DEFINICIÓN Y VISUALIZACIÓN DE SUPERFICIES RECORTADAS

Molina Carmona, R.; Puchol García, J.A. y Salas Pérez, F. (*)
Dpto. de Tecnología Informática y Computación
Universidad de Alicante
Apdo. Correos 99, 03080-Alicante
Tfn (96) 590 36 81 Fax (96) 590 36 81

Dpto. de Informática
Instituto Español del Calzado y Conexas - INESCOP
Apdo. Correos 253, 03600-Elda (Alicante)
Tfn (96) 539 52 13 Fax (96) 538 10 45

RESUMEN

Las superficies recortadas presentan algunas dificultades de diseño y visualización. El primer problema es consecuencia de realizar la definición de las curvas de recorte en un espacio (el paramétrico) diferente del de diseño ($\Re^3$). Para solucionarlo, proponemos un método de obtención de la curva de recorte a partir de curvas tridimensionales, presentando el caso general y un caso particular más simple. Por último se trata la visualización realista de la superficie. Planteamos, para ello, un algoritmo de poligonalización sobre el espacio paramétrico. Estas técnicas se han empleado para el diseño de pisos de calzado, proyecto conjunto de la Universidad de Alicante e INESCOP.

1. INTRODUCCIÓN

Las superficies recortadas constituyen una potente herramienta en los sistemas CAD/CAM actuales. Permiten la generación de superficies paramétricas libres y su recorte mediante curvas cerradas definidas en el espacio de los parámetros [1, 2]. Poseen grandes ventajas, tanto para el programador como para el usuario. La principal es la posibilidad de definir superficies libres, sin restricciones en la forma de la frontera de la superficie o en la topología del poliedro de control.

No obstante, también tienen algunos inconvenientes. Para el usuario, el diseño de la superficie puede resultar poco intuitivo debido a que la curva de recorte ha de definirse en el espacio de los parámetros, y no en el espacio de representación 3D sobre el que el usuario trabaja. Para el programador, por otro lado, la visualización en modo realista de la imagen se complica: las librerías gráficas disponibles están pensadas para polígonos, por lo que debe optarse entre la transformación a un modelo poligonal (un paso no tan sencillo como para una superficie sin recortar) y el desarrollo de algoritmos propios adaptados a este tipo de superficies.

El presente trabajo propone soluciones a cada uno de estos problemas. Para el primero se sugiere el diseño de curvas tridimensionales situadas sobre la superficie, calculadas como intersección de superficies o mediante proyección de curvas planas, que luego son transformadas al espacio de los parámetros; para el segundo la poligonalización de la superficie resultante sobre el espacio de los parámetros, tratando de aprovechar las ventajas que ofrece un espacio bidimensional y acotado.

Los resultados que aquí se presentan se enmarcan dentro de un proyecto más amplio, que intenta dar soluciones a la problemática del diseño y fabricación por ordenador de pisos y moldes para calzado, proyecto realizado conjuntamente por el Instituto Español del Calzado y Conexas (INESCOP) y el Departamento de Tecnología Informática y Computación de la Universidad de Alicante. Los ejemplos que se ofrecen corresponden a desarrollos de este proyecto, que se encuentran ya funcionando.

La organización del documento es la siguiente: En el apartado 2, se introduce la problemática del usuario a la hora de diseñar la curva de recorte y las dos soluciones propuestas. El desarrollo de la técnica de visualización y su solución se trata en el punto 3. El apartado cuarto presenta los resultados y algunos ejemplos, para dar paso a las conclusiones en el punto 5.

2. DISEÑO DE SUPERFICIES RECORTADAS

Veamos en primer lugar qué entendemos por superficie recortada: Sea S(u, v) una superficie en forma paramétrica, con parámetros (u, v) definidos sobre un espacio $\Re^2$ (en lo sucesivo, espacio paramétrico), y cuya imagen (x, y, z) se encuentra definida en un espacio $\Re^3$, que llamaremos espacio de representación. Sea C(t) una curva paramétrica cerrada, definida sobre un espacio paramétrico $\Re$, cuya imagen se encuentra definida en un espacio $\Re^2$, que coincide con el espacio paramétrico de S. Definimos la superficie S recortada con la curva C, S_C (t, u, v), como la imagen en el espacio de representación (x, y, z) de los puntos (u, v) del espacio paramétrico que se encuentran contenidos en la curva cerrada C(t) [1, 2]. La curva correspondiente a C(t) en el espacio de representación, forma un curva tridimensional que resulta ser la frontera de la superficie recortada, y que llamaremos curva L(t) (Fig.1).

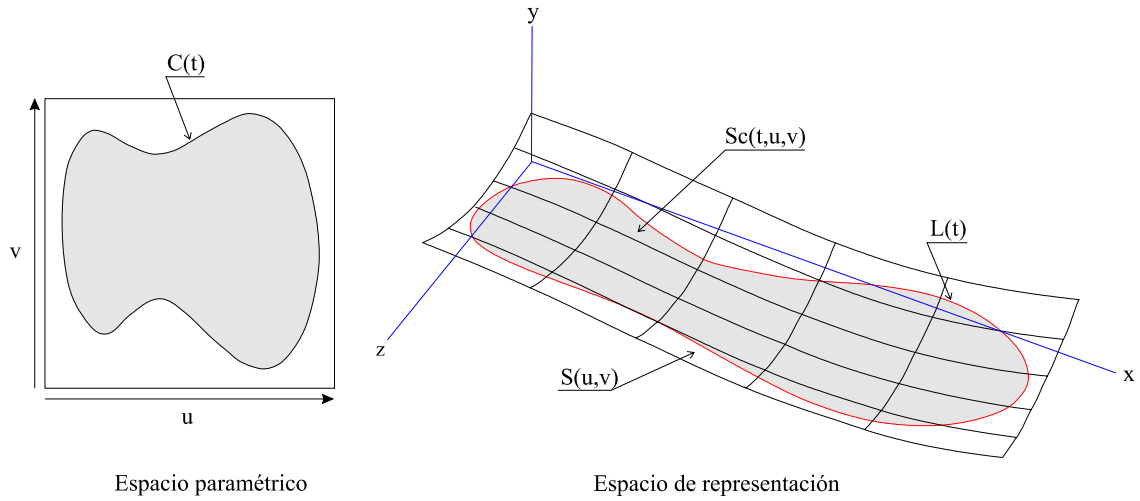

Fig. 1 - Superficie recortada

El problema con el que se encuentra el usuario es la definición de la curva C(t) para dar lugar a la curva L(t), ya que él trabaja sobre el espacio de representación y no sobre el paramétrico. El presente documento trata el problema inverso: encontrar una forma de definir C(t) a partir de L(t).

Planteamos inicialmente las siguientes restricciones:

• Utilizaremos superficies y curvas de grado cúbico. Esta restricción no afecta de forma importante a los sistemas CAD actuales, ya que son las de uso más extendido por sus buenas propiedades y su formulación sencilla. No obstante, es posible generalizar el método para superficies y curvas de cualquier grado.

• Las superficies se encuentran en forma Bézier. Esta segunda restricción no es tal, puesto que el paso de forma BSpline, Spline o NURBS a forma Bézier es inmediato, utilizando algoritmos bien conocidos de transformación [1]. Se ha adoptado esta decisión debido a que el manejo separado de los "patches" facilita mucho los cálculos.

• Tanto los parámetros de las superficies como los de las curvas toman valores en el intervalo [0, 1]. Esto tampoco supone pérdida de generalidad, bastando con una reparametrización si fuera necesaria.

Los siguientes apartados presentan los pasos necesarios para obtener la curva de recorte C(t) a partir de la L(t). Aunque los métodos de obtención de la curva tridimensional L(t) no influyen en el planteamiento del método, existe una posibilidad para la definición de L(t) que reduce los cálculos necesarios. Por esta razón se plantean dos propuestas: Obtención de la curva de recorte a partir de una curva 3D (método general), y obtención a partir de una curva 2D diseñada en un plano y proyectada sobre la superficie.

2.1. Diseño a partir de la curva 3D

La obtención de una curva 3D situada sobre una superficie se puede hacer de diversas maneras. En el presente documento proponemos su definición como intersección de superficies. De esta manera aprovechamos la potente herramienta de diseño que supone modelar objetos tridimensionales como intersección de sus caras formadas por superficies libres. Existen algunos algoritmos que resuelven la intersección entre superficies, tanto de manera analítica como por aproximación, estos últimos basados generalmente en divisiones sucesivas [1, 2]. Como resultado, se obtienen curvas 3D situadas sobre la superficie (Fig. 2).

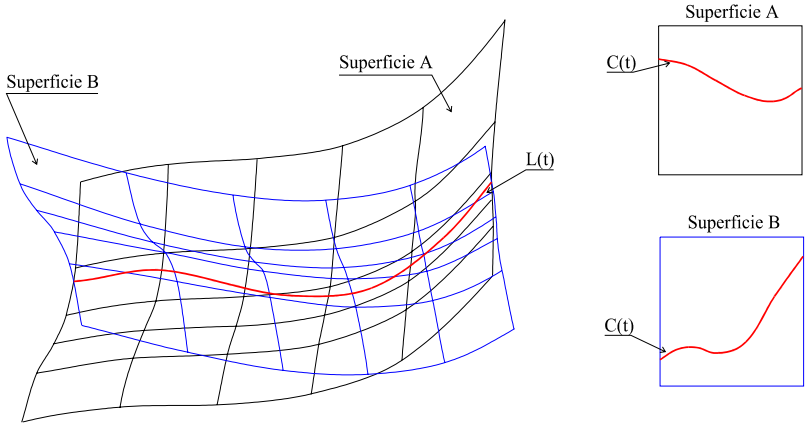

Fig. 2 - Diseño por intersección de superficies

Supongamos una curva tridimensional L(t) situada sobre la superficie, obtenida por el método propuesto o por cualquier otro. Nuestra aportación consiste en obtener la curva bidimensional C(t) que definida sobre el espacio paramétrico de la superficie permite recortarla para que su frontera coincida con L(t)..

Sea la superficie paramétrica bicúbica S(u, v) = (x, y, z) definida sobre un espacio paramétrico (u, v) y un espacio de representación (x, y, z), y calculada a partir de un conjunto de puntos de control Bézier p_{ij} dispuestos en forma de poliedro (poliedro de control). Podemos escribirla como

$$S(u, v) = \mathbf{UMPM}^T\mathbf{V} \qquad (1)$$

donde $\mathbf{U} = [1\ u\ u^2\ u^3]$, $\mathbf{V}^T = [1\ v\ v^2\ v^3]$, **P** es una matriz cuadrada de orden 4 donde guardamos los puntos de control y **M** una matriz cuadrada de coeficientes de orden 4 [1, 2]. Operando (1) obtenemos la siguiente ecuación vectorial

$$\begin{aligned} S(u, v) = {} & \mathbf{a}_{00}\, u^3v^3 + \mathbf{a}_{01}\, u^3v^2 + \mathbf{a}_{02}\, u^3v + \mathbf{a}_{03}\, u^3 + \\ & \mathbf{a}_{10}\, u^2v^3 + \mathbf{a}_{11}\, u^2v^2 + \mathbf{a}_{12}\, u^2v + \mathbf{a}_{13}\, u^2 + \\ & \mathbf{a}_{20}\, uv^3 + \mathbf{a}_{21}\, uv^2 + \mathbf{a}_{22}\, uv + \mathbf{a}_{23}\, u + \\ & \mathbf{a}_{30}\, v^3 + \mathbf{a}_{31}\, v^2 + \mathbf{a}_{32}\, v + \mathbf{a}_{33} = (x, y, z) \end{aligned} \qquad (2)$$

donde $\mathbf{a}_{ij}$ son coeficientes vectoriales que se calculan a partir de los puntos p_{ij} del poliedro de control. Esto da lugar a tres ecuaciones bicúbicas con dos incógnitas. Para cada valor (x, y, z) de la curva L(t), podemos plantear tres sistemas de ecuaciones con dos incógnitas cada uno (los correspondientes a las componentes xy, xz e yz). Resolviendo estos sistemas (por ejemplo, por el método de Newton-Raphson, que es el que se ha implementado) se obtienen tres conjuntos de tres soluciones para cada uno de ellos.

Los valores de los parámetros (u, v) para un punto (x, y ,z), serán aquellos pares (u, v) que cumplan:

1. Son solución real común a los tres sistemas.
2. Su valor se encuentra en el intervalo de definición de los parámetros [0, 1].

Si la superficie no presenta ningún cruce (no existen dos valores de (u, v) que tengan una misma imagen en el espacio de representación), y el punto de la curva se encuentra sobre la superficie, esta solución debe existir y ser única.

El procedimiento consiste, por lo tanto, en calcular la imagen en el espacio paramétrico de los puntos de la curva L(t) y así hallar los puntos de C(t). Sin embargo, no basta con calcular la imagen de los puntos de control de L(t), ya que la transformación al espacio paramétrico no es una transformación afín [1], es decir, la imagen de los puntos de control de L(t) en el espacio paramétrico no equivale a los puntos de control de C(t).

No obstante, la imagen de cada uno de los puntos de L(t) sí equivale a un punto de C(t), de manera que transformando un número suficiente de puntos de la curva, podemos obtener una buena aproximación de la misma. Aunque no es un método exacto, la calidad de la aproximación depende del número de puntos que se tomen. En la práctica, para un número razonable de puntos, la aproximación obtenida es suficiente.

El algoritmo propuesto, puede escribirse de la siguiente manera:

```
Sea S(u, v) la superficie que queremos recortar
Sea L(t) una curva 3D definida sobre la superficie
Sea Precisión el número de puntos que deseamos tomar de la curva

Para t=0 hasta 1 paso = 1/Precisión hacer
    (x, y, z) = L(t);
    SistemaXY = PlantearSistema (S, x, y);
    SistemaXZ = PlantearSistema (S, x, z);
    SistemaYZ = PlantearSistema (S, y, z);
    SolucionesXY = Resolver (SistemaXY);
    SolucionesXZ = Resolver (SistemaXZ);
    SolucionesYZ = Resolver (SistemaYZ);
    SolucionFinal = ExtraerSolucion (SolucionesXY, SolucionesXZ, SolucionesYZ);
    C = Añadir (C, SolucionFinal);
FinPara;
S_C (t, u, v) = RecortarSuperficie (S,C);
```

2.2. Diseño mediante curvas 2D

Dentro de los posibles métodos para el diseño de la curva L(t), existe la posibilidad de realizar la proyección de una curva bidimensional sobre la superficie a recortar y así obtener esta curva. Aunque se puede utilizar el caso general visto en el apartado anterior para realizar el proceso, el caso presenta algunas particularidades que permiten simplificar el problema.

Para poder aprovecharse de este método, es necesario que se cumplan las siguientes condiciones:

• El plano de definición de la curva debe ser uno de los planos coordenados. Si se utiliza otro plano, basta con hacer las transformaciones necesarias para llevar dicho plano a uno de los tres.

• Para facilitar los cálculos, conviene que la superficie sea univariada con respecto al plano elegido, es decir, que para cada punto de la curva definida sobre el plano su proyección sobre la superficie sea única.

Sea L'(t) una curva cúbica paramétrica definida sobre un espacio $\Re$ para el parámetro t, y con un espacio de representación $\Re^2$, diseñada sobre un plano coordenado (por ejemplo el XY). Para cada valor de t se obtiene un punto (x, y) sobre el plano. Podemos sustituir estos valores directamente en la ecuación (2), aunque en este caso sólo es posible plantear un sistema de ecuaciones (el correspondiente a las componentes x e y). De esta manera existe un único conjunto de tres soluciones, puesto que estamos calculando los valores de (u, v) para (x, y) y cualquier valor de z, es decir, todas las proyecciones del punto (x, y) sobre la superficie. Serán soluciones válidas aquellas soluciones reales que se encuentren dentro del intervalo de definición del parámetro [0,1]. Pueden, por lo tanto, existir hasta tres soluciones, lo cual es perfectamente posible pues las

superficies paramétricas bicúbicas pueden ser trivariadas para un único valor de (x, y). No obstante, si hemos tenido la precaución de elegir un plano como el propuesto esta solución será única si existe. Si no existiera, el punto no tiene proyección sobre la superficie (Fig. 3).

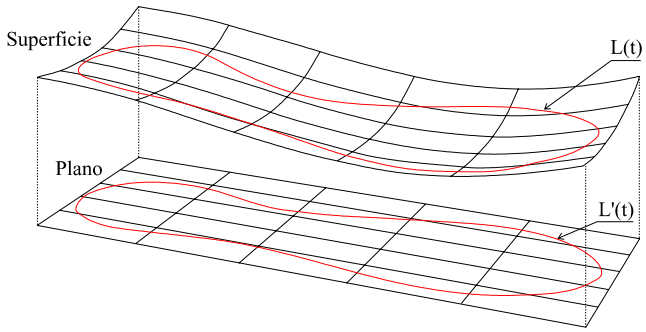

Fig. 3 - Proyección de curvas 2D sobre una superficie

El algoritmo es:

```
Sea S(u, v) la superficie que queremos recortar
Sea L'(t) una curva 2D definida sobre un plano (XY)
Sea Precisión el número de puntos que deseamos tomar de la curva

Para t=0 hasta 1 paso = 1/Precisión hacer
    (x, y) = L'(t);
    SistemaXY = PlantearSistema (S, x, y);
    Solucion = Resolver (SistemaXY);
    Si Existe (Solucion) entonces
        C = Añadir (C, SolucionFinal);
    FinSi;
FinPara;
SC(t, u, v) = RecortarSuperficie (S, C);
```

3. VISUALIZACIÓN DE SUPERFICIES RECORTADAS

Existen dos posibilidades inmediatas para la visualización realista de las superficies recortadas: La visualización directa de la superficie mediante la adaptación de los algoritmos de rendering, y la conversión a un modelo secundario poligonal, para aplicar posteriormente algoritmos conocidos de rendering de polígonos.

El primer método presenta dos ventajas importantes: se trata de una visualización directa sin modelo secundario y además, es exacta. Sin embargo, tropieza con un problema: es necesario desarrollar nuevos algoritmos de ray tracing, que tienen una complejidad muy superior a la de los algoritmos conocidos de ray tracing de polígonos [5, 6]. Además, no es posible utilizar las librerías gráficas actuales. Todo ello repercute en una velocidad de rendering muy baja.

Estos inconvenientes nos lleva a la utilización de un modelo secundario para la visualización, para el que existen gran cantidad de algoritmos conocidos. Se trata del modelo poligonal, que además tiene otras ventajas:

• La gran mayoría de librerías aceleradoras de gráficos están pensadas para visualizar polígonos, por lo que la visualización no requiere esfuerzos añadidos de programación.
• Es reutilizable para realizar otros cálculos (análisis sólido mediante descomposición en elementos finitos, ...).

• Permite homogeneizar el proceso de visualización cuando aparecen objetos diseñados con diferentes técnicas en una misma escena

3.1. Planteamiento

La definición de la curva de recorte sobre el espacio paramétrico resulta de gran utilidad a la hora de la visualización. Tenemos cuatro ventajas fundamentales:

• Trabaja sobre dos dimensiones: reduce el problema de intersección de planos al de intersección de rectas.

• La transformación resulta univariada.

• El espacio es topológicamente cuadrado y acotado.

• Si además tomamos polígonos cuadrados, el test de pertenencia de un punto al polígono y la intersección con rectas (sólo verticales y horizontales) ven reducida su complejidad al mínimo.

Lo que se pretende es definir una serie de polígonos bidimensionales sobre el espacio paramétrico, para después obtener el polígono real al llevarlo al espacio de representación. Estos polígonos, irán subdividiéndose cuando sean atravesados por la curva de recorte, dando lugar a otros nuevos. Finalmente serán etiquetados como interiores o exteriores, llevando al espacio de representación los interiores a la curva.

Sea $S_C(t, u, v)$ una superficie recortada, definida tal y como se ha presentado en los apartados anteriores. El algoritmo propuesto trabaja sobre el espacio paramétrico, para aprovechar la simplicidad que éste le ofrece. El método desarrollado se basa en una generalización de los algoritmos de recorte de polígonos (polygon clipping), en concreto, en el algoritmo de Weiler-Atherton [3]. Su funcionamiento es sencillo y se basa en la subdivisión de los polígonos atravesados por la curva.

A grandes rasgos, el método funciona de la siguiente manera: El primer paso es poligonalizar la superficie completa, sin recortar. Es posible utilizar cualquier tipo de polígono, pero si utilizamos polígonos rectangulares los cálculos se simplifican. Además, tenemos que discretizar la curva de recorte, convirtiéndola a polilínea, con la precisión deseada. A continuación se va recorriendo cada uno de los tramos de la curva hasta que intersecte con alguna arista de algún polígono, es decir hasta que entre dentro del polígono. El polígono se divide en dos, polígono derecho y polígono izquierdo, y se van añadiendo los siguientes puntos: El punto de intersección de la curva a la entrada del polígono se guarda en ambos polígonos; también se guardan en los dos todos los puntos de la curva hasta que salga del polígono, así como el punto de intersección a la salida del polígono; por último, en el polígono derecho se guardan los vértices del polígono original que queden a la derecha de los tramos de curva contemplados, y en el izquierdo los que queden a la izquierda. Sólo falta sustituir el polígono inicial por los nuevos polígonos derecho e izquierdo y repetir el proceso para el polígono contiguo, en el que se entra, hasta que se recorra toda la curva (Fig. 4).

En el algoritmo anterior, cada polígono puede ser recorrido y subdividido varias veces, de manera que si la curva sale y vuelve a entrar en el polígono, se divide una vez más. La última parte del algoritmo consiste en etiquetar como interiores los polígonos que se encuentren dentro de la curva, y como exteriores el resto y llevar al espacio de representación todos los polígonos interiores.

Este algoritmo puede dar lugar a diferentes tipos de polígonos, no necesariamente coplanares ni convexos. Algunas librerías, sin embargo, sólo son capaces de representar correctamente polígonos coplanares y convexos, por lo que es necesario incluir algún tipo de postproceso antes de llevar los polígonos al espacio de representación. Proponermos llevar a cabo una triangularización en ese punto, para asegurar la convexidad y la coplanareidad.

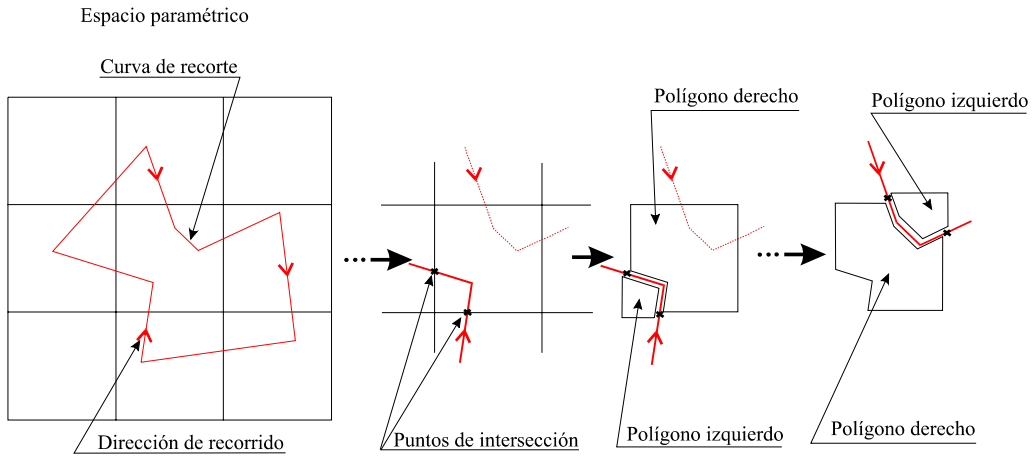

Fig. 4 - Funcionamiento del algoritmo de poligonalización

3.2. Algoritmo de poligonalización

Sea S_C (t, u, v) la superficie que queremos recortar
Sea C(t) una curva de recorte definida sobre el espacio paramétrico
Sea Precisión el número de puntos que deseamos tomar de la curva

```
Polígonos = Poligonalizar (S (u, v));
Polilínea = ConvertirAPolilínea (C, Precisión);
Tramo = PrimerTramo (Polilínea);
Mientras no Intersecta (Tramo, Polígonos) hacer
   Tramo = SiguienteTramo (Polilínea);
FinMientras;
Polígono = PoligonoIntersección (Tramo, Polígonos);
Mientras no Vacía (Polilínea) hacer
   PuntoEntrada = IntersectarEntrada (Tramo, Polígono);
   PuntoSalida = IntersectarSalida (Tramo, Polígono);
   Insertar (PolígonoDerecho, PuntoEntrada);
   Insertar (PolígonoIzquierdo, PuntoSalida);
   Para cada Punto desde PuntoEntrada a PuntoSalida hacer
        Insertar (PolígonoDerecho, Punto);
   FinPara;
   Para cada Punto desde PuntoSalida a PuntoEntrada hacer
        Insertar (PolígonoIzquierdo, Punto);
   FinPara;
   Insertar (PolígonoDerecho, PuntoSalida);
   Insertar (PolígonoIzquierdo, PuntoEntrada);
   Para cada Vértice de Polígono hacer
        Si Derecha (Vértice, PuntoEntrada, PuntoSalida)
             Insertar (PolígonoDerecho, Vértice);
        FinSi;
        Si Izquierda (Vértice, PuntoEntrada, PuntoSalida)
             Insertar (PolígonoIzquierda, Vértice);
        FinSi;
   FinPara;
   Eliminar (Polígono, Polígonos);
```

```
    Insertar (PolígonoDerecho, Polígonos);
    Insertar (PolígonoIzquierdo, Polígonos);
    Tramo = SiguienteTramo (Polilínea);
    Polígono =  PolígonoIntersección (Tramo, Polígonos);
FinMientras
Etiquetar (Polígonos);
ConvertirARepresentación (Polígonos);
```

Obsérvese que la inserción de puntos de la curva de recorte, se hace de forma inversa en los polígonos derecho e izquierdo, con el objeto de conservar la ordenación, horaria o antihoraria.

4. PRUEBAS Y RESULTADOS

Los algoritmos planteados han sido construidos sobre un entorno Windows, instalado en una máquina con procesador Intel Pentium a 90 MHz. y codificados en lenguaje C++. La visualización del modelo poligonal se ha realizado utilizando las librerías gráficas Intel 3DR. Este desarrollo se ha integrado en un proyecto de INESCOP para diseño y fabricación de pisos y moldes de calzado por ordenador. Los algoritmos han sido utilizados con casos reales. En las figuras 5, 6, 7, se muestran algunos de esos casos.

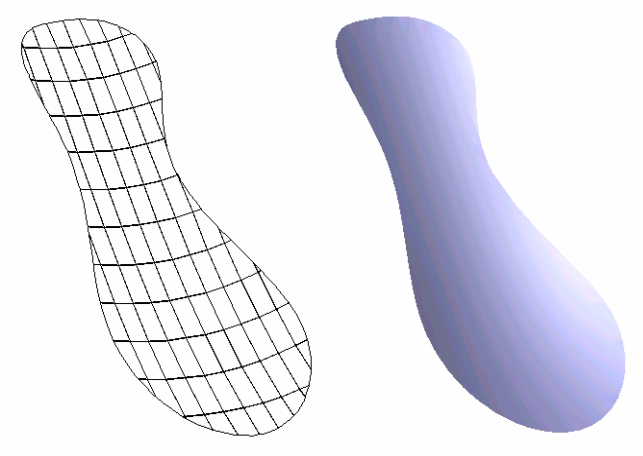

Fig. 5 - Poligonalización y visualización del objeto de las figuras 1 y 2 [1]

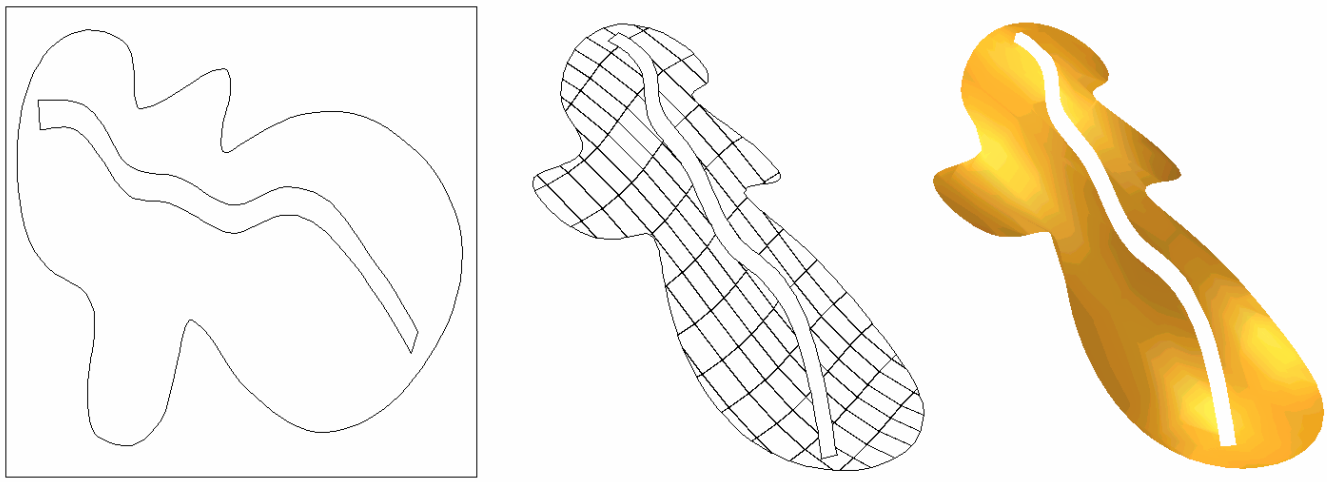

Fig. 6 - Superficie recortada por varias curvas [1]

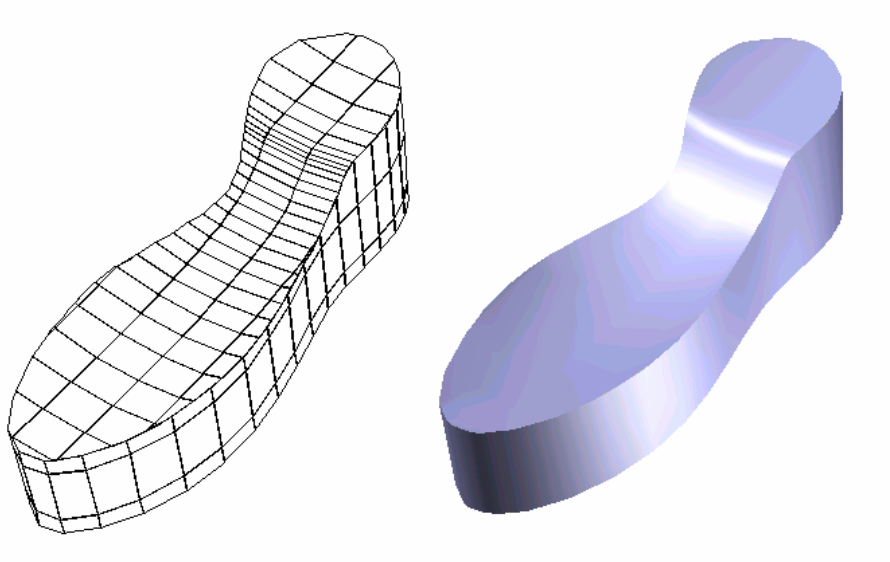

Fig. 7 - Poligonalización y visualización de un objeto formado por superficies recortadas [1]

A modo ilustrativo, se presentan en la tabla 1, los resultados obtenidos para las figuras anteriores. Para cada una de ellas se incluye el tiempo de poligonalización en segundos, incluyendo poligonalización inicial, recortado, etiquetado y transformación al espacio de representación, y el tiempo total del proceso de rendering, incluyendo la visualización utilizando las librerías gráficas Intel 3DR.

Figura	nº superficies	nº curvas	nº polígonos	nº tramos	t poligonalización	t total
Fig. 5	1	1	144	109	2,1 s	5,2 s
Fig. 6	1	2	144	153	5,4 s	9,1 s
Fig. 7	3	4	500	436	15,1 s	19,3 s

Tabla 1 - Resultados

5. CONCLUSIONES

Hemos presentado un conjunto de soluciones a algunos de los problemas que plantea el uso de superficies recortadas para el modelado de objetos tridimensionales en un sistema CAD. En concreto se ha dado solución a los siguientes problemas:

• Para el problema del diseño de las curvas de recorte, se ha planteado la necesidad de trabajar sobre el espacio tridimensional de representación, que es el que maneja el usuario del sistema. Se ha descrito un algoritmo que permite la transformación de la curva tridimensional en una curva de recorte en el espacio paramétrico. También se ha incluido una simplificación importante para un caso particular en el que la curva es diseñada sobre un plano y proyectada sobre la superficie. Las mayores ventajas del método son: permitir al usuario trabajar sobre el espacio de representación de forma transparente, una velocidad aceptable de proceso y una precisión definible por él mismo.

• La visualización se ha resuelto mediante el uso de un modelo secundario de tipo poligonal, para tratar de aprovechar los algoritmos y librerías gráficas existentes. El algoritmo descrito funciona sobre el espacio paramétrico, lo que disminuye su complejidad de manera muy importante. Los tiempos obtenidos para el render se pueden considerar buenos a la vista de los sistemas que utilizan visualización directa.

En un futuro, se pretende construir un modelo geométrico enteramente basado en la utilización de superficies recortadas. Se está ya trabajando en la eliminación de algunas de las restricciones planteadas: La

generalización del método de transformación al espacio de los parámetros para superficies y curvas de cualquier grado; la posibilidad de utilizar cualquier plano para la definición de las curvas de proyección (sea o no la superficie univariada con respecto a ese plano), etc.

6. REFERENCIAS

6.1. Referencias bibliográficas

[1] **CURVES AND SURFACES FOR COMPUTER AIDED GEOMETRIC DESIGN. A PRACTICAL GUIDE.**
Farin, G.
Academic Press, Inc.
ISBN 0-12-249052-5, 1993.

[2] **COMPUTATIONAL GEOMETRY FOR DESIGN AND MANUFACTURE**
Faux, I.D., Pratt, M.J.
Ellis Horwood Ltd, Publishers.
ISBN 0-85312-114-1, 1979

[3] **COMPUTER GRAPHICS**
Hearn, D., Baker, M.P.
Prentice Hall, Inc.
ISBN 0-13-159690-X, 1994.

[4] **RAY TRACING TRIMMED RATIONAL SURFACE PATCHES**
Nishita, T., Sederberg, T.W., Kakimoto, M.
Computer Graphics, Volume 24, Number 4. Págs 337-345.
ACM 0-89791-344-2/90/008/0337, 1990.

[5] **UN MODELO PARAMÉTRICO DE FRONTERAS PARA LA REPRESENTACIÓN DE SÓLIDOS POR GEOMETRÍA CONSTRUCTIVA.**
Vivó Hernando, R.
Tesis doctoral. Universidad Politécnica de Valencia.
1992.

6.2. Referencias a ilustraciones

[1] Imágenes de superficies utilizadas en el diseño de pisos para calzado, obtenidas del proyecto PYM (Diseño y Fabricación por Ordenador de Pisos y Moldes para Calzado), desarrollado por el Instituto Español del Calzado y Conexas - INESCOP.

NOTAS

(*) Molina Carmona R.
Departamento de Tecnología Informática y Computación (Universidad de Alicante)
Departamento de Informática (Instituto Español del Calzado y Conexas - INESCOP).

Puchol García, J.A.
Departamento de Tecnología Informática y Computación (Universidad de Alicante)

Salas Pérez, F.
Departamento de Informática (Instituto Español del Calzado y Conexas - INESCOP).