UNIVERSIDAD POLITÉCNICA DE VALENCIA
FACULTAD DE INFORMÁTICA

Reconstrucción de modelos sólidos 3D a partir de vistas ortográficas 2D utilizando técnicas de combinación de cuerpos elementales.

PROYECTO FIN DE CARRERA

PRESENTADO POR:
Lorenzo J. Muñiz Márquez

DIRIGIDO POR:
D. José María Gomis Martí

INDICE

Capítulo 1
INTRODUCCIÓN

En el proceso de diseño asistido por computador en la ingeniería mecánica y fabricación, la representación automática y la construcción de modelos sólidos con ordenadores es un paso vital. Aunque obtener proyecciones 2D para un objeto 3D dado es sencillo, la operación inversa llega a ser algo implícita y complicada. Estas dificultades se originan de la pérdida de información semántica cuando un objeto 3D es representado con proyecciones 2D. Este proceso de obtener un objeto 3D a partir de sus proyecciones 2D es lo que llamamos *Reconstrucción 3D.*

El objetivo de este capítulo es realizar una descripción detallada de todos los conceptos y definiciones que intervienen en las representaciones 2D de objetos. Además veremos distintos métodos de reconstrucción 3D de objetos, como pueden ser: la reconstrucción 3D a partir de una vista o a partir de varias vistas. Veremos también una serie de antecedentes de los procesos de reconstrucción 3D a partir de representaciones 2D. Describiremos de forma resumida este proyecto final de carrera, así como sus objetivos, y las herramientas y entorno de desarrollo utilizados en la resolución de este proyecto.

1.1. LA REPRESENTACIÓN 2D DE OBJETOS.

En esta sección se va a realizar una descripción detallada de todos los conceptos y definiciones que intervienen en las representaciones 2D de objetos, así como los distintos tipos de representaciones 2D que existen.

1.1.1. LA EXPRESIÓN GRÁFICA EN LA INGENIERÍA.

La actividad del ingeniero se centra en la resolución de determinados problemas y en la difusión de las diferentes tecnologías. Es evidente que la persona que debe cubrir estos objetivos deberá poseer unos sólidos conocimientos teóricos y en particular el dominio de determinadas tecnologías. Pero, además, para poder aplicarlas y para poder comunicar sus propuestas, necesitará igualmente de unos medios de expresión que se lo permitan plenamente. Necesitará, también, saber expresarse gráficamente.

Además, siendo la realización la principal función del ingeniero, su actividad forzosamente ha de ser eminentemente creadora. En consecuencia, en su formación deben valorarse positivamente todas aquellas disciplinas que fomenten y faciliten su creatividad. Y aquí ha de quedar patente el papel insustituible de la *expresión gráfica* ya que al igual que la escritura y el pentagrama constituyen elementos determinantes para el ejercicio creativo del escritor y el músico, el dibujo constituye el vehículo necesario para el técnico en su facultad creadora. No en balde, el grafismo y la intuición geométrica son recursos esenciales para el técnico de cara a la visión de los fenómenos en su conjunto, constituyendo, al mismo tiempo, un elemento de acercamiento entre sus conocimientos teóricos y la realidad. Todo ello hace que un eficaz y completo adiestramiento en el lenguaje gráfico sea de absoluta importancia para el ingeniero, lenguaje que, como veremos a continuación, va a precisar doblemente.

Por un lado, en la resolución de un determinado problema, el ingeniero, haciendo uso de la razón e imaginación, inicia un discurso mental en el que, forzosamente, la comunicación de cuantas formas imagina y procesa se produce a través del lenguaje gráfico. Por tanto, éste no solo tiene su razón de ser como técnica de representación formal a desarrollar sobre un determinado soporte, sino que su alcance sobrepasa el ámbito de la representación penetrando en el propio ejercicio intelectual.

Por otro lado, la intervención del ingeniero en las líneas de trabajo que le son propias, aún siendo decisiva, no protagoniza en toda su extensión el desarrollo de aquellas. En general, el proceso que abarca desde la concepción de la idea hasta la materialización de ésta, pasando por las correspondientes actividades proyectuales o de diseño, implica, cuando menos en su última etapa, a terceras personas. Por ello, el trabajo del ingeniero, en su última fase, se centra en la exposición de unos resultados que, para que alcancen los objetivos que los han motivado, han de ser correctamente interpretados. En definitiva, el técnico, llegado un determinado momento, ha de saber transmitir el fruto de sus esfuerzos y tal prerrogativa exige que la antedicha exposición reúna los requisitos necesarios que lo haga posible. Entre los más necesarios están, precisamente, aquellos que hacen de la expresión gráfica un lenguaje, un instrumento insustituible para los técnicos.

1.1.2. OBJETIVOS DEL DIBUJO TÉCNICO.

Anteriormente se ha señalado la función de la expresión gráfica como medio de enlace entre los conocimientos teóricos del técnico y la realidad que lo rodea. Siendo esta tridimensional, constituye un objetivo primordial en la formación del ingeniero el desarrollo de sus facultades mentales especiales, correspondiendo al dibujo técnico la misión de iniciarle a discurrir en tres dimensiones hasta hacerle conseguir un dominio suficiente del espacio.

La necesidad de plasmar los resultados, obliga, en la inmensa mayoría de los casos, al ingeniero a materializar formas tridimensionales por él creadas sobre soportes planos. La representación bidimensional de dichas formas implica el conocimiento de las distintas técnicas de conseguir la relación Espacio-Plano. Es por tanto un objetivo prioritario de esta disciplina el estudio de los *sistemas de representación* cuyos principios rige la *geometría descriptiva*. Además, como consecuencia de dicho estudio y, sobre todo, de la experiencia adquirida a través del subsiguiente ejercicio práctico, el futuro ingeniero debe quedar en condiciones de elegir el sistema de representación más adecuado en función del mensaje que pretende transmitir y del destinatario del mismo.

Las formas entran dentro del cómputo de los elementos habituales en el ejercicio profesional del ingeniero, complementando a las diferentes tecnologías en las distintas actividades de diseño. Por tanto, el conocimiento de su geometría, generación y propiedades resultan esenciales en el desarrollo de dichas actividades. En consecuencia, su estudio hasta conseguir una familiarización adecuada en su manejo, es otro objetivo del dibujo técnico; cuyo logro ha de proveer al ingeniero del vocabulario formal necesario y, en definitiva, argumentar y fomentar su creatividad.

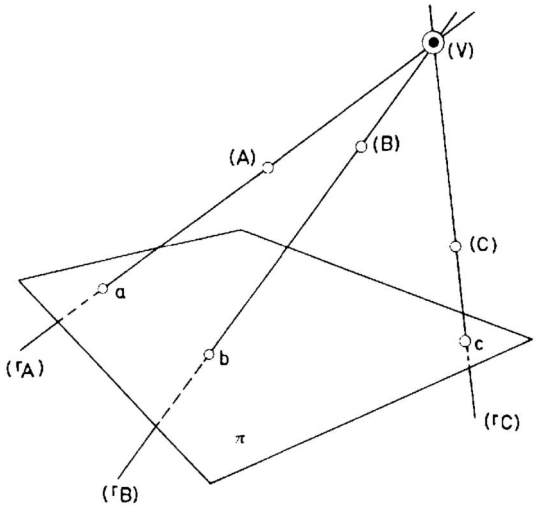
El dibujo técnico en sus distintas aplicaciones ha ido adaptándose a las diferentes tecnologías, adoptando características específicas en cada caso. Esto hace que el ingeniero deba conocer los convencionalismos específicos que dentro del lenguaje gráfico, son aceptados en función de su aplicación a cada técnica. El conocimiento y uso apropiado de dichos convencionalismos que a tal fin disponen las *normas* es otro de los objetivos a alcanzar en el estudio de esta disciplina.

1.1.3. OPERACIONES FUNDAMENTALES UTILIZADAS.

Los distintos sistemas que se emplean para representar sobre un plano las figuras del espacio están basados en la consideración de otras figuras deducidas de la figura dada mediante operaciones de proyección y sección.

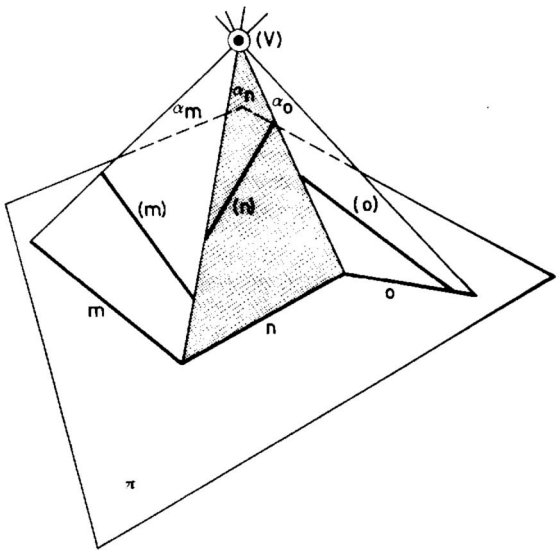
A) PROYECCIÓN

Dado un conjunto de puntos (A), (B), (C), ... y de rectas (m), (n), (o), ... se llama proyección de la figura compuesta por los puntos (A), (B), ... y de las rectas (m), (n), ... desde un punto fijo del espacio (V) (centro de proyección) a la figura compuesta de las rectas (V)(A), (V)(B), ... y de los planos (V)(m), (V)(n), ... que determina el punto (V) con cada uno de los puntos (A), (B), ... y con las rectas (m), (n), ... Podemos ver esto en la figura siguiente:



B) SECCIÓN

Dado un conjunto de planos (α_m), (α_n), (α_o), ... y de rectas (r_A), (r_B), (r_C), ... se llama sección de la figura compuesta de los planos (α_m), (α_n), ... y de las rectas (r_A), (r_B), ... con un plano fijo π (plano del cuadro), a la figura compuesta de las rectas m, n, ... y de los puntos a, b, ..., intersección del plano π con cada uno de los planos (α_m), (α_n), ... y con las rectas (r_A), (r_B), ... Los puntos a, b, ... y las rectas m, n, ... se llaman trazas de las rectas y planos (r_A), (r_B), ..., (α_m), (α_n), ... Podemos ver esto en la figura siguiente:



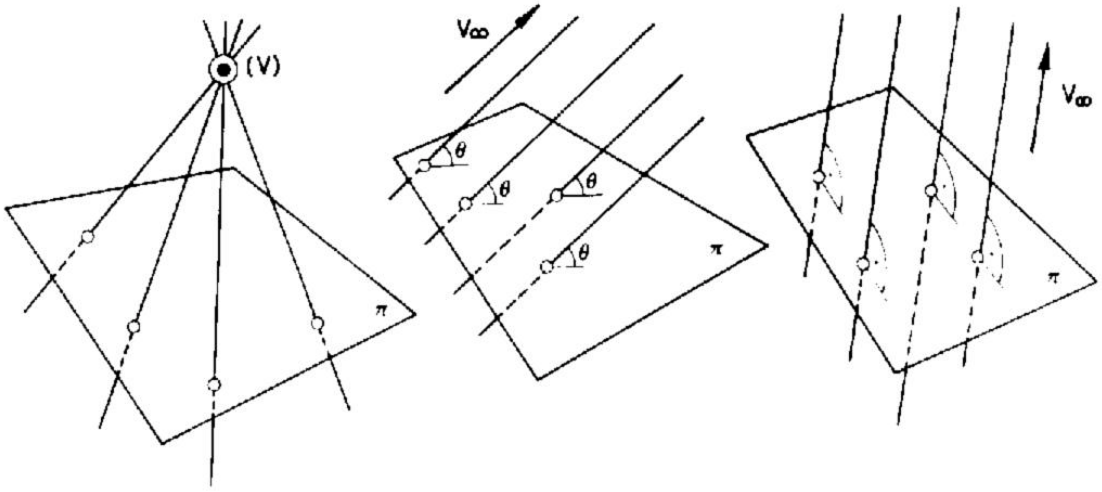
Estas dos operaciones, si se aplican sucesivamente, forman una operación que consiste en proyectar una figura desde un centro (V) y cortar la proyección así obtenida con un plano π que no pasa por (V). La figura así obtenida se dice que es la proyección desde en centro (V) sobre el plano π de la figura dada.

1.1.4. CLASES DE PROYECCIÓN.

El centro de proyección (V) puede ser un punto PROPIO o IMPROPIO; en el primer caso se dice que la proyección es ***central***, y en el segundo, que la proyección es ***paralela*** o ***cilíndrica***.

En este último caso, los rayos proyectantes son todos ellos paralelos a una dirección (la del punto impropio) y cabe distinguir, a su vez, dos clases de proyección, en función de la relación existente entre la dirección de proyección y la orientación del plano de proyección.

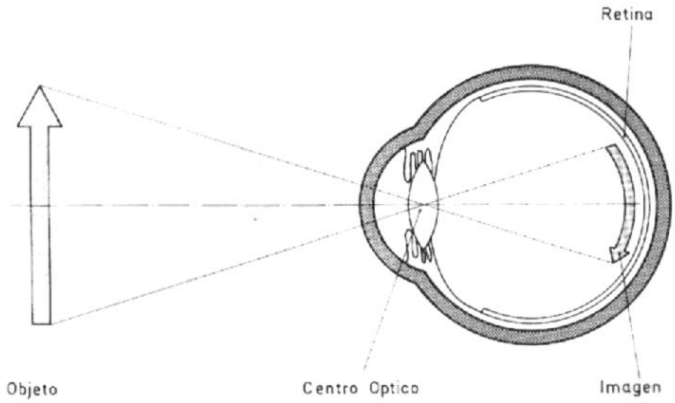
Se llama proyección ***ortogonal*** de una figura sobre un plano a la proyección paralela de la figura sobre este plano, cuando la dirección de los rayos proyectantes es perpendicular al plano; cuando esta dirección no sea ortogonal al plano diremos que es una proyección ***oblicua***.

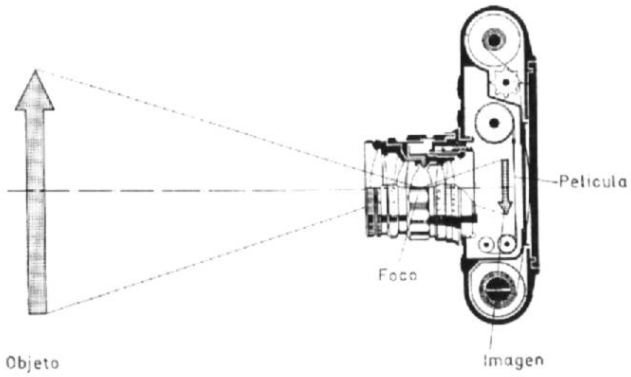


A) *PROYECCIÓN CENTRAL (Centro de Proyección PROPIO)*

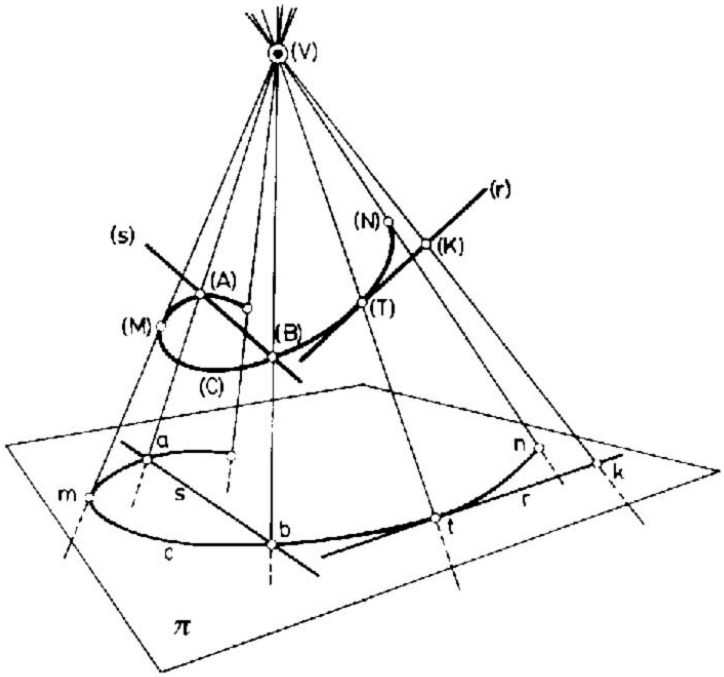
Esta clase de proyección guarda gran similitud con la visión humana y con la fotografía. En la primera el centro de proyección es el centro óptico del ojo y todas las visuales procedentes del objeto pasan por dicho punto y desde él se proyectan sobre la superficie (esférica) de la retina. En la segunda el centro de proyección es el foco del objetivo de la cámara y los rayos visuales del objeto, tras converger a él, se proyectan sobre la superficie (plana) de la película fotográfica. Estos dos ejemplos se pueden ver en la figura siguiente.

La proyección central presenta algún inconveniente, como es el no poder proyectar el propio centro de proyección ni los puntos del plano π_V paralelo al plano π de proyección que contiene al centro (V). De estos últimos, a lo más, podemos obtener las direcciones de sus proyecciones (puntos impropios del plano de proyección).





Las principales propiedades que se conservan INVARIANTES en la proyección central son las de Pertenencia, Intersección y Tangencia. (Además de la Razón Doble, Polaridad y la Separación y Ordenación, dentro de las proyectivas).



En la figura anterior podemos observar las siguientes propiedades:

a) Pertenencia.

Si un punto (M) pertenece a una curva, su proyección m pertenecerá a la de la curva. En particular, si un punto (K) pertenece a una recta, su proyección k pertenecerá también a la recta.

b) Intersección.

Si una recta corta a una curva, su proyección cortará a la de la curva en puntos que son proyecciones de los de intersección de aquéllas. En la figura se observa que si la recta (s) corta a la curva (c) en los puntos (A) y (B), su proyección s corta también a la curva *c* proyección de (C) en a y b, puesto que, por ser (A) y (B) comunes a la recta y a la curva, sus proyecciones también pertenecerán a las proyecciones de ambas.

c) Tangencia.

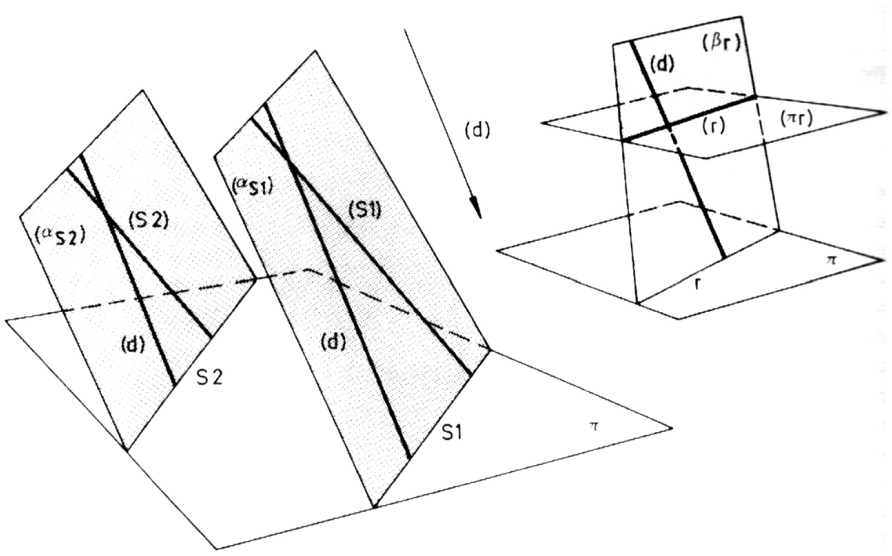
Si una recta (r) es tangente a una curva en un punto (T), su proyección r es también tangente a la proyección de la curva en la proyección t, del punto de tangencia. La constatación de esta invariante se obtiene sin más que particularizar la anterior en el límite, cuando los puntos de intersección tienden a confundirse.

B) PROYECCIÓN CILÍNDRICA OBLICUA (Centro de Proyección IMPROPIO, la dirección del cual forma con el plano de proyección un ángulo menor de 90º y mayor de 0º)

Puede considerarse como un caso particular de la proyección central, razón por la cual las INVARIANTES de la dicha proyección (Pertenencia, Intersección y Tangencia) son aplicables a la proyección cilíndrica. Además de éstas, la proyección paralela posee otras invariantes:

a) Paralelismo.

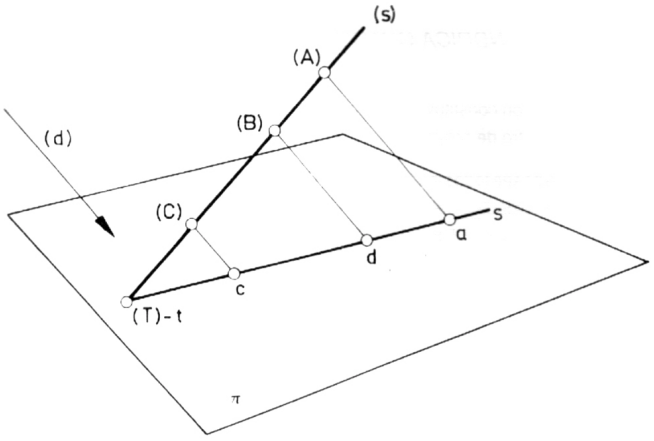
Dos rectas paralelas en el espacio se proyectan cilíndricamente según dos rectas paralelas. En efecto, consideremos dos rectas (s1) y (s2) paralelas en un sistema de proyección cilíndrica dado por un plano π y una dirección (d) de proyección. Las proyecciones de ambas rectas serán las rectas s1 y s2 resultado de intersectar los planos proyectantes (α_{s1}) y (α_{s2}) (que determinan (d) con (s1) y (s2) respectivamente) con el plano π. Al ser (α_{s1}) y (α_{s2}) paralelos, s1 y s2 necesariamente también lo serán, ya que la intersección de dos planos paralelos con un tercero no paralelo a los anteriores se produce siempre según dos rectas paralelas entre sí.



Obsérvese que, de acuerdo con esto último, la proyección de cualquier recta (r) paralela al plano de proyección π (y por tanto contenida en un plano π), se producirá siempre según una recta r paralela a la (r), ya que éstas pueden considerarse como el resultado de intersectar el plano proyectante (β_r) de (r) con los planos π y π_r.

b) Proporcionalidad de segmentos tomados sobre una recta (Razón simple).

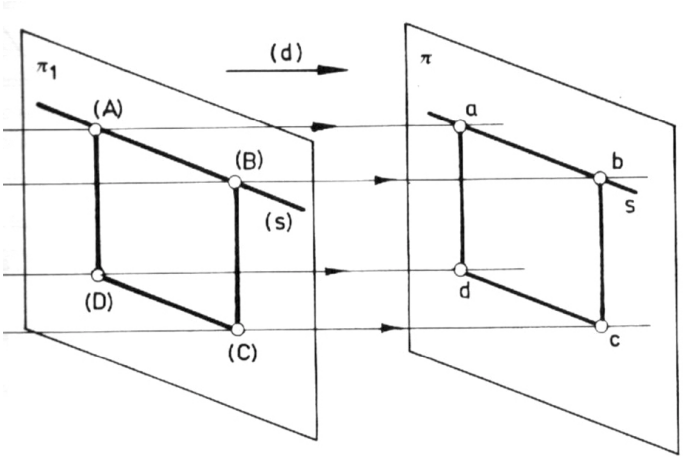
Dos segmentos tomados sobre una recta se proyectan según otros dos, de manera que la razón entre los dos primeros es igual a la razón entre los dos últimos. Dada una recta (s), que se proyecta según s en un sistema de proyección cilíndrica determinado por el plano π y la dirección (d), si tomamos los puntos (A), (B) y (C) sobre la misma, de manera que determinen una razón de proporcionalidad K = (A).(C) / (B).(C) y, a continuación, obtenemos las proyecciones a, b y c de dichos puntos; en el espacio (figura siguiente) se forman los triángulos (A)a(T), (B)b(T) y (C)c(T) semejantes entre sí que por aplicación del teorema de Thales permiten establecer que: ac/bc = (A)(C)/(B)(C) = K.



A todos estos invariantes se les añade una propiedad importante que se cumple en proyección paralela:

"La proyección cilíndrica de cualquier figura plana contenida en un plano paralelo al de proyección es otra figura idéntica a ella". Ya que las secciones producidas en cualquier superficie radiada de vértice impropio (prismas o cilindros) por dos planos paralelos son iguales.

La verificación de esta propiedad es fácilmente realizable. Consideremos una figura plana poligonal, por ejemplo un cuadrado contenido en un plano π1 paralelo al de proyección π. En la figura siguiente el lado (A)(B) del cuadrado se proyecta según un segmento ab, necesariamente paralelo a dicho lado, ya que la recta (s) que contiene a (A)(B) es paralela a π1. Por tanto (A)(B)ba determinan un paralelogramo en el que, necesariamente, (A)(B)=ba. Generalizando para los otros tres lados, se concluye que abcd es un cuadrado idéntico al (A)(B)(C)(D).



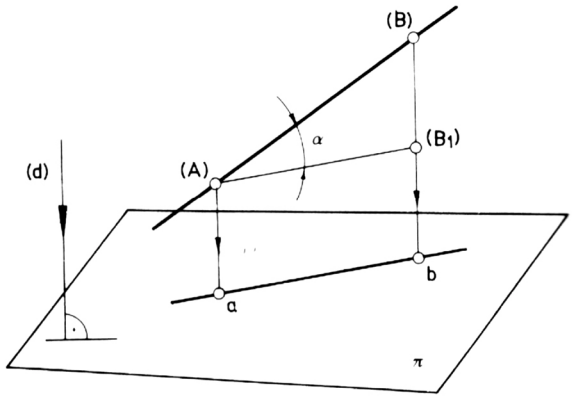
C) PROYECCIÓN CILÍNDRICA ORTOGONAL

Esta clase de proyección constituye un caso particular de la anterior y se origina cuando la dirección del punto IMPROPIO centro de proyección, es ortogonal al plano de proyección.

Consecuentemente posee todos los INVARIANTES de aquella: Pertenencia, Intersección, Tangencia, Paralelismo, Proporcionalidad de segmentos, y la propiedad de proyectar figuras contenidas en planos paralelos al de proyección según figuras idénticas a ellas.

Además en la proyección cilíndrica ortogonal se cumple:

a) La propiedad por la cual la proyección de un segmento es siempre menor o igual a dicho segmento. (Figura siguiente).



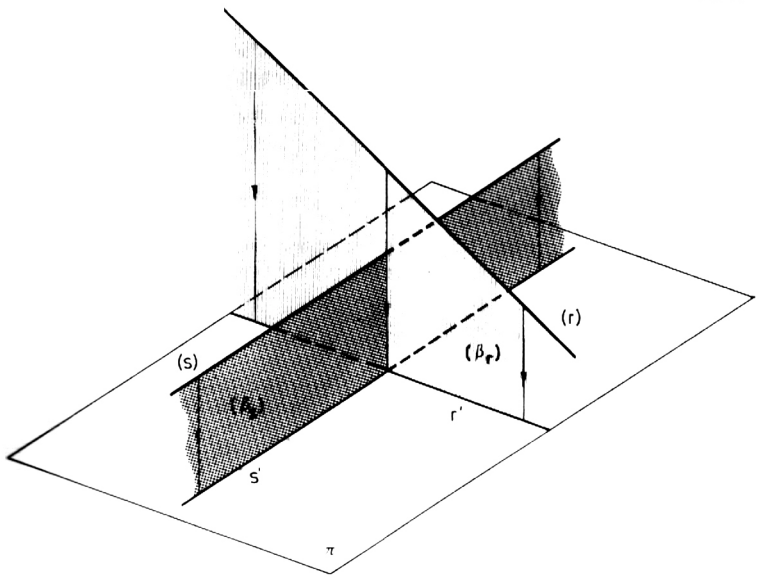
Consideremos el segmento (A)(B) tomado sobre la recta (s) y proyectémoslo ortogonalmente sobre un plano π, obteniendo otro segmento ab. Si por (A) hacemos pasar un segmento (A)(B1) igual al ab y paralelo a este último, podemos establecer que ab = (A)(B1) = (A)(B)cos(α), siendo α el ángulo que forman los segmentos (A)(B) y (A)(B1).

Como el coseno de un ángulo puede, en valor absoluto, adoptar valores entre 0 y 1, ab será siempre menor, o a lo sumo igual, que (A)(B).

b) El Teorema de las Tres Perpendiculares.

"Si dos rectas son perpendiculares en el espacio (cruzándose o cortándose), y una de ellas es paralela a un plano de proyección, las proyecciones ortogonales de ambas rectas sobre dicho plano, son también perpendiculares".

Consideremos (figura siguiente) dos rectas (s) y (r) en el espacio, de manera que (s) sea paralela al plano de proyección π y (r) sea perpendicular (s) (cruzándose con esta última).



Una recta perpendicular a otra se encuentra siempre contenida en un plano perpendicular a la segunda. Llamemos (β_r) al plano que, conteniendo a (r), sea perpendicular (s).

La proyección ortogonal de (s) sobre π será la recta s' resultado de intersectar el plano proyectante (β_s) de dicha recta con π. Como ya sabemos, s' será paralela (s) y en consecuencia perpendicular (β_r). Y, si tenemos en cuenta que todo plano perpendicular a una recta lo es también a todos los planos que contienen a dicha recta, (β_r) será perpendicular también π. Lo que nos indica que (β_r) es el plano proyectante de (r).

En definitiva los planos (β_s), (β_r) y π son perpendiculares entre sí definiendo un triedro trirrectángulo y las intersecciones r' y s' de los dos primeros con el tercero son dos rectas perpendiculares.

1.1.5. SISTEMAS DE REPRESENTACIÓN. CONDICIÓN NECESARIA PARA CUALQUIER SISTEMA DE REPRESENTACIÓN GRÁFICA.

La misión fundamental de los sistemas de representación gráfica consiste en permitir representar sobre una superficie plana, formas tridimensionales previamente creadas por el técnico, y viceversa, permitir la construcción tridimensional (restitución) de un objeto a partir de su representación gráfica (en un plano).

Como consecuencia de dicha misión, para que un sistema de representación sea válido será condición esencial la BIUNIVOCIDAD de la relación ESPACIO-PLANO, que en él se establezca. Es decir, que a cada punto del espacio le corresponda una única representación en el plano (del dibujo) y que a cada representación de un punto en el plano (dibujo) le corresponda un único punto del espacio.

En definitiva, la validez de un sistema de representación, conllevará la existencia de una relación BIUNIVOCA entre el espacio a representar (3D) y el plano de representación (2D).

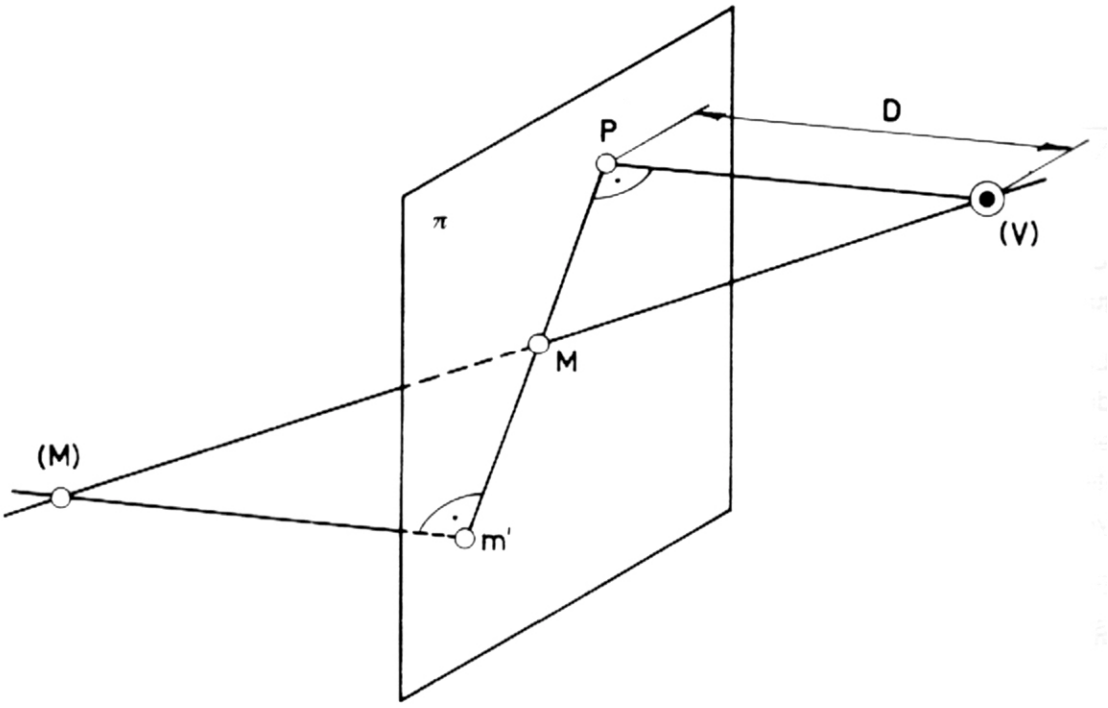
1.1.6. SISTEMAS DE REPRESENTACIÓN MÁS IMPORTANTES.

A) SISTEMA CÓNICO

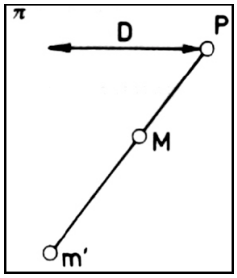
El sistema cónico está compuesto por el plano de proyección π y un centro de proyección (V) propio (siguiente figura).

El plano π se denomina plano del cuadro. Y, en este sistema un punto (M) queda determinado conociendo su proyección central M, intersección de su rayo proyectante (M)(V) con el plano π; y, a fin de garantizar su restitución al espacio, conociendo además su proyección ortogonal m' sobre π. En el sistema cónico existe

un punto notable: la proyección ortogonal del centro de proyección (V) sobre el plano del cuadro π. Este punto recibe el nombre de punto principal P, y la condición que han de cumplir los puntos M (proyección central) y m' (proyección ortogonal) es que la recta m'M pase por el punto P, lo cual es evidente, puesto que la recta m'MP es la traza del plano π con el plano formado por los tres puntos (V)-(M)-P, por ser paralelas las rectas (M)m' y (V)P.



Si hacemos coincidir el plano de proyección π, en el que tenemos representado el punto principal P y la distancia D del centro de proyección a dicho plano π, con el del dibujo (figura siguiente), en él tendremos los puntos M y m', proyección central y ortogonal, respectivamente, del punto (M) del espacio, quedando la biunivocidad del sistema garantizada.



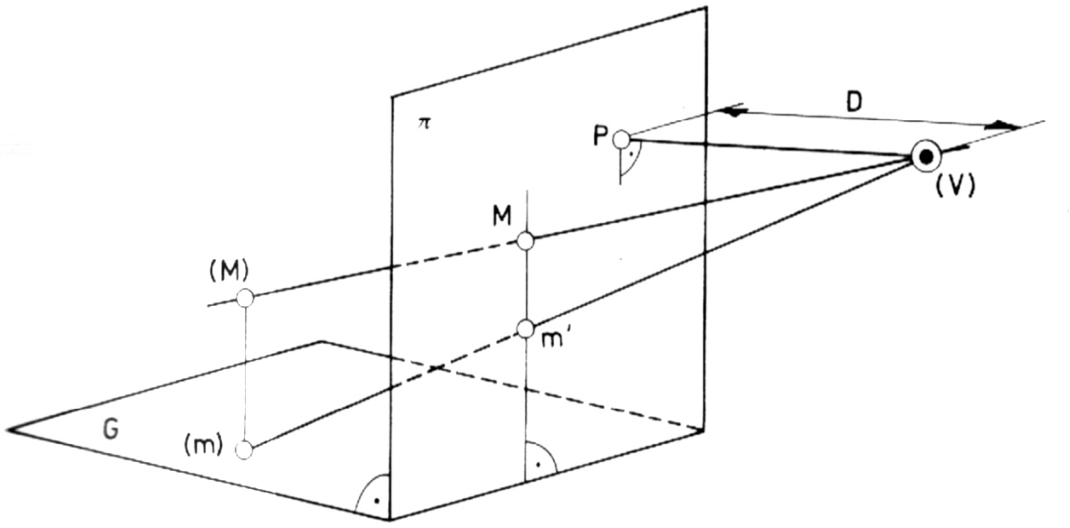
Ya que para restituir el punto (M) a su posición en el espacio bastará:

1º. Levantar a partir de P la perpendicular al plano del dibujo y tomar sobre ella la distancia D, obteniendo el punto (V).
2º. Unir el punto (V) así determinado con M.
3º. Trazar desde m' la perpendicular al plano del dibujo.

La intersección de esta última perpendicular con el rayo (V)M nos individualizará la posición del punto del espacio (M).

a) Sistema cónico con plano geometral.

Existe otra modalidad de representación en el sistema cónico, que aporta grandes ventajas como sistema general de representación central, y que, además facilita el cambio de sistema de representación (Diédrico/Cónico).



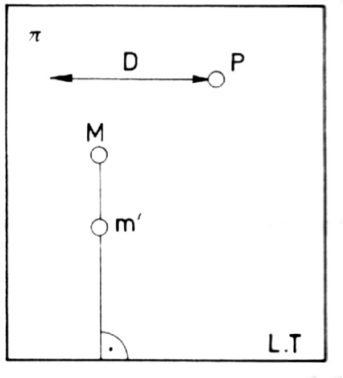
Dicha modalidad consiste en que, además del plano π de proyección y del centro (V) que definen este sistema, se utiliza un plano G perpendicular a π, que generalmente es horizontal, y que se denomina plano geometral. La recta LT, intersección de G con π, se llama también línea de tierra. En este sistema la representación de un punto (M) se consigue de la siguiente forma:

Se obtiene primeramente el punto M, proyección directa central, desde (V) sobre π del punto del espacio (M). Seguidamente se proyecta ortogonalmente el punto (M) en (m) sobre G, y nuevamente se efectúa la proyección central de (m) sobre el plano de proyección π, dando lugar a m'.

Siendo la recta (M)(m) del espacio perpendicular al plano geometral G, lo será también el plano (α) que determinan las rectas (M)(V), (m)(V) y (M)(m), por contener, precisamente, a esta última. Como π es también perpendicular a G, la intersección de (α) y π, es decir la recta Mm', también lo será. Y por tanto, Mm' será siempre perpendicular a la línea de tierra (LT).

Obsérvese que, como consecuencia de esto último, la totalidad de las rectas o aristas de un objeto perpendiculares al plano geometral G, aparecerán

representadas en este sistema mediante rectas ortogonales a la línea de tierra (figura siguiente).



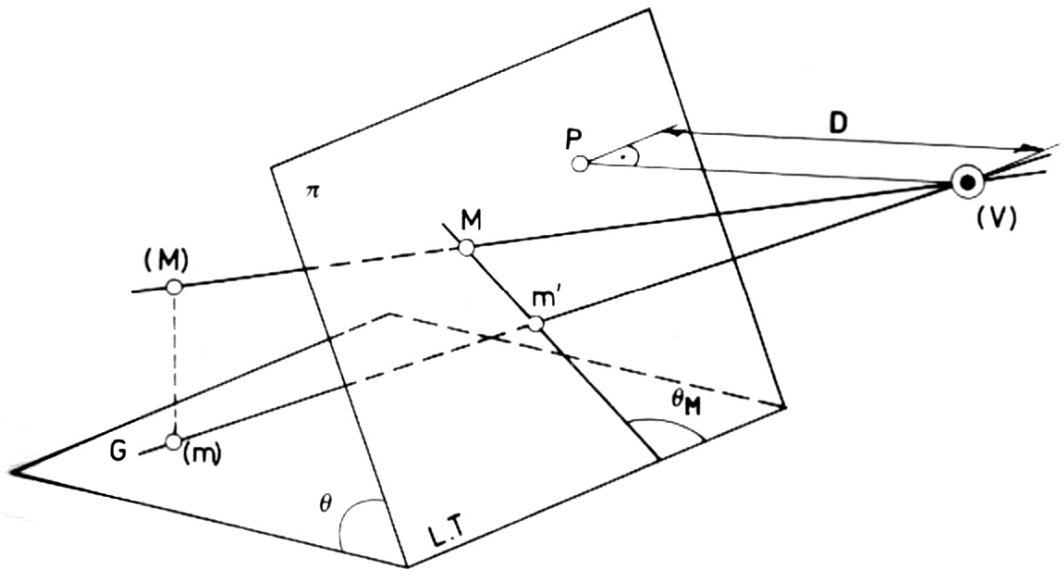
Al igual que en el caso anterior, la reversibilidad del sistema queda totalmente garantizada. Para su comprobación, dada la representación de un punto por sus proyecciones M y m', y conocidas P y D, bastaría efectuar las siguientes operaciones:

1º. Levantar, a partir de P, la perpendicular al plano del dibujo y tomar sobre ella la distancia D, obteniendo el punto (V).
2º. Unir el punto (V) con M.
3º. Unir el punto (V) con m' y obtener la intersección de dicha recta con G: punto (m).
4º. Levantar, a partir de (m), la perpendicular a G.

La intersección de esta última con el rayo proyectante (V)M nos determinará la posición en el espacio del punto (M).

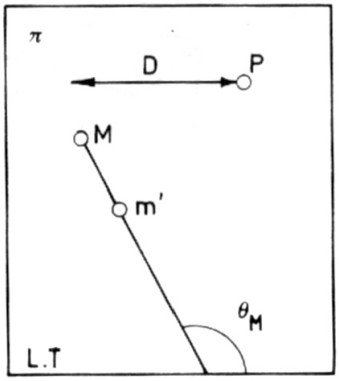
b) Sistema cónico con plano geometral y plano del cuadro inclinado.

Existe, por último, otra variante del sistema cónico, derivada de la anterior, que se obtiene en el caso de que el plano de proyección π forme un ángulo θ, cualquiera, distinto de 90º, con el plano geometral G.

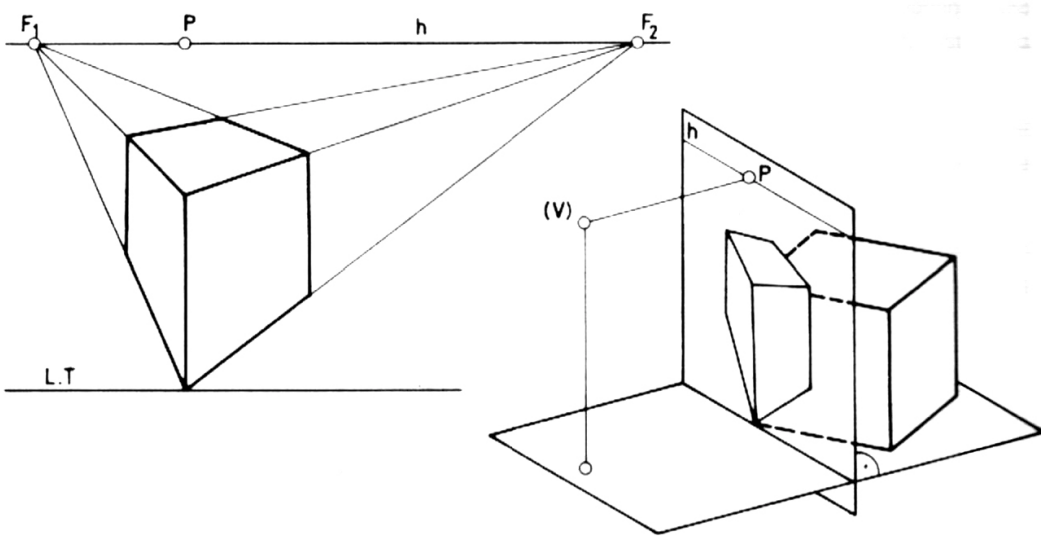


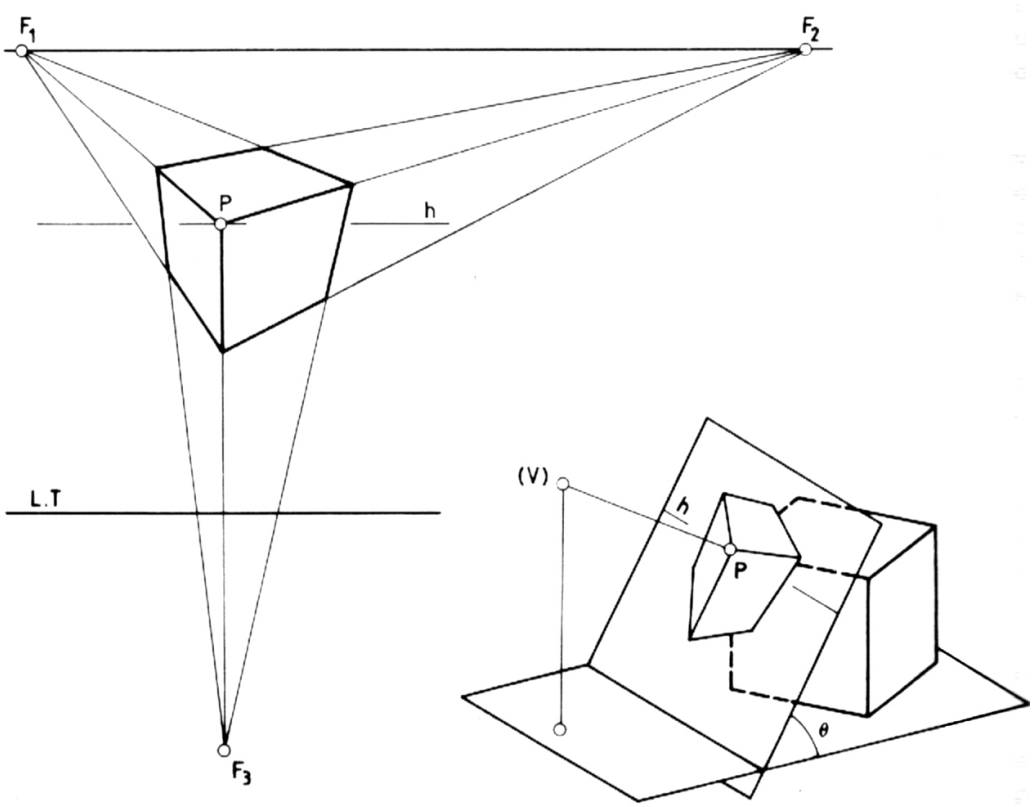
En este sistema la representación de un punto (M) se obtiene exactamente igual que en el anterior. Es decir a través de la proyección central M de dicho punto y de la proyección central m' del punto (m), proyección ortogonal, a su vez, del punto (M).

A diferencia del anterior sistema, aquí las proyecciones M y m' no determinarán, en general, alineaciones ortogonales con la L.T. formando éstas un ángulo θ_M, distinto de 90º, con la L.T. Ello es debido precisamente a que, contrariamente a lo que ocurría en el apartado anterior, los planos π y G no son perpendiculares.



Como consecuencia, en este sistema las alturas verticales ya no se proyectan paralelas y ortogonales a la L.T., sino que poseen un punto de fuga a una distancia finita (figura anterior). La contrastación de dicha diferencia queda reflejada en los dos ejemplos siguientes en los que aparece representado un mismo objeto en estas dos modalidades de sistemas cónicos.





La reversibilidad del sistema se puede comprobar realizando las mismas operaciones utilizadas en el caso en el que π es perpendicular a G.

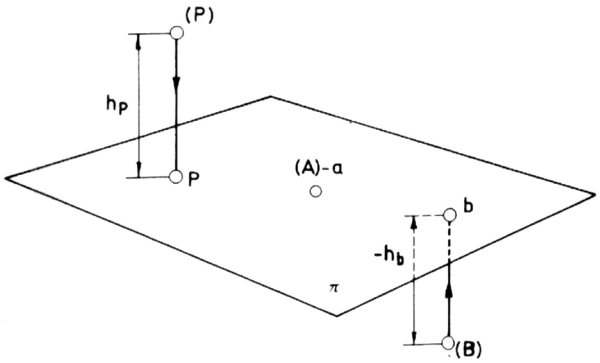
La perspectiva cónica, en cualquiera de sus modalidades, permite obtener imágenes semejantes a la visión humana. Por ello constituye un medio de comunicación adecuado para la comunicación entre expertos y profanos.

Sin embargo, presenta el inconveniente de precisar cierta laboriosidad y dificultad para el manejo de cuerpos y para la obtención de medidas.

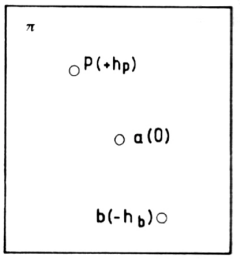
En cualquier caso, así como la fotografía permite obtener una imagen intuitiva de un objeto existente, en ocasiones, es de mucho interés presentar dibujos intuitivos y expresivos de edificios u objetos que se quieren realizar, promocionar o incluso, vender. Este es el objeto de la perspectiva cónica, utilizada por ingenieros y arquitectos para la representación de edificios, puentes, obras hidráulicas, naves industriales y, en general, objetos de tamaño considerable observados desde distancias relativamente próximas a los mismos.

B) SISTEMA DE PLANOS ACOTADOS

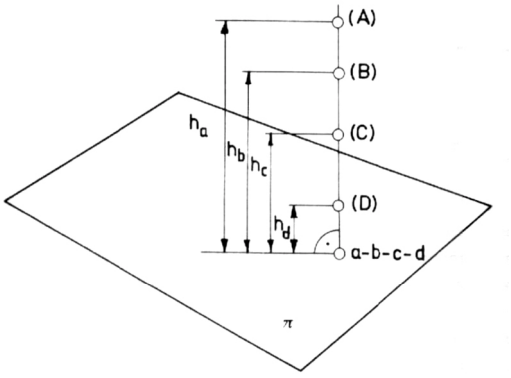
En el sistema de planos acotados la forma de conseguir la representación de un punto (P) del espacio consiste en obtener de él una proyección cilíndrica ortogonal sobre un plano de proyección π: punto P pie de la perpendicular trazada desde (P) al plano. Además, para conseguir la condición de reversibilidad se anota entre paréntesis al lado de P un número que nos indique la distancia del punto (P) al plano de proyección, o sea su cota h_P, la cual, como se aprecia, divide al espacio en dos partes, de las que una de ellas se afecta de cotas positivas y la otra de cotas negativas (figura siguiente).

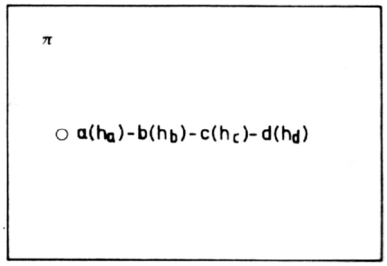


La comprobación del requisito de biunivocidad de este sistema es inmediata, puesto que, dada la representación de un punto (P) por su proyección y por su cota (siguiente figura), bastará levantar por P la perpendicular a π y llevar sobre ella, a partir de P, el valor de la cota, para restituir el punto (P) en el espacio.

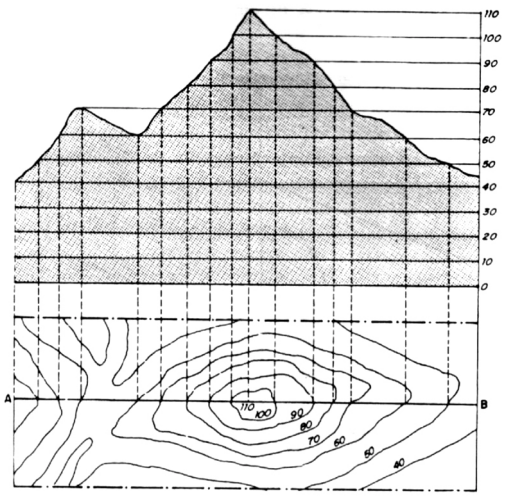


La operatoria en el sistema de planos acotados es probablemente la más sencilla, sin embargo, en este sistema de representación existe la posibilidad de confusión cuando se trate de relacionar en el espacio puntos que se hallen sobre proyectantes comunes (véase las dos figuras siguientes).





Por ello este sistema se utiliza cuando la forma de representar, tiene un solo punto por proyectante, lo cual sucede en las superficies topográficas (siguiente figura). Además de las superficies topográficas, este sistema tiene su campo de aplicación en la representación y el manejo constructivo de curvas y superficies empíricas, en la representación de algunos elementos de máquinas (álabes), en la arquitectura naval, en la representación de cubiertas, la geografía, la geología y la meteorología.



C) SISTEMA DIÉDRICO O DE MONGE

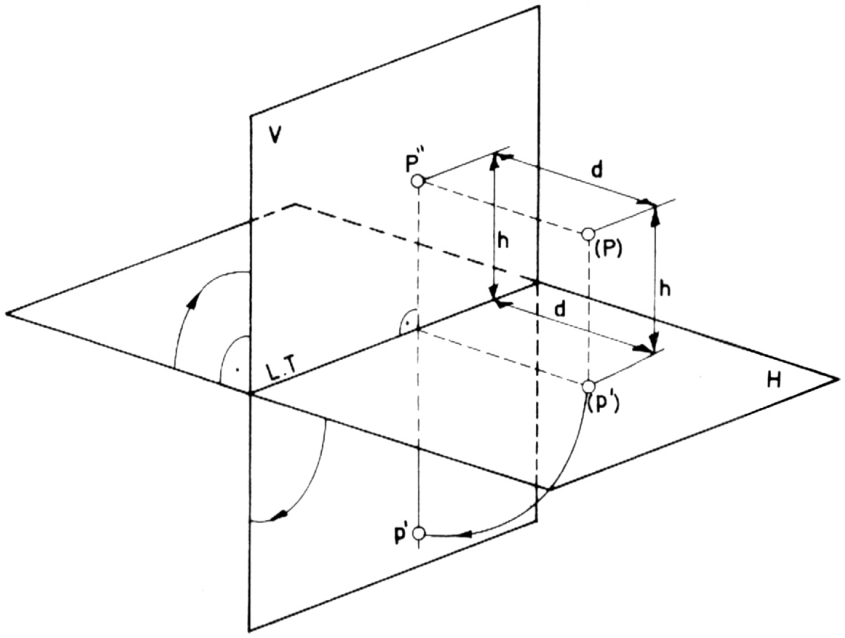
Para evitar el inconveniente que supone la relación en el espacio de puntos situados sobre la misma proyectante, cuando están representados en el sistema de planos acotados, se puede recurrir a una segunda proyección que nos evite afectar al mismo punto proyección de varias cotas.

El sistema Diédrico dispone de un conjunto formado por dos planos de proyección ortogonales entre sí, que se colocan, en general: uno de ellos, horizontal (Plano horizontal de proyección H), y el otro, por tanto, vertical (Plano vertical de proyección V).

Tal sistema, fue ideado para su aplicación a la construcción, en cuya aplicación los objetos que se han de representar presentan sus líneas principalmente orientadas según planos horizontales y verticales. Esta es la razón natural de que, de los dos planos de proyección uno se elija horizontal y otro vertical.

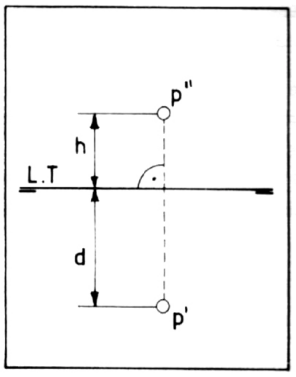
Debe advertirse, no obstante, que no es de ninguna manera esencial al sistema el que estos dos planos sean realmente uno horizontal y otro vertical. La razón originaria expuesta justifica la adopción de esos nombres convencionales, que se suelen aplicar aún cuando no correspondan en algunos casos a una realidad física. Incluso, es muy frecuente en la práctica cambiar de planos de proyección tomando nuevos planos horizontales o verticales a conveniencia.

En este sistema, la representación de un punto (P) se consigue proyectando ortogonalmente el punto (P) sobre el plano horizontal H, dando lugar a su proyección horizontal p', y proyectando el punto (P) también ortogonalmente sobre el plano vertical V, obteniendo así la proyección vertical p'' (ver figura siguiente).



De la exposición hecha, puede parecer deducirse la dificultad para la realización práctica de este sistema de representación, al haberse de efectuar los dibujos sobre dos planos distintos. Esta dificultad puede obviarse haciendo que esos dos planos sean realmente uno solo, es decir, que en vez de obtener un dibujo plano para cada una de las proyecciones, se obtengan los dos dibujos en un solo plano superposición de los dos.

Ello se consigue haciendo coincidir al plano H con el V (plano del dibujo) haciéndolo girar alrededor de su recta de intersección, que llamaremos en lo sucesivo línea de tierra L.T. De esta forma, p' viene a ocupar una posición tal que se encuentren p'' y p' sobre la misma perpendicular a la línea de tierra (figura siguiente), lo cual es evidente al observar en la figura anterior que el plano determinado por (P), p' y p'', que contiene a la trayectoria de p' ocasionada por el abatimiento de H sobre V, es perpendicular a V y H y por lo tanto a la L.T.



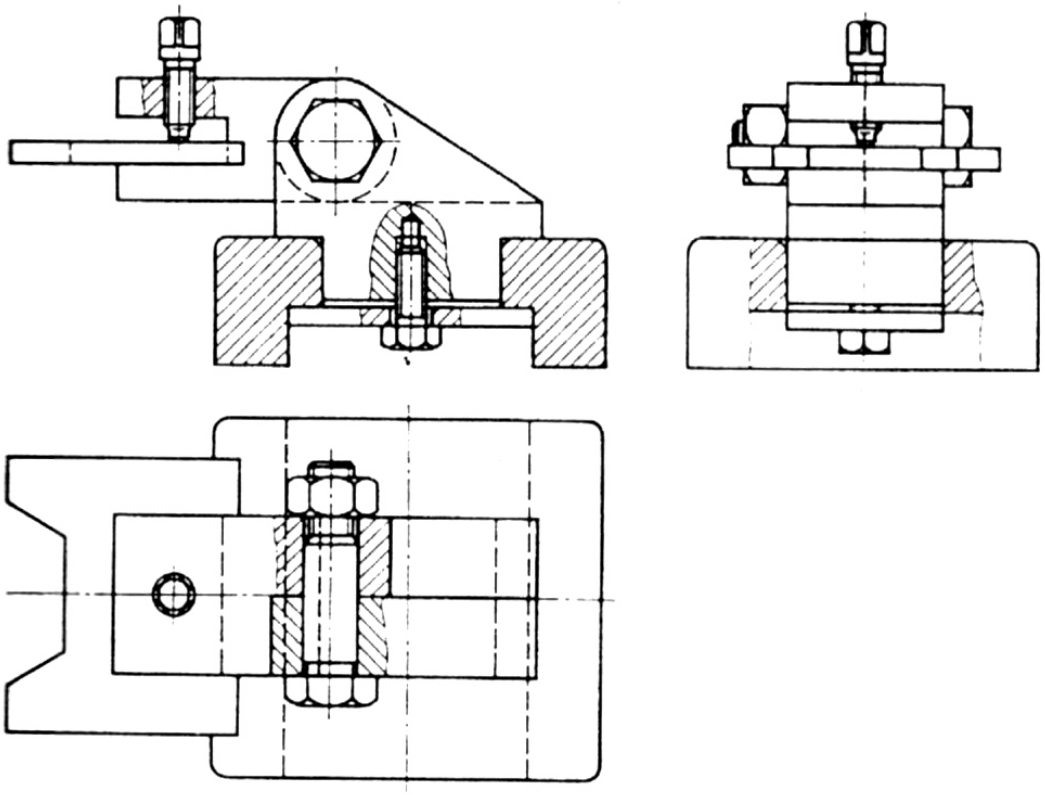
La distancia h que separa a la proyección vertical p'' de la línea de tierra, será igual a la altura del punto (P) sobre el plano horizontal de proyección H, es decir, la magnitud del segmento (P)-p'; de la misma forma, la distancia d que separa la proyección horizontal p' de la línea de tierra, representa un segmento igual a la magnitud (P)-p'', es decir, la distancia existente entre el punto del espacio (P) y el plano vertical V de proyección.

Dada la representación de un punto por sus proyecciones p' y p'', la reversibilidad del sistema es fácilmente demostrable efectuando las siguientes operaciones:

1º) A partir de la L.T., se coloca perpendicularmente al plano del dibujo, es decir, a V, el plano H.

2º) Desde p' levantamos la perpendicular a H, y desde p'' la perpendicular a V, estas dos perpendiculares se cortarán en un único punto del espacio, que será el punto (P), el cual dio origen a las dos proyecciones p' y p''.

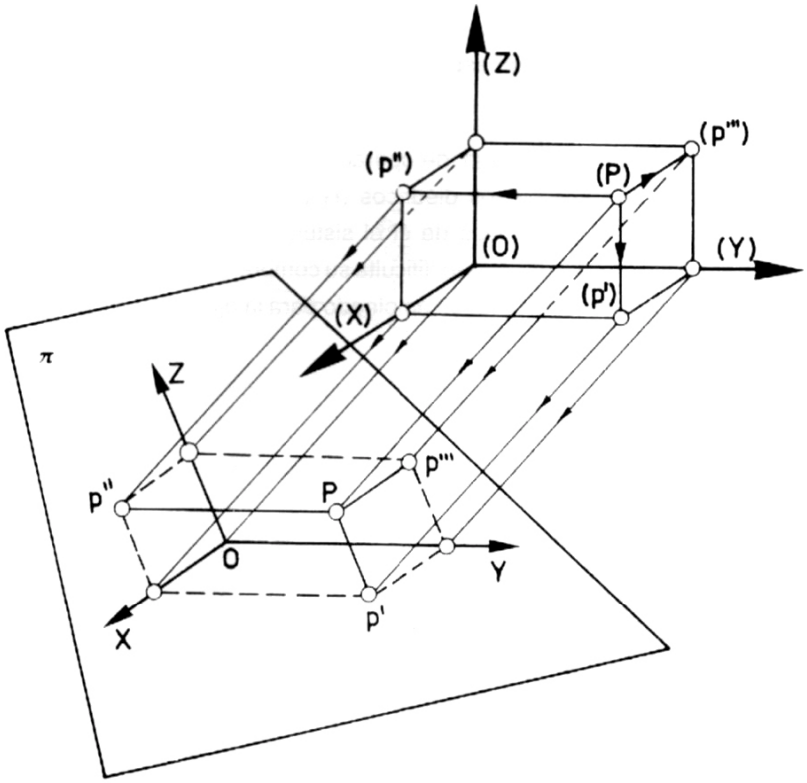
El sistema Diédrico precisa, en su manejo, de operaciones y construcciones gráficas sencillas. Y la propiedad de proyectarse sobre sus planos diédricos (H y V) sin deformarse la información gráfica contenida en planos paralelos a estos, hace de él el sistema de medida por excelencia. Posee, no obstante, un nivel de abstracción elevado, lo que dificulta su comprensión al no habituado. A pesar de ello, constituye el sistema de representación habitual empleado para la ejecución de los planos de ingeniería (figura siguiente).



D) SISTEMA AXONOMÉTRICO

El sistema de representación axonométrico es aquel en el que las figuras a representar se proyectan en proyección paralela sobre un plano del cuadro referidas a un sistema de ejes, sobre cuyos planos coordenados tienen las mismas sus correspondientes proyecciones ortogonales. La proyección sobre el cuadro de estas proyecciones sobre los planos coordenados acaba de determinar los elementos representados.

La representación de un objeto en sistema axonométrico se consigue pues, a través del siguiente proceso:



1º) Proyectando ortogonalmente (en el espacio) el objeto a representar sobre los tres planos coordenados de un triedro de referencia trirrectangular, obteniendo unas proyecciones que en adelante denominaremos PREVIAS.

2º) Proyectando, a continuación, el objeto, los ejes del triedro de referencia y las proyecciones previas, OBLICUA u ORTOGONALMENTE, sobre el plano del cuadro.

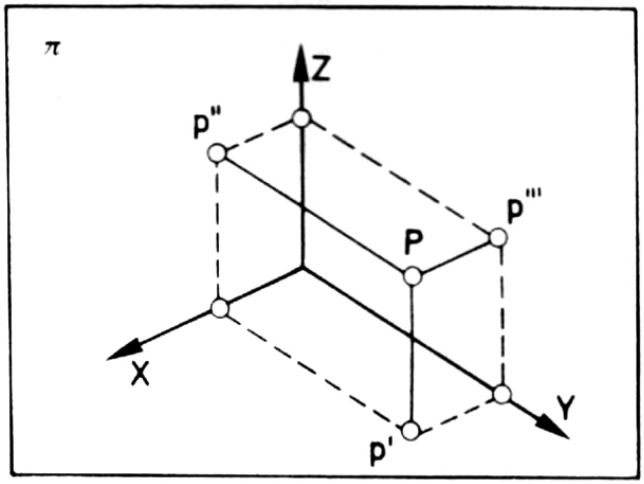
Como resultado de esta segunda proyección sobre el plano del cuadro (o del dibujo) el objeto lo tendremos representado por su proyección DIRECTA sobre π y por las tres proyecciones sobre π de las proyecciones previas, proyecciones que reciben el nombre de LATERALES.

En función de que la proyección cilíndrica empleada sea **oblicua** u **ortogonal**, dispondremos de un **sistema axonométrico oblicuo** u **ortogonal**.

En particular para representar un punto (P) del espacio (figura anterior), en primer lugar proyectaremos ortogonalmente este punto sobre las tres caras del triedro trirrectángulo, obteniendo así las proyecciones "previas" (p'), (p''), (p'''); de manera que los segmentos (P)-(p'''), (P)-(p'') y (P)-(p'), serán iguales, respectivamente a las coordenadas (x), (y), (z) del punto (P) con relación al sistema del espacio.

A continuación, proyectaremos cilíndricamente el conjunto del espacio constituido por el punto (P) y por sus respectivas proyecciones (p'), (p'') y (p''') sobre el plano del cuadro.

De esta forma obtendremos:



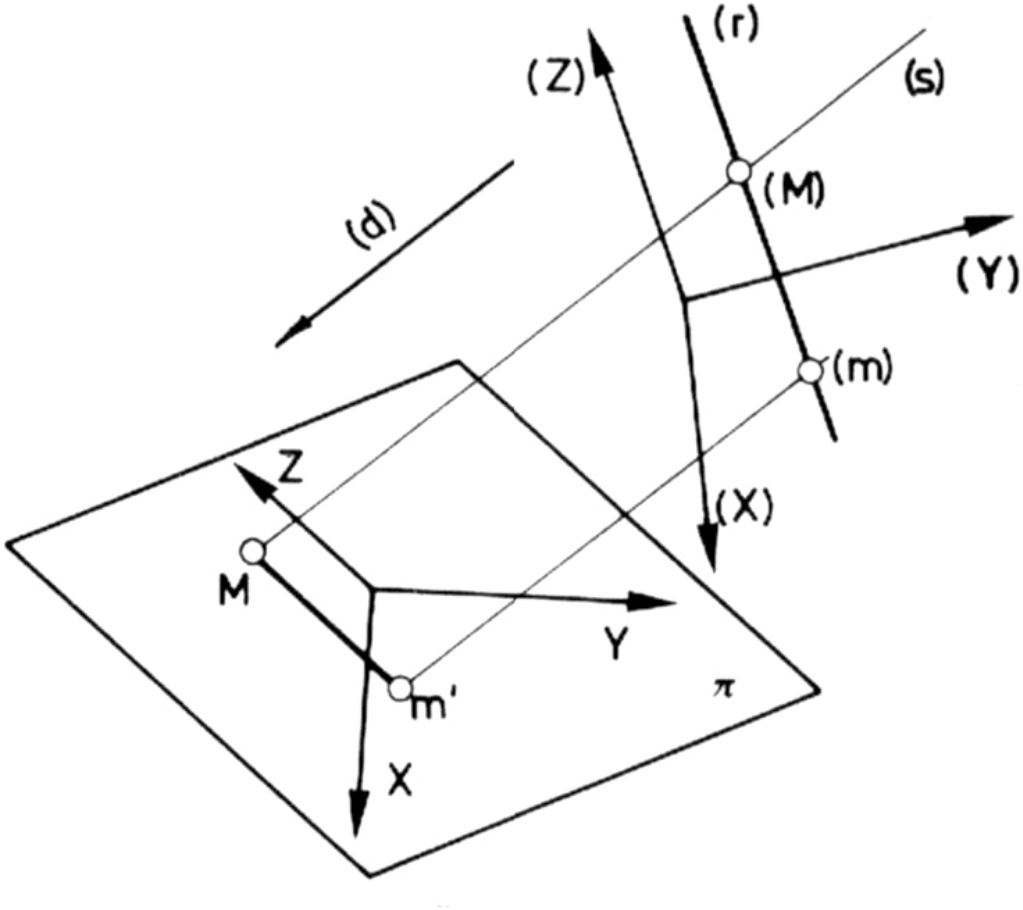
Una proyección directa (P) y tres proyecciones laterales p', p'', p''', de las previas, (p'), (p''), (p'''), situadas sobre las caras del triedro trirrectángulo. En esta nueva proyección se aprecian de una sola vez las proyecciones de las tres coordenadas del punto (P); es decir, los segmentos P-p''', P-p'', P-p'. La relación numérica entre dichas coordenadas y sus proyecciones se expresan en función de unos coeficientes e_i que, en virtud del carácter invariante de la proporcionalidad de segmentos en proyección cilíndrica, son de aplicación general para cualquier segmento (coordenada) tomado paralelamente a un eje.

Los segmentos P-p''', P-p'' y P-p' y las coordenadas (x), (y) y (z) de cualquier punto del espacio (P) guardarán, por tanto, una relación de proporcionalidad que, de ahora en adelante, expresaremos mediante los coeficientes e_x, e_y y e_z.

El valor de los coeficientes e_x, e_y, e_z, el de los ángulos zôx, zôy, xôy, que forman las proyecciones de los ejes entre sí, así como la existencia o no, de relación entre los coeficientes y dichos ángulos, depende de: el ángulo que forma la dirección de proyección con el plano del cuadro y de la orientación del triedro de referencia respecto a este último.

Para que un punto esté representado en este sistema bastará conocer dos de sus cuatro proyecciones; bien, la directa P y una de las laterales, por ejemplo p'; o bien, dos laterales cualesquiera, por ejemplo p' y p'', pues con ellas se obtienen las otras dos trazando paralelas a las proyecciones de los ejes hasta completar la proyección del paralelepípedo de la figura anterior.

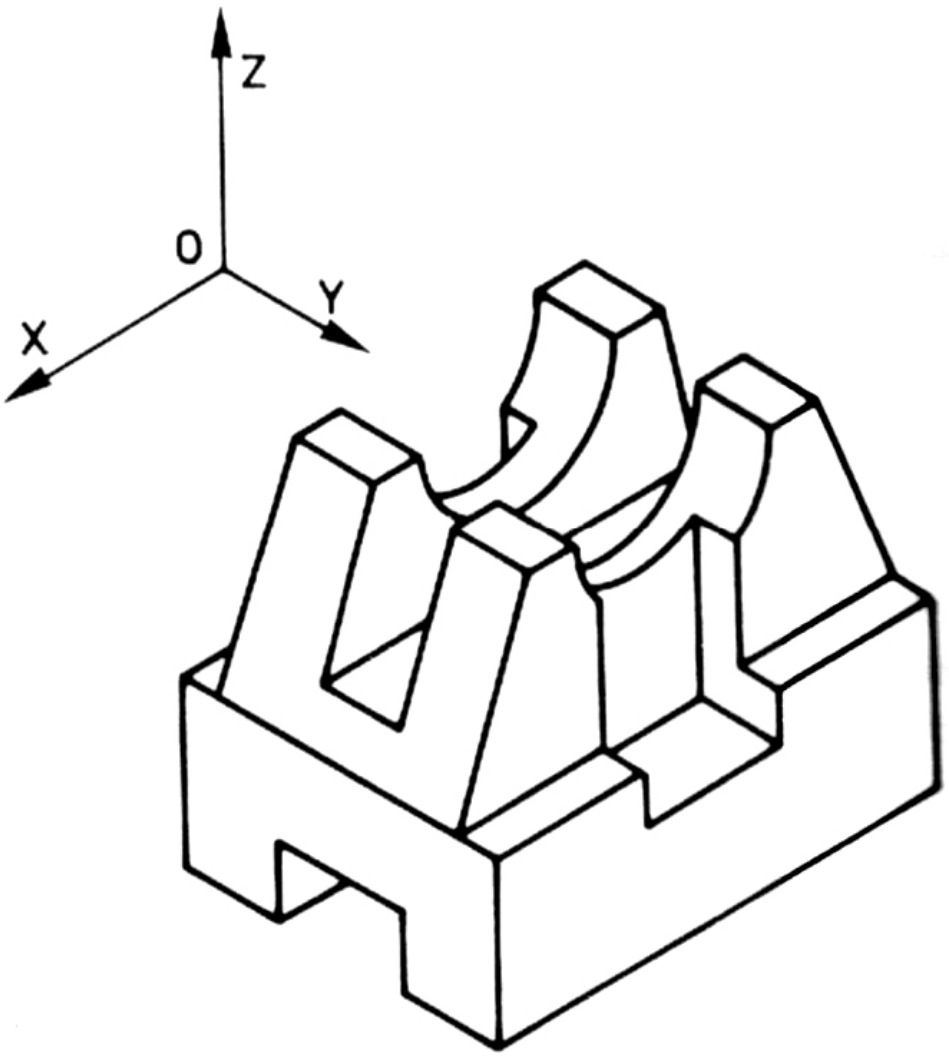
Queda por último, comprobar la reversibilidad del sistema. Para ello supongamos un punto (M) representado por sus proyecciones directa M y lateral m', en un sistema axonométrico dado por el plano de proyección π y una dirección de proyección (d) (figura siguiente).



Es evidente que el punto (M) en el espacio estará en la recta (s) paralela a (d) que pasa por (M). Además, la proyección previa (m') será el punto intersección de la recta paralela a (d) que pasa por m' con el plano coordenado xoy del sistema de referencia. Finalmente, el punto (M), único en el espacio, será el resultado de intersectar la recta (s) con la perpendicular (r) al plano xoy, que pasa por (m'). Obsérvese que ambas rectas se han de cortar forzosamente al estar los segmentos (M)(m') y Mm' contenidos en mismo plano proyectante.

Tanto el sistema axonométrico como el cónico, permiten visualizar en una sola proyección (la directa) las tres dimensiones principales a la vez, si bien ninguna de ellas resulta igual a su proyección sobre los planos de referencia.

Además el sistema axonométrico tiene la ventaja sobre el cónico de que tratándose de proyecciones paralelas, la razón entre un segmento y su proyección es constante para toda recta paralela a una dirección dada, por lo que resulta muy fácil construir las longitudes paralelas a una cierta dirección y en especial las paralelas a los ejes, lo que es de gran utilidad en la representación de cuerpos que posean tres direcciones principales según las cuales estén orientadas sus aristas o líneas más importantes (figura siguiente).



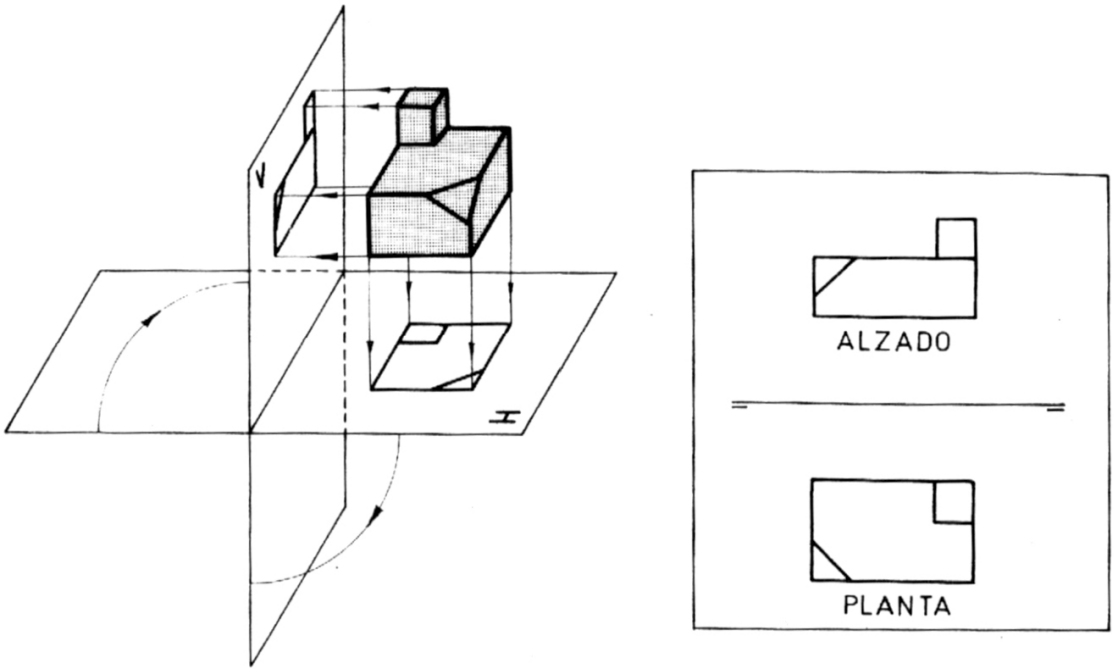
Por otro lado, se ha de tener presente que la construcción de una imagen en axonometría supone en general más trabajo de dibujo que si se efectúa en el sistema diédrico. Sin embargo, este mayor trabajo está totalmente justificado cuando el ingeniero desea ser comprendido rápidamente, bien sea por profanos en cuestiones técnicas, por profesionales, por operarios o por fabricantes.

1.1.7. INTRODUCCIÓN AL SISTEMA DIÉDRICO.

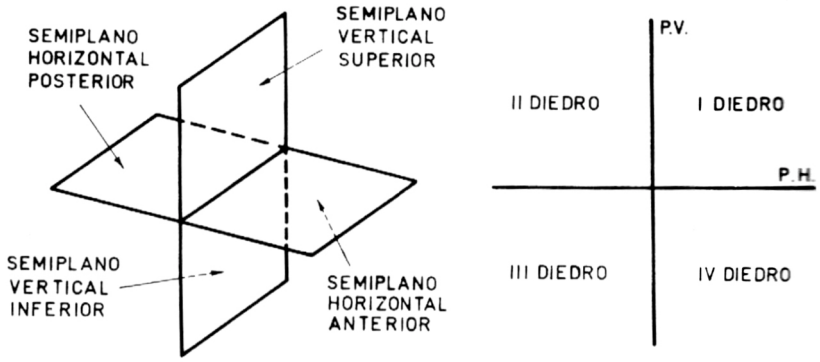
Se ha definido antes el sistema de representación de Monge como un sistema de proyección paralela ortogonal sobre dos planos perpendiculares entre sí, llamados uno horizontal y otro vertical.

En él, una figura cualquiera del espacio se representa mediante sus dos proyecciones ortogonales sobre los dos planos de proyección. La proyección ortogonal sobre el plano horizontal se denomina PROYECCIÓN HORIZONTAL o PLANTA, y la proyección sobre el vertical, PROYECCIÓN VERTICAL o ALZADO.

Para reunir la planta y el alzado de dicha figura en un mismo plano se hace girar el plano horizontal alrededor de la recta intersección de ambos planos hasta hacerlo coincidir con el vertical que, a su vez se hace coincidir con el del dibujo. Dicha recta intersección recibe el nombre de Línea de Tierra L.T. Veamos esto en las siguientes figuras.

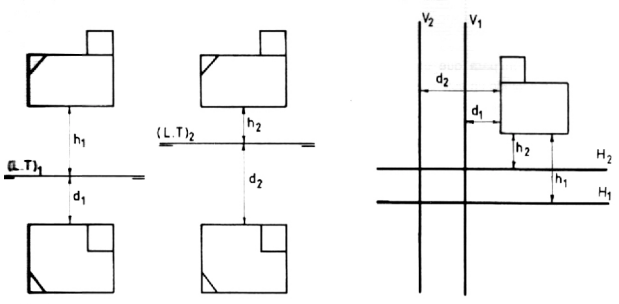


Los planos horizontal P.H. y vertical P.V. de proyección dividen al espacio en cuatro regiones. Para denominar los dos semiplanos de cada plano de proyección y las cuatro regiones diédricas, tomando a los planos efectivamente uno como horizontal y otro como vertical, se supone colocado al espectador en uno de los diedros. El semiplano horizontal sobre el que se apoya se llama horizontal anterior y el otro horizontal posterior, haciendo referencia estos dos calificativos de anterior y posterior a su situación con respecto al otro plano de proyección. El semiplano vertical que está por encima del horizontal se llama vertical superior y el otro vertical inferior. Se llama PRIMER DIEDRO al limitado por el horizontal anterior y el vertical superior en el que se supone situado el observador; SEGUNDO DIEDRO al limitado por el vertical superior y el horizontal posterior; TERCER DIEDRO al limitado por el horizontal posterior y el vertical inferior, y CUARTO DIEDRO al limitado por el vertical inferior y el horizontal anterior. Ver figura siguiente.



En el plano del dibujo la línea de tierra, que divide en dos zonas a éste, se subraya en sus extremos mediante dos trazos, de manera que la zona superior corresponde al semiplano vertical superior y al semiplano horizontal posterior, y la zona inferior corresponde al semiplano horizontal anterior y al semiplano vertical inferior.

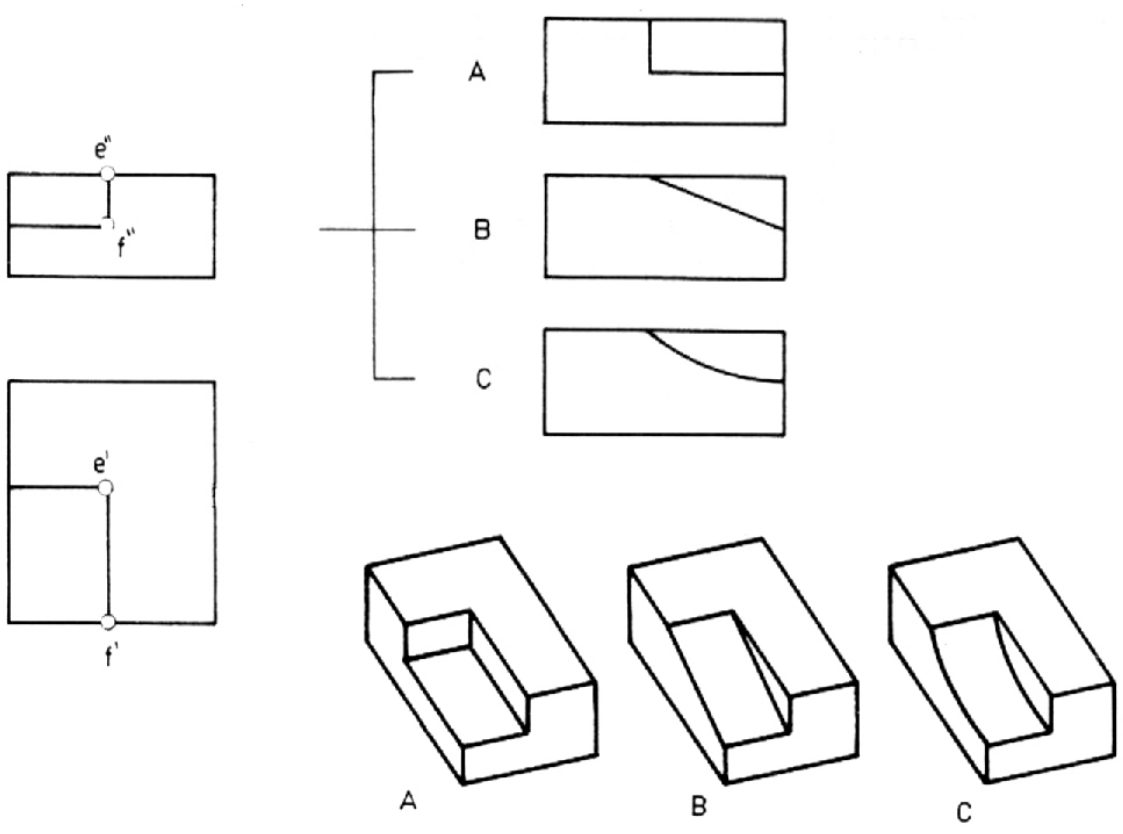
Debe observarse en la figura siguiente que si las dos proyecciones de un objeto, alzado y planta, permanecen fijas en el plano del dibujo y se traslada paralelamente a si misma a la línea de tierra, ello indica, únicamente que ha tenido lugar una traslación conjunta de los dos planos de proyección. Para estudiar las propiedades de un objeto por medio de sus proyecciones, una traslación de esta clase carece de importancia. Por este motivo, muy frecuentemente, la línea de tierra no se dibuja, ya que cada objeto queda determinado por su planta y alzado, sin necesidad de la línea de tierra.



1.1.8. TERCERA PROYECCIÓN. PLANO DE PERFIL.

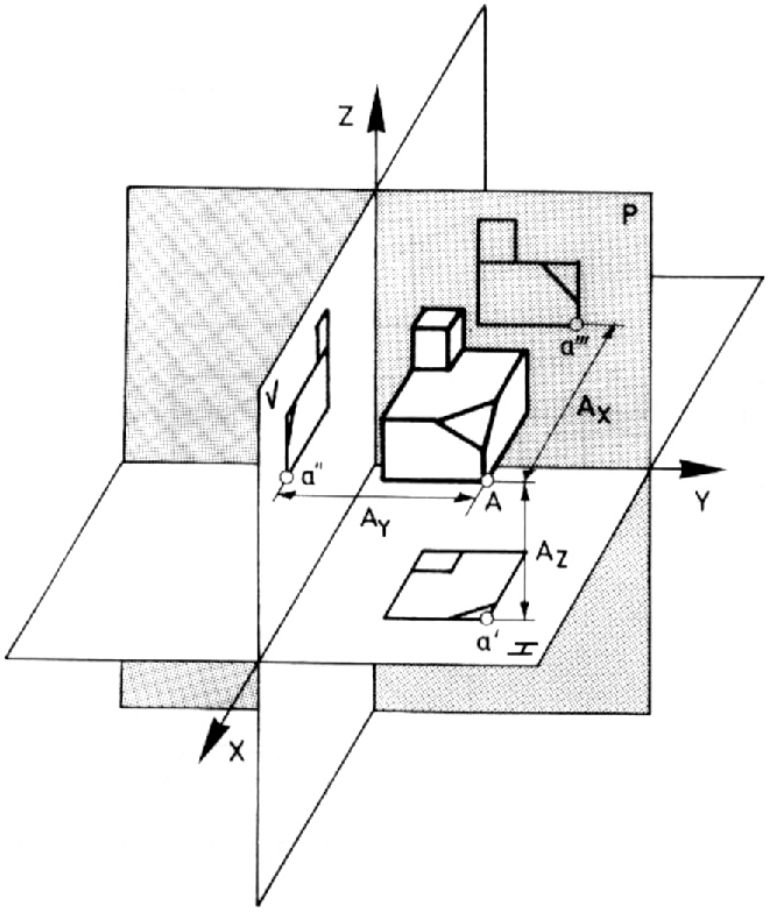
En la práctica, a pesar de la suficiencia (reversibilidad) del sistema diédrico, cuando las proyecciones horizontal y vertical de un objeto no resultan suficientemente exhaustivas para su correcta interpretación, se introduce un tercer plano de proyección.

A modo de ejemplo, obsérvese que el alzado y planta de la figura siguiente pueden corresponder a cualquiera de las tres piezas representadas en perspectiva axonométrica. Naturalmente si facilitáramos en dichas vistas la proyección de puntos suficientes contenidos en la línea EF, evitaríamos dicha indeterminación. Sin embargo, resulta mucho más claro y sencillo romper la indeterminación, adjuntando a la información que proporcionan el alzado y la planta, la obtenida proyectando la pieza sobre un tercer plano en el que se visualiza sin ambigüedad la línea EF.



Por este motivo y también para completar el sistema referencial y facilitar la situación de puntos por los valores numéricos de sus coordenadas, se fija un tercer plano perpendicular a la línea de tierra y, por lo tanto, al horizontal y al vertical, que se llama PLANO DE PERFIL.

El plano de perfil forma, con los dos planos de proyección, un triedro trirrectángulo (figura siguiente), cuyas aristas pueden ser tomadas como ejes de un sistema de coordenadas cartesianas ortogonales, siendo el origen del mismo el punto común a los tres planos.



El conjunto de los tres planos coordenados Horizontal, Vertical y de Perfil dividen el espacio en 8 octantes. Y un punto cualquiera del espacio se referirá a dichos planos a través de sus distancias a ellos.

El triedro trirrectángulo quedará integrado por los ejes:
- ox: intersección de los planos Horizontal y Vertical.
- oy: intersección de los planos Horizontal y de Perfil.
- oz: intersección de los planos Vertical y de Perfil.

Para dar numéricamente puntos, utilizaremos, pues, este sistema de coordenadas proporcionándolos como en Geometría Analítica por la terna de números (x, y, z) correspondientes a su:

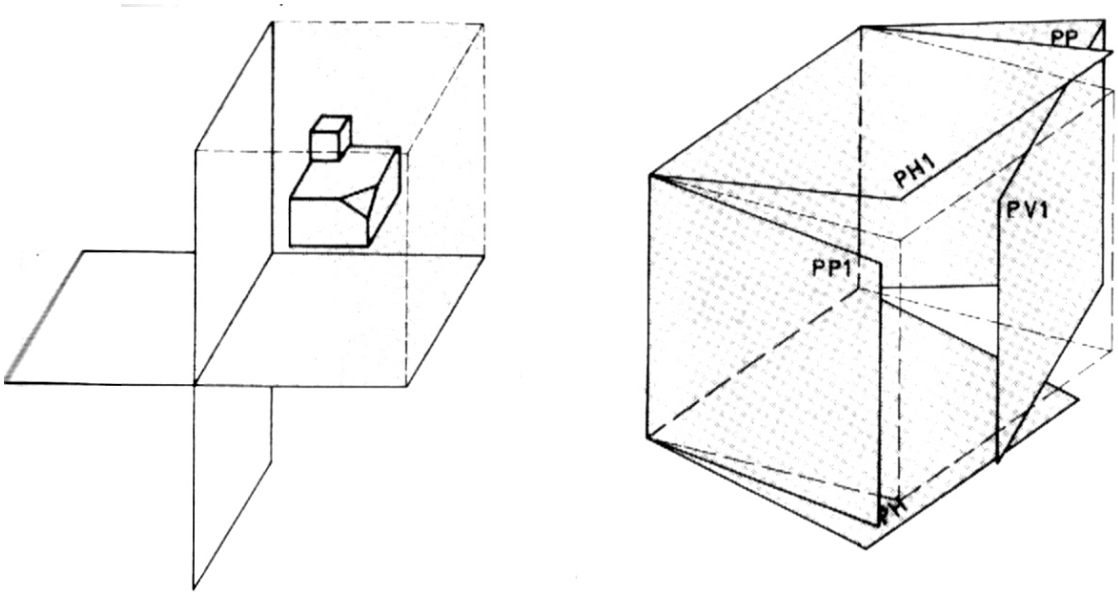
(x) Desviación o Referencia: distancia al plano de Perfil.
(y) Distancia o Alejamiento: distancia al plano Vertical.
(z) Altura o Cota: distancia al plano Horizontal.

La coordenada “x” adquirirá valores positivos cuando el punto se encuentre a la izquierda del plano de Perfil, la coordenada “y” será positiva cuando el punto se encuentre delante del plano Vertical y la coordenada “z” será positiva cuando el punto se encuentre por encima del plano Horizontal.

1.1.9. GENERALIZACIÓN DE LAS PROYECCIONES DIÉDRICAS. SISTEMA MULTIVISTA.

A la planta y alzado de la representación de un objeto se pueden añadir, además del perfil (objeto visto por la izquierda), otras proyecciones obtenidas sobre otros tres planos PH1, PV1, PP1 paralelos al Horizontal, Vertical y de Perfil, respectivamente.

La disposición de estos planos se efectúa de manera que los planos PH, PV, PP y los nuevos planos de proyección formen las caras laterales de un ortoedro en cuyo interior se encuentra el objeto. Obtenidas las proyecciones sobre las caras del ortoedro, al desarrollar éste sobre el plano del dibujo (plano Vertical) se obtienen las seis vistas que constituyen la representación multivista de dicho objeto.



Las proyecciones sobre los planos PP1, PV1 y PH1, que corresponden a puntos de vista situados a la derecha, detrás y abajo del objeto, se distinguen de la planta, alzado y perfil en que en las primeras unas partes del objeto se presentan visibles mientras que en las segundas no lo son y viceversa. Esta es precisamente la ventaja de este conjunto de “proyecciones opuestas”: las aristas que en una son visibles y en la opuesta son ocultas, se pueden suprimir de esta última y obtener, por tanto, dibujos más claros en la representación de objetos complicados (Eliminación de Líneas Ocultas).

En función del cuadrante en que se sitúe el objeto a representar y de la posición relativa entre observador - objeto - plano de proyección, se obtienen dos sistemas multivista diferentes normalizados en Europa y en América respectivamente.

1.2. LA RECONSTRUCCIÓN 3D.

En esta sección se va a realizar una descripción detallada de todos los conceptos y definiciones que intervienen en el proceso de reconstrucción 3D a partir de una representación 2D del objeto.

1.2.1. INTRODUCCIÓN.

La descripción de objetos tridimensionales en un plano, utilizando proyecciones bidimensionales, se remonta a más de dos mil años. Fue Monge el primero que sistematizó y simplificó los métodos existentes, dando lugar al nacimiento de la geometría descriptiva. El problema contrario de cómo reconstruir automáticamente la estructura de un objeto tridimensional (estructura geométrica y topológica) a partir de su proyección, empezó a atraer la atención sólo a finales de los 60, motivado por el desarrollo de los ordenadores digitales. Varios autores incluyen buenas referencias a los primeros estudios sobre este tema. Este problema, que se denomina "**reconstrucción**", implica determinar la relación geométrica y topológica de las partes atómicas de un objeto. No debe confundirse con el **reconocimiento** o **restitución**, que implica la identificación de un objeto mediante algún sistema de acoplamiento de plantillas.

La importancia de éste problema es obvia, dado que los dibujos de ingeniería (que proceden de proyecciones de objetos 3D) son el mecanismo estándar de comunicación entre los departamentos de diseño y fabricación. La reconstrucción de modelos 3D a partir de los dibujos lineales de ingeniería es crítica para el mantenimiento y desarrollo continuado de los objetos representados. La automatización de esta actividad permitiría tomar los objetos existentes e incorporarlos a paquetes de modelado de sólidos, donde se podrían modificar gradualmente para fabricar objetos más nuevos y modernos.

Por otra parte los sistemas de modelado actuales obligan al usuario a emplear un "lenguaje" muy restrictivo. El resultado es que los ordenadores no se utilizan en las fases más conceptuales del diseño. Un boceto permite a un diseñador una rápida exploración de ideas imprecisas. La información que un diseñador pone en una hoja de papel, puede ser interpretada por otro ser humano utilizando su "percepción visual". Nuestra cultura visual nos ayuda a pasar desde una representación axonométrica o perspectiva a una representación mental de los objetos 3D.

Najendra y Gujar publicaron un resumen de varios artículos que trataban la reconstrucción de objetos tridimensionales a partir de sus vistas 2D. Wang y Grinstein completaron el trabajo, realizando una taxonomía de la reconstrucción de objetos 3D a partir de dibujos lineales de proyección bidimensional. Estos autores basaron sus clasificaciones en aspectos tales como la representación interna usada en el proceso de reconstrucción, la naturaleza de los objetos reconstruidos, el número de vistas 2D requeridas, las premisas y el grado de interacción del usuario necesario para corregir la reconstrucción.

1.2.2. DEFINICIONES.

Puesto que los términos que se usan en el problema de la reconstrucción a menudo se solapan y en algunos casos tienen significados que difieren de su uso común, los definiremos brevemente aquí:

OBJETO: Un objeto es un cuerpo sólido tridimensional. En muchos casos usamos el termino "sólido" como sinónimo de objeto.

FRONTERA: La frontera de un objeto es el objeto menos su interior.

MUNDO DEL OBJETO: Es el sistema de Coordenadas del Mundo (WC) en el que se define al objeto.

MODELO: Un modelo es un objeto de forma primitiva predefinida, como un bloque paralelepipédico o una esfera. Un modelo se puede definir en el mundo del objeto o en su sistema de coordenadas locales.

CARA: Una cara de un objeto es una superficie cerrada no vacía y delimitada, no necesariamente plana, que pertenece a la frontera de ese objeto.

ARISTA: Una arista es el segmento lineal compartido por dos caras. Las aristas no tienen que ser necesariamente rectas; dependerá de si las caras compartidas son o no planas.

VERTICE: Un vértice es un punto terminal de una arista.

ESCENA: Una escena es un conjunto finito de objetos.

DIBUJO LINEAL: Un dibujo lineal es la proyección de una escena 3D.

LINEA: Una línea es la proyección de una arista en un dibujo lineal.

UNIÓN: Una unión es la proyección de un vértice en un dibujo lineal.

1.2.3. REPRESENTACIONES DE OBJETOS SÓLIDOS.

Las representaciones más comúnmente utilizadas en el problema de reconstrucción son la Geometría Constructiva de Sólidos (CSG) y la Representación de Fronteras (BRep).

La CSG es un método efectivo y directo para la reconstrucción de un objeto sólido. Este método esta conceptualmente próximo a las técnicas que actualmente se usan en ingeniería para el diseño de piezas mecánicas y, a menudo, da una representación compacta del objeto. Se usa el termino CSG* para incluir también aquellas representaciones que no se denominan a sí mismas CSG pero que se basan en un supuesto igual o similar para la construcción de un objeto complejo a partir de otros más simples, como Aldefeld 83 y Ho 86.

La BRep ofrece una representación del objeto más general, aunque no necesariamente más compacta. Un sólido se modela por un número finito de superficies delimitadas, cada una de las cuales se representa por un conjunto de aristas orientadas que la limitan. Cada arista se representa por dos vértices, definidos en algún sistema de coordenadas. Las superficies que se usan normalmente para describir los objetos sólidos incluyen superficies planas (por ejemplo polígonos), superficies cuadráticas (por ejemplo cilíndricas, cónicas y esféricas) y s*plines*. Las operaciones de construcción en BRep aseguran que el contorno esta bien formado; esto incluye requisitos tales como que "el contorno sea cerrado". La mayor parte de los trabajos de reconstrucción usan BRep.

1.2.4. NÚMERO DE VISTAS.

Por el número de vistas requeridas al dibujo de entrada, existen dos grandes categorías: métodos de vistas múltiples y métodos de vista única.

Los métodos basados en vistas múltiples están más adelantados. Obviamente es mucho más fácil reconstruir objetos 3D a partir de proyecciones de vistas múltiples que a partir de proyecciones de vista única. No obstante dichos métodos suelen estar limitados a considerar las vistas en alzado, planta y perfil. No aceptan convencionalismos normalizados.

La reconstrucción a partir de una única vista presenta más ambigüedades y da más soluciones.

En algunos casos, más de un objeto puede generar la misma, o mismas, vistas al proyectar. Un método se llama de "solución múltiple" si encuentra todos los objetos que puedan corresponderse con la(s) vista(s) 2D, y de "solución única" si se para después de encontrar un objeto.

Premisas y grado de interacción

Para simplificar el problema de la reconstrucción, generalmente se asume que en cualquier proyección de una escena sólo se representan las aristas y los contornos. Es decir, se trabaja con vistas "normalizadas" (en el sentido de que los objetos se representan por medio de aristas y contornos). No se consideran la textura, la gama, el sombreado y otros parámetros adicionales que se están usando en el reconocimiento de objetos. Algunos sistemas distinguen entre aristas vistas y ocultas; otros no distinguen, y los hay que exigen que no se empleen dichas aristas ocultas.

Se suele añadir una limitación adicional sobre la dirección de la proyección en proyecciones paralelas y sobre el centro de la proyección en proyecciones perspectivas. En las proyecciones paralelas, la dirección de la proyección no debe ser paralela a ninguna cara, ni paralela a ningún par de aristas no colineales. En las proyecciones perspectivas, el centro de la proyección no debe ser coplanar con ninguna cara ni coplanar con ningún par de aristas no colineales. A esta limitación se la denomina "Supuesto de Punto de Vista General" y generalmente elimina casos de degeneración potenciales en los que, por ejemplo, una cara se puede proyectar como una línea, o dos aristas distintas se pueden proyectar sobre una línea, ver Sugihara. La reconstrucción a partir de una única proyección no tiene mucho sentido sin este supuesto. Imaginemos el dibujo lineal de un cuadrado: éste podría ser la proyección ortográfica de un bloque de profundidad arbitraria.

Los sistemas existentes también se pueden clasificar en función de la colaboración que requieren por parte del usuario. La distinción básica es entre sistemas automáticos y sistemas guiados. Lafue y Liardet han desarrollado sistemas interactivos para interpretar dibujos lineales de vista única, en donde el problema es cómo realizar interacciones suaves y flexibles entre los usuarios y los ordenadores. Lamb y Bandopadha presentaron un sistema para determinar estructuras 3D a partir de un dibujo lineal impreciso. Su sistema usa reglas heurísticas, más algunas reglas perceptuales, para intentar ajustar automáticamente el dibujo y reconstruir su estructura 3D. Cuando no lo logra, depende de la interacción del usuario para conseguir que el proceso siga.

1.2.5. ESTADO DEL ARTE.

La historia de tratar de comprender el dibujo lineal data de los años 60. Varios autores incluyen buenas referencias a los primeros estudios sobre este tema. En este apartado se analizan los modelos que se han desarrollado en este campo de estudio y se presentan algunos algoritmos típicos de cada una de los distintos métodos.

El primer intento de interpretación del dibujo lineal fue el de Roberts, quien limitó los objetos a un conjunto de modelos predefinidos. Su enfoque intentaba identificar una transformación que (cuando se aplica a uno de los modelos) da una proyección que encaja con el dibujo lineal, en el sentido de mínimos cuadrados. Claramente, este método basado en modelos es muy limitado.

Intentos posteriores eliminaron esta restricción. Guzman desarrolló un programa llamado SEE para analizar dibujos lineales de poliedros sin usar modelos explícitos de objetos. Su programa incluso identifica objetos parcialmente ocultos. Basándose en algunos conceptos heurísticos sencillos sobre las uniones de los dibujos lineales, SEE examina cada unión y agrupa regiones alrededor de la unión si estas regiones están asignadas al mismo cuerpo. Además SEE fusiona los gráficos según otros supuestos heurísticos y crea nuevos nodos. Como muchas reglas heurísticas no tienen base teórica, SEE puede cometer errores. Sin embargo, ha influido en gran medida en los trabajos de investigación posteriores. Si se limita el problema a la interpretación de dibujos de vistas múltiples, tales como los dibujos de ingeniería compuestos de las vistas en alzado, planta y perfil, el problema es más fácil de solucionar.

Las dos aportaciones anteriores pueden considerarse como los precedentes de los trabajos más referenciados en este campo que se exponen a continuación. Dicha exposición se realiza desglosada en función de que el método de reconstrucción sea a partir de vistas múltiples o a partir de una sola vista. Además dentro de cada una de éstos se ha distinguido en función del modelo de representación utilizado y en función de la aproximación utilizada, respectivamente. En la tabla siguiente se ofrece un resumen global de dichos trabajos.

Métodos de vistas múltiples	Métodos de vista única
Métodos Brep	**Métodos de etiquetado**
Idesawa 1973	Huffman 1971 y Clowes 1971
Wesley-Markowsky 1980, 1981	Lees y otros 1985
Sakurai 1983 y Gu y otros 1985	Malik 1987
Preiss 1984	**Métodos del espacio gradiente**
Yan y otros 1994	Mackworth 1973
Masuda y otros 1996	Wei 1987
Métodos CSG	**Métodos de programación lineal**
Aldefeld 1983, 1984	Sugihara 1984, 1986
Ho 1986	**Método Perceptual**
Método de identificación de primitivas	Lamb y otros 1990
Meeran y Pratt 1993	**Método de identificación de primitivas**
	Wang y otros 1989, 1991, 1992
	Método de Optimización
	Lipson y Shpitalni 1996

Organización de los métodos de reconstrucción.

1.2.6. LA RECONSTRUCCIÓN 3D A PARTIR DE UNA VISTA.

Huffman y Clowes establecieron el primer *método de etiquetado* válido para poliedros. Los métodos de etiquetado posteriores trataron de ampliarlo a casos más generales, incluyendo dibujos con aristas ocultas, superficies sin espesor (u objetos "Origami" de papel), y dibujos con objetos curvos. Todos se basaban en reglas heurísticas, hasta que Malik publicó su trabajo. Su método trata los dibujos lineales de escenas formadas por objetos sólidos regulares opacos delimitados por piezas con superficies suaves, con lo que solucionó el problema de etiquetado general.

Los métodos de etiquetado son métodos de interpretación más que de reconstrucción: ofrecen sólo las condiciones necesarias para que un dibujo lineal 2D represente un sólido 3D válido. Además, un dibujo lineal que se puede etiquetar adecuadamente no necesariamente representa un sólido 3D verdadero.

Los *métodos de gradientes* parten del método de Mackworth, que interpreta los dibujos lineales construyendo la imagen de cada plano en el espacio de gradientes. Su programa POLY determina primero el tipo de aristas: las aristas enlazadas (ya sean cóncavas o convexas) y las aristas ocultas. Después resuelve si las aristas enlazadas son cóncavas o convexas mediante la construcción de la imagen de gradientes. Mackworth también demuestra que un dibujo lineal etiquetado puede considerarse incorrecto si no admite una figura reciproca. Por ello, su método del espacio de gradientes se puede usar para detectar dibujos lineales no realizables. No obstante, algunos dibujos lineales no son realizables incluso aunque se puedan construir sus imágenes de gradientes.

Wei amplió la idea de Mackworth, dando las cuatro condiciones necesarias y suficientes para que exista un poliedro en R^3. La diferencia estriba en el hecho de que el gradiente de Wei es un vector tridimensional, mientras que el de Mackworth es sólo bidimensional. Basándose en estas cuatro restricciones, dio las reglas para hallar la solución recursivamente. Wei también estudió posteriormente como ampliar su algoritmo a dibujos no perfectos. Sin embargo, la limitación del método de Wei es que sólo puede manejar poliedros con cuatro grados de libertad. Los poliedros con más grados de libertad son descompuestos en varios poliedros, cada uno de los cuales no puede tener más de cuatro grados de libertad.

El *método de programación lineal* se basa en plantear y resolver las ecuaciones lineales que describen las condiciones que un poliedro debe satisfacer. Sugihara consiguió dar una condición necesaria y suficiente que permitía que un dibujo lineal representase un objeto poliédrico en términos del problema de programación lineal. Su formulación permite resolver el problema de discriminar entre dibujos lineales correctos e incorrectos. Sin embargo, el método de Sugihara no es práctico para realizar la reconstrucción. La condición es tan precisa matemáticamente que algunos dibujos son rechazados simplemente porque los vértices se desvían ligeramente de las posiciones correctas. La dificultad estriba en el hecho de que hay ecuaciones redundantes en el sistema. Sugihara soluciona esta dificultad identificando las ecuaciones redundantes y permitiendo extraer un subconjunto. El método corregido da una solución completa al problema de la reconstrucción de un objeto poliédrico 3D, obteniendo su representación BRep, a partir de una proyección 2D de vista única. Su limitación más evidente es que sólo trata objetos poliédricos. La implementación del método requiere la solución de un gran problema de programación lineal de 3m+n variables, donde m es el número de caras visibles del poliedro y n el número de vértices visibles.

Lamb y Bandopadhy presentaron un sistema para extraer la estructura 3D, a partir de un dibujo lineal impreciso. La idea es dar a los diseñadores que utilizan CAD la capacidad de visualizar interactivamente la interpretación más probable de los dibujos disponibles en cada momento, los cuales pueden ser imprecisos, para permitir los cambios pertinentes cuando la interpretación de los mismos sea incorrecta.

Este "*método perceptual*" difiere de los otros en que no usa métodos numéricos. El algoritmo toma como dato un dibujo axonométrico. Y utiliza los invariantes que aseguran que las líneas paralelas aparecen paralelas, y que las aristas paralelas a los ejes principales se dibujan con longitudes proporcionales a las dimensiones reales. El algoritmo preprocesa el dibujo generando un gráfico de adyacencia (un mapa de vértices y aristas) y después etiqueta el dibujo usando el método de Waltz. Se le asignan al dibujo los ejes principales, los cuales, a su vez, forman los planos principales. También se selecciona el origen. El objetivo del algoritmo es asignar coordenadas 3D a todos los vértices del grafo.

Este modelo ofrece un buen interfaz del usuario para que los diseñadores que utilizan CAD puedan ver objetos 3D. Pero tiene limitaciones en el uso de las reglas de percepción heurísticas. Estas son ciertas en muchos casos, pero no siempre. Por ejemplo, este método corregirá líneas con una ligera inclinación y las hará horizontales o verticales, incluso aunque esta inclinación sea intencionada. Otro problema surge de la determinación precisa de si dos planos son paralelos, o si dos objetos son simétricos. A menos que exista algún supuesto sobre el mundo del objeto (como la rectilinearidad), estas propiedades no se pueden determinar antes de que se asignen las coordenadas. Así pues, este modelo no ofrece una reconstrucción exacta del objeto 3D.

En el campo de la reconstrucción de una representación CSG de un *sólido (método de identificación de primitivas*), se ha avanzado poco hasta fechas recientes. La construcción de una representación CSG a partir de vistas múltiples todavía depende mucho de la interacción del usuario. En la reconstrucción automática de la representación CSG a partir de un sólo dibujo la aportación más importante es la de Wang y Grinstein, quienes presentaron un algoritmo para extraer automáticamente los bloques primitivos de un poliedro. Se supone que el poliedro es rectangular, es decir, los tres planos que se unen para formar un vértice deben ser perpendiculares entre sí. Los autores proponen examinar el dibujo lineal etiquetado (con cualquier esquema de etiquetado válido) y extraer información de los bloques a partir de ciertos tipos de uniones. Wang amplió el algoritmo para abarcar objetos semi-rectilíneos y algunos objetos curvilíneos. Un objeto semi-rectilíneo tiene una dirección axial y cualquier superficie cuya normal no sea perpendicular al eje, es ella misma perpendicular al eje. Wang también introdujo los cilindros como primitivas. Wang dio una solución completa para encontrar la representación CSG de un dibujo lineal de un poliedro general con el tetraedro como nueva forma primitiva. También amplió los algoritmos semi-rectilíneos previos para poder abarcar un mundo del objeto más amplio que incluiría a los objetos curvos. Para cada algoritmo, se presentaban pruebas que muestran que si el dibujo representa un objeto válido el algoritmo termina y genera una representación CSG válida. Se daba una condición general y suficiente, basada en estos algoritmos, para que un dibujo lineal pueda representar un sólido válido. Sus algoritmos también podían manejar algunos dibujos incorrectos, que los esquemas de etiquetado existentes no podían distinguir.

El último método introducido es el *método de optimización*, donde la aportación más significativa es la de Lipson y Shpitalni. Su método de reconstrucción por "inflado" de una representación plana se basa en definir como variables las coordenadas Z de todos los vértices definidos por sus coordenadas (X,Y) en la representación plana. La función objetivo es una formulación matemática de las "regularidades" observables en el dibujo 2D. Los autores distinguen tres tipos de regularidades: (a) las que reflejan alguna relación espacial entre entidades individuales (por ejemplo el paralelismo), (b) las que reflejan alguna relación espacial entre un grupo de entidades (por ejemplo una simetría oblicua en las aristas que definen el contorno de una cara) y (c) las regularidades que afectan a todo el dibujo (como la proporcionalidad entre las longitudes del dibujo y las longitudes reales). Su método tolera las imperfecciones y permite reconstruir una gran variedad de objetos, incluyendo caras planas y cilíndricas. Pero el porcentaje de fallos aumenta al considerar objetos con superficies curvas.

La tabla siguiente muestra un breve resumen de los algoritmos analizados en los apartados anteriores recalcando el tipo de objetos que componen la escena, el número de vistas necesarias, la representación usada, el número de soluciones y si se requiere la interacción con el usuario.

Autores	**Tipos de superficies permitidas**	**Número de vistas necesarias**	**Representación usada**	**Número de soluciones**	**Interacción con usuario requerida**
Huffman 71	Plana	Una	Etiquetado	Múltiple/ única	Ninguna
Clowes 71					
Idesawa 73	Plana	Múltiple	BRep (Fronteras)	Única	Ninguna
Wesley 80,81	Plana	Múltiple	BRep (Fronteras)	Múltiple	Ninguna
Sakurai 83	Plana/ cilíndrica	Múltiple	BRep (Fronteras)	Múltiple	Ninguna
Gu 85	cónica/ esférica/ tórica				
Preiss 84	Plana/ cilíndrica	Múltiple	BRep (Fronteras)	Única	Ninguna
Aldefeld 83	Plana/ cilíndrica	Múltiple	BRep (Fronteras)	Única	Ninguna
Aldefeld 84	Plana/ cilíndrica	Múltiple	CSG*	Única	Mucha
Ho 86	Plana/ esférica cilíndrica	Múltiple	CSG	Única	Mucha
Malik 85	Plana/ cilíndrica	Una	Etiquetado	Múltiple/ única	Ninguna
	cónica/ esférica tórica				
Mackworth 73	Plana	Una	BRep (Fronteras)	Única	Ninguna
Sugihara 86	Plana	Múltiple	BRep (Fronteras)	General	Ninguna
Sugihara 86	Plana	Una	BRep (Fronteras)	General	Alguna
Wei 87	Plana	Una	BRep (Fronteras)	General	Ninguna
Lamb 90	Plana	Una	BRep (Fronteras)	Única	Alguna
Wang 92	Plana/ cilíndrica	Una	CSG	Única/ múltiple	Ninguna
Meeran 93	Plana/ cilíndrica	Múltiple		Única	Ninguna
Yan 94	Plana	Múltiple	BRep (Fronteras)	Única/ múltiple	Ninguna
Lipson 96	Plana/ cilíndrica	Una	BRep (Fronteras)	Única	Ninguna
Masuda 96	Plana/ cilíndrica	Múltiple	BRep (Fronteras)	Única	Ninguna

1.2.7. LA RECONSTRUCCIÓN 3D A PARTIR DE VARIAS VISTAS.

Idesawa estableció el primer método BRep, dando un tratamiento matemático a la geometría implicada en los objetos a reconstruir. En este método, se debe establecer la correspondencia entre el objeto sólido y las tres vistas, en caso contrario, se han de añadir las vistas auxiliares adecuadas para generar una solución única. Incluso si existe la correspondencia entre el sólido y las tres vistas, puede que no se establezca la correspondencia a nivel de punto y línea. Ello es debido a que las personas ven un sólido mediante la consideración simultánea de caras, aristas y vértices, mientras que las maquinas funcionan paso a paso. Así es posible que la figura tridimensional obtenida durante el proceso de reconstrucción implique lo que Idesawa llamo "figuras fantasma", que incluyen líneas fantasma y puntos fantasma, que no tienen ningún significado en el cuerpo sólido.

Idesawa también presentó un sistema práctico para la reconstrucción de poliedros. En él se usan distintos criterios para eliminar las figuras fantasma. Por ejemplo, un vértice 3D debe pertenecer a, al menos, tres aristas 3D, y una arista 3D debe pertenecer al menos a dos caras no coplanares. Un ejemplo de una condición necesaria para que las caras pertenezcan a un sólido es que un vértice definido como la intersección de *n* aristas debe pertenecer a *n* caras.
Aunque Idesawa da cuenta de la existencia de figuras fantasma (y en parte las elimina), su método aun puede generar falsos elementos, dependiendo de las diferentes condiciones usadas. Su método funciona solo para poliedros y no se menciona el caso de solución múltiple.

También dentro de los métodos BRep, Wesley y Markowsky presentaron algoritmos para crear estructuras alámbricas y proyecciones organizadas jerárquicamente desde niveles inferiores a superiores. Por ejemplo, generar vértices 3D a partir de proyecciones 2D, generar aristas 3D a partir de vértices 3D, y a continuación generar sub-caras 3D (caras virtuales) a partir de aristas 3D, agrupar sub-caras 3D para formar sub-objetos 3D (bloques virtuales), y después agrupar sub-objetos 3D, para formar los objetos que se correspondan con las proyecciones 2D dadas.

Estos algoritmos, denominados de proyección, dan buenos resultados en presencia de casos patológicos y de soluciones múltiples, que no se pueden manejar con los métodos anteriores (se pueden encontrar ejemplos en Wesley). La diferencia estriba en su método de agrupar las caras. Las caras se combinan en zonas cerradas que dividen el espacio infinito en subespacios. Las combinaciones de subespacios que dan las proyecciones de entrada son soluciones. De esta manera, el método puede eliminar todos los elementos falsos y asegurar que se pueden obtener todas las soluciones. Pero se limita a objetos poliédricos, o sea, geometrías poliédricas planas cuyas proyecciones contienen solo líneas, y solo funciona con vistas múltiples.

Sakurai amplió el algoritmo de Wesley-Markowsky introduciendo objetos con simetría axial, como cilindros, conos, toros, y esferas. La representación de datos empleada permite que los objetos tengan vértices y aristas implícitas y no-visibles; como aristas tangenciales (por ejemplo, cuando un cilindro es tangente a un plano) y aristas de contorno (por ejemplo, la proyección del contorno de una esfera como se ve en una vista). Su limitación está en el hecho de que (1) los ejes de estos objetos deben ser paralelos a uno de los ejes de coordenadas, y (2) no se permite la intersección de dos objetos si sus proyecciones en las tres vistas no son ni líneas rectas ni arcos circulares.

Gu y otros ampliaron el algoritmo de Sakurai para abarcar una clase mayor de objetos. Para objetos poliédricos y cilíndricos, la restricción de las tres vistas y las posiciones de los objetos se suavizan considerablemente:

1. El eje de cualquier cilindro sólo necesita ser paralelo a uno de los planos de las coordenadas en vez de a un eje de coordenada, y el plano de intersección de este cilindro sólo necesita ser paralelo o perpendicular a ese plano de coordenadas.

2. Se permiten arcos elípticos, hipérbolas y curvas regulares de orden superior en proyecciones ortográficas, con ejes simétricos paralelos a un eje de coordenadas; lo cual aumenta las diferentes clases de intersecciones entre objetos.

Finalmente Preiss modelizó el problema como un problema de propagación de restricciones. Se reduce el conjunto de vértices candidatos (es decir, vértices que se habían extraído de dibujos 2D), al aplicarle ciertas restricciones. En el segundo paso, al aplicar las restricciones a cada arista, se agrupan las aristas candidatas. En el tercer paso, al comparar las partes de las aristas, se suprimen las aristas inconsistentes. En el cuarto y último paso, se construyen bucles a partir de las aristas aplicando las restricciones a las aristas de un bucle y entre anillos. Este algoritmo solo trata el caso de solución única.

Dentro de los métodos que utilizan el modelo de representación CSG, Aldefeld establece el primer método, desarrollando una técnica que usa las tres vistas de una pieza mecánica formada por varios objetos elementales aislados de espesor uniforme. Su algoritmo automático clasifica los objetos en varias clases y usa los distintos parámetros para guiar el proceso de interpretación. Los pasos de entrada son:

1. Introducir la silueta de la base del objeto.

2. Buscar otra vista para determinar los rectángulos simples o compuestos que casen con la base.

3. Buscar la tercera vista para identificar los rectángulos que casen con la base y los rectángulos encontrados en el segundo estadio.

4. Encontrar un conjunto completo de segmentos lineales que construyan el objeto.

Este método necesita los dibujos de ingeniería de tres vistas, con proyecciones ortográficas en alzado, planta y perfil. Es más, se supone que la representación 2D de cada objeto elemental es completa en el sentido siguiente: los datos de entrada deben incluir, para cada objeto elemental, las proyecciones de todas sus aristas y de las de todos los contornos para los que la línea de visión sea tangente a la superficie. Esto significa que cada objeto elemental se debe representar como si fuera un cuerpo aislado. Posteriormente, en 1984 Aldefeld y Richter propusieron un algoritmo semiautomático. No es necesario que cada objeto elemental sea un cuerpo aislado. El procedimiento dirigido por el usuario requiere ciertos conocimientos por parte del usuario del objeto a generar. En muchos casos, el usuario inserta líneas auxiliares, arcos u otras formas primitivas 2D en las vistas especificas para proceder con los cuatro pasos mencionados anteriormente. El usuario itera en los cuatro pasos de la entrada, insertando cada vez las formas primitivas 2D perdidas para generar proyecciones completas de los objetos

elementales (en el sentido anterior). El objeto final se genera aplicando operaciones CSG a los objetos elementales.

El algoritmo propuesto por Ho usa el mismo modelo de objeto CSG*. Las primitivas se introducen a partir de un dibujo de ingeniería y son identificadas por el usuario. El algoritmo deduce todos los parámetros de dimensionado y transformación a partir de 5 puntos introducidos mediante un digitalizador basándose en dos o tres vistas del objeto. El algoritmo construye entonces la representación CSG del objeto. Como este es un modelo más orientado al CAD, también está capacitado para manejar varios tipos de dibujos de ingeniería. Se pueden manejar tanto volúmenes poliédricos (por ejemplo, cuboides y tetrapirámides) como volúmenes con superficies curvas sencillas (por ejemplo, conos, cilindros y esferas), con la restricción de que los ejes de estos volúmenes sean perpendiculares a uno de los planos de proyección u oblicuos a dos de ellos. Este algoritmo también se puede usar de forma plenamente interactiva, sin que genere ningún retraso notable en la respuesta que podría producirse al necesitar operaciones computacionales complejas.

S. Meeran y M.J. Pratt propusieron en 1993 un método enfocado hacia la fabricación, basado en la identificación de primitivas. Su aproximación utiliza la teoría del reconocimiento de patrones y la interrelación entre las formas de los objetos.
Mas recientemente, en 1994, Qing-Wen Yan y otros han propuesto un algoritmo en la línea de los de Wesley y Markowsky.

Finalmente, dentro del método de vistas múltiples, las últimas referencias obtenidas corresponden a H. Masuda y N. Masayaki quienes han trabajado en la elaboración de un algoritmo que reconstruye modelos BRep de objetos poliédricos y cilíndricos.

1.3. ANTECEDENTES.

Este tema de investigación se entronca en una línea de trabajo que se encuentra en fase de desarrollo desde el año 1994 por un grupo de investigación dirigido por José María Gomis Martí del Departamento de Expresión Gráfica en la Ingeniería (DEGI) de la Universidad Politécnica de Valencia (UPV).

Tras la revisión del estado del arte, en el momento de iniciarse los trabajos, los primeros pasos se dirigieron hacia la restitución de un sistema de coordenadas tridimensional a partir de su proyección axonométrica oblicua. Dicho trabajo, resuelto mediante el empleo de la programación simbólica y de un método numérico, sugirió la vía de la utilización de las axonometrías oblicuas en los trabajos que se efectuaron posteriormente.

A partir de aquí el planteamiento general de reconstrucción fue dirigido a través de las dos vías especificadas al resumir el estado del arte. Es decir, a través del método que utilizan vistas múltiples ortográficas y a través del método que realiza la reconstrucción a partir de una única vista. Ambas vías se enfocaron a partir de la utilización de axonometrías oblicuas. En la primera, se usa como elemento auxiliar para la determinación de la topología de los objetos, utilizando para ello un método derivado

del Teorema de Pohlke. En la segunda, se utiliza la propia axonometría, como dato de partida para el algoritmo de reconstrucción.

Los resultados del trabajo "restitución de un sistema de coordenadas tridimensional a partir de su proyección axonométrica oblicua" fueron presentados en el VII Congreso de Ingeniería Gráfica celebrado en Vigo en junio de 1995.

Por lo que respecta al trabajo realizado hasta la fecha en relación a la reconstrucción a partir de una vistas múltiples ortográficas, los primeros resultados se materializaron en la obtención de un algoritmo que permite la generación automática de axonometrías oblicuas a partir de las tres vistas diédricas de un objeto poliédrico. Dichos resultados fueron presentados en la comunicación "Generación automática de axonometrías oblicuas de objetos a partir de sus vistas ortográficas" en la Reunión Internacional de Expresión Gráfica en la Ingeniería y en la Arquitectura EGRAF 96 celebrada en Camagüey (Cuba) en Diciembre de 1996. Posteriormente dicho algoritmo fue complementado con otro que permite reconstruir dicho objeto en tres dimensiones. Una versión básica, ya operativa de dicho algoritmo fue presentada en el I Seminario Italo-Español de Diseño de máquinas celebrado en Nápoles en Junio de 1996. Los resultados definitivos hasta la fecha fueron presentados en la comunicación "Reconstrucción de Modelos Poliédricos a partir de sus vistas Normalizadas" en el XII Congreso Nacional de Ingeniería Mecánica celebrado en Bilbao en febrero de 1997.

Finalmente, en relación al método de reconstrucción a partir de una única vista, el grupo de trabajo ha desarrollado un algoritmo de reconstrucción que será presentado en el próximo Congreso de Ingeniería Gráfica a celebrar en Bilbao/San Sebastián el próximo mes de Junio.

1.4. OBJETIVOS.

A pesar de los importantes avances en CAD, los diseñadores todavía prefieren el lápiz y el papel. Especialmente en las fases más conceptuales del diseño, en las que se baraja una colección incompleta de requisitos e ideas abstractas sobre lo que el producto diseñado deberá ser. Por otra parte, los diseños ya existentes están especificados en el soporte tradicional: los planos normalizados. Por lo que la tarea de actualización y modificación de viejos diseños requiere generar modelos sólidos de los objetos existentes. Mejorar ambos aspectos, es el reto de la **reconstrucción** de modelos 3D a partir de dibujos de ingeniería y en base a ellos se formulan los dos objetivos generales de este trabajo que se exponen a continuación.

- El primer objetivo general es llegar a la reconstrucción automática de planos de ingeniería.

- El segundo objetivo general es crear una herramienta de diseño que permita especificar modelos utilizando bocetos.

Reconstrucción automática de planos

La finalidad de este objetivo es rescatar la información acumulada en planos en las oficinas de ingeniería.

Para que la reconstrucción fuera automática no basta con la capacidad de "vectorizar" los planos. De ese modo únicamente se obtiene una base de datos 2D: se mejora el almacenamiento de la información que ya está disponible y se incrementa la eficiencia en su reutilización, pero no se aumenta la capacidad de manipulación de los diseños especificados en dichos planos. Por contra, si se reconstruye el diseño a partir de las representaciones del mismo almacenadas en planos de ingeniería, se pasa a disponer de un modelo tridimensional, a partir del cual se pueden generar automáticamente nuevos planos. Pero también se pueden hacer modificaciones "remodelando" directamente en 3D. Por tanto, no sólo se incrementa la eficiencia sino que se potencia la creatividad, ya que se pasa de trabajar con información 2D a trabajar con información 3D, respetando e incorporando todo el "know-how" de la empresa.

Para alcanzar éste objetivo, las líneas de trabajo que es necesario explorar (además de la vectorización de planos y la reconstrucción de modelos a partir de sus vistas principales, que ya están siendo abordadas) es la consideración de la reconstrucción de toda la información contenida en los planos de ingeniería en forma de normas y convencionalismos. Esto significa disponer de la capacidad de reconstruir a partir de un conjunto variable de vistas, que pueden incluir diferentes combinaciones de vistas principales y auxiliares. Además significa reconstruir formas especificadas en vistas cortadas y convencionalismos. Por último significa poder incorporar la información dimensional contenida en la acotación, y la información de fabricación contenida en todo tipo de símbolos normalizados (tolerancias, acabados superficiales, procedimientos de fabricación, etc.).

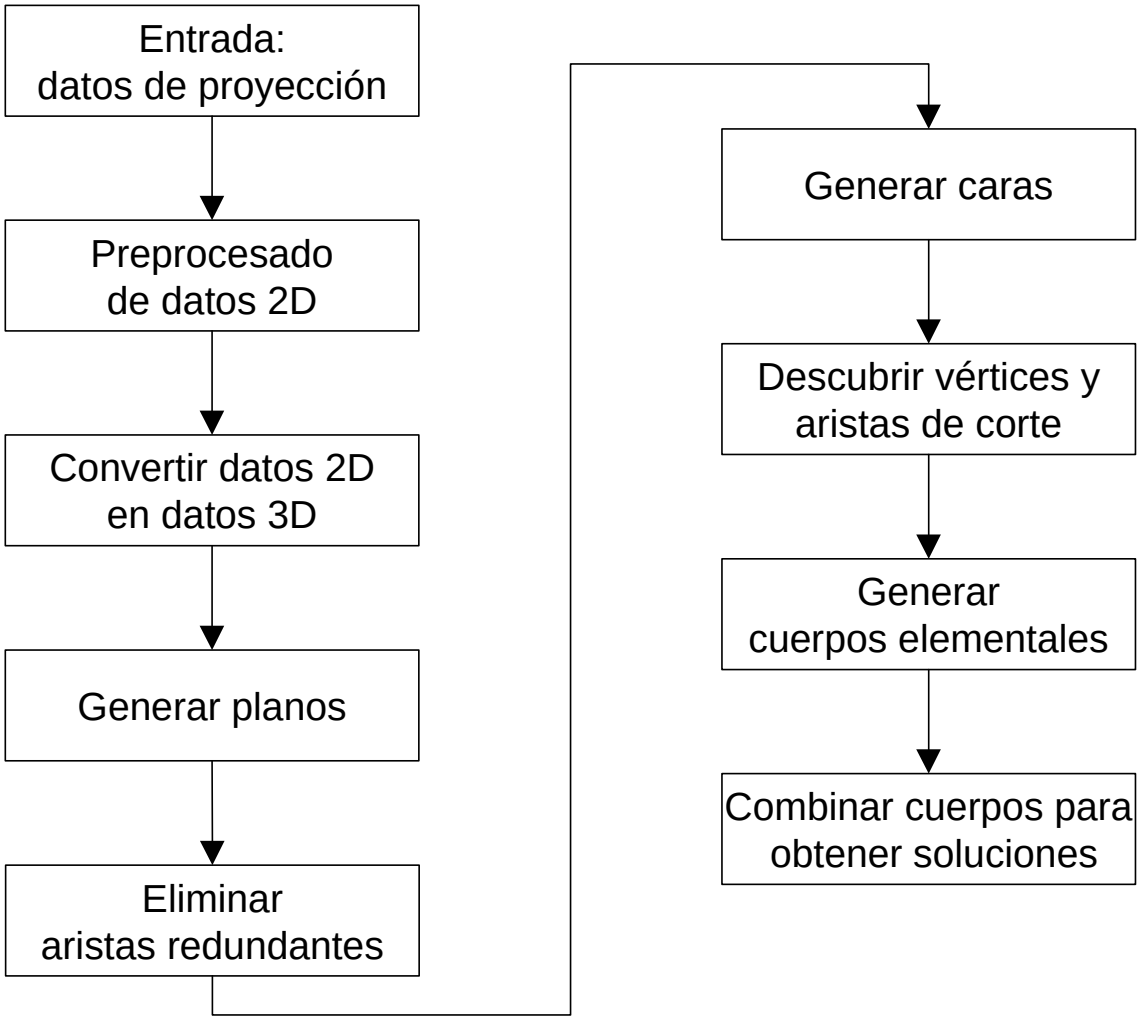
"De rebote" esta línea requiere examinar la normalización de las representaciones gráficas actualmente en uso, para eliminar las ambigüedades, redundancias e indefiniciones, que un técnico entrenado es capaz de detectar y corregir, pero que resultaría mucho más complicado conseguir que un sistema de reconstrucción adquiriese dicho grado de "experiencia" y capacidad de decisión. Tal línea de trabajo redundaría en una simplificación de la tarea de reconstrucción, y, lo que es más importante, en una mejora del lenguaje gráfico normalizado. Se obtendría una "normalización de la normalización".

1.5. DESCRIPCIÓN DEL PROYECTO.

La finalidad de este proyecto es la implementación del algoritmo eficiente de Qing-Wen Yan para la reconstrucción del modelo tridimensional de un cuerpo poliédrico a partir de sus proyecciones ortográficas de planta, alzado y perfil.

El algoritmo utiliza información tal como las líneas continuas o discontinuas en las proyecciones (datos de entrada) para reducir la cantidad de caras y cuerpos potencialmente computables. También se reduce eficazmente el posible número de

objetos ambiguos, las ambigüedades causadas por el hecho que muchos objetos pueden tener el mismo conjunto de proyecciones dado por los datos de entrada, y el algoritmo puede reconocer todos los sólidos correctos de un conjunto de proyecciones dado. Este algoritmo puede tratar las cuestiones de cómo identificar los datos de entrada, cómo generar el apropiado modelo alámbrico más eficazmente, cómo aprovechar los atributos de línea continua/discontinua para acelerar el proceso de encontrar las soluciones correctas, y cómo garantizar que todas las soluciones posibles y correctas son encontradas. Veamos ahora un diagrama de flujo de la estructura del algoritmo:



Vamos ahora a hacer una breve descripción de todos los pasos de este algoritmo de reconstrucción 3D:

I. *Preprocesado de datos.* Hay que realizar un análisis de los datos de entrada, porque la información de los datos de entrada es información en dos dimensiones y para realizar el cambio a tres dimensiones hay que obtener toda la información adicional que se ha perdido en el paso de tres a dos dimensiones.

II. *Construcción del modelo alámbrico.* Este paso consiste en a partir de toda la información generada por el paso anterior, información de vértices y aristas en

2D, obtener un objeto 3D, modelo alámbrico de vértices y aristas 3D, del cual, vamos a ir eliminando información en los pasos siguientes.

III. *Generación de planos.* A partir del modelo alámbrico, construir todos los posibles planos que se pueden generar con las aristas del objeto 3D.

IV. *Análisis de la información de líneas discontinuas.* Para cada arista 2D discontinua, vamos a comprobar si la arista 3D generada a partir de ella, tiene un plano que la oculta desde el punto de vista de la proyección, eliminando las aristas 3D que no se oculten detrás de un plano.

V. *Generación de caras.* En este paso se generan todas las caras que forman parte del objeto 3D obtenido en pasos anteriores.

VI. *Obtención de vértices y aristas de corte.* Examinar todas las caras generadas en el paso anterior para obtener los vértices o aristas de corte entre dos caras.

VII. *Generación cuerpos elementales.* A partir de las caras, se generan todos los cuerpos u objetos simples y cerrados que se puedan generar.

VIII. *Verificación final.* A partir de los cuerpos elementales, se obtiene todas las posibles combinaciones de estos cuerpos, pasando estas combinaciones una verificación que sólo la pasarán las combinaciones que formen objetos correctos. Esta verificación consiste en la proyección de los datos 3D para obtener los datos 2D y compararlos con los datos de entrada. Se pueden obtener varios objetos correctos.

El modelo obtenido es un modelo completo que nos sirve para poder representar sin pérdida de información, de manera inequívoca y completa la estructura topológica y la forma geométrica de un objeto tridimensional.

1.6. HERRAMIENTAS Y ENTORNO DE DESARROLLO.

El lenguaje de programación C.

El C es un lenguaje de programación ampliamente difundido. Está encuadrado dentro de los lenguajes a medio camino entre los lenguajes de alto nivel y los de bajo nivel (ensambladores), incluyendo las ventajas de ambos. Por una parte contiene elementos propios de lenguajes de alto nivel, entre ellos, un conjunto reducido de sentencias de control y manipulación de datos, que pueden utilizarse para definir construcciones de alto nivel. Pero por otra parte, a diferencia de un lenguaje de alto nivel, donde se ha diseñado y se ha incorporado previamente todo aquello que posiblemente va a querer el programador, el C permite definir rutinas para llevar a cabo comandos de alto nivel. Estas rutinas se llaman funciones y son muy importantes en el lenguaje C. Puede diseñarse una librería de funciones específicas de su programa.

El C manipula bits, bytes y puede operar directamente sobre los caracteres. En C estos procedimientos se llevan a cabo por funciones que no son propiamente parte del

lenguaje, sino que se proporcionan como parte de una librería estándar. Estas funciones son rutinas especialmente escritas en el propio C y que llevan a cabo esas operaciones. El código en C es muy portable, lo que facilita la tarea de poder volver a ser utilizado en un nuevo ordenador con un procesador diferente. La mayoría de programas de aplicación tendrán que ser escritos una sola vez y compilados en el compilador C escrito para el nuevo procesador, lo que se traduce en un considerable ahorro de tiempo y dinero.

Los programas generados por compiladores C suelen ejecutarse tan rápidamente como los escritos en lenguaje ensamblador, consiguiendo una ventaja adicional; *velocidad del ensamblador con la comodidad y legibilidad del C.*

Al contrario de como ocurre en Pascal, el compilador C está marcado por la rivalidad entre Microsoft y Borland en el mercado. Ambas compañías disponen de varios programas: Microsoft de su *QuickC* y el gran compilador de C *MSC*, Borland dispone de *Turbo C++* y *Borland C++*, ambos sistemas de desarrollo completos para C y C++. El compilador utilizado para la realización de este proyecto final de carrera ha sido el *Borland C++* versión 4.5.

El entorno WINDOWS.

Microsoft Windows es un entorno gráfico de ventanas para ordenadores personales que utilizan como sistema operativo MS-DOS en las versiones 3.X y que con la aparición del Windows 95, 98 ya pasa a denominarse como sistema operativo de 32 bits. La *multitarea* es una de sus características destacables que le capacita para ejecutar más de una aplicación al mismo tiempo. Puesto que múltiples aplicaciones pueden estar abiertas a la vez, no se necesita salir de una para pasar a otra, pudiendo conmutar entre ellas fácilmente. Esto se traduce en que, por ejemplo, se puede dejar al sistema ordenando una base de datos en una ventana, mientras se escribe una carta en otra. La multitarea sólo es posible en máquinas equipadas con procesadores i80386, i80486, siendo más recomendados los procesadores de última generación Pentium, Pentium II, AMD K6,... La aplicación desarrollada para este proyecto funcionará tanto en Windows 3.X, como en los últimos Windows 95 y Windows 98.

El modelo de programación de WINDOWS.

No importa las herramientas de desarrollo que utilice, programar para Windows es totalmente diferente al viejo modo de sentencias lineales y agrupadas o programas orientados a transacciones.

Cuando se escribe una aplicación en C para MS-DOS, el único e importante requisito es una función llamada *main*. El sistema operativo llama *main* a la ejecución del programa por parte del usuario, momento desde el cual puede utilizar cualquier estructura de programación. Cuando el sistema operativo de Windows carga un programa, éste le denomina función de programa principal (función *Winmain*). En algún lugar a lo largo de la aplicación debe haber la función *Winmain*, donde ejecuta algunas tareas específicas.

Su tarea más importante es la de crear la ventana principal de la aplicación, que debe tener su propio código para procesar los mensajes que Windows le envíe. La diferencia principal entre un programa escrito en MS-DOS y un programa escrito para

Windows, consiste en que la aplicación bajo MS-DOS llama al sistema operativo para conseguir la entrada del usuario, mientras que un programa de Windows procesa la entrada del usuario mediante mensajes del sistema operativo.

Muchos programas de MS-DOS escriben directamente en la memoria de vídeo o al puerto de la impresora. El inconveniente de esta técnica es suministrar software de dispositivos para cada tarjeta de vídeo y para cada modelo de impresora. Windows introdujo una capa de abstracción llamada *Interfaz de Dispositivo Gráfico (GDI)*. Windows suministra los controladores para vídeo e impresoras, de tal forma que su programa no necesita conocer el tipo de tarjeta de vídeo o impresora conectada a sistema.

Estas han sido algunas de las características más importantes que diferencian la programación bajo MS-DOS de la programación en entorno Windows, aunque hay varias más como puedan ser: programación basada en recursos, gestión de memoria, bibliotecas de enlace dinámico (DLL), interfaz de programación de aplicaciones (API), etc.

Capítulo 2
ALGORITMO PARA LA RECONSTRUCCIÓN 3D DE UN OBJETO

El objetivo de este capítulo es entrar en el detalle de todos los pasos que se siguen para generar un objeto en 3D a partir de su representación ortográfica 2D según el algoritmo de reconstrucción 3D de Qing-Wen Yan. Así mismo, se profundizará en todos los aspectos relacionados con los pasos del algoritmo, necesarios a la hora de entender perfectamente todo lo que se obtiene en cada paso. Además, hay que añadir que cada paso del algoritmo toma toda la información de entrada generada en el paso anterior, para que tras procesarla, sea generada información de salida para el paso siguiente. Podemos decir entonces que la información de entrada del primer paso son los datos de partida del algoritmo, y los datos de salida del último paso son los resultados del algoritmo.

Vamos ahora a hacer una breve descripción de todos los pasos de este algoritmo de reconstrucción 3D:

IX. *Preprocesado de datos.* Hay que realizar un análisis de los datos de entrada, porque la información de los datos de entrada es información en dos

dimensiones y para realizar el cambio a tres dimensiones hay que obtener toda la información adicional que se ha perdido en el paso de tres a dos dimensiones.

X. *Construcción del modelo alámbrico.* Este paso consiste en a partir de toda la información generada por el paso anterior, información de vértices y aristas en 2D, obtener un objeto 3D, modelo alámbrico de vértices y aristas 3D, del cual, vamos a ir eliminando información en los pasos siguientes.

XI. *Generación de planos.* A partir del modelo alámbrico, construir todos los posibles planos que se pueden generar con las aristas del objeto 3D.

XII. *Análisis de la información de líneas discontinuas.* Para cada arista 2D discontinua, vamos a comprobar si la arista 3D generada a partir de ella, tiene un plano que la oculta desde el punto de vista de la proyección, eliminando las aristas 3D que no se oculten detrás de un plano.

XIII. *Generación de caras.* En este paso se generan todas las caras que forman parte del objeto 3D obtenido en pasos anteriores.

XIV. *Obtención de vértices y aristas de corte.* Examinar todas las caras generadas en el paso anterior para obtener los vértices o aristas de corte entre dos caras.

XV. *Generación cuerpos elementales.* A partir de las caras, se generan todos los cuerpos u objetos simples y cerrados que se puedan generar.

XVI. *Verificación final.* A partir de los cuerpos elementales, se obtiene todas las posibles combinaciones de estos cuerpos, pasando estas combinaciones una verificación que sólo la pasarán las combinaciones que formen objetos correctos. Esta verificación consiste en la proyección de los datos 3D para obtener los datos 2D y compararlos con los datos de entrada. Se pueden obtener varios objetos correctos.

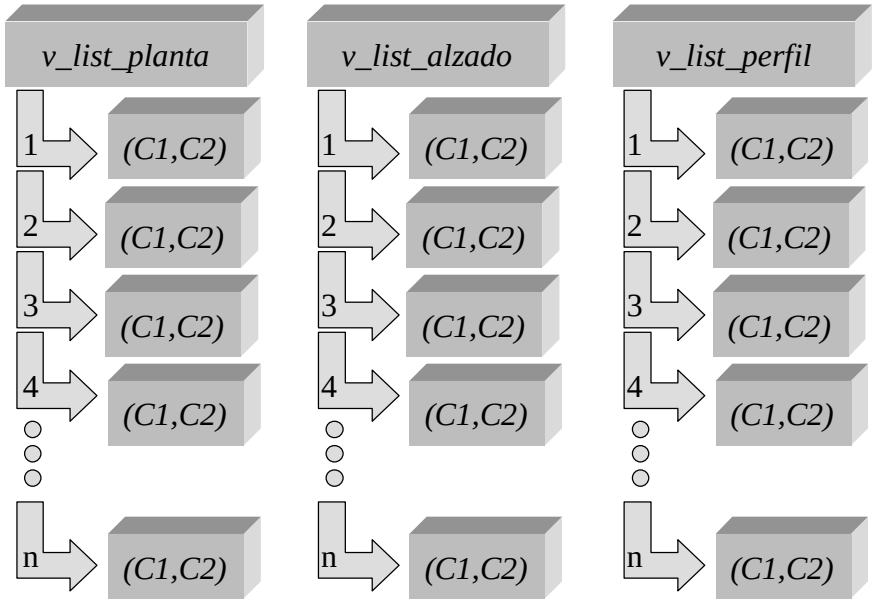
Pasamos entonces a detallar cada paso.

2.1. INFORMACION Y DATOS DE ENTRADA.

Para simplificar un poco, asumimos que todos los datos de vértices y aristas 2D los tenemos almacenados en varias listas dependiendo si son datos de la proyección de *planta, alzado* o *perfil*.

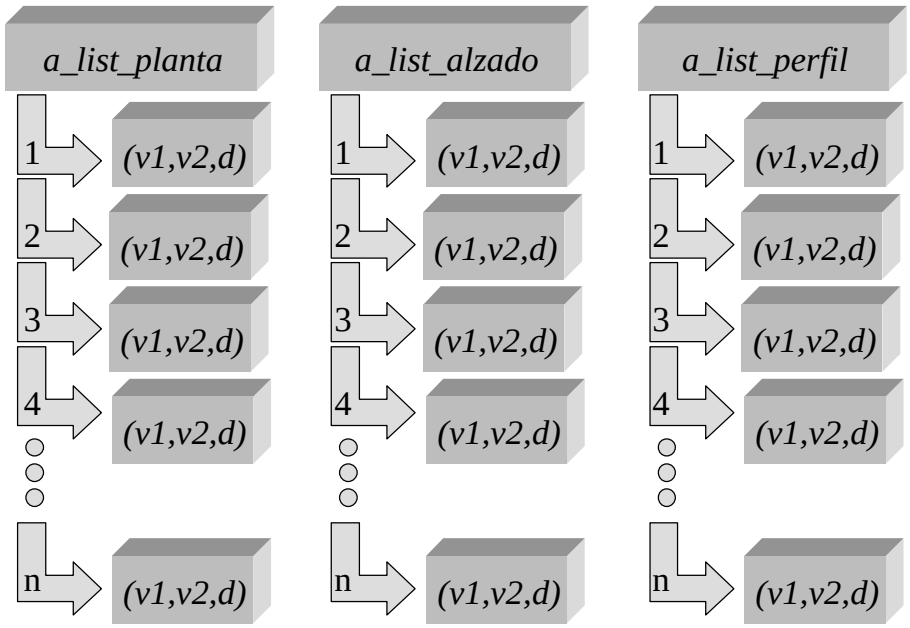
Cada entrada de la lista de vértices nos proporciona los datos: *(coord1, coord2)*, donde, para la planta, *coord1* será la coordenada X del vértice y *coord2* será la coordenada Y del vértice; para el alzado, *coord1* será la coordenada X del vértice y *coord2* será la coordenada Z del vértice; y para el perfil, *coord1* será la coordenada Y del vértice y *coord2* será la coordenada Z del vértice. También seleccionaremos el vértice deseado según su índice dentro de la lista de vértices. Nombraremos las tres listas con los siguientes nombres: *v_list_planta, v_list_alzado y v_list_perfil.*

En la figura siguiente podemos ver gráficamente la estructura de las listas de vértices:



Cada entrada de la lista de aristas nos proporciona los datos: *(v1, v2, discont)*, donde, para la planta, *v1* y *v2* serán los índices de la lista de vértices *v_list_planta*; para el alzado, *v1* y *v2* serán los índices de la lista de vértices *v_list_alzado*; y para el perfil, *v1* y *v2* serán los índices de la lista de vértices *v_list_perfil;* para cualquier caso la variable *discont* será una variable booleana que nos indica si la arista es discontinua (*true)* o continua (*false)*. También seleccionaremos la arista deseada según su índice dentro de la lista de aristas. Nombraremos las tres listas con los siguientes nombres: *a_list_planta, a_list_alzado y a_list_perfil.*

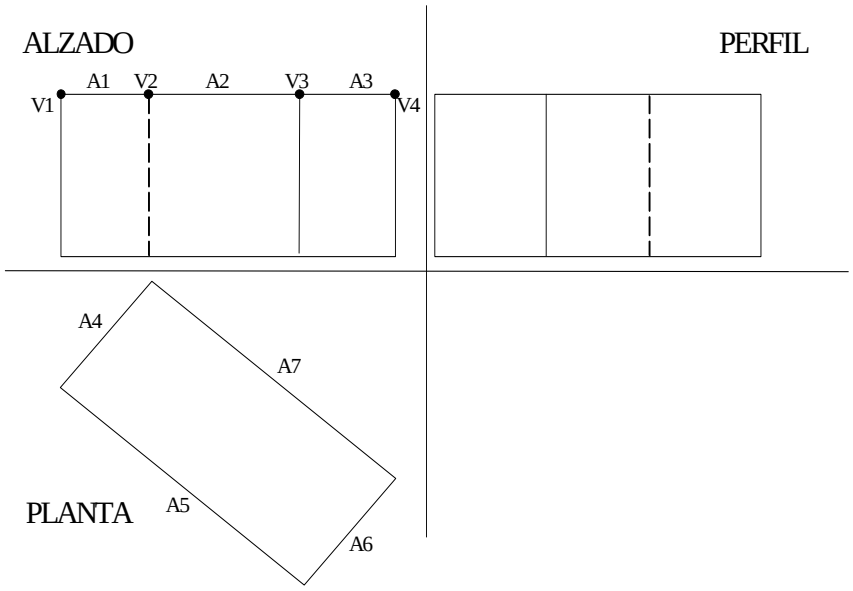
En la figura siguiente podemos ver gráficamente la estructura de las listas de aristas:



2.2. PREPROCESADO DE DATOS.

Este es el primer paso del algoritmo de reconstrucción. Consiste en un análisis de los datos de entrada, porque si no tratamos correctamente la información de los datos de entrada puede llevar al algoritmo a no generar las aristas 3D correctas. Vamos a ver primero un ejemplo de este problema, y a continuación se verá la solución a este caso.

2.2.1. PROBLEMA.



- Los datos de entrada nos proporcionan en el alzado, las aristas A1, A2 y A3 entre los vértices V1, V2, V3 y V4. En realidad nos tendrían que proporcionar las aristas A1 y A3, con sus equivalentes de la planta A4 y A6, para generar con sus equivalentes del perfil una arista 3D correcta, pero además necesitamos una arista entre V1 y V3 (equivalente a la A5 de la planta) y una arista entre V2 y V4 (equivalente a la A7 de la planta), además la arista A2 no interviene en la formación de ninguna arista 3D.

- También nos podían haber proporcionado sólo una arista entre V1 y V4, la cual habría que descomponer para poder obtener las aristas "útiles" en la formación de aristas 3D.

Por lo tanto tenemos que analizar los datos de entrada porque nos pueden proporcionar: aristas alineadas, solapadas o no; una única arista, en la que hay una serie de vértices 2D que corresponden a otras aristas 2D.

SOLUCIÓN: Para un conjunto de vértices dados y alineados (siendo el número de vértices mayor que dos), si hay una arista o varias que pasan por todos los puntos, obtener todas las combinaciones de aristas que se puedan generar a partir de esos vértices, e incluirlos (sólo los que no estén) en la lista de aristas de esa vista. Las aristas

que se generen y sean “inútiles” para la generación del objeto 3D serán eliminadas por el paso siguiente del algoritmo (construcción del modelo alámbrico).

2.2.2. OBTENCIÓN DE VÉRTICES ALINEADOS.

Uno de los problemas que surgen para llevar a cabo la solución anterior es cómo saber cuando dos aristas están alineadas, o lo que es lo mismo, cuando cuatro vértices (dos por arista) están alineados. Para ello hay que recurrir a una proposición de la geometría analítica que dice lo siguiente:

Proposición:

Sea una recta ***r*** determinada por un punto **P**(x_0, y_0, z_0) y un vector **v**(v_1, v_2, v_3), y una recta ***s*** determinada por un punto **Q**(x_1, y_1, z_1) y un vector **u**(u_1, u_2, u_3). Sean las matrices:

$$A = \begin{bmatrix} v_1 & v_2 & v_3 \\ u_1 & u_2 & u_3 \end{bmatrix} \qquad B = \begin{bmatrix} v_1 & v_2 & v_3 \\ u_1 & u_2 & u_3 \\ x_1 - x_0 & y_1 - y_0 & y_1 - y_0 \end{bmatrix}$$

Se tiene que si:

1. Rango(A)=Rango(B)=1, las rectas son coincidentes.
2. Rango(A)=1, Rango(B)=2, las rectas son paralelas no coincidentes.
3. Rango(A)=Rango(B)=2, las rectas se cortan en un punto.
4. Rango(A)=2, Rango(B)=3, las rectas se cruzan.

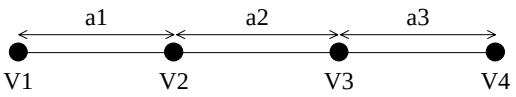
A dos aristas podemos aplicarle la proposición anterior para saber si están alineadas de la forma siguiente:

- La recta *r* definida por *P* y *v* se obtiene de los dos vértices de la primera arista (vert1, vert2), *P* será por ejemplo vert1, y el vector *v* se obtiene de vert2-vert1.
- La recta *s* definida por *Q* y *u* se obtiene de los dos vértices de la segunda arista (vert3, vert4), *Q* será por ejemplo vert3, y el vector *u* se obtiene de vert4-vert3.

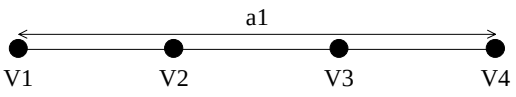
2.2.3. ALGORITMO APLICADO.

Considerando los siguientes casos que se pueden dar de los datos de entrada:

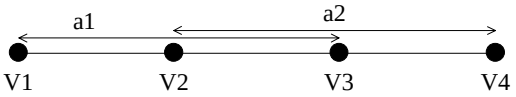
- Una arista cada dos vértices.



- Una arista que une los dos vértices más distantes.



- Aristas solapadas.



El algoritmo siguiente soluciona estos casos, e incluso si hay varios grupos de aristas alineados y entre los distintos grupos de aristas no están unidos mediante aristas.

1. Se crea una lista de todos los vértices de aristas alineadas, y se ordena esta lista de forma que el vértice anterior y posterior de la lista de un vértice dado son los que están más cerca, midiendo las distancias con respecto a la recta que alinea los vértices.
2. Se seleccionan los primeros vértices unidos por aristas de la lista de vértices (creada en el punto 1) y el resto de vértices se guardan en otra lista llamada "basura".
3. Se obtienen todas las combinaciones de dos vértices de la lista de vértices seleccionados, cada combinación de dos vértices formará una arista que se insertará en la correspondiente lista de aristas 2D (planta, alzado o perfil), sólo si no existe la arista.
4. Si la lista "basura" no tiene ningún vértice o sólo tiene dos vértices que forman una arista separada entonces el algoritmo acaba. En otro caso, se "reciclan" los vértices de la lista "basura", esto es, se seleccionan los siguientes vértices unidos por aristas de la lista "basura" (eliminándolos de esta lista) y se vuelve al paso 3.

Después de aplicar este algoritmo, las listas de los datos de entrada 2D están preparadas para aplicar el siguiente paso del algoritmo de reconstrucción, en el que pasaremos ya a trabajar en 3D.

2.3. CONSTRUCCIÓN DEL MODELO ALÁMBRICO.

En este punto se va a explicar el proceso, mediante el cual se generan todos los vértices y aristas 3D a partir de los vértices y aristas 2D de las proyecciones de los datos de entrada. Durante este proceso de reconstrucción se pueden generar aristas redundantes y patológicas, que serán tratadas en este y otros puntos.

En esta sección se expone el algoritmo de construcción del modelo alámbrico (*Wireframe Construction)* que puede eliminar aristas redundantes y patológicas. El algoritmo combina la *búsqueda en un árbol de decisión* basado en proyecciones del alzado, el algoritmo de *eliminación de aristas redundantes*, y un conjunto de reglas para *eliminar aristas patológicas*.

Veamos una serie de conceptos relacionados con el modelado geométrico. Podemos clasificar la información que tenemos almacenada de un objeto de dos formas: información geométrica e información topológica. La información geométrica es la información propia de los objetos, como por ejemplo los vértices y las aristas que forman el objeto. La información topológica es la que se establece mediante la relación entre distintos objetos. También podemos definir como modelo de proyecto a toda la información geométrica y topológica que tenemos en nuestra base de datos de los objetos. Entonces definimos en modelado geométrico como el proceso de creación y manipulación de la información de la base de datos.

Podemos realizar una clasificación de los modelos geométricos de la forma:

- Modelo alámbrico: almacena las coordenadas de los vértices y las aristas que unen los vértices. Tiene una serie de inconvenientes como la ambigüedad en la representación, también puede producir una pérdida de las líneas de silueta en algunos objetos, y además puede llevar a representar una serie de objetos imposibles o sin sentido.

- Modelo de superficies poliédricas: aproxima cualquier sólido por un conjunto determinado de caras planas. El inconveniente que tiene es que hay imprecisiones debido al facetado y si se precisa mayor precisión necesitamos mayor número de caras, lo que provoca un mayor coste.

- Modelo de superficies esculpidas: se suministra una serie de puntos y se genera un conjunto de curvas y/o un conjunto de superficies a partir de la nebulosa de puntos.

- Modelado sólido: considera el interior del objeto macizo, además de su geometría.

Este algoritmo genera el modelo alámbrico de un objeto sólido a partir de las listas de vértices y aristas 2D de cada proyección (*v_list_planta, v_list_alzado, v_list_perfil, a_list_planta, a_list_alzado, a_list_perfil*).

2.3.1. NOCIONES DE GEOMETRÍA ANALÍTICA.

Se va a realizar una serie de definiciones y proposiciones de la geometría analítica que se van a utilizar en los puntos siguientes.

2.3.1.1. Las ecuaciones de la recta en R^3.

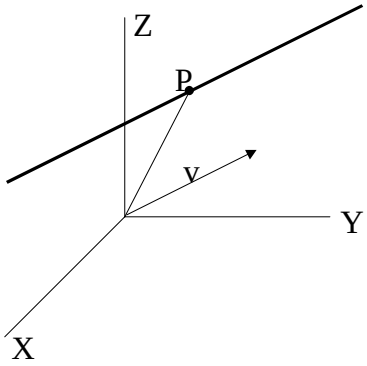
Una recta queda perfectamente determinada si conocemos:
a) Un punto *P* de una recta.
b) Un vector libre *v*, que indica la dirección de la recta en el espacio.

Una determinación alternativa, aunque equivalente, consiste en conocer dos puntos, *P* y *Q*, de la recta. Es equivalente porque el vector libre que admite como representante el vector PQ indica, evidentemente, la dirección de la recta.

Hallar las ecuaciones de una recta en R^3 consiste en explicitar la condición que debe verificar un punto cualquiera X de R^3 para pertenecer a la recta.

La condición lógica consiste en imponer que el vector libre *u*, que admite como representante el vector PX, sea paralelo al vector *v* citado antes.

Fijemos la notación para las coordenadas de los puntos y vectores citados:



$P(x_0, y_0, z_0)$
$X(x, y, z)$
$v = (v_1, v_2, v_3)$
$u = (u_1, u_2, u_3)$

Por la definición dada de *u*, y de su vector asociado, se tiene que $u_1 = x - x_0$, $u_2 = y - y_0$, $u_3 = z - z_0$, con lo que la condición $u = \alpha v$ se expresa así:

Ecuación Vectorial de la Recta: $(x - x_0, y - y_0, z - z_0) = \alpha \cdot (v_1, v_2, v_3)$

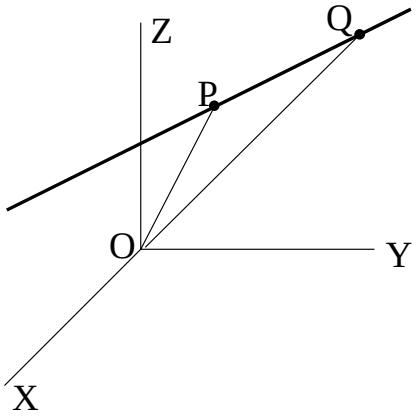
Teniendo en cuenta que $\alpha(v_1, v_2, v_3) = (\alpha v_1, \alpha v_2, \alpha v_3)$ y la igualdad entre vectores resultan las tres ecuaciones siguientes:

Ecuaciones Paramétricas de la Recta:
$$\begin{cases} x = x_0 + \alpha \cdot v_1 \\ y = y_0 + \alpha \cdot v_2 \\ z = z_0 + \alpha \cdot v_3 \end{cases}$$

Cuando v_1, v_2, v_3 son **no nulos**, se obtiene, eliminando el parámetro α:

Ecuaciones de la Recta en Forma Continua: $\dfrac{x - x_0}{v_1} = \dfrac{y - y_0}{v_2} = \dfrac{z - z_0}{v_3}$

Si se tienen como datos dos puntos de la recta, $P(x_0, y_0, z_0)$ y $Q(x_1, y_1, z_1)$, el vector PQ es un representante del *v*, por lo que $v_1 = x_1 - x_0$, $v_2 = y_1 - y_0$, $v_3 = z_1 - z_0$. Sustituyendo en las ecuaciones anteriores:



$$(x - x_0, y - y_0, z - z_0) = \alpha \cdot (x_1 - x_0, y_1 - y_0, z_1 - z_0)$$

$$\begin{cases} x = x_0 + \alpha \cdot (x_1 - x_0) \\ y = y_0 + \alpha \cdot (y_1 - y_0) \\ z = z_0 + \alpha \cdot (z_1 - z_0) \end{cases}$$

$$\frac{x - x_0}{x_1 - x_0} = \frac{y - y_0}{y_1 - y_0} = \frac{z - z_0}{z_1 - z_0}$$

con $x_1 - x_0$, $y_1 - y_0$, $z_1 - z_0$ distintos de cero.

2.3.1.2. Otra forma de analizar las posiciones de dos rectas.

Hay dos formas de calcular las posiciones relativas de dos rectas, la primera ya la hemos visto en el apartado **2.2.2.** que se basa en los puntos y vectores que definen la recta. La otra forma se deduce del análisis de las ecuaciones de la recta.

Proposición: Posiciones respectivas de dos rectas en R^3.Tenemos un sistema de cuatro ecuaciones y tres incógnitas:

$$r \equiv \begin{cases} A_1 x + B_1 y + C_1 z + D_1 = 0 \\ A_2 x + B_2 y + C_2 z + D_2 = 0 \end{cases}$$

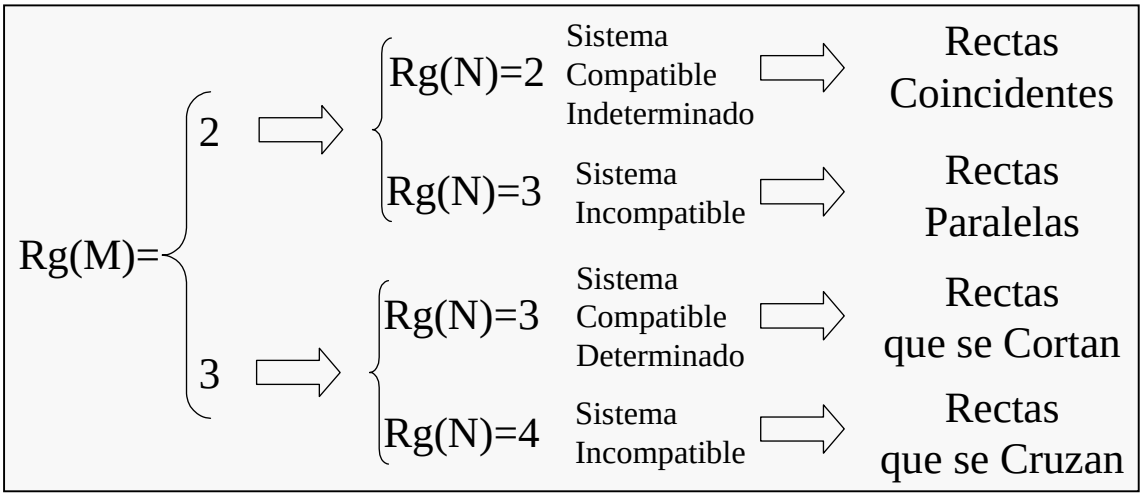
$$s \equiv \begin{cases} A_3 x + B_3 y + C_3 z + D_3 = 0 \\ A_4 x + B_4 y + C_4 z + D_4 = 0 \end{cases}$$

con las restricciones: $rg\left(\begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \end{bmatrix}\right) = rg\left(\begin{bmatrix} A_3 & B_3 & C_3 \\ A_4 & B_4 & C_4 \end{bmatrix}\right) = 2$

Las matrices a considerar son:

$$M = \begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \\ A_4 & B_4 & C_4 \end{bmatrix} \qquad N = \begin{bmatrix} A_1 & B_1 & C_1 & D_1 \\ A_2 & B_2 & C_2 & D_2 \\ A_3 & B_3 & C_3 & D_3 \\ A_4 & B_4 & C_4 & D_4 \end{bmatrix}$$

Posiciones de dos rectas:



Visto esto, pasamos a ver ahora los tres pasos principales para la construcción del modelo alámbrico: recorrido del árbol de decisión, eliminación de aristas redundantes y eliminación de aristas patológicas.

2.3.2. RECORRIDO DEL ÁRBOL DE DECISIÓN.

Antes de pasar a ver el árbol de decisión, tenemos que realizar una clasificación por casos de las aristas 2D de las listas *a_list_planta*, *a_list_alzado,* y *a_list_perfil.*

- **Caso 1:** Paralelas al eje X.
- **Caso 2:** Paralelas al eje Z.
- **Caso 3:** No paralelas a ningún eje de coordenadas.
- **Caso 4:** Punto - foco formado por la proyección.
- **Caso 5:** Paralelas al eje Y.

En las siguientes figuras se muestra las diferentes posiciones que pueden adoptar las aristas en las proyecciones de planta, alzado y perfil.

Figura 1

En esta figura tenemos que las proyecciones de planta y alzado de una arista 3D pertenecen al caso 1, paralelas al eje X, y como siempre que hay dos proyecciones paralelas al mismo eje, la proyección que falta (perfil) pertenece al caso 4, punto - foco.

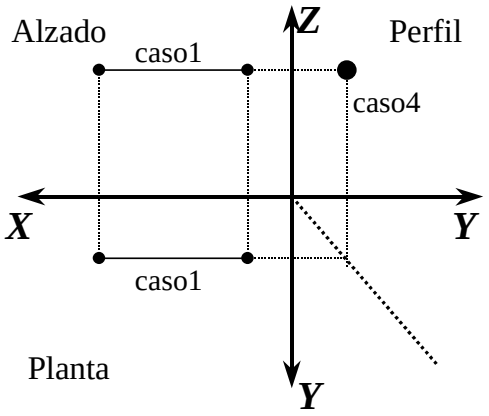


Figura 2

En esta figura tenemos que la proyección de alzado de la arista 3D pertenece al caso 1, paralela al eje X, y la proyección de perfil pertenece al caso 5, paralela al eje Y, y como siempre que hay dos proyecciones paralelas a distintos ejes, la proyección que falta (planta) pertenece al caso 3, arista no paralela a ningún eje.

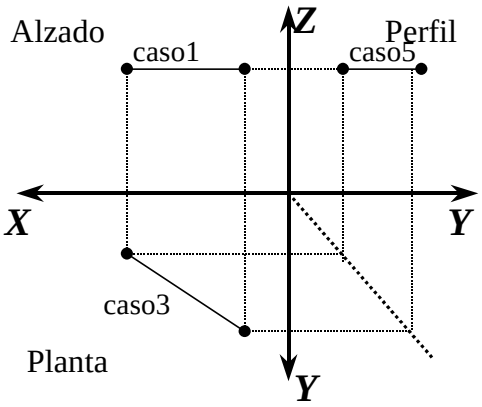
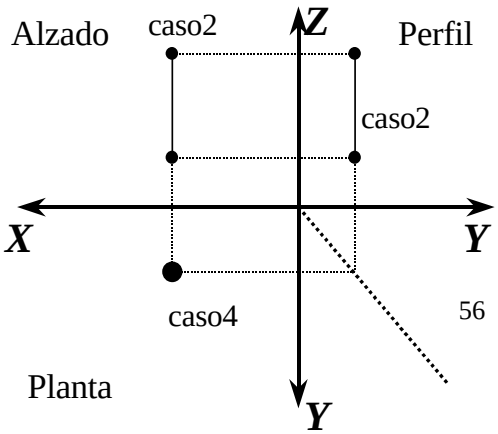


Figura 3

En esta figura tenemos que las proyecciones de perfil y alzado de una arista 3D



pertenecen al caso 2, paralelas al eje Z, y como siempre que hay dos proyecciones paralelas al mismo eje, la proyección que falta (planta) pertenece al caso 4, punto - foco.

Figura 4

En esta figura tenemos que la proyección de alzado de la arista 3D pertenece al caso 2, paralela al eje Z, y la proyección de planta pertenece al caso 5, paralela al eje Y, y como siempre que hay dos proyecciones paralelas a distintos ejes, la proyección que falta (perfil) pertenece al caso 3, arista no paralela a ningún eje.

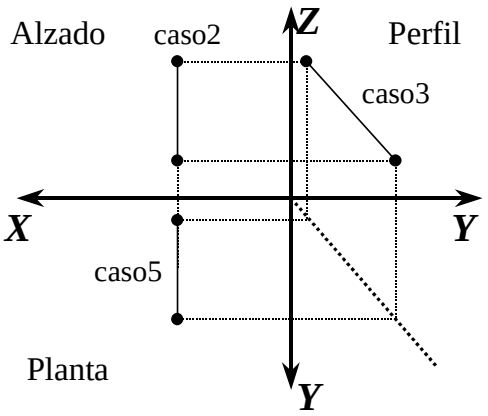


Figura 5

En esta figura tenemos que la proyección de planta de la arista 3D pertenece al caso 1, paralela al eje X, y la proyección de perfil pertenece al caso 2, paralela al eje Z, y como siempre que hay dos proyecciones paralelas a distintos ejes, la proyección que falta (alzado) pertenece al caso 3, arista no paralela a ningún eje.

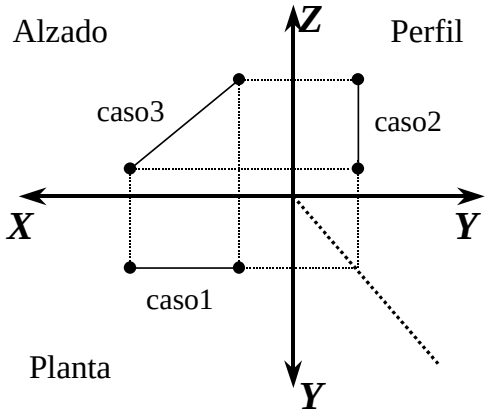
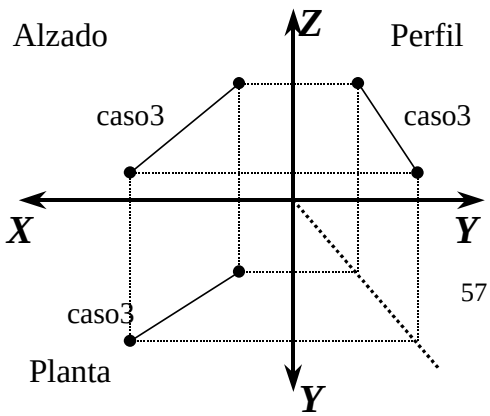


Figura 6

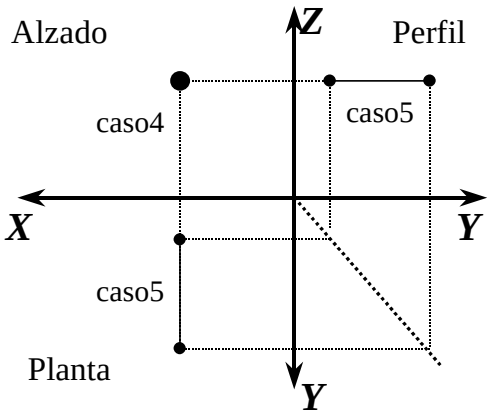
En esta figura tenemos que las tres proyecciones (planta, alzado y perfil) de la



arista 3D pertenecen al caso 3, lo que quiere decir que la arista 3D no será paralela a ningún eje de coordenadas.

Figura 7

En esta figura tenemos que las proyecciones de perfil y planta de una arista 3D pertenecen al caso 5, paralelas al eje Y, y como siempre que hay dos proyecciones paralelas al mismo eje, la proyección que falta (alzado) pertenece al caso 4, punto - foco.

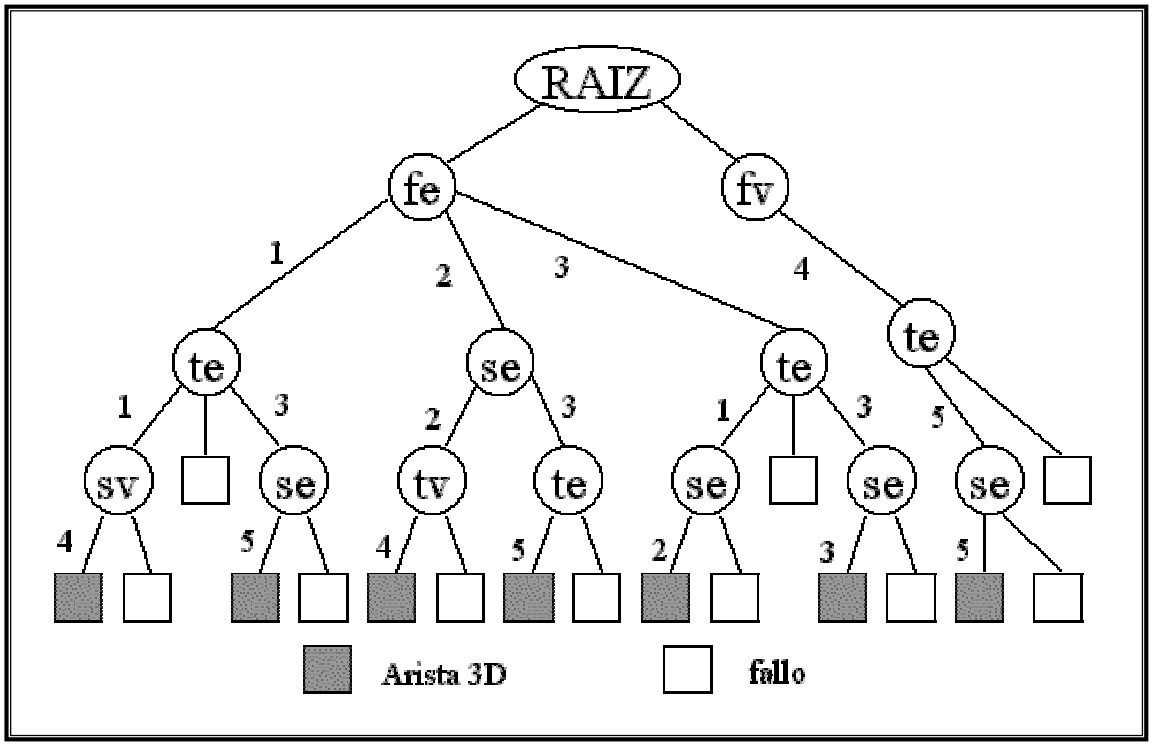


A partir de los casos anteriormente expuestos, se ha generado un árbol de decisión, que al ser recorrido por el algoritmo se obtienen una serie de aristas 3D, que en su conjunto forman el modelo alámbrico del objeto.

A continuación se listan una serie de símbolos utilizados en el árbol de decisión:

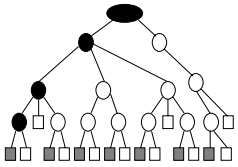
- **fe** : arista seleccionada de la lista *a_list_alzado.*
- **te** : arista seleccionada de la lista *a_list_planta.*
- **se** : arista seleccionada de la lista *a_list_perfil.*
- **fv** : vértice seleccionado de la lista *v_list_alzado.*
- **tv** : vértice seleccionado de la lista *v_list_planta.*
- **sv** : vértice seleccionado de la lista *v_list_perfil.*

En la figura siguiente tenemos en árbol de decisión que a continuación pasamos a explicar:

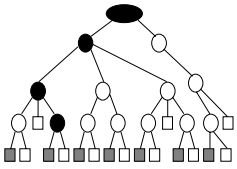


2.3.2.1. RECORRIDO DEL ÁRBOL DE DECISIÓN PARA GENERAR ARISTAS Y VÉRTICES 3D.

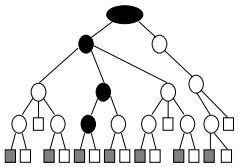
Como hemos explicado anteriormente el árbol de decisión está basado en la proyección de planta, por lo tanto, primero seleccionaremos una arista *fe* que a continuación pasamos a comprobar si es del caso 1, 2 ó 3. A continuación, seleccionamos una arista *te* para los casos 1 ó 3, o seleccionamos una arista *se* para el caso 2. Así continuaremos trazando el árbol hasta llegar a las hojas para ver si se genera la arista o no. Finalmente, seleccionamos un vértice *fv* y se traza el árbol hasta el nodo *te*. Entonces seleccionaremos una arista *te* del caso 5, si tiene éxito, para alcanzar el nodo *se*. En el nodo *se*, intentamos seleccionar una arista *se* del caso 5 para obtener una arista 3D. Si esto falla, se recorre el árbol hacia atrás para seleccionar una nueva arista *te*. Cuando alcanzamos los nodos hoja, creamos una lista VList3D, que consiste en los vértices no duplicados de cada arista 3D generada. Similarmente, creamos una lista AList3D que consiste en las aristas 3D generadas no duplicadas.



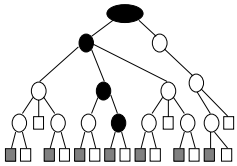
Con esta rama del árbol de decisión, se obtienen todas las aristas 3D cuyas proyecciones de alzado y planta pertenecen al caso 1 y la proyección de perfil pertenece al caso 4, representado en la *figura 1*.



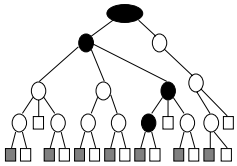
Con esta rama del árbol de decisión, se obtienen todas las aristas 3D cuya proyección de alzado pertenece al caso 1, la proyección de planta pertenece al caso 3 y la proyección de perfil pertenece al caso 5, representado en la *figura 2*.



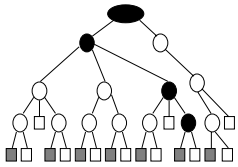
Con esta rama del árbol de decisión, se obtienen todas las aristas 3D cuyas proyecciones de alzado y perfil pertenecen al caso 2 y la proyección de planta pertenece al caso 4, representado en la *figura 3*.



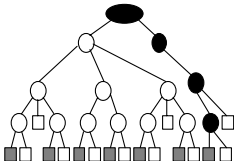
Con esta rama del árbol de decisión, se obtienen todas las aristas 3D cuya proyección de alzado pertenece al caso 2, la proyección de perfil pertenece al caso 3 y la proyección de planta pertenece al caso 5, representado en la *figura 4*.



Con esta rama del árbol de decisión, se obtienen todas las aristas 3D cuya proyección de alzado pertenece al caso 3, la proyección de planta pertenece al caso 1 y la proyección de perfil pertenece al caso 2, representado en la *figura 5*.



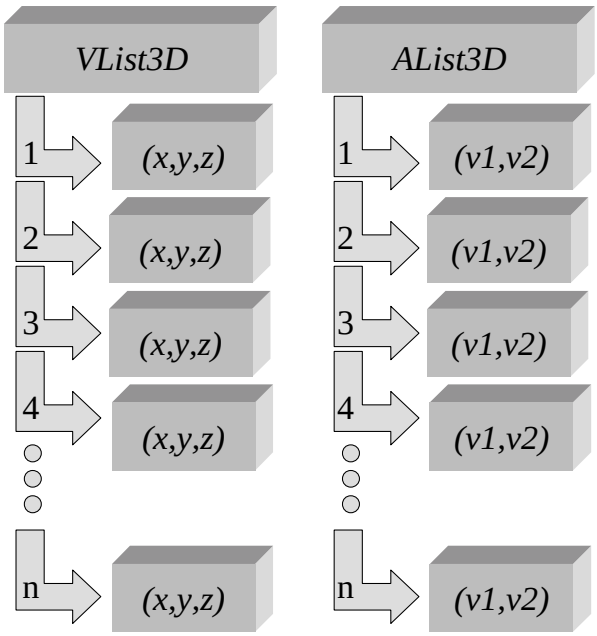
Con esta rama del árbol de decisión, se obtienen todas las aristas 3D cuyas proyecciones de planta, alzado y perfil pertenecen al caso 3, representado en la *figura 6*.



Con esta rama del árbol de decisión, se obtienen todas las aristas 3D cuya proyección de alzado pertenece al caso 4, y las proyecciones de planta y perfil pertenecen al caso 5, representado en la *figura 7*.

Las listas de vértices y aristas 3D, VList3D y AList3D respectivamente, contienen todas las posibles aristas y vértices 3D que se pueden generar recorriendo árbol de decisión.

Aquí podemos ver un esquema de la estructura de estas listas:

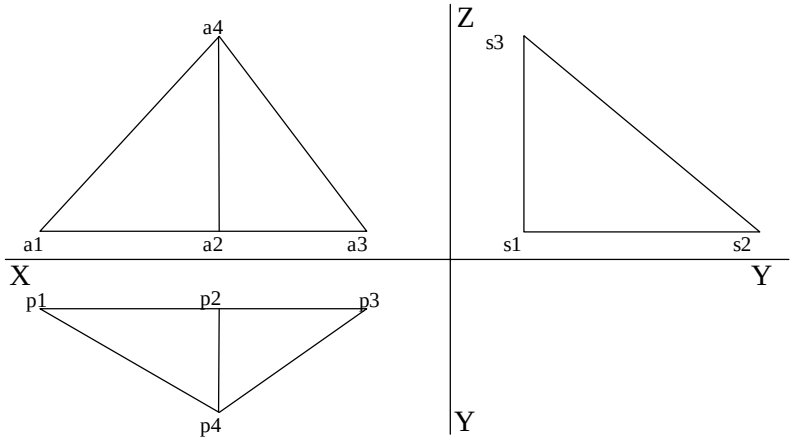


Cada entrada de la lista de vértices 3D, almacena las tres coordenadas x, y, z de los vértices generados, pudiendo hacer referencia a cada vértice mediante el índice de la lista. En la lista de aristas, en cada entrada se almacena dos índices, v1, v2, que hacen referencia a los dos vértices de la lista VList3D que forman la arista.

Algunas de estas aristas 3D pueden solaparse parcialmente, y deben ser tratadas por el algoritmo de eliminación de aristas redundantes, que es el paso siguiente.

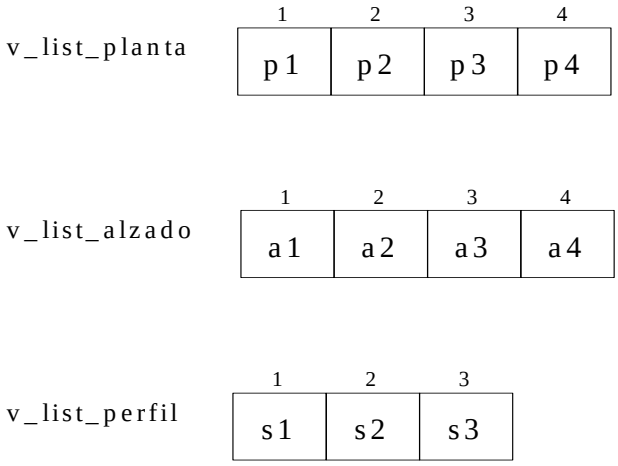
2.3.2.2. EJEMPLO SENCILLO DE RECORRIDO DEL ÁRBOL DE DECISIÓN.

En la figura siguiente podemos observar las proyecciones de planta alzado y perfil de un objeto sencillo:

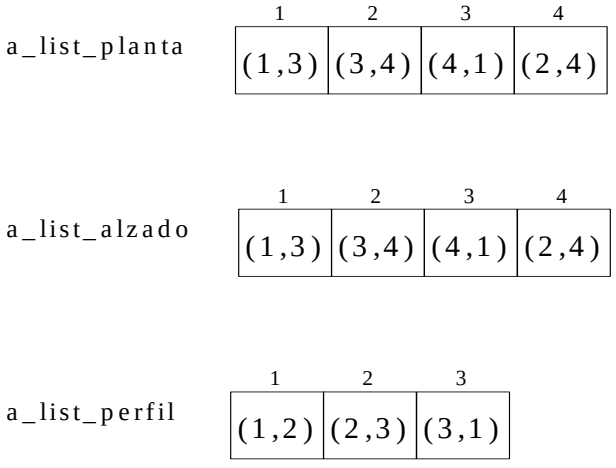


Suponemos que los vértices 2D de cada proyección en la figura anterior son, en realidad, las coordenadas de cada proyección. Así un vértice p_i de la proyección planta equivale a (x_i, y_i), un vértice a_j de la proyección alzado equivale a (x_j, z_j) y un vértice s_k de la proyección perfil equivale a (y_k, z_k).

Por lo tanto, tenemos las listas de vértices de las proyecciones siguientes:



Y las listas de aristas de las proyecciones son las siguientes:



El contenido de las listas de aristas son los índices de la lista de vértices que forman una arista 2D en cada proyección. Así, por ejemplo, la arista 1 (la arista que tiene índice de valor 1) de la proyección de planta la forman los vértices 1 y 3 (los vértices que tienen índice de valor 1 y 3) de la lista de vértices de la proyección de planta, que son los vértices p1 y p3. Por lo tanto, la arista 1 está formada por los vértices p1 y p3.

Primero aplicaríamos el algoritmo de preprocesado de los datos de entrada, que en este ejemplo generaría aristas nuevas en la planta y el alzado, ya que hay tres vértices unidos mediante una arista. En el caso de la planta, la arista 1 une los vértices p1, p2 y p3, y en el alzado, la arista 1 une los vértices a1, a2 y a3. Como ya tenemos la arista que une los vértices p1 y p3 en la planta, a partir de ahora la expresaremos como la arista

(p1, p3), este algoritmo generaría dos aristas más, las aristas (p1, p2) y (p2, p3), y en el alzado se generarían las aristas (a1, a2) y (a2, a3).

Las listas de aristas después del preprocesado de datos quedarían de la siguiente forma:

a_list_planta

1	2	3	4	5	6
(1,3)	(3,4)	(4,1)	(2,4)	(1,2)	(2,3)

a_list_alzado

1	2	3	4	5	6
(1,3)	(3,4)	(4,1)	(2,4)	(1,2)	(2,3)

a_list_perfil

1	2	3
(1,2)	(2,3)	(3,1)

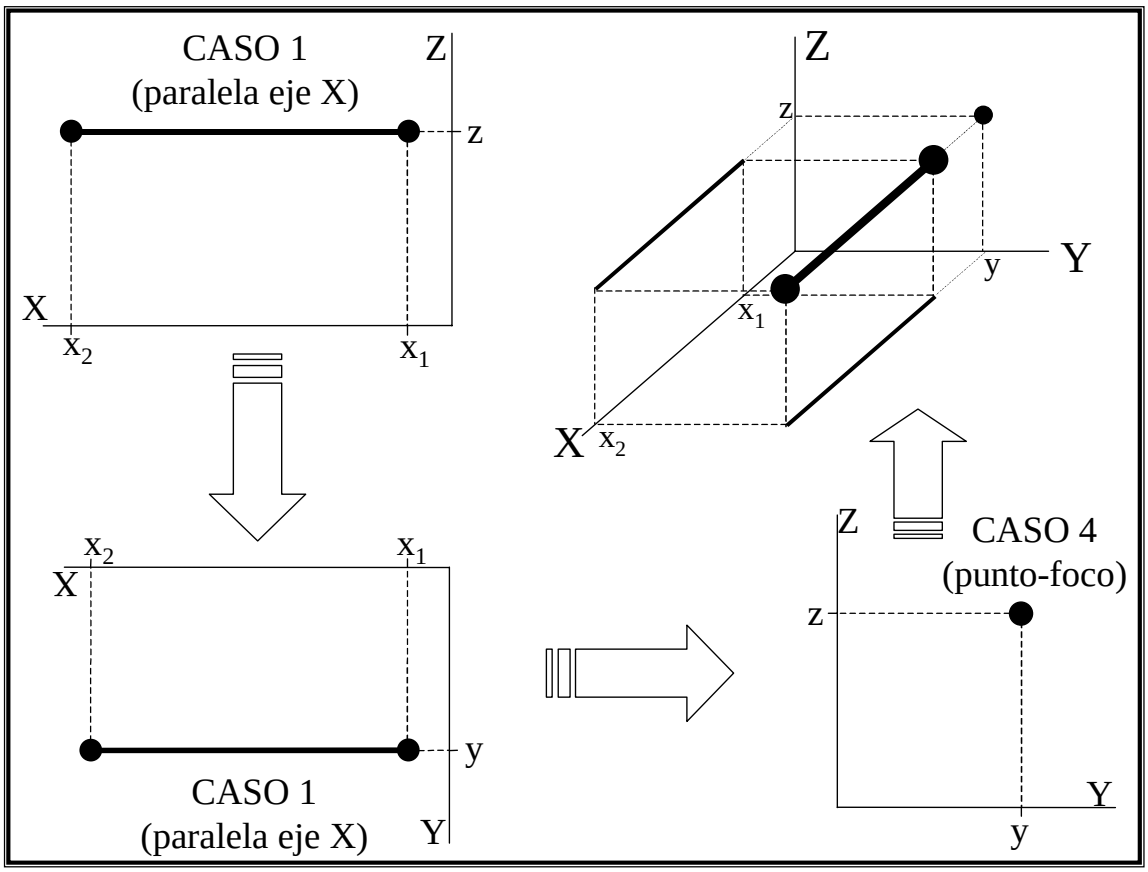
Las nuevas aristas generadas son las aristas 5 y 6 de la planta y el alzado. A partir de este momento se pasa a recorrer el árbol de decisión para obtener el modelo alámbrico del objeto.

El recorrido del árbol se inicia seleccionando las aristas de la lista de aristas del alzado, por lo tanto, seleccionamos la arista 1 de *a_list_alzado*, nodo *fe* del árbol de decisión. Como esta arista es paralela al eje X (caso 1), pasamos al nodo *te* del árbol de decisión. Ahora vamos a buscar dentro de la lista de aristas de la planta una arista que sea paralela al eje X (caso 1) o no paralela a ningún eje (caso 3). Esta arista que buscamos también tiene que cumplir que la coordenada X de los vértices que son extremos de la arista del alzado y de la arista de la planta sean iguales. Como sólo hay una arista en la planta que cumple las condiciones anteriores, seleccionamos la arista 1 de *a_list_planta*, que es paralela al eje X. Debido a esto, pasamos al nodo *sv*, pasando a buscar un vértice de la lista de vértices del perfil que sea punto – foco (caso 4). El punto – foco del perfil cumplirá que su coordenada Z será la misma que la de los dos vértices de la arista del alzado seleccionada; y su coordenada Y será la misma que la de los dos vértices de la arista de la planta seleccionada. Siguiendo esto, obtenemos el vértice 1 del perfil (s1) que es un punto – foco de la arista 1 del alzado y la arista 1 de la planta.

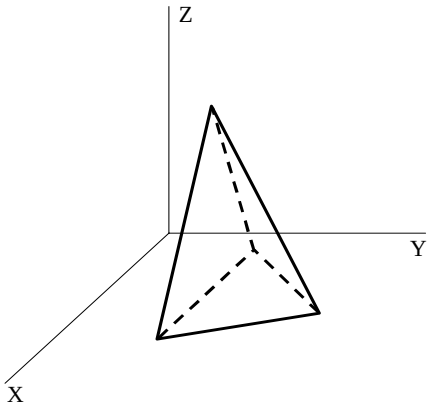
Ahora tenemos que formar dos vértices 3D y una arista 3D con los datos seleccionados anteriormente. Al tener un punto – foco se puede decir que sus dos coordenadas son fijas para toda la arista, por lo tanto si llamamos (S_Y, S_Z) a las coordenadas del punto – foco, tenemos dos coordenadas de los dos vértices 3D que forman la arista 3D. La coordenada X se obtiene de los datos de la planta o del alzado. Tenemos que los dos vértices del alzado (S_{X1}, S_Z) y (S_{X2}, S_Z) al ser la arista paralela al eje X y los dos vértices de la planta (S_{X1}, S_Y) y (S_{X2}, S_Y) al ser la arista paralela al eje X. Ya podemos obtener la arista 3D a partir de sus vértices 3D, que serán (S_{X1}, S_Y, S_Z) y (S_{X2}, S_Y, S_Z). Si no existen estos vértices en la lista de vértices 3D, *VList3D*, se insertan

y, con los índices de estos vértices dentro de la lista, si no existe esta arista (I_1, I_2) en la lista de aristas 3D, *AList3D*, entonces también se inserta.

En la figura siguiente se muestra gráficamente los pasos anteriores:



Para el resto de las aristas 2D se actuaría un modo similar dependiendo de la rama del árbol de decisión que se esté accediendo en cada momento. Cuando acaba este algoritmo de recorrido del árbol se obtiene el modelo alámbrico siguiente, a falta de la eliminación de aristas redundantes y patológicas, que en este ejemplo no supondría modificación alguna.



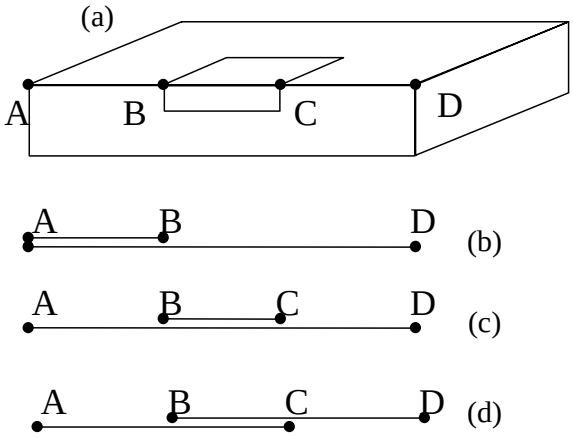
Después de la aplicación del algoritmo de recorrido del árbol de decisión, tenemos que las listas de vértices y aristas, *VList3D* y *AList3D*, contienen todas las

posibles aristas y vértices 3D. Algunas de estas aristas pueden solaparse parcialmente y deben ser tratadas en el paso siguiente.

2.3.3. ELIMINACIÓN DE ARISTAS REDUNDANTES.

A este procedimiento de eliminación de aristas redundantes lo vamos a llamar a partir de ahora *RER* que proviene de ***R****edundant* ***E****dges* ***R****emoval*. Este procedimiento se fundamenta en que las aristas solapadas son redundantes, no sólo incrementan la complejidad de la computación, sino que además, también introduce ambigüedades en el proceso de generación de *bucles de caras*[♣].

En la figura siguiente se muestra tres casos de solapamiento. Para el modelo alámbrico de la figura (a), los vértices A, B, C, D están en la lista *VList3D*. En la lista *AList3D*, la arista AB solapa con la AD, figura (b), o la arista BC solapa con la AD, figura (c), o la arista AC solapa con la BD, figura (d). Las partes solapadas de las aristas se consideran como '*aristas redundantes*' y son eliminadas por el procedimiento *RER*.



2.3.3.1. ALGORITMO APLICADO.

Partiendo de los datos generados por el procedimiento anterior, esto es, las listas de vértices y aristas *VList3D* y *AList3D*, hacemos lo siguiente:

1. Inicialmente se marcan todas las aristas en *AList3D* como 'no examinadas' y se inicializa el conjunto E=∅ (E será un conjunto en el que tendremos las aristas solapadas).
2. Seleccionar una arista 'no examinada' e_i de *AList3D*, y hacer $E \leftarrow \{e_i\}$. Si todas las aristas han sido examinadas, el procedimiento *RER* acaba.
3. Para cada arista $e_j \in$ *AList3D*, $e_j \neq e_i$, si e_j y e_i son colineales y e_j solapa con una arista de E, entonces $E \leftarrow E \cup \{e_j\}$.
4. Si sólo hay una arista en E, entonces ninguna otra arista solapa con la arista de E. Se marca esta arista como 'examinada', y pasar al paso 2, en otro caso continuar.
5. Si hay más de una arista en E, entonces eliminar estas aristas de *AList3D* y ordenar todos los vértices de acuerdo con sus valores de coordenadas. Después tenemos que

[♣] Conjunto de aristas en bucle cerrado que forman una cara del objeto en 3D, en inglés *face loops*.

v_1, v_2, ... , v_K serán una secuencia de vértices ordenados. Añadir las aristas (v_j, v_{j+1}), j = 1, ... , K-1 a *AList3D*, y marcarlas como 'examinadas'. Ir al paso 2.

Usando este procedimiento, las aristas en *AList3D* son todas únicas y no solapadas. Después del procedimiento *RER*, por ejemplo, sólo las aristas AB y BD pertenecen a *AList3D* para la figura anterior (b), y sólo las aristas AB, BC, CD para las figuras (c) y (d). Aunque se han eliminado las aristas solapadas redundantes, algunas aristas o vértices patológicos, tales como aristas colgantes, vértices y aristas aisladas, pueden existir y deben ser eliminadas. El procedimiento de eliminación de vértices y aristas patológicas se utiliza para eliminar estas aristas o vértices. Este procedimiento se va a ver en el paso siguiente.

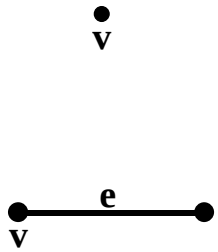
2.3.4. ELIMINACIÓN DE ARISTAS PATOLÓGICAS.

A este procedimiento de eliminación de vértices y aristas patológicas lo vamos a llamar a partir de ahora *PEVR* que viene de ***P**athological **E**dges/**V**ertices **R**emoval*. Este procedimiento se basa en que, al generar el modelo alámbrico de un objeto, recorriendo el árbol de decisión, se pueden generar una serie de aristas "*patológicas*" que no tienen sentido y que no forman parte del objeto 3D. Por lo tanto se aplica este procedimiento para la eliminación de algunas (no todas) de estas aristas patológicas, incidiendo esta eliminación de aristas en una reducción de la complejidad de la computación.

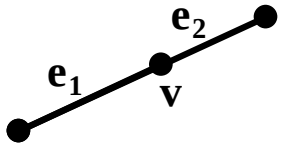
Vamos a definir la ***valencia*** de un vértice, como el número de aristas que inciden en sobre ese vértice. También lo representaremos como la función ρ(*vertex*), que nos proporciona el número de aristas compartidas por el vértice *vertex*. A continuación se van a describir los casos en los que hay vértices y aristas patológicas, dependiendo de la valencia de los vértices del modelo alámbrico generado.

2.3.4.1. DESCRIPCIÓN DE LOS CASOS POSIBLES.

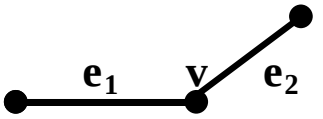
Los casos posibles en los que podemos encontrar aristas y vértices patológicos son los siguientes:



Los dos primeros casos son los que se muestran en la figura de la izquierda. El primero se trata de un vértice aislado, que como indica su nombre, no hay ninguna arista que pase por él, por lo tanto su valencia es 0, y debe de ser eliminado de la lista de vértices. El otro caso es el de una arista colgante, que sería una única arista *e* incidiendo sobre el vértice *v*. Este vértice tendría valencia 1 y sería eliminado, al igual que la arista *e*, porque no es lógico que un vértice 3D tenga conectada sólo una arista.

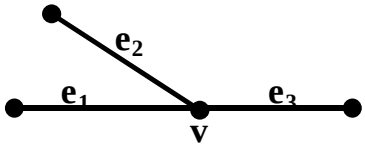


En la figura de la derecha se muestra dos casos más, ambos con la misma característica, la valencia del vértice *v* es 2. En el primer caso tenemos que las dos únicas aristas

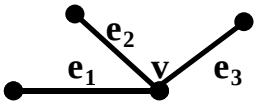


que une el vértice *v* son colineales, o lo que es lo mismo, que las dos aristas están sobre la misma recta. En tal caso, lo que se hace es generar una nueva arista, que resulta de la fusión de las aristas e_1 y e_2, y eliminar *v* de la lista de vértices, y e_1, e_2 de la lista de aristas.

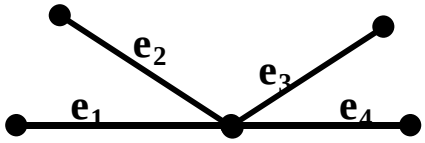
En el siguiente caso, las dos aristas no son colineales, por lo tanto no tienen sentido en 3D y deben ser eliminados *v*, e_1 y e_2 de sus respectivas listas.



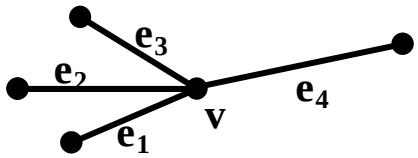
Los siguientes dos casos que se muestran en la figura de la izquierda, se caracterizan en que la valencia del vértice *v* que estudiamos es 3. El primero de los dos, además, se caracteriza por tener dos aristas colineales, que son e_1 y e_3, lo que no tiene mucho sentido en 3D. Lo que hay que hacer es eliminar el vértice *v* y la arista e_2 que son patológicos, y fusionar las dos aristas e_1 y e_3.



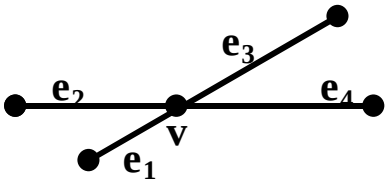
Otro caso que no tiene mucho sentido en 3D es el siguiente. Aquí tenemos a tres aristas que son no colineales y coplanares, esto es, que pertenecen al mismo plano. Por lo tanto se elimina el vértice y las tres aristas.



Los últimos tres casos se caracterizan por la valencia del vértice *v* que es 4, y los podemos ver en la figura de la derecha. Además para todos los casos las cuatro aristas son coplanares. En el primer caso tenemos que el vértice *v* y las dos aristas que no son colineales, e_2 y e_3, son patológicas, por lo tanto deben ser eliminados de sus respectivas listas, mientras que las aristas e_1 y e_4 se fusionan.



En los dos siguientes casos tenemos que ningún par de aristas son colineales, en el primero, y en el segundo tenemos que hay dos pares de aristas colineales. En estos dos últimos casos todas las aristas son patológicas, además del vértice *v*, por lo tanto deben ser eliminados de la lista de aristas y de la lista de vértices, respectivamente.



Después de ver los posibles casos de aristas y vértices patológicos vamos a pasar a ver de forma esquemática el algoritmo del procedimiento *PEVR*.

2.3.4.2. ALGORITMO APLICADO.

Dadas las listas de vértices y aristas 3D, *VList3D* y *AList3D* respectivamente, este algoritmo elimina los vértices y aristas patológicas que encuentra. En este procedimiento se va a utilizar la función ρ que aplicada sobre un vértice del modelo alámbrico, obtiene la valencia de este vértice. El algoritmo es el siguiente:

1. Seleccionar un vértice *v*, mientras haya vértices no analizados. Si se han analizado todos los vértices, pasar al paso 4.
2. Calcular la valencia de este vértice, ρ(*v*):
  2.1. Si la valencia es cero, ρ(*v*)=0, es un vértice aislado que hay que eliminar.
  2.2. Si la valencia es uno, ρ(*v*)=1, se trata de una arista colgante. Hay que eliminar la arista y el vértice *v*.
  2.3. Si la valencia es dos, ρ(*v*)=2:
    2.3.1. Si las dos aristas son colineales entonces fusionarlas y eliminar *v*.
    2.3.2. Si las dos aristas son no colineales entonces eliminar las aristas y *v*.
  2.4. Si la valencia es tres, ρ(*v*)=3:
    2.4.1. Si dos de las tres aristas son colineales entonces eliminar la arista no colineal y *v*, y fusionar las aristas colineales.
    2.4.2. Si las tres aristas son coplanares y no hay dos colineales entonces eliminar las aristas y *v*.
  2.5. Si la valencia es cuatro o más, ρ(*v*)≥4, las cuatro aristas coplanares:
    2.5.1. Si sólo hay dos aristas colineales entonces fusionarlas y eliminar el resto de aristas y *v*.
    2.5.2. Si dos pares de aristas son colineales entonces eliminar las aristas y *v*.
    2.5.3. Si todas las aristas son no colineales entonces eliminar las aristas y *v*.
3. Ir al paso 1.
4. Si se han encontrado vértices o aristas patológicas que se hayan eliminado, volver a aplicar el algoritmo, o lo que es lo mismo, volver al paso 1 en el estado inicial.

Es esencial mencionar que el procedimiento *PEVR* elimina las aristas y vértice patológicos generados por el algoritmo de recorrido del árbol de decisión. Sin embargo, algunas aristas patológicas pueden ser descubiertas en los procedimientos posteriores que se van a describir. El procedimiento *PEVR* se aplicará otra vez, si se encuentran aristas y vértices patológicos, hasta que no haya más. El procedimiento *PEVR* alcanza finalmente un estado estable de las listas *VList3D* y *AList3D*.

Hemos establecido el modelo alámbrico de un objeto con las listas *VList3D* y *AList3D* que contienen información acerca de los vértices y aristas 3D. Después de haber obtenido el modelo alámbrico, necesitamos construir los planos o gráficos planos♦ para conseguir la información de los bucles de caras.

2.4. GENERACIÓN DE PLANOS.

En el punto anterior se ha visto como el algoritmo *WC* construye el modelo alámbrico de los datos de entrada (vértices y aristas 2D). El modelo alámbrico no da explícitamente información sobre la superficie de los objetos. Necesitamos construir los

♦ Conjuntos de aristas del modelo alámbrico que están en el mismo plano, del inglés *planar graphs.*

gráficos planos (del inglés *planar graphs*) a partir del modelo alámbrico para generar la información de la superficie de los objetos. Un gráfico plano es una colección de aristas coplanares, es decir, que pertenecen al mismo plano.

En esta sección se describe el algoritmo de construcción de los gráficos planos (*Planar-Graph* Generation, *PGG*). Este algoritmo consta de varios pasos, como pueden ser: *búsqueda de aristas adyacentes a un vértice*, *determinación de planos*, *cálculo del vector normal al plano*, *eliminación de planos duplicados*, *búsqueda de aristas pertenecientes a un plano* y *verificación de los gráficos planos*.

2.4.1. NOCIONES DE GEOMETRÍA ANALÍTICA.

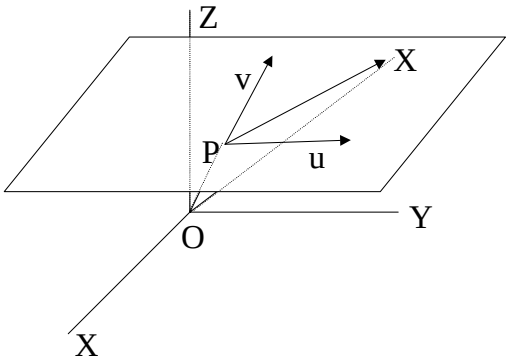
Se va a realizar una serie de definiciones y proposiciones de la geometría analítica que se van a utilizar en los puntos siguientes.

2.4.1.1. Ecuación de un plano en R^3.

Para determinar un plano en el espacio físico debemos conocer alternativamente:
a) Un punto *P* y dos vectores *u* y *v*, no paralelos.
b) Dos puntos *P* y *Q* y un vector *v*, siendo el vector [*PQ*] y *v* no paralelos.
c) Tres puntos no alineados *P*, *Q* y *R*.

Basta con resolver el caso a), ya que en los casos b) y c) podemos tomar:
b) *P*, *u*=[*PQ*], *v*.
c) *P*, *u*=[*PQ*], *v*=[*PR*].
(u otras elecciones similares).

Para que un punto cualquiera *X* de R^3 esté en el plano, debe cumplirse que $[PX]=\alpha\boldsymbol{u}+\beta\boldsymbol{v}$, siendo α y β escalares.



$P(x_0, y_0, z_0)$
$X(x, y, z)$
$u = (u_1, u_2, u_3)$
$v = (v_1, v_2, v_3)$
$PX=(x-x_0, y-y_0, z-z_0)$

De la figura anterior se deduce:

- Ecuación vectorial: $(x-x_0, y-y_0, z-z_0) = \alpha(u_1, u_2, u_3)+\beta(v_1, v_2, v_3)$

- Ecuaciones paramétricas:

$$x = x_0 + \alpha u_1 + \beta v_1$$
$$y = y_0 + \alpha u_2 + \beta v_2$$
$$z = z_0 + \alpha u_3 + \beta v_3$$

Una forma cómoda de eliminar los parámetros del sistema de ecuaciones anterior, consiste en lo siguiente: para que existan los escalares α y β, las matrices:

$$M_1 = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{bmatrix} \qquad M_2 = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ x - x_0 & y - y_0 & z - z_0 \end{bmatrix}$$

deben tener el mismo rango. Como el rango de M_1 es dos (por ser *u* y *v* no paralelos), el determinante de M_2 debe ser cero. Desarrollando el determinante:

$$A(x - x_0) + B(y - y_0) + C(z - z_0) = 0 \qquad (*)$$

siendo

$$A = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} \quad \mathrm{B} = -\begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix} \quad \mathrm{C} = \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}$$

Estos coeficientes A, B, C, son precisamente las coordenadas del producto vectorial $u \times v$. También podemos escribir:

$$Ax + By + Cz + D = 0 \ (**), \text{donde } D = -Ax_0 - By_0 - Cz_0$$

Las ecuaciones (*) y (**) son llamadas, indistintamente, ecuación implícita del plano en el espacio físico.

2.4.1.2. Posiciones respectivas de dos planos en R^3.

Consideramos los planos π y π' de ecuaciones:

π: Ax + By + Cz + D =0
π': A'x + B'y + C'z + D' =0

Las ecuaciones anteriores constituyen un sistema de dos ecuaciones con tres incógnitas. Formemos las matrices:

$$M = \begin{bmatrix} A & B & C \\ A' & B' & C' \end{bmatrix} \qquad N = \begin{bmatrix} A & B & C & D \\ A' & B' & C' & D' \end{bmatrix}$$

Las posibles combinaciones de rangos son:

a) Rango(M)=Rango(N)=1 **sistema compatible (indeterminado)**.
Por ser Rango(M)=1, los planos son paralelos, y al tener Rango(N)=1, las ecuaciones son proporcionales; por tanto, los planos *coinciden*, como corresponde a un sistema compatible indeterminado.

b) Rango(M)=1, Rango(N)=2 **sistema incompatible**.
Los planos son paralelos pero no tienen ningún punto en común (incompatibilidad): son paralelos *estrictamente*.

c) Rango(M)=Rango(N)=2 **sistema compatible (indeterminado)**.
Los planos no son paralelos, por lo que se cortan en una recta.

2.4.1.3. Distancia entre un punto y un plano.

Tenemos los datos siguientes: un punto $P(x_0, y_0, z_0)$ y un plano π que viene definido por la ecuación (*) Ax+By+Cz+D=0. Queremos saber la distancia que hay entre ese punto y el plano siguiendo el camino más corto, que será una recta perpendicular al plano π y que pase por el punto *P*.

Tracemos la recta que pasa por *P* y es perpendicular al plano. Sus ecuaciones paramétricas son (tomando como parámetro -α sin pérdida de generalidad):

$$\begin{aligned} x &= x_0 - A\alpha \\ y &= y_0 - B\alpha \qquad (**) \\ z &= z_0 - C\alpha \end{aligned}$$

La intersección de (*) y (**) es el punto $P'(x_1, y_1, z_1)$, proyección perpendicular de *P* sobre el plano, y verifica ambas ecuaciones:

$$\left.\begin{array}{c} x_1 = x_0 - A\alpha,\ y_1 = y_0 - B\alpha,\ z_1 = z_0 - C\alpha \\ Ax_1 + By_1 + Cz_1 + D = 0 \end{array}\right\} \Rightarrow$$

$$A(x_0 - A\alpha) + B(y_0 - B\alpha) + C(z_0 - C\alpha) + D = 0 \Rightarrow$$

$$\alpha = \frac{Ax_0 + By_0 + Cz_0 + D}{A^2 + B^2 + C^2}$$

Tenemos que:

$$d(P,\pi) = d(P,P') = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2} =$$

$$= \alpha\sqrt{A^2 + B^2 + C^2} = \frac{Ax_0 + By_0 + Cz_0 + D}{\sqrt{A^2 + B^2 + C^2}}$$

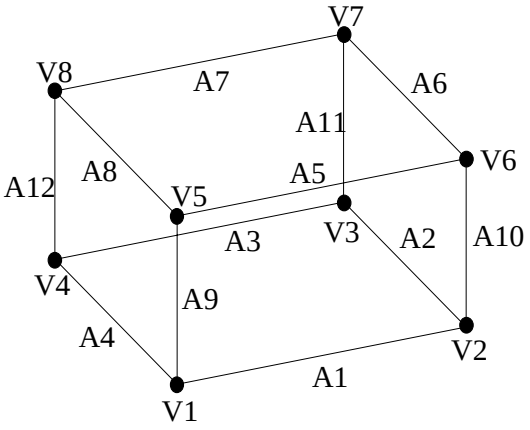
La fórmula resultante es la siguiente:

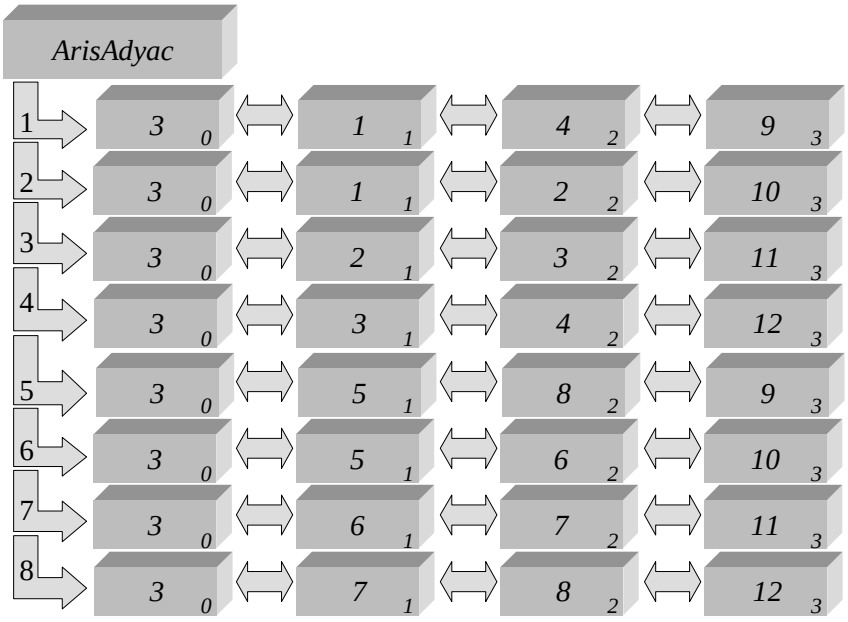
$$\boxed{d(P,\pi) = \frac{Ax_0 + By_0 + Cz_0 + D}{\sqrt{A^2 + B^2 + C^2}}}$$

2.4.2. BÚSQUEDA DE ARISTAS ADYACENTES A VÉRTICES.

Este es el primer paso a realizar dentro del algoritmo de generación de gráficos planos *PGG*. Se trata de registrar todas las aristas adyacentes para cada vértice de la lista de vértices 3D.

Se ha definido una estructura de datos de *array bidimensional*, el cual va a ser rellenado de información en este paso del algoritmo. Este *array* tendrá tantas filas como vértices tiene el modelo alámbrico, y dentro de cada fila tendremos tantas columnas como aristas adyacentes tenga el vértice indicado por el número de fila. Se ha puesto un máximo a este número de aristas adyacentes que es 10. Así por ejemplo, cuando queramos saber las aristas adyacentes al vértice 23 de la lista de vértices 3D, accederemos a la fila 23 del *array bidimensional* en la cual tendremos almacenadas: como primer dato el número de aristas adyacentes, y seguido a esto, los índices de la lista de aristas 3D que son adyacentes al vértice 23. En la figura siguiente, se puede ver un ejemplo de almacenamiento de los datos para un objeto 3D.





Como se puede ver en la figura anterior, las aristas adyacentes al vértice V1 nos las proporciona la primera fila de la estructura de datos. Así también, las aristas adyacentes al vértice V5 nos las proporciona la quinta fila. Dentro de cada fila, el

primer elemento, con índice 0, nos dice siempre el número de aristas que hay en su fila y los índices siguientes contienen índices de la lista de aristas 3D, por lo tanto nos proporciona las aristas adyacentes. Resumiendo, indicaremos una celda del *array* anterior de la forma siguiente: *ArisAdyac[i][j]*; por lo tanto diremos que *ArisAdyac[i]* serán todas las aristas adyacentes al vértice *i*; también diremos que el número de aristas adyacentes al vértice *i* serán *n=ArisAdyac[i][0]*; y las aristas adyacentes al vértice *i* vienen dadas por *ArisAdyac[i][1], ... , ArisAdyac[i][n]*.

El algoritmo que registra toda la información de las aristas adyacentes a un vértice, o lo que es lo mismo, rellena la estructura de datos anterior, es muy simple y se ve a continuación:

Para cada vértice de la lista de vértices 3D hacer:
- Recorrer la lista de aristas 3D.
- Cuando una arista 3D tenga el vértice seleccionado en cada momento:
  - Guardar el índice de la arista en la estructura *ArisAdyac*.
  - Incrementar el número de aristas adyacentes de dicha fila de *ArisAdyac*.

2.4.3. CONSTRUCCIÓN DE PLANOS Y OBTENCIÓN DE LA NORMAL.

Estos son el segundo y tercer paso a realizar dentro del algoritmo de generación de gráficos planos *PGG*. Se basa en que un plano puede ser determinado por dos aristas adyacentes no colineales (no pertenecen a la misma recta) en un vértice. Por lo tanto se trata de construir todos los planos que se puedan generar en cada vértice tomando sus aristas adyacentes de dos en dos. El plano construido vendrá definido por su propia ecuación del plano (Ax+By+Cz+D=0) que utilizaremos para realizar una hipótesis de cual es el vector normal[♣] exterior del plano, ya que el vector normal exterior del plano no puede ser determinado hasta que sepamos cuál es la cara exterior del plano.

Se ha definido una lista de planos como la estructura de datos que contendrá toda la información generada sobre los planos. Para cada plano, almacenaremos los coeficientes de la ecuación del plano (A, B, C, D), el hipotético vector normal al plano *N* que viene definido por tres coordenadas (n_x , n_y , n_z), y el conjunto de aristas que están incluidas en ese plano. Esta estructura de datos viene limitada a un número máximo de planos que se pueden almacenar, este número máximo es de 100 planos.

Antes de pasar a explicar el algoritmo utilizado en este paso hay que explicar dos acciones que se realizan en este punto. La primera es la comprobación de la colinealidad de dos aristas, y se utiliza una función (rutina) implementada para el punto **2.3.3. ELIMINACIÓN DE ARISTAS REDUNDANTES**, y que además se explica sus fundamentos en el punto **2.3.1.2.** . La segunda acción a explicar es la obtención de la ecuación del plano, esto es, la obtención de los coeficientes de la ecuación a partir de dos aristas con un vértice común, y que además son no colineales.

Obtención de los coeficientes de la ecuación del plano.

[♣] El hipotético vector normal exterior del plano es un vector perpendicular al plano, y lo llamaremos *N*.

Partimos de que tenemos dos aristas con un punto en común (vértice) y que además son no colineales, por lo tanto estas dos aristas pueden definirnos un plano y queremos saber los valores de los coeficientes de la ecuación del plano que forman. Por lo tanto sabemos:

- Arista A, definida por los vértices: v1 (v1.x, v1.y, v1.z) y v2 (v2.x, v2.y, v2.z).
- Arista B, definida por los vértices: v1 (v1.x, v1.y, v1.z) y v3 (v3.x, v3.y, v3.z).
- Vértice común: v1 (v1.x, v1.y, v1.z).

Según lo visto en el punto **2.4.1.1.** para obtener la ecuación de un plano necesitamos un punto y dos vectores no paralelos, por lo tanto obtendremos los vectores de las aristas de la forma siguiente:

- El vector *u*: (u_1, u_2, u_3)=(v2.x-v1.x, v2.y-v1.y, v2.z-v1.z).
- El vector *v*: (v_1, v_2, v_3)=(v3.x-v1.x, v3.y-v1.y, v3.z-v1.z).

Podemos obtener los valores de los coeficientes utilizando:

$$A = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} \quad \mathrm{B} = -\begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix} \quad \mathrm{C} = \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}$$

$$D = -Ax_0 - By_0 - Cz_0, \text{ donde } (x_0, y_0, z_0) \text{ es el vértice común.}$$

Por lo tanto, tenemos que:

$$\begin{aligned} A &= (v2.y - v1.y)(v3.z - v1.z) - (v2.z - v1.z)(v3.y - v1.y) \\ B &= (v2.z - v1.z)(v3.x - v1.x) - (v2.x - v1.x)(v3.z - v1.z) \\ C &= (v2.x - v1.x)(v3.y - v1.y) - (v2.y - v1.y)(v3.x - v1.x) \\ D &= -A(v1.x) - B(v1.y) - C(v1.z) \end{aligned}$$

Obtención del hipotético vector normal exterior al plano.

A partir de ahora llamaremos cara positiva del plano a la cara que a partir de la cual sale el hipotético vector normal exterior al plano, y a la otra cara, la llamaremos cara negativa. Así pues, *+N* será el vector normal exterior del plano en la cara positiva. Además según el artículo de la revista *Computer-Aided Design* de Qing-Wen Yan se propone para el cálculo del vector normal la siguiente fórmula:

$$N = \begin{cases} -(A\vec{i} + B\vec{j} + C\vec{k}) & \text{si } D < 0 \\ A\vec{i} + B\vec{j} + C\vec{k} & \text{en otro caso} \end{cases}$$

donde *i, j, k* son vectores unitarios en las direcciones *X, Y, Z* respectivamente.

Algoritmo de construcción de planos.

Este algoritmo construye todos los posibles planos a partir de todas las combinaciones de dos aristas adyacentes con un vértice común, o lo que es lo mismo, rellena toda la información posible en la lista de planos.

1) Seleccionar un vértice.
2) Generar todas las combinaciones de dos aristas adyacentes al vértice seleccionado.
3) Con cada par de aristas y el vértice obtener:
   a) Los coeficientes A, B, C, D de la ecuación del plano.
   b) El hipotético vector normal exterior del plano.
4) Si no quedan vértices a seleccionar, se acaba el algoritmo. Si quedan vértices ir a 1).

2.4.4. ELIMINACIÓN DE PLANOS DUPLICADOS.

Este es el cuarto paso a realizar dentro del algoritmo de generación de gráficos planos *PGG*. Se basa en que al generar todos los posibles planos con cada dos aristas adyacentes se pueden generar planos duplicados. Vamos a analizar lo que propone Qing-Wen Yan en su artículo:

"*Borrar el plano j si existe un plano i, $i \neq j$, tal que la distancia $D_j < \xi$ (ξ es una tolerancia, 10^{-5}), donde D_j es aproximado por $((a_i-a_j)^2+(b_i-b_j)^2+(c_i-c_j)^2+(d_i-d_j)^2)^{½}$, ($a_i$, b_i, c_i , d_i) y (a_j , b_j , c_j , d_j) son los coeficientes de los planos correspondientes.*"

Ahora bien, si comprobamos estos dos planos: y=0, -y=0; tenemos que si se calcula la distancia entre los planos conforme a la fórmula anterior tenemos que:

$$D_j = ((0-0)^2+(1-(-1))^2+(0-0)^2+ (0-0)^2)^{½} = (2^2)^{½} = 2$$

Por lo tanto, mediante esta fórmula nos da que la distancia es 2 y en realidad el plano y=0 es el mismo que el plano –y=0, debido a esto se deduce que la fórmula no sirve para el cálculo de la distancia entre dos planos.

La solución que se ha adoptado para corregir el error es el cálculo de la posición respectiva de los dos planos, para ello se utiliza lo explicado en el punto **2.4.1.2.** .

El algoritmo empleado también será muy sencillo:
(si el número total de planos es *m*)

```
Para i=1 hasta m-1 hacer
    Para j=i+1 hasta m hacer
        Si (PlanosCoincidentes( i , j )) entonces EliminarPlano( j ); FinSi
    FinPara
FinPara
```

2.4.5. BÚSQUEDA DE LAS ARISTAS DE CADA PLANO.

Este es el quinto paso a realizar dentro del algoritmo de generación de gráficos planos *PGG*. Se trata de asociar todos los planos generados en los pasos anteriores con las aristas 3D del modelo alámbrico. Esto se lleva a cabo calculando las distancias de las aristas a los planos. Veamos ahora lo propuesto por Qing-Wen Yan:

"Para cada arista 3D de la lista de aristas 3D, computar las distancias de sus puntos finales D_1 y D_2 al plano. Si $|D_1| \leq \xi$ y $|D_2| \leq \xi$, la arista pertenece al plano."

Por lo tanto para calcular la distancia de una arista a un plano, lo que haremos será calcular las distancias de los dos vértices de la arista al plano, para ello utilizaremos la fórmula de la distancia de un punto a un plano expuesta en el punto **2.4.1.3.** . Si las dos distancias calculadas son menores que una tolerancia $\xi=10^5$, sabremos que la arista pertenece al plano, y el índice de esta arista dentro de la lista de aristas será introducido dentro de la estructura de planos. Cuando este procedimiento termina, tenemos en cada plano todas las aristas que pertenecen a ese plano. Ahora vamos a ver el algoritmo de este procedimiento, teniendo *m* planos, y *n* aristas:

```
Para i=1 hasta m hacer
    Para j=1 hasta n hacer
        V1=AList3D[ j ].v1;  /* V1 índice del primer vértice de la arista j */
        V2=AList3D[ j ].v2;  /* V2 índice del segundo vértice de la arista j */
        D1=CalculaDistanciaPuntoPlano( V1 , i );
        D2=CalculaDistanciaPuntoPlano( V2 , i );
        Si (D1<=ξ) y (D2<=ξ) entonces GuardarAristaEnPlano( j , i ); FinSi
    FinPara
FinPara
```

2.4.6. VERIFICACIÓN DE LOS GRÁFICOS PLANOS.

Este es el sexto y último paso a realizar dentro del algoritmo de generación de gráficos planos *PGG*. Se trata de verificar todos los gráficos planos generados, ya que algunos gráficos planos son patológicos y necesitan ser ajustados o eliminados. Ahora vamos a ver el algoritmo de este procedimiento, teniendo en cuenta que tenemos *m* planos:

1) Recorrer los m planos comprobando el número de aristas que contienen, si este número es menor o igual que dos, hay que eliminar este plano porque con dos o una arista no se puede tener una cara cerrada.
2) Comprobar que todas las aristas pertenecen a más de un gráfico plano, si no es así eliminar esta arista y aplicar el procedimiento *PEVR* (explicado en el punto **2.3.4.**) para eliminar cualquier nueva arista o vértice patológico que aparezca.
3) Se comprueba que las aristas de cada gráfico plano formen una figura cerrada, si hay aristas colgantes, se eliminan y se vuelve a comprobar que es cerrado. Si hay algún gráfico plano no cerrado, entonces se elimina.

Llegados a este punto, hemos generado con éxito los gráficos planos y los vectores normales hipotéticos de los planos. Con estos gráficos planos, los atributos de

línea discontinua serán examinados en el punto siguiente del algoritmo para verificar la consistencia del modelo alámbrico construido.

2.5. ANÁLISIS DE INFORMACIÓN DE LÍNEAS DISCONTINUAS.

En el punto anterior se ha visto como el algoritmo *PGG* construye todos los gráficos planos válidos a partir de los datos del modelo alámbrico. Sin los planos y sus vectores normales exteriores, no somos capaces de determinar la relación de orientación entre planos y aristas. Cuando tenemos los planos y los vectores normales, podemos investigar la corrección de las aristas 3D que son generadas a partir de aristas 2D y, los tipos de líneas sólidas y discontinuas (*broken lines*) en las vistas planas. El procedimiento siguiente elimina las aristas 3D no válidas a partir de la información dada de las líneas discontinuas, a partir de los datos de entrada de las vistas. A partir de ahora llamaremos a este algoritmo *BLR* (*Broken Line Removal*).

Vamos a explicar de manera resumida el algoritmo de este procedimiento:

1. *Seleccionar una arista 3D de la lista de aristas 3D.*
2. *Verificar su proyección de vista Alzado.*
   A partir de los datos de entrada, sabemos el tipo de línea de proyección de cada arista 2D. Si su proyección de alzado es una línea discontinua, habría como mínimo un gráfico plano que no permite ver esta arista 3D cuando se mira en la dirección +OY. Si no existe tal gráfico plano, entonces esta arista 3D debe ser eliminada ya que no tiene ningún gráfico plano que impida su visión y por lo tanto no debería ser discontinua su proyección. Después de esto, ir al paso 5.
3. *Verificar su proyección de vista Planta.*
   Si su proyección de planta es una línea discontinua, habría como mínimo un gráfico plano que no permite ver esta arista 3D cuando se mira en la dirección +OZ. Si no existe tal gráfico plano, entonces esta arista 3D debe ser eliminada ya que no tiene ningún gráfico plano que impida su visión y por lo tanto no debería ser discontinua su proyección. Después de esto, ir al paso 5.
4. *Verificar su proyección de vista Perfil.*
   Si su proyección de perfil es una línea discontinua, habría como mínimo un gráfico plano que no permite ver esta arista 3D cuando se mira en la dirección +OX. Si no existe tal gráfico plano, entonces esta arista 3D debe ser eliminada ya que no tiene ningún gráfico plano que impida su visión y por lo tanto no debería ser discontinua su proyección. Después de esto, ir al paso 5.
5. *Si todas las aristas 3D han sido examinadas entonces ir al paso 6; en otro caso, ir al paso 1.*
6. *La eliminación de aristas puede crear nuevas aristas y vértices patológicos.*
   Por lo tanto, el procedimiento ***PEVR*** es aplicado para depurar nuevas aristas y vértices patológicos. Además, después de esto también se vuelve a aplicar la ***verificación de los gráficos planos*** para actualizar y comprobar los gráficos planos después de la eliminación de aristas.

El procesado de la información de las líneas discontinuas es útil antes de la generación de bucles de caras en los pasos siguientes, porque la eliminación de aristas irrelevantes produce una computación más rápida de bucles de caras y bucles de

cuerpos, debido a la reducción de ambigüedad que supone la eliminación de estas aristas. Además, con este procedimiento, muchas aristas redundantes son eliminadas. Esto evita mucha más redundancia en la generación de bucles de caras y cuerpos, y de este modo la computación es más eficiente.

2.6. GENERACIÓN DE CARAS.

En el punto anterior se ha visto como el algoritmo *BLR* utiliza la información de las líneas discontinuas de los datos de entrada para eliminar aristas irrelevantes en la generación del objeto. Esto incide en una mayor rapidez en la computación de este procedimiento del algoritmo, lo que provoca una mayor eficiencia del algoritmo de reconstrucción. En las secciones anteriores, hemos recogido información sobre vértices, aristas y gráficos planos. Para determinar los bucles de caras de un objeto, necesitamos generar todos los bucles básicos de cada gráfico plano. Llamaremos a partir de ahora a este procedimiento como *FLG* (*Face Loop Generation*, Generación de Bucles de Caras), el cual determina todos los bucles básicos que son usados para construir todos los bucles de caras para cada gráfico plano.

Tenemos que $\Psi=\{L_1, L_2, \ldots, L_k\}$ es un conjunto de bucles básicos, y además se sabe que $L_i \cap L_j = \{\text{aristas, vértices}\}$, y $L_i \not\subset L_j$ o $L_j \not\subset L_i$, para todo *i, j*. Además, cualquier bucle $L_k \notin \Psi$ puede ser obtenido de la forma:

$$L_k = \bigcup_{s=1}^{m} L_{k_s}$$

para $m \geq 1$, donde $L_{ks} \in \Psi$. El algoritmo *FLG* también determina la posición relativa entre dos bucles, y se clasifican entre bucles interiores y exteriores, y genera bucles de caras a partir de bucles básicos.

Debido a que este algoritmo es bastante complicado, a continuación pasaremos verlo de una forma general, y después se expondrán los puntos del algoritmo que precisen una mayor explicación. Partimos de un gráfico plano generado por el procedimiento *PGG*, por lo tanto el algoritmo siguiente se tendrá que aplicar a todos los gráficos planos.

Algoritmo:

1. Construir una tabla de vértices - aristas adyacentes *adj_tab*=(*v*, *ne*, *SE*), donde $v \in VList3D$, *ne* es el número de aristas adyacentes a *v*, y $SE=\{e_1, e_2, \ldots, e_{ne}\}$ es una lista de todas las aristas adyacentes a *v* en el gráfico plano.

2. Encontrar la arista adyacente ordenada de un vértice.

2.1. Seleccionar una entrada (v, ne, SE) de la tabla *adj_tab*.

2.2. OE(v) será un conjunto de aristas adyacentes a v, ordenado. Las aristas en OE(v) están organizadas en orden ascendente del ángulo formado respecto a una arista seleccionada e_k en el sentido de las agujas del reloj. El ángulo se calcula alrededor del hipotético vector normal exterior del plano.

2.3. Si todas las entradas de la tabla han sido procesadas, ir al paso **3**; en otro caso ir al paso **2.1.**.

3. Encontrar todos los bucles básicos dentro de un gráfico plano. Cada arista (v_i, v_j) tiene dos sentidos alternativos: de v_i a v_j, y viceversa. E_c denota una arista actual seleccionada, y V_{start} y V_{end} denotan un vértice inicial y final de un bucle básico, respectivamente. Inicialmente, el conjunto de bucles básicos se inicializa a $\Psi \leftarrow \varnothing$, y el conjunto de aristas del bucle básico se inicializa a $L \leftarrow \varnothing$.

3.1. Seleccionar una arista $e=(v_i, v_j)$ del gráfico plano, $E_c \leftarrow e$. Si todas las aristas han sido seleccionadas en este gráfico plano, entonces ir al paso **4.**; en otro caso, hacer $L \leftarrow L \cup E_c$, y continuar.

3.2. Seleccionar la arista adyacente a E_c en V_{start} de $OE(V_{start})$. Suponiendo que este conjunto sea $OE(V_{start})=\{e_1, e_2, \ldots, E_c, e_{c'}, \ldots, e_{ne}, e_1\}$. Entonces, $e_{c'}=(v_k, v_m)$ es la primera arista adyacente a E_c en V_{start}. Hacer $L \leftarrow L \cup e_{c'}$.

3.3. Si E_c ha visitado en las dos direcciones el bucle básico actual L, entonces E_c es un puente en el gráfico plano y debe ser eliminada, $L=\{e_1, \ldots, e_j, E_c, e_k, \ldots, e_m, E_c\}$. Se formará un bucle básico por las aristas $e_k, \ldots, e_m$ en L. Después hacer $\Psi \leftarrow \Psi \cup \{L\}$.

3.4. Si $V_{start}=V_{end}$, se ha descubierto un bucle básico; hacer $\Psi \leftarrow \Psi \cup L$, y $L \leftarrow \varnothing$, e ir al paso **3.1.**; en otro caso, ir al paso **3.2.**.

4. Identificar la relación de inclusión entre los bucles básicos en un gráfico plano. Para cada par de bucles L_i y L_j, si L_i incluye a L_j, entonces *incluye*(i,j)=1, en otro caso, *incluye*(i,j)=0.

5. Formar los bucles de caras. Dada la relación de inclusión entre bucles básicos:

Si L_i no incluye algún bucle, entonces L_i forma un bucle de cara;
Sino **Si** L_i incluye a L_j, entonces L_i es un bucle exterior, L_j es un bucle interior y L_i incluye a L_j directamente.
Sino **Si** el bucle L_i incluye más de un bucle, por ejemplo, $L_{j1}, L_{j2}, \ldots, L_{jn}$, verificar cuando los L_j son incluidos por otros bucles básicos.

Un bucle de cara se forma a partir de L_i y bucles básicos incluidos por L_i directamente. En este bucle de cara L_i es un bucle exterior, y los otros son bucles básicos interiores.

Para cada gráfico plano, usamos el algoritmo *FLG* para generar los bucles de caras. Como las aristas colgantes y aisladas son eliminadas cuando se generan los gráficos planos, cada arista pertenecería a dos o más bucles de caras. Si una arista conecta con menos de dos bucles de caras, esta arista es patológica, y sería eliminada de la lista de aristas 3D. En este caso, el procedimiento *PEVR* se aplica de nuevo para eliminar esta arista redundante.

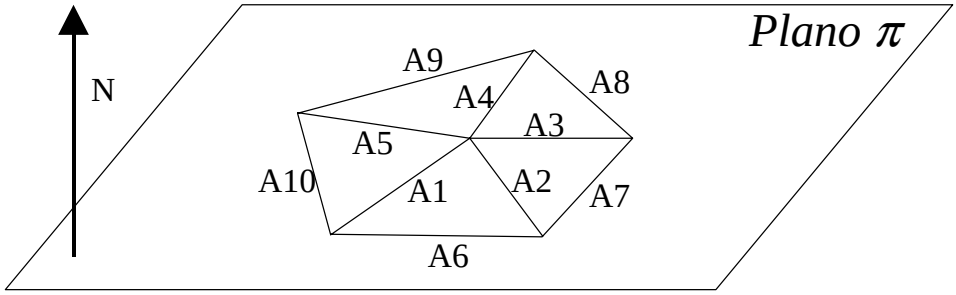
2.6.1. ORDENACIÓN DEL CONJUNTO DE ARISTAS ADYACENTES.

Este procedimiento es utilizado en el punto segundo del algoritmo *FLG*, en el cual se pide una ordenación del conjunto de aristas adyacentes a un vértice dado. En este punto vamos a estudiar el método empleado para llevar a cabo esta ordenación y

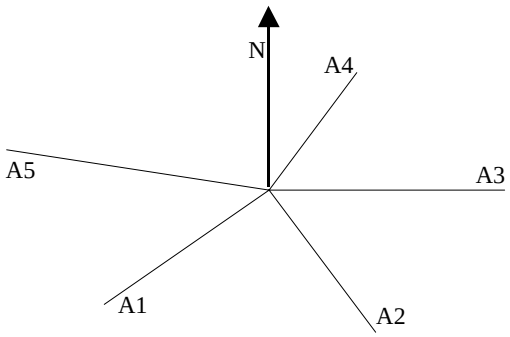
además veremos con todo detalle un ejemplo real. Veamos ahora lo que dice el artículo de Qing-Wen Yan acerca del punto que estamos tratando ahora:

"OE(v) será un conjunto de aristas adyacentes a v, ordenado. Las aristas en OE(v) están organizadas en orden ascendente del ángulo formado respecto a una arista seleccionada e_k en sentido de las agujas del reloj. El ángulo se calcula alrededor del vector normal exterior del plano."

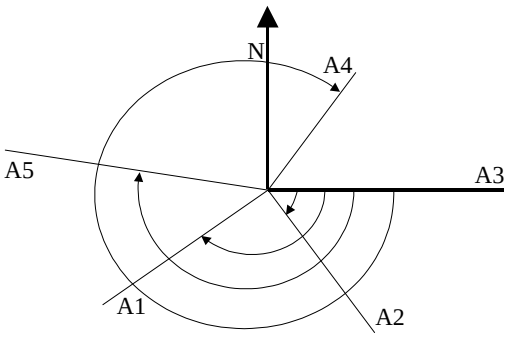
Hay que decir que en la implementación del algoritmo no se ha generado ningún conjunto *OE(v)*, en realidad lo que se hace es ordenar los conjuntos de aristas adyacentes, dentro de la estructura de datos *adj_tab* en el campo *SE*, o lo que es lo mismo se ha ordenado todos los conjuntos de aristas adyacentes *adj_tab.SE*. Pasamos ahora a ver un ejemplo sencillo de cómo se requiere la ordenación:



Tenemos el gráfico plano formado por las aristas {A1, A2, ... , A10}, este gráfico plano está incluido dentro del *plano* π, y tiene como vector normal exterior el vector *N*. Vamos a estudiar como se ordenarían las aristas A1, ... , A5 que tienen un vértice en común. Para ello ampliamos la zona en cuestión y damos un estado inicial al conjunto de aristas adyacentes:



	1	2	3	4	5
SE	A3	A5	A1	A4	A2



	1	2	3	4	5
SE	A3	A2	A1	A5	A4

Partimos de un posible estado inicial en el que tenemos el conjunto de aristas adyacentes de la forma siguiente: *SE*={A3, A5, A1, A4, A2}. Seleccionamos como arista inicial la arista A3, que es la primera del conjunto de aristas adyacentes. Luego se calculan los ángulos de esta arista inicial A3 con respecto al resto de aristas, teniendo en cuenta que el ángulo se calcula alrededor del vector normal exterior *N* y en sentido de

las agujas del reloj. Al final tenemos que ordenar el conjunto de aristas adyacentes en orden ascendente de los ángulos calculados, obteniendo el conjunto ordenado *SE*={A3, A2, A1, A5, A4}.

Ahora vamos a ver matemáticamente como calculamos los ángulos que forman las aristas dependiendo del criterio explicado en el párrafo anterior.

Angulo formado por dos rectas.

Sean ***u*** y ***v*** los vectores de dirección de dos rectas cualquiera. Designaremos como θ al ángulo formado entre las dos rectas. Por lo tanto, tenemos que:

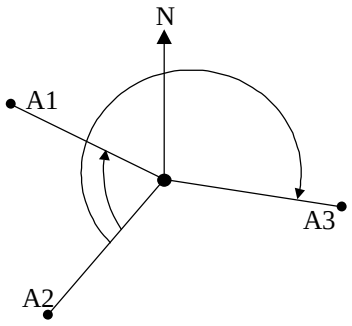
$$\cos(\theta) = \frac{u \cdot v}{|u| \cdot |v|}$$

En este caso lo que tenemos son dos aristas con un punto común, por lo tanto tenemos que calcular los vectores de dirección de las rectas que contienen las dos aristas. Si suponemos que la primera arista viene definida por los vértices (v1, v2) y la segunda arista por (v1, v3), el vector *u* calculará como v2-v1, y el vector *v* se calculará como v3-v1, siendo v1 el punto común de las dos aristas.

Con esta fórmula anterior se calcula el ángulo entre dos rectas sin tener en cuenta los criterios enumerados anteriormente, pero teniendo en cuenta estos criterios, el ángulo formado por las dos aristas puede ser, o el calculado con la fórmula anterior, o el ángulo complementario al calculado con la fórmula anterior. Estos criterios los vamos a ver a continuación.

Aplicación de los criterios al ángulo formado por dos aristas.

Vamos a explicar este apartado con un ejemplo muy sencillo:



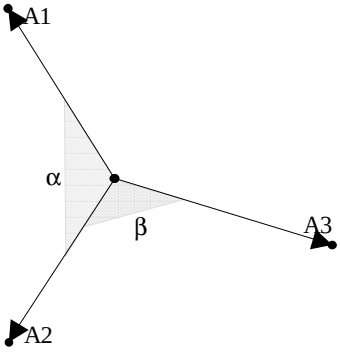
Inicialmente:

A2	A3	A1

Resultado:

A2	A1	A3

Inicialmente tenemos el conjunto de aristas adyacentes de la siguiente forma {A2, A3, A1}. Como la primera arista del conjunto es la arista A2, seleccionaremos ésta para calcular los ángulos que forma con el resto de aristas. Por lo tanto, con lo visto en el apartado anterior tenemos que calcular los ángulos entre A2-A3 y A2-A1.



Tenemos ahora que el ángulo(A2-A3)=β y que el ángulo(A2-A1)=α. Ahora formamos los vectores *A1*, *A2* y

A3, tal y como se ven en la figura de la izquierda, y se calculan los productos vectoriales: $A2 \times A1$ y $A2 \times A3$. Si los vectores resultantes de los productos vectoriales tienen el mismo sentido que el hipotético vector normal exterior del plano, entonces el ángulo γ calculado de esas dos aristas estaría en sentido antihorario, por lo tanto habría que quedarse con el ángulo $360^o-\gamma$.

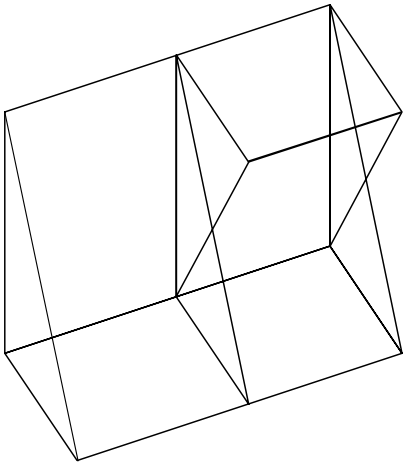
Si los vectores resultantes tienen el sentido contrario al hipotético vector normal exterior del plano, entonces el ángulo γ se ha calculado en sentido horario. Faltaría explicar como sabemos que dos vectores tienen el mismo sentido, o sentido contrario. Pues bien, esto se hace calculando el ángulo formado entre los dos vectores. Si el ángulo es 0^o entonces los vectores tienen el mismo sentido. Pero si el ángulo es 180^o, entonces tienen sentidos contrarios.

Ahora ya tenemos los ángulos formados entre la arista seleccionada y el resto de aristas. Sólo falta ordenar el conjunto de aristas adyacentes en orden creciente de ángulos. Se ha utilizado en la implementación de este procedimiento el método burbuja de ordenación de listas por su sencillez.

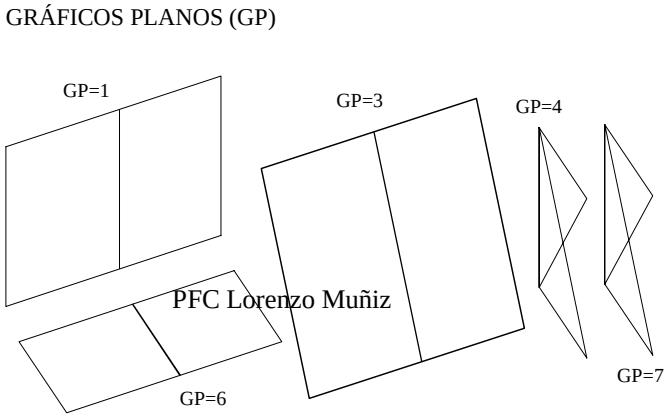
Pasamos ahora a analizar un ejemplo real, para que se vea como se ordenan todos los conjuntos de aristas adyacentes.

Ejemplo.

Se va a utilizar como ejemplo de la ordenación de aristas adyacentes el *Dibujo 1* de la aplicación Rec3D. En la figura siguiente tenemos el modelo alámbrico del objeto 3D en el momento anterior a aplicar el algoritmo *FLG*. A continuación se verán los casos en los que se realiza la ordenación, ya que cuando sólo tenemos dos aristas adyacentes no se aplica ninguna ordenación, puesto que dos aristas siempre están ordenadas.

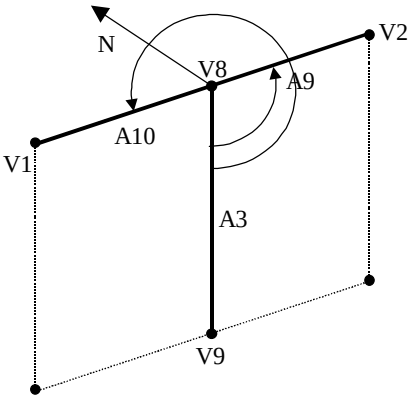


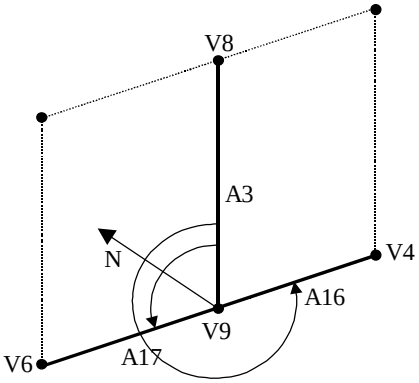
El algoritmo de ordenación se aplicaría en los siguientes gráficos planos:



En el gráfico plano 1 tenemos que hay 6 vértices de los cuales V8 y V9 son los únicos que tienen más de dos aristas adyacentes. En la figura siguiente se muestra como quedarían ordenadas las aristas adyacentes:

GP=1
Nº vértices=6
N(0,-30,0)



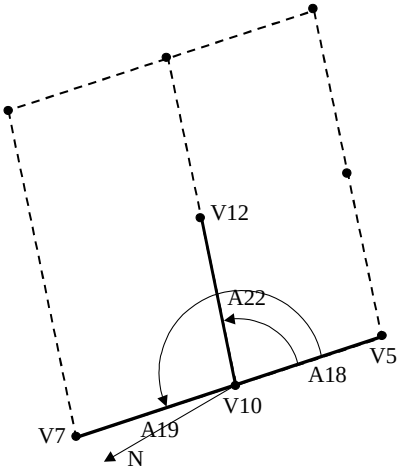


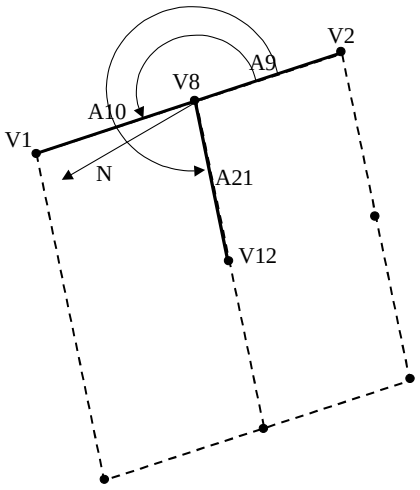
3	9	10

3	17	16

En el gráfico plano 3 tenemos que hay 8 vértices de los cuales V8 y V10 son los únicos que tienen más de dos aristas adyacentes. En la figura siguiente se muestra como quedarían ordenadas las aristas adyacentes:

GP=3
Nº vértices=8
N(0,-15,-4.5)



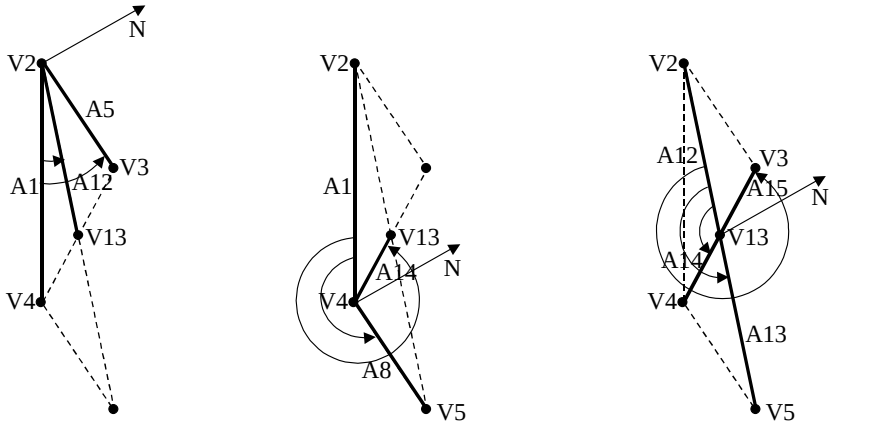


18	22	19

9	10	21

En el gráfico plano 4 tenemos que hay 5 vértices de los cuales V2, V4 y V13 son los únicos que tienen más de dos aristas adyacentes. Como el gráfico plano 7 es igual al gráfico plano 4, sólo veremos este último. En la figura siguiente se muestra como quedarían ordenadas las aristas adyacentes:

GP=4
Nº vértices=5
N(-30,0,0)



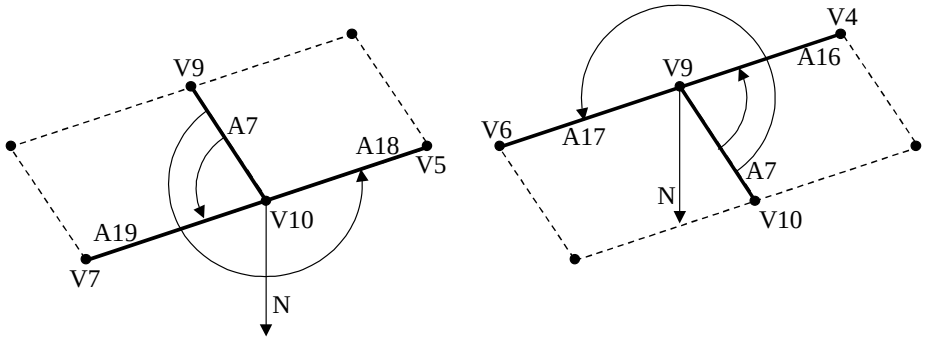
1	12	5

1	8	14

12	14	13	15

En el gráfico plano 6 tenemos que hay 6 vértices de los cuales V9 y V10 son los únicos que tienen más de dos aristas adyacentes. En la figura siguiente se muestra como quedarían ordenadas las aristas adyacentes:

GP=6
Nº vértices=6
N(0,0,-9)



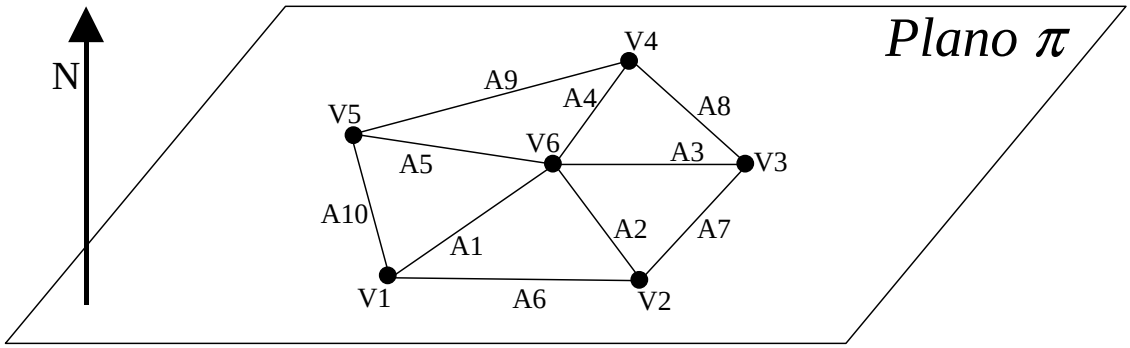
7	19	18

7	16	17

2.6.2. GENERACIÓN DE TODOS LOS BUCLES BÁSICOS.

Este procedimiento se lleva a cabo en el tercer punto del algoritmo *FLG*, en el cual se tienen que encontrar todos los bucles básicos dentro de un gráfico plano. En este punto se va a estudiar el método empleado para llevar a cabo esta búsqueda de bucles básicos. Vamos a explicar mediante dos ejemplos la forma de ir recorriendo los gráficos planos en busca de bucles básicos.

Se va a utilizar como primer ejemplo un gráfico plano utilizado en el punto anterior, y lo podemos ver en la siguiente figura:



Vista la figura anterior se va a proponer una configuración de la tabla de vértices y aristas adyacentes, de forma que los conjuntos de aristas adyacentes estén ordenados según el punto anterior.

Tabla adj_tab

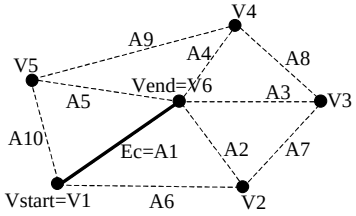
v	***ne***	***SE***
V1	3	{A1, A6, A10}
V2	3	{A2, A7, A6}
V3	3	{A3, A8, A7}
V4	3	{A4, A9, A8}
V5	3	{A5, A10, A9}
V6	5	{A1, A5, A4, A3, A2}

Ahora se va definir una serie de variables que se van a utilizar durante la traza del algoritmo, o proceso de búsqueda de bucles básicos, tenemos las variables:

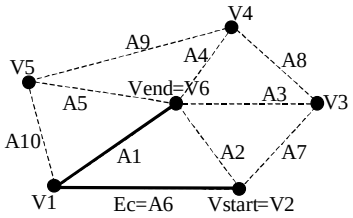
- E_c que indicará en cada momento la arista seleccionada.
- V_{start} será el vértice inicial del bucle básico que se está construyendo.
- V_{end} será el vértice final del bucle básico que se está construyendo.
- Ψ es el conjunto de bucles básicos. Inicialmente vacío ($\varnothing$).
- L es el conjunto de aristas de un bucle básico en formación. Inicialmente vacío ($\varnothing$).

Se va a utilizar con los conjuntos los operadores *asignación* ($\leftarrow$) y *unión* ($\cup$).

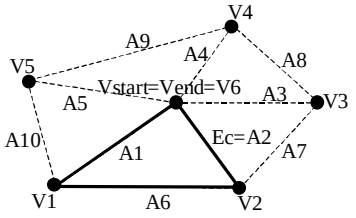
Inicialmente se selecciona la primera arista, en este caso A1, por lo tanto E_c=A1. Además también se selecciona un sentido de búsqueda de aristas, que se hace asignando a las variables V_{start} y V_{end} los vértices de la arista seleccionada. Tomamos el sentido: V_{start}=V1 y V_{end}=V6. También añadimos la arista seleccionada al conjunto de aristas del bucle básico: L←L∪E_c.



Ahora pasamos a realizar un bucle para ir encontrando las aristas sucesivas. Se selecciona la arista adyacente a E_c en V_{start}, tenemos que V_{start}=V1 y las aristas adyacentes en V1 son {A1, A6, A10}. Por lo tanto la arista adyacente a E_c=A1 en el vértice V1 es la arista A6, entonces realizamos las asignaciones: L←L∪{A6} y V_{start}=V2. Por lo que se tiene que L={A1, A6}, E_c=A6, V_{start}=V2 y V_{end}=V6.

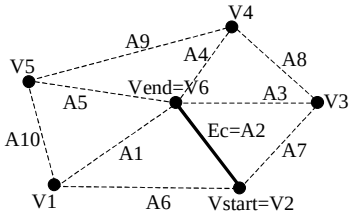


Se selecciona la arista adyacente a E_c en V_{start}, tenemos que V_{start}=V2 y las aristas adyacentes en V2 son {A2, A7, A6}. Por lo tanto la arista adyacente a E_c=A6 en el vértice V2 es la arista A2, entonces realizamos las asignaciones: L←L∪{A2} y V_{start}=V6. Por lo que se tiene que L={A1, A6, A2}, E_c=A2, V_{start}=V6 y V_{end}=V6. Como tenemos que V_{start}=V_{end}, nos indica que ya tenemos un bucle básico. Cuando se encuentra un bucle básico hay que hacer: Ψ←Ψ∪L y L←∅. Ahora tenemos un bucle básico: Ψ={{ A1, A6, A2}}, que vemos en la figura siguiente.

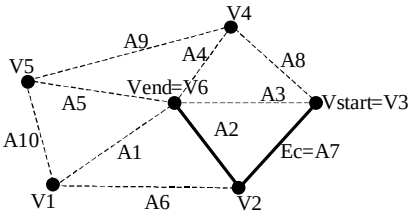


Cuando se empieza de nuevo la búsqueda de otro bucle básico, hay que seleccionar una arista que no se haya seleccionado antes en el paso inicial. En este caso, seleccionamos la arista siguiente: E_c=A2. Además, se selecciona un sentido de búsqueda de aristas, que se hace asignando a las variables V_{start} y V_{end} los vértices de la arista seleccionada. Tomamos el sentido: V_{start}=V2 y V_{end}=V6. También añadimos la arista seleccionada al conjunto de aristas del bucle básico: L←L∪E_c.

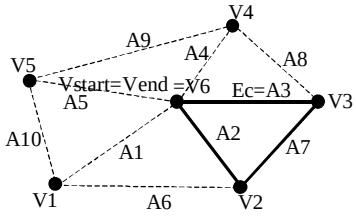
Tenemos inicialmente:



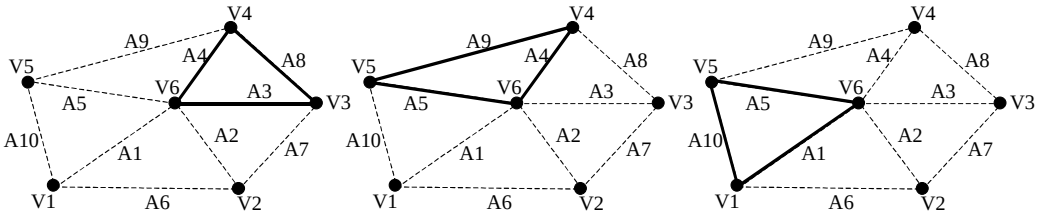
Ahora pasamos a realizar un bucle para ir encontrando las aristas sucesivas. Se selecciona la arista adyacente a E_c en V_{start}, tenemos que V_{start}=V2 y las aristas adyacentes en V2 son {A2, A7, A6}. Por lo tanto la arista adyacente a E_c=A2 en el vértice V2 es la arista A7, entonces realizamos las asignaciones: L←L∪{A7} y V_{start}=V3. Por lo que se tiene que L={A2, A7}, E_c=A7, V_{start}=V3 y V_{end}=V6.



Se selecciona la arista adyacente a E_c en V_{start}, tenemos que V_{start}=V3 y las aristas adyacentes en V3 son {A3, A8, A7}. Por lo tanto la arista adyacente a E_c=A7 en el vértice V3 es la arista A3, entonces realizamos las asignaciones: L←L∪{A3} y V_{start}=V6. Por lo que se tiene que L={A2, A7, A3}, E_c=A3, V_{start}=V6 y V_{end}=V6. Como tenemos que V_{start}=V_{end}, nos indica que ya tenemos un bucle básico. Cuando se encuentra un bucle básico hay que hacer: Ψ←Ψ∪L y L←∅. Ahora tenemos los bucles básicos: Ψ={{A1, A6, A2}, {A2, A7, A3}}, que vemos en la figura siguiente.



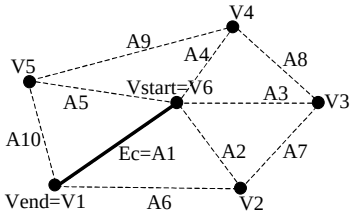
Si seleccionamos sucesivamente las aristas A3, A4, A5, A6, A7, A8, A9 y A10, con el mismo sentido de búsqueda que el empleado en los dos bucles básicos anteriores se obtienen los siguientes bucles básicos:



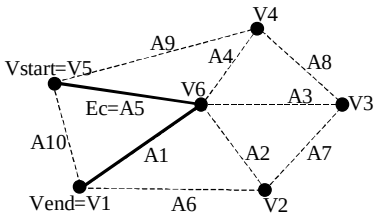
El conjunto de los bucles básicos tiene, por ahora, los siguientes bucles:

Ψ={{A1, A6, A2}, {A2, A7, A3}, {A3, A8, A4}, {A4, A9, A5}, {A5, A10, A1}}.

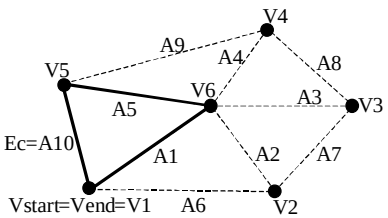
Ahora lo que hay que hacer es volver a ejecutar este procedimiento como si se empezase desde el principio, pero ahora cuando se inicializa las variables V_{start} y V_{end} se tomará el sentido contrario al tomado anteriormente. Por lo tanto, se selecciona la primera arista, en este caso A1, por lo tanto E_c=A1. Además ahora se selecciona el sentido de búsqueda de aristas contrario al seleccionado anteriormente, que se hace asignando a las variables V_{start} y V_{end} los vértices de la arista seleccionada. Tomamos el sentido: V_{start}=V6 y V_{end}=V1. También añadimos la arista seleccionada al conjunto de aristas del bucle básico: L←L∪E_c.



Ahora pasamos a realizar un bucle para ir encontrando las aristas sucesivas. Se selecciona la arista adyacente a E_c en V_{start}, tenemos que V_{start}=V6 y las aristas adyacentes en V6 son {A1, A5, A4, A3, A2}. Por lo tanto la arista adyacente a E_c=A1 en el vértice V6 es la arista A5, entonces realizamos las asignaciones: L←L∪{A5} y V_{start}=V5. Por lo que se tiene que L={A1, A5}, E_c=A5, V_{start}=V5 y V_{end}=V1.

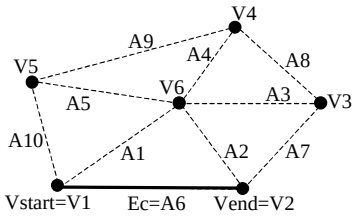


Se selecciona la arista adyacente a E_c en V_{start}, tenemos que V_{start}=V5 y las aristas adyacentes en V5 son {A5, A10, A9}. Por lo tanto la arista adyacente a E_c=A5 en el vértice V5 es la arista A10, entonces realizamos las asignaciones: L←L∪{A10} y V_{start}=V1. Por lo que se tiene que L={A1, A5, A10}, E_c=A10, V_{start}=V1 y V_{end}=V1. Como tenemos que V_{start}=V_{end}, nos indica que ya tenemos un bucle básico. Cuando se encuentra un bucle básico hay que hacer: Ψ←Ψ∪L y L←∅. Pero como este bucle básico ya lo tenemos en el conjunto Ψ, entonces no se añade el bucle básico. En la figura siguiente vemos el gráfico plano generado que no será incluido en Ψ:

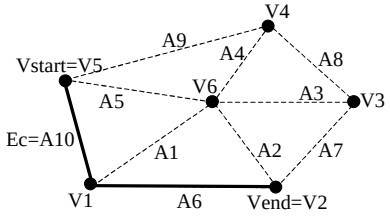


Con las aristas A2, A3, A4 y A5, con este sentido tampoco se generan nuevos bucles básicos. Pero veamos lo que pasa cuando se selecciona la arista A6:

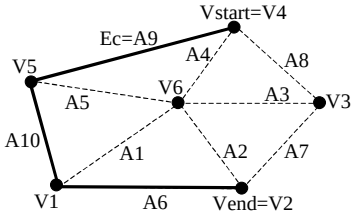
Seleccionamos la arista A6, por lo tanto E_c=A6. Además ahora se selecciona el sentido de búsqueda V_{start}=V1 y V_{end}=V2. También añadimos la arista seleccionada al conjunto de aristas del bucle básico: L←L∪E_c.



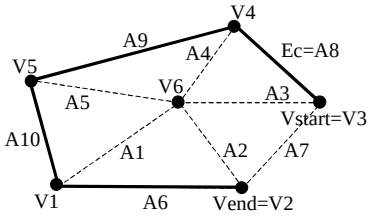
Ahora pasamos a realizar un bucle para ir encontrando las aristas sucesivas. Se selecciona la arista adyacente a E_c en V_{start}, tenemos que V_{start}=V1 y las aristas adyacentes en V1 son {A1, A6, A10}. Por lo tanto la arista adyacente a E_c=A6 en el vértice V1 es la arista A10, entonces realizamos las asignaciones: L←L∪{A10} y V_{start}=V5. Por lo que se tiene que L={A6, A10}, E_c=A10, V_{start}=V5 y V_{end}=V2.



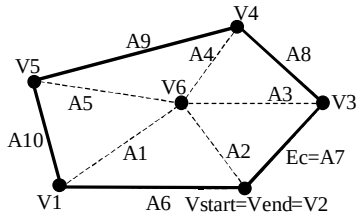
Se selecciona la arista adyacente a E_c en V_{start}, tenemos que V_{start}=V5 y las aristas adyacentes en V5 son {A5, A10, A9}. Por lo tanto la arista adyacente a E_c=A10 en el vértice V5 es la arista A9, entonces realizamos las asignaciones: L←L∪{A9} y V_{start}=V4. Por lo que se tiene que L={A6, A10, A9}, E_c=A9, V_{start}=V4 y V_{end}=V2.



Se selecciona la arista adyacente a E_c en V_{start}, tenemos que V_{start}=V4 y las aristas adyacentes en V4 son {A4, A9, A8}. Por lo tanto la arista adyacente a E_c=A9 en el vértice V4 es la arista A8, entonces realizamos las asignaciones: L←L∪{A8} y V_{start}=V3. Por lo que se tiene que L={A6, A10, A9, A8}, E_c=A8, V_{start}=V3 y V_{end}=V2.



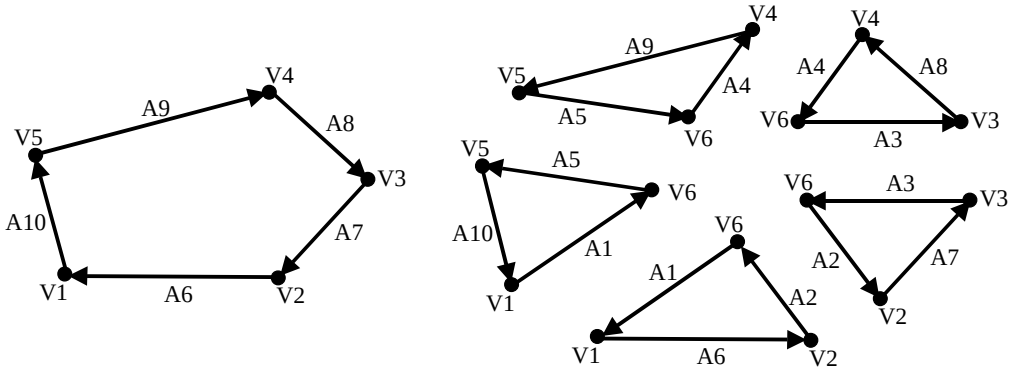
Se selecciona la arista adyacente a E_c en V_{start}, tenemos que V_{start}=V3 y las aristas adyacentes en V3 son {A3, A8, A7}. Por lo tanto la arista adyacente a E_c=A8 en el vértice V3 es la arista A7, entonces realizamos las asignaciones: L←L∪{A7} y V_{start}=V2. Por lo que se tiene que L={A6, A10, A9, A8, A7}, E_c=A7, V_{start}=V2 y V_{end}=V2. Como tenemos que V_{start}=V_{end}, nos indica que ya tenemos un bucle básico. Cuando se encuentra un bucle básico hay que hacer: Ψ←Ψ∪L y L←∅.



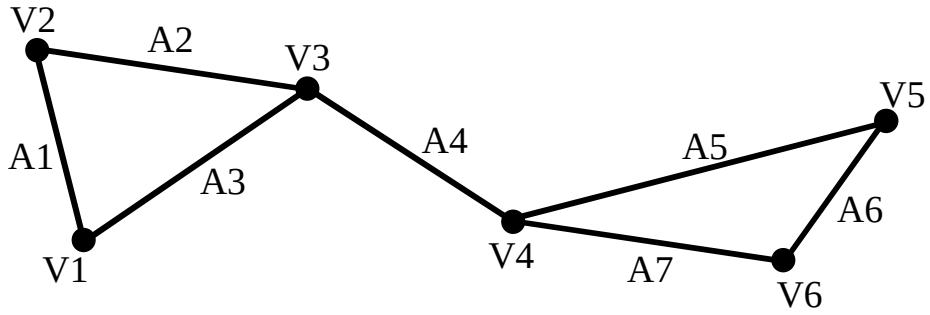
Ahora tenemos los bucles básicos:

Ψ={{A1, A6, A2}, {A2, A7, A3}, {A3, A8, A4},
{A4, A9, A5}, {A5, A10, A1}, {A6, A10, A9, A8, A7}}.

Ya no se añaden más bucles básicos debido a que las aristas que quedan por seleccionar, A7, A8, A9, A10, con el mismo sentido, generan el mismo bucle básico. Por lo tanto, ya se ha concluido la búsqueda de bucles básicos para este ejemplo que estamos viendo. Podemos ver en la figura siguiente todos los bucles básicos encontrados:



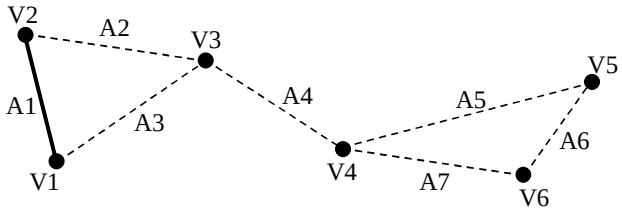
Vamos a ver ahora un ejemplo sencillo en el que se aplica una acción especial dentro de la generación de bucles básicos, supongamos el gráfico plano siguiente, con su tabla asociada de vértices y aristas adyacentes ordenada:



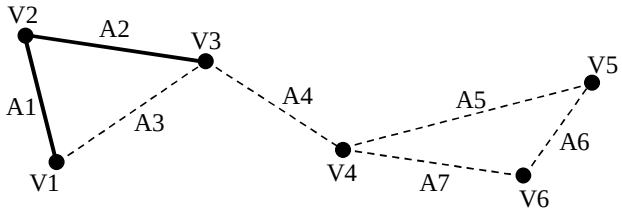
v	*ne*	*SE*
V1	2	{A1, A3}
V2	2	{A1, A2}
V3	3	{A4, A3, A2}
V4	3	{A4, A5, A7}
V5	2	{A5, A6}
V6	2	{A6, A7}

Inicialmente se selecciona la primera arista, en este caso A1, por lo tanto E_c=A1. Además también se selecciona un sentido de búsqueda de aristas, que se hace asignando a las variables V_{start} y V_{end} los vértices de la arista seleccionada. Tomamos el sentido: V_{start}=V2 y V_{end}=V1. También añadimos la arista seleccionada al conjunto de aristas del bucle básico: L←L∪E_c. En las siguientes figuras se muestra visualmente la búsqueda de aristas del bucle básico:

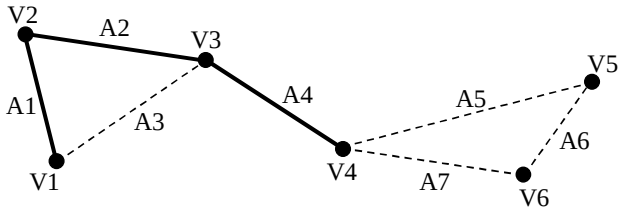
1º)



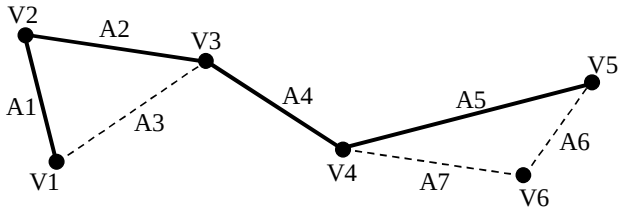
2º)



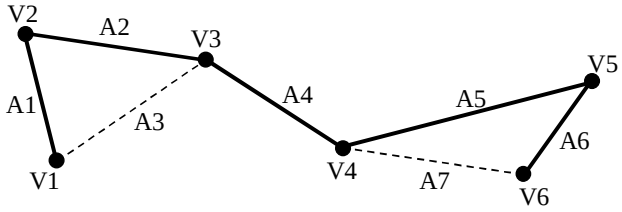
3º)



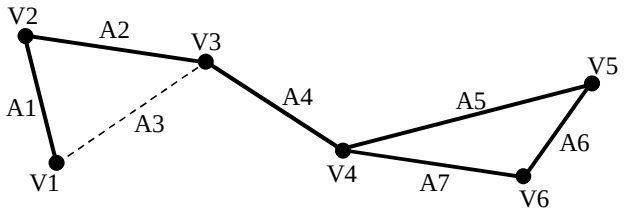
4º)



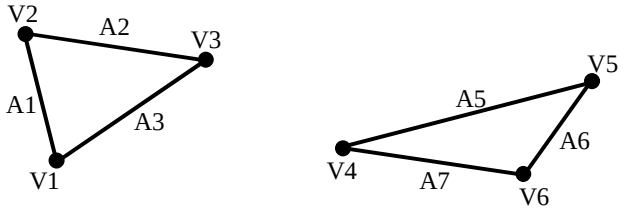
5º)



Después de la figura anterior tenemos que las variables tienen los valores siguientes: V_{start}=V6, V_{end}=V1, E_c=A6, Ψ=∅, L={A1, A2, A4, A5, A6}.



Ahora tenemos que las variables tienen los valores siguientes: V_{start}=V4, V_{end}=V1, E_c=A7, Ψ=∅, L={A1, A2, A4, A5, A6, A7}. En este punto vamos a llegar a una situación que no habíamos visto antes. Decimos que la arista A4 es una *arista puente* porque actúa como un puente entre dos bucles básicos y no forma parte de ningún bucle básico. Esta situación se detecta en la siguiente acción del algoritmo que modifica las variables de forma que: V_{start}=V3, V_{end}=V1, E_c=A4, Ψ=∅, L={A1, A2, A4, A5, A6, A7, A4}. En el conjunto L tenemos que la arista A4 está dos veces, por lo que nos indica que es una *arista puente*. El algoritmo nos dice que, en estos casos, hay que eliminar la arista E_c=A4, formar un bucle básico con las aristas de L que están entre las dos aristas A4, e inicializar la búsqueda de bucles básicos. Por lo tanto, tenemos ahora un bucle básico que es Ψ={{A5, A6, A7}}. Si volvemos a ejecutar la búsqueda de bucles básicos se obtiene el segundo y último bucle básico, quedando Ψ={{A5, A6, A7}, {A1, A2, A3}} como se muestra en la figura:



Hasta aquí se ha visto todo lo relacionado con la búsqueda de bucles básicos a partir de una tabla ordenada de vértices y aristas adyacentes. El apartado siguiente, dentro del procedimiento de generación de bucles de caras, es el encargado de la identificación de las relaciones de inclusión entre los bucles básicos dentro de un gráfico plano. Por lo tanto, la sección siguiente va a tomar como datos de entrada el conjunto de todos los bucles básicos, para establecer la relación entre ellos. Esta relación entre bucles básicos se utilizará en el último punto del procedimiento para formar los bucles de caras del objeto 3D.

2.6.3. IDENTIFICAR LA RELACIÓN ENTRE BUCLES BÁSICOS.

Este procedimiento es la cuarta parte del algoritmo *FLG*. Se va a estudiar la relación de inclusión entre todos los bucles básicos, dentro de un gráfico plano, para la generación final de los bucles de caras (que dependen de estas relaciones). Veamos ahora lo que dice el artículo de Qing-Wen Yan acerca del punto que estamos tratando ahora:

"Identificar la relación de inclusión entre los bucles básicos en un gráfico plano. Para cada par de bucles L_i y L_j, si L_i incluye a L_j, entonces incluye(i,j)=1, en otro caso, incluye(i,j)=0."

Luego el algoritmo de este procedimiento de identificación de relaciones podría ser definido básicamente, siendo *m* el número de gráficos planos:

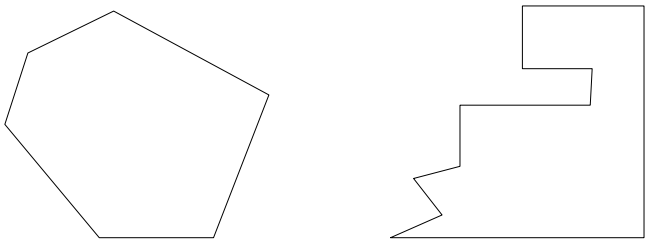
```
Para k=1 hasta m hacer
        n=ObtieneNumeroBuclesBasicos( k );
        Para i=1 hasta n-1 hacer
                Para j=i+1 hasta n hacer
                        Si (IdentificaRelacion( i , j )) entonces incluye[i][j]=1;
                        Sino incluye[i][j]=0;
                        FinSi
                FinPara
        FinPara
FinPara
```

Del algoritmo anterior hay que decir:

- La función *ObtieneNumeroBuclesBasicos(gp)* obtiene el número de bucles básicos que hay en el gráfico plano *gp*.
- La función *IdentificaRelacion(b1,b2)* devuelve **verdadero** si el bucle básico *b1* incluye al bucle básico *b2*. En otro caso, devuelve **falso**.
- La estructura de datos *incluye[b1][b2]*, que se trata de una matriz de ceros y unos donde *b1* es la fila, y *b2* es la columna.

2.6.3.1. Caras cóncavas y convexas.

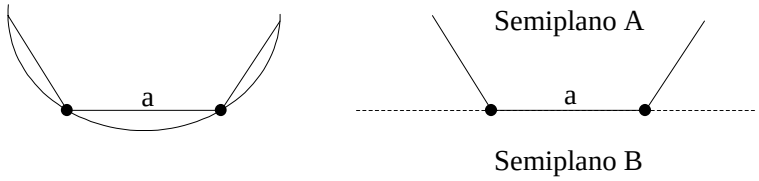
Una cara la consideramos convexa si todas sus aristas son convexas, en caso contrario la consideramos cóncava. En la figura siguiente tenemos dos ejemplos, uno de cara convexa y otro de cara cóncava.



CARA CONVEXA CARA CÓNCAVA

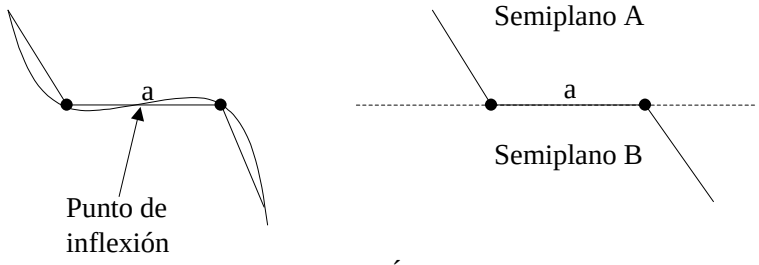
También podríamos considerar la cara convexa como aquella que al ser atravesada por una línea en cualquier ángulo, como máximo corta dos aristas de la cara, siendo cóncava cuando se cortan más de dos aristas.

Una arista será convexa en una cara si tanto la arista anterior como la posterior están en el mismo lado con relación a ella. Es decir, será convexa si las aristas contiguas quedan en el mismo semiplano respecto de la recta teórica que la contiene.



ARISTA CONVEXA

Una arista será cóncava en una cara si la arista anterior y posterior quedan en lados diferentes con relación a ella. Es decir, una arista será cóncava si sus aristas contiguas quedan en semiplanos distintos respecto a la recta teórica que la contiene. En este caso, si trazásemos una curva que pasase por cada uno de los vértices, comprobaríamos que el arco que comprende la arista cóncava posee un punto de inflexión.



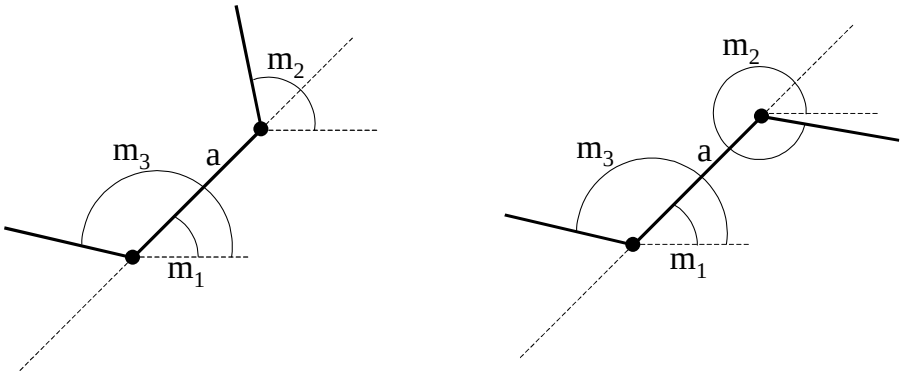
ARISTA CÓNCAVA

2.6.3.2. Forma de calcular la concavidad - convexidad de una arista.

Para averiguar si una arista es cóncava o convexa, lo haremos en 2D puesto que nos resultará más sencillo calcular las pendientes y hacer todas las operaciones necesarias.

Fácilmente, puede demostrarse que si una cara 3D posee una arista cóncava, al proyectar dicha cara sobre un plano, obtenemos un polígono en el que el lado proyección de la arista cóncava es también cóncavo. Ocurrirá otro tanto con cualquier arista convexa de una cara 3D, en cuyo caso, al proyectar sobre un plano, el lado proyección de dicha arista, es un lado convexo del polígono proyección de la cara.

Calcularemos la pendiente m_1 (ángulo de la arista actual). Si este ángulo es mayor de 180º entonces le restaremos 180, para así poder trabajar en el primer o segundo cuadrante.

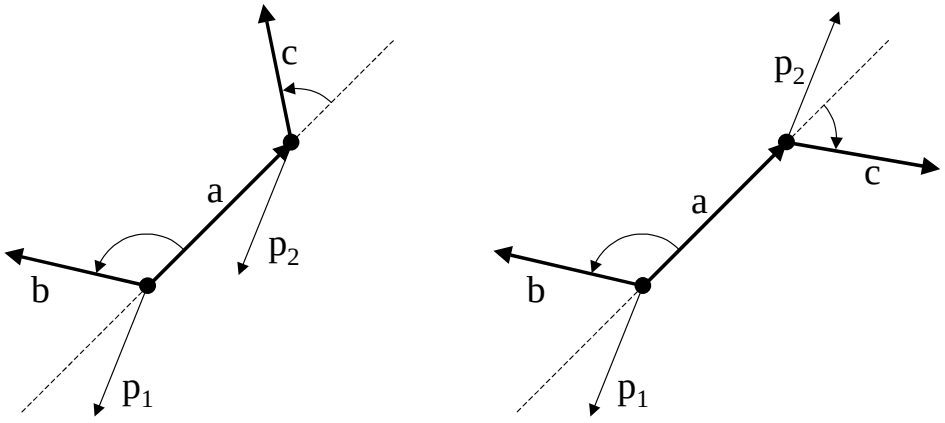


Calculamos las pendientes (los ángulos) de las aristas contiguas m_2 y m_3. Si estos están en el mismo semiplano respecto de la arista actual, entonces la arista será convexa, sino será cóncava.

2.6.3.3. Otra forma de calcular la concavidad - convexidad de una arista.

Esta forma de averiguar si una arista es cóncava o convexa es parecida al método anterior de cálculo, pero aquí nos basamos en los vectores y sus posiciones.

Primero, partimos de una arista *a* de la que queremos saber si es cóncava o convexa. Para ello obtenemos la arista anterior y la arista posterior, y calculamos un vector para cada arista de la forma siguiente:



Suponemos que las aristas a=(v1,v2), b=(v1,v3), c=(v2,v4) forman los vectores va=v2-v1, vb=v3-v1, vc=v4-v2. Ahora calculamos los productos vectoriales p1=va×vb y p2=va×vc y aplicamos la siguiente condición:

Si (p1 y p2 tienen el mismo sentido)
    Entonces la arista *a* es convexa;
    Sino la arista *a* es cóncava;
FinSi

2.6.3.4. Algoritmo de identificación de la relación entre bucles básicos.

Este algoritmo va a realizar una identificación de la relación de inclusión entre dos bucles básicos dentro de un gráfico plano, es decir, a este algoritmo le pasamos como datos de entrada dos bucles básicos, *b1* y *b2*, del mismo gráfico plano, y nos devolverá un valor lógico, que será **verdadero** si el bucle básico *b1* incluye al bucle *b2* y en otro caso será **falso**.

Dentro de este procedimiento decidiremos dos caminos de cálculo de esta relación. Si el bucle básico *b1* es una cara convexa, se aplicará un método de cálculo y si *b1* es una cara cóncava se aplicará otro mucho más complicado. Ahora vamos a ver un esquema sencillo de este algoritmo que desarrollaremos en puntos posteriores:

Función *IdentificaRelacion* **(***b1, b2***) devuelve VALOR_LÓGICO**

VL **variable tipo VALOR_LÓGICO**

Si (EsConvexa(b1)) **Entonces** VL=IncluyeConvexo(b1, b2);
Sino VL=IncluyeConcavo(b1, b2);
FinSi

Devuelve VL;

FinFunción

Hemos definido las funciones siguientes:

- La función *EsConvexa(b)* examina el bucle básico *b* devolviendo **verdadero** si es convexo, y **falso** si es cóncavo.

- La función *IncluyeConvexo(b1, b2)* analiza si el bucle básico convexo *b1* incluye al bucle básico *b2*, devolviendo **verdadero**. En otro caso, devuelve **falso**.

- La función *IncluyeConcavo(b1, b2)* analiza si el bucle básico cóncavo *b1* incluye al bucle básico *b2*, devolviendo **verdadero**. En otro caso, devuelve **falso**.

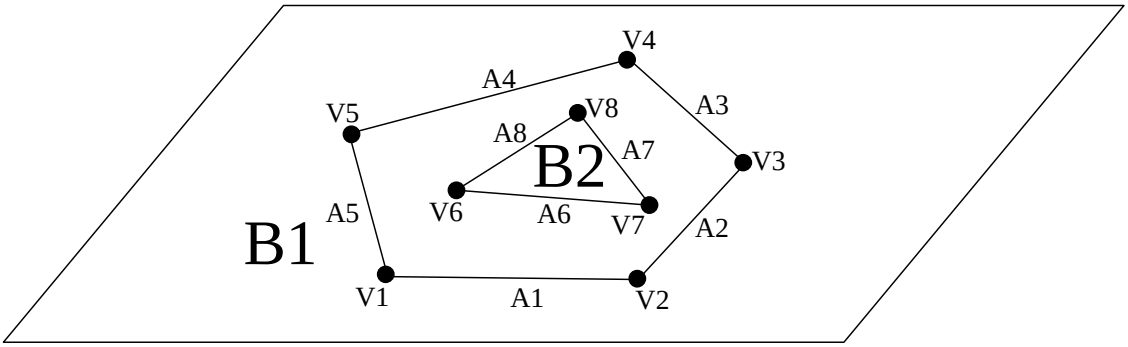
También se ha definido una variable lógica *VL* que almacena el valor devuelto por las funciones de *Incluye____( )*. El valor de esta variable es el que será devuelto por la función *IdentificaRelacion( )*.

A continuación veremos las bases utilizadas para el cálculo de la relación de inclusión entre dos bucles básicos de un gráfico plano dependiendo de la concavidad y convexidad de los bucles básicos. Hay que mencionar especialmente el cálculo de esta relación cuando el bucle básico es cóncavo. Debido a su complejidad hay que descomponer el bucle cóncavo en varios bucles convexos.

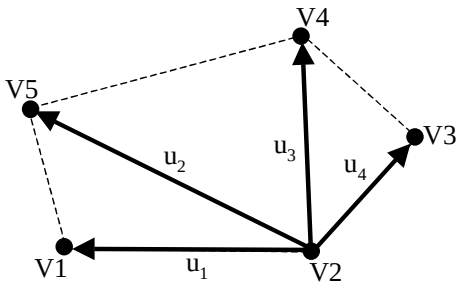
2.6.3.5. Cálculo de la relación de inclusión convexa.

En este apartado, vamos a desarrollar el método utilizado para saber si un bucle básico convexo *b1* incluye a otro bucle básico *b2* cualquiera, dentro del mismo gráfico plano. Hay que decir, que este procedimiento sólo se puede aplicar cuando tenemos la certeza de que el bucle básico *b1* es convexo. Para ver este punto vamos a utilizar unos ejemplos.

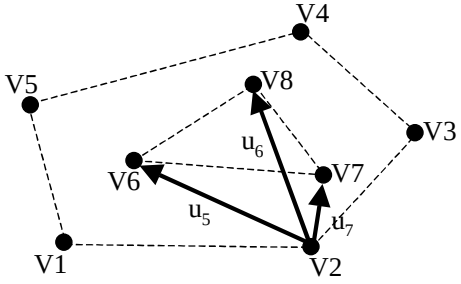
En el primer ejemplo tenemos el siguiente gráfico plano. En él hay dos bucles básicos y los dos son convexos:



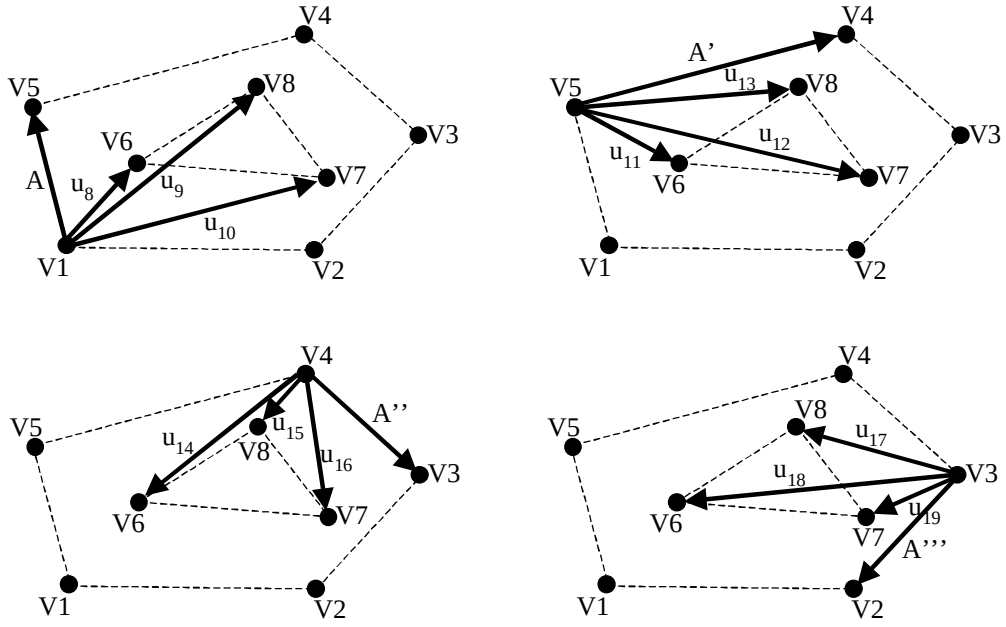
En esta figura tenemos dos bucles básicos: B1 y B2. El bucle básico B1 consta de las siguientes aristas {A1, A2, A3, A4, A5}, mientras que el bucle básico B2 esta formado por las aristas {A6, A7, A8}. Ahora calculamos los vectores siguientes:



Si calculamos los vectores resultantes de los productos vectoriales $u_1 \times u_2$, $u_1 \times u_3$, $u_1 \times u_4$ tenemos que tienen todos el mismo sentido. Si hacemos lo mismo con el resto de vértices de B1, y da también el mismo sentido, nos indica que este bucle básico es convexo. Luego podríamos utilizar este método para saber también cuando un bucle básico es convexo. Ahora pasamos a calcular los vectores siguientes:

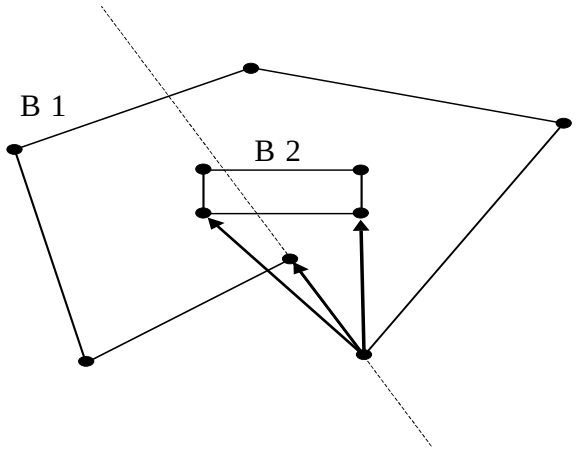


Sabemos que la arista A1 divide este plano en dos semiplanos. Ahora se calculan los vectores producto escalar $u_1 \times u_5$, $u_1 \times u_6$, $u_1 \times u_7$. Si estos productos dan vectores con el mismo sentido que los productos anteriores ($u_1 \times u_2$, $u_1 \times u_3$, $u_1 \times u_4$) tenemos que todos los puntos están en el mismo semiplano. Ahora calculamos los vectores:



Si los productos vectoriales $A \times u_8$, $A \times u_9$, $A \times u_{10}$, $A' \times u_{11}$, $A' \times u_{12}$, $A' \times u_{13}$, $A'' \times u_{14}$, $A'' \times u_{15}$, $A'' \times u_{16}$, $A''' \times u_{17}$, $A''' \times u_{18}$, $A''' \times u_{19}$, nos proporcionan vectores con el mismo sentido que todos los anteriores, podemos decir que el bucle básico B1 incluye al bucle básico B2.

Comprobando este método con un bucle cóncavo:



Podemos comprobar que la arista cóncava no cumple lo especificado de los productos vectoriales. Como se puede ver en la figura anterior, hay productos vectoriales que dan un vector en un sentido, y otros productos vectoriales dan el sentido contrario. Esto es debido a que hay puntos del bucle básico B2 que están en ambos semiplanos generados por la arista cóncava. Por lo tanto, se puede deducir que este método sólo es válido cuando el bucle básico externo es convexo.

2.6.3.6. Cálculo del punto de corte de dos rectas.

En este apartado, vamos a desarrollar el método de cálculo utilizado para obtener el punto de corte entre dos rectas. Esto nos será útil para el punto siguiente que se encarga del cálculo de la relación de inclusión cóncava.

Como vimos en el punto **2.2.2.**, partimos de una recta ***r*** determinada por dos puntos **P***(x₀, y₀, z₀)* y **P'***(x₂, y₂, z₂)* de una arista; y la recta ***s*** determinada por dos puntos **Q***(x₁, y₁, z₁)* y **Q'***(x₃, y₃, z₃)* de la otra arista. Formamos los vectores **v***(v₁, v₂, v₃)* y **u***(u₁, u₂, u₃)* de la forma siguiente:

$$v_1 = x_2 - x_0 \quad u_1 = x_3 - x_1$$
$$v_2 = y_2 - y_0 \quad u_2 = y_3 - y_1$$
$$v_3 = z_2 - z_0 \quad u_3 = z_3 - z_1$$

Sean las matrices:

$$A = \begin{bmatrix} v_1 & v_2 & v_3 \\ u_1 & u_2 & u_3 \end{bmatrix} \qquad B = \begin{bmatrix} v_1 & v_2 & v_3 \\ u_1 & u_2 & u_3 \\ x_1 - x_0 & y_1 - y_0 & y_1 - y_0 \end{bmatrix}$$

Se tiene que si:

5. Rango(A)=Rango(B)=1, las rectas son coincidentes.
6. Rango(A)=1, Rango(B)=2, las rectas son paralelas no coincidentes.
7. Rango(A)=Rango(B)=2, las rectas se cortan en un punto.
8. Rango(A)=2, Rango(B)=3, las rectas se cruzan.

El proceso de cálculo del punto de corte será:

1. Comprobar que el Rango(B)≠3 para ello hay que comprobar que

$$v_1 \cdot u_2 \cdot (z_1 - z_0) + v_3 \cdot u_1 \cdot (y_1 - y_0) + v_2 \cdot u_3 \cdot (x_1 - x_0) - v_3 \cdot u_2 \cdot (x_1 - x_0) - v_1 \cdot u_3 \cdot (y_1 - y_0) - v_2 \cdot u_1 \cdot (z_1 - z_0) = 0$$

2. Si no es cero, tenemos que las dos rectas se cruzan, por lo tanto no hay punto de corte y este proceso finaliza.

3. Comprobar el rango de A, porque para que las rectas se corten Rango(A)= Rango(B)= 2. Cuando se sabe que cumple la condición calcular α y β. Podemos definir las rectas *r* y *s* de la forma siguiente mediante ecuaciones paramétricas:

$$\begin{cases} x = x_0 + \alpha \cdot v_1 \\ y = y_0 + \alpha \cdot v_2 \\ z = z_0 + \alpha \cdot v_3 \end{cases} \quad \begin{cases} x = x_1 + \beta \cdot u_1 \\ y = y_1 + \beta \cdot u_2 \\ z = z_1 + \beta \cdot u_3 \end{cases}$$

Si $\begin{vmatrix} v_1 & v_2 \\ u_1 & u_2 \end{vmatrix} \neq 0$ **Entonces** calcular α y β de la forma:

$$\alpha = \frac{(y_0 - y_1)\cdot u_1 + (x_1 - x_0)\cdot u_2}{u_2 \cdot v_1 - u_1 \cdot v_2}$$
$$\beta = \frac{(y_0 - y_1)\cdot v_1 + (x_1 - x_0)\cdot v_2}{u_2 \cdot v_1 - u_1 \cdot v_2}$$

Sino **Si** $\begin{vmatrix} v_1 & v_3 \\ u_1 & u_3 \end{vmatrix} \neq 0$ **Entonces** calcular α y β de la forma:

$$\alpha = \frac{(z_0 - z_1)\cdot u_1 + (x_1 - x_0)\cdot u_3}{u_3 \cdot v_1 - u_1 \cdot v_3}$$
$$\beta = \frac{(z_0 - z_1)\cdot v_1 + (x_1 - x_0)\cdot v_3}{u_3 \cdot v_1 - u_1 \cdot v_3}$$

Sino **Si** $\begin{vmatrix} v_2 & v_3 \\ u_2 & u_3 \end{vmatrix} \neq 0$ **Entonces** calcular α y β de la forma:

$$\alpha = \frac{(z_0 - z_1)\cdot u_2 + (y_1 - y_0)\cdot u_3}{u_3 \cdot v_2 - u_2 \cdot v_3}$$
$$\beta = \frac{(z_0 - z_1)\cdot v_2 + (y_1 - y_0)\cdot v_3}{u_3 \cdot v_2 - u_2 \cdot v_3}$$

Sino Acaba el proceso porque Rango(A)=1 en cuyo caso las rectas son coincidentes o paralelas.

4. Se obtiene el punto de corte (*x*, *y*, *z*) de las dos formas siguientes:

$$\begin{cases} x = x_0 + \alpha \cdot v_1 \\ y = y_0 + \alpha \cdot v_2 \\ z = z_0 + \alpha \cdot v_3 \end{cases}$$

$$\begin{cases} x = x_1 + \beta \cdot u_1 \\ y = y_1 + \beta \cdot u_2 \\ z = z_1 + \beta \cdot u_3 \end{cases}$$

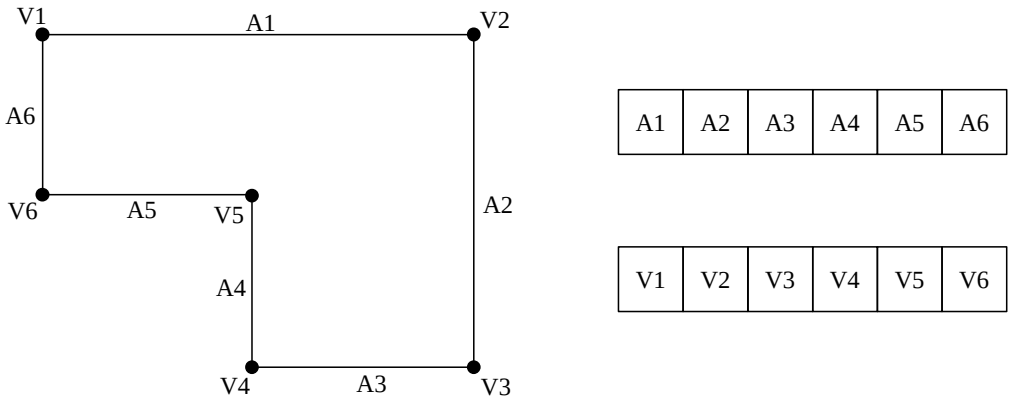
Por lo tanto, aquí tenemos un algoritmo que nos calcula el punto de corte de dos aristas coplanares, sólo cuando estas dos aristas se corten. En el caso de que no se corten las dos aristas devolveríamos un valor significativo, para que el programa sepa que las dos aristas son paralelas, ya que este algoritmo sólo se aplicará entre aristas del mismo plano.

2.6.3.7. Cálculo de la relación de inclusión cóncava.

En este apartado, vamos a desarrollar el método utilizado para saber si un bucle básico cóncavo *b1* incluye a otro bucle básico *b2* cualquiera, dentro del mismo gráfico plano. Hay que decir, que este procedimiento sólo se puede aplicar cuando tenemos la certeza de que el bucle básico *b1* es cóncavo. Para ver este punto vamos a utilizar unos ejemplos.

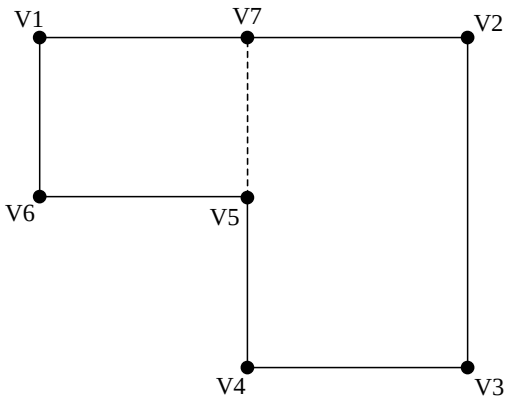
Para resolver este problema se utiliza el viejo método de *"divide y vencerás"* que tantas veces se aplica en el mundo de la programación. Este método nos dice que se puede resolver un problema complejo descomponiéndolo en problemas más sencillos, que se pueden resolver más fácilmente. Aplicando esto a lo que nos ocupa, vamos a tratar de descomponer el bucle básico cóncavo en bucles básicos convexos. Sabremos que el bucle básico cóncavo *b1* incluye al bucle básico *b2*, porque todos los vértices del bucle básico *b2* están dentro de los bucles convexos que componen el bucle básico cóncavo *b1*. Así pues, ahora vamos a ver el método empleado para obtener el conjunto de bucles convexos que componen el bucle básico cóncavo *b1*. Se utilizará primero un ejemplo sencillo y luego veremos un ejemplo más complejo.

Tenemos el bucle básico cóncavo siguiente, con las aristas ordenadas en una lista, y obtenemos la lista ordenada de los vértices:



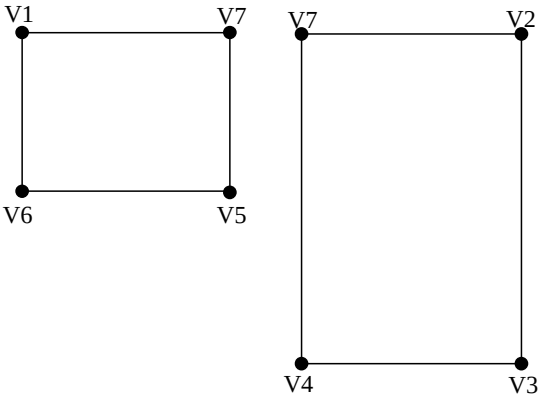
A partir de ahora utilizaremos sólo la lista de vértices, que la supondremos ordenada de forma que entre dos vértices contiguos hay una arista, y entre el primer vértice y el último hay también una arista.

Primero comprobaremos que la arista V1-V2 (primeros dos vértices de la lista de vértices) es convexa, si no es así recorreremos la lista buscando una arista convexa, y ordenando la lista de forma que esta arista convexa sea la primera de la lista. Como la arista V1-V2 es convexa, continuamos con la segunda arista V2-V3. Y seguimos así hasta que encontremos una arista cóncava. En este caso, tenemos que la arista V4-V5 es cóncava. Ahora pasamos a calcular los puntos de corte entre la arista V4-V5 y el resto de aristas. Nos quedaremos con el punto de corte más cercano a la arista V4-V5 que no sean estos vértices (V4, V5). Por lo tanto, en este caso tenemos, como se muestra en la figura siguiente, que el punto de corte V7 entre las aristas V4-V5 y V1-V2 es el punto de corte que cumple estos requisitos, además se añade a la lista ordenada de vértices entre el vértice V1 y el vértice V2, y entre V4 y V5, para que quede ordenada.



V1	V7	V2	V3	V4	V7	V5	V6

Entonces, ahora podemos formar dos bucles básicos convexos. Con la lista de vértices, seleccionamos los vértices comprendidos entre el vértice nuevo, y los extraemos a otra lista de vértices (junto con un vértice nuevo). Ahora, tenemos dos listas de vértices ordenados, que nos proporciona los bucles básicos convexos que forman el bucle básico cóncavo de partida.



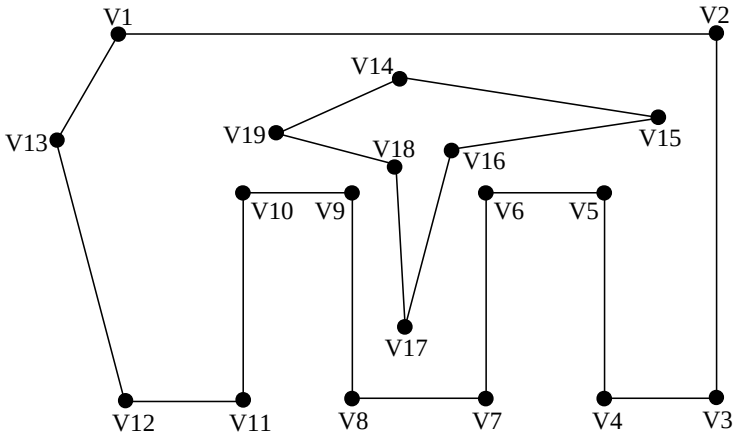
V1	V7	V5	V6

V7	V2	V3	V4

Cuando ya hemos obtenido la división del bucle básico cóncavo en sus bucles básicos convexos, pasamos a realizar la comprobación de inclusión con otro bucle básico. Este cálculo de la relación de inclusión, se haría según lo explicado en el apartado **2.6.3.5.**, pero en este caso, marcamos inicialmente todos los vértices del bucle básico, que vamos a comprobar si es exterior o interior al bucle cóncavo, con la marca *"exterior"*. Los vértices que sean interiores a los bucles básicos convexos serán marcados con la marca "*interior"*, y los vértices que sean exteriores no se modifican las marcas que tienen asignadas.

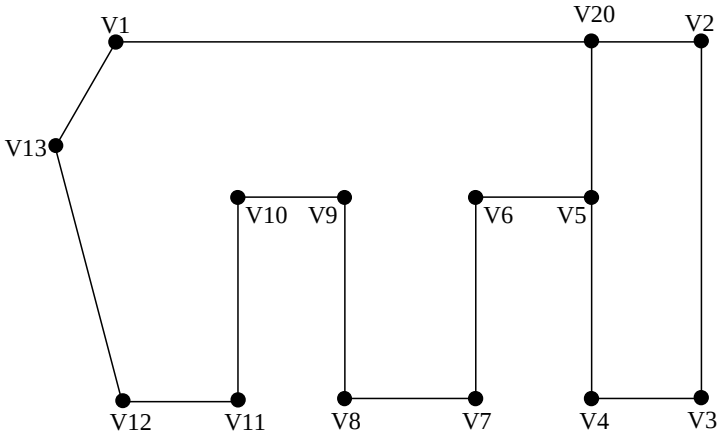
Por lo tanto, el bucle básico cóncavo *b1* incluye al bucle básico *b2* si las marcas de todos los vértices del bucle básico *b2* son iguales a *"interior"*. Si hay una sola marca de vértice igual a *"exterior"*, entonces se puede decir que *b1* no incluye a *b2*.

Ahora vamos a ver un ejemplo más completo y más complejo del cálculo de la relación de inclusión entre un bucle cóncavo *B1* y otro bucle *B2*. En la figura siguiente tenemos la situación inicial de la que partimos cuando vamos a ejecutar el algoritmo de cálculo de inclusión cóncava:

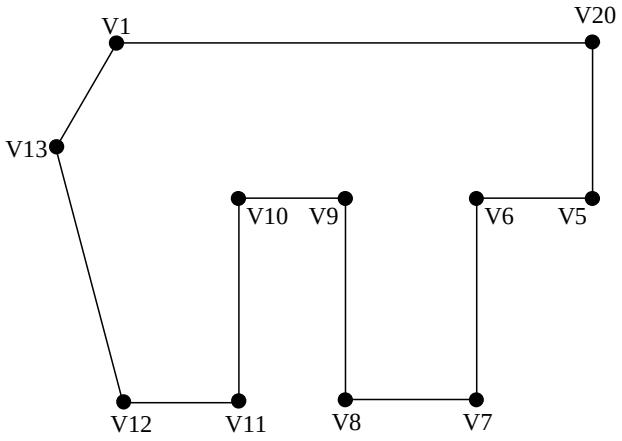


Tenemos que el bucle básico cóncavo *B1* esta formado por la lista de vértices ordenada: {V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13}, y el bucle básico *B2* esta formado por la lista de vértices ordenada: {V14, V15, V16, V17, V18, V19}. Como se puede ver en la figura anterior el bucle *B1* incluye al bucle *B2*. Veamos ahora como se va descomponiendo el bucle básico cóncavo *B1* en distintos bucles convexos.

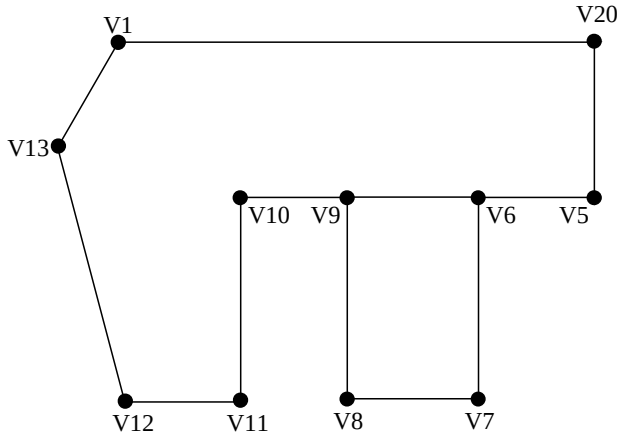
Inicialmente vamos seleccionando las aristas de *B1* y comprobando si son cóncavas o convexas. Si son convexas, pasamos a seleccionar la siguiente hasta que encontramos una cóncava, por lo tanto, las aristas V1-V2, V2-V3, V3-V4 son convexas y no se hará nada con ellas. Pero hemos llegado a la arista cóncava V4-V5. Ahora calcularemos los puntos de corte con las otras aristas del mismo bucle y nos quedaremos con el punto de corte V20 que se ve en la figura siguiente, quedando la lista de vértices: {V1, V20, V2, V3, V4, V20, V5, V6, V7, V8, V9, V10, V11, V12, V13}. Ahora se divide esta lista en dos, la primera es un bucle convexo {V20, V2, V3, V4} y la segunda sigue siendo un bucle cóncavo {V1, V20, V5, V6, V7, V8, V9, V10, V11, V12, V13} por lo tanto hay que seguir dividiendo este bucle.



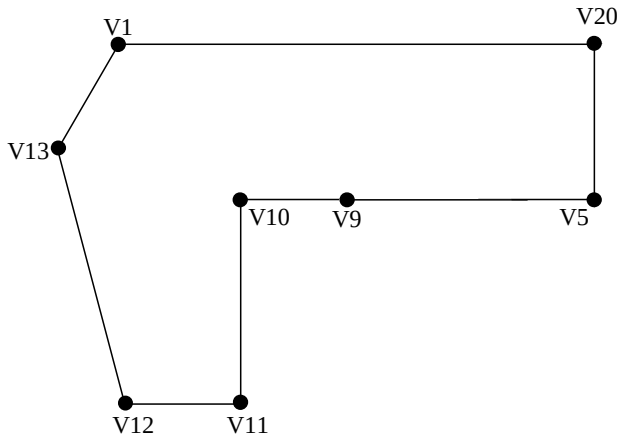
Ahora partimos del bucle cóncavo de la figura siguiente, con la lista de vértices ordenada {V1, V20, V5, V6, V7, V8, V9, V10, V11, V12, V13}, el cual vamos a descomponer en bucles convexos.



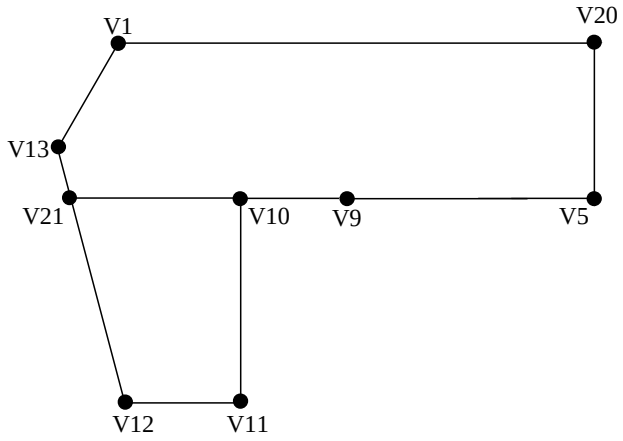
Empezamos a buscar aristas cóncavas siguiendo el orden de la lista de vértices. Vamos probando las aristas V1-V20, V20-V5 que son convexas, y llegamos a la arista V5-V6 que es cóncava. Por lo tanto, se obtienen los puntos de corte con el resto de aristas, y tenemos que el punto de corte correcto es el vértice V9. Por lo tanto añadimos este vértice entre V5-V6, y no hace falta añadirlo entre V8-V9 porque ya lo tenemos. Tenemos la lista de vértices {V1, V20, V5, V9, V6, V7, V8, V9, V10, V11, V12, V13} que dividimos en dos listas formando dos nuevos bucles: el primero {V9, V6, V7, V8} que es convexo, y el segundo {V1, V20, V5, V9, V10, V11, V12, V13} que sigue siendo cóncavo. Veamos los dos nuevos bucles en la figura siguiente:



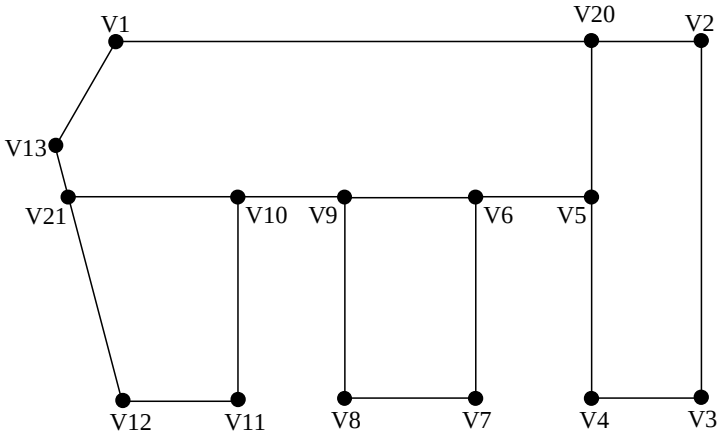
Entonces, ahora partiremos de este nuevo bucle cóncavo que hemos formado. Este bucle viene definido por la lista de vértices ordenada {V1, V20, V5, V9, V10, V11, V12, V13}. Como vemos en la figura siguiente, tenemos dos aristas alineadas que son V5-V9 y V9-V10. Para el cálculo de concavidad – convexidad de aristas, haremos lo siguiente, si tenemos varias aristas alineadas, según el sentido que hayamos tomado, las primeras aristas alineadas serán convexas, y la última alineada dependerá de las aristas anterior y posterior del conjunto de aristas alineadas. Por lo tanto la arista V5-V9 será convexa, y la arista V9-V10 será cóncava. Pasemos a ver en la figura siguiente el bucle básico cóncavo:



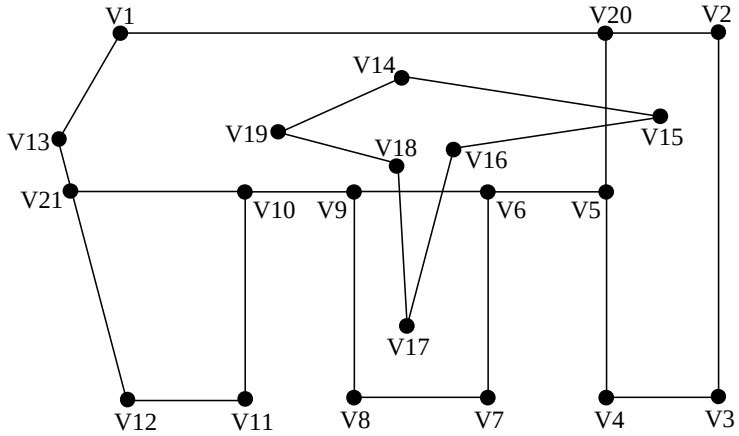
Ahora empezamos el recorrido de la lista de vértices en busca de aristas cóncavas. Tenemos que la primera arista cóncava es la arista V9-V10. Después calculamos los cortes con otras aristas, obteniendo el punto de corte V21, que se ve en la figura siguiente. Por lo tanto, tenemos que insertar el vértice V21 entre los vértices V9-V10 y entre V12-V13. Por lo que nos queda la lista de vértices: {V1, V20, V5, V9, V21, V10, V11, V12, V21, V13}. Que dividiendo esta lista de vértices obtenemos los bucles: {V1, V20, V5, V9, V21, V13} y {V21, V10, V11, V12}, siendo convexos estos dos nuevos bucles como se puede ver en la figura siguiente.



En la figura siguiente vemos todos los bucles convexos que hemos obtenido a partir del bucle inicial cóncavo:



Después de haber obtenido la descomposición del bucle básico cóncavo *B1* en varios bucles convexos, hay que comprobar si el bucle básico *B2* está incluido en el bucle básico *B1*. Para ello, lo que hacemos es analizar las posiciones de todos los vértices del bucle básico *B2*. Cada vértice de *B2* tiene que estar dentro de algún bucle convexo de los que forman *B1*, si hay algún vértice que cae fuera de todos los bucles convexos de *B1*, entonces quiere decir que este vértice cae fuera del bucle básico cóncavo *B1*. En la figura siguiente se puede comprobar que todos los vértices del bucle básico *B2* están dentro de los bucles convexos de *B1*, por lo tanto el bucle básico cóncavo *B1* incluye al bucle básico *B2*.



Tenemos que el bucle básico cóncavo *B1*={V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13} se ha dividido en cuatro bucles convexos que son: *B1.1*={V1, V20, V5, V9, V21, V13}, *B1.2*={V20, V2, V3, V4}, *B1.3*={V9, V6, V7, V8}, *B1.4*={V21, V10, V11, V12}. El bucle básico *B2* está formado por los vértices {V14, V15, V16, V17, V18, V19}. Como se puede ver en la figura anterior, los vértices V14, V16, V18, V19 están incluidos en el bucle convexo *B1.1*, el vértice V15 está incluido en el bucle convexo *B1.2*, y el vértice V17 está incluido en el vértice *B1.3*. De aquí se deduce que el bucle básico cóncavo *B1* incluye al bucle básico *B2*.

2.6.4. FORMACIÓN DE BUCLES DE CARAS.

Este procedimiento es la quinta parte del algoritmo *FLG*. Partiendo de las relaciones de inclusión entre todos los bucles básicos, dentro de un gráfico plano, obtenidas en el punto **2.6.3.** se van a generar los bucles de caras definitivos. Este va a ser el último punto dentro del algoritmo de generación de bucles de caras *FLG*.

Veamos ahora lo que dice el artículo de Qing-Wen Yan acerca del punto que estamos tratando ahora:

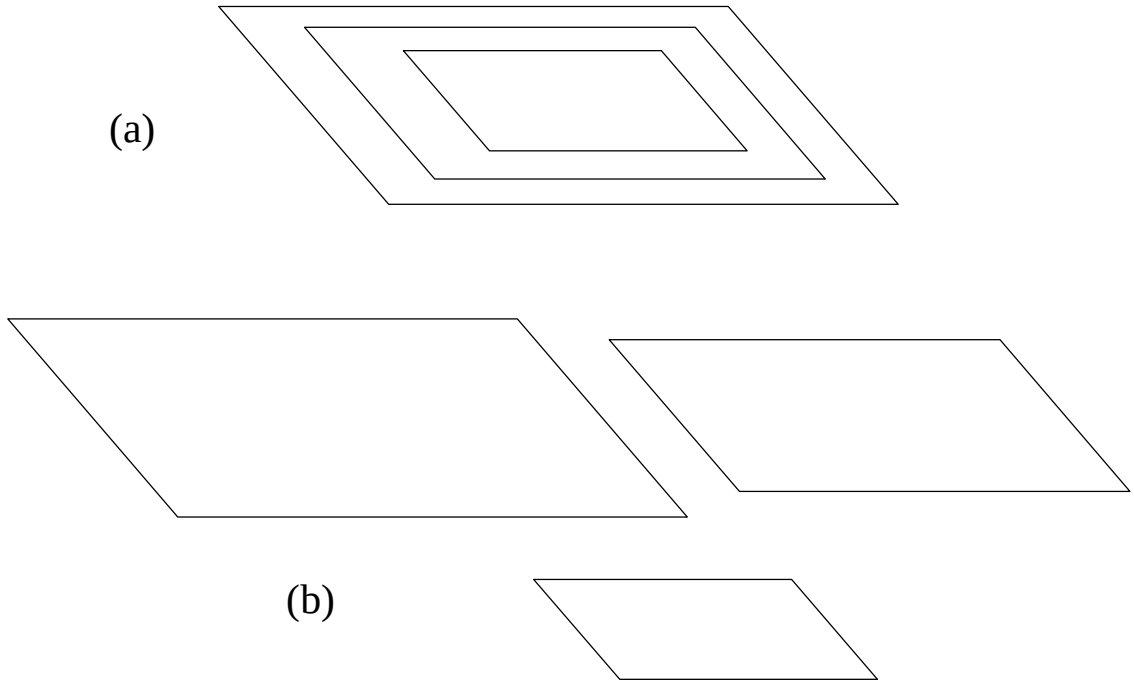
"Dada la relación de inclusión entre bucles básicos:

Si *L_i no incluye algún bucle, entonces L_i forma un bucle de cara;*
Sino ***Si*** *L_i incluye a L_j, entonces L_i es un bucle exterior, L_j es un bucle interior y L_i incluye a L_j directamente.*

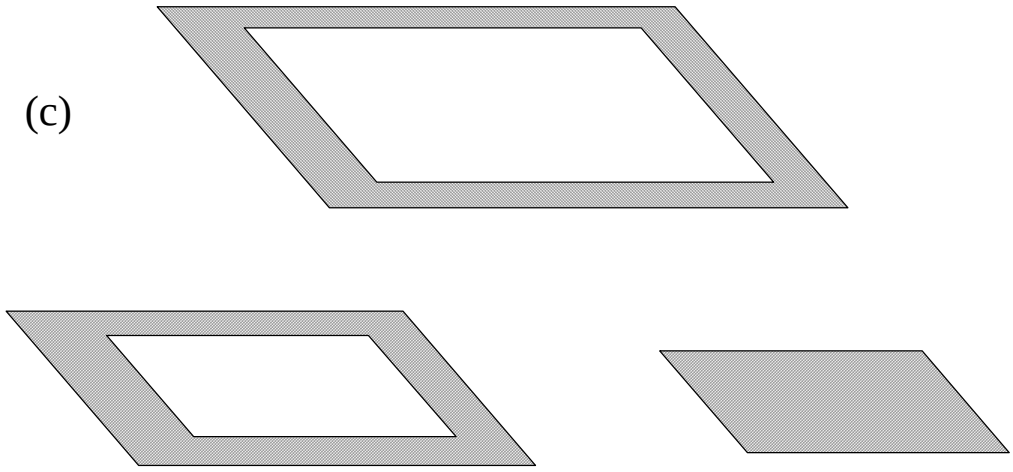
Sino Si *el bucle L_i incluye más de un bucle, por ejemplo, L_{j1}, L_{j2}, ... , L_{jn}, verificar cuando los L_j son incluidos por otros bucles básicos.*

Un bucle de cara se forma a partir de L_i y bucles básicos incluidos por L_i directamente. En este bucle de cara L_i es un bucle exterior, y los otros son bucles básicos interiores."

El párrafo anterior lo que nos viene a decir es que formarán bucles de caras: los bucles básicos que no incluyan a ningún bucle básico, y los bucles básicos que sólo tengan un nivel de anidamiento de bucles básicos interiores a él. En la figura siguiente podemos ver un ejemplo de un gráfico plano, sus bucles básicos, y los bucles de caras generados.



(a) Gráfico plano; (b) Bucles básicos.



(c) Bucles de caras formados.

Hemos visto como se forman los bucles de caras a partir de las relaciones de inclusión, pero antes de acabar la ejecución del algoritmo de generación de bucles de caras *FLG*, hay que ejecutar el procedimiento *PEVR* (eliminación de aristas y vértices patológicos) para eliminar la posible aparición de aristas redundantes.

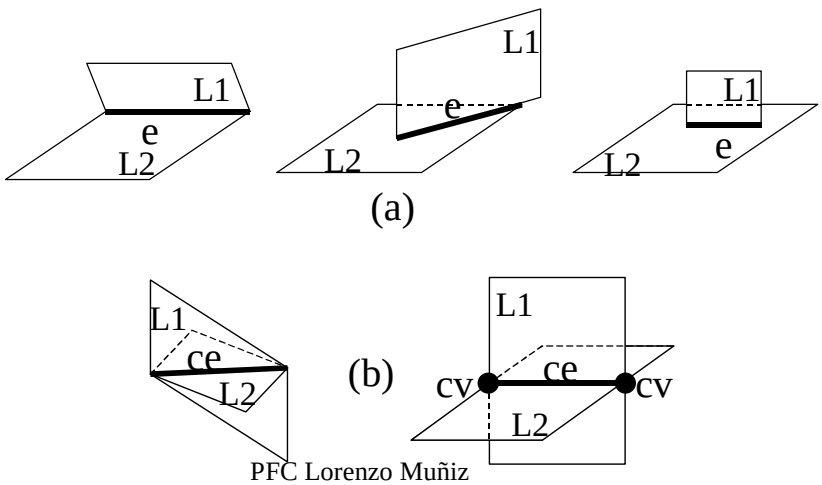
Después de haber visto el punto **2.6.**, ya sabemos como se generan todas las caras de un objeto. Estas caras nos serán útiles en la generación de los bucles de cuerpos, ya que se van a ir generando dependiendo de las caras adyacentes a una dada. Pero antes de pasar a la generación de bucles de cuerpos hay que realizar una comprobación. Hay que verificar si hay bucles de caras que se cortan, si es así, hay que generar las aristas y vértices de corte apropiados, y dividir los bucles de caras dependiendo de las aristas de corte. Éste va a ser el siguiente paso del algoritmo que vamos a ver a continuación.

2.7. OBTENCIÓN DE VÉRTICES Y ARISTAS DE CORTE.

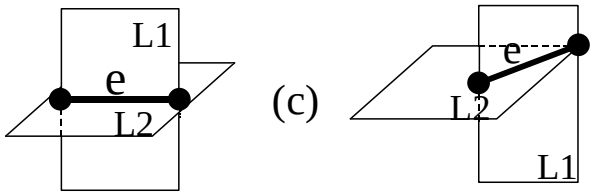
En la sección anterior fueron generados todos los bucles de caras recorriendo los gráficos planos mediante el algoritmo *FLG*. Los bucles de caras de diferentes planos se pueden intersectar, creando vértices y aristas. A estos vértices y aristas los llamamos vértices y aristas de corte. Con los vértices y aristas de corte se pueden descubrir más bucles de caras de diferentes planos. Llamaremos a partir de ahora a este procedimiento como *CEV* (*Cutting Edges and Vertices*, Vértices y Aristas de Corte), el cual determina todos los vértices y aristas de corte entre varios bucles de caras, generando la división apropiada de estos bucles de caras, en nuevos bucles de caras. Veamos lo que nos dice Qing-Wen Yan en su artículo sobre esto:

"Dados dos bucles de caras no coplanares, la línea de intersección entre ellos se puede clasificar en los siguientes tres casos:

- ***Caso 1**: La línea de intersección es una arista de un bucle de cara, figura (a).*
- ***Caso 2**: La línea de intersección entre dos bucles de caras y sus dos puntos finales están en la frontera de los bucles de caras, figura (b).*
- ***Caso 3**: La línea de intersección entre dos bucles de caras, pero no como en el segundo caso, figura (c).*



La intersección de bucles de caras en el caso 1 es regular. En el caso 2, las aristas/vértices de corte son necesarios, mientras en el caso 3 son patológicos. Como resultado, en el caso 3, el bucle de cara que incluye ambos puntos finales de la línea de



intersección será eliminado."

Es necesario mencionar que las aristas y vértices de corte no existen realmente en el objeto. Se presentan sólo para asistir en la correcta generación de objetos. Además las aristas y vértices de corte serán eliminados cuando el objeto final sea construido.

Sin entrar mucho en detalles, a continuación se muestra en lenguaje natural el método empleado para la resolución de este apartado. El método sería el siguiente:

1. Comparar dos bucles de caras no coplanares.

2. Si la intersección es una arista que forma parte de alguno de los bucles de caras del objeto, pasar al punto 5.

3. Si los vértices de la arista de corte (de los dos planos comparados) están en la frontera de los dos bucles de caras, entonces generar los vértices de corte y la arista de corte; dividir los dos bucles de caras, generando nuevos bucles de caras.

4. Si hay intersección y no es de los casos anteriores, entonces eliminar el bucle de cara que contiene los puntos finales de la línea de intersección.

5. Si no quedan pares de bucles de caras no coplanares a comparar, acabar el algoritmo. En otro caso, seleccionar dos bucles de caras no coplanares y pasar al punto 1.

Hay tres procedimientos empleados en el algoritmo anterior que deben ser mejor explicados. El primero es la comprobación de que dos bucles de caras sean no coplanares, que ha sido desarrollado en el punto **2.4.1.2.**. Los otros dos procedimientos, descubrir la recta intersección de dos bucles de caras (planos) y la división de los bucles de caras se explicarán a continuación.

2.7.1. CÁLCULO DE LA RECTA INTERSECCIÓN ENTRE DOS PLANOS.

Tenemos dos planos no coplanares π y π', y queremos encontrar la recta *r* que es la intersección entre esos dos planos. Estos planos vienen definidos por las ecuaciones:

$$\pi \equiv Ax + By + Cz + D = 0$$
$$\pi' \equiv A'x + B'y + C'z + D' = 0$$

Además también definiremos la recta intersección *r* mediante un vector *v* y un punto *p*, dependiendo de las condiciones que se muestran a continuación:

$$\textbf{Si } \begin{vmatrix} A & B \\ A' & B' \end{vmatrix} \neq 0 \textbf{ Entonces } r \equiv \begin{cases} x = \alpha \cdot z + p \\ y = \beta \cdot z + q \\ z = z \end{cases} \quad \begin{matrix} v = (\alpha, \beta, 1) \\ p = (p, q, 0) \end{matrix} \qquad (1)$$

$$\textbf{Sino Si } \begin{vmatrix} A & C \\ A' & C' \end{vmatrix} \neq 0 \textbf{ Entonces } r \equiv \begin{cases} x = \alpha \cdot y + p \\ y = y \\ z = \beta \cdot y + q \end{cases} \quad \begin{matrix} v = (\alpha, 1, \beta) \\ p = (p, 0, q) \end{matrix} \qquad (2)$$

$$\textbf{Sino Si } \begin{vmatrix} B & C \\ B' & C' \end{vmatrix} \neq 0 \textbf{ Entonces } r \equiv \begin{cases} x = x \\ y = \alpha \cdot x + p \\ z = \beta \cdot x + q \end{cases} \quad \begin{matrix} v = (1, \alpha, \beta) \\ p = (0, p, q) \end{matrix} \qquad (3)$$

Para el caso (1) tenemos que resolver el sistema de ecuaciones formado por las ecuaciones de los planos obteniendo valores de *x* e *y* en función de la variable *z*, para obtener los valores α, β, *p*, *q*. Como se ve a continuación:

$$\left.\begin{matrix} Ax + By + Cz + D = 0 \\ -\dfrac{B}{B'}(A'x + B'y + C'z + D' = 0) \end{matrix}\right\} \qquad \begin{matrix} Ax + By + Cz + D = 0 \\ -\dfrac{A'B}{B'}x - By - \dfrac{C'B}{B'}z - \dfrac{D'B}{B'} = 0 \end{matrix}$$

Sumando las dos ecuaciones anteriores, tenemos:

$$\frac{AB' - A'B}{B'}x + \frac{CB' - C'B}{B'}z + \frac{DB' - D'B}{B'} = 0 \Rightarrow (AB' - A'B)x + (CB' - C'B)z + DB' - D'B = 0$$

Despejando la variable *x* obtenemos:

$$x = \frac{(C'B - CB')z + D'B - DB'}{B'A - A'B}$$

Los valores de α y *p* serán:

$$\boxed{\alpha = \frac{C'B - CB'}{B'A - BA'} \qquad p = \frac{D'B - DB'}{B'A - BA'}}$$

Ahora realizamos un cálculo similar para obtener *y* en función de *z*, de forma:

$$\left.\begin{array}{l} Ax+By+Cz+D=0 \\ -\dfrac{A}{A'}(A'x+B'y+C'z+D'=0) \end{array}\right\} \qquad Ax+By+Cz+D=0$$
$$-Ax-\frac{B'A}{A'}y-\frac{C'A}{A'}z-\frac{D'A}{A'}=0$$

Sumando las dos ecuaciones anteriores, tenemos:

$$(BA'-B'A)y+(CA'-C'A)z+DA'-D'A=0$$

Despejando la variable *y* obtenemos:

$$y=\frac{(C'A-CA')z+D'A-DA'}{BA'-AB'}$$

Y los valores de β y *q* serán:

$$\boxed{\beta=\frac{C'A-CA'}{BA'-B'A} \qquad q=\frac{D'A-DA'}{BA'-B'A}}$$

Para el caso (2) tenemos que resolver el sistema de ecuaciones formado por las ecuaciones de los planos obteniendo valores de *x* y *z* en función de la variable *y*, para obtener los valores α, β, *p*, *q*. Estos valores serán:

$$\boxed{\alpha=\frac{B'C-BC'}{C'A-CA'} \qquad p=\frac{D'C-DC'}{C'A-CA'}}$$

$$\boxed{\beta=\frac{B'A-BA'}{A'C-AC'} \qquad q=\frac{D'A-DA'}{A'C-AC'}}$$

Y para el caso (3) tenemos que resolver el sistema de ecuaciones formado por las ecuaciones de los planos obteniendo valores de *y* y *z* en función de la variable *x*, para obtener los valores α, β, *p*, *q*. Estos valores serán:

$$\boxed{\alpha=\frac{A'C-AC'}{BC'-B'C} \qquad p=\frac{D'C-DC'}{BC'-B'C}}$$

$$\boxed{\beta=\frac{A'B-AB'}{CB'-C'B} \qquad q=\frac{D'B-DB'}{CB'-C'B}}$$

Resumiendo, tenemos que:

$$a=AB'-A'B \quad c=AD'-A'D \quad e=BD'-B'D$$
$$b=AC'-A'C \quad d=BC'-B'C \quad f=CD'-C'D$$

- Caso (1):..... $\alpha=\dfrac{d}{a} \quad \beta=\dfrac{b}{-a} \quad p=\dfrac{e}{a} \quad q=\dfrac{c}{-a}$

- Caso (2):..... $\alpha = \frac{-d}{b} \quad \beta = \frac{a}{-b} \quad p = \frac{f}{b} \quad q = \frac{c}{-b}$

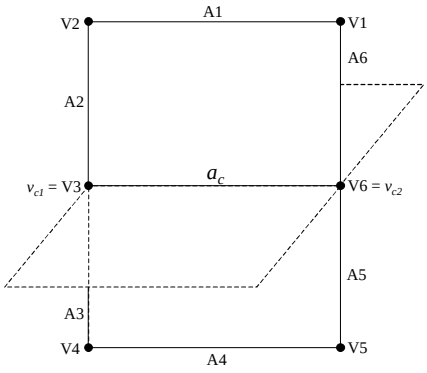
- Caso (3):..... $\alpha = \frac{-b}{d} \quad \beta = \frac{-a}{-d} \quad p = \frac{f}{d} \quad q = \frac{e}{-d}$

Por lo tanto, ahora sabemos un método sencillo de obtener la recta intersección entre dos planos no coplanares, sólo tenemos que saber las ecuaciones de estos dos planos.

2.7.2. DIVISIÓN DE BUCLES DE CARAS.

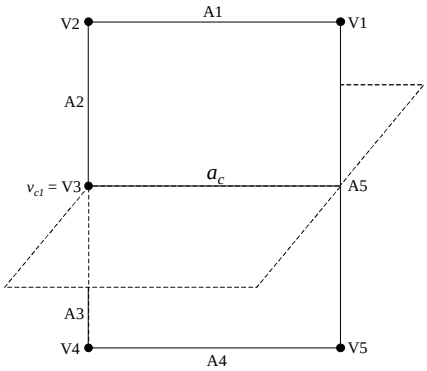
Dados dos bucles de caras no coplanares de un objeto 3D que se cortan en una arista de corte a_c (formada por dos vértices de corte v_{c1} y v_{c2}), tenemos tres casos posibles a contemplar:

- Caso (1): Los vértices de corte existen en el bucle de cara, por lo tanto no hay que dividir ninguna arista.



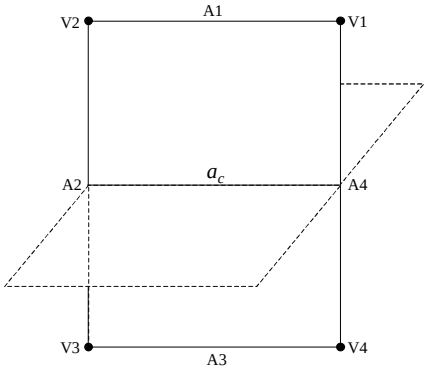
Como podemos ver en la figura anterior, los dos vértices de corte (v_{c1}, v_{c2}) están ya incluidos en el bucle de cara (V3, V6), por lo tanto ya existían, y no hace falta dividir ninguna arista porque no hay ningún vértice de corte que caiga en medio de una arista.

- Caso (2): Existe sólo un vértice de corte en el bucle de cara, por lo tanto sólo hay que dividir una arista en dos por medio del otro vértice de corte.



Como podemos ver en la figura anterior, el vértice de corte v_{c1} está ya incluido en el bucle de cara (V3), por lo tanto ya existe, pero nos damos cuenta que el vértice de corte v_{c2} no existe, por lo tanto tenemos que crearlo y dividir la arista sobre la que recae este vértice, en este caso sería la arista A5. Esta división requiere la eliminación de la arista A5 del bucle de cara, y la adición de dos nuevas aristas, que serían (V1, v_{c2}) y (v_{c2}, V5).

- Caso (3): No existe ningún vértice de corte en el bucle de cara, por lo tanto hay que dividir dos aristas por medio de los vértices de corte.



Como podemos ver en la figura anterior, los vértices de corte v_{c1} y v_{c2} no existen, por lo tanto tenemos que crearlos y dividir las aristas sobre las que recaen estos vértices, en este caso serían las aristas A2 y A4. Esta división requiere la eliminación de las aristas A2 y A4 del bucle de cara, y la adición de cuatro nuevas aristas, que serían (V2, v_{c1}), (v_{c1}, V3), (V1, v_{c2}) y (v_{c2}, V4).

Después de la división de las aristas, dependiendo del caso de entre los anteriores en el que nos encontremos, tendremos que pasar a la división del bucle de cara en dos nuevos bucles de caras. El método utilizado para llevar a cabo este propósito se va a exponer a continuación.

Vamos a definir dos listas de aristas, en las cuales, el primer valor será el número de aristas que contiene la lista, y los siguientes valores serán las propias aristas. Inicialmente, asignamos a la primera lista las aristas que componen el bucle de cara que

queremos dividir y a la segunda lista estará vacía. Veamos ahora como estarían inicializadas las listas para los tres casos anteriores:

- Caso (1): L1={6, A1, A2, A3, A4, A5, A6}, L2=∅.
- Caso (2): L1={5, A1, A2, A3, A4, A5}, L2=∅.
- Caso (3): L1={4, A1, A2, A3, A4}, L2=∅.

Ahora vamos a proceder a la descomposición de las aristas que tengan que ser divididas porque un vértice de corte caiga entre sus dos vértices. Como se puede ver en las figuras anteriores, las listas de aristas quedarían de la siguiente forma:

- Caso (1): L1={6, A1, A2, A3, A4, A5, A6}, L2=∅.
- Caso (2): L1={6, A1, A2, A3, A4, A5', A5''}, L2=∅.
- Caso (3): L1={6, A1, A2', A2'', A3, A4', A4''}, L2=∅.

Como se puede observar, para el caso (2) se ha dividido la arista A5 en las aristas A5' y A5'', para el caso (3) se han dividido las aristas A2 y A4 en las aristas A2', A2'' y A4', A4'', y para el caso (1) no se divide ninguna arista.

Ahora se va a proceder a insertar la arista de corte en las listas de aristas. Insertaremos la arista de corte a_c entre cada par de aristas de L1 que sean adyacentes al mismo vértice de corte. Por lo tanto las listas de aristas dependiendo de cada caso quedarán de la forma siguiente:

- Caso (1): L1={8, A1, A2, a_c, A3, A4, A5, a_c, A6}, L2=∅.
- Caso (2): L1={8, A1, A2, a_c, A3, A4, A5', a_c, A5''}, L2=∅.
- Caso (3): L1={8, A1, A2', a_c, A2'', A3, A4', a_c, A4''}, L2=∅.

Entonces ahora pasamos a dividir las la lista L1, formando dos bucles de caras. Guardaremos en la lista L2 las aristas comprendidas entre el primer a_c y el segundo a_c, guardando también en L2 un único a_c. En la lista L1 guardaremos las aristas que estén entre la primera arista de la lista L1 y el primer a_c (incluido), y las aristas que estén a partir del segundo a_c (no incluido). Por lo tanto, las listas de aristas quedarán así:

- Caso (1): L1={4, A1, A2, a_c, A6}, L2={4, A3, A4, A5, a_c}.
- Caso (2): L1={4, A1, A2, a_c, A5''}, L2={4, A3, A4, A5', a_c}.
- Caso (3): L1={4, A1, A2', a_c, A4''}, L2={4, A2'', A3, A4', a_c}.

Por lo tanto hemos obtenido en las listas L1 y L2 los dos bucles de caras fruto de la división de un bucle de cara mediante una arista de corte.

Visto este apartado, ya sabemos como dividir los bucles de caras no coplanares que se cortan en el espacio. Ahora ya tenemos todos los bucles de caras que se podían generar para pasar al punto siguiente, que es la generación de los bucles de cuerpos. Al igual que se han utilizado las aristas adyacentes para obtener un conjunto de bucles de caras, se va a utilizar un método similar básicamente, pero mucho más complejo, para obtener los bucles de cuerpos mediante los bucles de caras adyacentes.

2.8. GENERACIÓN DE CUERPOS ELEMENTALES.

En las dos secciones anteriores se han generado todos los bucles de caras recorriendo los gráficos planos mediante el algoritmo *FLG* y dividiendo los bucles de caras de diferentes planos que se intersectan mediante vértices y aristas de corte con el algoritmo *CEV*. Ahora en este punto vamos a tratar la generación de cuerpos elementales o bucles de cuerpos a partir de los bucles de caras. Llamaremos cuerpos elementales o bucles de cuerpos a unos objetos elementales 3D delimitados mediante caras o bucles de caras. Los bucles de cuerpos pueden ser vistos como subobjetos que no tienen puntos interiores comunes, pueden compartir vértices, aristas o caras. Llamaremos a partir de ahora a este procedimiento como *BLG* (*Body Loop Generation*, Generación de Bucles de Cuerpos). Veamos ahora lo que dice el artículo de Qing-Wen Yan acerca del punto que estamos tratando:

"Algoritmo BLG:

Dados los bucles de caras F_1, F_2, ... , F_m , el algoritmo *BLG* descubre todos los bucles de cuerpos.

(B1) [Inicialización].

Un bucle de cara F_i tiene dos lados. El lado del vector normal exterior es el lado positivo, denotado como $+F_i$, mientras que el otro es el negativo, $-F_i$. Marcar cada lado de F_i como 'sin usar' poniendo $\chi(i, +) \leftarrow 0$, $\chi(i, -) \leftarrow 0$, $1 \leq i \leq m$. Inicializar el conjunto de bucles de caras $S(F) \leftarrow \varnothing$.

(B2) [Seleccionar una cara de inicio y su lado].

Seleccionar un F_j como bucle de cara inicial para empezar la búsqueda de bucles de cuerpos. Si $\chi(j, +)=0$, entonces $S(F) \leftarrow S(F) \cup \{+F_j\}$ y $\chi(j, +) \leftarrow 1$, y marcar $+F_j$ en S(F) como 'sin expandir'; ir al paso B3. Si $\chi(j, -)=0$, entonces $S(F) \leftarrow S(F) \cup \{-F_j\}$ y $\chi(j, -) \leftarrow 1$, y marcar $-F_j$ en S(F) como 'sin expandir'.

(B3) [Seleccionar un bucle de cara 'sin expandir' $\oplus F_K$ ($\oplus$ puede ser + ó -) de S(F), y computar los sucesivos bucles de caras de cada arista en la frontera de $\oplus F_K$].

Los bucles de caras adyacentes a la arista *e* en F_K son F_1, F_2, ... , F_n, $n \geq 2$. Los sucesivos bucles de caras de $\oplus F_K$ en la arista *e* es $\oplus F_S$, $1 \leq s \leq n$ y $s \neq k$, F_S necesita el ángulo de rotación más pequeño para coincidir con $\oplus F_K$. α será el ángulo de rotación entre $\oplus F_K$ y $\oplus F_S$. n_K y n_S serán los hipotéticos vectores normales de F_K y F_S, respectivamente. θ será el ángulo entre n_K y n_S. Entonces, tenemos

$$\theta = \cos^{-1}\left[\frac{n_S \cdot n_K}{|n_S| \cdot |n_K|}\right]$$

El vector unitario **e** será un vector a lo largo de la arista *e*, y satisface la regla de la mano derecha con n_K. Si $\oplus F_K$ es $+F_K$ su bucle de cara sucesivo a la arista *e* puede ser computado con el siguiente criterio. Obtenemos los valores de α y de selección del lado $\oplus$ para cada bucle de cara adyacente en la arista *e* de modo que elegimos el bucle de cara con el menor valor de α.

- Si $n_S \times n_K$ está en la misma dirección que **e**, y $+F_S$ está a la derecha de **e**, entonces $\alpha = \pi - \theta$; asignar $+F_S$ a $\oplus F_S$ (ver figura 12a).
- Si $n_S \times n_K$ está en la dirección contraria a **e**, y $+F_S$ está a la izquierda de **e**, entonces $\alpha = \theta$; asignar $-F_S$ a $\oplus F_S$ (ver figura 12b).
- Si $n_S \times n_K$ está en la misma dirección que **e**, y $+F_S$ está a la izquierda de **e**, entonces $\alpha = 2\pi - \theta$; asignar $-F_S$ a $\oplus F_S$ (ver figura 12c).
- Si $n_S \times n_K$ está en la dirección contraria a **e**, y $+F_S$ está a la derecha de **e**, entonces $\alpha = \pi + \theta$; asignar $+F_S$ a $\oplus F_S$ (ver figura 12d).

Si $+F_S \in S(F)$, entonces $S(F) \leftarrow S(F) \cup \{+F_S\}$ y $\chi(s, +) \leftarrow 1$. Marcar $+F_S$ como 'sin expandir' en S(F). Este procedimiento de selección del lado garantiza que todos los bucles de caras en S(F) que forman las caras interiores y exteriores del bucle serán formados. Cuando el bucle de cara sucesivo de F_K en cada arista sobre F_K ha sido determinado usando el criterio anterior, marcar $+F_K$ como 'expandido'. Ir al paso B4. Similarmente, si $\oplus F_K$ es $-F_K$, su sucesivo bucle de cara en la arista *e* puede ser determinado de acuerdo con esto; ir al paso B4. Como un ejemplo, los vectores normales de los bucles de caras adyacentes F_1, F_5 en la figura 13a se muestran en la figura 13b. Las equivalencias de bucles de caras sucesivos son $+F_1$ y $-F_2$, $-F_1$ y $-F_5$, $+F_4$ y $-F_3$, y $-F_4$ y $+F_5$.

(B4) Repetir el paso B3 hasta que no haya nuevos bucles de caras para ser expandidos en S(F). Los bucles de caras en S(F) pueden formar un bucle de cuerpo. Ir al paso B5.

(B5) [Verificar la corrección del bucle de cuerpo].

Un bucle de cuerpo debe estar cerrado por los bucles de caras, sin caras colgantes. Así para el bucle de cuerpo actual en S(F), si cada arista de este bucle de cuerpo es compartida por dos y sólo dos bucles de caras, entonces los bucles de caras en el actual S(F) forman un bucle de cuerpo válido. En otro caso, los bucles de caras en S(F) no pueden formar un cuerpo cerrado; ir al paso B6 para iniciar otra búsqueda.

(B6) Poner $S(F) \leftarrow \varnothing$, y repetir los pasos B1-B5 hasta que cada bucle de cara F_i, $1 \leq i \leq m$, haya sido visitado en sus dos lados.

Fin del algoritmo BLG"

Las figuras que a continuación se muestran son las figuras a las que se hace referencia en el artículo de Qing-Wen Yan, se trata de las figuras 12 (a), (b), (c), (d) y las figuras 13 (a) y (b). En estas figuras se muestran ejemplos de los casos posibles de

bucles de caras adyacentes (figura 12) y un ejemplo de varios bucles de caras adyacentes a una arista.

Figura 12:

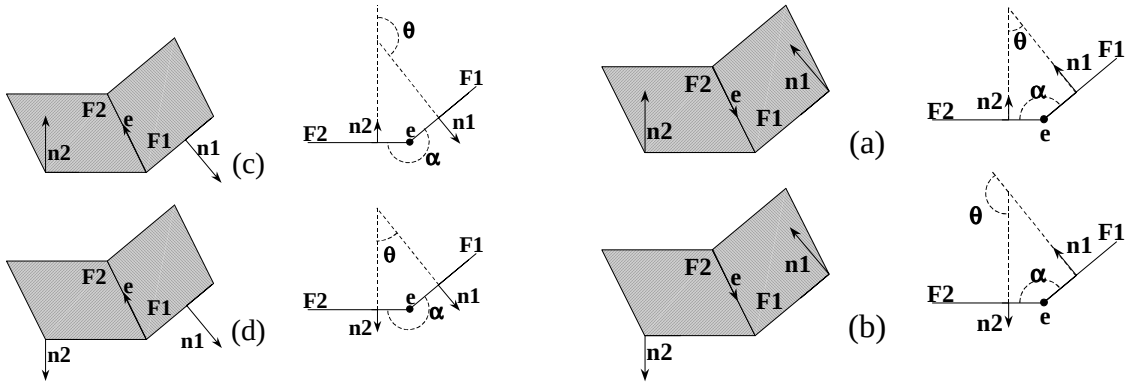
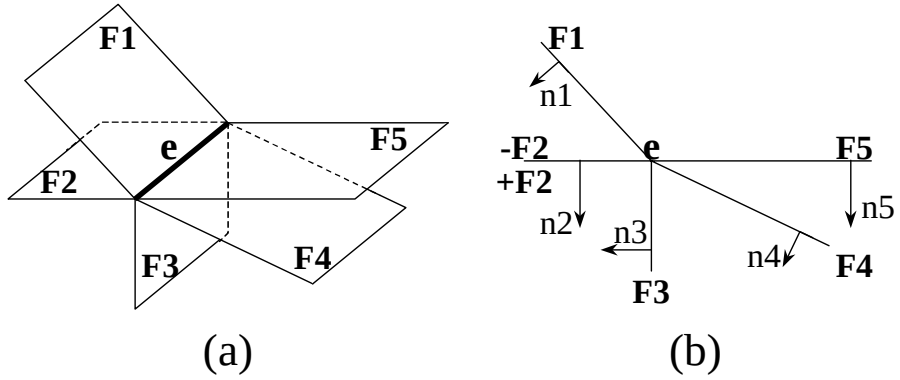


Figura 13:



"El algoritmo *BLG* genera todos los bucles de cuerpos a partir de los bucles de caras. Algunos de los bucles de cuerpos generados son ilimitados, y se llaman bucles de cuerpos exteriores. Los otros son interiores, porque son finitos. Los bucles de cuerpos exteriores son inútiles para construir objetos, y por eso se descartan.

Para clasificar los bucles de cuerpos interiores y exteriores, la información de selección de lado (por ejemplo el valor de ⊕) de los bucles de caras en la generación de los bucles de cuerpos es importante. En un bucle de cuerpo, los lados de todos los bucles de caras seleccionados usando el algoritmo anterior forman las caras interiores del bucle de cuerpo. Como resultado, el interior de un bucle de cuerpo exterior es ilimitado, y el exterior es un 'sólido' vacío finito. Para clasificar los bucles de cuerpos en interiores y exteriores, utilizamos los siguientes pasos:

- Para un bucle de cuerpo, seleccionar dos bucles de caras adyacentes $\oplus F_i$ y $\oplus F_j$ a su arista compartida *e*, y construir un rayo L que empiece en el punto medio de *e* y en la dirección de

$$L = \frac{\oplus n_1}{|n_1|} + \frac{\oplus n_2}{|n_2|}$$

donde $\oplus n_1$ y $\oplus n_2$ son los vectores normales hipotéticos de $\oplus F_i$ y $\oplus F_j$, respectivamente, y $\oplus$ corresponde a su selección de lado.

- Analizar las situaciones en las cuales la línea L intersecta con el bucle de cara del bucle de cuerpo. Si el número de puntos de intersección formados por L y todos los bucles de caras en un bucle de cuerpo (excluyendo F_i y F_j) es uno o más, entonces el bucle de cuerpo es interior; en otro caso, es exterior.

Usamos los vectores normales hipotéticos de los bucles de caras. Para todo bucle de cara F de un bucle de cuerpo, si F es seleccionado por el lado negativo, por ejemplo la cara -F, entonces su hipotético vector normal exterior +n apunta al exterior del bucle de cuerpo, mientras -n apunta al interior.

Similarmente, si F es seleccionado por el lado positivo, por ejemplo +F, entonces su hipotético vector normal exterior +n apunta al interior del bucle de cuerpo, mientras -n apunta al exterior. Esto implica que el auténtico vector normal exterior de un bucle de cara -F de un cuerpo es +n, o el auténtico vector normal exterior de un bucle de cara +F en el bucle de cuerpo es -n.

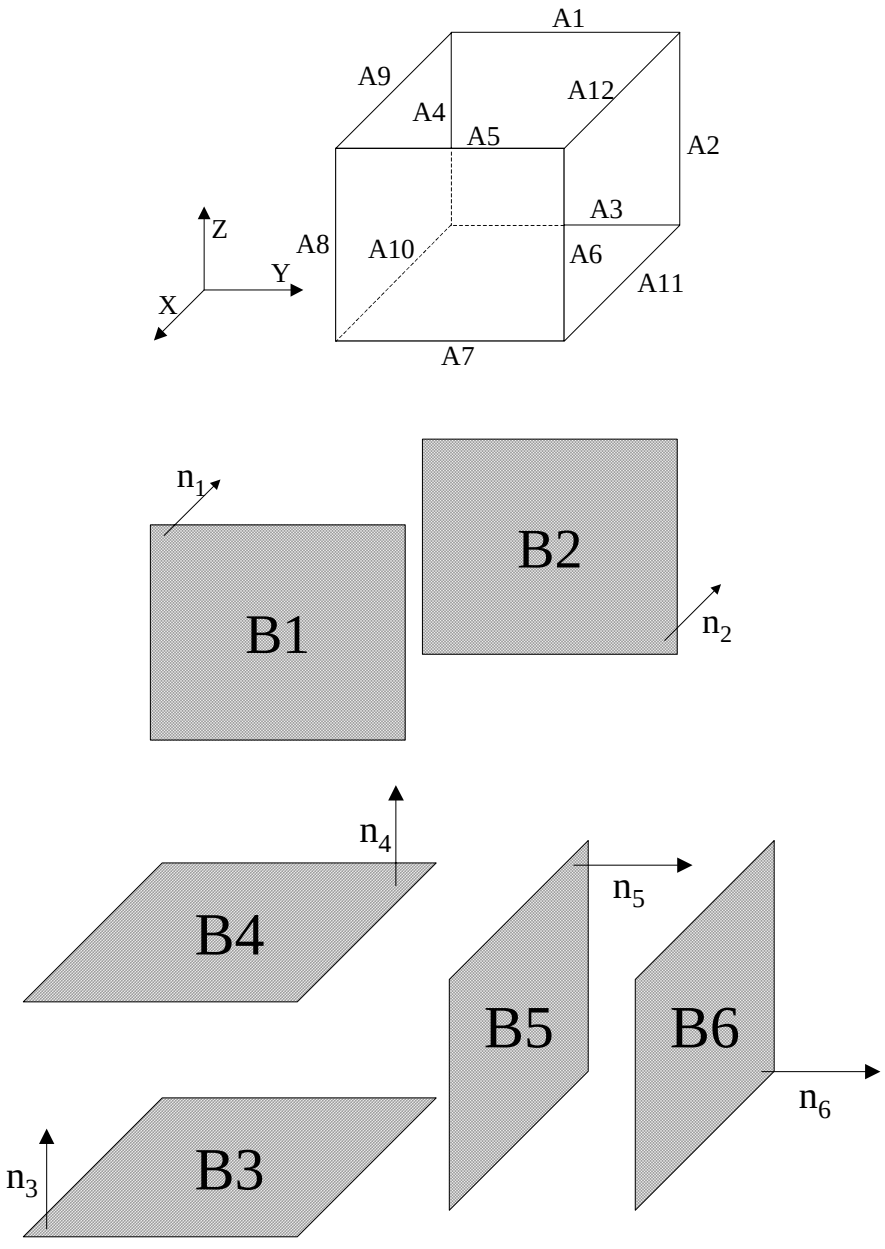
Por lo tanto, podemos obtener los auténticos vectores normales exteriores para todas las caras de cualquier cuerpo. La información del auténtico vector normal exterior puede ser usada para construir la representación de fronteras (*boundary representations, R-rep)* completa de los objetos."

Resumiendo el contenido de este apartado del artículo de Qing-Wen Yan, se puede decir que este procedimiento de generación de bucles de cuerpos consta de tres puntos clave. El primer punto es la generación de los bucles de cuerpos con la información de los bucles de caras y sus hipotéticos vectores normales exteriores. El segundo punto es la verificación de los bucles de cuerpos para comprobar que son cerrados. Estos dos puntos anteriores se repiten conjuntamente dentro de un bucle, así realizamos la verificación del bucle de cuerpo cada vez que obtenemos un bucle de cuerpo. El tercer y último punto en el que se basa este algoritmo es la comprobación de si los bucles de cuerpos generados son exteriores o interiores. Así eliminaremos los bucles de cuerpos generados que sean exteriores y nos quedaremos con los bucles de cuerpos que sean interiores.

En este apartado nos vamos a centrar en la resolución del método de generación de bucles de cuerpos mediante ejemplos. Inicialmente se hará con un ejemplo muy sencillo para familiarizarnos con el método, pasando seguidamente a la resolución de un ejemplo más complicado, verificando su corrección. Y después nos centraremos en el método para descubrir si un bucle de cuerpo es exterior o interior añadiendo unas nociones de geometría analítica necesarias en la resolución de este apartado.

2.8.1. EJEMPLO SENCILLO DE GENERACIÓN DE BUCLES DE CUERPOS.

En este apartado vamos a ver un ejemplo sencillo de generación de bucles de cuerpos, como se puede ver en la figura siguiente vamos a utilizar un objeto sencillo, un cubo. Para este objeto suponemos que se han aplicado todos los puntos anteriores del algoritmo, por lo tanto disponemos de los bucles de caras del objeto y los hipotéticos vectores normales exteriores de cada bucle de cara.



Tenemos seis bucles de caras que son: B1, B2, B3, B4, B5 y B6. Cada bucle de cara tiene su vector normal n_1, n_2, n_3, n_4, n_5 y n_6, respectivamente. Sabemos que los bucles de caras constan de los siguientes conjuntos de aristas: B1={A5, A6, A7, A8}, B2={A1, A2, A3, A4}, B3={A3, A10, A7, A11}, B4={A1, A9, A5, A12}, B5={A9, A8, A10, A4} y B6={A12, A6, A11, A2}. Estos son los datos de partida para el procedimiento de generación de bucles de cuerpos que a continuación vamos a ver.

Realizamos la inicialización de los conjuntos:

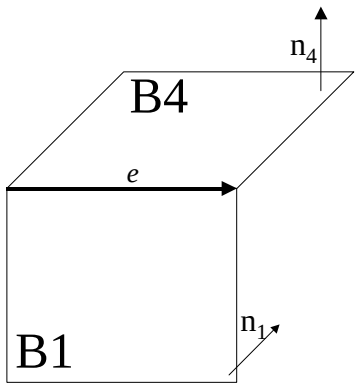
- Conjunto de marcas de caras positivas: $\chi_+()=\{0, 0, 0, 0, 0, 0\}$.
- Conjunto de marcas de caras negativas: $\chi_-()=\{0, 0, 0, 0, 0, 0\}$.
- Conjunto de bucles de caras: $S(F)=\varnothing$.

La posición (índice) dentro de cada conjunto de marcas indica el bucle de cara al que nos referimos, el cero indica *"sin usar"* y el uno indica *"usado"*. Así pues, si queremos decir que hemos utilizado la cara positiva (la que nos proporciona el vector normal $+n$) del bucle de cara B3 realizaremos la asignación $\chi_+(3)=1$. Y si queremos decir que hemos utilizado la cara negativa (la que nos proporciona el vector normal $-n$) del bucle de cara B5 realizaremos la asignación $\chi_-(5)=1$.

Ahora pasamos a seleccionar una cara de inicio y su lado. Seleccionaremos el primer bucle de cara B1 y como los dos lados están *"sin usar"* elegimos el positivo. Nos referiremos a la marca *"sin expandir"* como S-E y la marca *"expandido"* como EX. Por lo tanto introducimos esta cara, su lado y lo marcamos como *"sin expandir"* dentro del conjunto de bucles de caras: $S(F)=\{\{+B1, S\text{-}E\}\}$; y marcamos $\chi_+(1)=1$.

Entonces ahora elegimos un bucle de cara sin expandir de S(F), como solo hay uno, tomamos +B1. Tenemos que obtener las caras adyacentes a B1 con respecto a las aristas de B1. Por lo tanto, vamos a ir recorriendo las aristas de B1 obteniendo las caras adyacentes y realizando unas operaciones para seleccionar la cara adyacente apropiada y su lado apropiado.

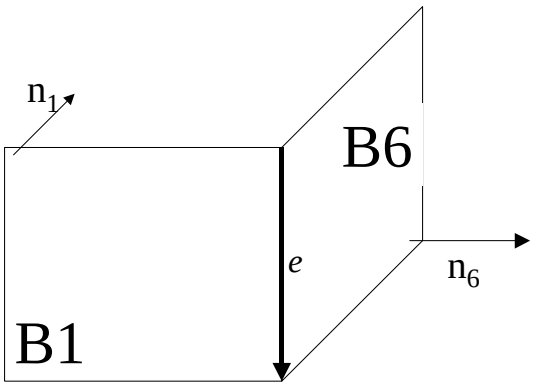
Tenemos que B1={A5, A6, A7, A8}, y seleccionamos la primera arista que es A5 y formamos el vector *e* sobre esta arista, de forma que satisface la regla de la mano derecha con el vector normal n_1. Veamos en la figura siguiente la situación:



Podemos ver que $n_4 \times n_1$ es un vector de sentido contrario al vector *e*, además +B4 está a la izquierda de *e*, este caso sería igual al de la figura 12b, por lo tanto el ángulo formado por las caras será $\alpha=\theta$ y se añade la cara –B4 al conjunto S(F). Tenemos que los conjuntos tendrán la siguiente configuración:

- Conjunto de marcas de caras positivas: $\chi_+()=\{1, 0, 0, 0, 0, 0\}$.
- Conjunto de marcas de caras negativas: $\chi_-()=\{0, 0, 0, 1, 0, 0\}$.
- Conjunto de bucles de caras: $S(F)=\{\{+B1, S\text{-}E\}, \{-B4, S\text{-}E\}\}$.

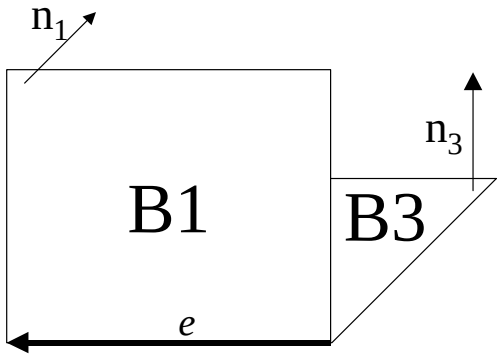
Seleccionamos la siguiente arista de B1 que será la arista A6, y tenemos:



Sabemos que $n_6 \times n_1$ es un vector de sentido contrario al vector *e*, además +B6 está a la izquierda de *e*, sería como el caso anterior, igual a la figura 12b, por lo tanto el ángulo formado por las caras será $\alpha=\theta$ y se añade la cara –B6 al conjunto S(F). Tenemos que los conjuntos tendrán la siguiente configuración:

- Conjunto de marcas de caras positivas: $\chi_+()=\{1, 0, 0, 0, 0, 0\}$.
- Conjunto de marcas de caras negativas: $\chi_-()=\{0, 0, 0, 1, 0, 1\}$.
- Conjunto de bucles de caras: S(F)={{+B1, S-E}, {-B4, S-E}, {-B6, S-E}}.

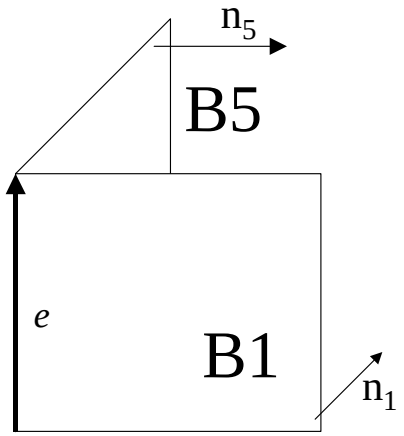
Después seleccionamos la siguiente arista de B1 que será la arista A7, y tenemos:



Además $n_3 \times n_1$ es un vector con el mismo sentido que el vector *e*, y +B3 está a la derecha de *e*, el ángulo formado por las caras será $\alpha=\pi-\theta$ y se añade la cara +B3 al conjunto S(F). Este caso es igual al de la figura 12a. Tenemos que los conjuntos tendrán la siguiente configuración:

- Conjunto de marcas de caras positivas: $\chi_+()=\{1, 0, 1, 0, 0, 0\}$.
- Conjunto de marcas de caras negativas: $\chi_-()=\{0, 0, 0, 1, 0, 1\}$.
- Conjunto de bucles de caras: S(F)={{+B1, S-E}, {-B4, S-E}, {-B6, S-E}, {+B3, S-E}}.

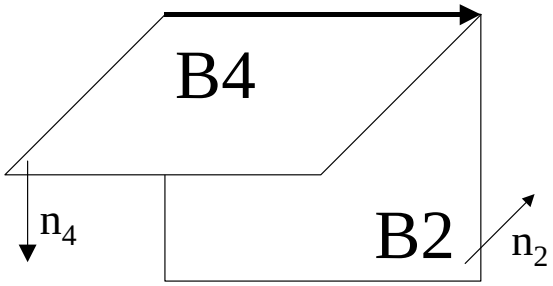
Ahora seleccionamos la última arista de B1 que será la arista A8, tenemos:



Este caso es igual al anterior, por lo tanto se selecciona la cara +B5. Como se han analizado todas las aristas del bucle de cara +B1, se marca como *"expandido"* en S(F). Los conjuntos tendrán la siguiente configuración:

- Conjunto de marcas de caras positivas: $\chi_{+}()=\{1, 0, 1, 0, 1, 0\}$.
- Conjunto de marcas de caras negativas: $\chi_{-}()=\{0, 0, 0, 1, 0, 1\}$.
- Conjunto de bucles de caras: S(F)={{+B1, EX}, {-B4, S-E}, {-B6, S-E}, {+B3, S-E}, {+B5, S-E}}.

Pasamos entonces a analizar la siguiente cara de S(F), en este caso será el bucle de cara –B4. Hay que tener en cuenta que en este caso como la cara seleccionada es la cara –B4, entonces cambiamos el sentido del vector normal. La primera arista de B4 es la arista A1, por lo tanto:



Este caso es idéntico al de la figura 12b. Tenemos que $n_2 \times (-n_4)$ tiene sentido contrario al vector *e*, y +B2 está a la izquierda de *e*, añadiremos la cara –B2 al conjunto S(F). Los conjuntos tendrán la siguiente configuración:

- Conjunto de marcas de caras positivas: $\chi_{+}()=\{1, 0, 1, 0, 1, 0\}$.
- Conjunto de marcas de caras negativas: $\chi_{-}()=\{0, 1, 0, 1, 0, 1\}$.
- Conjunto de bucles de caras: S(F)={{+B1, EX}, {-B4, S-E}, {-B6, S-E}, {+B3, S-E}, {+B5, S-E}, {-B2, S-E}}.

Siguiendo la misma forma de actuar, después de seleccionar las aristas A12, A5 y A9 se eligen las caras –B6, +B1 y +B5, respectivamente, que como ya las tenemos en el conjunto S(F), este conjunto no varía, pasando a marcar –B4 en S(F) como *"expandido"*.

Como se puede comprobar, seleccionando los sucesivos bucles de caras de S(F), que son –B6, +B3, +B5 y –B2, no se obtienen bucles de caras que no tengamos en el conjunto S(F). Llegados a este punto, tenemos la siguiente configuración de las variables:

- Conjunto de marcas de caras positivas: $\chi_{+}()=\{1, 0, 1, 0, 1, 0\}$.
- Conjunto de marcas de caras negativas: $\chi_{-}()=\{0, 1, 0, 1, 0, 1\}$.
- Conjunto de bucles de caras: S(F)={{+B1, EX}, {-B4, EX}, {-B6, EX}, {+B3, EX}, {+B5, EX}, {-B2, EX}}.

Por lo tanto, cuando llegamos a esta parte en la que no se añaden más bucles de caras al conjunto S(F) y se han expandido todos los bucles de caras del conjunto S(F), entonces hemos obtenido un bucle de cuerpo que estaría formado por las caras:

$$C_1=\{+B1, -B4, -B6, +B3, +B5, -B2\}$$

Después de esto, tenemos que S(F)=∅ y seleccionamos una cara *"sin usar"* del conjunto $\chi_{+}()$ o del conjunto $\chi_{-}()$. Si trazásemos a partir de aquí este procedimiento de generación de bucles de cuerpos de forma similar a la traza anterior, llegaríamos a una configuración de las variables siguientes:

- Conjunto de marcas de caras positivas: $\chi_{+}()=\{1, 1, 1, 1, 1, 1\}$.
- Conjunto de marcas de caras negativas: $\chi_{-}()=\{1, 1, 1, 1, 1, 1\}$.
- Conjunto de bucles de caras: S(F)={{+B2, EX}, {+B4, EX}, {+B6, EX}, {-B3, EX}, {-B5, EX}, {-B1, EX}}.

Y habríamos obtenido otro bucle de cuerpo formado por las caras:

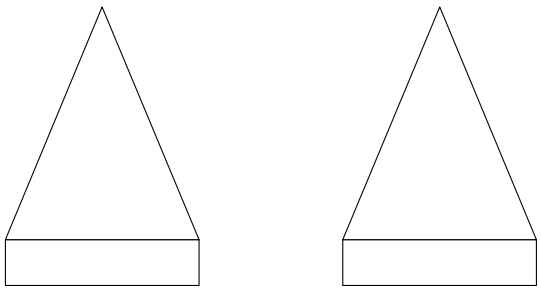
$$C_2=\{+B2, +B4, +B6, -B3, -B5, -B1\}$$

El procedimiento acaba cuando se han usado todas las caras posibles en la generación de bucles de cuerpos, o lo que es lo mismo, cuando los conjuntos $\chi_{+}()$ y $\chi_{-}()$ son todo unos: {1, 1, 1, 1, ...}.

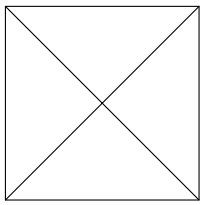
Aplicando el método para descubrir bucles de cuerpos exteriores que explicaremos posteriormente, nos damos cuenta que el bucle de cuerpo C_1 es interior y el bucle de cuerpo C_2 es exterior, por lo tanto, tendremos que eliminar el bucle C_2. Es decir, tenemos que este objeto sólo tiene un bucle de cuerpo, C_1, que por ser único, será el objeto correcto.

2.8.2. EJEMPLO DE GENERACIÓN DE BUCLES DE CUERPOS.

En este apartado vamos a ver un ejemplo un poco más complejo de generación de bucles de cuerpos, como se puede ver en la figura siguiente tenemos las vistas de planta, alzado y perfil del objeto que vamos a estudiar. Para este objeto suponemos que se han aplicado todos los puntos anteriores del algoritmo, por lo tanto disponemos de los bucles de caras del objeto y los hipotéticos vectores normales exteriores de cada bucle de cara.

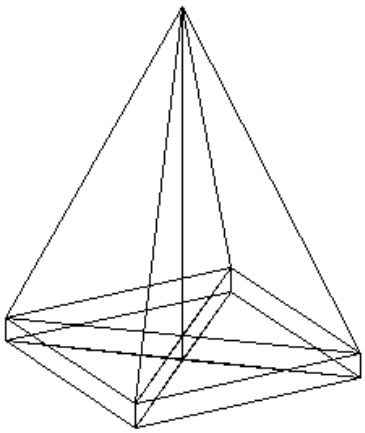


ALZADO PERFIL



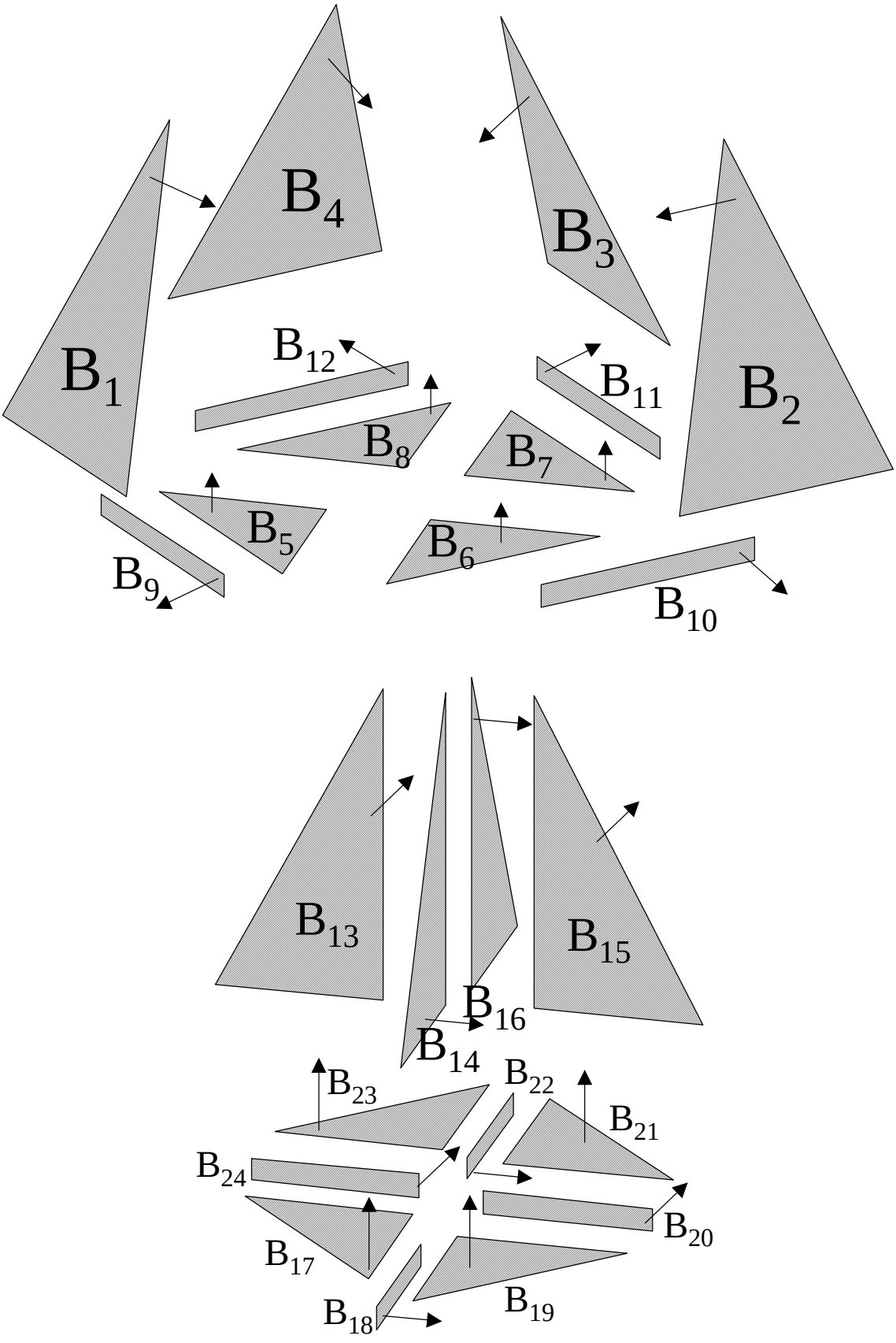
PLANTA

En la figura siguiente podemos observar el modelo alámbrico del objeto, generado con los procedimientos del algoritmo: *WC* (construcción del modelo alámbrico), *PGG* (generación de gráficos planos), *BLR* (procesado de la información de las líneas discontinuas), *FLG* (generación de bucles de caras), *CEV* (generación de vértices y aristas de corte).



Como se puede apreciar en la figura anterior, hay dos aristas en el centro del objeto (que forman una especie de eje del objeto) que son líneas de corte generadas en el procedimiento del algoritmo *CEV*.

Ahora podemos observar en la figura siguiente todos los bucles de caras del objeto generados en el procedimiento *FLG*, así como sus hipotéticos vectores normales exteriores, calculados en el procedimiento *PGG* del algoritmo de reconstrucción.



Para todo este apartado, denominaremos a n_i como el hipotético vector normal exterior del bucle de cara B_i. Además tenemos los bucles de caras formados por las aristas:

- B_1={A5, A6, A18}.
- B_2={A2, A6, A8}.
- B_3={A7, A17, A8}.
- B_4={A1, A7, A5}.
- B_5={A20, A26, A14}.
- B_6={A12, A13, A26}.
- B_7={A19, A25, A13}.
- B_8={A11, A14, A25}.
- B_9={A15, A20, A16, A18}.
- B_{10}={A2, A16, A12, A10}.
- B_{11}={A9, A17, A10, A19}.
- B_{12}={A1, A9, A11, A15}.
- B_{13}={A5, A4, A23}.
- B_{14}={A6, A22, A4}.
- B_{15}={A8, A24, A4}.
- B_{16}={A7, A4, A21}.
- B_{17}={A18, A22, A23}.
- B_{18}={A16, A26, A3, A22}.
- B_{19}={A2, A24, A22}.
- B_{20}={A10, A13, A3, A24}.
- B_{21}={A17, A21, A24}.
- B_{22}={A9, A21, A3, A25}.
- B_{23}={A1, A23, A21}.
- B_{24}={A15, A14, A3, A23}.

Por lo tanto, ya tenemos todos los datos necesarios para la generación de los bucles de cuerpos, por parte del procedimiento *BLG*. A partir de ahora, vamos a ir viendo una traza del algoritmo de generación de bucles de cuerpos para el ejemplo especificado en las figuras anteriores.

Durante el seguimiento de la traza del algoritmo emplearemos una serie de variables que iremos modificando dependiendo de las acciones del algoritmo. Estas variables que vamos a utilizar en la traza, las tendremos inicializadas a los valores siguientes:

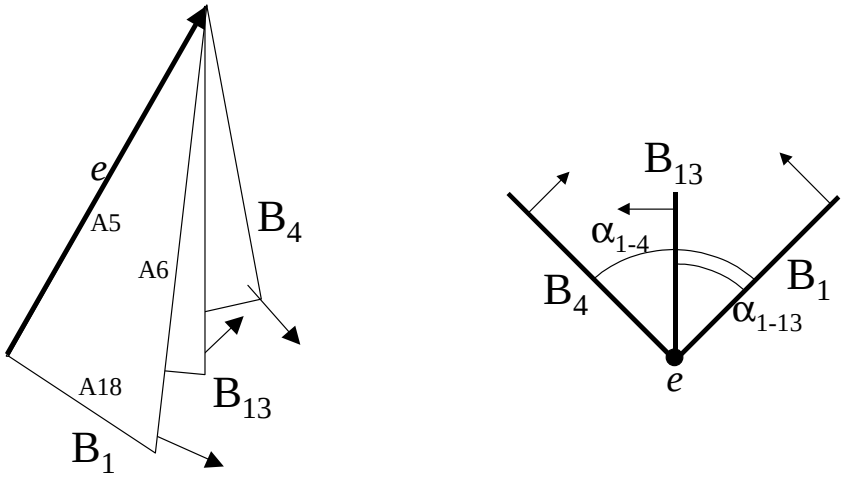
- Conjunto de marcas de caras positivas:
  $\chi_+()$={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}.
- Conjunto de marcas de caras negativas:
  $\chi_-()$={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}.
- Conjunto de bucles de caras: S(F)=∅.
- Conjunto de bucles de cuerpos: BL=∅.

La posición (índice) dentro de cada conjunto de marcas indica el bucle de cara al que nos referimos, el valor cero indica *"sin usar"* y el uno indica *"usado"*. Así pues, si queremos decir que hemos utilizado la cara positiva (la que nos proporciona el vector normal $+n$) del bucle de cara B_3 realizaremos la asignación $\chi_+(3)=1$. Y si queremos decir que hemos utilizado la cara negativa (la que nos proporciona el vector normal $-n$) del bucle de cara B_5 realizaremos la asignación $\chi_-(5)=1$.

Ahora pasamos a seleccionar una cara de inicio y su lado. Seleccionaremos el primer bucle de cara B_1 y como los dos lados están *"sin usar"* elegimos el positivo. Nos referiremos a la marca *"sin expandir"* como S-E y la marca *"expandido"* como EX. Por lo tanto introducimos esta cara, su lado y lo marcamos como *"sin expandir"* dentro del conjunto de bucles de caras: $S(F)=\{\{+B_1, \text{S-E}\}\}$; y marcamos $\chi_+(1)=1$.

Entonces ahora elegimos un bucle de cara sin expandir de S(F), como solo hay uno, tomamos $+B_1$. Tenemos que obtener las caras adyacentes a B_1 con respecto a las aristas de B_1. Por lo tanto, vamos a ir recorriendo las aristas de B_1 obteniendo las caras adyacentes y realizando unas operaciones para seleccionar la cara adyacente apropiada y su lado apropiado.

Tenemos que $B_1=\{A5, A6, A18\}$, seleccionamos la primera arista que es A5 y formamos el vector *e* sobre esta arista, de forma que satisface la regla de la mano derecha con el vector normal n_1. Veamos en la figura siguiente la situación:



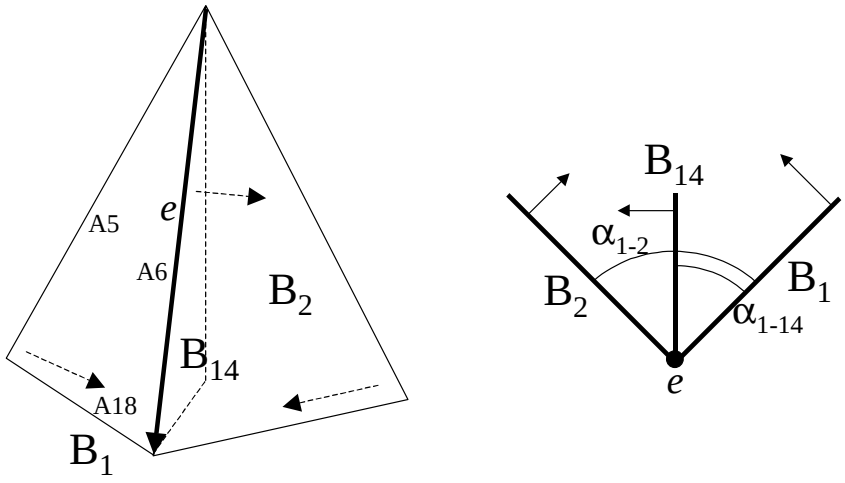
Como se puede ver en las figuras anteriores, la cara adyacente a la cara $+B_1$ será la cara $-B_{13}$. Esto se descubre calculando el ángulo formado entre las caras $+B_1$ y $-B_{13}$, que llamamos $\alpha_{1\text{-}13}$, y el ángulo formado entre las caras $+B_1$ y $+B_4$, que llamamos $\alpha_{1\text{-}4}$. Los ángulos α_i se calculan a partir de los ángulos θ_i. Estos ángulos se calculan a partir de los vectores normales de las caras implicadas mediante la fórmula:

$$\theta_{k-s} = \cos^{-1}\left[\frac{n_S \cdot n_K}{|n_S| \cdot |n_K|}\right]$$

Para el cálculo del ángulo $\alpha_{1\text{-}13}$ tenemos que el vector $n_{13}\times n_1$ está en la dirección contraria (tiene sentido contrario) al vector *e*, y $+B_{13}$ está a la izquierda de *e*. Por lo tanto, $\alpha_{1\text{-}13}=\theta_{13\text{-}1}$ y seleccionamos la cara negativa de B_{13} ($-B_{13}$). Esto es equivalente a la figura 12b. Para el cálculo del ángulo $\alpha_{1\text{-}4}$ tenemos que el vector $n_4\times n_1$ está en la misma dirección (tiene el mismo sentido) que el vector *e*, y $+B_4$ está a la derecha de *e*. Por lo tanto, $\alpha_{1\text{-}4}=\pi-\theta_{4\text{-}1}$ y seleccionamos la cara positiva de B_4 ($+B_4$). Esto es equivalente a la figura 12a. Después de todo esto, podemos deducir que la cara adyacente a $+B_1$ será la que tenga un ángulo α menor, y como sabemos que $\alpha_{1\text{-}13}<\alpha_{1\text{-}4}$, por lo tanto podemos decir que la cara adyacente a $+B_1$ es la cara $-B_{13}$. Ahora pasamos a tener las variables con la siguiente configuración:

- Conjunto de marcas de caras positivas:
  $\chi_+()=\{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de marcas de caras negativas:
  $\chi_-()=\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de bucles de caras: $S(F)=\{\{+B_1, S\text{-}E\}, \{-B_{13}, S\text{-}E\}\}$.
- Conjunto de bucles de cuerpos: $BL=\varnothing$.

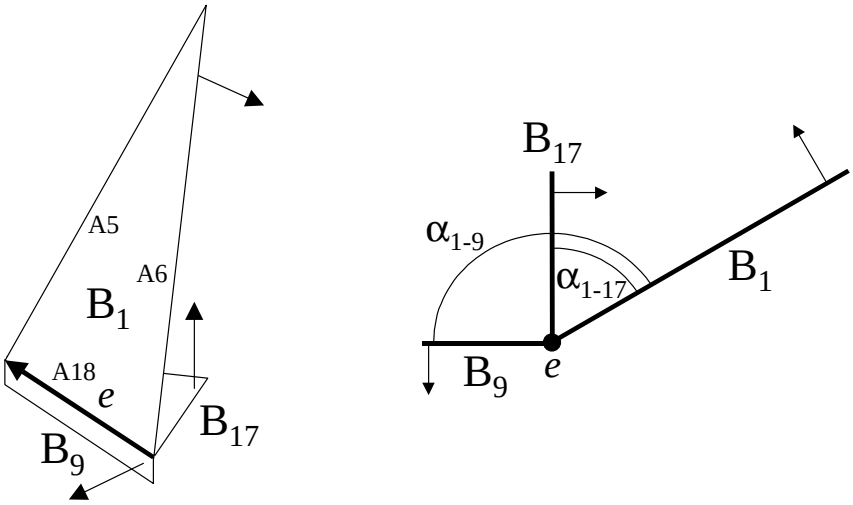
Ahora seleccionamos la siguiente arista de B_1 que es A6 y formamos el vector *e* sobre esta arista, de forma que satisface la regla de la mano derecha con el vector normal n_1. Veamos en la figura siguiente la situación:



Para el cálculo del ángulo $\alpha_{1\text{-}14}$ tenemos que el vector $n_{14}\times n_1$ está en la dirección contraria al vector *e*, y $+B_{14}$ está a la izquierda de *e*. Por lo tanto, $\alpha_{1\text{-}14}=\theta_{14\text{-}1}$ y seleccionamos la cara negativa de B_{14} ($-B_{14}$). Esto es equivalente a la figura 12b. Para el cálculo del ángulo $\alpha_{1\text{-}2}$ tenemos que el vector $n_2\times n_1$ está en la misma dirección que el vector *e*, y $+B_2$ está a la derecha de *e*. Por lo tanto, $\alpha_{1\text{-}2}=\pi-\theta_{2\text{-}1}$ y seleccionamos la cara positiva de B_2 ($+B_2$). Esto es equivalente a la figura 12a. Podemos decir que la cara adyacente a $+B_1$ será la que tenga un ángulo α menor, y como sabemos que $\alpha_{1\text{-}14}<\alpha_{1\text{-}2}$, por lo tanto podemos decir que la cara adyacente a $+B_1$ es la cara $-B_{14}$. Ahora pasamos a tener las variables con la siguiente configuración:

- Conjunto de marcas de caras positivas:
  $\chi_+()$={1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}.
- Conjunto de marcas de caras negativas:
  $\chi_-()$={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}.
- Conjunto de bucles de caras: S(F)= {{+B_1, S-E}, {-B_{13}, S-E}, {-B_{14}, S-E}}.
- Conjunto de bucles de cuerpos: BL=∅.

Ahora seleccionamos la siguiente arista de B_1 que es A18 y formamos el vector *e* sobre esta arista, de forma que satisface la regla de la mano derecha con el vector normal n_1. Veamos en la figura siguiente la situación:

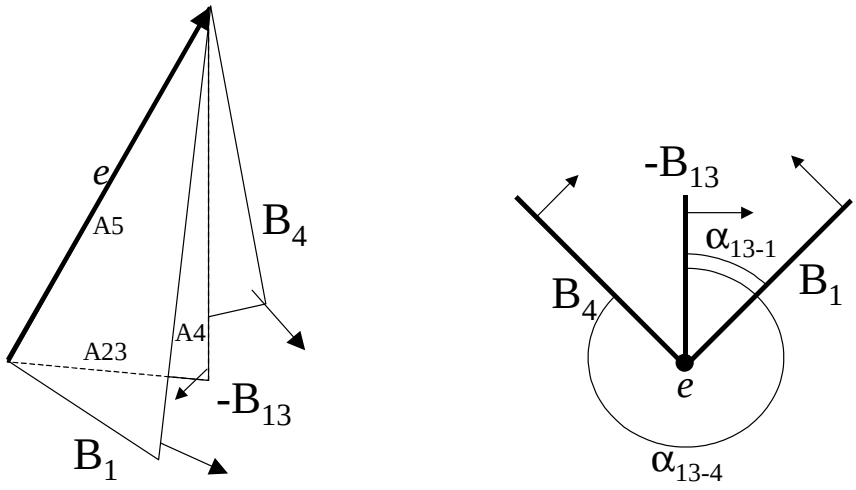


Para el cálculo del ángulo $\alpha_{1\text{-}17}$ tenemos que el vector $n_{17}\times n_1$ está en la misma dirección que el vector *e*, y +B_{17} está a la derecha de *e*. Por lo tanto, $\alpha_{1\text{-}17}=\pi-\theta_{17\text{-}1}$ y seleccionamos la cara positiva de B_{17} (+B_{17}). Esto es equivalente a la figura 12a. Para el cálculo del ángulo $\alpha_{1\text{-}9}$ tenemos que el vector $n_9\times n_1$ está en dirección contraria al vector *e*, y +B_2 está a la izquierda de *e*. Por lo tanto, $\alpha_{1\text{-}9}=\theta_{9\text{-}1}$ y seleccionamos la cara negativa de B_2 (-B_2). Esto es equivalente a la figura 12b. Podemos decir que la cara adyacente a +B_1 será la que tenga un ángulo α menor, y como sabemos que $\alpha_{1\text{-}17}<\alpha_{1\text{-}9}$, por lo tanto podemos decir que la cara adyacente a +B_1 es la cara +B_{17}. Ahora pasamos a tener las variables con la siguiente configuración:

- Conjunto de marcas de caras positivas:
  $\chi_+()$={1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0}.
- Conjunto de marcas de caras negativas:
  $\chi_-()$={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}.
- Conjunto de bucles de caras:
  S(F)= {{+B_1, EX}, {-B_{13}, S-E}, {-B_{14}, S-E}, {+B_{17}, S-E}}.
- Conjunto de bucles de cuerpos: BL=∅.

Como ya se han obtenido todas las caras adyacentes de +B_1, marcamos +B_1 como *"expandido"* en S(F) y seleccionamos la siguiente cara, en este caso, -B_{13}. Tenemos que B_{13}={A5, A4, A23}, seleccionamos la primera arista que es A5 y

formamos el vector *e* sobre esta arista, de forma que satisface la regla de la mano derecha con el vector normal n_{13}. Veamos en la figura siguiente la situación:

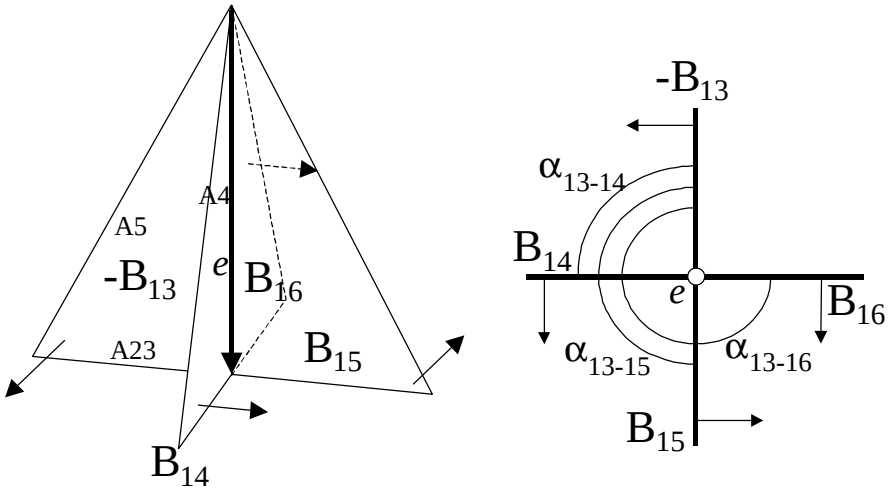


Para este caso que tenemos seleccionada la cara $-B_{13}$ lo que haremos será tratar las condiciones como si se hubiera seleccionado la cara $+B_{13}$, y luego cuando obtenemos los ángulos α hacemos $\alpha=2\pi-\alpha$ y cambiamos la selección de lado. Para el cálculo del ángulo $\alpha_{13\text{-}1}$ tenemos que el vector resultado de $n_1 \times n_{13}$ está en la misma dirección que el vector *e*, y $+B_1$ está a la izquierda de *e*. Por lo tanto, $\alpha_{13\text{-}1}=2\pi-\theta_{1\text{-}13}$ y seleccionamos la cara negativa de B_1 ($-B_1$). Esto es equivalente a la figura 12c. Pero como la cara de partida es negativa tendremos que $\alpha_{13\text{-}1}=\theta_{1\text{-}13}$ y se selecciona $+B_1$. Para el cálculo del ángulo $\alpha_{13\text{-}4}$ tenemos que el vector $n_4 \times n_{13}$ está en la misma dirección que el vector *e*, y $+B_4$ está a la derecha de *e*. Por lo tanto, $\alpha_{13\text{-}4}=\pi-\theta_{4\text{-}13}$ y seleccionamos la cara positiva de B_4 ($+B_4$). Esto es equivalente a la figura 12a. Pero como la cara de partida es negativa tendremos que $\alpha_{13\text{-}4}=\pi+\theta_{4\text{-}13}$ y se selecciona $-B_4$. Podemos decir que la cara adyacente a $-B_{13}$ será la que tenga un ángulo α menor, y como sabemos que la relación de los ángulos es $\alpha_{13\text{-}1}<\alpha_{13\text{-}4}$, podemos decir que la cara adyacente a $-B_{13}$ es la cara $+B_1$. Ahora pasamos a tener las variables con la siguiente configuración:

- Conjunto de marcas de caras positivas:
  $\chi_+()=\{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de marcas de caras negativas:
  $\chi_-()=\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de bucles de caras:
  $S(F)= \{\{+B_1, EX\}, \{-B_{13}, S\text{-}E\}, \{-B_{14}, S\text{-}E\}, \{+B_{17}, S\text{-}E\}\}$.
- Conjunto de bucles de cuerpos: $BL=\varnothing$.

Como se puede ver, las variables anteriores no se han modificado de forma alguna debido a que la cara $+B_1$ ya la teníamos en el conjunto S(F), o lo que es lo mismo, ya se había usado $\chi_+(1)=1$.

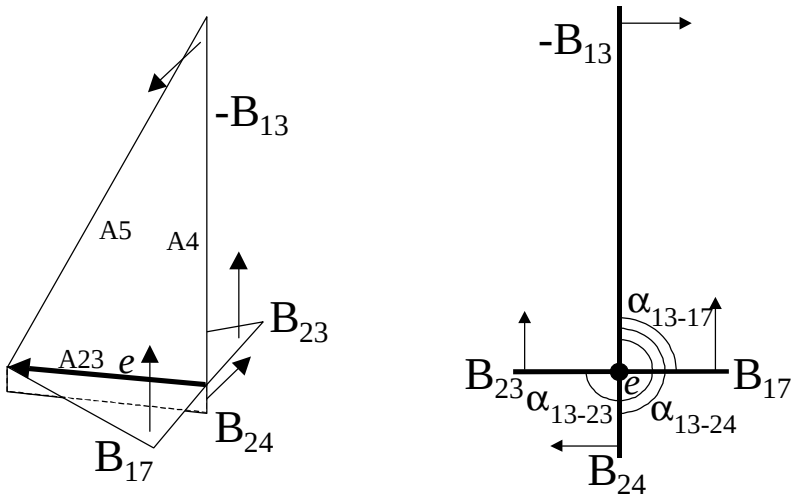
Ahora seleccionamos la siguiente arista de B_{13} que es la arista A4 y formamos el vector *e* sobre esta arista, de forma que satisface la regla de la mano derecha con el vector normal n_{13}. Veamos en la figura siguiente la situación:



Como en el caso anterior tenemos seleccionada la cara $-B_{13}$ y lo que haremos será tratar las condiciones como si se hubiera seleccionado la cara $+B_{13}$, y luego cuando obtenemos los ángulos α hacemos $\alpha=2\pi-\alpha$ y cambiamos la selección de lado. Para el cálculo del ángulo $\alpha_{13\text{-}14}$ tenemos que el vector resultado de $n_{14}\times n_{13}$ está en dirección contraria al vector *e*, y $+B_{14}$ está a la derecha de *e*. Por lo tanto, $\alpha_{13\text{-}14}=\pi+\theta_{14\text{-}13}$ y seleccionamos la cara positiva de B_{14} ($+B_{14}$). Esto es equivalente a la figura 12d. Pero como la cara de partida es negativa tendremos que $\alpha_{13\text{-}14}=\pi-\theta_{14\text{-}13}$ y se selecciona $-B_{14}$. Para el cálculo del ángulo $\alpha_{13\text{-}15}$ tenemos que el vector $n_{15}\times n_{13}$ está en dirección contraria al vector *e*, y $+B_{15}$ está a la derecha de *e*. Por lo tanto, $\alpha_{13\text{-}15}=\pi+\theta_{15\text{-}13}$ y seleccionamos la cara positiva de B_{15} ($+B_{15}$). Esto es equivalente a la figura 12d. Pero como la cara de partida es negativa tendremos que $\alpha_{13\text{-}15}=\pi-\theta_{15\text{-}13}$ y se selecciona $-B_{15}$. Para el cálculo del ángulo $\alpha_{13\text{-}16}$ tenemos que el vector resultado de $n_{16}\times n_{13}$ está en dirección contraria al vector *e*, y $+B_{16}$ está a la izquierda de *e*. Por lo tanto, el ángulo es $\alpha_{13\text{-}16}=\theta_{16\text{-}13}$ y seleccionamos la cara negativa de B_{16} ($-B_{16}$). Esto es equivalente a la figura 12b. Pero como la cara de partida es negativa tendremos que $\alpha_{13\text{-}16}=2\pi-\theta_{16\text{-}13}$ y se selecciona $+B_{16}$. Podemos decir que la cara adyacente a $-B_{13}$ será la que tenga un ángulo α menor, y como sabemos que $\alpha_{13\text{-}14}<\alpha_{13\text{-}15}<\alpha_{13\text{-}16}$, por lo tanto podemos decir que la cara adyacente a $-B_{13}$ es la cara $-B_{14}$. Las variables de la traza no se modifican de forma alguna debido a que la cara $-B_{14}$ ya la teníamos en el conjunto S(F), o lo que es lo mismo, ya se había usado $\chi_{-}(14)=1$:

- Conjunto de marcas de caras positivas:
  $\chi_{+}()=\{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de marcas de caras negativas:
  $\chi_{-}()=\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de bucles de caras:
  $S(F)=\{\{+B_1, EX\}, \{-B_{13}, S\text{-}E\}, \{-B_{14}, S\text{-}E\}, \{+B_{17}, S\text{-}E\}\}$.
- Conjunto de bucles de cuerpos: $BL=\varnothing$.

Ahora seleccionamos la última arista de B_{13} que es la arista A23 y formamos el vector *e* sobre esta arista, de forma que satisface la regla de la mano derecha con el vector normal n_{13}. Veamos en la figura siguiente la situación:



Como en el caso anterior tenemos seleccionada la cara $-B_{13}$ y lo que haremos será tratar las condiciones como si se hubiera seleccionado la cara $+B_{13}$, y luego cuando obtenemos los ángulos α hacemos $\alpha=2\pi-\alpha$ y cambiamos la selección de lado. Para el cálculo del ángulo $\alpha_{13\text{-}17}$ tenemos que el vector resultado de $n_{17}\times n_{13}$ está en la misma dirección que el vector *e*, y $+B_{17}$ está a la izquierda de *e*. Por lo tanto, $\alpha_{13\text{-}17}=2\pi-\theta_{17\text{-}13}$ y seleccionamos la cara negativa de B_{17} ($-B_{17}$). Esto es equivalente a la figura 12c. Pero como la cara de partida es negativa tendremos que $\alpha_{13\text{-}17}=\theta_{17\text{-}13}$ y se selecciona $+B_{17}$. Para el cálculo del ángulo $\alpha_{13\text{-}24}$ tenemos que el vector $n_{24}\times n_{13}$ está en la misma dirección que el vector *e*, y $+B_{24}$ está a la derecha de *e*. Por lo tanto, $\alpha_{13\text{-}24}=\pi-\theta_{24\text{-}13}$ y seleccionamos la cara positiva de B_{24} ($+B_{24}$). Esto es equivalente a la figura 12a. Pero como la cara de partida es negativa tendremos que $\alpha_{13\text{-}24}=\pi+\theta_{24\text{-}13}$ y se selecciona $-B_{24}$. Para el cálculo del ángulo $\alpha_{13\text{-}23}$ tenemos que el vector resultado de $n_{23}\times n_{13}$ está en la misma dirección que el vector *e*, y $+B_{16}$ está a la derecha de *e*. Por lo tanto, el ángulo es $\alpha_{13\text{-}23}=\pi-\theta_{23\text{-}13}$ y seleccionamos la cara positiva de B_{23} ($+B_{23}$). Esto es equivalente a la figura 12a. Pero como la cara de partida es negativa tendremos que $\alpha_{13\text{-}23}=\pi+\theta_{23\text{-}13}$ y se selecciona $-B_{23}$. Podemos decir que la cara adyacente a $-B_{13}$ será la que tenga un ángulo α menor, y como sabemos que $\alpha_{13\text{-}17}<\alpha_{13\text{-}24}<\alpha_{13\text{-}23}$, por lo tanto podemos decir que la cara adyacente a $-B_{13}$ es la cara $+B_{17}$. Las variables de la traza no se modifican de forma alguna debido a que la cara $+B_{17}$ ya la teníamos en el conjunto S(F), o lo que es lo mismo, ya se había usado $\chi_{+}(17)=1$, además se han examinado todas sus aristas, por lo tanto:

- Conjunto de marcas de caras positivas:
  $\chi_{+}()=\{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de marcas de caras negativas:
  $\chi_{-}()=\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de bucles de caras:
  $S(F)=\{\{+B_1, EX\}, \{-B_{13}, EX\}, \{-B_{14}, S\text{-}E\}, \{+B_{17}, S\text{-}E\}\}$.
- Conjunto de bucles de cuerpos: $BL=\varnothing$.

Otra forma de actuar cuando pasamos a examinar una cara negativa es la siguiente: ir seleccionando las aristas de este bucle de cara, pero a diferencia del método anterior, se forma el vector *e* sobre la arista seleccionada de forma que satisface la regla de la mano derecha con el vector normal de la cara negativa -*n*. El resto del procedimiento de generación de bucles de cuerpos de Qing-Wen Yan se aplica igual que para el caso de la cara positiva, pero hay que tener en cuenta que se ha cambiado el vector normal y el vector a lo largo de la arista *e*.

Después de haber examinado la cara $-B_{13}$, pasamos a examinar la cara $-B_{14}$. Como se puede comprobar, las caras adyacentes a las aristas {A6, A22, A4} en la cara negativa seleccionada ($-B_{14}$) son las caras $+B_1$, $-B_{13}$ y $+B_{17}$, que como las tenemos incluidas en el conjunto S(F), este conjunto no varía.

$$S(F)= \{\{+B_1, EX\}, \{-B_{13}, EX\}, \{-B_{14}, EX\}, \{+B_{17}, S\text{-}E\}\}.$$

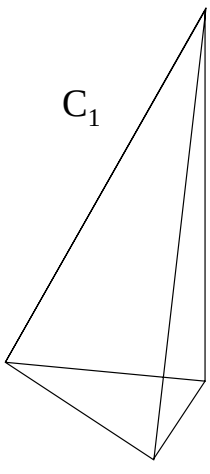
Y ahora pasamos a seleccionar la última cara *"no expandida"*, que es la cara $+B_{17}$. Como podemos comprobar, las caras adyacentes a las aristas {A18, A22, A23} en la cara positiva seleccionada ($+B_{17}$) son las caras $+B_1$, $-B_{13}$ y $-B_{14}$, que como las tenemos incluidas en el conjunto S(F), este conjunto no varía. Por lo tanto, el conjunto de bucles de caras queda de la forma siguiente:

$$S(F)= \{\{+B_1, EX\}, \{-B_{13}, EX\}, \{-B_{14}, EX\}, \{+B_{17}, EX\}\}.$$

Como se han expandido todas las caras, esto quiere decir que se ha generado un bucle de cuerpo que llamaremos C_1, asignaremos $S(F)= \varnothing$ y haremos $BL=BL\cup\{C_1\}$. Por lo tanto, tenemos:

- $C_1=\{+B_1, -B_{13}, -B_{14}, +B_{17}\}$.
- $BL=\{C_1\}$.

En la figura siguiente podemos ver el primer bucle de cuerpo formado (C_1):



Ahora seleccionamos $+B_2$ y lo insertamos en el conjunto S(F). Aplicamos el algoritmo de construcción de bucles de cuerpos hasta que llegamos a la situación siguiente:

- Conjunto de marcas de caras positivas:
  $\chi_+()=\{1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0\}$.
- Conjunto de marcas de caras negativas:
  $\chi_-()=\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de bucles de caras:
  $S(F)= \{\{+B_2, EX\}, \{-B_{15}, EX\}, \{+B_{14}, EX\}, \{+B_{19}, EX\}\}$.
- Conjunto de bucles de cuerpos: $BL=\{C_1\}$.

Como se han expandido todas las caras, esto quiere decir que se ha generado un bucle de cuerpo que llamaremos C_2, asignaremos $S(F)= \varnothing$ y haremos $BL=BL\cup\{C_2\}$. Por lo tanto, tenemos:

- $C_2=\{+B_2, -B_{15}, +B_{14}, +B_{19}\}$.
- $BL=\{C_1, C_2\}$.

Entonces seleccionamos $+B_3$ y lo insertamos en el conjunto S(F). Aplicamos el algoritmo de construcción de bucles de cuerpos hasta que llegamos a la situación siguiente:

- Conjunto de marcas de caras positivas:
  $\chi_+()=\{1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0\}$.
- Conjunto de marcas de caras negativas:
  $\chi_-()=\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de bucles de caras:
  $S(F)= \{\{+B_3, EX\}, \{+B_{15}, EX\}, \{+B_{16}, EX\}, \{+B_{21}, EX\}\}$.
- Conjunto de bucles de cuerpos: $BL=\{C_1, C_2\}$.

Como se han expandido todas las caras, esto quiere decir que se ha generado un bucle de cuerpo que llamaremos C_3, asignaremos $S(F)= \varnothing$ y haremos $BL=BL\cup\{C_3\}$. Por lo tanto, tenemos:

- $C_3=\{+B_3, +B_{15}, +B_{16}, +B_{21}\}$.
- $BL=\{C_1, C_2, C_3\}$.

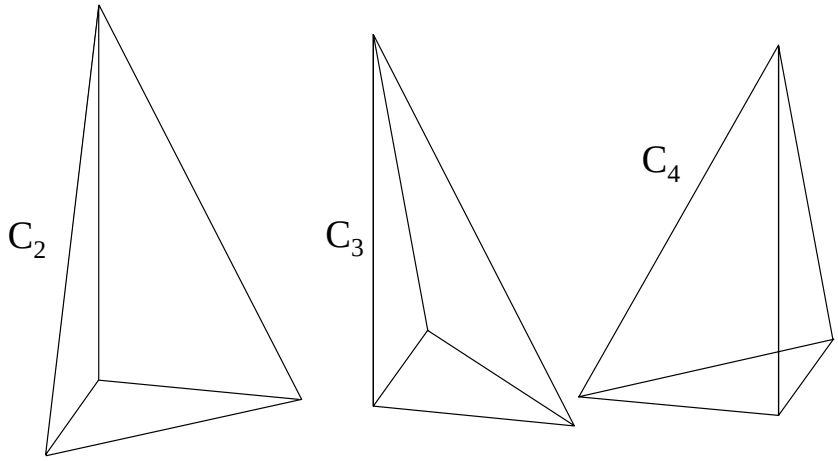
Después de esto seleccionamos $+B_4$ y lo insertamos en el conjunto S(F). Aplicamos el algoritmo de construcción de bucles de cuerpos hasta que llegamos a la situación siguiente:

- Conjunto de marcas de caras positivas:
  $\chi_+()=\{1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0\}$.
- Conjunto de marcas de caras negativas:
  $\chi_-()=\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0\}$.
- Conjunto de bucles de caras:
  $S(F)= \{\{+B_4, EX\}, \{+B_{13}, EX\}, \{-B_{16}, EX\}, \{+B_{23}, EX\}\}$.
- Conjunto de bucles de cuerpos: $BL=\{C_1, C_2, C_3\}$.

Como se han expandido todas las caras, esto quiere decir que se ha generado un bucle de cuerpo que llamaremos C_4, asignaremos $S(F)= \varnothing$ y haremos $BL=BL\cup\{C_4\}$. Por lo tanto, tenemos:

- $C_4=\{+B_4, +B_{13}, -B_{16}, +B_{23}\}$.
- $BL=\{C_1, C_2, C_3, C_4\}$.

En la figura siguiente podemos ver los bucles de cuerpos C_2, C_3, C_4:



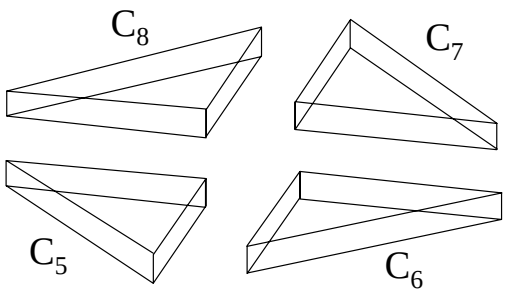
Aplicando el mismo método que se ha aplicado para generar los bucles de cuerpos anteriores se obtiene los siguientes bucles de cuerpos:

- $C_5=\{+B_5, -B_9, -B_{17}, -B_{18}, -B_{24}\}$.
- $C_6=\{+B_6, -B_{10}, -B_{19}, +B_{18}, -B_{20}\}$.
- $C_7=\{+B_7, -B_{11}, +B_{20}, -B_{21}, +B_{22}\}$.
- $C_8=\{+B_8, -B_{12}, -B_{22}, -B_{23}, +B_{24}\}$.

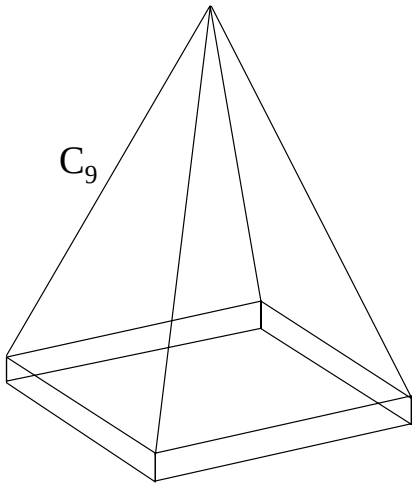
Quedando los conjuntos de la siguiente forma:

- Conjunto de marcas de caras positivas:
  $\chi_+()=\{1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$.
- Conjunto de marcas de caras negativas:
  $\chi_-()=\{0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$.
- Conjunto de bucles de cuerpos: $BL=\{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$.

En la figura siguiente podemos ver los bucles de cuerpos:



Y por último, se genera el bucle de cuerpo que representamos en la figura siguiente. Sabemos que ya no se van a generar más bucles de cuerpos porque los conjuntos $\chi_+()$ y $\chi_-()$ son todo unos.



$$C_9=\{-B_1, -B_2, -B_3, -B_4, -B_5, -B_6, -B_7, -B_8, +B_9, +B_{10}, +B_{11}, +B_{12}\}$$

Quedando los conjuntos de la siguiente forma:

- Conjunto de marcas de caras positivas:

$$\chi_+()=\{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}.$$

- Conjunto de marcas de caras negativas:

$$\chi_-()=\{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}.$$

- Conjunto de bucles de cuerpos:

$$BL=\{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9\}.$$

Llegados a este punto, hemos realizado la generación de todos los bucles de cuerpos a partir de los bucles de caras y los vectores normales de un objeto 3D. Antes de acabar este procedimiento tenemos que verificar todos los bucles de cuerpos para comprobar si son interiores o exteriores. Algunos de los bucles de cuerpos generados son ilimitados, y se llaman bucles de cuerpos exteriores. Los otros son interiores porque son finitos. Los bucles de cuerpos exteriores son inútiles para construir objetos, y por eso se descartan. Esta clasificación de los bucles de cuerpos en interiores y exteriores la veremos en el apartado siguiente, ahora sólo se puede decir que los bucles de cuerpos C_1, C_2, C_3, C_4, C_5, C_6, C_7 y C_8 son interiores, y el bucle de cuerpo C_9 es exterior.

2.8.3. CLASIFICACIÓN DE LOS BUCLES DE CUERPOS.

En este apartado vamos a ver el método empleado para la clasificación de los bucles de cuerpos. El algoritmo *BLG* genera todos los bucles de cuerpos a partir de los bucles de caras. Algunos de los bucles de cuerpos generados son ilimitados, y se llaman bucles de cuerpos exteriores. El resto de bucles de cuerpos son interiores, porque son finitos. Los bucles de cuerpos exteriores no son útiles en la construcción de objetos 3D, y por eso se descartan.

Para clasificar los bucles de cuerpos en interiores y exteriores, necesitamos la información referente a la selección del lado para cada bucle de cara del bucle de cuerpo. En un bucle de cuerpo, los lados de todos los bucles de caras seleccionados usando el algoritmo anterior forman las caras interiores del bucle de cuerpo. Como resultado, el interior de un bucle de cuerpo exterior es ilimitado, y el exterior es un sólido vacío finito. Para clasificar los bucles de cuerpos en interiores y exteriores, utilizamos los siguientes pasos:

- Para un bucle de cuerpo, seleccionar dos bucles de caras adyacentes $\oplus F_i$ y $\oplus F_j$ a su arista compartida *e*, y construir un rayo L que empiece en el punto medio de *e* y en la dirección de

$$L = \frac{\oplus n_1}{|n_1|} + \frac{\oplus n_2}{|n_2|}$$

  donde $\oplus n_1$ y $\oplus n_2$ son los vectores normales de $\oplus F_i$ y $\oplus F_j$, respectivamente, y $\oplus$ corresponde a su selección de lado.

- Analizar las situaciones en las cuales la línea L intersecta con un bucle de cara del bucle de cuerpo. Si el número de puntos de intersección formados por L y todos los bucles de caras en un bucle de cuerpo (excluyendo F_i y F_j) es uno o más, entonces el bucle de cuerpo es interior; en otro caso, es exterior.

Para todo bucle de cara F de un bucle de cuerpo, si F es seleccionado por el lado negativo, por ejemplo la cara -F, entonces su hipotético vector normal exterior +n apunta al exterior del bucle de cuerpo, mientras -n apunta al interior. Similarmente, si F es seleccionado por el lado positivo, por ejemplo +F, entonces su hipotético vector normal exterior +n apunta al interior del bucle de cuerpo, mientras -n apunta al exterior. Esto implica que el auténtico vector normal exterior de un bucle de cara -F de un cuerpo es +n, o el auténtico vector normal exterior de un bucle de cara +F en el bucle de cuerpo es -n. Por lo tanto, podemos obtener los auténticos vectores normales exteriores para todas las caras de cualquier cuerpo.

Vamos a realizar un análisis de la fórmula utilizada para el cálculo del rayo L que interviene en la clasificación de los bucles de cuerpos. Tenemos la siguiente fórmula:

$$L = \frac{\oplus n_1}{|n_1|} + \frac{\oplus n_2}{|n_2|}$$

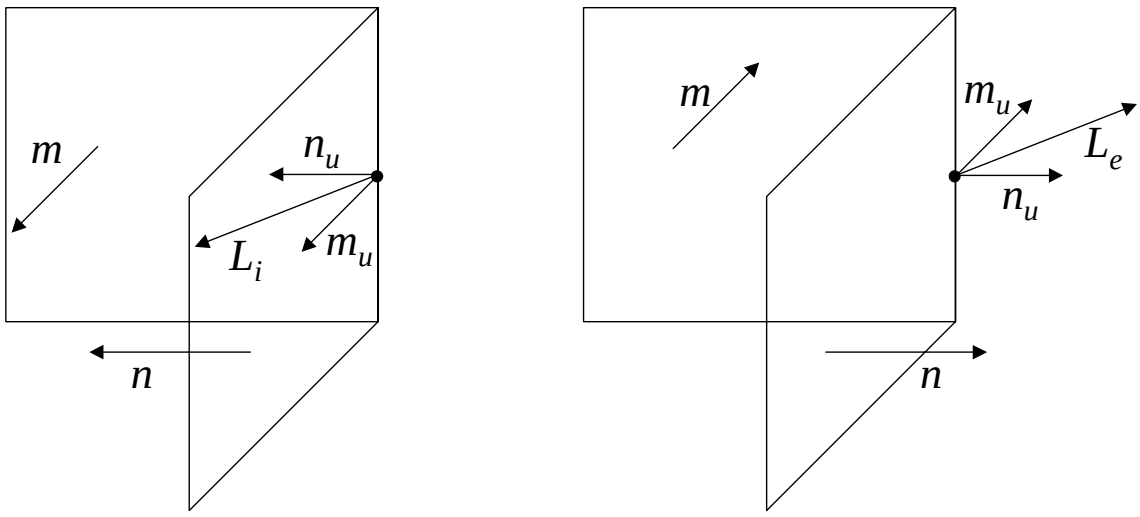
De la fórmula anterior podemos decir que n_1 y n_2 son los vectores normales de un par de bucles de caras adyacentes del mismo bucle de cuerpo. El símbolo $\oplus$ nos indica la selección de lado de cada bucle de cara que utilizamos.

Ahora utilizaremos la siguiente definición. Un *vector unitario* es un vector cuyo módulo es la unidad. Decimos que *normalizar* un vector es conseguir otro de la misma dirección y sentido que el dado, pero de módulo unidad. Para conseguirlo, basta multiplicarlo por el inverso de su módulo:

$$n_u = \frac{n}{|n|}$$

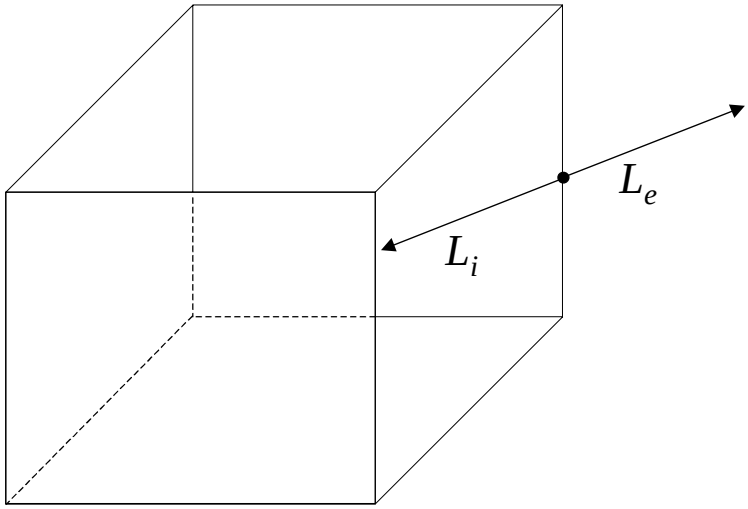
Además, según lo explicado anteriormente, si hemos seleccionado un bucle de cara $\oplus$F, el vector normal $\oplus n$ al bucle de cara F apuntará hacia el interior del bucle de cuerpo.

Por lo tanto podemos decir que el rayo *L* está formado por la suma de dos vectores unitarios y que apuntan al interior del bucle de cuerpo. Por lo tanto, si el bucle de cuerpo es interior, el rayo *L* cruzará una o varias caras del bucle de cuerpo, pero si el bucle de cuerpo es exterior, el rayo *L* no cruzará ninguna cara del bucle de cuerpo. Esto lo podemos comprobar en las figuras siguientes:



En la figura anterior podemos ver como a la izquierda tenemos que los dos vectores normales y el rayo L_i apuntan a una zona del espacio dividida por los dos bucles de caras y a en la derecha los vectores normales y L_e apuntan a la otra zona del espacio.

Podremos comprobar si el bucle de cuerpo es exterior o interior calculando si los rayos *L* cruzan algún bucle de cara. Si a los ejemplos anteriores los situamos en un objeto cualquiera podemos ver los rayos orientados hacia el interior y los rayos orientados hacia el exterior:



Como se puede ver en la figura anterior las caras con las que se ha formado el rayo L_i darán lugar a un bucle de cuerpo interior, mientras que las caras que forman el rayo L_e darán lugar a un bucle de cuerpo exterior. Por lo tanto, ya sabemos como clasificar los bucles de cuerpos en interiores y exteriores.

Llegados a este punto podemos decir que prácticamente se ha visto el algoritmo de reconstrucción. Hemos realizado el proceso de pasar de 2D a 3D, inicialmente, generando vértices y aristas 3D a partir de vértices y aristas 2D; después hemos obtenido todos los planos que se podían generan con las aristas 3D y sus vectores normales; hemos obtenido los bucles de caras a partir de los gráficos planos y ahora hemos generado los bucles de cuerpos a partir de los bucles de caras. En el siguiente apartado trataremos el último punto del algoritmo de reconstrucción, que consiste en la combinación de todos los bucles de cuerpos y su comprobación, para la generación de todos los objetos 3D válidos, ya que a una representación de planta, alzado y perfil se le pueden asociar distintos objetos 3D que la cumplen.

2.9. VERIFICACIÓN DEL OBJETO 3D.

En la sección anterior se han generado todos los bucles de cuerpos recorriendo los bucles de caras mediante el algoritmo *BLG* y eliminando los bucles de cuerpos exteriores del conjunto de bucles de cuerpos. Ahora en este punto vamos a tratar la combinación de estos cuerpos elementales o bucles de cuerpos para la generación de un posible objeto candidato correcto. Aplicaremos a cada objeto candidato un procedimiento de eliminación de caras, aristas y vértices redundantes en la generación del objeto, y finalmente, proyectaremos cada objeto candidato en las tres vistas (planta, alzado y perfil) con el objeto de realizar una comparación con los datos de entrada del

algoritmo. Llamaremos a partir de ahora a este procedimiento como *FT* (*Final Test*, Verificación Final). Este procedimiento lo dividiremos en los tres siguientes bloques: combinación de bucles de cuerpos, corrección del objeto candidato y consistencia del objeto candidato con las tres vistas de los datos de entrada.

El modo de actuar en este procedimiento será el siguiente:

1. Generar un objeto candidato a partir de una combinación de bucles de cuerpos.

2. Corregir el objeto candidato, es decir, eliminar una serie de caras, aristas o vértices que son redundantes en la generación del objeto.

3. Comprobar la consistencia del objeto candidato con las tres vistas de los datos de entrada. Esto consiste en proyectar el objeto candidato sobre las tres vistas y comparar con los datos de entrada. Si son iguales, entonces hemos descubierto un objeto válido.

4. Si quedan combinaciones de bucles de cuerpos, entonces pasar al punto 1. En otro caso, acaba el procedimiento.

2.9.1. GENERACIÓN DE OBJETOS CANDIDATOS.

En este apartado vamos a ver como se generan las combinaciones de bucles de cuerpos para obtener posteriormente un objeto candidato. Para los bucles de cuerpos generados anteriormente por el procedimiento *BLG*, comprobamos todas las posibles combinaciones de esos bucles de cuerpos y para garantizar que podemos descubrir todos los objetos que encajan con las tres vistas originales de los datos de entrada. Si las proyecciones del objeto candidato son consistentes con sus correspondientes tres vistas, el objeto candidato es una solución de las vistas. Dados *N* bucles de cuerpos, el número total de objetos candidato o combinaciones de bucles de cuerpos es 2^N-1.

La solución adoptada para la obtención de las combinaciones es la siguiente: si suponemos un número binario de *N* bits o dígitos binarios, y suponemos que tenemos *N* bucles de cuerpos, entonces cada número binario desde el 00...00 hasta el 11...111 nos da una combinación de bucles de cuerpos, donde el valor 0 en la posición *i* nos indica que el bucle de cuerpo *i* no estará incluido en ese objeto candidato y el valor 1 en la posición *i* nos indica que el bucle de cuerpo *i* está incluido en ese objeto candidato. De las 2^N combinaciones posibles hay que restarle una (00...00), por lo tanto tenemos todas las combinaciones de bucles de cuerpos que generan objetos candidatos.

Por lo tanto, suponiendo que tenemos cuatro bucles de cuerpos (b_1, b_2, b_3, b_4) tenemos las siguientes combinaciones de bucles de cuerpos:

Conta.	**b_1 b_2 b_3 b_4**
1	0001
2	0010

3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Nosotros tendremos un bucle que tendrá un contador que recorrerá la tabla. En cada pasada del bucle el valor del contador en binario nos indicará los bucles de cuerpos que tendrá el objeto candidato. El objeto candidato será una variable de tipo registro que contendrá los bucles de cuerpos que forman el objeto candidato según la variable contador. A este objeto candidato le aplicaremos los siguientes dos apartados de este procedimiento de verificación, pasando luego a generar otra combinación.

2.9.2. CORRECCIÓN DEL OBJETO CANDIDATO.

En este apartado, vamos a realizar una verificación de la corrección del objeto candidato, además trataremos de descubrir los bucles de caras, aristas y vértices redundantes entre los distintos bucles de cuerpos que componen el objeto candidato. Veamos ahora la información aportada por el artículo de Qing-Wen Yan:

"Desde que un bucle de cuerpo tiene una única orientación en el espacio, la relación entre cualesquier dos bucles de cuerpos puede ser clasificada en los cuatro casos siguientes:

- *Caso 1: Dos bucles de cuerpos separados, figura 14a.*
- *Caso 2: Dos cuerpos compartiendo algunos vértices, figura 14b.*
- *Caso 3: Dos cuerpos compartiendo algunas aristas, figura 14c.*
- *Caso 4: Dos cuerpos compartiendo algunas caras, figura 14d.*

Para cada objeto candidato, verificamos su corrección en función a las siguientes reglas.

(1) [Eliminar bucles de caras redundantes]. Eliminar cada bucle de cara para el cual ambos lados fueron seleccionados en el mismo sólido, desde que sabemos que esos bucles de caras son las caras interiores del sólido, y no la superficie del sólido. La cara F_2 en la figura 15a, por ejemplo, es redundante.

(2) [Eliminar aristas redundantes]. Eliminar esas aristas que son compartidas por dos y sólo dos caras coplanares. Después de eliminarlas, unir las dos caras en una, formando un nuevo bucle de cara del sólido (figura 15b).

(3) [Descartar el candidato incorrecto usando la información de los vértices]. Si dos caras sólo comparten algunos vértices, este objeto candidato no es regular, y es incorrecto.

(4) [Descartar el candidato incorrecto usando la información de las aristas]. Puesto que cada arista pertenece a dos y sólo dos caras del objeto sólido, si una arista tiene más de dos caras adyacentes, y la conexión de una arista es más de dos, este objeto es también no regular.

Cada candidato correcto se examinará más para la consistencia de sus proyecciones con las vistas."

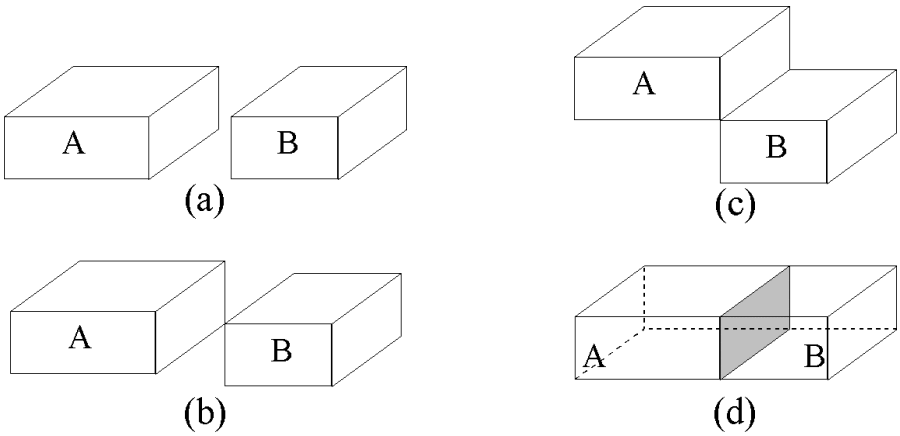


Figura 14: *Relación entre dos objetos en el espacio.*

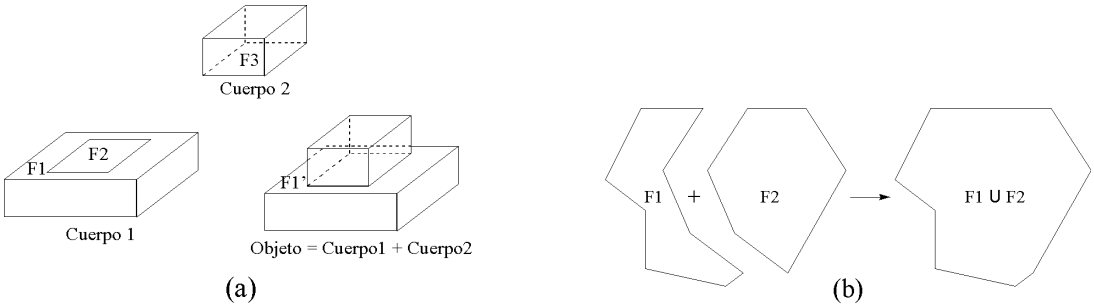


Figura 15: *Algunas situaciones especiales para la construcción de objetos; (a) eliminación de caras duplicadas, (b) eliminación de aristas duplicadas.*

Hay que añadir que se ha realizado una modificación a este apartado debido a su incorrección en la práctica. Esto es debido a un problema en la ordenación de los pasos que solucionan la corrección del objeto candidato. Según lo expuesto en el artículo de Qing-Wen Yan, este procedimiento se soluciona aplicando secuencialmente los pasos

(1)-(2)-(3)-(4), pero esto tiene un problema. Si se aplica el punto (4), que trata de descartar el objeto candidato incorrecto usando la información de las aristas, posteriormente al punto (2), que elimina las aristas redundantes, el algoritmo no funciona bien. Por lo tanto se ha reordenado la secuencia de la forma (1)-(4)-(2)-(3), teniendo un resultado óptimo.

2.9.3. CONSISTENCIA DEL OBJETO CANDIDATO CON LAS VISTAS.

En este apartado, vamos a realizar una verificación de la corrección del objeto candidato comprobando la consistencia del objeto con los datos de entrada, para ello tendremos que proyectar el objeto candidato sobre las tres vistas (planta, alzado y perfil) y compararlo con los datos de entrada. Veamos ahora la información aportada por el artículo de Qing-Wen Yan:

"Primero, un objeto candidato se proyecta a los correspondientes planos de proyección (xOz, xOy, yOz), y se forman tres nuevas vistas. Segundo, comparar las tres nuevas vistas ortográficas con las vistas originales. Si son iguales completamente, este objeto es una solución.

Durante la proyección, una arista 3D puede llegar a ser un vértice 2D o una arista 2D. Necesitamos considerar los siguientes pasos. Si una arista 3D no está bloqueada por algún plano, la proyección de esta arista 3D es una arista sólida; en otro caso, es una arista discontinua. Si una arista discontinua solapa con una arista sólida, las secciones solapadas podrían ser sólidas. Por lo tanto, las aristas 3D pueden ser proyectadas en una combinación de varias aristas sólidas y varias aristas discontinuas.

Con las aristas 2D de la proyección, unimos las aristas adyacentes del mismo tipo (sólidas o discontinuas). Comparar estas aristas con los de los datos de entrada. Si las aristas se solapan completamente y son iguales, el objeto construido es una solución correcta. En otro caso, no hay solución."

Ahora veremos un esquema del algoritmo aplicado para resolver este procedimiento:

1. GENERACIÓN DE PROYECCIONES:

1.1. Seleccionar una arista 3D del objeto.

1.2. Proyección de planta:

1.2.1. Proyectar la arista 3D sobre el plano xOy (planta).

1.2.2. Verificar si hay planos en sentido +Z que la ocultan.

1.2.3. Si existen planos que la ocultan pasará a ser una arista 2D discontinua. En otro caso, pasará a ser una arista 2D continua.

1.2.4. Si ya existía la arista 2D, comprobar si una es continua y la otra discontinua. En este caso, se modificaría el atributo a arista continua.

1.2.5. Si no existía la arista 2D, guardar en una lista de aristas de la planta.

1.3. Proyección de alzado:

1.3.1. Proyectar la arista 3D sobre el plano xOz (alzado).
1.3.2. Verificar si hay planos en sentido +Y que la ocultan.
1.3.3. Si existen planos que la ocultan pasará a ser una arista 2D discontinua. En otro caso, pasará a ser una arista 2D continua.
1.3.4. Si ya existía la arista 2D, comprobar si una es continua y la otra discontinua. En este caso, se modificaría el atributo a arista continua.
1.3.5. Si no existía la arista 2D, guardar en una lista de aristas del alzado.

1.4. Proyección de perfil:

1.4.1. Proyectar la arista 3D sobre el plano yOz (perfil).
1.4.2. Verificar si hay planos en sentido +X que la ocultan.
1.4.3. Si existen planos que la ocultan pasará a ser una arista 2D discontinua. En otro caso, pasará a ser una arista 2D continua.
1.4.4. Si ya existía la arista 2D, comprobar si una es continua y la otra discontinua. En este caso, se modificaría el atributo a arista continua.
1.4.5. Si no existía la arista 2D, guardar en una lista de aristas del perfil.

1.5. Buscar todos los vértices que estén alineados por aristas 2D y generar todas las combinaciones.

2. VERIFICACIÓN DE PROYECCIONES.

Comprobar que todas las aristas de los vectores de planta, alzado y perfil de los datos de entrada, estén en los vectores de planta, alzado y perfil proyectados en el paso anterior, y no tiene que sobrar ninguna arista 2D.

Explicado esto, podemos decir que se ha definido minuciosamente el algoritmo de reconstrucción 3D de objetos a partir de las vistas de planta, alzado y perfil de Qing-Wen Yan. Por lo tanto cuando acaba el algoritmo, tenemos una serie de objetos válidos definidos a partir de sus caras que pueden representarse gráficamente en la pantalla mediante un algoritmo de ocultación de caras, o utilizar estos datos para aplicarle algún otro algoritmo de sombreado, modelos de iluminación, etc.

Capítulo 3
ESTRUCTURA DE DATOS Y ALGORITMOS

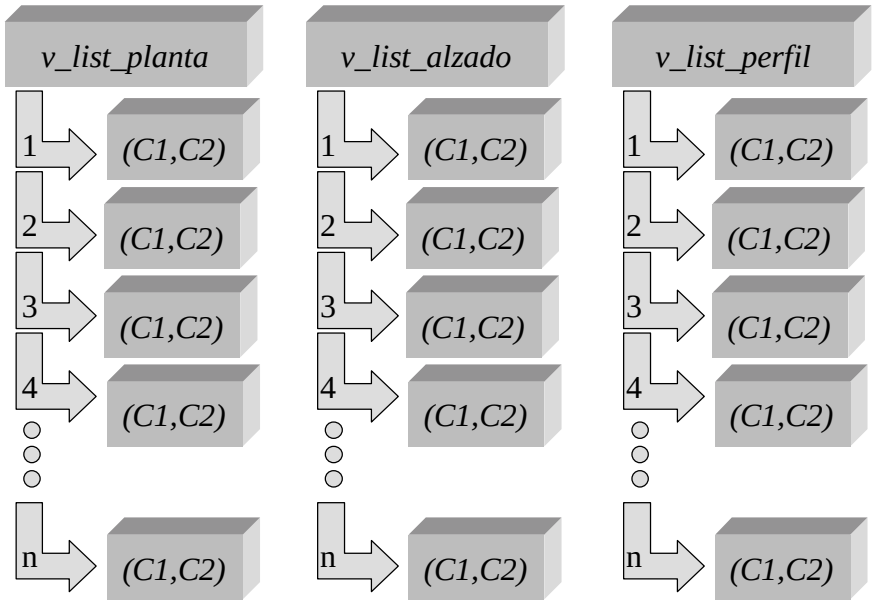
En este capítulo se va a incidir de un modo descriptivo sobre las estructuras de datos y los algoritmos construidos para la implementación del algoritmo mediante un lenguaje de programación. Además, se realizará un estudio teórico de los costes temporales de las diferentes partes que componen el algoritmo de reconstrucción 3D.

3.1. DESCRIPCIÓN DE ESTRUCTURAS DE DATOS.

El acceso a los datos para realizar los cálculos pertinentes es de vital importancia para la eficiencia del algoritmo. Con buena distribución de los datos podemos, además, controlar la seguridad de los datos y que estos no interactúen con otros elementos del programa. En este apartado se va a definir las estructuras de datos utilizadas por el algoritmo para almacenamiento y cálculo de distintos datos empleados en su resolución. Las estructuras de datos constarán pues de combinaciones de tipos de datos básicos formando una estructura más compleja de datos simples relacionados entre sí.

A la hora de construir las estructuras tenemos que diferenciar los vértices y aristas bidimensionales de los vértices y aristas tridimensionales.

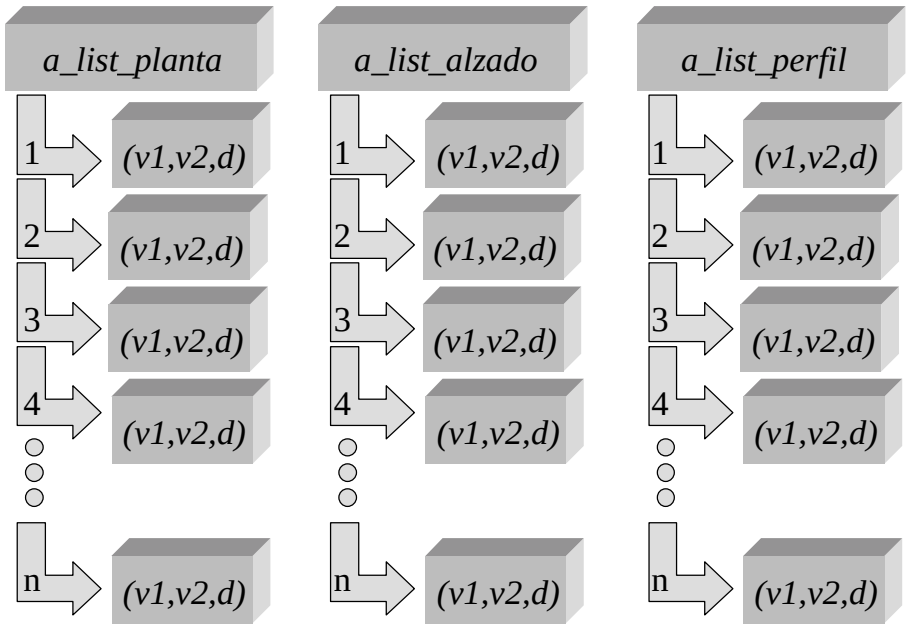
Por lo tanto como tenemos tres vistas ortográficas (planta, alzado y perfil), tenemos que tener tres listas de vértices 2D y tres listas de aristas 2D. Con el orden o índice dentro de la lista podemos nombrar los vértices y las aristas 2D, por ejemplo, la arista 15 de la proyección perfil será la que ocupe la posición 15 dentro de la lista de aristas de la proyección perfil, que por ejemplo estará formada por los vértices 2 y 8, que serán los vértices que ocupen la posición 2 y 8, respectivamente, dentro de la lista de vértices 2D de la proyección perfil. Además, en las listas de aristas 2D tendremos otro campo que será el atributo de tipo de línea (continua o discontinua) dentro de cada proyección. En la figura siguiente vemos las estructuras de las listas de vértices 2D:



Aquí tenemos las tres listas de vértices 2D que se llamarán *v_list_planta*, *v_list_alzado*, *v_list_perfil* que serán respectivamente como su nombre indica las listas de la planta, alzado y perfil. Cada elemento básico o celda de esta lista está compuesto por dos variables (C1, C2). Estas dos variables serán las coordenadas del vértice dentro de cada proyección, y serán del tipo de datos REAL. Así pues, tendremos que las coordenadas C1 y C2 en la proyección planta, en realidad, son las coordenadas X e Y, respectivamente, del vértice 2D dentro de esta proyección. Las coordenadas C1 y C2 en la proyección alzado, en realidad, son las coordenadas X y Z, respectivamente, del vértice 2D dentro de esta proyección. Y las coordenadas C1 y C2 en la proyección perfil, en realidad, son las coordenadas Y y Z, respectivamente, del vértice 2D dentro de esta proyección.

Por lo tanto cuando queramos hacer referencia al vértice 12 de la proyección alzado, este vértice será el que ocupe la posición 12 dentro de la lista de vértices de la proyección alzado, y haremos referencia a él mediante *v_list_alzado[12]*. Cuando queramos referenciar las coordenadas de este mismo vértice utilizaremos *v_list_alzado[12].C1* y *v_list_alzado[12].C2*.

Ahora en la figura siguiente vemos las estructuras de las listas de aristas 2D:



Cada entrada de la lista de aristas nos proporciona los datos: *(v1, v2, discont)*, donde, para la planta, *v1* y *v2* serán los índices de la lista de vértices *v_list_planta*; para el alzado, *v1* y *v2* serán los índices de la lista de vértices *v_list_alzado*; y para el perfil, *v1* y *v2* serán los índices de la lista de vértices *v_list_perfil;* para cualquier caso la variable *discont* será una variable booleana que nos indica si la arista es discontinua (*true)* o continua (*false)*. También seleccionaremos la arista deseada según su índice dentro de la lista de aristas. Nombraremos las tres listas con los siguientes nombres: *a_list_planta, a_list_alzado y a_list_perfil.*

Por lo tanto cuando queramos hacer referencia a la arista 8 de la proyección planta, esta arista será la que ocupe la posición 8 dentro de la lista de aristas de la proyección planta, y haremos referencia a ella mediante *a_list_planta[8]*. Cuando queramos referenciar los vértices de esta misma arista 2D y su *"flag"* de línea discontinua, utilizaremos *a_list_planta[8].v1*, *a_list_planta[8].v2* y *a_list_planta[8].d*.

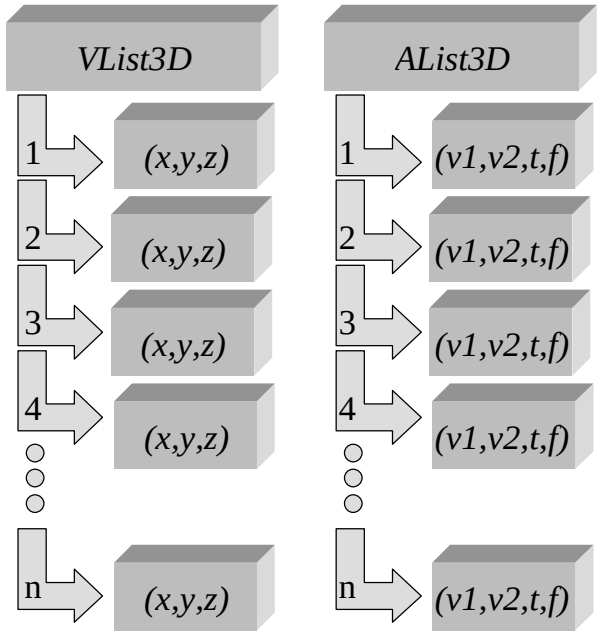
Y veamos como se definen en C estas estructuras:

```
#define MaxVistas  200

struct Vertice
  { float Coord1, Coord2; };

struct Arista{
        int v1, v2;
        char Tipo;
        };

extern struct Vertice  V_List_Planta[MaxVistas], V_List_Alzado[MaxVistas], V_List_Perfil[MaxVistas];

extern struct Arista   A_List_Planta[MaxVistas], A_List_Alzado[MaxVistas], A_List_Perfil[MaxVistas];
```

Ahora vamos a pasar a describir las estructuras de datos 3D. En primer lugar tenemos que nombrar a las listas de vértices y aristas 3D, después tendremos que describir una estructura de datos para almacenar los planos y los bucles de caras, pasando por último a describir una estructura de datos para almacenar los bucles de cuerpos y los objetos válidos.

Las listas de vértices y aristas 3D, *VList3D* y *AList3D* respectivamente, contienen todas las posibles aristas y vértices 3D que se pueden generar mediante el algoritmo de reconstrucción.

Aquí podemos ver un esquema de la estructura de estas listas:

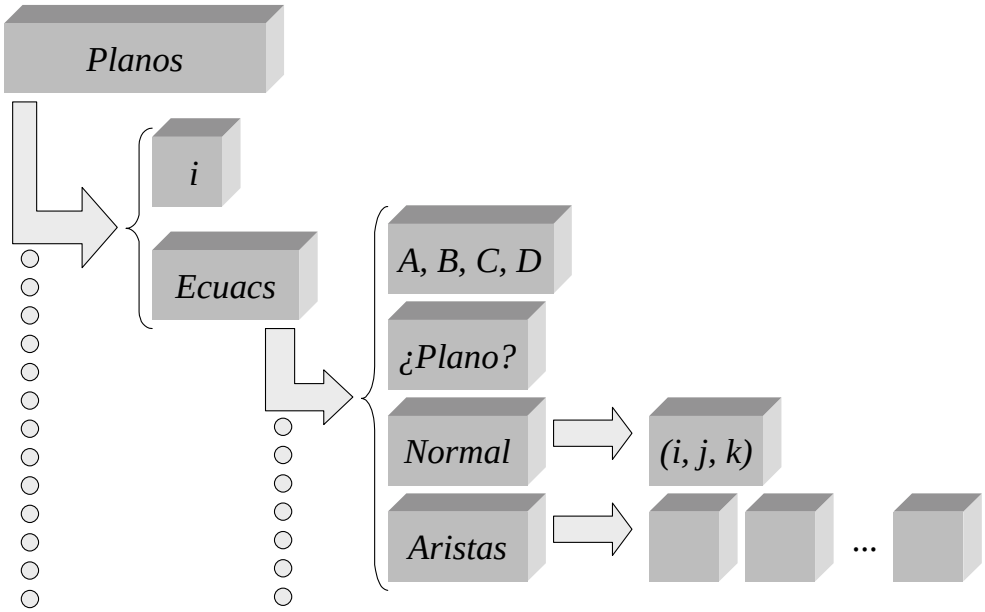


Cada entrada de la lista de vértices 3D, almacena las tres coordenadas x, y, z de los vértices generados, pudiendo hacer referencia a cada vértice mediante el índice de la lista. En la lista de aristas, en cada entrada se almacena dos índices, *v1*, *v2*, que hacen referencia a los dos vértices de la lista *VList3D* que forman la arista, y además tenemos dos variables más, *t* (tipo) y *f* (flag), que se utilizarán mediante el algoritmo para realizar marcas especiales sobre las aristas 3D.

Y, por lo tanto, veamos como se definen en C estas estructuras:

```
#define MaxA     200
#define MaxV     100

struct Vertice3D
 { double x,y,z;};
struct Arista3D
 { int v1,v2; char Tipo,Flag;};

extern struct Vertice3D  VList3D[MaxV];
extern struct Arista3D   AList3D[MaxA];
```

Veamos ahora gráficamente la estructura de almacenamiento de los planos:

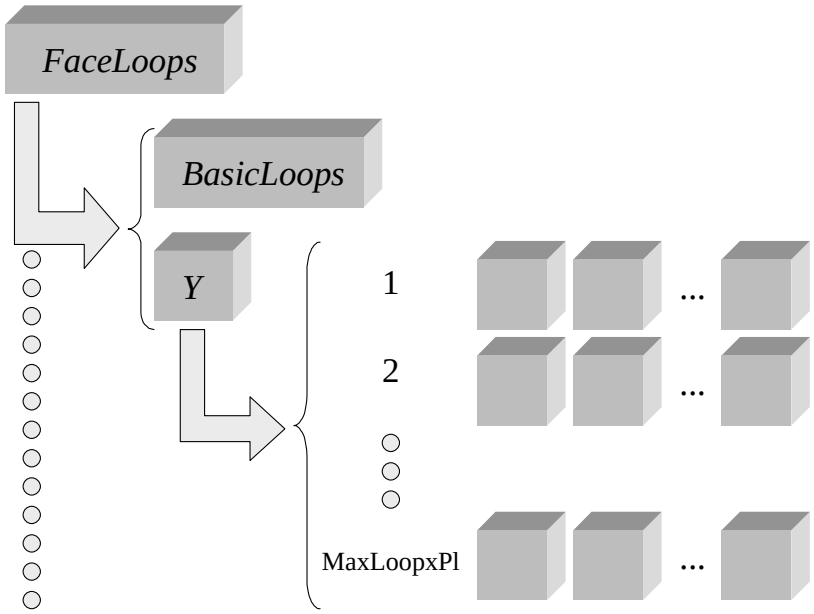


Tenemos que la estructura *Planos* es una lista con tantos componentes como vértices 3D existan, por lo tanto el elemento 5 de la lista serán todos los planos que se pueden generar con el vértice 3D número 5, y dentro de este elemento de la lista tenemos referenciado con la variable *i* el número de planos que se generan con el vértice número 5 y en la variable *Ecuacs* será la lista donde tenemos definidos los planos que se generan con el vértice 5. La definición de los planos viene dada por: las variables *A*, *B*, *C*, *D* que serán los coeficientes de la ecuación general del plano; el hipotético vector normal exterior al plano y una lista de las aristas que están incluidas en este plano. Además, la variable *¿Plano?* Será de tipo booleano y la utilizaremos para validar o invalidar el plano.

Veamos como se definen en C esta estructura:

```
#define MaxV  100
#define PLporV     10

struct N
{ double i,j,k;};

struct EcuacionPlano
{ double a,b,c,d; int Plano;
  struct N Normal;
  char Aristas[20];
};

struct Planos
{ int i;
  EcuacionPlano Ecuacs[PLporV];
};

extern struct Planos huge Pl[MaxV];
```

En la siguiente figura podemos ver la estructura de almacenamiento de los bucles de caras:



En la figura anterior podemos ver que la estructura de almacenamiento de los bucles de caras es tipo lista, donde la variable encargada del almacenamiento se llama *FaceLoops*. Cada elemento de la lista tiene la posibilidad de almacenar varios bucles de caras que estén contenidos en el mismo plano. En concreto la variable *BasicLoops* nos dice cuantos bucles se han formado con este plano y en la variable *Y* es una variable tipo array o matriz donde para cada bucle de cara se almacenan sus aristas asociadas. Este número de bucles de caras por plano viene limitado por la constante MaxLoopxPl.

Implementando en C la estructura, queda de la forma:

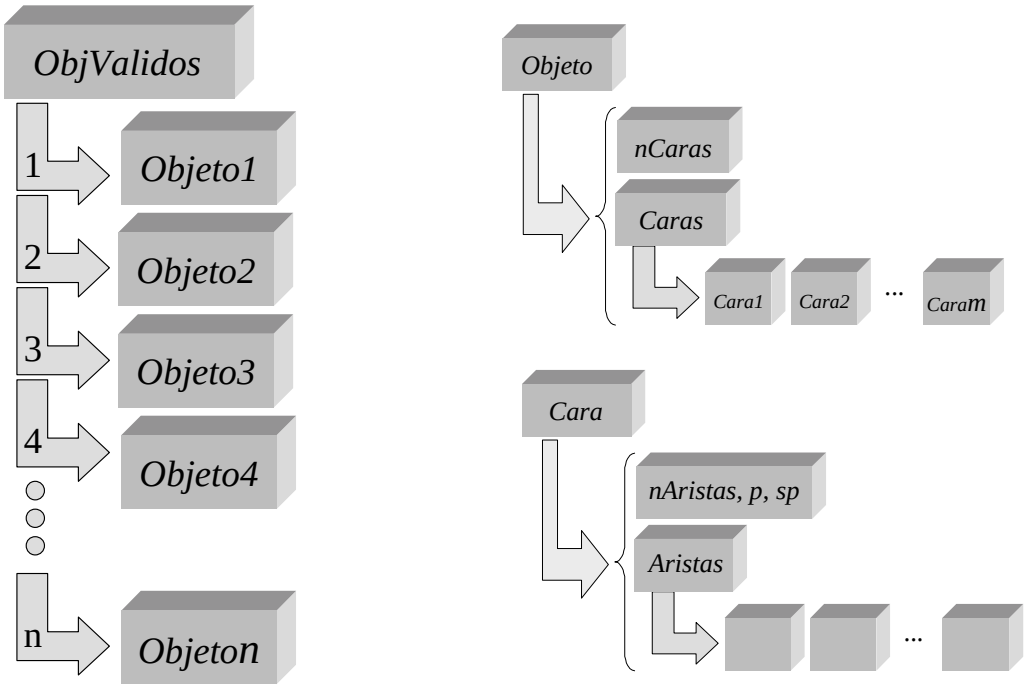
```
#define MaxVerxPl  30
#define MaxArixV   20
#define MaxLoopxPl 15
#define MaxBucCara 80

struct CBUCLES
{ int BasicLoops;
  int Y[MaxLoopxPl][MaxVerxPl+1];
 };

extern struct CBUCLES  huge FaceLoops[MaxBucCara];
```

Ahora vamos a pasar a describir la estructura de datos utilizada para el almacenamiento de los bucles de cuerpos generados por el procedimiento *BLR* del algoritmo de reconstrucción 3D.

Podemos decir que los bucles de cuerpos estarán almacenados en una estructura tipo lista llamada *BodyLoops* y en cada elemento de la lista se almacena un bucle de cuerpo, o cuerpo elemental. Por lo tanto, estos elementos que almacenan el bucle de cuerpo constarán de una variable que nos indica el número de caras que componen el bucle de cuerpo y de una lista de índices a la estructura de bucles de caras que nos están indicando las caras que componen el bucle de cuerpo. Esta estructura escrita en C quedaría de la siguiente forma:

```
#define MinBBxO        4
#define MaxBBxO        60
#define MaxBucObj      60

struct OBUCLES
{ int NBucCaras;
  int Caras[MaxBBxO];
};

extern struct OBUCLES huge BodyLoops[MaxBucObj];
```

Para almacenar los objetos válidos utilizaremos la estructura de la figura siguiente:



Aquí tenemos que la lista *ObjValidos* está formada por unos elementos que van a almacenar un objeto válido (*Objeto*). Este elemento está formado por una variable que indica el número de caras que tiene el objeto válido y otra variable (*Caras*) que forma una lista de elementos, que cada uno nos describirá una de las caras que forman el objeto. A su vez, estos elementos están formados con: una variable que indica el número de aristas que forman la cara (*nAristas*), dos índices (*p* y *sp*) que nos relacionan esta cara con su plano asociado mediante la estructura de planos y, por último, una lista de aristas que forman la cara (*Aristas*).

A continuación podemos ver la definición en C de esta estructura:

```
#define MaxVerxPl  30
#define MaxBucCara 80
#define MaxObjVal  20

struct Cara
{ int nAristas,p,sp;
  int Aristas[MaxVerxPl];
};

struct TipoObjeto
{ int nCaras;
  struct Cara Caras[MaxBucCara];
};

extern struct TipoObjeto huge ObjValidos[MaxObjVal];
```

Por lo tanto, ya hemos visto las principales características de las principales estructuras de datos para informar de como están organizados los datos, así como de las restricciones que imponen las estructuras de datos sobre los datos de entrada y sus características 3D. Una mejora de esta aplicación vendría dada por la reestructuración de los tipos de datos de forma que se utilizase memoria dinámica, ya que esto eliminaría muchas restricciones de las impuestas por el uso de variables estáticas.

3.2. ANÁLISIS DE COSTES DE LOS ALGORITMOS.

Como indica su propia definición, un algoritmo es un conjunto de reglas básicas que permiten resolver de forma metódica un problema. También se puede definir como una lista de instrucciones que especifican una secuencia de operaciones deterministas que llevadas a cabo por un agente ejecutor (modelo computacional) da una solución a cualquier instancia de un problema determinado en un tiempo finito. Lo que intentaremos es estimar el orden de magnitud de los recursos computacionales que requiere un algoritmo para resolver un problema. Los recursos computacionales se pueden dividir en dos grandes apartados: Tiempo y Espacio.

En este apartado vamos a ver una breve descripción de los algoritmos empleados en la reconstrucción 3D mediante pseudocódigo o lenguaje algorítmico. El propósito de esta descripción es realizar un estudio de la complejidad temporal (o costes temporales) de los algoritmos, por lo tanto la descripción de los algoritmos va a ser muy básica.

Se han establecido ocho procedimientos que realizan las funciones principales del algoritmo de reconstrucción. Estos procedimientos están interconectados entre sí y cualquier resultado o parámetro de uno servirá para realizar cálculos en los posteriores. La secuencia ordenada de estos procedimientos es la siguiente:

1. Preprocesado de datos.
2. Generación del modelo alámbrico.
3. Generación de planos.
4. Análisis de la información de aristas discontinuas.

5. Generación de caras.
6. Obtención de vértices y aristas de corte.
7. Generación de bucles de cuerpos.
8. Combinación de bucles de objetos y verificación.

Una vez establecidos los pasos, pasamos a describir cada uno de los algoritmos en pseudocódigo.

Función PreprocesadorDatos(listas de vértices 2D de los datos de entrada)

Variables

LV:ListaVertices2D;

Inicio

Para i=1 **hasta** NV_{planta} **hacer**
Para j=i+1 **hasta** NV_{planta} **hacer**
Para k=j+1 **hasta** NV_{planta} **hacer**
Si VerticesColineales(i,j,k) **entonces** LV=AñadeVertices(i,j,k);
FinSi
FinPara
EliminarVerticesNoUnidos(LV);
GenerarAristasPlanta(LV);
FinPara
FinPara

Para i=1 **hasta** NV_{alzado} **hacer**
Para j=i+1 **hasta** NV_{alzado} **hacer**
Para k=j+1 **hasta** NV_{alzado} **hacer**
Si VerticesColineales(i,j,k) **entonces** LV=AñadeVertices(i,j,k);
FinSi
FinPara
EliminarVerticesNoUnidos(LV);
GenerarAristasAlzado(LV);
FinPara
FinPara

Para i=1 **hasta** NV_{perfil} **hacer**
Para j=i+1 **hasta** NV_{perfil} **hacer**
Para k=j+1 **hasta** NV_{perfil} **hacer**
Si VerticesColineales(i,j,k) **entonces** LV=AñadeVertices(i,j,k);
FinSi
FinPara
EliminarVerticesNoUnidos(LV);
GenerarAristasPerfil(LV);
FinPara
FinPara

Fin.

Vamos a definir una serie de funciones de carácter asintótico para poder definir la complejidad temporal de los algoritmos. Sea $t:N \rightarrow R^{\geq 0}$ una función de coste o el

tiempo de ejecución del algoritmo. Definimos orden de f(n) como: $O(f(n))=\{ t:N\rightarrow R^{\geq 0} / (\exists c\in R^{\geq 0}) (\exists n_0\in N) \forall n\geq n_0 [t(n)\leq c.f(n)] \}$, podemos decir que t crece a lo sumo como f(n) que es equivalente al peor caso. También definimos $\Omega(f(n))= \{ t:N\rightarrow R^{\geq 0} / (\exists c\in R^{\geq 0}) (\exists n_0\in N) \forall n\geq n_0 [t(n)\geq c.f(n)] \}$, que quiere decir que t crece al menos como f(n) que es equivalente al mejor caso. Y por último, definimos $\theta(f(n))= \{ t:N\rightarrow R^{\geq 0} / (\exists c_1,c_2\in R^{\geq 0}) (\exists n_0\in N) \forall n\geq n_0 [c_1.f(n)\leq t(n) \leq c_2.f(n)] \}$, o también, $\theta(f(n))= O(f(n)) \cap \Omega(f(n))$ que es equivalente ni al mejor, ni al peor caso.

Tenemos que NV_{planta}, NV_{alzado} y NV_{perfil} son datos que nos indican el número de vértices 2D de las proyecciones de planta, alzado y perfil, respectivamente. Además podemos decir que las funciones *AñadeVertices* y *EliminarVerticesNoUnidos* tienen un coste temporal unitario; la función *VerticesColineales* tiene un coste temporal equivalente a 3 operaciones en coma flotante (una resta y dos multiplicaciones); y la función *GenerarAristas* tiene un coste temporal que depende de la longitud de la lista LV (que llamamos N_{LV}) con la siguiente formula: $N_{LV}!/(2!.(N_{LV}-2)!)$ que son las combinaciones de N_{LV} elementos de dos en dos, pero no se realiza ninguna operación en coma flotante. Por lo tanto tenemos que el procedimiento debido a los bucles "para" que posee y a que la carga temporal viene impuesta por las operaciones en coma flotante, si llamamos NV al número de vértices 2D total, tenemos un coste temporal de: $O(NV^3)$.

Función ConstruccModeloAlámbrico(listas de vértices y aristas 2D)

Inicio

```
RecorrerArbol(datos2D);
RER(datos3D);
PEVR(datos3D);
```

Fin.

Función RecorrerArbol(datos2D)

Inicio

```
Para i=1 hasta NA_alzado hacer
   Si EsCaso1(i) entonces
      Para j=1 hasta NA_planta hacer
         Si EsCaso1(j) entonces
            Para k=1 hasta NV_perfil hacer
               Si EsCaso4(k) entonces
                  GenerarVertices3D(i,j,k);
                  GenerarArista3D(i,j,k);
               FinSi
            FinPara
         FinSi
         Si EsCaso3(j) entonces
            Para k=1 hasta NA_perfil hacer
               Si EsCaso5(k) entonces
                  GenerarVertices3D(i,j,k);
                  GenerarArista3D(i,j,k);
```

```
                FinSi
            FinPara
        FinSi
    FinPara
FinSi
Si EsCaso2(i) entonces
    Para j=1 hasta NA_perfil hacer
        Si EsCaso2(j) entonces
            Para k=1 hasta NV_planta hacer
                Si EsCaso4(k) entonces
                    GenerarVertices3D(i,j,k);
                    GenerarArista3D(i,j,k);
                FinSi
            FinPara
        FinSi
        Si EsCaso3(j) entonces
            Para k=1 hasta NA_planta hacer
                Si EsCaso5(k) entonces
                    GenerarVertices3D(i,j,k);
                    GenerarArista3D(i,j,k);
                FinSi
            FinPara
        FinSi
    FinPara
FinSi

Si EsCaso3(i) entonces
    Para j=1 hasta NA_planta hacer
        Si EsCaso1(j) entonces
            Para k=1 hasta NA_perfil hacer
                Si EsCaso2(k) entonces
                    GenerarVertices3D(i,j,k);
                    GenerarArista3D(i,j,k);
                FinSi
            FinPara
        FinSi
        Si EsCaso3(j) entonces
            Para k=1 hasta NA_perfil hacer
                Si EsCaso3(k) entonces
                    GenerarVertices3D(i,j,k);
                    GenerarArista3D(i,j,k);
                FinSi
            FinPara
        FinSi
    FinPara
FinSi
FinPara

Para i=1 hasta NV_alzado hacer
    Si EsCaso4(i) entonces
```

```
            Para j=1 hasta NA_planta hacer
               Si EsCaso5(j) entonces
                  Para k=1 hasta NA_perfil hacer
                     Si EsCaso5(k) entonces
                        GenerarVertices3D(i,j,k);
                        GenerarArista3D(i,j,k);
                     FinSi
                  FinPara
               FinSi
            FinPara
         FinSi
      FinPara
```

Fin. /* Fin del procedimiento RecorrerArbol */

Función RER (listas de vértices y aristas 3D)

Inicio

```
   Para i=1 hasta NA_3D hacer
      Para j=1 hasta NA_3D hacer
         Si ColinealesySolapadas(i,j) entonces
            ObtenerAristasNuevas(i,j);
            EliminarAristas(i,j);
         FinSi
      FinPara
   FinPara
```

Fin.

Función PEVR (listas de vértices y aristas 3D)

Inicio

```
   Para i=1 hasta NV_3D hacer
      V=CalcularValenciaDelVertice(i);
      Si V=0 entonces EliminarVertice(i); FinSi
      Si V=1 entonces EliminarVertice(i); EliminarAristasDe(i); FinSi
      Si V=2 y AristasColineales() entonces EliminarVertice(i);EliminarAristasDe(i);
      FinSi
      Si V=3 entonces ResolverCasoV3(i); FinSi
      Si V>3 entonces ResolverCasoV4(i); FinSi
   FinPara
```

Fin.

Tenemos que NV_{planta}, NV_{alzado} y NV_{perfil} son datos que nos indican el número de vértices 2D de las proyecciones de planta, alzado y perfil, respectivamente. NA_{planta}, NA_{alzado} y NA_{perfil} son datos que nos indican el número de aristas 2D de las proyecciones de planta, alzado y perfil, respectivamente. Y, por tanto, NV_{3D} y NA_{3D} son variables que

nos indican el número de vértices y aristas 3D, respectivamente, que tenemos en las listas de vértices y aristas 3D. Como se puede ver en el primer procedimiento hemos descompuesto la construcción del modelo alámbrico en tres procedimientos más: *RecorrerArbol*, *RER* y *PEVR*. Por lo tanto, el coste del procedimiento será la suma de los costes de los tres procedimientos anteriores.

Pasamos primero a analizar el procedimiento *RecorrerArbol*. Las funciones *EsCasoX* son funciones que comprueban los casos de aristas y vértices 2D, siendo ciertas cuando cumplen las condiciones vistas en el capítulo 2 de esta memoria, y además sabemos no tienen coste alguno. Además sabemos que las funciones de generación de vértices y aristas 3D tienen coste unitario. Por lo tanto con estos datos, tenemos que el coste del procedimiento de recorrido del árbol de decisión tiene un coste: $\theta(NA_{alzado} . NA_{planta} . NA_{perfil})$.

Ahora analizaremos el procedimiento *RER*. En este procedimiento la carga temporal viene impuesta por la función *ColinealesySolapadas* en la cual en el mejor caso se realiza: una raíz cuadrada, dos sumas nueve restas y nueve multiplicaciones; y en el peor caso: las operaciones anteriores más nueve restas y seis multiplicaciones. El coste que daríamos para este procedimiento será: $O(NA_{3D}^2)$.

Por último, tenemos el procedimiento *PEVR*. Tenemos que las funciones *EliminarVertice* y *EliminarAristasDe* no tienen casi carga temporal sobre el procedimiento, luego la suponemos nula. Por lo tanto, las funciones que gastan el mayor tiempo de este procedimiento son *AristasColineales*, que hemos visto antes las operaciones que realiza, *ResolverCasoV3* y *ResolverCasoV4* que realizan varios cálculos de colinearidad y coplanaridad. Por lo tanto tenemos que el coste temporal de *PEVR* será: $\theta(NV_{3D} . NA_{3D})$.

Función GenerarPlanos(listas de vértices y aristas 3D)

Inicio

```
InicializaciónyObtenerAristasAdyacentes(aristas 3D);
ConstruirPlanos();
EliminarPlanosDuplicados();
AsignarAristasAPlanos();
VerificarPlanos();
```

Fin.

El primer procedimiento, *InicializaciónyObtenerAristasAdyacentes*, en el que hemos dividido al procedimiento de generación de planos tiene una complejidad temporal pequeña del orden de $O(NA_{3D})$. En el siguiente procedimiento, *ConstruirPlanos*, ya se empiezan a utilizar las operaciones en coma flotante para el cálculo de los coeficientes de las ecuaciones de cada plano, llevando este procedimiento una complejidad de $O(NV_{3D}.NA_{3D})$. En el siguiente procedimiento, *EliminarPlanosDuplicados*, tenemos que se realiza una comparación entre todos los planos generados por el procedimiento anterior eliminando los duplicados, esto nos lleva a un coste de $O(N_{planos}^2)$, siendo N_{planos} el número de planos generados. Después tenemos el procedimiento de *AsignarAristasAPlanos* que se trata de asignarle a cada

plano todas las aristas que contenga, así tenemos un coste de $O(NA_{3D}.N_{planos})$. Y por último, tenemos el procedimiento de verificación y corrección de los planos que se llama *VerificarPlanos* tenemos que la cota inferior viene dada por $\Omega(NA_{3D}.N_{planos})$ y el peor caso tenemos que se haría NA_{3D} veces el procedimiento *PEVR*, por lo tanto, $O(NV_{3D} \cdot NA_{3D}^2)$. Por lo tanto, este último procedimiento es el que lleva el peso de la mayoría del tiempo de ejecución del procedimiento *GenerarPlanos*.
Ahora analizamos el procedimiento:

Función AnalisisLineasDiscontinuas(listas de vértices y aristas 2D y 3D, planos)

Inicio

```
Para i=1 hasta NA3D  hacer
   Si VerificarAlzado(i) entonces EliminarArista(i);
   Sino Si VerificarPlanta(i) entonces EliminarArista(i);
         Sino Si VerificarPerfil(i) entonces EliminarArista(i);
               FinSi
         FinSi
   FinSi
FinPara
Si Elimina alguna arista entonces PEVR();
FinSi
```

Fin.

Tenemos que las funciones del tipo *Verificar____* lo que hacen es un barrido de todos los planos buscando un plano proyectado por delante de la arista proyectada, por lo tanto estas funciones tienen un coste de $O(N_{planos})$. El peor caso sería recorrer las tres funciones todas las veces y eliminar alguna arista, dando un coste temporal de $O(3.N_{planos}.NA_{3D}+ NV_{3D} . NA_{3D})$.

El siguiente procedimiento será:

Función GenerarCaras(listas de vértices y aristas 3D, planos)

Inicio

```
InicializarEstructuras();
Para i=1 hasta Nplanos hacer
   OrdenarAristasAdyacentes();
   EncontrarBuclesBasicos();
   IdentificarRelacion();
   FormarBucles();
FinPara
```

Fin.

La función *InicializarEstructuras* tiene un coste temporal despreciable frente al resto del algoritmo, por lo tanto, no hace falta analizarlo. La función *OrdenarAristasAdyacentes* tiene un coste de $O(NV_{3D}.$[Coste de ordenación burbuja de talla máxima 20]) debido a que para cada vértice aplica el algoritmo de ordenación "*burbuja*" para un número máximo de *MaxArixV* (20) aristas. Ahora tenemos la función *EncontrarBuclesBasicos*, que tendrá una cota superior de complejidad $O(2 \cdot NA_{3D}^2)$. Para

el siguiente procedimiento, *IdentificarRelación*, tenemos que la cota inferior de la complejidad viene definida cuando no tenemos ninguna cara cóncava y su coste será $\Omega(4 \cdot N_{caras}^2 \cdot MaxVerxPl^2)$, siendo *MaxVerxPl* (30) una constante en el algoritmo que limita en número máximo de vértices que podemos tener por plano y N_{caras} el número de bucles básicos formados en la función anterior. Por último, la función *FormarBucles* no representa ninguna complicación por lo tanto tendrá un coste bajo de $O(N_{caras}.MaxArixPl)$.
El siguiente procedimiento será:

Función ObtenerVerticesyAristasCorte(listas de vértices y aristas 3D, bucles de caras)

Inicio

```
Para i=1 hasta N_planos hacer
   Para j=i+1 hasta N_planos hacer
      Si CortanPlanos(i,j) entonces
         Para k=1 hasta N_caras hacer
            Para t=1 hasta N_caras hacer
               Si NoHayAristaEnInterseccion(k,t) entonces GestionaCorte(k,t);
               FinSi
            FinPara
         FinPara
      FinSi
   FinPara
FinPara
Fin.
```

La función *CortanPlanos* comprueba si se cortan dos planos, esto es equivalente a una serie de operaciones en coma flotante. El mejor caso sería que no se cortase ningún plano, que es imposible, y el peor caso sería que se cortasen todos los planos y no hubiera aristas en la intersección de los planos, que viene determinado por la función *NoHayAristaEnInterseccion*, y también es imposible. Por lo tanto, calculamos el coste sin la función *GestionaCorte* y luego calculamos otro coste ejecutando siempre esta última función. Tenemos que un posible límite inferior sería $\Omega(N_{planos}^2 \cdot N_{caras}^2)$ y una posible cota superior sería $O(N_{planos}^2 \cdot N_{caras}^3)$.

El siguiente procedimiento será:

Función GenerarBuclesCuerpos(listas de vértices y aristas 3D, bucles de caras)

Inicio

```
Inicialización();
Mientras NO(Fin) hacer
   InicializaConjuntoS();
   Fin=SeleccionarCara();
   Si NO(Fin) entonces
      FormarBucleCuerpo();
      Si VerificarBucle() y EsInterior() entonces GuardaBucle();
      FinSi
   FinSi
FinMientras
```

Fin.

Aquí tenemos al procedimiento más costoso, en cuanto al tiempo se refiere, de los algoritmos que hemos visto hasta ahora. Esto es debido a su gran cantidad de cómputo o uso de operaciones en coma flotante que realiza, y además por su estructura genera una complejidad temporal de $O(N_{caras}^4)$.

Y por último tenemos el procedimiento:

Función CombinacionYVerificacion(listas v/a 2D, listas v/a 3D, bucles de cuerpos)

Inicio

Para i=1 **hasta** 2^N **hacer**
GeneraCandidato();
Si VerificaCandidato() **y** ProyeccionCorrecta() **entonces** GeneraObjeto();
FinSi
FinPara

Fin.

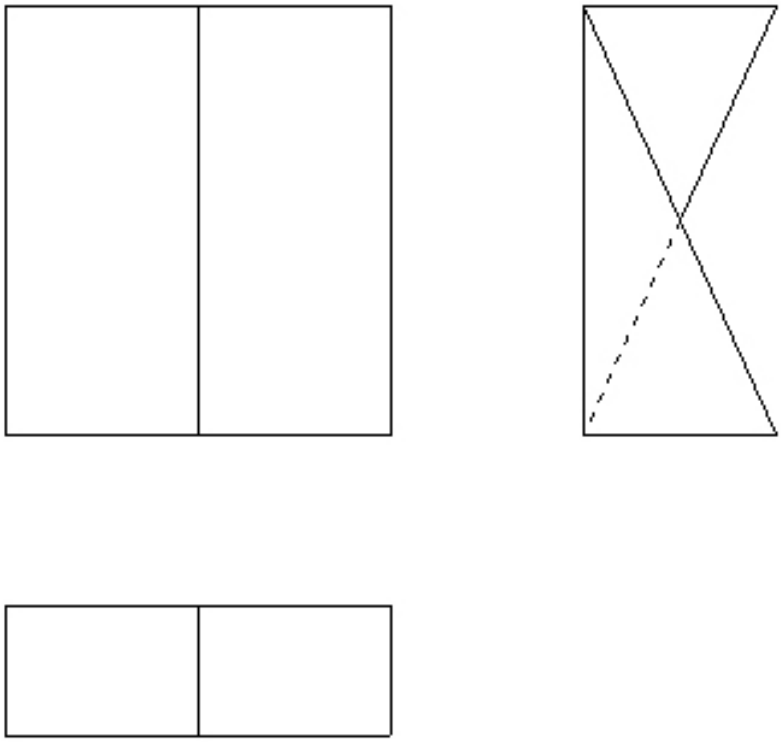
Este último algoritmo es el que tiene una mayor complejidad temporal de todos los algoritmos vistos hasta ahora, esto es debido a que antes teníamos algoritmos con costes polinómicos, y este último tiene coste exponencial, concretamente $O(2^N)$ donde N es el número de bucles de cuerpos o cuerpos elementales que se han generado. Por lo tanto, cuando tengamos muchos bucles de cuerpos este procedimiento se va a llevar del orden del 90% del porcentaje de ejecución del algoritmo de reconstrucción.

Capítulo 4
EJEMPLOS DE RECONSTRUCCIÓN

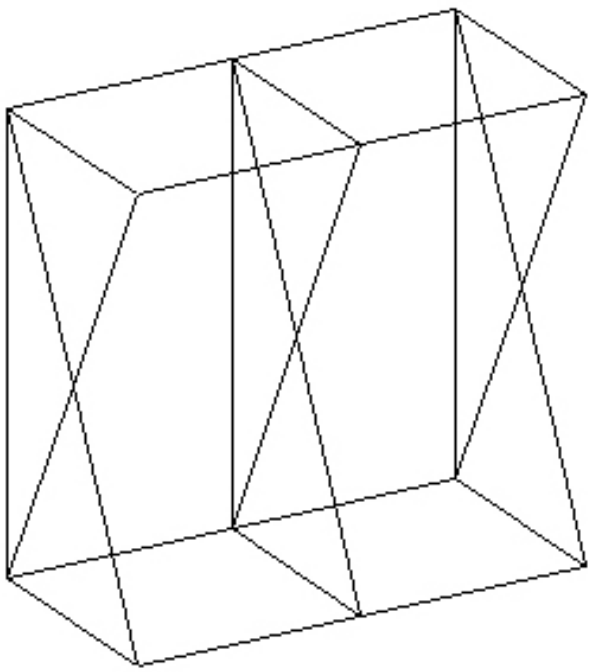
En este capítulo se incluyen una serie de ejemplos demostrativos de la ejecución de la aplicación Rec3D, creada como implementación del algoritmo de reconstrucción 3D de Qing-Wen Yan. Dentro de los ejemplos podemos distinguir desde los más simples hasta los diseños más complejos que plantean mayor problemática.

Dentro de cada ejemplo se van a especificar todos los datos posibles que se vayan generando en los distintos pasos del algoritmo de reconstrucción implementado. Inicialmente se mostrará gráficamente los datos de entrada del algoritmo, que serán las proyecciones de planta, alzado y perfil del objeto que queremos obtener. Luego se mostrará el modelo alámbrico construido mediante el procedimiento *WC* del algoritmo de reconstrucción. Después veremos los planos generados mediante el procedimiento *PGG* y luego veremos si el procedimiento *BLR* modifica el modelo alámbrico, dando la justificación necesaria. Visto esto, pasaremos a ver los bucles de caras que genera el algoritmo *FLG* y después pasaremos a ver como se modifica el modelo alámbrico cuando se obtienen los vértices y aristas de corte en el procedimiento *CEV*. Finalmente obtenemos los cuerpos elementales mediante el procedimiento *BLR* y mediante la combinación de los cuerpos elementales obtendremos los objetos válidos.

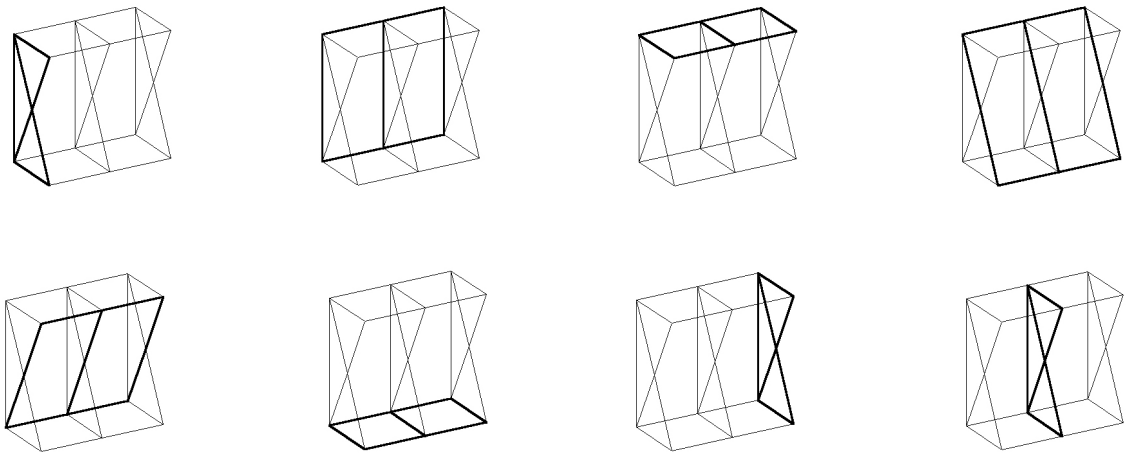
4.1. EJEMPLO 1.



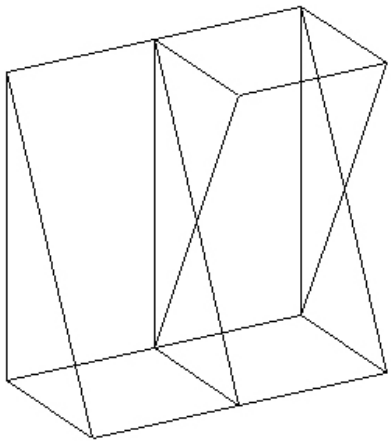
Con la representación anterior de planta, alzado y perfil, tenemos que el procedimiento de construcción del modelo alámbrico genera los siguientes vértices y aristas 3D:

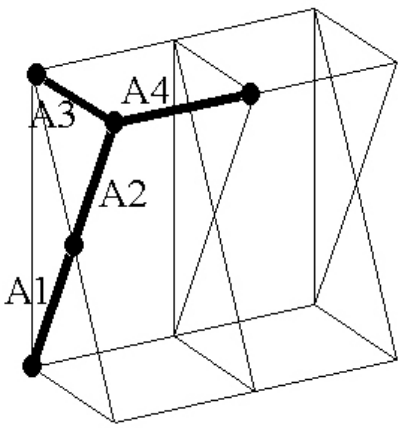


Del anterior modelo alámbrico se obtienen los siguientes gráficos planos mediante el procedimiento *PGG*:



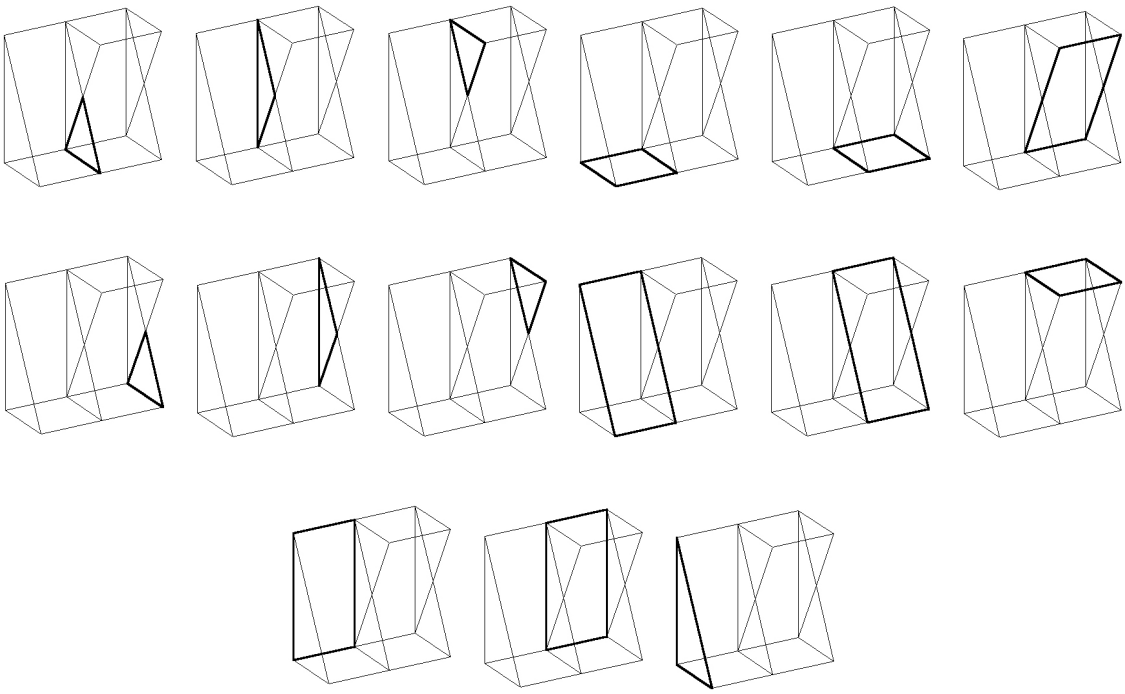
Ahora se aplica el procedimiento de análisis de las líneas discontinuas de los datos de entrada, *BLR*, obteniendo el siguiente modelo alámbrico.





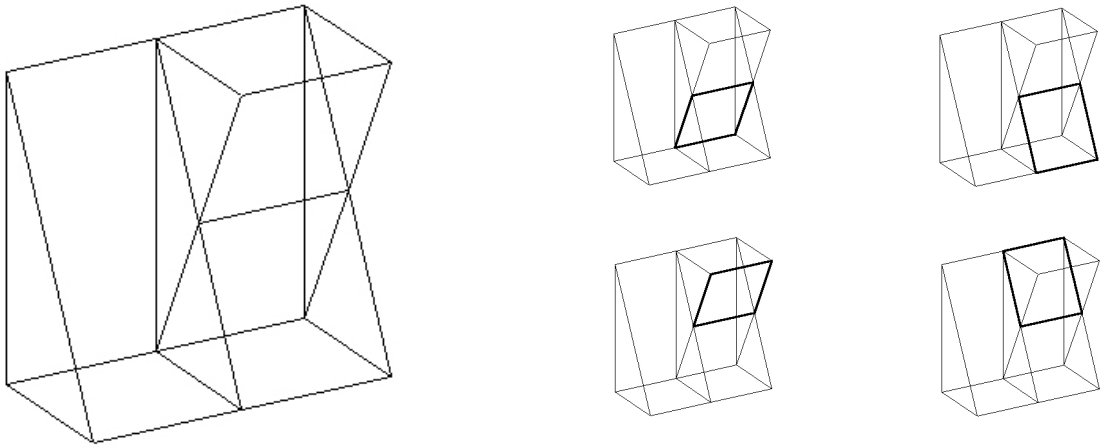
En la figura de la izquierda podemos ver las aristas que ha eliminado el procedimiento *BLR*. Primero ha eliminado la arista A1 porque en la dirección de +X no hemos encontrado ningún gráfico plano por delante de la arista, y además, su proyección asociada del perfil es una línea discontinua, por lo tanto, si A1 existiera, en la proyección de perfil sería una arista continua. Como hay una contradicción con los datos de entrada la arista A1 tiene que ser eliminada. Al eliminar la arista A1, tenemos que aplicar el procedimiento *PEVR* que elimina A2, A3 y A4.

En este momento se aplica el procedimiento de generación de bucles de caras *FLG*, obteniendo las siguientes:

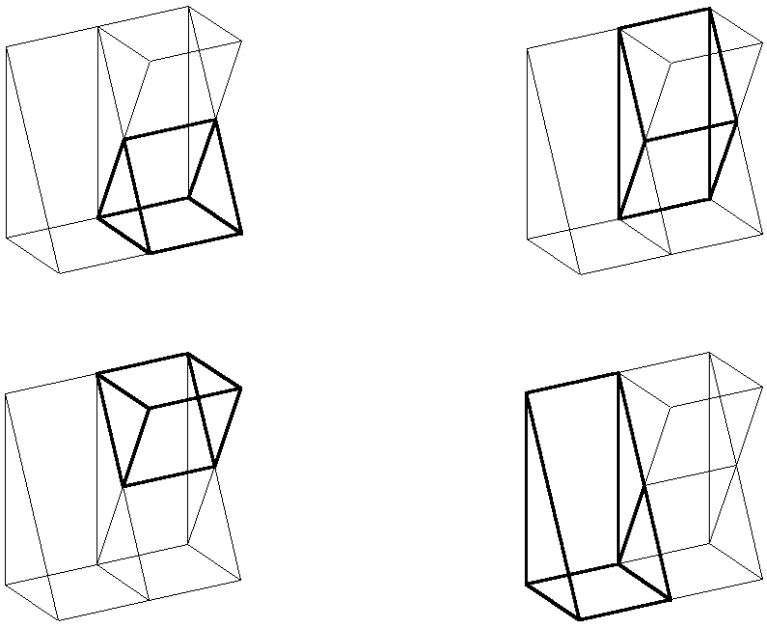


Entonces, ahora, aplicamos el procedimiento que se encargará de encontrar las aristas y vértices de corte, *CEV*. En este caso, sólo tenemos un arista de corte que se genera mediante el corte de los dos bucles de caras siguientes:

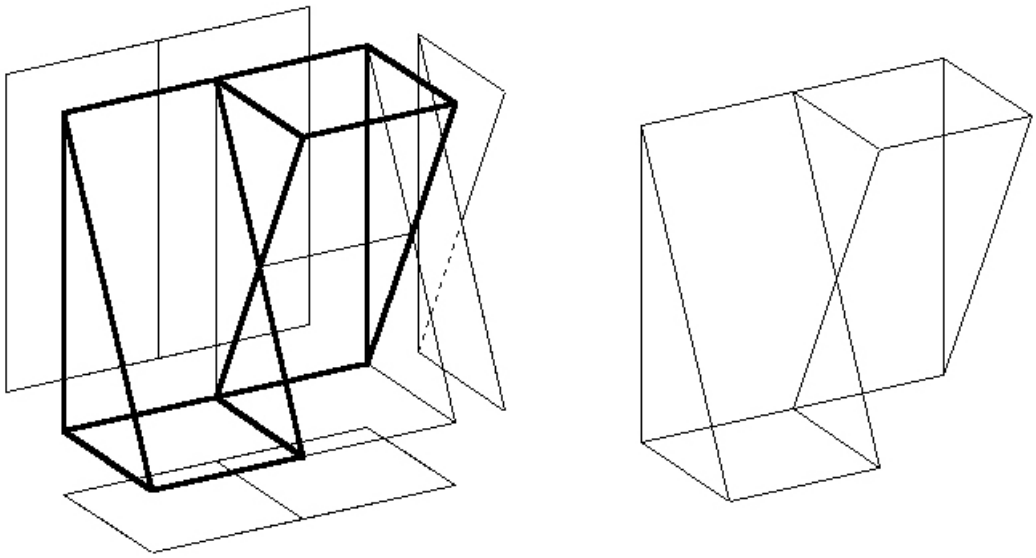
El corte de estos dos bucles de caras se traduce en la figura siguiente mediante una arista de corte que nos deja el siguiente modelo alámbrico y, además provoca que los dos bucles de caras se dividan como se indica a continuación:



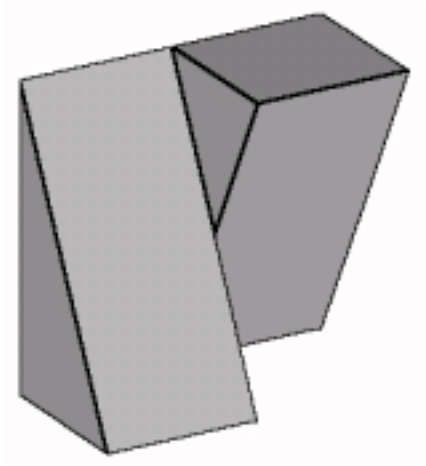
Con todos los bucles de caras generados anteriormente, se generan los bucles de cuerpos que podemos ver en la figura siguiente mediante el procedimiento *BLG*.



Por lo tanto, ahora solo queda realizar las combinaciones de todos los bucles de cuerpos para, mediante sus proyecciones, obtener todos los objetos válidos posibles, que en este caso sólo es el siguiente.



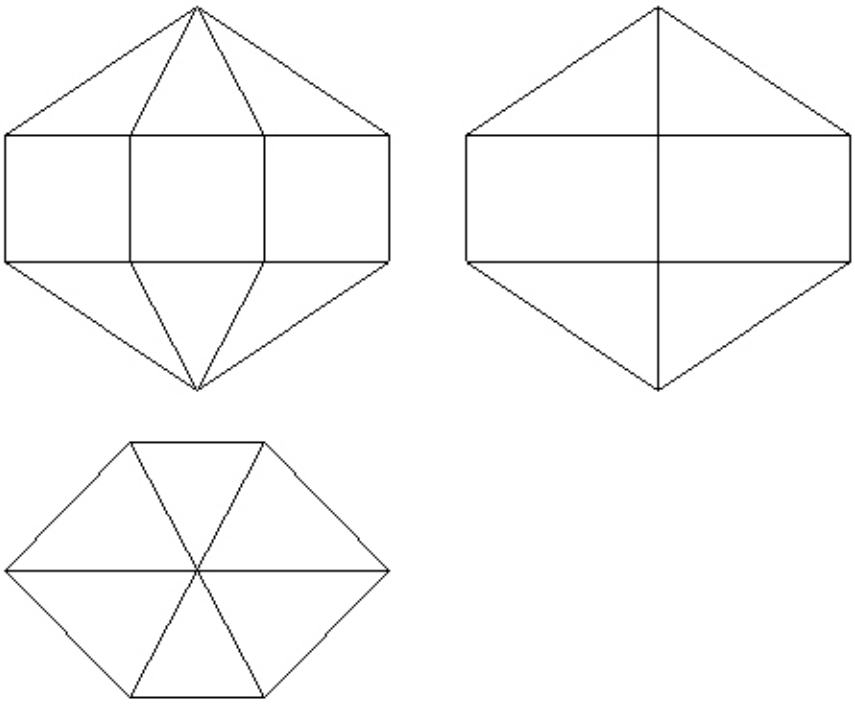
Aplicando sombreado al objeto válido, podemos ver en la figura siguiente la única solución correcta u objeto válido 3D para las proyecciones de entrada.



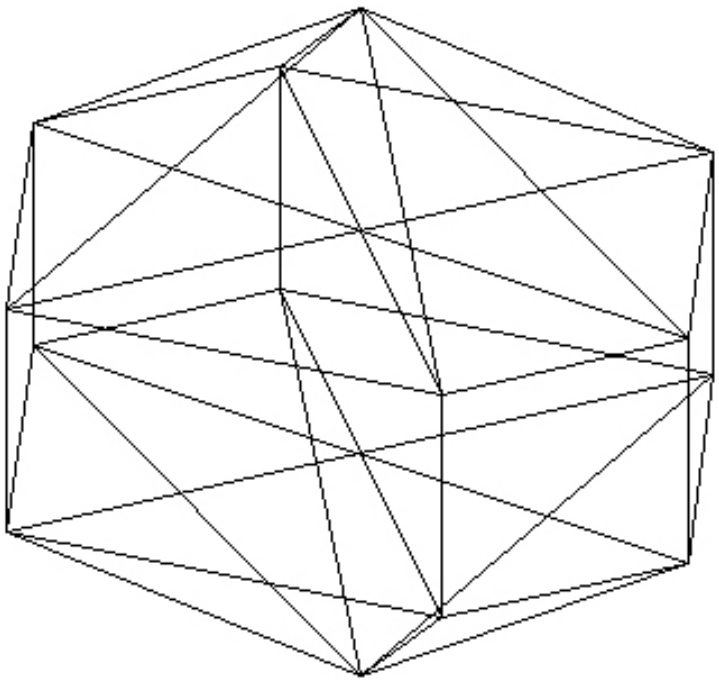
Para finalizar, mostramos los datos y estadísticas de la reconstrucción tal y como nos la presenta la aplicación Rec3D en un fichero de texto.

```
INFORME DE RECONSTRUCCION 3D
____________________________

Nombre del dibujo:  Dibujo 1 de la aplicacion REc3D

Datos de entrada:
  - Planta: 9 vertice(s), 5 arista(s).
  - Alzado: 9 vertice(s), 5 arista(s).
  - Perfil: 5 vertice(s), 6 arista(s).

Aristas despues del preprocesado de datos:
  - Planta: 15 arista(s).
  - Alzado: 15 arista(s).
  - Perfil: 9 arista(s).

Datos de la reconstruccion 3D:
  - Vertices 3D: 13.
  - Aristas 3D: 25.
  - Graficos Planos: 8.
  - Bucles de caras: 17.
  - Bucles de cuerpos: 4.
  - Total de Objetos candidatos: 15.
  - Objetos Validos: 1.

Estadisticas de la reconstruccion 3D:
  - Preproceso: t=0 ms, 0 %.
  - WC: t=20 ms, 6 %.
  - PGG: t=25 ms, 8 %.
  - BLR: t=10 ms, 3 %.
  - FLG: t=25 ms, 8 %.
  - CEV: t=10 ms, 3 %.
  - BLG: t=95 ms, 31 %.
  - Verificacion: t=125 ms, 40 %.
  - Total: t=310 ms, 100 %.

          PFC'98, Lorenzo J. Muñiz Marquez.
```

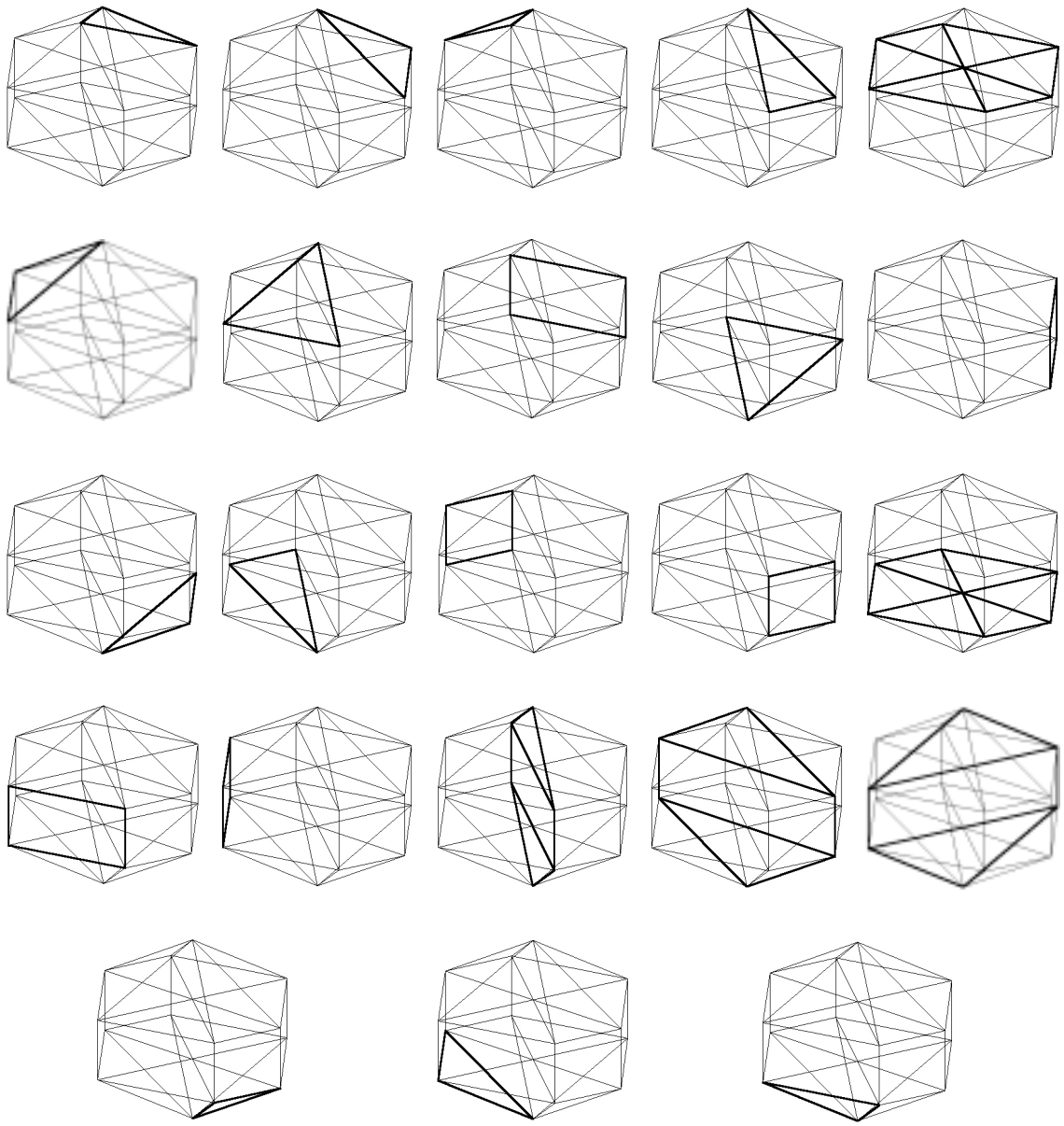
4.2. EJEMPLO 2.



Con la representación anterior de planta, alzado y perfil, tenemos que el procedimiento de construcción del modelo alámbrico genera los siguientes vértices y aristas 3D:

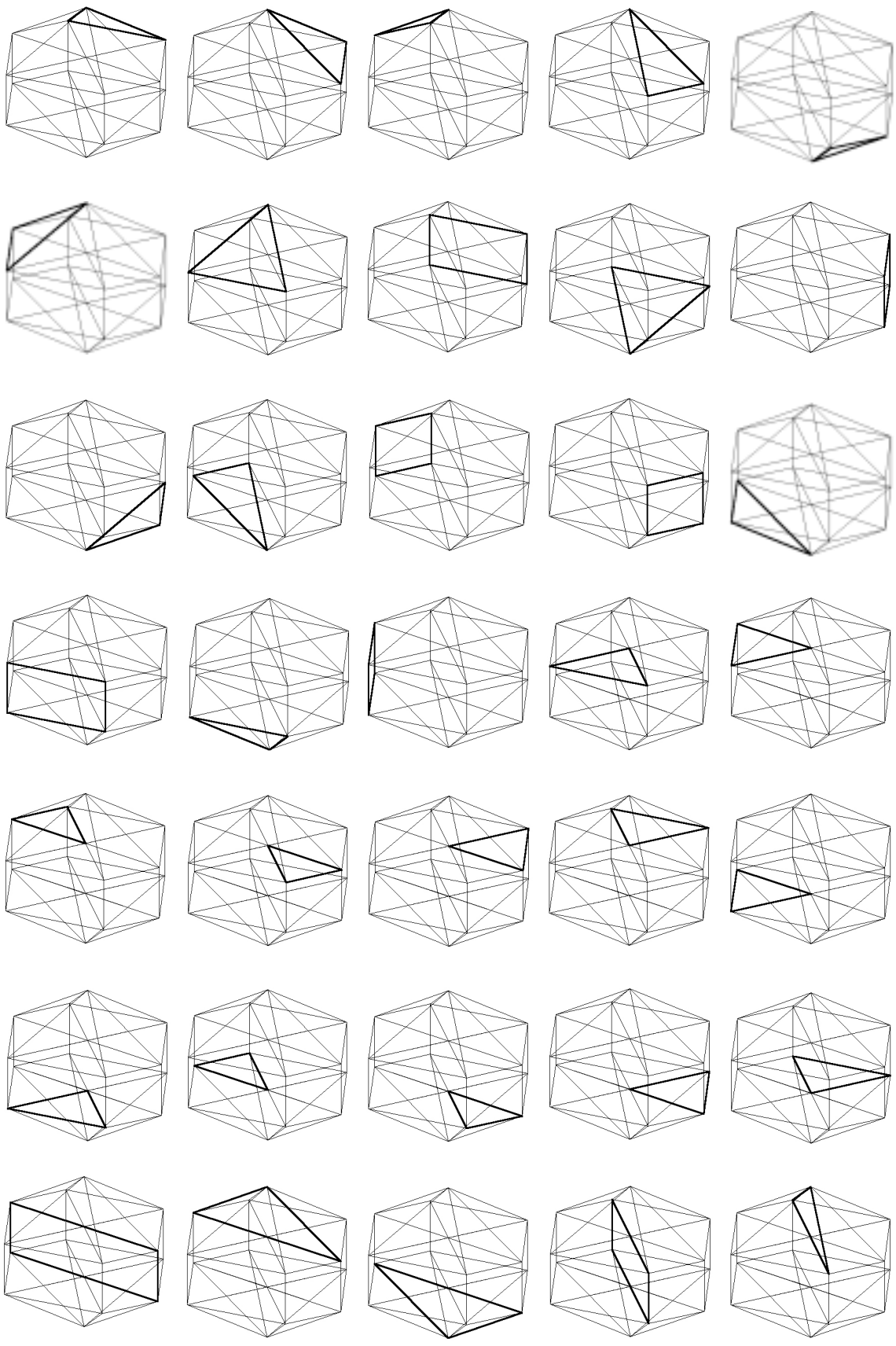


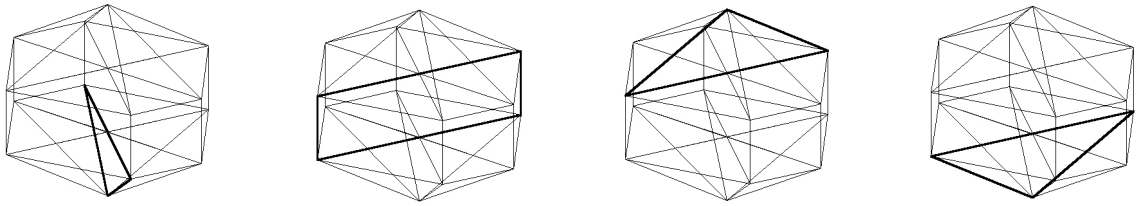
Del anterior modelo alámbrico se obtienen los siguientes gráficos planos mediante el procedimiento *PGG*:



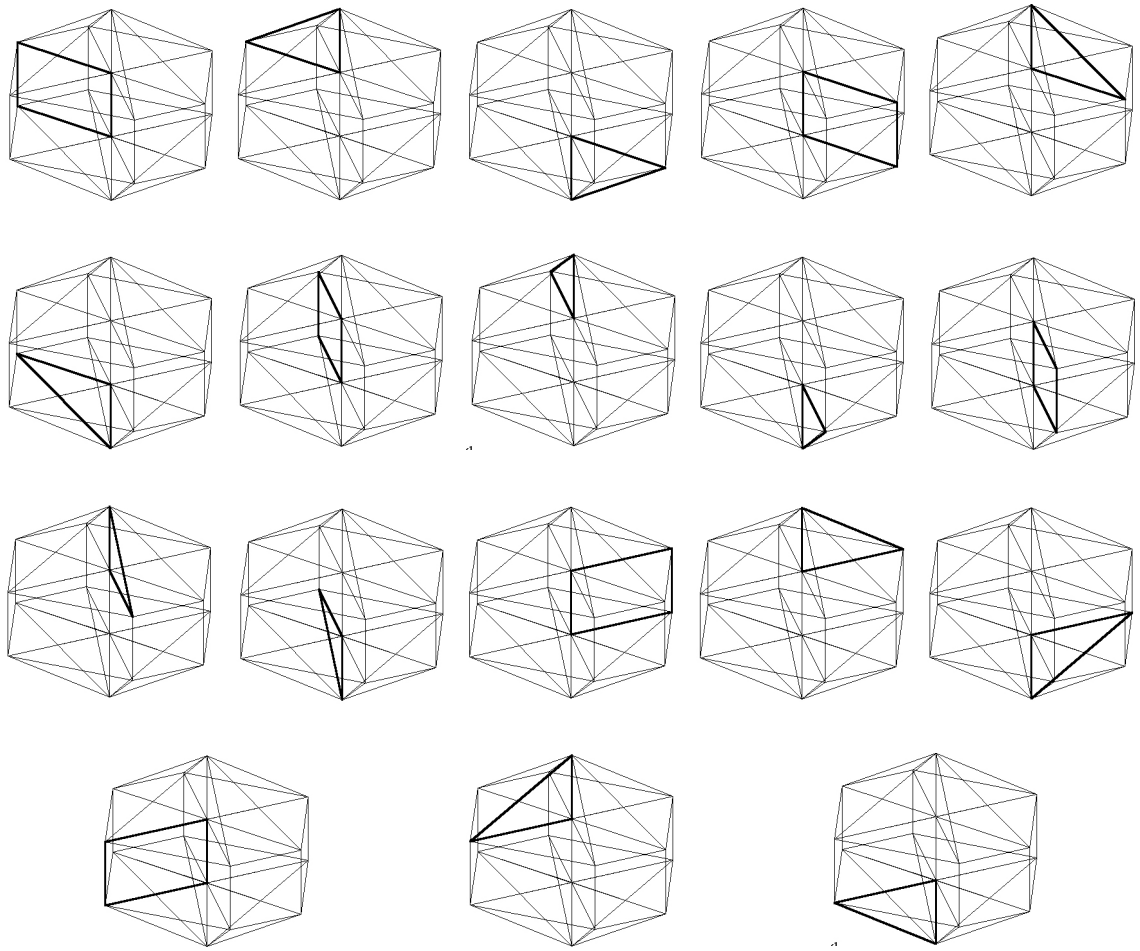
Ahora se aplica el procedimiento de análisis de las líneas discontinuas de los datos de entrada, *BLR*, obteniendo el mismo modelo alámbrico.

En este momento se aplica el procedimiento de generación de bucles de caras *FLG*, obteniendo las siguientes:

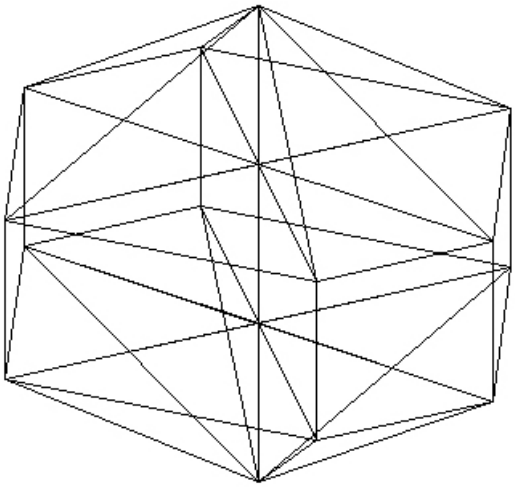




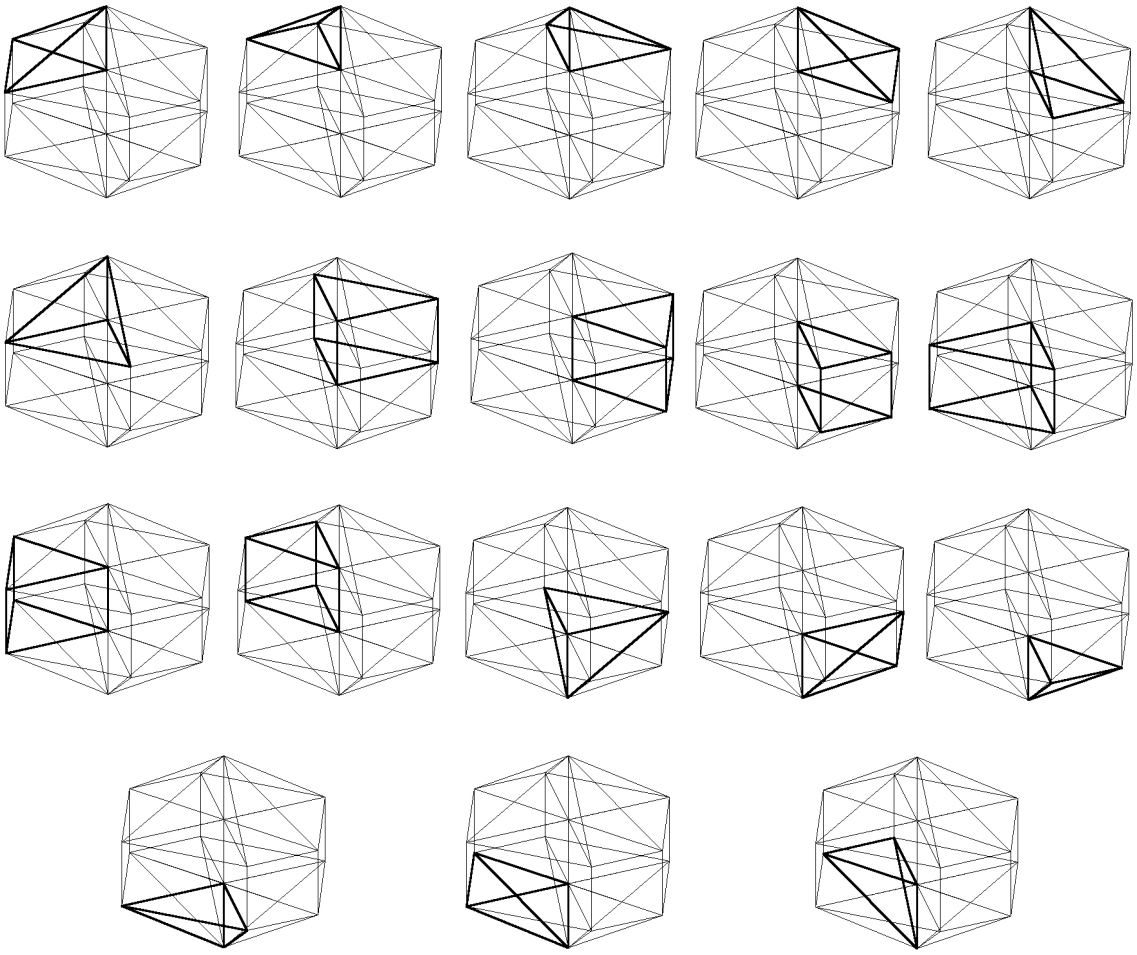
Entonces, ahora, aplicamos el procedimiento que se encargará de encontrar las aristas y vértices de corte, *CEV*. En este caso, tenemos tres aristas de corte. El corte de los bucles de caras con las aristas de corte genera los siguientes bucles de caras:



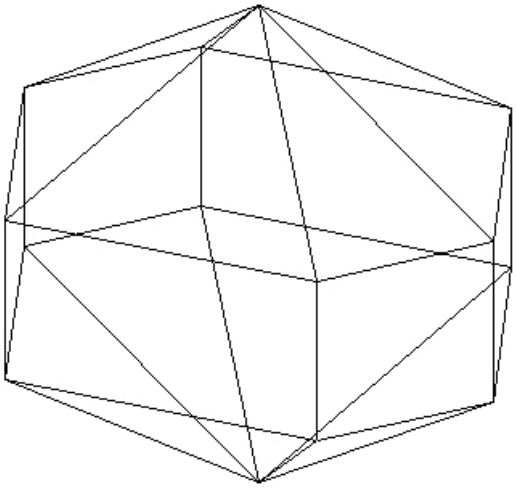
En la figura siguiente podemos ver como queda el modelo alámbrico después de aplicar el procedimiento *CEV*.



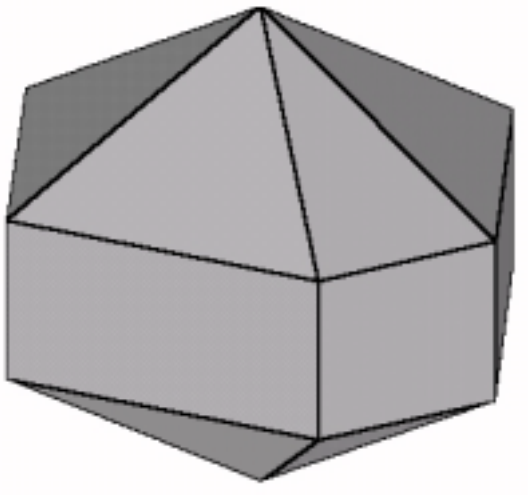
Con todos los bucles de caras generados anteriormente, se generan los bucles de cuerpos que podemos ver en las figuras siguientes mediante el procedimiento *BLG*.



Por lo tanto, ahora solo queda realizar las combinaciones de todos los bucles de cuerpos para, mediante sus proyecciones, obtener todos los objetos válidos posibles, que en este caso sólo es el siguiente.



Aplicando sombreado al objeto válido, podemos ver en la figura siguiente la única solución correcta u objeto válido 3D para las proyecciones de entrada.

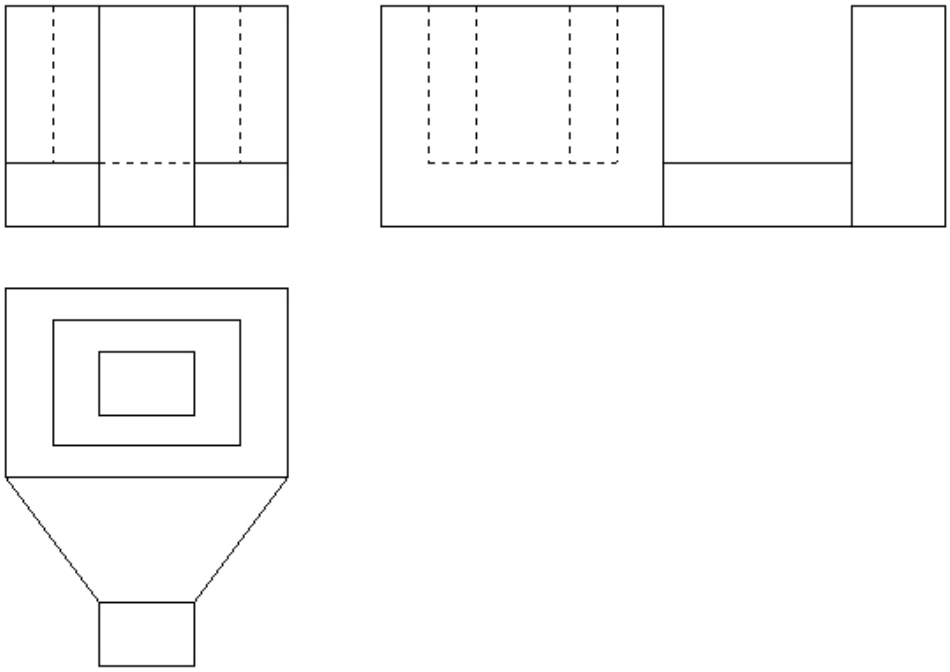


Para finalizar, mostramos los datos y estadísticas de la reconstrucción tal y como nos la presenta la aplicación Rec3D en un fichero de texto.

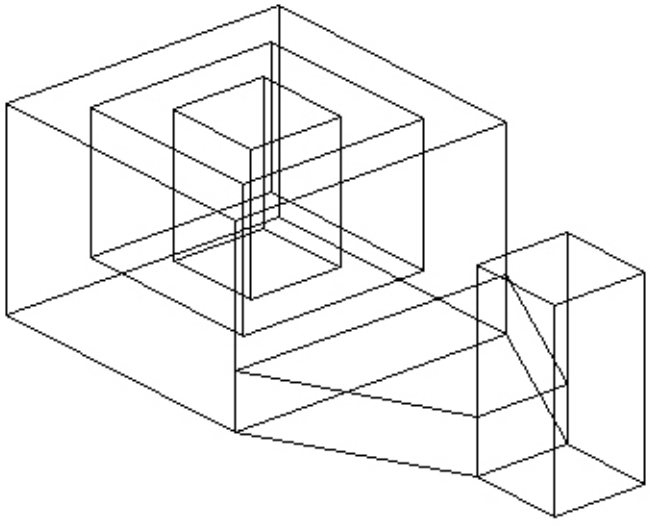
```
INFORME DE RECONSTRUCCION 3D
____________________________

Nombre del dibujo:  C:\PROYECTO\PROTOWIN\DIAMANTE.GRA

Datos de entrada:
  - Planta: 7 vertice(s), 9 arista(s).
  - Alzado: 10 vertice(s), 14 arista(s).
  - Perfil: 8 vertice(s), 9 arista(s).

Aristas despues del preprocesado de datos:
  - Planta: 15 arista(s).
  - Alzado: 24 arista(s).
  - Perfil: 18 arista(s).

Datos de la reconstruccion 3D:
  - Vertices 3D: 16.
  - Aristas 3D: 52.
  - Graficos Planos: 23.
  - Bucles de caras: 48.
  - Bucles de cuerpos: 18.
  - Total de Objetos candidatos: 262143.
  - Objetos Validos: 1.

Estadisticas de la reconstruccion 3D:
  - Preproceso: t=0 ms, 0 %.
  - WC: t=70 ms, 0 %.
  - PGG: t=45 ms, 0 %.
  - BLR: t=5 ms, 0 %.
  - FLG: t=35 ms, 0 %.
  - CEV: t=115 ms, 0 %.
  - BLG: t=695 ms, 0 %.
  - Verificacion: t=6135017 ms, 100 %.
  - Total: t=6135982 ms, 100 %.

          PFC'98, Lorenzo J. Muñiz Marquez.
```

Como se puede ver, el coste de este ejemplo es excesivamente elevado. Esto es debido a la gran cantidad de bucles de cuerpos que se obtienen, que provocan un número de objetos candidatos de 262143, con un tiempo de ejecución del procedimiento de verificación de los objetos candidatos de 6135017 ms = 102 minutos aproximadamente. Estos tiempos de ejecución se han medido sobre un procesador Pentium MMX 233 MHz. Por lo tanto, podemos deducir que el tiempo de este algoritmo recae principalmente sobre el procedimiento de verificación, por lo tanto, para mejorar su rendimiento habría que realizar una optimización de este procedimiento, ya que tiene un coste temporal exponencial.

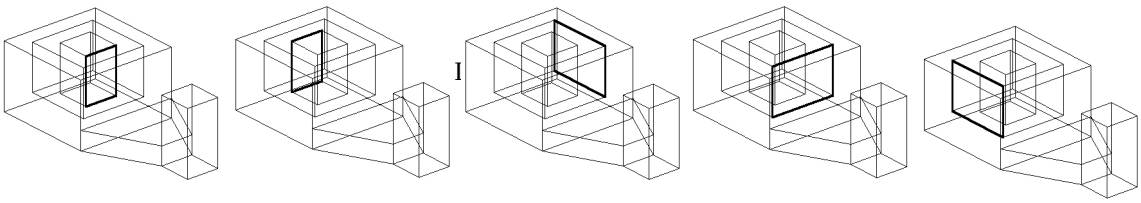
4.3. EJEMPLO 3.



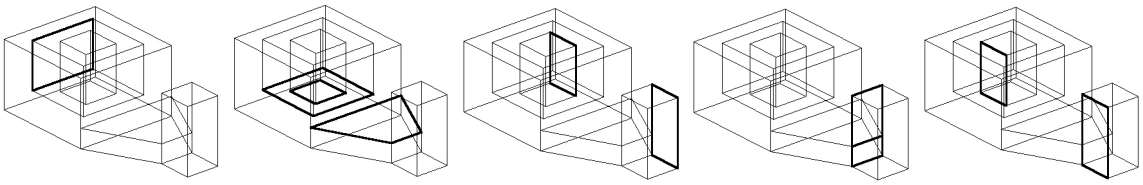
Con la representación anterior de planta, alzado y perfil, tenemos que el procedimiento de construcción del modelo alámbrico genera los siguientes vértices y aristas 3D:



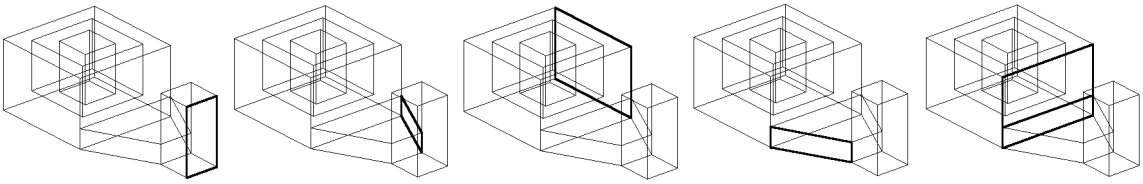
Del anterior modelo alámbrico se obtienen los siguientes gráficos planos mediante el procedimiento *PGG*:



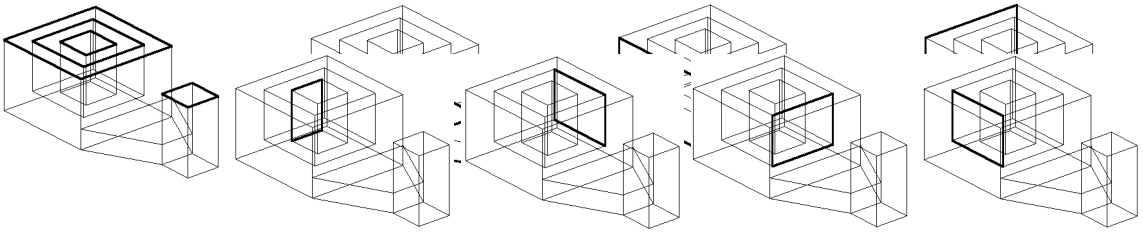
Ahora se aplica el procedimiento de análisis de las líneas discontinuas de los datos de entrada, *BLR*. Este procedimiento no modifica de forma alguna el modelo alámbrico anteriormente generado.

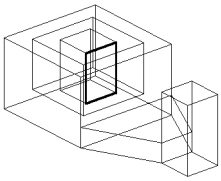


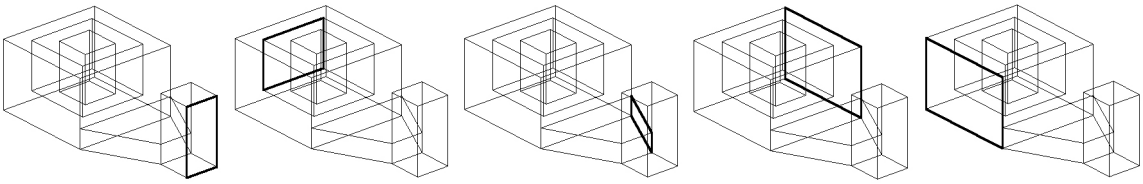
En este momento se aplica el procedimiento de generación de bucles de caras

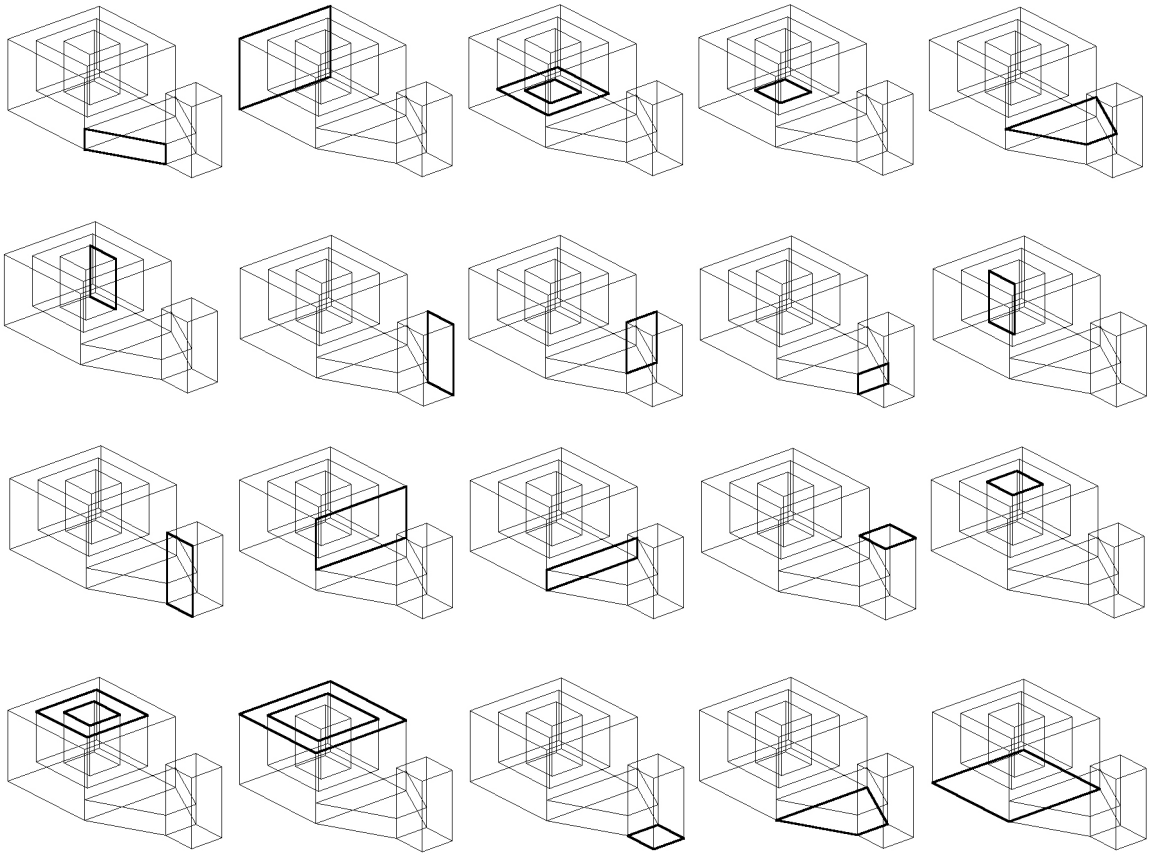


FLG, obteniendo las siguientes:



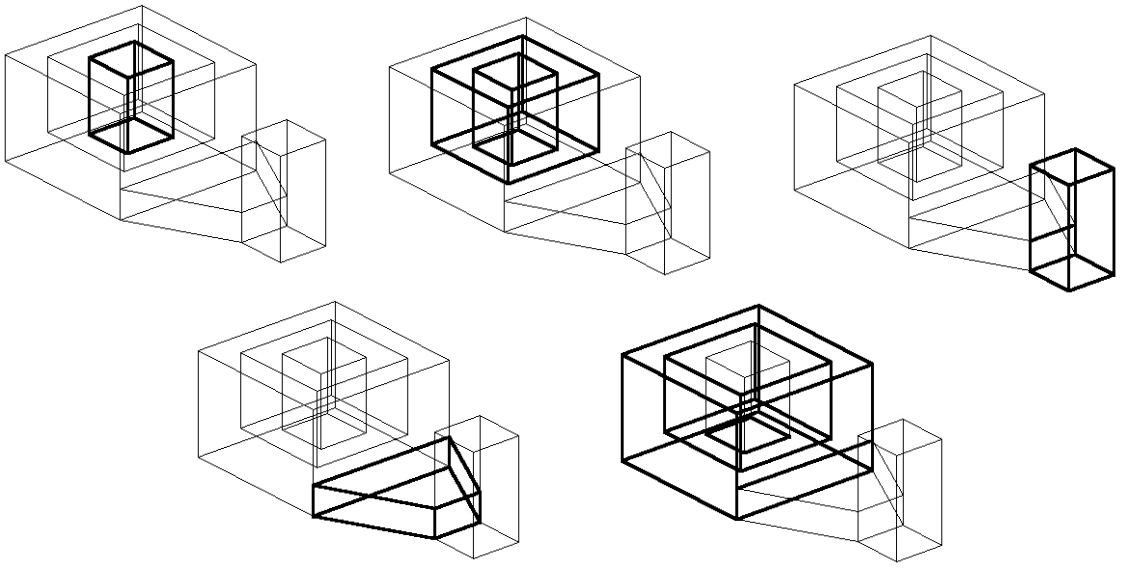




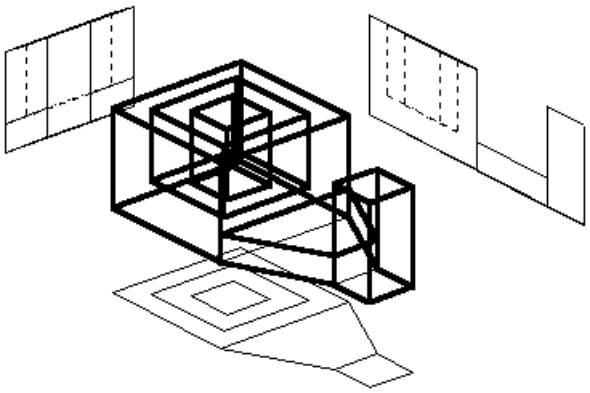


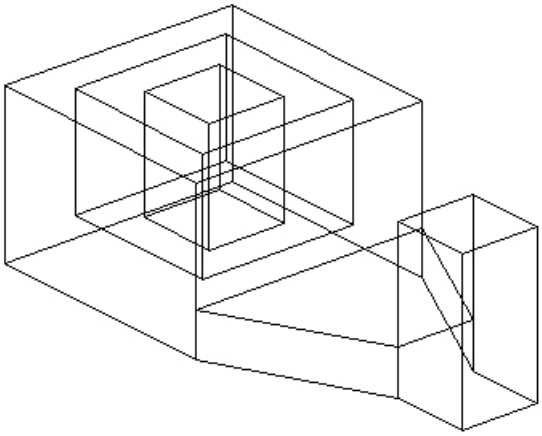
Entonces, ahora, aplicamos el procedimiento que se encargará de encontrar las aristas y vértices de corte, *CEV*. En este caso, no tenemos ninguna arista de corte que se genere mediante el corte de los dos bucles de caras.

Con todos los bucles de caras generados anteriormente, se generan los bucles de cuerpos que podemos ver en la figura siguiente mediante el procedimiento *BLG*.

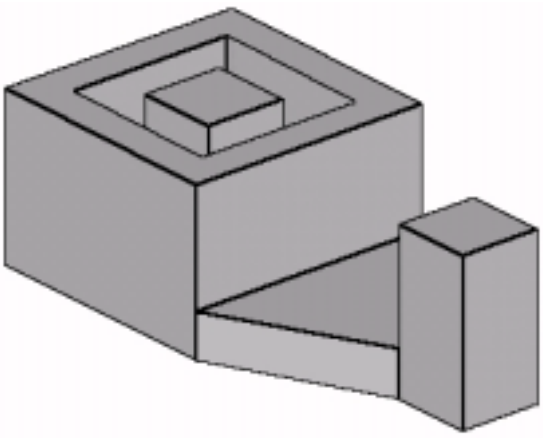


Por lo tanto, ahora solo queda realizar las combinaciones de todos los bucles de cuerpos para, mediante sus proyecciones, obtener todos los objetos válidos posibles, que en este caso sólo es el siguiente.





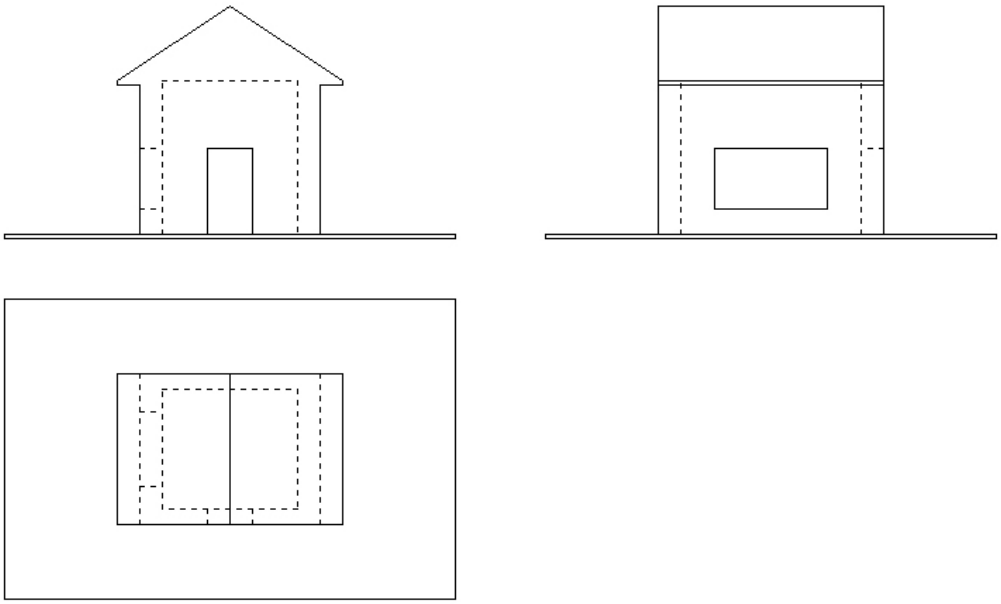
Aplicando sombreado al objeto válido, podemos ver en la figura siguiente la única solución correcta u objeto válido 3D para las proyecciones de entrada.



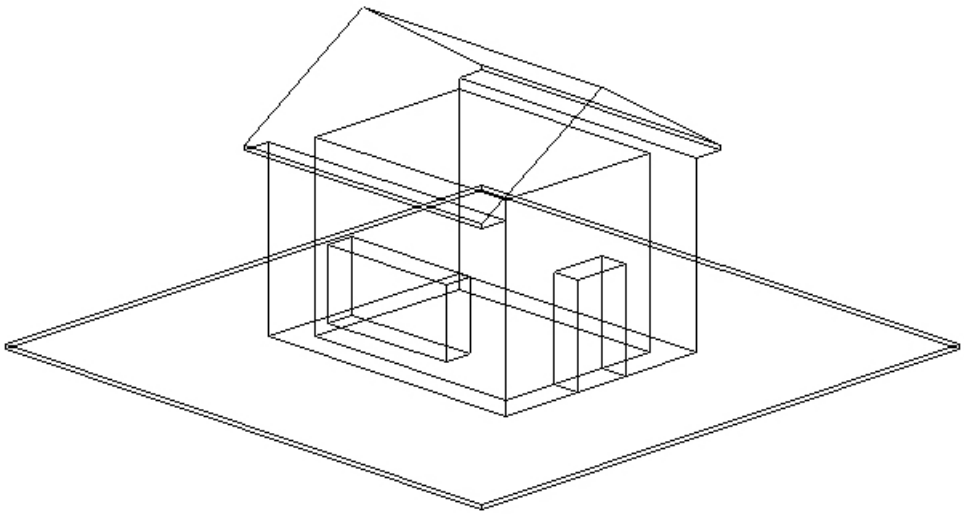
Para finalizar, mostramos los datos y estadísticas de la reconstrucción tal y como nos la presenta la aplicación Rec3D en un fichero de texto.

```
INFORME DE RECONSTRUCCION 3D
____________________________

Nombre del dibujo:  Dibujo 10 de la aplicacion REc3D

Datos de entrada:
  - Planta: 16 vertice(s), 18 arista(s).
  - Alzado: 16 vertice(s), 11 arista(s).
  - Perfil: 18 vertice(s), 13 arista(s).

Aristas despues del preprocesado de datos:
  - Planta: 18 arista(s).
  - Alzado: 50 arista(s).
  - Perfil: 41 arista(s).

Datos de la reconstruccion 3D:
  - Vertices 3D: 36.
  - Aristas 3D: 58.
  - Graficos Planos: 19.
  - Bucles de caras: 30.
  - Bucles de cuerpos: 5.
  - Total de Objetos candidatos: 31.
  - Objetos Validos: 1.

Estadisticas de la reconstruccion 3D:
  - Preproceso: t=10 ms, 1 %.
  - WC: t=100 ms, 6 %.
  - PGG: t=50 ms, 3 %.
  - BLR: t=10 ms, 1 %.
  - FLG: t=35 ms, 2 %.
  - CEV: t=30 ms, 2 %.
  - BLG: t=360 ms, 23 %.
  - Verificacion: t=1000 ms, 63 %.
  - Total: t=1595 ms, 100 %.

          PFC'98, Lorenzo J. Muñiz Marquez.
```

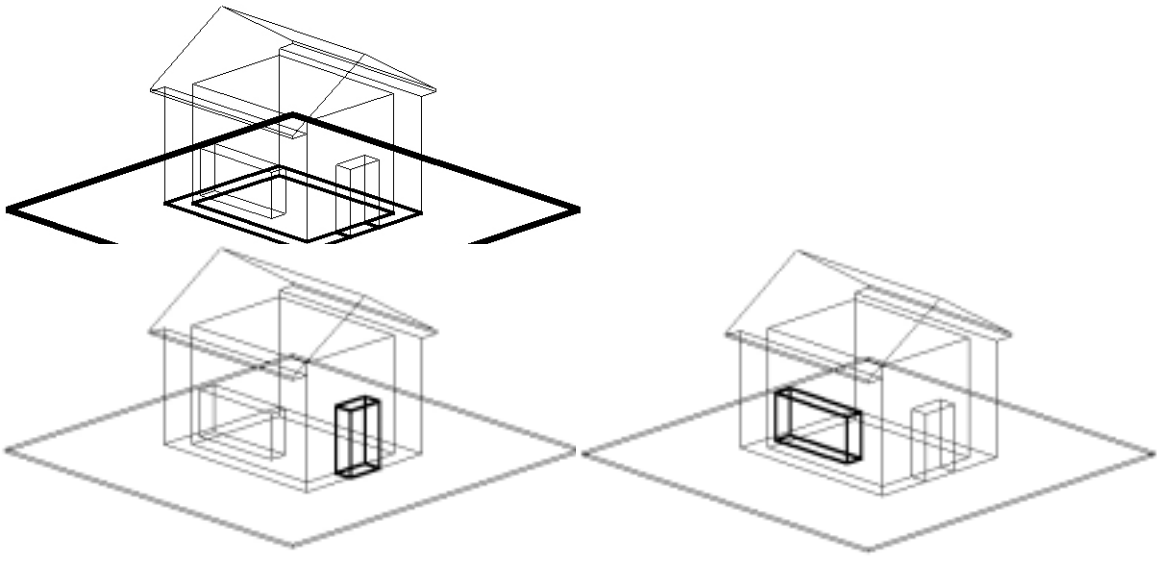
4.4. EJEMPLO 4.



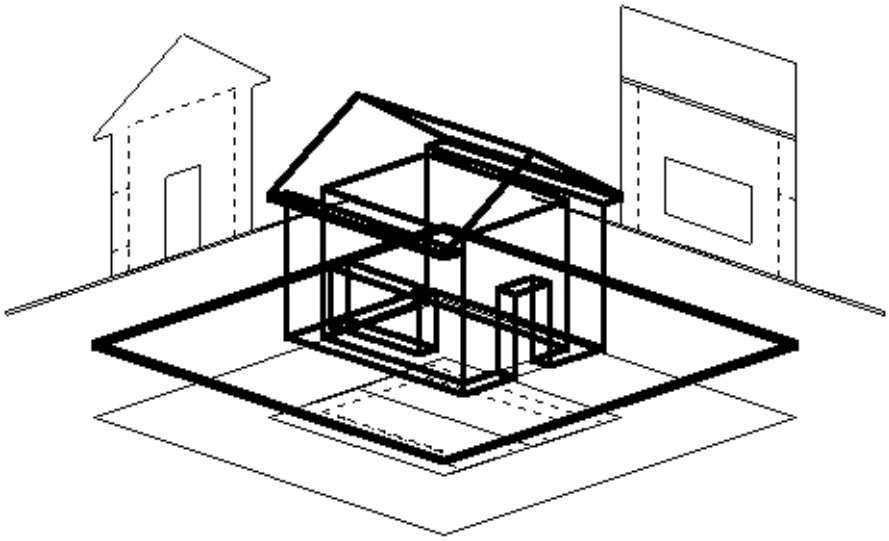
Con la representación anterior de planta, alzado y perfil, tenemos que el procedimiento de construcción del modelo alámbrico genera los siguientes vértices y aristas 3D:

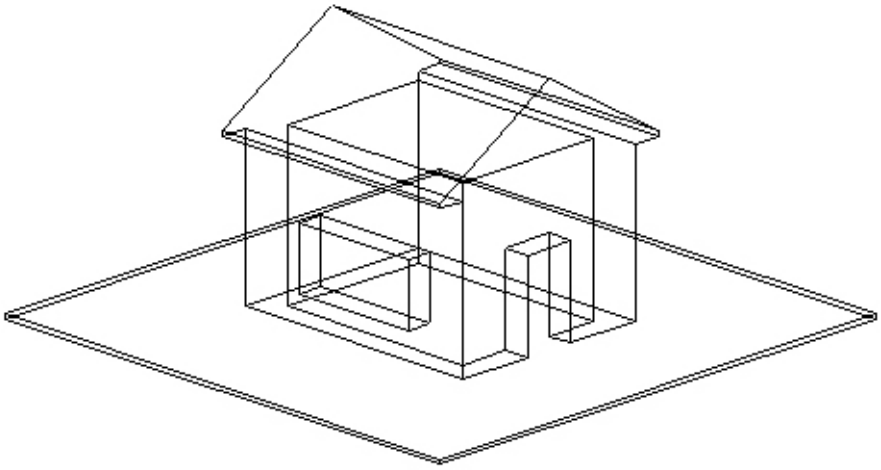


Del anterior modelo alámbrico se obtienen los gráficos planos mediante el procedimiento *PGG*. Después se aplica el procedimiento de análisis de las líneas discontinuas de los datos de entrada, *BLR*. Este procedimiento no modifica de forma alguna el modelo alámbrico anteriormente generado. En este momento se aplica el procedimiento de generación de bucles de caras *FLG*, obteniendo un conjunto de caras que se utilizarán en los siguientes procedimientos. Entonces, ahora, aplicamos el procedimiento que se encargará de encontrar las aristas y vértices de corte, *CEV*. En este caso, no tenemos ninguna arista de corte que se genere mediante el corte de los dos bucles de caras. Con todos los bucles de caras generados anteriormente, se generan los bucles de cuerpos que podemos ver en la figura siguiente mediante el procedimiento *BLG*.

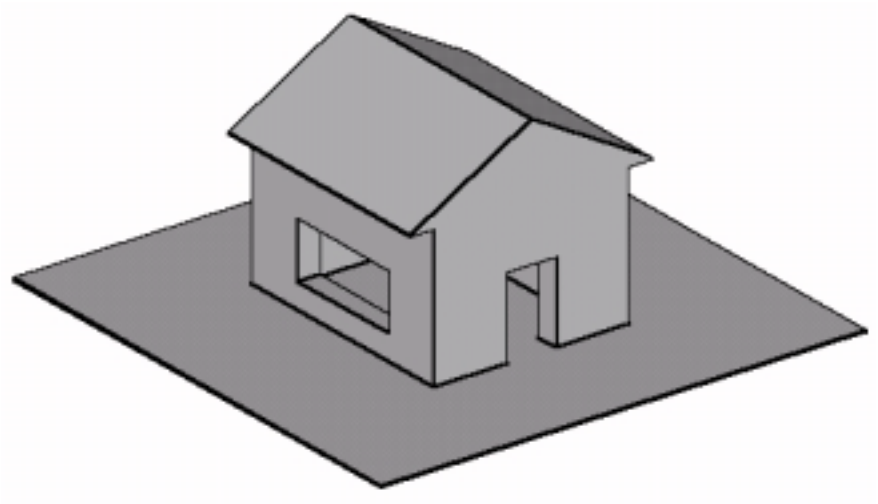


Por lo tanto, ahora solo queda realizar las combinaciones de todos los bucles de cuerpos para, mediante sus proyecciones, obtener todos los objetos válidos posibles, que en este caso sólo es el siguiente.





Aplicando sombreado al objeto válido, podemos ver en la figura siguiente la única solución correcta u objeto válido 3D para las proyecciones de entrada.



Para finalizar, mostramos los datos y estadísticas de la reconstrucción tal y como nos la presenta la aplicación Rec3D en un fichero de texto.

```
INFORME DE RECONSTRUCCION 3D
____________________________

Nombre del dibujo:  Dibujo 13 de la aplicacion REc3D

Datos de entrada:
  - Planta: 26 vertice(s), 19 arista(s).
  - Alzado: 25 vertice(s), 20 arista(s).
  - Perfil: 22 vertice(s), 16 arista(s).

Aristas despues del preprocesado de datos:
  - Planta: 63 arista(s).
  - Alzado: 57 arista(s).
  - Perfil: 51 arista(s).

Datos de la reconstruccion 3D:
  - Vertices 3D: 50.
  - Aristas 3D: 77.
  - Graficos Planos: 26.
  - Bucles de caras: 35.
  - Bucles de cuerpos: 5.
  - Total de Objetos candidatos: 31.
  - Objetos Validos: 1.

Estadisticas de la reconstruccion 3D:
  - Preproceso: t=25 ms, 1 %.
  - WC: t=205 ms, 6 %.
  - PGG: t=70 ms, 2 %.
  - BLR: t=24 ms, 1 %.
  - FLG: t=65 ms, 2 %.
  - CEV: t=65 ms, 2 %.
  - BLG: t=820 ms, 25 %.
  - Verificacion: t=1970 ms, 61 %.
  - Total: t=3244 ms, 100 %.

          PFC'98, Lorenzo J. Muñiz Marquez.
```

Capítulo 5
DESCRIPCIÓN DE LA APLICACIÓN Rec3D

En los capítulos anteriores hemos realizado una descripción de un proceso de reconstrucción 3D a partir de una serie de vistas según Qing-Wen Yan. Con el fin de realizar una implementación en lenguaje de programación C, se ha creado una aplicación que permite la reconstrucción 3D de objetos a partir de sus vistas de planta, alzado y perfil. He llamado a esta aplicación *Rec3D*, y es la que se va a describir a continuación para que no haya dudas sobre su manejo y pasos a seguir para obtener el objeto 3D. Realizaremos una descripción del menú, todas sus opciones y barras de herramientas.

Hay que decir, que esta aplicación surgió en primera instancia como un prototipo para MS-DOS en la que se resolvía el problema de la reconstrucción hasta el procedimiento de generación de bucles de caras, pasando luego a realizar la versión para entornos Windows, por su mejor presentación y calidad. Esta versión de la aplicación se ha generado con la opción del compilador de 16 bits, por lo tanto podemos ejecutar esta aplicación tanto en Windows 3.1, como en Windows 95 o Windows 98.

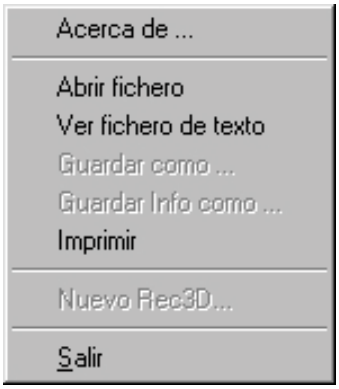
5.1. DESCRIPCIÓN DEL MENÚ.

Como todos los programas estilo Windows tenemos un menú, en la parte superior de la pantalla, de opciones desplegables. Como se puede ver en la figura siguiente, tenemos las siguientes opciones:

Vamos ahora a realizar una descripción de cada una de las opciones del menú anterior, explicando las acciones que realizan o las opciones que modifican.

5.1.1. EL MENÚ "Archivo".

En este menú tenemos una serie de acciones que tienen que ver con el manejo de ficheros o con la propia aplicación. Dentro de esta opción del menú tenemos las siguientes acciones:



- *Acerca de ...* : Muestra información acerca del programa y su autor.
- *Abrir fichero* : Muestra un típico cuadro de diálogo de Windows para cargar un fichero en memoria con los datos de las vistas. El formato de este fichero de entrada es uno diseñado para el almacenamiento de los datos de las vistas con extensión *.GRA y que está definido en el apéndice B.
- *Ver fichero de texto* : Muestra un típico cuadro de diálogo de Windows como el de *abrir fichero*, con el objeto de seleccionar un fichero de texto que será abierto con el programa NOTEPAD. Esta opción tiene interés cuando queremos ver el contenido de los ficheros *.GRA de los datos de entrada y queramos modificar su contenido. También podremos ver los ficheros de texto con la información y estadísticas de la reconstrucción que se guardan con la opción *guardar info como*.
- *Guardar como ...* : Muestra un típico cuadro de diálogo de *guardar como*, en él se nos pide un nombre de fichero, con el cual se guardará el objeto válido 3D seleccionado en formato DXF. Si no se selecciona un objeto válido, aparece en el menú desplegable en color gris, lo que significa que no se puede seleccionar.
- *Guardar Info como ...* : Muestra un típico cuadro de diálogo de *guardar como*, en él se nos pide un nombre de fichero, con el cual se guardará un fichero de texto con toda la información concerniente al proceso de reconstrucción, así como de las estadísticas generadas por el propio programa.

- *Imprimir* : Muestra un típico cuadro de diálogo de Windows de impresión. Se imprimirá en la impresora seleccionada todo lo que se vea en la pantalla que puede ser sólo el objeto 3D o las vistas y el objeto 3D a la vez.
- *Nuevo Rec3D ...* : Opción no implementada.
- *Salir* : Acaba el programa, previa confirmación.

5.1.2. LOS MENÚS “Dibujos1”, “Dibujos2” Y “Dibujos3”.

Aquí tenemos una serie de dibujos predefinidos que se reconstruyen seleccionando el dibujo deseado:

5.1.3. EL MENÚ “Acciones”.

En este menú desplegable tenemos una serie de opciones que en su mayoría están en la barra de herramientas "Acciones", estas opciones tienen que ver con las características del objeto:

- *Vértices* : Muestra los vértices 3D del objeto.
- *Aristas* : Muestra las aristas 3D del objeto.
- *Gráficos Planos* : Muestra los gráficos planos del objeto.
- *Bucles de caras* : Muestra los bucles de caras del objeto.
- *Bucles de cuerpos* : Muestra los bucles de cuerpos del objeto.
- *Objetos Válidos* : Muestra los objetos válidos de la reconstrucción.
- *Datos de reconstrucción* : Muestra las cantidades de información generadas por el algoritmo de reconstrucción: vértices y aristas 2D, vértices y aristas 3D, gráficos planos, bucles de caras, bucles de cuerpos y sus combinaciones, y los objetos

válidos. Estos datos se guardan en fichero de texto con la opción *guardar info como* del menú *Archivo*.

- *Estadísticas* : Muestra los tiempos de ejecución de cada módulo del algoritmo, el tiempo total de ejecución y los porcentajes del tiempo de ejecución de cada módulo con respecto al tiempo de ejecución total. Estos datos también se guardan en fichero de texto con la opción *guardar info como* del menú *Archivo*.
- *Animación* : Si se ha reconstruido algún objeto y se selecciona esta opción del menú, aparece un primer cuadro de dialogo en el que podemos introducir tres datos: incremento del ángulo de longitud y latitud de cada movimiento de la animación y la velocidad de la animación. En el siguiente cuadro de dialogo le indicamos al programa como realizar la animación, además hay un botón de pausa y otro para salir de la animación.
- *Render* : Esta opción del menú sólo permanecerá activa cuando hayamos seleccionado algún objeto válido. Entonces al seleccionarla aparecerá un cuadro de diálogo de tipo paleta de colores en el cual seleccionamos un color. Cuando hagamos click en el botón aceptar, podremos ver como el objeto valido activo se sombrea con el color que hayamos seleccionado teniendo la apariencia de un objeto sólido, en lugar del modelo alámbrico.
- *Ver proceso de reconstrucción ...* : Con esta opción, se va mostrando el proceso de reconstrucción paso a paso para poder ver como se va modificando el modelo alámbrico.

5.1.4. EL MENÚ “Herramientas”.

En este menú desplegable tenemos una serie de opciones que están en la barra de herramientas "Herramientas", estas opciones tienen que ver con la visualización del objeto:

- *Zoom +/-* : Acerca o aleja el objeto 3D.
- *Vistas Zoom +/-* : Acerca o aleja las vistas ortográficas.
- *Longitud +/-* : Realiza una rotación en longitud.
- *Latitud +/-* : Realiza una rotación en latitud.
- *Vistas/Objeto* : Conmuta entre cuatro ventanas (vistas y objeto) y sólo una (objeto).

- *Traslación* : Realiza una traslación del objeto, hay que proporcionarle mediante un cuadro de diálogo la traslación en X, en Y, y en Z, con respecto a la posición actual.
- *Ángulos de rotación* : Fija el incremento del ángulo de rotación en longitud o en latitud que queramos para las rotaciones.
- *Perspectiva Isométrica* : Fija la ventana del objeto 3D en perspectiva isométrica.

5.1.5. EL MENÚ “Opciones”.

En este menú desplegable tenemos una serie de opciones que en su mayoría están en la barra de herramientas "Acciones", estas opciones tienen que ver con las partes del algoritmo; además hay otra serie de opciones de visualización de objetos del programa:

En el primer bloque tenemos unas casillas de verificación que indican a la aplicación los pasos del algoritmo de reconstrucción que queremos que se ejecuten de forma secuencial. En el segundo bloque, tenemos las opciones de visualización de los Ejes 3D (X, Y, Z); de las dos barras de herramientas ("Acciones" y "Herramientas"); del Monitor de la Reconstrucción, que es un cuadro de diálogo que aparece cuando se está realizando la reconstrucción y nos va indicando en cada momento la situación actual; y por último, si hemos realizado un *render*, podemos conmutar entre el objeto “renderizado” y en modelo alámbrico con esta opción (*Ver objeto renderizado*). La última opción se utiliza para limitar el numero de combinaciones de bucles de cuerpos que genera el algoritmo de reconstrucción, por defecto tiene 1024.

5.1.6. EL MENÚ “Ayuda”.

Como todos los programas de entorno Windows, y en general todos los programas, deben disponer de un elemento de ayuda que permita al usuario, ante un problema que haya surgido, obtener información sobre la aplicación y su manejo. En este menú se puede disponer de una sencilla ayuda sobre todas las acciones de los menús y su utilización.

5.2. BARRAS DE HERRAMIENTAS.

Como la mayoría de programas estilo Windows tenemos una serie de barras de herramientas, flotantes por la pantalla, que representan opciones de los menús desplegables en forma de botones, teniendo un acceso más rápido a estas opciones o acciones. Tenemos dos barras de herramientas flotantes que vamos a ver a continuación.

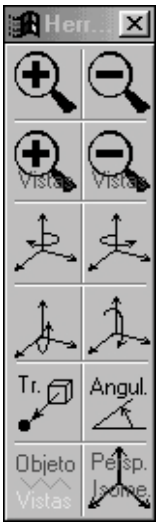
5.2.1. BARRA "Acciones".

Tenemos que esta barra de herramientas está formada por acciones del menú desplegable "Acciones" y por opciones del menú desplegable "Opciones".

Tenemos que los botones de esta barra de herramientas: *Vert.*, *Arist.*, *Plan*, *Bucl. Cara*, *Bucles de Objetos* y *Objetos válidos* muestran respectivamente los vértices 3D, las aristas 3D, los gráficos planos, los bucles de caras, los bucles de cuerpos y los objetos válidos del objeto sobre el que hemos realizado la reconstrucción 3D. Por lo tanto, estos botones son equivalentes a las primeras seis acciones que tenemos en el menú desplegable "Acciones".

A continuación se tiene una serie de cuadros de verificación u opciones que marcan las partes del algoritmo que se van a ejecutar en la reconstrucción 3D del objeto. El primer paso del algoritmo es WC, el último BLG, y no se pueden tener varias opciones salteadas y seleccionadas porque los pasos del algoritmo son secuenciales. Por lo tanto, estas opciones son equivalentes a las primeras seis opciones que tenemos en el menú desplegable "Opciones".

5.2.2. BARRA "Herramientas".



Tenemos que esta barra de herramientas está formada por acciones del menú desplegable "Herramientas".

En esta barra de herramientas tenemos que los cuatro primeros botones (los que tienen una lupa dibujada) sirven para realizar zoom de acercamiento (+) o zoom de alejamiento (-) sobre el objeto 3D o sobre las vistas. Los siguientes botones realizan una rotación en longitud o latitud en el sentido de las flechas dibujadas en los botones. También tenemos los botones de traslación y ángulos de rotación, perspectiva isométrica y conmutación entre ventana de vistas y objeto, y ventana de objeto.

Como se puede comprobar, tenemos que todas las acciones del menú herramientas están incluidas en esta barra de herramientas para tener un acceso más rápido a estas acciones.

Capítulo 6
CONCLUSIONES

El número de investigaciones en la reconstrucción de información 3D, a partir de la información 2D, se ha incrementado rápidamente en los últimos tiempos. Se han realizado continuos logros en la recuperación de modelos sólidos a partir de sus vistas planas, pero los métodos de los algoritmos descritos en la literatura no son completos, en mayor o menor grado.

En este proyecto final de carrera, se propone el algoritmo de Qing-Wen Yan que maneja la reconstrucción de modelos sólidos poliédricos 3D a partir de sus vistas planas. El algoritmo puede encontrar una única solución o múltiples soluciones, pero además elimina casos patológicos, si hay alguno. El algoritmo garantiza que se va a encontrar la solución correcta a partir de unos datos de entrada, con tal que las vistas sean correctas. El algoritmo toma toda la ventaja de la información de las líneas discontinuas para eliminar otros casos patológicos. Este algoritmo utiliza los hipotéticos vectores normales exteriores para clasificar los bucles de caras y cuerpos, y asistir en la construcción de bucles de cuerpos. Eventualmente, ajustando los hipotéticos vectores normales exteriores para cada bucle de cara del objeto, se obtiene el auténtico vector normal que podemos utilizar para la construcción de la representación de fronteras completa del objeto.

Con este proyecto se ha conseguido implementar una herramienta de ayuda al dibujo. Se ha repasado históricamente la mayoría de algoritmos de reconstrucción 3D que se han investigado; hemos visto las distintas formas de representar objetos

tridimensionales en un plano (o dos dimensiones); se ha estudiado con profundidad el algoritmo de reconstrucción 3D de Qing-Wen Yan, entrando en todos los detalles que llevan a cabo la implementación de este algoritmo; y se ha realizado un estudio de coste temporal de las partes que componen este algoritmo.

Todos estos aspectos se han cubierto sobradamente e incluso se han ampliado y mejorado alguno de ellos que no habían sido mencionados en la bibliografía. Por ejemplo, podemos citar el cálculo de la inclusión y otros análisis entre bucles de caras que sean cóncavos, que en el artículo de Qing-Wen Yan no menciona nada, y su resolución ha sido una de las partes más complicadas en la implementación de este algoritmo.

En la tabla siguiente, podemos ver una comparativa de los tiempos de ejecución, de los ejemplos estudiados en el capítulo 4, en distintos ordenadores con distinto procesador. Sólo hacemos referencia a la frecuencia de reloj del procesador porque suponemos, como ocurre en realidad, que el programa y sus datos cabe completamente en memoria RAM y no realiza intercambios a disco duro. Por lo tanto, en la ejecución del algoritmo, sólo interviene la velocidad del procesador.

	i486DX 33MHz		Pentium 100MHz		Pentium 233MHz	
	T.ejec.	% Verific.	T.ejec.	% Verific.	T.ejec.	% Verific.
Ejemplo 1	4146 ms	37	744 ms	41	295 ms	40
Ejemplo 2	6,5 horas	99,9	4,3 horas	99,9	97 min.	99,9
Ejemplo 3	20764 ms	61	4183 ms	59	1535 ms	62
Ejemplo 4	42258 ms	60	7791 ms	60	3025 ms	61

Como se puede ver en la tabla anterior, el desarrollo de la tecnología posibilita que problemas que hace algunos años eran intratables, ahora es posible su análisis pese a su alto coste temporal. Esto es debido a que el procedimiento de *Verificación*, dentro del algoritmo de reconstrucción, tiene un coste temporal exponencial, y el porcentaje de ejecución de este procedimiento con respecto al tiempo total de ejecución del algoritmo llega a ser de casi el 100%.

Como posible ampliación y mejora, recomendaría: Principalmente, la optimización del procedimiento de *Verificación* para mejorar el tiempo de ejecución; otra mejora, sería la utilización de un fichero gráfico, como por ejemplo DXF, para introducir los datos de entrada; y por último utilizar un esquema de memoria dinámica, para el almacenamiento de todas las variables, así se eliminan limitaciones de los objetos.

Apéndice A: Artículo de Qing-Wen Yan

ALGORITMO EFICIENTE PARA LA RECONSTRUCCIÓN DE OBJETOS 3D A PARTIR DE PROYECCIONES ORTOGRÁFICAS.♣

Qing-Wen Yan, C L Philip Chen* y Zesheng Tang✦

Un algoritmo eficiente para la reconstrucción de todos los modelos de sólidos 3D poliédricos a partir de proyecciones ortográficas 2D ha sido dirigido. El algoritmo puede manejar casos patológicos e identificar posibles soluciones correctas eficientemente. El algoritmo posee técnicas eficientes para construir apropiadas figuras alámbricas, manejar líneas discontinuas de los datos de entrada, generar bucles de caras, determinar bordes y vértices a eliminar, construir bucles de cuerpos y ensamblar los bucles de cuerpos en los objetos apropiados. Algunas proyecciones ortográficas complejas y simétricas son dadas para mostrar la integridad del algoritmo.

Palabras clave: reconstrucción, modelos sólidos, proyecciones ortográficas.

En el proceso del diseño asistido por computador en la ingeniería mecánica y fabricación, la representación automática y construcción de modelos sólidos con ordenadores es un paso vital. Tres aproximaciones son comúnmente usadas para crear

♣ Traducido por Lorenzo J. Muñiz Márquez

Scanning Management Systems, Canoga Park, CA 91304, USA

* Departamento de Informática e ingeniería, Wright State University, Dayton, Ohio 45435, USA

✦ Departamento de Informática y tecnología, Tsinghua University, Pekín, China

modelos sólidos: (a) modelado por primitivas, (b) barrido por rotación y traslación, y (c) reconstrucción de proyección planar. El modelado por primitivas usa directamente objetos 3D simples, incluyendo cubos, cilindros y conos. Un conjunto de operaciones booleanas normalizadas, tales como intersección, unión y diferencia, son aplicadas a estos objetos para definir estructuras complejas. El modelado por barrido es una técnica que usa líneas base de objetos para modelar un objeto sólido. El modelado por barrido construye modelos geométricos trasladando el contorno de un objeto a lo largo de una trayectoria o rotando una sección alrededor de su eje. Los primeros dos métodos han sido aplicados en muchos paquetes de software comerciales. Por el contrario la reconstrucción por proyección es una aproximación orientada a 2D que se basa en una analogía con los dibujos de la ingeniería convencional. Genera modelos 3D a partir de un conjunto de proyecciones 2D planares, tales como planta, alzado, perfil, y vistas locales. Es un procedimiento de recuperar información de una dimensión inferior a una dimensión superior. Aunque obtener proyecciones 2D para un objeto 3D dado es sencillo, la operación inversa llega a ser algo implícito y difícil. Las dificultades de esta aproximación se originan de la pérdida de información semántica cuando un objeto 3D es representado con proyecciones 2D.

Aunque la literatura sobre reconstrucción 3D es extensa, unos pocos autores han dado un algoritmo completo y han elegido representar los objetos formalmente [1] [2] . Normalmente, los algoritmos no son capaces de manejar el rango completo de casos patológicos y ambigüedades que ocurren en la práctica. La mayoría de los métodos primarios anteriores consisten en los pasos siguientes:

- Generar vértices 3D apropiados a partir de nodos 2D.
- Construir cantos 3D a partir de vértices 3D.
- Formar las caras de los objetos a partir de los lados 3D.
- Construir componentes de objetos a partir de las caras.
- Combinar componentes para encontrar soluciones.

El algoritmo de Idesawa usó la aproximación 'bottom-up' de arriba y dio un conjunto de reglas para juzgar los bordes y vértices patológicos en el proceso de reconstrucción. El trabajo se repartió con objetos simples sin casos patológicos o casos con múltiples soluciones. Aldefeld [2] [3] intentó interpretar los objetos 3D sobre la base del reconocimiento de patrones, usando búsquedas heurísticas en lugar del método 'bottom-up'. Una proyección 2D es considerada como un conjunto de elementos gráficos 2D primarios, tales como líneas rectas, rectángulos, arcos y círculos. Estos elementos están clasificados en diferentes grupos de acuerdo a distintas características de patrones. Una búsqueda heurística es usada entonces para buscar las posibles soluciones en el espacio. El algoritmo está restringido a objetos gruesos uniformes, y se puede generar una posible solución en la cual no siempre se puede recuperar la semántica 3D completa de los objetos. El algoritmo de Markowsky y Wesley[4] también utiliza la aproximación 'bottom-up'. El método puede utilizar objetos poliédricos, y permite la existencia de objetos no múltiples, pero no puede manejar proyecciones de líneas rotas (por ejemplo, líneas discontinuas). La aproximación puede tratar casos patológicos o con múltiples soluciones. Sakurai y Gossard [5] extendieron el trabajo de Markowsky y Wesley. Consideraron objetos con superficies esféricas, cónicas y cilíndricas, exigiendo que los ejes de la superficie curva fueran paralelos a uno de los ejes de coordenadas. Gujar [6] propuso un algoritmo que puede procesar sólo objetos poliédricos. Sin embargo, el algoritmo no puede rechazar casos patológicos, y sólo maneja proyecciones de línea continua. Gu et al. [7] combinó el método 'bottom-up' con la aproximación por reconocimiento de patrones, y permitió a los ejes cilíndricos ser paralelos a uno de los planos de coordenadas. Sin embargo, no son estudiadas las proyecciones de líneas rotas.

Nosotros ideamos un algoritmo de reconstrucción para encontrar todos los objetos poliédricos para un conjunto dado de proyecciones ortográficas 2D. El algoritmo usa información tal como líneas continuas o discontinuas en las proyecciones para reducir la cantidad de caras y cuerpos potencialmente computables. También reduce eficazmente el posible número de objetos ambiguos. Las ambigüedades causadas por el hecho que muchos objetos pueden tener el mismo conjunto de proyecciones dado, y el algoritmo puede reconocer todos los sólidos correctos de un conjunto de proyecciones dado. Nuestro algoritmo puede tratar las cuestiones de cómo identificar los datos de entrada, cómo generar el apropiado modelo alámbrico más eficazmente, cómo tomar la ventaja de los atributos de línea continua/discontinua para acelerar el proceso de encontrar las soluciones correctas, y cómo garantizar que todas las soluciones posibles y correctas, para unas entradas dadas, son encontradas.

Resolvemos estas cuestiones en las siguientes secciones. La próxima sección describe nuestro algoritmo. Varios ejemplos ejecutados por nuestro algoritmo son mostrados y analizados. Finalmente, resumimos el algoritmo, y sugerimos un futuro trabajo. En el apéndice, definimos los conceptos geométricos usados en el artículo, tales como objeto, cara, bucle, bucle de cara, bucle de cuerpo, modelo alámbrico y aristas/vértices de corte.

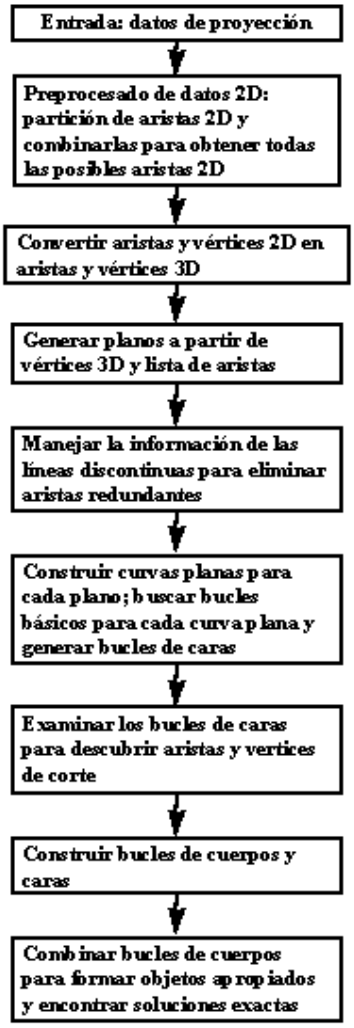


Figura 1: Algoritmo.

ALGORITMO DE RECONSTRUCCIÓN

El diagrama de bloques de nuestro algoritmo se muestra en la figura 1. En cada etapa el algoritmo genera toda la información posible, y elimina vértices, aristas y caras incorrectas usando un conjunto de criterios para acelerar la formación de un objeto correcto. Antes de construir el objeto, p_vertex y p_edge son, respectivamente, un vértice y una artista 3D potencialmente para la construcción del objeto. Los pasos principales del algoritmo son brevemente descritos a continuación.

(1) Convertir los vértices y aristas 2D estructurados en una lista de vértices 2D y una lista de aristas 2D. Las proyecciones son representadas mediante elementos gráficos 2D tales como líneas, rectángulos, polilíneas, polígonos, círculos y arcos (ver figura 2a).

(2) Construir todos los posibles vértices 3D y sus correspondientes aristas a partir de la lista de vértices y la lista de aristas, y verificar su corrección (ver figura 2b).

(3) Construir todas las posibles caras planas para generar los bucles de caras, y verificar su unicidad y corrección.

(4) Manejar la información de las líneas discontinuas para incrementar la eficiencia de la computación.

(5) Formar los bucles de caras para generar los bucles de cuerpos como se muestra en la figura 2d.

- **(5.1)** Encontrar todos los gráficos planos.
- **(5.2)** Buscar los generadores de bucles.
- **(5.3)** Determinar la relación entre los bucles básicos, y construir una tabla de relación de inclusión para cada gráfico plano.
- **(5.4)** Generar los bucles de caras.

(6) Examinar la relación entre los bucles de caras si existen algunos vértices y aristas de corte e_1 y e_2; la figura 2f muestra los nuevos bucles de caras formados por los bucles de caras de corte F_1 - F_2 en la figura 2d.

(7) Construir todos los bucles de cuerpos, por ejemplo bloques para generar objetos como se muestra en la figura 2g.

(7.1) Encontrar los sucesivos bucles de caras de un bucle de cuerpo dado.
(7.2) Establecer bucles de cuerpo elementales.
(7.3) Verificar la proximidad de cada bucle de cuerpo
(7.4) Clasificar los bucles de cuerpos.

(8) Combinar los bloques en objetos, y tomar decisiones finales como se muestra en la figura 2h.

(8.1) Reorganizar los bucles de caras para generar nuevos bucles de caras.
(8.2) Verificar la corrección de un objeto hipotético.
(8.3) Verificar la consistencia entre el objeto generado y las vistas del objeto.

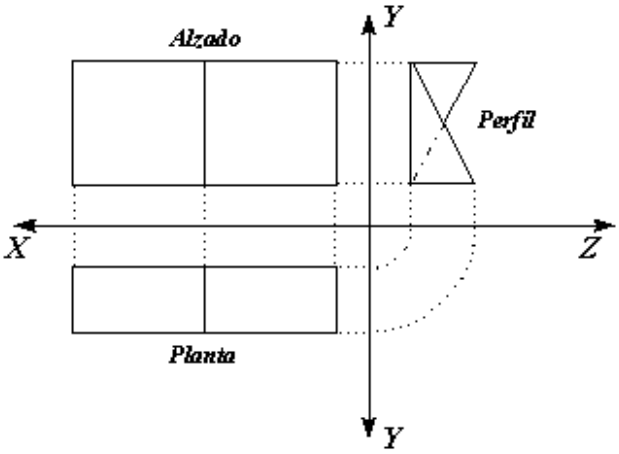


Figura 2a

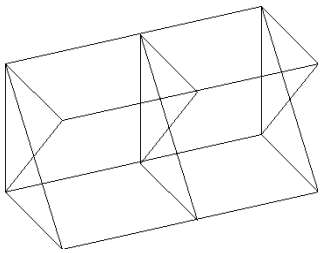


Figura 2b

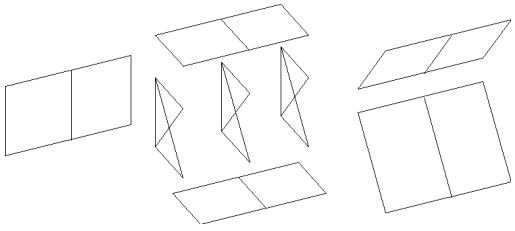


Figura 2c

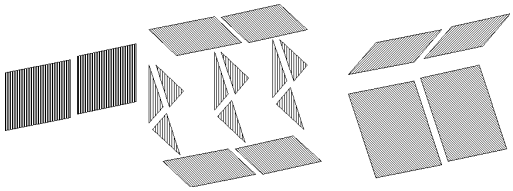


Figura 2d

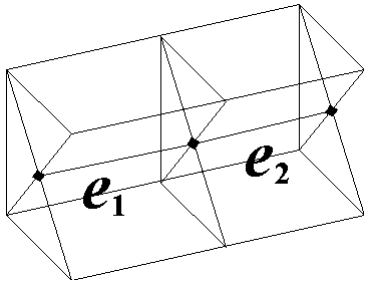


Figura 2e

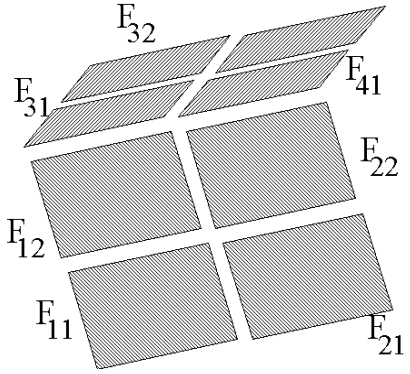


Figura 2f

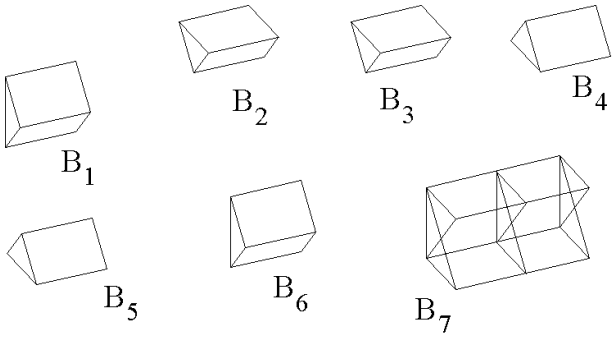


Figura 2g

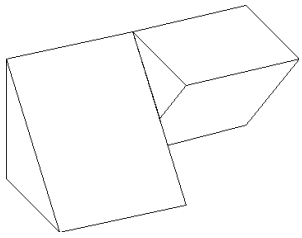


Figura 2h

Figura 2, Ejemplo: (a) tres vistas de un objeto, (b) modelo alámbrico del objeto de la figura 2a, (c) gráficos planos de la figura 2b, (d) bucles de caras formados de cada gráfico plano, (e) vértices y aristas de corte, (f) nuevos bucles de caras creados por corte de los bucles de caras, (e) bucles de cuerpos B_1 - B_6 y bucle de cuerpo exterior B_7, (h) objeto correcto.

El algoritmo es detallado a partir de ahora.

CONSTRUCCIÓN DEL MODELO ALÁMBRICO

Durante el proceso de reconstrucción, las aristas redundantes o patológicas pueden ser generadas a través de trazar los vértices y aristas 2D en vértices y aristas 3D. **v_list(•)** será una lista de vértices 2D, en la cual cada vértice consiste en un valor de coordenadas *(x,y)*, donde **(•)** puede ser **f**, **t**, o **s**, representando las proyecciones del alzado (front), planta (top) y perfil (side), respectivamente. **e_list(•)** será una lista de aristas 2D, en las cuales cada arista registra índices de **v_list(•)** que son puntos finales, la información más detallada (líneas continuas y discontinuas) de cada arista, y el tipo de arista (arista simple o completa). En esta sección, ideamos un algoritmo de construcción del modelo alámbrico que puede eliminar aristas redundantes y patológicas. El algoritmo combina la búsqueda en árbol de decisión basado en proyecciones del alzado, algoritmo de eliminación de aristas redundantes, y un conjunto de reglas para eliminar aristas patológicas y redundantes.

Trabajos anteriores [1 3 6] han mostrado que las aristas y vértices 3D pueden ser generados en dos pasos. Primero, generar vértices 3D eligiendo un vértice de cada lista **v_list(•)** para cada proyección, y comparar las coordenadas de los tres vértices en sus ejes de coordenadas comunes. La comparación termina cuando no se crean más vértices 3D. Segundo, generar aristas 3D seleccionando un par de vértices (asumir que hay una arista entre los dos vértices). Si las tres proyecciones de esta hipotética arista están todas en sus correspondientes **e_list(•)**, se crea una arista p_edge; en otro caso, seleccionar otro par de vértices, y repetir este proceso de hipótesis y verificación hasta que no se creen más aristas 3D.

Este método requiere una gran cantidad de computación para la comparación de aristas y vértices, y carece de eficiencia. Nosotros diseñamos una aproximación eficiente para la generación de aristas y vértices 3D directamente desde las listas **v_list(•)** y **e_list(•)** en *solo un paso*, sin la generación separada de vértices y aristas 3D. Las aristas 2D en **e_list(•)** pueden ser clasificadas en los siguientes cinco grupos:

- Caso 1: paralelas al eje ***X***.
- Caso 2: paralelas al eje ***Z***.
- Caso 3: no paralelas a ningún eje de coordenadas.
- Caso 4: un punto - foco formado por acumulación.
- Caso 5: paralelas al eje ***Y***.

La figura 3 muestra cada caso para diferentes posiciones y operaciones de una arista. En la figura 3a, las proyecciones de alzado y planta de una arista p_edge pertenecen al caso 1, mientras en la figura 3b, su proyección de planta cae en el caso 3. Las proyecciones de perfil en las figuras 3a y 3b pertenecen al caso 4 y caso 5 respectivamente. Todas las demás proyecciones se muestran en las figuras 3c-g. En la base de la proyección del alzado, se construye un árbol de decisión, como se muestra en la figura 4, para revelar información 3D sobre una arista dada en **e_list(•)**.

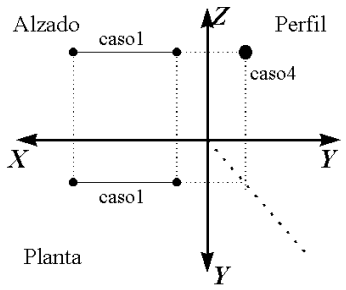


Figura 3a

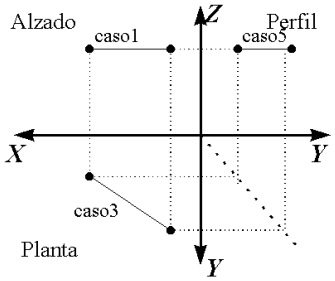


Figura 3b

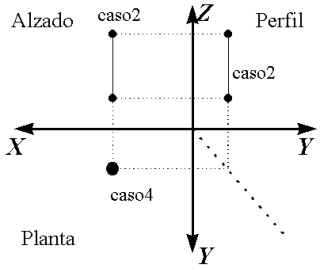


Figura 3c

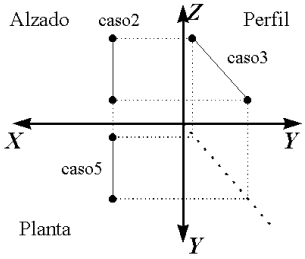


Figura 3d

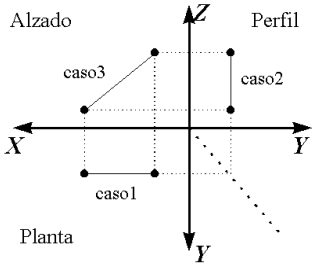


Figura 3e

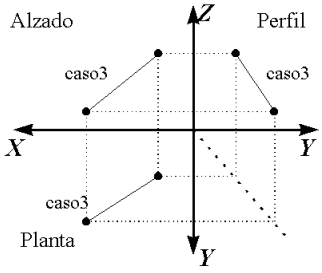


Figura 3f

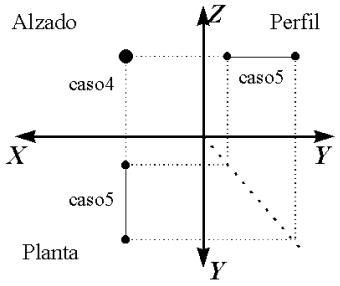


Figura 3g

Los vértices y aristas 3D pueden ser generados recorriendo este árbol de decisión con el algoritmo de construcción del modelo alámbrico, el cual se va a describir a continuación. Los símbolos utilizados para este algoritmo son los siguientes:

fe = arista seleccionada de **e_list(f)**.
te = arista seleccionada de **e_list(t)**.
se = arista seleccionada de **e_list(s)**.
fv = vértice seleccionado de **v_list(f)**.
tv = vértice seleccionado de **v_list(t)**.
sv = vértice seleccionado de **v_list(s)**.

En la figura 4; 1, 2, 3, 4 y 5 corresponden a los casos discutidos antes.

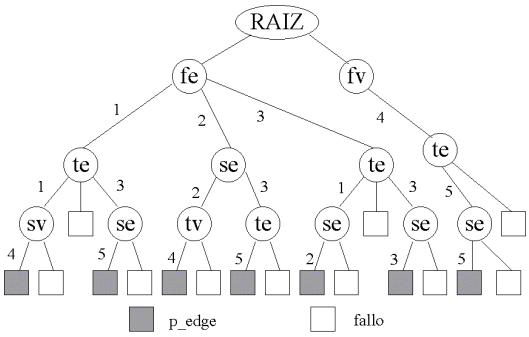


Figura 4

Algoritmo WC (wireframe construction, construcción del modelo alámbrico): Dadas las listas v_list(•) y e_list(•), el algoritmo genera el modelo alámbrico de un objeto sólido.

(W1) [Recorrer el árbol de decisión (como se muestra en la figura 4) para generar aristas y vértices 3D]. Primero seleccionar una arista *fe* y trazar el árbol de decisión para decidir a que caso (1,2,3) pertenece la arista. A continuación, seleccionar *te* para el caso 1 ó 3, o seleccionar*se* para el caso 2. Trazar el árbol hasta alcanzar las hojas para ver si una arista p_edge puede ser generada. Finalmente, seleccionar un *fv* y trazar el árbol hasta el nodo *te* . Entonces seleccionar un *te* del caso 5, si tiene éxito, para alcanzar el nodo *se*. En el nodo *se* , intentamos seleccionar un *se* del caso 5 para obtener un p_edge. Si esto falla, se recorre el árbol hacia atrás para seleccionar un nuevo *te*. Cuando alcanzamos los nodos hoja, creamos una lista **p_vlist** que consiste en los vértices no duplicados de cada arista p_edge. Similarmente, creamos una lista **p_elist** que consiste en las aristas p_edge no duplicadas.
La arista p_edge se representa por un par de vértices p_vertices. Las listas p_vlist y p_elist contienen todas las posibles aristas y vértices 3D. Algunas de estas aristas pueden solaparse parcialmente y deben ser tratadas en el paso siguiente.

(W2) [Resolver las aristas solapadas en p_elist]. Las aristas solapadas son redundantes. No sólo incrementan la complejidad de computación, sino también introduce ambigüedades en el proceso de generación de bucles básicos y bucles de caras. La figura 5 muestra tres casos de solapamiento. Para el modelo alámbrico de la figura 5a, los vértices A, B, C, D están en la lista p_vlist. En la lista p_elist, la arista AB solapa con la AD, figura 5b, o la arista BC solapa con la AD, figura 5c, o la arista AC solapa con la BD, figura 5d. Las partes solapadas de las aristas se consideran como 'aristas redundantes', son eliminadas por el procedimiento RER (Redundant - Edges Removal, eliminación de aristas redundadntes).

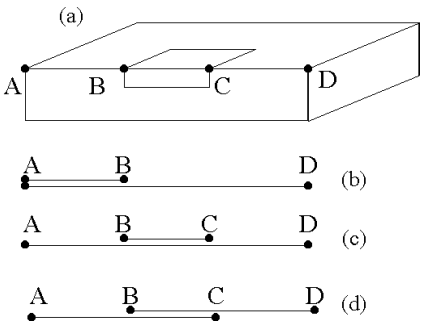


Figura 5

Procedimiento RER : dada una lista p_elist generada en el paso W1, el procedimiento elimina aristas redundantes de p_elist.

(W2.1) Inicialmente se marcan todas las aristas en p_elist como 'no examinadas'.
(W2.2) Seleccionar una arista 'no examinada' e_i de p_elist, y hacer $E \leftarrow \{ e_i \}$. Si todas las aristas han sido examinadas, el procedimiento RER acaba.

(W2.3) Para cada arista $e_j \in$ p_elist, $e_j \neq e_i$, si e_j y e_i son colineales y e_j solapa con una arista de E, entonces $E \leftarrow E \cup \{ e_j \}$.
(W2.4) Si sólo hay una arista en E, entonces ninguna otra arista solapa con la arista de E. Se marca esta arista como 'examinada', y pasar al paso W2.2, en otro caso continuar.
(W2.5) Si hay más de una arista en E, entonces eliminar estas aristas de p_elist y ordenar todos los vértices de acuerdo con sus valores de coordenadas. $v_1, v_2, \ldots, v_K$ serán una secuencia de vértices ordenados. Añadir las aristas (v_j , v_{j+1}), j = 1, ... , K-1 a p_elist, y marcarlas como 'examinadas'. Ir al paso W2.2.

Usando este procedimiento, las aristas en p_elist son todas únicas y no solapadas. Después del procedimiento RER, por ejemplo, sólo las aristas AB y BD pertenecen a p_elist para la figura 5b,y sólo las aristas AB, BC, CD para las figuras 5c y 5d. Además se han eliminado aristas solapadas redundantes, algunas aristas / vértices patológicos, tales como aristas colgantes , vértices y aristas aisladas, pueden existir y deben ser eliminadas. El procedimiento PEVR (Pathological Edges/Vertices Removal, eliminación de vértices / aristas patológicas) es designado para eliminar estas aristas / vértices. Este procedimiento se da en el paso W3.

(W3) [Eliminar aristas y vértices patológicos]. ρ(p_vertex) representa el número de aristas p_edges compartidas en p_vertex. Las aristas o vértices patológicos son eliminados por el procedimiento PEVR.

Procedimiento PEVR : Dadas las listas p_vlist y p_elist, el procedimiento elimina aristas y vértices patológicos.

(W3.1) Borrar un vértice aislado p_vertex si ρ(p_vertex)=0. Ver el vértice *v* en la figura 6a.
(W3.2) Si ρ(p_vertex)=1, eliminar la línea colgante de p_elist en el vértice p_vertex, y eliminar p_vertex de p_vlist. Como se muestra en la figura 6b, la arista *e* y el vértice *v* son eliminados.
(W3.3) Si ρ(p_vertex)=2, y dos aristas son colineales, entonces borrar p_vertex y unir las dos aristas adyacentes en p_vertex. Como se muestra en la figura 6c, *v* es eliminada, y e_1 , e_2 son unidas. Si ρ(p_vertex)=2 y dos aristas no son colineales, entonces borrar p_vertex, y las dos aristas adyacentes al punto p_vertex. En la figura 6d, *v*, e_1 y e_2 son eliminados.
(W3.4) Si ρ(p_vertex)=3, y dos de tres aristas son colineales, entonces borrar p_vertex y la tercera arista. Unir las dos aristas colineales. En la figura 6e, *v* y e_2 son eliminados; e_1 y e_3 son unidos.
(W3.5) Si ρ(p_vertex)=3, y tres aristas son coplanares, pero no hay dos aristas colineales, entonces borrar p_vertex y esas tres aristas. En la figura 6f, *v* , e_1 , e_2 y e_3 son eliminados.
(W3.6) Si ρ(p_vertex)≥4, las 4 aristas son coplanares, y sólo dos aristas son colineales, entonces se unen estas dos aristas, y se borra p_vertex y las aristas no colineales. En la figura 6g, *v*, e_2 y e_3 se eliminan, y e_1 y e_4 se unen. Si dos pares de aristas son colineales o todas las aristas son no colineales, entonces se borra p_vertex y

todas las aristas. En la figura 6h y 6i, v , e_1 , e_2 , e_3 y e_4 son eliminadas.

Fin del Algoritmo WC.

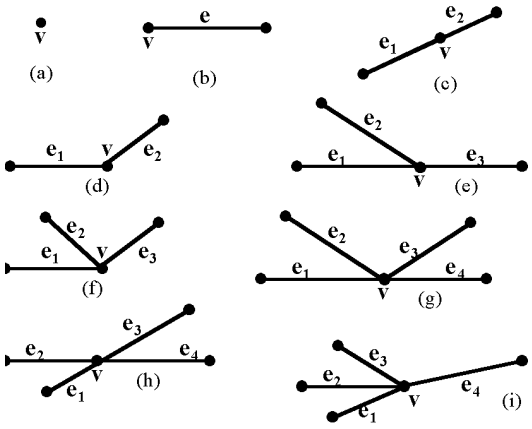


Figura 6: eliminación de aristas redundantes; (a) deg=0; (b) deg=1; (c) deg=2; (d) deg=2; (e) deg=3; (f) deg=3; (g) deg=4; (h) deg=4; (i) deg=4.

Hemos establecido el modelo alámbrico de un objeto con las listas p_vlist y p_elist que contienen información acerca de las aristas y vértices 3D. Después de haber obtenido el modelo alámbrico, necesitamos construir los gráficos planos para conseguir la información de las curvas de caras. Es esencial mencionar que el procedimiento PEVR elimina las aristas y vértices patológicos actuales generados en el paso W2. Sin embargo, algunas aristas patológicas potenciales pueden ser descubiertas en los procedimientos de generación de curvas de caras que se van a describir. El procedimiento PEVR se aplica otra vez, si se encuentran aristas y vértices patológicos, hasta que no haya más. El procedimiento PEVR alcanza finalmente un estado estable en p_elist y p_vlist.

CONSTRUCCIÓN DE GRAFICOS PLANOS

El algoritmo WC construye el modelo alámbrico de los datos de entrada (aristas y vértices 2D). El modelo alámbrico no da explícitamente información sobre la superficie de los objetos. Necesitamos construir los gráficos planos (planar graphs) a partir del modelo alámbrico para generar la información de la superficie de los objetos. Un gráfico planar es una colección de aristas coplanares. El algoritmo para generar todos los gráficos planos se discute en esta sección. Los datos de entrada de este algoritmo son las listas p_elist y p_vlist obtenidos del algoritmo WC.

Algoritmo PGG (planar-graph generation, generación de gráficos planos) :

(P1) Registrar las aristas adyacentes para cada vértice de p_vlist.

(P2) [Construir planos]. Un plano puede ser determinado por dos aristas adyacentes no colineales en un vértice. Para cada v_i de p_vlist, construir un plano determinado por e_1 y e_2, donde $e_1 = (v_i , v_j)$ y $e_2=(v_i,v_k)$, $i \neq k$.

(P3) [Hipotetizar el vector normal exterior del plano]. Hasta no saber cuál es la cara exterior del plano construido arriba, el vector normal exterior del plano no puede ser determinado. El hipotético vector normal de un plano es un vector perpendicular al plano. Como se muestra en la figura 7. **OP** es una normal exterior hipotética de ese plano. La cara, a partir de la cual, sale el hipotético vector normal del plano es la cara positiva del plano, y la otra cara, la cara negativa. En la figura 7, +N es el vector normal exterior hipotético del plano en la cara positiva. Puede ser obtenido:

$$N = \begin{cases} -(a\vec{i} + b\vec{j} + c\vec{k}) & d < 0 \\ a\vec{i} + b\vec{j} + c\vec{k} & \text{en otro caso} \end{cases}$$

donde i , j , k son vectores unitarios en las direcciones X, Y, Z, respectivamente.

(P4) [Eliminar un plano duplicado]. Borrar el plano j si existe un plano i, $i \neq j$, tal que la distancia $D_j \leq \xi$ (ξ es una tolerancia, 10^{-5}), donde D_j es aproximado por $((a_i-a_j)^2+(b_i-b_j)^2+(c_i-c_j)^2+(d_i-d_j)^2)^{1/2}$, ($a_i$, b_i, c_i, d_i) y (a_j, b_j, c_j, d_j) son los coeficientes de las ecuaciones de los planos correspondientes.

(P5) [Búsqueda de todas las aristas del plano]. Para cada arista p_edge de p_elist, computar las distancias de sus dos puntos finales D_1 y D_2 al plano. Si $|D_1| \leq \xi$ y $|D_2| \leq \xi$, p_edge está en el plano. Las aristas p_edge que están en el plano serán usadas para generar las curvas de caras.

(P6) [Verificar la corrección de cada gráfico plano]. Aunque todos los gráficos planos son generados, algunos son patológicos, y necesitan ser ajustados o eliminados.

(P6.1) Si un gráfico plano consta de dos o menos aristas, descartarlos.
(P6.2) Si una arista p_edge de p_elist pertenece a sólo un gráfico plano, borrar p_edge. Aplicar el procedimiento PEVR para eliminar cualquier nueva arista o vértice patológico que aparezca.
(P6.3) Si la frontera de un gráfico plano no es cerrada, por ejemplo, que existan aristas colgantes o aisladas, como se muestra en la figura 8, eliminar las aristas colgantes o aisladas para este gráfico plano, y verificar de nuevo que es cerrado. Eliminar todos los gráficos planos no cerrados, si hay alguno.

Fin del Algoritmo PGG.

En este punto, hemos generado con éxito los gráficos planos y los vectores normales hipotéticos de los planos. Con estos gráficos planos, los atributos de línea continua o discontinua son examinados para verificar la consistencia del modelo alámbrico construido.

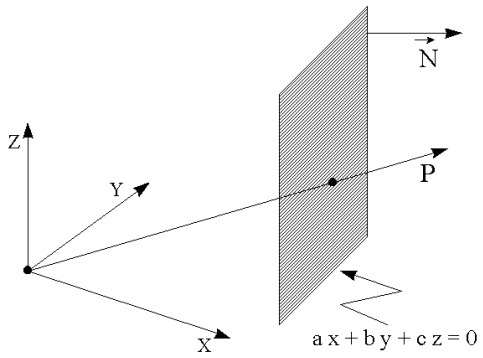


Figura 7: Definición del vector normal exterior hipotético del plano.

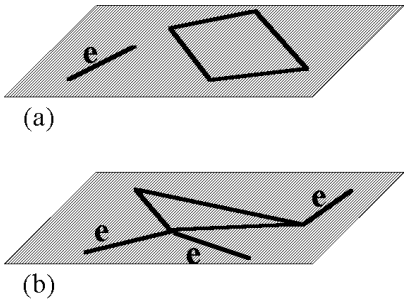


Figura 8: Gráficos planos irregulares; (a) arista aislada; (b) aristas colgantes.

MANEJO DE LÍNEAS ROTAS EN VISTAS PLANAS

Sin los planos y sus vectores normales exteriores, no somos capaces de determinar la relación de orientación entre planos y aristas. Cuando tengamos los planos y los vectores normales, podemos investigar la corrección de las aristas p_edge que son generadas a partir de aristas 2D y, tipos de líneas continuas (o sólidas) y discontinuas (*broken lines*) en las vistas planas. El procedimiento siguiente elimina las aristas 3D no válidas a partir de la información dada de líneas discontinuas a partir de los datos de entrada de las vistas.

Procedimiento: eliminación de aristas no válidas a partir de la información de líneas rotas.

Para cualquier arista p_edge de p_elist, los pasos siguientes son aplicados.
(I1) [Verificar su proyección de vista - alzado]. A partir de los datos de entrada, sabemos el tipo de línea de proyección de cada arista 2D. Si su proyección de alzado es una línea discontinua, habría como mínimo un gráfico plano que no permite ver esta arista p_edge cuando se mira en la dirección de +OY. Si no existe tal gráfico plano, esta p_edge debe ser eliminada; ir al paso I4. Si la arista p_edge es válida, entonces continuar.
(I2) [Verificar su proyección de vista - planta]. Si su proyección de la planta es una línea discontinua, habría como mínimo un gráfico plano que no permite ver esta arista p_edge cuando se mira en la dirección de +OZ. Si no existe tal gráfico plano, esta p_edge debe ser eliminada; ir al paso I4. Si la arista p_edge es válida, entonces continuar.

(I3) [Verificar su proyección de vista - perfil]. Si su proyección de perfil es una línea discontinua, habría como mínimo un gráfico plano que no permite ver esta arista p_edge cuando se mira en la dirección de +OX. Si no existe tal gráfico plano, esta p_edge debe ser eliminada; ir al paso I4.
(I4) Si todas las aristas p_edge han sido eliminadas, entonces ir al paso I5; en otro caso, seleccionar otra arista p_edge, e ir al paso I1.
(I5) La eliminación de aristas no válidas puede crear nuevas aristas y vértices patológicos. Por lo tanto, el procedimiento PEVR es aplicado para depurar nuevas aristas y vértices patológicos.

El procesado de la información de líneas discontinuas es útil antes de la generación de bucles de caras en los pasos siguientes, porque la eliminación de aristas irrelevantes produce una computación más rápida de bucles básicos, bucles de caras, y bucles de cuerpos, debido a una reducción de ambigüedad causada por aristas irrelevantes. Con el procesamiento de información de líneas discontinuas, muchas aristas redundantes son eliminadas. Esto evita mucha más redundancia cuando los bucles de caras y cuerpos son generados, y de este modo la computación es eficiente.

GENERACIÓN DE BUCLES DE CARAS (FACE LOOPS)

En secciones anteriores, hemos recogido información sobre vértices, aristas y gráficos planos. Para determinar los bucles de caras de un objeto, necesitamos generar todas los bucles básicos en cada gráfico plano. Presentamos el algoritmo de generación de bucles de caras (FLG), el cual determina todas los bucles básicos que son usadas para construir todas los bucles de caras para cada gráfico plano. $\Psi=\{L_1,L_2,...,L_k\}$ será un conjunto de bucles básicos en las cuales $L_i \cap L_j$ ={aristas, vértices}, y $L_i \not\subseteq L_j$ o $L_j \not\subseteq L_i$, para todo *i*, *j* . Cualquier bucle $L_k \notin \Psi$ puede ser obtenida de la forma:

$$L_K = \bigcup_{s=1}^{m} L_{K_s}$$

para m ≥ 1, donde $L_{Ks} \in \Psi$. El algoritmo FLG también determina la posición relativa entre dos bucles (por ejemplo, dos bucles están separados o solapados), y se clasifican en bucles interiores/exteriores y generan bucles de caras a partir de bucles básicos. Se da un ejemplo al final del algoritmo.

Algoritmo FLG (face-loop generation, generación de curvas de caras) : Dado un gráfico plano generado con el algoritmo PGG, este algoritmo genera todas los bucles de caras de los gráficos planos.

(F1) [Construir una tabla de vértices - aristas adyacentes para los datos de entrada, los gráficos planos]. Para cada vértice en el gráfico plano, construir una lista de aristas - vértices (*v*, *ne*, *SE*), donde *v* ∈ p_vlist, *ne* es el número de aristas adyacentes a *v* en el gráfico plano. La tabla de aristas - vértices *adj_tab* es una colección de todas las listas de aristas - vértices.
(F2) [Encontrar la arista adyacente ordenada de un vértice].

(F2.1) Seleccionar una entrada (*v*, *ne*, *SE*) de la tabla *adj_tab* .
(F2.2) OE(v) será un conjunto de aristas adyacentes a *v* , ordenado. Las aristas en OE(v) están organizadas en orden ascendente del ángulo formado respecto a una arista seleccionada e_k en el sentido de las agujas del reloj. El ángulo se calcula alrededor del vector normal exterior del plano.
(F2.3) Si todas las entradas tienen que ser procesadas, ir al paso F3; en otro caso, ir al paso F2.1 .

(F3) [Encontrar todas los bucles básicos dentro de un gráfico plano]. Cada arista (v_i , v_j) tiene dos direcciones alternativas: de v_i a v_j , y viceversa. E_C denota una arista actual seleccionada, y V_{start} y V_{end} denotan su vértice inicial y final de una curva básica, respectivamente. Inicialmente, el conjunto de bucles básicos se inicializa a $\Psi \leftarrow \varnothing$, y el conjunto de aristas del bucle básico se inicializa a $L \leftarrow \varnothing$.

(F3.1) Seleccionar una arista e = (v_i , v_j) en el gráfico plano, $E_C \leftarrow$ e. Si todas las aristas han sido seleccionadas en este gráfico plano, entonces ir al paso F4;
en otro caso, $L \leftarrow L \cap E_C$, y continuar.
(F3.2) Seleccionar la arista adyacente de E_C en V_{start} de OE(V_{start}). Suponiendo que $OE(V_{start})=<e_1,...,E_C,e_{c'},...,e_{ne},e_1>$.
Entonces, $e_{c'}=(v_k,v_m)$ es la primera arista adyacente de E_C en V_{start}. Poner $L=L \cup e_{c'}$.
(F3.3) Si E_C ha visitado en las dos direcciones el bucle básico actual L, entonces E_C es un *puente* en el gráfico plano, y es eliminado.
$L=<e_1,...,ej,EC,e_k,...,e_t,E_C>$. Un bucle básico es formado por las aristas $e_k,...,e_t$ en L.
Poner $\Psi \leftarrow \Psi \cup \{ L \}$.

(F3.4) Si $V_{start}=V_{end}$, se ha descubierto un bucle básico; Poner $\Psi \leftarrow \Psi \cup L$, y L=∅, e ir al paso F3.1; en otro caso, ir al paso F3.2 .

(F4) [Identificar la relación de inclusión entres los bucles básicos en un gráfico plano]. Para cada par de bucles L_i y L_j ,si L_i incluye a L_j , entonces incluye (i, j) = 1, si no, incluye (i , j) = 0.

(F5) [Formar los bucles de caras]. Dada la relación de inclusión entre bucles básicos,

Si L_i no incluye algún bucle, entonces L_i forma un bucle de cara;
Sino Si L_i incluye L_j , entonces L_i es un bucle exterior, L_j es un bucle interior y L_i incluye a L_j directamente;
Sino Si el bucle L_i incluye más de un bucle, por ejemplo, L_{j1}, L_{j2}, ... , L_{jn}, verificar cuando los L_j son incluidos como otros bucles básicos.

Un bucle de cara se forma a partir de L_i y bucles básicos incluidos por L_i directamente. En este bucle de cara, L_i es un bucle básico exterior, y los otros son bucles básicos interiores.

Fin del algoritmo FLG

Para cada gráfico plano, usamos el algoritmo FLG para generar los bucles de caras. Como las aristas colgantes y aisladas son eliminadas cuando se encuentran los gráficos planos, cada arista pertenecería a dos o más bucles de caras después de haber construido todos los bucles de caras. Si una arista conecta con menos de dos bucles de caras, esta arista es patológica, y sería eliminada de la lista p_elist. En este caso, el procedimiento PEVR se aplica de nuevo para eliminar esta arista redundante.

Ejemplo : En el ejemplo de la figura 9a, trazamos el algoritmo FLG. Los ocho bucles básicos se muestran en las figuras 9b - 9i. Las direcciones de los bucles las indican las flechas. L_1, L_7 y L_8 son bucles exteriores. No participan en la generación de bucles de caras y son eliminadas. Los bucles básicos interiores son L_2, L_3, L_4, L_5 y L_6. Con L_2 - L_6, todos los bucles de caras del gráfico plano son generados en el paso F4. Los bucles de caras generados se muestran en la figura 10.

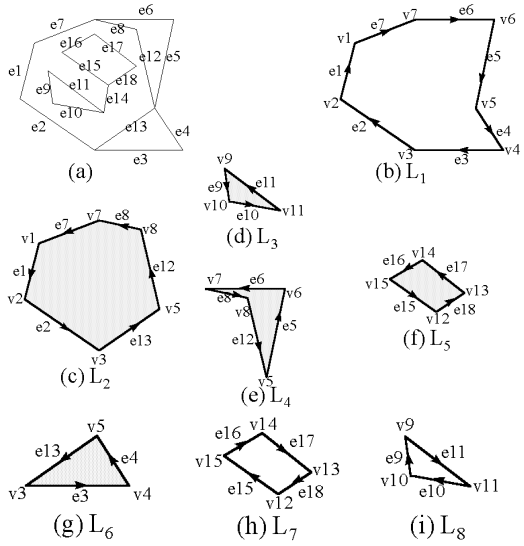


Figura 9: Gráfico plano típico y sus bucles básicos; (a)gráfico, (b) L_1, (c) L_2, (d) L_3, (e) L_4, (f) L_5, (g) L_6, (h) L_7, (i) L_8.

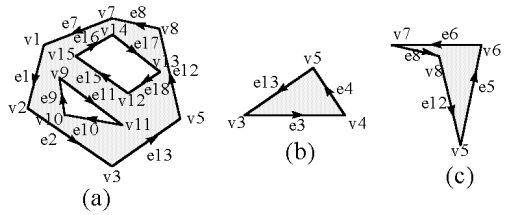


Figura 10: Bucles de caras generados a partir del gráfico plano de la figura 9a; (a) F_1, (b) F_2, (c) F_3.

VÉRTICES Y ARISTAS DE CORTE

En la sección anterior, todos los bucles de caras fueron generados recorriendo los gráficos planos de cada plano. Los bucles de caras de diferentes planos se pueden intersectar, creando aristas y vértices; por eso se llaman aristas y vértices de

corte. Con las aristas de corte y los vértices de corte, se pueden descubrir más bucles de caras de diferentes planos. Dados dos bucles de caras no coplanares, la línea de intersección entre ellos se puede clasificar en los siguientes tres casos.

- Caso 1: La línea de intersección es una arista de un bucle de cara, figura 11a.
- Caso 2: La línea de intersección entre dos bucles de caras, y sus dos puntos finales están en la frontera de los bucles de caras, figura 11b.
- Caso 3: La línea de intersección entre dos bucles de caras, pero no como el caso 2, figura 11c.

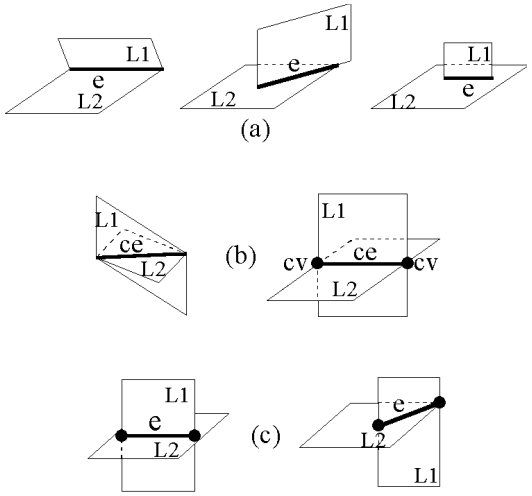


Figura 11: Clasificación de las líneas de intersección entre dos bucles de caras.

La intersección de bucles de caras en el caso 1 es regular. En el caso 2, las aristas / vértices de corte son necesarias, mientras en el caso 3 son patológicas. Como resultado, en el caso 3, el bucle de cara que incluye ambos puntos finales de la línea de intersección serían eliminados.

Después de haber encontrado las aristas y vértices de corte, la relación con cualquiera de los dos bucles de caras es de intersección o están separados. Es necesario mencionar que esas aristas y vértices de corte no existen realmente en el objeto. Se presentan sólo para asistir en la correcta generación de objetos. Además, las aristas y vértices de corte serán eliminadas cuando el objeto final sea construido.

GENERACION DE BUCLES DE CUERPOS (BODY LOOPS)

Los bucles de cuerpos pueden ser vistos como subobjetos, los cuales no tienen puntos interiores comunes. Pueden compartir algunos vértices, aristas o caras. En la base de las tres vistas ortográficas, cada bucle de cuerpo tiene una única localización en el espacio.

Algoritmo BLG (body-loop generation, generación de bucles de cuerpos) : Dados los bucles de caras $F_1, F_2, \ldots, F_m$, el algoritmo BLG descubre todos los bucles de cuerpos.

(B1) [Inicialización]. Un bucle de cara F_i tiene dos lados. El lado del vector normal exterior es el lado positivo, denotado como $+F_i$, mientras que el otro es el negativo, $-F_i$. Marcar cada lado de F_i como 'sin usar' poniendo $\chi(i, +) \leftarrow 0$, $\chi(i, -) \leftarrow 0$, $1 \le i \le m$. Inicializar el conjunto de bucles de caras $S(F) \leftarrow \varnothing$.

(B2) [Seleccionar una cara de inicio y su lado]. Seleccionar un F_j como bucle de cara inicial para empezar la búsqueda de bucles de cuerpos. Si $\chi(j,+)=0$, entonces $S(F) \leftarrow S(F) \cup \{ +F_j \}$ y $\chi(j,+) \leftarrow 1$, y marcar $+F_j$ en S(F) como 'sin expandir'; ir al paso B3. Si $\chi(j,-)=0$, entonces $S(F) \leftarrow S(F) \cup \{-F_j\}$ y $\chi(j,-) \leftarrow 1$, y marcar $-F_j$ en S(F) como 'sin expandir'.

(B3) [Seleccionar un bucle de cara 'sin expandir' $\oplus F_K$ ($\oplus$ puede ser + ó -) de S(F), y computar los sucesivos bucles de caras de cada arista en la frontera de $\oplus F_K$]. Los bucles de caras adyacentes a la arista *e* en F_K son $F_1, F_2, \ldots, F_n$, $n \ge 2$. Los sucesivos bucles de caras de $\oplus F_K$ en la arista *e* es $\oplus F_S$, $1 \le s \le n$ y $s \ne k$, F_S necesita el ángulo de rotación más pequeño para coincidir con $\oplus F_K$. α será el ángulo de rotación entre $\oplus F_K$ y $\oplus F_S$. n_K y n_S serán los hipotéticos vectores normales de F_K y F_S , respectivamente. θ será el ángulo entre n_K y n_S . Entonces, tenemos

$$\theta = \cos^{-1}\left[\frac{n_S \cdot n_K}{|n_S| \cdot |n_K|}\right]$$

El vector unitario **e** será un vector a lo largo de la arista *e*, y satisface la regla de la mano derecha con n_K. Si $\oplus F_K$ es $+F_K$ su bucle de cara sucesivo a la arista *e* puede ser computado con el siguiente criterio. Obtenemos los valores de α y de selección del lado $\oplus$ para cada bucle de cara adyacente en la arista *e* de modo que elegimos el bucle de cara con el menor valor de α.

• Si $n_S \times n_K$ está en la misma dirección que **e**, y $+F_S$ está a la derecha de **e**, entonces $\alpha = \pi - \theta$; asignar $+F_S$ a $\oplus F_S$ (ver figura 12a).
• Si $n_S \times n_K$ está en la dirección contraria a **e**, y $+F_S$ está a la izquierda de **e**, entonces $\alpha = \theta$; asignar $-F_S$ a $\oplus F_S$ (ver figura 12b).
• Si $n_S \times n_K$ está en la misma dirección que **e**, y $+F_S$ está a la izquierda de **e**, entonces $\alpha = 2\pi - \theta$; asignar $-F_S$ a $\oplus F_S$ (ver figura 12c).
• Si $n_S \times n_K$ está en la dirección contraria a **e**, y $+F_S$ está a la derecha de **e**, entonces $\alpha = \pi + \theta$; asignar $+F_S$ a $\oplus F_S$ (ver figura 12d).

Si $+F_S \in S(F)$, entonces $S(F) \leftarrow S(F) \cup \{ +F_S \}$ y $\chi(s,+) \leftarrow 1$. Marcar $+F_S$ como 'sin expandir' en S(F). Este procedimiento de selección del lado garantiza que todos los bucles de caras en S(F) que forman las caras interiores y exteriores del bucle serán formados. Cuando el bucle de cara sucesivo de F_K en cada arista sobre F_K ha sido determinado usando el criterio anterior, marcar $+F_K$ como 'expandido'. Ir al paso B4. Similarmente, si $\oplus F_K$ es $-F_K$, su sucesivo bucle de cara en la arista *e* puede ser determinado de acuerdo con esto; ir al paso B4. Como un ejemplo, los vectores normales de los bucles de caras adyacentes F_1, F_5 en la figura 13a se muestran en la figura 13b. Las equivalencias de bucles de caras sucesivos son $+F_1$ y $-F_2$, $-F_1$ y $-F_5$, $+F_4$ y $-F_3$, y $-F_4$ y $+F_5$.

(B4) Repetir el paso B3 hasta que no haya nuevos bucles de caras para ser expandidos en S(F). Los bucles de caras en S(F) pueden formar un bucle de cuerpo. Ir al paso B5.

(B5) [Verificar la corrección del bucle de cuerpo]. Un bucle de cuerpo debe estar cerrado por los bucles de caras, sin caras colgantes. Así para el bucle de cuerpo actual en S(F), si cada arista de este bucle de cuerpo es compartida por dos y sólo dos bucles de caras, entonces los bucles de caras en el actual S(F) forman un bucle de cuerpo válido. En otro caso, los bucles de caras en S(F) no pueden formar un cuerpo cerrado; ir al paso B6 para iniciar otra búsqueda.

(B6) Poner $S(F) \leftarrow \varnothing$, y repetir los pasos B1-B5 hasta que cada bucle de cara F_i, $1 \le i \le m$, haya sido visitado en sus dos lados.

Fin del algoritmo BLG

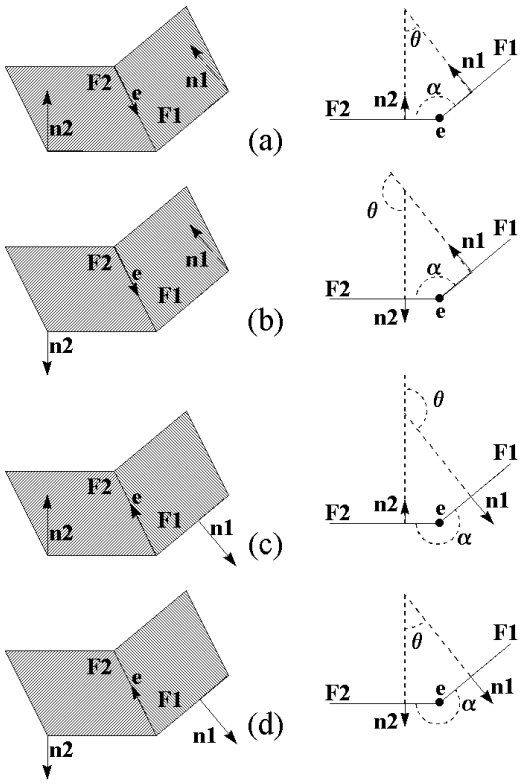


Figura 12: Descubrimientos de bucles de caras sucesivos.

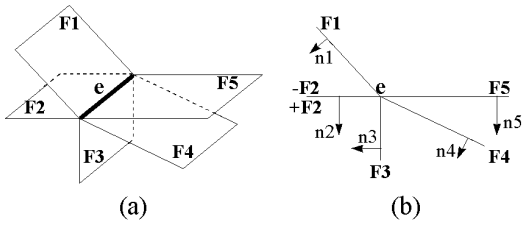


Figura 13: Ejemplos de bucles de caras; (a) caras adyacentes a la arista *e*, (b) proyección a lo largo de la arista *e*.

El algoritmo BLG genera todos los bucles de cuerpos a partir de los bucles de caras. Algunos de los bucles de cuerpos generados son ilimitados, y se llaman bucles de cuerpos exteriores. Los otros son interiores, porque son finitos. Los bucles de cuerpos exteriores son inútiles para construir objetos, y por eso se descartan.

Para clasificar los bucles de cuerpos interiores y exteriores, la información de selección de lado (por ejemplo el valor de ⊕) de los bucles de caras en la generación de los bucles de cuerpos es importante. En un bucle de cuerpo, los lados de todos los bucles de caras seleccionados usando el algoritmo anterior forman las caras interiores del bucle de cuerpo. Como resultado, el interior de un bucle de cuerpo exterior es ilimitado, y el exterior es un 'sólido' vacío finito. Para clasificarlos bucles de caras en interiores y exteriores, utilizamos los siguientes pasos:

- Para un bucle de cuerpo, seleccionar dos bucles de caras adyacentes $\oplus F_i$ y $\oplus F_j$ a su arista compartida *e*, y construir un rayo L que empiece en el punto medio de *e* y en la dirección de

$$L = \frac{\oplus\, n_1}{|n_1|} + \frac{\oplus\, n_2}{|n_2|}$$

  donde $\oplus\, n_1$ y $\oplus\, n_2$ son los vectores normales hipotéticos de $\oplus F_i$ y $\oplus F_j$, respectivamente, y ⊕ corresponde a su selección de lado.
- Analizar las situaciones en las cuales la línea L intersecta con el bucle de cara del bucle de cuerpo. Si el número de puntos de intersección formados por L y todos los bucles de caras en un bucle de cuerpo (excluyendo F_i y F_j) es uno o más, entonces el bucle de cuerpo es interior; en otro caso, es exterior.

Usamos los vectores normales hipotéticos de los bucles de caras. Para todo bucle de cara F de un bucle de cuerpo, si F es seleccionado por el lado negativo, por ejemplo -F, entonces su hipotético vector normal exterior +n apunta al exterior del bucle de cuerpo, mientras -n apunta al interior. Similarmente, si F es seleccionado por el lado positivo, por ejemplo +F, entonces su hipotético vector normal exterior +n apunta al interior del bucle de cuerpo, mientras -n apunta al exterior. Esto implica que el auténtico vector normal exterior de un bucle de cara -F de un cuerpo es +n, o el auténtico vector normal exterior de un bucle de cara +F en el bucle de cuerpo es -n. Por lo tanto, podemos obtener los auténticos vectores normales exteriores para todas las caras de cualquier cuerpo. La información del auténtico vector normal exterior puede ser usada para construir la representación de fronteras (*boundary representations, B-rep)* completa de los objetos.

COMBINACIÓN DE BUCLES DE CUERPOS Y TOMA DE DECISIONES

Para los bucles de cuerpos generados antes, comprobamos todas las posibles combinaciones de esos bucles de cuerpos para garantizar que podemos descubrir todos los objetos que encajan con las tres vistas originales de los datos de entrada. Si las proyecciones del objeto candidato son consistentes con sus correspondientes tres vistas, el objeto candidato es una solución de las vistas. Dados N bucles de cuerpos, el número total de candidatos es 2^N-1. Verificamos la corrección de cada objeto candidato, y comparamos sus proyecciones con las vistas de los datos de entrada, por consistencia.

CORRECCIÓN DEL OBJETO CANDIDATO

Desde que un bucle de cuerpo tiene una única orientación en el espacio, la relación entre cualquiera dos bucles de cuerpos puede ser clasificada en los cuatro casos siguientes:

- Caso 1: Dos bucles de cuerpos separados, figura 14a.
- Caso 2: Dos cuerpos compartiendo algunos vértices, figura 14b.
- Caso 3: Dos cuerpos compartiendo algunas aristas, figura 14c.
- Caso 4: Dos cuerpos compartiendo algunas caras, figura 14d.

Para cada objeto candidato, verificamos su corrección en función a las siguientes reglas.

(1) [Eliminar bucles de caras redundantes]. Eliminar cada bucle de cara para el cual ambos lados fueron seleccionados en el mismo sólido, desde que sabemos que esos bucles de caras son las caras interiores del sólido, y no la superficie del sólido. La cara F_2 en la figura 15a, por ejemplo, es redundante.

(2) [Eliminar aristas redundantes]. Eliminar esas aristas que son compartidas por dos y sólo dos caras coplanares. Después de eliminarlas, unir las dos caras en una, formando un nuevo bucle de cara del sólido (figura 15b).

(3) [Descartar el candidato incorrecto usando la información de los vértices]. Si dos caras sólo comparten algunos vértices, este objeto candidato no es regular, y es incorrecto.
(4) [Descartar el candidato incorrecto usando la información de las aristas]. Puesto que cada arista pertenece a dos y sólo dos caras del objeto sólido, si una arista tiene más de dos caras adyacentes, y la conexión de una arista es más de dos, este objeto es también no regular.

Cada candidato correcto se examinará más para la consistencia de sus proyecciones con las vistas.

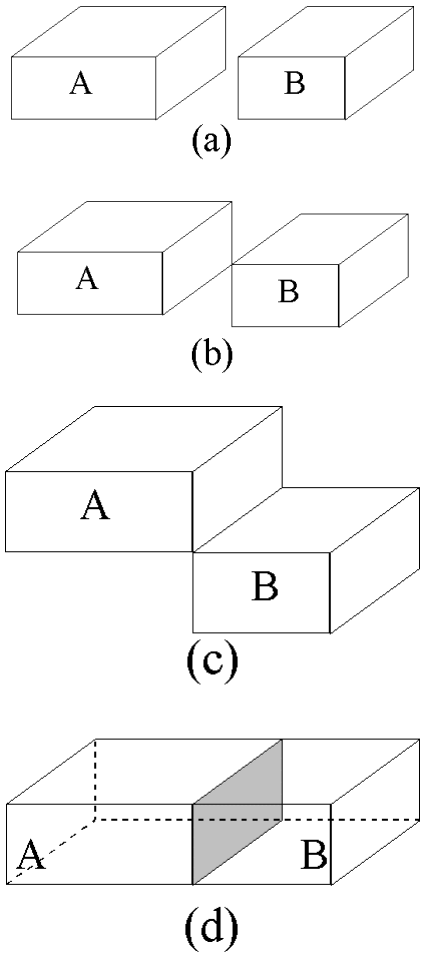
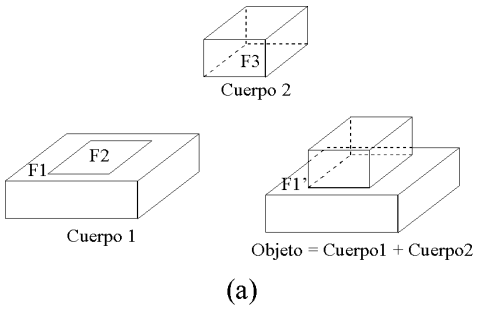


Figura 14: Relación entre dos objetos en el espacio.



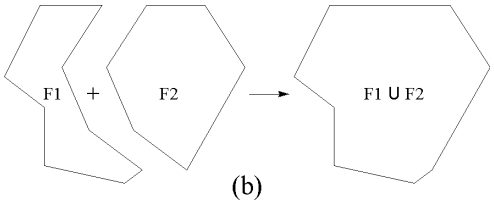


Figura 15: Algunas situaciones especiales para la construcción de objetos; (a) eliminación de caras duplicadas, (b) eliminación de aristas duplicadas.

CONSISTENCIA DEL OBJETO CANDIDATO CON LAS TRES VISTAS DE LOS DATOS DE ENTRADA

Primero, un objeto candidato se proyecta a los correspondientes planos de proyección (*xOz*, *xOy*, *yOz*), y se forman tres nuevas vistas. Segundo, comparar las tres nuevas vistas ortográficas con las vistas originales. Si son iguales completamente, este objeto es una solución.

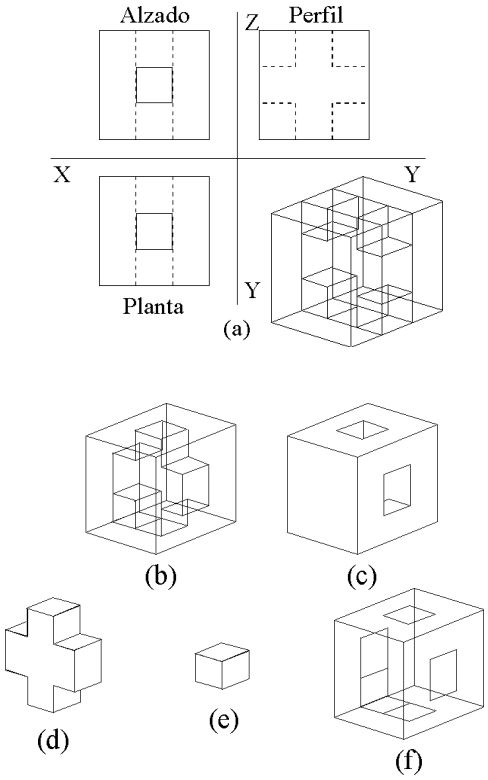
Durante la proyección, una arista 3D puede llegar a ser un vértice 2D o un segmento de línea 2D. Necesitamos considerar los siguientes pasos. Si una arista 3D no está bloqueada por algún plano, la proyección de esta arista 3D es un segmento sólido; en otro caso, es un segmento de línea discontinua. Si un segmento de línea discontinua solapa con un segmento de línea sólida, las secciones solapadas podrían ser sólidas. Por lo tanto, las aristas 3D pueden ser proyectadas en una combinación de varios segmentos de líneas sólidas y varios segmentos de líneas discontinuas.

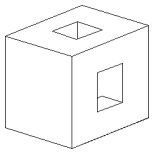
Con los segmentos de líneas 2D de la proyección, unimos los segmentos de líneas adyacentes del mismo tipo (sólidas o discontinuas). Comparar estos segmentos de líneas con los de los datos de entrada. Si los segmentos de líneas se solapan completamente y son iguales, el objeto construido es una solución correcta. En otro caso, no hay solución. En la sección siguiente, damos unos ejemplos para demostrar la corrección y funcionamiento de los algoritmos.

EJEMPLOS Y ANÁLISIS

El algoritmo ha sido implementado en C para una estación de trabajo SUN 3/60. Cada paso del procedimiento de construcción de modelos sólidos se muestra en tiempo real. Cada ejemplo realizado por el algoritmo consiste en tres vistas ortográficas del objeto y el correspondiente modelo alámbrico 3D.

La figura 16 muestra un ejemplo en el cual se demuestra el procesado de la información de líneas rotas. La figura 16a muestra el modelo alámbrico antes de procesar la información de las líneas rotas, mientras la figura 16b muestra el modelo alámbrico después del procesamiento. Las figuras 16c-f muestran los bucles de cuerpos del modelo alámbrico. El objeto correcto se muestra en la figura 16g. La eficiencia de la computación sufre un gran incremento por la reducción de aristas redundantes.





(g)

Figura 16: Ejemplo de procesamiento de líneas discontinuas.

La figura 17 muestra otro ejemplo en el cual es necesario descubrir aristas/vértices de corte. Con este descubrimiento, las vistas planas pueden ser interpretadas correctamente para garantizar que los objetos correctos son reconstruidos. La figura 17a muestra las tres vistas y el modelo alámbrico generado por el algoritmo WC. La figura 17b muestra las aristas de corte descubiertas y el vértice de corte introducido. Los bucles de cuerpos para este modelo alámbrico se listan en las figuras 17d-m. En la base de diferentes interpretaciones de líneas discontinuas, tiene tres soluciones, como se muestra en las figuras 17n-p. Sin considerar la información de líneas rotas, estas vistas tendrían cuatro modelos sólidos correspondientes.

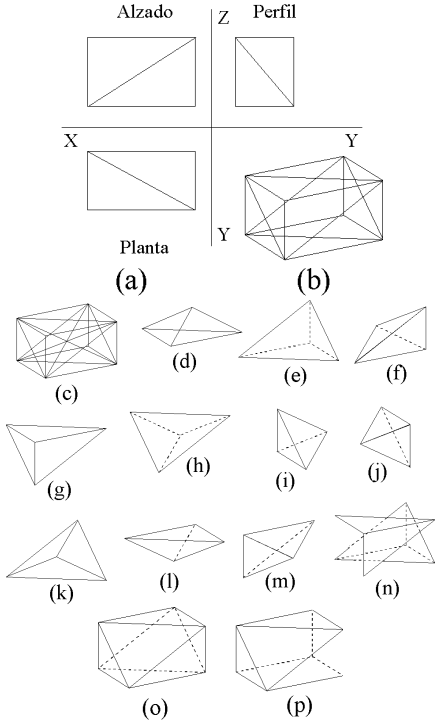


Figura 17: Ejemplo de objetos correctos múltiples; (a)tres vistas, (b) modelo alámbrico, (c) vértices y aristas de corte, (d) bucle de cuerpo exterior, (e) (m) bucles de cuerpo interiores, (n) (p) tres soluciones correctas.

A partir de los ejemplos anteriores, determinamos los siguientes hechos:

- La mayoría de los datos de entrada, por ejemplo los modelos sólidos más complejos, necesitan un mayor tiempo de computación para construir el sólido.
- No puede haber siempre soluciones múltiples para vistas planas altamente simétricas. La figura 17 tiene múltiples soluciones, mientras la figura 16 no.
- La eficiencia de procesamiento se incrementa cuando consideramos la información de líneas rotas en el algoritmo. En la figura 16, si se ignora la información de líneas rotas, hay seis posibles bucles de cuerpos. Después de procesar la información de líneas rotas, hay dos bucles de cuerpos. El número de objetos candidatos decrece desde 2^6-1=63 hasta 2^2-1=3. Por lo tanto, la eficiencia de la computación para identificar los objetos correctos mejora exponencialmente.

CONCLUSIONES

El número de investigaciones de reconstrucción de información 3D, a partir de la información 2D, se ha incrementado rápidamente. Se han realizado continuos logros en la recuperación de modelos sólidos a partir de sus vistas planas, pero los métodos de los algoritmos descritos en la literatura no son, en mayor o menor grado, completos. En este artículo, se propone un algoritmo que maneja la reconstrucción de modelos sólidos poliédricos 3D a partir de sus vistas planas. El algoritmo no puede encontrar solo la solución correcta o múltiples soluciones, pero además elimina casos patológicos, si hay alguno. El algoritmo está garantizado para encontrar la solución correcta a partir de unos datos de entrada, con tal que las vistas sean correctas. El algoritmo toma toda la ventaja de la información de líneas discontinuas para eliminar los casos patológicos. Nuestro algoritmo

utiliza los hipotéticos vectores normales exteriores para clasificar los bucles de caras y cuerpos, y asistir en la construcción de bucles de cuerpos. Eventualmente, ajustando los vectores normales exteriores hipotéticos para cada bucle de cara del objeto, se obtiene el auténtico vector normal para la construcción de la representación de fronteras completa del objeto.

REFERENCIAS

1 Idesawa, M. "A system to generate a solid figure from a three view" Bull. Jap. Soc. Mech. Eng. Vol 16 (Feb 1973) pp 216-225

2 Aldefeld, B. "On automatic recognition of 3D structures from 2D representations" Computer-Aided Design Vol 15 Nº 2 (1983) pp 59-64

3 Aldefeld, B. and Ricjter, H. "Semiautomatic three dimensional interpretation of line drawings" Comput. & Graph. Vol 8 Nº 4 (1984) pp 371-380

4 Markowsky, G. and Wesley, M. "Fleshing out projections" IBM J. Res. & Develop., Vol 25 Nº 6 (1981) pp 934-954

5 Sakurai, H. and Gossard, D. "Solid model input through orthographic views" Comput. & Graph. Vol 17 Nº 3 (1983) pp 243-252 (Proc ACM/SIGGRAPH Conf.)

6 Gujar, U. and Nagendra, I. "Construction of 3D solid objects from orthographic views" Comput. & Graph. Vol 13 Nº 4 (1989) pp 505-521

7 Gu, K.; Tang, Z. and Sun, J. "Reconstruction of 3D solid objects from orthographic projections" Comput. Graph. Forum Vol 10 Nº 5 (1986) pp 317-324

APÉNDICE

Conceptos de geometría.

Este apéndice da definiciones de la geometría del objeto, cara, vértice, arista, etc. En este artículo, todos los objetos se refieren a poliedros, si no son explícitamente especificados.

Objeto (object)

El objeto O es una región bordeada no vacía de $\Re^3$ denotada como O={δO, iO}, donde δO representa la frontera o superficie de O, y iO es el interior de O. El espacio complementario cO de O se llama exterior de O. Propiedades de δO:

- iO y cO son subespacios disjuntos separados por δO.
- Eliminando un punto de δO hace que iO y cO estén conectados.
- Para cualquier punto p∈ δO, si hay una tangente plana a p, entonces el vector normal de O en el punto p, apunta en la dirección del subespacio cO.

Cara (face)

La cara F de un objeto es un plano no vacío y rodeado denotado como F={δF, iF}, donde δF es la circunferencia de F, y iF es el interior de F. En nuestro dominio, δF está compuesto de segmentos de líneas rectas y δO se hace con un número limitado de caras.

Conjunto de vértices (vertex set)

El conjunto de vértices V(F) de la cara F es una colección de todos los puntos de intersección de cualquiera dos segmentos de líneas. El conjunto de vértices V(O) del objeto O es una colección de todos los puntos de intersección de cualquiera tres caras no coplanares.

Conjunto de aristas (edge set)

El conjunto de aristas E(F) de la cara F es la circunferencia de la cara F, o δF. Cada punto final pertenece a V(F), y cualquier punto interior de la arista no pertenece a V(F). El conjunto de aristas de E(O) del objeto O es la colección de las circunferencias de todas la caras en δO. Cada arista en E(O) satisface lo siguiente:

- Ambos puntos finales están en V(O).
- Cualquier punto interior de una arista no está en V(O).
- Para cualquier punto p en una arista, existen siempre dos caras $F_1 \in \delta O$ y $F_2 \in \delta O$ tal que $p \in \delta F_1 \cup \delta F_2$.

Modelo alámbrico (wireframe)

El modelo alámbrico WF(O) del objeto O se define como un conjunto de todos los pares ordenados [V(O),E(O)].

Bucle (loop)

El bucle L es una colección de aristas coplanares $L=\{e_1,e_2, \ldots ,e_k\}$, donde e_i es una arista. Cada arista en L satisface:

- El punto de intersección de cualquiera dos aristas e_i y e_j debe ser V(F).
- El punto final de cada arista puede estar conectado a dos y sólo dos aristas.
- Los bucles son todos disjuntos.

Los bucles se clasifican en interiores y exteriores. Si el área está rodeada, el bucle es interior, en otro caso, es exterior.

Bucle de cara (face loop)

Una cara puede estar representada en términos de bucles. L_0, L_1, ... , L_k serán los bucles sobre la cara F, donde L_0 es un bucle interior que incluye L_1, ... , L_k directamente, y no hay solapamiento entre bucles por ejemplo, $L_i \cap L_j = \varnothing$ el bucle de cara F se define:

- La frontera de F está determinada por

$$\partial F = \bigcup_{i=0}^{k} L_i$$

- F consta de todos los puntos dentro de L_0 y fuera de $\bigcup_{i=1}^{k} L_i$ y los puntos en $\bigcup_{j=0}^{k} L_j$.

Bucle de cuerpo (body loop)

Un bucle de cuerpo es una colección de bucles de caras. El bucle de cuerpo B={F_1, F_2, ... , F_m}, donde F_i es un bucle de cara tal como

- Para cualquier F_i , F_j, $i \neq j$, sus líneas de intersección caen dentro de $\partial F_i \cap \partial F_j$, si intersectan.
- Para cualquier arista $e \in E(F)$, existen dos y sólo dos bucles de caras F_i y $F_j \in B$, $i \neq j$, tal que $e \in \partial F_i \cap \partial F_j$, el cual asegura la clausura del bucle de cuerpo.

Similarmente, hay bucles de cuerpos interiores y exteriores en términos de volumen que los bucles de cuerpos contienen. Un bucle de cuerpo rodeado será interior, y si no está rodeado será exterior. Usando la definición de bucle de cuerpo B, un objeto se puede definir como sigue.

Objeto (object)

Un objeto es una colección de un número limitado de bucles de cuerpos interiores. O={B_1, B_2, ... , B_m} donde B_i, i=1, 2, ... , m, son bucles de cuerpos interiores, tal que:

- $\partial O = \bigcup_{i=1}^{m} \partial B_i$ donde **U** es una operación de conjunto regularizada.
- El interior de O incluye todos los puntos de $\bigcup_{i=1}^{m} B_i$.
- No hay solapamiento entre cualquiera dos bucles de cuerpos B_i y B_j, $i \neq j$.

Aristas y vértices de corte (cutting vertices/edges)

Después de la construcción desde los datos de la proyección 2D hasta los vértices y aristas 3D, esos vértices y aristas pueden no formar un modelo alámbrico 3D correcto, y los siguientes casos patológicos pueden existir:

- Caso 1: El punto de intersección de dos aristas 3D no es el punto final de las aristas.
- Caso 2: La línea de intersección de dos planos que son generados a partir de los vértices y aristas del modelo alámbrico está en la frontera de los planos, pero la línea de intersección no es una arista del modelo alámbrico.

Para remediar el caso patológico 1, añadimos el punto de intersección en el conjunto de vértices del modelo alámbrico, este vértice se llama vértice de corte. Para remediar el caso patológico 2, introducimos la línea de intersección en el conjunto de aristas del modelo alámbrico, esta arista se llama arista de corte. Con estos vértices y aristas de corte, el modelo alámbrico es correcto en el concepto del sentido geométrico.

Apéndice B: Formato GRA

En este apéndice se va describir el formato de los ficheros GRA, que son unos ficheros de texto con la extensión GRA en los cuales tenemos definidos los datos de entrada al algoritmo de reconstrucción, es decir, en estos ficheros definimos los datos de las proyecciones de planta, alzado y perfil. Después de describir la estructura, veremos un ejemplo correspondiente a algún dibujo del capítulo 4.

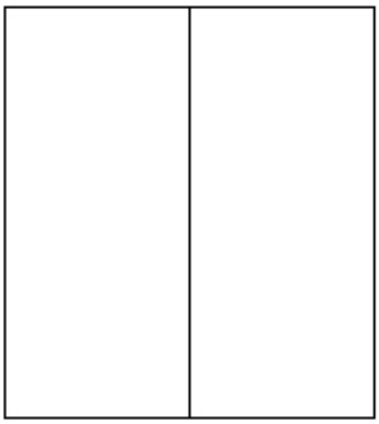
La estructura de un fichero GRA será la siguiente: tenemos unas cadenas de caracteres que actúan como delimitadores de distintas zonas, que serán las cadenas *[Planta]*, *[Alzado]*, *[Perfil]* y *[Fin]*. Estos marcadores siempre van en el mismo orden con el que se han expuesto. Dentro de cada sección, excepto *[Fin]* que indica el final de los datos, tienen dos subsecciones que son: *[Vertices]* y *[Aristas]*. La sección *[Vertices]* nos describe los vértices 2D (sólo tienen dos coordenadas) y la sección *[Aristas]* nos describe las aristas 2D, mediante dos índices a vértices de la sección anterior. La sección *[Vertices]* viene definida mediante los vértices según el siguiente formato: (c_1,c_2), (c_3,c_4), (c_5,c_6), ... , (c_n,c_{n+1}). acabando en un punto; y la sección *[Aristas]* viene definida con el formato: (i_1,i_2), (i_3,i_4)r, (i_5,i_6), ... , (i_m,i_{m+1}). acabando en un punto, donde los valores c_1, c_2, ... , c_{n+1} son los valores de tipo de datos real que definen las coordenadas 2D y los valores i_1, i_2, ... , i_{m+1} son los valores de tipo de datos entero que definen los índices de los vértices 2D que se utilizan para formar aristas. Si alguna arista 2D tiene a continuación el carácter "*r*", como por ejemplo (i_3,i_4)r, esto nos indica que

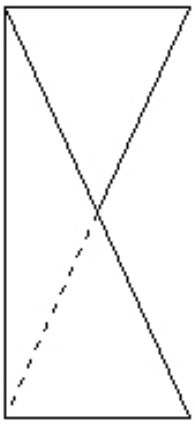
esta arista en esta proyección es una línea discontinua, si no lleva el carácter entonces será continua. Veamos ahora un esquema del formato.

```
[PLANTA]
[VERTICES]
(CP1,CP2),(CP3,CP4), ... ,(CPvp,CPvp+1).
[ARISTAS]
(IP1,IP2),(IP3,IP4)r, ... ,(IPap,IPap+1).
[ALZADO]
[VERTICES]
(CA1,CA2),(CA3,CA4), ... ,(CAva,CAva+1).
[ARISTAS]
(IA1,IA2)r,(IA3,IA4), ... ,(IAaa,IAaa+1).
[PERFIL]
[VERTICES]
(CPE1,CPE2),(CPE3,CPE4), ... ,(CPEvpe,CPEvpe+1).
[ARISTAS]
(IPE1,IPE2),(IPE3,IPE4), ... ,(IPEape,IPEape+1)r.
[FIN]
```

Como se puede ver se ha definido un formato de almacenamiento de los datos de entrada muy sencillo, con el objetivo que sea fácil al usuario el introducir los datos, aunque costoso en el sentido que hay que obtener las coordenadas de todos los vértices y definir las aristas. Con el propósito de mejorar se puede añadir, en futuros trabajos, el reconocimiento de los datos de entrada a partir de ficheros gráficos estándar, como por ejemplo en formato DXF, IGES, ...

A continuación vamos a pasar a ver cómo estaría definido el ejemplo 1 del capítulo 4 en el formato GRA. En la figura siguiente podemos ver la planta, alzado y perfil del ejemplo 1 para, a continuación, mostrar su descripción en formato GRA.





```
[PLANTA]
[VERTICES]
(7,2),(4,2),(1,2),(7,5),(4,5),(1,5),(7,3.5),(4,3.5),(1,3.5).
[ARISTAS]
(1,4),(2,5),(3,6),(1,3),(4,6).
[ALZADO]
[VERTICES]
(7,12),(4,12),(1,12),(7,2),(4,2),(1,2),(7,7),(4,7),(1,7).
[ARISTAS]
(1,3),(3,6),(6,4),(4,1),(2,5).
[PERFIL]
[VERTICES]
(2,12),(5,12),(2,2),(5,2),(3.5,7).
[ARISTAS]
(1,2),(1,3),(3,4),(4,1),(2,5),(5,3)r.
[FIN]
```

BIBLIOGRAFIA

- Qing-Wen Yan, C. L. Philip Chen, Zesheng Tang: “Efficient algorithm for the reconstruction of 3D objects from orthographics projections”. Computer-Aided Design. Volume 26. Number 9. September 1994.

- Foley et al.: “Principles of interactive computer graphics”. Addison-Wesley, 1982.

- Foley et al.: “Computer graphics”. Addison-Wesley, 1990.

- Foley, Van Dam, Feiner, Hughes, Phillips: “Introducción a la graficación por computador”. Addison-Wesley, 1996.

- José María Gomis Martí: “Dibujo técnico I”. SPUPV-90.439.

- José García Resta: “Reconstrucción 3D de objetos poliédricos a partir de su representación axonométrica oblicua 2D”. Proyecto Fin de Carrera, 1997.

- Charles Petzold: “Programación en Windows™”. Ediciones Anaya Multimedia.

- P. Pérez Carreras, A. Pérez Machado: “Matemáticas para COU”. McGraw-Hill, 1988.

- Carlos Mataix Aracil: “Tratado de geometría analítica”. Editorial Dossat, S. A.

- Charles H. Lehmann: “Geometría analítica”. Ed. Limusa, 1982.

- Alfred V. Aho, John E. Hopcroft: “Data structures and algorithms”. Addison-Wesley, 1983.

- Brian W. Kernighan, Dennis M. Ritchie: “The C programming language”

- I.V. Nagendra and U.G. Gujar, “3-D objects from 2-D orthographic views - A survey”. *Computers & Graphics,* vol. 12, No. 1, pp. 11 *1-1* 14, (1988).

- M.A. Wesley and G. Markowsky, “Generation of solid models from two-diniensional and three-dimensional data”. *Solid Modeling by, Computer: from Theory.to Application,* pp. 23-51, (1986).

- Weidong Wang, “On the automatic reconstruction problem of a 3D object's Constructive Solid Geometry representation from its 2D projection”. A *Thesis Proposal for D.Sc. Submitted to the Graduate Committee of Computer Science Dept.., University of Lowell,* (May 1988).

- Aldefeld, “On automatic recognition of 3D structures from 2D representations”. *Computer Aided Design, vol.* 15 No. 2, pp. 59-64, (Mar 1983).

- Masanori Idesawa: “A system to generate a solid figure from three views”. *Bull.* JSME**,** vol. 16, pp.216-225, (Feb 1973).

- M.A. Wesley and G. Markowsky: “Fleshing out projections”. *IBM Journal of Research and Development,* vol. 25 No. 6, (Nov 1981).

- H. Sakurai: “Solid model input through orthographic views”. *Computer Aided Design vol.* 17 No.3, (July 1983).

- Kaining Gu, Zesheng Tang, and Jiaguang Sun: “Reconstruction of 3D objects from orthographic projections”. *COMPINT 85' Proceedings,* pp. 807-811, (1985).

- Uday Gujar and I.V. Nagendra: “Construction of 3D solid objects from orthographic views”. *Comput. & Graphics,* vol. 13 No. 4, pp. 505-521, (1989).

- Weidong Wang and Georges Grinstein: “A polyhedral object's CSG-Rep reconstruction from a single 2D line drawing”. *Proc. of 1989 SPIE Intelligent Robots and Computer Vision III.- AIgorithms and Techniques,* vol. 1192, pp. 230-238, (Nov 1989).

- Weidong Wang: “Regular curved object's CSG-rep reconstruction from a single 2D line drawing”. *Proc. of 1991 SPIE Intelligent Robots and Computer Vision. X: Neural, Biological and 3-D Methods,* vol. 1608, Boston, USA, (Nov 1991).

- Weidong Wang: “On the automatic reconstruction problem of a 3D object's Constructive Solid Geometry representation from its 2D projection," A *Doctor of Science dissertation, Computer Science Dept., University of Massachusetts* at *Lowell,* (Mar 1992).