UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MÁSTER

MÁSTER EN MATEMÁTICAS

Geometría y visualización

Autor
Jesús Bueno Urbano

Directores
Pedro A. García Sánchez
Carlos Ureña Almagro

ESCUELA INTERNACIONAL DE POSGRADO

—

Curso académico 17/18

Geometría y visualización

Jesús Bueno Urbano

Resumen

En este Trabajo Fin de Máster hemos querido hacer un recorrido por los conceptos básicos de del Ray Tracing para poder representar y visualizar superficies implícitas. Para ello en el primer capítulo haremos un repaso sobre métodos elementales para implicitar superficies, es decir, para poder transformar una expresión no implícita, paramétrica en nuestro caso, de una superficie en la expresión implícita de ésta. Sobre todo nos centraremos en el método de la base de Gröbner y presentaremos los métodos de la resultante y de Wu-Ritt.

Ya en el segundo capítulo el tema principal de éste será buscar métodos de visualización de superficies como la poligonalización de superficies, en particular, presentaremos un algoritmo de triangulación extraído de [Har03]. A continuación explicaremos el concepto de Ray Tracing.

A continuación haremos una introducción al Análisis de Intervalos y a sus propiedades, a los Intervalos Modales y a las llamadas extensiones semánticas. Todo esto nos servirá para, finalmente, aplicarlo al Ray Tracing y así presentar varios algoritmos mejorados cuya eficiencia compararemos.

Yo, **Jesús Bueno Urbano**, estudiante del Máster Universitario en Matemáticas de la **Escuela Internacional de Posgrado de la Universidad de Granada**, con DNI 20078941X, autorizo que la siguiente copia de mi Trabajo Fin de Máster pueda ser consultada por las personas que lo deseen.

Fdo: Jesús Bueno Urbano

En Granada, a 21 de septiembre de 2018.

Agradecimientos

En primer lugar me gustaría agradecer el apoyo que me ha dado mi familia durante todos mis estudios universitarios, en especial a mis padres, Jesús y María Luisa, por el apoyo emocional y económico. También a mi pareja, David, por entender esos momentos en los que tenía que estar más atento de redactar que de él.

Además me gustaría agradecer el trabajo de mis tutores Pedro y Carlos por buscar y proponerme un tema como éste, menos convencional que los habituales que se explican durante el Grado y Máster, y por guiarme y corregir el camino que he seguido durante estos 10 meses.

Índice general

Índice de figuras

Capítulo 1

Implicitación de superficies

1.1. Introducción

Las superficies implícitas se usan en la Ciencia de Computación Gráfica desde los años 70 para modelar objetos geométricos, pero su uso e importancia han ido creciendo en años recientes ya que pueden ser usadas para describir objetos en espacios de dimensión arbitraria. En este Trabajo Fin de Máster nos centraremos en objetos de dimensión dos y tres. En muchas ocasiones, un mismo objeto matemático se puede definir de una forma particular, por tanto nos cabe preguntarnos por qué las superficies en forma implícita se han popularizado tanto. La respuesta la encontramos en que la expresión implícita de una superficie, en contraste con la definición paramétrica del de la misma, suele ser compacta, manejable y es trivial comprobar la pertenencia de un punto al objeto dado. Por ejemplo, una expresión paramétrica de la esfera unidad es

$$X(u,v) = (\cos(u)\sin(v), \sin(u)\sin(v), \cos(v)), \qquad (u,v) \in [0, 2\pi] \times [0, \pi],$$

mientras que su expresión en forma implícita es

$$f(x,y,z) = x^2 + y^2 + z^2 - 1.$$

Aunque hemos mencionado algunas ventajas de usar la forma implícita para la modelización y visualización de superficies, su principal debilidad es la cantidad de tiempo que necesitan para la visualización directa, por ejemplo usando Ray Tracing [GW05]. Otra de las debilidades de las superficies en forma implícita es la dificultad de controlar la forma de las superficies durante una visualización rápida en un entorno interactivo. Esto lleva a que las representaciones paramétricas sigan siendo populares hoy en día gracias a la relativamente rápida renderización que presentan.

Aún presentando estas debilidades, las superficies en forma implícita son una forma flexible de crear objetos complejos ya que ofrecen una clasificación manejable y clara de

conjuntos de puntos, es decir, es fácil saber si un punto del espacio se encuentra *dentro*, *fuera* o *en* la superficie.

Además también se pueden usar para la representación de nubes de puntos. Por ejemplo, en imágenes de datos médicos y reconstrucción de objetos representados como medias de conjuntos de puntos [B+05, P+06]. En [Uhl03] se describe el llamado *método RBF* basado en técnicas variacionales donde, dados una serie de puntos de una superficie S de la cual no conocemos su expresión, procedemos a calcular una función en forma implícita que modele una superficie S' que sea una aproximación razonable de la superficie inicial e interseque a los puntos que conocemos.

1.2. Descripción del problema

Una representación implícita de una superficie S se define como el conjunto de puntos $p \equiv (x, y, z) \in \mathbb{R}^3$ que son solución de la ecuación $f(p) = 0$, donde $f : \mathbb{R}^3 \to \mathbb{R}$ es una función que asigna un valor escalar a cada punto del espacio. Es decir,

$$S = \{p \in \mathbb{R}^3 : f(p) = 0\}.$$

Sea el sólido A el espacio descrito por la preimagen de la función f en $]-\infty, 0]$. Por uniformidad consideraremos el siguiente criterio:

$$\begin{array}{lll} p \in \text{int}(A) & \text{si} & f(p) < 0, \\ p \in \partial A & \text{si} & f(p) = 0, \\ p \in \text{ext}(A) & \text{si} & f(p) > 0. \end{array}$$

Donde $\text{int}(A)$, ∂A y $\text{ext}(A)$ denotan el interior, frontera y exterior de A respectivamente. Esto establece por medio de la vía topológica que la función implícita es negativa en el *interior*, cero en la superficie, véase *frontera*, y positiva en el *exterior* [Har01].

Este sistema es un estándar que se suele usar por comodidad y por ser el más común entre la literatura de este tipo de trabajos. Otros ejemplos incluyen a [Uhl03] o [Bli82] donde las funciones implícitas definidas eran positivas en el interior y negativas en el exterior del sólido, o por ejemplo [Ric73] donde eran siempre positivas y alcanzaban el valor unidad en la superficie, menos uno en el interior y mayor que uno en el exterior.

Está claro que estas definiciones no afectan al problema principal, véase que, volviendo al ejemplo de la esfera, en [Uhl03] usan lo que llaman la forma inversa que sería

$$f(x, y, z) = -x^2 - y^2 - z^2 + 1.$$

Ahora que hemos descrito las ventajas de las superficies dadas de forma implícita queremos ver los métodos para crear la representación implícita de objetos arbitrarios.

Hay infinidad de métodos para la realizar la implicitación de superficies, algunos ya los hemos mencionado como los basados en nubes de puntos y/o técnicas variacionales, pero aquí nos centraremos en los métodos de eliminación de variables que parten de expresiones paramétricas de la superficie, o parte de a superficie, para después modificarlas y así obtener una expresión implícita de ésta mediante operaciones simbólicas.

1.3. Métodos de eliminación de variables

Eliminación es una disciplina matemática para suprimir variables de sistemas de ecuaciones polinomiales. En nuestro caso este método se aplica a la parametrización de la superficie que se expresa como una sistema de ecuaciones

$$\begin{aligned} x_1 &= f_1(t_1, \ldots, t_m), \\ &\vdots \\ x_n &= f_n(t_1, \ldots, t_m). \end{aligned}$$

En donde las funciones f_i son polinomios o funciones racionales de éstos.

El método consiste encontrar las relaciones entre las variables y, así pues, la ecuación implícita que nos expresa la misma superficie que el sistema de ecuaciones de su expresión paramétrica.

En [Hof93] se hace una clasificación del método de la resultante, el método de la base de Gröbner y el método de Wu-Ritt. Todos los métodos que vamos a describir tienen una propiedad común y ésta es que el resultado del algoritmo es una ecuación que puede ser utilizada por métodos de visualización directa.

1.3.1. Método de la base de Gröbner

Este método se basa en encontrar una base de Gröbner para un ideal I, donde éste es un conjunto de polinomios que cumple con el requisito de existencia de una base de Gröbner. La búsqueda de una base de Gröbner reducida se basa en la búsqueda de una solución exacta de un sistema de ecuaciones polinomiales. Si el sistema de ecuaciones polinomiales tiene una solución, entonces las variables del sistema son eliminadas y el conjunto original de ecuaciones se transforma. Este nuevo conjunto de ecuaciones transformadas sí puede ser solucionado de forma sencilla.

La transformación de la expresión paramétrica en un expresión implícita puede ser resuelto de una forma satisfactoria usando una base de Gröbner de un ideal. Las parametrizaciones que tenemos como entrada pueden ser tanto polinomiales como racionales.

Ideal

Un subconjunto $I \subset k[x_1, \ldots, x_n]$ se dice ideal en $k[x_1, \ldots, x_n]$ si se verifican las siguientes dos condiciones.

1. Si $f, g \in I$, entonces $f + g \in I$.
2. Si $f \in I$, entonces $fg \in I$ para todo $g \in k[x_1, \ldots, x_n]$.

Sean $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$, el conjunto $I = \{\sum_{i=1}^{s} g_i f_i \, : \, g_i \in k[x_1, \ldots, x_n]\}$ es un ideal de $k[x_1, \ldots, x_n]$ y además es el menor ideal que contiene a los polinomios $f_1, \ldots, f_s$. El conjunto $\{f_1, \ldots, f_s\}$ se llama conjunto generador o base del ideal I.

Ordenamiento de los polinomios

Para la computación de la base de Gröbner, necesitamos el concepto de orden monomial.

Definición 1.3.1. Un orden monomial se define como un orden total sobre el conjunto de los monomios en un anillo polinomial verificando esta propiedad respecto del producto, i.e., dados dos monomios u y v tales que $u \leq v$ y sea w otro monomio, entonces $uw \leq vw$.

En el caso de una cantidad finita de variables se tiene la siguiente forma equivalente.

Definición 1.3.2. Un orden sobre el conjunto de monomios en un anillo polinomial se dice monomial si se verifican las siguientes condiciones:

- El orden es total.
- Si u es un monomio cualquiera se tiene que $1 \leq u$.

Aunque existen varios órdenes monomiales, y se pueden escoger según la situación, los más comunes son los siguientes.

Orden lexicográfico

Dado el conjunto de monomios en n variables $x_1, \ldots, x_n$ tales que establecemos que $x_1 \prec x_2 \prec \ldots \prec x_n$ el orden lexicográfico se define como

$$1 \prec x_1 \prec x_1^2 \prec \ldots \prec x_2 \prec x_1 x_2 \prec x_1^2 x_2 \prec \ldots \prec x_2^2 \prec x_1 x_2^2 \prec \ldots \prec x_n \prec \ldots$$

Orden lexicográfico graduado

A diferencia del método anterior, este método primero ordena los términos por su grado y los términos de igual grado se ordenan de manera lexicográfica. Tomando el ejemplo anterior nos quedaría:

$$1 \prec x_1 \prec x_2 \prec \ldots \prec x_n \prec x_1^2 \prec x_1 x_2 \prec x_1 x_2 \prec \ldots \prec x_1 x_n \prec x_2^2 \prec x_2 x_3 \prec \ldots$$

Reducción polinomial

Para calcular la base de Gröbner es importante elegir un orden $\prec$, por ello tras haberlo elegido pasamos a definir los siguientes términos.

Definición 1.3.3. Para cada polinomio $f(x_1, \dots, x_n)$ se define el monomio líder como el mayor término de f bajo $\prec$ con coeficientes no nulos. Se denota por $LM(f)$.

Nota. El coeficiente del monomio líder se llama coeficiente líder y se denota por $LC(f)$.

Definición 1.3.4. El término líder de un polinomio f se define como el producto del monomio líder. Se denota por $LT(f)$.

Definición 1.3.5. La cola de un polinomio $f(x_1, \dots, x_n)$, denotado por $TT(f)$ se obtiene separando el término líder del resto del polinomio.

Con las definiciones dadas se puede reescribir un polinomio $f(x_1, \dots, x_n)$ como

$$f = LT(f) + TT(f).$$

Ahora podemos proceder a la reducción polinomial propiamente dicha.

Dados dos polinomios $f(x_1, \dots, x_n)$ y $g(x_1, \dots, x_m)$ se dice que g reduce a un polinomio h respecto de f si, y sólo si, $LT(g)$ se puede eliminar mediante la resta de un múltiplo apropiado de f. Esta operación se denota por $g \to_f h$. Por tanto, la reducción $g \to_f h$ es posible si, y sólo si, existe un escalar b y un monomio u tales que $h = g - buf$ donde $b = \frac{LC(g)}{LC(f)}$ y $u = \frac{LM(g)}{LM(f)}$.

Se dice que un polinomio g se reduce respecto de un conjunto, o base, de polinomios $F = \{f_1, \dots, f_n\}$ si g es reducible respecto de uno o más polinomios de F. En tal caso la reducción de un polinomio puede conducir a una secuencia de reducciones, lo que es un proceso finito. Se puede probar además que cada polinomio g_i en la secuencia de reducciones y el propio polinomio g es un elemento del ideal $(f_1, \dots, f_n)$.

S-polinomios

Este proceso que hemos llevado a cabo nos conduce al concepto de **S-polinomios**. Para dos polinomios f y g se define su S-polinomio como:

$$S(f, g) = \frac{x^\gamma}{LT(f)} \cdot f - \frac{x^\gamma}{LT(g)} \cdot g.$$

Donde x^γ representa el máximo común divisor entre los monomios líderes de f y g.

Base de Gröbner

Después de escoger un orden, el conjunto $G = \{g_1, \dots, g_l\}$ del ideal I es una base de Gröbner si

$$\langle LT(g_1), \dots, LT(g_l)\rangle = \langle LT(I)\rangle.$$

Es decir, el conjunto $G \subset I$ es la base de Gröbner si, y sólo si, el término líder de cualquier elemento de I es divisible entre $LT(g_i)$ para algún $i = 1, \dots, l$. En consecuenia una base de Gröbner de un conjunto de polinomios es un tipo concreto de base del ideal que generan que cumple:

- Todo polinomio en el ideal se reduce a cero respecto a la base.
- Todo polinomio tiene una única forma normal respecto de la base.

Cuando la parametrización se compone de funciones polinomiales, ésta se puede expresar como

$$\begin{aligned} x_1 &= f_1(t_1, \dots, t_m), \\ &\vdots \\ x_n &= f_n(t_1, \dots, t_m). \end{aligned}$$

Donde $f_1, \dots, f_n$ son polinomios en $K[t_1, \dots, t_m]$ con K un cuerpo.

Este sistema se puede ver como la proyección $F : K^m \to K^n$ definida por

$$F(t_1, \dots, t_m) = (f_1(t_1, \dots, t_m), \dots, f_n(t_1, \dots, t_m)).$$

Entonces, la imagen es un subconjunto de K^n parametrizado por el sistema previo. Teniendo en cuenta que $F(K^m)$ no es una variedad afín, se obtiene que la solución del problema de conversión de ecuaciones paramétricas a implícitas es equivalente a encontrar la variedad mínima que contiene a $F(K^m)$, i.e., el problema de implicitación consiste en la eliminación de parámetros de la descripción paramétrica. La ecuación final contiene solo las variables $x_1, \dots, x_n$.
La eliminación de variables se puede realizar calculando la base de Gröbner reducida para un ideal $I = \langle x_1 - f_1, \dots, x_n - f_n\rangle$. Para enfrentarse a este problema solo es necesario tener en cuenta el orden $\prec$.

El segundo método es la implicitación racional, la cual puede ser expresada como

$$\begin{aligned} x_1 &= \tfrac{f_1(t_1,\dots,t_m)}{g_1(t_1,\dots,t_m)}, \\ &\vdots \\ x_n &= \tfrac{f_n(t_1,\dots,t_m)}{g_n(t_1,\dots,t_m)}. \end{aligned}$$

Donde $f_1, \dots, f_n, g_1, \dots, g_n \in K[t_1, \dots, t_m]$.

Sabemos que $F : K^m \to K^n$ no se puede definir en todo K^m ya que, obviamente, hay que excluir el conjunto de raíces de los polinomios g_i para todo $i = 1, \dots, n$. Si denotamos como $W \subset K^m$, entonces

$$F(t_1, \dots, t_m) = \left(\frac{f_1(t_1, \dots, t_m)}{g_1(t_1, \dots, t_m)}, \dots, \frac{f_n(t_1, \dots, t_m)}{g_n(t_1, \dots, t_m)} \right)$$

define la proyección $F : K^m \setminus W \to K^n$.

El objetivo es encontrar la variedad mínima en K^n que contenga $F(K^m/W)$. En la parametrización definida se eliminan las fracciones multiplicando la iésima coordenada por el polinomio g_i. Entonces la ecuación $1 - g_1 \dots g_n y = 0$, para polinomios g_i no nulos en la variedad definida, se añade y se evalúa la base de Gröbner reducida. Los elementos de la base de Gröbner que no contienen a las variables $t_1, \dots, t_n, y$ definen la representación implícita de la variedad afín dada. Una explicación más exhaustiva se puede encontrar en [Hof93].

Ejemplo

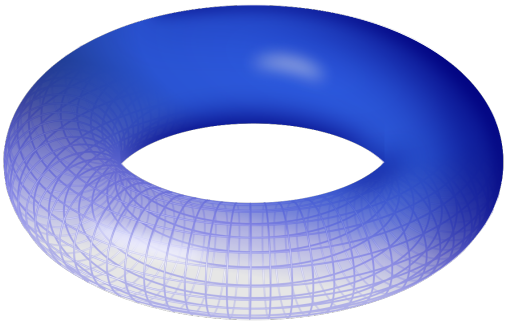

Figura 1.1: Representación clásica de un toro. Imagen extraída de [Wik15].

La expresión paramétrica de un toro es

$$\begin{aligned} x &= r \cos u \cos t + R \cos t, \\ y &= r \cos u \sin t + R \sin t, \\ z &= r \sin u. \end{aligned}$$

Si renombramos esta expresión como

$$c_u = \cos u, \quad c_t = \cos t, \quad s_u = \sin u \quad \text{y} \quad s_t = \sin t, \tag{1.1}$$

podemos representar la expresión paramétrica inicial en polinomios como

$$\begin{aligned} x - r c_u c_t - R c_t &= 0, \\ y - r c_u s_t - R s_t &= 0, \\ z - r s_u &= 0. \end{aligned} \tag{1.2}$$

Añadiendo las identidades

$$\begin{aligned} c_u^2 + s_u^2 - 1 = 0 &= 0, \\ c_t^2 + s_t^2 - 1 = 0 &= 0. \end{aligned} \tag{1.3}$$

La base de Gröbner reducida para el ideal I, generado por los polinomios (1.1) y (1.2), contiene 9 elementos. Uno solo de estos elementos no contiene variables en c_u, s_u, c_t ó s_t, y tiene la forma

$$(x^2 + y^2 + z^2 - r^2 - R^2)^2 = 4R^2(z^2 - r^2),$$

la cual es la expresión implícita del toro.

1.3.2. Método de la resultante

El término *resultante* se suele introducir si se presenta la siguiente cuestión: ¿Cuándo dos polinomios en el anillo de polinomios $K[x]$ tienen un divisor común? Los métodos que usan la evaluación de la resultante se pueden usar para eliminar un subconjunto de variables del sistema inicial de ecuaciones algebraicas no lineales. Un dato interesante de la resultante para polinomios en varias variables es que para $n+1$ polinomios ésta elimina n variables a la par. A diferencia del método de la base de Gröbner, este método es no secuencial. La idea básica que subyace en las resultantes multidimensionales es la conversión de un problema de eliminación no lineal en uno lineal, lo cual ayuda a aplicar métodos conocidos de Álgebra Lineal para resolver el sistema.

Existen distintos tipos de resultante. La definición básica involucra dos polinomios en una variable, por ejemplo la resultante de Sylvester o Bézout, y a partir de ahí se puede ir generalizando a dos polinomios en dos variables y después a tres polinomios en dos variables, la resultante de Dixon. Esta última puede generalizarse a $n+1$ polinomios en n variables. Aquí daremos una simple pincelada para dar la idea de las resultantes ya nombradas. Podemos encontrar más información en [Ber00].

Resultante de Sylvester

El principal problema es la tendencia a encontrar si dos polinomios $f, g \in K[x]$ tienen divisor común. Existen varias maneras de encontrarlo, por ejemplo, el algoritmo de Euclides se puede usar para descomponer los polinomios en productos de factores simples. O por ejemplo, el siguiente lema.

Lema 1.3.1. Sean $f, g \in K[x]$ tales que $deg(f) = n > 0$ y $deg(g) = m > 0$. Se tiene que f y g tienen un divisor común si y sólo si existen polinomios $A, B \in K[x]$ verificando que

1. ambos polinomios A y B son no nulos,
2. A y B tienen como mínimo grado $m-1$ y $n-1$ respectivamente y

3. $Af + Bg = 0$.

Definición 1.3.6. Sean $f, g \in K[x]$ dados como $f = a_n x^n + \cdots + a_0$ y $g = b_m x^m + \cdots + b_0$ donde $a_n, b_m \neq 0$, entonces la resultante de Sylvester de f y g es de la forma

$$\text{Res}(f, g) = \text{Det}(\text{Syl}(f, g)).$$

Donde $\text{Syl}(f, g)$ denota

$$\begin{pmatrix}
a_n & & & & & b_m & & & & \\
a_{n-1} & a_n & & & & b_{m-1} & b_m & & & \\
a_{n-2} & a_{n-1} & a_n & & & b_{m-2} & b_{m-1} & b_m & & \\
\vdots & \vdots & & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \\
a_1 & \dots & \dots & \dots & a_n & b_1 & \dots & \dots & \dots & b_m \\
a_0 & \dots & \dots & \dots & a_{n-1} & b_0 & \dots & \dots & \dots & b_{m-1} \\
 & a_0 & \dots & \dots & a_{n-2} & & b_0 & \dots & \dots & b_{m-2} \\
 & & \ddots & \vdots & \vdots & & & \ddots & \vdots & \vdots \\
 & & & a_1 & a_0 & & & & b_1 & b_0 \\
 & & & & a_0 & & & & & b_0
\end{pmatrix}$$

y los espacios en blanco de la matriz denotan ceros.

Ahora realizaremos un ejemplo sencillo para comparar este método con el de la base de Gröbner.

Sean por ejemplo los polinomios en la variable x cuyos coeficeintes son polinomios en la variable y

$$f = x^2 y - 1, \qquad g = x^2 + y^2 + xy - 4.$$

Aplicando el método de la resultante de Sylvester tenemos

$$\text{Res}(f, g) = \text{Det}\begin{pmatrix} y & 0 & 1 & 0 \\ 0 & y & y & 1 \\ -1 & 0 & y^2 - 4 & y \\ 0 & -1 & 0 & y^2 - 4 \end{pmatrix} = y^6 - 8y^4 + y^3 + 16y^2 - 8y + 1.$$

Para comparar podemos ver la solución obtenida mediante el método de la base de Gröbner para el ideal $I = \langle f, g \rangle$ cuya base de Gröbner reducida respecto del orden lexicográfico sería

$$\langle x - 4y^5 - y^4 + 32y^3 + 4y^2 - 64y + 16, y^6 - 8y^4 + y^3 + 16y^2 - 8y + 1 \rangle.$$

Resultante de Bézout

Es similar a la resultante de Sylvester, salvo que la definición de matriz de Bézout es un poco más dificultosa que ésta, pero a cambio ésta tiene dimensión $n \times n$ en lugar de la dimensión $(n+m) \times (n+m)$. En consecuencia, la evaluación del determinante de la matriz Bézout es mucho más rápido.

Resultante de Dixon

Es una versión generalizada de la resultante y matriz de Bézout para tres polinomios en dos variables. Entonces la resultante de Dixon se generaliza para $n+1$ polinomios en n variables.

1.3.3. El método de Wu-Ritt

En esta sección daremos una breve introducción a la teoría de este método. Éste se basa en la aproximación de Wu-Ritt para encontrar un conjunto característico para un sistema de ecuaciones no lineales. Dado un sistema de ecuaciones polinomiales $S = \{f_1, \ldots, f_m\}$ se transforma en una forma triangular S'. Es importante notar que si el número n de variables es mayor que el número de ecuaciones del conjunto S, entonces el conjunto de variables se divide en dos subconjuntos: las variables independientes, que denotaremos por $\{u_1, \ldots, u_k\}$, y las dependientes, que denotaremos por $\{y_1, \ldots, y_l\}$.

La pseudodivisión de polinomios de varias variables es la operación clave en la computación de conjuntos característicos. Para realizar la pseudodivisión se da uso de la representación recursiva de los polinomios con lo cual se define la siguiente reducción polinomial.

Un polinomio f_i se reduce respecto de otro polinomio f_j si verifican una de las dos siguientes condiciones:

1. La mayor variable de f_i es menor, con respecto a $\prec$, que la mayor variable de f_j.
2. El grado de la mayor variable en f_j es mayor que el grado de la mayor variable en f_i.

Si f_i no es reducible respecto de f_j, entonces f_i se reduce a r mediante la pseudodivisión entre f_j.

Definición 1.3.7. Dado un conjunto finito Σ de polinomios $u_1, \ldots, u_k, y_1, \ldots, y_l$ un conjunto característico Φ de Σ se define de cualquiera de las siguientes maneras:

1. $\{g_1\}$ donde g_1 es un polinomio de $\{u_1, \ldots, u_k\}$.

2. Una cadena $\langle g_1, \ldots, g_l \rangle$ donde cada g_i es un polinomio en $\{u_1, \ldots, u_k, y_1, \ldots, y_i\}$, con coeficiente líder $LC(g_i)$, tales que:

   - Cualquier cero de Σ es un cero de Φ.
   - Cualquier cero de Φ que no es cero de ninguno de los coeficientes líderes $LC(g_i)$ es un cero de Σ.

Podemos encontrar más información sobre el presente método en [Ber00], [G$^+$91a] o [G$^+$91b].

1.4. Conclusiones

Todos los métodos mencionados tienen una característica común: si hacemos uso de ellos para convertir la expresión paramétrica de un objeto a la expresión implícita de éste entonces lo que obtenemos es una sola ecuación. A partir de ahí las superficies pueden ser visualizadas con métodos de visualización directa.

Por supuesto, esto es solo un tipo de métodos de implicitación de superficies, ya que existe una gran variedad. Alguno ya ha sido mencionado al comienzo de este capítulo y otros pueden ser consultados en la bibliografía. En este tipo de métodos no es necesario conocer la expresión paramétrica del objeto.

Capítulo 2

Representación y visualización de superficies implícitas

Tras el primer capítulo donde introdujimos el concepto de superficie implícita y distintos métodos para, a partir de superficies expresadas en forma paramétrica, obtener la expresión implícita de ésta. Nos cabe preguntarnos cuáles son los métodos más básicos para poder representar las superficies en forma implícita.

Claro está que estos métodos no tienen por qué ser los mejores computacionalmente hablando, pero nos dan una primera aproximación al tema principal de este Trabajo Fin de Máster.

Por tanto en este capítulo se basará en presentarnos dos métodos de visualización: poligonalización, para lo cual explicaremos un algoritmo en detalle, y Ray Tracing.

2.1. Representación de superficies algebraicas y blobs

2.1.1. Superficies algebraicas

Una superficie se dice algebraica si se define por polinomios cuyo grado refleja el grado de la superficie. El grado indica el número de intersecciones de la superficie y una recta, véase [BW97]. Por ejemplo un plano tiene grado 1 mientras que una esfera tiene grado 2. El uso de polinomios tiene la ventaja de ser menos costoso, en tiempo de renderización, que cualquier otra representación analítica general.

Las superficies algebraicas más utilizadas son las cuadráticas. Este tipo de superficies son fáciles de renderizar, son necesarios muy pocos parámetros para controlar su forma y, además, es posible usar coordenadas homogéneas para aplicar transformaciones afines. A continuación mostramos típicas superficies cuadráticas como son la esfera, el cilindro y el cono.

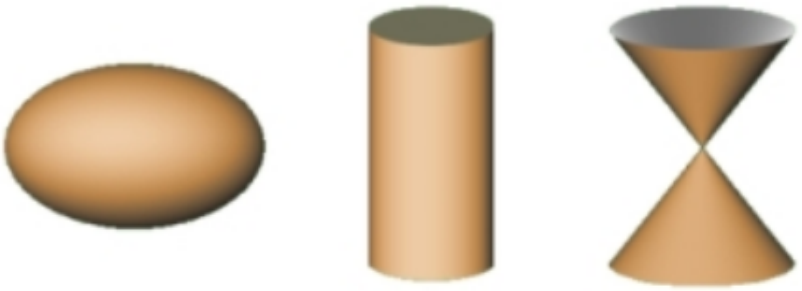

Figura 2.1: Ejemplos básicos de superficies cuadráticas.

2.1.2. Blobs

Los blobs son sumas de distribuciones gaussianas inspiradas en la distribución de densidad de las moléculas. Este tipo de superficies fueron usadas por primera vez por Blinn [Bli82] para renderizar una animación del ADN para el programa de televisión Cosmos de Carl Sagan, en donde cada átomo era aproximado por una esfera gaussiana.

La suma de esferas gaussianas genera una unión entre las superficies. Blinn propuso la función

$$f(x,y,z) = \sum_{i=1}^{n} b_i e^{-a_i r_i(x,y,z)^2} - T,$$

donde cada sumando está centrado en un término r_i y T representa el umbral. El término r_i se calcula como

$$r_i(x,y,z) = \sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2}.$$

El término b representa la altura de la función y el término a es la desviación estándar. El efecto de la función blob se puede modificar cambiando los parámetros.

La función exponencial define una esfera gaussiana que tiende a infinito a la par que la exponencial tiende a cero. Esto significa que cada esfera gaussiana influye sobre las demás sin importar la separación que exista entre ellas. Los objetos flexibles [W$^+$86] computan las esferas usando aproximaciones polinomiales.

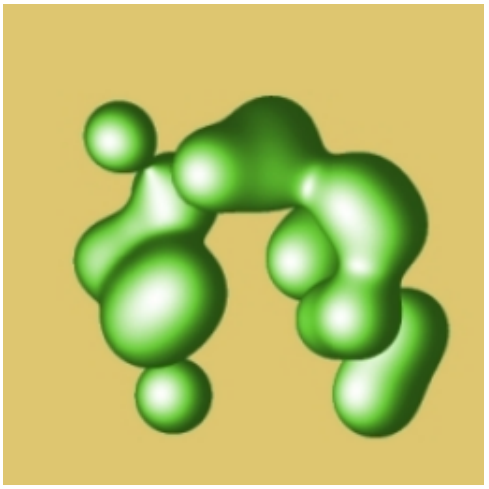

Figura 2.2: Ejemplo de sumas de esferas gaussianas.

2.2. Métodos de visualización

En esta sección cubrimos dos de los métodos de visualización usados y referenciados en la bibliografía de superficies implícitas de forma más frecuente: poligonalización y Ray Tracing. Cubriremos ambos con detalle, pero cabe destacar que en el resto del Trabajo Fin de Máster nos centraremos en el desarrollo del segundo.

2.2.1. Poligonalización de superficies implícitas

La idea general de los métodos de poligonalización consiste en la creación de polígonos que representen la superficie implícita. Los polígonos son fáciles de renderizar en los sistemas gráficos modernos, por tanto, este método suele ser el elegido cuando vamos a realizar visualización interactiva.

Existen varios métodos de poligonalización de superficies, siendo los más populares:

- El llamado *método de las celdas fijas* [BW90] consistente en dividir el espacio en poliedros de forma conveniente (cubos, tetraedros,...) y, tras esto, en calcular la intersección de la superficie con las aristas de los poliedros para así definir los vértices de la poligonalización. Estos vértices han de ordenarse para crear polígonos convexos.

  Obviamente la calidad del resultado depende en gran medida de como de *fina* sea la división del espacio, esto es, si los poliedros son demasiado grandes entonces se perderán una gran cantidad de detalles y si son demasiado pequeños crearán polígono en exceso que realentizarán la renderización.

  Este problema se puede solucionar con una serie de métodos adaptativos donde el tamaño de la celda según el detalle de la superficie, véase, si en una zona nos

interesa hacer una poligonalización más detallada allí habrá una división más fina. Aunque estos métodos son difíciles de implementar y aún no son muy populares por esta razón y las celdas de tamaño fijo suelen ser la opción más común.

- El segundo método más común son los *marching methods* que consisten en crear, de forma sucesiva, un mallado triangular comenzando con un punto o un polígono dado. Este método lo explicaremos mejor a continuación con un ejemplo claro donde mostraremos un algoritmo concreto extraído de [Har03].

2.2.2. El algoritmo de triangulación

La formulación del algoritmo de triangulación extraído de [Har03][1] no usa ninguna representación especial de la superficie para ser triangulada. Las operaciones dependientes de la representación están implícitamente en el apartado de `surfacepoint` que se define a continuación. En él presentaremos las ideas básicas del algoritmo. Después introduciremos el procedimiento `surfacepoint` y la estructura de los datos usados. Finalmente se explicaremos los pasos del algoritmo en detalle.

La idea del algoritmo

S0 Escoge un punto s cercano a la superficie. Determina el correspondiente punto p_1 de la superficie. Rodea p_1 de un hexágono regular $q_2, \ldots, q_7$ en el plano tangente. Con el procedimiento `surfacepoint` determina los puntos $p_2, \ldots, p_7$ correspondientes a los puntos iniciales $q_2, \ldots, q_7$. Ya hemos construido los primeros seis triángulos de la triangulación. Entonces al conjunto ordenado de puntos $p_2, \ldots, p_7$ lo llamaremos *polígono delantero actual* Π_0. Si la triangulación se puede limitar por curvas cerradas $\Gamma_1, \Gamma_2, \ldots$ podemos determinar los polígonos delanteros $\Pi_1, \Pi_2, \ldots$ ligados a las curvas.

S1 Para cada punto del polígono Π_0 determinamos el ángulo del área aún por triangular. A estos ángulos los llamamos *ángulos delanteros*.

S2 Revisamos si algún punto p_i de Π_0 está cerca. Con cerca entendemos:

- Un punto de Π_0 distinto de p_i y de su entorno.
- Un punto de cualquier otro polígono Π_k distinto.

En el primer caso dividimos el polígono Π_0 en dos nuevos polígonos Π_0 y Π_1. En el segundo caso unimos los dos polígonos en uno nuevo al que llamaremos Π_0 y será nuestro nuevo polígono delantero actual.

[1] Al aparecer en [Har98] los derechos pertenecen a Springer-Verlag.

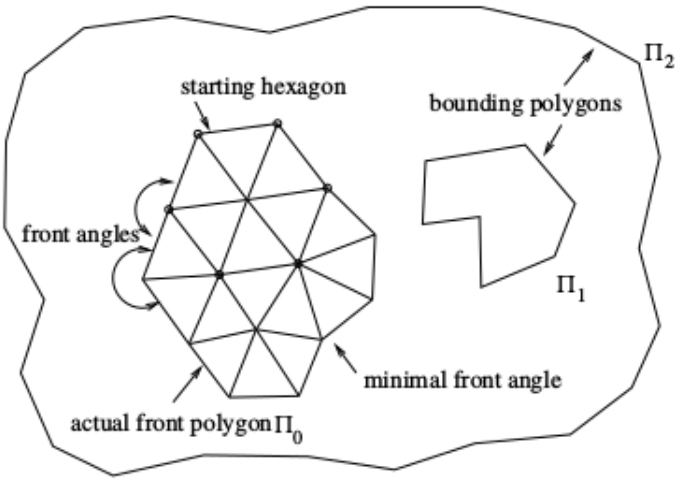

Figura 2.3: Nociones básicas del algoritmo de triangulación.

S3 Determinar un punto p_m del polinomio Π_0 con un ángulo delantero mínimo. Rodea p_m por triángulos con ángulos cercanos a $\frac{\pi}{6}$. Elimina p_m del polígono Π_0 e inserta los nuevos puntos en Π_0.

S4 Repite los pasos 1, 2 y 3 hasta que Π_0 consista sólo en tres puntos que generan un nuevo triángulo. Si aún queda algún polígono restante se convierte en el nuevo Π_0 y se repiten los pasos 1, 2 y 3. Una vez ya no queden más polígonos, habremos terminado y la triangulación estará completada.

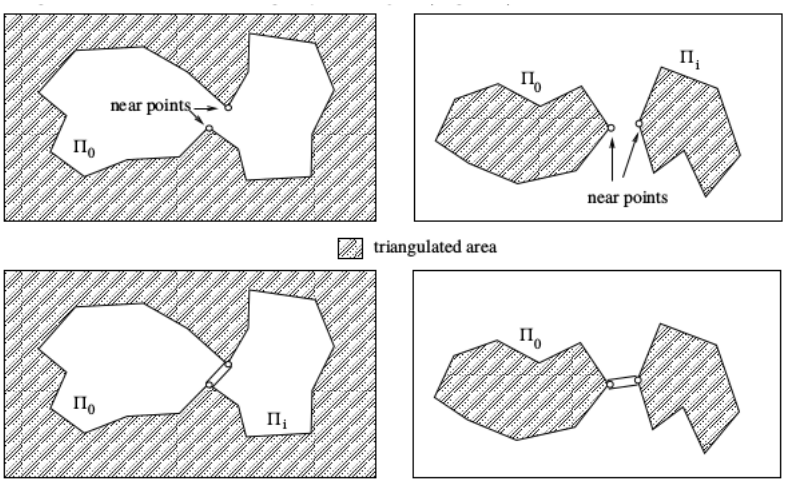

Figura 2.4: Dividiendo y uniendo el polígono Π_0.

El procedimiento `surfacepoint`

Un paso fundamental del algoritmo es determinar el punto p de la superficie que está cerca de un punto q en un entorno de la superficie. El vector $q - p$ no ha de ser necesa-

riamente perpenticular a la superficie. Debido a que casi todas las superficies pueden ser numéricamente implicitadas, daremos a solución para superficies implícitas.

Comenzamos con una superficie implícita Φ definida por una función $f(x) = 0$ para la cual su gradiente ∇f existe y no se anula para ningún punto de la superficie y un punto q en un entorno de la superficie. El siguiente procedimiento calcula un punto de la superficie p, un normal y dos vectores tangentes al punto p.

1. - $u_0 := q$.
   - Repetir $u_{k+1} := u_k - \frac{f(u_k)}{\nabla f(u_k)^2} \nabla f(u_k)$ hasta que $\|u_{k+1} - u_k\|$ es suficientemente pequeño.
   - $p := u_{k+1}$.
2. Definimos el normal a la superficie en p como $n := \frac{\nabla f(p)}{\|\nabla f(p)\|}$.
3. Para los vectores tangentes:
   - $t_1 := \frac{1}{\|\sqrt{n_x^2+n_y^2}\|}(n_y, -n_x, 0)$ si $n_x > 0.5$ ó $n_y > 0.5$.
   - En otro caso elegimos $t_1 := \frac{1}{\|\sqrt{n_x^2+n_z^2}\|}(-n_z, 0, n_x)$.
   - $t_2 := n \times t_1$.

La estructura

Para la construcción de los triángulos necesitamos un paso de longitud $\delta_t > 0$ que es aproximadamente la longitud de las aristas.

Para cada punto p_i guardamos la siguiente información:

- Las coordenadas.
- El normal y los tangentes a la superficie en p_i tales que son ortonormales entre sí.
- El ángulo delantero de p_i si es un punto delantero de Π_0.
- La variable booleana `angle_changed` con `angle_changed = true` si el ángulo delantero cambió y tiene que ser recalculado.
- La variable booleana `border_point`, con el valor `true` si el punto p_i es en el borde de la triangulación y debería ser ignorado para futuros cálculos.

Los triángulos serán numerados de forma consecutiva. Para cada triángulo guardaremos el valor numérico de sus vértices.

S0

Sea s un punto inicial en un entorno de la superficie. Entonces `surfacepoint` nos determina el primer punto p_1 de la triangulación y el sistema ortonormal n_1, t_{11} y t_{12}. El resto de puntos $p_2, \ldots, p_7$ son el resultado de aplicar `surfacepoint` a

$$q_{i+2} = p_1 + \delta_t cos\left(\frac{i\pi}{3}\right) t_{11} + \delta_t sen\left(\frac{i\pi}{3}\right) t_{12}, \qquad i = 0, \ldots, 5.$$

Ya hemos obtenido los primeros seis triángulos.

S1

Si un punto p_{0i} del polígono delantero $\Pi_0 = (p_{01}, \ldots, p_{0N_0})$ acaba de ser incluido o un punto cercano a p_{0i} es un nuevo punto, entonces es necesario recalcular el ángulo delantero ω del punto p_{0i}. Sea:

- $v_1 := p_{0,i-1}$ si $i > 1$ ó $v_1 := p_{0N_0}$ si $i = 1$.
- $v_2 := p_{0,i+1}$ si $i < N_0$ ó $v_2 := p_{01}$ si $i = N_0$.
- (ξ_1, η_1, ζ_1) y(ξ_2, η_2, ζ_2) las coordenadas de v_1 y v_2, respectivamente, en el sistema local ortonormal n, t_1 y t_2 en el punto p_{0i}.
- ω_i el ángulo polar de (ξ_i, η_i).

Entonces el ángulo delantero en el punto p_{0i} es $\omega = \omega_2 - \omega_1$ si $\omega_2 \geq \omega_1$ ó $\omega = \omega_2 - \omega_1 + 2\pi$ en caso contrario.

S2

Para prevenir el solapamiento de nuevos triángulos sobre los ya existentes comprobaremos:

- Las distancias dos a dos de los puntos que componen Π_0. Si hay puntos p_{0i} y p_{0j}, con $i < j$, que no son vecinos, ni vecinos de vecino y $\|p_{0i} - p_{0j}\| < \delta_t$ entonces Π_0 se separa en dos polígonos.
- Las distancias de puntos de Π_0 a puntos del resto de polígonos Π_k. Si hay puntos $p_{0i} \in \Pi_0$ y $p_{mj} \in \Pi_m$ tales que $\|p_{0i} - p_{mj}\| < \delta_t$, entonces los polígonos Π_0 y Π_m se unen.

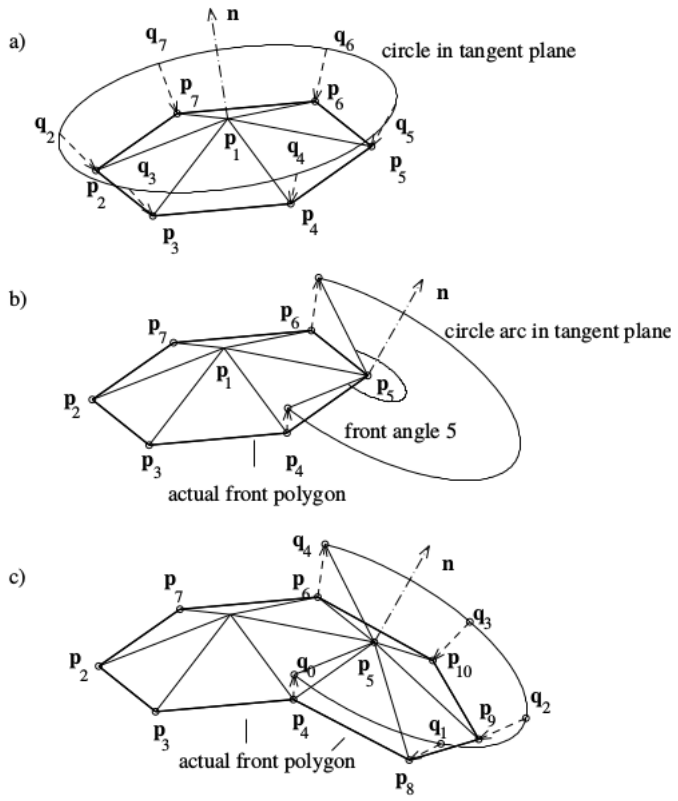

Figura 2.5: Los primeros pasos del algoritmo.

S3

Sea p_{0m} un punto de Π_0 con un ángulo delantero mínimo ω. Completamos la triangulación en p_{0m} de la siguiente manera:

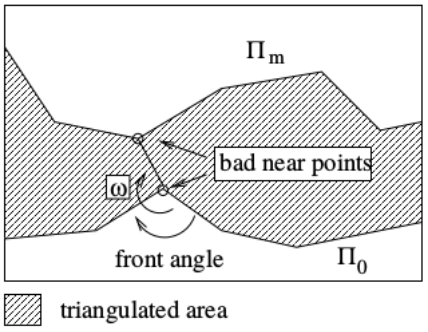

Figura 2.6: Puntos cercanos *malos* y su detección.

1. Determinamos los vecinos v_1 y v_2 de p_{0m}.

2. Determinamos el número n_t de triángulos que van a ser generados:

$$n_t := \texttt{trunc}\left(\frac{3\omega}{\pi}\right) + 1 \quad \text{y} \quad \Delta\omega := \frac{\omega}{n_t}$$

   Corrección de $\Delta\omega$ para casos extremos:

   - Si $\Delta\omega < 0.8$ y $n_t > 1$ entonces $n_t \to n_t - 1$ y $\Delta\omega = \frac{\omega}{n_t}$.
   - Si $\Delta\omega < 0.8$, $n_t = 1$ y $\|v_1 - v_2\| > \frac{5}{4}\delta_t$ entonces $n_t = 2$ y $\Delta\omega \to \frac{\Delta\omega}{2}$.
   - Si $\omega < 3$ y $\|v_1 - p_{0m}\| \leq \frac{1}{2}\delta_t$ (ó $\|v_1 - p_{0m}\| \leq \frac{1}{2}\delta_t$) entonces $n_t = 1$.

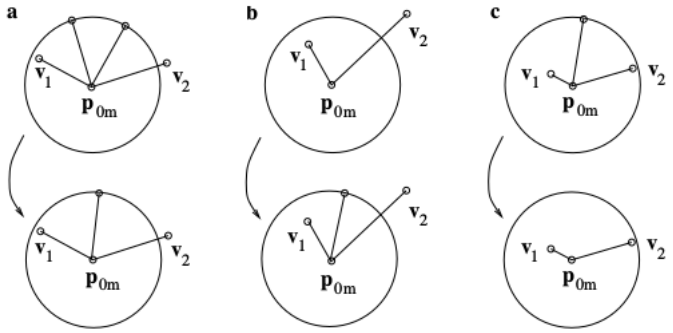

Figura 2.7: Correcciones de los casos extremos.

3. Generamos los triángulos:

   Si $n_t = 1$ entonces tenemos un nuevo triángulo (v_1, v_2, p_{0m}) en otro caso sean q_0 y q_{n_t} las proyecciones ortogonales de v_1 y v_2 en el plano tangente en el punto p_{0m} y sea q_i el resultado de una rotación de ángulo $i\Delta\omega$ alrededor del normal a la superficie en p_{0m} aplicada a $p_{0m} + \delta_t \frac{q_0 - p_{0m}}{\|q_0 - p_{0m}\|}$. Aplicando el procedimiento `surfacepoint` a q_i obtendremos nuevo puntos p_{N+i} $i = 1, \ldots, n_t - 1$, donde N es el total de puntos existentes hasta el momento, y n_t nuevos triángulos.

4. Renovamos el polígono Π_0:

   Borramos el punto p_{0m} y, si $n_t > 1$, insertamos en su posición los nuevos puntos $p_{N+1}, \ldots, p_{N+n_t-1}$. Todas las variables booleanas tiene el valor `true` para asegurarnos de que los nuevos cálculos se realizan.

Ejemplos de superficies trianguladas

Esfera

Triangulación de la esfera $x^2 + y^2 + z^2 - 4 = 0$ comenzando por el punto $(1, 1, 1)$ y paso de longitud $\delta_t = 0.3$. La siguiente imagen muestra los primeros cuatro polígonos delanteros y la situación tras 101 y 1531 triángulos. La triangulación total involucra 1534 triángulos.

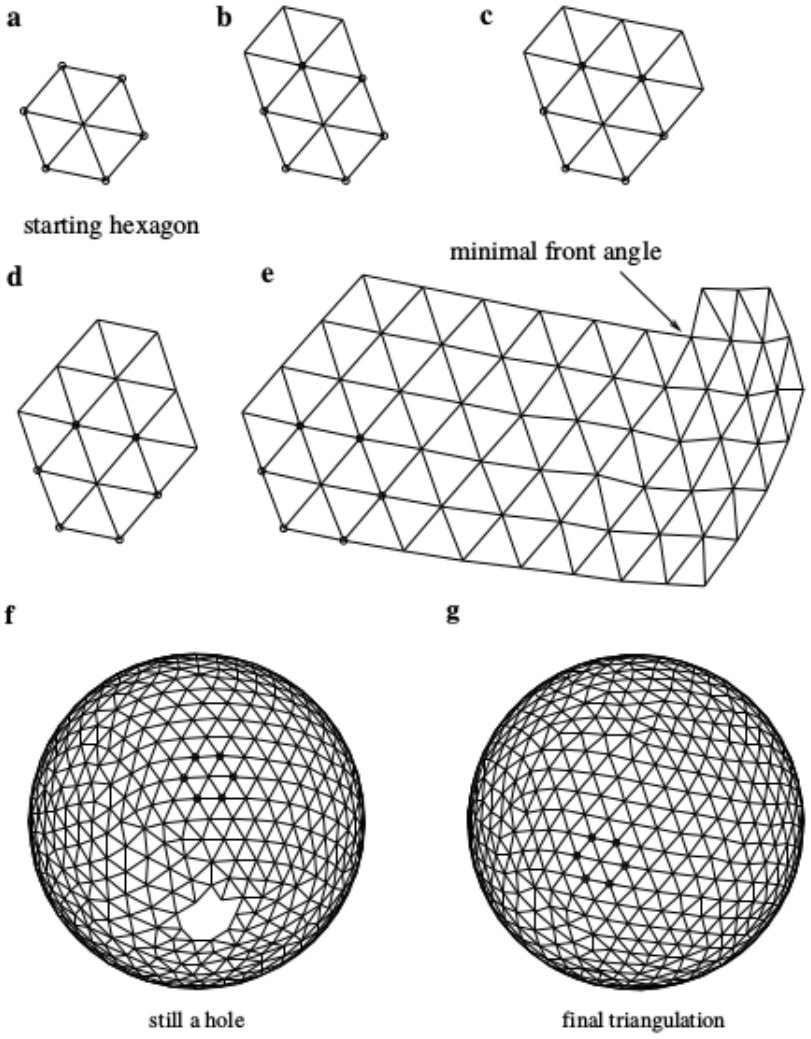

Figura 2.8: Proceso de triangulación de la esfera.

Cilindro

Triangulación del cilindro $x^2 + y^2 - 1 = 0$ comenzando por el punto $(1, 0, 0)$ y paso de longitud $\delta_t = 0.2$.

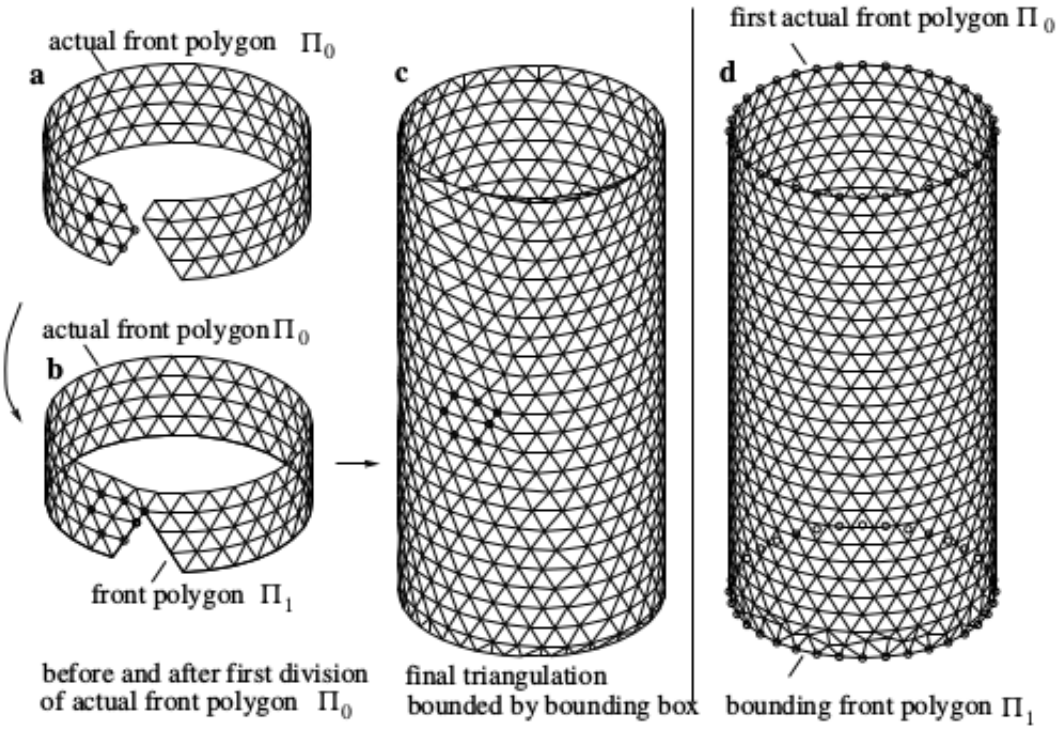

Figura 2.9: Proceso de triangulación del cilindro.

Toro

Triangulación del cilindro $(x^2 + y^2 + z^2 + 0.8775) - 4(x^2 + y^2) = 0$ comenzando por el punto $(1, 0, 0.5)$ y paso de longitud $\delta_t = 0.1$.

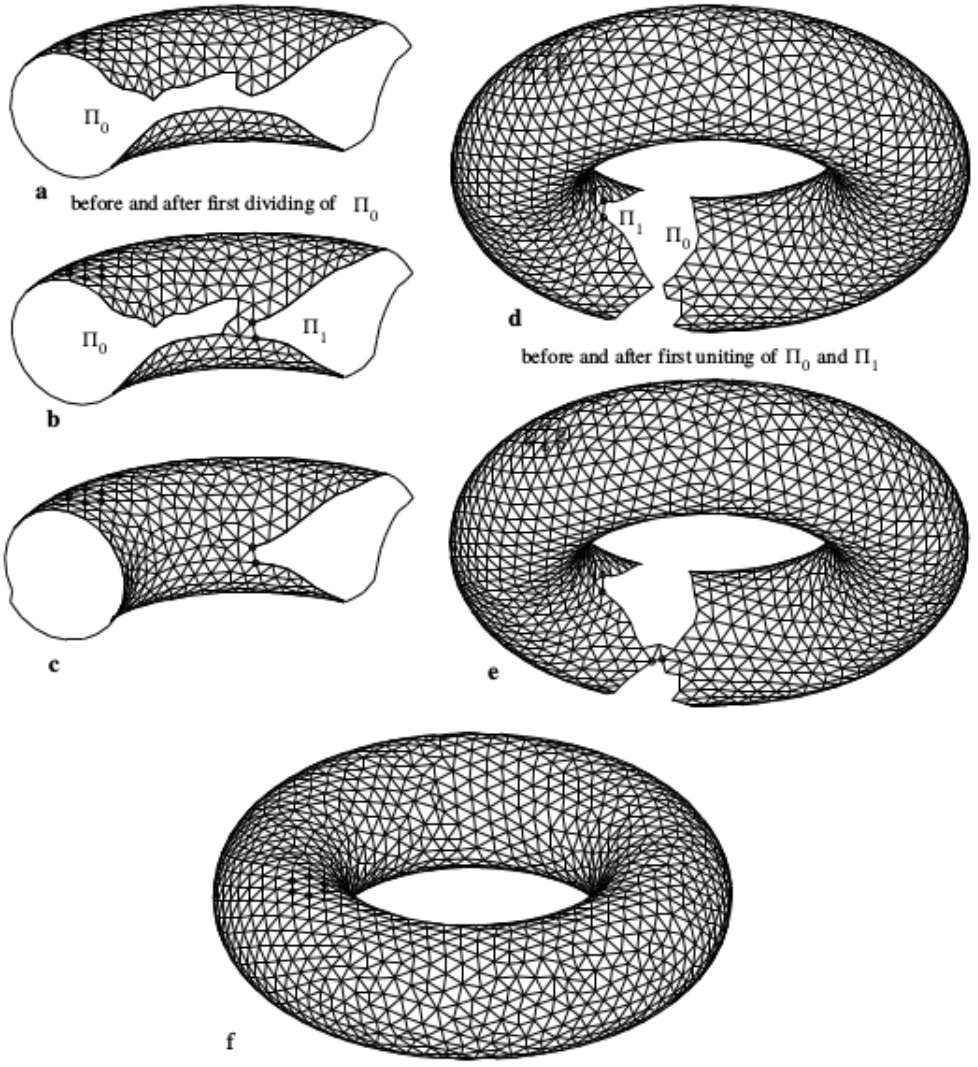

Figura 2.10: Proceso de triangulación del toro.

2.2.3. Ray Tracing en superficies implícitas

En 1980, Whitted [Whi80] propuso un método para generar imágenes de alta calidad usando modelos geométricos básicos. Este método evolucionó en lo que hoy se conoce como Ray Tracing, donde la interacción entre rayos de luz y objetos en la naturaleza es simulado: la luz, ya sea artificial o natural, rebota con los objetos y esto es lo que llega a nuestros ojos e interpretamos las propiedades de los objetos, véase color, transparencia, brillo... Además la luz indirecta puede rebotar en varios objetos antes de llegar a nuestros ojos.

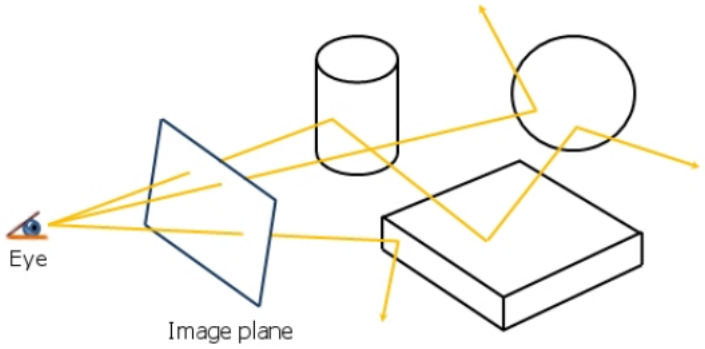

Figura 2.11: Rayos de luz rebotando en distintos objetos antes de llegar a nuestros ojos. El plano imagen en la figura se ve cruzado por los rayos; este plano puede contener una representación bidimensional de la escena tridimensional.

Usando esta técnica de renderización es posible obtener una representación realista de de distintos tipos de escenas.

En la naturaleza la luz proviene de las llamadas fuentes de luz y, tras rebotar en los objetos del medio, algunos de los rayos llegan a nuestros ojos. Por tanto, analizando el problema, podemos darnos cuenta que, computacionalmente hablando, es muy costoso intentar simular todos los rayos provenientes del foco de luz basándonos en que muchos de los rayos nunca llegarán a nuestro ojo. Por esa razón es mejor modelar el proceso inverso, esto es, trazar los rayos desde el ojo y buscar las intersecciones con los objetos del medio.

2.2.4. El algoritmo de Ray Tracing

El método del Ray Tracing es de los llamados *píxel a píxel*. Uno o más rayos de luz son trazados en cada uno de los píxeles en un plano imagen o pantalla. El objetivo es encontrar las intersecciones con los objetos del medio conforme vayamos realizando el trazado de los rayos. Generalmente se suele buscar la primera intersección ya que, en

caso de haber más de una, suelen estar obstruidas por el propio objeto.

En la figura 2.12 introducimos el proceso de construcción de un rayo. El punto c representa el origen. Un rayo que comienza en este punto se envía a través de un píxel en la pantalla con dirección $\vec{cs}$. Estos rayos tienen coordenadas con respecto al sistema uvw, mientras que el objeto tiene su propio sistema de referencia xyz. Esta independencia de los sistemas de referencia permite ver la escena desde cualquier posición arbitraria.

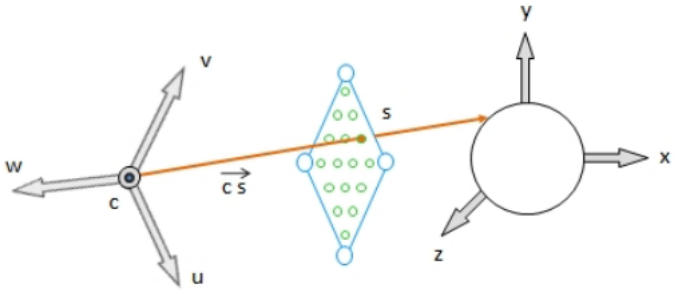

Figura 2.12: Definición de un rayo cruzando una pantalla. Si el rayo interseca alguna superficie, el color se calcula y asignado al píxel en la pantalla.

Si hay alguna intersección entre el rayo y el objeto el color es calculando tomando en cuenta la dirección del vector normal del punto de intersección y la posición de las luces. La aportación de las luces indirectas también se calcula. La suma de todas las posibles aportaciones se usa para asignar el color correspondiente en el píxel.

Análisis de la intersección

Un rayo se define como

$$\begin{aligned} x &= c_x + t(x_s - c_x), \\ y &= c_y + t(y_s - c_y), \\ z &= c_z + t(y_z - c_z). \end{aligned}$$

Donde (c_x, c_y, c_z) es origen o punto de vista, (x_s, y_s, z_s) es el punto donde el rayo cruza la pantalla y t es el parámetro del rayo.

La intersección entre la superficie implícita $f(x, y, z) = 0$ y el rayo viene definida por la ecuación

$$g(t) = 0,$$

donde $g(t)$ es la función $g : \mathbb{R}_0 \to \mathbb{R}^3$ definida por

$$g(t) \equiv f(c_x + t(x_s - c_x), c_y + t(y_s - c_y), c_z + t(y_z - c_z)).$$

De este modo, el problema se reduce a encontrar las raíces de la función g.

Estas soluciones se sustituyen en la definición paramétrica del rayo para encontrar el punto de intersección y el valor del vector normal.

Existen muchas maneras de encontrar las raíces de g. Las manera más clásicas son el método de la bisección o de Newton [Har01]. También existen de otro tipo como: técnicas fuzzy [FBB96], aritmética de intervalos [Mit90] o constantes de Lipschitz [KB89].

2.2.5. Ray Tracing eficiente

Como ya hemos comentado el principal problema del algoritmo de Ray Tracing es la falta de eficiencia. Muchos autores han propuesto diferentes técnicas de mejora de la eficiencia pero se pueden clasificar en tres grupos:

- Intersecciones más rápidas.
- Trazar menor cantidad de rayos.
- Generalización de los rayos.

La técnica más básica para acelerar el Ray Tracing es el llamado *volumen delimitador o límite*,que consiste en crear un volumen que contenga a nuestro objeto y que el cálculo de la intersección de los rayos con éste sea menos costoso que con el objeto en sí. La primera propuesta de volumen delimitador fue la esfera [Whi80] por la simplicidad y por la facilidad para probar la intersección de los rayos con ésta. La idea es recubrir cada objeto con una esfera, si un rayo interseca a una esfera, entonces se analiza el caso dentro de la esfera, en caso contrario no es necesario. Si además añadimos lo que se suele llamar *jerarquía* el orden del algoritmo se puede reducir a $log(n)$.

Otra técnica bastante común en la literatura especializada es la subdivisión del espacio en la cual el espacio que rodea a nuestro objeto se divide y se descartan las subdivisiones que no contienen parte del objeto alguno. Podemos dividir el espacio de manera regular o de manera adaptativa como con la poligonalización, pero conlleva los mismos inconvenientes.
La ventaja que tiene este método es que, al ordenar las divisiones, sólo hay que probar el rayo para cada una de éstas en lugar de para el espacio completo para buscar la primera intersección.

Ya como final tenemos las llamadas técnicas direccionales en la que se mejora la técnica anterior discretizando también las direcciones.

2.3. Conclusiones

En este capítulo hemos visto distintas maneras de representar y visualizar superficies expresadas de forma implícita, de las cuales ya hemos nombrado sus ventajas e inconvenientes. En términos de coste computacional y velocidad de renderización éstas no presentan ningún reto y presentan una serie de propiedades que las hacen muy atractivas.

En el siguiente capítulo haremos una introducción al Análisis de Intervalos para mejorar la capacidad de computación y tratar de solventar los problemas de redondeo de las máquinas debido a los puntos flotantes.

Capítulo 3

Análisis de intervalos

El Análisis de Intervalos es una rama de las Matemáticas que lidia con los problemas de redondeo debido al uso de la aritmética de coma flotante. Los ordenadores tienen registros de coma flotante para representar los números reales. Como es bien sabido, hay números reales que no tienen representación finita, por esa razón estos números tienen que ser redondeados.
Este tipo de situación crea problemas de imprecisión numérica que se puede propagar y acumular a los largo de los algoritmos, en especial en aquellos que son recursivos.
Aunque es posible trabajar con un gran tamaño de bits para representar a los números, este conjunto de números es, de hecho, una representación digital que, obviamente, no tiene las mismas propiedades que el conjunto de números reales.
Un ejemplo muy básico puede mostrarnos este problema:

- a = `random()`
- b = `random()`
- c = a + b
- c = c - a - b

Por lógica de cómo trabajamos con el conjunto de números reales se esperaría que el valor final de *c* sea cero, pero este puede no ser el caso dependiendo del programa de cálculo que tomemos en un ordenador convencional.

Hay una gran cantidad de áreas de investigación en las cuales en Análisis de Intervalos ha sido aplicado para mantener la precisión numérica: Ingeniería de control y supervisión, modelización geométrica y diseño por ordenador.
Aparte también hay una serie de *catástrofes* documentadas que podrían haber sido evitadas usando Análisis de Intervalos. Por ejemplo:

- Un misil Patriot falló debido a la acumulación de errores de redondeo.

- La explosión del Arianne 5 causada por un error de exceso.

En este capítulo presentaremos la nociones, operaciones y propiedades básicas del Análisis de Intervalos. Además también haremos una introducción al Análisis de Intervalos Modal que completa la definición clásica de Análisis de Intervalos.

3.1. Planteamiento

La computación numérica de problemas teóricos en un ordenador requiere que los números reales $\mathbb{R}$ sean representados con una cantidad limitada de cifras decimales. Por supuesto, es posible usar un conjunto de números suficientemente grande de números con una gran cantidad de cifras decimales, pero aún así habrá números reales que no podrán ser representados.
Esto significa que, cuando traducimos un problema teórico a un problema computacional, estamos trabajando con un conjunto de *números digitales*, llámese DI, también conocidos como números en coma flotante.

Los números reales dan soporte a aquellos modelos donde las magnitudes continuas están presentes. Sin embargo, como sabemos, los ordenadores trabajan con truncamiento de números reales. Por tanto, es posible que parte de la información se pierda. Las operaciones deberían restringirse a un intervalo obtenido por medias de redondeo, lo que nos da una identificación operacional de los valores calculados.

Dados dos valores reales $\underline{a}$ y $\overline{a}$, un intervalo A se define como

$$A = [\underline{a}, \overline{a}] = \{x \in \mathbb{R} : \underline{a} \leq x \leq \overline{a}\},$$

donde $\underline{a}$ y $\overline{a}$ se conocen como el ínfimo y el supremo del intervalo respectivamente.

Para mantener la representación exacta del número, debemos realizar redondeo al número digital más próximo. Este redondeo es necesario para que la definición anterior sea válida. El uso del Análisis de Intervalos con números digitales nos proporciona un control automático de los errores de redondeo.

En la construcción de intervalos, denotados por $I(\mathbb{R})$, muchas de las propiedades de los números reales se pierden, véase la propiedad distributiva, y se obtienen otras como la relación de inclusión.
Los ordenadores trabajan con el conjunto DI, por esa razón el conjunto de intervalos con base en DI, denotado por $I(DI)$, es el usado para las operaciones algorítmicas. Esto significa que, mientras $I(DI)$ admita operaciones sobre el conjunto DI para ser tratadas por ordenador, se puede establecer un modelo analítico de operaciones y relaciones de los intervalos análogo al de los intervalos $I(\mathbb{R})$.

Como los intervalos que trataremos a a partir de ahora no son intervalos de números reales significa que los algoritmos que aplicamos a éstos deberán ser modificados para adaptarse a la nueva aritmética.

3.2. Relaciones entre intervalos

Las relaciones entre los intervalos son equivalentes a algunas relaciones entre los límites del intervalo. Aunque la definición de las relaciones a veces no son evidentes, hay una norma general de tomar la definición más útil en cada caso. Lo importante es obtener relaciones parecidas a los intervalos de números reales.

Definición 3.2.1. Se dice que dos intervalos A y B son iguales, denotado por $A = B$, si para todo $a \in A$ existe $b \in B$ tal que $a = b$ y viceversa. En función de los límites:

$$A = B \iff \underline{a} = \underline{b} \text{ y } \overline{a} = \overline{b}.$$

Definición 3.2.2. Se dice que dados dos intervalos, A y B, $A < B$ si para todo $a \in A$ y $b \in B$ se tiene que $a < b$. En función de los límites:

$$A < B \text{ si y sólo si } \overline{a} < \underline{b}.$$

Definición 3.2.3. Se dice que dados dos intervalos, A y B, $A \leq B$ si para todo $a \in A$ existe $b \in B$ tal que $a \leq b$ y para todo $b \in B$ existe $a \in A$ tal que $a \leq b$. En función de los límites:

$$A \leq B \iff \underline{a} \leq \underline{b} \text{ y } \overline{a} \leq \overline{b}.$$

Definición 3.2.4. Se dice que dados dos intervalos, A y B, $A \subset B$ si para todo $a \in A$ se tiene que $a \in B$. En función de los límites:

$$A \subset B \iff \underline{a} \geq \underline{b} \text{ y } \overline{a} \leq \overline{b}.$$

Definición 3.2.5. Se dice que dados dos intervalos, A y B, A es incidente en B, $A =\!\!\!|\!| B$, si $A \cap B \neq \emptyset$. En función de los límites:

$$A =\!\!\!|\!| B \iff \text{máx}(\underline{a}, \underline{b}) \leq \text{mín}(\overline{a}, \overline{b}).$$

3.3. Operaciones de Aritmética de Intervalos

Las operaciones de los intervalos se basan en la Teoría de Conjuntos. De esta manera es posible definir las operaciones mediante medias de los límites de los intervalos.

Hay ciertas condiciones que deben cumplir las operaciones:

- El resultado de una operación entre intervalos ha de ser otro intervalo.

- La restricción de una operación de intervalos entre dos intervalos particulares debe coincidir con la misma operación realizada entre intervalos reales.
- Todas las operaciones entre elementos particulares de ambos intervalos deben de estar contenidas en el intervalo final. Esto es conocido como el Principio de Inclusión.

De acuerdo a estas condiciones, las operaciones se definen por la expresión

$$AwB = \{awb : a \in A, b \in B\}, \tag{3.1}$$

donde w representa cualquiera de las operaciones.

La ecuación general de las operaciones de Aritmética de Intervalos es

$$AwB = [\text{mín}(\underline{a}w\underline{b}, \underline{a}w\overline{b}, \overline{a}w\underline{b}, \overline{a}w\overline{b}), \text{máx}(\underline{a}w\underline{b}, \underline{a}w\overline{b}, \overline{a}w\underline{b}, \overline{a}w\overline{b})]$$

Por tanto, las cuatro operaciones básicas se definen como

$$\begin{aligned}
&[\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}],\\
&[\underline{a}, \overline{a}] - [\underline{b}, \overline{b}] = [\underline{a} - \overline{b}, \overline{a} - \underline{b}],\\
&[\underline{a}, \overline{a}] * [\underline{b}, \overline{b}] = [min(\underline{ab}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}), \text{máx}(\underline{ab}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b})],\\
&[\underline{a}, \overline{a}] / [\underline{b}, \overline{b}] = [\underline{a}, \overline{a}] * [\tfrac{1}{\overline{b}}, \tfrac{1}{\underline{b}}] \text{ si } 0 \notin [\underline{b}, \overline{b}].
\end{aligned}$$

3.4. Intervalos modales

El Análisis de Intervalos Modales, MIA por sus siglas en inglés, es un complemento lógico del Análisis de Intervalos clásico que incluye herramientas para resolver incertidumbre cuantificada. Para alcanzar este objetivo, la versión clásica se asocia con un cuantificador.

Un intervalo modal se define como un par (I, Q) donde I es un intervalo clásico y Q es un cuantificador modal, véase $\forall$ o $\exists$ llamados universal y existencial, respectivamente. El conjunto de intervalos modales se representa por $I^*(\mathbb{R})$. Si el intervalo modal se asocia con un cuantificador existencial se llama *propio* y en caso de que esté asociado con un cuantificador universal se llama *impropio*. La representación canónica de un intervalo modal es:

- Intervalo propio: $X = [a, b] = ([a, b]', \exists)$ si $a \leq b$.
- Intervalo impropio: $X = [a, b] = ([b, a]', \forall)$ si $a \geq b$.
- Intervalo puntual: $X = [a, b] = ([a, b]', \{\exists, \forall\})$ si $a = b$.

Donde la notación $'$ indica un intervalo clásico.

Nota. Un intervalo puntual puede ser considerado tanto propio como impropio.

El proceso de construcción de intervalos modales se completa con el concepto de cuantificador modal Q definido por

$$Q(x,X)P(x) := \begin{cases} (\exists x \in X')P(x) & X = (X', \exists), \\ \\ (\forall x \in X')P(x) & X = (X', \forall), \end{cases}$$

donde $P(x)$ es un predicado con variable libre x. Esto define el conjunto de predicados reales aceptados por un intervalo modal $A = (A', Q_A)$ como

$$Pred((A', Q_A)) := \{P(.) \in Pred(\mathbb{R}) : Q(x, (A', Q_A))P(x)\}.$$

donde $Pred(\mathbb{R})$ es el conjunto de predicados con variable real.

Esta definición del cuantificador modal Q nos obliga a introducir un cambio en la notación clásica para los cuantificadores. En lo sucesivo usaremos:

- $\exists(x, X')$ en lugar de $\exists x \in X'$.
- $\forall(x, X')$ en lugar de $\forall x \in X'$.

A través de la identificación de un intervalo modal con el conjunto de tales predicados reales en los que aceptamos $X \iff P(X)$, surge la inclusión de dos intervalos como la inclusión del conjunto de predicados que aceptan, esto es, si $X, Y \in I^*(\mathbb{R})$ se tiene

$$X \subset Y \iff Pred(X) \subset Pred(Y).$$

Usando las coordenadas canónicas $X = [x_1, x_2]$ e $Y = [y_1, y_2]$ esta inclusión mantiene el mismo *modus operandi* tradicional, esto es,

$$[x_1, x_2] \subset [y_1, y_2] \iff y_1 \leq x_1 \text{ y } x_2 \leq y_2.$$

La siguiente figura nos muestra la representación geométrica de los intervalos modales y la relación de inclusión.

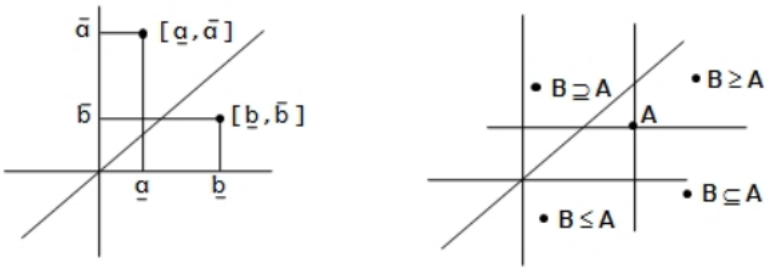

Figura 3.1: La primera imagen nos muestra la representación de los intervalos modales y la segunda las inclusiones y las desigualdades.

Las operaciones *meet* y *join* en $I^*(\mathbb{R})$ para una familia acotada de intervalos modales $A(I) := \{A(i) = [a_1(i), a_2(i)] \in I^*(\mathbb{R}) : i \in I\}$, donde I es el el dominio de índices, se definen por

$$\wedge\,(i, I)A(i) = A \in I^*(\mathbb{R}) \text{ es tal que } \forall (i, I)X \subset A(i) \iff X \subset A,$$

$$\vee\,(i, I)A(i) = B \in I^*(\mathbb{R}) \text{ es tal que } \forall (i, I)X \supset A(i) \iff X \supset B.$$

notado como $(A \wedge B)$ y $(A \vee B)$ para el caso correspondiente. El resultado, visto como función de los límites de los intervalos, es

$$\bigwedge_{i\in I} A(i) = [\max_{i\in I} a_1(i), \min_{i\in I} a_2(i)],$$

$$\bigvee_{i\in I} A(i) = [\min_{i\in I} a_1(i), \max_{i\in I} a_2(i)].$$

Con estas operaciones el conjunto de intervalos modales es un retículo para la $\subset$-relación, mientras que los intervalos clásicos no lo son, por tanto, estamos completando el conjunto de los intervalos clásicos.

Ambos operadores son isotónicos, i.e., si $A_i \subset B_i$ para cada $i \in I$, entonces

$$\bigwedge_{i\in I} A_i \subset \bigwedge_{i\in I} B_i,$$

$$\bigvee_{i\in I} A_i \subset \bigvee_{i\in I} B_i.$$

En el conjunto de los números conocemos que hay dos relaciones $\leq$ y $\geq$, y las extensiones de estas relaciones a los intervalos se definen por

$$[x_1, x_2] \leq [y_1, y_2] \iff x_i \leq y_i \quad i = 1, 2.$$

Lo que nos conduce a los operadores mín y máx para una familia acotada de intervalos modales $A(I) := \{A(i) \in I^*(\mathbb{R}) : i \in I\}$ como:

$$\min_{i\in I} A(i) = A \in I^*(\mathbb{R}) \text{ es tal que } \forall (i, I)X \leq A(i) \iff X \leq A$$

$$\max_{i\in I} A(i) = B \in I^*(\mathbb{R}) \text{ es tal que } \forall(i, I)X \geq A(i) \iff X \geq B$$

Computacionalmente expresado:

$$\min_{i\in I} A(i) = [\min_{i\in I} a_1(i), \min_{i\in I} a_2(i)]$$
$$\max_{i\in I} A(i) = [\max_{i\in I} a_1(i), \max_{i\in I} a_2(i)]$$

El conjunto de los intervalos modales es, por tanto, un retículo bajo la $\leq$-relación. La siguiente figura nos muestra la representación geométrica de los operadores que hemos definido para dos intervalos.

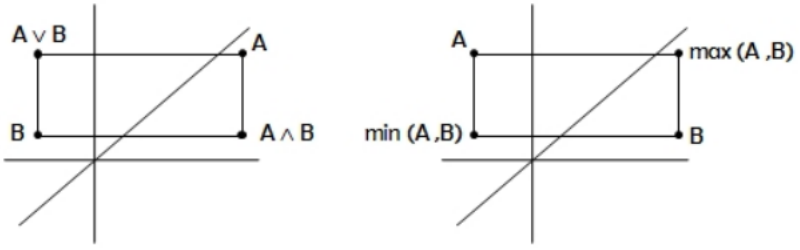

Figura 3.2: Representación de los operadores meet, join, máx y mín.

3.4.1. Extensiones semánticas

En la teoría clásica de Análisis de Intervalos una extensión de una función de $\mathbb{R}^n$ a $\mathbb{R}$ dada por $z = f(x_1, \ldots, x_n)$ es el intervalo de extensión unida R_f de f. Para el intervalo $X' = (X'_1, \ldots, X'_n) \in I(\mathbb{R}^n)$ se define el rango de f-valores en X' como

$$R_f(X'_1, \ldots, X'_n) := \{f(x_1, \ldots, x_n) : x_i \in X'_i \; \forall i \in \{1, \ldots, n\}\} =$$

$$[\min\{f(x_1, \ldots, x_n) : x_i \in X'_i \; \forall i \in \{1, \ldots, n\}\}, \max\{f(x_1, \ldots, x_n) : x_i \in X'_i \; \forall i \in \{1, \ldots, n\}\}].$$

Para obtener la estimación para la extensión unida, las extensiones racionales del intervalo teórico $fR(X'_1, \ldots, X'_n)$ se definen como su correspondiente función real-racional $f(x_1, \ldots, x_n)$ reemplazando:

1. Los argumentos numéricos x_i por sus argumentos de intervalos X'_i.
2. Los operadores aritméticos *reales* ω por su correspondiente operación entre intervalos la cual, en los casos más comunes de computaciones truncadas de cualquier aritmética tiene que dirigirse al exterior ω^R debido a la inclusión

   $$X'\omega Y' \subset X'\omega^R Y' := Out(X'\omega Y'),$$

   donde *Out* representa el redondeo exterior del intervalo $X'\omega Y'$.

Las funciones de intervalos racionales tienen la propiedad, fundamental para todo el cuerpo de Análisis de Intervalos, de ser inclusiva, esto es, para un conjunto de intervalos cumpliendo $X'_1 \subset Y'_1, \ldots, X'_n \subset Y'_n$ se cumple

$$fR(X'_1, \ldots, X'_n) \subset fR(Y'_1, \ldots, Y'_n).$$

Suponiendo que no ocurre ninguna división entre un intervalo que contenga al cero.

La relación entre ambas extensiones es

$$R_f(X'_1, \ldots, X'_n) \subset fR(X'_1, \ldots, X'_n).$$

Donde fR es computable a parir de los límites de los intervalos $X'_1, \ldots, X'_n$ y normalmente representan una sobrestimación de $R_f(X'_1, \ldots, X'_n)$.

En el Análisis de Intervalos Modales, un rol similar al de R_f se ve cubierto de forma semántica por las funciones * y **, denotadas por f^* y f^{**} (funciones estrella y doble estrella) y definidas por:

$$f^*(X) := \bigvee_{x_p \in X'_p} \bigwedge_{x_i \in X'_i} [f(x_p, x_i), f(x_p, x_i)] = \left[\min_{x_p \in X'_p} \max_{x_i \in X'_i} f(x_p, x_i), \max_{x_p \in X'_p} \min_{x_i \in X'_i} f(x_p, x_i) \right]$$

$$f^{**}(X) := \bigwedge_{x_i \in X'_i} \bigvee_{x_p \in X'_p} [f(x_p, x_i), f(x_p, x_i)] = \left[\max_{x_p \in X'_p} \min_{x_i \in X'_i} f(x_p, x_i), \min_{x_p \in X'_p} \max_{x_i \in X'_i} f(x_p, x_i) \right]$$

Las cuales tienen, por supuesto, la propiedad de inclusión $f^*(X) \subset f^{**}(X)$. Además

$$X \subset Y \Rightarrow f^*(X) \subset f^*(Y) \text{ y } f^{**}(X) \subset f^{**}(Y).$$

En algunos casos puede ocurrir que $f^* \equiv f^{**}$.

Los próximos teoremas nos dan una interpretación lógica de las extensiones semánticas.

Teorema 3.4.1. Teorema de la semántica *: Sea $X \in I^*(\mathbb{R}^n)$ y $Z \in I^*(, \mathbb{R})$, se tiene

$$f^*(X) \subset Z \iff \forall(x_p, X'_p)\ Q(z, Z)\ \exists(x_i, X'_i)\ z = f(x_p, x_i).$$

Teorema 3.4.2. Teorema de la semántica **: Sea $X \in I^*(\mathbb{R}^n)$ y $Z \in I^*(, \mathbb{R})$, se tiene

$$f^*(X) \supset Z \iff \forall(x_i, X'_i)\ Q(z, Dual(Z))\ \exists(x_p, X'_p)\ z = f(x_p, x_i).$$

Esto significa que es posible reducir una expresión lógica a inclusiones entre intervalos. Ambos teoremas hacen equivalente una fórmula lógica, con intervalos y predicados funcionales donde los cuantificadores universales preceden a los existenciales, a una inclusión de intervalos.

Por ejemplo, la función real $f(x,y) = x + y$ con los intervalos $X = [1,3]$ e $Y = [2,3]$ el resultado es $X + Y = [4,5]$. De acuerdo al teorema de la semántica * obtenemos

$$\forall(x,[1,3]')\exists(z,[4,5]')\exists(y,[2,3]')x + y = z.$$

Y de acuerdo al teorema de la semántica ** obtenemos

$$\forall(y,[2,3]')\exists(z,[4,5]')\exists(x,[1,3]')x + y = z.$$

Incluso pensando que las funciones f^* y f^{**} son óptimas en el sentido semántico, estos teoremas no explicitan el proceso de computación del intervalo Z que verifica $f^* \subset Z$ ó $f^{**} \supset Z$, i.e., intervalos que son una estimación exterior e interior de f^* y f^{**} respectivamente. De hecho, exceptuando los operadores aritméticos, el cálculo de f^* y f^{**} no se puede lograr mediante computación directa. Si la función continua f es una función racional, entonces existe una extensión modal racional que se obtiene usando el programa computacional definido por el árbol sintáctico de las expresiones de la función:

- Si f es una función racional de $\mathbb{R}^n$ a $\mathbb{R}$, entonces su extensión modal racional a los intervalos modales $X_1, \ldots, X_n$ representada por $f\ R(X_1, \ldots, X_n)$ es la función $f\ R$ de $I^*(\mathbb{R}^n)$ a $I^*(\mathbb{R})$ definida por el programa computacional indicado por la sintaxis de f cuando el operador ral se transforma en su extensión semántica.

Las funciones de intervalos modales racionales no son interpretables, pero tiene la propiedad de ser isotónicas, i. e., para intervalos $X_1 \subset Y_1, \ldots, X_n \subset Y_n$ se verifica la relación

$$f\ R(X_1, \ldots, X_n) \subset f\ R(Y_1, \ldots, Y_n).$$

Asumiendo que no se producen divisiones por intervalos que contengan al cero.

3.4.2. Interpretabilidad y optimalidad

La solución al problema de computar las extensiones semánticas f^* y f^{**} consiste en relacionarlo mediante relaciones de inclusión a algunas extensiones racionales. Las computaciones con $f\ R(X)$ deben de hacerse con truncación externa de cada operador para obtener $f^*(X) \subset f\ R(X)$ y con la truncación interna obtenemos $f\ R(X) \subset f^{**}(X)$. En muchos de los casos la extensión racional $f\ R(X)$ es óptima, i. e.,

$$f^*(X) = f\ R(X) = f^{**}(X).$$

Y, exceptuando el redondeo, ambos teoremas semánticos son aplicables al intervalo $f\ R(X)$ proporcionándole un significado lógico a éste.

El Análisis de Intervalos Modales nos proporciona una serie de resultados sobre inclusiones o igualdades que resuelven parte del problema doble de interpretabilidad de

extensiones modales racionales y computabilidad de las expresiones semánticas. Ahora mostraremos varios teoremas que nos muestran resultados sobre la interpretabilidad de extensiones racionales.

Teorema 3.4.3. * interpretabilidad de funciones modales racionales: Si las componentes impropias de X son uni-incidentes[1] en $f\ R(X)$ y si $Out(f\ R(Prop(X)))$ existe, entonces

$$Out(f\ R(X)) \supset f^*(X).$$

Donde Out representa el redondeo exterior del intervalo $f\ R(X)$.

Teorema 3.4.4. ** interpretabilidad de las funciones modales racionales: Si las componentes propias de X son uni-incidentes en $fR(X)$ y si $Out(f\ R(Prop(X)))$, entonces

$$Inn(f\ R(X)) \subset f^{**}(X).$$

Donde Inn representa el redondeo interno del intervalo $f\ R(X)$.

Una función real f se llama x-totalmente monótona para una variable multi-incidente $x \in \mathbb{R}$ si es uniformemente monótona para esta variable y para cada una de sus incidencias, consideradas como variables independientes.

Teorema 3.4.5. * interpretabilidad con monotonía total: Sea X un vector intervalo y $f\ R$ definido en $Prop(X)$ y totalmente monótona para un subconjunto Z de componentes multi-incidentes. Sea $X\ Dt^*$ el vector agrandado de X, esto es, cada incidencia de cada componente multi-incidente del subconjunto con monotonía total está incluida en $X\ Dt^*$ como una componente independiente, pero transformada en su dual si el correspondiente punto incidente tiene un sentido monótono contrario al global de la correspondiente componente Z. Para el resto, las componentes impropias muti-indicentes se transforman en un intervalo puntual definido por cualquiera de sus puntos. Entonces

$$f^*(X) \subset f\ R(X\ Dt^*).$$

Teorema 3.4.6. ** interpretabilidad con monotonía total: Sea X un vector intervalo y $f\ R$ definido en $Prop(X)$ y totalmente monótona para un subconjunto Z de componentes multi-incidentes. Sea $X\ Dt^{**}$ el vector agrandado de X, esto es, cada incidencia de cada componente multi-incidente del subconjunto con monotonía total está incluida en $X\ Dt^{**}$ como una componente independiente, pero transformada en su dual si el correspondiente punto incidente tiene un sentido monótono contrario al global de la correspondiente componente Z. Para el resto, las componentes propias muti-indicentes se transforman en un intervalo puntual definido por cualquiera de sus puntos. Entonces

$$f^{**}(X) \supset f\ R(X\ Dt^{**}).$$

[1]Una componente x_i de una variable x se dice *uni-incidente* en una función continua real f si ocupa una hoja del árbol sintáctico del árbol de f. En cualquier otro caso se dice *multi-incidente*. Por ejemplo en la función $f : \mathbb{R}^2 \to \mathbb{R}$, definida por $f(x_1, x_2) = x_2 + \frac{x_1}{x_2}$, x_1 es uni-incidente y x_2 es multi-incidente.

Teorema 3.4.7. Interpretabilidad en el caso general multi-incidente: Si $f\ R(X)$ tiene componentes multi-incidentes impropias y Xt^* se obtiene reemplazando tales componentes por intervalos puntual definido por cualesquiera de sus puntos de sus dominios, entonces

$$f^*(X) \subset f\ R(X\ t^*).$$

Este teorema es útil cuando no es posible realizar test de monotonía.

Es posible obtener mejores resultados si se aplica el concepto de optimalidad.

Definición 3.4.1. Una función modal racional $f\ R(X)$ se dice óptima si cualquiera de sus operadores no uniformemente monótonos es seguido por operadores de una única variable.

Teorema 3.4.8. Coerción a la optimalidad: Sean X, $f\ R$ y $X\ D$ definidos bajo las condiciones de los teoremas 3.4.5 y 3.4.6. Y sea $f\ R$ óptimo en $Prop(X)$. En tal caso

$$f^*(X) = f\ R(X\ D) = f^{**}(X).$$

Este teorema es útil para resolver el problema, especialmente en el caso cuando la función de la fórmula lógica verifica las condiciones de optimalidad ya que, en este caso, la computación de $f\ R(X\ D)$ es igual a $f^*(X)$, excepto por el redondeo, y el teorema 3.4.1 lo hace equivalente a la fórmula lógica.

Ejemplo:

Consideremos una función continua $f : \mathbb{R} \to \mathbb{R}$ definida por

$$f(x, y) = \frac{xy}{x + y},$$

con $X = [2, 3]$ e $Y = [4, 3]$.

La función f es totalmente monótona respecto de x y de y ya que sus derivadas parciales son mayores que cero en los dominios de las variables. Tomando las componentes multi-incidentes como componentes independientes, es decir,

$$f(x_1, x_2, y_1, y_2) = \frac{x_1 y_1}{x_2 + y_2},$$

se tiene que:

$$\frac{\partial f}{\partial x_1} > 0, \quad \frac{\partial f}{\partial x_2} < 0, \quad \frac{\partial f}{\partial y_1} > 0 \quad \text{y} \quad \frac{\partial f}{\partial y_2} > 0.$$

Con estas condiciones mostraremos las aplicaciones de tres teoremas:

1. De acuerdo al teorema 3.4.7 las componentes impropias multi-incidentes se reemplazan por intervalos puntual, $Xt^* = ([2, 3], [4, 4], [2, 3], [4, 4])$ el resultado es

$$f\ R(Xt^*) = \frac{X\ [4, 4]}{X + [4, 4]} = [1.1428, 2].$$

2. De acuerdo al teorema 3.4.5 tenemos $X\ Dt^* = ([2,3],[4,4],[3,2],[4,4])$ y el resultado es
$$f\ R(X\ Dt^*) = \frac{X\ [4,4]}{Dual(X) + [4,4]} = [1.3333, 1.7144].$$

3. De acuerdo al teorema 3.4.8 tenemos $X\ D = ([2,3],[4,3],[3,2],[3,4])$ y el resultado es
$$f^*(X) = \frac{XY}{Dual(X) + Dual(Y)} = [1.3333, 1.5].$$

La mejor aproximación se obtiene mediante el teorema 3.4.8. En el lado contrario tenemos la aproximación obtenida mediante el teorema 3.4.7. En este ejemplo es

$$f^*(X) \subset f\ R(X\ Dt^*) \subset f\ R(Xt^*).$$

3.5. Conclusiones

En este capítulo hemos definido e introducido las diferentes propiedades del Análisis de Intervalos. Como hemos explicado, la Aritmética de Intervalos se puede aplicar a diversos campos de investigación para solucionar los problemas de redondeo. Por supuesto las gráficas por ordenador no iban a ser una excepción.

También hay una visión del Análisis de Intervalos Modales, que completa la definición clásica de Análisis de Intervalos mediante aplicación de los cuantificadores a la definición de intervalo. La teoría de Intervalos Modales nos da los intervalos impropios que en la teoría clásica carece de sentido y, en la mayoría de casos, la transformación lleva tal intervalo impropio en uno propio sin explicación lógica. El Análisis de Intervalos Modales nos concede una explicación para tales casos por medio de las aplicaciones de los teoremas incluidos en este capítulo.

Los teoremas desarrollados son la base para la teoría de Intervalos Modales y, además, nos ayudarán a mejorar el Ray Tracing, o trazado, de superficies implícitas.

Capítulo 4

Aplicaciones del Análisis de Intervalos a las superficies implícitas

Como ya hemos explicado en el capítulo anterior, el Análisis de Intervalos es una herramienta bastante fiable con respecto a los problemas de redondeo de los ordenadores. Además tiene aplicaciones en muchas áreas como:

- Gráficas por ordenador.
- Detección de colisiones.
- Errores de aproximación en la transferencia de datos entre sistemas CAD/CAM.
- Ray Tracing de superficies paramétricas.
- Ray Tracing de superficies implícitas.

El Análisis de Intervalos ha sido usado en Infografía en especial para la creación de algoritmos de subdivisión fidedignos. Estos algoritmos pueden evaluar áreas o espacio para detectar la *existencia* de superficies o curvas. Esto permite la creación de estructuras como árboles octales o cuaternarios, octrees y quadtrees en inglés de forma respectiva.

Este capítulo comienza con la introducción de diferentes algoritmos de subdivisión de intervalos. A continuación presentamos diferentes técnicas para desarrollar Ray Tracing, centrándonos en las basadas en Aritmética de Intervalos. Finalmente se incluye una comparación entre aproximaciones basadas en intervalos para encontrar las raíces de las intersecciones para conocer la eficiencia.

4.1. Algoritmos de subdivisión recursiva

Existen infinidad de técnicas para representar objetos geométricos como representaciones poligonales, técnicas de subdivisión espaciales, parches paramétricos... El proceso de visualización mediante Análisis de Intervalos conbina características de de todos las técnicas nombradas, aunque la más común suele ser la subdivisión del espacio. Éste es usado para representar objetos geométricamente y luego aplicar estrategias de renderizado como el Ray Tracing.

Los métodos de subdivisión del espacio tienen como propósito *encerrar* al objeto. Esto es, por ejemplo en el método de los árboles octales, si una región cúbica encierra un objeto entonces podemos dividir el cubo en ocho octantes iguales. Cada nuevo cubo, u octante, es evaluado para asegurarnos que contiene alguna parte del objeto. Este proceso continua hasta un nivel predefinido de subdivisión.
Este proceso crea una estructura octal con información sobre el objeto que puede ser utilizada para crear una visualización directa o acelerar otras técnicas de visualización.

4.1.1. Subdivisión espacial usando Aritmética de Intervalos

El algoritmo de a subdivisión se basa en la subdivisión recursiva del espacio que contiene al objeto para generar ocho cubos llamados octantes.

Un octante consiste en una región cúbica definida por tres intervalos, en el que cada uno de ellos representa los valores de frontera del octante en cada una de las dimensiones. Los octantes que no contienen ningún punto de la superficie son descartados.

Este proceso permite la creación de estructuras que describan el objeto de forma geométrica. La creación de estructuras como los árboles octales es directa. Un árbol octal es un tipo de estructura que representa el espacio ocupado por varios objetos en una escena tridimensional. La versión bidimensional es llamado árbol cuaternario.

En conclusión estamos ante un tipo de árbol en el que cada nodo representa regiones y sus hojas son regiones aún más pequeñas que contienen parte de los objetos

Es posible usar este algoritmo para representar operaciones entre diferentes funciones implícitas. Por ejemplo, Suffern et al. [SB03] introdujo una técnica para renderizar la intersección entre dos superficies implícitas basada en un árbol octal. Esta técnica usaba Aritmética de Intervalos para descartar regiones que no contenían alguna de ambas superficies. Si la región contiene a ambas superficies se subdivide y se repite el mismo análisis.

Los octantes que contienen alguna parte de la superficie se subdividen en otras ocho regiones que se evalúan para saber si siguen conteniendo alguna parte de la superficie.

Dada una superficie implícita, definida por $f(x, y, z) = 0$, y una región cúbica definida por tres intervalos X, Y y Z. Usaremos la extensión unida de f definida en el capítulo anterior para saber si la superficie interseca a la región cúbica. A esta extensión la denotaremos por $F(X, Y, Z)$.

El algoritmo funciona como sigue: Si $0 \in F(X, Y, Z)$ significa que la región interseca a al superficie, en caso contrario, la región puede descartarse.

Las regiones del espacio que contienen parte de la superficie se subdividen de forma recursiva y se evalúan de nuevo hasta que alcanzamos el tamaño de mallado deseado, el cual habremos seleccionado en base a la resolución que necesitemos. Al final del algoritmo tendremos una lista de cubos que contienen partes de la superficie, pero parte de esos cubos contendrán regiones que no son solución a nuestro problema. El algoritmo de subdivisión usando funciones inclusión sería el siguiente:

```
Evaluate(X,Y,Z):
   If(0 belongs to F(X,Y,Z))
      If(X or Y or Z <= Threshold)
         Add (X,Y,Z) to solution list
      Else
         Subdivide X into X_1 and X_2
         Subdivide Y into Y_1 and Y_2
         Subdivide Z into Z_1 and Z_2
         Evaluate (X_i,Y_j,Z_k) for i,j,k in {1,2}
   Else
      The octant is rejected
```

Este algoritmo tiene muchas debilidades. La principal es que el algoritmo es muy dependiente del número de subdivisiones requeridas para alcanzar la resolución deseada, lo que provoca un gran coste computacional.

Aunque este algoritmo es muy útil en el caso tridimensional se puede usar para casos bidimensionales e incluso para rasterizar curvas algebraicas, véase [dOdF00, Tau94].

4.1.2. Mejorando el proceso de subdivisión

Existen muchas técnicas para mejorar el proceso de subdivisión. El principal objetivo de tales técnicas es permitir el diseño de algoritmos de subdivisión más eficientes y robustos.

Una versión adaptativa del algoritmo trabaja con la curvatura de la superficie durante el propio proceso de subdivisión. Balsys et al. [BS01] desarrollaron un algoritmo adaptativo para mejorar la creación de la estructura del árbol octal. Esto soluciona los problemas

surgidos por la diferencias de profundidad entre nodos adyacentes.

En [dOdF00, SB03] las técnicas adaptativas se usan para garantizar que las subdivisiones son lo suficientemente pequeñas como para contener partes de una curva con una curvatura relativamente grande. Esto se consigue evaluando el gradiente de la curva en cada subdivisión: un gradiente muy grande significa que la curva varía mucho dentro de la región en la que estamos trabajando. Por tanto tenemos que generar cubos pequeños para representar la curvatura.

Carvalho et al. [C+98] proporcionó ciertas condiciones para detener el proceso de subdivisión recursiva cuando las celdas son lo suficientemente pequeñas y no contienen intersecciones cerradas de la curva.

Bühler [Bü02] desarrolló una estimación lineal implícita basada en intervalos, ILIE por su siglas en inglés, que consiste en un cierre lineal de un objeto adaptado a la topología intrínseca del mismo. Cada celda se reduce a partes que sólo contienen la correspondiente ILIE, lo que reduce el número de subdivisiones del proceso total. Esto, por supuesto, reduce el tiempo de computación.

Standder y Hart [SH97] nos muestran cómo los puntos críticos de la función afectan a la topología de la superficie implícita. Usaron la Aritmética de Intervalos para encontrar tales puntos y, tras esto, modificaron la poligonalización para acomodarla a los cambios topológicos.

Duff [Duf92] desarrolló una aplicación geométrica de los intervalos usando árboles en donde las hojas son funciones implícitas. El algoritmo tiene en cuenta la información del árbol completo en cada subdivisión, obteniendo así una escena completa con varios objetos correctamente renderizados en la misma escena.

4.2. Ray Tracing eficiente aplicado a superficies implícitas

En esta sección abarcaremos distintas mejoras desarrolladas por distintos autores para realizar un Ray Tracing eficiente en superficies implícitas. Estos trabajos se centran en la creación de intersecciones fidedignas. Los autores aplican distintas técnicas numéricas para controlar la pérdida de raíces debido al uso de algoritmos basados en números de coma flotante. Esto puede generar que ciertas partes de las superficies más *finas* puedan perderse durante el proceso de renderización. Aunque este tipo de superficies son casos especiales sería deseable obtener un método que también fuera capaz de cubrir estos casos.

Otro problema relacionado con el Ray Tracing es la eficiencia de los algoritmos. Whitted [Whi80] menciona que el 95 por ciento del gasto de tiempo de renderizado era debido a las intersecciones en escenas complejas. El Ray Tracing es lento debido a que necesita

de muchas intersecciones para cada píxel. Esto se incrementa en las superficies implícitas e incluso más con la Aritmética de Intervalos.

4.2.1. Aproximación por muestreo puntual

Los test de intersección se pueden desarrollar en dos pasos:

1. Encontrar la raíz, en donde el primer intervalo que la contiene es seleccionado.
2. Refinamiento de la raíz, en donde el intervalo que la contiene se reduce hasta llegar a un tamaño prefijado.

La clave en el primer paso es garantizar que el intervalo contiene una única raíz de la función implícita. Este paso normalmente es más complicado y puede resolverse con métodos como el de la bisección. Sin embargo, este método puede causar problemas como la convergencia a la raíz equivocada o, directamente, la no convergencia.

```
Bisection(t_1,t_2):
   t_m = (t_1 + t_2)/2
   If(f(t_m) == 0)
      t_m is a root
   Else
      If(abs(t_2 - t_1) < Threshold)
         There are no roots
      Elif(f(t_1)*f(t_m) < 0)
         Bisection(t_1,t_m)
      Else
         Bisection(t_m,t_2)
```

Una alternativa al método de la bisección es el método de Newton, el cual tiene la ventaja de que, en caso de converger, lo hace de forma cuadrática en lugar de lineal como el método de la bisección. Sin embargo, el método de Newton puede no converger a ninguna de las raíces o, en algunos casos, divergir [Har01].

Un ejemplo de los problemas del muestreo puntual es el de la imagen 4.1. Si los intervalos iniciales son demasiado grandes, imagen a), se pierde mucha información y, por tanto, raíces. Si usamos un tamaño más pequeño de intervalos el problema no se resuelve del todo, como se puede ver en b). Además, intervalos pequeños significan pérdida de eficiencia. El problema se soluciona usando Aritmética de Intervalos, casos c) y d). Incluso usando una precisión pequeña en el método de la bisección, caso c), el resultado es mejor que en los anteriores. El mejor resultado posible sería el d) que es cuando el método de la bisección alcanza la precisión de máquina.

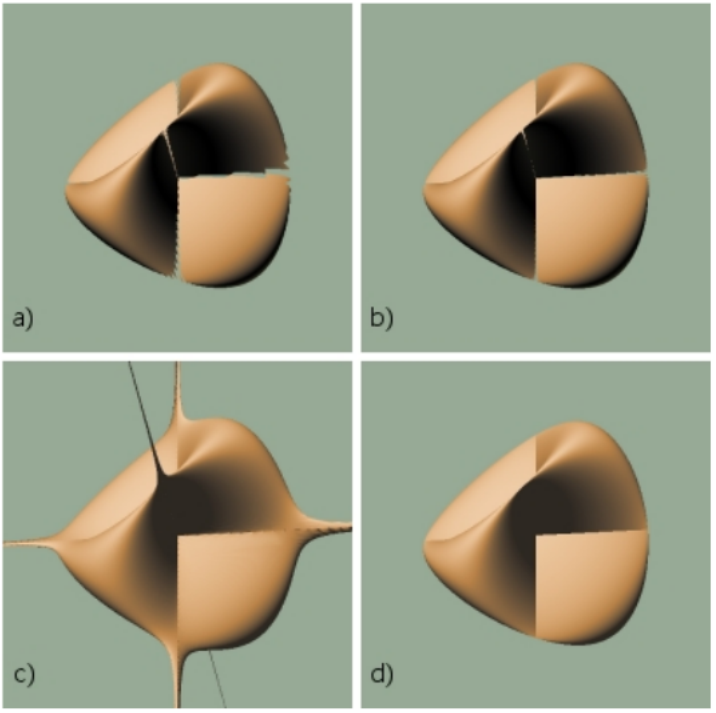

Figura 4.1: Imágenes obtenidas por muestreo puntual. a) Sin Aritmética de Intervalos e intervalo de tamaño 0.001. b) Sin Aritmética de Intervalos e intervalo de tamaño 0.0001. c) Con Aritmética de Intervalos y bisección con precisión 0.0001. d) Con Aritmética de Intervalos y bisección con precisión máquina.

4.2.2. Aproximación por Aritmética de Intervalos

Mitchell [Mit90] propuso el uso de Aritmética de Intervalos para encontrar la raíz de una forma segura. La extensión unida de la función intersección f es

$$F(T) = f(c_x + T(x_s - c_x), c_y + T(y_s - c_y), c_z + T(y_z - c_z)).$$

Los valores c_x, c_y y c_z indica el punto de vista del observador y los valores x_s, y_s y z_s indican el punto donde el rayo interseca la pantalla y T es el parámetro del intervalo. La diferencia con la definición clásica es el uso del parámetro del intervalo T, que toma intervalos como valores en lugar de puntos reales.

La ecuación anterior se conoce por *función inclusión* para la superficie implícita. Esta función puede ser evaluada usando cualquier intervalo de T. Si el resultado de la evaluación es $0 \notin F(T)$ podemos asegurar que no hay ninguna raíz de f en el valor actual de T. Aún así esta función no consigue el resultado contrario, esto es, para conocer si existen raíces para cada valor de T. Esto es debido a que en cada operación los límites del intervalo se redondean al alza y a la baja para garantizar que cualquier resultado posible no se pierda. Este redondeo incluye valores que no son parte de la evaluación de la función, por ello el cero puede ser incluido mediante cualquiera de las operaciones de redondeo.

Por tanto, la función inclusión se usa como función descarte.

Para saber si hay una única raíz en el intervalo usamos la extensión unida F' de la derivada f'. El algoritmo propuesto por Mitchell es el siguiente:

```
Mitchell(T as [t_1,t_2]):
   If(0 not in F(T))
      If()
         If(f(t_1)*f(t_2) <= 0)
            Root refinement over T using Bisection or Newthon method
      Else
         T_1 = [t_1,(t_1 + t_2)/2]
         T_2 = [(t_1 + t_2)/2,t_1]

         If(width(T_1) >= threshold)
            Mitchell(T_1)
         Else
            Root refinement over T_1 using Bisection or Newthon method

         If(width(T_2) >= threshold)
            Mitchell(T_2)
         Else
            Root refinement over T_2 using Bisection or Newthon method
   Else
      Reject T
```

El algoritmo comienza evaluando el intervalo T usando la función inclusión. En el caso de que el cero esté contenido en el resultado, la derivada de la función implícita es se evalúa usando el valor del intervalo de T. Si el cero no pertenece al resultado de la evaluación de la derivada se comprueba que los valores de los extremos del intervalo sean de signo contrario y, en caso de darse, sabemos que sólo hay una sola raíz en el intervalo T. Por tanto podríamos realizar el proceso de refinamiento del intervalo.
En el caso de que al evaluar la función inclusión y la derivada el cero esté contenido, subdividimos el intervalo T en dos y repetimos el proceso con los nuevos intervalos.

Otra consideración a tener en cuenta es la anchura que a la que los intervalos pueden llegar a alcanzar antes de finalizar la búsqueda de la raíz y comenzar el refinamiento del intervalo. Por ejemplo, cuando las raíces se encuentran en la tangente, la convergencia será siempre hacia estas raíces. En tal caso, la derivada es siempre cero y el proceso siempre termina cuando se alcanza un valor mínimo de anchura.

En otros casos no hay garantía de que el mínimo de anchura permitido contenga al-

guna raíz. En tal caso se puede establecer el mínimo en la precisión del ordenador, lo que incrementa significativamente las probabilidades de que contenga una raíz [C$^+$00].

Mitchell no usó redondeo en sus operaciones entre intervalos. Consideraba que el uso de coma flotante en la aritmética proporcionaba suficiente precisión para la renderización de las superficies. La razón que esgrimía era que la Aritmética de Intervalos era que aumentaba el tiempo de renderización.

Finalmente, este algoritmo propone resolver el refinamiento del intervalo usando técnicas clásicas como Íos métodos de bisección, Newton o cualquier otro suficientemente rápido para renderizar las superficies. Esto es debido a que, según Mitchell, La Aritmética de Intervalos es demasiado lenta para realizar este proceso.

4.2.3. Mejorando la aproximación por Aritmética de Intervalos

Si el algoritmo para encontrar la raíz se desarrolla usando la bisección clásica de intervalos, mismo algoritmo que en la sección anterior pero sin la verificación de la derivada, la eficiencia del algoritmo puede bajar drásticamente.

Capriani et al. [C$^+$00] nos muestran que hay algoritmos basados en intervalos para encontrar raíces eficaces y suficientemente rápidos como para renderizar una superficie implícita.

```
Newton(T as [t_1,t_2]):
   If(0 in F(T))
      If(0 not in F'(T))
         t_m = t_1 + midpoint(T)
         NT = t_m - f(t_m)/F'(T)
         NTT = NT intersecting with T

         If(NTT is empty)
            There's no root
         Else
            Newton(NTT)

      Else
         T_1 = [t_1,midpoint(T)]
         T_2 = [midpoint(T), t_2]

         If(width(T_1) >= Threshold)
            Newton(T_1)
         Else
```

```
            T_1 is the root

         If(width(T_2) >= Threshold)
            Newton(T_2)
         Else
            T_2 is the root
   Else
      Reject T
```

El método de intervalos de Newton es una aproximación muy buena que es más rápida que la versión clásica de la bisección. El método de intervalos de Newton tiene a menudo convergencia cuadrática, pero requiere del cálculo de las derivadas.

Capriani et al. propusieron una importante mejora del método mediante un operador Alefeld-Hansen. El operador es obtenido cuando $0 \in F'(T)$. Si $T = [t_1, t_2]$ se define

$$\frac{1}{F'(T)} = \begin{cases} \left[\frac{1}{t_2}, \infty\right] & \text{si } t_1 = 0, \\ \left[-\infty, \frac{1}{t_1}\right] & \text{si } t_2 = 0, \\ \left[-\infty, \frac{1}{t_1}\right] \cup \left[\frac{1}{t_2}, \infty\right] & \text{Cualquier otro caso.} \end{cases}$$

Véase [Ale70, Han78].

Usando aritmética extendida [Han03] el operador es

$$AH(T) = \left[midpoint(T) - \frac{f(midpoint(T))}{F'(T)}\right] \cap T.$$

El operador de Alefeld-Hansen se puede añadir al método de intervalos de Newton para comprobar los casos en los que $0 \in F'(T)$. Como se dice en [C$^+$00], el resultado de la evaluación de este operador puede ser el conjunto vacío, un sólo intervalo o dos intervalos de anchura máxima la mitad del intervalo de partida. Esto significa que el operador tiene convergencia cuadrática en la mayoría de casos. Cuando el operador devuelve dos intervalos ambos deben de ser usados para evaluar la función de forma recursiva. El algoritmo es el siguiente:

```
AH(T as [t_1,t_2]):
   If(0 in F(T))
      If(0 not in F'(T))
         t_m = t_1 + midpoint(T)
         NT = t_m - f(t_m)/F'(T)
         NTT = NT intersecting with T
```

```
      If(NTT is empty)
         There's no root
      Else
         AH(NTT)

   Elif(0 not in f(t_m))
      AH(T) = (t_m - f(t_m)/F'(T)) intersecting T
      If(AH(T) = emptyset)
         There's no root
      Else
         If(AH(T) generates one interval T_1)
            AH(T_1)
         Elif(AH(T) generates two intervals T_1 and T_2)
            AH(T_1)
            AH(T_2)

   Else
      T_1 = [t_1,midpoint(T)]
      T_2 = [midpoint(T),T_2]

      If(width(T_1) >= Threshold)
         Newton(T_1)
      Else
         T_1 is the root

      If(width(T_2) >= Threshold)
         Newton(T_2)
      Else
         T_2 is the root

Else
   Reject T
```

Los algoritmos previos requerían del uso de derivadas de la función. Esto es una desventaja cuando tenemos que renderizar funciones no derivables. En estos casos podemos usar el método de la bisección clásica es una alternativa que funciona bien a pesar de la falta de eficiencia.

Estrada et al. [SE+03] propusieron un algoritmo para optimizar la estrategia base del método de intervalos de la bisección. Básicamente proponen una serie de condiciones que

tienen que alcanzarse durante el proceso de bisección. Sostenían que, dado un intervalo $T = [t_1, t_2]$ y su punto medio t_m, si $\overline{F([t_1, t_1]) * F([t_m, t_m])} \leq 0$ entonces el intervalo $[t_m, t_2]$ se rechaza. Este algoritmo se llama MRF.

En otro algoritmo llamado MRFro [SE+03] introducen algunas condiciones para mejorar el método de la bisección. Este algoritmo evalúa $F([t_1, t_2])$ solo cuando

$$\underline{F([t_1, t_1]) * F([t_2, t_2])} > 0,$$

ya que $\overline{([t_1, t_1]) * F([t_2, t_2])} \leq 0$ te asegura que $0 \in F([t_1, t_2])$, lo cual no es verdad si tenemos en cuenta el redondeo. El redondeo puede introducir valores que verifiquen la ecuación, pero eso no es verdad para los valores originales del intervalo. Sin embargo, esta suposición fue suficiente para las superficies renderizadas en el trabajo de Estrada et al.

4.2.4. Comparación de diferentes aproximaciones por intervalos

Como hemos podido ver en las secciones anteriores hay diferentes maneras de realizar aproximaciones mediante intervalos que nos garanticen cierto nivel de fiabilidad en las intersecciones mediante Ray Tracing. Esta sección cubre una comparación entre la eficiencia de distintos métodos. Las superficies utilizadas en las pruebas están presentes en la figura 4.2. Estas superficies fueron renderizadas mediante rayos primarios usando uno por píxel.

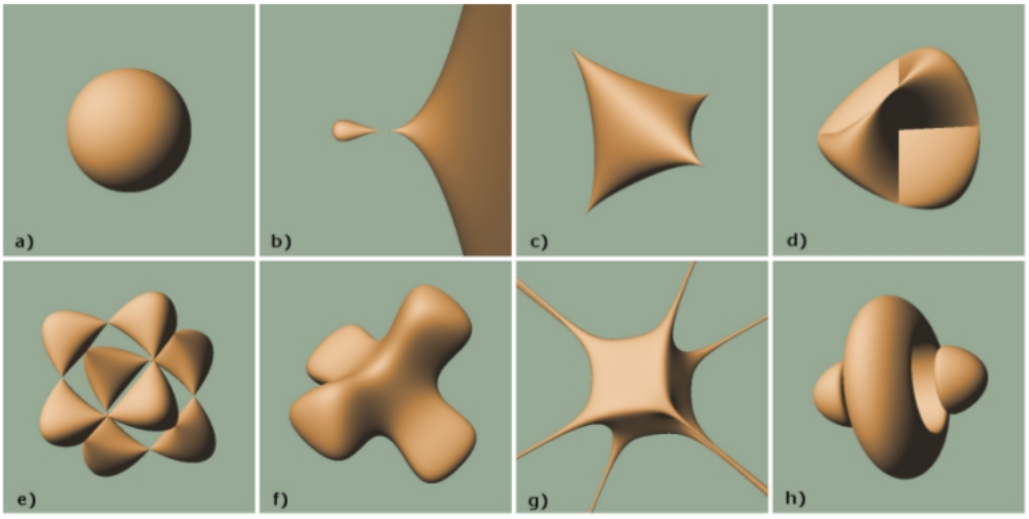

Figura 4.2: Superficies usadas para comparar distintos métodos basados en intervalos para realizar Ray Tracing en superficies implícitas. a) Esfera. b) Superficie *gota*. c) Kusner-Schmitt. d) Crosscap. e) Superficie Chubs. f) Modelo K3 McMullen g) Cubo astado. h) Toro Gumdrop.

Como vemos en los resultados que extrae [Flo08], cinco han sido los métodos analizados: el algoritmo propuesto por Mitchell, el método de Newton basado en intervalos, el método de Newton usando el operador de Alefeld-Hansen, el algoritmo MRFro y un algoritmo que combina las condiciones del MRFro y las del método de Newton basado en intervalos.

Nota. Los resultados presentados por [Flo08] se basan en imágenes de 300x300 píxeles de resolución y fueron renderizadas en un ordenador con un procesador Pentium 4.

	Sphere	Drop	Chubs	Crossc.	Gumd. Torus	H. cube	McMul. K3	Kusn.-Schm.
Mitchell	20	36	98	80	240	99	102	75
Newton	28	47	94	96	260	180	123	94
N.+Al.-Han.	27	46	92	99	202	166	110	86
MRFro	48	103	188	105	541	272	276	210
MRFro+Newt.	55	96	174	151	380	283	221	173

Figura 4.3: Comparación de diferentes métodos.

Obviamente los resultados son mejores para la esfera en todos los métodos y peores para una superficie compleja como el toro Gumdrop. El algoritmo de Mitchell muestra una mejor eficiencia para una superficie tan sencilla como es la esfera, pero cuando la complicación va aumentando el mejor método es el de Newton+AH. Además el método de Newton también mejora relativamente la velocidad, pero no lo suficiente. Esto nos indica dos cosas:

- Las comprobaciones extra que se necesitan en el operador de Alefeld-Jansen se compensa con la mejor convergencia en superficies complejas.
- En el caso de superficies simples, como la esfera, es mejor aplicar métodos simples como el de Mitchell.

El algoritmo MRFro es el más lento en todos y esto es debido a que no toma información extra de la superficie como puede ser la derivada. Cuando se añade información extra, como la subdivisión de Newton, el algoritmo mejora su eficiencia en superficies complejas pero no para superficies simples.

La imagen 4.4 nos muestra gráficamente los resultados de la tabla 4.3. Las superficies fueron intencionalmente ordenadas de menos a más computacionalmente costosa. Esto indica que la complejidad se incrementa de izquierda a derecha. Además un mayor tiempo indica menor eficiencia y viceversa.

Los algoritmos de Mitchell, Newton y Alefeld-Hansen tiene una eficiencia casi lineal cuando la complejidad crece. Además, la eficiencia es parecida en estos tres métodos que usan la derivada como fuente de información.

Por otro lado, el hecho de que tengan una eficiencia tan parecida nos hace darnos cuenta de que podemos usar operaciones con intervalos en todo el proceso sin afectar al tiempo de computación.

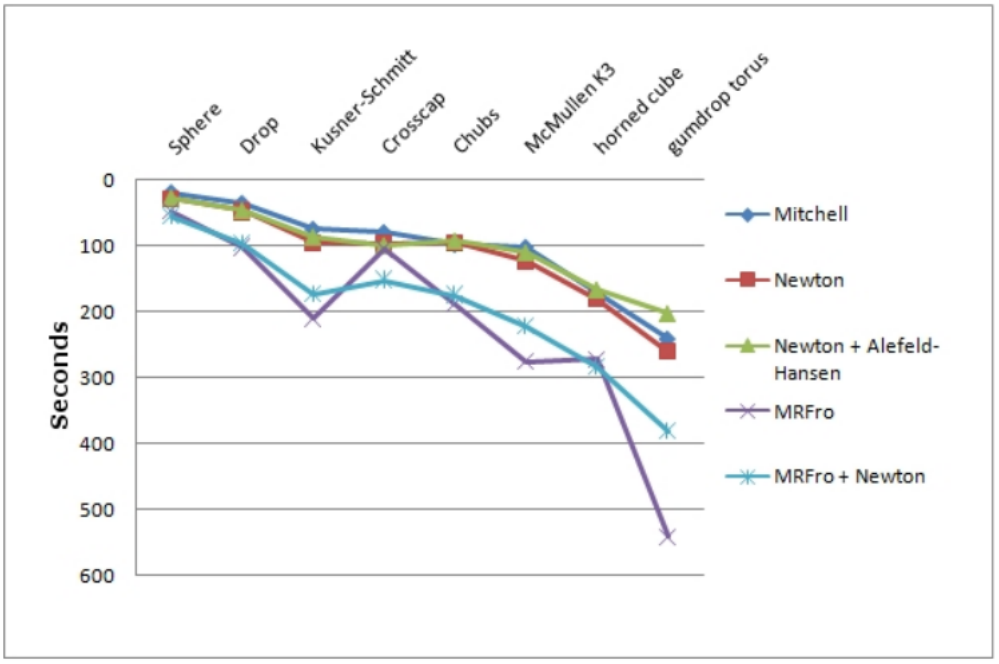

Figura 4.4: Resultados de la comparación de diferentes métodos de Ray Tracing para las superficies analizadas. Las diferencias en calidad de renderización no son apreciables entre los métodos.

4.2.5. Otras aproximaciones destacables

Métodos basados en constantes de Lipschitz

Dada una función implícita $h(t)$, si existe una constante positiva L tal que para cualesquiera valores t_1 y t_2 se verifica

$$|h(t_1) - h(t_2)| < L|t_1 - t_2|,$$

entonces se dice que H satisface la condición de Lipschitz y L se conoce por el nombre de constante de Lipschitz. Si t_1 se encuentra relativamente cerca de t_2, entonces

$$\frac{|h(t_1) - h(t_2)|}{|t_1 - t_2|} < L$$

representa la derivada de la función h. En tal caso L mide la variación de la función entre t_1 y t_2.

El método de las superficies L-G [KB89] se basa en las constantes de Lipschitz para desarrollar un método eficaz y de confianza de Ray Tracing en funciones implícitas. Este método trabaja con dos pasos:

1. Se crea una estructura octal para la función implícita $f(x)$ con $x \in \mathbb{R}^3$ y y para la que existe una constante de Lipschitz L. Sea x_0 el centro de uno de los cubos y d la distancia entre éste y cualquiera de los vértices, si $|f(x_0)| > Ld$ entonces podemos descartar el cubo. De otro modo el cubo se subdivide en otros ocho cubos nuevos. La mayoría de cubos se aceptará o rechazará sin problema, pero puede haber casos conflictivos como los que se muestran en la figura 4.5.

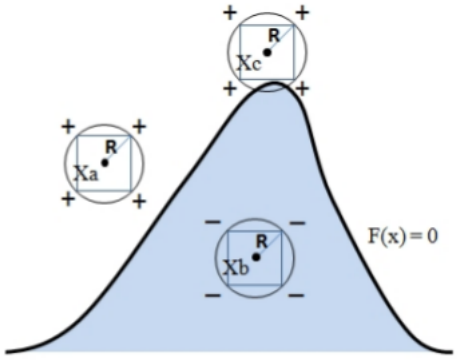

Figura 4.5: Esferas con centro en x_a y x_b no intersecan a la superficie. Mientras que la esfera con centro en x_c puede que sí la interseque.

2. Se realiza Ray Tracing sobre la superficie usando la estructura creada en el primer paso. Si ahora consideramos la función intersección $g(t)$ con constante de Lipschitz G. Sean t_1 y t_2 los puntos de entrada de g en un octal, entonces $t_m = \frac{t_1+t_2}{2}$ y $d = \frac{t_2-t_1}{2}$. Si $|g(t_m)| > Gd$ entonces existe una única raíz entre t_1 y t_2.

El algoritmo sería el que sigue:

```
Lipschitz(t_1,t_2):
   Compute G for t_1, t_2
   t_m = midpoint(t_1,t_2)
   d = (t_2 - t_1)/2
   If(abs(t_m) > G*d)
      If(F(t_1)*F(t_2) < 0)
         Compute using Newton
      Else
         There's no intersection
   Else
      Lipschitz(t_1,t_m)
      Lipschitz(t_m,t_2)
```

Otro método basado en constantes de Lipschitz es el Sphere Tracing [Har96]. Este

método es similar al de las superficies L-G pero usa cotas de Lipschitz en lugar de constantes de Lipschitz. Una cota de Lipschitz es cualquier constante que satisface la condición de Lipschitz, pero no necesariamente la más pequeña. En este método, debemos primero encontrar una cota de Lipschitz para una determinada función según qué distancia consideremos.

Aritmética afín

De Cusatis et al. [C$^+$99] introdujeron el uso de aritmética afín al Ray Casting de superficies implícitas. La Aritmética de Intervalos puede generar una sobre estimación en algunas operaciones con intervalos. La Arimética afín se propuso para controlar esta sobrestimación ya que la aproximación se realiza sobre un paralelogamo, véase 4.6. La Aritmética afín converge cuadráticamente mientras que la bisección usando Aritmética de Intervalos lo hace de forma lineal.

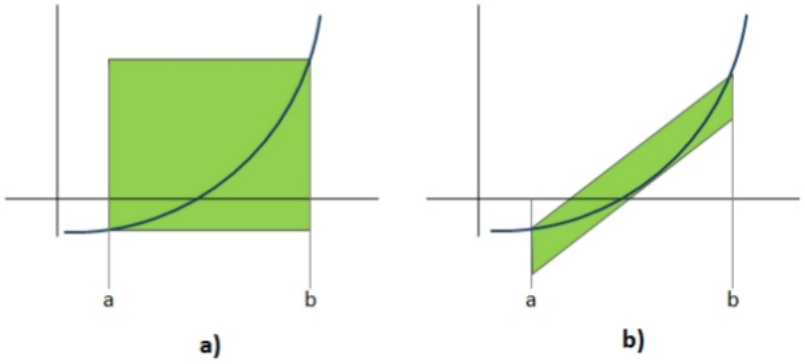

Figura 4.6: Aproximación de una función en un intervalo $[a, b]$. a) Usando Aritmética de Intervalos. b) Usando Aritmética afín.

En la Aritmética afín una cantidad x se define mediante la forma afín

$$\hat{x} = x_0 + x_1\epsilon_1 + \cdots + x_n\epsilon_n,$$

la cual es un polinomio de grado uno. Los elementos ϵ_i son desconocidos pero sabemos que oscilan en el intervalo $[-1, 1]$. Éstos pueden contribuir a la incertidumbre de dos o más cantidades. Esto indica cierta dependencia subyacente entre las cantidades.

El esquema de este nuevo algoritmo es básicamente el mismo que el que describe este mismo concepto en la Aritmética de Intervalos. La diferencia es qué librería ha de usarse para el cálculo de las extensiones unidas. Como pasa en la Aritmética de Intervalos, las operaciones de aritmética básicas y funciones pueden ser extendidas a formas afines que sean manejables [SF97].

Sin embargo, las mejoras introducidas por el uso de Aritmética afín van en detrimento del algoritmo básico propuesto por Mitchell [C+99], pero no en detrimento de en un método más veloz como el de Newton con intervalos. Además, ha sido probado que la Aritmética afín es un caso especial de la forma centrada de la Aritmética de Intervalos [Gav05].

4.3. Conclusiones

En este capítulo hemos hecho una introducción de diferentes aplicaciones de la Aritmética de Intervalos en la creación de estructuras de subdivisión y también en la creación de test de intersección de Ray Tracing en superficies implícitas. La Aritmética de Intervalos nos proporciona una manera sencilla de comprobar los tres ejes de un cubo para saber si interseca a la superficie. Esta técnica puede ser mejorada mediante el estudio de la superficie para realizar más subdivisiones en secciones críticas de ésta.

Además, también hemos presentado diferentes técnicas para realizar test de intersección entre rayos y superficies implícitas. La mayoría de métodos para los test de intersección están destinados a trabajar con Aritmética de Intervalos, la cual remplaza las operaciones sobre número reales por operaciones sobre intervalos.

Hay diferentes aproximaciones, por ejemplo, las basadas en derivadas y otras que crean ciertas condiciones para mejorar el proceso de bisección. Aunque el uso de derivadas mejora la eficiencia, puede que ésta no sea fácil de obtener para algunos tipos de superficies implícitas. Dimos también un rápido vistazo a la comparación de distintos métodos para saber su comportamiento y eficiencia ante las mismas superficies con distinta complejidad y vimos que en superficies sencillas son máś eficientes algoritmos sencillos y en superficies complejas es más eficaz el uso de algoritmos algo más complejos debido a que suelen presentar convergencia de orden cuadrático frente a la lineal de los más simples, lo que compensa las comprobaciones extras que deben realizarse.

Finalmente también vimos como hay otra serie de técnicas que no usan Aritmética de Intervalos como las basadas en constantes de Lipschitz o en Aritmética afín. Aunque su perdida de generalidad les hace menos atractivas y, por tanto, su uso está muy poco popularizado.

Capítulo 5

Próximo paso

Una vez presentado tanto la teoría de Aritmética de Intervalos y una serie de algoritmos de Ray Tracing que usan esta teoría, está claro que el siguiente paso es encontrar técnicas para acelerar estos algoritmos o mejorar la renderización de la imagen sin afectar al coste computacional de éstos. De acuerdo con [AK87] podemos acelerar las técnicas de Ray Tracing de tres maneras: reduciendo del coste medio del test de intersección, reduciendo del número total de rayos intersecados ó reemplazando una cierta cantidad de rayos individuales por un haz de rayos.

En [Flo08] se presenta un nuevo método para sacar rendimiento de la llamada coherencia de los rayos a la hora de acelerar el proceso de Ray Tracing en superficies implícitas que ya hemos presentado. En él podemos adaptar la teoría de Intervalos para estudiar el comportamiento de haces de rayos en lugar de cada rayo de forma individual, esto significa que podemos analizar secciones del espacio para conocer la coherencia de los objetos.

Tras generalizar la definición de rayo para poder tratar un haz como si fuera un solo objeto pasamos a desarrollar dos sencillos algoritmos llamados de *rechazo* y de *inclusión* que nos permiten clasificar los conjuntos de rayos según si todos los rayos del haz intersecan la superficie o si, por el contrario, ninguno lo hace. En el caso de que ambos algoritmos fallen, i.e., estemos ante un haz que interseque parcialmente al objeto, se procede a un proceso de división del espacio en las áreas *problemáticas* y de evaluación de éstas. Como se resultado se obtiene una mejora de la eficiencia, llegando a doblarse en algunos casos.

Por supuesto esto es solo un primer paso de ampliación y esta rama está mucho más desarrollada con trabajos que recrean sombras mediantes haces de rayos secundarios, aplicaciones para el desarrollo de animaciones, mejoras mediante el uso de GPU... Pero eso ya son temas que se acercan más a la rama de la informática.

Bibliografía

[AK87] J. Arvo and D. Kirk. Fast ray tracing by ray classification. In *SIGGRAPH*, 1987.

[Ale70] G. Alefeld. Eine modification des newtonverfarens zur bestimmung der reellen nullstellen einer reellen funktion. *Numeriche Matematik*, pages 32 – 32, 1970.

[B+05] V. Benedet et al. Volumetric of unorganized set of points with implicit surfaces. *Computational Science and Its Applications*, 3480:838 – 846, 2005.

[Ber00] J. Berchtold. *The Bernstein Basis in Set-Theoretic Gaometric Modelling*. PhD thesis, Department of Mechanical Engineering from University of Bath, 2000.

[Bli82] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–236, julio 1982.

[BS01] R. Balsys and K. Suffern. Visualisation of implicit surfaces. *Computers & Graphics*, 25:89–107, febrero 2001.

[BW90] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. *SIGGRAPH Comput. Graph.*, 24(2):109 – 116, febrero 1990.

[BW97] J. Bloomenthal and B. Wyvill. *Introduccion to Implicit Surfaces*. Morgan Kaufman Publishers, San Francisco, California, 1997.

[Bü02] K. Bühler. Implicit linear interval estimations. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, pages 123–132, marzo 2002.

[C+98] P. C. Carvalho et al. Computing arrangements of implicit curves. *Extended abstract in Anais do VERMAC*, pages 19–22, 1998.

[C+99] A. De Cusatis et al. Interval methods for ray casting implicit surfaces with affine arithmetic. In *XII Brazilian Symposium on Computer Graphics and Image Processing (Cat. No.PR00481)*, pages 65–71, 1999.

[C+00] O. Capriani et al. Robust and efficient ray intersection of implicit surfaces. *Reliable Computing*, 6(1):9 – 21, febrero 2000.

[dOdF00] J. B. S. de Oliveira and L. H. de Figueiredo. Robust approximation of offsets and bisectors of plane curves. In *Proceedings 13th Brazilian Symposium on Computer Graphics and Image Processing (Cat. No.PR00878)*, pages 139–145, 2000.

[Duf92] T. Duff. Interval arithmetic recursive subdivision for implicit functions and constructive solid geometry. *SIGGRAPH Comput. Graph.*, 26(2):131–138, julio 1992.

[FBB96] S. Foufou, J. Brunand, and A. Bouras. Surface/surface intersections: A three states classification. *International Conference of Knowledge transfer visualization and graphics 96*, pages 499 – 406, 1996.

[Flo08] J. Florez. *Improvements in the ray tracing of implicit surfaces based on interval arithmetic*. PhD thesis, Department d'Electrònica, Informàtica i Automàtica de la Universitat de Girona, Gerona, España, 2008.

[G+91a] G. Gallo et al. *Efficient Algorithms and Bounds for Wu-Ritt Characteristic Sets*, pages 119–142. Birkhäuser Boston, Boston, Massachusetts, 1991.

[G+91b] G. Gallo et al. *Wu-Ritt characteristic sets and their complexity*, pages 111–136. Dimacs Series on Discrete Mathematics and Theorectical Computer Science. American Mathematical Society and Association for Computing Machinery, 1991.

[Gav05] M. Gavriliu. *Towards more efficient interval analysis: corner forms and remainder interval Newton methods*. PhD thesis, California Institute of Technology, 2005.

[GW05] E. Groot and B. Wyvill. Rayskip: faster ray tracing of implicit surface animations. *Computer graphics and interactive techniques in Australasia and South East Asia*, pages 31 – 36, 2005.

[Han78] E. Hansen. A globally convergent interval method for computing and bounding real roots. *BIT Numerical Mathematics*, 18(4):415 – 424, diciembre 1978.

[Han03] E. Hansen. *Geometric Computations with Interval and New Robust Method*. Horwood publishing limited, 2003.

[Har96] J. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, diciembre 1996.

[Har98] E. Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14:95–108, 1998.

[Har01] J. C. Hart. *Siggraph 93 Course Notes: Design, Visualization and Animation of Implicit Surfaces*, chapter Ray Tracing Implicit Surfaces. Washington State University, Pullman, WA 99164-2752, mayo 2001.

[Har03] E. Hartmann. *Triangulation of Implicit Surfaces*, pages 81 – 92. Technische Hochschule Darmstadt, Darmstadt, Alemania, octubre 2003.

[Hof93] C. M. Hoffmann. Implicit curves and surfaces in cagd. *IEEE Computer Graphics and Applications*, 13:79–88, enero 1993.

[KB89] D. Kalra and A. Barr. Guaranteed ray intersections with implicit surfaces. *SIGGRAPH Comput. Graph.*, 23(3):297 – 306, julio 1989.

[Mit90] D. Mitchell. Robust ray intersection with interval arithmetic. In *Proceedings on Graphics interface '90*, pages 68 – 74, 1990.

[P+06] J. Peiró et al. Shape reconstruction from medical images and quality mesh generation via implicit surfaces. *International Journal for Numerical Methods in Fluids*, 343580(8):1339 – 1360, 2006.

[Ric73] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, mayo 1973.

[SB03] K. Suffern and R. Balsys. Rendering the intersections of implicit surfaces. *IEEE Computer Graphics and Applications*, 23:70–77, septiembre 2003.

[SE+03] J. F. Sanjuan-Estrada et al. Reliable algorithms for ray intersection in computer graphics based on interval arithmetic. In *16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, pages 35–42, octubre 2003.

[SF97] J. Stolfi and L. H. De Figueiredo. Self-validated numerical methods and applications, 1997.

[SH97] B. Stander and J. C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 279–286, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[Tau94] Gabriel Taubin. Rasterizing algebraic curves and surfaces. *IEEE Comput. Graph. Appl.*, 14(2):14–23, marzo 1994.

[Uhl03] K. Uhlir. Modeling methods with implicitly defined objects. Technical Report DCSE/TR-2003-04, Department of Computer Science and Engineering from University of West Bohemia in Pilsen, Univerzitni 8, 30614 Pilse, Czech Republic, febrero 2003.

[W+86] G. Wyvill et al. Data structure for soft objects. *The Visual Computer*, 2:227 – 234, 1986.

[Whi80] T. Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343 – 349, junio 1980.

[Wik15] Wikipedia. Torus.svg, 2015. [Online; acceso el 9 de abril de 2018].