Poligonização de Superfícies Implícitas usando Técnicas Intervalares

Fabricio Lira
Dimas Martínez
Instituto de Matemática - IM
Universidade Federal de Alagoas - UFAL
Maceió, Brasil
fabriciomlira@gmail.com dimas@mat.ufal.br

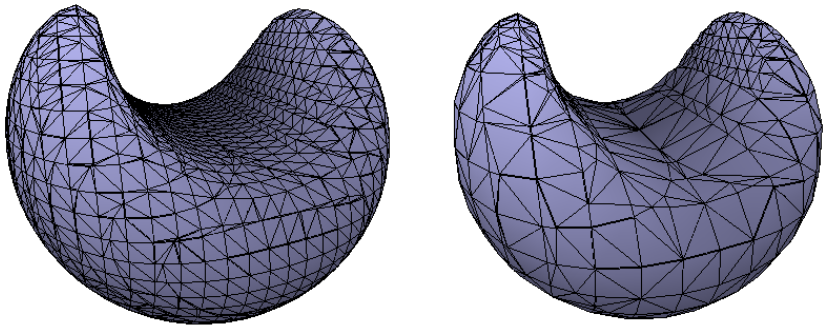


Figura 1. Resultados obtidos com os algoritmos adaptativos implementados para a superfície implícita dada pela equação $(y - x^2 - y^2 + 1)^4 + (x^2 + y^2 + z^2)^4 - 1 = 0$, com o método de Adaptação Espacial (esquerda), e com Adaptação Geométrica (direita).

***Resumo*—Neste trabalho implementamos o algoritmo para poligonização de superfícies implícitas conhecido como "Marching Cubes". A partir daqui, implementamos duas variantes adaptativas: a primeira se adapta à localização espacial da superfície enquanto a outra variante se adapta à geometria da superfície, usando informação proveniente das derivadas. Os algoritmos aqui apresentados estão baseados no uso da Aritmética Intervalar. Fazemos uma comparação entre estes algoritmos com relação a sua eficiência computacional.**

***Keywords*-superfície implícita, aritmética intervalar, diferenciação automática**

***Abstract*—In this work we implement the poligonization algorithm for implicit surfaces known as "Marching Cubes". We also implement two adaptive variants: the first one is adapted to the spatial location of the surface while the second variant is adapted to the geometry of the surface, using the information coming from derivatives. The algorithms presented here are based on the use of Interval Arithmetic. We compare these algorithms with respect to their computational efficiency.**

***Keywords*-implicit surface, interval arithmetic, automatic differentiation**

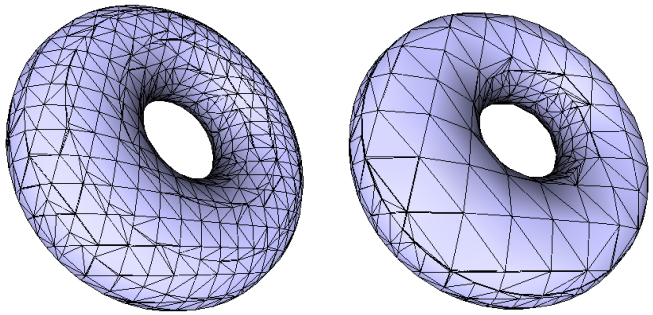
I. Introdução

A renderização de superfícies implícitas é, em geral, um problema difícil. No caso das superfícies paramétricas, a amostragem e representação das mesmas é mais simples, dado que temos uma equação para obter pontos sobre a superfície. Por outro lado, para amostrarmos superfícies implícitas é necessário resolver equações não lineares. E mesmo quando conseguimos amostrar uma superfície implícita, achar uma boa representação dela continua sendo um problema difícil.

Uma forma de resolver este problema é subdividir o espaço em regiões muito pequenas, identificando aquelas por onde a superfície passa. Este processo é chamado de enumeração e existem várias formas de testar se a superfície passa por uma destas regiões. Neste trabalho usamos Aritmética Intervalar [1] para decidir se uma superfície passa por uma região ou não.

O processo de enumeração descrito acima é muito caro, pois quase sempre descartaríamos a grande maioria das regiões. Uma forma de melhorá-lo é o uso de técnicas adaptativas, que permitem descartar grandes regiões onde a superfície não passa [2], [3]. Este processo é conhecido como Adaptação Espacial. Adicionalmente, quando temos informação da geometria da superfície, como sua curvatura, podemos reduzir ainda mais o número de regiões a serem testadas [3], [4]. Chamamos esta técnica de Adaptação Geométrica.

Neste trabalho implementamos dois algoritmos adaptativos, um espacial e outro geométrico [3], que fazem uso da Aritmética Intervalar para poligonizar uma superfície implícita. O segundo usa também Diferenciação Automática [5] para reduzir os cálculos a partir das informações das derivadas. A Figure 2 mostra um exemplo da utilização dos algoritmos adaptativos para poligonização de superfícies implícitas implementados.



(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 2. Superfície implícita dada por $(x^2+y^2+z^2+0.75)^2-4(x^2+y^2) = 0$ com $L = 5$, $\varepsilon = 7$ e nível $= 7$.

O texto está dividido da seguinte forma: começamos na seção II enunciando as definições fundamentais e dando uma idéia geral das técnicas de Aritmética Intervalar e Diferenciação Automática; a continuação, na seção III, apresentamos o algoritmo clássico Marching Cubes para renderização de superfícies implícitas e os algoritmos adaptativos implementados; alguns resultados obtidos e uma comparação da eficiência dos algoritmos estudados são apresentados na seção IV; finalmente, na seção V fazemos as considerações finais.

II. Definições

Nesta seção apresentamos as principais definições e alguns conceitos relacionados que serão úteis para este trabalho.

A. Superfícies Implícitas

Objetos implícitos são conjuntos de nível de funções em várias variáveis. Para valores regulares da função, o teorema da função implícita nos garante que no $\mathbb{R}^3$ esse conjunto é uma superfície, e que no $\mathbb{R}^2$ esse conjunto é uma curva [6]. Um estudo amplo sobre objetos implícitos pode ser encontrado em [7].

Definição (Superfície Implícita)**.** *Dada uma função $f(x,y,z) : \Omega \subseteq \mathbb{R}^3 \to \mathbb{R}$, uma superfície implícita é o conjunto de zeros da função f, se 0 é um valor regular de f. Ou seja, o conjunto $\mathcal{S} = \{(x,y,z) \in \Omega \ / \ f(x,y,z) = 0\}$.*

Por exemplo, as superfícies implícitas dadas por um cilindro e uma esfera se correspondem com as equações $x^2+y^2-9 = 0$ e $x^2 + y^2 + z^2 - 9 = 0$ respectivamente.

B. Aritmética Intervalar

A Aritmética Intervalar surgiu no final da década de 50 com os trabalhos de Moore (veja [1], por exemplo) visando dar suporte a problemas que lidam com a incerteza. Os números representados como intervalos servem como controladores da propagação de erros, uma vez que garante que a resposta correta de determinado problema pertence ao intervalo obtido.

A Aritmética Intervalar trata da representação numérica através de intervalos e das operações aritméticas neles realizadas. Em computação gráfica [8]–[10], as técnicas intervalares são usadas para encontar soluções de vários problemas. Em particular para construir aproximações poligonais de curvas e superfícies implícitas [3], [11], [12].

A seguir, são apresentadas as principais definições e operações da Aritmética Intervalar descritas em [13], fundamentais na construção dos métodos adaptativos [3] pois possibilitam determinar em quais regiões do espaço encontra-se a superfície.

Definição (Conjunto $\mathbb{IR}$)**.** *Define-se o conjunto $\mathbb{IR}$ como sendo o conjunto de todos os intervalos fechados $[a,b]$, ou seja, $\mathbb{IR} = \{[a,b] : a,b \in \mathbb{R} \ e \ a \leq b\}$.*

As operações com intervalos são definidas de modo que o resultado seja um intervalo contendo todas as possíveis operações entre elementos dos operandos – intervalos neste caso. Por exemplo, as operações de soma, subtração, multiplicação e divisão são descritas por:

$$[a,b] + [c,d] = [a+c, b+d]$$

$$[a,b] - [c,d] = [a-d, b-c]$$

$$[a,b] \cdot [c,d] = [\min\{a\cdot c, a\cdot d, b\cdot c, b\cdot d\}, \max\{a\cdot c, a\cdot d, b\cdot c, b\cdot d\}]$$

$$[a,b]/[c,d] = [\min\{a/c, a/d, b/c, b/d\}, \max\{a/c, a/d, b/c, b/d\}],$$

com $0 \notin [c,d]$.

Definição (Função Inclusão)**.** *Seja $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ uma função. Uma função inclusão F de f é aquela definida sobre subconjuntos X de Ω tais que $f(X) \subseteq F(X)$.*

Dada uma função $f : \mathbb{R}^n \to \mathbb{R}$, podemos obter uma função intervalar $F : \mathbb{IR}^n \to \mathbb{IR}$ substituindo as operações entre números reais pelas respectivas operações com intervalos. Esta função é chamada de *extensão intervalar de f*. O Teorema Fundamental da Aritmética Intervalar [1], enunciado a seguir, garante que a extensão intervalar F de f é uma função inclusão.

Teorema (Teorema Fundamental da AI)**.** *Se F é uma extensão intervalar de $f : \mathbb{R}^n \to \mathbb{R}$, então*

$$f(I_1, \cdots, I_n) \subset F(I_1, \cdots, I_n)$$

onde $I_1, \cdots, I_n$ são intervalos em $\mathbb{IR}$.

Este teorema constitui uma ferramenta muito poderosa, pois nos da garantias teóricas de que uma superfície não passa por uma determinada região. Por exemplo, se F é a extensão intervalar de f (ou qualquer outra função inclusão de f), e $0 \notin F([0,1],[0,1],[0,1])$, então temos certeza de que a superfície $\mathcal{S} = \{f(x,y,z) = 0\}$ não passa pelo cubo unitário $[0,1] \times [0,1] \times [0,1]$.

C. Diferenciação Automática

A técnica conhecida como Diferenciação Automática [5], [14] fornece um método para a obtenção exata da derivada de uma função, a partir de uma aplicação criteriosa da regra da cadeia.

Similar a como é feito em Aritmética Intervalar, as operações aritméticas e funções são aplicadas a uma $n-$upla que representa o valor da função e de suas derivadas. Para calcular as primeiras derivadas de uma função $f : \mathbb{R}^3 \to \mathbb{R}$ consideramos a $4-$upla (f, f_x, f_y, f_z), que representa os valores da imagem da função e de suas derivadas parciais. Por exemplo, definimos a soma, a raiz quadrada e o cosseno de f como:

$$(f, f_x, f_y, f_z)+(g, g_x, g_y, g_z) = (f+g, f_x+g_x, f_y+g_y, f_z+g_z)$$

$$\sqrt{(f, f_x, f_y, f_z)} = (\sqrt{f}, \frac{f_x}{2\sqrt{f}}, \frac{f_y}{2\sqrt{f}}, \frac{f_z}{2\sqrt{f}})$$

$$\cos(f, f_x, f_y, f_z) = (\cos f, -f_x \operatorname{sen} f, -f_y \operatorname{sen} f, -f_z \operatorname{sen} f).$$

Note que, para os cálculos, f, f_x, f_y e f_z são números. Portanto, eles podem ser substituídos por intervalos. Isto nos fornce um método intervalar de cálculo de derivadas, que usaremos para estimar o gradiente de uma função (o vetor normal a uma superfície implícita) em uma região.

III. Poligonização de Superfícies Implícitas

As técnicas adaptativas de enumeração de superfícies implícitas representam uma variante mais eficiente para construirmos uma boa amostragem e representação desta por malhas triangulares. Nesta seção decreveremos o algoritmo Marching Cubes e dois métodos adaptativos utilizando Aritmética Intervalar e Diferenciação Automática.

A. Marching Cubes

O algoritmo Marching Cubes [15] constitui um dos métodos mais notórios para renderização de superfícies implícitas e dados volumétricos. Este particiona o domínio em n partes ao longo dos três eixos coordenados criando um grid de cubos, percorrendo-os numa sequência e processando-os de acordo com o sinal de cada um dos seus 8 vérices. De acordo com a classificação, são geradas triagulações locais, resultando na triangulação da superfície ao percorrer-se todo o grid.

Nossa implementação esta fundamentada na atualização proposta por Lewiner et al [16] com adaptação à Aritmética Intervalar, como mostrado no pseudocódigo abaixo.

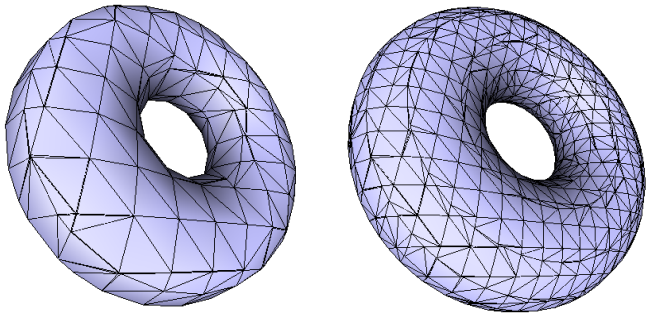
Algoritmo 1:

marchingcubes(f, X, Y, Z, n)
for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    for $k = 1$ to n **do**
      Criar o cubo $X_i \times Y_j \times Z_k$
      Classificar o cubo de acordo com o sinal de cada vértice
      Processar a classificação em relação às ambiguidades
      Calcular os pontos de intersecção nas arestas
      Gerar a triangulação do cubo
    end for
  end for
end for

Na Figure 3 mostramos um exemplo do uso deste algoritmo. Como podemos ver, quanto maior o número de partições nos três eixos coordenados, melhor é a aproximação poligonal da superfície.

B. Adaptação Espacial

Dada uma função $f : \mathbb{R}^3 \to \mathbb{R}$ e um cubo $\mathbf{B}$, é possível, com o uso de técnicas intervalares, descobrir se a superfície implícita $\mathcal{S} = \{f(x, y, z) = 0\}$ passa no cubo $\mathbf{B}$.



(a) Grid 64x64x64 (b) Grid 128x128x128

Figure 3. Superfície implícita dada por $(x^2+y^2+z^2+0.75)^2-4(x^2+y^2) = 0$ com $L = 5$.

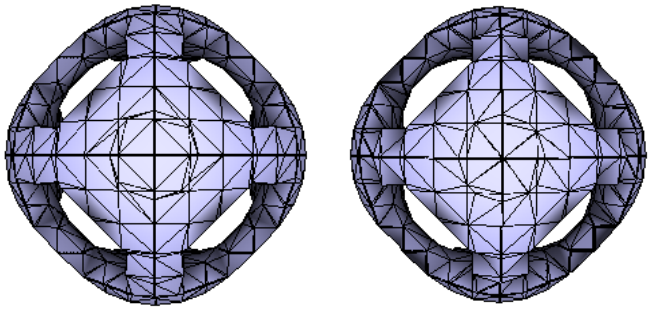
Seja F a função inclusão associada a f, e sejam X, Y e Z três intervalos de $\mathbb{IR}$ tais que $\mathbf{B} = X \times Y \times Z$, pelo teorema fundamental da Aritmética Intervalar sabemos que $f(x, y, z) \subseteq F(X, Y, Z)$. Assim, se o intervalo $K = F(X, Y, Z)$ não contém o número zero em seu interior, então podemos garantir que $\mathcal{S}$ não passa por $\mathbf{B}$. Por outro lado, se $0 \in K$, não temos certeza se a superfície passa ou não por $\mathbf{B}$. No entanto, na medida em que o tamanho de $\mathbf{B}$ diminui, esta certeza aumenta [12].

Baseados neste teste, podemos usar o seguinte algoritmo para renderizar uma superfície implícita. Primeiramente verificamos se $\mathcal{S}$ não passa por $\mathbf{B}$, ou seja, se $0 \notin K$. Neste caso, sabemos que a superfície não passa por $\mathbf{B}$ e podemos descartar este cubo completamente. Se isso não acontecer, então $\mathcal{S}$ pode passar por $\mathbf{B}$. Neste caso dividimos $\mathbf{B}$ em oito cubos $\mathbf{B_1}$, $\mathbf{B_2}$, $\mathbf{B_3}$, $\mathbf{B_4}$, $\mathbf{B_5}$, $\mathbf{B_6}$, $\mathbf{B_7}$ e $\mathbf{B_8}$, repetindo este passo recursivamente até que a árvore gerada neste processo alcance a profundidade dada. Em seguida, em cada um dos cubos que formam as folhas desta árvore, aplicamos o processamento do algoritmo 1 determinando os pontos da superfície e sua respectiva triangulação, pois os cubos obtidos têm tamanhos iguais, definidos em função da profundidade da árvore. Isto garante que a utilização do algoritmo Marching Cubes não criará buracos na malha final obtida. Veja a seguir o pseudocódigo deste método.

Algoritmo 2:

rasterizacao($f, X, Y, Z, nivel$)
if $0 \notin F(X, Y, Z)$ **then**
  return
else
  if $nivel == 1$ **then**
    Gerar a triangulação do cubo $X \times Y \times Z$
  else
    Dividir X, Y, Z em intervalos $X_1, X_2, Y_1, Y_2, Z_1, Z_2$
    rasterizacao($f, X_1, Y_1, Z_1, nivel - 1$)
    rasterizacao($f, X_2, Y_1, Z_1, nivel - 1$)
    rasterizacao($f, X_2, Y_2, Z_1, nivel - 1$)
    rasterizacao($f, X_1, Y_2, Z_1, nivel - 1$)
    rasterizacao($f, X_1, Y_1, Z_2, nivel - 1$)
    rasterizacao($f, X_2, Y_1, Z_2, nivel - 1$)
    rasterizacao($f, X_2, Y_2, Z_2, nivel - 1$)
    rasterizacao($f, X_1, Y_2, Z_2, nivel - 1$)
  end if
end if

Para encontrar os pontos de intersecção das arestas do cubo $X \times Y \times Z$ com a superfície utilizamos o método da bisseção intervalar, pois encontrar este ponto de intersecção se reduz a encontrar o zero de uma função real em um intervalo. No entanto, para resultados mais precisos podemos usar o método de Newton. Na Figure 4 - a) mostramos o resultado da implementação do algoritmo descrito.



(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 4. Superfície implícita dada por $((x^2+y^2-1)^2 + z^2)((y^2+z^2-1)^2 + x^2)((z^2+x^2-1)^2 + y^2) - 0.005625(1 + 3(x^2+y^2+z^2)) = 0$ com $L = 3$, $\varepsilon = 5$ e nível $= 6$.

Com a técnica adaptativa descrita acima, utilizamos um critério de subdivisão adaptado à localização espacial da superfície, descartando rapidamente aquelas regiões por onde a superfície não passa. Como resultado, o algoritmo toma menos tempo para poligonizar a superfície, já que temos um menor número de células a serem processadas. Já o Marching Cubes processa todas as regiões do grid e quase sempre descarta a maioria delas. Note que, se os lados do cubo inicial foram todos divididos por uma potência de 2, ambos os algoritmos geram exatamente a mesma malha.

C. Adaptação Geométrica

O processo de Adaptação Espacial apresentado acima se baseia exclusivamente na localização da superfície, e assim são amostradas a mesma quantidade de pontos em regiões tanto de alta como de baixa curvatura.

Este método pode ser aprimorado utilizando uma adaptação à curvatura da superfície [3], que resulta em uma redução drástica na quantidade de regiões visitadas para desenhar a superfície. Note que regiões onde a curvatura se mantém próxima de zero têm vetor normal quase constante. Então basta encontrar onde o vetor normal não varia muito para detectar regiões de baixa curvatura.

Com a ajuda da diferenciação automática podemos obter informações da curvatura da superfície. Aplicando-se conjuntamente com a avaliação intervalar da função, é possível obter estimativas da variação do vetor normal. Podemos determinar em quais cubos a superfície possui baixa variação da curvatura e portanto podemos obter aproximações à superfície nesses cubos. Estes cubos não serão subdivididos, e assim diminuímos a quantidade de avaliações da função, necessárias para obtermos uma representação consistente da superfície.

Dada a função inclusão $\overrightarrow{N}(X,Y,Z)$ para o vetor normal à superfície $\overrightarrow{n}(x,y,z) = \nabla f(x,y,z) / \parallel \nabla f(x,y,z) \parallel$, no ponto (x,y,z), podemos afirmar que $\overrightarrow{n}(x,y,z) \subseteq \overrightarrow{N}(X,Y,Z)$.

Incluindo essa estimativa para a normal à superfície, obtemos uma adaptação geométrica dela. Se o diâmetro$(\overrightarrow{N}) = \max\{\text{diâmetro}(\overrightarrow{N}_X), \text{diâmetro}(\overrightarrow{N}_Y), \text{diâmetro}(\overrightarrow{N}_Z)\}$ é menor que uma tolerância ε, então podemos concluir que a normal à superfície varia pouco no cubo $X \times Y \times Z$, e portanto podemos aproximar a superfície sem continuar subdividindo. O Algoritmo 3 faz duas verificações: a primeira para a localização da superfície e uma segunda verificação para a variação do vetor normal na região. Desse modo teremos aproximado a superfície por triângulos maiores nas regiões de baixa variação do vetor normal e por triângulos menores nas outras regiões. Na Figure 6 - b) mostramos o resultado da implementação do algoritmo descrito.

Algoritmo 3:

rasterizacao$(f, X, Y, Z, nivel, \varepsilon)$
if $0 \notin F(X,Y,Z)$ **then**
  return
else
  if $diametro(\overrightarrow{N}(X,Y,Z)) < \varepsilon$ **or** $nivel == 1$ **then**
    Gerar a triangulação do cubo $X \times Y \times Z$
  else
    Dividir X, Y, Z em intervalos $X_1, X_2, Y_1, Y_2, Z_1, Z_2$
    rasterizacao$(f, X_1, Y_1, Z_1, nivel-1, \varepsilon)$
    rasterizacao$(f, X_2, Y_1, Z_1, nivel-1, \varepsilon)$
    rasterizacao$(f, X_2, Y_2, Z_1, nivel-1, \varepsilon)$
    rasterizacao$(f, X_1, Y_2, Z_1, nivel-1, \varepsilon)$
    rasterizacao$(f, X_1, Y_1, Z_2, nivel-1, \varepsilon)$
    rasterizacao$(f, X_2, Y_1, Z_2, nivel-1, \varepsilon)$
    rasterizacao$(f, X_2, Y_2, Z_2, nivel-1, \varepsilon)$
    rasterizacao$(f, X_1, Y_2, Z_2, nivel-1, \varepsilon)$
  end if
end if

Semelhantemente à Adaptação Espacial, para gerar a triangulação em cada cubo utilizamos o algoritmo Marching Cubes. No entanto, neste caso cubos vizinhos podem ter tamanhos diferentes, e isto pode ocasionar quebras na triangulação obtida neste cubo em relação à triangulação de seus vizinhos, gerando buracos na malha resultante, como pode ser visto na Figure 5.

Visando solucionar este problema adotamos o seguinte critério para validarmos a triangulação gerada: se algum vizinho ao cubo possui tamanho diferente dele, então identificamos os pontos que formam a quebra, na fronteira entre ambos os cubos, e criamos uma triangulação da quebra, com o intuito de fechar a brecha criada por esta. Uma outra maneira de contornar este problema seria usarmos o Marching Cubes Adaptativo [17].

Embora este procedimento resolve o problema de quebra na triangulação, o resultado tem aparência artificial. Para melhorar a aparência da triangulação, aplicamos localmente uma suavização na malha formada pela quebra, junto com a triangulação do cubo. Esta suavização é alcançada usando o critério de Delaunay nesta pequena malha. Estes passos descritos estão esquematizados na Figure 5.

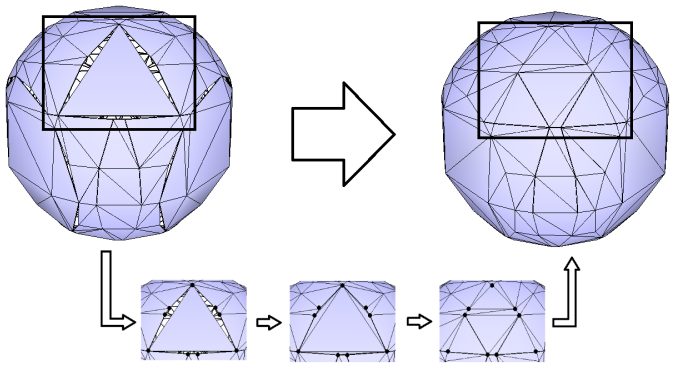
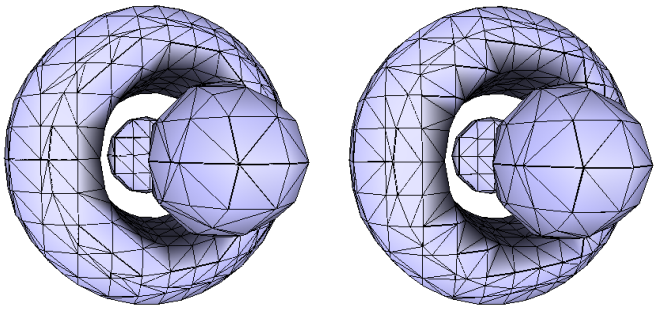


Figure 5. Construção da triangulação no cubo que apresenta tamanho distinto aos seus vizinhos.

IV. Resultados

Os algoritmos descritos acima foram implementados de forma completa em Lua, com saída em OpenGL.

As figuras a seguir mostram alguns resultados obtidos nesta implementação. Em cada figura, o lado esquerdo mostra o resultado do algoritmo de Adaptação Espacial, enquanto o lado direito mostra o algoritmo de Adaptação Geométrica para a mesma superfície. Em todos os exemplos adotamos $\Omega = [-L, L] \times [-L, L] \times [-L, L]$.

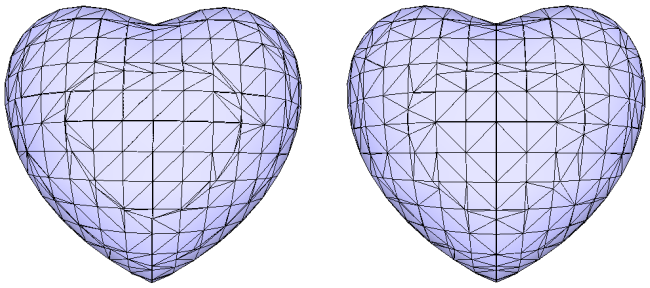


(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 6. Superfície implícita dada por $4(x^4 + (y^2 + z^2)^2) + 17(y^2 + z^2)x^2 - 20(x^2 + y^2 + z^2) + 17 = 0$ com $L = 5$, $\varepsilon = 25$ e nível $= 6$.

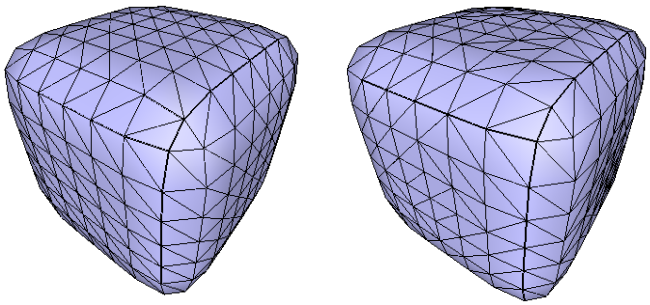
Para comparar os algoritmos, definimos como cubos *visitados* aqueles nos quais avaliamos a função inclusão durante a execução do algoritmo. Chamamos de *finais* aqueles cubos que possuem a tolerância ε adotada ou aqueles que representam as folhas da árvore gerada no processo de recursão dos algoritmos. É nos cubos finais que a superfície é aproximada por uma triangulação.

Na tabela I comparamos os resultados obtidos nas execuções dos exemplos mostrados em cada figura. Como pode ser visto na tabela, para a maioria dos exemplos testados o algoritmo de adaptação geométrica foi consideravelmente mais eficiente do que o algoritmo de adaptação espacial. O algoritmo de adaptação espacial visitou um número de cubos expressivamente maior, enquanto o algoritmo de adaptação geométrica retornou uma quantidade menor com a tolerância dada.



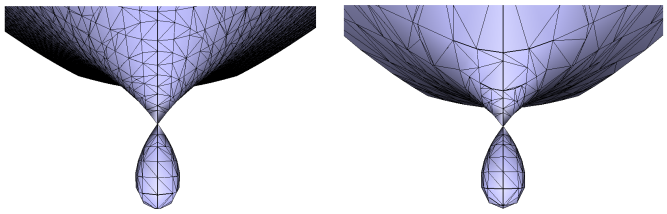
(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 7. Superfície implícita dada por $(x^2 + \frac{9}{4}y^2 + z^2 - 1)^3 - x^2z^3 - \frac{9}{80}y^2z^3 = 0$ com $L = 5$, $\varepsilon = 25$ e nível $= 7$.



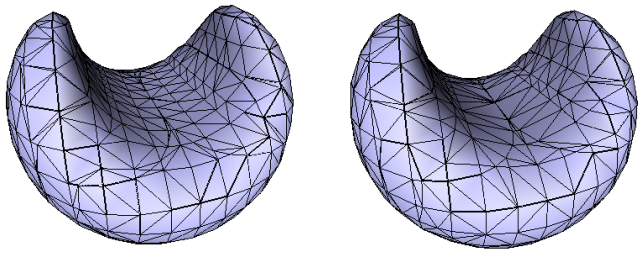
(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 8. Superfície implícita dada por $(0.5625 - x^2)y^2 - (x^2 + 1.5y - 0.75)^2 = 0$ com $L = 3$, $\varepsilon = 6$ e nível $= 5$.



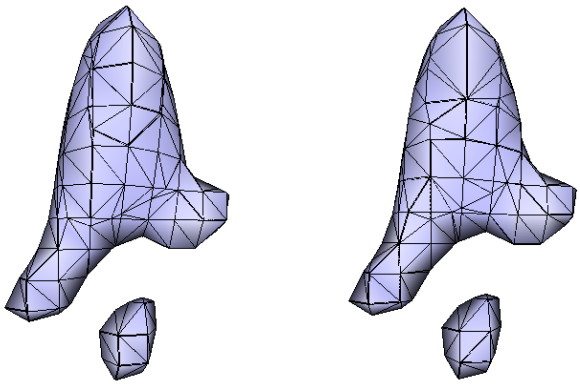
(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 9. Superfície implícita dada por $x^2 + y^2 - 0.5(0.995z^2 + 0.005 - z^3) + 0.0025 = 0$ com $L = 5$, $\varepsilon = 2$ e $nivel = 7$.



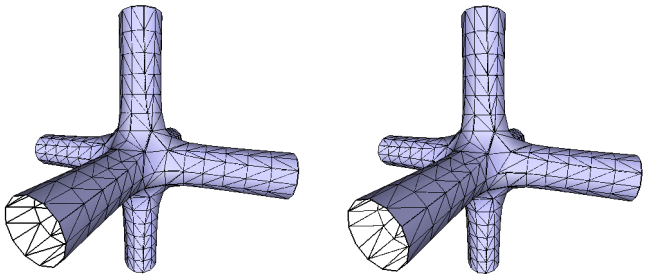
(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 10. Superfície implícita dada por $(y - x^2 - y^2 + 1)^4 + (x^2 + y^2 + z^2)^4 - 1 = 0$ com $L = 5$, $\varepsilon = 26$ e nível $= 7$.



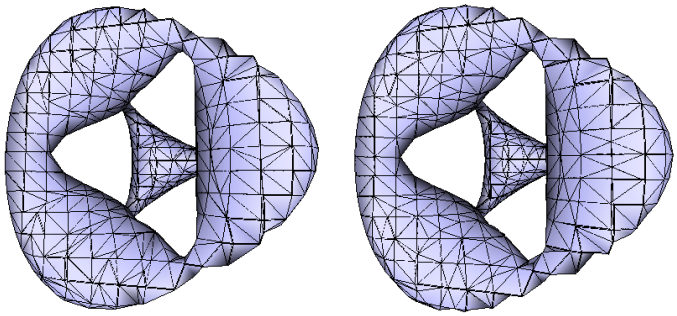
(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 11. Superfície implícita dada por $z^2 + 0.004 + 0.110x - 0.177y - 0.174x^2 + 0.224xy - 0.303y^2 - 0.168x^3 + 0.327x^2y - 0.087xy^2 - 0.013y^3 + 0.235x^4 - 0.667x^3y + 0.745x^2y^2 - 0.029xy^3 + 0.072y^4 = 0$ com $L = 5$, $\varepsilon = 25$ e nível $= 6$.



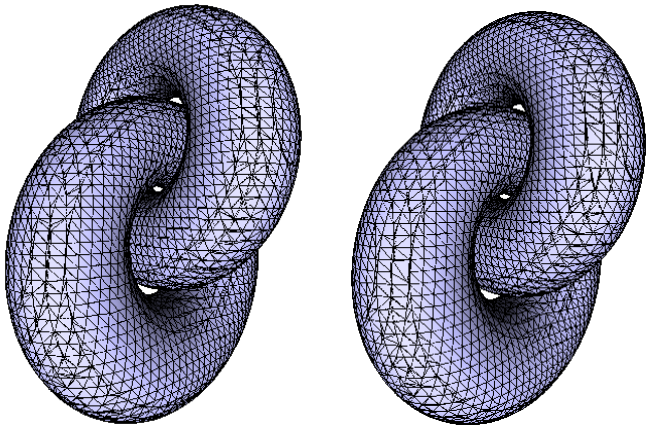
(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 12. Superfície implícita dada por $x^2y^2 + y^2z^2 + x^2z^2 - 2(x^2 + y^2 + z^2) - 4 = 0$ com $L = 10$, $\varepsilon = 15$ e nível $= 5$.



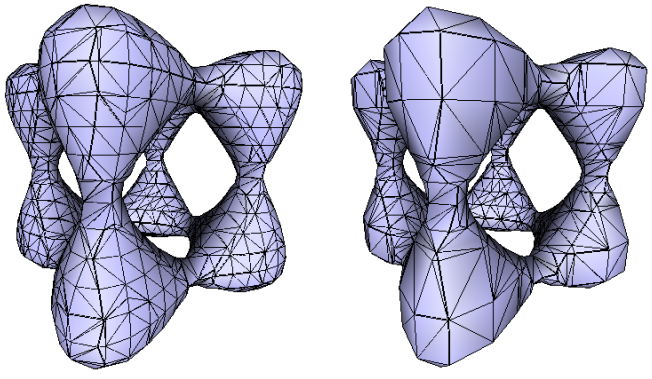
(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 13. Superfície implícita dada por $(x^2+y^2+z^2-23.75)^2 - 0.8((z-5)^2-2x^2)((z+5)^2-2y^2) = 0$ com $L = 5$, $\varepsilon = 25$ e nível $= 5$.



(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 14. Superfície implícita dada por $((x^2+y^2+z^2+12.5775)^2 - 64(x^2+y^2))((x^2+(y+4)^2+z^2+12.5775)^2 - 64((y+4)^2+z^2)) = 0$ com $L = 10$, $\varepsilon = 35$ e nível $= 7$.



(a) Adaptação Espacial (b) Adaptação Geométrica

Figure 15. Superfície implícita dada por $x^4-5x^2+y^4-5y^2+z^4-5z^2+11.8 = 0$ com $L = 5$, $\varepsilon = 30$ e nível $= 6$.

V. Conclusão

Neste trabalho abordamos dois problemas clássicos da Computação Gráfica que estão intimamente relacionados: o desenho de superfícies implícitas e a representação por malhas triangulares destas superfícies. Implementamos duas técnicas adaptativas, que permitem uma diminuição substancial da quantidade de cálculos necessários, para o tratamento deste problema.

A Adaptação Espacial, por utilizar exclusivamente técnicas intervalares constitui uma boa abordagem adaptativa, pois os métodos intervalares fornecem garantias teóricas da existência de pontos da superfície em certa região do espaço que a contém. No entanto, a Adaptação Geométrica, por considerar informações extras da superfície, proporciona uma excelente representação da superfíce com uma menor quantidade de informação, o que ocasiona uma grande redução do custo computacional no que se refere ao armazenamento e processamento de dados no computador.

Table I
Resultados obtidos com os métodos de Adaptação Espacial e Adaptação Geométrica.

Superfície	Visitados		Finais		Triângulos	
	Esp.	Geo.	Esp.	Geo.	Esp.	Geo.
Figure 2	11721	2825	4208	2008	2448	1456
Figure 4	2761	2761	1096	1096	1344	1344
Figure 6	6089	2313	2176	1848	1376	1376
Figure 7	3305	2729	1048	1044	1416	1404
Figure 8	585	585	296	296	584	584
Figure 9	19529	1225	7204	432	14176	704
Figure 10	2601	1833	956	856	1296	1216
Figure 11	14521	4105	4988	3192	388	372
Figure 12	1609	1161	664	620	992	992
Figure 13	3209	2633	1880	1824	1872	1856
Figure 14	67913	43177	34824	30564	16584	16372
Figure 15	10249	2121	4728	1704	3328	1984

Agradecimentos

Os autores agradecem aos revisores anônimos e a Thales Vieira por seus oportunos comentários e sugestões. Este projeto de Iniciação Científica foi dessenvolvido na UFAL, financiado pelo PIBIC / CNPq.

References

[1] R. E. Moore, *Interval Analysis*. Prentice-Hall, 1966.

[2] J. Bloomenthal, "Polygonization of implicit surfaces," *Computer Aided Geometric Design*, vol. 5, no. 4, pp. 341–355, 1988.

[3] A. Paiva, H. Lopes, T. Lewiner, and L. H. de Figueiredo, "Robust adaptive meshes for implicit surfaces," in *Sibgrapi*, 2006.

[4] B. R. de Araujo and J. A. P. Jorge, "Adaptive polygonization of implicit surfaces," *Computers & Graphics*, vol. 29, no. 5, pp. 686–696, 2005.

[5] L. B. Rall, "The arithmetic of differentiation," *Mathematics Magazine*, vol. 59, no. 5, pp. 275–282, 1986.

[6] M. P. Carmo, *Geometria Diferencial de Curvas e Superfícies*, ser. Textos Universitários. Sociedade Brasileira de Matemática/SBM, Rio de Janeiro, 2005.

[7] J. M. Gomes and L. Velho, *Implicit Objects in Computer Graphics*, ser. Monografias de Matemática. IMPA, 1992.

[8] J. M. Snyder, "Interval analisys for computer graphics," *Computer & Graphics*, vol. 26, no. 2, pp. 121–130, 1992.

[9] T. Duff, "Interval arithmetic and recursive subdivision for implicit functions and construtive solid geometry," *Computer & Graphics*, vol. 26, no. 2, pp. 131–138, 1992.

[10] K. G. Suffern and E. D. Fackerell, "Interval methods in computer graphics," *Computers & Graphics*, vol. 15, no. 3, pp. 331–340, 1991.

[11] L. H. Figueiredo and J. Stolfi, "Adaptive enumeration of implicit surfaces with affine arithmetic," *Computer Graphics Forum*, vol. 15, no. 5, pp. 287–296, 1996.

[12] H. Lopes, J. B. Oliveira, and L. H. Figueiredo, "Robust adaptive polygonal approximation of implicit curves," *Computer & Graphics*, no. 26, pp. 841–852, 2002.

[13] R. E. Moore, *Methods and Applications of Interval Analysis*. SIAM, 1979.

[14] H. Kagiwada, R. Kalaba, N. Rasakhoo, and K. Spingarn, "Numerical derivatives and nonlinear analisys," *Plenum, New York*, 1986.

[15] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *Computer & Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[16] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares, "Efficient implementation of marching cubes cases with topological guarantees," *Journal of Graphics Tools*, vol. 8, no. 2, pp. 1–15, 2003.

[17] R. Shu, C. Zhou, and M. S. Kankanhalli, "Adaptive marching cubes," *The Visual Computer*, vol. 11, pp. 202–217, 1995.