que alternan segmentos cortos y largos. Las líneas discontinuas que se detectan no tienen porque ser rectas. No parece haber ningún problema para adaptar este método a nuestro sistema, aunque parece que el coste temporal del algoritmo será bastante elevado.

A partir de este punto, el articulo de Langrana [19] parece constituir una aproximación consistente al problema y parece que se podría adaptar adecuadamente a las necesidades de nuestro sistema. Además, el mismo autor en sus artículos [17, 18] completa el sistema con módulos que interpretan la información contenida en estos conjuntos de acotación y la emplea para complementar la forma de los objetos representados en el dibujo.

3. Vectorizador por emparejamiento de contornos.

La Figura 2 muestra un diagrama en el que se aprecian todas las fases en las que se descompone el vectorizador que se va a desarrollar en este proyecto fin de carrera.

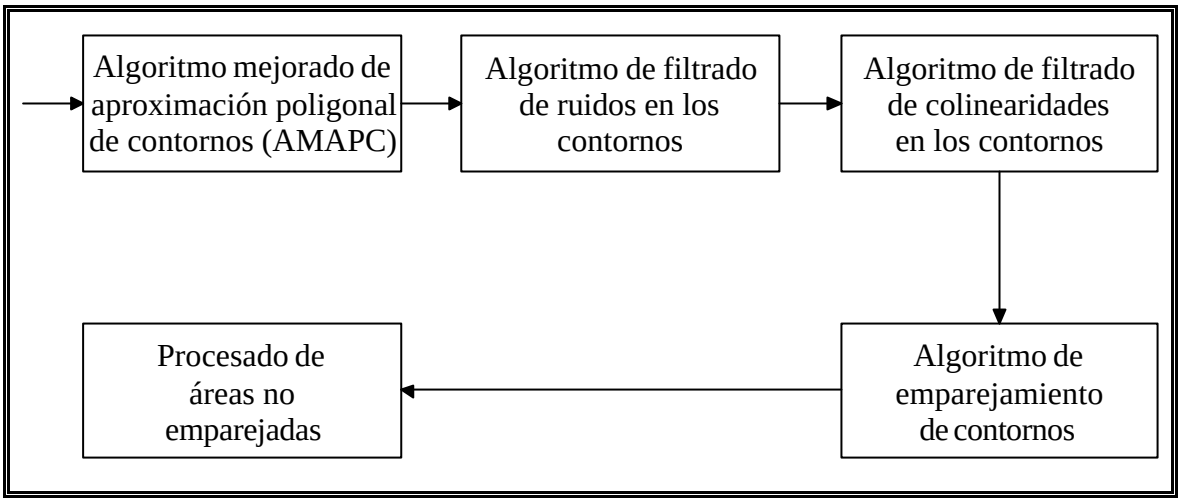


Figura 2

La primera fase corresponderá con la generación de polígonos que aproximen los contornos de las figuras. Para realizar esta operación se ha creado un algoritmo basado en el de Capson [1] que pretende obtener las citadas aproximaciones poligonales.

Los polígonos de contorno generados por el anterior algoritmo tienen el problema de verse bastante afectados por el ruido que se produce frecuentemente en los contornos de las figuras. Un contorno con numerosos pequeños salientes y entrantes de un pixel de altura generará un polígono con gran número de aristas que aportarán información superflua. El algoritmo de filtrado de ruido en los polígonos intentará reducir los efectos causados por este tipo de contornos.

Muchas de las aristas que aparecen en los polígonos de los contornos podrían considerarse colineares con las que se encuentran a continuación de ellas si relajáramos ligeramente el concepto de colinearidad considerando colineares aquellas aristas que formen un ángulo llano o uno próximo a este. Aplicando el algoritmo de filtrado de colinearidades en los contornos, pretendemos eliminar las aristas redundantes de los polígonos aproximadores sin que estos sufran deformaciones de relevancia.

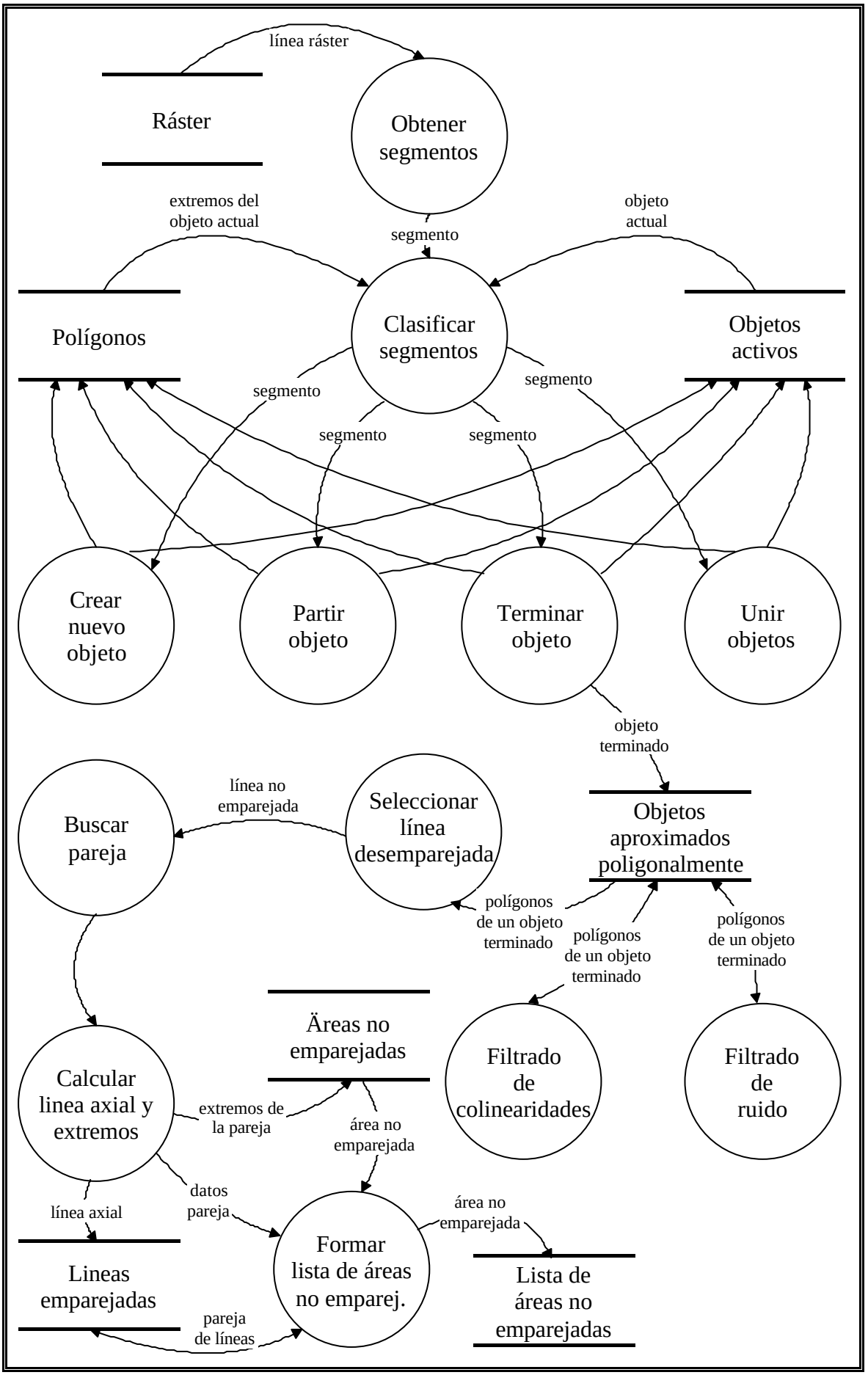
Una vez concluida la fase de filtrado tendremos todos los polígonos que aproximan los contornos de una figura de la imagen simplificados. Hay que destacar

que el filtrado de una figura se produce inmediatamente que se conocen todos los polígonos que aproximan sus contornos, sin tener que esperar a que se terminen de procesar el resto de los polígonos pertenecientes a otras figuras. Procediendo de este modo, podemos liberar la memoria que utilizaban las aristas superfluas de los contornos y dejarla disponible para el resto de las figuras que todavía se están procesando.

Siguiendo con el diagrama, cuando los contornos de una figura han sido simplificados, ya se encuentran listos para proceder a la identificación de las aristas que se encuentran a ambos lados del contorno de una línea. Al proceso que hace corresponder dos de estas aristas situadas en contornos de una misma línea enfrentados, lo denominaremos emparejamiento. El algoritmo de emparejamiento de contornos que se ha implementado está basado en el descrito en el articulo [2]. Este algoritmo produce una potente estructura de datos que contendrá la parte más significativa de la información contenida en el ráster de la imagen original. La estructura consistirá en un grafo en el que los arcos representarán las rectas que simbolizan las parejas detectadas en el proceso de emparejamiento y los nodos serán las áreas que no han emparejado. Todas las aristas formarán parte de una lista enlazada y se conocerá el grosor medio de cada línea. Las áreas desemparejadas estarán también incluidas en otra lista enlazada. El contorno de cada área no emparejada estará aproximado mediante un polígono.

La última fase del procesamiento de una figura debería ser la identificación de primitivas dentro del grafo de la figura tales como rectas, circunferencias y arcos de circunferencia. Una vez identificados deberíamos eliminar de la estructura de datos todas las parejas y áreas desemparejadas y sustituirlas por las primitivas que las aproximan. Todo este proceso resulta bastante complicado y por causas de tiempo y sólo a título experimental, ha sido sustituido por un sencillo algoritmo que se encarga de procesar los desemparejamientos en los que inciden únicamente dos parejas. El proceso consiste en calcular la intersección de las dos líneas incidentes y por tanto eliminar el área del desemparejamiento.

Todo el proceso se puede contemplar desde el punto de vista de los datos en el siguiente diagrama de flujo de datos que muestra una versión muy simplificada del sistema.


línea ráster
Ráster
Obtener segmentos
extremos del objeto actual
segmento
objeto actual
Polígonos
Clasificar segmentos
Objetos activos
segmento
segmento
segmento
segmento
Crear nuevo objeto
Partir objeto
Terminar objeto
Unir objetos
objeto terminado
Buscar pareja
línea no emparejada
Seleccionar línea desemparejada
Objetos aproximados poligonalmente
polígonos de un objeto terminado
polígonos de un objeto terminado
polígonos de un objeto terminado
Äreas no emparejadas
Filtrado de colinearidades
Filtrado de ruido
Calcular linea axial y extremos
extremos de la pareja
área no emparejada
datos pareja
línea axial
Formar lista de áreas no emparej.
área no emparejada
Lineas emparejadas
Lista de áreas no emparejadas
pareja de líneas

4. Extracción de contornos.

4.1 Introducción.

El presente capítulo presenta el algoritmo desarrollado por David W. Capson para la extracción de contornos [1] y .las modificaciones que se le han realizado.

El algoritmo toma una imagen binaria en formato ráster y produce una lista ordenada de coordenadas cartesianas que definen una aproximación poligonal de todos y cada uno de los contornos que aparecen en ella. El método opera con una única línea de la imagen y la ejecución es secuencial en una única pasada de derecha a izquierda en un tiempo de O(n), donde n es el número de transiciones entre figura y fondo detectadas en la línea. Aunque para nuestros propósitos nos resulta indiferente, el algoritmo está especialmente indicado para realizar la extracción de contornos sobre la marcha sobre un ráster generado automáticamente por una cámara o un dispositivo similar puesto que para que esto sea posible debe de completarse el procesamiento de una línea como máximo en el tiempo que dura el refresco horizontal del ráster. Dado un tiempo máximo que será el tiempo de refresco del ráster, se puede especificar el número mínimo de puntos de transición para los que se garantiza el procesamiento, es decir que podemos fijar la anchura del ráster.

El algoritmo usa una estructura de datos dinámica que se actualizará a medida que se recorre el ráster simplemente con cambiar un número fijo de punteros de esta estructura. La imagen del ráster puede incluir cualquier número de objetos, cada uno de los cuales puede contener cualquier número de agujeros. No existen restricciones en cuanto a complejidad: los objetos se pueden entrelazar, salirse por los bordes de la imagen, o aparecer como “islas” dentro de agujeros de otros objetos, además esto último puede producirse con cualquier grado de anidamiento. Este método también se amolda a objetos que consisten en un simple pixel o contienen protrusiones o agujeros de una anchura de un pixel. No hay necesidad de almacenar la imagen completa; solo se requieren los puntos de transición de la línea del ráster que esté siendo procesada en ese momento. Los contornos de las figuras estarán enlazados en sentido antihorario y los contornos de los agujeros en sentido horario y en cualquier caso cierto tipo de puntos colineares será eliminado.

4.2 Preliminares

Damos por sentado que las imágenes a procesar han sido digitalizadas en un ráster y han sido procesadas para obtener siluetas sobre un fondo de color diferente. Además, la imagen binaria deberá ser codificada en series de puntos consecutivos para proporcionar las coordenadas x de los puntos de transición entre las siluetas y el fondo en cada una de las líneas de la imagen. De este modo, la imagen binaria original es aproximada línea a línea con pares de puntos de transición como ilustra la Figura 3. Cualquier par de puntos de transición sucesivos de la misma línea definen un *segmento* y las coordenadas (x, y) de esos puntos y serán aludidos como *puntos de contorno*. Un *punto izquierdo de contorno* tendrá la menor coordenada x del par de puntos; un *punto derecho de contorno* tendrá la mayor. El número de línea de la imagen se tomará como la coordenada y. Por ejemplo, en la Figura 3-B los segmentos válidos se especificarán como $[x_1, x_2]$, $[x_3, x_4]$, $[x_5, x_6]$, $[x_7, x_8]$, etc. Los puntos de contorno correspondientes al segmento $[x_1, x_2]$ son (x_1, y_i) y (x_2, y_i). Para el segmento $[x_3, x_4]$ son (x_3, y_{i+1}) y (x_4, y_{i+1}) y así sucesivamente.

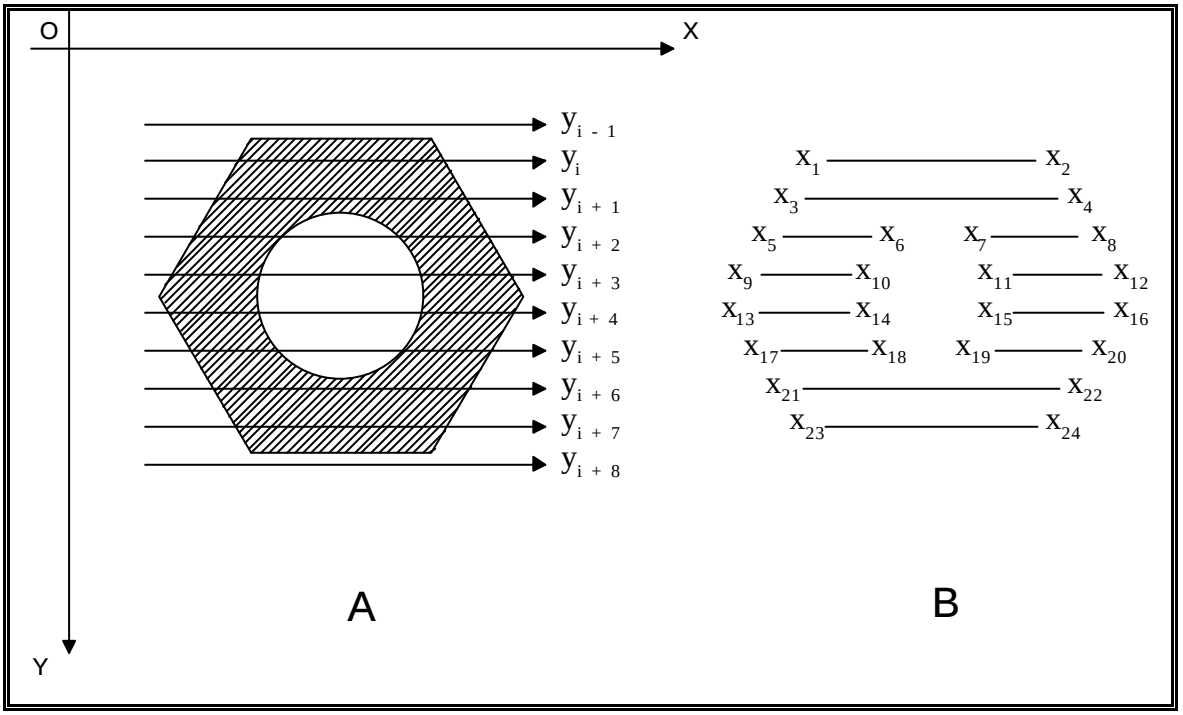


Figura 3

Un *área de objeto* es un conjunto de segmentos conectados los cuales son del color distinto al fondo. Áreas de hueco son conjuntos de segmentos conectados del mismo color del fondo pero completamente rodeados por un área de objeto. Dos segmentos se consideran conectados si se encuentran en líneas consecutivas y hay un

solapamiento entre los intervalos sobre los que se extienden en el eje x. Cuando cualquier porción de un área de objeto aparece en la línea del ráster que esta siendo procesada, ese objeto se dice que es un *objeto activo.* En el momento en que aparece una línea que no contiene ninguna parte de un objeto que previamente ha sido activo, pasamos a denominar ese objeto como *objeto terminado*.

Los contornos del área de objeto de nuestro ejemplo está definida por la lista ordenada de puntos de contorno (x_1, y_i), (x_3, y_{i+1}), ..., (x_2, y_i). Del mismo modo, el contorno del área de hueco es (x_7, y_{i+2}), (x_{11}, y_{i+3}), ..., (x_6, y_{i+2}). Esos puntos pueden ser considerados como los vértices de una aproximación poligonal de un área y obviamente los puntos colineares pueden ser eliminados de la lista de puntos de contorno sin perdida alguna de información sobre la forma. Los puntos restantes se denominarán *vértices* y en la Figura 4-a se muestra un ejemplo.

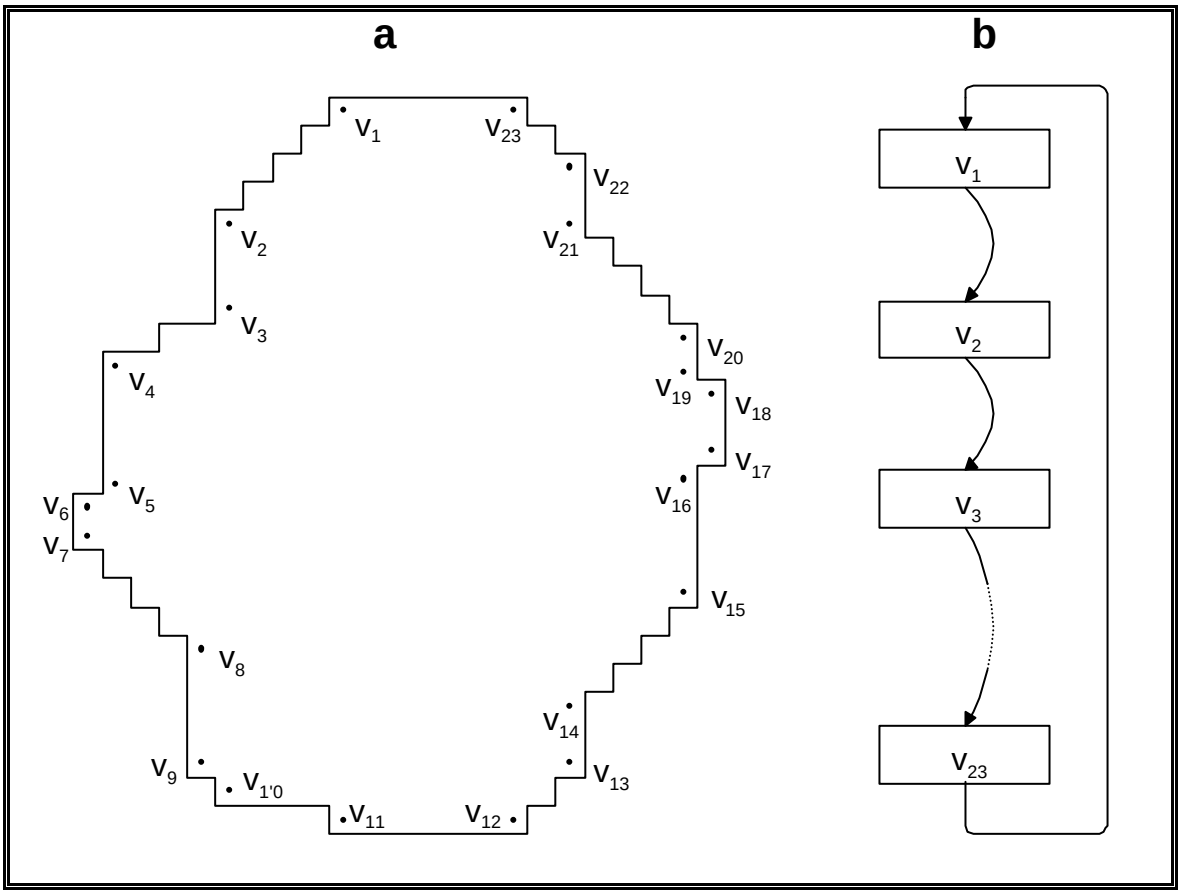


Figura 4

El número de vértices necesarios para representar un contorno es siempre menor o igual al número de puntos de contorno. En la práctica el número de vértices es considerablemente menor que el número de puntos de contorno. Es conveniente representar los vértices en una lista enlazada de coordenadas (x, y) como la que se

muestra en la Figura 4-b. A esta estructura se le llamará *lista de vértices*. Los vértices se listarán de acuerdo a los siguientes estándares:

Los vértices de las áreas de objeto estarán enlazados en sentido antihorario y uno de estos vértices será considerado el vértice inicial. Este vértice será el punto izquierdo de contorno del segmento que teniendo mayor coordenada y esté más a la derecha (v_{11} en la Figura 4).

Las áreas de hueco estarán enlazadas en sentido horario. El punto de inicio para una lista de vértices de hueco estará en el punto de mayor coordenada y con la coordenada x de más a la derecha (v12 en la Figura 4).

4.3 Descripción del algoritmo y las estructuras de datos.

El algoritmo acepta como entrada el conjunto de segmentos que han sido detectados para una única línea de la imagen estando ya procesados los segmentos de la línea anterior. Por lo tanto, la tarea fundamental es decidir sobre la conexión que se debe hacer entre los puntos de contorno de ambas líneas. Esto se puede lograr teniendo en cuenta los segmentos de la línea anterior secuencialmente de derecha a izquierda (tal y como aparecían en el ráster) y procesando los de la nueva línea en el mismo orden. Tal y como se puede ver en la Figura 5 para la anterior línea (y_{k-1}), el segmento que se está considerando en este momento estará determinado por su punto izquierdo de contorno a, su punto derecho de contorno b y el punto izquierdo de contorno del siguiente segmento c. Del mismo modo, el segmento que se está teniendo en cuenta en la nueva línea (y_k) vendrá determinado por los puntos de contorno d, e y f. Puesto que todas las conexiones para a, b, c, d, e y f se determinan antes de que se avance a los siguientes segmentos de cada línea, no hay nunca necesidad de considerar los puntos de contorno de segmentos previos al [a, b] o al [d, e]. Si un segmento es el último en cualquiera de las dos líneas entonces c (o f) se fija a un valor que es más grande (está más a la derecha) que cualquier coordenada de una línea.

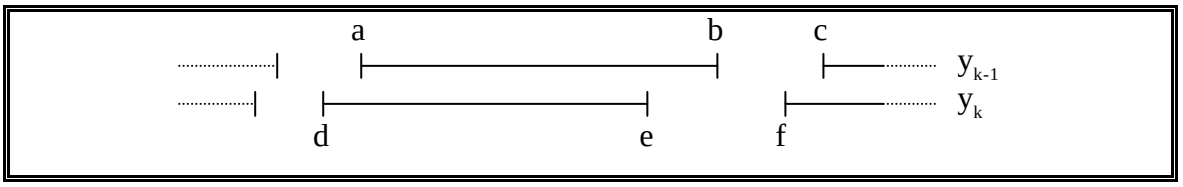


Figura 5

La Tabla 2 muestra las condiciones que se aplicaran para identificar la situación que se detecta en cada momento del proceso. Dependiendo de la situación detectada en

un momento dado, deberemos establecer conexiones entre los puntos correspondientes con a, b, c, d y e para simbolizar la aparición de líneas que constituirán los lados de los polígonos que aproximarán las figuras.

Condición	**Interpretación**
e < a	El segmento [d, e] es el comienzo de un nuevo objeto de la imagen. Conectar e con d.
d < b	El segmento [a, b] es la terminación de un objeto existente en la imagen. Conectar a con b.
Ninguna de las anteriores	El segmento [d, e] se solapa con el segmento [a, b]. Conectar a con d.
f < b	El segmento [a, b] se ha partido y [e, f] podría ser el comienzo de un hueco. Conectar e con f.
e > c	El segmento [d, e] se está uniendo con el segmento [a, b] y el siguiente segmento y [b, c] pueden ser la terminación de un agujero. Conectar c con b.
Ninguna de las anteriores	Conectar e con b.

Tabla 2

La estructura *buffer de líneas* se usa para mantener los puntos de contorno de la línea que está apunto de ser analizada. El número máximo de puntos que se almacenarán será igual al número de pixeles en una línea de la imagen. También deberemos almacenar el número de la línea que está siendo procesada en ese momento. Los punteros d y e y f contendrán los índices de las posiciones que ocupan en el buffer los puntos correspondientes. La coordenada x de cada uno de estos puntos vendrá dada por el valor almacenado en el elemento del buffer indicado por su correspondiente puntero, mientras que la coordenada y vendrá dada por el número de la línea que en ese momento se está procesando y que previamente habíamos almacenado. La *lista de vértices* será una estructura en la que se almacenarán los vértices que vayamos obteniendo durante el proceso. Cada elemento de esta estructura contendrá además de las coordenadas de estos vértices un campo que permitirá enlazar con el siguiente vértice dentro de la lista. Para conectar un vértice con el siguiente bastará con cambiar

el valor de este campo para que apunte a la posición de la lista donde se encuentra el siguiente vértice.

En este momento ya podemos escribir el algoritmo en pseudo-código que nos permitirá hacernos una idea del funcionamiento:

```
Calcular_contornos()
        Inicializar estructuras
        Mientras queden líneas en la imagen
                Apuntar a, b y c al comienzo de los segmentos de la línea anterior.
                Leer los segmentos de la nueva línea.
                Apuntar d, e y f al comienzo de los nuevos segmentos.
                Mientras queden segmentos en la línea anterior o en la actual
                Si e < a
                        Crear_nuevo_objeto()
                sino
                        Si d > b
                                Terminar_objeto()
                        sino
                                Conectar a con d.
                                Mientras (f <= b) o (e >= c)
                                        Si f <= b
                                                Partir_objeto()
                                        sino

                                        Fsi
                                Fmientras
                                Conectar e con b.
                                Avanzar hasta el siguiente segmento en la línea anterior.
                                Avanzar hasta el siguiente segmento en la nueva línea.
                        Fsi
                Fmientras
        Fmientras
Fcalcular_contonos

Crear_nuevo_objeto()
        Conectar e con d.
        Actualización_por_creación()
        Avanzar hasta el siguiente segmento en la nueva línea.
FCrear_nuevo_objeto

Terminar_objeto()
        Conectar a con b.
        Actualización_por_terminación()
Avanzar hasta el siguiente segmento en la línea anterior.
FTerminar_objeto()

Partir_objeto()
        Conectar e con f.
        Avanzar hasta el siguiente segmento en la nueva línea.
        Actualización_por_partición()
FPartir_objeto
```

Unir_objetos()
Conectar c con b.
Actualización_por_unión()
Avanzar hasta el siguiente segmento en la línea anterior.
FUnir_objetos

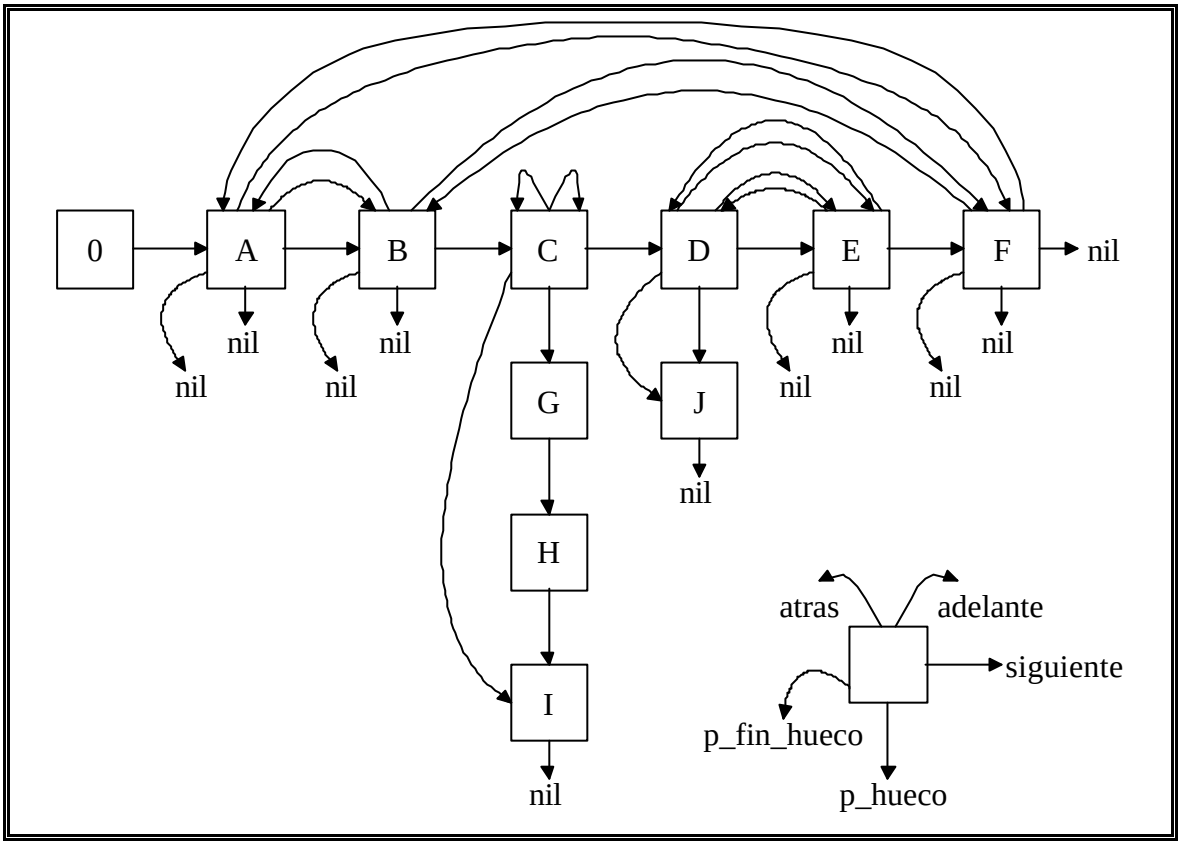


Figura 6

La *lista de objetos activos* será una estructura de datos que tal y como se muestra en la Figura 6 representa dinámicamente los objetos de la imagen que están activos en un momento determinado y las relaciones que existen entre ellos. Hay cinco campos de enlace en cada nodo:

siguiente. Es el puntero al siguiente nodo de la lista de objetos activos. La ordenación de izquierda a derecha preserva el orden en el que aparecieron los objetos en la línea anterior.

p_hueco. Apunta al nodo que representa el primer agujero en la lista de agujeros de un objeto activo. En los nodos de hueco *p_hueco* apuntará al siguiente hueco en la lista de huecos.

p_fin_hueco. Se usa para apuntar al último hueco de la lista de huecos de un objeto activo. Se utiliza para facilitar la concatenación de dos listas de hueco.

atrás, adelante. Se usan para conectar nodos de los que se conoce que pertenecen a la misma área de objeto. Cuando un nodo se parte su puntero *adelante* apunta al nuevo nodo creado y el puntero *atrás* del nuevo nodo se apunta sobre el nodo original. Los enlaces *adelante* y *atrás* forman una lista circular de nodos doblemente enlazada dentro de la lista definida por los enlaces *siguiente.*

Además de los enlaces, en cada nodo se definen los siguientes campos:

izquierda, derecha. Serán índices que apuntarán a la posición en la lista de vértices de los puntos de contorno izquierdo y derecho de los segmentos de un objeto activo que apareció en la línea anterior. Cuando los nodos de objeto se mueven a la lista de objetos terminados, el último valor del campo izquierdo se usará como punto de inicio del contorno.

deltax_izq, deltax_der. En estos campos se almacenará el cambio en la coordenada x de los puntos de contorno izquierdo y derecho entre dos líneas consecutivas. Cuando esos cambios se calculen en la línea actual se compararán con los anteriores *deltax_izq* y *deltax_der*. Si alguno de los dos coincide entonces estaremos ante un punto colinear que no hará falta introducir en la lista de vértices.

Los punteros *objeto_actual* y *objeto_anterior* están al tanto de la posición de los nodos de la lista de objetos activos que representan el objeto actual y el objeto anterior respectivamente.

En el ejemplo mostrado en la Figura 6 hay 6 objetos activos (A, B, C, D, E, F). Se han detectado 3 huecos para el objeto C (G, H, I) y uno en el objeto D (J). Las listas de huecos están enlazadas mediante el campo p_hueco. El objeto A se ha partido formando el objeto B el cual a su vez también se ha partido para formar el objeto F. De un modo parecido, el objeto D se ha partido dando el objeto E. Los objetos B y F representan huecos potenciales en el objeto A, del mismo modo que E dentro del objeto D. Además, los objetos C y D se encuentran entre los objetos B y F.

El primer nodo de la lista de objetos activos (0) es un nodo ficticio que se añade a la estructura porque permite simplificar la inserción y el borrado al principio de la lista. El campo siguiente del nodo ficticio se considera que apunta al primer objeto

activo. Los procedimientos para manipular las estructuras de datos de la lista de objetos activos se perfilan en la siguiente sección.

Sobre la lista de objetos activos se definirán tres operaciones básicas: **Insertar(**ptr**)**, **Borrar(**ptr**)** y **Concatenar(**ptr_nodo, ptr_primero, ptr_ultimo**)**. La primera operación inserta un nodo en la lista de objetos activos a la derecha del nodo al que apunta ptr. La segunda operación borra un nodo de la lista de objetos activos; ptr apunta a un nodo cuyo campo *siguiente* apuntará al nodo que será borrado. La última operación concatenará la lista definida por ptr_primero, ptr_ultimo al final de la lista de huecos del objeto apuntado por ptr_nodo.

4.3.1 Actualización por partición.

Cuando un área de objeto se parte (f < b) un nodo se insertará en la lista de objetos activos inmediatamente a la derecha del nodo actual. Los punteros *adelante* y *atrás* se ajustarán (como se puede ver en la Figura 7-b) para indicar que se conoce que los dos nodos forman parte de la misma área de objeto. En este momento se desconoce si la partición es el comienzo de un área de hueco o no. Hasta el momento en que se determine esto los dos nodos se tratarán como objetos separados.

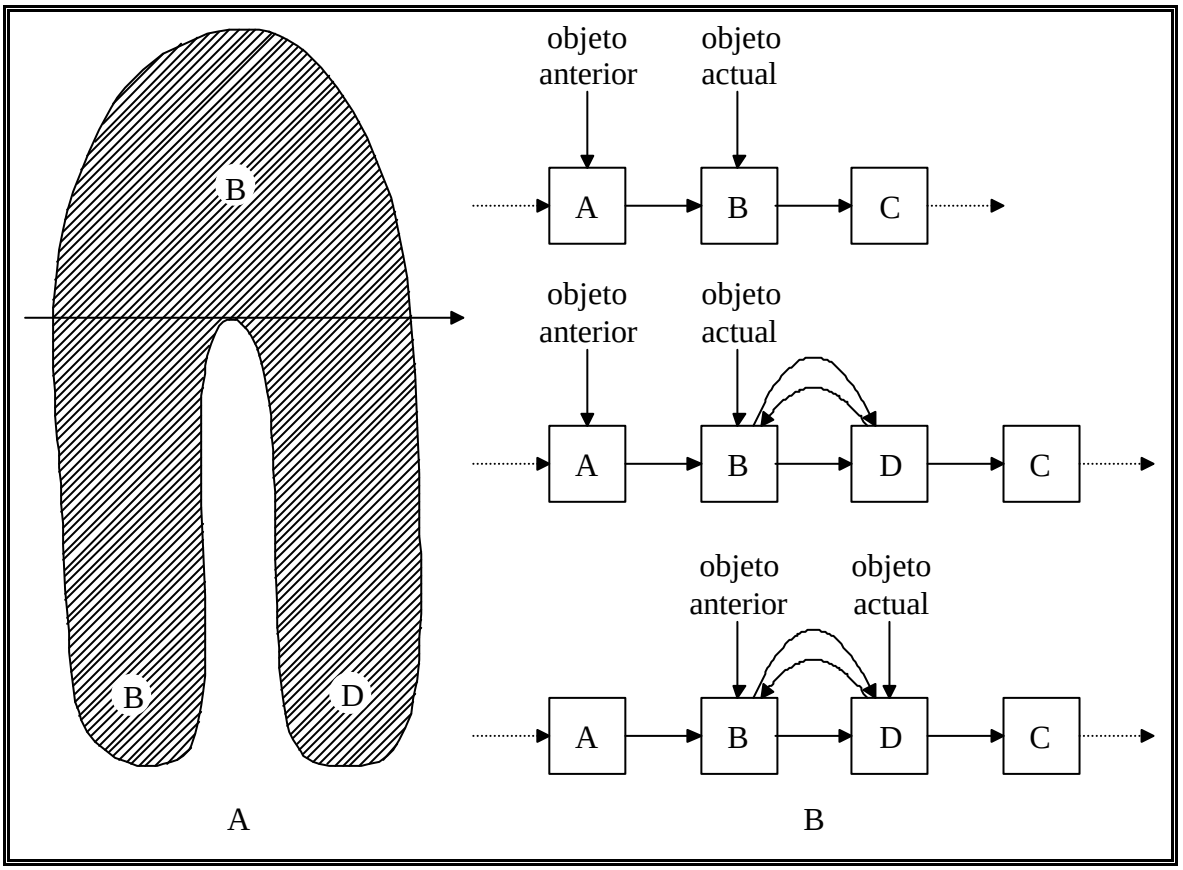


Figura 7

Actualización_por_partición()
Insertar(objeto_actual **)**
Fijar los punteros atrás y adelante.
objeto_anterior = objeto_actual
objeto_actual = objeto_actual.siguente
FActualización_por_partición

4.3.2 Actualización por unión.

Una operación de unión se realiza siempre entre el nodo actual y su inmediato vecino de la derecha en la lista de objetos activos.

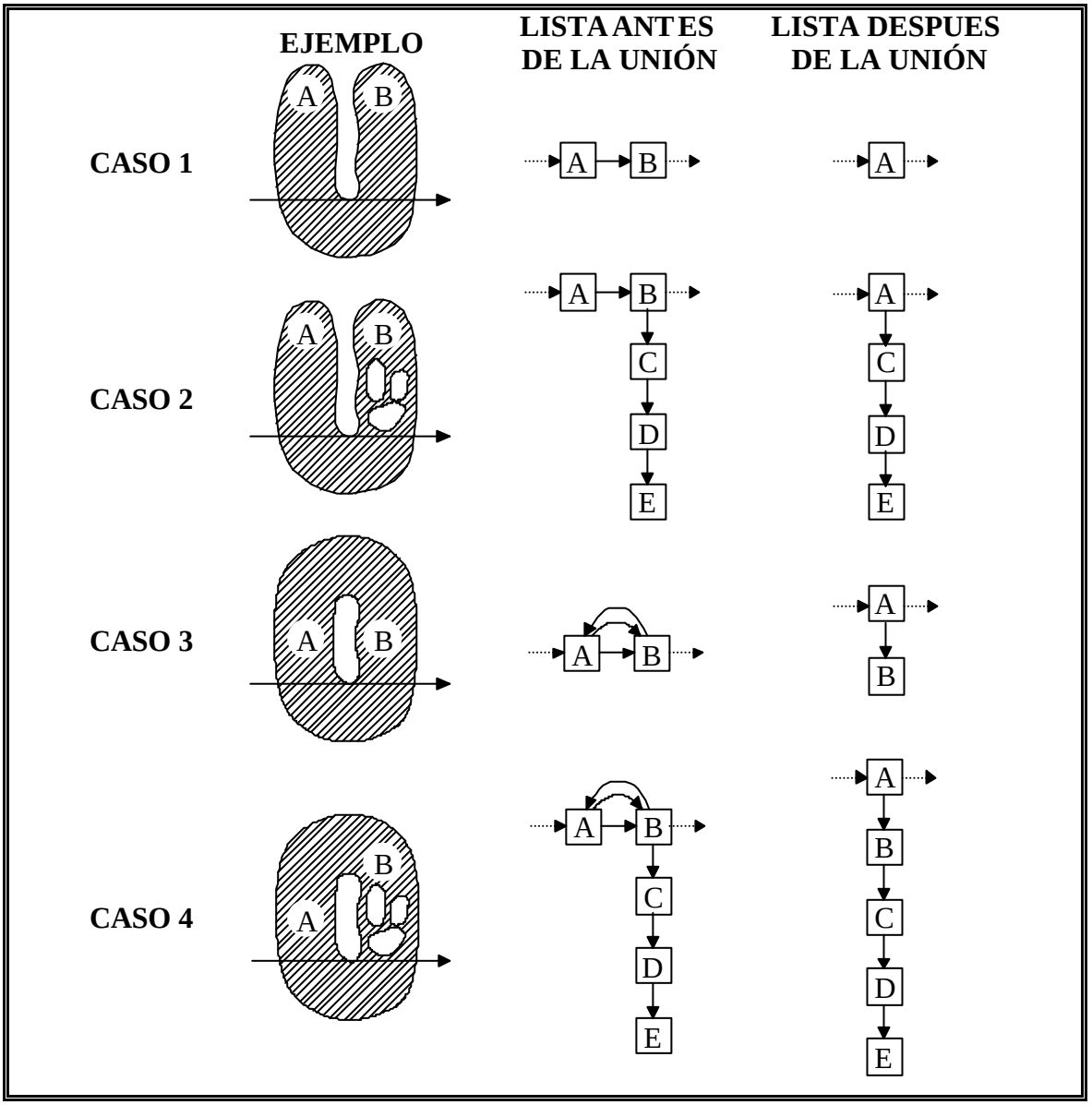


Figura 8

Hay cuatro casos a considerar en una unión que están ilustrados en la Figura 8:

Caso 1. El nodo derecho era un área de región distinta.

Caso 2. El nodo derecho era un área de región diferente con huecos.

Caso 3. El nodo derecho era el resultado de partir el nodo izquierdo. Por tanto el nodo derecho representa un área de hueco contenida en el nodo izquierdo.

Caso 4. El nodo derecho era el resultado de partir el nodo izquierdo pero este último también contenía una lista de huecos.

```
Actualización_por_unión()
    Si el nodo siguiente proviene de partir el objeto_actual (Caso 3 y Caso 4)
        Unir el nodo siguiente y su lista de huecos a la lista de huecos del objeto_actual.
    sino
        Si el nodo siguente tiene una lista de huecos
            Unir la lista de huecos del nodo siguiente a la del objeto_actual.
        Fsi
    Fsi
    Eliminar el nodo siguiente de la lista adelante/atrás.
    Copiar el campo derecha del nodo siguiente al campo derecha del objeto_actual.
    Borrar( objeto_actual )
Factualización_por_unión
```

4.3.3 Actualización por creación.

Cuando se encuentra un objeto por primera vez en la imagen, se crea un nodo en la lista de objetos activos. La condición (e < a) para detectar el comienzo de un nuevo objeto implica que el nuevo objeto esta situado a la derecha del objeto actual. Esto significa que debe ser insertado a la inmediatamente a la derecha del nodo que está inmediatamente a la izquierda del objeto actual. De este modo, la misma rutina puede utilizarse para insertar a la izquierda o a la derecha. Una vez que el nodo se ha insertado, el puntero al objeto anterior debe adelantarse (Ver Figura 9).

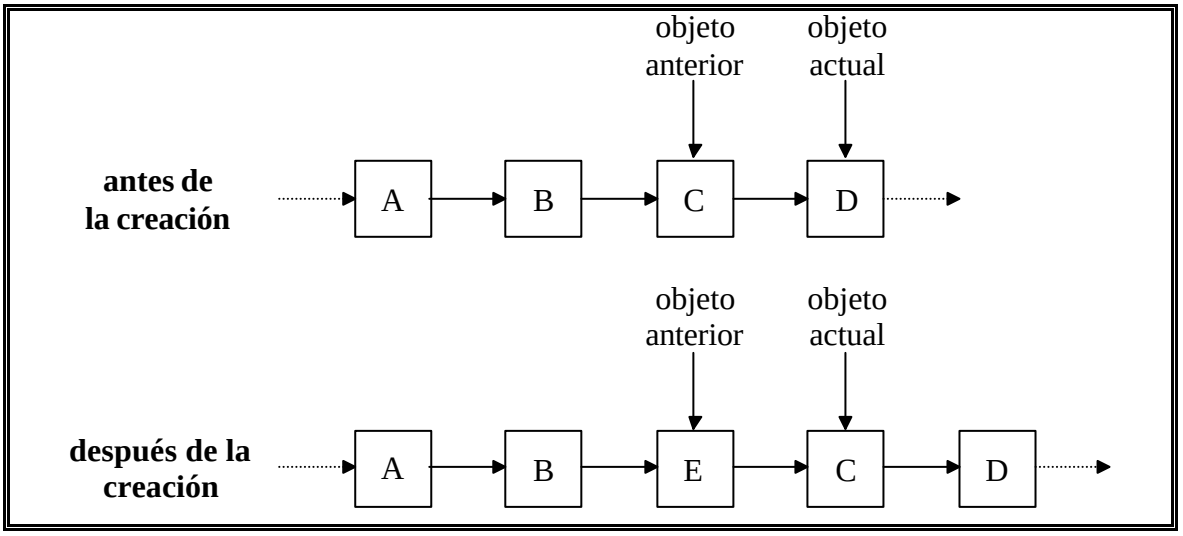


Figura 9

```
Actualización_por_creación()
        Insertar( objeto_anterior )
Objeto_anterior = objeto_anterior.siguiente
FActualización_por_creación()
```

4.3.4 Actualización por terminación.

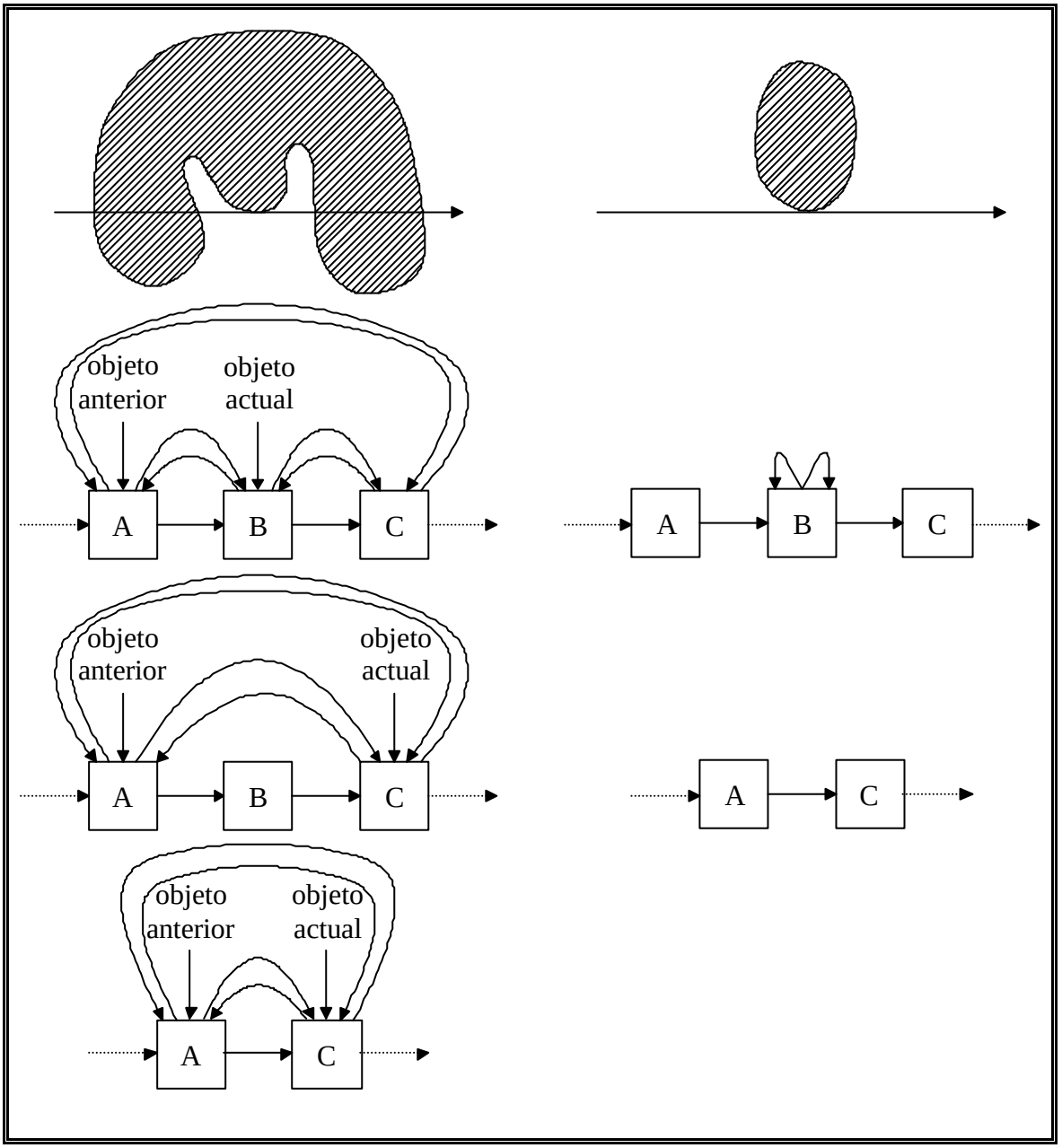


Figura 10

La condición d > b indica la terminación de un nodo activo. En general, hay dos casos a considerar:

Caso 1. Aunque un nodo ha terminado, el objeto está todavía activo (Figura 10 izquierda). Esto se detecta chequeando si el nodo que termina se ha partido o es

el resultado de una partición, esto es, chequeando los campos *adelante* y *atrás*. En caso de que el nodo que termina tenga huecos se debe concatenar su lista de huecos a la de un nodo que se conserve.

Caso 2. El área de objeto está terminando (Figura 10 derecha). Hay que quitar el nodo de la lista de objetos activos que representa al objeto y se debe añadir una nueva entrada en la lista de objetos terminados que apunte a este nodo.

```
Actualización_por_terminación()
        Extraer el nodo de la lista enlazada adelante/atrás
        Si el área es completa (Caso 2)
                Añadir un puntero al nodo en la lista de objetos terminados.
        Sino
                Si el nodo contiene algún hueco
                        Añadir su lista de huecos a la del nodo que se conserve.
                Fsi
        Fsi
        Borrar( objeto_anterior )
        objeto_actual = objeto_anterior.siguente
FActualización_por_terminación()
```

4.3.5 La lista de objetos terminados.

La lista de objetos terminados almacenará la información de los nodos de la lista de objetos activos que se conoce que representan objetos completos. Tras todo el proceso, cada uno de estos nodos en su campo *izquierda* contendrá un índice que señalará la posición de la lista de vértices en la que empieza la lista que corresponde con ese contorno. Además, su puntero p_hueco apuntará a la lista de huecos que se han encontrado dentro de ese objeto completo en particular.

4.3.6 Ejemplo.

En la Figura 11-A podemos ver una sencilla imagen que nos ayudará a ilustrar el funcionamiento del algoritmo. Los números indican el número del nodo que representa a esa parte del objeto en el esquema de la lista de nodos activos de la Tabla 3. En la parte derecha (Figura 11-B) podemos ver la misma imagen una vez se ha discretizado y convertido en ráster. Los puntos negros indican que en ese punto del barrido de la imagen se ha detectado la presencia de un objeto y por tanto ese pixel del ráster será del color distinto al del fondo. La silueta de la figura una vez discretizada aparece representada como la línea escalonada que rodea a los puntos negros del objeto en la

Figura 11-B y pretende representar la apariencia que tendría el ráster si representáramos cada uno de sus puntos como un cuadrado.

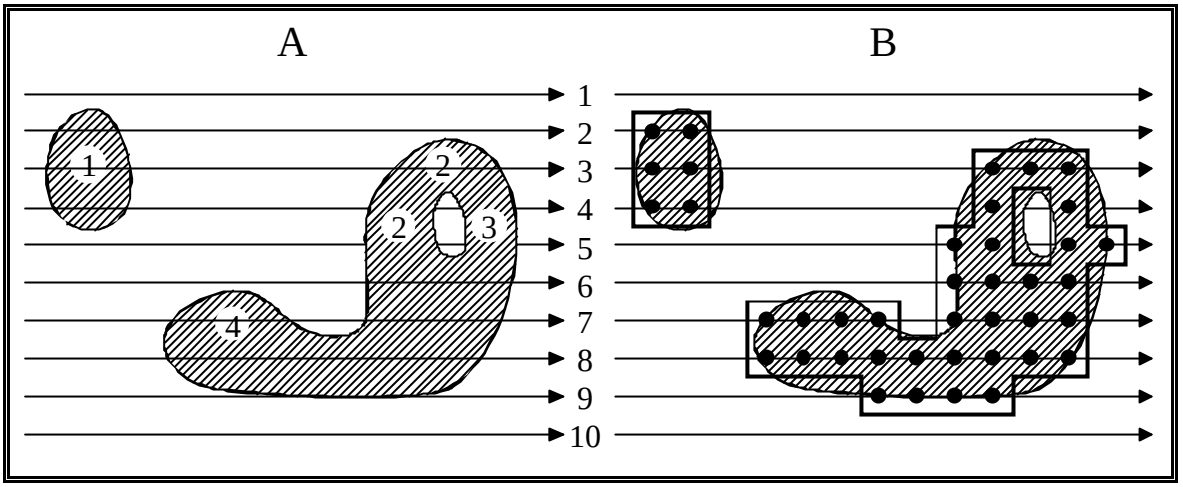


Figura 11

En la Tabla 3 aparece la evolución que va sufriendo la lista de objetos activos a medida que se va procesando la imagen de la Figura 11 línea por línea. Al final del proceso la lista de objetos terminados contendrá dos entradas una por cada área de objeto y además el nodo correspondiente a la segunda de estas áreas contendrá un puntero a un tercer nodo que representará el hueco de esta. Cada uno de estos tres nodos contendrá un índice a la posición de la lista de vértices en donde se encuentra el primer vértice de la lista correspondiente a cada contorno.

Línea	**Lista de objetos activos**	**Comentario**
1	objeto_anterior → Ficticio → nil	La lista de objetos activos se inicializa del modo mostrado. Como la linea número 1 no contiene segmentos no hay cambios en la estructura. objeto_siguiente = nil
2	Ficticio → 1 → nil	Se crea el objeto 1.
3	Ficticio → 1 → 2 → nil	Se actualiza el objeto 1. Se crea el objeto 2.
4	Ficticio → 1 → 2 → 3 → nil	Se actualiza el objeto 3. El objeto 2 se parte produciendo el objeto 3 (un hueco potencial).

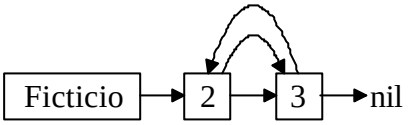
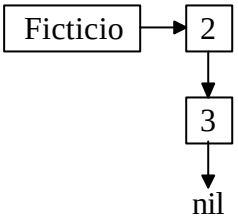
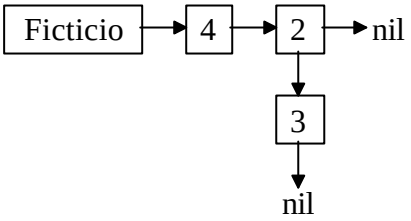
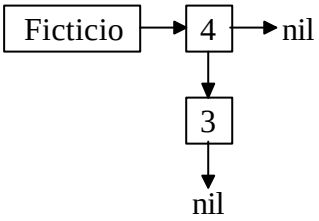
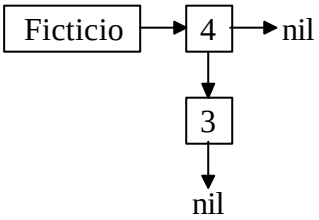
5		El objeto 1 termina y se añade a la lista de objetos terminados. Se actualizan los objetos 2 y 3.
6		Se unen los objetos 2 y 3. Ya que el objeto 3 era un agujero potencial en el objeto 2, ahora se convierte en un nodo de hueco en el objeto 2.
7		Se crea el objeto 4. Se actualiza el objeto 2.
8		Se unen los objetos 4 y 2. La lista de huecos del objeto 2 se une a la del objeto 4 y el objeto 2 termina y es descartado.
9		El objeto 4 es actualizado.
10		Termina el objeto 4 y es añadido a la lista de objetos termiandos. La lista de objetos terminados contiene ahora punteros a los nodos 1 y 4. El nodo 4 contiene un puntero a su hueco.

Tabla 3

4.3.7 Conclusiones.

El algoritmo de extracción de vértices es un proceso totalmente secuencial para obtener contornos de áreas en imágenes binarias mediante una única pasada sobre el ráster. La eficiencia en la ejecución se obtiene manipulando punteros en una estructura de datos dinámica. Una vez obtenidas, las estructuras de datos permiten que las listas de vértices sean fácilmente postprocesadas a gran velocidad para determinar informaciones

tales como área, perímetro, momentos, curvatura, cajas de circunscripción, etc. Es probable que algunas de estas informaciones puedan ser extraídas durante el proceso añadiendo campos adicionales a la lista de objetos activos.

4.4 Modificaciones al algoritmo de Capson

El algoritmo anteriormente expuesto consigue de un modo bastante simple una aproximación poligonal de los contornos de la figura. Esta simplicidad está justificada por el hecho de que el algoritmo fue inicialmente diseñado para aplicaciones de procesamiento de vídeo en tiempo real y además se pretendía realizar una implementación del mismo usando hardware dedicado.

En algunas ocasiones esta simplicidad provoca inconvenientes para el uso que pretendemos dar a este algoritmo. Dado que nuestro objetivo es diferente y nuestra aplicación carece de las fuertes restricciones temporales exigidas por el procesamiento en tiempo real podremos sacrificar parte de esa simplicidad en aras de obtener una mayor calidad en los contornos extraídos. En este caso, con la expresión mayor calidad queremos indicar que los contornos obtenidos tendrán menor número de puntos y estos se situarán en lugares adecuados para intentar que la aproximación poligonal obtenida se ciña de la manera más fiel posible a los contornos de las figuras.

Los inconvenientes antes mencionados consisten en que no se detectan más que cierto tipo de colinearidades y en el desajuste sistemático de la posición de determinadas aristas de los polígonos dentro de los huecos.

Finalmente, también se han introducido modificaciones para calcular sobre la marcha las cajas de circunscripción de todos los contornos. El término caja de circunscripción pretende ser una traducción de la expresión inglesa bounding box y consistirán en aquellos rectángulos tales que circunscriben a los polígonos que aproximan nuestros contornos.

4.4.1 Tratamiento de la colinearidad.

En el algoritmo propuesto por Capson dentro de cada nodo de la lista de objetos activos tenemos dos campos (deltax_izq y deltax_der) que se utilizan para determinar la colinearidad o no de los puntos de contorno que aparecen en la nueva línea con respecto a anteriores puntos de contorno. Cada vez que obtenemos un segmento de la nueva línea el mecanismo propuesto consiste en calcular para cada uno de los dos puntos del segmento la variación que han sufrido sus coordenadas x con respecto a las de los puntos correspondientes de la línea anterior. Los puntos de la línea anterior se obtendrán

a partir de los campos izquierda y derecha de un objeto activo que almacenará en su campo deltax_izq y deltax_der la variación sufrida por las coordenadas x de estos puntos con respecto a los correspondientes de la línea precedente a la anterior. Si la variación calculada para esta línea coincide con la variación almacenada en el objeto activo estaremos ante un caso de colinearidad y por los tanto no será necesario añadir un nuevo punto para describir el contorno.

Estudiemos el ejemplo de la Figura 12 que consiste en un fragmento de la parte izquierda del objeto representado en la Figura 4. El proceso comenzaría procesando las líneas que pudieran preceder al objeto hasta llegar a la línea que contiene el segmento [v1, v23]. En ese momento se ejecutaría el procedimiento crear_nuevo_objeto que introduciría un nuevo nodo en la lista de objetos activos y además unirá el punto e con el d después de crearlos en la lista de vértices, es decir creará la arista (v23, v1). El nuevo nodo de la lista de objetos activos contendrá en su campo izquierda un índice a la posición de v1 en la lista de vértices y en su campo derecha otro a v23 mientras que sus campos deltax_izq y deltax_der estarán inicializados de modo que no sea posible que se produzca ninguna colinearidad con la línea siguiente. Esta inicialización se puede conseguir por ejemplo, utilizando valores por encima del tamaño horizontal del ráster y en adelante llamaremos a esta técnica forzar la no colinearidad.

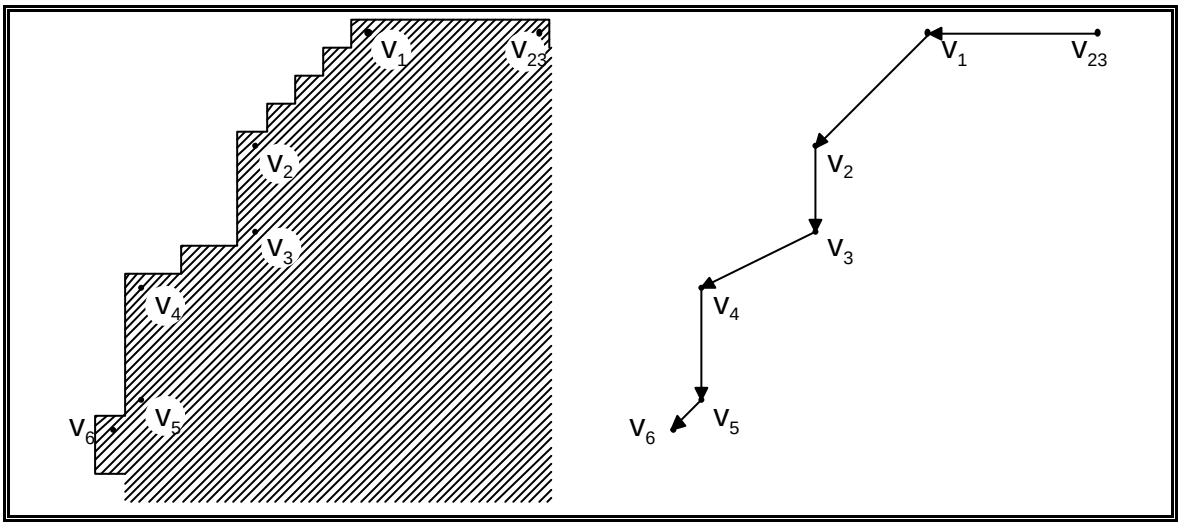


Figura 12

Una vez procesados todos los demás segmentos que pudieran aparecer en la línea pertenecientes a otros objetos, pasaríamos a la línea siguiente y llegaría un momento en el que el segmento [a, b] en la línea anterior correspondería con el [v1, v23]. A los puntos extremos de este segmento accederíamos a través de los índices izquierda y derecha del objeto activo antes creado. En la línea actual tendríamos un

nuevo segmento que se solaparía con el segmento [v1, v23], lo que según el algoritmo nos llevaría en primer lugar a ejecutar la acción “conectar a con d” y posteriormente “conectar e con b”. La primera de las acciones se encargará de generar las aristas del lado izquierdo de los objetos activos mientras que la segunda lo hará en el lado derecho. Como podemos observar, en nuestro ejemplo para simplificar solo nos preocuparemos del lado izquierdo del contorno pero en la parte derecha se deberían realizar acciones análogas. Antes de realizar cualquiera de estas dos acciones se debería calcular la variación de la componente x entre los puntos correspondientes de la línea anterior y la actual realizando las diferencias entre las abscisas de a – d y b – e en cada caso. Si el valor de la primera diferencia coincidiera con el valor almacenado en el campo deltax_izq de nuestro objeto activo, únicamente deberíamos modificar las coordenadas del punto a para que ocupara la posición del punto d. En nuestro caso como hemos forzado la no colinearidad al inicializar el campo deltax_izq del objeto activo, deberíamos realizar la acción “conectar a con d” creando el nuevo punto d en la lista de vértices y enlazándolo a continuación de a que en este caso es v1 pero además deberemos modificar el valor del campo deltax_izq al valor 1 que es el resultado de la variación de la coordenada x entre a y d.

La evolución de nuestro contorno seguiría en la línea siguiente y al calcular la variación de la coordenada x entre los nuevos puntos a y d y compararla con el valor de deltax_izq veríamos que coincide por lo que estaríamos ante una colinearidad y por tanto solo sobreescribiríamos las coordenadas del punto a con las coordenadas del punto d. La situación de colinearidad se prolongaría hasta analizar la línea siguiente a v2, porque en ese momento el valor de la variación sería 0 y diferiría del valor almacenado en deltax_izq por lo que tendríamos que crear un nuevo punto justo debajo de v2 y actualizar a 0 el valor de deltax_izq. La variación cero de la x se prolongaría hasta la línea de v3 momento en el que terminaría la colinearidad que generaría la arista entre v2 y v3. En ese momento las variaciones pasarían a tener valor 2 hasta la línea de debajo de v4 en donde volverían a ser 0 hasta v5. En resumen, mientras las variaciones calculadas coinciden con las almacenadas desplazaremos el punto de la línea actual pero en el momento en que difieran deberemos crear un nuevo punto a continuación que será el que se desplazará mientras dure la nueva colinearidad.

Este sistema resulta bastante simple y permite detectar colinearidades verticales así como todas aquellas con un ángulo comprendido entre 45º y – 45º (Figura 13-A) pero sin embargo no se consideran colineares casos en los que el ángulo sea distinto

(Figura 13-B) ni aquellas líneas horizontales que no sean un inicio o final de contorno. Los resultados que se deberían obtener en caso de hallar colinearidad en las líneas de la Figura 13-B aparecen en el apartado C de esta misma figura.

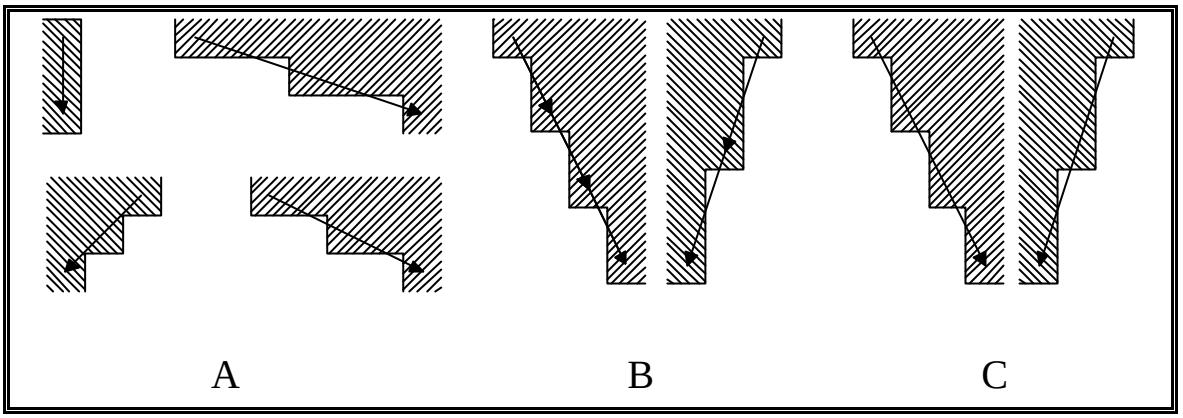


Figura 13

Como podemos ver, el ajuste de las aristas al contorno en los casos B y C es igual de bueno, pero en el caso C aparecen menos aristas lo cual redundará en un importante ahorro cuando se procesen grandes imágenes.

Un nuevo ejemplo del otro defecto de ajuste referente a las líneas horizontales lo podemos ver en la Figura 14 que nos muestra la aproximación del mismo fragmento de contorno aplicando el sistema anteriormente expuesto (A) y un mejor ajuste de la esquina del contorno (B).

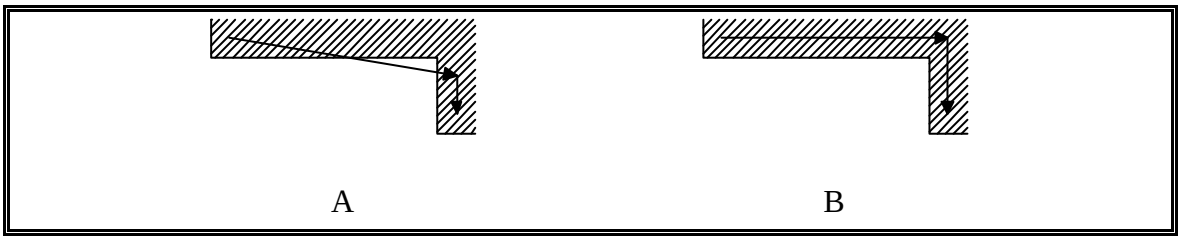


Figura 14

Para tratar de solventar estos defectos del algoritmo de Capson se aplicarán unas modificaciones que básicamente consistirán en aplicar una tabla de transformaciones dependiendo de la topología local del contorno que estamos aproximando polígonamente.

En la Tabla 4 podemos ver la citada tabla de transformaciones que como podemos apreciar se divide en cuatro subtablas, cada una de las cuales contempla 16 casos. Las subtablas de izquierda a derecha y de arriba abajo las nombraremos dependiendo de su forma como pendiente positiva, entrantes, salientes y pendiente negativa. Los esquemas que representan cada uno de los casos muestran fragmentos del lado izquierdo de un objeto activo aunque esto simplemente sea a efectos simbólicos ya

que el caso engloba las ocurrencias tanto en el lado izquierdo como en el derecho del objeto activo. La representación del lado derecho de un objeto activo sería un esquema simétrico al del lado izquierdo con respecto a la vertical, pero su significado sería el mismo puesto que las acciones que indicaría el esquema serían análogas a las indicadas por el representado en la tabla.

		Comienzo pendiente positiva				Comienzo pendiente negativa			
		Corto largo	Largo corto	Largo largo	Corto corto	Corto largo	Largo corto	Largo largo	Corto corto
Final pendiente positiva	Corto largo				fnc				
	Largo corto					fnc	fnc	fnc	fnc
	Largo largo								
	Corto corto								
Final pendiente negativa	Corto largo								
	Largo corto	fnc	fnc	fnc	fnc	fnc		fnc	fnc
	Largo largo								
	Corto corto								

Tabla 4

Cada uno de los casos comprende dos cambios en la variación de la coordenada x de los segmentos correspondientes de líneas consecutivas. Cada uno de los esquemas muestra cinco segmentos que forman cuatro esquinas que se corresponden con los

citados cambios en la variación. Si las ordenamos según aparecen al recorrer los segmentos empezando por el de más arriba, la primera y la tercera esquinas de cada diagrama representarán cada una un cambio en la citada variación de la coordenada x. La longitud de cada segmento vertical de cada una de estas dos esquinas indica si la variación que representan se ha mantenido igual a cero durante más de una línea (segmento largo) o si por el contrario, las dos variaciones que hay entre tres líneas consecutivas son diferentes (segmento corto). Los segmentos horizontales representan la variación que ha originado el cambio y del mismo modo que sucede con los segmentos verticales en los esquemas sólo se distingue entre un valor absoluto de la variación igual a uno (segmento corto) o diferente a uno (segmento largo).

El último segmento de cada esquema siempre es corto y tiene a su lado un punto con el interior blanco que se utilizará para recordar que en esa posición es donde se encuentra el punto del segmento actual correspondiente a ese lado del contorno. Aunque en ese momento del análisis no conozcamos todavía la longitud real de ese segmento vertical, el segmento del esquema tiene una longitud uno puesto que como mínimo será de ese tamaño.

Algunos de los casos aparecen etiquetados con las siglas fnc que indican que se debe forzar la no colinearidad entre el punto blanco y los futuros puntos que aparezcan a continuación de él al continuar el análisis de ese lado del contorno.

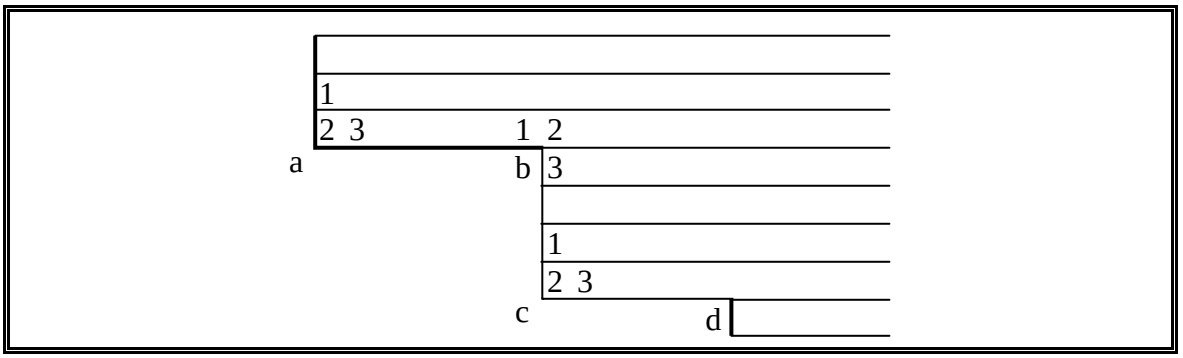


Figura 15

Los puntos oscuros indican que en esa posición se deberá insertar un nuevo punto. Los puntos deberán estar enlazados en orden ascendente para el lado izquierdo de un objeto activo y en orden descendente en el lado derecho. La Figura 15 nos muestra un pequeño fragmento del lado izquierdo de un objeto activo correspondiente a dos cambios en la variación de la coordenada x de los puntos de ese lado del contorno. Las esquinas originadas por los mencionados cambios aparecen etiquetadas con letras en el orden fijado con anterioridad. En cada una de estas esquinas en las que podemos

insertar vértices hay numeradas tres posibles posiciones que pueden ocupar los vértices del polígono con el que trataremos de aproximar el contorno. Cada uno de los esquemas de la tabla nos indicarán en que esquinas deben insertarse puntos y en cual de las tres posiciones.

Cada uno de los casos de las transformaciones puede identificarse conociendo la longitud de los segmentos de sus dos cambios de variación en x y la subtabla a la que pertenece. La primera de las variaciones se considerará el comienzo del caso y la segunda el final. En los márgenes de la tabla podemos ver la longitud de los segmentos de cada una de estas variaciones que sirven para designar los casos.

Un caso se encuadrará dentro de una subtabla según sea la inclinación de las rectas con las que aproximaremos el contorno en cada uno de los cambios de variación. Los casos en los que ambos cambios se aproximen mediante líneas con inclinación hacia el mismo lado serán clasificados en las subtablas de pendiente positiva o negativa dependiendo del lado hacia el que se inclinen. Los casos en los que las inclinaciones de los cambios sean hacia lados opuestos se incluirán en las subtablas de entrantes o salientes dependiendo de la forma cóncava o convexa de la aproximación.

Esta manera de ordenar la tabla tiene la ventaja de permitir conocer que casos pueden producirse a continuación de uno dado o cuáles pueden precederlo. Los casos que pueden producirse a continuación de uno dado son aquellos en los que la forma y pendiente de su cambio final coincide con las del cambio de comienzo. Estarán todos en una única columna de las dos que tienen en su margen superior la misma etiqueta que nuestro caso tenia en su margen derecho. La columna adecuada será aquella de las dos que esté en la mitad de la tabla marcada con la misma pendiente de comienzo que la del final de nuestro caso. Los casos que pueden haberse producido antes de uno dado estarán en la fila cuyo cambio final esté etiquetado con la misma pendiente y longitud de segmentos que el cambio inicial del caso del que buscamos los antecesores.

Como podemos ver, en la tabla hay algunas celdas con fondo gris. Estas celdas representan casos en los que de producirse colinearidad no deben de ejecutarse las acciones indicadas en la Tabla 4 sino que deberemos de ejecutar las correspondientes de la Tabla 5. Estos casos excepcionales son los que nos permiten una vez detectadas las colinearidades reducir el número de vértices de nuestra aproximación poligonal del contorno.

Ambas tablas de transformaciones han sido creadas teniendo en cuenta para cada caso los posibles casos antecesores y sucesores de modo que las acciones que se ejecutan en un caso no generen conflictos con las ejecutadas al aplicar otro caso.

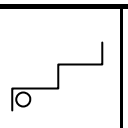
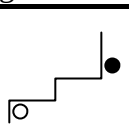
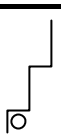
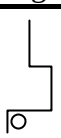
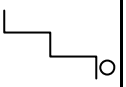
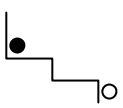
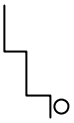
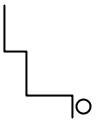
Comienzo	Corto largo	Largo largo	Largo corto	
Final	Corto largo		Largo corto	Largo largo
				
				

Tabla 5

Para poder reconocer los cambios en la variación de la coordenada x entre segmentos de líneas consecutivas introduciremos seis nuevos campos en la estructura de los nodos de los objetos activos. Tres de estos campos se utilizarán para el lado izquierdo y los otros tres para el derecho:

ant_deltax_izq, *ant_deltax_der*. Se utilizarán para almacenar el valor que produjo el anterior cambio de la variación de la coordenada x entre dos líneas consecutivas. Se corresponderá con la longitud del segmento horizontal de la primera esquina (segundo segmento) de los esquemas de los casos.

Deltay_izq, *deltay_der*. Contendrán el número de líneas que hay entre la nueva línea que se está procesando y el anterior cambio de la variación de la coordenada x. Es nuestros esquemas se identificará con la longitud del segmento vertical de la tercera esquina (tercer segmento).

Ant_deltay_izq, *ant_deltay_der*. Tendrán un papel análogo al de los dos campos anteriores pero en este caso se contarán las líneas entre el último cambio y el que le precedía. Será el segmento vertical de la primera esquina de los esquemas (primer segmento).

El proceso consistirá en ir calculando la variación que sufre la coordenada x en cada cambio de línea y si esa variación es cero únicamente incrementaremos el contador de la coordenada y del lado correspondiente (*deltay_izq* o *deltay_der*) y desplazaremos hacia abajo el vértice al que apunta ese lado del objeto activo. En el

momento en que la variación sea distinta de cero pasaremos a identificar el caso de la tabla del que se trata. Para ello consultaremos el valor de los nuevos campos teniendo en cuenta las correspondencias que existen entre ellos y los segmentos de los esquemas. Del mismo modo se identificará si se esta ante un caso excepcional de los que aparecen representados sobre un fondo gris. Una vez se conoce el caso en el que nos encontramos pasaremos a ejecutar las acciones indicadas en la tabla correspondiente y finalmente se inicializará el contador *delta_y* a 1 y *ant_deltax* pasará a tener el valor calculado de la variación de x.

Las acciones a realizar en cada caso se pueden codificar en un solo byte reservando tres grupos de dos bits para representar las tres esquinas en las que se pueden realizar inserciones. En cada uno de estos grupos de dos bits se almacenará un número que indicará la posición en la que se insertará el vértice correspondiente a esa esquina de acuerdo con los valores mostrados en la Figura 15. En caso de no tener que realizar una inserción en una esquina se marcará con el valor cero. Todavía deberemos reservar un bit de los dos restantes del byte para indicar si se debe forzar la no colinearidad. El último bit podría utilizarse para indicar si se trata de un posible caso extraordinario o para cualquier otra utilidad pero en la implementación realizada no se le ha dado ningún uso.

La Figura 16 nos muestra un ejemplo del resultado del algoritmo modificado al ejecutarse sobre un fragmento de la parte derecha de una figura. La parte de la figura marcada como A muestra el resultado de aplicar el algoritmo de Capson sin modificar, mientras que la parte derecha aplica todas las modificaciones que acabamos de explicar. Hay que destacar que el contorno de la derecha tiene menos vértices que el de la izquierda y pese a ello sigue representando el contorno con toda fidelidad.

Es posible que se pueda reprochar a la aproximación el no haber continuado la línea detectada entre las líneas 5 y 9 hasta la línea 12, pero según veremos a continuación el algoritmo considera el cambio entre las líneas 10 y 11 como una esquina y los casos están estudiados para preservar las esquinas. A continuación trataremos de explicar línea a línea los casos que se detectan durante el procesamiento de esta figura.

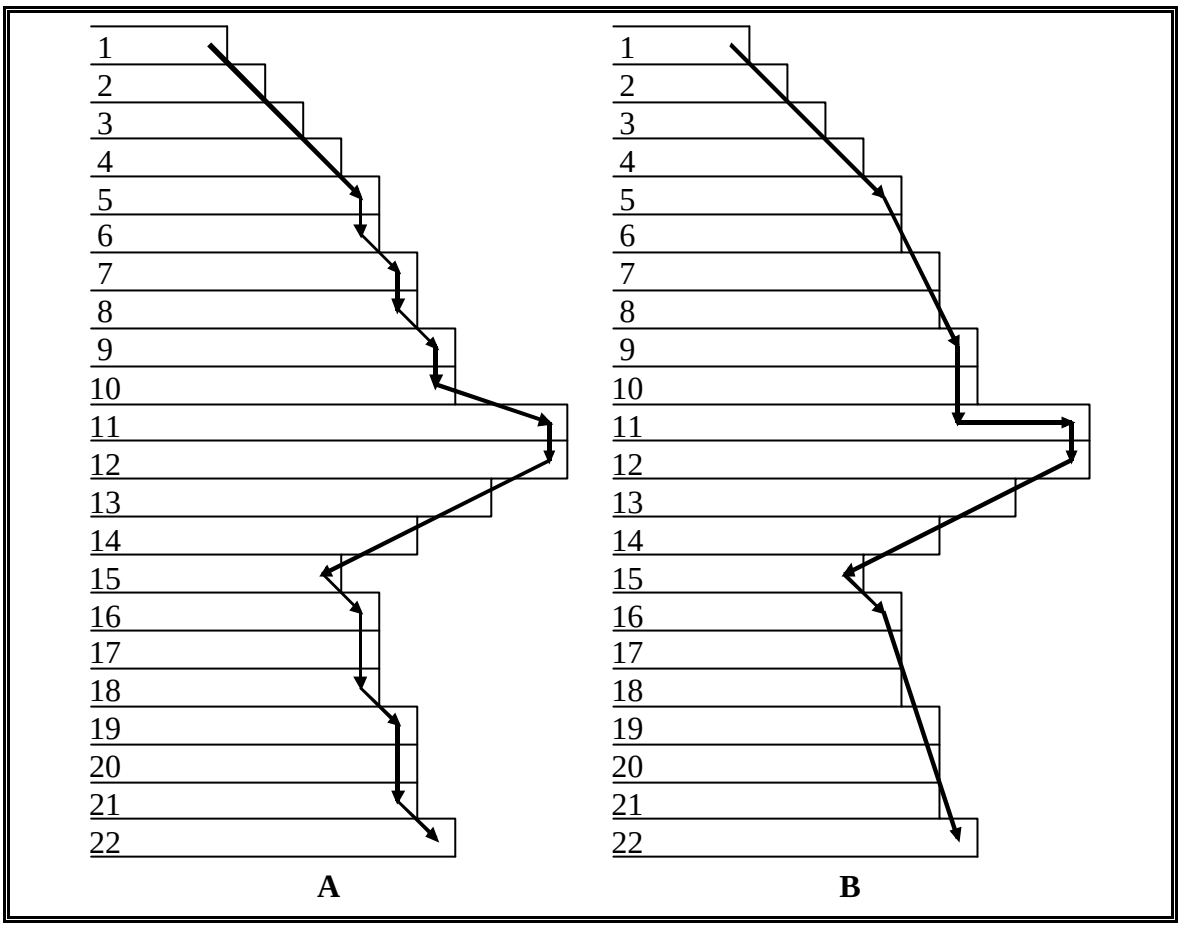


Figura 16

El primer cambio en la variación entre líneas de la coordenada x se produce entre la línea 2 y la línea 3 y tendrá un valor de –1 que coincide con la variación que había sucedido entre las líneas 1 y 2. Además *ant_deltay_der* y *deltay_der* tienen ambas valor 1. Como estamos en un contorno de la parte derecha de un objeto activo y la Tabla 4 está expresada para contornos de la parte izquierda todo el tema de las pendientes funcionará al revés, es decir que aunque la pendiente del cambio del comienzo y el del final sean las dos negativas deberemos de buscarlos en la subtabla correspondiente al comienzo y final positivos que será la de arriba a la izquierda. Como el valor absoluto de las variaciones es 1 para los cuatro segmentos, estaremos en el caso que tiene segmentos cortos en todas sus posiciones. El esquema que encaja con esta descripción es el de la cuarta fila y cuarta columna de la mencionada subtabla. Como podemos ver en el esquema elegido, en este caso no hay que realizar ninguna acción.

Se repetirán los mismos valores anteriores y por tanto se realizará la misma elección en todas las líneas entre la 3 y la 6. En esta última línea la variación de la coordenada x con respecto a la de la línea 5 será 0 y en consecuencia lo único que se

hará será incrementar el contador *deltay_der* y mover las coordenadas del punto indicado por el campo derecha del objeto activo una línea hacia abajo.

En la línea siguiente la variación de x será –1 y por tanto deberemos volver a buscar en la mitad superior de la tabla correspondiente al final con pendiente positiva[1]. Como el valor almacenado en *ant_deltax_der* es también –1 deberemos consultar en la mitad izquierda de la tabla que es la que incluye los comienzos con pendiente positiva. El valor de *ant_deltay_der* será 1 lo que junto con el valor 1 de *ant_deltax_der* indica que tenemos un comienzo con una variación corto-corto y *deltay_der* tiene el valor 2 por lo que el final será largo-corto. Todos estos datos nos llevan al caso que ocupa la segunda fila y cuarta columna de la tabla que nos indica que tenemos que insertar un punto en la posición 2 de la segunda esquina.

En la línea 8 la variación es cero por lo que no sucede nada interesante y a partir de ahora no comentaremos las líneas cuya variación sea 0. La línea 9 tiene una variación de –1, *ant_deltax_der* será también –1 y *deltay_der* y *ant_deltay_der* serán ambos 2. Con estos datos llegamos al caso de la segunda fila y segunda columna de la tabla pero como es un posible caso extraordinario deberemos comprobar si existe colinearidad. En este caso como *deltay_der* y *ant_deltay_der* son iguales y el valor absoluto de la variación de x es igual a *ant_deltax_der* y además es 1 podemos concluir que existe colinearidad y por tanto deberemos ejecutar la acción correspondiente de la tabla de casos extraordinarios (Tabla 5) que indica que no hay que hacer ninguna inserción.

La línea 11 tendrá pendiente negativa y un cambio largo-corto en el comienzo y uno largo-largo al final lo que nos conduce a la fila 3 y columna 2 de la tabla. El caso correspondiente a la línea 13 será el de la fila 7 y la columna 3 ya que es un saliente largo-largo en el comienzo y el final.

La línea 14 producirá una excepción a la tabla ya que tenemos un tramo que coincide con la primera excepción de la subtabla de la pendiente negativa. Como las variaciones de la x son iguales en los dos cambios y el número de líneas del primer segmento de ambos tramos es igual a 1 podemos decir que existe colinearidad y por tanto deberemos ejecutar la excepción correspondiente de la Tabla 5 que sólo inserta un punto en la posición 2 de la primera esquina. En la línea 15 también se produce una colinearidad por lo que se ejecuta la excepción correspondiente al primer caso de la subtabla de pendiente negativa.

[1] Recordar que de estar en el lado izquierdo de un objeto activo, un valor negativo en la variación de x indicaría pendiente negativa.

La línea 16 termina con la colinearidad anterior y se identifica con el caso de la última fila y la quinta columna. La siguiente línea con interés será la 19 y en ella se produce un caso correspondiente con la celda de la cuarta fila y la segunda columna. La nueva colinearidad que comienza en esta línea se detectará en la línea 22, por lo que para finalizar se ejecutará la excepción del caso de la segunda fila y columna.

El lugar del algoritmo donde se deben realizar todas estas comprobaciones y acciones coincide con las líneas del pseudo-código que indican las acciones “conectar a con d” y “conectar e con b”. Estas dos acciones son las que se encargan de generar las aristas de los lados de las figuras correspondiendose respectivamente con el lado izquierdo y el derecho. También tenemos que inicializar los nuevos campos de los objetos activos en el momento en que se crean, labor que corresponde a las funciones *crear_nuevo_objeto* y *partir_objeto.* Además deberemos comprobar los casos que se producen al finalizar los contornos para ejecutar la acción que mejor se adapte a cada final. La finalización de los contornos consiste en cerrar por la parte de abajo las aristas de los lados izquierdo y derecho de cada objeto activo que se han generado con todo el proceso anterior. Los objetos activos desaparecen en las funciones *terminar_objeto* y *unir_huecos* por lo que serán tambien las encargadas de finalizar los contornos. Para finalizar los contornos deberemos tomar los lados izquierdo y derecho del contorno que termina y estudiar los casos que se producen. El conjunto de casos que se puede producir en un final de contorno es más restringido que el que teníamos hasta ahora porque solo nos preocupa el primer cambio de la variación de x y el segmento vertical del segundo cambio. Por ejemplo, para la terminación de un objeto mediante la función *terminar_objeto* podremos consultar solo los casos de la quinta y septima filas de la mitad inferior de la Tabla 4 que contiene los casos que tienen final con pendiente negativa. En la función *unir_huecos* podríamos buscar solo en la tercera y cuarta filas de la mitad superior de la tabla que corresponde con los casos que finalizan con pendiente positiva.

4.4.2 Ajuste de los huecos.

Los huecos de las figuras en el algoritmo se detectan inicialmente como un objeto activo que se parte en dos ramas que cierto número de líneas más tarde volverán a unirse. Así pues tenemos tres momentos significativos en la evolución del hueco de un objeto: el momento en el que aparece como la partición del objeto inicial, el desarrollo de sus lados y por último el momento en que se termina al unirse dos objetos activos.

Según el algoritmo en pseudo-código que hemos visto con anterioridad las funciones *partir_objeto* y *unir_objetos* corresponderán respectivamente con la aparición y la terminación del hueco mientras el desarrollo de sus lados se llevará a cabo al realizar las acciones de conectar a con d y e con b. La primera de estas dos acciones se encargará de generar las aristas del lado izquierdo del polígono con el que pretendemos aproximar el hueco mientras que la otra se encargará de generar las del lado derecho.

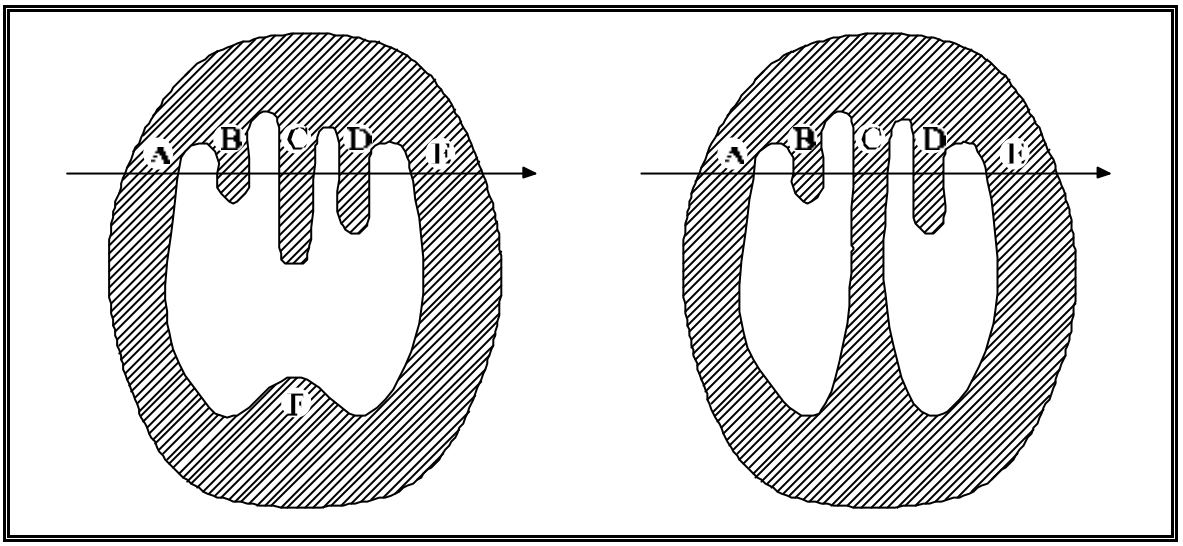


Figura 17

Hay que tener en cuenta que un hueco (al igual que un contorno exterior) también se puede formar a partir de dos o más objetos activos. En un momento dado un objeto puede partirse varias veces formando cierto número de huecos potenciales. Algunas de las ramas en las que se ha dividido el objeto pueden unirse para formar un hueco, pero otras terminarán sin más, formando un saliente del objeto hacia el interior del hueco. En la Figura 17 se muestran dos siluetas, la flecha indica una línea de escaneado y como se puede apreciar, en el momento de procesar esta línea lo único que conocemos de la figura es que anteriormente se ha partido en cinco ramas (A, B, C, D, E) pero no podemos saber cual será la evolución de la figura. En el momento en que se analiza la línea indicada por la flecha las cinco ramas están formando cuatro huecos potenciales. En momentos posteriores primero la rama B y posteriormente C terminarán, con lo que los huecos potenciales a los que daban lugar desaparecerán. En la silueta de la parte izquierda, la rama C tomará el mismo camino que las dos anteriores pero en la parte derecha se producirán dos uniones entre las ramas A, C y E que darán lugar a los dos huecos. Como podemos ver dos figuras que hasta cierto momento resultaban idénticas, sufren una evolución diferente en una de sus ramas (C) que hace que el resultado final sea muy distinto.

Según el algoritmo, cada vez que una figura se parte se ejecutará el procedimiento *partir_objeto* que comienza conectando los puntos e y f. Del mismo modo, cada vez que se termine un hueco se ejecutará el procedimiento *unir_objetos* que unirá los puntos c y b. Si nos fijamos en la Figura 18-B podemos ver un ejemplo del momento en que se produce la partición del objeto que muestra la arista que se genera en ese momento y en la Figura 18-C tenemos el momento de la unión que cierra el hueco. Como podemos apreciar, en ambos casos la arista generada aparece en una posición tal que se sale del área del objeto y intersecta el área del hueco.

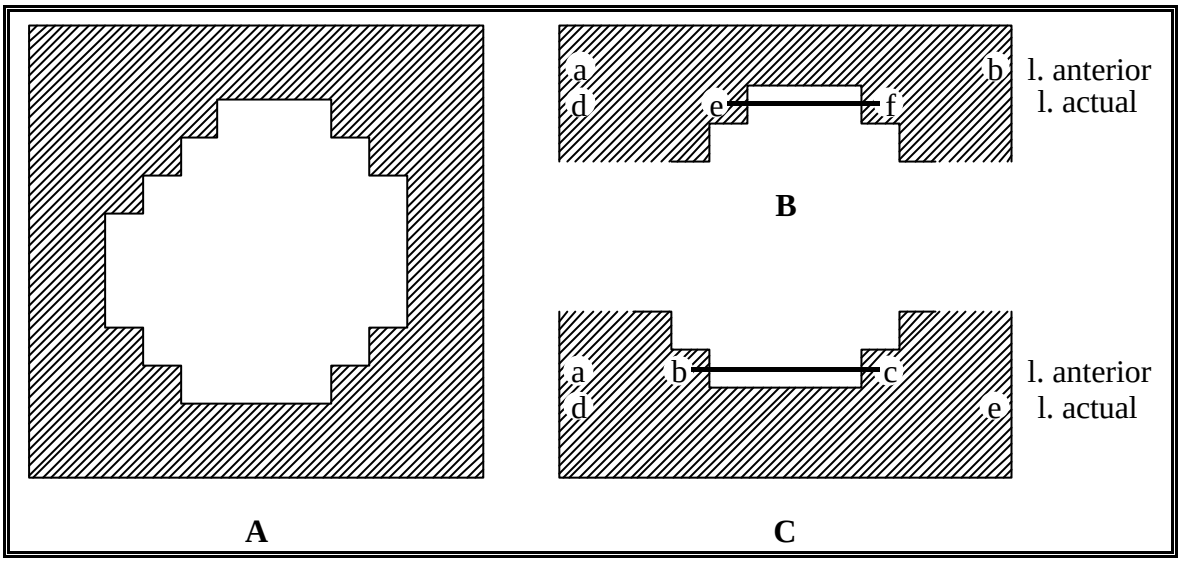


Figura 18

Pero este problema no solo surge cuando tenemos huecos reales, sino que aparece siempre que se tengan que ejecutar *partir_objeto* o *unir_objetos* y para ello basta con que sean posibles huecos.

Para eliminar este tipo de problemas la solución pasa por subir los puntos e y f a la línea anterior en cuanto se detecte la aparición de un posible hueco y bajar los puntos b y c a la nueva línea cuando se detecte la terminación de este. Los desplazamientos anteriormente comentados resultan sencillos puesto que únicamente consisten en incrementar o disminuir la coordenada y de los puntos según el caso.

El mayor problema aparece a la hora de ajustar la coordenada x de estos nuevos puntos, puesto que si simplemente desplazamos verticalmente los segmentos obtenidos incurriremos en errores en los lados del segmento dado que las nuevas aristas que generemos no se ajustarán adecuadamente a estos. En el ejemplo de la Figura 18 los puntos del lado izquierdo del hueco deberán desplazarse a la derecha y los de la derecha hacia la izquierda como podemos ver en la Figura 19. Pero por ejemplo, en caso de que

uno de los lados del hueco fuera vertical el correspondiente punto no debería desplazarse horizontalmente.

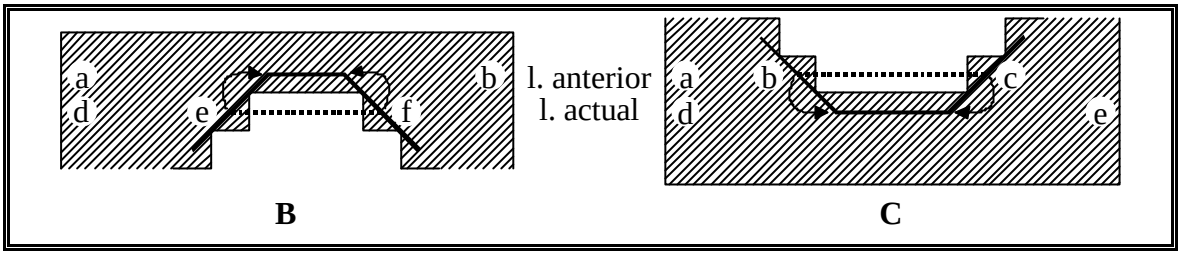


Figura 19

Así pues, en el momento en que aparece o termina un posible hueco(*partir_objeto* y *unir_objetos* respectivamente), habrá que decidir si modificamos o no las coordenadas de los puntos izquierdo y derecho de la nueva arista en función de la dirección que tomen los respectivos lados del hueco.

En el caso de la aparición de un posible hueco lo que haremos será simplemente inicializar los nuevos campos de los objetos activos que intervendrán a ambos lados de la partición. De este modo, en la línea siguiente a la partición cuando se ejecuten las modificaciones sobre las acciones unir a con d y unir e con b explicadas en el anterior punto, se podrá detectar que nos encontramos ante un contorno que proviene de una partición. Como en el momento de ejecutar estas acciones ya se conocerá la orientación que toma cada uno de los lados de los contornos podremos decidir si modificar la componente x o no. Dependiendo de la elección tomada ajustaremos los nuevos campos añadidos en las modificaciones del punto anterior para que a partir de ese momento se pueda continuar con la identificación de casos del modo que allí se explicó.

En la finalización de un posible hueco, las modificaciones realizadas en el punto anterior sobre la función *unir_objetos* del algoritmo original, nos proporcionan todos los puntos hasta la línea anterior a la unión. Así pues, solo restan por añadir los puntos que corresponden a esta última línea que los añadiremos teniendo en cuenta el caso detectado en la última variación. Conocer este caso nos permitirá decidir sobre la posición horizontal de los puntos de la última línea.

4.4.3 Cajas de circunscripción.

Como ya se comentó cualquier figura estará inscrita dentro de su caja de circunscripción. El papel de las cajas de circunscripción es importante a la hora de dibujar en pantalla los polígonos que aproximan los contornos porque en muchos casos

nos permiten conocer rápidamente aquellos contornos que no necesitan ser redibujados dentro de una ventana que muestra parte de una imagen. Además, en el programa implementado se utilizará el tradicional algoritmo de recortado de segmentos contra ventanas de Sutherland-Cohen para recortar las aristas de los polígonos cuya caja de circunscripción intersecte el área de la ventana de dibujo.

Para calcular la caja de circunscripción de un objeto bastará con conocer cuatro valores que se incluirán como nuevos campos en la estructura de datos de los objetos activos:

x_max, *y_max*. Contendrán el mayor valor encontrado entre todas las coordenadas horizontales y verticales respectivamente de los vértices que aproximan un contorno.

x_min, *y_min*. Se corresponden con el menor valor encontrado entre todas las coordenadas horizontales y verticales respectivamente de los vértices que aproximan un contorno.

Las cajas de circunscripción se pueden representar por un rectangulo expresado como sus esquinas superior izquierda[2] (*x_min*, *y_min*) e inferior derecha (*x_max*, *y_max*).

El campo *y_max* de un objeto activo no se conocerá hasta el momento en que se produzca la terminación del objeto puesto que se corresponderá con la coordenada y de los puntos del último de sus segmentos. El valor de los campos *x_max* y *x_min* se actualizará conforme se vayan analizando nuevas líneas. El campo *y_min* se inicializará con el valor de la línea en la que aparezca el nuevo objeto pero en caso de unión de dos objetos activos el campo del objeto que permanezca deberá contener el menor de los dos valores que se corresponderá con la línea de comienzo del objeto más antiguo.

Cuando se parta una figura, el nuevo objeto que aparece en la lista de objetos activos se insertará a la derecha del objeto actual y representará la rama derecha de la figura. Los campos *x_min* y *x_max* de este nuevo objeto corresponderán respectivamente con los puntos e y f, mientras que el campo *y_min* contendrá el número de la línea anterior que es el punto donde aparecerá la primera arista de la nueva cadena de vértices que representará el posible hueco que aparece a causa de la partición.

[2] Consideraremos el origen de coordenadas situado en la esquina superior izquierda y el sentido creciente del eje de ordenadas hacia abajo.

Al unirse dos objetos activos deberemos tener en cuenta si estamos cerrando un hueco (los dos objetos activos representan ramas de una misma figura) o si estamos realizando una simple unión. En el primer caso deberemos tener en cuenta que se cierra el polígono con el que aproximaremos el hueco y por tanto ya podemos asignar como valor del campo *y_max* el número de la línea actual. En caso de que estemos ante una simple unión, el campo *x_max* del objeto persistente deberá contener el mayor de los dos valores de este campo entre este objeto y el que desaparece. Deberemos realizar el mismo cálculo en el caso de los campos *x_min* e *y_min* pero en este caso buscaremos el mínimo valor de los dos.

En la terminación de un objeto activo también aparecen dos casos dependiendo de si quedan todavía más ramas activas de la figura a la que pertenece o si la figura termina definitivamente. En el caso de una figura que termina definitivamente, sólo deberemos ocuparnos de rellenar el campo *y_max* con el número de línea actual menos uno puesto que en esa línea estará el punto más bajo de la figura. En el otro caso habrá que realizar acciones análogas a las realizadas en la unión de dos objetos pero aquí los objetos activos entre los que compararemos sus campos serán el objeto que termina y el apuntado por el campo *adelante* de este que será otro objeto activo perteneciente a la misma figura.

El momento en que se deben actualizar los campos *x_max* y *x_min* cuando estamos en una línea en la que no se ejecuta ningún suceso de los anteriores coincide con las acciones conectar a con d y conectar e con b. En el momento en que se ejecuta la primera acción, si la figura no está dividida en varias ramas, simplemente deberemos comparar el punto d con el valor almacenado en el campo *x_min* del objeto activo actual y de ser menor aquel primero, asignar su valor al campo. En caso de tener una figura con varias ramas además del campo *x_min* también deberemos actualizar el campo *x_max* si el punto d es mayor que el valor almacenado en este último campo. Si la figura está formada por una única rama cuando se debería ejecutar la acción conectar e con b deberemos comparar el valor de *x_max* con la e y si esta última es mayor actualizar el valor del campo. En el caso de tener varios objetos activos referentes a la misma figura, deberemos actualizar con el valor e y del mismo modo que hemos venido indicando hasta ahora los campos *x_max* y *x_min* del objeto apuntado por el campo *adelante* del objeto activo actual. Actualizando con este valor el objeto apuntado por *adelante* conseguimos que al procesar la rama de más a la derecha de la figura los

valores máximos obtenidos se almacenarán en los campos del objeto activo de más a la izquierda que será el que contendrá las coordenadas del contorno exterior de la figura.

5. Filtrado de polígonos de contorno.

5.1 Introducción.

Los contornos obtenidos después de procesar una imagen con el algoritmo mejorado de aproximación poligonal (AMAPC) no son todavía perfectamente aptos para proceder directamente a su emparejamiento. Se presentan problemas al estar los polígonos obtenidos afectados por el ruido que aparece durante el escaneado en los contornos de las figuras. Este ruido se muestra en nuestros contornos en forma de pequeños salientes y entrantes de unos pocos pixeles que hacen que los contornos del ráster tomen un aspecto dentado. La profundidad de estos dientes en muchos casos no supera un pixel pero a pesar de ello estos saltos son detectados por el AMAPC como parte del contorno y por tanto los polígonos que genera se ajustarán a ellos produciendo zigzags, saltos y otros efectos indeseables. El principal problema de estos efectos es que dificultarán el proceso de emparejamiento al incrementarse el número de las aristas que componen el contorno. Además las aristas que representan el ruido suelen tomar direcciones que nada tienen que ver con las del lado opuesto del contorno de la línea con el que deberían emparejar por lo que difícilmente podrán reconocerse como pareja.

Como acabamos de comentar el ruido de los contornos no hace otra cosa que añadir confusión a nuestro proceso por lo que la solución más inmediata parece ser eliminarlo del ráster mediante operaciones morfológicas. Sin embargo, la eliminación de este tipo de ruido con operaciones morfológicas tradicionales es un proceso delicado y que requiere de un considerable grado de interacción por parte del usuario. Esta dificultad proviene de la elección de la operación morfológica junto con sus parámetros más adecuados para cada circunstancia. Además hay que tener en cuenta que para obtener los mejores resultados deberemos aplicar estos filtros a zonas determinadas de la imagen y no a toda ella en su conjunto. Pensemos que el mismo filtro o incluso el mismo valor de un parámetro que resulta adecuado para un área de la imagen, puede dar resultados decepcionantes aplicado sobre otra área de la misma imagen. Por todas estas razones, no parece muy adecuado el utilizar operaciones morfológicas tradicionales para obtener el grado de automatización que se pretende obtener en este proyecto.

Una vía alternativa sería utilizar sobre el ráster algún otro tipo de filtro más sofisticado que consiguiera la homogeneización y suavización de los contornos, pero el desarrollo de tales filtros está fuera de los objetivos de este proyecto. Por nuestra parte

hemos optado por un tercer camino alternativo que no excluye la utilización de ninguno de los anteriores. Esta tercera opción consiste en la implementación de un filtro que actúe sobre las aristas de los polígonos con los que aproximamos los contornos e intente eliminar los defectos provocados por el ruido sobre las líneas horizontales y verticales que es el lugar donde estos resultan más evidentes.

Existe un segundo problema que se puede resolver aprovechando este último filtro. El problema consiste en que bajo ciertas circunstancias muy particulares, el AMAPC sitúa algunos puntos superpuestos encima de otros, es decir que pueden aparecer dos puntos consecutivos de un polígono con las mismas coordenadas originando el equivalente a una arista de longitud cero. Aprovechando que tendremos que analizar con nuestro filtro una por una todas las aristas de nuestros polígonos, podemos detectar y eliminar todos los puntos redundantes.

Otro tipo de problemas viene originado por el estricto concepto de colinearidad que se maneja en el AMAPC. Recordemos que en él, dos aristas consecutivas solo se consideran colineares si en ambos ejes las variaciones entre los vértices de sus extremos son exactamente iguales. Esta inflexibilidad que resultaba imprescindible en el AMAPC provoca que en el momento que una de sus variaciones se vea alterada en un sólo pixel las dos aristas serán consideradas no colineares y por tanto no se podrán aproximar por una arista más larga. Esta última situación, causa un incremento del número de aristas que como ya hemos visto, resulta perjudicial.

Algunas de las alteraciones en las variaciones son achacables al ruido de los contornos del ráster pero en muchos casos resultan ser intrínsecas a la propia línea. Basta con que observemos con detalle en una imagen escaneada cualquier línea recta con pendiente próxima a la horizontal o la vertical para darnos cuenta de que las dimensiones de los escalones que forman sus contornos, en escasas ocasiones guardan una regularidad tan perfecta como la que el AMAPC requiere para considerar una colinearidad.

Por todo lo anterior, se hace imprescindible un filtro que analice las aristas de los polígonos que aproximan nuestros contornos y que con un criterio más flexible que el utilizado hasta ahora determine cuando se está ante una colinearidad y en ese caso sustituya las aristas involucradas por una equivalente de mayor longitud.

5.2 Filtro de ruido.

Como acabamos de comentar, trataremos de crear un algoritmo que recorriendo una sola vez cada una de las aristas de nuestros polígonos, detecte las situaciones en las que estas adoptan configuraciones características que hagan sospechar que se han producido a causa de algún ruido en el contorno e intente eliminar las distorsiones producidas.

En la Figura 20 podemos ver fragmentos de un par de capturas de pantalla realizadas durante la ejecución de nuestro programa. La parte de arriba de la figura muestra los resultados que produce el AMAPC sin someterlos a ningún tipo de procesamiento extra, mientras que en la parte de abajo podemos ver los resultados obtenidos cuando además se aplica el filtro de ruido propuesto. Como se puede apreciar, los resultados obtenidos son sensiblemente mejores en el segundo caso puesto que aparecen un menor número de aristas (trazos de color rojo). Aunque el ajuste sobre el área del objeto (trazo negro) del segundo caso no sea tan preciso como en el primero esto en lugar de suponer un problema realmente es una ventaja porque incluso se aproxima más a la línea ideal que un ser humano percibiría al ver esta imagen.

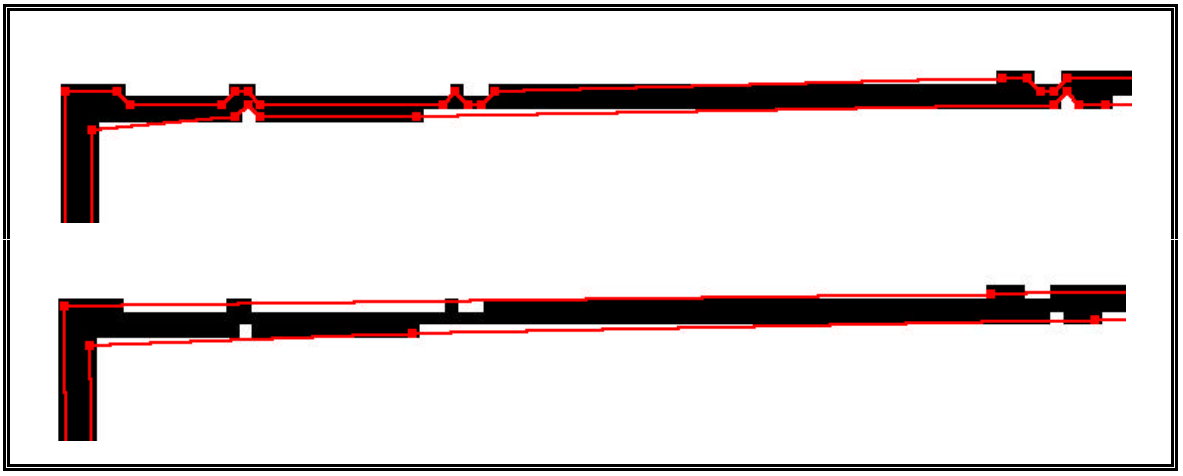


Figura 20

Hay que destacar que la imagen se ha extraido de un plano escaneado a una resolución de 300 x 300 p.p.p. y la imagen aquí mostrada ha sido ampliada con un zoom del 850 % aproximadamente. No se ha sometido al plano a ningún tipo de procesamiento previo y merece la pena mencionar que en caso de aplicar sobre este fragmento de la imagen una operación morfológica clásica como una erosión o un opening, se perdería la conectividad de los pixeles de la recta aunque la profundidad de

la operación fuera uno, ya que como se puede ver, hay zonas en las que el grosor de la recta es de solamente un pixel.

En la parte de arriba de la figura se ve claramente como se repiten las mismas formas características en la configuración de las aristas cada vez que el contorno esta afectado por un ruido. Todas estas configuraciones tienen en común que aparecen en ellas unas peculiares líneas de poca longitud ($\sqrt{2}$ pixeles) que van desde un pixel a otro contiguo que se encuentra en una de sus diagonales (ángulo de $\pm 45^{o}$). Estas pequeñas rectas serán las que al aparecer entre las aristas de un polígono de contorno, nos alertarán acerca de la posible presencia de un ruido. Teniendo esto en cuenta, ya solo nos queda identificar la configuración de las demás aristas del posible ruido y si se corresponde con una forma habitual de ruido adoptar las acciones oportunas.

Consideraremos ruido todos aquellos entrantes, salientes y escalones en los que intervienen una o más de las pequeñas rectas que hemos mencionado anteriormente. En la Figura 21 tenemos algunos ejemplos habituales de ellos que podemos identificar con los que aparecen en la Figura 20. Pero hay que tener en cuenta que cada uno de estos ejemplos también puede aparecer en posición vertical (con el sentido de las flechas hacia arriba o hacia abajo) o simetrico con respecto a la horizontal.

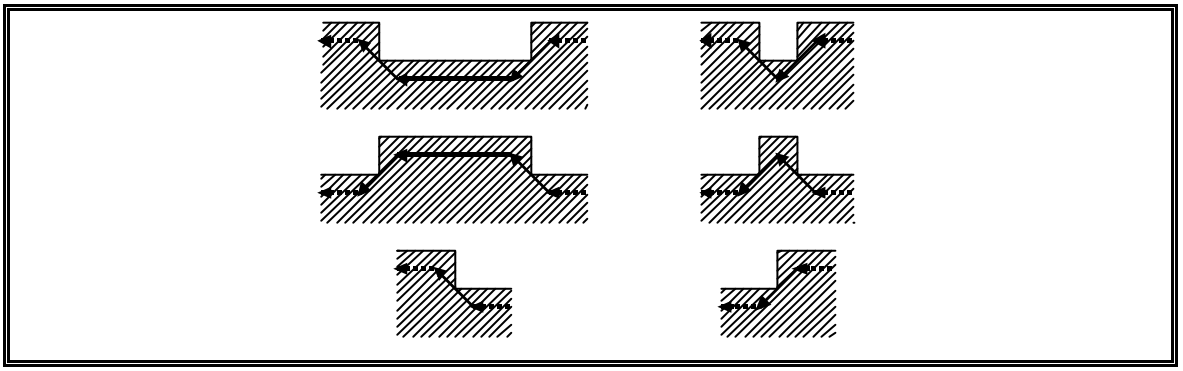


Figura 21

Además, los ruidos no deben tratarse individualmente sino que por el contrario deben de agruparse todas las aristas que correspondan a una serie de ruidos consecutivos que estén relacionados para poder aproximarlos del mejor modo posible. Para poder conseguir esto deberemos escribir un algoritmo que se encargue de contemplar todos los casos que aparecen en la Figura 22. Esta figura nos muestra un grafo en el que las transiciones entre estados corresponderán con las aristas que nos van apareciendo al recorrer los polígonos con los que aproximamos los contornos.

Como se puede apreciar, tan solo hay tres símbolos de transición entre estados. La barra inclinada hacia la derecha o hacia la izquierda indica la aparición en nuestro polígono de una pequeña arista indicadora de ruido, mientras que la barra horizontal hace referencia a cualquier otra arista diferente de estas. La inclinación hacia la derecha o la izquierda de los símbolos que representan a las pequeñas aristas de ruido, no indica inclinación real sino que es una simple manera de expresar gráficamente que la inclinación de una de estas aristas es hacia el lado contrario al de la última arista de este tipo que apareció antes del estado del que parte el arco etiquetado por el símbolo. Hay que recordar que aunque los diagramas aquí representados siempre aparecen en horizontal esto no tiene que ver nada con la posición real de las aristas, simplemente se utilizan para expresar una configuración particular de estas que en la realidad puede estar en cualquier posición.

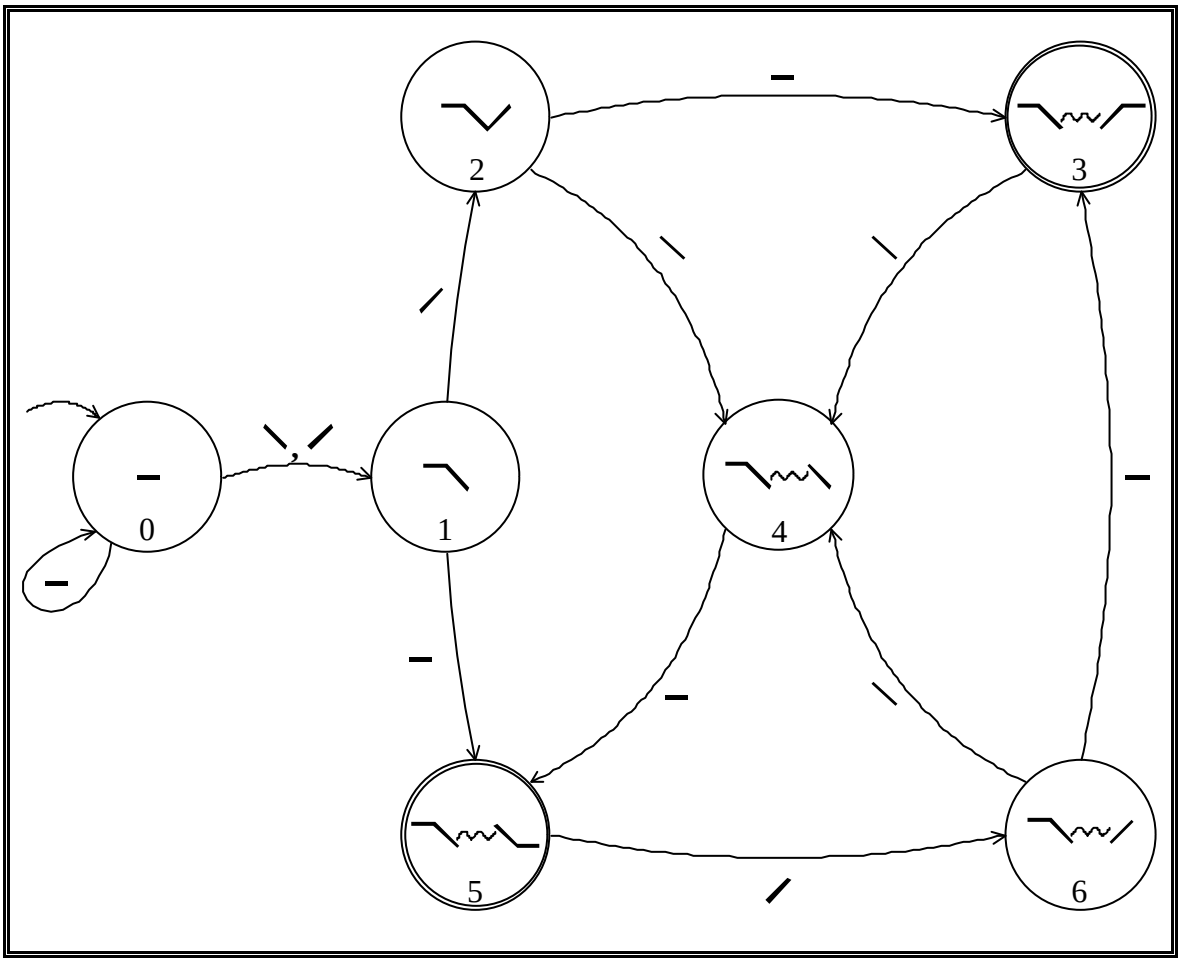


Figura 22

En la implementación realizada, para identificar las aristas indicadoras de ruido se calculan los incrementos entre las coordenadas verticales y horizontales de los extremos de las aristas y se calcula el producto entre ambos. Cuando el valor absoluto

del resultado del producto sea igual a uno estaremos ante una de estas aristas. Esta manera de detectar las aristas indicadoras resulta muy sencilla y tiene la ventaja de funcionar en cualquier sentido. Además, para saber si dos aristas tienen inclinación opuesta bastará con multiplicar entre sí los productos de cada una de las aristas y en caso de ser cierto obtendremos el resultado –1.

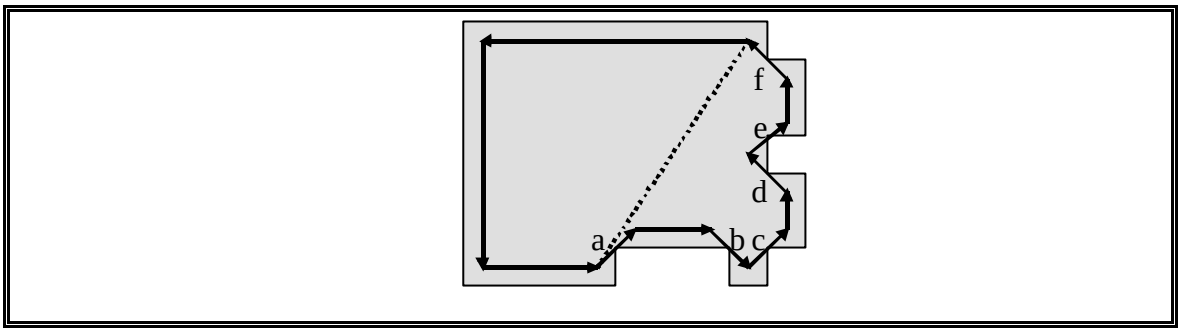


Figura 23

Cuando estamos en un caso como el de la Figura 23 y otros similares, podemos tener problemas al detectar los ruidos. Si sobre esta figura aplicamos el método de los productos junto con el grafo de la Figura 22, la agrupación de ruidos que se detecta va desde la arista indicadora a hasta la d. Si eliminàramos las aristas involucradas en este presunto ruido cometeríamos un error de ajuste en nuestra aproximación al contorno puesto que aparecería una arista en la posición indicada por la línea discontinua. Aunque en este caso el error no es muy importante podrían presentarse casos en los que sí lo fuera y por tanto deberemos tener en cuenta otros parámetros para discriminar casos de este estilo.

Los parámetros adicionales que hemos considerado en nuestra implementación para tratar de solucionar estos problemas son la orientación, el sentido y el tamaño de cada una de las componentes de una agrupación de ruido. En la figura del ejemplo podemos ver las dos componentes básicas que forman las agrupaciones de ruido. La primera de ellas es la más simple de las dos y está formada únicamente por dos aristas indicadoras consecutivas que formarán un pequeño entrante o un saliente como el que en la figura forman las aristas d y e. La segunda componente estará formada por dos aristas indicadoras entre las que se intercala una línea horizontal o vertical formando configuraciones del estilo de la que en la figura forman las aristas que van de la a hasta la b. La orientación de este último tipo de componentes será la misma que la de la arista no-diagonal de las tres, mientras que la de una componente del primer tipo coincidirá con la de mayor longitud de esta. En el programa implementado se consideran las

siguientes orientaciones: horizontal, vertical, diagonal, casi horizontal, casi vertical y casi diagonal. Dentro de cada una de estas orientaciones tendremos dos sentidos: hacia la izquierda o hacia la derecha, y en los casos vertical y casi vertical hacia arriba o hacia abajo. El sentido vendrá dado por el orden seguido por las aristas. El tamaño se utilizará para determinar si en un caso particular la componente de ruido a eliminar tiene un tamaño considerable en comparación con las aristas que la rodean o por el contrario es prescindible.

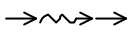
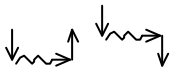
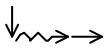
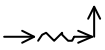
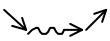
Diagrama	Descripción	Acción
	El ruido, su arista predecesora y su sucesora tienen la misma orientación y sentido.	[a, d]
	El ruido, su arista predecesora y su sucesora tienen la misma orientación pero el sentido de la arista sucesora es contrario al del ruido.	[a, c[
	El ruido, su arista predecesora y su sucesora tienen la misma orientación pero el sentido de la arista predecesora es contrario al del ruido.	[b, c]
	La arista predecesora y la sucesora tienen la misma orientación pero no es la misma que la del ruido.	[b, c]
	El ruido y su arista sucesora tienen la misma orientación pero es distinta de la de la arista predecesora.	[b, d]
	El ruido y su arista predecesora tienen la misma orientación pero es distinta de la de la arista sucesora.	[a, c]
	El ruido y su arista predecesora tienen la misma orientación pero sentidos opuestos. La arista sucesora tiene distinta orientación.	[b, c]
	El ruido, su arista predecesora y su sucesora tienen distinta orientación.	[b, c]

Tabla 6

Una vez decidida la eliminación de un ruido, debemos decidir que puntos de sus aristas se van a eliminar. Esta decisión se tomará teniendo en cuenta la orientación y el

sentido del ruido que queremos eliminar y de las aristas que se encuentran inmediatamente antes y después de él.

La Tabla 6 describe todos los casos de borrado de conjuntos de ruidos que se han contemplado en el filtro implementado. La primera columna proporciona una representación gráfica del caso que corresponde a cada fila. Del mismo modo que ocurría anteriormente, estos diagramas no pretenden representar estrictamente una sola situación sino que abarcarán un conjunto de ellas, algunas de las cuales pueden tener aspecto distinto al del diagrama. La segunda columna nos da una descripción del caso en la que se indica la orientación y el sentido de las aristas que en él intervienen. Por último, en la columna acción podemos ver el conjunto de puntos que se eliminarán expresado como un intervalo. El significado de las letras que aparecen en el interior del intervalo se muestra en la Figura 24. Esta figura representa un conjunto de ruidos tipo en el que se han etiquetado los puntos extremos de la primera y última aristas indicadoras.

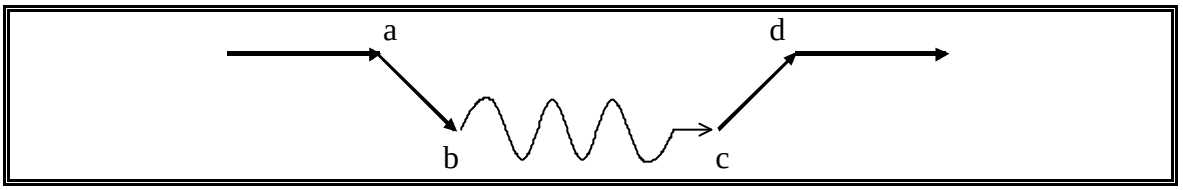


Figura 24

5.3 Filtrado de colinearidades.

El filtro que hemos diseñado recorrerá una por una todas las aristas de los polígonos que aproximan una figura e irá comparando parejas de aristas consecutivas entre sí. Cuando encuentre una pareja de aristas que se puedan considerar colineares continuará comprobando las siguientes aristas hasta encontrar alguna que ya no lo sea. En ese momento eliminará todos los puntos intermedios y aproximará el conjunto de aristas por una sola que irá desde el primer vértice del conjunto hasta el primer vértice de la arista no colinear.

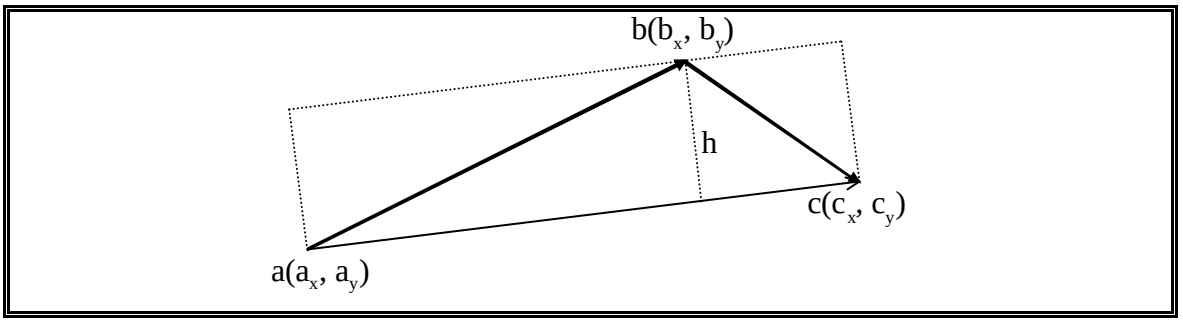


Figura 25

En la Figura 25 tenemos dos aristas consecutivas ($\overrightarrow{ab}$ y $\overrightarrow{bc}$) que vamos a comprobar si son colineares o no. Para ello en primer lugar calcularemos la superficie del rectángulo que aparece en línea discontinua resolviendo el determinante :

$$\text{Área} = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} = (a_y - c_y)b_x + (c_x - a_x)b_y + (a_x c_y - c_x a_y)$$

Para calcular h que será la altura del rectángulo de trazo discontinuo bastará con dividir el área por su base, la cual calcularemos mediante el teorema de Pitágoras:

$$h = \frac{\text{Área}}{\sqrt{(a_y - c_y)^2 + (c_x - a_x)^2}}$$

Una vez calculada la altura debemos compararla con la máxima desviación tolerable entre dos aristas colineares y si h es menor que este parámetro podemos considerar colineares las dos aristas.

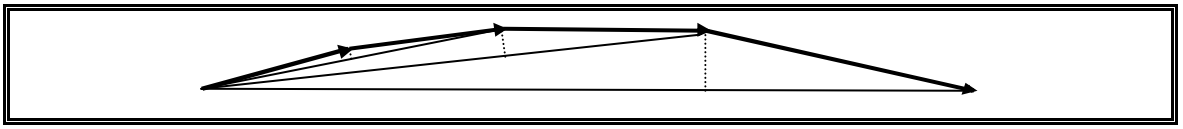


Figura 26

Cuando tenemos más de dos aristas seguidas que se consideran colineares como en el caso que muestra la Figura 26 podemos llegar a cometer un error considerable si no tenemos en cuenta que las diferentes alturas que vamos calculando están siempre al mismo lado de la recta con la que aproximaríamos al conjunto y además se va incrementando su valor cada vez más.

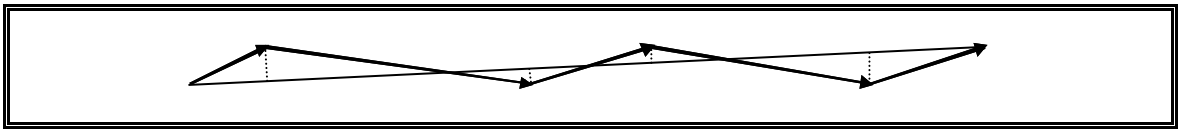


Figura 27

Por otro lado, en un caso como el de la Figura 27 podemos ver que las alturas están a ambos lados de la recta y que no existen los problemas anteriores. Para poder resolver el problema iremos sumando las alturas a medida que las vamos calculando y en el momento en que su suma supere un umbral máximo consideraremos que la colinearidad ha terminado. Este método funcionará porque en el primer caso todas las alturas tienen el mismo signo mientras que en el segundo los signos se van alternado al estar los extremos de las aristas a ambos lados de la recta y por lo tanto se compensarán las desviaciones que se van produciendo.

5.4 Conclusiones.

Se han diseñado dos filtros que aplicados sobre las aproximaciones poligonales pretenden reducir el número de aristas de estas sin producir perdidas significativas de información en la forma de los contornos. En ambos casos el coste temporal de las operaciones será lineal con el número de aristas de la figura a simplificar.

En el caso del filtro de ruido la técnica que se ha utilizado, desde un punto de vista formal resulta ser un autómata reconocedor de un determinado lenguaje. El reconocedor que se ha implementado utiliza un lenguaje muy simple pese a lo cual se obtienen resultados más que aceptables. Podía ser de interés enriquecer el lenguaje utilizado para así poder abarcar un mayor número de casos que se adaptaran con mayor fidelidad a determinadas situaciones.

El filtro de colinearidades podía mejorarse utilizando algún método que empleara expresiones más simples para el calculo de las alturas pero de todos modos el coste asintótico no se modificaría.

6. Emparejamiento de aristas de contorno.

6.1 Introducción.

Como ya hemos visto, en la imagen ráster obtenida al escanear un dibujo de ingeniería aparecen una serie de componentes 4-conexas a las que llamaremos figuras. De dichas figuras podemos extraer los contornos y aproximarlos mediante circuitos cerrados de vectores con lo que hemos obtenido una primera representación vectorial del ráster. Desde el punto de vista de su aspecto, las figuras del ráster estarán compuestas por componentes lineales y por áreas rellenas. El proceso de emparejamiento se encargará de detectar las componentes lineales de una figura y relacionarlas unas con otras a través de áreas rellenas.

De un modo intuitivo, el emparejamiento de una figura consistirá en encontrar para cada línea del ráster aquellos contornos de la figura que se encuentran a ambos lados de la línea y que la contienen, para relacionar entre sí sus aristas y así delimitar vectorialmente la citada línea. En principio, todas aquellas zonas en las que no es posible encontrar parejas para las líneas de contorno se considerarán desemparejamientos. En este punto hay que resaltar la correspondencia entre los conceptos de pareja y componente lineal y entre desemparejamiento y área rellena.

En la Figura 28 se puede ver una figura extraída de un plano sobre la que están representados todos los elementos de los que acabamos de hablar. La figura o componente 4-conexa que aparecía en el ráster original está representada como la figura negra escalonada que aparece debajo de las demás líneas. Hay que resaltar que el tamaño de esta figura ha sido ampliado 3000 veces pues el ráster original se obtuvo escaneando un plano a 300 ppp y en la ilustración cada escalón se corresponde con un solo pixel.

En la ilustración también se aprecian dos contornos, uno interior y otro exterior representados como dos polilíneas cerradas de colores rojo y azul respectivamente con las uniones entre rectas resaltadas por cuadrados de fondo blanco. Las líneas que componen las polilíneas han sido numeradas indicando un orden para las aristas que las componen. Siguiendo este orden en una polilínea cualquiera, avanzaremos en un sentido antihorario para una polilínea que represente un contorno exterior, y horario para las de contornos interiores. Hay que tener en cuenta que como norma, el contorno exterior de una figura lo consideraremos el contorno 0 y los contornos interiores se numerarán

sucesivamente a partir de este. En nuestro ejemplo, podemos observar que solo hay dos contornos, el exterior coloreado en rojo que como acabamos de indicar tendrá el número 0 y el interior de color azul que será el 1. De este modo podremos identificar unívocamente cada una de las aristas de una figura cualquiera como un par con la forma: (nº contorno, nº arista).

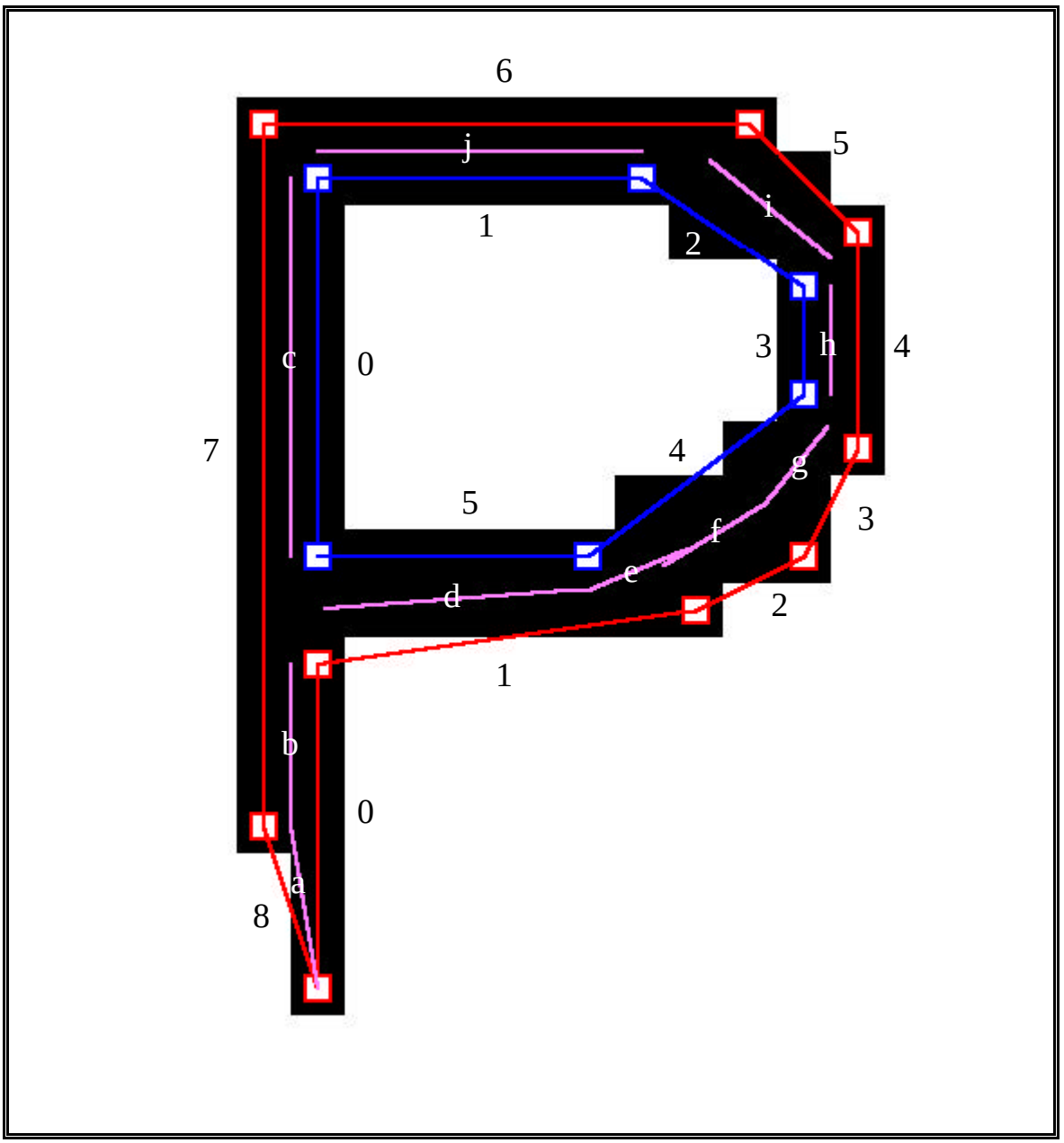


Figura 28

Las líneas de color rosa que aparecen entre las dos líneas de contorno representan las parejas que fueron halladas como resultado del proceso de emparejamiento. Así pues, podemos decir que la arista 7 del contorno exterior (0, 7) empareja con la arista 0 del mismo contorno (0, 0) y con la arista 0 del contorno interior

(1, 0) y las parejas calculadas son b y c respectivamente. También hay que resaltar que la arista (0, 7) no está emparejada en toda su longitud y por tanto aparecen 3 zonas en las que la arista no tiene pareja. Dos de estos desemparejamientos se producen en las zonas donde el arco de la P se une con el trazo vertical y se puede apreciar que en estas zonas no aparece la línea rosa que indica emparejamiento. El desemparejamiento restante está situado en el extremo inferior de la arista (0, 7) entre las parejas a y b. Este desemparejamiento resulta un poco especial porque la arista (0, 0) además de la pareja b también tiene la a y entre ellas no hay ningún espacio libre y por lo tanto desde cierto punto de vista se podría considerar que en ese punto no existe desemparejamiento. Para nosotros resultará más conveniente considerar que antes y después de un emparejamiento siempre habrá un desemparejamiento. Más adelante nos ocuparemos de desarrollar con detenimiento el concepto de desemparejamiento el cual resultará clave para posteriores desarrollos y permitirá afrontar problemas como el que aparece entre las parejas e y f. Por el momento bastará con conservar la idea intuitiva de lo que es un desemparejamiento.

6.2 Comparación de dos aristas.

Para encontrar las parejas de una arista dada a1 deberemos compararla con las demás aristas de los contornos de la figura a la que pertenece. Con el fin de facilitar el proceso de comparación, es conveniente que la arista que pretendemos comparar se encuentre en una posición prefijada. La posición que tomaremos en este caso se alcanzará con un simple giro con respecto al origen de coordenadas. El giro elegido será aquel que deje al vector que representa a la arista en posición horizontal y con sentido de izquierda a derecha.

Como es sabido, para realizar un giro basta con conocer el seno y el coseno del ángulo que pretendemos girar y en el caso de nuestro giro dichas razones trigonométricas pueden calcularse sin necesidad de conocer la medida de dicho ángulo.

En la Figura 29 se muestran las seis diferentes posiciones que puede tener una arista de un contorno. Hay que indicar que ya que todos los puntos del ráster tienen coordenadas positivas, cualquier figura se encontrará siempre en el primer cuadrante por lo que las componentes de sus aristas también tendrán siempre valores positivos. También hay que destacar que en la citada figura el origen de coordenadas se encuentra en la esquina superior izquierda y que el sentido creciente del eje de ordenadas es hacia

abajo, es decir que el primer cuadrante estará situado abajo a la derecha. Este convenio es debido a que esa es la posición por defecto del origen de coordenadas en las ventanas de Windows.

El ángulo α corresponderá al ángulo que forma nuestra arista a1 con respecto al semieje positivo de la x, mientras que β será el ángulo que pretendemos girar la citada arista para alcanzar la posición anteriormente comentada. Definiremos Δx y Δy como la diferencia entre las respectivas componentes de los puntos final e inicial de la arista a1. En la figura aparecen representados por la punta y la cola de la flecha respectivamente. Asimismo definiremos |a1| como la distancia entre los citados puntos.

Teniendo en cuenta las anteriores definiciones así como los conceptos de seno y coseno y a la vista de cada uno de los casos de la figura, podemos escribir para cada uno de ellos:

$\cos \alpha = \Delta x / \lvert a1\rvert$	$\operatorname{sen} \alpha = - \Delta y / \lvert a1\rvert$	α es opuesto a β.
$\cos \alpha = - \Delta x / \lvert a1\rvert$	$\operatorname{sen} \alpha = \Delta y / \lvert a1\rvert$	α es suplementario de β.
$\cos \alpha = \Delta x / \lvert a1\rvert$	$\operatorname{sen} \alpha = - \Delta y / \lvert a1\rvert$	α es opuesto a β.
$\cos \alpha = - \Delta x / \lvert a1\rvert$	$\operatorname{sen} \alpha = \Delta y / \lvert a1\rvert$	α es suplementario de β.
$\cos \alpha = \Delta x / \lvert a1\rvert$	$\operatorname{sen} \alpha = - \Delta y / \lvert a1\rvert$	α es opuesto a β.
$\cos \alpha = - \Delta x / \lvert a1\rvert$	$\operatorname{sen} \alpha = \Delta y / \lvert a1\rvert$	α es suplementario de β.

Teniendo en cuenta que las razones trigonométricas para los ángulos suplementarios son $\cos \beta = - \cos \alpha$ y $\operatorname{sen} \beta = \operatorname{sen} \alpha$ y que para los ángulos opuestos son $\cos \beta = \cos \alpha$ y $\operatorname{sen} \beta = - \operatorname{sen} \alpha$, si ponemos el ángulo β en función del ángulo α para todos los casos anteriores obtenemos que:

$$\cos \beta = \Delta x / |a1|$$

$$\operatorname{sen} \beta = \Delta y / |a1|$$

Con este resultado y con las siguientes formulas que dan la posición de un punto tras un giro β podemos obtener fácilmente las coordenadas de la arista en su nueva posición aplicándolas a los puntos inicial y final de la arista a1:

$$x' = x \cos \beta + y \operatorname{sen} \beta$$

$$y' = y \cos \beta - x \operatorname{sen} \beta$$

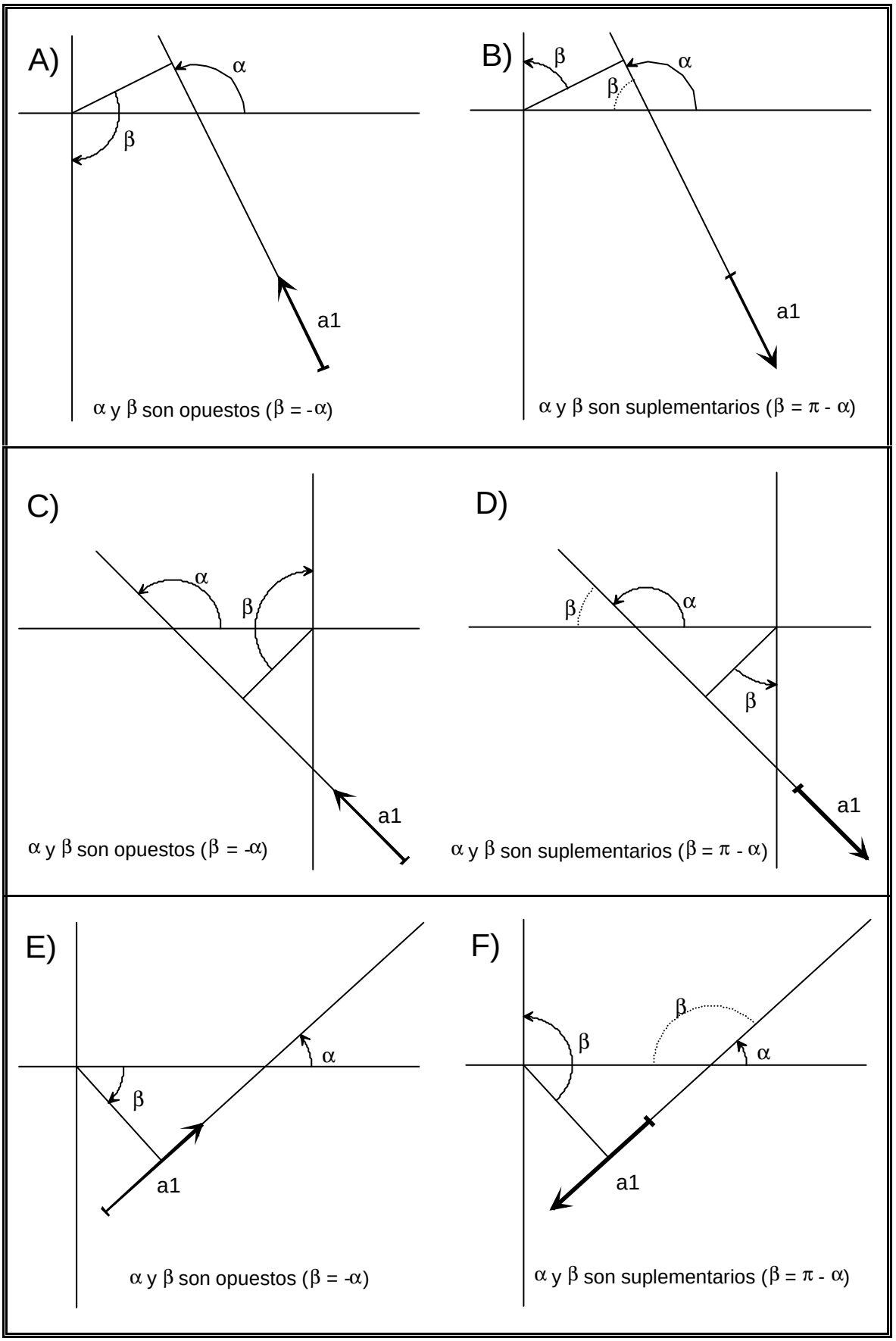

A)
α
β
a1
α y β son opuestos (β = -α)
B)
β
α
β
a1
α y β son suplementarios (β = π - α)
C)
α
β
a1
α y β son opuestos (β = -α)
D)
β
α
β
a1
α y β son suplementarios (β = π - α)
E)
α
β
a1
α y β son opuestos (β = -α)
F)
β
β
α
a1
α y β son suplementarios (β = π - α)

Figura 29

De este modo conseguimos girar la arista a1 desde cualquier posición inicial hasta la posición que pretendíamos, conociendo únicamente las coordenadas de sus extremos y realizando tan solo operaciones aritméticas elementales. Pero para poder comparar a1 con otra arista a2 deberemos someter a esta última al mismo giro que a a1, aunque en este caso ya conoceremos los valores del seno y el coseno de β y por tanto no tendremos que calcularlos de nuevo. Tras los citados giros dos aristas que emparejan tendrán el aspecto que se muestra en la Figura 30 .

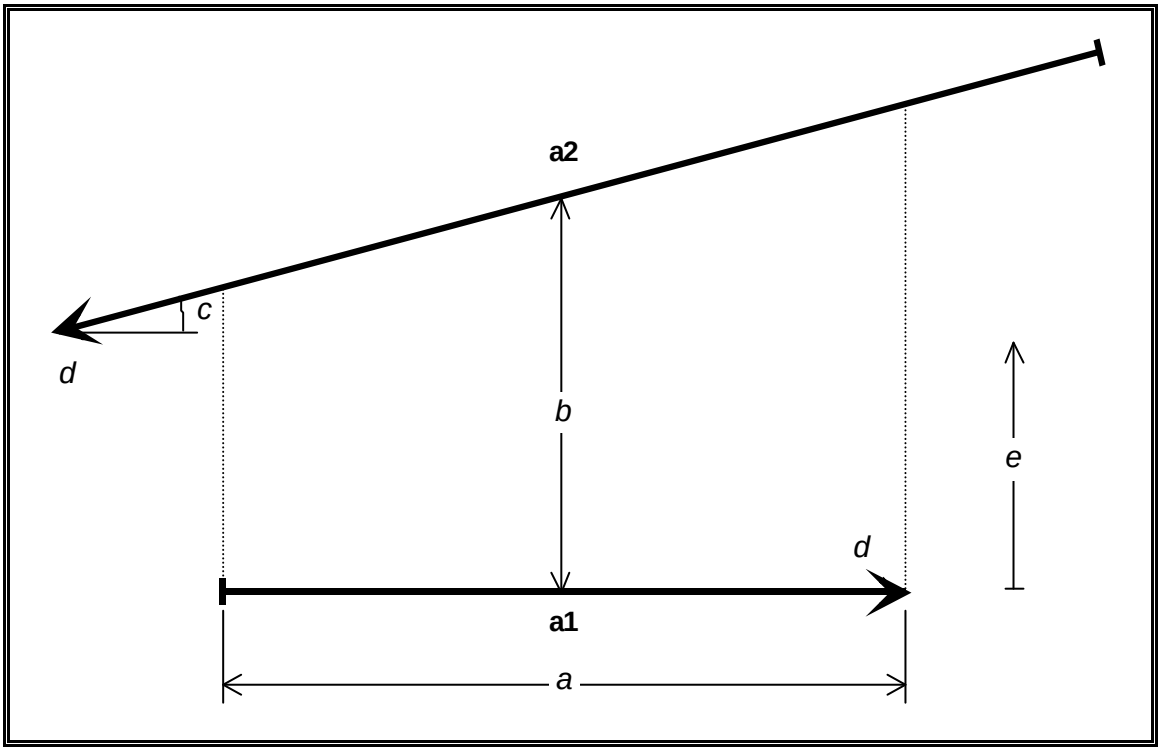


Figura 30

6.3 Criterios para el establecimiento de un emparejamiento.

En la Figura 30 se aprecian las aristas a1 y a2 en la posición alcanzada después del giro, es decir que a1 está en posición horizontal y su sentido es de izquierda a derecha, mientras que a a2 se la ha girado el mismo ángulo que a a1 y ha quedado en la posición que se muestra. También aparecen representados en la figura los criterios que nos permitirán decidir si las dos aristas emparejan o no. Los citados criterios son:

Grado de solapamiento. El grado de solapamiento coincide en este caso con la distancia a y resulta sencillo de calcular a partir de las diferencias entre las componentes x de los extremos de las aristas solapadas una vez giradas. Se pueden

descartar como parejas todas aquellas en las que no exista solapamiento o este sea demasiado pequeño.

Distancia entre aristas. La distancia entre ambas aristas (b en este caso) se medirá en el centro del área de solapamiento y representará el grosor medio de dicha área. El grosor del área de solapamiento se puede considerar como el grosor de la línea detectada. También resulta sencillo calcular esta distancia a partir de los extremos del área de solapamiento y de la pendiente de la arista a2 utilizando la ecuación punto-pendiente de la recta ($y = y0 + m\ (x - x0)$). Una vez calculada esta distancia, debemos compararla con una distancia máxima d_{max} a partir de la cual dos aristas no se considerarán pareja. La elección del valor del parámetro d_{max} resulta sencilla puesto que será el grosor máximo de la línea más gruesa que aparezca en el dibujo. Como ya veremos más adelante este valor tendrá gran influencia sobre los resultados obtenidos y resultará delicado de fijar a priori.

Ángulo entre aristas. Se corresponderá con el ángulo de a2 con respecto a la horizontal y puede ser calculado como el arco tangente de la pendiente de a2. El ángulo calculado deberá ser comparado con un ángulo α_{min} por encima del cual dos aristas no se considerarán pareja. El valor de este parámetro α_{min} resulta difícil de fijar puesto que para un mismo dibujo pueden aparecer algunas diferencias en los resultados obtenidos dependiendo de este parámetro. Un valor de 30º produce resultados más que aceptables en todos los casos que se han probado aunque de surgir algún problema se debería ajustar dicho parámetro de manera empírica.

Sentidos relativos. Aprovechando que el algoritmo de extracción de contornos asigna claramente un sentido fijo a cada uno de los contornos que componen una figura y que estos sentidos son opuestos entre contornos interiores y exteriores de una misma figura, podemos afirmar que dos aristas que se encuentran en lados opuestos de una misma línea tendrán sentidos opuestos. Si las citadas aristas pertenecen a un mismo contorno ya sea el exterior o uno interior, al estar una enfrente de otra deberán tener sentidos opuestos ya que los contornos son cerrados y todas sus aristas tienen el mismo sentido de giro (ver Figura 31 a y b). Por otra parte, si las dos líneas son de contornos diferentes y uno de los contornos es el exterior los sentidos seguirán siendo opuestos ya que las aristas de contornos interiores tienen sentido horario y las del contorno exterior lo tendrán opuesto, por lo que si ambas aristas están en lados enfrentados de una línea también tendrán

sentidos opuestos (ver Figura 31 c). Por último, si las aristas pertenecen a dos contornos interiores distintos y son de la misma línea, esto significa que la línea se encuentra entre los dos contornos. Así pues nuestras aristas también tendrán sentido horario ya que ambos contornos tienen sentido horario y están en lados opuestos de la línea (ver Figura 31 d). Puesto que el sentido de a1 está fijado de izquierda a derecha, para saber si a2 tiene el sentido opuesto (de derecha a izquierda), nos bastará con comprobar que después del giro, la componente x de la cabeza de a2 es menor que la de la cola.

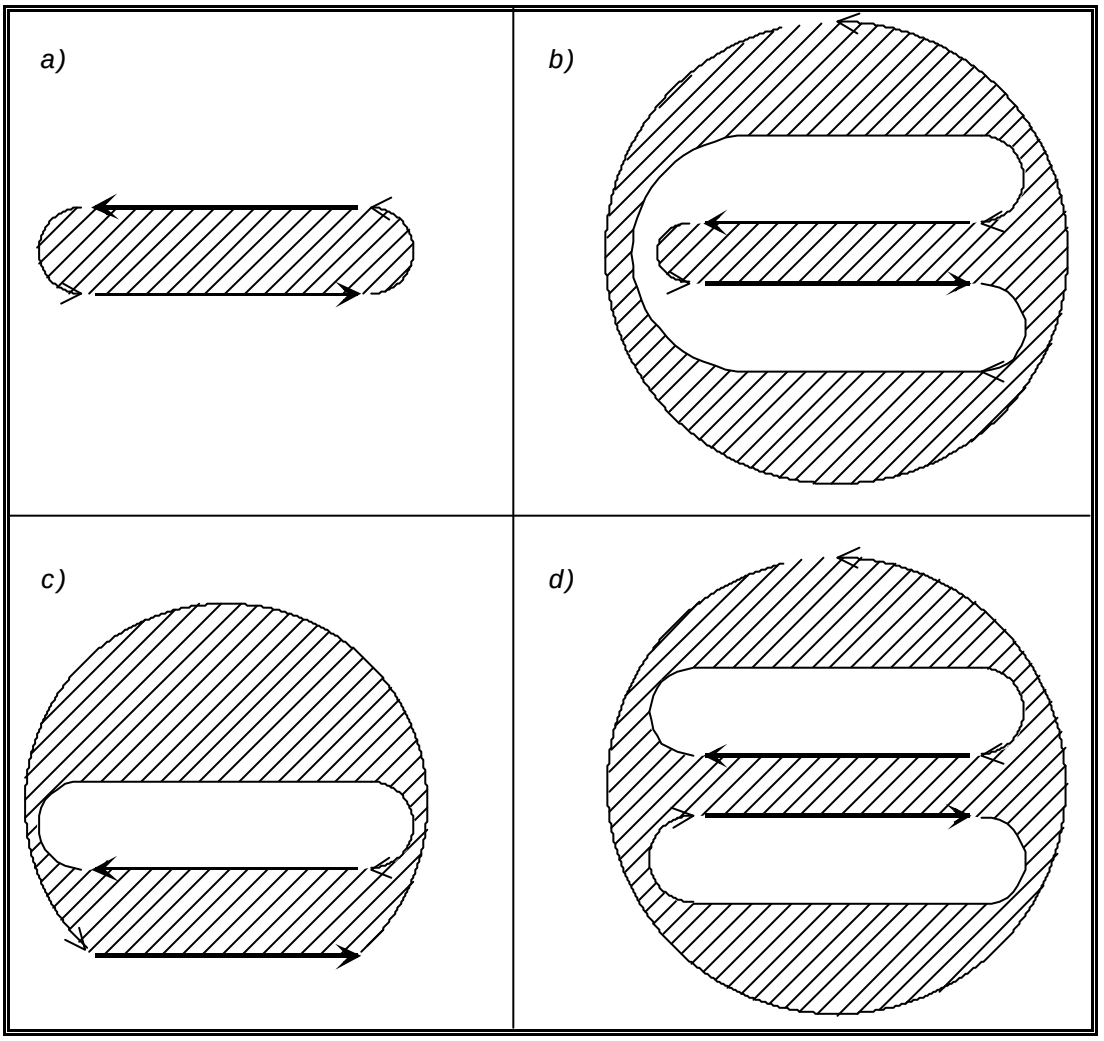


Figura 31

Lado correcto para el emparejamiento. Este criterio también hace uso del sentido de los contornos. Con el sentido que hemos fijado, las líneas siempre quedarán a la izquierda del contorno al que pertenecen según el sentido de la arista. Un contorno exterior envolverá siempre a las líneas en su interior, por lo que al tener sentido antihorario el interior del contorno siempre estará a la izquierda de cualquiera de sus aristas. Con los contornos interiores tenemos la misma situación, con la diferencia

de que en este caso las líneas siempre estarán en el exterior del contorno y este tendrá sentido horario por lo que de todos modos, también quedarán a la izquierda de las aristas del contorno según el sentido de estas. El hecho de que todas las posibles parejas de a1 queden a su izquierda según su sentido se traduce en que estas parejas aparecerán por encima de a1 una vez hayamos realizado el giro ya que el sentido de a1 lo hemos fijado de izquierda a derecha. Por tanto podremos saber si una arista es una candidata a pareja con solo comprobar que después del giro ambas componentes *y* de los extremos de a2 son mayores que la *y* de la arista a1.

6.4 Criterios adicionales.

Con los anteriores criterios se puede establecer fácilmente y con un coste reducido cuando emparejan dos líneas. Por otra parte, también aparecen algunos problemas debidos a la diferencia de grosor entre las líneas de un dibujo. Como ya hemos comentado el parámetro d_{max} se utiliza para descartar aquellas aristas que se encuentran demasiado alejadas de a1 y su valor se debe ajustar al máximo grosor de la línea más ancha. En algunos dibujos con gran variación entre el grosor de unas líneas y otras, este valor resulta demasiado grande para algunas áreas del dibujo en las que aparecen detalles muy pequeños y el grosor de las líneas es pequeño comparado con el valor del parámetro.

Un ejemplo de lo anterior lo tenemos en la Figura 32 que es la misma imagen que aparece en la Figura 28 pero con la diferencia de que en su procesamiento se ha empleado un algoritmo que utiliza únicamente los criterios comentados anteriormente. En el ráster del plano del que se extrajeron ambas imágenes aparecen líneas de un grosor de 10 pixeles por lo que el parámetro d_{max} ha sido fijado en este grosor para que las citadas líneas pudieran ser reconocidas por el algoritmo. Esto ha tenido como efecto colateral la aparición de las dos nuevas líneas de pareja del interior de la P marcadas como *X* e *Y*. Estas líneas señalan el emparejamiento de las aristas (0,1) con (0,6) y (0,6) con (0,2) respectivamente. Estos inesperados emparejamientos se producen porque en ambos casos se cumplen todos los criterios incluido el de distancia entre aristas que en este caso es el que debería habernos indicado que no son parejas. Si nos fijamos con detenimiento podemos darnos cuenta que el número de pixeles entre ambos pares de aristas es aproximadamente 8 por lo que la distancia calculada entre las aristas de estas

falsas parejas rondará este número que al ser inferior a d_{max} hace que la condición fracase.

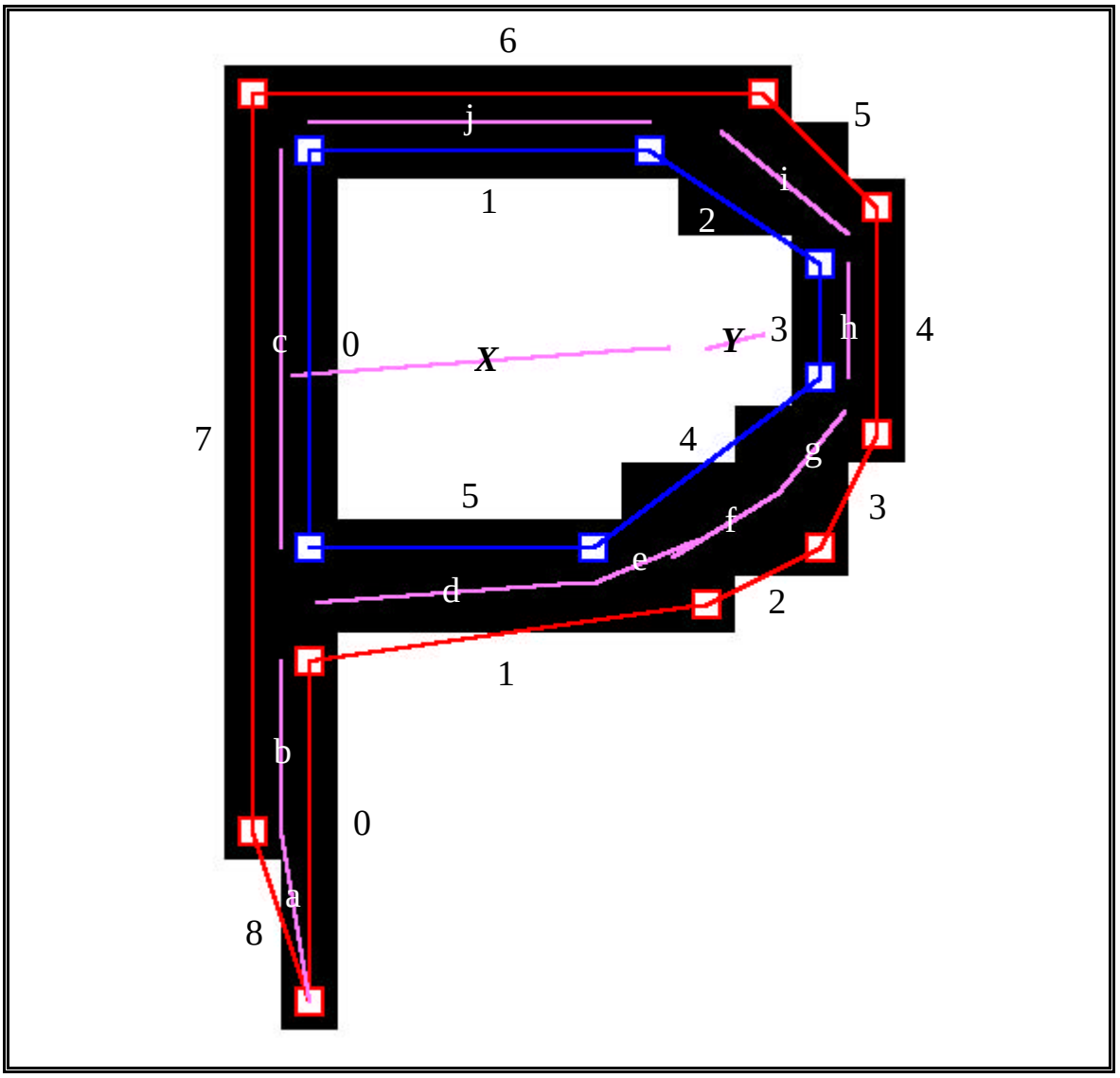


Figura 32

El problema se ha solucionado introduciendo un nuevo criterio que consistirá en comprobar entre sí todas las posibles parejas que han superado los criterios anteriores y en caso de que encontremos dos de ellas que se solapen nos quedaremos únicamente con aquella de las dos cuya distancia respecto a a1 sea menor.

En la Figura 33 aparece un esquema que ilustra la situación que acabamos de comentar para la arista *X*. Los vectores o flechas que en ella aparecen representan las aristas de los contornos. Dichos vectores se encuentran en la posición que tienen después del giro que deja a la arista a1 en posición horizontal y apuntando hacia la derecha.

La parte izquierda de la figura representa la posición en la que a1 es (0, 1) y la derecha aquella en la que lo es (0, 6). Como se puede apreciar, después del giro la posición relativa y los ángulos de unas aristas con respecto a otras se conservan aunque las posiciones absolutas de estas aristas hayan cambiado.

También están representadas con línea discontinua las proyecciones que nos ayudan a calcular los solapamientos de unas aristas con otras. En el ejemplo de la figura entre las candidatas a parejas de (0, 1) tendremos a (1, 5), (1, 4) y (0, 6) que se corresponden con las axiales d, e y *X* respectivamente y entre las candidatas de (0, 6) tendremos a (0, 2), (1, 1) y (0, 1) correspondientes a f, j y *Z*. La pareja *Z* no aparece representada en la Figura 32, esto se debe a que tenemos que elegir uno de los dos posibles emparejamientos entre (0, 1) y (0, 6) que aparecen en la Figura 33.

A partir de ahora fijaremos la notación indicando en primer lugar la arista que toma el papel de a1. Así pues, según esta notación la pareja *X* se corresponde con el par (0, 1)-(0, 6), del mismo modo, la pareja *Y* se corresponderá con las aristas (0, 6)-(0, 2) y estas son las falsas parejas que aparecen representadas en la Figura 32. La decisión de cual de los dos posibles emparejamientos entre dos aristas se toma en cada caso vendrá determinada por el orden en que se procesen las aristas dentro del algoritmo de emparejamiento, tema del que nos ocuparemos más adelante.

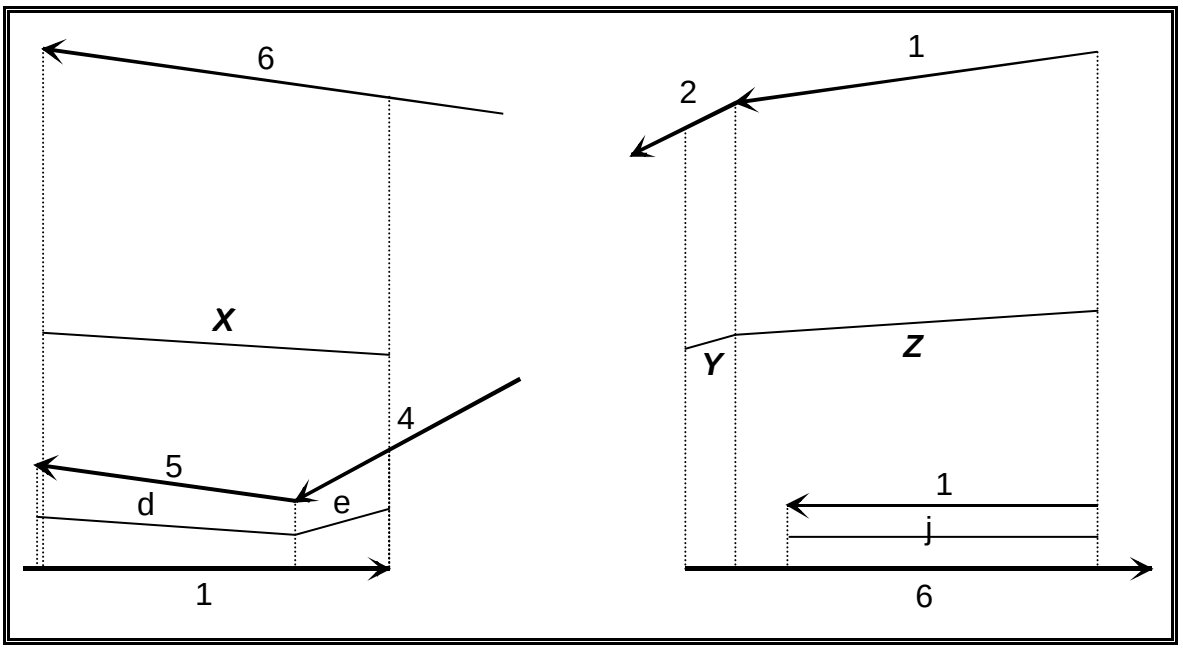


Figura 33

La pareja *X* aparece solapada con d e y pero según el nuevo criterio expuesto será rechazada como candidata a pareja de (0, 1) porque la distancia que hay entre (0, 1)

y (0, 6) es mayor que la que existe hasta cualquiera de las otras dos aristas que producen estas parejas. Una situación análoga se produciría si consideráramos el punto de vista de (0, 6) con respecto a la *Z* al solaparse las aristas (0, 1) y (1, 1).

El caso de la falsa pareja *Y* será un poco más complicado. La Figura 34 representa todas las aristas implicadas y las proyecciones que se realizan entre ellas para calcular los emparejamientos. A diferencia de la Figura 33 las aristas no han sido giradas respecto de su posición inicial para hacer más sencilla la interpretación del dibujo. Podemos ver que existen proyecciones verticales que parten de la arista (0, 6) y otras con cierto ángulo que parten de (0, 2) que son respectivamente para calcular las candidatas a parejas de dichas aristas.

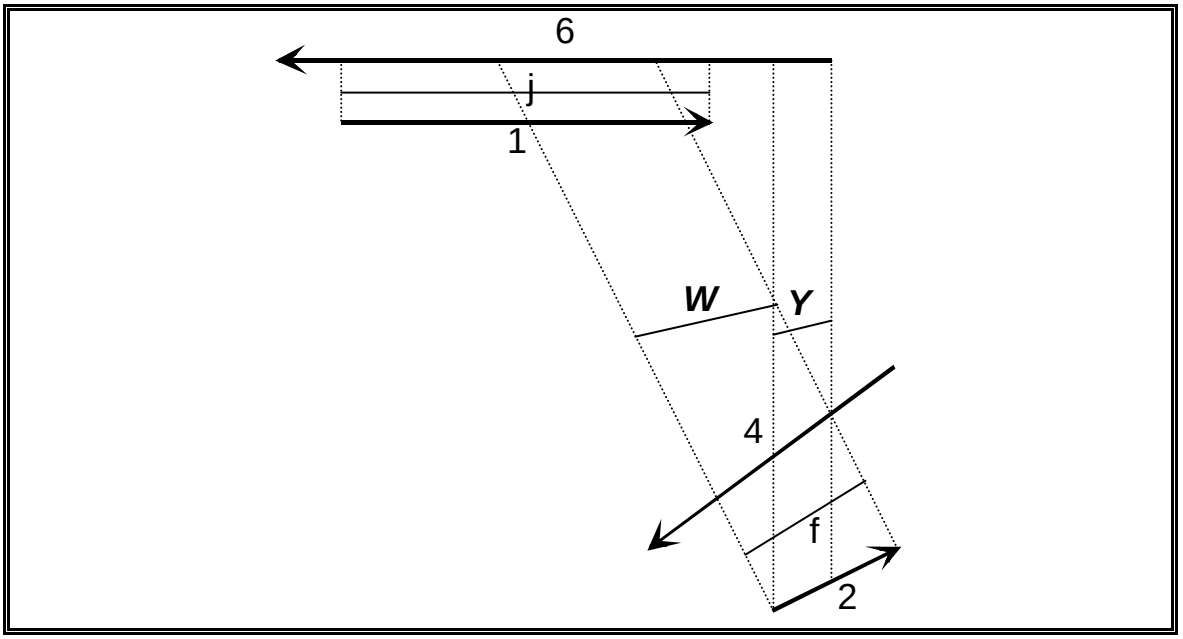


Figura 34

Las candidatas de (0, 6) son (1, 1) y (0, 2) correspondientes con j e *Y* y las candidatas de (0, 2) son f y la nueva pareja *W* generadas por (1, 4) y (0, 6). Vale la pena indicar que la falsa pareja *W* no aparece representada en la Figura 32 por las mismas razones por las que no lo hacia la *Z* en esta misma figura. La arista (0, 6) se solapa con la (1, 5) desde el punto de vista de (0, 2) por lo que según el nuevo criterio no podrá ser válida la pareja *Z*, es decir (0, 2)-(0, 6).

Ahora fijémonos en lo que sucede con la falsa pareja *Y* desde el punto de vista de (0, 6). Como se puede apreciar tanto en la Figura 33 como en la Figura 34, la *Y* no se solapa con ninguna otra de las candidatas a pareja de (0,6) por lo que desde el punto de

vista de esta resultará una pareja totalmente válida y nuestro criterio no habrá conseguido descartarla.

Para resolver esta situación consideraremos que para poder decir que dos aristas forman pareja, entre las candidatas a parejas de la primera tendrá que estar la segunda y entre las candidatas de la segunda deberá aparecer la primera. En nuestro ejemplo, para poder decir que (0, 6) y (0, 2) forman pareja, entre las candidatas a parejas de (0, 6) debería de estar (0, 2) y entre las de (0, 2) debería de estar (0, 6), pero como esta última candidata ha sido eliminada por el criterio de solapamiento con (1, 4), la única candidata válida de (0, 2) será (1, 5) a través de f y por tanto podemos concluir que no existe emparejamiento entre las dos aristas primeras.

6.5 Algoritmo de emparejamiento.

Como hemos visto, para emparejar una arista a1 de un contorno de una figura necesitaremos en primer lugar, calcular el giro que la deja en posición horizontal y también conocer las aristas a las que una vez que les hemos aplicado el dicho giro logran superar los criterios de emparejamiento. En segundo lugar para cada una de estas últimas aristas también deberemos calcular el giro que nos permite conocer sus aristas candidatas a parejas para finalmente comprobar si la arista inicial está entre este último grupo de aristas de modo que podremos decidir ya definitivamente si son pareja o no. Ya que para decidir si entre dos parejas hay un emparejamiento tenemos que calcular las candidatas a pareja de ambas, sería absurdo ocuparnos únicamente de la arista a1 y sus candidatas a pareja y no aprovechar para encontrar sus propios emparejamientos las candidatas a pareja de la segunda arista involucrada que ya tenemos calculadas. Para poder conseguir esto deberemos establecer un orden de procesamiento adecuado. Este orden consistirá en que una vez hemos decidido si ha existido emparejamiento o no, continuaremos el procesamiento por las candidatas de a1 de las que como ya sabemos, tenemos calculadas las candidatas. A continuación podemos ver una propuesta de algoritmo en pseudo-código que permite procesar figuras siguiendo este orden. El citado algoritmo es el que se ha implementado en nuestro programa, con la diferencia de que este es recursivo y la implementación realizada es iterativa por razones de eficiencia. El orden seguido por este algoritmo no exactamente el mismo que el de la versión implementada pero en la práctica las diferencias son mínimas y a efectos ilustrativos la versión recursiva aquí mostrada resulta más sencilla de entender.

```
Producir_parejas (Figura)
    Mientras queden aristas de Figura sin procesar
        a1 = siguiente arista no procesada
        Calcular las razones trigonométricas del giro para procesar a1
        Lista = Calcular_candidatas (a1, razones_trigonométricas)
        Marcar a1 procesada
        Si Lista ≠ ∅
            Emparejar ( a1, Lista)
        Fsi
    Fmientras
Fproducir_parejas

Emparejar (arista_base, lista_candidatas)
    Para Actual = cada una de las aristas de lista_candidatas
        Si Actual no está procesada
            Calcular las razones trigonométricas del giro para procesar
    Actual
            Lista_actual = Calcular_candidatas (Actual,
    razones_trigonométricas)
            Si arista_base ∈ Lista_actual
                Calcular la línea axial que representa el emparejamiento
                Borrar arista_base de Lista_actual
            Fsi
            Marcar Actual procesada
            Si Lista_actual ≠ ∅
                Emparejar (Actual, Lista_actual)
            Fsi
        Fsi
    Fpara
Femparejar

Lista_candidatas Calcular_candidatas (Arista, razones_trigonométricas)
    Aplicar el giro indicado por las razones trigonométricas a Arista
    Para Actual = cada una de las aristas de la figura que no están marcadas como
        procesadas
        Aplicar el giro indicado por las razones_trigonométricas a Actual
        Si supera los criterios de emparejamiento iniciales
            Si Actual se solapa con alguna de las de Lista_candidatas
                Si distancia (Arista, Actual) < distancia (Arista, Solapada)
                    Sustituir A por Actual en Lista_candidatas
                Fsi
            sino
                Añadir Actual a Lista_candidatas
            Fsi
        Fsi
    Fpara
```

Fcalcular_candidatas

Como podemos ver se ha desglosado el algoritmo en tres funciones para mayor claridad y se ha incluido el pseudo-código de Calcular_candidatas para mostrar el papel que ocupan los giros y los criterios de emparejamiento dentro del algoritmo.

Tabla 7. Traza de la versión recursiva del algoritmo de emparejamiento sobre la Figura 1.

Iteración	*Nivel de recursión 1*			*Nivel de recursión 2*			*Nivel de recursión 3*			*Nivel de recursión 4*		
	(arista_base, lista_candidatas)	**Actual**	***Lista_actual***	**(arista_base, lista_candidatas)**	**Actual**	***Lista_actual***	**(arista_base, lista_candidatas)**	**Actual**	***Lista_actual***	**(arista_base, lista_candidatas)**	**Actual**	***Lista_actual***
1ª	((0, 0), { (0, 7), (0, 8) })											
		(0, 7)	{ (1, 0), ~~(0, 8)~~ }	((0, 7), { (1, 0) })								
					(1, 0)	{ ~~(0, 7)~~ }						
		(0, 8)	{ ~~(0, 8)~~ }									
2ª	((0, 1), { (1, 4), (0, 6), (1, 5) })											
		(1, 4)	{ ~~(0, 1)~~, (0, 2), (0, 3) }	((1, 4), { (0, 2), (0, 3) })								
					(0, 2)	{ ~~(1, 4)~~, (0, 6) }						
							((0, 2), { (0, 6) })					
								(0, 6)	{ (1, 1), ~~(0, 2)~~ }	((0, 6), { (1, 1) })		
											(1, 1)	{ ~~(0, 6)~~ }
					(0, 3)	{ ~~(1, 4)~~ }						
		(1, 5)	{ ~~(0, 1)~~ }									
3ª	((0, 4), { (1, 3) })	(1, 3)	{ ~~(0, 4)~~ }									
4ª	((0, 5), { (1, 2) })	(1, 2)	{ ~~(0, 5)~~ }									[3]

[3] Las aristas tachadas con un aspa indican que se ha producido emparejamiento entre la arista_base y la Actual por lo que se elimina la arista_base de entre las posibles candidatas.

En la Tabla 7 podemos ver una traza del algoritmo aplicado sobre la figura que nos está sirviendo de ejemplo a lo largo de todo el capítulo. La citada traza nos muestra el orden en que se producen las llamadas y los parámetros de estas. El nivel de recursión 1 se corresponde con las llamadas que realiza la función Producir_parejas a la función Emparejar mientras que los demás niveles se corresponden con verdaderas llamadas recursivas realizadas por Emparejar sobre sí misma. Por su parte la columna Iteración indica el número de la iteración del bucle de la función Producir_parejas. La sintaxis que se ha adoptado para las llamadas consiste en un primer parámetro que será una arista expresada como de costumbre por un par (nº contorno, nº arista) y un segundo parámetro que será la lista de las posibles parejas de nuestra primera arista expresadas como una serie de aristas entre llaves. Para interpretar la tabla en el orden de ejecución de las llamadas deberemos recorrerla de izquierda a derecha y de arriba abajo.

Aunque en la tabla no aparece representada, para realizar la traza es imprescindible una estructura que nos permita marcar aquellas aristas que ya han sido procesadas y por tanto no serán consultadas de nuevo a la hora de obtener candidatas para los emparejamientos.

6.6 Desemparejamientos.

Por el momento nos hemos ocupado únicamente de los emparejamientos pero no hemos dicho nada de lo que ocurre con las partes del dibujo que no emparejan. Desde nuestro punto de vista, menospreciar la importancia de las áreas no emparejadas puede redundar en una importante perdida de información que llegado el momento puede resultar de gran importancia para evitar distorsiones y otros problemas. Además los desemparejamientos pueden ser empleados para la detección de determinados objetos de importancia tales como puntas de flecha como puede verse en [2]. De todos modos el verdadero potencial de los desemparejamientos surge cuando los combinamos con las líneas emparejadas, porque con esta combinación conseguimos explicar el dibujo en su totalidad.

Como ya indicamos al principio del capitulo consideraremos que un emparejamiento siempre estará precedido y sucedido por un desemparejamiento, aunque en algunos casos los desemparejamientos resulten artificiosos. La importancia de esta regla reside en que nos permite explicar una figura como un grafo en el que los arcos representan los emparejamientos y los nodos representan los desemparejamientos. Este grafo describirá la topología del dibujo y

nos permitirá recorrerlo elemento a elemento. Así pues, desde este enfoque los desemparejamientos serán los nexos de unión entre unos emparejamientos y otros es decir entre las líneas detectadas, permitiéndonos por tanto relacionarlas unas con otras según un orden topológico. Ahora veremos como conseguir los citados desemparejamientos.

Como ya vimos, una de las condiciones que tienen que cumplir dos aristas de contorno para ser pareja es que estén solapadas y una vez giradas, debemos calcular las proyecciones verticales de los extremos de las aristas que se solapan para poder calcular el grosor medio del emparejamiento. Expresado de manera intuitiva, para conseguir los desemparejamientos bastará con convertir las líneas de proyección en aristas ya que estas proyecciones unen los puntos extremos de los emparejamientos.

Veamos un ejemplo para aclarar la situación, en la Figura 35-a tenemos una nueva porción del ejemplo que venimos utilizando a lo largo del capítulo. En ella se puede ver, un conjunto de aristas que emparejan junto con las proyecciones que hemos utilizado para calcular los emparejamientos. También se muestran tres puntos etiquetados como a, b y c que son tres de los extremos de las proyecciones que hemos realizado para calcular los emparejamientos y que tendrán especial interés en este ejemplo. Además podemos observar que las tres parejas dejan en el centro un desemparejamiento que será el que pretendemos describir.

Como ya hemos visto, en el ejemplo de la P aparecen tan solo dos contornos que podemos describirlos como dos listas cerradas de la forma: a_0, a_1, a_2, ..., a_n, a_0 donde n será 8 para el contorno exterior y 5 para el interior. Fijaremos el siguiente convenio, cada a_i contendrá las coordenadas de la cola de la arista con su mismo índice, así pues para conseguir las coordenadas de la cabeza de una arista sólo tenemos que consultar el siguiente elemento de la lista.

Si insertamos en las anteriores listas los puntos a, b, y c obtenemos el resultado que se puede apreciar en la Figura 35-b donde se puede apreciar que los nuevos vectores que surgen de los antiguos contornos, junto con las líneas que representan las proyecciones describen el perímetro del desemparejamiento que es lo que veníamos describiendo. Así pues, para obtener nuestro desemparejamiento lo único que nos resta es convertir las líneas de nuestras proyecciones en nuevas aristas e insertarlas adecuadamente en nuestras listas de contorno.

Como hemos podido ver en la Tabla 7, la primera pareja que se calcula de las que están involucradas en nuestro actual ejemplo es (0, 7)-(0, 0). Como ya vimos

anteriormente, para obtener este emparejamiento hemos girado la arista (0, 7) hasta la horizontal del modo ya explicado y hemos tenido que calcular las proyecciones de la cabeza de (0, 7) sobre (0, 0) y de la de (0, 0) sobre (0, 7). Si pensamos en la posición que tomarían los vectores después del giro veremos que la primera proyección queda a la derecha del emparejamiento mientras que la segunda está a su izquierda. Del mismo modo, en cualquier emparejamiento podremos hablar de la proyección de la derecha o de la de la izquierda del mismo. Debido al hecho de que hemos considerado que antes y después de un emparejamiento siempre habrán desemparejamientos, también podemos decir análogamente al caso anterior que tenemos un desemparejamiento izquierdo y otro derecho para cada emparejamiento y además se corresponderán con las respectivas proyecciones de las que hemos hablado. En este momento, ya que en nuestro ejemplo el desemparejamiento queda a la izquierda de la pareja (0, 7)-(0, 0), la proyección que nos interesa a nosotros es evidentemente, la de la izquierda del emparejamiento, es decir la de la cabeza de (0, 0) sobre (0, 7).

Ya que el sentido de la arista horizontal después del giro lo hemos fijado de izquierda a derecha, podemos decir que la proyección de la izquierda siempre tendrá sentido “hacia arriba” mientras que la de la derecha lo tendrá “hacia abajo”, o dicho de otro modo, la nueva arista generada a partir de la proyección izquierda tendrá su cola en la arista a1 ((0, 7) en nuestro caso) y su cabeza estará en la pareja de a1 ((0, 0) del ejemplo) en tanto que la de la derecha tendrá sentido contrario (de (0, 0) a (0, 7)).

Según lo que acabamos de indicar, para añadir la arista procedente de la proyección izquierda, bastará con modificar las listas indicando que después de la arista (0, 7) va una nueva arista cuya cola será el punto a y a continuación de esta viene la cabeza de (0, 0). Supongamos que partimos de la siguiente lista: cola(0, 7), cola(0, 8), cola(0, 0), cola(0, 1), ... , cola(0, 7). También la podemos expresar de manera equivalente como: cola(0, 7), cabeza(0, 7), cola(0, 0), cabeza(0, 0), ... , cola(0, 7). Tras realizar las modificaciones que acabamos de indicar obtendremos: cola(0, 7), a, cabeza(0, 0), ... , cola(0, 7). Pero como podemos ver nos ha quedado un trozo de lista abierta que es: cabeza(0, 7), cola(0, 0). Por tanto en la parte derecha deberemos realizar modificaciones análogas para conseguir cerrar la anterior lista: cabeza(0, 7), cola(0, 0), proyección de cabeza(0, 7), cabeza(0, 7).

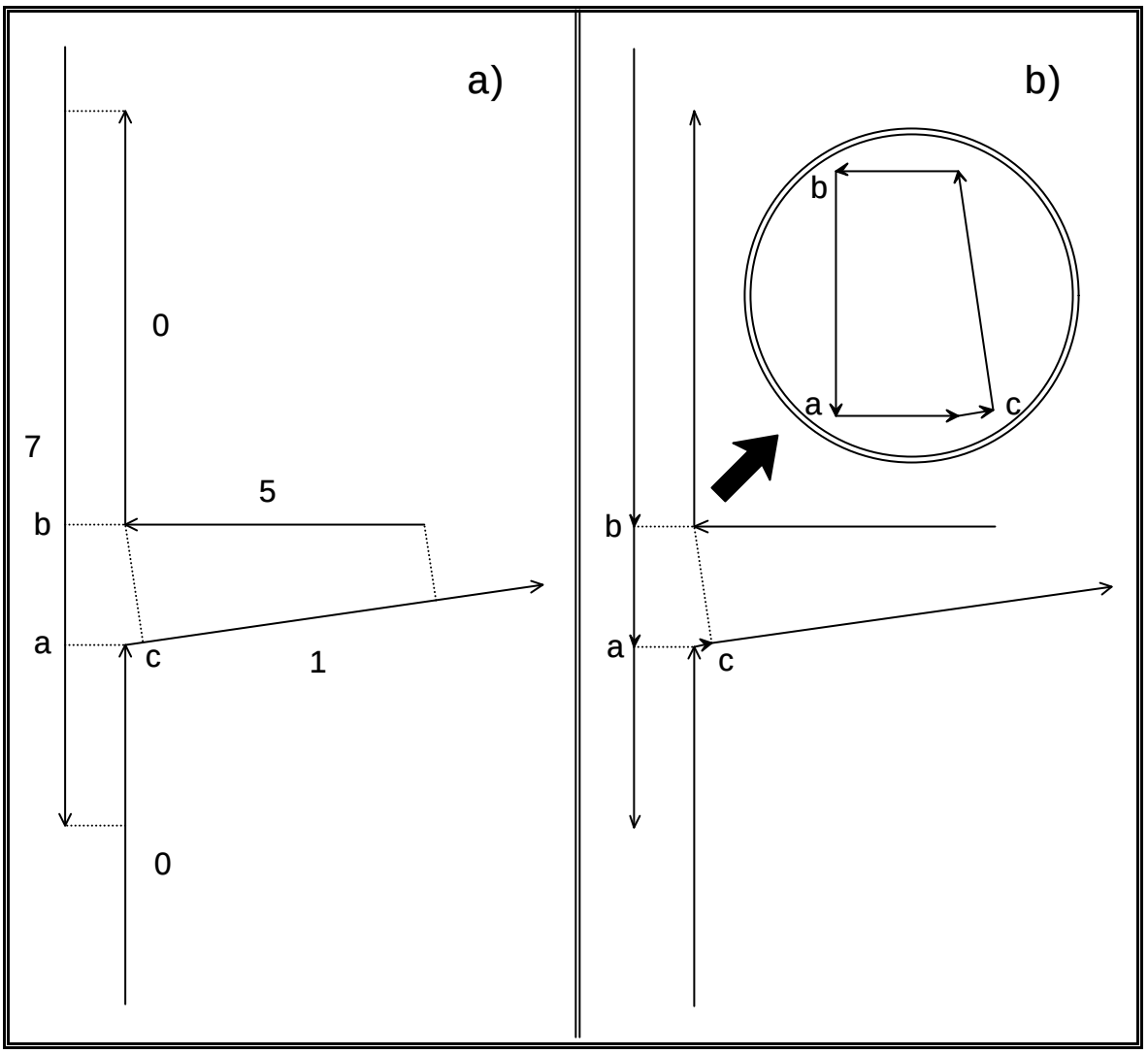


Figura 35

De acuerdo con lo que acabamos de exponer, cada vez que encontremos un emparejamiento generaremos dos sublistas que se corresponderán con los desemparejamientos izquierdo y derecho del emparejamiento. Desde cierto punto de vista podemos considerar que inicialmente partimos de un gran desemparejamiento que comprende todos los contornos de la figura y a medida que van apareciendo parejas entre sus aristas lo vamos descomponiendo en otros desemparejamientos más pequeños hasta que ya no surjan más parejas.

Continuando el proceso en nuestro ejemplo, la siguiente pareja de interés que se calculará será (0, 7)-(1, 0) por lo que se combinarán la lista procedente del desemparejamiento izquierdo de la pareja (0, 7)-(0, 0) con la lista del contorno interior. La lista resultante de este proceso será: cola(1, 0), b, a, cola(0, 1), ... , cola(0, 7), proyección de cabeza(1, 0), cabeza(1, 0), ... , cola(1, 0). Como podemos observar en este caso y a diferencia del anterior tras procesar ambos lados del emparejamiento

únicamente aparece una lista, o dicho de otra forma los desemparejamientos izquierdo y derecho son el mismo. Por último, se calcula la pareja (0, 1)-(1, 5) y al procesar su parte derecha obtenemos la lista: cola(1, 5), proyección de cola(1, 5), cabeza(0, 1), ... , cola(0, 7), proyección de cabeza(1, 0), cabeza(1, 0), ... , cola(1, 5). En la parte izquierda obtendremos la lista: cola(0, 1), c, cola(1, 0), b, a, cola(0, 1). Esta última lista será la que finalmente representará el desemparejamiento que estamos calculando, mientras que la otra, seguirá evolucionando conforme vayan apareciendo nuevas parejas, del mismo modo que lo ha hecho hasta el momento.

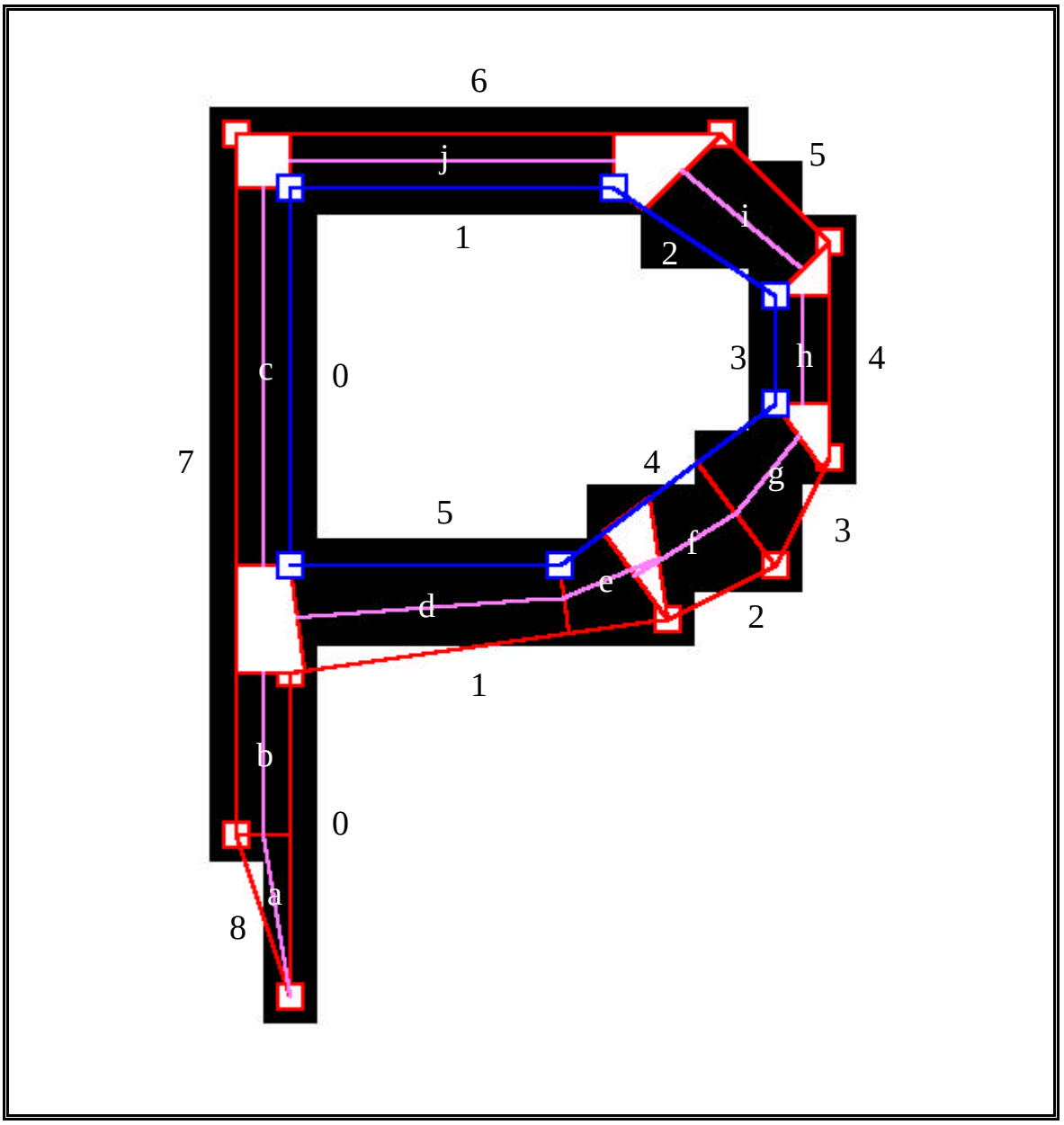


Figura 36

Pero no en todos los casos los desemparejamientos van a resultar tan sencillos como el caso que acabamos de ver. En la misma figura que nos está sirviendo de ejemplo a lo largo de todo el capítulo podemos observar un desemparejamiento un poco más extraño entre las parejas etiquetadas como e y f. En la Figura 36 aparecen representados los desemparejamientos que se calculan en la figura de ejemplo como polígonos de perímetro rojo y fondo blanco.

Si observamos el desemparejamiento al que nos acabamos de referir podemos observar que las parejas detectadas se solapan con el área desemparejada debido a que las proyecciones de ambas parejas se solapaban. En principio este hecho no es ningún problema pero habrá que tener en cuenta para posteriores desarrollos que en algunas ocasiones los perímetros que describen los desemparejamientos no tienen significado poligonal porque las aristas que los componen llegan incluso a cruzarse unas sobre otras produciendo formas anómalas. Un ejemplo de esto lo tenemos en la Figura 37 en la que podemos observar una de estas anomalías en el desemparejamiento con esquinas en los puntos 8, 7, 15, 16. La anomalía tiene su origen en que hay dos áreas emparejadas solapadas entre sí del mismo modo que ocurría con las parejas e y f de la Figura 36. En este caso el solapamiento no es tan grande como el de la citada figura y por tanto no llegan a superponerse las líneas axiales de los emparejamientos. Las parejas superpuestas en este caso, tienen sus extremos en (16- 17)-(6- 8) y en (14- 15)-(7- 8). Hay que destacar que en el citado desemparejamiento existe una sola arista entre los puntos 16 y 8 y entre 7 y 15. Además podemos ver que el punto 8 pertenece a dos desemparejamientos, es decir que pertenecerá a dos listas. Dichos desemparejamientos son el ya citado 8, 7, 15, 16 y el 13, 14, 8. Para resolver este problema en nuestra implementación deberemos realizar una copia de este punto para poder incluirlo en la lista del segundo desemparejamiento. Además deberemos cambiar el sentido del trozo de arista que aparece entre los puntos 7 y 8 que según el sentido de los contornos exteriores, en este caso debería ser de derecha a izquierda pero para que el desemparejamiento tenga consistencia como un circuito deberá ser de derecha a izquierda como vemos en el detalle de la Figura 37.

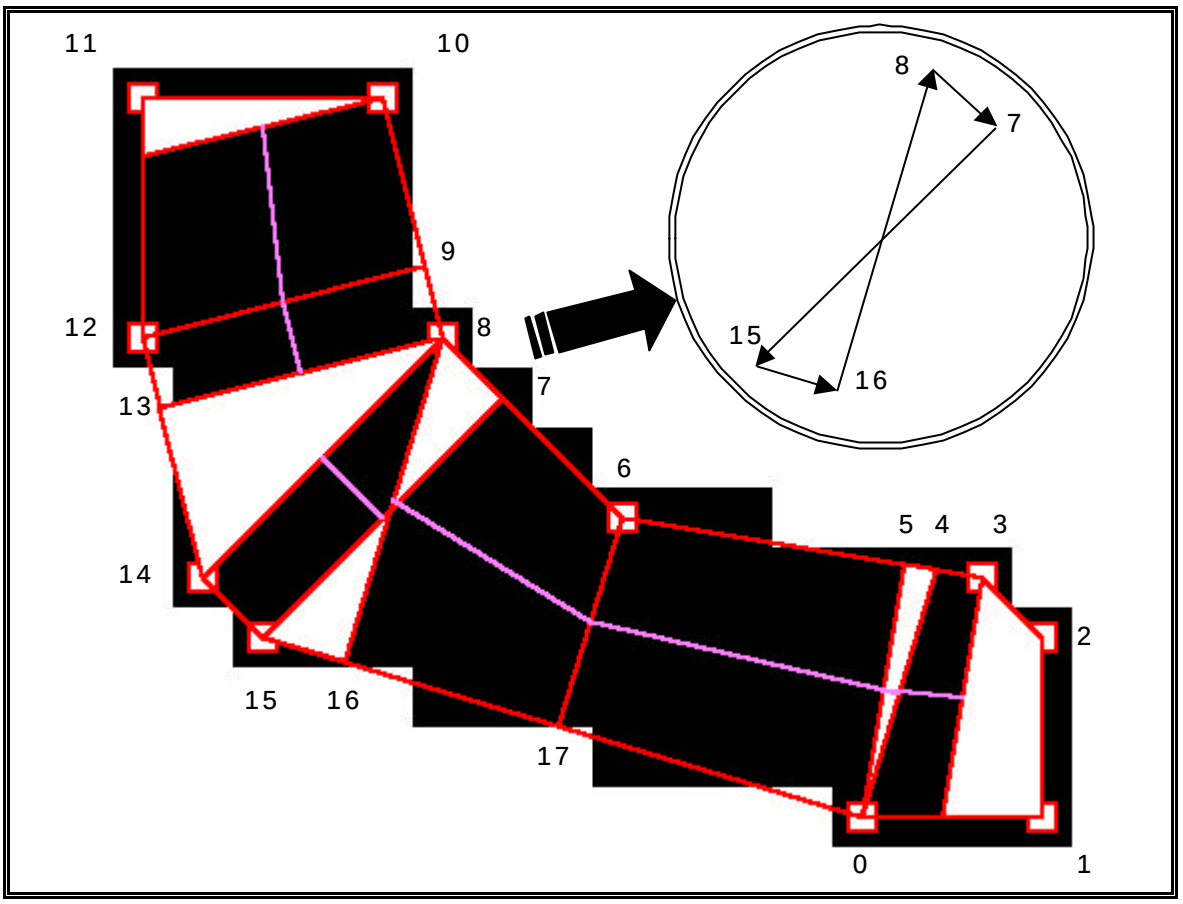


Figura 37

En nuestro programa para cada figura pretendemos obtener dos listas relacionadas entre sí. La primera contendrá todas las parejas que se encuentren en la figura, mientras que la segunda contendrá todos los desemparejamientos. Una pareja de la lista de parejas relacionará entre sí a los dos contornos que la producen con la distancia que existe entre ellos, que ha sido calculada como uno de los criterios de emparejamiento y constituirá el grosor de la línea detectada. Como ya hemos visto, una pareja también relaciona dos desemparejamientos que hemos llamado desemparejamiento derecho e izquierdo y esta relación también deberá estar presente en la estructura de datos de nuestra lista de parejas. Por su parte la lista de desemparejamientos contendrá en cada uno de sus elementos la lista con los vectores que describen su perímetro y a su vez cada uno de estos vectores nos indicará si proviene de un contorno o de transformar una proyección en arista y en este último caso la pareja que generó la proyección. De este modo podremos conocer para cada desemparejamiento que parejas intervienen en su generación o visto desde otro punto de vista, que líneas confluyen en él. También es conveniente almacenar el número de aristas de perímetro de cada desemparejamiento así como el número de ellas que

provienen de líneas confluentes porque este último parámetro nos puede servir para realizar fácilmente una primera clasificación de los desemparejamientos.

Para conseguir generar la lista de desemparejamientos utilizaremos un proceso en dos fases. La primera fase será la generación de las listas de vectores que describen el perímetro de los desemparejamientos. A medida que se van calculando las parejas de una arista las iremos introduciendo en una lista ordenándolas según la posición del solapamiento que producen sobre la arista girada hasta la horizontal y una vez obtenida esta lista ordenada se recorrerá aplicando el proceso que hemos visto anteriormente el cual convertía las proyecciones de los extremos de cada solapamiento en aristas. La segunda parte del proceso se realizará una vez hayamos calculado todas las parejas de la figura y por tanto ya estarán calculadas todas las listas de los desemparejamientos de esa figura. Consistirá en recorrer la lista de parejas de la figura e ir añadiendo los desemparejamientos izquierdo y derecho de cada pareja a la lista de desemparejamientos de la figura. Deberemos tener cuidado de establecer algún mecanismo para no añadir varias veces el mismo desemparejamiento a la lista ya que un desemparejamiento suele relacionar a mas de una pareja y por tanto al recorrer la lista de parejas podemos acceder desde varias de ellas al mismo desemparejamiento.

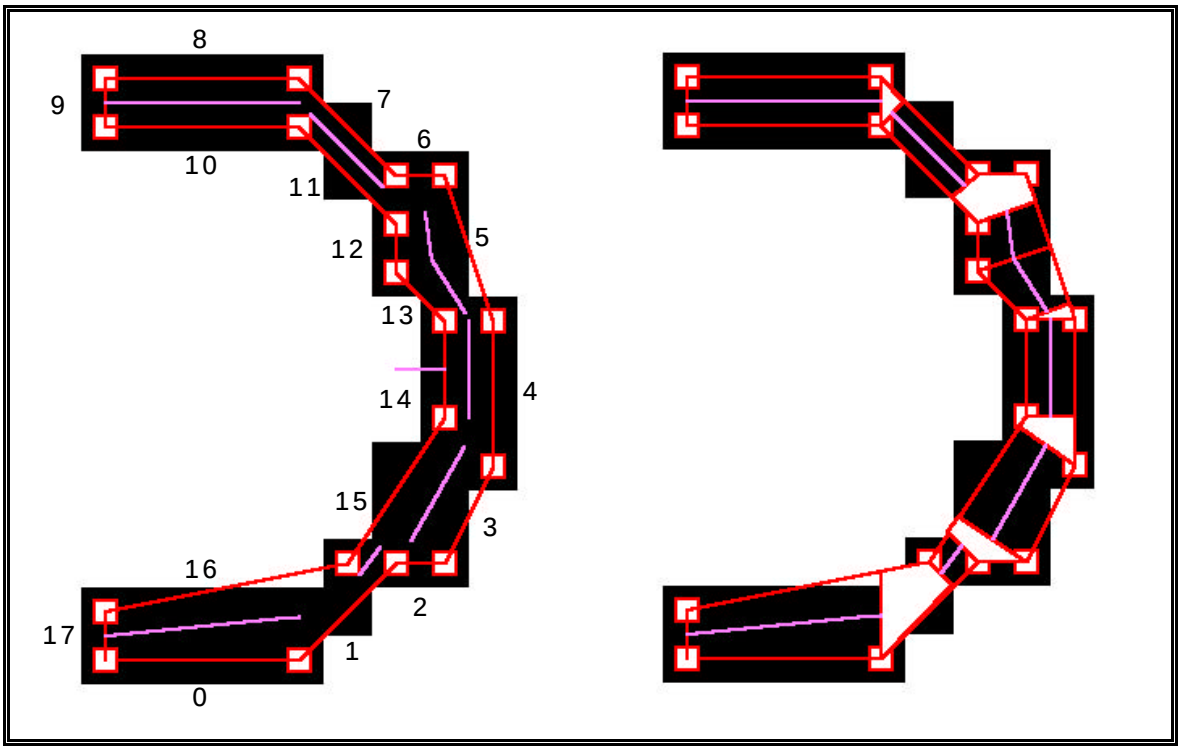


Figura 38

En la Figura 38 a la izquierda se puede observar una figura en la que el procesamiento que ha recibido ha llegado únicamente hasta el emparejamiento pero sin procesar los desemparejamientos y a la derecha tenemos la misma figura pero con el proceso de generación de desemparejamientos aplicado. Esta figura nos servirá para entender mejor el procesamiento de los desemparejamientos y para introducir un nuevo problema. Se puede apreciar que en el centro de la figura de la izquierda aparece un emparejamiento incorrecto entre las aristas 6 y 2. Dicho emparejamiento es similar a los que comentamos con anterioridad pero no ha sido eliminado por los criterios de emparejamiento ya expuestos dado que dicho emparejamiento no se solapa con ninguna otra pareja de estas aristas ya que es la única. Como se puede ver en la figura de la derecha el falso emparejamiento ha sido detectado y eliminado durante la segunda fase del proceso de generación de desemparejamientos.

Para mejorar la comprensión del proceso de generación de desemparejamientos incluimos la Figura 39 y la Figura 40. En los esquemas del **a** al **j** podemos observar la secuencia que sigue la primera fase del proceso de generación de desemparejamientos. El esquema marcado como **k** corresponde a la figura una vez se ha completado el proceso en su totalidad y se puede apreciar como se ha detectado y corregido el error producido por la falsa pareja. Por último, el esquema **l** se ha incluido para que sea más sencillo apreciar sobre el contorno original las proyecciones que se calculan durante el proceso. En estas figuras las flechas de mayor grosor simbolizan las aristas procedentes de los contornos originales, mientras que las delgadas indican que proceden de una proyección.

Como puede apreciar el esquema **a** de la Figura 39 muestra el desemparejamiento inicial formado por todos los contornos de la figura que constituye la situación previa al calculo de cualquier pareja. El esquema b nos muestra la situación después de haberse procesado la pareja (0, 0)-(0, 16).En adelante por simplicidad y aprovechando que en el ejemplo hay un solo contorno nos referiremos a esta pareja como 0-16 y las demás parejas y aristas se expresarán del mismo modo utilizando esta notación simplificada. En el citado esquema podemos ver como la parte solapada de ambas aristas ha desaparecido y la única parte no solapada que pertenecía a la arista 16 ha pasado a formar parte del desemparejamiento derecho de esta pareja mientras que el izquierdo ha quedado constituido por dos aristas superpuestas con sentidos opuestos.

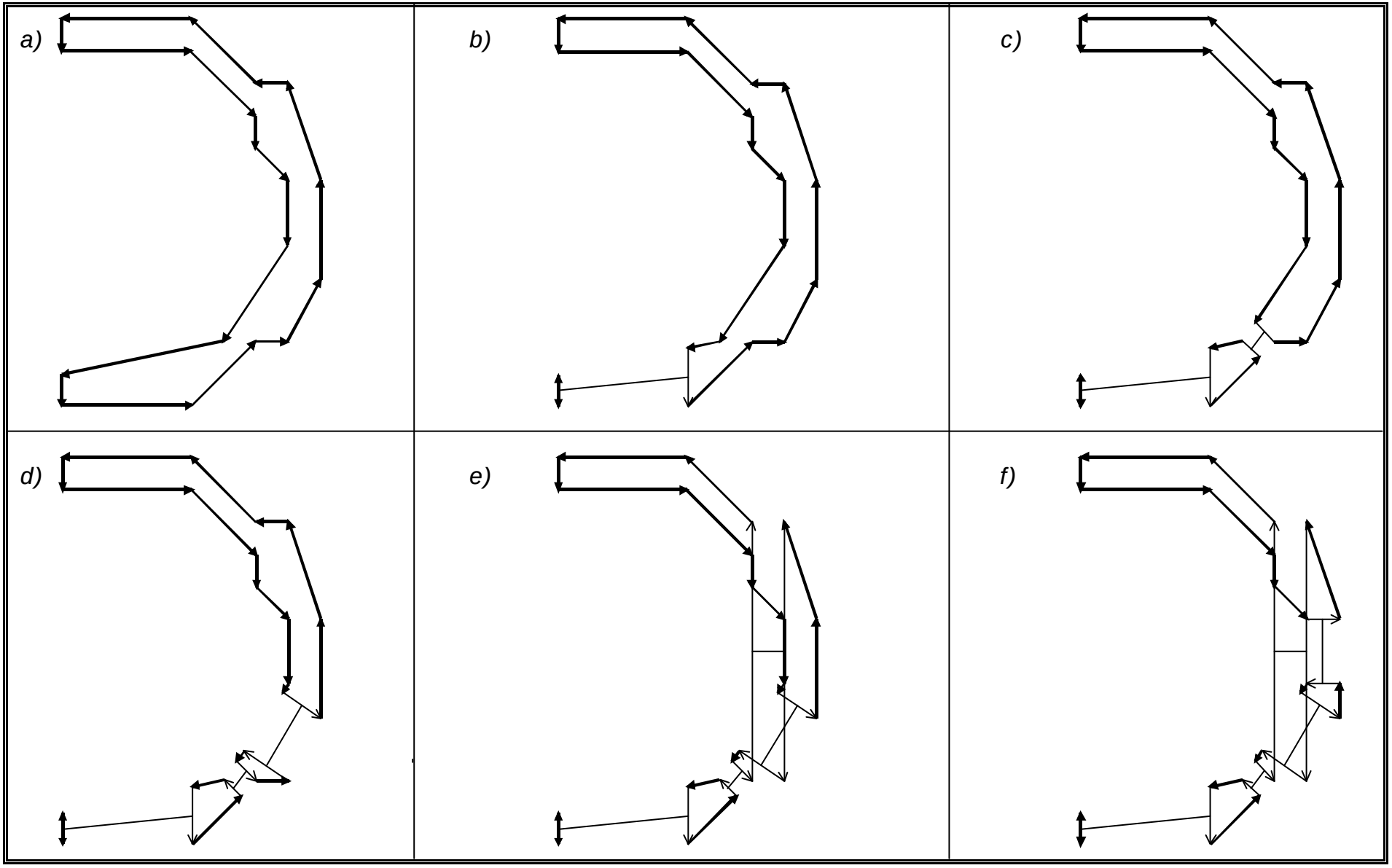


Figura 39

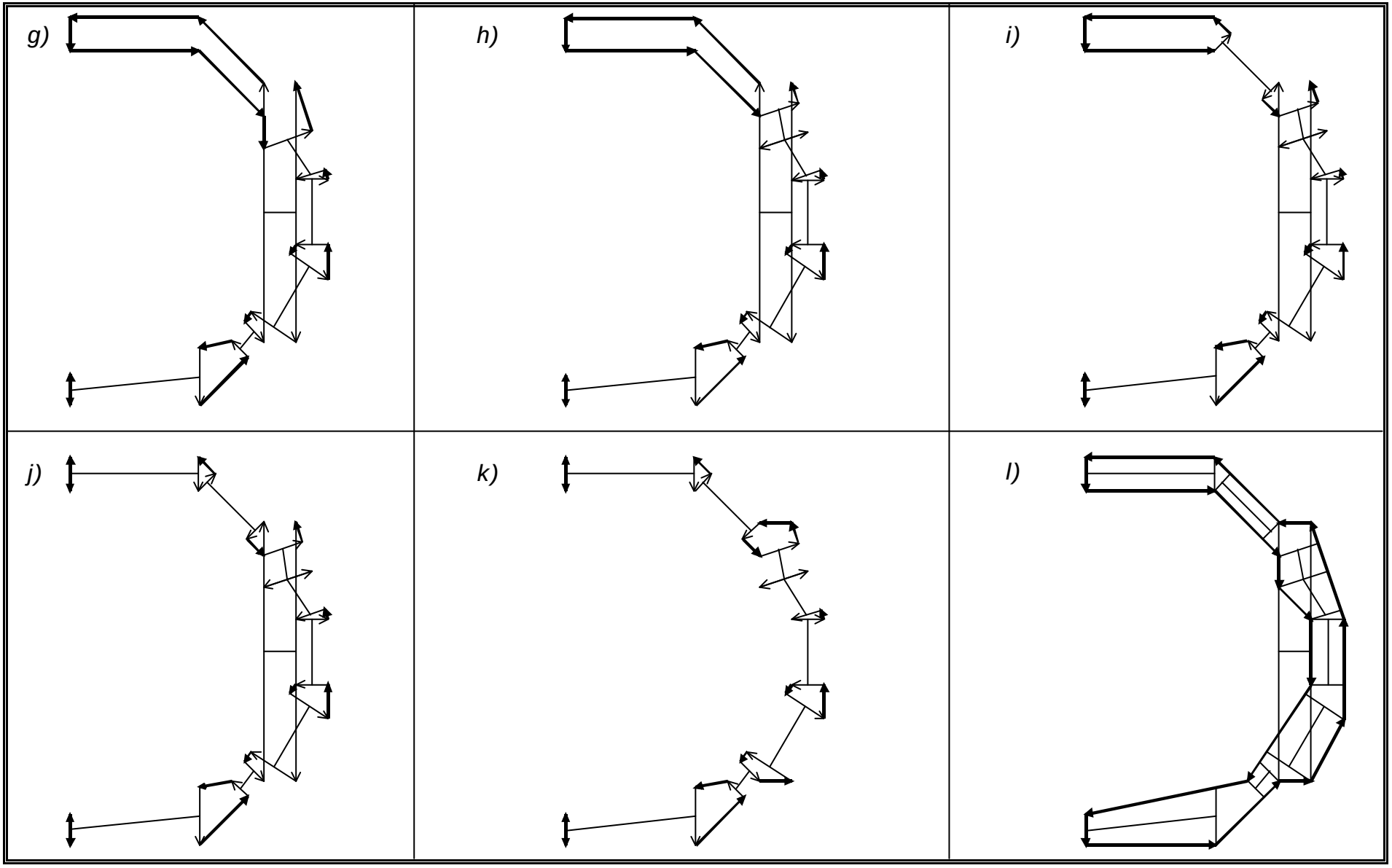


Figura 40

Los esquemas **c** y **d** nos muestran los resultados de los emparejamientos 1-15 y 15-3 que son similares al ya comentado. Mucho más interesante resulta el esquema **e** que nos muestra el momento en que aparece la falsa pareja 2-6. En él podemos apreciar que la pareja supera todos los criterios establecidos para considerarla una pareja válida y como tal es tratada, es decir que en su parte izquierda se transforma la proyección en una arista con sentido hacia arriba y en la derecha lo hace en una con sentido hacia abajo como se ha venido haciendo hasta ahora en todas las parejas.

El resto de los esquemas de la Figura 39 y los de la Figura 40 hasta el **j** no muestran nada digno de mención excepto el orden en el que van surgiendo las parejas y por tanto sus desemparejamientos asociados.

El esquema **k** representa el resultado después de aplicada la segunda fase del proceso. En esta fase, realizamos un recorrido de todas las parejas y a través de ellas alcanzamos los desemparejamientos. Cada vez que encontramos un desemparejamiento nuevo que no aparece en la lista que estamos creando, deberemos chequearlo para ver si no contiene falsas parejas y en caso de ser correcto lo incluiremos en nuestra lista. Todas aristas falsas observadas se caracterizan por ser interiores a un desemparejamiento, dicho de otro modo, los desemparejamientos izquierdo y derecho de la falsa pareja serán el mismo. La anterior caracterización será empleada para detectar las citadas falsas parejas del siguiente modo: se realizará un recorrido del circuito de aristas que constituyen el perímetro de un desemparejamiento, en el momento que aparezca una arista que converge en el desemparejamiento se añadirá a una lista después de comprobar que no había sido añadida anteriormente, en cuyo caso significaría que la pareja era falsa. Una vez detectada una falsa pareja, deberemos restaurar las aristas de contorno que originaron el falso emparejamiento inicial y eliminar la pareja proceso en el cual se generarán dos nuevos desemparejamientos. Después se iniciará un nuevo chequeo del perímetro de los dos nuevos desemparejamientos para buscar otras posibles falsas parejas en cada uno de ellos. Todo el proceso anterior se puede repetir recursívamente en cada desemparejamiento hasta el momento en que no aparezcan nuevas falsas parejas.

En la parte izquierda de la Figura 41 se puede ver el falso desemparejamiento en el momento en que se inicia la segunda fase del proceso de generación de desemparejamientos y en su parte derecha podemos ver el resultado final del proceso. Supondremos que empezamos el procesamiento de este desemparejamiento por la arista etiquetada 1, es decir que lo alcanzaremos desde la parte derecha del emparejamiento entre las aristas 1-15 de la Figura 38. Como la arista 1 proviene de un emparejamiento

introduciremos el emparejamiento que la originó en una lista que llamaremos lista de parejas confluentes que incluirá todas las parejas que convergen en el desemparejamiento. A partir de este punto iremos visitando aristas sucesivas hasta encontrar alguna que haya sido generada a partir de una proyección y no provenga de una arista de contorno. En nuestro caso esto sucede por primera vez con la arista 2 y en ese momento almacenaremos en la lista de parejas confluentes la pareja que la generó no sin antes comprobar que esa pareja no se encuentra ya en la lista. Posteriormente entrarán en la lista las parejas de las aristas 3 y 5. Finalmente llegaremos a la arista 7 y al intentar introducir la pareja de la que procede en la lista, nos daremos cuenta que ya está en ella puesto que ya la habíamos incluido al procesar la arista 2. En este momento conocemos que existe una falsa pareja en el desemparejamiento, pero es posible que existan otras, por eso seguimos con el proceso y en caso de aparecer más elegiremos para ser eliminada aquella pareja que tenga mayor grosor.

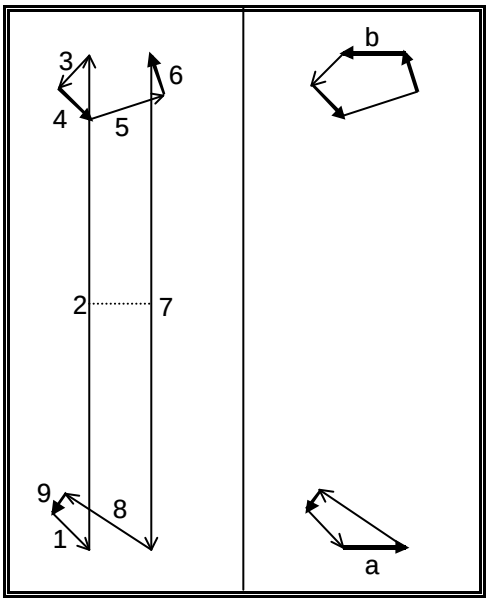


Figura 41

En nuestro ejemplo solo existe una falsa pareja y una vez detectada debemos deshacer las falsas aristas 2 y 7 uniendo en cada uno de los casos la cola de una arista con la cabeza de la otra, con lo que regeneraremos las aristas a y b de la derecha de la Figura 38. También deberemos eliminar la pareja de la lista de parejas de la figura y además tendremos que comprobar que eliminando estas aristas el problema se ha resuelto. Para asegurarnos de esto deberemos repetir todo el proceso recursivamente en los dos nuevos desemparejamientos que hemos obtenido en el caso de que exista más de una arista interior. Como podemos ver en nuestro ejemplo solo existe una arista interior

y por tanto no se realizarán llamadas recursivas y por tanto los nuevos desemparejamientos podrán añadirse a la lista de desemparejamientos.

6.7 La estructura de contornos emparejados.

La estructura de contornos emparejados es el resultado del proceso de emparejamiento e intenta sintetizar la información más relevante que aparece en el ráster.

La Figura 42 muestra una sencilla componente conexa de la que representaremos la estructura de datos que se obtiene al aplicar el algoritmo expuesto en los apartados anteriores. Las cifras que aparecen en negrita representan la numeración que se ha obtenido para las aristas del único polígono de contorno que se obtiene de esta figura. Los pares entre paréntesis indican las coordenadas de las esquinas de los polígonos de contorno (representados en color rojo) y de los polígonos del contorno de las áreas no emparejadas(color verde). Las líneas axiales fruto del emparejamiento se representan en color rosa.

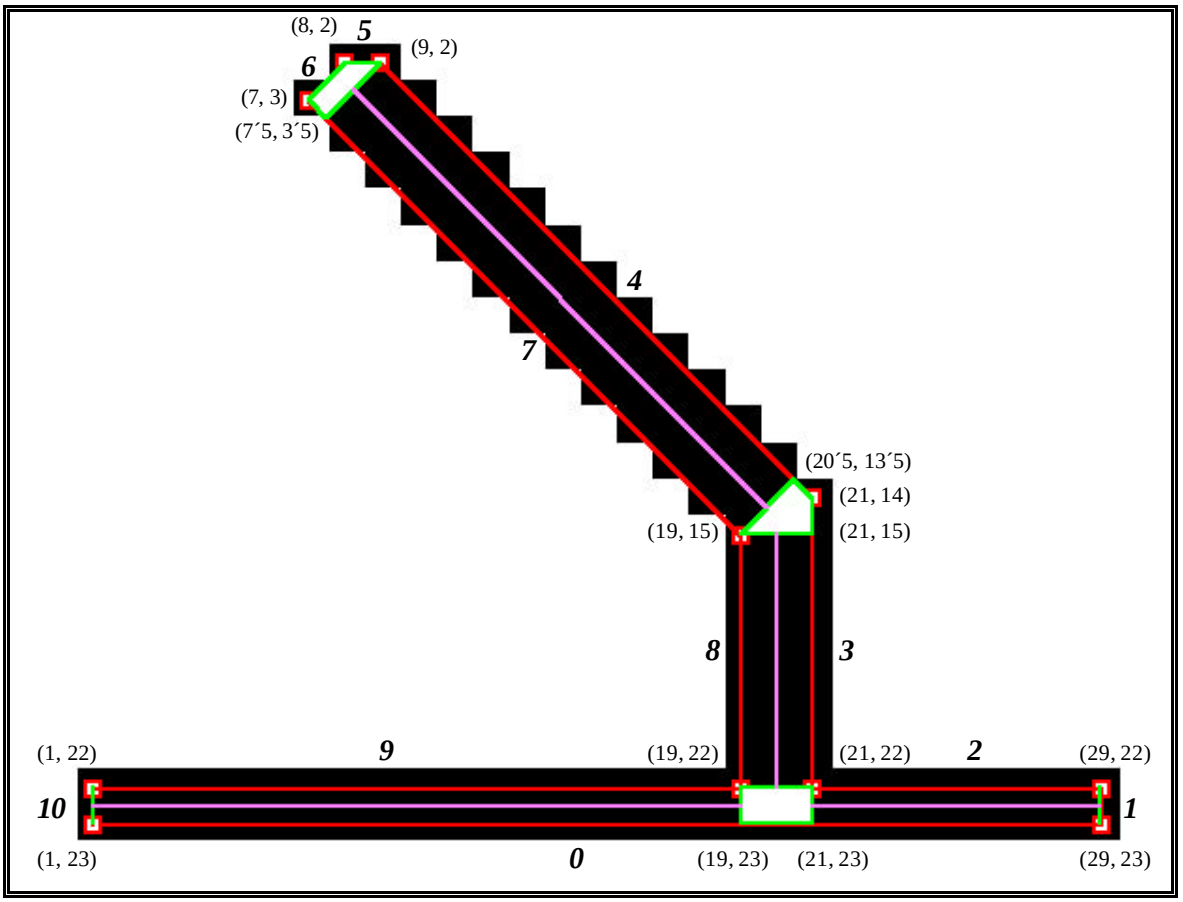


Figura 42

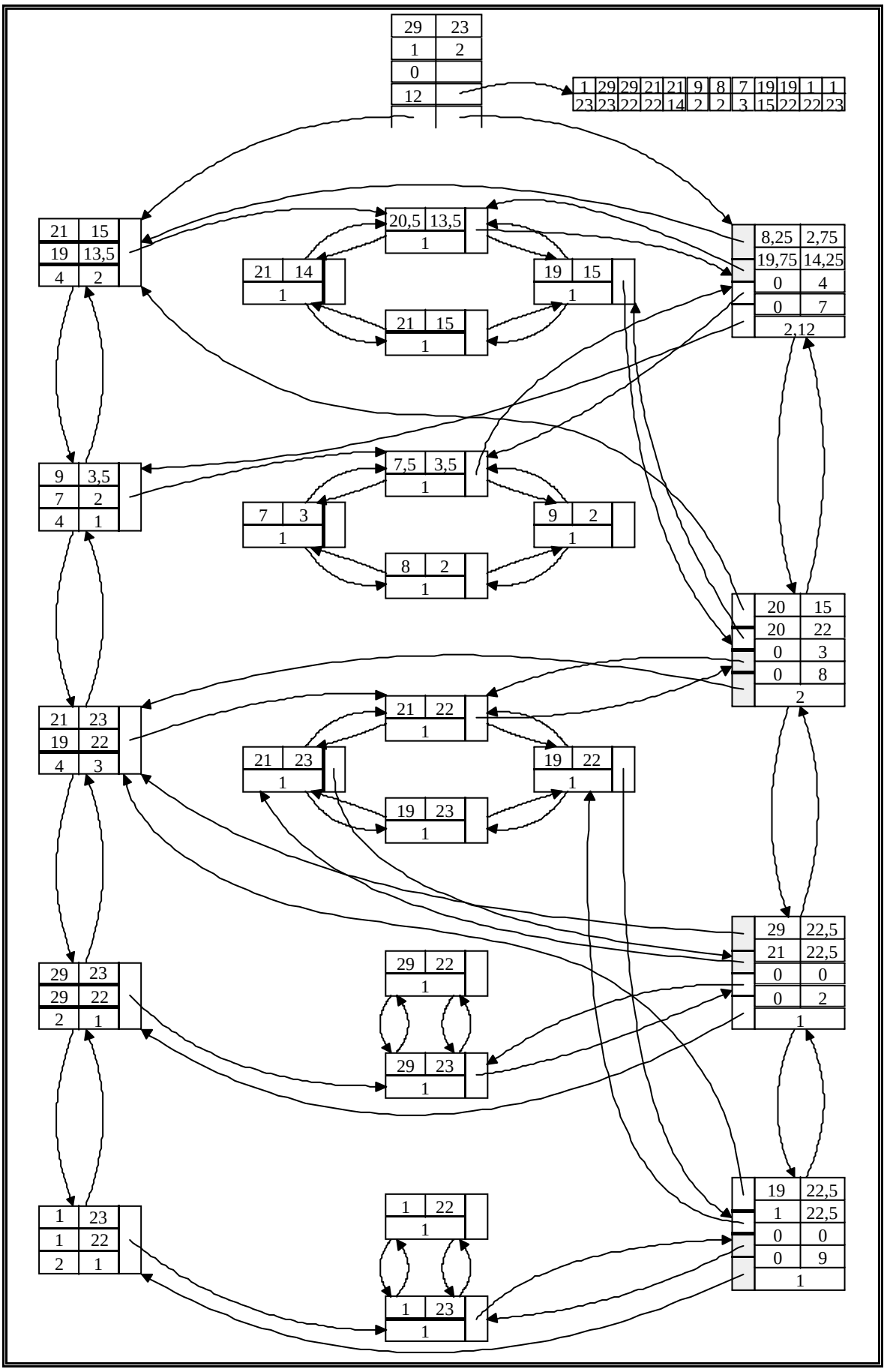

29 23
1 2
0
12
1 29 29 21 21 9 8 7 19 19 1 1
23 23 22 22 14 2 2 3 15 22 22 23
21 15
19 13,5
4 2
20,5 13,5
1
21 14
1
19 15
1
21 15
1
8,25 2,75
19,75 14,25
0 4
0 7
2,12
9 3,5
7 2
4 1
7,5 3,5
1
7 3
1
9 2
1
8 2
1
20 15
20 22
0 3
0 8
2
21 23
19 22
4 3
21 22
1
21 23
1
19 22
1
19 23
1
29 22,5
21 22,5
0 0
0 2
1
29 23
29 22
2 1
29 22
1
29 23
1
19 22,5
1 22,5
0 0
0 9
1
1 23
1 22
2 1
1 22
1
1 23
1

Figura 43

La Figura 43 muestra la estructura de contornos emparejados que se obtiene al procesar la figura del ejemplo.

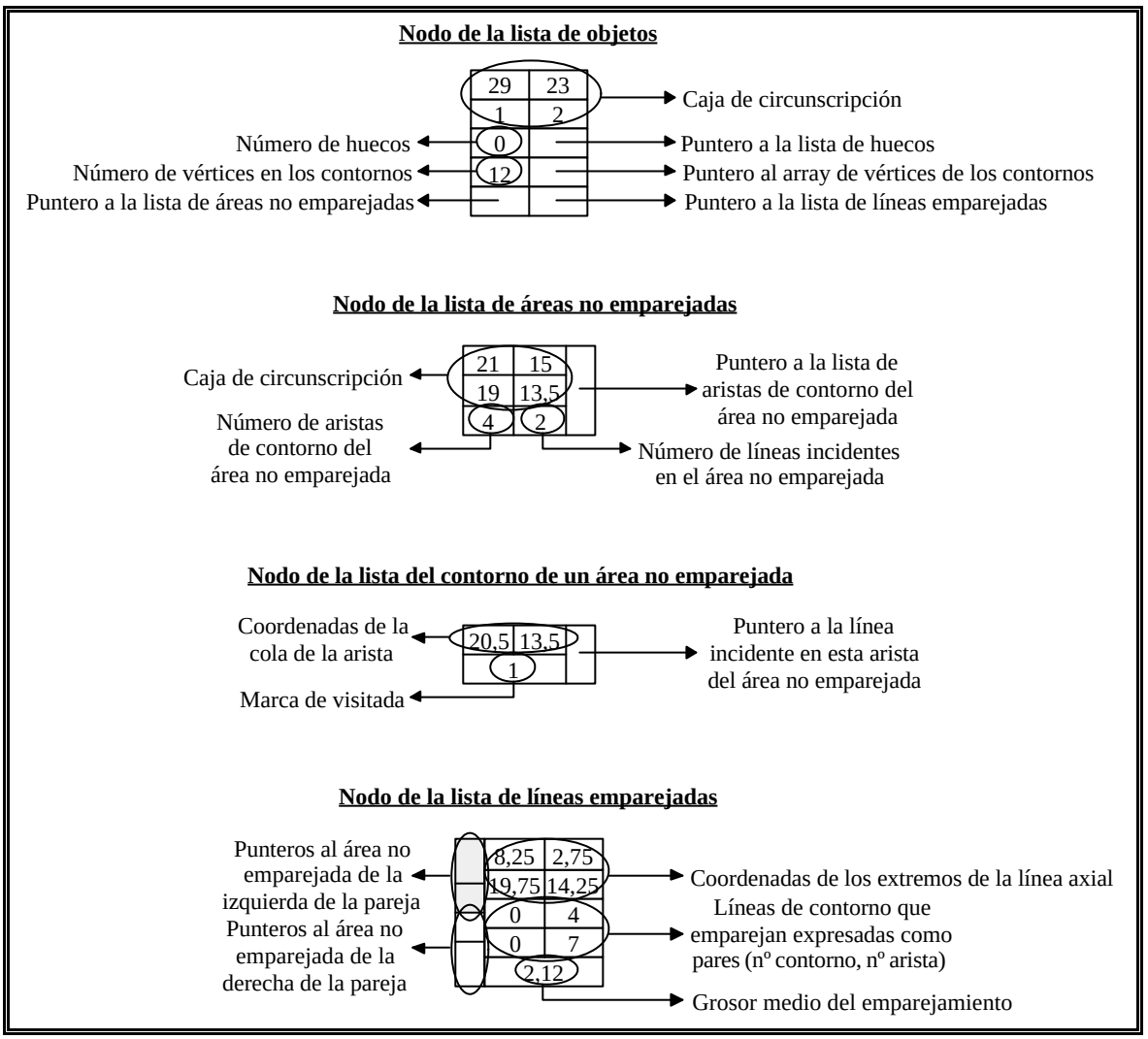


Figura 44

La Figura 44 explica el significado de cada uno de los campos de los nodos de la Figura 43.

6.8 Resolución de desemparejamientos.

La resolución de desemparejamientos es un tema complicado y en nuestra opinión lo más adecuado sería abordarlo durante un proceso de identificación de primitivas de dibujo tales como rectas, circunferencias o arcos de circunferencia. En una primera aproximación al problema se podría seguir la misma estrategia de clasificación mediante sucesivos tests aplicada por Vicente Escuderos en [12] aunque adaptada a nuestras necesidades. La técnica propuesta consistiría en recorrer el grafo representado por la estructura de contornos emparejados a partir de un área no emparejada con más

de dos líneas incidentes mediante un recorrido primero en profundidad (DFS) de cada uno de los caminos que parten de él. En el momento en que en uno de estos caminos aparezca otra área no emparejada que no tenga dos líneas incidentes deberemos realizar un test para comprobar si podemos unir esta última área con la anterior donde se inició el camino mediante una recta. De no ser así, se trataría de aproximar mediante un arco de circunferencia todas las líneas del camino. En caso de volver a fracasar deberíamos elegir el área desemparejada que más alejada esté de la línea con la que hemos realizado el primer test para dividir el camino en dos y repetir sucesivamente el proceso en cada uno de los dos caminos resultantes hasta que no sea posible partir el camino o se tenga éxito en algún test. Cuando esto último suceda se sustituirán en el grafo todas las líneas y áreas desemparejadas detectadas como parte de una primitiva por ella. A la primitiva se le asignará un grosor igual a la media de todas las líneas que toman parte en ella.

Como podemos ver el proceso anterior no resulta de ningún modo trivial y por razones de tiempo se ha decidido dejarlo pendiente. A título experimental se ha implementado código para simplificar las áreas no emparejadas en las que únicamente inciden dos líneas. Estos desemparejamientos serán uniones entre líneas o esquinas. El proceso implementado no tiene demasiada utilidad práctica en sí, aunque reduce el número de aristas implicadas en los desemparejamientos con dos líneas incidentes. Su utilidad es el estudio de este tipo de desemparejamientos para intentar aproximarlos de una manera sencilla mediante un punto, de modo que las líneas incidentes se desvíen lo menos posible con respecto a su posición inicial. Parte del código implementado como puede ser el cálculo del ángulo entre dos líneas o el cálculo de la intersección de estas podrá ser de utilidad para la implementación del algoritmo anteriormente descrito.

El código implementado recorre la lista de áreas desemparejadas en busca de las que solo tienen dos aristas incidentes. Cuando encuentra una de ellas, calcula el ángulo de las líneas que intervienen y si supera un umbral (fijado en este caso en 30º) se calculará el punto de intersección entre las dos líneas. En caso de tener un ángulo por debajo del umbral el punto por el que se aproximará el desemparejamiento será la media de los dos extremos de las líneas que inciden en el desemparejamiento.

Los resultados obtenidos con este método son bastante buenos en general pero aparecen problemas en algunas pequeñas curvas que dejan un desemparejamiento entre dos líneas que forman un ángulo por encima del umbral pero que al calcular su intersección esta se aleja del desemparejamiento. Esta situación se produce en general

en líneas que forman parte de caracteres por lo que en principio no nos preocupa ya que estos se discriminarán previamente.

6.9 Costes.

El cálculo del coste asintótico del algoritmo de emparejamiento resultaría muy delicado y laborioso de realizar de manera analítica por lo que resulta más práctico realizar una aproximación de manera intuitiva.

Estudiaremos el coste de aplicar el algoritmo de emparejamiento sobre una única figura puesto que para un dibujo completo bastará con multiplicar por el número de figuras que aparecen en él. En el peor de los casos ninguna de las aristas de la figura tendrá pareja por lo que para calcular la lista de candidatas de cada una de ellas tendremos que recorrer todas las aristas de todos los contornos de la figura pero esa arista la marcaremos ya como procesada y no volverá a ser comprobada. Por tanto si consideramos n como el número de aristas que componen los contornos de la figura, podemos decir que el coste en el peor de los casos para calcular las parejas de una figura será O(n log n). En el mejor de los casos todas las aristas de la figura emparejarían con la primera y por tanto el coste sería Ω(n). Pero en la realidad tanto el mejor como el peor caso son muy improbables y el orden en que se procesan las aristas influirá decisivamente en los tiempos reales obtenidos.

El coste de la primera fase de la generación de desemparejamientos que se realiza simultáneamente a la detección de parejas, no empeorará el coste asintótico del proceso puesto que se limita a realizar un número determinado de operaciones sobre cada una de las parejas detectadas. En cuanto a la segunda fase, cada desemparejamiento será recorrido por lo menos una vez para buscar falsas parejas y se deberá realizar un nuevo recorrido de sus aristas por cada falsa pareja que aparezca. Así pues, el coste de esta segunda fase para cada desemparejamiento será O(m n) donde n será el número de aristas que describen el contorno del desemparejamiento y m será el número de falsas parejas que aparecen en el desemparejamiento. Los emparejamientos donde aparecen falsas parejas no se presentan con demasiada frecuencia, y mucho más extraño es que aparezca más de una falsa pareja en un desemparejamiento. Teniendo esto en cuenta el valor de m será 1 en la mayoría de los casos y en un pequeño número de ocasiones será 2, siendo mayor de dos en muy extrañas circunstancias. Así pues, el coste de la operación será lineal con el número de aristas del contorno de los desemparejamientos en la mayoría de las ocasiones.

6.10 Mejoras.

Las posibles mejoras del algoritmo de emparejamiento pueden consistir en nuevos criterios que permitan descartar las aristas candidatas a pareja con mayor celeridad o detener la búsqueda de nuevas parejas en el momento que se cumplan determinadas condiciones sin llegar a completar el proceso de búsqueda exhaustiva que se realiza en la actual implementación.

Siguiendo esta línea, se podría detener el proceso de búsqueda de nuevas parejas en el momento en que la arista de la que estamos buscando parejas esté totalmente solapada razón por la que no deberían de aparecer nuevas parejas entre las aristas que restan por comprobar. Esto sería posible siempre que estemos seguros de que entre las aristas candidatas a pareja no se nos ha colado una falsa pareja. Por tanto para poder estar seguros de esto podíamos introducir criterios nuevos como comparar el grosor de la pareja que genera una arista candidata con el de otras parejas de la misma arista ya calculadas o incluso con grosores estándar o grosores habituales en la figura detectados a priori por otros mecanismos. En caso de ser viable la detención prematura de la búsqueda sería muy interesante el anticipar la comprobación de las aristas próximas a una pareja ya detectada, puesto que existe una cierta localidad espacial entre las aristas que son parejas de una dada. Con el término “aristas próximas” queremos indicar aquellas que son inmediatamente anteriores o posteriores a una dada dentro de un contorno. Si tuviéramos un caso en que las aristas a, b y c por una parte y d y e por otra fueran consecutivas y existieran los emparejamientos b-d y b-e, lo más lógico haciendo uso del mencionado principio de localidad espacial será comenzar a buscar las parejas de d y e entre las que se encuentran próximas a b, es decir, comenzar a buscar empezando por a y c y continuar en la dirección de aquella de las dos que resultara pareja si ese fuera el caso.

Todas estas mejoras deberían ser objeto de un minucioso estudio antes de pasar a su implementación puesto que pueden originar una sobrecarga de código que haga que los tiempos no mejoren o incluso que empeoren además de existir la posibilidad de que aparezcan problemas inesperados debidos a configuraciones particulares de las aristas de los contornos o a otras causas que no se han tenido en cuenta. Según lo visto habrá que ser muy cauto a la hora de sopesar el esfuerzo, las desventajas y los beneficios que podemos obtener con estas modificaciones.

7. Costes temporales: aproximación empírica.

El objetivo de la aproximación empírica es la verificación de los resultados teóricos de los costes de cada una de las fases del algoritmo indicados en su momento. Para ello se ha elegido un conjunto significativo de planos de ingeniería que permiten comprobar el funcionamiento de la aplicación.

Los planos se han elegido teniendo en cuenta que fueran todos en el formato TIFF sin comprimir para así evitar distorsiones en los tiempos debidas a los algoritmos de lectura de los diferentes formatos de ficheros y los de descompresión. Todos los planos son dibujos escaneados a una resolución de 300 puntos por pulgada a los que no se ha sometido a ningún tipo de preproceso. A los planos tampoco se les ha eliminado el texto que en ellos pudiera aparecer. Las calidades de los diferentes rásters obtenidos tras el escaneado son diversas pero podemos calificarlas en todos los casos como aceptables.

No se ha tenido en cuenta aspectos como el diferente grosor de las líneas ni el tipo de dibujo que aparece en el ráster, así que podemos encontrar todo tipo de características en el conjunto de dibujos de prueba. Por lo tanto, desde este punto de vista el conjunto de prueba puede considerarse bastante heterogéneo ya que aparecen desde esquemas eléctricos hasta planos de maquetas pasando por planos de arquitectura piezas mecánicas y simples dibujos de una vista de alambre.

Por su parte el programa también ha sido modificado para eliminar el contador de líneas de la barra de estado que provoca un innecesario incremento en los tiempos que en algunos casos resulta significativo. Este contador de líneas únicamente proporciona feedback durante el procesamiento de los planos pero debería de buscarse un mejor medio de implementar este feedback que frenase menos el funcionamiento del algoritmo.

Los tiempos que devuelve el programa al terminar el procesamiento de la imagen sólo miden el tiempo que transcurre entre el comienzo de la rutina de vectorización y su final y no incluyen los tiempos de dibujado de los vectores y de liberación de memoria al destruir la estructura de contornos emparejados ya que estas tareas dependen básicamente del sistema operativo.

Para poder obtener los tiempos de cada una de las fases se han recompilado versiones especiales del programa en las que se han ido eliminando los diferentes

módulos convirtiendo en comentarios las líneas en las que se realizan las llamadas a ellos. Dada la estructura secuencial en que se aplican los diferentes módulos sobre las figuras que se van encontrando en el ráster, podemos obtener los tiempos consumidos por cada fase (módulo) de la vectorización como la diferencia de los tiempos obtenidos entre la versión del programa que incluye hasta un módulo en particular y la versión que sólo ejecuta hasta la fase que le precede en la secuencia.

La Tabla 8 incluye un resumen de las medias de los tiempos obtenidas para cada una de las fases. En esta tabla las diferentes fases aparecen ordenadas de izquierda a derecha según la secuencia de ejecución, por tanto los tiempos correspondientes a la ejecución del proceso completo aparecerán en la columna de más a la derecha. También aparece intercalada una columna extra correspondiente a la media conjunta de los dos filtrados, debido a que aparecen problemas a la hora de medir con precisión los tiempos de cada uno de los filtros por separado. Tal y como se comentará más adelante, se ha optado por agrupar conjuntamente ambas operaciones.

Plano	Media de la aproximación de contornos	Media del filtrado de ruido	Media del filtrado de colinearidades	Media conjunta de los dos filtrados	Media de la generación de parejas
1	0,108	0,002	0,01	0,012	0,926
2	0,606	0,052	0,01	0,062	12,13
3	0,474	0,02	0,01	0,03	71,818
4	4,188	0,14	0,144	0,284	604,082
5	2,196	0,11	0,054	0,164	231,566
6	7,162	0,078	0,06	0,138	190,51
7	8,988	0,084	0,254	0,338	405,954
8	0,386	0,002	0,018	0,02	47,092
9	0,530	0,024	0,018	0,042	2,67
10	0,438	0,012	0,014	0,026	25,936

Tabla 8

Los tiempos que aparecen en la tabla han sido tomados en un PC con un Pentium II a 300 Mhz. con 64 Mb. de RAM corriendo bajo Windows 98. Los tiempos que se obtienen dependen en gran medida del sistema operativo y del tamaño de la RAM debido al empleo de técnicas de reserva de memoria dinámica y al uso masivo que se hace de esta a causa del gran tamaño que alcanzan las estructuras de datos.

Para que nos hagamos una idea de cómo se han obtenido los datos de la tabla pondremos como ejemplo la columna correspondiente al filtro de ruido. El filtro de ruido se ejecuta después de obtener la aproximación poligonal de los contornos de una

figura y antes de que se ejecute el filtro de colinearidades, por tanto, podemos obtener el tiempo que emplea esta operación como la diferencia entre la versión del programa que ejecuta solamente la aproximación poligonal y la versión que ejecuta el filtrado de ruido pero no el filtrado de contornos ni las fases siguientes.

7.1 Generación de contornos

En la Tabla 9 podemos ver algunos datos acerca del tamaño de los planos utilizados. El tamaño real de los documentos puede obtenerse fácilmente si tenemos en cuenta que todos los planos han sido escaneados a una resolución de 300 p.p.p. y tenemos el tamaño en pixeles de la imagen.

Plano	Tamaño de la imagen (pixeles)		Tamaño de la imagen en bits	Tamaño del fichero en Kb.	Cambios entre fondo y dibujo
	Ancho	Alto			
1	1013	1517	1536721	189	34818
2	2490	2871	7148790	875	198476
3	2806	2174	6100244	746	138340
4	5776	12583	72679408	8883	810872
5	10400	3744	38937600	4757	387654
6	10784	13959	150533856	18387	1041826
7	10784	16362	176447808	21556	1416668
8	1876	2959	5551084	680	89647
9	2536	4166	10564976	1291	75640
10	3535	2318	8194130	1002	79630

Tabla 9

La columna de más a la derecha de la tabla muestra el número de cambios entre el color de fondo y el de dibujo que se producen al recorrer el ráster de izquierda a derecha y de arriba abajo. Esta última columna resulta de especial importancia en este caso puesto que el algoritmo modificado de aproximación poligonal de contornos (AMAPC) tiene un coste teórico directamente proporcional al número de cambios producidos (O(n)).

El número de cambios será el doble del número de segmentos encontrados en una codificación run-lenght en horizontal de la imagen.

La Tabla 10 muestra los resultados obtenidos tras ejecutar en cinco ocasiones la versión del programa que únicamente aplica el AMAPC. En la columna de más a la derecha aparecen los tiempos medios calculados con los datos obtenidos de las columnas anteriores.

Plano	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Media
1	0,1	0,11	0,11	0,11	0,11	0,108
2	0,61	0,61	0,6	0,61	0,6	0,606
3	0,49	0,44	0,44	0,5	0,5	0,474
4	4,18	4,23	4,17	4,18	4,18	4,188
5	2,2	2,19	2,19	2,2	2,2	2,196
6	7,2	7,19	7,14	7,14	7,14	7,162
7	8,96	9,01	8,95	9,06	8,96	8,988
8	0,39	0,39	0,39	0,38	0,38	0,386
9	0,5	0,55	0,55	0,55	0,5	0,530
10	0,43	0,44	0,44	0,44	0,44	0,438

Tabla 10

El Gráfico 1 muestra el resultado de los tiempos medios obtenidos de la ejecución del AMAPC frente al número de cambios entre el fondo y la figura de cada uno de los planos. Como puede apreciarse en la figura el coste temporal de este algoritmo es claramente lineal con respecto al número de cambios.

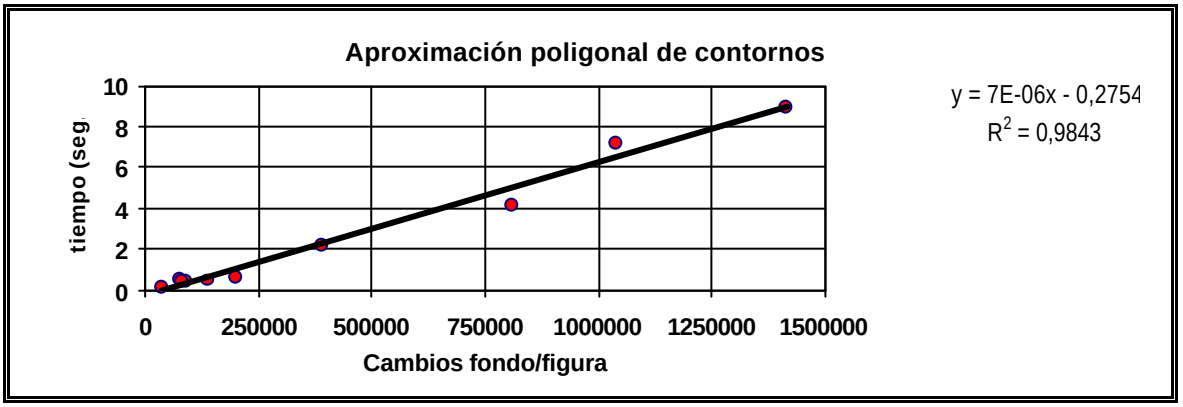


Gráfico 1

7.2 Filtrado

Como ya sabemos, después de la generación de los contornos someteremos a los polígonos obtenidos de esta fase a un proceso de filtrado en el que se aplicarán dos algoritmos distintos. El primero de los filtros en aplicarse será el de eliminación de ruido de los contornos, y después de él, se aplicará el filtro de eliminación de colinearidades. Ambos filtros tienen un coste teórico lineal.

En la Tabla 11 podemos ver el resultado de la fase de generación de contornos y en las dos últimas columnas, el número de aristas que se obtienen tras la aplicación de los filtros. Como se puede apreciar, la reducción del número de aristas es significativa en todos los casos y todavía resulta más importante al tener en cuenta que la mayor

parte del coste del proceso viene determinada por el algoritmo de emparejado que como veremos depende del número de aristas.

Plano	Contornos externos	Contornos interiores	Polígonos	Aristas en los polígonos	Aristas después del filtro de ruido	Aristas después del filtro de colinearidades
1	310	220	530	12446	9621	7722
2	2266	1844	4110	72621	67532	49401
3	526	2372	2898	53393	48592	36561
4	7938	3407	11345	310403	267430	204153
5	3719	2115	5834	169461	140966	113053
6	3122	2166	5288	176139	120819	92260
7	5409	5062	10471	356277	294151	226241
8	245	1579	1824	35489	28101	21636
9	904	442	1346	25377	16960	13568
10	671	569	1240	19698	14846	12307

Tabla 11

El tiempo medio de ejecución de la fase de filtrado se puede obtener como la diferencia de los tiempos medios de las dos columnas de la izquierda de la Tabla 12.

Plano	Media tras el filtrado	Media del AMAPC	Media del filtrado
1	0,120	0,108	0,012
2	0,668	0,606	0,062
3	0,504	0,474	0,03
4	4,472	4,188	0,284
5	2,360	2,196	0,164
6	7,300	7,162	0,138
7	9,326	8,988	0,338
8	0,406	0,386	0,02
9	0,572	0,530	0,042
10	0,464	0,438	0,026

Tabla 12

Dado que el reloj que se ha utilizado para medir los tiempos solo tiene una resolución de centésimas de segundo y que el tiempo de ejecución de las operaciones de filtrado es muy pequeño en la mayoría de los dibujos de ejemplo, los tiempos medios obtenidos para cada uno de los dos filtros por separado no resultan demasiado fiables.

Una posible solución al problema podía ser repetir varias veces el proceso de filtrado sobre la misma imagen, pero teniendo en cuenta que durante el proceso se

modifica la imagen, no es posible repetir el proceso sobre el mismo conjunto de datos sin tener que realizar grandes cambios en el programa.

La solución por la que se ha optado consiste en agrupar los costes de ejecución de ambas operaciones en uno solo, aprovechando que ambas tienen un coste lineal y se ejecutan secuencialmente. Al estar las operaciones agrupadas los tiempos medios de ejecución tienen un valor mayor y por tanto se ven afectados en menor medida por los errores derivados de la falta de resolución del reloj. Además, sólo se pretende comprobar la linearidad de ambas operaciones y este objetivo se puede cumplir también mediante la solución que se ha adoptado.

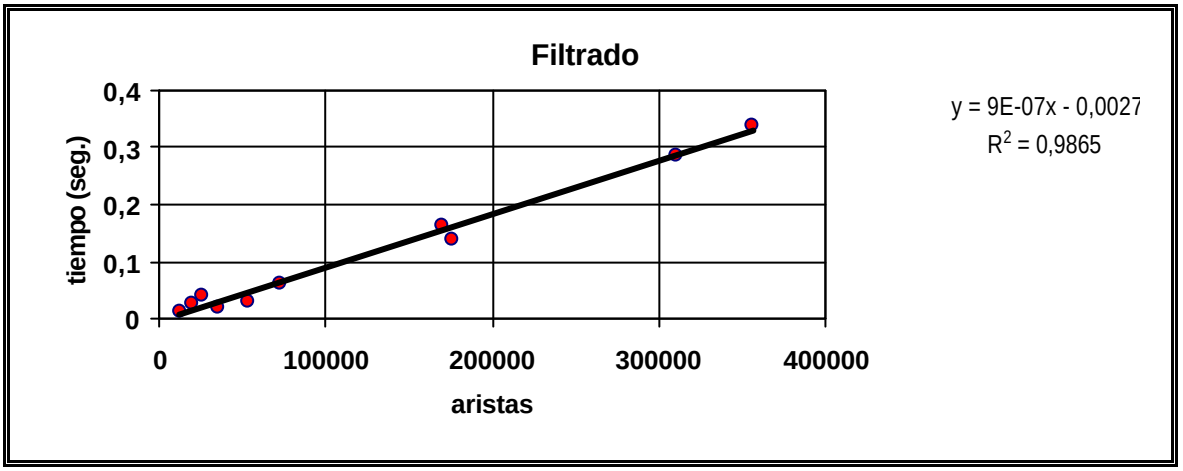


Gráfico 2

En el Gráfico 2 se puede apreciar claramente el carácter lineal de la fase de filtrado ya que se obtiene un coeficiente de correlación bastante bueno. Las tablas 6 y 7 y los gráficos 3 y 4 muestran la información correspondiente a cada uno de los filtros por separado y como se puede apreciar por los coeficientes de correlación el ajuste lineal no es tan bueno como con los filtros agrupados.

Plano	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Media
1	0,11	0,11	0,11	0,11	0,11	0,110
2	0,66	0,66	0,66	0,65	0,66	0,658
3	0,5	0,49	0,49	0,49	0,5	0,494
4	4,34	4,34	4,34	4,28	4,34	4,328
5	2,3	2,31	2,31	2,31	2,3	2,306
6	7,2	7,25	7,25	7,25	7,25	7,240
7	9,12	9,06	9,06	9,06	9,06	9,072
8	0,39	0,38	0,39	0,39	0,39	0,388
9	0,55	0,54	0,54	0,54	0,6	0,554
10	0,44	0,44	0,44	0,44	0,49	0,450

Tabla 13

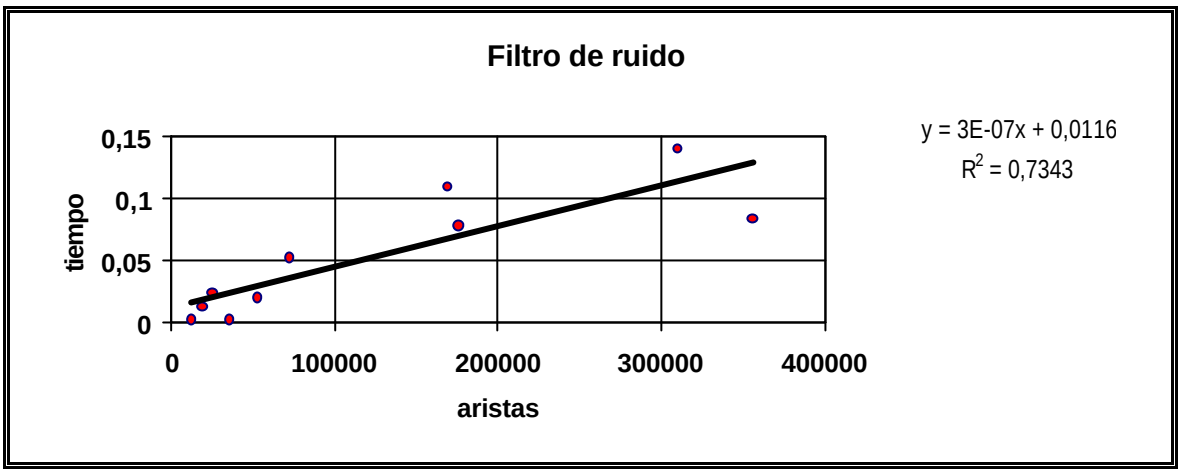


Gráfico 3

Plano	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Media
1	0,11	0,11	0,16	0,11	0,11	0,120
2	0,66	0,65	0,66	0,71	0,66	0,668
3	0,5	0,49	0,54	0,5	0,49	0,504
4	4,45	4,45	4,51	4,5	4,45	4,472
5	2,36	2,36	2,36	2,36	2,36	2,360
6	7,3	7,3	7,3	7,3	7,3	7,300
7	9,83	9,17	9,17	9,17	9,29	9,326
8	0,39	0,38	0,38	0,44	0,44	0,406
9	0,55	0,61	0,54	0,61	0,55	0,572
10	0,44	0,5	0,44	0,44	0,5	0,464

Tabla 14

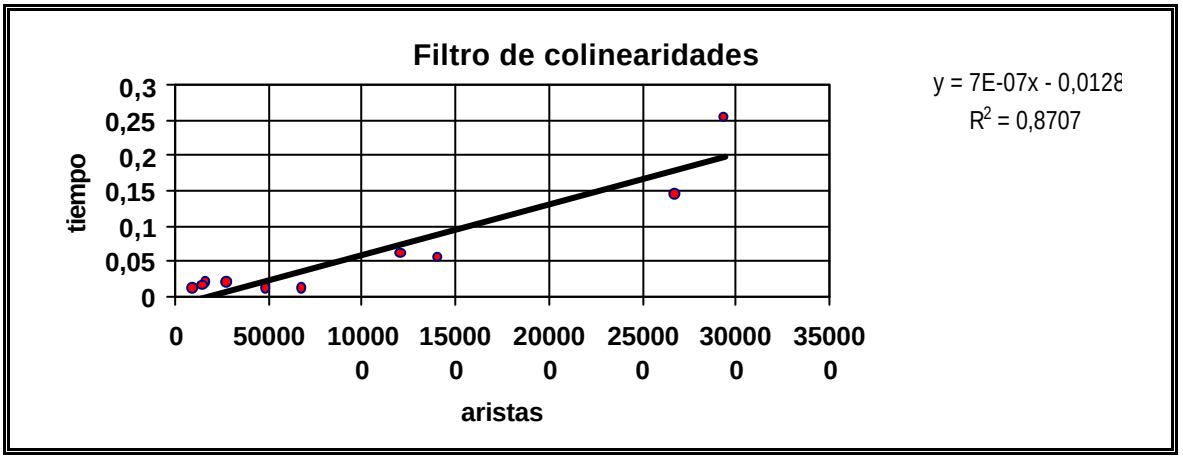


Gráfico 4

7.3 Emparejamiento.

La Tabla 15 muestra algunos datos obtenidos tras emparejar las aristas de los polígonos de contorno que se han obtenido tras la fase de filtrado. Las áreas no emparejadas han sido clasificadas en extremos, uniones y cruces dependiendo de que el

número de parejas que confluyen en ellas sea una, dos o más de dos respectivamente. Asimismo, se incluyen en la tabla el número de aristas de los polígonos que forman las áreas desemparejadas.

Plano	Parejas	Extremos	Uniones	Cruces	Áreas no emparejadas	Lados de las áreas no emparejadas
1	3480	594	2578	398	3570	13245
2	24599	4601	16871	3583	25055	89010
3	17171	1151	10822	3432	15405	65074
4	87540	14529	72257	5377	92163	355974
5	42041	7873	31413	4365	43651	187396
6	40756	6190	30504	4842	41536	163041
7	100129	13008	74548	12652	100208	402225
8	8349	433	4896	1910	7239	36367
9	5713	781	4077	1281	6139	23013
10	4934	574	3122	1312	5008	20480

Tabla 15

La Tabla 16 muestra los tiempos que se han obtenido al ejecutar la versión del programa que realiza los emparejamientos.

Plano	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Media
1	1,04	1,05	1,05	1,05	1,04	1,046
2	12,8	12,74	12,85	12,8	12,8	12,798
3	72,28	72,44	72,28	72,5	72,11	72,322
4	607,53	608,3	610,39	610,28	606,27	608,554
5	233,33	233,21	234,47	233,87	234,75	233,926
6	198,5	198,44	196,91	197,63	197,57	197,810
7	419,69	422,92	414,25	420,79	398,75	415,280
8	47,24	47,62	47,67	47,4	47,56	47,498
9	3,24	3,24	3,25	3,24	3,24	3,242
10	26,42	26,42	26,42	26,37	26,37	26,400

Tabla 16

El Gráfico 5 muestra la nube de puntos que se obtiene al comparar el número de aristas procedentes de la etapa de filtrado que se pretende emparejar con los tiempos medios obtenidos al restar a la media de la Tabla 16 la media de las fases anteriores al emparejamiento que aparece en la Tabla 15.

El ajuste que mejor se adapta a esta nube de puntos es el ajuste lineal con un coeficiente de correlación de 0,8937. Estos datos parecen indicar que aunque en el peor de los casos el coste de la operación de emparejamiento es de O(n log n) con respecto al

número de aristas a emparejar, en los casos más habituales los costes temporales se explican mejor linealmente.

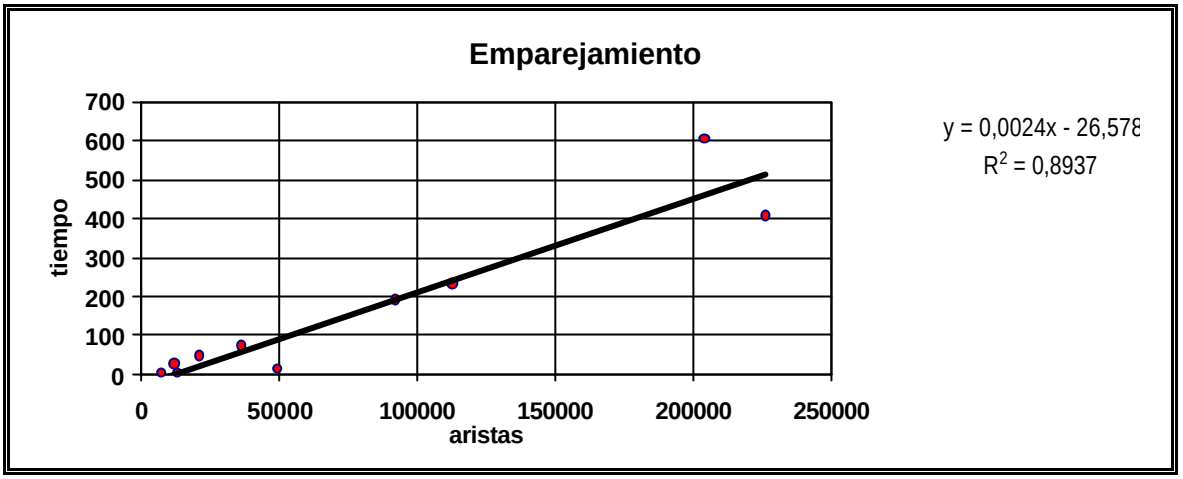


Gráfico 5

8. Conclusiones.

En el presente trabajo se ha realizado una aproximación al problema de la vectorización en general, y en particular a su aplicación a los dibujos de ingeniería.

La vectorización automática no sólo es un modo más eficiente que la vectorización manual de introducir en un ordenador la información de los numerosos documentos ya existentes en papel, sino que además al combinarse con las técnicas de reconstrucción 3D abre nuevos caminos para mejorar los flujos de trabajo que se establecen entre el hombre y los programas de CAD.

También se ha intentado obtener una visión comparativa de las herramientas de vectorización que se encontraban a nuestro alcance y se ha comprobado que las herramientas de conversión de ráster a vector suelen estar orientadas para obtener mejores resultados en un campo de aplicación en particular.

Para profundizar en el conocimiento de los sistemas de vectorización se ha realizado una amplia búsqueda bibliográfica centrada en los aspectos más básicos del proceso de conversión de ráster a vectores. Se han estudiado diferentes sistemas propuestos por diversos autores comparando las soluciones que se adoptan en temas tan diversos como la extracción de líneas, la discriminación de texto, la detección de líneas discontinuas, sombreados, cabezas de flechas y etcétera y como resultado se ha propuesto un sistema propio.

El sistema propuesto se centra en un algoritmo de extracción de líneas basado en el emparejamiento de contornos. Se ha juzgado que la técnica propuesta presenta diversas ventajas sobre la más clásica y extendida del adelgazamiento de contornos (thinning) porque parece afrontar de un modo más razonable la detección de áreas problemáticas como los cruces y las áreas rellenas desechando la menor información posible y además, con el método propuesto la información resultante del proceso se obtiene en una forma adecuada para su posterior procesamiento en fases más avanzadas con un menor esfuerzo.

Finalmente se ha realizado una implementación de la serie de algoritmos que permiten la extracción y emparejamiento de los contornos de imágenes binarias y se ha probado sobre dibujos que reúnen características que aparecen en la realidad como un tamaño considerable, ruido, etc. Los resultados obtenidos han sido bastante

esperanzadores en cuanto al funcionamiento y la calidad del resultado obtenido. Por todas estas razones parece que se está ante un punto de comienzo apropiado para el desarrollo de un sistema más completo y complejo.

Todavía quedan muchas fases por desarrollar y son posibles diversas mejoras en las ya desarrolladas, pero también a este respecto se ha intentado realizar una humilde aportación de ideas. En el campo de las mejoras parece que la calidad del proceso podía mejorarse más utilizando técnicas de filtrado de los contornos vectorizados más potentes. Si se filtrase el ruido y las colinearidades con mayor precisión y se consiguiera eliminar un mayor número de aristas de contorno, aunque el coste temporal real de estas operaciones fuera superior el tiempo de procesamiento global del algoritmo sería menor o por lo menos la calidad de los resultados sería superior.

Apéndices.

Apéndice A: Manual del usuario.

Apéndice B: Ejemplos.

Apéndice A: Manual del usuario.

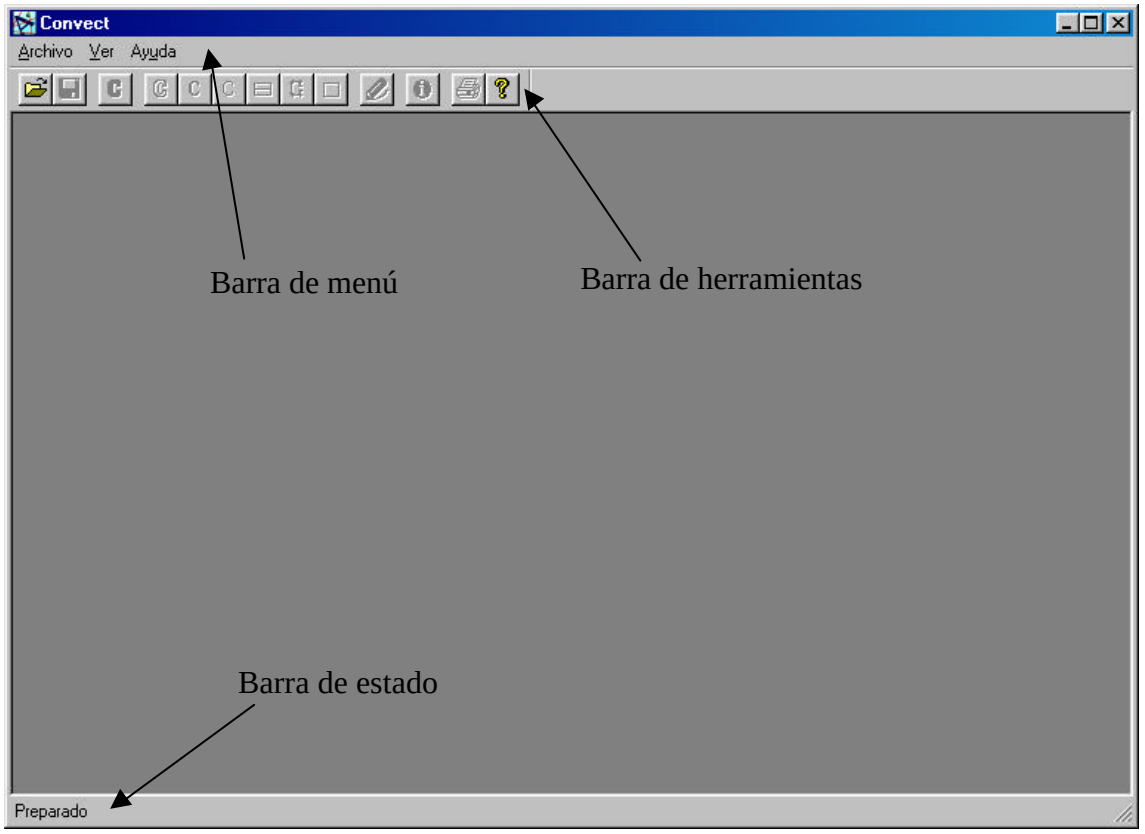
Instalación.

La aplicación consta únicamente de dos ficheros: Convect.exe y Lead51n.dll. Para que el programa funcione correctamente ambos ficheros deberán encontrarse en el mismo directorio. En Windows 95/98 como alternativa se puede situar el fichero Lead51n.dll en el directorio C:\windows\system.

Ejecución.

Para lanzar el programa bastará con hacer doble click en el icono del programa:

Al ejecutar el programa aparecerá la siguiente ventana:

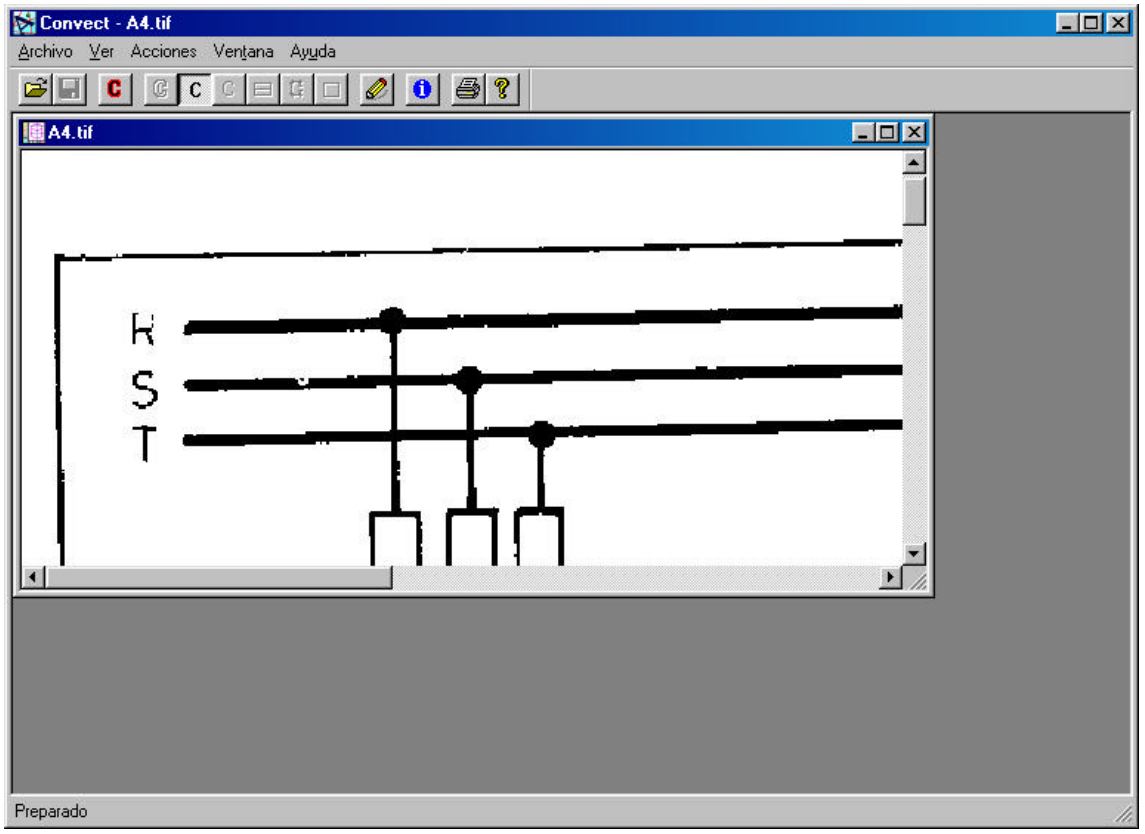


Para abrir un fichero bastará con pulsar en Archivo/Abrir o sobre el icono  . Los tipos de ficheros que se pueden abrir son: LEAD, JFIF, LEAD1JFIF, LEAD2JFIF, JTIF, LEAD1JTIF, LEAD2JTIF, TIFF, MPT, TIFF LZW, TIFF CCITT, TIFF CCITT Group 3, TIFF CCITT Group 4, IOCA (ICA), WinFax Group 3, WinFax Group 4, FAX Group 3, FAX Group 4, Truevision TGA (TARGA), GIF, PNG, Photoshop 3.0 (PSD), Windows Bitmap (BMP), Windows Metafile (WMF), PCX, DCX, PostScript Raster

(Encapsulated PostScript), OS/2 Bitmap (OS/2 BMP), CALS Raster, MacPaint (MAC), GEM Image (IMG), Microsoft Paint (MSP), WordPerfect (WPG), SUN Raster (RAS), Macintosh Pict (PCT), LEAD 1BIT, PCD.

El programa sólo funcionará correctamente con aquellos ficheros de los anteriores tipos que contengan imágenes binarias ya que no se permite trabajar con colores.

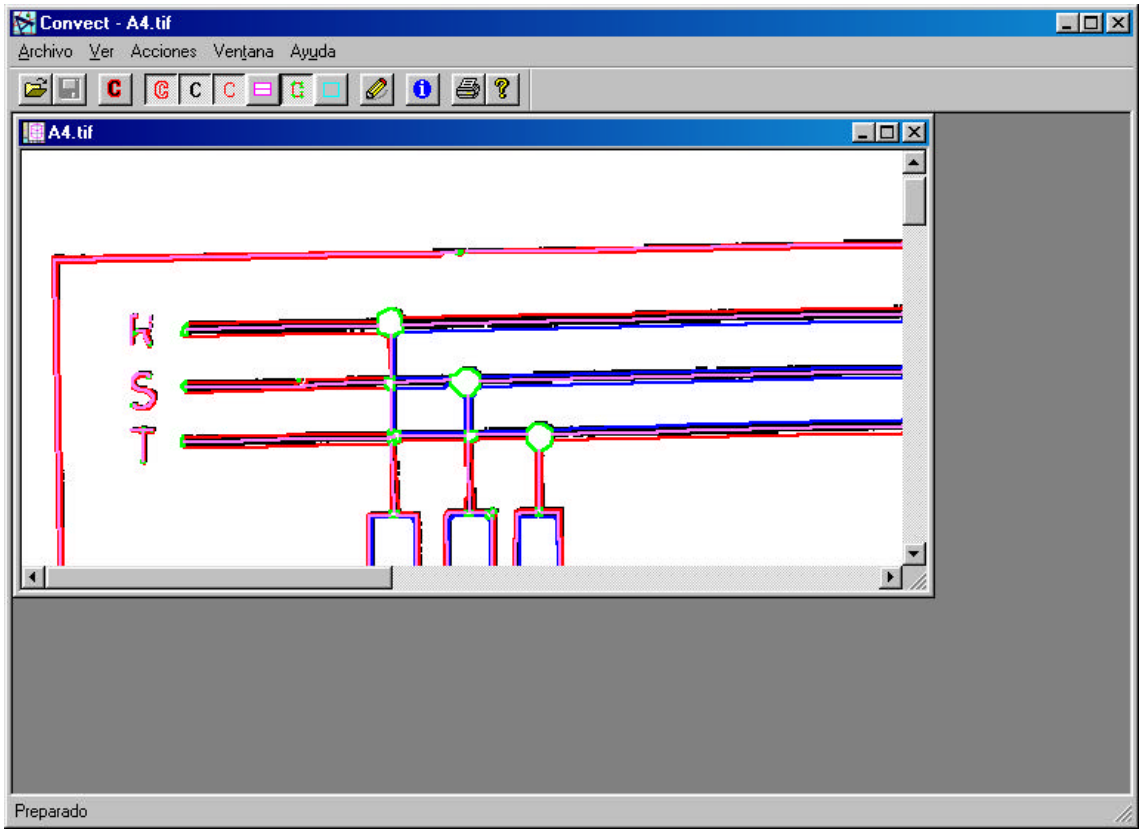
Cuando abrimos un archivo el aspecto de las barras de herramientas y de menú cambiará y aparecerá una ventana con la imagen del fichero:



Pulsando con el ratón sobre la entrada de menú Acciones\Procesar o sobre el icono  se iniciará el procesamiento de la imagen. Una vez iniciado el procesamiento, en la barra de estado se sustituirá el mensaje “Preparado” por un contador que indica la línea de la imagen que se está procesando en ese momento. En algunas imágenes el contador se detiene en una línea por un periodo prolongado de tiempo, esto no indica mal funcionamiento, sino que debido a que en esa línea de la imagen termina una figura que resulta muy costosa de procesar debido a su complejidad.

Al terminar de procesarse la imagen aparecerá un mensaje que nos informa del tiempo que ha consumido el programa en procesarla. Sobre la imagen de la ventana

podremos observar que aparecen dibujados los resultados del proceso en distintos colores y además se iluminarán el resto de los iconos de la barra de herramientas.

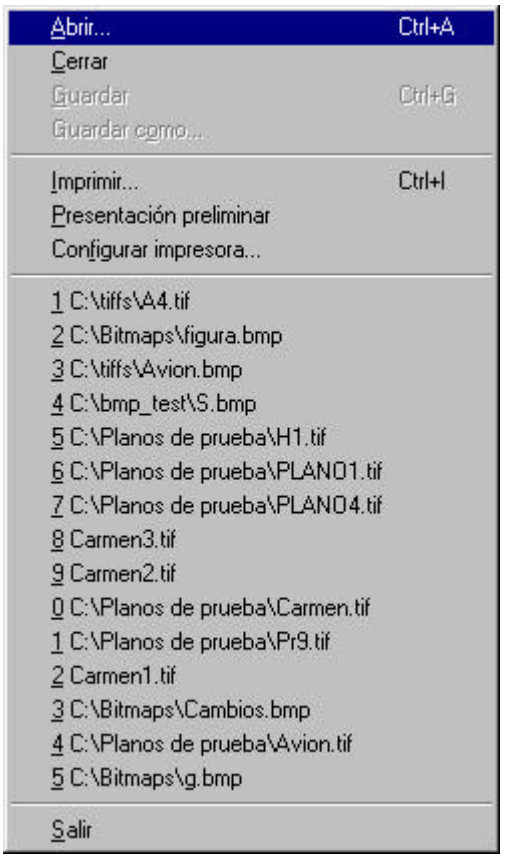


La imagen original leída del fichero aparecerá en color negro y superpuesta sobre ella tendremos líneas de color rojo y azul que representan los contornos exteriores e interiores respectivamente. Entre algunas de estas líneas aparecen líneas de color rosa que representarán que las líneas de los contornos que se encuentran a ambos lados han emparejado y en ese punto se considera que se ha detectado una línea. Las partes del dibujo donde no se detectan líneas aparecerán señaladas mediante polígonos de color verde con el interior blanco.

Menús.

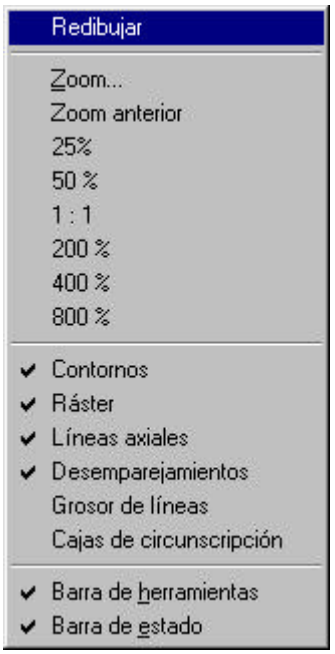
1. Archivo.

El menú archivo contiene las mismas entradas que aparecen en la mayoría de las aplicaciones de Windows. Las únicas notas a tener en cuenta son que ya que el programa no produce ningún fichero de salida para evitar estropear los ficheros ya existentes se han inhabilitado las funciones de Guardar y Guardar como. Además el mecanismo de impresión es muy rudimentario y solo permite imprimir a tamaño 1:1 aunque en pantalla la imagen se pueda visualizar ampliada o reducida.



2. Ver.

El menú Ver permite manipular el modo en como se presenta la información en pantalla.



2.1 Redibujar.

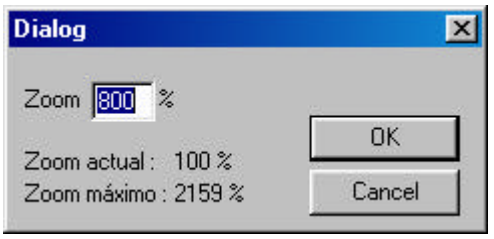
Vuelve a dibujar la imagen en pantalla. Debería usarse en momentos en los que la imagen que se presenta en pantalla está incompleta o defectuosa debido a defectos producidos durante el scroll. También se puede accionar el redibujado de la imagen en pantalla pulsando el icono  de la barra de herramientas.

2.2 Zoom, Zoom anterior, 25%, 50%, 1:1, 200%, 400%, 800%.

Todas estas entradas de menú permiten controlar el tamaño de la imagen en pantalla. Las 6 últimas amplían (200%, 400%, 800%), reducen (25%, 50%) o muestran un único pixel de la imagen por cada pixel del fichero (1:1).

El comando Zoom anterior ajusta la imagen con las mismas opciones de tamaño que se tenían en la operación de zoom inmediatamente anterior.

Al presionar sobre el comando Zoom aparece el siguiente cuadro de dialogo:



En el se nos permite introducir el porcentaje de zoom exacto que queremos que se aplique a la imagen tanto para ampliarla (valor mayor que100) como para reducirla. También aparece indicado el porcentaje de zoom que se esta aplicando en esos momentos a la imagen, así como el porcentaje máximo que se permite. Los porcentajes de zoom deberán ser mayores que cero y menores que el valor indicado en el campo Zoom máximo. Esta cantidad variará dependiendo de las dimensiones de la imagen con la que se esté trabajando en ese momento.

Existe una forma alternativa de realizar zoom de ampliación que consiste en pulsar con el botón izquierdo dentro de la imagen y arrastrarlo sin soltar el botón. Veremos que aparece un rectángulo sombreado entre el punto donde empezamos a pulsar el botón y el puntero del ratón y en el momento que soltemos el botón la parte de la imagen que se encuentra encerrada por el rectángulo sombreado se ampliará hasta ajustarse al tamaño de la ventana.

Otra posibilidad de zoom consiste simplemente en hacer click sobre la superficie de la imagen sin arrastrar el ratón. Con esta acción conseguimos

rápidamente que el área de la imagen próxima al punto donde realizó la pulsación se amplíe en un porcentaje arbitrario.

Manipulando el zoom con el ratón podemos conseguir porcentajes de zoom por encima del valor indicado como máximo en el cuadro de diálogo anteriormente comentado. Los porcentajes obtenidos jamas superarán el 3000%. Hay que advertir que el precio que debemos pagar para conseguir estos porcentajes es no poder hacer scroll por la superficie completa de la imagen mientras el valor del zoom actual sea superior al del zoom máximo.

El botón derecho del ratón se ha programado para que al hacer click dentro de la imagen se devuelva el zoom al modo 1:1.

2.3 Contornos, ráster, líneas axiales, desemparejamientos, grosor de líneas, cajas de circunscripción.

Este grupo de entradas de menú controlan que información de la que se ha obtenido durante el procesamiento se debe mostrar en pantalla. Estos comandos producen los mismos resultados que pulsar los siguientes iconos de la barra de herramientas:

 Contornos. Controla el dibujado de los polígonos que aproximan los contornos de las figuras de la imagen.

 Ráster. Controla el dibujado de la imagen contenida en el fichero original.

 Líneas axiales. Controla el dibujado de las líneas centrales que representan las parejas encontradas.

 Desemparejamientos. Controla el dibujado de los polígonos que representan el perímetro de las áreas en las que no aparecen emparejamientos.

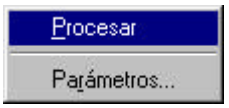
 Grosor de líneas. Controla el dibujado del grosor medio detectado entre las líneas que forman una pareja.

 Cajas de circunscripción. Controla el dibujado de las cajas que contienen en su interior los polígonos que aproximan los contornos.

Cuando uno de los botones anteriores está pulsado o su correspondiente entrada de menú está marcada como seleccionada (Õ) significa que está activado el dibujado de las correspondientes entidades y por tanto estas aparecerán en pantalla. Teniendo esto en cuenta, si dejamos todas las opciones desactivadas podemos dejar la imagen de la ventana en blanco.

2.4 Barra de herramientas, barra de estado.

Estas dos entradas de menú se comportan del mismo modo que las anteriores pero muestran u ocultan elementos de la ventana del programa como la barra de herramientas o la barra de estado.



3. Acciones.

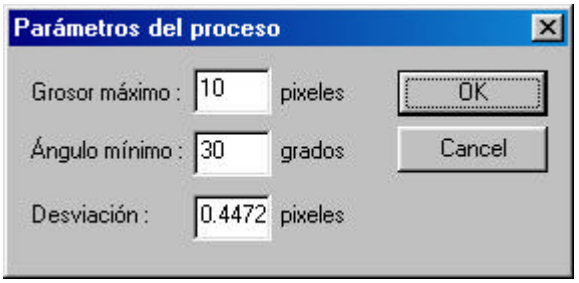
En este menú disponemos de comandos para lanzar el procesamiento y ajustar los parámetros de este.

3. 1 Procesar.

Accionar esta entrada de menú tiene el mismo efecto que el icono  de la barra de herramientas que como ya vimos anteriormente inicia el procesamiento de la imagen.

3.2 Parámetros.

Al pulsar sobre esta entrada de menú aparecerá el siguiente cuadro de dialogo en el podemos ajustar los tres parámetros que intervienen en el procesamiento de la imagen:



El primero de los parámetros, el grosor máximo, es el la distancia por encima de la cual, dos aristas que se solapan desde el punto de vista de una de ellas, no pueden considerarse una pareja.

El ángulo mínimo es el valor por debajo del cual, el ángulo que forman dos aristas que se solapan se considera aceptable para aceptarlas como pareja.

La desviación indica el valor por encima del cual, se considera que la distancia máxima entre una línea con la que pretendemos aproximar un conjunto de aristas de un contorno, es demasiado grande y por tanto no se realizará la aproximación.

4. Ventana.

La aplicación permite tener varias ventanas abiertas al mismo tiempo y por tanto trabajar sobre las imágenes de varios ficheros simultáneamente o sobre un mismo fichero pero con diferentes parámetros. Este menú se encarga de gestionar todas las ventanas que representan las diferentes imágenes con las que se está trabajando.

Las entradas del menú son totalmente estándar y por lo tanto no las comentaremos, solo destacaremos que pulsando en la entrada “Nueva ventana” se abre una nueva ventana con la misma imagen original que la ventana activa lo que nos permite introducir otro juego de parámetros distinto para procesar la imagen y poder comparar los resultados obtenidos en ambos casos.

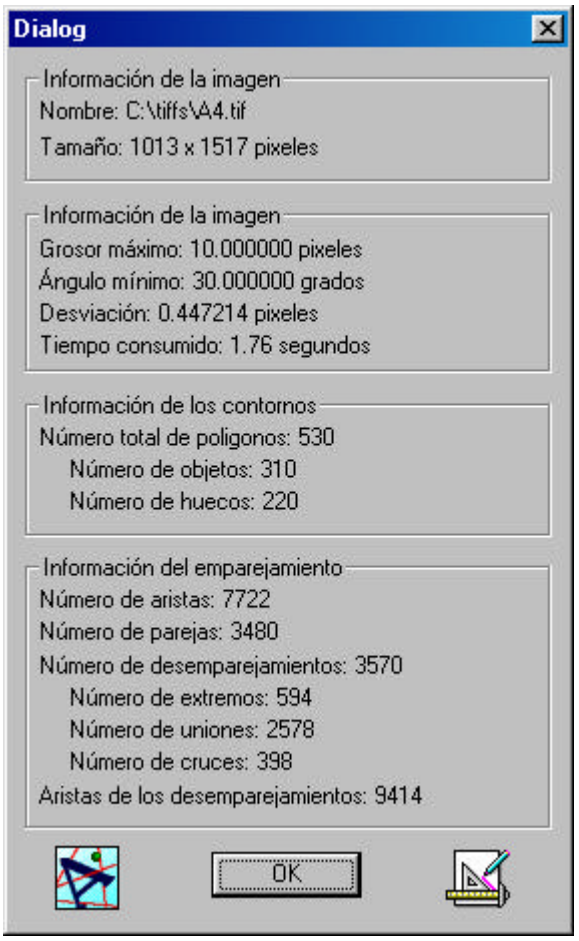
5. Ayuda.

En el menú ayuda encontramos la típica opción “Acerca de” que proporciona información sobre la versión del programa y el autor.

Información.

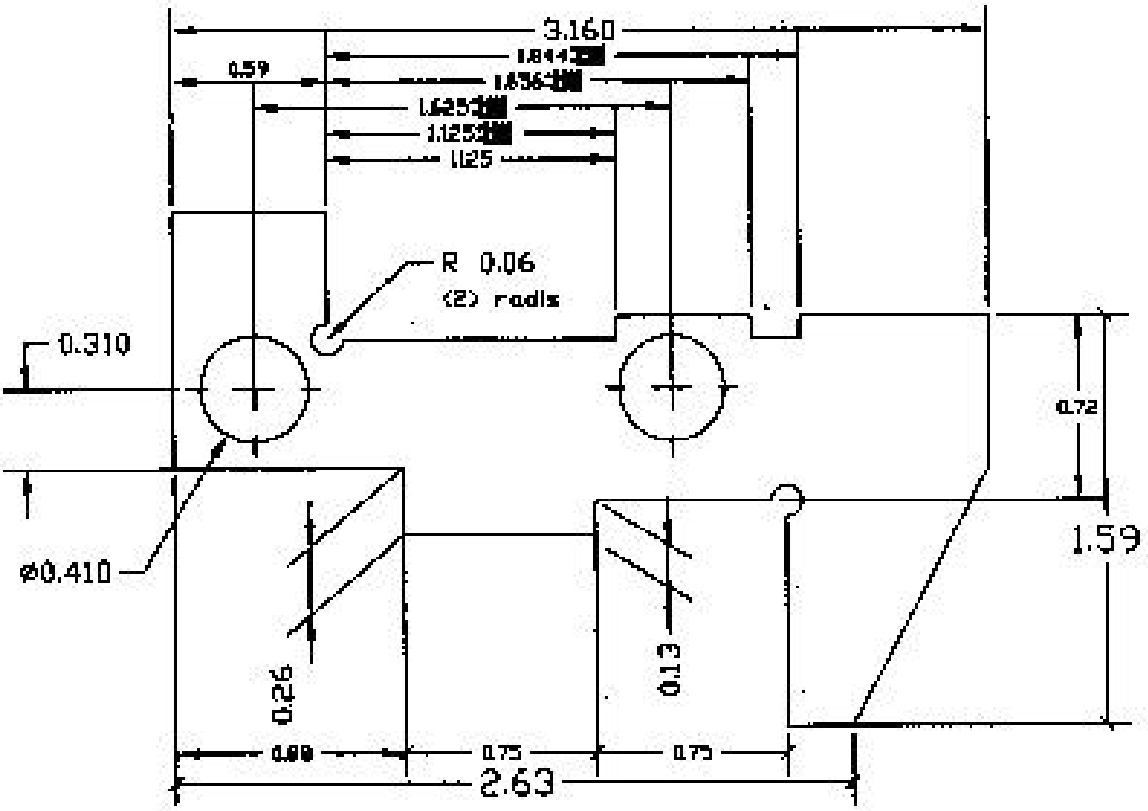
Proporciona información sobre el fichero correspondiente a la ventana activa y sobre los resultados del procesamiento de esa ventana. En caso de que no se haya procesado todavía la imagen, las tres últimas secciones de la caja de dialogo no mostrarán información alguna.

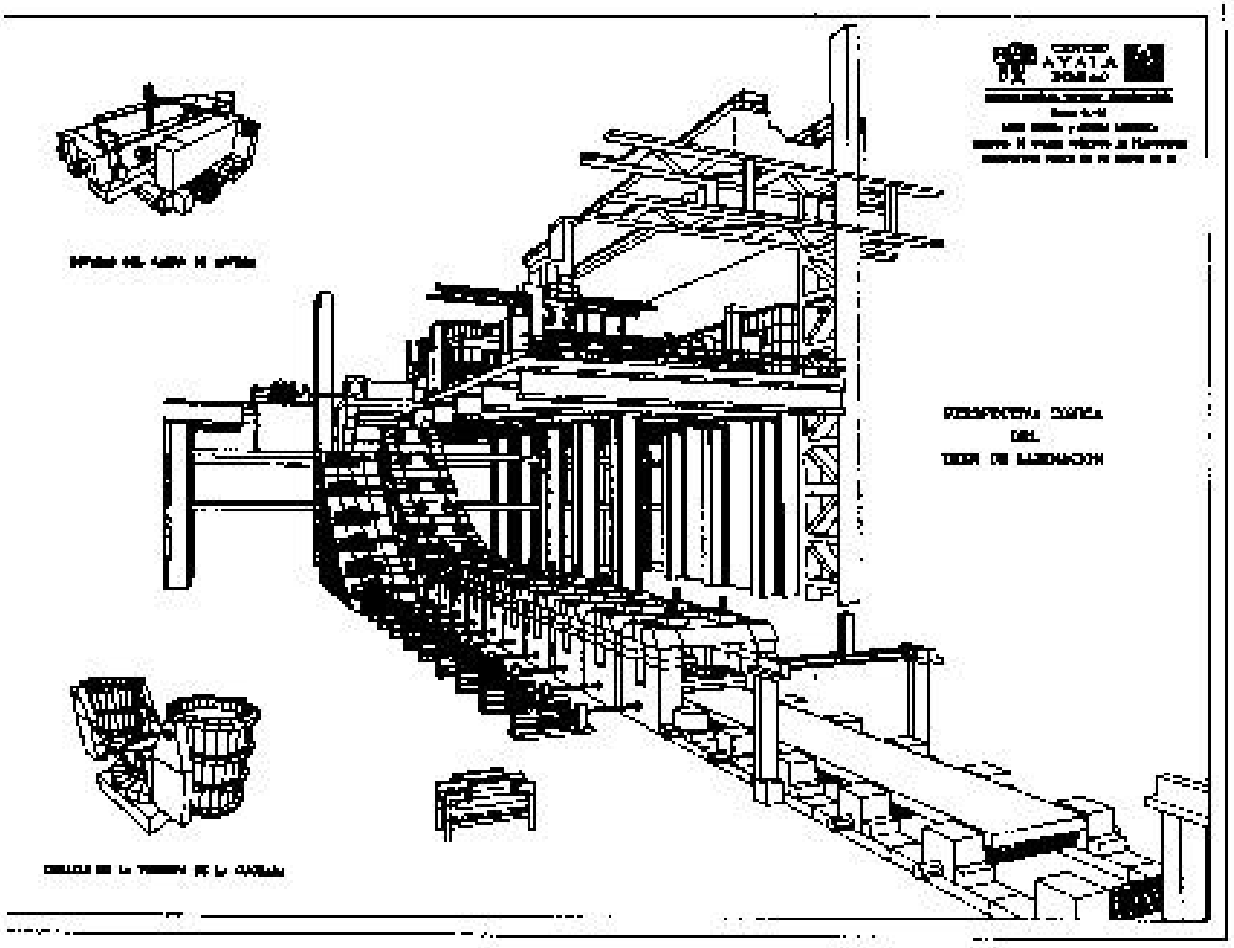
También se puede activar este comando accionando el icono  que mostrará el mismo cuadro de dialogo que la entrada de menú:


Dialog
Información de la imagen
Nombre: C:\tiffs\A4.tif
Tamaño: 1013 x 1517 pixeles
Información de la imagen
Grosor máximo: 10.000000 pixeles
Ángulo mínimo: 30.000000 grados
Desviación: 0.447214 pixeles
Tiempo consumido: 1.76 segundos
Información de los contornos
Número total de poligonos: 530
Número de objetos: 310
Número de huecos: 220
Información del emparejamiento
Número de aristas: 7722
Número de parejas: 3480
Número de desemparejamientos: 3570
Número de extremos: 594
Número de uniones: 2578
Número de cruces: 398
Aristas de los desemparejamientos: 9414
OK

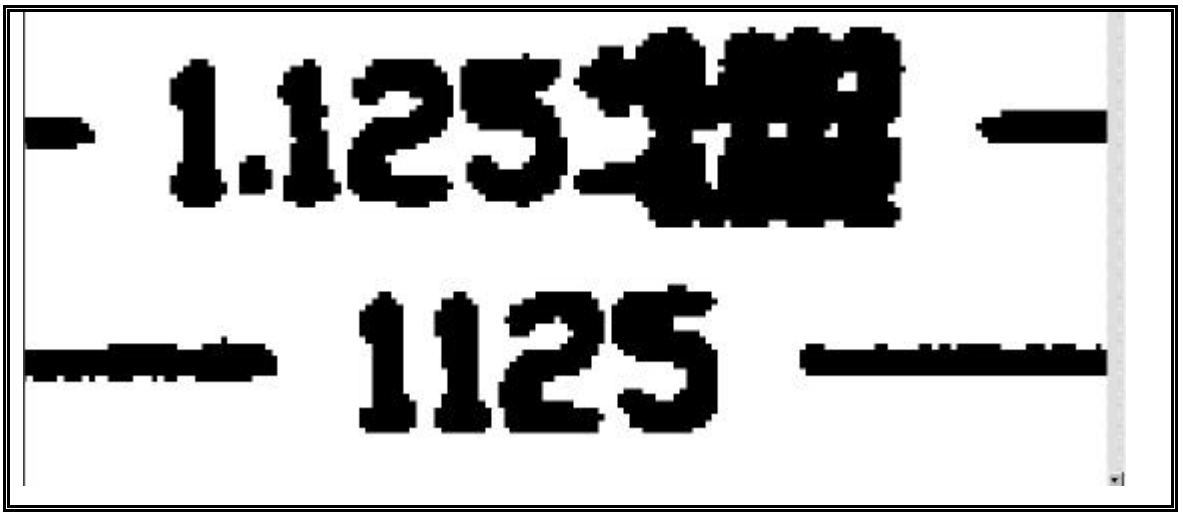
Apéndice B: Ejemplos.

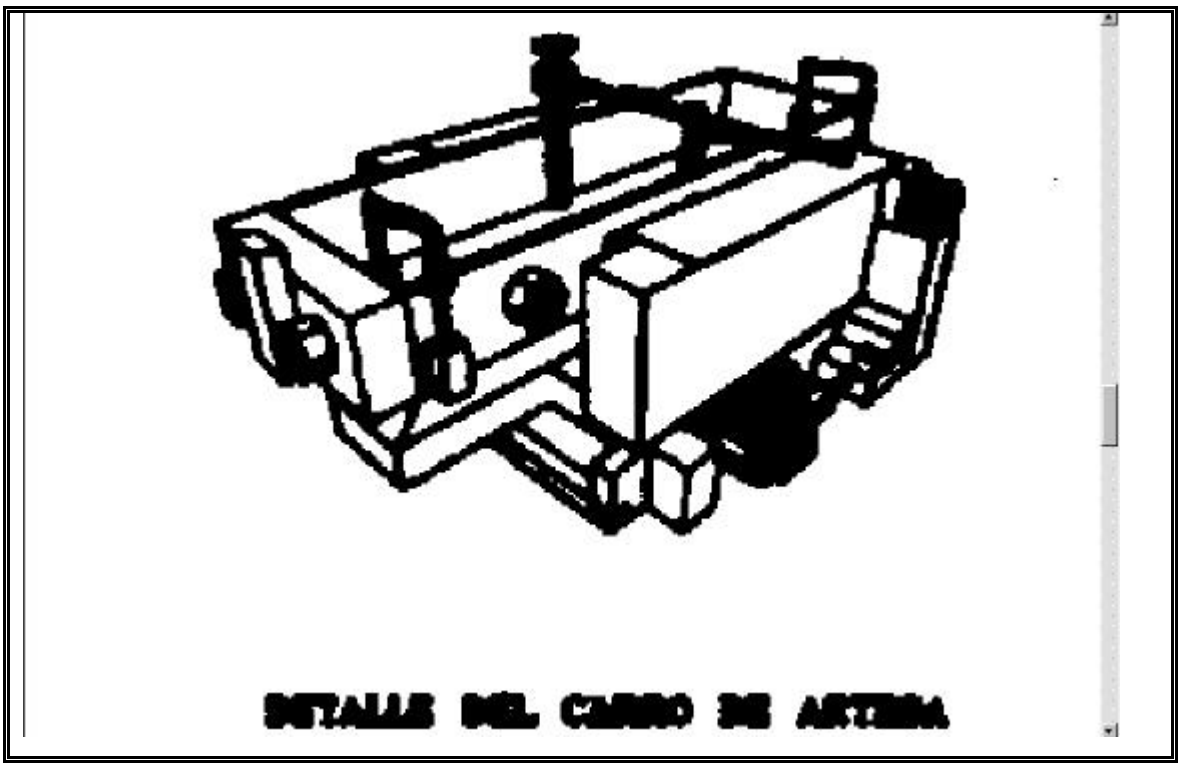
Ejemplo 1


3.160
0.59
1.125
R 0.06
(2) radis
0.310
0.72
1.59
ø0.410
0.26
0.13
0.75
0.75
2.63

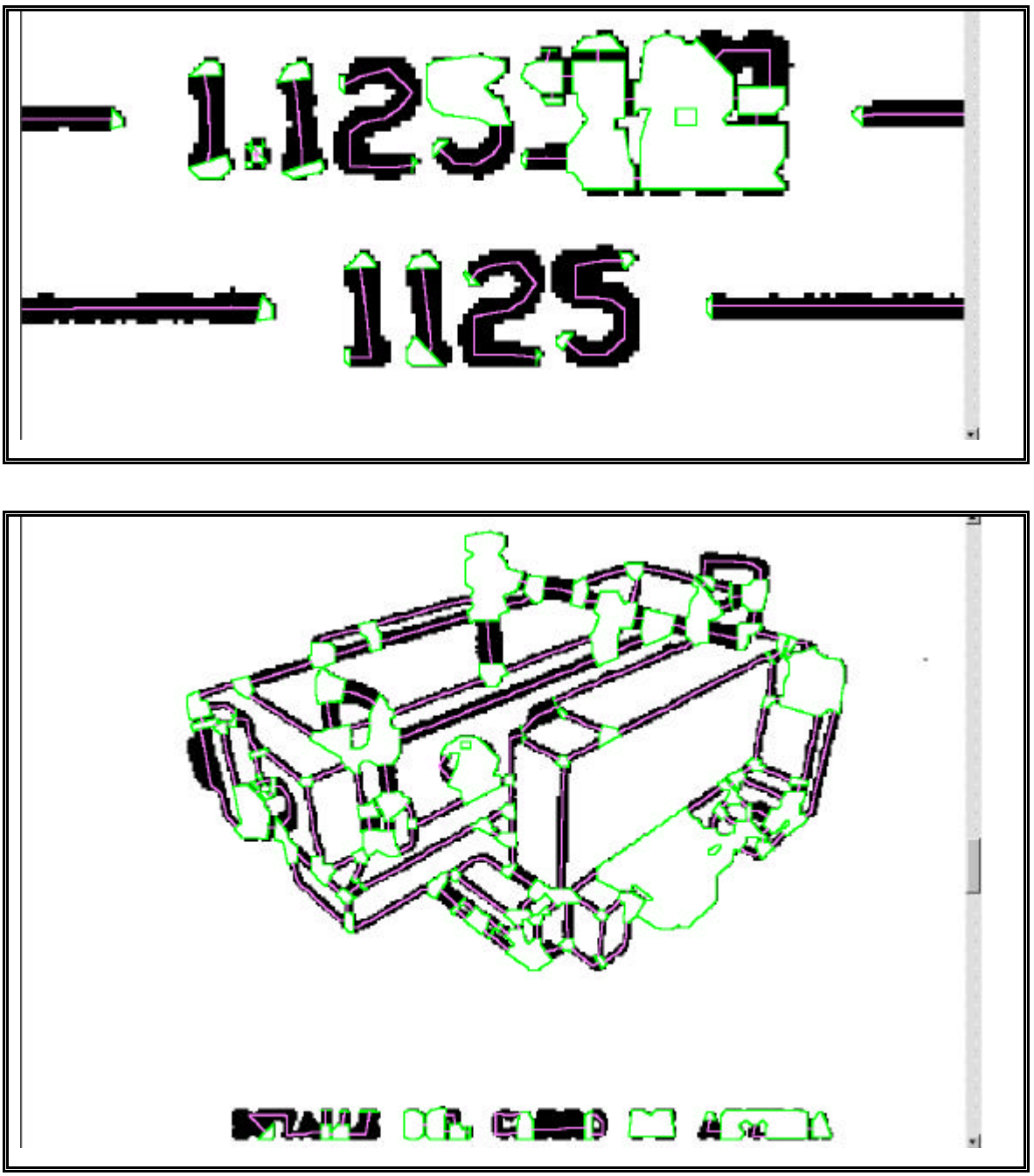


En la página anterior aparece un documento que se reproduce en la forma en que fue recibido. Simplemente se ha escalado para adaptarlo al formato de la página. Su origen parece ser una fotocopia de dos documentos que posteriormente ha sido escaneada. Como se puede apreciar en los siguientes detalles la calidad del documento deja bastante que desear:

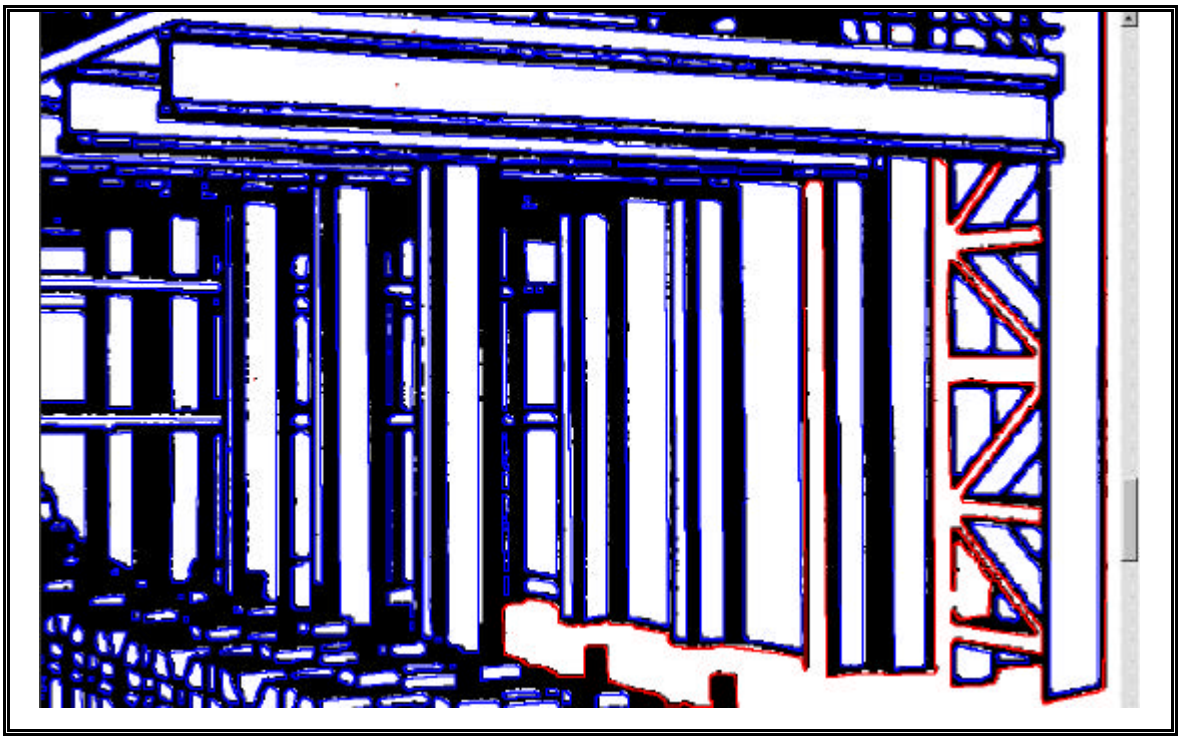


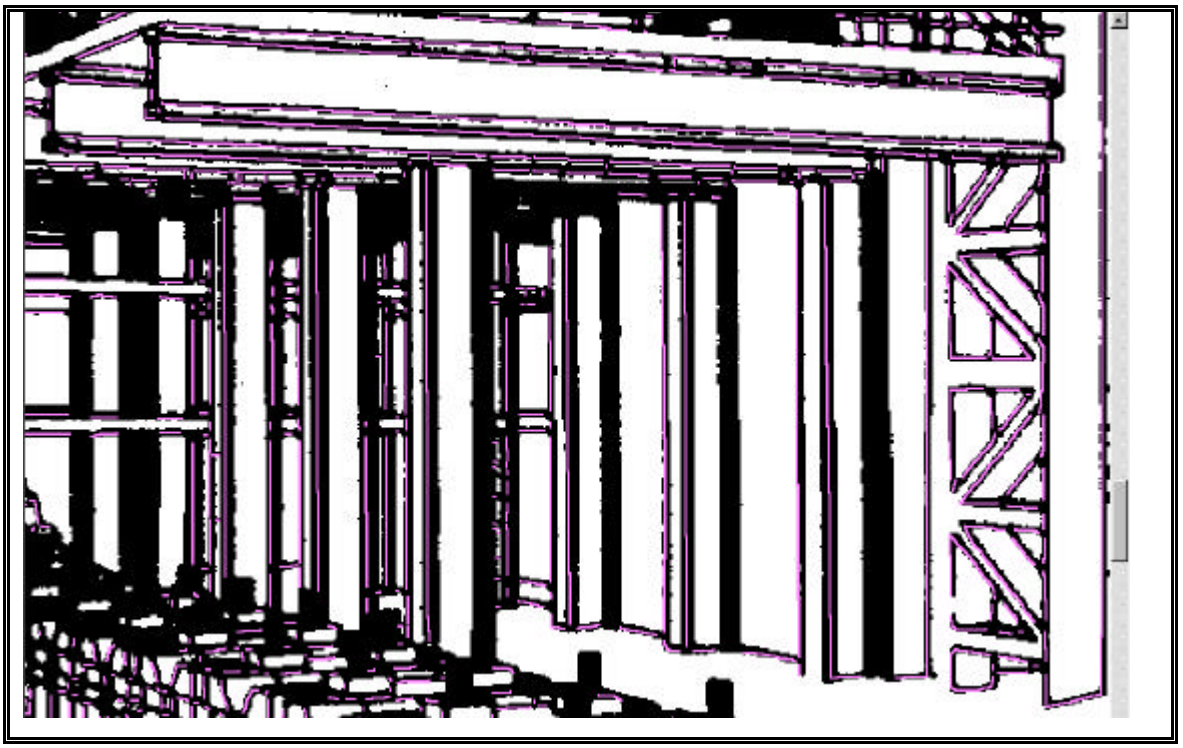


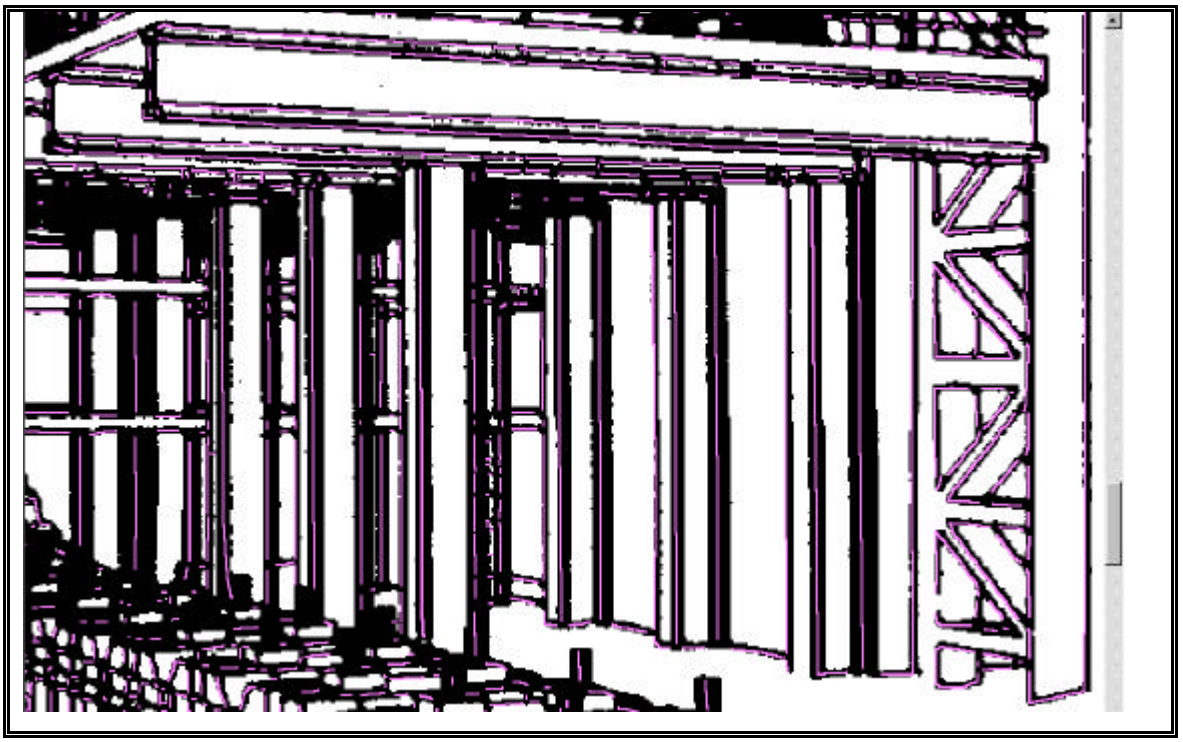
Pese a la calidad deficiente del original tras el procesamiento los resultados son los que se muestran a continuación:



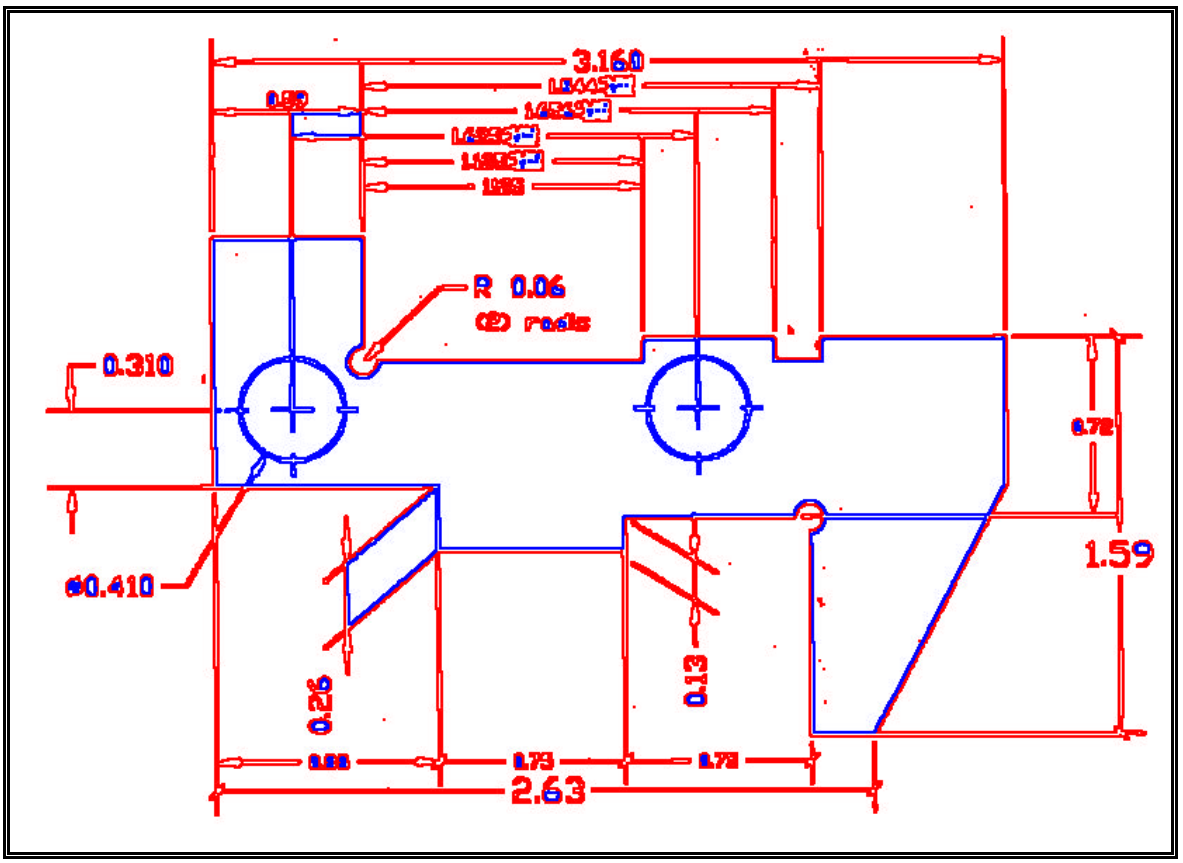
En las siguientes figuras se muestra en primer lugar un nuevo detalle del ejemplo anterior en el que se muestra la aproximación de los contornos y después el resultado del procesado con un parámetro de grosor máximo de 10 pixeles en el primer caso y de 20 pixeles en el segundo. Como se puede apreciar las gruesas líneas verticales del centro de la figura que representan los pilares del tren de laminación no se han detectado como rectas al utilizar el primer valor para el parámetro y por el contrario con el valor 20 han sido detectadas todas. Esto se debe a que el grosor de estas líneas está por encima de los 10 pixeles que el programa toma por defecto como parámetro.





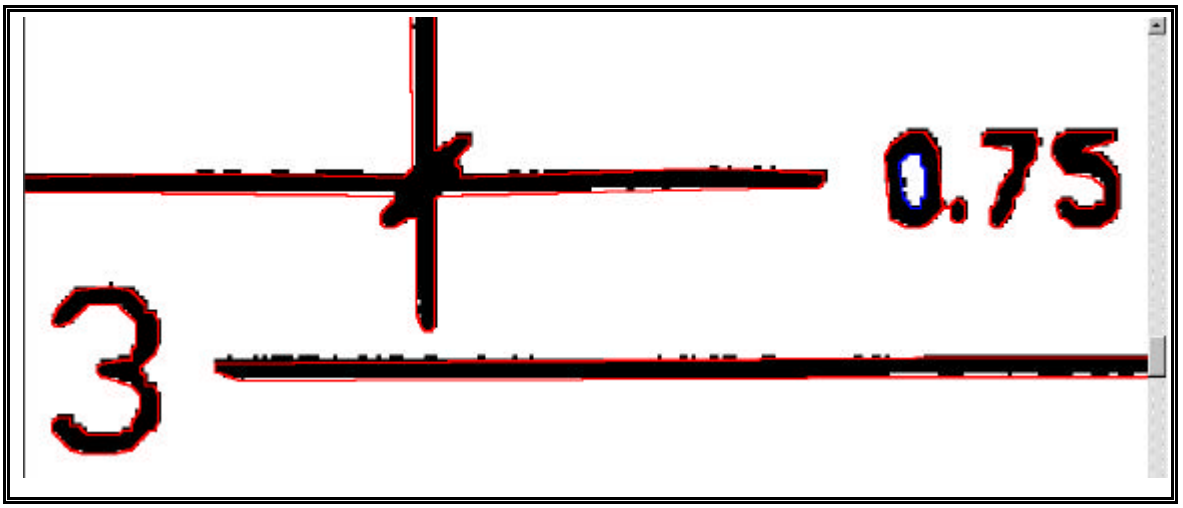


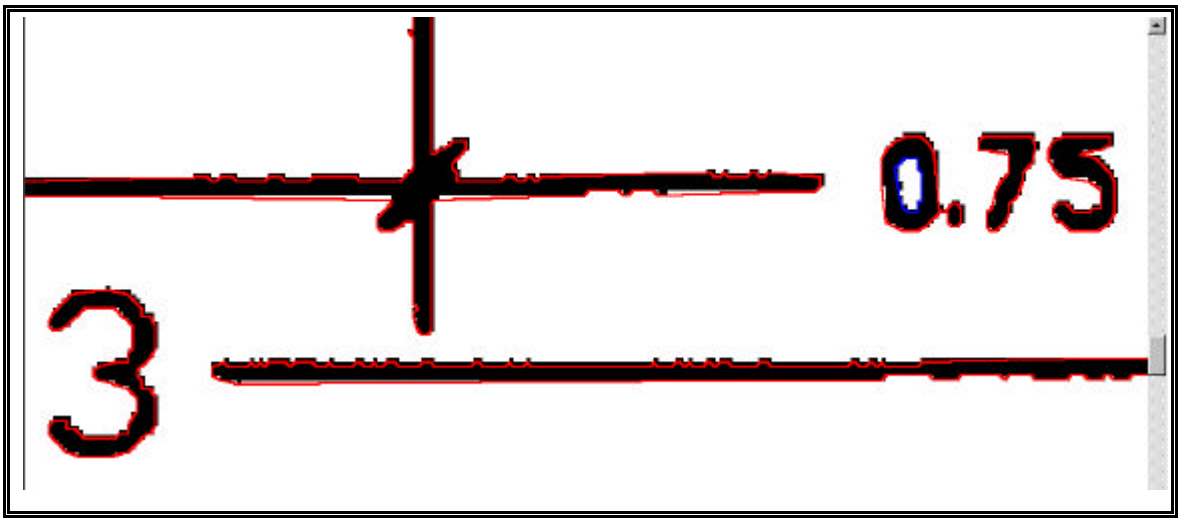
A continuación veremos un detalle de los contornos de la parte de arriba del ejemplo, obtenidos tras el procesamiento:



Ahora veremos todavía más ampliada la zona inmediatamente a la derecha de la cifra de cota 2.63. Hay que destacar el abundante ruido que aparece en los contornos del

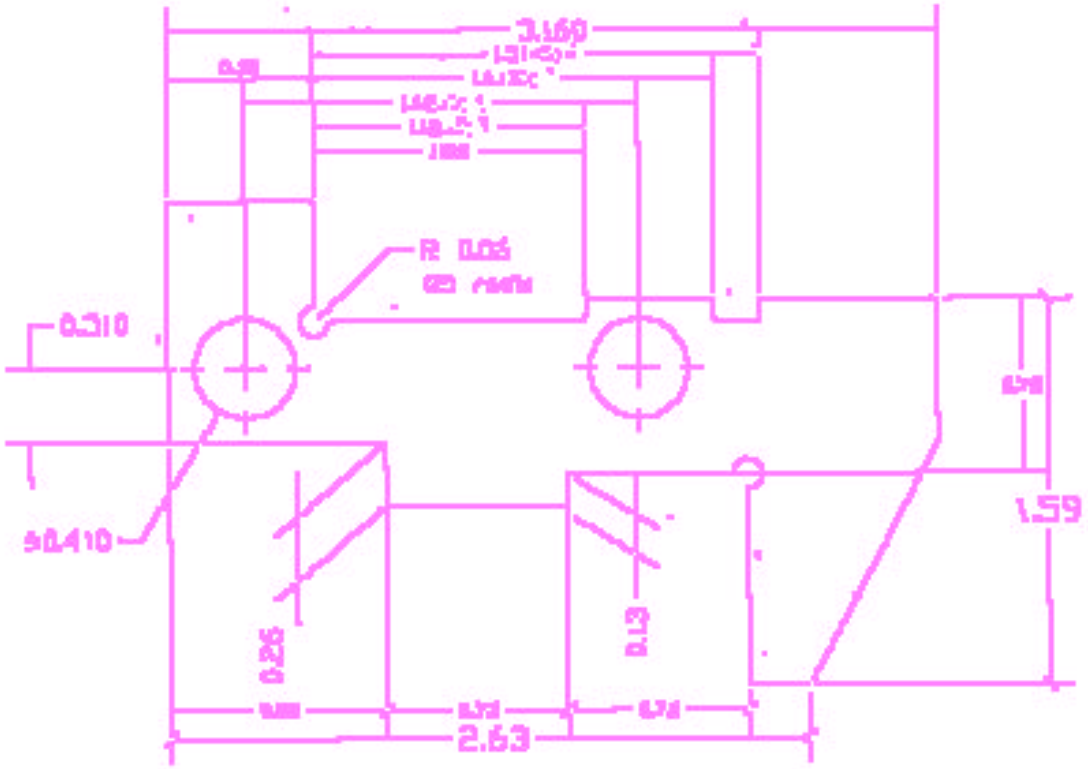
fragmento de la línea de cota que aparece a la derecha. En la figura siguiente veremos la misma área pero procesada con una versión especial del programa que no filtra los contornos.

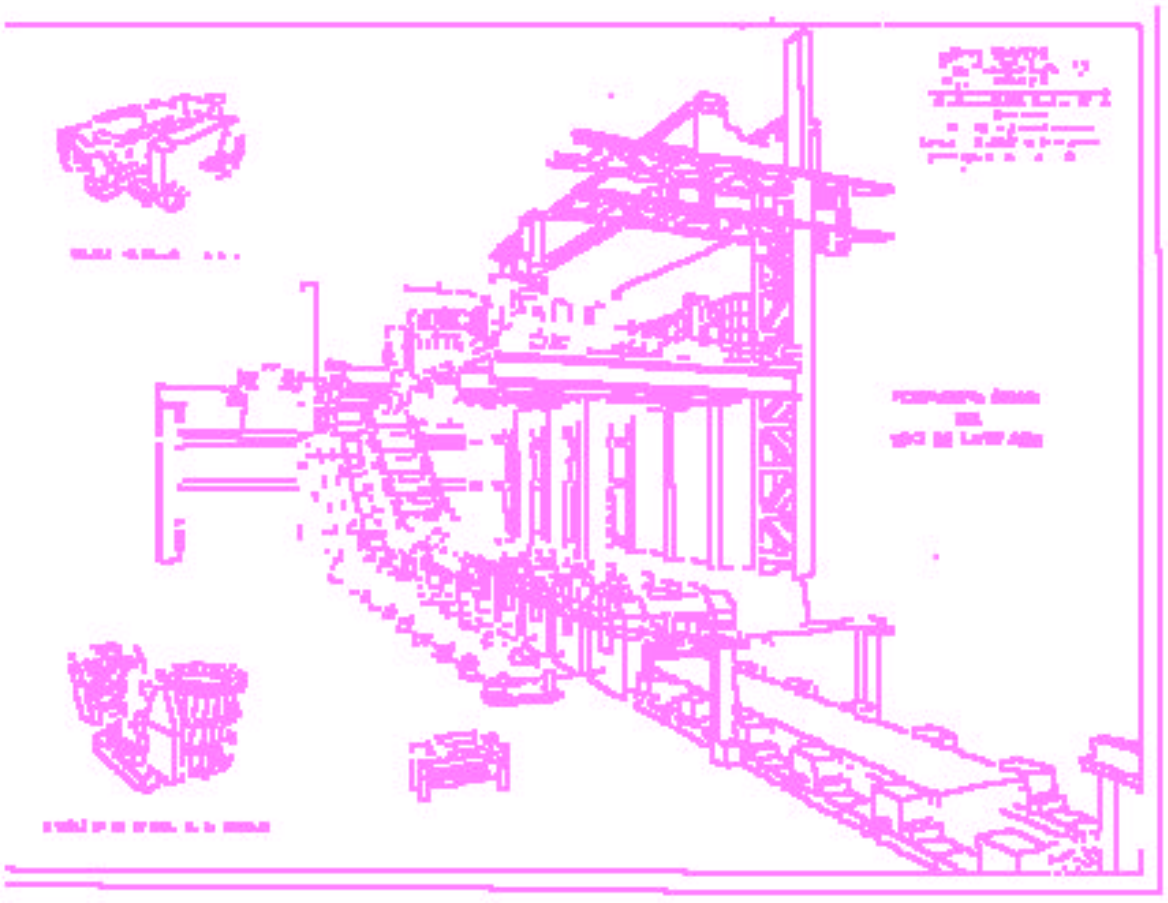


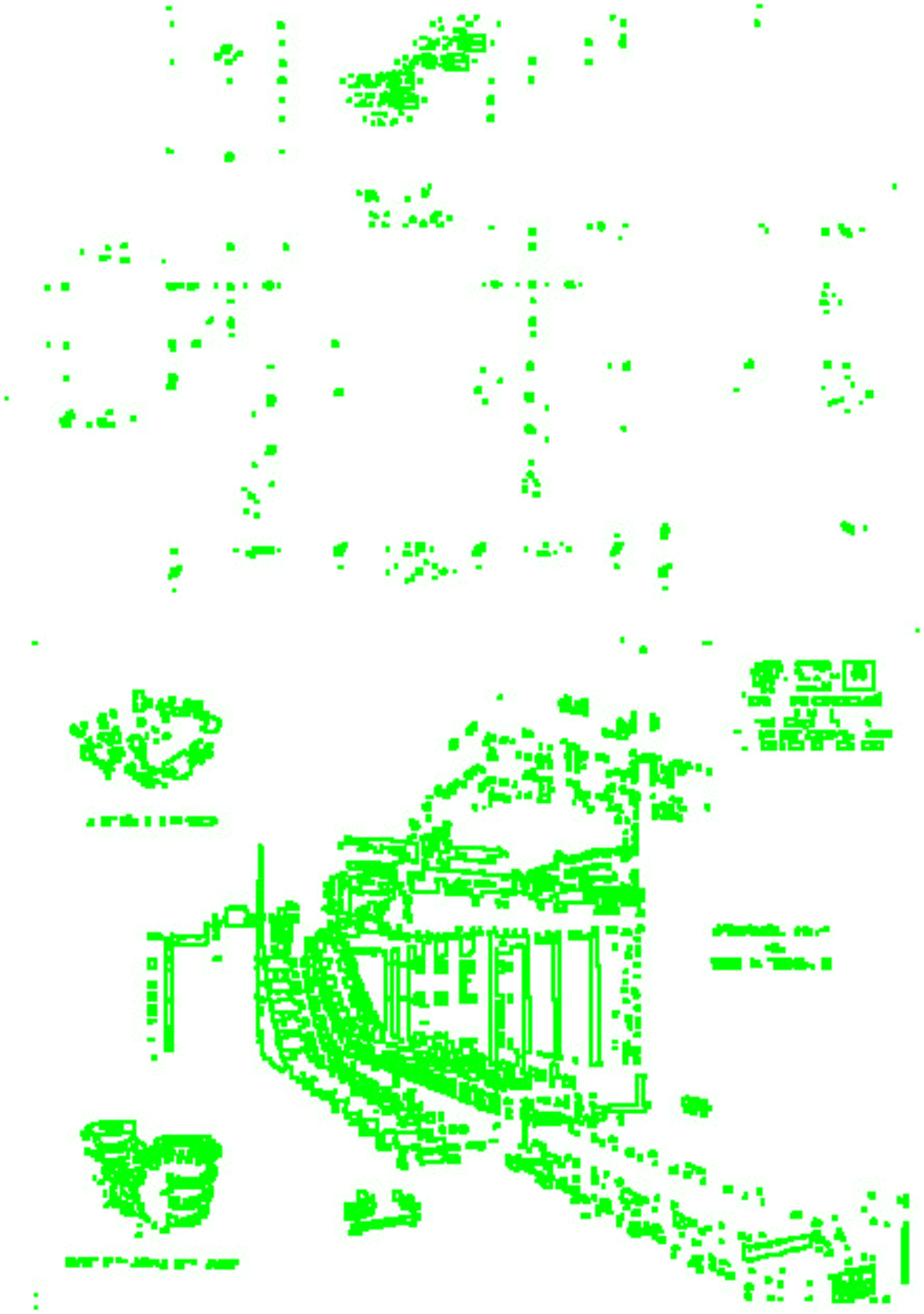


Como puede apreciarse en esta segunda figura en la que el contorno no se ha sometido a ningún filtrado aparecen multitud de dientes de sierra en los contornos que provocarían que la línea no emparejase en toda su longitud y por tanto quedaría dividida en multitud de pequeños segmentos.

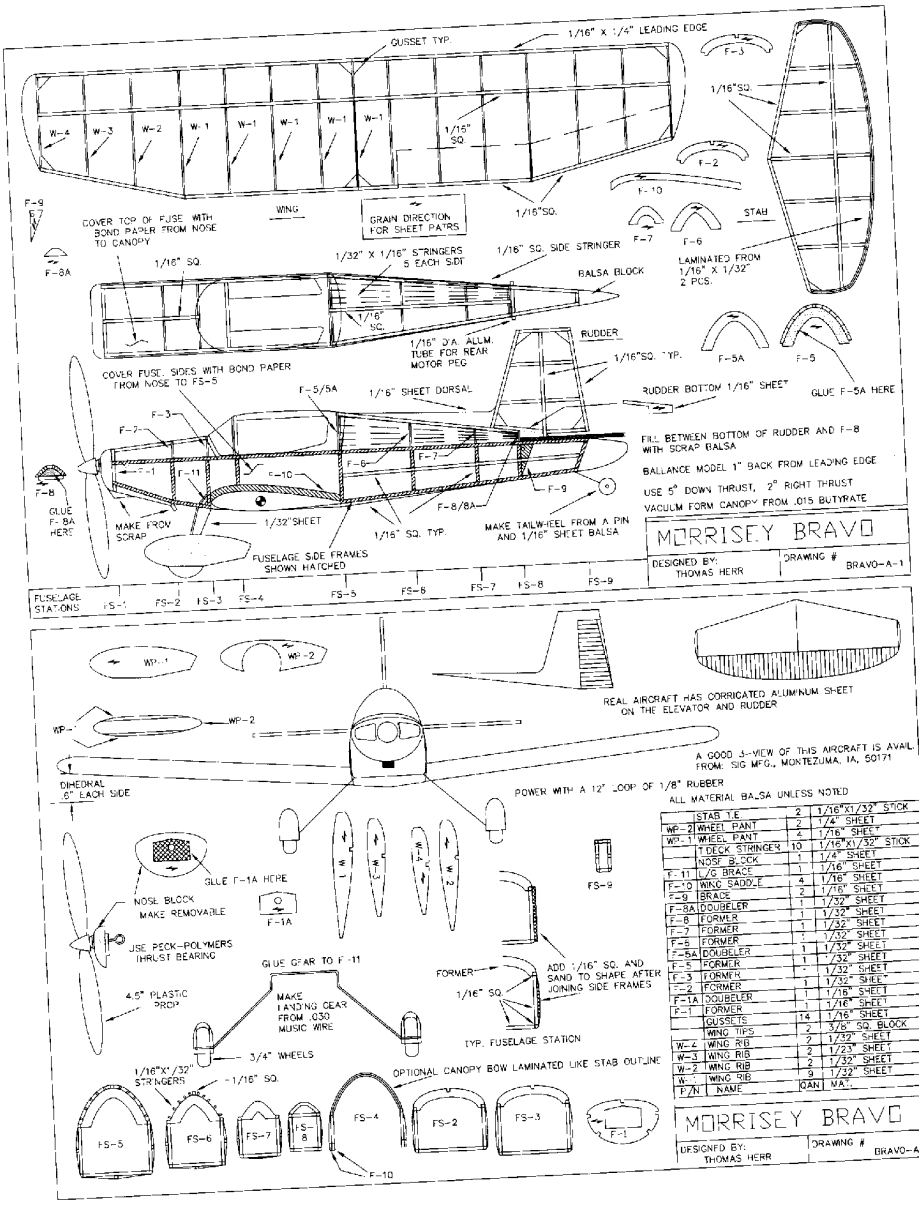
Para finalizar con este ejemplo mostraremos por separado las líneas detectadas y las áreas no emparejadas que se detectan en todo el ejemplo sometiéndolo al proceso completo con el grosor máximo fijado a 20 pixeles.


0.510
0.410
0.26
0.13
1.59
2.63



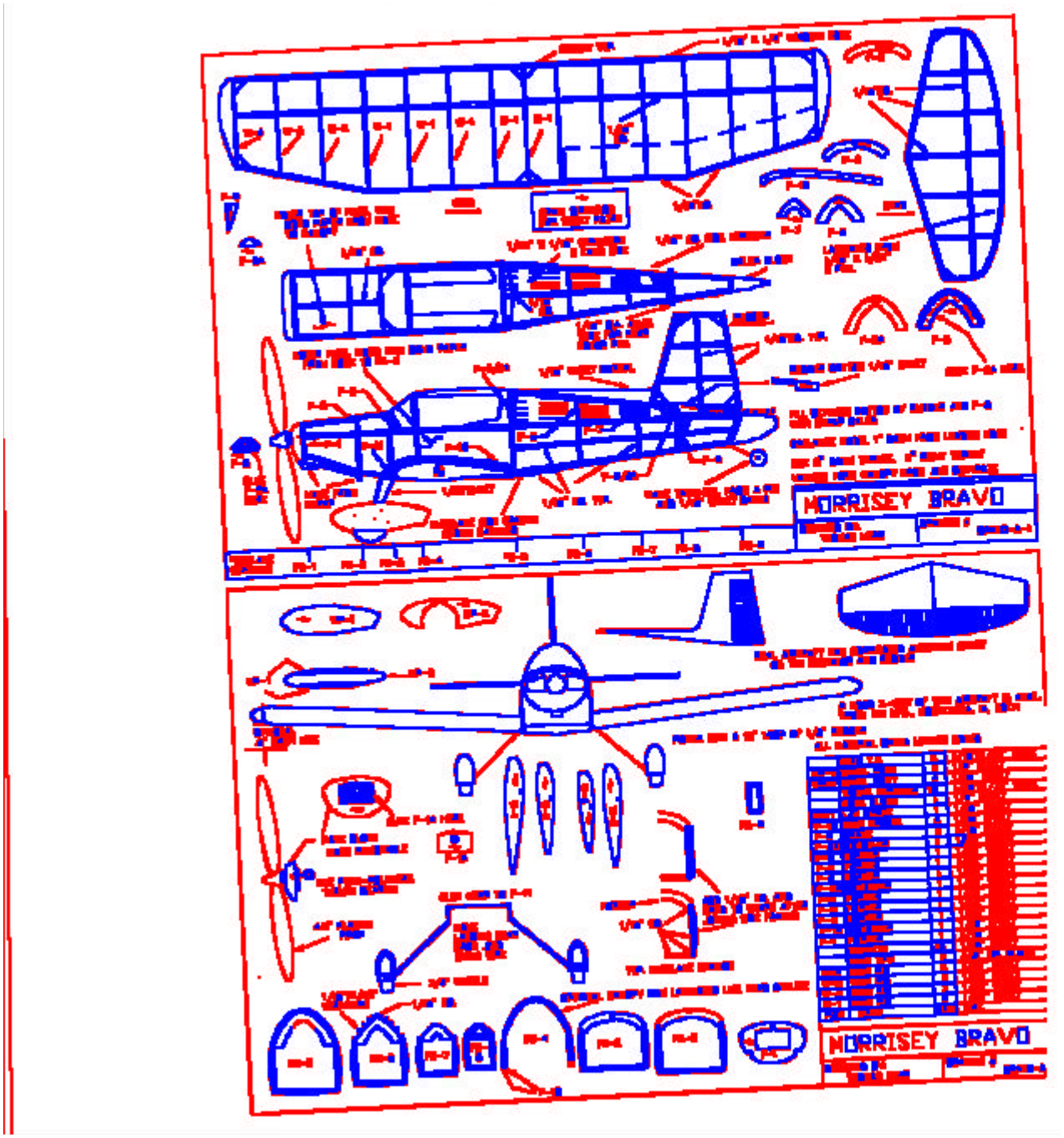


Ejemplo 2

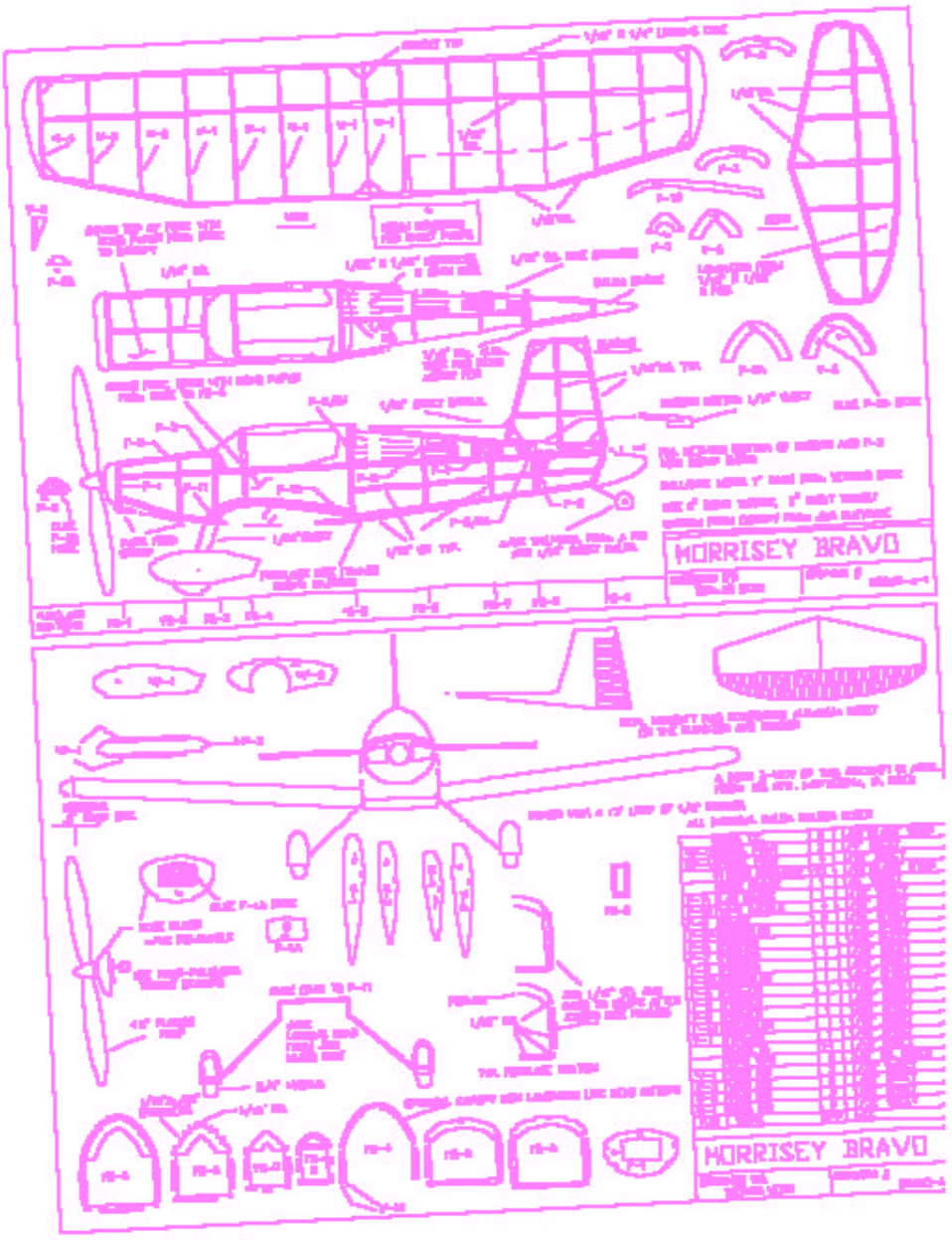


F/N	NAME	QAN	MAT.
	STAB T.E.	2	1/16"X1/32" STICK
WP-2	WHEEL PANT	2	1/4" SHEET
WP-1	WHEEL PANT	4	1/16" SHEET
	T DECK STRINGER	10	1/16"X1/32" STICK
	NOSE BLOCK	1	1/4" SHEET
F-11	L/G BRACE	1	1/16" SHEET
F-10	WING SADDLE	4	1/16" SHEET
F-9	BRACE	2	1/16" SHEET
F-8A	DOUBLER	1	1/32" SHEET
F-8	FORMER	1	1/32" SHEET
F-7	FORMER	1	1/32" SHEET
F-6	FORMER	1	1/32" SHEET
F-5A	DOUBLER	1	1/32" SHEET
F-5	FORMER	1	1/32" SHEET
F-3	FORMER	1	1/32" SHEET
F-2	FORMER	1	1/32" SHEET
F-1A	DOUBLER	1	1/16" SHEET
F-1	FORMER	1	1/16" SHEET
	GUSSETS	14	1/16" SHEET
	WING TIPS	2	3/8" SQ. BLOCK
W-4	WING RIB	2	1/32" SHEET
W-3	WING RIB	2	1/32" SHEET
W-2	WING RIB	2	1/32" SHEET
W-1	WING RIB	9	1/32" SHEET

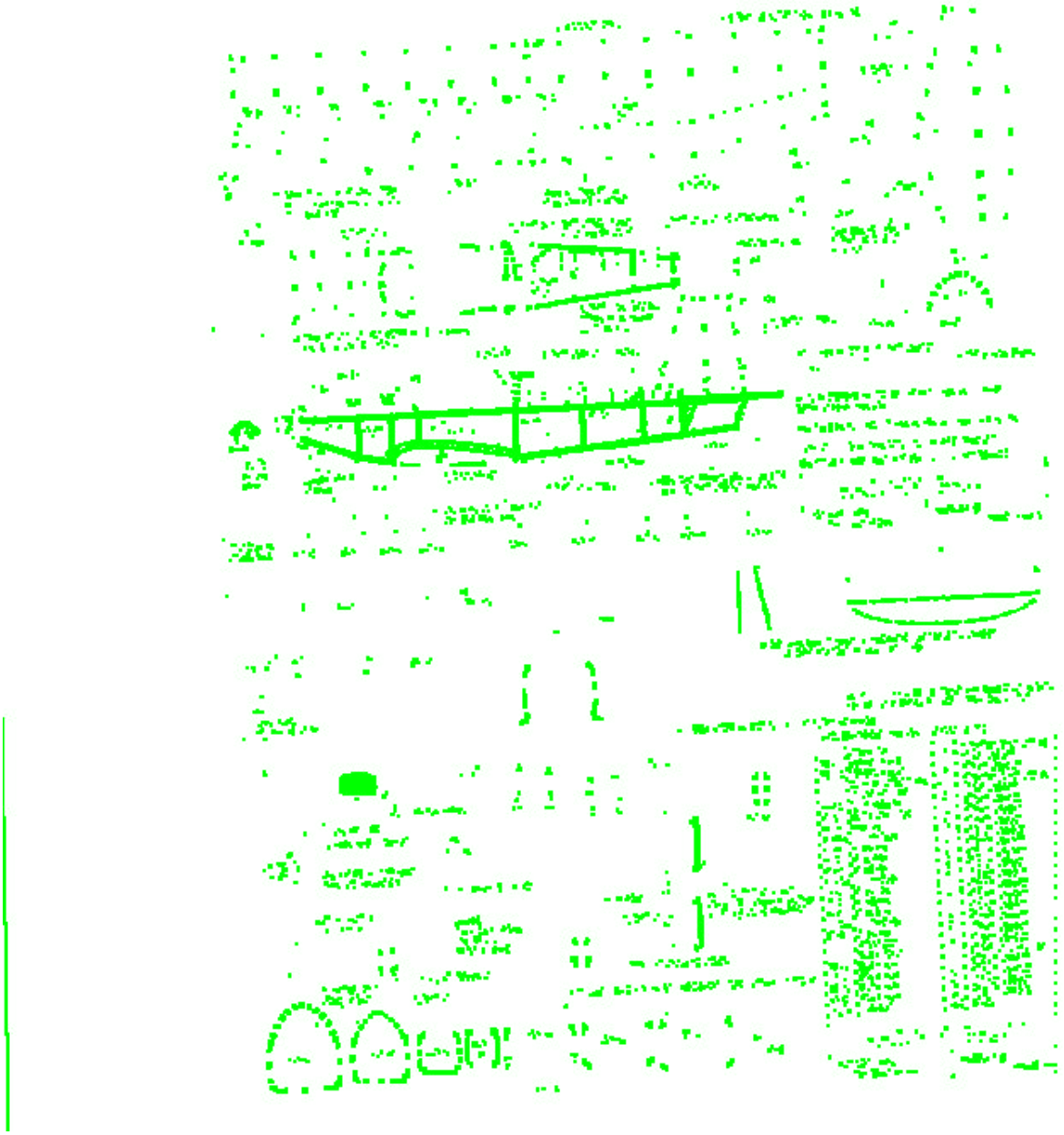
La imagen muestra el plano escaneado de una maqueta de aeromodelismo. La inclinación y el incorrecto centrado se deben a la falta de cuidado al escanear el plano.



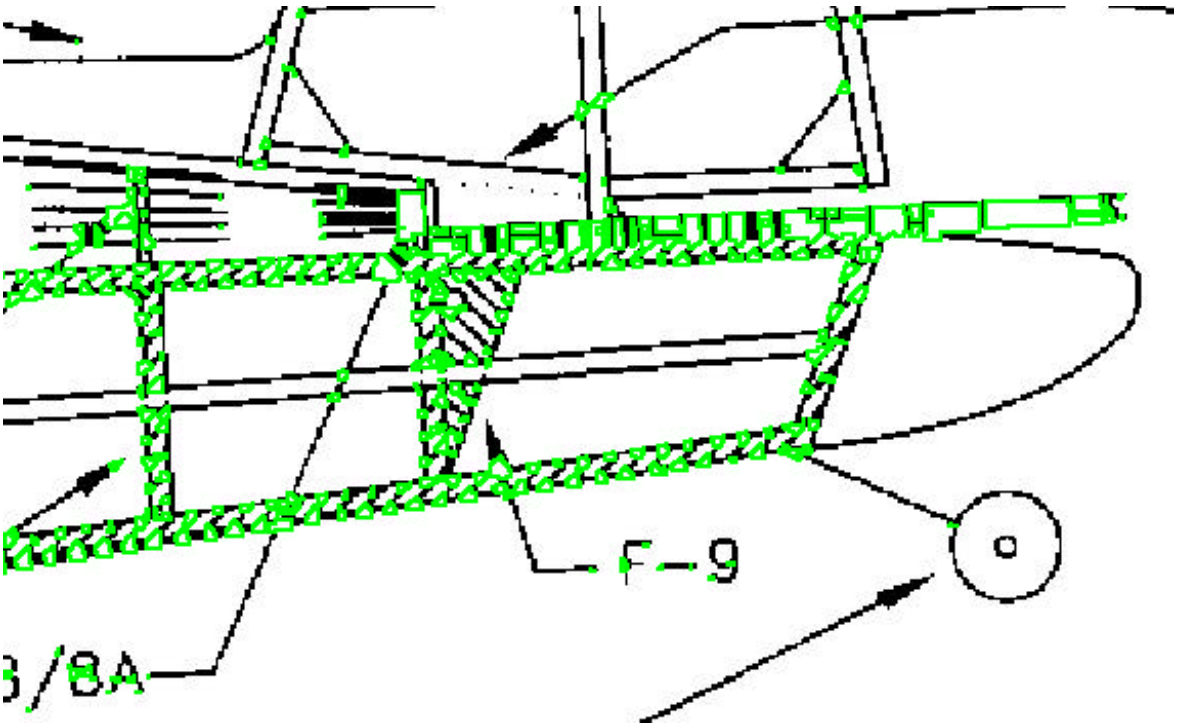
El dibujo muestra los contornos interiores (azul) y exteriores (rojo) del plano una vez procesado por nuestro algortimo.



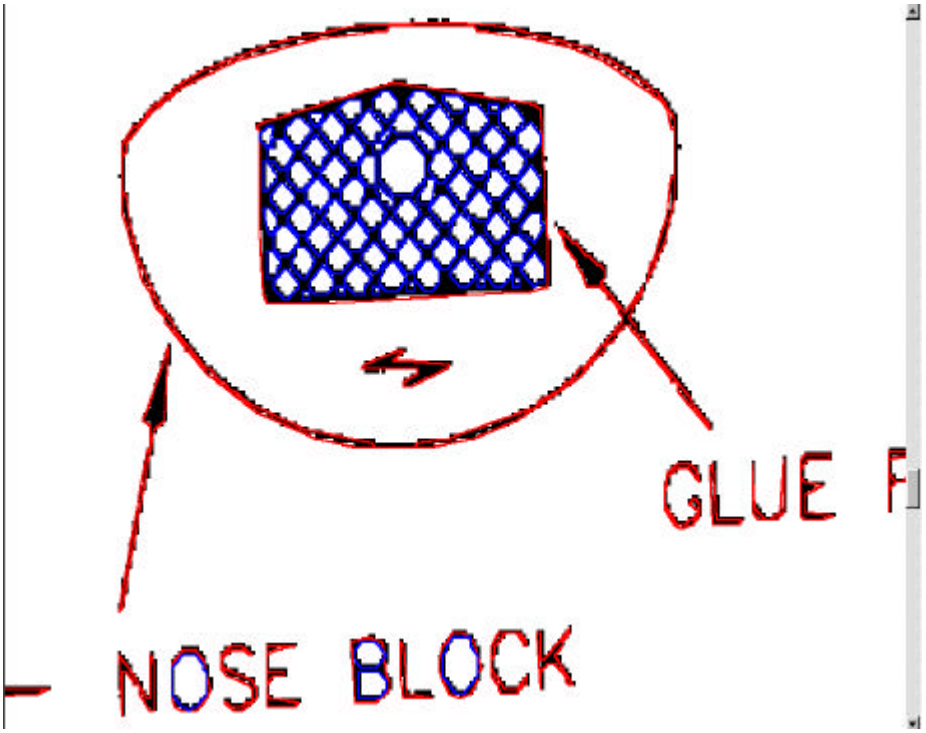
La figura muestra las líneas axiales correspondientes a las parejas detectadas en el proceso de emparejamiento.



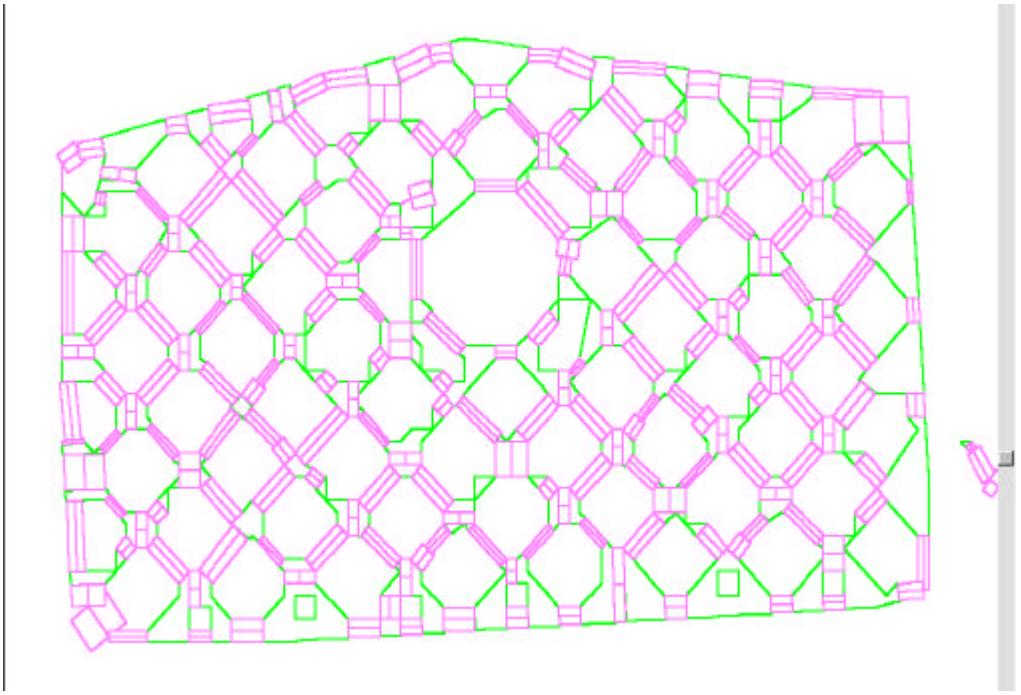
En la figura podemos observar las áreas no emparejadas obtenidas como resultado de engrosamientos en el ráster.



La anterior figura es un detalle del centro de la imagen en donde aparentemente hay una gran área desemparejada que realmente es gran número de desemparejamientos de pequeño tamaño originados por un rayado del plano original situados muy próximos.



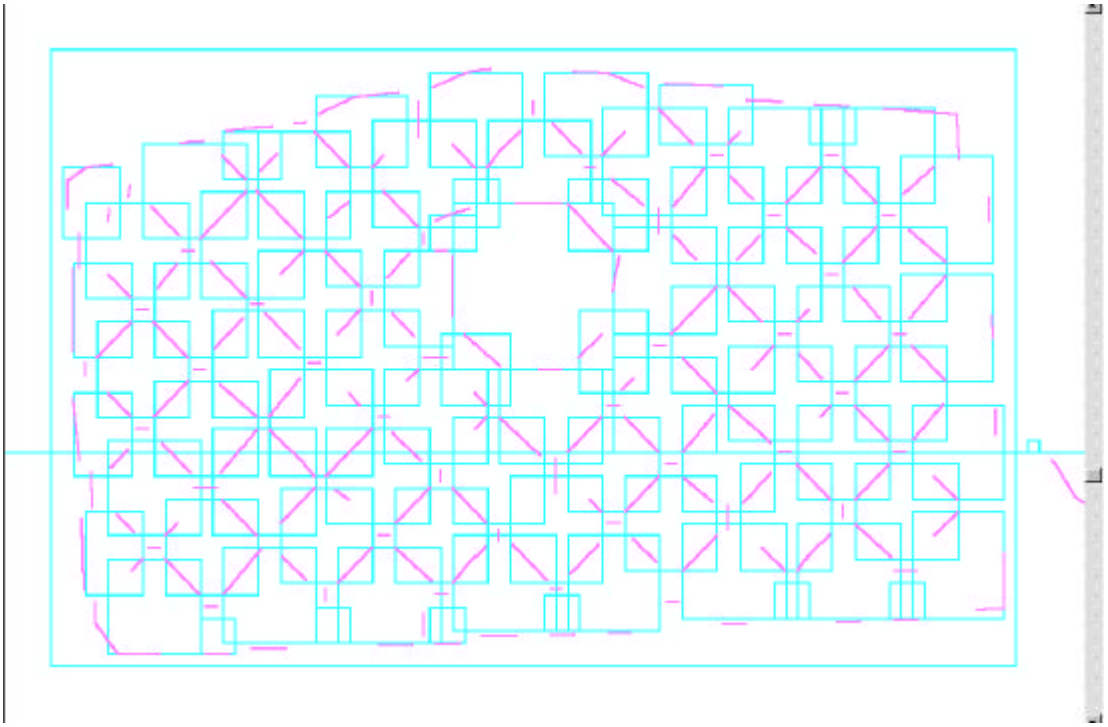
En la figura vemos un detalle de los contornos de un área correspondiente a un rayado con pendientes a 45º y a -45º.



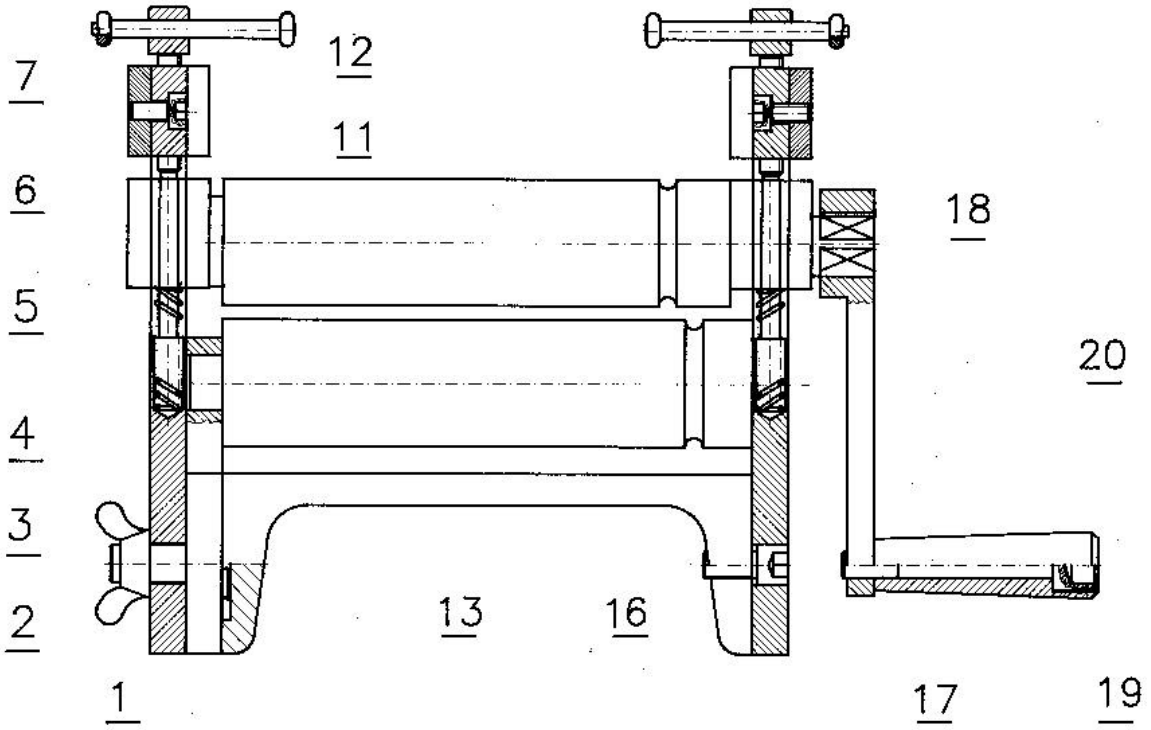
En esta ocasión vemos un detalle todavía más ampliado de las líneas detectadas (color rosa) y de las áreas desemparejadas (color verde). En las líneas se ha representado el grosor detectado, como un rectángulo entorno a ellas.

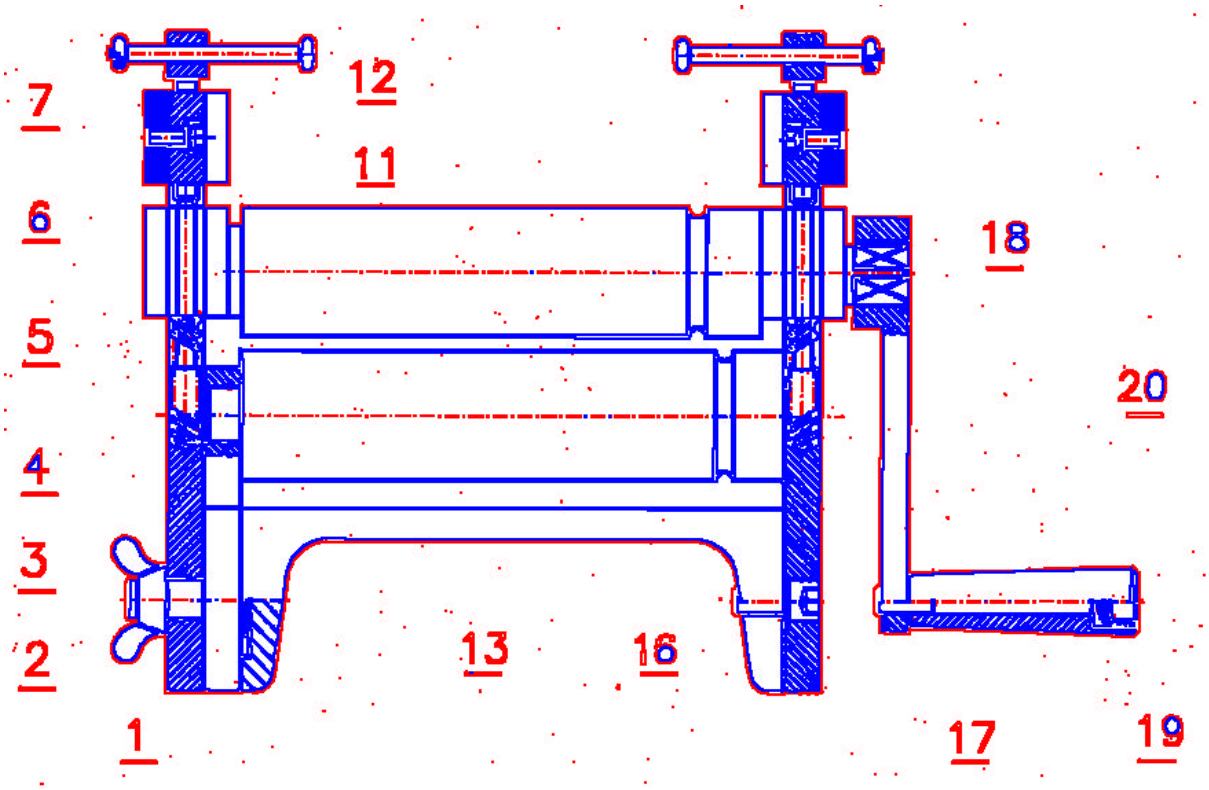
Se puede apreciar cierto patrón repetido de líneas y emparejamientos alternándose en un orden particular y con ángulos muy similares. En caso de poder ser detectados los circuitos así formados en la estructura de datos de contornos desemparejados se podría aproximar todo este gran número de líneas y polígonos por simples etiquetas que indicarían la presencia de un rayado y el tipo de este.

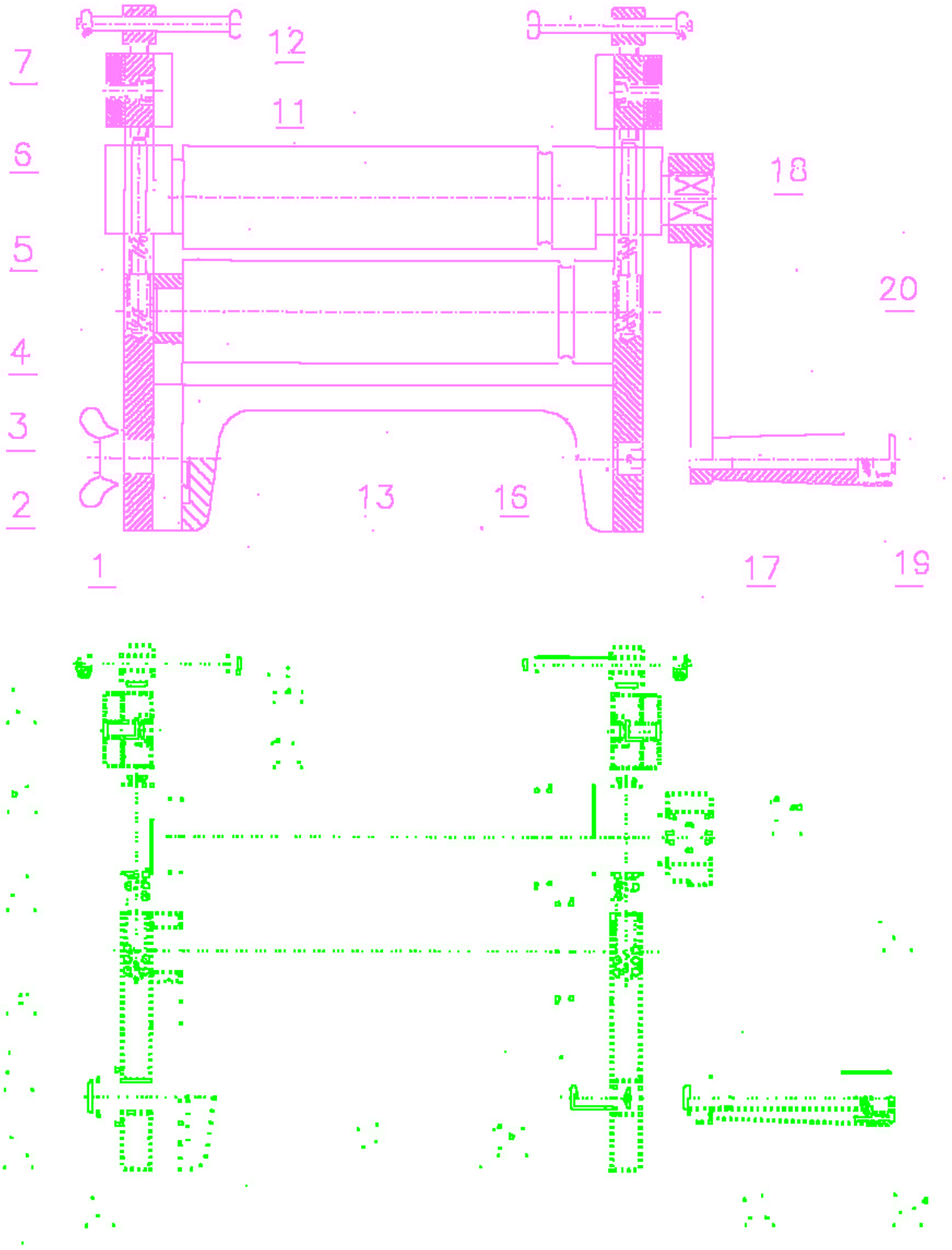
También podían ser de ayuda para este proceso las cajas de circunscripción que aparecen representadas en la siguiente figura. En ellas también se puede apreciar la regularidad que puede permitir al programa estar alerta ante un posible rayado.



Ejemplo 3

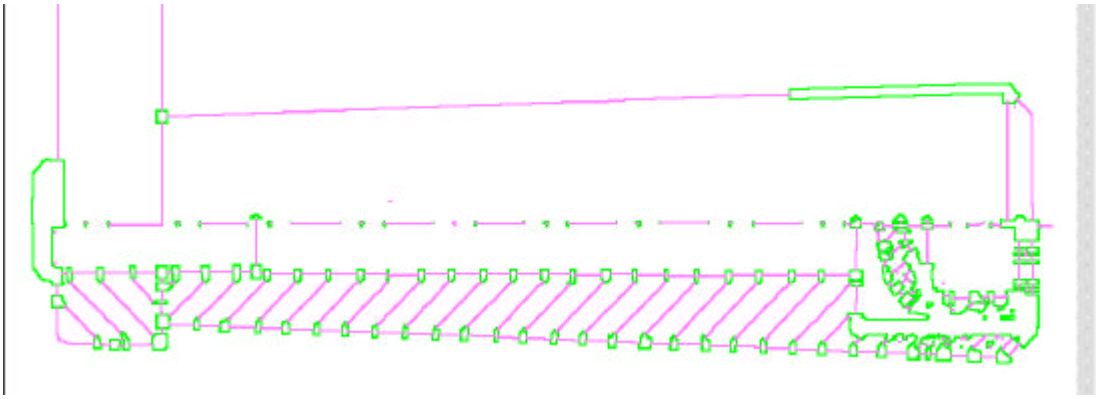

7
6
5
4
3
2
1
12
11
18
20
13
16
17
19

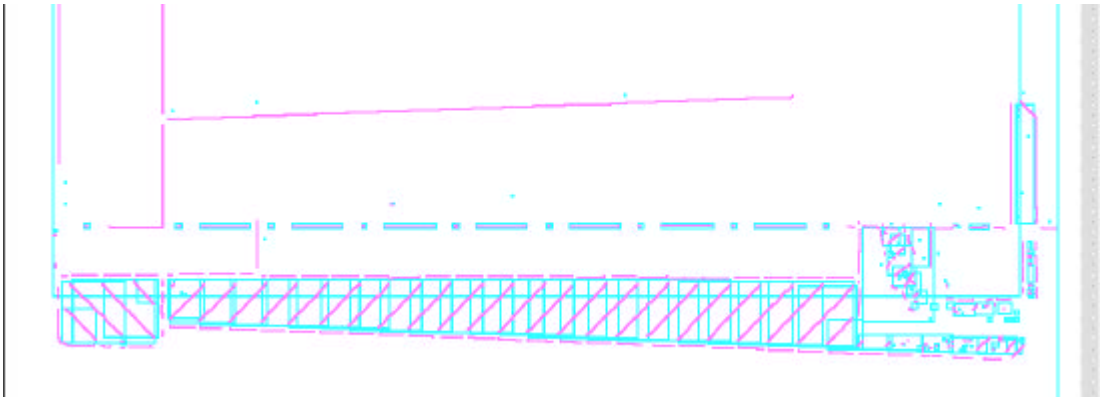

7
6
5
4
3
2
1
12
11
18
20
13
16
17
19



Anteriormente hemos podido ver las cuatro imágenes de costumbre que representan las estructuras calculadas por el programa desde sus diferentes puntos de vista.

A continuación podemos ver un detalle del rayado de la manivela. Este rayado es de diferente tipo al visto con anterioridad pero todos los comentarios que se realizaron entonces siguen siendo igualmente válidos.





Ejemplo 4

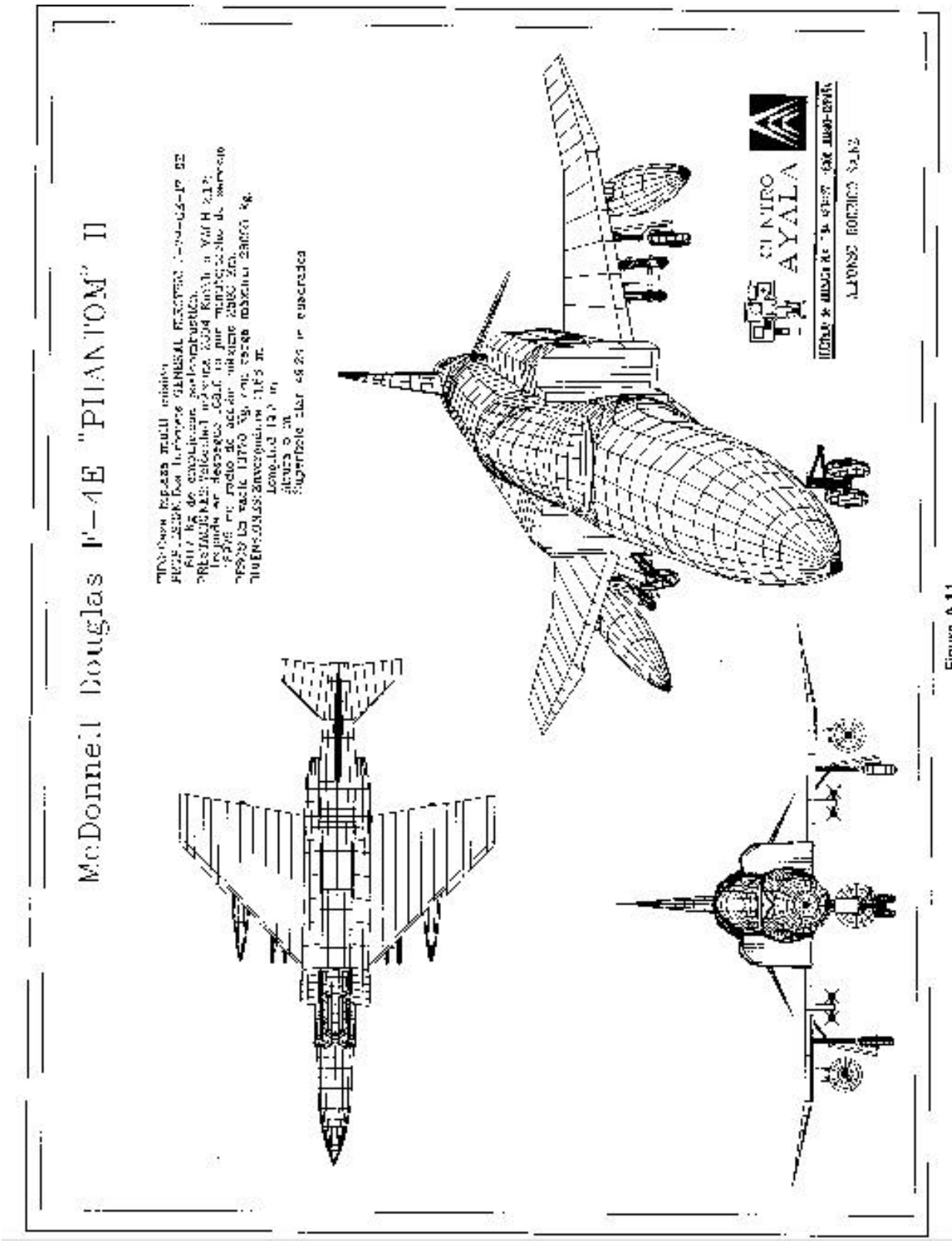
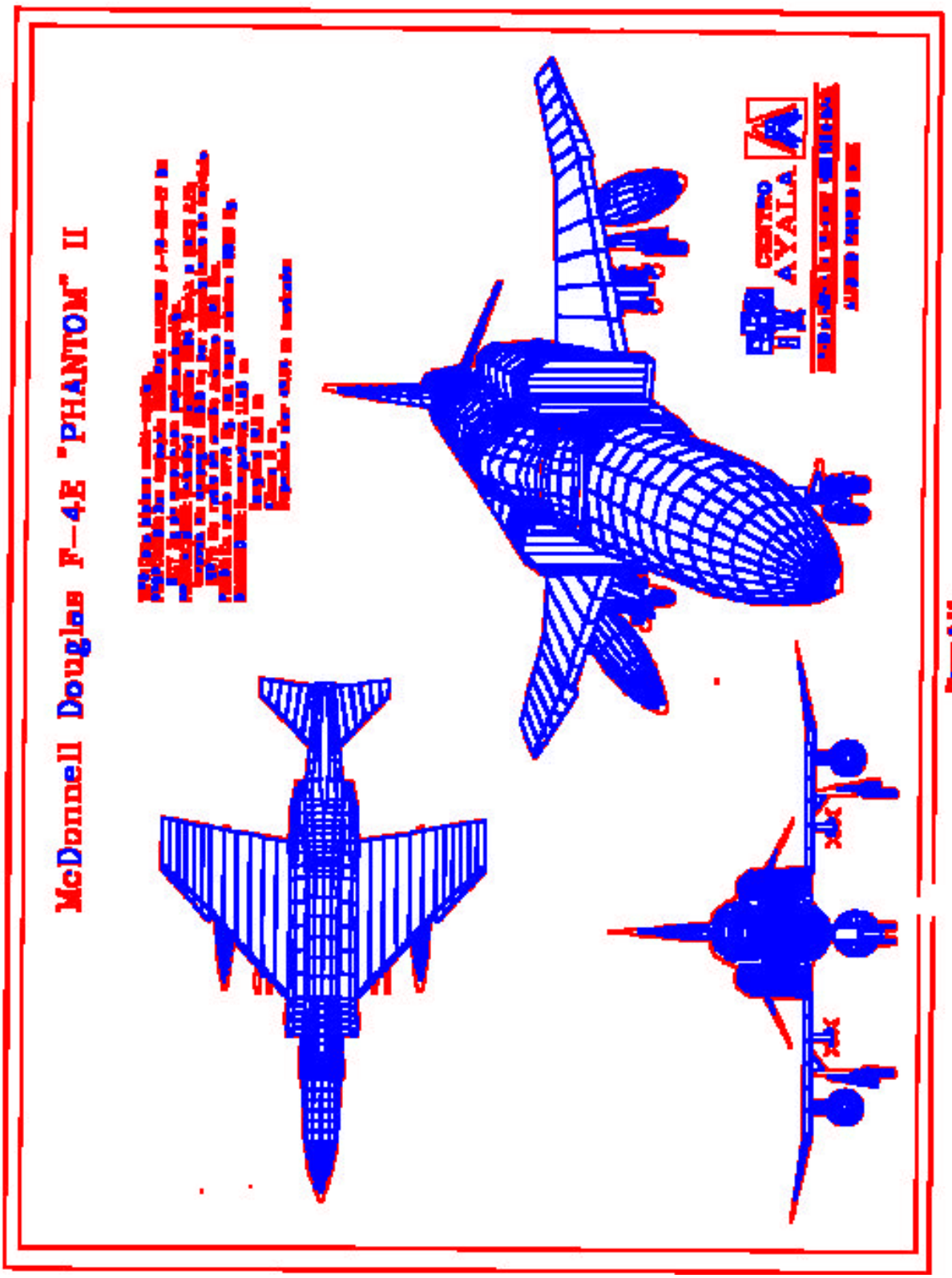
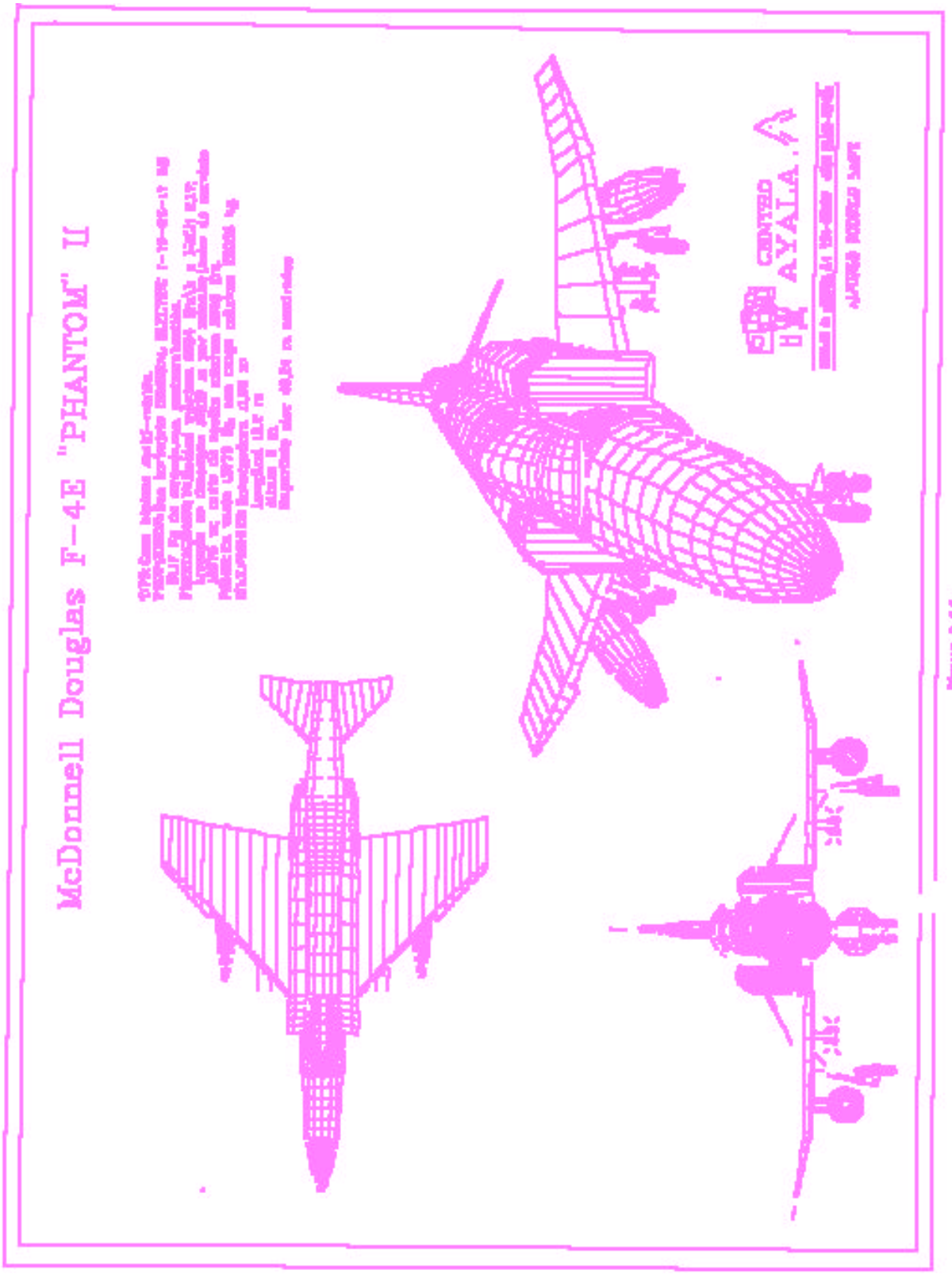
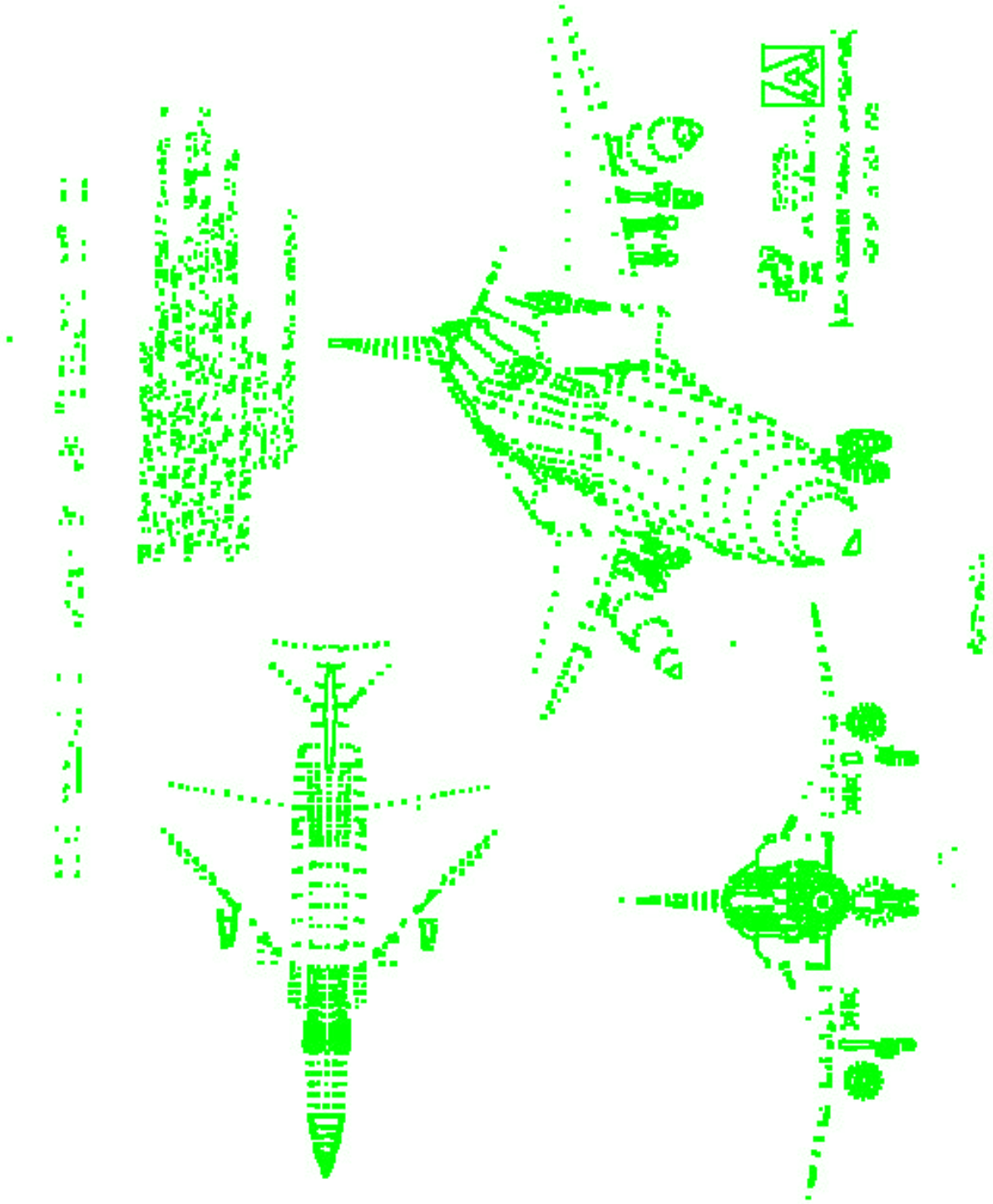


Figura A.11.


McDonnell Douglas F-4E "PHANTOM" II
CENTRO AYALA


McDonnell Douglas F-4E "PHANTOM" II
CENTRO
AYALA



Bibliografía

[1] An improved algorithm for the secuential extraction of boundaries from a raster scan. David W. Capson. Computer Vision, graphics and image procesing 28,109-125 (1984).

[2] Arrowhead recognition during automated data capture. G. Priestnall, R. E. Marston y D. G. Elliman. Pattern recognition letters 17 pags. 277-286 (1996).

[3] Skeleton generation of engineering drawings via contour matching. C. C. Han y K. C. Fan. Pattern recognition vol. 27 nº2 pags. 261-275 (1994).

[4] The theory and measurement of a silhouette descriptor for image pre-procesing and recognition. Paul P. Nahin. Pattern Recognition vol. 6, 85-95 (1974).

[5] Curve-fitting with piecewise parametric cubics. Michael Plass y Maureen Stone. Computer Graphics vol. 17 nº 3 Julio 1983.

[6] A system for interpretation of line drawings. Rangachar Kasturi, Sing T. Bow, Wassim El-Masri, Jayesh Shah, James R. Gattiker y Umesh B. Mokate. IEEE transactions on pattern analysis and machine intelligence vol. 12, nº 10 Octubre 1990.

[7] Detection of dimension sets in engineering drawings. Chang Pyng Lai y Rangachar Kasturi. IEEE transactions on pattern analysis and machine intelligence vol. 16, nº 8 Agosto 1994.

[8] Dimensioning Analysis. Toward automatic understanding of engineering drawings. Dov Dori. Communications of the ACM. Vol. 35 nº 10 Octubre 1992.

[9] Vector-based arc segmentation in the machine drawing understanding system environment. Dov Dori. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 17 nº 11 Noviembre 1995.

[10] Representing pattern recognition-embedded systems through object-process diagrams_the case of the machine drawing understanding system. Dov Dori. Pattern Recognition Letters 16 p. 377-384 (1995).

[11] Line structure extraction from line-drawing images. Toru Kaneko. Pattern Recognition. Vol. 25 nº 9 p. 963-973, 1992.

[12] Implementación de técnicas avanzadas de preproceso de ficheros raster para postidentificación de entidades. Aplicación a la interpretación de planos de ingeniería

digitalizados. Vicente Escuderos Abella. UPV PFC Facultad de Informática de Valencia 1995.

[13] Algoritmo robusto para la discriminación de texto-gráficos en planos de ingenieria. Mª Carmen Juan Lizandra. UPV PFC Facultad de Informática de Valencia 1995.

[14] Contribuciones al desarrollo de un sistema de regeneración de dibujos a mano alzada por ordenador. F. Brusola Simón. Tesis doctoral UPV DEGI (1989).

[15] Decomposing a plane figure into lines and nodes. Silvano Di Zenzo. Pattern Recognition Letters 14. p. 935-943 (1993).

[16] Engineering Drawing processing and vectorization system. Vijay Nagasamy y Noshir A. Langrana. Computer Vision Graphics and Image Processing 49 (1990).

[17] Perfecting vectorized Mechanical drawings. Yuan Chen, Noshir A. Langrana y Atish K. Das. Computer Vision an Image Understanding. Vol. 63 nº 2 p.273-286 Marzo 1996.

[18] Feature identification from vectorized mechanical drawings. Noshir A. Langrana, Yuan Chen y Atish K. Das. Computer Vision and image understanding. Vol. 68 nº 2 p. 127-145 Noviembre 1997.

[19] Recognition and integration of dimension sets in vectorized engineering drawings. Atish K. Das y Noshir A. Langrana. Computer Vision and Image Understanding. Vol. 68 nº 12 p. 90-108 Octubre 1997

[20] Adaptative vectorization of line drawing images. Rik D. T. Janssen y Albert M. Vossepoel. Computer Vision and Image Understanding vol. 65, nº 1 Enero, p. 38-56, 1997.

[21] The role of scanners, automatic data capture, and document storage and distribution. Richard N. Stover. Capitulo 7 de The CAD/CAM handbook. Carl Machover. Pags. 57-65 Ed. McGraw-Hill 1996.

[22] From paper drawings to computer-aided design. Medhat Karima, Kuldip S. Sadhal y Tim O. McNeil. IEEE Computer Graphics and Applications. Vol. 5 nº 2 Febrero 1995.

[23] A systematic evaluation of skeletonization algorithms. Seong-Whan Lee, Louisa Lam y Ching Y. Suen. International Journal of Pattern Recognition and Artificial Intelligence. Vol. 7 nº 5 p. 1203-1225 (1993).

[24] Analytical comparison of thinning algorithms. Y. Y. Zhang y P. S. P. Wang. International Journal of Pattern Recognition and Artificial Intelligence. Vol. 7 nº 5 p. 1227-1246 (1993).

[25] Thinning metodologies – A comprehensive survey. Louisa Lam, Seong-Whan Lee y Ching Y. Suen. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 14 nº 9 p. 869-885 Septiembre 1992.

[26] An X-crossing preserving skeletonization algorithm. Gongzhu Hu y Ze-Nian Li. International Journal of Pattern Recognition and Artificial Intelligence. Vol. 7 nº 5 p. 1031-1053 (1993).

[27] A new parallel thinning methodology. Y. Y. Zhang y P. S. P. Wang. International Journal of Pattern Recognition and Artificial Intelligence. Vol. 7 nº 5 p. 999-1011 (1993).

[28] A fast and economic scan-to-line-conversion algorithm. Gerd Woetzeld. Gesellsschaft für Mathematik und Datenverarbeitung (GMD) 1978 Siggraph Conf. Atlanta.

[29] Syntactic analysis of technical drawing dimensions. Suzanne Collin y Dominique Colnet. International Journal of Pattern Recognition and Artificial Intelligence. p. 1131-1148 (1994).

[30] Geometric constraints and reasoning for geometrical CAD systems. Hiromasa Suzuki, Hidetoshi Ando y Fumihiko Kimura. Computers & Graphics. Vol. 14 nº 2 p. 211-224 (1990).

[31] Evaluation of binarization methods for document images. ∅ivind Due Trier y Torfinn Taxt. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 17 nº 3 Marzo 1995.

[32] Analisis of form images. Dacheng Wang y Sargur N. Srihari. International Journal of Pattern Recogniton and Artificial Intelligence. p. 1031-1051, 1994.

[33] Reconstrucción geométrica tridimensional. José María Gomis Martí y Pedro Company Calleja.

Artículos sobre la transformada de Hough y detección de curvas en imágenes:

[34] Digital image processing. Rafael C. Gonzalez, Paul Wintz. Pags. 130-135 Ed. Addison-Wesley Publishing Company (1987).

[35] Fuzzy Hough transform. Joon H. Han, László T. Kóczy, Timothy Poston. Pattern recognition letters 15 (1994) pags. 649-658.

[36] Modification of Hough transform for circles and ellipses detection using a 2-dimensional array. Raymond K. K. Yip, Peter K. S. Tam, Dennis N. K. Leung. Pattern Recognition Vol 25 nº9 p.1007-1022, 1992.

[37] Parallel algorithm for circle detection in images. Senthil Kumar, N. Ranganathan, Dimitry Goldgof. Pattern Recognition Vol.27 nº 8 p.1019-1028, 1994.

[38] A new curve detection method: randomized Hough transform (RHT). Lei Xu, Erkki Oja, Pekka Kultanen. Pattern Recognition Letters 11 (1990) p.331-338.

[39] A conectionist approach for peak detection in Hough space. V.V. Vinod, Santanu Chaudhury, S. Ghose, J. Mukherjee. Pattern Recognition Vol.25 nº10 p.1253-1264, 1992.

[40] Generalized fuzzy c-shells clustering and detection of circular and elliptical boundaries. Rajesh N. Dave. Pattern Recognition Vol. 25 nº7 p. 713-721, 1992.

Artículos sobre métodos de vectorización automática citados en [20]:

[41] A fast sequential method for polygonal approximation of digitized curves. K. Wall y P. E. Danielsson. Computer Vision Graphics and Image Processing. Vol 28 p. 220-227 (1984).

[42] Curve fitting based on polygonal approximation. K. Wall. Proc. 8th Int. Conf. Patt. Rec., Paris p. 1273-1275, IEEE Comput. Soc. Press (1986).

[43] Digital image processing. R. C. Gonzalez y R.E. Woods. Addison-Wesley, Reading, MA, 1992.

[44] An efficient algorithm for the piecewise linear approximation of planar curves. C. M. Williams. Computer Graphics and Image Processing. Vol. 8 p. 286-293 (1978).

[45] Bounded straight-line approximation of digitized planar curves and lines. C. M. Williams. Computer Graphics and Image Processing. Vol. 16 p. 370-381 (1981).

[46] Fast polygonal approximation of digitized curves. J. Sklansky y V. Gonzalez. Pattern Recognition. Vol. 12 p. 327-331 (1980).

[47] Polygonal approximation by the minimax method. Y. Kurozumi y W. A. Davis. Computer Graphics and Image Processing. Vol. 19 p. 248-264 (1982).

[48] An on-line algorithm for polygonal approximation of digitized plane curves. Proc. 6th Int. Conf. Patt. Rec., Munich, Vol. 2 p. 739-741 Octubre 1982.

[49] Dynamic two-step algorithm in curve fitting. M. K. Leung y Y. H. Yang. Pattern Recognition. Vol. 23(1/2) p. 69-79 (1990).

[50] A hybrid vectorization algorithm. T. Pavlidis Proc. 7th Int. Conf. Patt. Rec. Montreal p. 490-492 (1984).

[51] A vectorizer and feature extractor for document recognition. T. Pavlidis. Computer Vision Graphics and Image Processing. Vol. 35 p. 111-127 (1986).

[52] An algorithm for polygonal approximation of a digital object. A. Sirjani y G. R. Cross. Pattern Recognition Letters. Vol. 7(5) p. 299-303 (1988).

[53] On the detection of dominant points on digital curves. C. H. Teh y R. T. Chin. IEEE Trans. Pattern Anal. Machine Intell. 11(8) p. 859-872 (1989).

[54] Detection of significant points and polygonal approximation of digitized curves. B. K. Ray y K. S. Ray. Pattern Recognition Letters. Vol.13(6) p. 443-452 (1992).

[55] Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. D. H. Douglas y T. K. Peucker. Can. Cartographer 10(2) p. 112-122 (1973).

[56] Polygonal approximation of 2-D shapes through boundary merging. J. G. Leu y L. Chen. Pattern Recognition Letters7(4) p. 231-238 (1988).

[57] An iterative procedure for the polygonal approximation of plane curves. U. Ramer. Computer Graphics and Image Processing. Vol. 1 p. 244-256 (1972).

Indice

[i]NULO: no posee herramientas de limpieza. BAJO: sólo dispone de herramientas de limpieza de manchas. ALTO: además de limpieza de manchas tiene herramientas para cerrar huecos en la imagen, adelgazamiento de líneas, suavizado de contornos, etc.
[ii]NULO: no posee herramientas de edición raster. BAJO: posee herramientas de zoom, selección y dibujo ALTO: posee las anteriores más herramientas de transformación del raster (giros, escalados, recortado, deformaciones, ...).
[iii]NULO: no reconoce entidades. BAJO: reconoce arcos, rectas y círculos. MEDIO: reconoce las anteriores más texto. ALTO: reconoce los anteriores además de otras entidades como grosores y tipos de línea, rayados, puntas de flecha, etc.
[iv]No dispone de herramientas de limpieza, posee de numerosos filtros para tratamiento de imágenes en color y escala de grises.
[v]Discrimina y reconoce textos con un OCR.
[vi]Sólo discrimina textos, no dispone de OCR.
[7]Múltiples imágenes indica la posibilidad de trabajar con varias imágenes simultáneamente en el entorno de trabajo.