UNIVERSIDAD DE MURCIA

FACULTAD DE INFORMÁTICA
SISTEMAS Y
PROYECTOS
INFORMÁTICOS

Visión de alto nivel:

Interpretación de dibujos de líneas mediante comparación de grafos

---

Autor:
Ginés García Mateos

Profesor tutor:
Alberto Ruiz García

INDICE

Resumen del proyecto

En este proyecto se estudia y desarrolla de forma completa un sistema de visión, enfocado hacia la interpretación de escenas tridimensionales, dentro de un entorno controlado. A partir de las imágenes de entrada, serán extraídas las líneas correspondientes a los bordes de los objetos en la escena, dando lugar a las descripciones conocidas como dibujos de líneas. La característica fundamental del acercamiento propuesto es la interpretación a través de una comparación de grafos, que contendrán las relaciones más relevantes entre las líneas de borde de la escena.

Se ha considerado de manera especial la integración del proceso de interpretación, propiamente dicho, con todos los niveles previos de procesamiento, cuyo objetivo es la obtención del dibujo de líneas asociado a la escena. En concreto, el sistema de visión está compuesto por tres etapas (bajo, medio y alto nivel), actuando de forma coordinada a un nivel simbólico creciente.

El propósito de la etapa de bajo nivel es encontrar una descripción de la imagen original en función de los bordes de intensidad. Estos bordes dividen la imagen en zonas cuyos lugares correspondientes en la escena se espera que sean superficies con características de profundidad homogéneas (es decir, las caras de los objetos). La aplicación de esta etapa es relativamente sencilla y automática, obteniéndose en la mayoría de los casos resultados bastante buenos. Con esto se consigue que el proceso sea invariante a color, intensidad y a las características de iluminación de la escena. Por contra, el proceso será muy sensible en caso de existir de texturas muy destacadas, marcas, dibujos o pliegues suaves en los objetos.

El dibujo de bordes será recibido por el proceso de medio nivel, en el cual se realizará una agrupación de los puntos de borde en segmentos de líneas rectas. Idealmente el dibujo de líneas de la escena contiene la misma información que el dibujo de bordes pero a un mayor nivel descriptivo. Los puntos individuales dejan de ser importantes, fijándonos en las agrupaciones de los mismos en estructuras lineales. Puesto que los bordes delimitan las zonas con propiedades homogéneas, una línea de borde describirá una arista de alguno de los objetos en la escena y una unión entre líneas corresponderá a un vértice del objeto. El algoritmo de detección de segmentos utilizado, el algoritmo EMH, es capaz de obtener buenos resultados en un tiempo razonable. Además dispone de cierta dinámica que le permite adaptarse rápidamente a pequeñas variaciones de la imagen de entrada. El proceso de detección de segmentos impone una fuerte restricción sobre el dominio del problema: sólo serán admitidos objetos poliédricos, es decir formados por caras planas.

Por último, en la etapa de alto nivel se lleva a cabo el proceso de interpretación propiamente dicho. Como un paso intermedio entre el medio y el alto nivel, se transformará la descripción de la escena una estructura de grafo, en la que se extraen las propiedades relevantes del dibujo de líneas. Después se realizará el paso central de la interpretación: la comparación de grafos, entre el grafo asociado al dibujo de líneas de la escena y los grafos que representan modelos. El resultado será una hipótesis de interpretación, en la que se describirán los objetos reconocidos en la escena y las posiciones en las que se encuentran. Esta hipótesis deberá ser verificada de una manera geométricamente más precisa, mediante una técnica de alineamiento, con la cual es posible obtener la posición y orientación tridimensional de los objetos en la escena.

En la estructura de grafos utilizada, los nodos representan los segmentos del dibujo de líneas y los arcos relacionan los segmentos unidos en el dibujo. De esta forma, el grafo es un reflejo exacto de la disposición de las líneas en el dibujo. La creación de los grafos a partir de los dibujos de líneas consistirá en un proceso de búsqueda de relaciones no accidentales entre los segmentos. En este paso, algunos errores de los procesos anteriores podrán ser eliminados, quedándonos con la información relevante y no redundante, más adecuada para la posterior comparación.

La comparación de grafos ha sido un problema ampliamente estudiado, debido a su utilidad potencial, pero con una complejidad computacional de orden exponencial. Para resolverlo se utiliza un algoritmo desarrollado recientemente, que enfoca la comparación a través del conocido proble-

ma de asignación. Este algoritmo dispone de un bajo orden de complejidad, obteniendo en muchos casos resultados óptimos o próximos al óptimo. Además dispone de un parámetro que permite controlar la velocidad de ejecución.

El algoritmo trabaja con unos costes de comparación básicos, dependientes del problema tratado. Para el caso de los grafos asociados a los dibujos de líneas, ha sido realizado en este proyecto un estudio exhaustivo de cuales deberían ser estos costes. Se llega a la conclusión de que un nodo está definido en función de sus relaciones con otros nodos, siendo el coste básico de la comparación de grafos el coste de comparación entre tipos de uniones de segmentos.

Finalmente, se ve la necesidad de comprobar la hipótesis de interpretación, obtenida en la comparación de grafos, considerando las propiedades tridimensionales de los objetos de forma más precisa. Esto se realiza con los conocidos métodos de alineamiento, cuya efecto es similar a la proyección de un modelo 3D en un dibujo 2D. La comparación a bajo nivel de la imagen alineada con la original permitirá decidir si la escena observada puede ser considerada como una instancia del modelo correspondiente.

El sistema de visión estudiado a lo largo este proyecto ha sido implementado, utilizando una herramienta de programación visual orientada a objetos. Después se ha puesto a prueba, usando un conjunto de más de 70 imágenes adquiridas con una cámara, sobre un conjunto de 7 modelos de poliedros. El resultado ha demostrado que el acercamiento propuesto es capaz de obtener buenos resultados dentro de este reducido dominio. Se ha comprobado también, que la integración de las distintas etapas no sólo es posible, sino que además mejora el sistema en cuanto a la modularidad y la capacidad de los procesos de actuar de forma paralela.

Un análisis más extenso de los resultados y sus implicaciones es realizado en las conclusiones. Se presenta también una visión más global de este acercamiento, en cuanto a su posible aplicación práctica y el modo en que se deberían afrontar las posibles mejoras futuras.

1. Introducción. Motivación del proyecto.

1.1. La Visión Artificial.

Sin duda alguna, la visión es la actividad dominante del cerebro humano. La información visual constituye la mayoría de los 10^9 bits por segundo de información que se estima que recibe del sistema sensorial. La visión artificial tiene el ambicioso objetivo de conseguir emular la compleja capacidad humana para reconocer e interpretar escenas cualquiera que sea su naturaleza.

La meta final de la investigación en visión artificial es conseguir sistemas visuales genéricos, capaces de reconocer e interpretar escenas a partir de información óptica independientemente del dominio. Actualmente, los sistemas que usan la visión por computador son aplicaciones altamente especializadas en dominios particulares, que usan conocimiento de la tarea específica que se desea resolver. Esto ha limitado en gran medida el uso de los sistemas de visión, ya que cada nueva aplicación requiere una enorme cantidad de desarrollo.

Las aplicaciones potenciales de la visión artificial abarcan un amplio rango de campos, que aumentará con la mejora en el rendimiento de los sistemas construidos. Entre las aplicaciones actuales podemos destacar:

- Sistemas de procesamiento digital de imágenes, para restauración y mejora.
- Reconocimiento óptico de caracteres (OCR- Optical Character Recognition).
- Procesamiento automatizado de huellas dactilares.
- Navegación de robots en entornos controlados.
- Procesamiento de imágenes aéreas o de satélite.
- Sistemas de inspección de circuitos impresos.
- Calibrado y clasificación de la producción en función del color, textura y tamaño (por ejemplo, en producciones agrícolas).
- Máquinas de visión industriales, para ensamblaje e inspección de la producción.

A pesar de la disparidad de propósitos de estas aplicaciones, podemos encontrar un esquema básico de arquitectura, común a todos los sistemas de visión por computador, reflejado en la Figura 1.

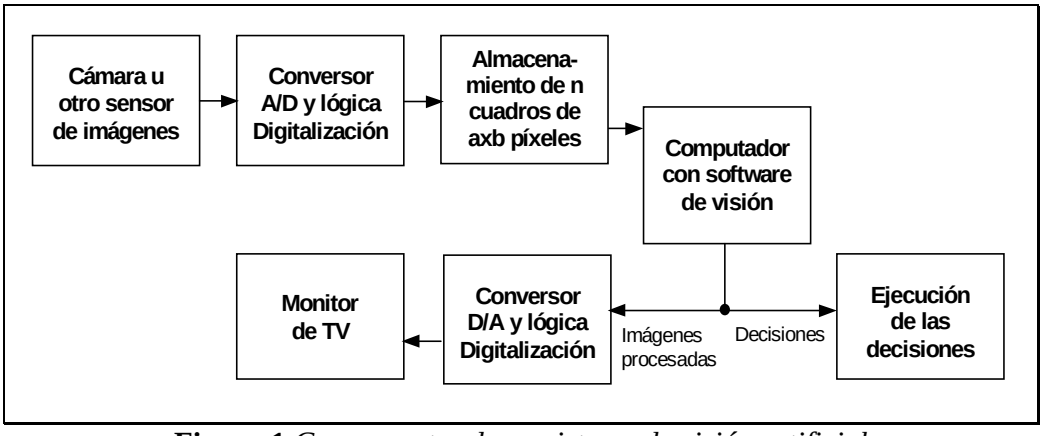

Figura 1 *Componentes de un sistema de visión artificial*

El núcleo de la arquitectura básica es un ordenador que, mediante la ejecución del software de visión, lleva a cabo los procesos de transformación para obtener a partir de la imagen, o secuencia de imágenes, alguna información útil. La adquisición de estas imágenes es realizada por un conjunto de elementos hardware, entre los que típicamente encontramos cámaras y tarjetas de interface. La información de salida puede referirse a decisiones que se deben adoptar (por ejemplo, para el control de algún proceso) o imágenes procesadas para ser mostradas a los usuarios.

Nuestro interés se centra en este núcleo informatizado del sistema, en las técnicas de procesamiento que nos permiten obtener una información útil y precisa a partir de la enorme, pero a menudo superflua, cantidad de información proporcionada por las imágenes. Más concretamente, las entradas utilizadas serán imágenes reales de objetos tridimensionales y las salidas serán interpretaciones acerca del tipo y posición de los objetos presentes en esas imágenes.

1.2. Etapas en el proceso de visión.

Normalmente, el conjunto de procesos involucrados en la percepción visual es descompuesto como una jerarquía que va desde el bajo nivel, siguiendo por el medio nivel hasta la visión de alto nivel. No existe una definición precisa y aceptada de los límites entre los distintos niveles, pero existe un gran consenso en cuanto a la división entre técnicas preliminares de procesamiento de imagen a nivel de píxel y técnicas que implican reconocimiento e interpretación.

En las etapas de bajo nivel encontramos un amplio conjunto de técnicas bien establecidas, para el filtrado, restauración y mejora de las imágenes. Tiene sentido utilizarlas por sí mismas y en algunos casos se implementan mediante hardware. Para los propósitos de este proyecto estamos interesados en las técnicas de suavizado y detección de bordes.

En el medio nivel se localizan técnicas que, sin llegar a ser interpretativas, utilizan las salidas del bajo nivel para conseguir descripciones de la imagen a un mayor nivel que simples píxeles. Las técnicas de segmentación tratan de determinar las distintas zonas de la imagen, teniendo en cuenta la uniformidad de color, textura, formas. Las de detección de líneas buscan la agrupación de los puntos de bordes en líneas u otras formas paramétricas predefinidas.

Las técnicas de alto nivel se encuentran más próximas al dominio de la inteligencia artificial, aunque dadas las características del problema deben desarrollar sus propios procedimientos y heurísticas. Existen varios acercamientos alternativos pero ninguno aplicable a todos los dominios, de forma genérica. Dentro de este nivel, los problemas están relacionados principalmente con el reconocimiento de patrones y la interpretación de escenas, infiriendo información acerca de los objetos presentes en la imagen. El estudio e implementación de una técnica dentro de este área constituye el principal objetivo de este proyecto.

Juntando todo lo anterior, podemos obtener una estructura global de las etapas de procesamiento en un sistema de visión genérico, que es mostrada en la Figura 2. Este esquema supone una ejecución secuencial de las etapas, desde las de bajo nivel hasta las de alto nivel, siendo la salida de cada una la entrada de la siguiente. Aparece un elemento nuevo, la base de conocimiento, en la que se encapsulan las particularidades de cada dominio concreto.

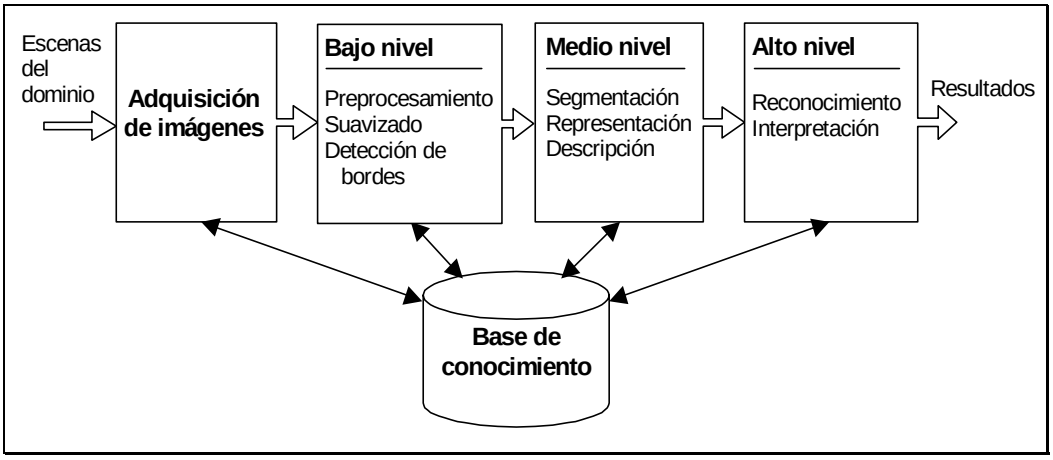

Figura 2 *Estructura de procesamiento de un sistema de visión genérico*

Algunos autores consideran que los niveles no deben ser interpretados como etapas de ejecución secuencial, sino que son técnicas con propósitos distintos que pueden ser o no necesarias

según la aplicación concreta. Por ejemplo Shimon Ullman, [Ull96], propone que los distintos procesos puedan actuar de forma paralela, existiendo interacciones entre etapas en ambos sentidos. De esta manera la etapa de alto nivel no depende completamente de las etapas anteriores ni requiere de su finalización para ejecutarse.

1.3. La visión de alto nivel.

Como hemos comentado, los procesos de bajo y medio nivel utilizan técnicas basadas en un conjunto razonablemente bien definido de formulaciones teóricas. Por contra, los principios de la visión de alto nivel son menos precisos y más especulativos, fundamentalmente en lo que se refiere a la interpretación. En consecuencia, es un problema aun muy abierto y que requiere cierta labor de investigación.

Un dominio particular dentro del problema de reconocimiento de formas, es el referido al reconocimiento de objetos tridimensionales. La mayoría de las aplicaciones prácticas actuales de la visión artificial (casi todas las enumeradas anteriormente) tratan con imágenes bidimensionales procedentes de un dominio bidimensional. En el entorno de la visión tridimensional se debe trabajar con imágenes 2D procedentes de un mundo 3D. Este hecho da lugar a la complejidad inherente al problema, no sólo por la ambigüedad existente al reducir la dimensión, sino por la enorme variación que puede causar en las imágenes la modificación de la posición del observador. Esto se une a los efectos provocados por los cambios de iluminación y las características de los objetos.

Independientemente de que se trate de un dominio 2D o 3D, podemos encontrar básicamente tres acercamientos al problema del reconocimiento de objetos: los métodos de propiedades invariantes, los métodos de descomposición en partes y los métodos de alineamiento. Esta es realmente una clasificación de las ideas subyacentes, más que de los esquemas utilizados, ya que en general se pueden utilizar conceptos de más de un acercamiento.

El acercamiento de propiedades invariantes presupone que cualquier imagen de un objeto tendrá siempre un conjunto de características que permanecerán inalteradas, o sobre un rango de valores definido. A partir de aquí surge la idea del “espacio de características”, donde cada dimensión del espacio es una característica medida y cada vista de un objeto se corresponderá con un punto del espacio. De esta manera, el problema del reconocimiento es descompuesto en el problema de calcular los valores de las características y el problema de encontrar las regiones del espacio correspondientes a cada clase.

La idea de los métodos de descomposición es más intuitiva. Cualquier objeto se puede descomponer en un conjunto de partes constituyentes, o componentes genéricos, y estos componentes se mantienen de una vista a otra. La descripción del objeto a través del conjunto de sus partes es una descripción estructural, en la que se pueden utilizar grafos, redes semánticas o lenguajes formales.

En los métodos de alineamiento, se parte de un conjunto de modelos de objetos y se realiza un proceso de compensación de las transformaciones tridimensionales implicadas en los objetos de la escena que es vista. Una cuestión fundamental es determinar los puntos característicos de la escena que se corresponden con puntos de los modelos, en base a los cuales se hace el alineamiento.

1.4. Interpretación de dibujos de líneas.

La interpretación de dibujos de líneas es un problema concreto dentro del ámbito de la visión de alto nivel, cuya característica fundamental es el tipo de entradas que recibe el proceso de reconocimiento, esto es, la descripción de la escena que es utilizada. En concreto, la representación está basada en un conjunto de líneas que se desprenden de la escena observada. Normalmente estas líneas son líneas de bordes, es decir describen los contornos más destacados de la imagen, produci-

dos por un cambio de intensidad. El problema de encontrar estas líneas de bordes cae en el dominio del bajo y medio nivel.

Los métodos más simples de reconocimiento en el dominio 2D, suelen utilizar como entrada la salida de algún proceso de bajo nivel, cuyo resultado puede ser, por ejemplo, una imagen modificada (aunque aun descrita por píxeles) o un conjunto de coeficientes transformados (por ejemplo, de Fourier). Sin embargo, en la visión tridimensional este acercamiento tan directo puede resultar inadecuado debido, como hemos comentado, a las enormes variaciones que se producen en las imágenes al modificar el punto de vista del observador. Por lo tanto, se hace necesario usar descripciones de mayor nivel, una de las cuales es la de las líneas de borde de la imagen.

En las técnicas de interpretación de dibujos de líneas toman especial relevancia las relaciones espaciales entre las líneas, más que las propias características de las líneas de forma individual (como longitud o pendiente). Por esta razón, la descripción inicial del conjunto de líneas suele ser transformada en una representación mediante grafos, denotando relaciones entre las líneas. El objetivo es buscar las relaciones que permanecen inalteradas en un amplio rango de variaciones de las condiciones de observación de la escena, resultando más adecuadas para la interpretación.

De esta forma, dado un modelo y una escena observada, el reconocimiento estará basada en un proceso de comparación de grafos. Esta comparación de grafos es un problema NP-completo, de complejidad exponencial. Además debería ser aplicable en los casos de búsqueda de un subgrafo dentro de un grafo (por ejemplo, si hay varios objetos en la escena) y tener tolerancia a cierto nivel de ruido (por líneas faltantes o sobrantes).

Aun teniendo resuelto lo anterior, el proceso de comparación de grafos no es suficiente en sí mismo. Muchos objetos distintos pueden tener grafos isomorfos, es decir pueden tener relaciones idénticas entre sus líneas de bordes. En estos casos será necesario aplicar también otras técnicas geométricamente más precisas, como las de alineamiento.

Aunque este proyecto se centra en el desarrollo de una técnica de interpretación de dibujos de líneas, se considerará en todo momento su aplicabilidad e integración con todas las etapas de procesamiento anteriores. Es decir, las entradas al proceso de alto nivel serán lo que realmente los procesos de medio y bajo nivel nos puedan ofrecer en situaciones reales.

2. Objetivos del proyecto.

El objetivo final de este proyecto es el diseño e implementación de un sistema de visión artificial completo, capaz de interpretar escenas reales tridimensionales dentro de un entorno controlado, a partir de imágenes del mismo, en un tiempo razonablemente pequeño, integrando todas las etapas de procesamiento, y haciendo especial hincapié en las técnicas de alto nivel de interpretación de dibujos de líneas.

Por interpretación entendemos la capacidad del sistema de inferir información acerca de los objetos presentes en la escena, así como de su posición y orientación. Esta información será la salida última del sistema.

Para la entrada al proceso hay un claro propósito de partida en cuanto a la utilización de imágenes adquiridas de escenas reales. Aunque las imágenes sintéticas pueden considerarse una cierta aproximación a casos reales, a menudo ocultan los verdaderos problemas que se pueden dar al usar entradas del mundo exterior y, lo que es peor, las técnicas desarrolladas pueden no tener mucha aplicación práctica. Este propósito llega un poco más lejos. Estamos interesados especialmente en utilizar un sistema de adquisición de alta disponibilidad, debido a su bajo coste, en nuestro caso una cámara QuickCam, como un punto más en favor de la utilidad práctica del sistema.

Los modelos utilizados para las escenas a reconocer serán un conjunto de objetos tridimensionales con caras planas, es decir poliedros. Las escenas se encuentran dentro de un entorno controlado, donde podemos ajustar los valores de iluminación, fondo y la oclusión de figuras.

Un requisito importante es el tiempo de respuesta del proceso de forma global. Idealmente el sistema deberá ser capaz de hacer un seguimiento de los objetos en tiempo real. Este tiempo depende de todos los procesos, empezando por la frecuencia de refresco de la cámara, implicando fuertes requerimientos de eficiencia en todos los niveles.

Para la construcción del sistema no se parte desde cero, sino que se dispone de las implementaciones de los procesos relacionados con la adquisición, la detección de bordes y la detección de líneas, aunque en módulos independientes. Por lo tanto, será necesario construir el proceso de alto nivel y realizar un diseño de arquitectura que integre todas las etapas de procesamiento, desde la adquisición hasta el alto nivel. En la implementación de los programas, se tendrán en cuenta especialmente los criterios de calidad del software. Estos criterios hacen referencia a aspectos bien conocidos como eficiencia, reutilización, extensibilidad, legibilidad, robustez, documentación.

A parte de las cuestiones de integración, el principal objetivo de desarrollo del proyecto se centra en los procesos de alto nivel. Más específicamente, se desarrollará una técnica de interpretación de dibujos de líneas, que serán el tipo de entradas procedentes del nivel anterior. Esto supone el estudio de las técnicas existentes y cómo se adaptan al conjunto de forma global que, recordemos, partirá de imágenes reales.

3. Desarrollo del proyecto.

3.1. Estructura global del proceso.

Debido a la complejidad de los problemas que se desean resolver, la ejecución de un sistema de visión artificial es vista como una estructura de procesos interactuando, más que como la simple ejecución secuencial de un algoritmo. En esta estructura se describe cuales son, cómo se disponen y cómo se comunican todos los elementos que constituyen el proceso completo, desde la adquisición hasta la interpretación en el alto nivel.

Por lo tanto, el análisis y establecimiento de esta estructura global del proceso, para nuestro caso concreto, es el primer paso antes de entrar de lleno en el desarrollo del proyecto. Para llevarlo a cabo se deben tener en cuenta las características más importantes del dominio del problema tratado, que han sido ya comentadas en la introducción. A continuación enumeraremos las que consideramos más influyentes para el diseño de la estructura global:

- Las imágenes de entrada proceden de un entorno tridimensional formado por objetos poligonales. Esto conlleva la necesidad de utilizar un proceso capaz de manejar las grandes variaciones que se pueden dar de una vista a otra de un mismo objeto. Además, se hace factible una descripción de las imágenes mediante líneas.
- Dentro del dominio tridimensional se busca una estructura lo más genérica posible, modular y capaz de adaptarse y extenderse según las particularidades de las aplicaciones prácticas que pudieran realizarse.
- Puesto que las imágenes utilizadas son reales, adquiridas del mundo exterior, se debe considerar la existencia de ruido y la posible falta de calidad en las imágenes. Este problema debe ser tratado en todas las etapas, ya que en ninguna podemos estar seguros de haberlo eliminado completamente. Es más, es posible que alguna etapa introduzca cierto ruido, lo cual debe ser evitado.
- El requisito del tiempo de respuesta del sistema implica que los procesos encadenados deben ser lo suficientemente rápidos como para no bloquear la ejecución global en ciertos puntos críticos. Los procesos deben estar muy bien acoplados entre sí para no provocar un retardo adicional. En la situación ideal, los procesos podrían actuar de forma paralela, necesitando comunicación con los otros sólo cuando se requiera determinada información.

En la introducción vimos la estructura de procesamiento de un sistema de visión genérico. Esta es la estructura más general posible, ya que integra todos los niveles de procesamiento existentes. En líneas generales, este enfoque se adapta bastante bien a nuestro dominio, ya que resulta posible y adecuada la aplicación de técnicas en todos los niveles de abstracción. Será necesario concretar las técnicas aplicadas en cada nivel, así como cuestiones de representación y control, que establecerán el modo en que se integran los distintos procesos.

La estructura de procesamiento desarrollada en este proyecto es mostrada en la Figura 3. Aparece una clara descomposición del proceso global en cuatro etapas: adquisición, bajo, medio y alto nivel, capaces de actuar por sí mismas. Cada etapa supone un nivel de abstracción superior respecto a la anterior, en cuanto a la descripción de las imágenes, repartiendo la complejidad del problema de forma precisa y equitativa. Esto es una ventaja frente a métodos más directos, donde básicamente toda la complejidad se encuentra en el alto nivel. Sin embargo, hay una clara desventaja debida al relativamente largo encadenamiento en los resultados de los procesos.

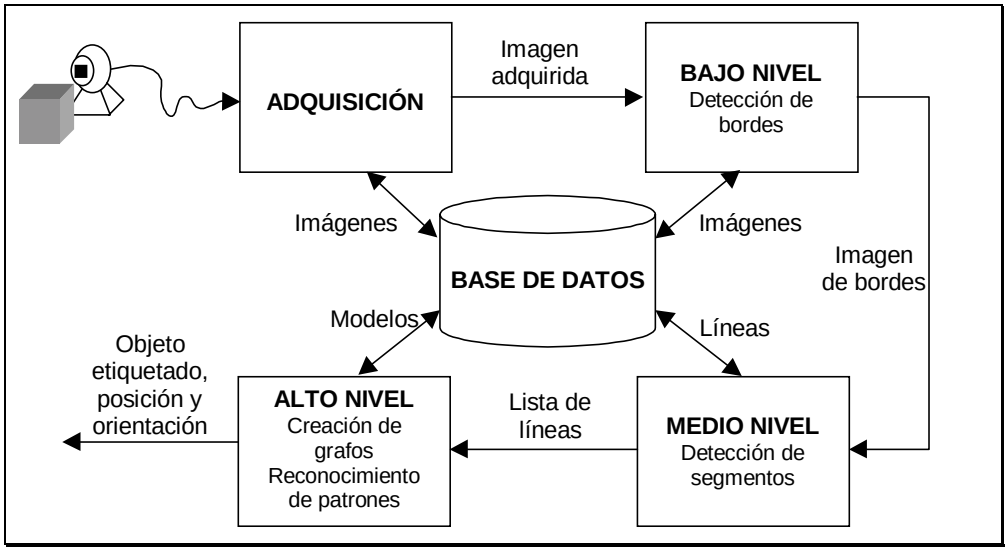

Figura 3 *Estructura global de procesos del sistema*

Las flechas entre los procesos representan flujos de información, más que flujos de control. Todos los procesos tienen la capacidad de actuar de forma paralela, requiriendo, a lo sumo, recibir al comienzo los datos iniciales y ser actualizados cada cierto tiempo con nuevos datos de entrada. Con ello se consigue mejorar la respuesta del sistema, no sólo por la potencial aceleración con una implementación paralela, sino porque se le dota de cierta dinámica, por ejemplo frente a una posible secuencia continua de imágenes de entrada. Aun con una sola imagen, el proceso de alto nivel puede empezar a obtener resultados sin necesitar que todos los procesos anteriores hayan acabado su ejecución completamente.

A continuación explicaremos brevemente las etapas que constituyen el esquema global, que serán descritas más ampliamente en los siguientes apartados del desarrollo.

La primera etapa, común a cualquier sistema de visión, es la de adquisición u obtención de imágenes del exterior. Los resultados de la adquisición pueden condicionar las etapas siguientes, aunque estas deberían permitir cierta falta de calidad o definición en las entradas. En esta etapa, la escena es descrita como un mapa de *axb* píxeles. Para este proyecto, se ha considerado el uso de los dos siguientes sistemas de adquisición:

- **QuickCam**, de Connectix, en blanco y negro, con máxima resolución de 64 niveles de gris con imágenes de hasta 320x240 píxeles.
- **Meteor**, de Matrox, también en blanco y negro, con mayores tamaños de imagen y resoluciones de hasta 256 niveles de gris.

En la etapa de bajo nivel se aplicarán técnicas de detección de bordes. Estas parten de las imágenes adquiridas y su función es encontrar los puntos donde existe un borde, en nuestro caso debido una discontinuidad en la intensidad de la imagen. El borde, en conjunto, representa un determinado cambio en los puntos correspondientes de la escena observada, y es de esperar que este cambio refleje alguna variación significativa dentro de la estructura tridimensional del objeto. El borde será descrito como una lista de puntos que lo forman. Para esta etapa, entre otras técnicas de detección de bordes, se cuenta con una implementación del algoritmo de Canny que, como veremos, es uno de los que ofrecen mejores resultados.

La imagen de bordes es la entrada para el proceso de medio nivel. Su objetivo será agregar los puntos de bordes en estructuras paramétricas conocidas, en concreto segmentos de líneas rectas. La imagen es representada como una lista de segmentos, que básicamente describen la misma información que la imagen de bordes, pero a un mayor nivel de abstracción, ya que los píxeles individuales dejan de ser importantes. Existen varias técnicas de detección de segmentos, en este proyecto se utilizará el algoritmo EMH.

La última etapa será la de alto nivel, en la cual se realiza la interpretación de los resultados obtenidos hasta este punto, es decir del dibujo de líneas. Esta parte centra la labor de estudio y desarrollo de este proyecto. En el método que se propone, la lista de líneas recibida como entrada es transformada en una estructura de grafo que describe las relaciones espaciales no accidentales entre las líneas. Este grafo será comparado con una serie de grafos que representan modelos, dando como resultado el modelo que mejor se ajusta y un emparejamiento entre escena y modelo. Por último, se aplicará un alineamiento para verificar la hipótesis del modelo.

En la siguiente figura se muestra un ejemplo de la salida de las distintas etapas para una escena muy simple, en la que se pretende mostrar gráficamente el tipo de tarea que debe resolver cada técnica, así como las entradas y salidas con las que actúa.

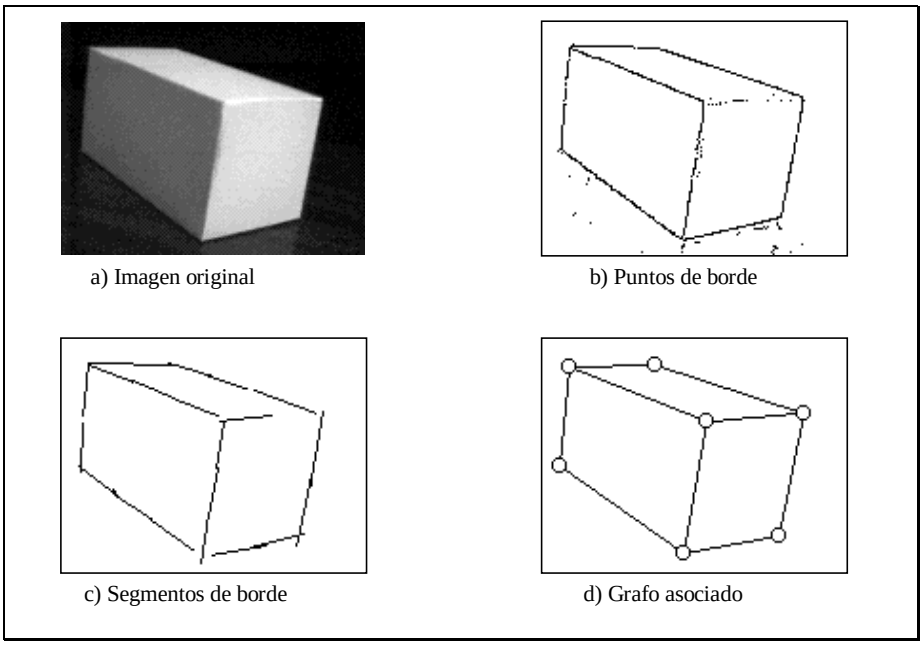

Figura 4 *Ejemplo de salidas de los procesos*

Tras la aplicación de la detección de bordes (b), se reduce enormemente la información de la imagen original (a), siendo importantes sólo los lugares en que hay variaciones de la intensidad. Estos puntos son agregados en segmentos que los contienen, dando lugar a una mejor descripción como una lista de segmentos (c), que elimina cierto ruido y también el volumen de la información (en este caso se pasa de 623 puntos de borde a 20 líneas). En el alto nivel se utiliza conocimiento específico del problema para hacer ciertas suposiciones, que nos indiquen qué relaciones entre los segmentos son relevantes y cómo deben ser descritas. Esto da lugar a una representación de grafo (d), que contiene 9 segmentos y relaciones de adyacencia entre ellos.

3.2. La etapa de bajo nivel: Detección de bordes.

3.2.1. Introducción. Motivación para el uso de imágenes de bordes.

Por procesamiento de imágenes a bajo nivel entendemos un conjunto de técnicas de preprocesamiento cuyo objetivo es la extracción de descripciones de la imagen, que a menudo son también imágenes. Pueden ser vistas como procesos automáticos, ya que no requieren ninguna capacidad inteligente por parte del sistema, en el sentido de que no se necesita conocer nada de los objetos presentes en la escena o la posición relativa del observador. El marco teórico de estas técnicas viene dado por la teoría de señales. Las técnicas de procesamiento son tratadas como sistemas lineales invariantes a desplazamientos, calculados como convoluciones discretas que son aplicadas por igual a todos los puntos de una imagen.

En muchas ocasiones, estas técnicas son aplicadas sin otro propósito que el de mejorar la calidad de la imagen de cara al usuario. Esto ocurre con las técnicas de eliminación de ruido, de mejora del contraste o de desemborronado. En este proyecto estamos interesados en las técnicas de bajo nivel para la obtención de dibujos de bordes, como un paso previo a la obtención del dibujo de líneas de borde la escena.

Se dice que una región de la imagen es una zona de borde si existe una variación en alguna de las características de la imagen. En el caso más sencillo, la detección de bordes se basa en la búsqueda de un cambio brusco en la intensidad (o luminosidad) de la imagen. Esto es lo que se conoce como un borde de intensidad. Aunque no sería descartable el uso de otras características más complejas, como el color, la textura o la forma, evidentemente en estos casos el bajo nivel se convertiría en una etapa de elevada complejidad, como para ser considerada como un sencillo proceso automático. En este proyecto se usará la detección de bordes de intensidad.

Algunos experimentos realizados con el sistema visual humano, han demostrado que la información de bordes es muy importante para el reconocimiento de objetos. La capacidad de los humanos de interpretar escenas a partir de simples dibujos de bordes, en base a la distinción de regiones con características homogéneas, parece ser uno de los aspectos determinantes en el proceso visual biológico. Esta es la principal motivación para la utilización de representaciones de bordes en los sistemas de visión artificial. No obstante, hay que tener en cuenta que en los humanos la detección de bordes se basa en características más complejas, como la textura y la profundidad tridimensional. En general, la aparición de un borde de intensidad puede ser debida a varias causas físicas, algunas de las cuales son mostradas en la siguiente figura.

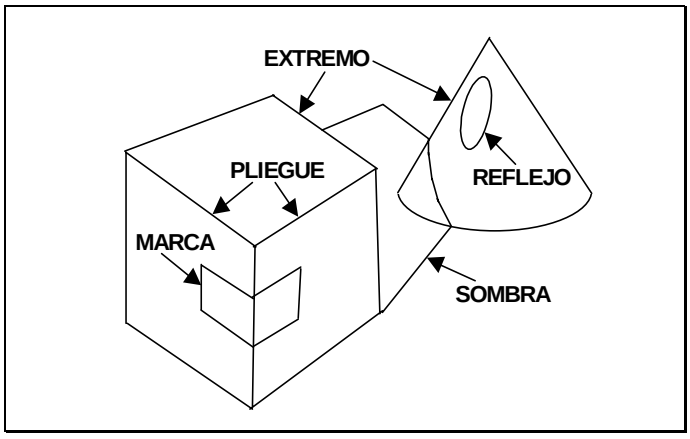

Figura 5 *Tipos de orígenes de los bordes*

En el contexto de la interpretación de escenas, estamos interesados en los bordes debidos a pliegues de los objetos o a los extremos donde estos acaban. En ambos casos, el cambio de intensidad indica una cambio brusco en la profundidad del punto de la escena correspondiente, que es interpretado como un cambio en la estructura 3D de los objetos.

Sin embargo, el proceso de detección de bordes de intensidad también detectaría las otras causas de borde: las marcas o dibujos en los objetos, las sombras y las zonas de reflejos. Además, hay que tener en cuenta que los pliegues en los objetos pueden ser más o menos agudos. Un pliegue agudo facilitará su detección, pero en el peor caso un pliegue suave permanecerá indetectable para el proceso de búsqueda de bordes. Puesto que la etapa de bajo nivel no tiene ningún conocimiento específico sobre la escena, estos errores deberían ser resueltos en las etapas posteriores. En la práctica, esto supone una seria limitación para la utilización de dibujos de borde en entornos no controlados.

3.2.2. Métodos básicos de detección de bordes.

La forma más sencilla de realizar la detección de bordes consiste en la utilización de operadores de gradiente. Estos operadores están formados por máscaras de convolución (normalmente

dos), que aplicadas sobre cada pixel y combinadas de manera adecuada, devuelven un valor de gradiente, indicando el grado de variación en ese punto. Después podemos elegir un umbral mínimo y considerar como borde todos los puntos que superen ese valor de variación.

Las máscaras de convolución utilizadas son máscaras diferenciales, es decir realizan una diferencia de los valores en un sentido menos los valores en el sentido opuesto. Estos valores son los niveles de gris de los píxeles correspondientes. Existen muchos tipos de operadores de bordes, y la mayoría de ellos se pueden adaptar a distintos tamaños de máscara. Algunos de los más usados son mostrados en la figura 6.

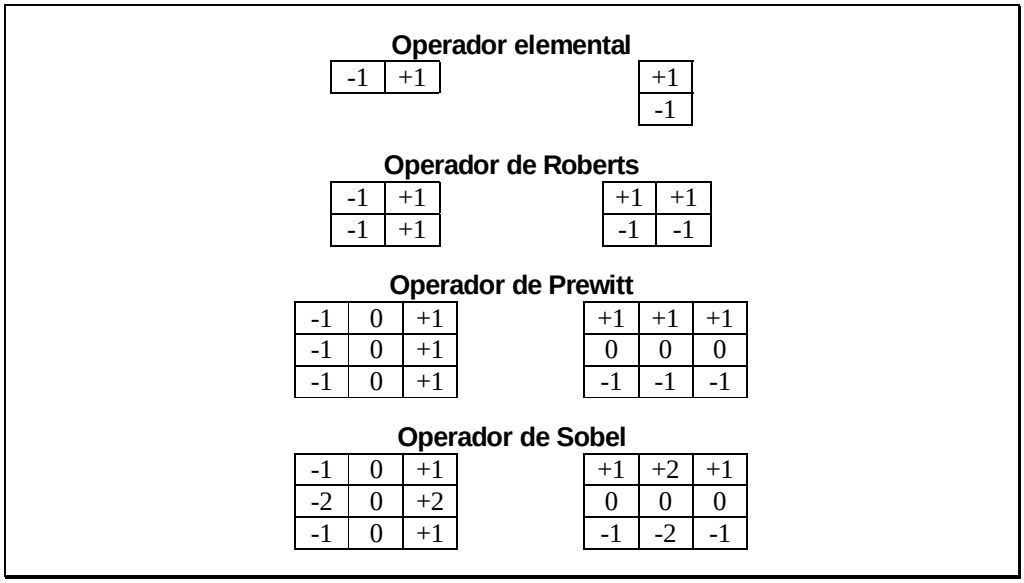

Figura 6 *Máscaras de convolución de los operadores de borde básicos*

Con la aplicación de cada máscara, obtenemos el gradiente en las dos direcciones principales, ∇_x y ∇_y. El módulo del gradiente en cada punto vendrá dado por:

$$|\nabla| = \sqrt{\nabla_x^2 + \nabla_y^2}$$

Este valor, tras la aplicación de un umbralizado, será el que determine si un punto es considerado o no como punto de borde.

Las técnicas de detección de bordes dan lugar implícitamente a un aumento del ruido, por lo que normalmente suelen ir acompañadas del uso de operadores de suavizado. Los suavizados están asociados a máscaras de convolución de tamaño ajustable. Las más simples son del tipo:

+1	+1	+1
+1	+1	+1
+1	+1	+1

Con el operador de Sobel este suavizado se realiza de forma implícita. Otro tipo es el suavizado gausiano, donde las máscaras utilizadas son funciones gausianas bidimensionales. Existe un parámetro σ que representa el tamaño del suavizado, en cuanto a la distancia en píxeles considerados. Cuanto mayor sea σ mayor es el suavizado y menos se tienen en cuenta los detalles. Se dice que σ determina la escala a la que vemos la escena.

En caso de disponer de una secuencia temporal de imágenes de una misma escena, adquiridas por la cámara, existe la posibilidad de realizar un suavizado temporal de la entrada. El valor suavizado correspondiente a un pixel dependerá del valor actual de la imagen para ese pixel y del valor anterior del mismo, es decir:

$$Gris_t := (1-\alpha)\cdot Entrada_t + \alpha\cdot Gris_{t-1}$$

Donde α es la inercia temporal del suavizado, que deberá ser fijada convenientemente. Esta técnica sólo es aplicable cuando no hay muchos movimientos, bruscos o continuos, en la escena de entrada.

A pesar de las diferencias entre los distintos tipos de máscaras usadas, los resultados de los operadores son bastante parecidos. En la siguiente figura se muestra un ejemplo comparativo de los dibujos de bordes obtenidos para cada operador.

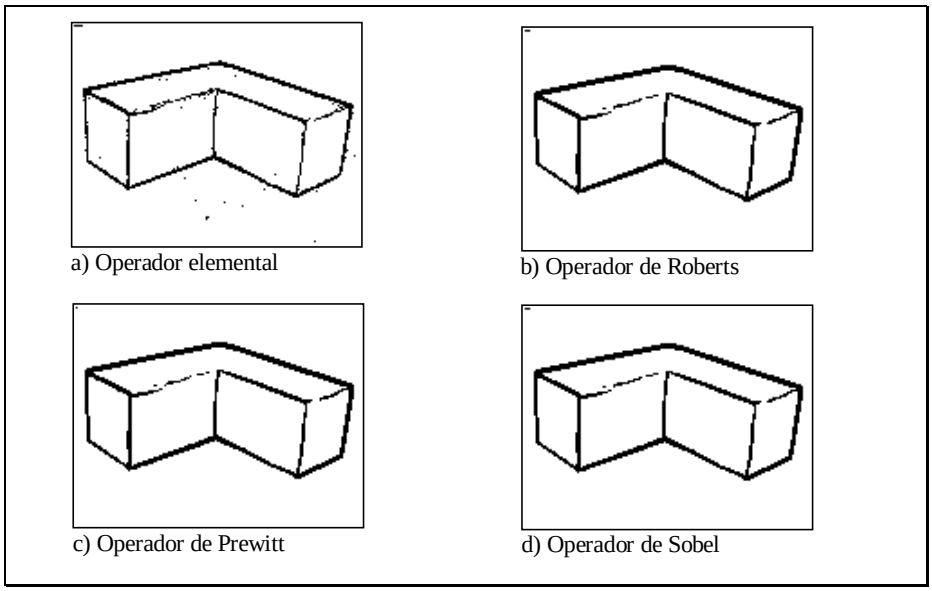

Figura 7 *Imágenes de bordes obtenidas con distintos operadores básicos*

Los resultados destacan por su gran similitud, sólo la salida del operador elemental (a) es algo distinta del resto. Se puede apreciar que este operador ha reconocido como borde algunos puntos de ruido. Los otros operadores eliminan este ruido, pero dan lugar a bordes más anchos, lo cual puede afectar negativamente a los procesos posteriores. Además, las imágenes de bordes fueron obtenidas utilizando valores de umbral distintos para cada operador. Es decir, no existe un criterio numérico único para el umbral, sino que es específico de cada operador.

En conclusión, para obtener una buena solución al problema de la detección de bordes, es necesario realizar otros tratamientos, que eviten la introducción de ruido a la vez que la formación de bordes gruesos. El operador de Canny, avanzando en los trabajos de Marr y Hildreth, realiza estos procesos para conseguir resultados más que aceptables.

3.2.3. El operador de Canny.

En general, un cambio en la intensidad de la imagen en una zona, dará lugar a un alto valor del gradiente en varios píxeles próximos a la misma. La idea fundamental del operador de Canny consiste en considerar sólo los puntos con valor máximo del gradiente en la dirección de este. Con ello se pretende evitar la formación de los bordes gruesos antes vistos, sólo los máximos serán tenidos en cuenta. El algoritmo trabaja con dos valores de umbral, umbral máximo y mínimo, y se puede resumir en los siguientes pasos:

1. Aplicar un operador de suavizado en la imagen original.
2. Calcular el valor del gradiente, módulo y ángulo, en cada punto de la imagen.
3. Buscar los puntos con valor de módulo máximo, en la dirección del gradiente en ese punto.
   Para cada punto que sea máximo hacer:
   Si el módulo del gradiente supera el umbral máximo, añadirlo como punto de borde. Si está entre el umbral máximo y el mínimo, almacenarlo para tratarlo en el siguiente paso. En otro caso, se rechaza como punto de borde.
4. Para cada punto almacenado en el paso anterior, si está al lado de algún punto de borde, entonces añadirlo como punto de borde.

Los resultados encontrados por este algoritmo son bastante buenos y muy adecuados al contexto en el que se aplica la imagen de bordes en los procesos posteriores. En la siguiente figura se muestran algunos ejemplos, en comparación con lo obtenido por los otros operadores.

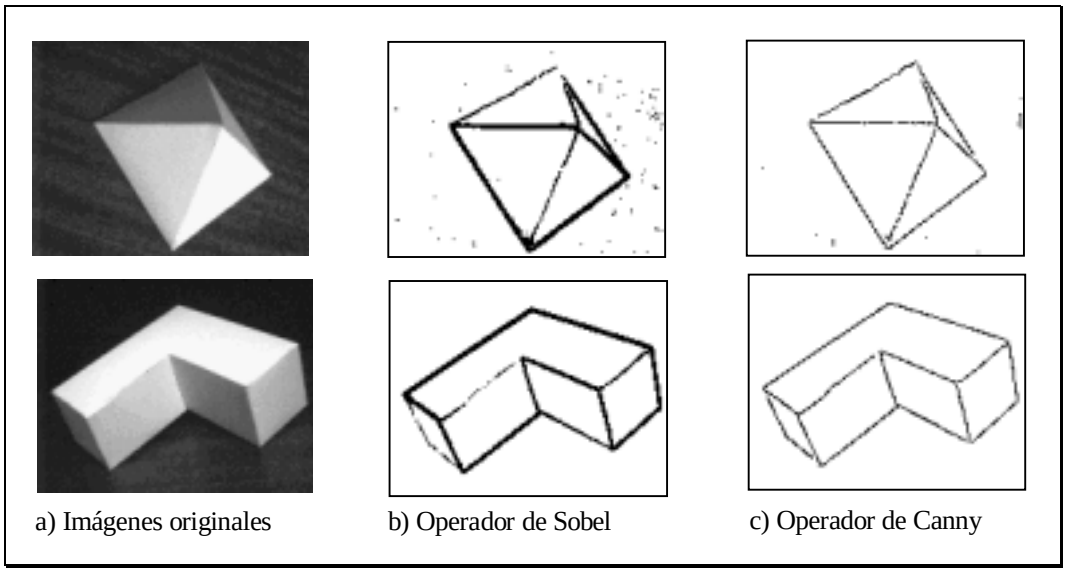

Figura 8 *Resultados de algunos de los procesos de detección de bordes*

Como se puede ver, con el operador de Canny se obtienen imágenes de bordes conectadas, bastante limpias de ruido y en la mayoría de los casos de un sólo pixel de anchura. Aunque utiliza un algoritmo más complejo que los operadores simples, la ejecución del proceso es bastante rápida, como puede comprobarse en el anexo B. El suavizado utilizado en este caso es un suavizado gausiano con valor de σ=3.

Otra gran ventaja del operador de Canny es la relativa a los parámetros del algoritmo: los umbrales máximo y mínimo. El uso de dos umbrales, en lugar de uno, podría interpretarse como un aumento en la complejidad del uso, ya que se deberán calibrar los valores adecuados para distintas condiciones de iluminación. Sin embargo, en la práctica el resultado es el contrario. Se ha podido comprobar, empíricamente, que se pueden encontrar unos valores de umbrales adecuados, que se ajustan bastante bien a la mayoría de los casos bajo un, relativamente amplio, rango de variaciones de las características de la escena.

En definitiva, la etapa de bajo nivel tiene por objetivo encontrar una descripción de la imagen original en función de los bordes de intensidad. Estos bordes dividen la imagen en zonas cuyos lugares correspondientes en la escena se espera que sean superficies con características de profundidad homogéneas. La aplicación de esta etapa es relativamente sencilla y automática, obteniéndose con el mejor de los operadores estudiados, el de Canny, resultados bastante buenos. Con ello se consigue que el proceso sea invariante a color, intensidad y a las características de iluminación de la escena. Por contra, el proceso será muy sensible en caso de existir de texturas muy destacadas, marcas, dibujos o pliegues suaves en los objetos.

3.3. La etapa de medio nivel: Detección de segmentos.

3.3.1. Introducción.

El procesamiento de imágenes a medio nivel está relacionado principalmente con la extracción de descripciones, a partir de las obtenidas por el bajo nivel. Estas descripciones se encuentran en un formato más simbólico, y normalmente definen la forma y posición de determinadas porciones de la escena. El problema fundamental es la distinción entre el fondo y la forma. Al igual que

en el bajo nivel, el proceso no necesita ningún conocimiento específico sobre los objetos presentes en la escena, pero sí de los tipos de formas que pueden aparecer.

Dentro de las técnicas de medio nivel encontramos las de segmentación, de las cuales existe una gran variedad (segmentación por color, bordes, textura, movimiento). En nuestro caso, la técnica a utilizar viene impuesta por las necesidades de los procesos posteriores y por la salida procedente del bajo nivel. Por lo tanto, se deberán aplicar técnicas de segmentación por agregación de bordes.

En general, las técnicas de agregación de bordes parten de una lista de puntos de borde, y realizan un proceso de combinación de estos puntos en estructuras paramétricas predefinidas, como por ejemplo, segmentos de líneas, arcos de circunferencia, de parábola o splines (trozos de curvas polinomiales de grado 2 ó 3). La representación en base a estas estructuras agregadas es más simbólica, en el sentido de que los puntos individuales dejan de tener importancia. Además, se obtiene una reducción significativa en la cantidad de información almacenada.

Las estructuras paramétricas que se utilizarán en este proyecto serán segmentos de líneas. Hablamos, por tanto, de un proceso de detección de segmentos. Esto da lugar a una importante implicación en el tipo de entradas permitidas: los objetos a utilizar deberán ser objetos poligonales, es decir formados por caras planas. Esta restricción es bastante importante, ya que la mayoría de los objetos del mundo real están formados por superficies más o menos curvadas. Sin embargo, se ha optado por el uso de líneas rectas ya que el problema de agregación de bordes resulta más sencillo de tratar.

3.3.2. Acercamientos al problema de detección de segmentos.

Básicamente existen dos grandes acercamientos al problema de encontrar segmentos en una imagen de puntos de borde: los modelos de agregación y la transformación del espacio de características.

Las técnicas que utilizan modelos de agregación intentan hacer la agrupación de bordes a través de una ordenación de los puntos de borde, para después encontrar tramos de líneas rectas. El problema de estos métodos es su elevada complejidad computacional, ya que normalmente se basan en búsquedas iterativas sobre todo el conjunto de puntos, para encontrar los que más probablemente siguen la secuencia.

En las técnicas de transformación del espacio de características, se aplica una simple transformación a los puntos de bordes, obteniendo un mapeo de cada punto al espacio de parámetros. En el caso de los segmentos de líneas, el espacio de parámetros está formado por dos dimensiones, θ y ρ, que corresponden en el espacio de características a los parámetros de una línea según la fórmula:

$$x \cdot \cos(\theta) + y \cdot \mathrm{sen}(\theta) = \rho$$

Esta es la ampliamente conocida y estudiada transformada de Hough, tratada extensamente en [Gon93]. Adicionalmente, estas técnicas realizan una discretización del espacio de parámetros en celdas, a cada una de las cuales se le asocia un contador. Al realizar el mapeo de cada punto de borde, se incrementan los contadores de las celdas correspondientes del espacio transformado. Posteriormente, para encontrar las rectas se realiza una búsqueda de los máximos locales en los contadores, y se determinan los límites de cada segmento.

Planteado de esta forma, esta técnica aun deja importantes problemas por resolver, como la elevada carga computacional, el error producido por la discretización del espacio de parámetros y la complejidad de los procesos posteriores de búsqueda de máximos y determinación de los límites de los segmentos. No obstante, existen variaciones del algoritmo que consiguen resolver estos problemas de forma efectiva y en un tiempo relativamente rápido. La idea de estas consiste en no realizar el mapeo para todos los puntos de borde, sino para unos grupos de puntos próximos que presentan una afinidad lineal.

3.3.3. El algoritmo EMH para detección de segmentos.

Frente a estas técnicas clásicas, aparece en el artículo [Lop97] un nuevo acercamiento al problema de la detección de las líneas, desde un punto de vista probabilístico. El problema es reformulado como un problema de estimación de parámetros usando modelos de mezcla. Un modelo de mezcla es una forma de representar una función de densidad de probabilidad compleja, como la suma de varias funciones de densidad básicas. La imagen es considerada como una muestra aleatoria de esta mezcla. Las funciones de densidad básicas corresponden a los segmentos, y su forma será parecida a una tienda de campaña. Además, en la mezcla se añade un componente uniforme para considerar el ruido.

Para el cálculo de los parámetros de la mezcla se usa una versión adaptada del algoritmo Expectación-Maximización, conocido como algoritmo EM. El proceso consiste básicamente en ajuste iterativo en dos pasos: calcular la probabilidad de que cada punto sea generado por un componente de la mezcla y recalcular después los parámetros de cada componente (cada uno correspondiente a un segmento) según esas probabilidades.

En la versión modificada, se añade un paso heurístico para eliminar, añadir o dividir segmentos de la mezcla. El proceso se puede seguir repitiendo de forma indefinida no existiendo, en principio, comprobaciones de convergencia globales, aunque normalmente estas se alcanzan muy rápidamente. En caso de utilizar una secuencia de imágenes, esta comprobación no tendría ningún sentido. En resumen, la estructura del algoritmo de detección de segmentos es la siguiente:

1. Inicialización.
   Generar un conjunto inicial de N segmentos aleatorios, dados por los puntos extremos (x1, y1) y (x2, y2).
2. Repetir hasta que no se necesite más procesamiento visual.
   2.1. Repetir hasta que no haya un cambio significativo de los segmentos.
      2.1.1. Paso E, de expectación.
         Para cada punto y cada segmento calcular $q_{i,e}$, la probabilidad a posteriori de que el punto *e* haya sido generado por el segmento *i*.
      2.1.2. Paso M, de maximización.
         Para cada segmento *i*, calcular sus parámetros (x1, y1), (x2, y2), teniendo en cuenta las probabilidades de pertenencia de los puntos a ese segmento. Esto se hace utilizando la media y la covarianza de los puntos, ponderados por las probabilidades de pertenencia a cada segmento.
   2.2. Asignar cada punto al segmento en el que tiene más probabilidad de pertenencia.
   2.3. Paso H, heurístico.
      Eliminar segmentos muy cortos o con pocos puntos asignados.
      Dividir los segmentos cuyos puntos no pasan un test de linealidad.
      Añadir algunos segmentos aleatoriamente, si se considera necesario.

Gracias al paso heurístico, el algoritmo adquiere la capacidad de buscar el número óptimo de líneas de forma automática. A pesar de ello, se requiere un número inicial de segmentos N, con el que empezará a trabajar el modelo de mezcla.

La condición del paso 2.1 suele ser sustituida por una más sencilla, ya que se trata de una simple comprobación local de convergencia, para introducir un paso heurístico. Además, en ocasiones la convergencia del proceso es lenta, por lo que interesa ejecutar la heurística más a menudo para eliminar las líneas que provocan el problema. La solución es repetir este punto un número fijado de veces.

Este algoritmo tiene varias ventajas sobre los acercamientos clásicos:

- La propia naturaleza del algoritmo EMH permite realizar un seguimiento en tiempo real de los segmentos moviéndose en una secuencia de imágenes. Es decir, el proceso tiene una cierta dinámica que le permite encontrar rápidamente la solución cuando hay pequeñas variaciones en la entrada. Esto no ocurre con las técnicas clásicas.

- El tratamiento del ruido es implícito, por lo que normalmente se comporta bien frente a cierto grado de ruido, sobre todo si es uniforme.
- Puesto que el algoritmo trabaja con estructura global, más que con proximidad local, es muy adaptable a la variación en el número total de puntos de entrada. Esto permite seleccionar aleatoriamente una porción de los puntos de la entrada, para aplicar el algoritmo sobre ellos. Con ello, se podría obtener una solución aceptable, pero en un menor tiempo ya que el algoritmo trabaja con menos puntos.
- El proceso puede funcionar de forma bastante automática, en cuanto a los parámetros requeridos. El número inicial de líneas N, no es una cuestión crítica ya que el paso H lo ajustará al número adecuado a cada caso. El valor del número de iteraciones en el paso 2.1 no suele ser tampoco muy determinante, influyendo solamente en el tiempo que se tarda en encontrar la solución. Por otro lado, el parámetro del tamaño de las líneas suele ser establecido a un valor fijo que depende de la resolución de la imagen.

Para este proyecto, se dispone de una implementación del algoritmo EMH, desarrollada por los creadores de este método. A continuación mostramos algunos ejemplos de los resultados obtenidos por el algoritmo, cuyas entradas son los dibujos de bordes resultantes de la etapa de bajo nivel.

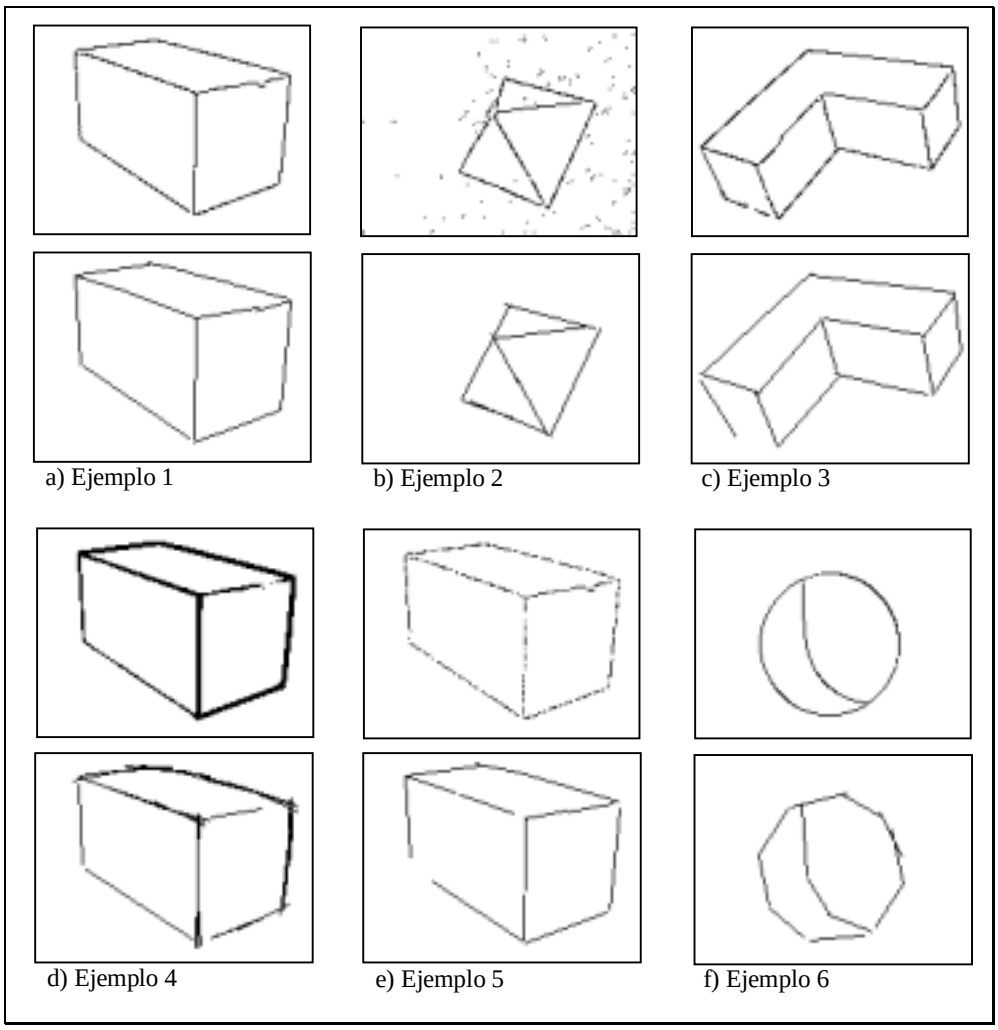

Figura 9 *Resultados del algoritmo EMH*

En la parte superior se muestran los dibujos de bordes de entrada al algoritmo, que en todos los casos, excepto en el ejemplo 4, corresponden a la aplicación del operador de Canny con imágenes tomadas de escenas reales. En la parte inferior están los resultados obtenidos por el algoritmo

de detección de segmentos para cada entrada. Se puede apreciar que en la mayoría de los casos la solución es razonablemente buena. En todos ellos, la solución corresponde a 10 ejecuciones del paso principal, el paso 2, donde en cada una de ellas el paso 2.1 se repite 8 veces. El tiempo de ejecución varía de un caso a otro, dependiendo del número de puntos de entrada.

El ejemplo 1 es una situación bastante simple, donde se consigue una buena solución rápidamente. No obstante, hay que tener en cuenta que la solución encontrada contiene 14 segmentos (cinco más del número óptimo teórico) ya que algunas líneas largas han sido divididas en varios segmentos colineares. En la entrada del ejemplo 2 se ha añadido intencionadamente cierto grado de ruido en la entrada, modificando los umbrales del operador de Canny. Exactamente, el 26.3% de los puntos de entrada son de ruido, a pesar de lo cual las líneas encontradas son una buena solución, debido a que el ruido existente es uniforme. En caso de existir agrupaciones aleatorias de puntos de ruido se puede comprobar que el algoritmo no funciona tan bien, ya que trata de representarlo introduciendo una gran cantidad de segmentos. El ejemplo 3 contiene más segmentos que en los otros casos, para los cuales se encuentra la representación de líneas adecuada. En este caso hay un segmento faltante, aunque es añadido en posteriores pasos de ejecución del algoritmo.

Para la entrada del ejemplo 4 se ha utilizado el operador de bordes de Prewitt, sobre la misma imagen del primer ejemplo. Como se aprecia, esto ha dado lugar a la aparición de bordes gruesos, que intentan ser representados con muchos segmentos paralelos y muy próximos entre sí. En total hay 38 segmentos en la solución, cuando en teoría deberían haber 9. Esto es un problema para las etapas posteriores, ya que hay una gran cantidad de información redundante y ruidosa. En el ejemplo 5 se ha utilizado el operador de Canny, realizando después una eliminación aleatoria del 30% de los puntos de borde. Con ello se consigue mejorar la velocidad de ejecución, sin perjudicar la obtención de una buena solución. No obstante, esta reducción presenta el problema de que la solución es muy oscilante, en cuanto a lo que se obtiene de una iteración a otra. Además, cuanto más reduzcamos el número de puntos de entrada más probabilidad hay de que la muestra no represente bien los puntos de borde originales, con lo que la solución será mala. Por último, en el ejemplo 6 se ha aplicado el algoritmo EMH para una figura con forma esférica. Aunque el resultado es una buena representación de la entrada (desde un punto de vista subjetivo) no será muy aplicable a las etapas posteriores ya que no hay ninguna indicación de que los segmentos sean más o menos curvados.

3.4. La etapa de alto nivel.

3.4.1. Introducción. La visión de alto nivel en el dominio 3D.

La etapa de alto nivel constituye el nivel más simbólico de procesamiento, en cuanto que se basa en el uso de información estructural relevante, dentro del dominio del problema, sin considerar las descripciones de las partes en términos absolutos. En el sentido más general, el objetivo de estas técnicas es la interpretación de las escenas, esto es, la capacidad de inferir información acerca de: ante qué tipo de escena nos encontramos, qué objetos están presentes en ella, con qué características y en qué posiciones. Se trata pues, de emular una de las principales capacidades inteligentes de la mente humana, algunos autores hablan incluso de la posibilidad de superar las capacidades humanas, [Nal93]. Sin embargo, por el momento las aplicaciones de la visión artificial se reducen a dominios muy concretos, dentro de entornos controlados o sin excesiva variabilidad.

Según el tipo de dominio, la interpretación puede ser más o menos directa. Para un dominio bidimensional, por ejemplo, puede no ser necesario llegar a un excesivo nivel simbólico antes de realizar la interpretación. En este caso, podríamos hablar de una interpretación de bajo nivel. Sin embargo, a medida que aumenta la complejidad del problema, introduciendo dominios más próximos al mundo real, se hace necesario utilizar una descripción más simbólica y elaborada.

Supongamos un sistema de reconocimiento óptico de caracteres. La técnica utilizada debería ser al menos invariante a desplazamientos, escala y giros. Esto implica que se deberán conside-

rar las relaciones, proporciones, ángulos o posiciones que se mantienen, más o menos fijos, de una realización a otra de un mismo carácter. En el dominio tridimensional la técnica de comparación de modelos también debería ser invariante a desplazamientos, giros y escala, además de luminosidad y color. Sin embargo, en una transformación 3D no tienen porqué conservarse tamaños, proporciones ni ángulos relativos entre las partes constituyentes del dibujo. Ello es debido a la enorme reducción de información implícita en la proyección de una escena 3D en una imagen 2D. A modo ilustrativo, se muestran en la siguiente figura varias realizaciones de un mismo objeto tridimensional.

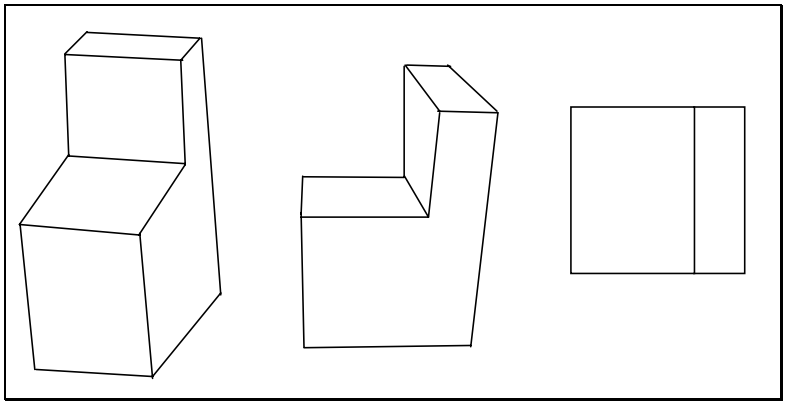

Figura 10 *Tres proyecciones de un objeto tridimensional*

Como se puede apreciar, una transformación tridimensional puede dar lugar a enormes variaciones en las distintas vistas de un mismo objeto. Las proporciones entre las longitudes de las líneas no se mantienen fijas, siendo posible que una línea sea claramente más larga que otra en una vista, pero en otra vista sea mucho menor. Para las líneas paralelas en el espacio, las proporciones de tamaños se mantienen en la imagen si la proyección es paralela, pero no si es perspectiva. En este caso además, puede que no se conserve el paralelismo de estas líneas. Evidentemente, los ángulos tampoco se conservan, pudiendo pasar fácilmente un ángulo agudo en una vista a ser mayor que 90º en otra vista. En el peor caso, se pueden perder algunas caras, pueden aparecer otras nuevas y una arista de un objeto puede proyectarse en un simple punto. En consecuencia, necesitamos buscar las propiedades que realmente sean relevantes en la estructura 3D de los objetos.

El dibujo de líneas de borde supone una descripción bastante elaborada de la imagen de entrada, procedente de la cámara. Tal y como es construido, es de esperar que las líneas del dibujo se correspondan con aristas de los objetos en la escena y los puntos de unión entre líneas con los vértices. Como vimos, su uso tiene una justificación biológica bastante razonable. Sin embargo, las líneas por sí mismas no tienen un gran valor, sino que es necesario describirlas en relación con las demás. Entre estas relaciones podemos encontrar una gran cantidad de posibilidades: una línea está unida o no a otra, es N veces más larga o corta que otra, tiene un cierto ángulo con otra línea, corta a otra línea, etc. La primera relación de las anteriores, es bastante probable que se mantenga de una vista a otra, aunque la información que ofrece no es muy rica. Las otras relaciones aportan más información pero, como hemos visto, pueden variar ampliamente de una a otra vista.

Por lo tanto, la primera cuestión será definir el conjunto de relaciones invariantes entre las líneas, para aplicarlas después al dibujo de líneas de borde. De esta forma, la representación de líneas se transforma en la representación propia del alto nivel, que contendrá las propiedades previamente definidas. Aun quedará el problema de decidir qué hacer con esta información, es decir, cómo hacer la interpretación.

3.4.2. Métodos de interpretación de dibujos de líneas.

Varios métodos han sido propuestos para tratar el problema de la interpretación de dibujos de líneas. Entre ellos podemos diferenciar básicamente tres acercamientos: etiquetado de nodos, comparación de grafos y técnicas de alineamiento. Las técnicas basadas en etiquetado de nodos realizan un proceso de búsqueda de un etiquetado consistente entre los nodos, donde las etiquetas

indican alguna información sobre los nodos a los que son asignados. Las de comparación de grafos están orientadas a la resolución de problemas de clasificación, más que de interpretación, aunque se pueden extender para tratar problemas de ese tipo. Las técnicas de alineamiento no son exclusivas del uso de dibujos de borde, y requieren una proceso previo de comparación o de búsqueda de equivalencias. Son muy adecuadas en problemas del tipo de reconocimiento de caras de personas.

El proceso de interpretación de dibujos de líneas desarrollado en este proyecto se basa fundamentalmente en una comparación de grafos, aunque incorporando muchas de las ideas expuestas en los otros acercamientos. El proceso de alto nivel al completo es presentado en el apartado 3.4.3. Antes comentaremos brevemente las principales características de los dos primeros acercamientos, haciendo hincapié en el etiquetado de nodos, como una técnica subyacente a la comparación de grafos. El método de alineamiento será expuesto más adelante, ya que es utilizado en una etapa posterior del proceso desarrollado.

3.4.2.1. Técnicas de etiquetado de nodos.

El etiquetado de nodos es un método general de interpretación que se ha usado no sólo en visión por computador, sino también en otros ámbitos de la inteligencia artificial. Partimos de una representación de un objeto desconocido, descompuesto en partes o nodos, en la que queremos identificar qué es cada uno de ellos, dentro de un conjunto de posibles valores o etiquetas. Existen una serie de restricciones definidas sobre las posibles etiquetas de los nodos, en función de la información almacenada. Estas restricciones pueden ser unarias (afectan a una sola parte) o n-arias (se basan en la relación existente entre varias partes).

De esta forma, el proceso de interpretación se basaría en una propagación de las restricciones a través de la estructura de representación. Partiendo con todas las posibles etiquetas en todos los nodos, se eliminarán aquellas que sean incompatibles con las restricciones definidas, hasta llegar a una situación de estabilidad. En la figura 11 aparece un ejemplo de un problema de etiquetado de nodos en una escena de una habitación.

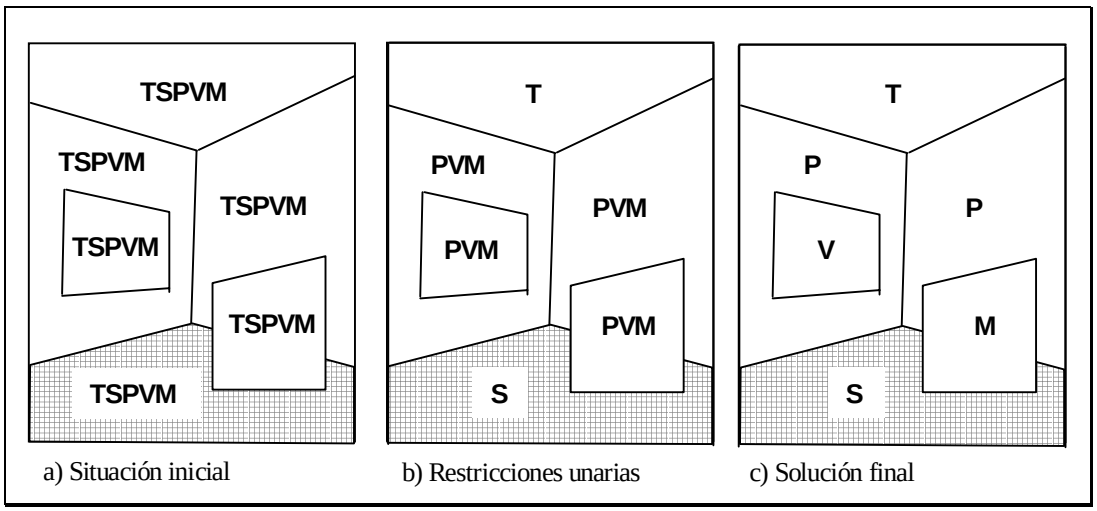

Figura 11 *Etiquetado de regiones en la escena de una habitación*

Inicialmente, cada región de la imagen tiene todas las etiquetas: {Techo (T), Suelo (S), Pared (P), Ventana (V), Mueble (M)}. En un primer paso, se aplican las restricciones unarias, en este caso: el techo siempre está arriba, y el suelo es a cuadros. Después se utilizan restricciones n-arias del tipo: una pared está unida al techo y al suelo, una ventana está en una pared, y el mueble está apoyado en el suelo. Con esto, se llega a una solución única y consistente, en la que tenemos información de qué representa cada zona de la imagen.

Sin embargo, en un problema de etiquetado de nodos en general, no está garantizado que se obtenga una única solución para una entrada correcta, que la solución obtenida se corresponda con la que lógicamente se esperaría o que el proceso tenga alguna utilidad en caso de existir cierto

ruido en la entrada. Todo depende del tipo de información almacenada y de las restricciones definidas.

Utilizando estas ideas, D. A. Huffman definió los aspectos básicos para aplicar el etiquetado de nodos a un problema genérico de interpretación de dibujos de líneas, donde los nodos son cada uno de los segmentos del dibujo. Según Huffman, estos segmentos pueden ser etiquetados, inicialmente, en uno los siguientes tipos:

- **Líneas de frontera**, que ocurren donde un objeto acaba. La etiqueta para este tipo de líneas es una flecha, dibujada en la misma dirección de la línea, y en el sentido horario respecto al objeto.
- **Líneas interiores**, que separan dos caras de un mismo objeto. Según el ángulo de las caras, estas líneas pueden ser de borde convexo, denotadas por "+", o de borde cóncavo, denotadas con "-".

Un ejemplo de este tipo de etiquetas se ilustra en la figura 12, donde aparece una figura conocida como "Ele".

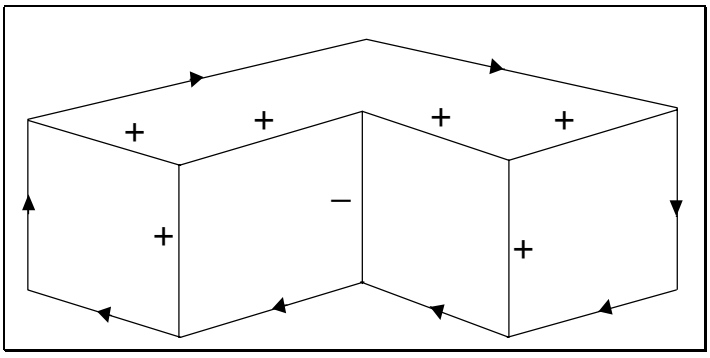

Figura 12 *Ejemplo de los tres etiquetados básicos de una línea*

A la hora de definir las restricciones en los etiquetados, Huffman encontró que en las uniones entre líneas (es decir, en los puntos de intersección) sólo ciertos conjuntos de asignaciones de etiquetas son físicamente realizables, según el tipo de unión. Por lo tanto, no existen restricciones unarias y las restricciones n-arias definidas son del tipo: si varias líneas tienen un punto de unión común, entonces las etiquetas de esas líneas deben ser algunas de las definidas dentro de un catálogo de etiquetados posibles para ese tipo de unión. Este catálogo depende del tipo de objetos que nos podemos encontrar y de otras características del dominio en que se trabaje. En el caso más simple, se pueden hacer las siguientes suposiciones:

- No existen sombras ni marcas en las caras de los objetos.
- Todos los vértices están formados por la intersección de tres caras.
- Una pequeña variación en el punto de vista no provocará un cambio en los tipos de uniones del dibujo. Esta es la que normalmente se conoce como "suposición del punto de vista generalizado", que trata de evitar situaciones anómalas (como la proyección más a la derecha de la figura 10) aunque físicamente factibles.

Con estas restricciones, el catálogo de uniones físicamente realizables consta de 18 tipos frente a los 208 posibles sin considerar la restricción de ser físicamente factible. Este catálogo es mostrado en la figura 13.

De esta manera, tenemos definidos todos los aspectos necesarios para realizar la interpretación de un dibujo de líneas, que cumpla con las suposiciones establecidas. Inicialmente asignamos las tres etiquetas a todos los nodos y después vamos comprobando las restricciones y eliminando etiquetas, hasta llegar a una situación en la que no haya ningún cambio. Podemos llegar a una solución única, a una situación de ambigüedad con varias etiquetas en algún nodo o a la falta de etiquetas en alguno de los nodos, indicando una inconsistencia del dibujo.

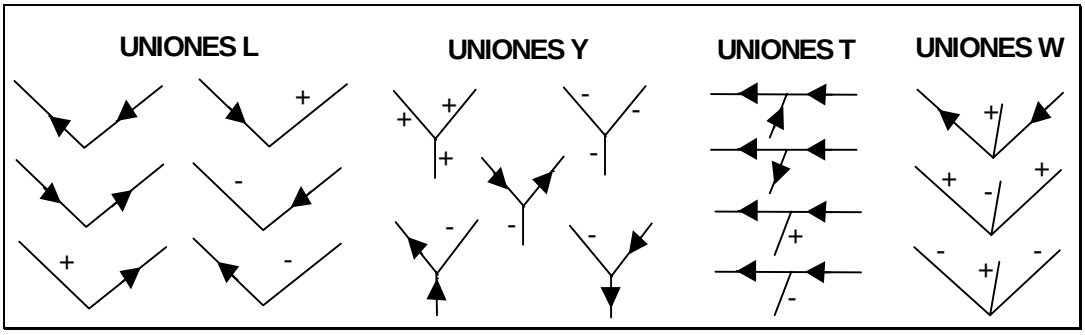

Figura 13 *Catálogo de posibles etiquetados en las uniones*

Este sencillo ejemplo de etiquetado de líneas fue extendido por Waltz, quién además definió un procedimiento para resolver de forma eficiente el problema de la propagación de restricciones. Waltz elimina la suposición del punto de vista generalizado, la restricción de las aristas de sólo tres vértices, y considera las sombras, los tipos de iluminación y las líneas de rotura entre dos objetos. Con ello, obtiene un total de unas 50 posibles etiquetas por nodo, que amplían el catálogo de uniones posibles, pero ofreciendo mucha más información que el sencillo caso de sólo 3 etiquetas. Dentro del catálogo aparecen otros tipos de uniones, donde convergen 4 o más líneas. El número de elementos del catálogo asciende a más de 1.800, un número pequeño en comparación con la enorme cantidad de etiquetados posibles, si no se consideran sólo los físicamente realizables.

Sin embargo, aun se sigue sin considerar la existencia de ruido, debida a la aparición de líneas faltantes o sobrantes. Este ruido dará lugar a que el proceso de interpretación llegue a una solución que no se corresponda con la esperada o, en la mayoría de los casos, a una situación de inconsistencia. Para resolver este problema sería necesario definir un nuevo valor de etiqueta para las líneas sobrantes, de ruido. Después, se debería ampliar el catálogo, haciendo que en cada unión hayan ninguna, una o varias líneas faltantes o sobrantes. Esto provocaría un considerable aumento del catálogo, con los problemas de complejidad computacional que implica, y en el peor de los casos la aparición de numerosas interpretaciones ambiguas de una misma escena, por la falta de información que supone la existencia de ruido.

Por otro lado, el resultado obtenido es una interpretación a nivel de líneas. Sería necesaria una fase posterior para delimitar los objetos dentro de la escena, en base a las etiquetas, así como para encontrar propiedades relevantes dentro de la estructura tridimensional de estos objetos. Posteriormente se debería realizar algún proceso de clasificación para identificar los objetos dentro de alguna categoría existente.

En conclusión, aunque el método expuesto resulte muy sensible al ruido y realice una interpretación no global de la escena, podemos extraer varias ideas interesantes. Primero, que el proceso de interpretación debe considerar la relación entre las líneas, más que las líneas en sí, y en especial las relaciones de unión entre las líneas. Segundo, que además de la relación de “una línea está o no unida a otra”, existe otra propiedad que se mantiene invariante a lo largo de un gran rango de vistas, que es el tipo de uniones en que dos o más líneas se juntan, ya sean los tipos L, Y, T, W u otros del mismo estilo.

3.4.2.2. Técnicas de comparación de grafos.

En muchas aplicaciones de la inteligencia artificial, resulta posible representar la información mediante una estructura de grafo dirigido o no dirigido. El conjunto de nodos representa ciertas partes de las cuales conocemos determinadas características y los enlaces representan relaciones entre esas partes. Los grafos no dirigidos se utilizan cuando las relaciones son simétricas, por ejemplo relaciones de líneas paralelas o perpendiculares, mientras que los dirigidos son utilizados para relaciones no simétricas, como las de mayor o menor que. Cuando los enlaces del grafo pueden ser etiquetados dentro de un conjunto de posibles valores, se dice que tenemos un grafo atribuido. Los grafos atribuidos resultan muy interesantes, ya que el contenido de un nodo se define

estructuralmente, en función de sus relaciones con los otros nodos, dentro de un amplio dominio de posibles relaciones.

En este acercamiento, el paso central del proceso de interpretación es la comparación de grafos, entre los grafos que representan modelos y el grafo del dibujo a interpretar. Otras etapas pueden ser necesarias antes y después de la comparación, por ejemplo para verificar la hipótesis de interpretación o para crear el grafo a partir de la información de partida. Planteado de esta forma, parece claro que este acercamiento está orientado principalmente a la resolución de problemas de clasificación. No obstante, veremos que es lo suficientemente general como para poder ser aplicado a problemas de interpretación.

Supongamos, por ejemplo, un problema de reconocimiento de caracteres basado en grafos dirigidos y atribuidos, del tipo mostrado en la figura 14. Los nodos son las líneas que forman el carácter y los enlaces indican relaciones entre líneas, etiquetadas con distintos valores de posiciones relativas. El reconocimiento de caracteres consistiría en resolver un problema de isomorfismo de grafos. Dado el grafo con el carácter a reconocer, se debería buscar un grafo del conjunto de modelos isomorfo al primero. Informalmente, se dice que dos grafos son isomorfos si existe una función biyectiva (que llamaremos emparejamiento), de los nodos del modelo a los del carácter a reconocer, tal que se preservan todos los enlaces. Puesto que estos enlaces son etiquetados, deberíamos considerar también que se mantengan los tipos de los enlaces, entre los nodos del modelo y los correspondientes en el otro grafo.

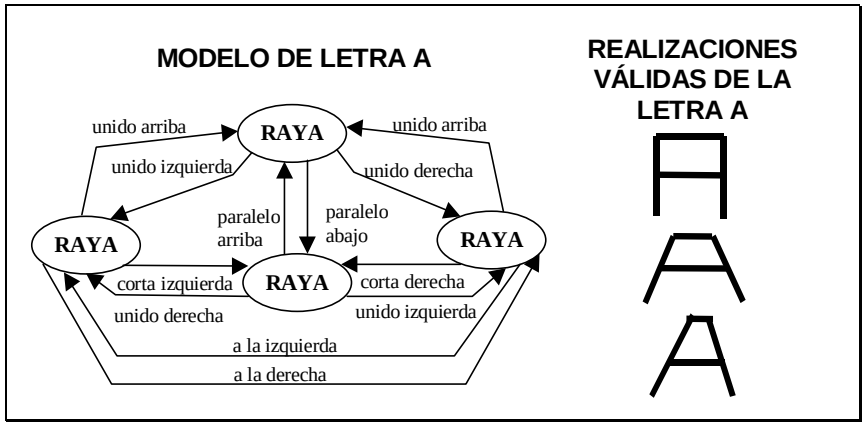

Figura 14 *Ejemplo de grafos de modelos*

El isomorfismo de grafos es de poca utilidad práctica en los problemas de interpretación de escenas, ya que no se puede esperar que los niveles inferiores sean capaces de obtener siempre una segmentación perfecta de la imagen. Una posible solución sería la definición de una medida de distancia entre grafos, en función del número de nodos o enlaces que deben ser añadidos o eliminados para conseguir que los grafos sean isomorfos.

Sin embargo, el verdadero problema es que normalmente la escena a interpretar puede constar de varios objetos, que están modelados de forma independiente. Por ello, un acercamiento más adecuado es la resolución de un problema de isomorfismo de subgrafos, es decir la búsqueda de isomorfismos pero a nivel de subgrafos. Este problema es computacionalmente más complejo, ya que no se conocen a priori los nodos que intervienen en el isomorfismo. Además, en general será posible que varios modelos puedan ser encontrados dentro de una misma escena.

Para aplicar la técnica de comparación de grafos, necesitamos estudiar y definir tres aspectos estrechamente relacionados: cómo están definidos y se crean los grafos a partir de la entrada (es decir, cuales serán los nodos y los enlaces), cómo se calculará la distancia entre dos grafos (posiblemente en función de la distancia básica entre nodos) y cuál será el algoritmo utilizado para resolver el problema del isomorfismo de subgrafos (ya que el problema es de elevada complejidad computacional). Todos estos aspectos serán tratados extensamente en los posteriores apartados del desarrollo.

3.4.3. Estructura del proceso de alto nivel desarrollado.

El proceso de alto nivel desarrollado en este proyecto se enmarca dentro de las técnicas de comparación de grafos, haciendo uso además, de muchas de las aportaciones de los otros acercamientos, para conseguir un tratamiento completo del problema. Visto como una caja negra, el proceso de alto nivel recibe en la entrada un conjunto de líneas de borde de la escena y obtiene a la salida una descripción del objeto u objetos reconocidos, posiblemente con información sobre la posición y orientación de estos objetos.

A lo largo de esta etapa se usará una representación de grafos, donde el paso central del proceso es la comparación. Esta comparación servirá como punto de partida para realizar una hipótesis de interpretación, que después deberá ser verificada utilizado un proceso de alineamiento, geométricamente más preciso. Por otro lado, antes de realizar la comparación se deberá ejecutar un procedimiento de creación del grafo a partir del dibujo de líneas. En conjunto, el proceso de alto nivel, mostrado en la siguiente figura, integra todos estos pasos dentro de una estructura de ejecución secuencial.

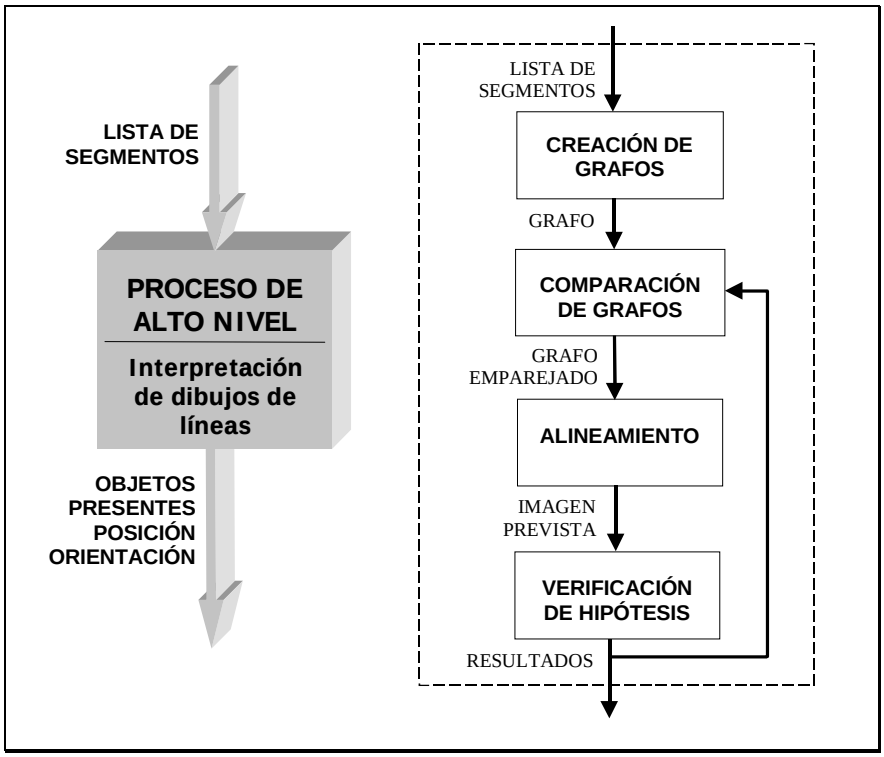

Figura 15 *Estructura global del proceso de alto nivel*

En este esquema, las flechas representan a la vez un flujo de información y de control, ya que en este caso carece de sentido ejecutar cada etapa de forma paralela. A lo largo de los siguientes apartados se describirán detalladamente los pasos que constituyen el proceso de interpretación de dibujos de líneas, desde la definición de la estructura de grafos hasta la verificación de la hipótesis.

3.4.4. Grafos asociados al dibujo de líneas.

La definición de los grafos va estrechamente unida al problema de determinar qué información es relevante para el problema y cuál no lo es. En general, una propiedad será considerada como relevante si se mantiene invariante de una realización a otra de un mismo objeto. Esto es, la representación de grafos debería describir características de los objetos que se mantienen invariantes para todos los ejemplares de un mismo objeto. Sin duda alguna, este es un aspecto clave para la obtención de buenos resultados, ya que todo el proceso de alto nivel utilizará la estructura de representación de grafos.

Ya hemos visto que en un dominio tridimensional la mayoría de las propiedades del dibujo de líneas sufren grandes variaciones al cambiar el punto de vista, por lo que ángulos y proporciones entre las líneas no pueden ser considerados como relevantes. Sin embargo, las técnicas de etiquetado de nodos sugerían, en cierto sentido, que la principal propiedad invariante del dibujo se encontraba en las uniones entre segmentos. Básicamente, dos o más líneas unidas en un punto permanecían siempre unidas y, bajo un gran rango de puntos de vista, el tipo de unión (unión L, Y, W ó T) seguía siendo el mismo.

Por lo tanto, podríamos pensar en una primera definición del grafo asociado al dibujo de líneas como la expuesta en el siguiente punto.

3.4.4.1. Grafos de vértices.

El grafo G asociado a un dibujo de líneas, es un grafo no dirigido formado por un par G= (Nodos, Enlaces) donde:

- Los nodos representan cada uno de los puntos de intersección entre las líneas. Estos pueden ser etiquetados con los tipos: L, T, W, Y u otro no predefinido. (Se puede añadir un tipo I, para representar el punto extremo de una línea que no tiene intersección con ninguna otra.)
- Los enlaces entre dos nodos representan las líneas del dibujo que pasan por esos dos puntos de unión.

Esto tipo de estructura es lo que podríamos denominar "grafo de vértices asociado al dibujo de líneas", ya que la representación está centrada en las uniones entre las líneas (los vértices en la escena). Utilizando esta representación, los grafos que tendríamos serían como los del ejemplo mostrado en la figura 16.

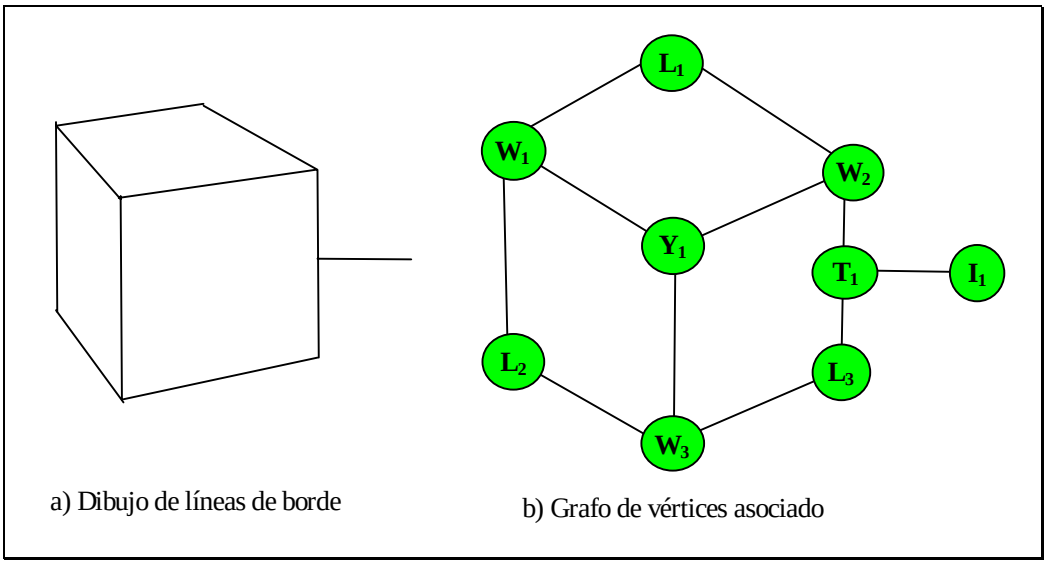

Figura 16 *Grafo de vértices asociado a un dibujo de líneas de borde*

Con esta estructura, el coste de comparación de dos nodos de dos grafos distintos, podría basarse en los tipos de los nodos y en la existencia o no de enlaces con los restantes nodos. Sin embargo, esta simple condición de existencia de enlaces es insuficiente en sí misma, ya que no proporciona ninguna información de posiciones relativas, que podría resultar fundamental para la posterior comparación. Esto podría solucionarse haciendo que los grafos sean dirigidos y atribuidos, de manera que el enlace de un nodo *A* con otro nodo *B* indique la posición de esa línea (por definición los enlaces representan líneas) dentro del tipo de unión de *A*. Por ejemplo, el enlace $W_3 \rightarrow Y_1$ contendría una etiqueta indicando que corresponde a la línea central del tipo de unión W.

Esto es sólo una solución parcial, que no resuelve todos los problemas. En el tipo de unión Y, por ejemplo, no existe ninguna línea destacada (como sí ocurre en los otros casos). Aunque esto puede ser resuelto con otras comprobaciones adicionales al hacer la comparación, el resultado es que el coste computacional de los procesos de creación y comparación de grafos crece sensiblemente, y el estudio teórico de cuales deberían ser los costes básicos de comparación entre nodos se hace más complejo.

Además, conceptualmente este tipo de representación tiene otro problema: de un dibujo de líneas pasamos a una representación de puntos, ya que en última instancia los nodos son puntos de unión. Se puede ver, por ejemplo, que en la figura anterior la línea más a la derecha del dibujo, que es una línea sobrante debida a un error, ha provocado la introducción de dos nodos erróneos en el grafo (I_1 y T_1), y la división de una línea en dos. En el peor caso, si hay poca conectividad entre las líneas, una entrada de N líneas podría convertirse en un grafo con 2N nodos.

3.4.4.2. Grafos de aristas.

De lo anterior se puede concluir que sería más adecuado otro tipo de representación de grafo, donde los nodos se corresponderán con las líneas y no con las uniones del dibujo de líneas. A esta representación la llamaremos “grafo de aristas asociado al dibujo de líneas”, y su definición es la siguiente.

> Un grafo de aristas G asociado a un dibujo de líneas, es un grafo no dirigido formado por un par G= (Nodos, Enlaces) donde:
>
> - Los nodos representan cada una de las líneas del dibujo de líneas.
> - Un nodo *A* está enlazado a otro nodo *B* si las líneas correspondientes a *A* y *B* están unidas.

Existe una clara dualidad entre los conceptos de grafos de aristas y de vértices. Los nodos de un tipo son los arcos en el otro y viceversa. Esto da lugar a que para cierto tipo de operaciones resulte más adecuada una representación y para otro conjunto de operaciones resulte lo contrario. Sin embargo, veremos que la representación de aristas resuelve los problemas anteriores, además de resultar conceptualmente más razonable. Utilizando el mismo dibujo de líneas de la figura 16, el grafo de aristas asociado sería como el siguiente dibujo. Para facilitar su comprensión, se pintan los nodos de forma alargada.

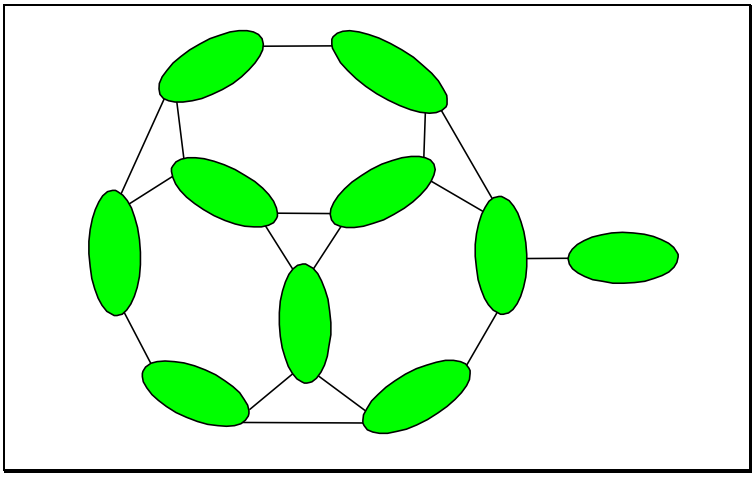

Figura 17 *Grafo de aristas asociado a un dibujo de líneas de borde*

Al contrario del tipo de grafos que aparecen en otros problemas, por ejemplo en el reconocimiento de caracteres antes visto, la estructura del grafo es un fiel reflejo de la disposición geométrica de las líneas dentro del dibujo de líneas. Dos nodos están enlazados si las líneas correspondientes en el dibujo están unidas, en otro caso no se considerará ninguna relación entre

las líneas. Aunque esto es discutible (ya que perdemos una gran cantidad de información para las líneas no unidas) esta decisión se justifica porque de estas relaciones no se deducen propiedades invariantes, al menos de forma directa.

Sin embargo, la simple información de unión o no entre líneas resulta insuficiente y, como antes, la estructura de grafo puede ser ampliada a una representación de grafo dirigido y atribuido para considerar las posiciones relativas, en función de los tipos de uniones predefinidos. De esta forma, un enlace entre dos nodos $A \rightarrow B$ podría ser etiquetado con el tipo de unión correspondiente, y las posiciones de *A* y *B* en este tipo. Por ejemplo, la etiqueta podría indicar que *A* y *B* se intersectan en una unión de tipo W, donde *A* es la línea central y *B* la de más a la derecha. Desafortunadamente, de esta forma aparecen los mismos problemas que antes, el proceso de comparación se hace más complejo computacional y conceptualmente, el tipo de nodo Y no aporta por sí mismo ninguna información de posición y se debe tratar el problema de los tipos no predefinidos. Además, aparece un problema nuevo, ya que no se diferencia si el enlace de una línea con otra es en uno u otro extremo del segmento.

3.4.4.3. Grafo de aristas dirigido y atribuido.

Todo lo anterior sugiere que se debería diseñar algún modo alternativo de representar de las posiciones relativas entre los segmentos, sin utilizar de forma explícita los tipos de uniones. Este método debería ser sencillo de crear y utilizar, pero manteniendo toda la información relevante para el proceso de comparación.

Supongamos que tenemos dos nodos, *A* y *B*, y dos arcos entre ellos, $A \rightarrow B$ y $B \rightarrow A$, indicando que los segmentos correspondientes a *A* y *B* se intersectan en un cierto punto. Adicionalmente, otros segmentos podrían pasar también por este punto. La etiqueta del arco $A \rightarrow B$ describe la unión desde el punto de vista de *A*. En la figura 18 aparece un ejemplo correspondiente a esta situación.

Empezando por lo más básico, esta etiqueta $A \rightarrow B$ debería indicar si el punto de intersección se encuentra en uno u otro extremo del segmento *A*. De alguna forma, el coste de comparación debería ser capaz de distinguir los enlaces que tiene *A* en un extremo del segmento y los que tiene en el otro. Con esto, los segmentos adquieren una nueva propiedad, que podríamos denominar orientación, según la cual uno de sus extremos se clasifica como “cola” y el otro como “cabeza”. Podemos decir que *A* se une con *B* en la cabeza.

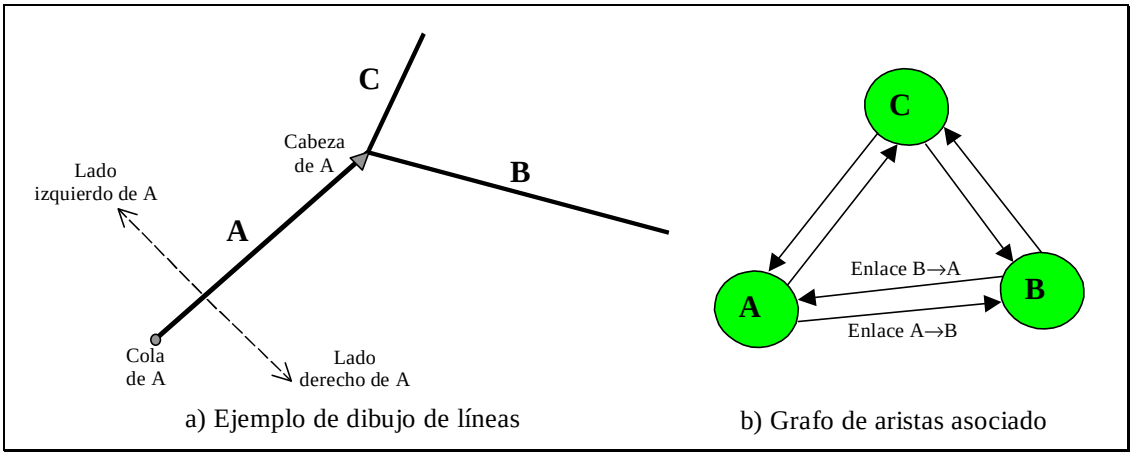

Figura 18 *Ejemplos de segmentos y posiciones relativas*

Esta propiedad de orientación facilita la definición de una nueva información de etiquetado, que se desprende de las ideas de las técnicas de etiquetado de nodos: el lado de una línea respecto de otra. Si nos fijamos en la división de tipos de uniones posibles (uniones tipo L, Y, W, T) podemos deducir que se basa en los lados relativos de las líneas: una línea está a la derecha o a la izquierda de la otra. Por ejemplo, en el tipo Y cada una de las líneas está unida a otras dos líneas, una a cada lado, mientras que en el tipo W las líneas exteriores tienen en su intersección a las otras dos

líneas a un mismo lado. En el tipo L se puede decir que una línea tiene a la otra a la derecha, y la otra tiene a la primera a la izquierda, y esto no varía aunque se giren las dos líneas a la vez.

El lado de una línea respecto de otra es realmente una información de ángulo pero discretizado de forma extrema, y fundamentado nuevamente en la búsqueda de características invariantes. Puesto que los segmentos tienen un sentido de orientación (de cola a cabeza) los lados derecho e izquierdo del segmento serán definidos en función de esta orientación.

Hasta ahora hemos visto y justificado que un arco $A \rightarrow B$ debería indicar si la intersección está en la cabeza o en la cola de A, y si B se encuentra a la derecha o a la izquierda de A. Para evitar una redundancia de información, este enlace no indicará nada sobre el resto de líneas que también pasan por ese punto (en el ejemplo, anterior este dato se tendrá en cuenta en el enlace $A \rightarrow C$).

Además de los anteriores, otros tipos de uniones entre dos líneas son posibles. Estos ocurren cuando el punto de intersección no es un extremo de uno de los segmentos, sino que está en medio de ambos. El tipo de unión T representa este caso, el segmento vertical intersecta en un punto intermedio del segmento horizontal de la T. Para tratar este caso, añadimos al conjunto de posiciones {Cola, Cabeza} una posición "Medio", y dos nuevos lados {Delante, Detrás} indicarán que un segmento tiene al otro segmento delante o detrás, extendiéndose del lado izquierdo al derecho.

Recopilando lo visto hasta este punto, tenemos que el conjunto de posibles valores de etiquetas para los arcos es el siguiente:

EtiquetasArco= {cabeza derecha, cabeza izquierda, cola derecha, cola izquierda,
medio izquierda, medio derecha, delante, detrás}

En la siguiente figura se muestra gráficamente el significado de este tipo de etiquetas. Los nodos son representados como flechas, ya que tienen la propiedad de orientación, uno de los extremos del segmento es la cabeza y el otro la cola.

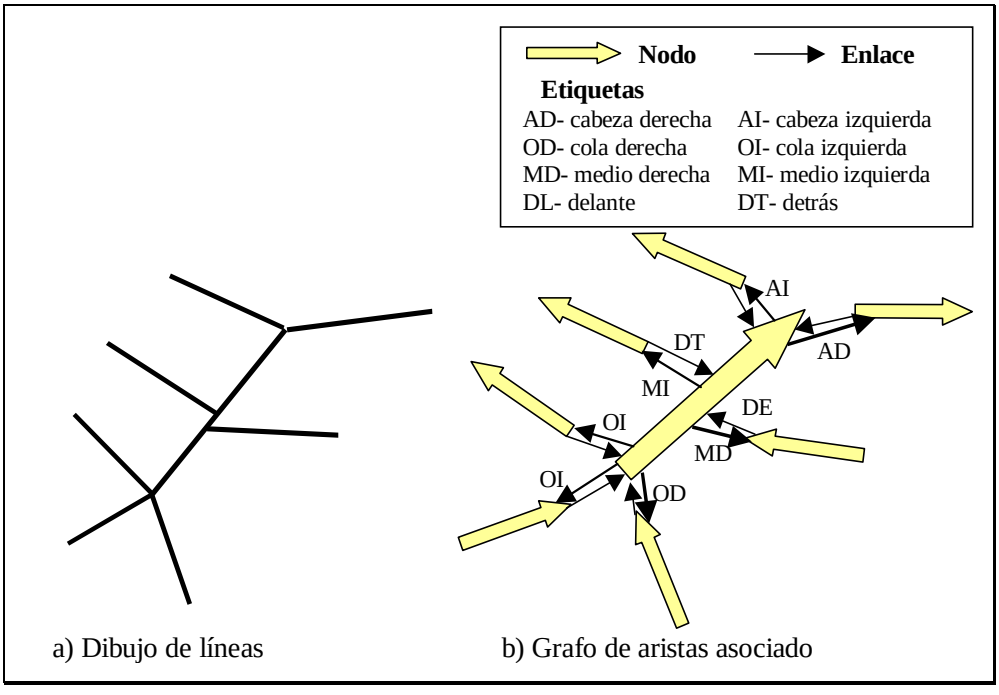

Figura 19 *Ejemplo de los posibles etiquetados de un grafo de aristas*

En resumen, a lo largo de toda la etapa de alto nivel se usará una representación mediante grafos de aristas, dirigidos y atribuidos, donde los nodos son las líneas del grafo, los arcos serán las intersecciones entre las líneas y la etiqueta de un arco indicará la posición relativa de una línea respecto de la otra.

3.4.5. Creación de grafos.

El proceso de creación de grafos puede ser considerado como un interface entre las etapas de medio y alto nivel, encargado de recoger el dibujo de líneas de borde (resultante de la detección de segmentos) y transformarlo en una representación de grafos de aristas (eliminando toda la información superflua para los procesos posteriores). Por lo tanto, está muy influido por el tipo de salidas que sea capaz de obtener la detección de segmentos y por la definición de estructura de grafos antes vista.

En caso de disponer de un dibujo preciso de las aristas de la escena, el cálculo del grafo resultaría extremadamente muy sencillo. Simplemente deberíamos añadir un nodo para cada segmento, unir cada nodo con los que intersecta y buscar después las relaciones geométricas entre los nodos unidos, etiquetando los arcos del grafo. Sin embargo, no se puede esperar que el dibujo de líneas sea una representación exacta de las aristas de la escena. En la mayoría de los casos las líneas contienen información muy ruidosa, superflua, duplicada o incluso contradictoria. Los siguientes son los tipos de errores más frecuentes que aparecen, en la práctica, en los dibujos de líneas obtenidos con escenas reales:

- Debido a una falta de contraste en los dibujos, los extremos de los segmentos quedan acortados llegando incluso a desaparecer completamente en algunos casos.
- El ruido de la imagen o los reflejos en las superficies, pueden provocar que los segmentos sean alargados, aunque la mayoría de las veces no excesivamente.
- Las aristas relativamente largas suelen ser modeladas con varios segmentos colineares y muy próximos entre sí.
- Las aristas muy cortas no son detectadas por el algoritmo de detección de segmentos. Aunque esto se puede solucionar ajustando un parámetro (la longitud mínima de las rectas), en la práctica existe un compromiso entre este problema y el anterior.
- Si existe ruido no uniforme, agrupado en ciertas zonas, el proceso tiende a introducir muchos segmentos, de forma casi incontrolada, para tratar de modelarlos. Un caso particular es la existencia de texturas, que aun no siendo ruido, provocan esta agregación de los puntos de borde en estructuras difíciles de modelar mediante segmentos.

Muchos de estos errores parten realmente de las primeras etapas del proceso de visión, de la adquisición o la detección de bordes, y se van propagando sin solucionarse al resto de etapas. Por lo tanto, la creación de grafos no debe limitarse a buscar las relaciones entre las líneas del grafo, sino que deberá solucionar los problemas anteriores, actuando en lo posible como un supresor de ruido.

Con todo esto, podemos definir la creación de grafos como un proceso de búsqueda de propiedades no accidentales. Un grupo de líneas existentes en una escena concreta, se dice que están relacionados de manera no accidental, si es poco probable que la relación observada ocurriera por causas aleatorias. La justificación para el uso de propiedades no accidentales se encuentra en el proceso visual humano [Low87]. Las personas parecen tender a hacer este agrupamiento, lo que facilita la interpretación de los dibujos de líneas. Veamos por ejemplo, la siguiente figura, donde se han añadido algunas relaciones no accidentales a un conjunto aleatorio de líneas.

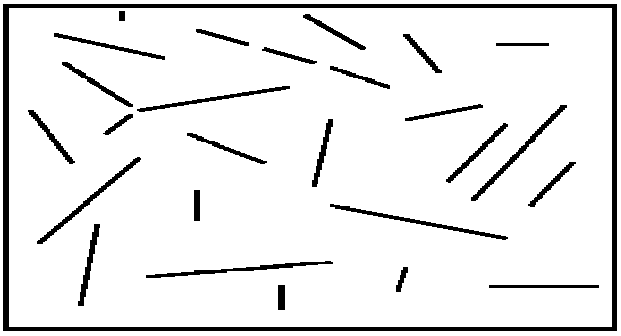

Figura 20 *Líneas aleatorias y grupos con propiedades no accidentales*

Rápidamente nuestra atención se enfoca en tres grupos de líneas: las tres líneas colineares (de arriba en el centro), las tres que casi convergen en un punto (a la izquierda) y las líneas paralelas (a la derecha). Aunque hay otras muchas relaciones entre las líneas, se les presta menos importancia ya que asumimos que se produjeron por causas aleatorias.

Los tipos de propiedades a considerar en el proceso de creación del grafo se encuentran dentro de un catálogo de propiedades no accidentales. En general, su contenido puede variar según el dominio de la aplicación. El catálogo usado para este proyecto es el mostrado en la figura 21.

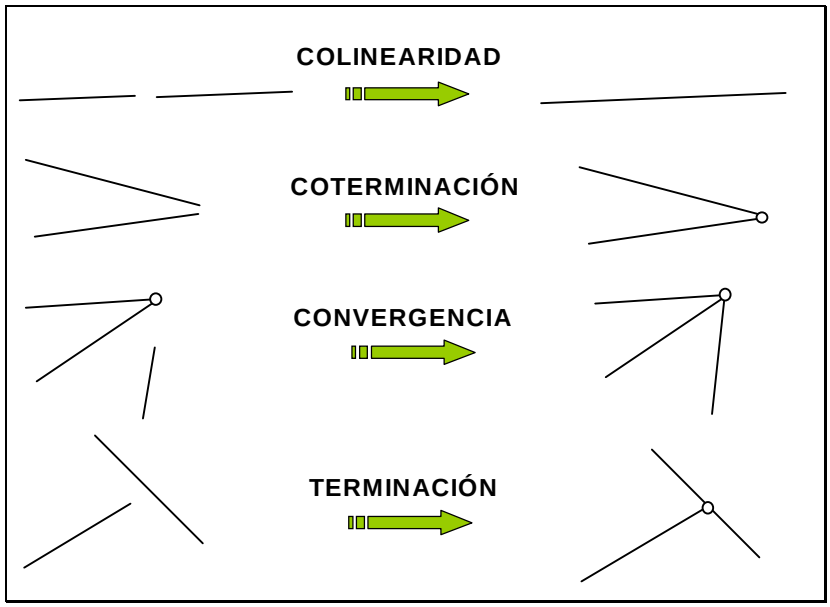

Figura 21 *Catálogo de propiedades no accidentales*

Para cada uno de los tipos de propiedades no accidentales catalogados, es necesario definir un modo de determinar su existencia y una acción adecuada en caso afirmativo. Adicionalmente, habrá que establecer un orden para la ejecución de las comprobaciones, del cual pueden depender los resultados obtenidos.

Las acciones asociadas a cada propiedad pueden ser del tipo: modificar los extremos de los segmentos, unir varios segmentos en uno, eliminar segmentos o introducir enlaces etiquetados entre los nodos. Por otro lado, la comprobación de las propiedades se basa en unas medidas tomadas sobre los segmentos y su comparación con ciertos umbrales. Estas comprobaciones deberían ser invariantes a escala y rotación.

Idealmente, el orden de ejecución de las acciones asociadas a cada propiedad, debería ser igual al grado (en sentido decreciente) en que podemos estimar que esas propiedades ocurrieron de manera no accidental (matemáticamente la probabilidad de que la propiedad sea no accidental). Es decir, deberíamos realizar primero las acciones asociadas a las propiedades que más confianza tenemos de que sean ciertas. Sin embargo, en la práctica esto resulta problemático, ya que todas las comprobaciones deberían ser realizadas al principio, después habría que ordenarlas y ejecutar sus acciones asociadas. Cada acción ejecutada puede variar todas las probabilidades, por lo que deberían ser recalculadas nuevamente.

Una estrategia más adecuada es realizar el proceso en varias fases de comprobación, donde en cada una de ellas se buscarán uno o varios tipos de propiedades determinados. Estas fases son ordenadas por las probabilidades a priori de sus tipos de propiedades correspondientes. Dentro de cada fase se realizaría un proceso iterativo de comprobación-acción entre pares de líneas, hasta que no se produzca ningún cambio. Por ejemplo, es claro que la probabilidad a priori de que aparezca una colinearidad es menor que la de una terminación, ya que hay más ángulos y posiciones posibles que den lugar a este segundo caso.

En consecuencia, la estructura del proceso de creación de grafos es la siguiente:

1. Crear un nuevo nodo en el grafo para cada segmento de entrada (asignándole una cierta orientación).
2. Comprobar la colinearidad, ejecutando las acciones correspondientes.
3. Comprobar la coterminación, ejecutando las acciones correspondientes.
4. Comprobar la convergencia y terminación, ejecutando las acciones correspondientes.
5. Heurísticas de acabado.

3.4.5.1. Propiedad de coterminación.

Dados dos segmentos, se dice que hay una coterminación si tienen un punto extremo común. Si dos segmentos coterminan en un punto, podemos inferir con cierto grado de confianza que las aristas de la escena asociadas a estos segmentos se unen en un vértice de algún objeto. En este caso, el grafo debería incluir enlaces entre esos nodos, etiquetados con las posiciones relativas de los segmentos (cabeza-cola, izquierda-derecha).

En la práctica, no podemos imponer la condición de que los segmentos tengan un mismo punto extremo, sino que la comprobación debe ser más flexible. En concreto, ¿cómo debería hacerse esta comprobación? Supongamos, por ejemplo, las líneas de la siguiente figura.

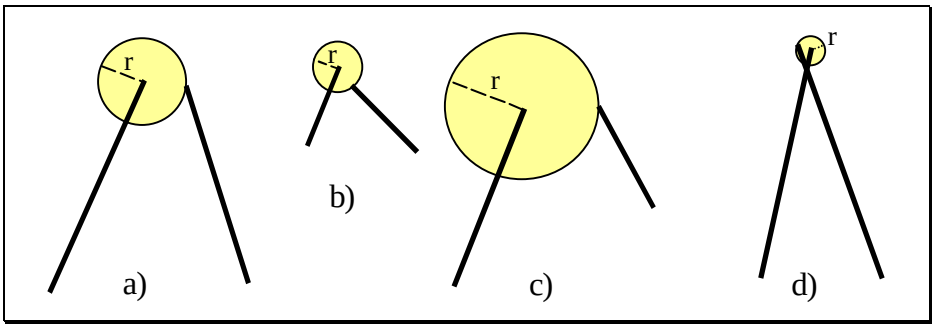

Figura 22 *Ejemplos de líneas que coterminan*

Una solución sencilla es establecer que hay una coterminación si la distancia entre los extremos de los dos segmentos *r*, es menor que cierto umbral de distancia r_{max}. Como en general esta distancia será pequeña, no nos importa que el punto de intersección esté fuera de los segmentos (como en los casos a y c) o en un punto interior a ambos (caso d). Para que el método sea invariante a escala, podríamos establecer el valor de r_{max} como una proporción de las longitudes de los segmentos implicados.

El mismo problema puede ser abordado mediante el cálculo de las probabilidades. Para medir cómo de significativa es la relación de coterminación entre dos líneas con distancia *r*, supongamos que establecemos r_{max} a *r* (todas las líneas con distancia menor que *r* coterminan). Entonces, debemos contar cuantos pares de líneas colocados de forma completamente aleatoria serían encontrados como coterminantes según esa distancia. Si el valor es muy grande, entonces lo más probable es que la coterminación entre las líneas iniciales ocurriera de manera casual.

Para calcular este número, supongamos que fijamos uno de los segmentos en una posición. Para que haya una coterminación, uno de los dos extremos del otro segmento debería encontrarse dentro de la circunferencia de radio *r*. Si la distribución de puntos es uniforme y con una densidad de puntos de extremo por unidad de área de *d*, entonces el número de extremos de segmentos que habrá dentro del área permitida será:

$$\text{Nº de puntos} = \text{Densidad} \cdot \text{Volumen} = d\pi r^2$$

Donde la densidad *d* está en función de las longitudes de las líneas existentes, y se puede calcular con la expresión: $2D/l^2$, siendo *D* una constante de densidad independiente de la escala, *l* la longitud de los segmentos y *2* el número de extremos de un segmento. Por lo tanto, podríamos definir una regla del tipo:

Si $N = 2D\pi r^2/l^2 << 1$ Entonces es improbable que la coterminación observada ocurriera accidentalmente

Teniendo en cuenta que $2D\pi$ es constante, la anterior condición $N << 1$ puede ser sustituida por $r^2/l^2 << V$, o equivalentemente por $r << P*l$. En definitiva, llegamos al mismo resultado que el obtenido en la primera aproximación al problema: se considerará que dos segmentos tienen un punto de coterminación si la distancia entre los extremos es menor que una cierta proporción *P* de la longitud de los segmentos.

En caso de encontrarse dos segmentos que coterminan, sus nodos correspondientes en el grafos serán enlazados, y etiquetados con las posiciones relativas de ambos. Además, se modificarán los puntos extremos de los segmentos para que ambos lleguen hasta el mismo punto de intersección. Estos serán los nuevos valores de los segmentos para las siguientes etapas del proceso de creación de grafos.

Para esta propiedad, el análisis probabilístico de las condiciones de no accidentalidad resulta relativamente sencillo. Hemos visto además que se llega a la misma condición que con un acercamiento más intuitivo al problema. Aunque un similar desarrollo formal podría llevarse a cabo para las demás propiedades, creemos que resulta más adecuado el acercamiento directo, que en última instancia deberá ser validado empíricamente.

3.4.5.2. Propiedad de colinearidad.

Dos segmentos se dice que son colineares si son parte de la misma línea (no siendo suficiente con que sean paralelos). Si además estos segmentos se encuentran próximos entre sí, entonces podemos sospechar que forman parte de una misma arista en la escena observada. En tal caso, el grafo debería incluir un sólo nodo para ambos segmentos, por lo que la acción asociada a esta comprobación es unir los dos nodos en uno. No obstante existe una excepción: si la aparición de los segmentos colineares fue debida a una oclusión (algún objeto tapa un trozo de la arista), estos segmentos no deberían ser unidos, ya que cortarían las líneas del otro objeto. Este caso aparece en la figura 22, donde también se muestran otros ejemplos de segmentos colineares.

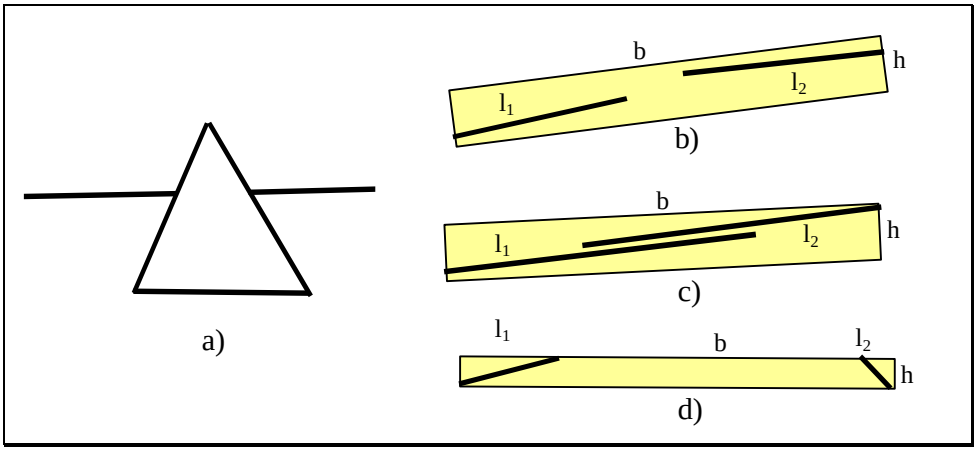

Figura 23 *Ejemplos de segmentos colineares*

El ejemplo c) representa el tipo de relaciones que más aparecen en la práctica, donde ambos segmentos comparten un mismo tramo. Igual que antes, existirá un cierto grado de confianza de que dos líneas cualesquiera sean colineares. Este grado puede ser medido en función del ángulo entre los segmentos y las distancias entre las líneas correspondientes. Un modo más adecuado, que combina las dos propiedades anteriores, es considerar el mínimo rectángulo que contiene ambos segmentos. Según este rectángulo sea más o menos achatado, podremos establecer que las líneas son más o menos colineares.

No obstante, el problema de encontrar el mínimo rectángulo que contiene cuatro puntos puede resultar excesivamente complejo para el propósito que se requiere. En su lugar, podemos

hacer una aproximación de la base *b* y la altura *h* de este rectángulo a partir de los dos autovalores de la matriz de covarianzas de los puntos extremos de los segmentos. La condición de colinearidad se basará en la proporción entre la base y la altura del rectángulo, que no deberá ser inferior a cierta cantidad. Es decir, la base debe ser varias veces más larga que la altura.

Un problema de este criterio es mostrado en el ejemplo d). Para dos segmentos pequeños y alejados entre sí, el valor de la proporción puede ser muy bueno, aun cuando resulte evidente que no existe colinearidad. Aun en caso de haber colineridad, puede ser más adecuado no unir los segmentos muy alejados. Por lo tanto, debe añadirse una nueva condición: la longitud del nuevo segmento (aproximada por *b*) no puede ser mucho mayor que la suma de las longitudes de los segmentos unidos. La diferencia de *b* menos las longitudes *l1+l2* es lo que podemos denominar "alargamiento de los segmentos", la longitud del trozo de segmento que se añade. La comprobación usará un umbral de alargamiento máximo, que será una proporción de las longitudes de los segmentos originales. Unido con lo anterior, el criterio de colinearidad será:

Si (b/h<Umbral_colinearidad) y (b<(l1+l2)*Maximo_alargamiento) Entonces
es improbable que la colinearidad observada ocurriera accidentalmente

Si una colinearidad es encontrada, el segmento que sustituye a los dos segmentos originales debería modelar longitudinalmente el rectángulo mínimo que los contiene. Puesto que este rectángulo no es calculado de forma explícita, se puede adoptar una solución más sencilla, aunque aproximada, que consiste en colocar los extremos del nuevo segmento en el par de puntos más alejados, de los cuatro existentes.

3.4.5.3. Propiedades de convergencia y terminación.

Al contrario que las propiedades anteriores, estas presentan una gran probabilidad a priori de ocurrencia accidental, por lo que deben ser tratadas de forma especial. Por un lado, si un segmento está unido a otro por un extremo, entonces no será necesario comprobar estas propiedades para ese extremo ya unido. Por otro lado, el orden en que se realicen las acciones asociadas a estas propiedades puede ser muy importante, por lo que será necesario calcular todas las probabilidades de ocurrencia no accidental para empezar por la de mayor valor.

Decimos que dos o más segmentos convergen si tienen un punto de intersección no interior a ambos, y un segmento termina en otro cuando uno de sus extremos es un punto intermedio del otro segmento. En la siguiente figura se muestran algunos ejemplos de estos tipos de propiedades.

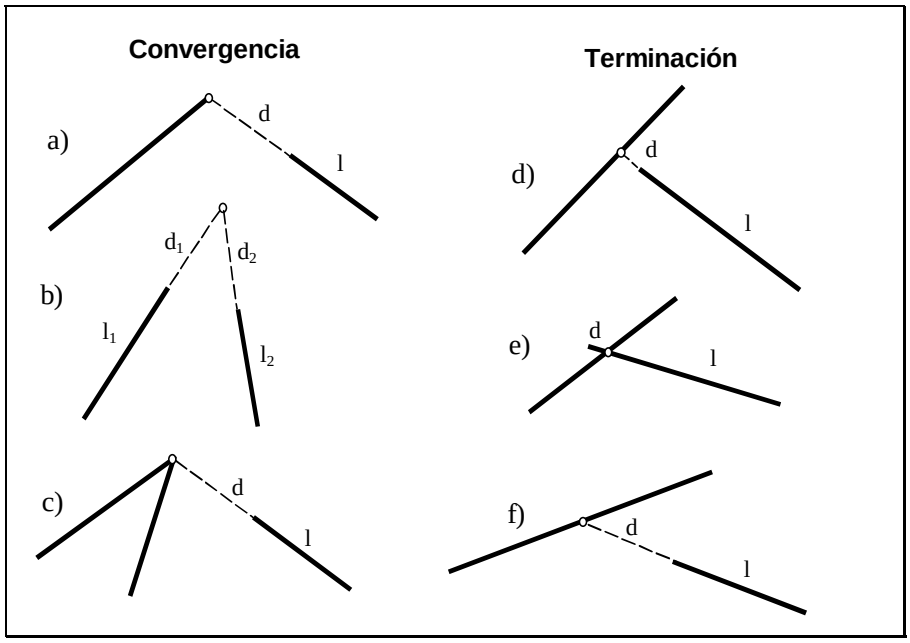

Figura 24 *Ejemplos de convergencia y terminación*

La convergencia es bastante parecida a la coterminación, pero a diferencia de esta, las distancias en que se pueden alargar las líneas pueden ser proporcionalmente mucho mayores. Además esta distancia es medida con relación al punto de intersección entre las líneas, y no entre los puntos extremos de los segmentos. Una situación especial es el caso c), donde un segmento converge en un punto en el que ya hay una unión (por ejemplo, por coterminación). Esto reforzaría la probabilidad de que la propiedad no haya ocurrido accidentalmente, pero hay que tener en cuenta que los extremos de los segmentos ya unidos no deberían ser modificados.

La terminación es el tipo de relación que da lugar a las uniones de tipo T, un segmento acaba en un punto intermedio de otro. El segundo no sufre modificaciones, pero el alargamiento del primero puede ser relativamente grande. Puede darse incluso, el caso de que este segmento deba ser acortado, como ocurre en e), aunque normalmente en una menor proporción.

En ambos casos, existe una distancia *d* que indica cuanto se deberían alargar (o acortar si es negativo) los segmentos para que llegaran a unirse realmente. La probabilidad de que una de estas relaciones haya ocurrido de forma no accidental depende inversamente de esta distancia, tomada en proporción de la longitud del segmento correspondiente.

Por lo tanto, el proceso de comprobación consistirá en buscar, para cada segmento con un extremo libre, la relación de las anteriores que ocurre a una mínima distancia *d*. Si la distancia es menor que el alargamiento máximo, entonces alargar el segmento (es decir, cambiar su punto extremo por el punto de intersección), cambiando también en el caso b) el punto extremo del otro segmento. Además se deberán crear enlaces entre los nodos, etiquetándolos con los valores adecuados.

Igual que vimos antes, el máximo alargamiento de un segmento es una proporción de la longitud de ese segmento. Esta proporción puede ser distinta para la convergencia y la terminación. Además, en este segundo tipo resultaría adecuado definir una proporción de acortamiento máximo, para el caso en que la distancia *d* sea negativa.

3.4.5.4. Heurísticas de acabado.

En el último paso del proceso de creación de grafos se comprueban una serie de propiedades de más difícil catalogación, cuyo contenido depende del uso específico que se vaya a hacer del grafo. Por su propia naturaleza, estas propiedades deben ser verificadas después de haber aplicado las estudiadas hasta ahora. En su mayoría son reglas heurísticas que se basan en cuestiones prácticas y que tratan de evitar situaciones excepcionales de ruido o inconsistencias. Las heurísticas definidas en este proyecto para la creación de grafos son las siguientes.

Eliminación de segmentos aislados.

Si después de aplicar todas las etapas anteriores queda algún nodo que no esté conectado con ningún otro, entonces podrá ser eliminado del grafo. Visto como un segmento, el nodo aporta una información de posición en el dibujo que no puede ser despreciada. Sin embargo, para el proceso de comparación la existencia de este nodo es completamente indiferente al no estar relacionado con otros nodos.

En la mayoría de los casos se trata de líneas debidas a ruido, que pueden ser eliminadas sin ningún inconveniente posterior. Aunque correspondieran a aristas del dibujo, al no incluir ninguna relación la comparación de estos nodos no tendría ningún valor, y sólo podría dar lugar a un aumento del tiempo de ejecución.

División de segmentos que se cortan.

Si el punto de intersección de dos segmentos es interior a ambos, entonces decimos que se cortan. Este tipo de relación no debería aparecer nunca, utilizando objetos poliédricos y bajo la suposición del punto de vista generalizado. Por ello, no se incluía en los tipos de posibles relacio-

nes entre dos segmentos. Sin embargo, en la práctica es muy posible que ocurra, provocando problemas que deberán ser resueltos.

La solución consistirá en dividir los dos segmentos originales en cuatro nuevos segmentos, a partir del punto de corte. Para estos nodos se añadirán relaciones de unión, estableciendo los valores de lado (izquierdo o derecho) de forma arbitraria, para los pares de segmentos que resultan de la división de un mismo nodo. Realmente necesitaríamos un lado "enfrente de", que no ha sido considerado por tratarse de una situación excepcional.

Eliminación de segmentos con varios cortes.

En algunas ocasiones, el algoritmo de detección de segmentos añade segmentos que recogen puntos de ruido, por lo que su disposición puede ser totalmente imprevisible. Algunos de estos segmentos erróneos cortan uno o varios de los segmentos correctos. En caso de provocar un sólo corte no podemos saber si el segmento es de ruido y cual de los dos es (en caso de serlo).

Si el segmento corta dos o más líneas, podemos sospechar que es un segmento erróneo, pues como hemos visto, la existencia de un simple corte para segmentos correctos es considerada como muy improbable. En consecuencia, estos nodos serán eliminados del grafo. Esto no impedirá la existencia de errores, pero se evitará que varios segmentos correctos sean divididos, provocando otros errores posteriores.

3.4.6. Comparación de grafos.

Una vez con el grafo creado a partir de la escena de entrada, el siguiente paso dentro del proceso de alto nivel es la comparación de grafos. El objetivo de esta etapa es la asignación de uno o varios conjuntos de segmentos del dibujo de líneas, a uno o varios modelos almacenados en la base de datos del problema. Generalmente una escena puede constar de varios objetos.

Este proceso está basado en la resolución del problema más simple de comparar dos grafos, repetido iterativamente para todos los modelos. En base a estos conjuntos de segmentos asignados a modelos, se formulará la hipótesis de interpretación: la escena está formada por varios objetos, cuyo tipo, posición y orientación viene determinada por la equivalencia de los segmentos con los modelos. Esta hipótesis será después verificada de una forma geométricamente más precisa, mediante un alineamiento.

Primero estudiemos el problema básico de comparación de dos grafos. Después veremos como este algoritmo puede ser utilizado para el problema de la interpretación, aunque esto no ha sido implementado completamente en este proyecto.

3.4.6.1. El problema de la comparación de grafos.

Evidentemente, la comparación de dos objetos representados con grafos de aristas es más simple que si pretendiéramos hacerla sobre las imágenes de entrada o sobre los dibujos de borde, más aun dentro de un dominio tridimensional. Aun así, el problema de comparación de grafos es un problema NP-completo, de elevada complejidad computacional, por lo que será necesario llegar a un compromiso entre velocidad y exactitud de la solución. Muchos métodos han sido diseñados para la resolución de este problema, entre los cuales podemos encontrar dos grandes acercamientos: la búsqueda en un espacio de estados y los métodos de optimización no lineal. Cada una de las técnicas implica un cierto compromiso entre exactitud y tiempo de ejecución.

Adicionalmente, el método de comparación debería tolerar cierto nivel de ruido, devolviendo para cada par de grafos un coste de comparación y una lista de correspondencias entre nodos de uno y otro grafo. Este coste expresa el grado de similitud entre los dos grafos, y puede venir dado en función de los segmentos que habría que añadir, eliminar o modificar para conseguir que ambos grafos sean iguales. Ya hemos visto que el verdadero problema es un subisomorfismo de grafos, por lo que lo anterior más que una definición precisa del coste será considerado como una idea para el cálculo de los costes subyacentes al proceso, los de asignación de un nodo a otro.

3.4.6.2. Comparación por búsqueda en el espacio de estados.

Supongamos el siguiente caso de ejemplo, donde se desea resolver el problema de comparación de grafos, independientemente de que los arcos sean o no etiquetados.

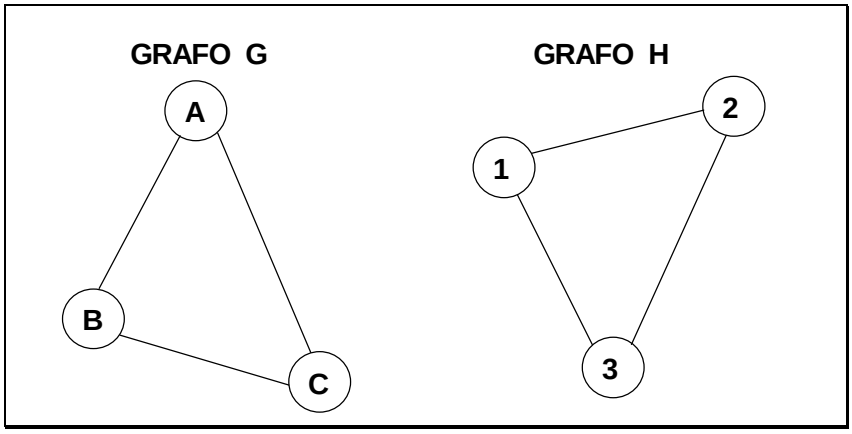

Figura 25 *Ejemplo de grafos para un problema de comparación de grafos*

El primero de los acercamientos implica la construcción de un espacio de estados, dentro del cual se realizará una búsqueda de la solución. En este caso, el espacio de estados tendría forma de árbol, donde cada nodo contiene una asignación de un nodo de G con otro de H, dándose por supuesta esta asignación en el subárbol correspondiente. Se podrán aplicar técnicas de ramificación y poda. La siguiente figura muestra un posible espacio de estados para el problema. La asignación de un nodo *X* del grafo G con otro nodo *Y* del grafo H es representada como *(X, Y)*.

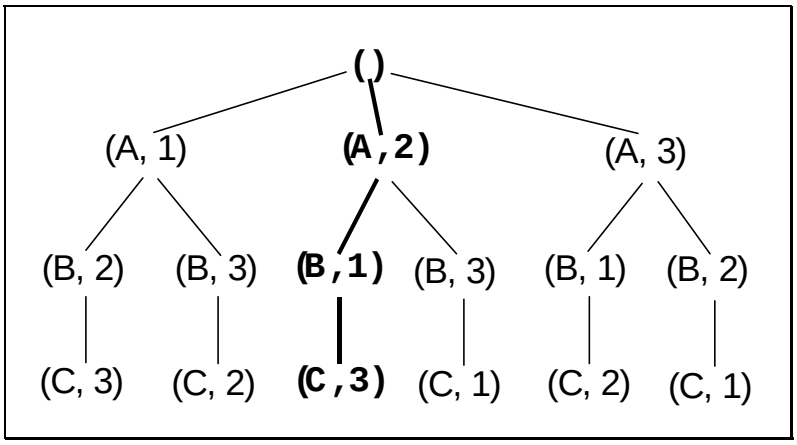

Figura 26 *Espacio de estados en la comparación de grafos*

Cualquier nodo hoja representa una posible solución. El resultado viene dado por el conjunto de asignaciones contenidas en ella misma y en todos sus nodos antecesores hasta la raíz del árbol. Por ejemplo, el nodo hoja *(C, 3)* en negrita corresponde a la solución que asigna *A* con *2*, *B* con *1* y *C* con *3*. Algún valor de coste debería ser introducido para encontrar cual es la mejor solución y para poder aplicar la poda del árbol.

En la anterior figura se puede ver claramente el problema de la complejidad exponencial de este acercamiento. A pesar de que se han aplicado algunos sencillos criterios para reducir el número de nodos del árbol (expandir en cada nivel las asignaciones de un nodo de G y no repetir en los subárboles los nodos de H ya asignados), el algoritmo expande un número factorial de nodos. Aunque las técnicas que usan este acercamiento suponen que con la ayuda de heurísticas el número de nodos puede ser reducido a un orden polinómico, en la práctica estos órdenes están aproximadamente sobre $O(l^3m^2)$, donde l y m son el número de arcos de los dos grafos. El problema se hace aun más costoso si consideramos el isomorfismo de subgrafos.

3.4.6.3. Comparación mediante técnicas de optimización no lineal.

Dentro de las técnicas de optimización no lineal, se abandona la idea de búsqueda en un árbol. Una gran variedad de métodos han sido diseñados en este acercamiento, haciendo uso de redes neuronales, programación lineal, algoritmos genéticos o relajación de etiquetado (en inglés, *relaxation labeling*). Este último es uno de los que obtienen mejores resultados, siendo además de baja complejidad computacional, *O(lm)*.

Conceptualmente, los métodos de relajación de etiquetado parten de las ideas de las técnicas de etiquetado de nodos, vistas en 3.4.2.1. En este caso, se considera que un nodo puede tener a la vez varias etiquetas, con distintas probabilidades de que sean correctas. De forma iterativa, estas probabilidades son recalculadas para cada nodo, según las restricciones definidas en los grafos, hasta llegar a una solución estable. Esta solución será un mínimo local, que dependerá de la inicialización aleatoria de las probabilidades.

La comparación de grafos puede ser realizada utilizando una relajación de etiquetado. De los dos grafos a comparar, uno será el modelo y el otro la escena que se desea identificar. El problema consistirá en etiquetar los nodos de la escena, donde las etiquetas harán referencia a cada uno de los nodos del modelo. Las restricciones y los cálculos de las probabilidades son definidos en función de los arcos de cada nodo con el resto de nodos. Nótese que hay una clara distinción entre las etiquetas de los arcos (calculadas en la creación del grafo) y las de los nodos (usadas sólo para realizar la comparación).

Las probabilidades de etiquetado de cada nodo de la escena con todos los nodos del modelo pueden ser representadas de forma tabular. Para el caso anterior, podríamos tener algo como lo siguiente en un cierto paso de ejecución del algoritmo.

P(X|i)	Nodo 1	Nodo 2	Nodo 3
Nodo A	0.10	0.80	0.10
Nodo B	0.60	0.20	0.20
Nodo C	0.10	0.15	0.75

Cada fila representa la probabilidad de que un nodo de la escena (A, B ó C) tenga las distintas etiquetas de nodos del modelo (1, 2 ó 3). Por lo tanto, todas las filas deben sumar 1. Sin embargo, esta condición no se impone sobre las columnas, ya que el etiquetado es sobre nodos de la escena, no sobre el modelo. De esta forma, varias ocurrencias de un modelo podrían aparecer en una misma escena.

Una mejora de este método para el problema específico de la comparación de grafos es el algoritmo de asignación graduado, propuesto en [Gol96]. Igual que la relajación de etiquetado, no realiza ninguna búsqueda en el espacio de estados y tiene un bajo orden de complejidad computacional de *O(lm)*. Además, se añade la condición de que la asignación de nodos sea unívoca (en el ejemplo anterior, que las columnas sumen 1) y se incorpora un método para evitar que el proceso caiga en un mínimo local.

3.4.6.4. El algoritmo de asignación graduado.

El algoritmo de asignación graduado para comparación de grafos, de S. Gold y A. Rangarajan, se encuentra dentro del grupo de técnicas de optimización no lineal. Puede ser visto, en cierto sentido, como una mejora del método de relajación de etiquetado en problemas de comparación de grafos, pero su verdadero interés es que se replantea el problema desde un nuevo punto de vista. No realiza ninguna distinción explícita entre el grafo de la escena y el del modelo, sino que ambos grafos juegan el mismo papel en el algoritmo.

Supongamos que tenemos dos grafos que queremos comparar, por ejemplo como los de la figura 25. El problema que queremos resolver consiste en asignar cada nodo del grafo G con un nodo del grafo H. Obviamente, la relación es reflexiva, si un nodo *a* del grafo G está asignado a *i* en el grafo H, entonces *i* está asignado a *a*. Esta asignación entre nodos puede ser descrita mediante

una matriz de asignación *M(a, i)*, de manera que *M(a,i)=M(i, a)*= 1 si los nodos *a* e *i* están asignados entre sí, y *M(a, i)*= 0 en otro caso. Con esto, las soluciones al problema de comparación de grafos serían del tipo:

M(a, i)	Nodo 1	Nodo 2	Nodo 3
Nodo A	0	1	0
Nodo B	1	0	0
Nodo C	0	0	1

Al contrario que antes, existen restricciones en los dos sentidos: sólo puede haber un uno por fila y por columna. El algoritmo debería "colocar" estos unos de forma que se optimice una cierta función de coste *C(a, i)*, de cada nodo *a* con cualquier otro nodo *i*. Estos *C(a, i)* pueden expresar un grado de compatibilidad entre los nodos (basada en la compatibilidad de los segmentos correspondientes) o bien una medida del coste de asignar ambos nodos (entendida como coste bajo para nodos equivalentes).

C(a, i)	Nodo 1	Nodo 2	Nodo 3
Nodo A	C_{A1}	C_{A2}	C_{A3}
Nodo B	C_{B1}	C_{B2}	C_{B3}
Nodo C	C_{C1}	C_{C2}	C_{C3}

Por lo tanto, se tratará de minimizar (o maximizar, según los *C(a, i)* sean tomados como costes o como compatibilidades) una determinada función objetivo. Esta función es una medida del coste total de la asignación, calculado mediante la fórmula:

$$E_g(M) = \sum_{a \in G} \sum_{i \in H} M(a, i) \cdot C(a, i) \qquad (1)$$

Definido de esta manera, hemos reducido la comparación de grafos a un tipo de problemas conocidos como "el problema de asignación". Existen técnicas de programación lineal capaces de obtener de forma eficiente la solución óptima para el problema de asignación. Sin embargo, estas no pueden ser aplicadas a nuestro caso (al menos de forma directa) ya que requieren que los costes *C(a, i)* sean constantes, lo cual veremos que no se cumple en la comparación de grafos.

La definición de los costes *C(a, i)* es un aspecto básico del algoritmo, ya que determina cuando dos nodos serán considerados o no como equivalentes. Este problema estaba subyacente en el resto de etapas del proceso de visión, llegando a la conclusión de que dos nodos sólo deberían ser comparados en función de las propiedades que permanecían invariantes en todas las realizaciones de los objetos. Vimos también que lo importante eran las propiedades estructurales, es decir aquellas que relacionan varias partes. Ahora se trata de determinar la forma de la función de costo, para cumplir con estos requisitos.

En el artículo [Gol96], Gold y Rangarajan proponen que la función objetivo de la comparación debería ser de la forma:

$$E_{wg}(M) = -\frac{1}{2} \sum_{a \in G} \sum_{i \in H} \sum_{b \in G} \sum_{j \in H} M(a, i) \cdot M(b, j) \cdot C'(a, i, b, j) \qquad (2)$$

Donde *C'(a, i, b, j)* es el coste de asignar a la vez los nodos (*a*, *i*) y (*b*, *j*). Visto desde el punto de vista de un problema de asignación, podemos igualar esta expresión con la fórmula (1), obteniendo que la función del coste de comparación de nodos *C(a, i)* sería del tipo:

$$C(a, i) = \sum_{b \in G} \sum_{j \in H} M(b, j) \cdot C'(a, i, b, j) \qquad (3)$$

En los mejores de estos costes es donde deberíamos "colocar" los unos (uno por fila y por columna) en las posiciones correspondientes de la matriz *M(a, i)*. En el artículo, los autores llegan a la misma conclusión (la reducción del problema a un problema de asignación) mediante una descomposición en series de Taylor. Veamos ahora cual es el significado de la fórmula (3).

En un problema de asignación típico, el coste de asignar un nodo *a* a otro nodo *i* depende exclusivamente de las características de ambos nodos. En este caso sin embargo, según la fórmula anterior el costo de la asignación *(a, i)* depende de todos los demás nodos de ambos grafos. Este costo se define a través a una función de costo básica *C'(a, i, b, j)*, que mide el grado de compatibilidad o similitud del arco *a→b*, en el grafo *G*, con el arco *i→j* en el grafo *H*. Por lo tanto, *C'(a, i, b, j)* puede ser interpretada como un coste de comparación de enlaces (más que de simples nodos). Estos costes están multiplicados por *M(b, j)*, haciendo que el coste *C'(a, i, b, j)* sólo se tenga en cuenta si en la matriz de asignación actual *b* y *j* están emparejados.

En definitiva, el coste de asignación del nodo *a* al *i* depende de los enlaces de *a* con otros nodos *b* (en el grafo *G*) y de los enlaces de *i* con otros nodos *j* (en el grafo *H*), para los nodos *b* y *j* emparejados en la solución actual (para los cuales *M(b, j)*=1). En la siguiente figura se muestra gráficamente el significado de estos costes. Para mayor claridad se ha modificado la notación de la función *C'*, tomando la forma *C'(Enlace1, Enlace2)*.

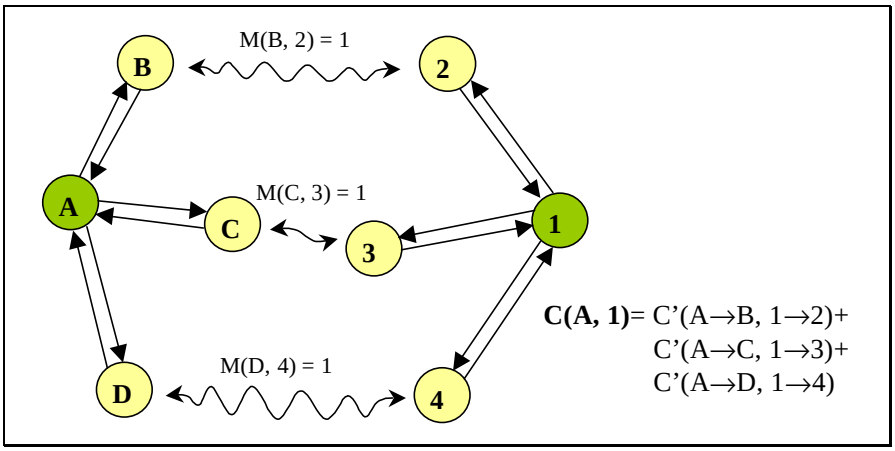

Figura 27 *Ejemplo de valores en un punto de ejecución del algoritmo*

Con la definición de los costes realizada, resulta clara la doble dependencia existente: la matriz *M* de asignación se calcula en función de la matriz de costes *C*, pero a su vez la matriz *C* es calculada según la asignación actual en *M*. Este es el motivo por el que, como comentamos antes, no se pueden aplicar algoritmos de asignación tradicionales. En su lugar, se procederá de forma parecida a la relajación de etiquetado: la matriz *M* de asignación podrá tomar valores en el intervalo [0, 1], indicando el grado de confianza de la asignación para cada par de nodos, en un momento concreto del algoritmo. De esta forma, todos los nodos de G estarán emparejados en un cierto grado con todos los nodos de H.

El algoritmo de asignación graduado, en su forma más simplificada, tendrá la siguiente estructura:

1. Inicialización aleatoria de M.
2. Repetir hasta que haya convergencia.
    2.1. Calcular C según los valores de M.
    2.2. Calcular M según los valores de C.
3. Obtener la solución.

Inicialmente, se asignan valores aleatorios a la matriz *M*, para evitar problemas de simetría en los grafos. Después se repite el paso 2 hasta que se cumpla una cierta condición de convergencia. En este paso se realizan sucesivos cálculos de *C* según *M* y viceversa. En el paso 2.1, para calcular *C* según el valor actual de *M* se utiliza la fórmula (3).

El cálculo de *M* según los valores de *C* (en el paso 2.2) debería ser tal que optimizara la función objetivo (1), lo cual se conseguiría "colocando" los unos, por filas y por columnas, en los lugares correspondientes a las mejores posiciones de *C*. Supongamos, por ejemplo, una sola fila de la matriz *C*. Deberíamos buscar el máximo valor de esa fila y poner en *M* un uno en esa posición, resultando algo como lo siguiente.

Valores de C(a, ·)				
1.5	2.5	1.0	7.3	2.5
Valores de M(a, ·) usando el máximo				
0	0	0	1	0

Sin embargo, esto presenta un grave problema, ya que se pierden los grados de asignación y la matriz *M* toma valores {0, 1} en la primera ejecución del paso 2.2. El resultado es que, con toda probabilidad, el algoritmo se estancaría rápidamente en una mala solución de óptimo local. Para evitarlo, se utiliza una técnica llamada "asignación soft" (en inglés *softassign*) consistente aplicar una función sobre *C*, y después normalizar por filas y por columnas. Con ello, tenemos una especie de máximo, pero que asigna valores reales en el intervalo [0, 1] según el número esté más o menos próximo al máximo.

La función aplicada inicialmente sobre *C*, para hacer la asignación soft, es del tipo:

$$M(a, i) = e^{\beta C(a, i)} \qquad (4)$$

Si β es grande, los valores máximos serán proporcionalmente mayores. En el límite, cuando β tiende a infinito el máximo soft tiende a la función máximo. Por otro lado, si β es pequeño las diferencias entre los máximos y los mínimos se reducen, y cuando β valga 0 el valor de la función será el mismo para todos los *C(a, i)*. Después de aplicar esta función, se realizarán sucesivas normalizaciones por filas y por columnas en la matriz *M*, hasta conseguir que todas las filas y todas las columnas sumen 1. Está demostrado que si todos los valores son positivos, esta simple repetición de normalizaciones hará que el resultado converja a una matriz donde todas las filas y columnas suman 1. En caso de tomar valores enteros esto es lo que se llama una matriz de permutación.

El parámetro β del máximo soft, puede verse como un controlador de la temperatura (relacionado con el grado de "convexificación" de la función objetivo). Inicialmente será pequeño, irá aumentando sucesivamente y al final será un valor grande, haciendo que la solución se asemeje a una matriz de permutación binaria. Por lo tanto, la condición de convergencia incluirá que β sea grande y que el valor de *M* no varíe significativamente.

Finalmente, en el paso 3 se obtiene la solución definitiva, a partir del valor de la matriz *M* final. Una forma sencilla de obtenerla es crear una lista de asignaciones con los pares de nodos cuyas asignaciones *M(a, i)* presentan el mayor valor, por filas o por columnas.

En definitiva, el algoritmo de asignación graduado tiene la siguiente estructura:

1. Inicializar β a $\beta_{inicial}$. Inicializar M(a, i) de forma aleatoria.
2. Repetir hasta que $\beta > \beta_{final}$.
    2.1. Repetir hasta que M converja, o se ejecuten más de I_0 iteraciones.
        2.1.1. Calcular: $C(a, i) := \Sigma\Sigma\, M(b, j) \cdot C'(a, i, b, j)$
        2.1.2. Calcular: $M(a, i) := e^{\beta C(a, i)}$
        2.1.3. Repetir hasta que M converja, o se ejecuten más de I_1 iteraciones.
            2.1.3.1. Normalizar M por filas.
            2.1.3.2. Normalizar M por columnas.
    2.2. $\beta := \beta$*Incremento_β;
3. Obtener la solución.

Los parámetros I_0 e I_1 restringen el número máximo de iteraciones, en los pasos que requieren una convergencia local. Los valores de $\beta_{inicial}$ y β_{final} indican los valores entre los cuales se moverá el parámetro de control de la asignación soft, β. Es aconsejable que β_{final} sea suficientemente

alto para que los valores finales de *M* sean siempre próximos a 0 ó 1. Por otro lado, *Incremento_β* indica la velocidad de ejecución del algoritmo, la velocidad a la que aumentará β.

3.4.6.5. Coste de comparación de enlaces.

Como hemos visto, los costes de asignación entre dos nodos son calculados a través de los enlaces existentes para esos nodos. En consecuencia, todo el proceso de comparación parte del coste básico de comparación entre enlaces *C'*. Dados dos nodos de un mismo grafo, puede que exista o no un enlace entre ellos. En caso de existir, puede ser etiquetado con un valor, indicando alguna información relevante sobre los nodos unidos.

En este proyecto, los grafos de aristas sobre los que se aplicará el algoritmo de comparación, son grafos dirigidos y atribuidos, en los cuales la etiqueta de un arco $A \rightarrow B$ indica la posición relativa del segmento *B* respecto de *A*. El coste de comparación de enlaces definirá un grado de compatibilidad entre los distintos tipos de enlace. En total, existen ocho posibles tipos de etiquetas, más la posibilidad de inexistencia de enlace entre los nodos. Estos valores son representados en la siguiente figura.

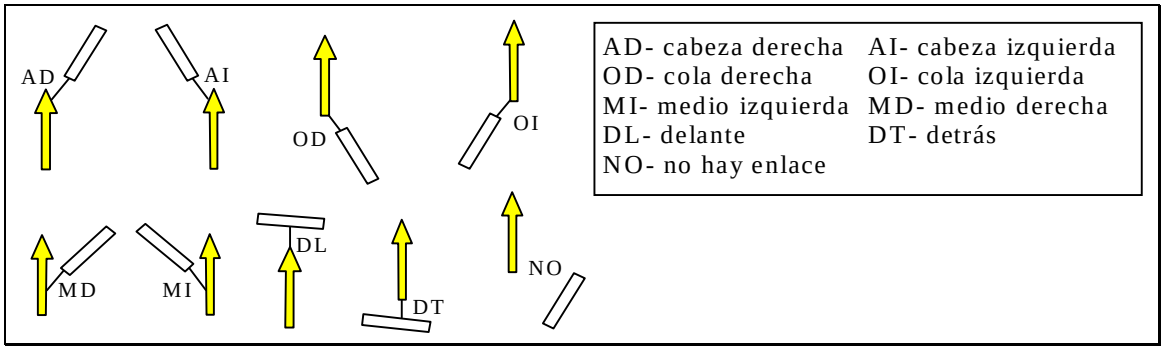

Figura 28 *Posibles relaciones entre los nodos*

Una posible opción, consistiría en definir el coste de comparación entre enlaces como el número de segmentos que hay que añadir, eliminar o modificar para que los dos enlaces sean iguales (en el dibujo de líneas correspondiente). Nótese que la modificación es un caso especial, ya que el sentido más amplio consistiría en una eliminación más una inserción de una nueva línea. Consideraremos que la modificación sólo permite un alargamiento o acortamiento de los extremos de un segmento.

De esta forma, los costes estarán entre siempre entre 0 (los enlaces son exactamente iguales) y 2 (un segmento debe ser eliminado y después insertado en la posición correcta). En la mayoría de los casos, esta definición se puede aplicar de forma sencilla para obtener los costes entre los tipos predefinidos de enlaces. Sin embargo, esto resulta más complejo en caso de no existir un enlace entre los nodos, ya que no tenemos ninguna información de posiciones relativas. Es más, entre los segmentos correspondientes podrían existir otros segmentos en el dibujo, por lo que el coste sería mayor de 2. Salvando este problema, los costes básicos de comparación entre enlaces son los siguientes.

AD	AI	OD	OI	MD	MI	DL	DT	NO	C'
0	2	2	2	1	2	1	2	≥1	AD
	0	2	2	2	1	1	2	≥1	AI
		0	2	1	2	2	1	≥1	OD
			0	2	1	2	1	≥1	OI
				0	2	1	1	≥1	MD
					0	1	1	≥1	MI
						0	2	≥1	DL
							0	≥1	DT
								0	NO

En esta tabla, los valores positivos indican coste. Sería necesario cambiar los signos para aplicar el algoritmo, tal y como ha sido definido.

Una gran parte de los costes de comparación entre enlaces contiene el valor 2, indicando el máximo grado de incompatibilidad entre enlaces. Sólo con los tipos de enlaces que ocurren en las uniones T (los tipos MD, MI, DL y DT) aparecen costes con valor 1. Un enlace de estos tipos, tendrá más probabilidad de quedar asignado a otros enlaces, ya que sus costes son menores. Esto es debido a que, acortando o alargando alguno de los dos segmentos de la T, el enlace puede convertirse en muchos otros tipos de enlaces.

En la práctica, puede que no sea deseable una mayor compatibilidad para estos enlaces, por lo que también podríamos asignar a estos costes el valor 2. Empíricamente, una definición de coste que funciona bastante bien es la siguiente:

$$C'(\text{Enlace1}, \text{Enlace2}) = \begin{cases} C_1 & \text{Si Enlace1} = \text{Enlace2} \\ C_2 & \text{Si Enlace1} \neq \text{Enlace2 y Enlace1,2} \neq \text{NO} \\ C_3 & \text{En otro caso} \end{cases}$$

Si los enlaces son iguales, entonces tenemos un coste bueno C_1. Si son distintos pero ambos distintos de "no enlace", entonces el coste es malo, C_2. Por último, si en un grafo existe enlace y en el otro no, habrá otro coste C_3. Por ejemplo, un conjunto de valores que, en general, da buenos resultados es $C_1 = 2$, $C_2 = -1$ y $C_3 = 0$.

En la definición de los grafos vimos la necesidad de introducir una propiedad de orientación en los segmentos. Todos los tipos de enlaces definidos, y utilizados ahora, están en función de esta orientación. Sin embargo, resulta claro que el coste de comparar dos dibujos de líneas debería ser independiente de la orientación de los segmentos, ya que esta es una propiedad asignada de forma completamente arbitraria, por necesidades internas del proceso. Los segmentos por sí mismos, carecen de la propiedad de orientación, y no existe ninguna forma de asignar orientaciones de forma universal y adecuada a las necesidades del proceso de comparación.

Por lo tanto, necesitamos deshacer el cambio de alguna manera, consiguiendo que dos segmentos equivalentes (en la escena y en el modelo) tengan un buen coste de comparación independientemente de las orientaciones que les hayan sido asignadas. Supongamos que queremos hallar el coste de asignación de dos nodos *C(a, i)*, sin considerar las orientaciones, pero utilizando alguno de los costes de comparación de enlaces *C'(Enlace1, Enlace2)*, antes definidos. El valor de *C(a, i)* puede ser calculado de la siguiente manera:

1. Utilizando la fórmula:

$$C^1(a,i) = \sum_{b \in G} \sum_{j \in H} M(b,j) \cdot C'(\text{Enlace}(a,b), \text{Enlace}(i,j))$$

Obtenemos un coste C^1 que considera cierta equivalencia entre los puntos extremos del segmento *a* y los correspondientes en *i*.

2. Definimos una función *Opuesto*, que para cada tipo de enlace $a \rightarrow b$, devuelve el tipo considerando que el segmento *a* hubiera sido tomado en el sentido contrario. Aplicamos otra vez la fórmula pero haciendo uso de esta función:

$$C^2(a,i) = \sum_{b \in G} \sum_{j \in H} M(b,j) \cdot C'(\text{Opuesto}(\text{Enlace}(a,b)), \text{Enlace}(i,j))$$

Ahora C^2 considera la equivalencia contraria entre los extremos de los segmentos.

3. Combinando los dos valores anteriores con un máximo, tenemos el valor del coste de comparación entre nodos independiente de orientaciones (ya que se toma la mejor de las dos posibles equivalencias de extremos). Por lo tanto:

$C(a, i) = \max (C^1(a, i), C^2(a, i))$

Los valores de la función *Opuesto* son los siguientes:

Enlace	AD	AI	OD	OI	MD	MI	DL	DT	NO
Opuesto	OI	OD	AI	AD	MI	MD	DT	DL	NO

El inconveniente de esta solución es que aumenta el tiempo del algoritmo de comparación, ya que básicamente la fórmula de los costes se debe aplicar dos veces, para cada par de nodos. No obstante, en la práctica el aumento no es muy grande ya que este no es el proceso más costoso.

3.4.6.6. Tratamiento del subisomorfismo.

El algoritmo de asignación graduado es capaz de resolver de forma eficiente el problema de la comparación de grafos, obteniendo resultados bastante próximos al óptimo en la mayoría de las ocasiones. Sin embargo, ya hemos visto que el verdadero problema implicado en la interpretación de dibujos de líneas es un problema de isomorfismo de subgrafos. El algoritmo debería ser capaz de encontrar subgrafos isomorfos dentro de los grafos de partida.

En la mayoría de las técnicas de comparación, la extensión a un problema de isomorfismo de subgrafos da lugar a un aumento de la complejidad computacional, o a una degradación de los resultados obtenidos. Por el contrario, en el algoritmo de asignación graduado esta extensión no sólo resulta muy sencilla de aplicar sino que no provoca ninguna variación considerable en el tiempo de ejecución.

Para considerar el problema del isomorfismo de subgrafos, se aplica una técnica estándar utilizada en programación lineal: la introducción de variables ficticias. Dada la matriz de asignación *M*, que consta de una fila y una columna para cada nodo de uno y otro grafo, simplemente debemos añadir filas o columnas adicionales, que representarán nodos ficticios. Si un nodo *A* de uno de los grafos originales es emparejado con un nodo ficticio, esto significará que el nodo *A* no participa en el subgrafo que ha sido emparejado.

Supongamos que tenemos dos grafos G y H, que constan de l y m nodos respectivamente. Si $l=m$ entonces no sería necesario añadir nodos ficticios a ninguno de los grafos. Por el contrario, si $l<m$ podríamos:

- Añadir $m-l$ columnas a *M*, representando nodos ficticios de G. En este caso, el algoritmo de comparación no varía.
- Añadir una sola columna a *M*. Este será un nodo ficticio de G, que podrá ser asignado a varios nodos de H. El paso de normalización debe ser modificado para que esta columna sume $m-l$.

De esta forma, tenemos un caso especial del problema de subisomorfismo. El algoritmo tenderá a buscar un subgrafo dentro del grafo mayor, que sea isomorfo al grafo menor (mayor o menor en cuanto al número de nodos). Aunque esto no es equivalente al problema del subisomorfismo, sin embargo es aplicable para nuestros propósitos ya que el grafo del modelo debería ser encontrado de forma completa dentro del grafo de la escena. En general, sería de esperar que el grafo menor fuera el correspondiente al modelo. No obstante, se pueden plantear algunos problemas, como el mostrado en la siguiente figura. En el dibujo, los segmentos emparejados son etiquetados con el mismo número.

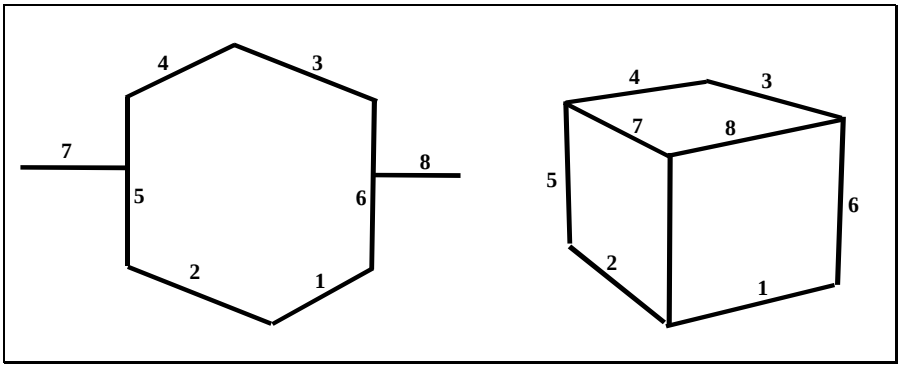

Figura 29 *Resultado del algoritmo de comparación de grafos*

Todos los nodos del grafo de menor tamaño (en este caso a la izquierda) son emparejados con algún nodo del otro grafo, aun cuando no contribuyan a que la solución sea mejor. Esto ocurre con los emparejamientos 7 y 8, que son claramente erróneos, pero que siempre aparecerán asignados de una u otra forma. Para solucionarlo podemos utilizar el coste final de comparación entre nodos, que nos indica claramente que estos emparejamientos tienen mucho error, por lo que no deberían ser tenidos en cuenta en el resultado.

Otra posible solución sería añadir nodos ficticios a ambos grafos, de forma que los nodos del grafo menor puedan no intervenir en el subisomorfismo. En la práctica, esto es equivalente a la solución anterior, donde el criterio para considerar o no un emparejamiento sería que su coste sea mayor o menor que un cierto umbral.

3.4.6.7. El coste de comparación total.

Una vez con el proceso de comparación de grafos completamente definido, sólo nos queda por determinar cual será el coste final de comparación entre dos grafos. Este coste está en función de los costes de comparación entre nodos, para la matriz de asignación final (que contendrá valores {0, 1}, según dos nodos sean emparejados o no). Sin embargo, debemos buscar una medida más significativa que el valor de la función objetivo (1).

Imaginemos que hemos encontrado un subisomorfismo perfecto entre un grafo G con *l* nodos, y un grafo H con *m* nodos, utilizando el segundo de los costes de comparación de enlaces *C'*. Supongamos que G representa la escena, H el modelo y que el número de emparejamientos encontrados es *e* (obviamente $e \leq l$ y $e \leq m$).

El coste de comparación de un nodo de G asignado a un nodo no ficticio de H será: $C_1*(e-1)$, puesto que las asignaciones con nodos ficticios no influyen en el coste y todas las *e-1* restantes influyen con un costo positivo C_1. Sumando estos costes para todos los nodos emparejados tendremos: $C_1*(e-1)*e$. Así, haciendo la suma y normalizando por este valor tendremos una medida que se aproximará a 1 cuanto mejor sea la asignación, sea cual sea el número de nodos asignados.

No obstante, la medida también debería tener en cuenta el número de nodos asignados *e*. En general, el emparejamiento será bueno si todos los nodos del modelo son asignados, independientemente de que hayan más o menos segmentos no emparejados en la escena. El valor *e/m* nos dará la proporción del modelo que ha sido emparejada. Uniéndolo a lo anterior, la suma debería ser multiplicada por: $(e/m)/(C1*(e-1)*e)=m/(C1*(e-1))$.

En definitiva, el valor total del coste de comparación entre dos grafos será:

$$\text{Coste} = \sum_{k=1}^{e} C(a_{\text{emparej}(k)}, i_{\text{emparej}(k)}) \frac{m}{C1*(e-1)} \qquad (5)$$

Este coste valdrá 1 cuando el modelo sea encontrado completamente en la escena y, con valores adecuados de C_1, C_2 y C_3, será próxima a 0 para emparejamientos malos.

3.4.6.8. Interpretación mediante comparación.

La resolución del problema del subisomorfismo permite ir más allá de la simple clasificación de objetos, pudiendo afrontar un problema más general de interpretación de escenas. Aunque esto no ha sido abordado completamente en este proyecto, daremos algunas ideas para el posible trabajo en este ámbito. La salida de la interpretación, implementada en este proyecto, consta de un único objeto con una información de posición y orientación determinada.

En el sentido más amplio, dada una escena representada como un dibujo de líneas, una interpretación de la misma consistiría en una descripción simbólica de todos los objetos que la constituyen y de posibles relaciones entre ellos. Por ejemplo, ante la figura 30, el proceso de interpretación debería deducir que hay un cubo a la derecha de una pirámide, y que detrás del primero se encuentra otra pirámide.

El proceso de comparación de grafos definido, es capaz de encontrar los objetos en la escena uno a uno. Por lo tanto, sería necesario repetir la comparación con todos los modelos existentes. Para todos aquellos cuyo coste de comparación total supere un cierto umbral, suponemos que la comparación es buena. Estos objetos, con la equivalencia de segmentos correspondiente, serían añadidos a la hipótesis de interpretación. Nótese que hablamos de hipótesis ya que posteriormente deberá ser verificada.

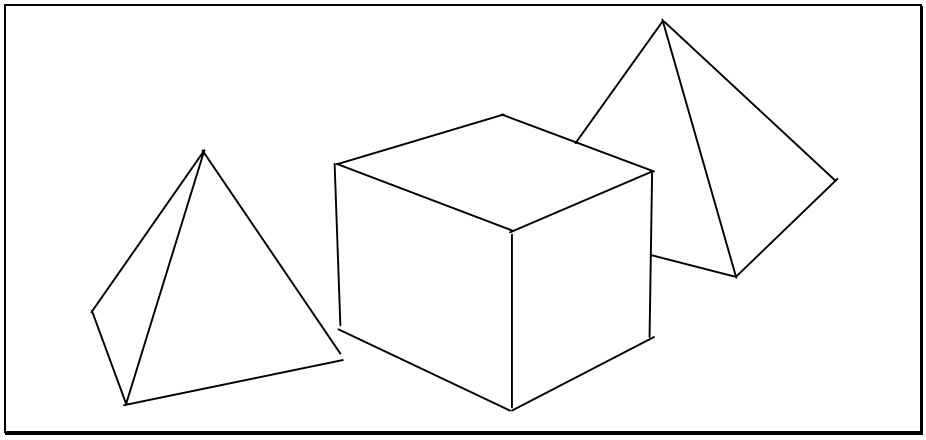

Figura 30 *Posible entrada para una interpretación*

Sin embargo, este proceso iterativo no es suficiente. Por ejemplo, para la entrada anterior, el modelo de la pirámide siempre sería emparejado con la pirámide de la izquierda (ya que es la solución óptima). Por este motivo, es necesario que después de detectar un objeto, este sea eliminado de alguna forma de la escena. El proceso de interpretación podría tener la siguiente estructura:

1. Inicializar la hipótesis de interpretación a vacío.
2. Repetir
    2.1. Para cada modelo de la base de datos hacer:
        2.1.1. Comparar el modelo con la escena.
        2.1.2. Si el coste de comparación es mayor que un umbral entonces:
            Añadir el tipo del modelo a la hipótesis de interpretación, junto con la equivalencia de segmentos encontrada.
            Eliminar los segmentos emparejados de la escena.
3. Hasta que no queden líneas en la escena o no cambie la hipótesis de interpretación.

De esta forma obtendríamos en la hipótesis de interpretación una lista con todos los objetos que se encuentran en la escena, junto con las líneas que lo forman cada uno.

3.4.7. Alineamiento y verificación de hipótesis.

La última fase dentro de la etapa de alto nivel es la de alineamiento, a partir de cuya salida se verificará la hipótesis de interpretación. Ambos procesos (alineamiento y verificación) se encuentran muy estrechamente relacionados, por lo que serán descritos de forma conjunta. El uso del alineamiento se hace imprescindible para obtener una comprobación precisa de la hipótesis, suponiendo una vuelta atrás en el nivel simbólico de descripción, para considerar las propiedades geométricas de los objetos de forma exacta.

El alineamiento parte de un conjunto de vistas de un modelo y de una asociación con los puntos correspondientes en la escena, que se suponen pertenecientes a ese modelo. Su objetivo es la obtención de una nueva vista, a través de una combinación lineal de las vistas del modelo, que será una predicción de cual debería ser el aspecto del dibujo de la escena, en caso de ser realmente una instancia de ese modelo.

Por último, la comparación a bajo nivel entre la escena y la predicción determinará el grado de parecido entre ambos. Para ello será necesario usar una medida de distancia adecuada. Aunque los aspectos teóricos básicos de estos procesos son tratados en los siguientes apartados, estos no han sido abordados de forma muy extensa en la implementación del sistema, ya que esta se centra en las cuestiones propias de la interpretación más que de la verificación.

3.4.7.1. El problema de la comparación estructural.

Tras la fase de comparación de grafos, obtenemos una hipótesis de interpretación en la que se describen cuales son los objetos encontrados en la escena y cuales son las correspondencias entre líneas del modelo y de la escena. Esta salida, por sí misma, podría considerarse como el resultado último del proceso de visión: la escena consta de una serie de objetos identificados, situados en unas posiciones determinadas.

Sin embargo, no podemos olvidar que la comparación de grafos está basada exclusivamente en propiedades estructurales de los dibujos: un segmento está unido a otro, está situado en un lado determinado respecto a los otros, corta o termina en otro. El método es completamente invariante a escala, rotaciones en el espacio 3D y variaciones en las proporciones y ángulos entre las caras de los objetos. Aunque esto era el objetivo perseguido en la comparación de objetos, resultará finalmente en una extremada ambigüedad de las descripciones sobre las que trabaja. Por ejemplo, todos los grafos asociados a los dibujos de líneas de la siguiente figura serán considerados como isomorfos. Es más, los grafos asociados a estos dibujos de líneas son isomorfos a sí mismos para varias asignaciones diferentes. Por ejemplo, en la figura alargada no hay ningún conocimiento sobre cuales son las aristas alargadas y cuales las cortas.

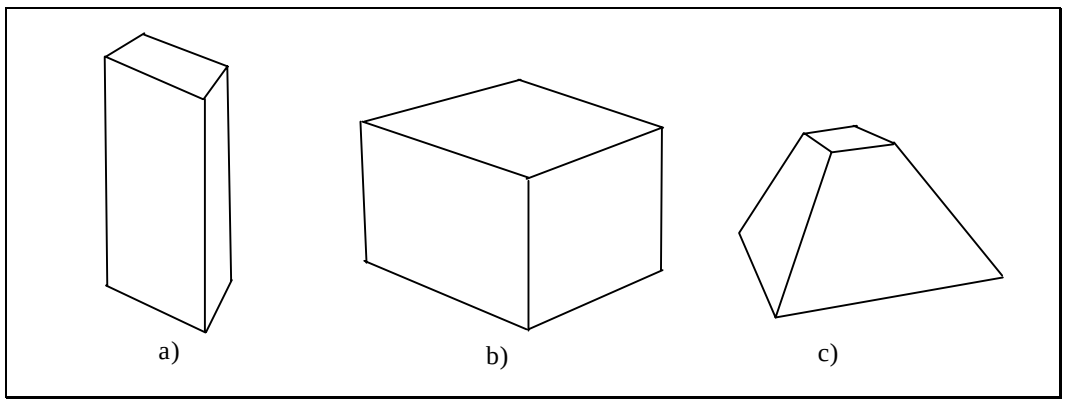

Figura 31 *Dibujos de líneas con grafos de aristas isomorfos*

Estos dibujos representan objetos claramente diferentes, cuya estructura 3D puede ser fácilmente identificada por un humano. A pesar de ello, las propiedades introducidas en los grafos no contienen ninguna información que permita distinguirlos. Pero la introducción de estas propiedades

diferenciadoras podría causar problemas en el sentido contrario, varias vistas de un mismo objeto podrían ser consideradas como incompatibles.

Por lo tanto, se hace evidente la necesidad de comprobar las hipótesis resultantes de la comparación de grafos, con un procedimiento a más bajo nivel que tenga en cuenta la forma, disposición y tamaños de las caras de los objetos. Este proceso es el alineamiento de vistas, que será descrito en los siguientes apartados.

3.4.7.2. Alineamiento mediante combinación de vistas.

Supongamos que disponemos de un dibujo de líneas de borde, como los de la figura 31, del cual estamos interesados en conocer si es una instancia de un cierto modelo de objeto tridimensional. Hablando en términos geométricos, el dibujo de líneas de borde es una proyección 2D de un modelo 3D. Si el dibujo procede de una escena real, entonces la proyección es perspectiva, aunque normalmente será muy próxima a una proyección paralela.

Para conocer si el dibujo de líneas corresponde al modelo, deberíamos comprobar si alguna de las proyecciones del modelo da lugar a un dibujo como el de partida. Algunos cálculos serían necesarios para determinar la posición y orientación del centro de proyección, antes de hacer la proyección propiamente dicha. Una vez con esto, el resultado indicaría de forma precisa e irrefutable, si el dibujo puede ser considerado o no como perteneciente al modelo proyectado. Esto es lo que se conoce como alineamiento mediante modelos 3D.

Puesto que disponemos de una hipótesis de modelo para el dibujo de líneas y una equivalencia de segmentos entre escena y modelo, la aplicación de esta técnica sería perfectamente posible. Sin embargo, este método implica (de forma explícita) la construcción y manipulación de modelos tridimensionales de los objetos. Esto puede ser problemático en caso de existir objetos con formas complejas. En la práctica, podemos usar otros métodos, también dentro de los esquemas del alineamiento, pero sin manejar explícitamente descripciones 3D de los modelos. Estos son conocidos como métodos de alineamiento por combinación de vistas 2D.

Si nos fijamos en las representaciones de planta, alzado y perfil, utilizadas para describir objetos poliédricos, sabemos que estas tres vistas son suficientes para describir la posición tridimensional de todos los vértices del objeto. En general, unas pocas vistas de un objeto, junto con algunas correspondencias entre vértices, serán suficientes para recuperar las posiciones relativas de las partes del objeto en el espacio. Shimon Ullman [Ull96], demostró que tres imágenes, con cuatro puntos correspondientes en cada una, son siempre suficientes para determinar estas posiciones relativas de los vértices. Incluso en algunos casos, es suficiente con sólo dos vistas.

Dadas tres vistas 2D de un objeto, podemos obtener una nueva vista del mismo, mediante una combinación de las anteriores. Partiendo del hecho de que la proyección de un modelo 3D puede ser expresada con ecuaciones lineales, es posible demostrar que la combinación de vistas para obtener una nueva, es una simple una combinación lineal. Si $M = \{M_1, M_2, M_3\}$ es el conjunto de vistas 2D de un modelo, entonces cualquier nueva vista P puede ser expresada como:

$$P = \sum_{i=1}^{3} \alpha_i M_i \qquad (6)$$

Más específicamente, si suponemos que P está formado por el conjunto de puntos $\{p_1 = (p_{1x}, p_{1y}), ..., p_n = (p_{nx}, p_{ny})\}$, y $M_i = \{m_{i1} = (m_{i1x}, m_{i1y}), ..., m_{in} = (m_{inx}, m_{iny})\}$, para i=1, 2, 3, entonces tenemos:

$$p_{kx} = a_0 + \sum_{i=1}^{3} a_i m_{ikx} \;;\; p_{ky} = b_0 + \sum_{i=1}^{3} b_i m_{iky} \;;\; \forall\, k=1..n \qquad (7)$$

Los parámetros a_i y b_i de las combinaciones lineales pueden ser calculados fácilmente, resolviendo el sistema de ecuaciones con los valores (x, y) de los puntos correspondientes en escena y modelo. Dos parámetros constantes han sido añadidos, a_0 y b_0, para modelar las traslaciones. Por lo

tanto, necesitamos que el proceso de comparación asigne un mínimo de 4 puntos para poder calcular los valores de las incógnitas.

El valor de los parámetros a_i y b_i tiene un significado especial. Como hemos visto, a_0 y b_0 representan la traslación, en el dibujo 2D, respecto del punto medio. Los valores de a_1 y b_1 indican la proporción de M_1 que es considerada en la vista nueva. Lo mismo ocurre para M_2 y M_3. Puesto que los M_i son una proyección del modelo 3D con una cierta orientación y escala, a partir de los parámetros a_i y b_i (que ponderan cada una de estas proyecciones) podemos predecir la posición y orientación 3D del objeto alineado. Además, estos valores deben cumplir ciertas restricciones para que la combinación sea una proyección físicamente posible. No obstante, esta interpretación de los parámetros requiere la extracción de propiedades 3D de los modelos, puesto que necesitamos conocer cual es la orientación y posición tridimensional de los modelos.

3.4.7.3. Aplicación del alineamiento.

El método básico de alineamiento puede extenderse para ser aplicado a problemas de reconocimiento con objetos de bordes suaves, descompuestos en partes, imágenes en escala de grises o dibujos de bordes de objetos complejos. El alineamiento por combinación de vistas puede resolver estos problemas eficientemente, requiriendo cálculos más simples que con otros acercamientos. A pesar de ello, este método deja una cuestión fundamental por determinar: la búsqueda de los puntos característicos correspondientes entre las distintas vistas.

En este proyecto, el alineamiento es usado con el propósito de realizar una verificación de las hipótesis de interpretación, por lo que este problema se encuentra resuelto. La determinación del modelo a alinear y de los puntos característicos correspondientes procede del resultado obtenido en la comparación de grafos.

Para realizar el alineamiento, las posiciones (x, y) de los extremos de cada segmento serán añadidos a la descripción de los nodos, dentro de los grafos de aristas. De esta forma, el modelo de un objeto estará formado por tres grafos de aristas, junto con una lista de todas las asignaciones de nodos de los tres grafos.

La estructura del proceso de alineamiento de un modelo M, formado por las vistas {M_1, M_2, M_3}, con una escena P, suponiendo que e nodos han sido emparejados (necesariamente e debe ser mayor que 3), es la siguiente:

1. Calcular las medias en x é y de los puntos de todos los grafos:
   a_0= media(P_{kx}); b_0= media(P_{ky});
2. Centrar los dibujos, restando a todos los puntos la media correspondiente.
3. Crear la matriz del sistema de ecuaciones:
   3.1. Buscar los tres puntos correspondientes a los emparejamientos con mejores costes.
   3.2. Introducir los valores (x, y) de los puntos de M y P en las posiciones adecuadas de la matriz del sistema.
4. Resolver el sistema de ecuaciones, aplicando el método de Cramer, obteniendo los valores de a_i, b_i para i=1, 2, 3.
5. Crear un nuevo grafo, con tantos nodos como los del modelo, calculando las posiciones mediante la fórmula (7).

Los cálculos de las medias en el paso 1, se basan únicamente en los nodos que hayan sido emparejados. De esta forma evitamos problemas debidos a que la escena conste de otros objetos. En el paso 3, se buscan los mejores puntos, en base a los cuales hacer el alineamiento. Para ello se puede usar el criterio del coste de asignación entre nodos. Se debe tener en cuenta que los nodos representan segmentos, cada uno con dos puntos extremos. El cálculo del sistema de ecuaciones es realizado en el paso 4. Una vez con los parámetros, se realizará la combinación lineal de las vistas para obtener la imagen prevista de la escena. Esta imagen estará descrita con un grafo de aristas (incluyendo los valores de las posiciones), que contendrá un segmento por cada uno del modelo.

De esta manera, será posible que la imagen prevista conste de más nodos que la de entrada, esta imagen muestra cómo sería visto el objeto si fuera observado sin ningún error.

3.4.7.4. Verificación de la hipótesis.

Una vez obtenido el alineamiento de la escena con el modelo, el siguiente paso será comprobar de la forma lo más precisa posible, si la escena observada se corresponde realmente con la prevista. En otras palabras, se trata de determinar si el alineamiento ha sido satisfactorio. El problema clave será definir una medida de distancia adecuada, para ser aplicada entre dibujos cuyos tamaños, formas y posiciones deberían ser teóricamente los mismos.

Aunque la imagen prevista está dada como un grafo de aristas, es obvio que no será de utilidad la medida del coste de comparación entre grafos. Es necesario utilizar una métrica de bajo nivel, que tenga en cuenta las propiedades geométricas del modelo y de la escena de forma precisa. En principio, podemos pensar en dos posibilidades: realizar la comparación a nivel de segmentos o a nivel de puntos de borde.

A nivel de segmentos, la medida de distancia estaría basada en una suma de los errores cometidos para todos los segmentos. Por ejemplo, sumando las distancias (entendidas como distancias euclídeas) entre los puntos medios de los segmentos equivalentes escena/modelo obtendríamos un grado de aproximación del alineamiento a la escena real. Esta medida podría ser normalizada por el número de segmentos y la longitud de los mismos, para obtener un porcentaje medio de variación en distancia.

A pesar de su sencillez y su aparente validez, esta definición supone una solución aproximada, poco adecuada al propósito último de la verificación de hipótesis. El dibujo de líneas con el que se compara ha sido obtenido tras la aplicación de los procesos de detección de bordes, detección de segmentos y búsqueda de propiedades no accidentales. En consecuencia, la medida estaría acumulando los errores introducidos en todas estas etapas. Es más, puesto que la diferencia es calculada para los pares de segmentos equivalentes, también serían acumulados los posibles errores de la comparación de grafos. Por lo tanto, necesitamos una comparación de bajo nivel.

La comparación a nivel de puntos de borde (a bajo nivel) supone el paso contrario. A partir del grafo previsto obtenemos un dibujo de bordes sintético, que será comparado con el dibujo de bordes original. Intuitivamente, la distancia consistiría en superponer ambos dibujos, observando aquellos puntos que están lejos de los puntos de borde del otro dibujo. En los ejemplos a) y b) de la siguiente figura se muestran estos dibujos de borde superpuestos.

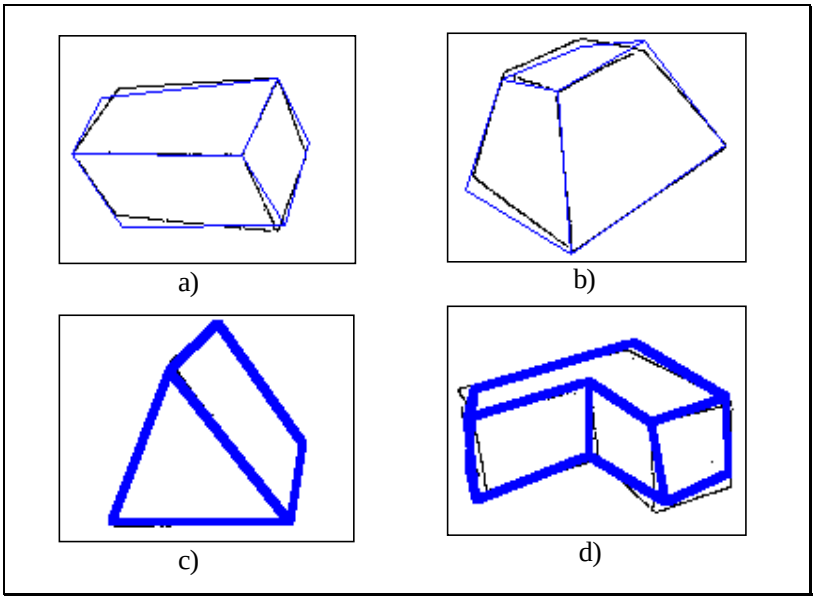

Figura 32 *Dibujos de borde originales y tras el alineamiento*

Las diferencias encontradas en la superposición pueden ser medidas como una distancia de Hamming. El error será el porcentaje de puntos que estén en un dibujo pero no en el otro. Esta sería la medida ideal, teniendo en cuenta que hay un alineamiento previo. Sin embargo, es bien conocido que la distancia de Hamming resulta extremadamente sensible a ligeras traslaciones o escalados. Por ejemplo, en las situaciones anteriores con toda seguridad se obtendría un valor muy malo, aun cuando los alineamientos son relativamente buenos.

Una solución sencilla consistiría en hacer más flexible la distancia de Hamming. Para ello, podemos hacer que las líneas del dibujo de bordes sintético sean pintadas con un cierto grosor, de varios píxeles. La nueva distancia vendría dada por la proporción de puntos de la escena que no aparecen en el dibujo sintético. De esta forma conseguimos cierta flexibilidad en la medida (según el grosor de las líneas) sin perder la simplicidad del cálculo de la distancia de Hamming original. En los ejemplos c) y d) se muestra esta nueva medida.

Esta modificación supone que la distancia deja de ser simétrica, uno de los dibujos es la escena y otro el modelo. Además, en caso de existir múltiples objetos en el dibujo, deberían compararse sólo los bordes pertenecientes a un objeto en cada caso. Este es un problema que requeriría un estudio más profundo, que cae fuera del área de interés fundamental de este proyecto.

4. Conclusiones.

4.1. Integración de etapas de procesamiento.

Como se establecía en los objetivos del proyecto, se ha hecho un especial hincapié en la integración de todas las etapas dentro de una estructura de procesamiento global. Hemos visto que cada uno de los tres grandes grupos en que pueden ser divididas las técnicas de visión artificial debe resolver un tipo de problemas completamente distinto, en niveles de abstracción diferentes. Esto hace que, en algunas ocasiones, los desarrollos realizados en un área específica no tengan en cuenta la verdadera utilidad del resultado en un marco de acciones más amplio.

Tal es el caso de las técnicas de interpretación de dibujos de líneas mediante etiquetado de nodos, cuyo uso fue considerado en el presente proyecto. En sí misma, esta técnica es capaz de proporcionar mucha información sobre escenas complejas, sin necesidad de contar con conocimiento previo de los posibles modelos. Sin embargo, si intentamos aplicarla a una situación real, en la que exista algo de ruido, el método fallará ya que todo el desarrollo está realizado sobre dibujos de líneas perfectos, obtenidos sintéticamente. No obstante, como hemos podido comprobar después, no son descartables las ideas aportadas en estas técnicas, consideradas desde un punto de vista más general.

En este proyecto, la aplicabilidad del método de alto nivel desarrollado ha sido garantizada mediante el uso de escenas reales, y el encadenamiento de los resultados obtenidos por los distintos niveles de procesamiento. De esta forma, el proceso de alto nivel es adaptado a las capacidades que nos ofrecen los procesos de detección de bordes y de segmentos. Por ejemplo, un elaborado paso de creación de grafos tuvo que ser añadido, debido a que el proceso de detección de segmentos difícilmente era capaz de obtener una representación exacta de todas las líneas de borde.

El resultado final demuestra que la integración no sólo es posible, sino que también contribuye al reparto equilibrado de la complejidad del problema a lo largo de todas las etapas. Computacionalmente, se obtiene una buena coordinación entre los procesos de los distintos niveles, que serán capaces de actuar de forma paralela. Aun en una ejecución secuencial, el tiempo de respuesta del sistema es lo suficientemente rápido como para permitir un seguimiento de escenas en tiempo real, con varias ejecuciones completas por segundo en los mejores casos.

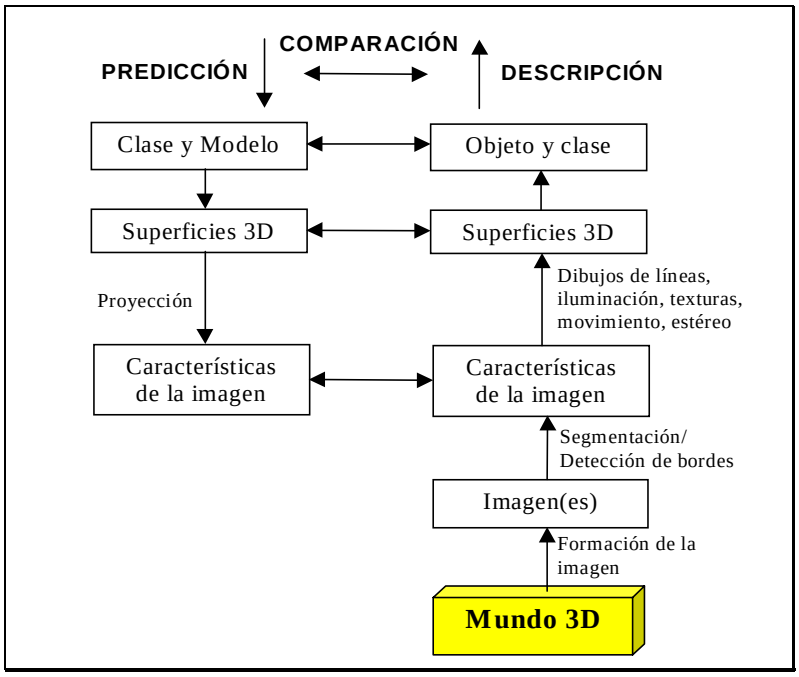

Figura 33 *Un paradigma para el reconocimiento de objetos*

Por otro lado, la unión de las etapas más que provocar una degradación sucesiva de las descripciones de la escena, da lugar a una mejora en la calidad. La razón es que ciertos errores que no pueden ser resueltos en una etapa pueden ser tratados convenientemente en la siguiente. Por ejemplo, si en una arista no se detecta un borde, por falta de contraste, esto no podrá ser solucionado por la detección de segmentos. Sin embargo, la comparación de grafos puede detectar que en esa posición falta un segmento.

En la última parte del proceso de alto nivel, hemos visto la necesidad de retroceder en el grado simbólico de descripción, para considerar la escena nuevamente desde una representación a bajo nivel. De esta manera, tendríamos una más compleja integración de las etapas con posibles interacciones en ambos sentidos. El camino ascendente (de bajo a alto nivel) es un proceso de abstracción en las descripciones de las imágenes, cada vez más próximas al concepto de objeto. El camino descendente es una predicción de cómo debería ser la escena, desde el punto de vista de una interpretación realizada a cualquier nivel (en nuestro caso en el nivel superior).

Con esto, estaríamos más cerca de la visión de futuro propuesta por V. S. Nalwa, en [Nal93]. Una representación gráfica de este paradigma es mostrada es la figura 33.

4.2. Extensiones y aplicabilidad práctica.

Aunque el problema de la interpretación ha sido enfocado hacia un dominio de objetos poligonales 3D dentro de un entorno controlado, creemos que la estructura del acercamiento expuesto, de interpretación basada en comparación de grafos, podría ser aplicada a la resolución de problemas con dominios más complejos. Asimismo, puesto que el sistema hace uso de técnicas existentes en todos los niveles de procesamiento, podría verse beneficiado de las posibles mejoras en esos niveles, para ampliar su dominio de entrada. Muchas de las restricciones podrían verse eliminadas con la introducción de nuevos métodos de bajo o medio nivel.

A continuación damos una visión crítica sobre cuales pensamos que son las principales restricciones del proceso desarrollado y exponemos algunas ideas de cómo podría ser extendido el sistema para poder superar estas limitaciones en el dominio de entrada.

4.2.1. Detección de bordes.

Una limitación fundamental introducida por esta etapa es la imposibilidad de obtener una buena imagen de bordes para los objetos con texturas. Gran parte de los objetos del mundo real disponen de una textura no uniforme. Para tamaños pequeños de textura el suavizado aplicado a la imagen podría solucionar el problema, haciendo que la superficie fuera reconocida como homogénea. Sin embargo, para tamaños medios el dibujo de bordes puede ser completamente errático, y muy difícil de tratar para cualquier posible procesamiento posterior. Un problema muy parecido ocurre cuando los objetos tienen marcas o dibujos en su superficie.

El reconocimiento de las texturas es un problema muy complejo, y aun no resuelto completamente en el ámbito de la visión artificial, que se encuentra dentro de las técnicas de medio nivel. Primero se deberían distinguir las zonas con igual textura (lo que es conocido como segmentación por textura) para después obtener las líneas de separación entre zonas.

Otra limitación de la detección de bordes es la necesidad de disponer de un buen contraste en las imágenes. Esto dificulta el proceso en caso de objetos con contornos suaves o cuando la iluminación es difusa y no es un parámetro que pueda ser controlado. Para resolver este problema se podría utilizar información adicional, por ejemplo de profundidad o movimiento de los objetos.

4.2.2. Detección de segmentos.

La etapa de medio nivel impone una restricción sobre los tipos de objetos a utilizar, debida al uso de segmentos de líneas. Puesto que la escena es descrita mediante líneas rectas, sólo podrán usarse objetos con caras planas, es decir objetos poliédricos.

Nuevamente nos encontramos con una limitación que excluye a la mayoría de los objetos del mundo real. Aunque las curvas pueden ser descritas mediante varios trozos de rectas, esto no será de ninguna utilidad para los procesos posteriores (al menos de la manera en que han sido definidos) ya que una misma curva podría ser dividida en más o menos trozos según la escala o el ángulo de visión, de forma completamente incontrolable.

Por lo tanto, sería necesario utilizar técnicas de agregación de bordes con otro tipo de estructuras, por ejemplo trozos de parábola o splines. En este caso además, también debería ser modificada la etapa de alto nivel para admitir el uso de líneas curvas.

4.2.3. Interpretación de la escena.

En la etapa de alto nivel se acumulan muchas de las limitaciones de las etapas previas. Para que un objeto pueda ser reconocido a alto nivel, debe tener un borde que sea significativo del mismo y este borde debe estar descrito con tramos rectos. Además, añade otra importante restricción. Como hemos visto, la etapa de alto nivel realiza una interpretación basada en modelos. Para que cualquier objeto pueda ser reconocido primero debemos tener un modelo del mismo. Igualmente, para poder aplicar el alineamiento necesitamos tres vistas de ese modelo.

La estructura de los grafos podría ser modificada fácilmente para tratar objetos con curvas. Los nodos tendrían una propiedad de ser curvados o no, y deberíamos considerar algunos tipos más de uniones entre nodos. El cálculo de los costes debería ser modificado para tratar estos casos, teniendo en cuenta que un borde curvado puede ser visto, bajo ciertos puntos de vista, como si fuera recto. Aunque no está completamente garantizado el funcionamiento fiable de este proceso para objetos con bordes curvados, creemos que resultaría previsible la obtención de buenos resultados en caso de que los objetos contengan a la vez caras curvadas y planas, o caras curvadas pero bien diferenciadas. Sin embargo, aun permanece una limitación fundamental y más difícil de superar, que es la necesidad de que los bordes internos de los objetos sean significativos. En otro caso, se deberían buscar otras propiedades y quizá también un método más adecuado a tales situaciones.

4.3. Conclusiones finales.

A lo largo del presente proyecto se ha desarrollado un acercamiento al problema de la visión de alto nivel con representaciones de dibujos de líneas, abordando también de forma muy extensa los demás aspectos del sistema de visión en el que se enmarca. Se ha pretendido dar un acercamiento modular y extensible, al problema de la interpretación de escenas tridimensionales, más que solucionar el problema concreto de reconocer determinados objetos poliédricos sin más.

Todas las principales decisiones en cuanto a las técnicas a utilizar en cada parte, han sido precedidas de un estudio más o menos pormenorizado de algunas de las más importantes aproximaciones en la resolución de esos tipos de problemas. Gracias a este estudio, la solución propuesta en este proyecto hace uso de una mezcla de muchas de las técnicas desarrolladas en áreas muy variadas dentro de la visión artificial, junto con algunos desarrollos propios, específicos del sistema construido. Una estructura de procesamiento global integra todas estas técnicas, en un marco de acciones dirigido hacia el propósito último de la interpretación tridimensional de escenas.

Los resultados obtenidos por la aplicación implementada, que se pueden consultar en el anexo B, son un reflejo claro de que el objetivo del proyecto ha sido cumplido satisfactoriamente. Para el dominio de entrada considerado, el programa es capaz de obtener un resultado positivo y en un tiempo reducido, en la mayoría de los casos. La hipótesis de interpretación tiene una sólida base en las propiedades de la escena, extraídas y depuradas por todas las fases previas a la interpretación. Otra ventaja fundamental del acercamiento es la facilidad en el manejo de los modelos de objetos, a partir de simples vistas de los mismos, sin necesidad de trabajar explícitamente con las propiedades tridimensionales, y ello a pesar de encontrarnos dentro de un dominio tridimensional.

No obstante, hemos visto que la simple emisión de una hipótesis de interpretación no era suficiente, debido a un problema que afectaba al proceso de comparación de modelos: la elevada ambigüedad existente en la estructura de representación de alto nivel, es decir en los grafos de aristas. Es necesario que la hipótesis de interpretación sea reforzada con la aplicación de los métodos de alineamiento, llegando a una solución con mayor fiabilidad, basada en las características de los objetos a más bajo nivel.

Vimos en su momento que en la definición de los grafos mucha información del dibujo de líneas era rechazada porque se consideraba que no era una propiedad invariante a transformaciones tridimensionales. A pesar de ello, creemos que resultaría necesario evitar la ambigüedad implícita en los grafos (tal y como han sido definidos) introduciendo alguna propiedad más, que proporcione información sobre los objetos válida para un determinado rango de variación. En especial, sería interesante considerar relaciones tales como las de paralelismo entre segmentos y las proporciones en las longitudes de las líneas que pertenecen a una misma cara de un objeto.

Por otro lado, resulta claro que las peculiaridades del proceso desarrollado no son aplicables de forma directa a una situación práctica real. El universo de entrada para el sistema se encuentra dentro de un dominio muy reducido, en el que no hay posibilidad de permitir muchos de los objetos del mundo real. Además todo el entorno de la escena debe ser controlable, permitiendo el ajuste de ciertos parámetros tales como la iluminación. No obstante, debemos tener en cuenta la envergadura del problema tratado. De hecho, por el momento todos los sistemas de visión necesitan trabajar sobre dominios muy específicos para poder obtener resultados útiles. Esto es aun más crítico cuando nos encontramos en un entorno tridimensional, trabajando en la resolución de problemas de alto nivel.

Yendo más allá de esta valoración puntual, creemos que los resultados son bastantes esperanzadores en cuanto a las posibilidades que ofrece la línea de acción propuesta, para la resolución de problemas de interpretación en un contexto más general. Podemos extraer algunas ideas interesantes, en relación a la resolución de problemas complejos en el marco de la visión artificial.

Primero, que llegar a un nivel de descripción más simbólico parece ser una necesidad relacionada de forma directa con el aumento en la complejidad de los problemas tratados. La construcción de estas descripciones contribuye a un reparto de la complejidad entre las distintas etapas, lo cual se dificulta en un acercamiento más directo. La representación final de las escenas hará énfasis en las propiedades invariantes de las mismas (cualesquiera que sean consideradas), mediante una extracción de las características más relevantes de las descripciones de nivel inferior.

Segundo, que la definición de relaciones entre partes o propiedades de los objetos enriquece en gran medida las posibilidades de descripción de los mismos. La definición de estas relaciones añade una propiedad estructural a la representación, en la que cada parte está reforzada por su relación con el resto. En cierto modo se trata de un esquema de descomposición en partes, pero en un sentido más amplio. Un dibujo de líneas, por ejemplo, puede ser visto como una representación por partes donde los componentes son los segmentos.

Tercero, que los distintos niveles no deben ser entendidos como procesos de ejecución secuencial, sino que pueden existir relaciones más complejas entre los mismos. Una etapa puede necesitar recibir cierta información de las etapas de nivel más bajo, y puede generar resultados que serán pasados a esas etapas inferiores para su verificación. Por otro lado, en muchos casos será posible y tendrá sentido la ejecución paralela de los procesos de distinto nivel, fundamentalmente cuando sea requerido algún tipo de seguimiento de escenas en tiempo real.

5. Bibliografía.

[Ull96] Shimon Ullman
High-level Vision. Object Recognition and Visual Cognition.
The MIT Press, 1.996

[Gon93] Rafael C. González, Richard E. Woods
Digital Image Processing
Addison-Wesley, 1.993

[Nal93] Vishvjit S. Nalwa
A Guided Tour of Computer Vision
Addison-Wesley, 1.993

[Win92] Patrick Henry Winston
Artificial Intelligence, 3rd Edition
Págs: 249-273, 531-551
Addison-Wesley, 1.992

[Gol96] Steven Gold, Anand Rangarajan
A Graduated Assignment Algorithm for Graph Matching
IEEE Transactions on Pattern Analysis and Machine Intelligence
Vol. 18, No. 4, Abril 1.996

[Lop97] Pedro E. López-de-Teruel, Alberto Ruiz
A Probabilistic Learning Algorithm for Real-Time Line Detection
Technical Report DIS 8-97

[Bal82] Dana H. Ballard, Christopher M. Brown
Computer Vision
Prentice-Hall, 1.982

[Can86] J. Canny
A Computational Approach to Edge Detection
IEEE Transactions on Pattern Analysis and Machine Intelligence
Vol. PAMI-8, No. 6, Noviembre 1.986

[Low87] D. Lowe.
Three-dimensional object recognition from single two-dimensional images.
Artificial Intelligence, 31:355--395, 1987.

[Marble]
MARBLE: Interactive Vision
http://www.icbl.hw.ac.uk/marble/vision

[Heriot]
Heriot-Watt University Reserch in Computer Vision
http://www.cee.hw.ac.uk/Vision/www.html

Anexo A. Diseño e implementación del sistema.

A.1. Introducción.

El proceso de visión desarrollado en el presente proyecto ha sido implementado utilizando una herramienta de programación visual orientada a objetos. En este anexo se comentarán las principales cuestiones relativas al diseño e implementación del sistema de visión propuesto.

Se ha creado un prototipo del sistema final, en el que se integran todas las etapas del sistema de visión completo, desde la adquisición hasta la interpretación. También se han creado varias aplicaciones, con el fin de probar de forma independiente cada una de las etapas del proceso de visión, en especial las relativas al proceso de alto nivel. Los programas realizados funcionan bajo el sistema operativo Windows 95. El uso de los programas adjuntos al proyecto resulta muy intuitivo, por lo que no se ha creído necesaria la creación de un manual de usuario. No obstante, estos programas incluyen una ayuda en línea en la que se explican las funciones más importantes.

El entorno de desarrollo utilizado para la creación de las aplicaciones es **Delphi 3**. Esta es una herramienta de programación visual muy extendida en la creación de aplicación bajo Windows 95, que incorpora las características de la programación orientada a objetos. En este sentido, creemos que el uso de la programación orientada a objetos es una de las bases fundamentales para la construcción de programas que satisfagan los criterios de calidad del software, principalmente en cuanto a extensibilidad y reutilización.

La decisión de la herramienta de programación a utilizar ha estado basada fundamentalmente en la disponibilidad de librerías existentes para algunos de los procesos necesarios en el sistema de visión completo. También se han tenido en cuenta los criterios de calidad, en cuanto a las posibilidades ofrecidas por las distintas herramientas para la creación de programas eficientes, una cuestión fundamental debido a la elevada complejidad computacional de los procesos implicados. Se ha considerado igualmente la facilidad en la creación de interfaces, teniendo en cuenta que la aplicación está encaminada a ser un prototipo sencillo de utilizar.

En cuanto a las librerías de funciones, este proyecto no parte desde cero, sino que se cuenta con algunas implementaciones de los procesos de nivel inferior y medio. Recordemos que el desarrollo, tanto teórico como práctico, de este proyecto se centra en las técnicas de interpretación a alto nivel. En concreto, se ha dispuesto de los siguientes módulos:

- **TScap**. Este módulo proporciona un componente para el acceso a la cámara QuickCam. Permite la adquisición de imágenes en blanco y negro, además del ajuste de muchos de los parámetros de la cámara, que podrán ser modificados por el usuario final. Se trata de un componente “shareware” gratuito para el uso con propósitos educativos, y disponible en la dirección: http://www.micg.et.fh-stralsund.de/~tstuefe.
- **Meteor**. Módulo para la adquisición de imágenes con la cámara Matrox, de Meteor. Esta cámara tiene una resolución mayor que la cámara QuickCam, pero tiene el inconveniente de requerir una tarjeta de interface específica. Este módulo ha sido desarrollado por el tutor del proyecto.
- **LowLevel**. Este módulo es una librería de funciones de procesamiento de imagen a bajo nivel, desarrollado por los profesores de la universidad de Murcia: Pedro E. López de Teruel Alcolea y Alberto Ruiz García. De las funciones incluidas se utilizará la implementación del algoritmo de Canny.
- **EMSegmentos**. Implementación del algoritmo EMH para la detección de segmentos. Este módulo aportará la funcionalidad del proceso de medio nivel. Ha sido realizado por los mismos creadores del anterior, que en este caso son los autores de este método.

A.2. Estructura de módulos del sistema.

La implementación completa del sistema contiene todas las etapas de proceso definidas, desde la adquisición hasta la etapa de alto nivel. El sistema está dividido en una serie de módulos, cada uno de los cuales tratará de una cuestión específica. Por lo tanto, es necesario crear una estructura adecuada que permita la integración de todos estos módulos. El objetivo que se persigue con esto es facilitar la extensibilidad del sistema, ocultando en los módulos las funcionalidades que no sean requeridas por los mismos.

La estructura de módulos creada es la que se muestra en la siguiente figura. Las flechas del dibujo indican una relación de uso entre los distintos módulos, que en algunos casos implican un intercambio de información en el sentido opuesto.

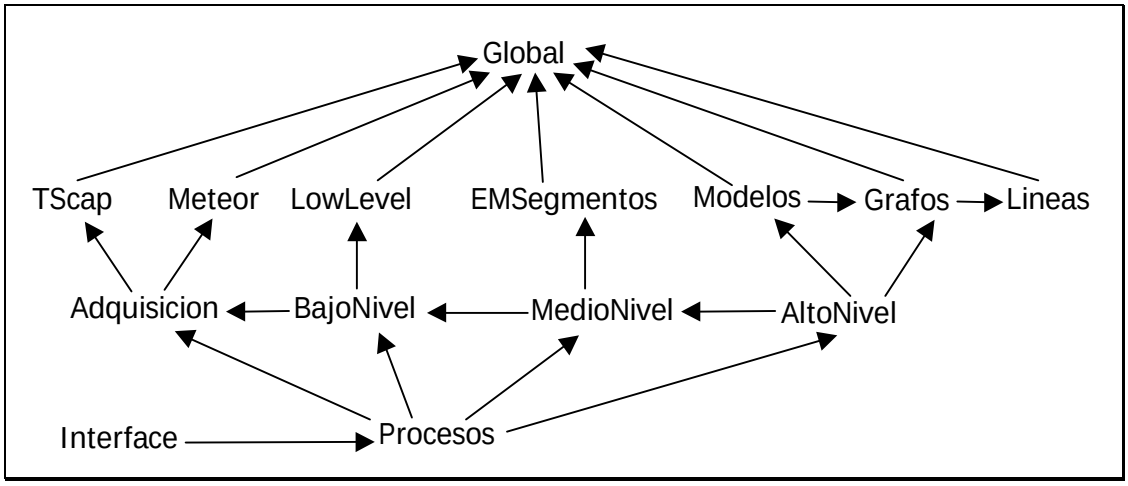

Figura 34 *Estructura de módulos del sistema*

Los módulos han sido divididos en varios niveles, representados horizontalmente. En la parte superior se encuentra el módulo *Global*, en el que se incluyen todas las constantes y variables utilizadas por todos los demás módulos. A un segundo nivel se encuentra un conjunto de módulos que implementan todas las funciones básicas, aunque cada una de forma independiente. En el siguiente nivel hay otro conjunto de módulos (desde *Adquisición* hasta *AltoNivel*) que utilizan los anteriores y cuyo objetivo es garantizar la integración de todas las funcionalidades, para realizar el paso de los resultados entre los distintos niveles. El módulo *Procesos* implementa los procesos concurrentes que se ejecutan a cada uno de los niveles anteriores. Por último, se encuentra el módulo *Interface*, para la creación del interface de usuario.

Como hemos visto, estas etapas tienen la posibilidad de actuar de forma paralela, requiriendo el intercambio de información cada cierto tiempo. Para optimizar la ejecución de los distintos procesos paralelos se ha utilizado una técnica conocida como “multithreading”. Esta técnica consiste en la creación de procesos (llamados “threads”), que comparten la misma zona de memoria, pudiendo acceder a las mismas variables globales. La ejecución concurrente de los threads es manejada por el sistema operativo, optimizando el tiempo de uso de la CPU. De esta manera conseguimos que, en caso de disponer de varios procesadores, los procesos se puedan ejecutar de forma realmente paralela, sin necesidad de realizar ninguna modificación en el programa. El propio sistema operativo se encargará de la paralelización, si admite el uso de “multithreading”.

A continuación comentaremos brevemente la función específica de cada uno de los módulos creados, expuestos en el anterior esquema.

- **Lineas.** En este módulo se implementan todas las operaciones básicas realizadas sobre los dibujos de líneas. Dentro del mismo se definen las clase *TPunto* y *TLinea*, con todas las

operaciones geométricamente más importantes. Este módulo es la base del manejo del dibujo de líneas, ya que se definen las funciones aplicadas sobre las mismas.

- **Grafos.** Dentro de este módulo se encuentran todas las operaciones relacionadas con los grafos de aristas. Estas operaciones van desde la creación de los grafos a partir del dibujo de líneas hasta la comparación de grafos. Se han creado las clases *TGrafo*, para los grafos de aristas, y *TEmparej* para representar un emparejamiento entre grafos.
- **Modelos.** Haciendo uso de los grafos, en este módulo son implementadas las descripciones de los modelos de objetos, incluyendo la comparación entre escena y modelo, y el posterior alineamiento de modelos. La clase definida en este módulo es llamada *TModelo*.
- **Adquisicion.** Esta unidad hace uso de los módulos de las cámaras, para recibir los resultados de la adquisición, haciéndolos disponibles para la etapa de bajo nivel. Se incluye también la posibilidad de realizar la adquisición a través de un fichero del disco.
- **BajoNivel.** Este módulo es el encargado del proceso de detección de bordes. Se usan las funciones de *LowLevel*, para aplicar el operador de Canny, y se implementan los operadores de borde básicos. Además se hace una extracción de los puntos de borde para poder ser utilizados por el siguiente nivel.
- **MedioNivel.** La principal función de este módulo es el intercambio de información con las etapas anterior y posterior. Se usa el módulo *EMSegmentos* para realizar la detección de segmentos. También se incluyen funciones para la entrada/salida del dibujo de líneas.
- **AltoNivel.** El módulo de alto nivel implementa el proceso de interpretación, haciendo uso de las funciones definidas para los grafos y los modelos. Por lo tanto, dentro de este módulo se encuentra la estructura de procesamiento iterativa realizada para la interpretación, en la cual la escena es comparada con todos los modelos.
- **Procesos.** En esta unidad se hace uso de las unidades anteriores, definiendo la estructura de ejecución principal de los procesos que se ejecutarán de forma paralela, es decir los threads. Concretamente, se definen las clases *TProcAdquisic*, *TThreadContorno*, *TThreadLineas* y *TThreadGrafo*.
- **Interface.** Dentro de este módulo se encapsulan todas las cuestiones relacionadas con la interface de usuario. De esta manera, todo el resto de los módulos será completamente independiente de la interface, garantizando de esta forma la reutilización y extensibilidad del sistema.

A.3. Diseño de clases.

Como ya hemos comentado, la implementación ha sido realizada utilizando un lenguaje de programación orientado a objetos. De esta forma, la cuestión fundamental en el análisis y diseño será el diseño de clases. En este apartado realizaremos una breve descripción de las principales clases implementadas, indicando para cada una de ellas los métodos y atributos más relevantes.

Clase TPunto.

Esta clase describe objetos que representan los puntos de la imagen, ya sea para representar los dibujos de borde o los extremos de los segmentos.

Atributos. *X*, *Y* : de tipo entero, representan las coordenadas del punto en la imagen.

Métodos. *Create(X, Y: entero)*: crea un nuevo objeto punto.

Distancia(OtroPunto: TPunto): calcula la distancia con otro objeto punto.

Clase TLinea.

Los objetos de esta clase son segmentos, representados por los puntos extremos del mismo, en los que se incluye una propiedad de orientación (un punto es la cabeza y el otro la cola).

Atributos. *Cabeza, Cola:* de tipo *TPunto*, representan los puntos extremos.

Métodos. *Create(Cabeza, Cola: TPunto)*: crea un nuevo objeto línea, a partir de dos puntos.

LadoPunto(Pt: TPunto): calcula el lado (derecho o izquierdo) de un punto respecto del segmento (según la orientación definida).

PosPuntoSegm(Pt: *TPunto)*: devuelve la posición relativa al segmento, de un punto que está en la misma línea que el segmento (la posición puede ser: en el medio del segmento, en uno de los extremos o fuera del segmento).

Intersecta(OtraLinea: TLinea): calcula y devuelve el punto intersección con la otra línea. En caso de no existir devuelve *nulo*.

Pinta(Lugar: *TCanvas)*: pinta el segmento en el lugar indicado como parámetro.

Clase TEnlace.

Esta clase representa las posibles posiciones relativas de un segmento respecto de otro.

Métodos. *Create(Valor: entero)*: crea un nuevo enlace, del tipo dado en *Valor*.

Opuesto: devuelve el valor del enlace suponiendo que el segmento es considerado con la orientación contraria.

Coste(OtroEnlace: TEnlace): devuelve el valor del coste de comparación entre enlaces, entre el objeto receptor y el objeto *OtroEnlace*.

Clase TGrafo.

Los objetos de esta clase representan grafos de aristas, en los cuales los nodos son segmentos y los arcos son los tipos de enlaces entre segmentos. Los grafos son representados mediante matrices de adyacencia.

Atributos. *Nodos:* array de *TLinea*, representa cada uno de los segmentos asociados a los nodos del grafo.

MatAdy: array bidimensional de *TEnlace*, es la matriz de adyacencia donde se encuentran los arcos etiquetados.

Métodos. *Create*: crea un nuevo objeto grafo vacío, es decir sin nodos.

CreateLista(Nodos: lista de TLinea): crea un nuevo grafo, incluyendo un nodo para cada uno de los segmentos de la lista pasada como parámetro.

CalculaGrafo: realiza el proceso de búsqueda de propiedades no accidentales, para un grafo nuevo, introduciendo las etiquetas en los arcos del grafo.

Pinta(Lugar: *TCanvas)*: pinta el grafo en el lugar indicado como parámetro.

Clase TEmparej.

Un objeto de la clase *TEmparej* representa un emparejamiento entre dos grafos, que es calculado mediante la ejecución del algoritmo de asignación graduado para comparación de grafos. El emparejamiento describe los nodos de un grafo que se corresponden con los del otro grafo.

Atributos. *Grafo1, Grafo2:* de tipo *TGrafo*, contienen los dos grafos comparados.

NodosAsigna: lista de pares de enteros, contiene la equivalencia entre los números de nodos de los dos grafos, tras ejecutar el algoritmo.

BetaInic, BetaFin, BetaIncr, MaxIterPpal, MaxIterNorm, CritConvPpal, CritConvNorm: de tipo real, indican los parámetros necesarios para la ejecución del algoritmo. Toman valores por defecto adecuados, al crear el objeto.

CosteTotal: Indica el coste total de comparación entre los grafos, calculado después de la ejecución del algoritmo.

Métodos. *Create(Grafo1, Grafo2: TGrafo)*: crea un nuevo objeto emparejamiento, sin ejecutar el algoritmo, es decir la lista de emparejamientos es vacía.

Empareja: ejecuta el algoritmo de comparación de grafos, con los parámetros indicados en los atributos. Se calcula el valor del coste total de asignación y la lista de nodos emparejados.

Pinta(Lugar1, Lugar2: *TCanvas)*: pinta los grafos en los lugares indicados como parámetros, etiquetando los nodos emparejados con un mismo número.

Clase TModelo.

Esta clase describe modelos tridimensionales de los objetos del dominio, que son representados con tres vistas distintas del objeto. Contiene métodos para realizar la comparación y el alineamiento.

Atributos. *Vistas:* array de tamaño 3 de *TGrafo*, en el que se representa cada una de las vistas consideradas.

EmpModelo12, EmpModelo13: de tipo *TEmparej*, describe las equivalencias entre los segmentos del grafo asociado a la vista 1, con los segmentos de los otros dos grafos.

Métodos. *Create(Vista1, Vista2, Vista3: TGrafo)*: crea un nuevo objeto modelo, con los tres grafos pasados como parámetro, buscando los emparejamientos entre los segmentos. Los grafos deben ser isomorfos, para que se pueda crear el modelo.

ComparaModelo(Escena: TGrafo): realiza una comparación del modelo con el grafo pasado como parámetro. Devuelve un objeto de la clase *TEmparej*, en el que se indica el valor del emparejamiento.

AlineaModelo(Escena: TGrafo; Emparej: TEmparej): lleva a cabo el proceso de alineamiento del modelo con el dibujo de líneas correspondiente a la escena. Las equivalencias entre los nodos de *Escena* y los del modelo están dadas en *Emparej*. Devuelve un grafo, que es el resultado del alineamiento.

Anexo B. Pruebas realizadas.

B.1. Introducción.

A lo largo de todo el desarrollo teórico del proceso de visión, hemos hecho un énfasis especial en la comprobación práctica de las ideas propuestas. Algunos ejemplos han sido incluidos en las partes en las que se introducía una nueva técnica.

Adicionalmente, se ha realizado un extenso conjunto de pruebas, sobre las distintas etapas y sobre el proceso de visión completo, para evaluar empíricamente la validez del enfoque desarrollado. La descripción y los resultados de estas pruebas se encuentran incluidos dentro de este anexo. El conjunto de objetos sobre los que se han realizado las pruebas consta de los 7 poliedros mostrados en la siguiente figura. Para simplificar se usarán nombres simbólicos para los objetos, y no los nombres geométricamente correctos.

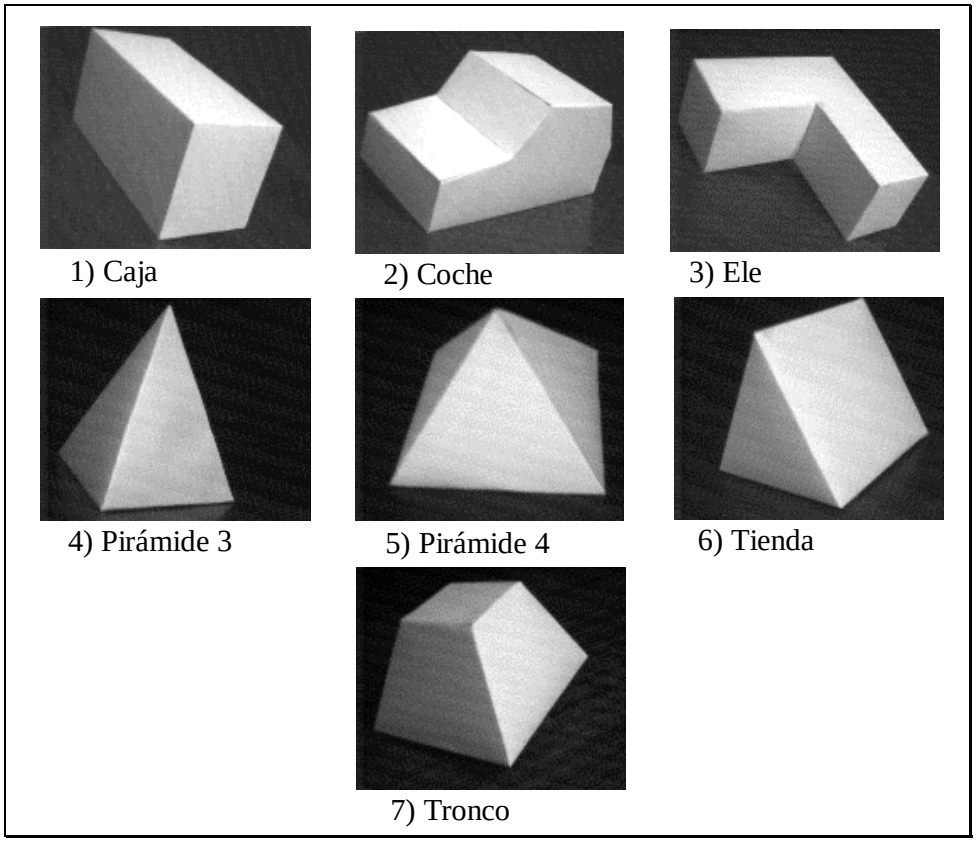

Figura 35 *Objetos de entrada para la realización de pruebas*

Algunas de las pruebas son, en bastante medida, independientes de los dibujos concretos sobre los que son aplicadas, por lo que no se indicarán expresamente los objetos que fueron usados. Esto ocurre principalmente con los niveles inferiores. El conjunto de pruebas ha sido dividido en tres bloques, cada uno de los cuales se refiere a uno de los tres niveles de procesamiento existentes. Las pruebas de alto nivel constituyen una validación de todo el sistema, ya que se basan en los resultados obtenidos por los procesos anteriores. Tras cada una de las pruebas se realizan unos breves comentarios sobre los resultados obtenidos.

Todas las pruebas han sido ejecutadas utilizando un ordenador **Pentium** a 200 Mhz, con 32 Mbytes de memoria. El sistema operativo sobre el que se ejecutan es Windows 95. Estos parámetros son especialmente relevantes para las pruebas relativas a tiempos de ejecución. Para la adquisición se ha utilizado la cámara QuickCam, capaz de obtener imágenes de calidad media, con una resolución máxima de 64 niveles de grises.

B.2. Detección de bordes.

Las pruebas de detección de bordes han sido aplicadas sobre los cinco operadores de borde que se vieron en la memoria del proyecto. En concreto, los operadores utilizados son:

Operador	Tamaño de máscara
Elemental	2x1
Roberts	2x2
Prewitt	3x3
Sobel	3x3
Canny	3x3

En el operador de Canny, el tamaño de la máscara se refiere al valor de σ para el filtrado gausiano. Los valores de los umbrales utilizados son ajustados en cada caso.

Las pruebas ejecutadas hacen referencia a dos aspectos: el tiempo de ejecución de los operadores y la calidad de los bordes obtenidos.

Tiempos de ejecución de los operadores.

Para la medida del tiempo de ejecución se han realizado pruebas variando el tamaño de las imágenes y el número de puntos de borde obtenido en cada caso. Estos tiempos incluyen la aplicación del operador y el almacenamiento de los puntos de borde resultantes en una lista. Para resoluciones de 160x120 se han realizado 500 ejecuciones del algoritmo, mientras que para el tamaño de 320x240 se ha ejecutado 125 veces. El número de puntos de borde ha sido modificado cambiando las imágenes de entrada y los umbrales de los algoritmos para conseguir ese valor. Los resultados obtenidos son los siguientes.

Tamaño de imágenes: 160x120 **Ejecuciones: 500**

	500 puntos de borde		1.000 puntos de borde		2.000 puntos de borde	
Operador	Tiempo total (s)	Tiempo medio (ms)	Tiempo total (s)	Tiempo medio (ms)	Tiempo total (s)	Tiempo medio (ms)
Elemental	20.59	41.18	20.73	41.46	20.84	41.68
Roberts	23.78	47.56	23.93	47.86	23.99	47.98
Prewitt	32.98	65.96	33.29	66.58	33.34	66.68
Sobel	34.05	68.1	34.4	68.8	34.74	69.48
Canny	62.35	124.7	92.92	185.84	84.23	168.46

Tamaño de imágenes: 320x240 **Ejecuciones: 125**

	500 puntos de borde		1.000 puntos de borde		2.000 puntos de borde	
Operador	Tiempo total (s)	Tiempo medio (ms)	Tiempo total (s)	Tiempo medio (ms)	Tiempo total (s)	Tiempo medio (ms)
Elemental	22.31	178.48	23.24	185.92	23.04	184.32
Roberts	23.12	184.96	23.25	186	23.4	187.2
Prewitt	34.3	274.4	33.9	271.2	34.23	273.84
Sobel	34.32	274.56	34.43	275.44	34.32	274.56
Canny	68.78	550.24	63.53	508.24	69.96	559.68

Se puede apreciar que, en general, todos los operadores son muy rápidos, no tardando ninguno de ellos más de una fracción de segundo en ejecutarse. En todos los casos, el operador elemental es el más rápido y prácticamente independiente del número de puntos de borde resultantes. Los tiempos de los otros operadores básicos son mayores, aunque se encuentran dentro del mismo rango de tiempo. Para los operadores con tamaño de máscara de 3x3, el aumento del tiempo está entorno al 50%. El aumento del tiempo de ejecución es mayor en el operador de Canny. Para las imágenes de 160x120 el tiempo es 4 veces superior al tiempo del operador elemental, y para 320x240 es algo menos de 3 veces más lento.

A pesar de ello, los resultados obtenidos con el operador de Canny pueden ser considerados como muy buenos. Recordemos que este operador realiza un procesamiento más completo de la imagen que la simple aplicación de una máscara de convolución, por lo que resulta lógico este elevado incremento del tiempo. Para tamaños de imagen pequeños, el tiempo de una ejecución no supera los 200 ms, lo cual permitiría un refresco de la imagen de bordes de 5 veces por segundo, suficiente para los propósitos requeridos. Para la resolución de 320x240 el tiempo es superior al medio segundo, permitiendo un refresco de algo menos de 2 veces por segundo.

Calidad de los operadores.

Las pruebas de calidad son más subjetivas, ya que dependen de lo que sea considerado como un buen borde. Para llevarlas a cabo, se ha definido el concepto de "borde completo". Consideraremos que el borde es completo si todas las aristas del dibujo son representadas por un borde continuo de puntos (es decir, si no existen discontinuidades de borde en las aristas). De esta forma, la medición consiste en ajustar los umbrales para cada operador, hasta conseguir que el borde sea continuo, contando después el número de puntos de borde obtenidos por el operador.

El valor del número de puntos de borde está relacionado inversamente con las propiedades de eliminación de ruido y de generación de bordes gruesos. Un borde completo será mejor cuantos menos puntos contenga (ya que se describe mejor la imagen, generando menos redundancia). Este test ha sido realizado sobre los siguientes 5 ejemplos, con imágenes de tamaño 160x120.

Número de puntos de borde

Operador	1. Caja	2. Coche	3. Ele	4. Tienda	5. Tronco
Elemental	1322	3262	2022	788	1745
Roberts	1002	2266	1790	722	1414
Prewitt	1007	1886	1859	810	1410
Sobel	998	1846	1669	785	1390
Canny	530	727	734	496	563

Como se ve, el operador de Canny obtiene en todos los casos bordes de más calidad, que contienen muchos menos puntos de borde. El operador elemental, que tenía el menor tiempo de ejecución, resulta ser el que genera bordes de mayor tamaño, debido al ruido y al grosor de los bordes. Aunque este problema es menor en los otros operadores básicos, sin embargo estos también dan lugar a bordes malos. De media, estos operadores obtienen bordes con 2.2 veces más puntos que el operador de Canny. Con el operador de Canny, el borde no sólo es menor sino que se aproxima mucho al borde óptimo, es decir el que consta de un sólo pixel de anchura por línea.

La cuestión del número de puntos por borde es fundamental para el posterior proceso de detección de segmentos, como se verá más adelante. Además, se ha visto que todos los operadores tienen un reducido tiempo de ejecución, que resultará despreciable frente a los requerimientos de otros niveles. Por lo tanto, de todos los operadores vistos el que será más adecuado para este proyecto es el operador de Canny.

En cuanto a los umbrales, los valores típicos para la obtención de un buen borde son diferentes para cada uno de los operadores. Estos valores son aproximadamente sobre los siguientes.

Operador	Umbral mínimo	Umbral máximo
Elemental	40	-
Roberts	50	-
Prewitt	115	-
Sobel	160	-
Canny	10	25

En general, el valor del umbral depende de los tamaños de máscara utilizados, del nivel de contraste y la resolución (en cuanto al número de niveles de gris) de las imágenes empleadas. El operador de Canny tiene la gran ventaja de que los umbrales son aplicables para un amplio rango de variación del contraste de las imágenes, lo que no ocurre con los otros.

B.3. Detección de segmentos.

Tiempos de ejecución del algoritmo.

En la ejecución del algoritmo EMH, la medida del tiempo de ejecución es un tanto subjetiva. Como vimos en el desarrollo, el algoritmo realiza continuamente un proceso repetitivo en el que no hay pruebas de convergencia global, por lo que no se puede identificar una finalización exacta del proceso. Además, esta comprobación no sería de gran utilidad ya que en muchas ocasiones no se alcanza una verdadera situación estable de convergencia.

Por lo tanto, la medida del tiempo de ejecución se basa en el número de veces que debe ejecutarse el paso principal del algoritmo (enumerado como paso 2, en el esquema descrito en la página 14) hasta alcanzar una solución que contenga una línea adecuada por cada una de las líneas del dibujo de bordes. Nótese que este es un concepto subjetivo, ya que debemos determinar cuando las líneas describen de forma adecuada el dibujo de bordes.

Para la entrada al algoritmo utilizada el operador de Canny. Se han realizado pruebas variando el número de puntos de borde de la entrada, y el número de iteraciones de los pasos E-M (paso 2.1 en el algoritmo). Para cada una de las pruebas se ha medido el tiempo medio de ejecución de un ciclo (es decir del paso principal del algoritmo, el paso 2) y el número de ciclos necesarios para llegar a una solución buena. Lógicamente, el tiempo de ejecución total es el producto de estas cantidades. Los resultados obtenidos son los siguientes.

	Iteraciones paso E-M: 8			**Iteraciones paso E-M: 15**		
Nº puntos de borde	Tiempo/ciclo (s)	Nº de ciclos	Tiempo total (s)	Tiempo/ciclo (s)	Nº de ciclos	Tiempo total (s)
200	0.102	11	1.122	0.151	9	1.359
400	0.2484	9	2.2356	0.413	7	2.891
600	0.378	8	3.024	0.612	7	4.284

Los tiempos de ejecución son comparativamente superiores a los de los operadores de bajo nivel. Estos tiempos están en todos los casos por encima de un segundo, necesitando en el peor caso hasta 4 segundos. Se puede ver que el tiempo aumenta de directamente con el número de puntos de borde, debido a que el algoritmo realiza varios cálculos sobre todos estos puntos. El tiempo de ejecución total también aumenta cuanto mayor es el número de segmentos existentes (independientemente del número inicial de segmentos).

Por otro lado, se ve que al aumentar el número de iteraciones del paso E-M (de 8 a 15) disminuye el número de ciclos, pero al aumentar el tiempo de cada ciclo el resultado es que el tiempo total aumenta.

Calidad de los dibujos de líneas obtenidos.

Se han aplicado pruebas utilizando los distintos operadores de borde, para comprobar la calidad de la solución obtenida en cada caso. En estas pruebas nos fijamos fundamentalmente en el número de líneas resultantes. El valor óptimo corresponde al número de aristas que realmente existen en la escena, según el objeto que aparece. Estos son los resultados obtenidos.

Número de líneas por figura

Operador	1. Caja	2. Coche	3. Ele	4. Tienda	5. Tronco
Elemental	24	37	19	12	25
Roberts	32	38	25	15	30
Prewitt	37	40	36	11	36
Sobel	34	42	34	10	35
Canny	11	16	17	8	13
Número óptimo	9	16	15	6	9

Como era de esperar, los mejores resultados son los obtenidos con el operador de Canny. El número de líneas está muy próximo al óptimo, no superándolo en más de 2 segmentos en la mayoría de los casos. Para los otros operadores el efecto de la aparición de bordes gruesos es que se introduce un número de líneas muy grande, hasta 4 veces superior al óptimo. Esto quiere decir que una misma arista será modelada por muchos segmentos colineares, lo que podrá provocar errores en los procesos posteriores. Esto no ocurrirá si utilizamos el operador de Canny, para el que además se ha comprobado que el algoritmo EMH converge más rápidamente.

B.4. Interpretación de dibujos de líneas.

Igual que las anteriores, las pruebas realizadas en la etapa de alto nivel han sido dirigidas hacia dos aspectos diferenciados: los tiempos de ejecución y los resultados de la interpretación. Todos los dibujos de líneas de entrada para las pruebas han sido los resultantes de la fase de detección de segmentos, aplicada sobre dibujos de borde de escenas reales. De esta forma, los resultados de la interpretación serán una prueba acumulada de todo el proceso.

En cuanto al tiempo de ejecución, ha sido medido de forma independiente para los pasos de que consta el proceso de alto nivel: la creación de grafos y la comparación de grafos.

Tiempo de ejecución del proceso de creación de grafos.

El tiempo de ejecución de la creación de los grafos está en función del número de segmentos de entrada, así como del número de segmentos que resultarán tras la creación del grafo. Para realizar las pruebas, ha sido ejecutado el algoritmo 1.000 veces con 8 ejemplos de prueba de diferentes objetos, obteniendo los resultados siguientes.

Tiempo de creación de grafos **Ejecuciones: 1.000**

Ejemplo	Segmentos iniciales	Nodos resultantes	Tiempo total (s)	Tiempo medio (ms)
1	7	5	10.66	10.66
2	9	6	12.9	12.9
3	12	8	17.42	17.42
4	14	9	21.17	21.17
5	17	12	26.42	26.42
6	20	16	48.37	48.37
7	26	9	23.12	23.12
8	35	9	29.03	29.03

Como puede verse, el tiempo de ejecución es muy reducido para todos los casos, no superando en la peor de las situaciones los 50 ms. Por lo tanto, este tiempo es despreciable frente al consumido por la mayoría de los restantes procesos. Curiosamente, se puede apreciar que el tiempo de ejecución depende más del número de nodos resultantes que de los segmentos de partida. Esto es debido a que normalmente la mayoría de los segmentos eliminados son debidos a la existencia de colinearidad, por lo que serán suprimidos en la primera comprobación del proceso de creación de grafos. Si después de esta fase no se han eliminado muchos segmentos (como es el ejemplo 6) todas las demás comprobaciones deberán realizarse sobre un número muy grande de segmentos, por lo que el tiempo será mayor.

Tiempo de ejecución del proceso de comparación de grafos.

Para el algoritmo de comparación de grafos, el tiempo de ejecución depende fundamentalmente de dos parámetros: el valor del incremento de β y el número de nodos. También puede variar sensiblemente en función de los grados de similitud de los grafos de entrada, aunque esta variación es más difícil de cuantificar.

El parámetro *Incremento_β* controla la velocidad a la que converge la solución. Debe ser siempre mayor que 1, ya que el incremento es realizado mediante una multiplicación (ver punto

2.2, en la página 38). A lo largo del algoritmo, el valor de β varía entre un valor inicial y un valor final. Cuanto menor sea *Incremento_β* más lentamente convergerá, y más tiempo tardará en ejecutarse el algoritmo. Los valores inicial y final de β han sido establecidos de forma constante, siendo de 0.5 y 10, respectivamente. Los tiempos de ejecución obtenidos con una comparación de 12 nodos (diferentes en 3 de ellos) son los siguientes.

Tiempo de comparación Ejecuciones: 100

Incremento_β	Tiempo total (s)	Tiempo medio (ms)
1.075	92.35	923.5
1.15	57.53	575.3
1.3	40.11	401.1
1.5	32.59	325.9
1.8	28.41	284.1
2	27.64	276.4

En el caso más lento, el tiempo de comparación está próximo a 1 segundo. Sin embargo, a medida que aumentamos *Incremento_β*, el tiempo de ejecución disminuye de forma exponencial, llegando a casi un cuarto de segundo para el valor 2. Aunque este valor puede ser aumentado, no debe ser muy alto ya que en otro caso la comparación puede dar un mal resultado. En la práctica, se ha comprobado que con *Incremento_β*=1.5 se obtiene una buena relación entre tiempo de ejecución y bondad de la solución (y en algunos casos incluso con un valor algo mayor).

En cuanto al número de nodos, el tiempo de ejecución será mayor cuantos más nodos hayan. Como se vio en el desarrollo, este tiempo era de orden *O(lm)*, donde *l* y *m* son el número de arcos de los grafos comparados. En la práctica, puesto que el número de enlaces máximo depende del cuadrado del número de nodos, el tiempo de ejecución será del tipo $O(n^4)$, siendo *n* el número de nodos por grafo (supuesto el mismo para los dos grafos). Este rápido aumento del tiempo de ejecución se puede comprobar en las siguientes ejecuciones. El valor de *Incremento_β* utilizado es de 1.5, y los grafos comparados constan ambos del mismo número de nodos.

Tiempo de comparación Ejecuciones: 100

Número de nodos	Tiempo total (s)	Tiempo medio (ms)
5	1.19	11.9
7	2.05	20.5
10	12.97	129.7
12	32.59	325.9
15	64.34	643.4
20	250.3	2503

Para un número reducido de nodos, el tiempo de ejecución es extremadamente rápido, no superando los 12 ms. Sin embargo, a medida que aumenta el número de nodos, el tiempo aumenta muy rápidamente. Para el caso de 20 nodos este tiempo se eleva a más de 2.5 segundos. Aunque esto es una evidente limitación para el uso de grafos grandes, normalmente los grafos constan entre 5 y 16 nodos, por lo que el algoritmo podrá ser aplicado en un tiempo no excesivamente grande.

Resultados de la creación de grafos.

Para las pruebas de interpretación, se ha utilizado el conjunto de 7 modelos antes visto. Para cada uno de los modelos se han adquirido un total de 10 imágenes (con un tamaño de 160x120), a las cuales se les han aplicado todos los procesos hasta la interpretación. Estas suponen unas pruebas globales de todo el sistema, ya que se acumulan todas los resultados de los distintos procesos, no sólo en el alto nivel.

Una primera medida es el porcentaje de líneas de ruido existentes en el grafo de aristas asociado a cada una de las imágenes de entrada. Con estos valores se mide desde la calidad de la ad-

quisición, pasando por las etapas de bajo y medio nivel, hasta la creación del grafo. Los resultados medios obtenidos para los 10 dibujos utilizados en cada objeto son los mostrados en la siguiente tabla. El porcentaje de error indica la cantidad de líneas faltantes o sobrantes, frente a una representación perfecta del objeto correspondiente.

Figura	Nº de segmentos óptimo	Porcentaje de error
1. Caja	9	1.1
2. Coche	16	12.4
3. Ele	15	8.6
4. Pirámide 3	5	0
5. Pirámide 4	8	10.7
6. Tienda	6	0
7. Tronco	9	7.3

Los grafos que contienen la representación perfecta de los objetos son los que se muestran en la figura siguiente. Las uniones entre segmentos son marcadas con un pequeño círculo.

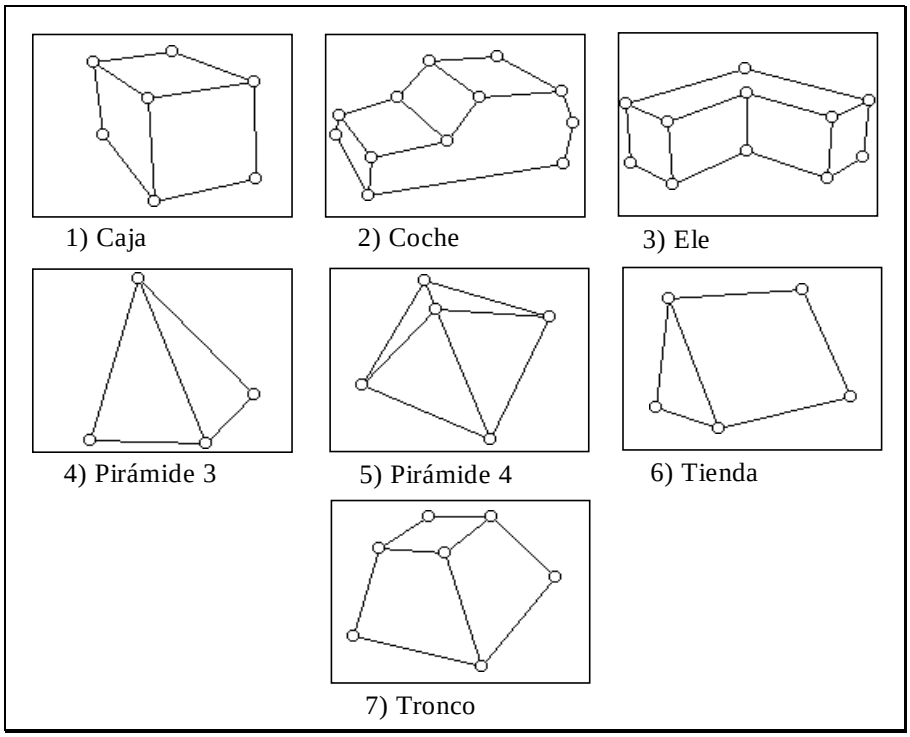

Figura 36 *Grafos óptimos asociados a los objetos*

Para los tipos de objetos más simples (la pirámide 3 y la tienda), el proceso es capaz de obtener una representación perfecta de las aristas en todos los casos, no cometiendo ningún error. Para el objeto con forma de caja, el resultado es también bastante bueno, aunque en este caso se cometen algunos errores, debidos a líneas inexistentes por falta de contraste en las imágenes de entrada. Para las figuras más complejas, los grafos contienen más error. No obstante, en el peor caso (para la figura 2) este error no está por encima del 12.4%, lo que supone una media de 2 líneas erróneas por cada grafo.

En conjunto, el error cometido considerando todas las figuras es del 5.7%, lo cual es un resultado que puede ser considerado como bastante positivo.

Grado de bondad de la medida de comparación de grafos.

En general, se puede decir que una medida es buena si es discriminante e invariante. Será discriminante cuando obtenga valores malos para objetos poco parecidos entre sí, y se dirá que es invariante si devuelve buenos valores para distintas realizaciones de un mismo objeto.

Para medir el grado de discriminación de la medida, se han comprobado los valores de los costes de comparación entre los grafos perfectos asociados a los objetos. El objetivo es que el coste de comparación sea 1 cuando se compara un objeto consigo mismo y muy bajo cuando son comparados objetos distintos. La medida de coste utilizada es la que aparece en la página 40, utilizando los valores: C_1= 2, C_2= -1 y C_3= 0.

Costes de comparación de grafos

	Modelo						
Escena	1. Caja	2. Coche	3. Ele	4. Pirámide 3	5. Pirámide 4	6. Tienda	7. Tronco
1. Caja	1	0.53	0.53	0.4	0.36	0.73	1
2. Coche	0.89	1	0.89	0.4	0.36	0.6	0.89
3. Ele	0.89	0.89	1	0.4	0.36	0.73	0.89
4. Pirámide 3	0.22	0.13	0.13	1	0.63	0.67	0.22
5. Pirámide 4	0.27	0.19	0.23	0.63	1	0.8	0.27
6. Tienda	0.49	0.24	0.29	0.67	0.6	1	0.49
7. Tronco	1	0.53	0.53	0.4	0.36	0.73	1

Como se puede ver, la medida no es simétrica, uno de los grafos comparados es el modelo y el otro el objeto desconocido de la escena. El coste está en función del porcentaje de nodos del modelo que han sido emparejados.

La medida es claramente discriminante en la mayoría de los casos. En casi todos ellos el coste de comparación entre objetos distintos es inferior a 0.67, con lo cual el coste será claramente mayor para el objeto adecuado (aun existiendo algo de ruido).

Por otro lado, destaca el hecho de que los costes para las filas y las columnas de los objetos 1 y 7 son exactamente iguales. Como habíamos previsto durante el desarrollo teórico, esto es debido a que los grafos asociados a estos objetos son isomorfos, por lo que permanecerán indistinguibles para el proceso de comparación. Algo parecido ocurre con los objetos 2 y 3, que estructuralmente resultan muy parecidos entre sí. Se trata de una carencia de la estructura de grafos definida, más que de un problema del algoritmo de comparación.

Por último, para medir la invarianza de la métrica, se ha realizado una prueba de clasificación. Cada una de las 10 vistas de los objetos es comparada con todos los modelos, obteniendo como resultado el modelo con mejor valor de coste. Debido al isomorfismo antes comentado entre las figuras 1 y 7, se ha eliminado de estas pruebas la figura número 7. A continuación se muestra la matriz de clasificación, donde se indica el número de ejemplos de cada objeto clasificado en cada uno de los modelos existentes.

	Clase de salida						
Objeto escena	1. Caja	2. Coche	3. Ele	4. Pirámide 3	5. Pirámide 4	6. Tienda	% Acierto
1. Caja	10	0	0	0	0	0	100
2. Coche	0	6	4	0	0	0	60
3. Ele	0	2	8	0	0	0	80
4. Pirámide 3	0	0	0	10	0	0	100
5. Pirámide 4	0	0	0	3	7	0	70
6. Tienda	0	0	0	0	0	10	100
						TOTAL	**85**

El porcentaje de acierto indica el número de vistas de los objetos que son clasificadas en las clases correctas, es decir los valores de la diagonal principal.

Para los objetos que dieron lugar a poco error, los porcentajes de acierto son del 100%, todas las vistas son bien clasificadas. Para objetos que contienen más ruido el porcentaje de acierto disminuye, aunque nunca por debajo del 60%. El peor resultado se obtiene para la figura con forma de coche, debido a su gran similitud con la ele. Estos resultados podrían ser mejorados por un proceso posterior de alineamiento, en el que la hipótesis fuera verificada de forma más precisa. No

obstante esto no ha sido comprobado extensamente, ya que el alineamiento sólo ha sido implementado a modo experimental.

En definitiva, después de aplicar todo el proceso de interpretación desarrollado (excepto el alineamiento) sobre un conjunto 70 imágenes pertenecientes a 7 modelos, obtenemos que el porcentaje de aciertos total es del 85%. Es decir, en 61 de las pruebas realizadas se obtiene un buen resultado y sólo en 9 la clasificación final es errónea. Esto puede ser valorado como un buen resultado, teniendo en cuenta que para la adquisición se ha utilizado una cámara muy sencilla, capaz de ofrecer una calidad limitada.

Por otro lado, la valoración final también debe tener en cuenta el tiempo de ejecución del proceso de forma conjunta. Hemos visto que los procesos más costosos se encontraban en la detección de segmentos y en el algoritmo de comparación de grafos. En cualquier caso, el tiempo acumulado en todas las etapas no supera los 2.5 segundos, para el tipo de entradas utilizadas. En el caso del seguimiento de objetos en tiempo real, este tiempo disminuye de manera muy significativa, debido a que el algoritmo de detección de segmentos EMH (el que más retardo genera) es capaz de obtener soluciones muy rápidamente cuando hay una pequeña variación en la entrada.