Segmentação de Malhas Triangulares Baseada em Convexidade Relaxada

Rui S. V. Rodrigues José F. M. Morgado
Escola Superior de Tecnologia e Gestão de Viseu, IPV
Viseu
{rsrodrigues,fmorgado}@estv.ipv.pt

Abel J. P. Gomes
Instituto de Telecomunicações, UBI
Covilhã
agomes@di.ubi.pt

Resumo

Este artigo introduz uma nova técnica de segmentação de malhas triangulares, designada por segmentação de montes e vales *(MeV), que combina técnicas de segmentação baseadas na fronteira e no interior de regiões. O algoritmo MeV usa uma função altura com sinal com o objectivo de distinguir entre montes (+) e vales (-) e zonas planares (0). Cada região é construída à volta de extremos locais (i.e., máximos e mínimos) da malha. Um monte é construído a partir de um máximo e um vale a partir de um mínimo. No entanto, é possível às fronteiras dos montes invadirem parcialmente os vales e vice-versa. Consequentemente, vamos ter montes que formam regiões convexas relaxadas e vales que formam regiões côncavas relaxadas. Ao contrário do actual estado da arte da segmentação de malhas, a existência destas regiões relaxadas torna o algoritmo eficaz na segmentação de diferentes tipos de objectos (de forma livre ou não).*

Palavras-Chave

Malhas triangulares, segmentação de malhas, convexidade relaxada.

1 INTRODUÇÂO

A segmentação de malhas é importante em áreas tão diversas como a modelação geométrica, desenho assistido por computador e computação gráfica. A segmentação de malhas de triângulos consiste na partição da malha num conjunto de sub-malhas. Mas, como refere Attene et al. [Attene 06a], a segmentação de malhas pode ser feita com base em critérios geométricos ou critérios semânticos. Nas técnicas de segmentação baseadas em propriedades geométricas, a malha é dividida num conjunto de sub-malhas que satisfazem uma determinada propriedade geométrica (por exemplo, curvatura ou distância a um plano). Por outro lado, nas técnicas de segmentação baseadas em semântica, a divisão da malha em sub-malhas tem lugar quando cada sub-malha delimita um região perceptualmente significativa (por exemplo, um braço do corpo humano).

Na verdade, os métodos de análise dos objectos não fornecem uma descrição semântica das suas formas, mas eles permitem uma caraterização geométrica e estrutural dos objectos. No entanto, acredita-se que esta informação, aliada ao conhecimento das ciências cognitivas acerca da percepção humana, permite a definição de novos mecanismos para estruturar e perceber a forma dos objectos. Neste âmbito, Biederman [Biederman 87] diz que as pessoas percepcionam os objectos como uma coleção de partes, enquanto Hoffman [Hoffman 97] refere que a visão humana define fronteiras das partes ao longo dos mínimos negativos das curvaturas principais. Note que, de certa maneira, a asserção de Hoffman implica que as partes significativas dos objectos são convexas. Como vamos ver mais à frente, o algoritmos proposto neste artigo segue esta asserção de Hoffman.

1.1 Trabalho relacionado

A segmentação de objetos consiste na divisão de um objeto em sub-objetos significativos em termos de forma. Na nossa opinião, a melhor maneira de classificar os algoritmos de segmentação de malhas é ter em conta a dimensão estrutural dos objetos. Assim sendo, pode dizer-se que há três categorias principais de técnicas de segmentação:

1. *Segmentação baseada em volume*. Neste caso, os segmentos são volumes. A entrada é uma malha volumétrica 3D, que é particionada em sub-malhas volumétricas 3D [Chazelle 81] [Bajaj 92] [Lien 04] [Ghosh 13].

2. *Segmentação baseada em superfície*. Nesta técnica, os segmentos são regiões definidas por malhas de triângulos 2D. Cada região consiste num conjunto de faces ligadas que tem uma propriedade geométrica semelhante (por exemplo, convexidade e curvatura).

3. *Segmentação baseada em esqueleto (skeleton*). Nesta técnica, também conhecida como esqueletização, os segmentos são segmentos de reta. A entrada é uma

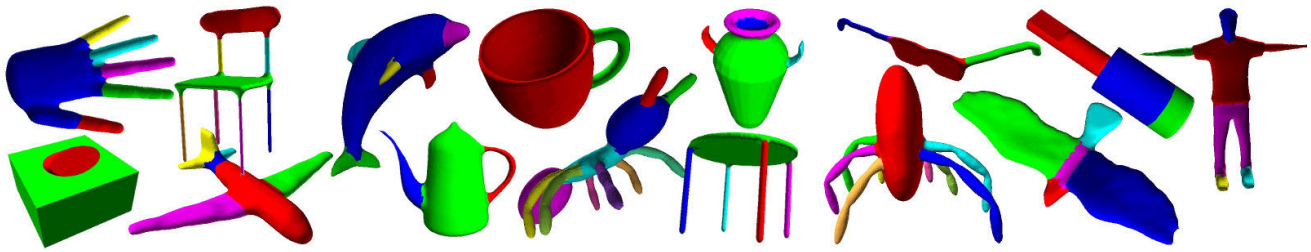

Figura 1. Um conjunto diverso de malhas, segmentadas usando o algoritmo MeV.

malha volumétrica 3D ou uma malha de superfície 2D, mas a saída é um esqueleto 1D que representa a forma estrutural da malha. Gerar este *esqueleto* é um processo conhecido como esqueletização (*skeleton extraction* ou *skeletonization*) [Li 01] [Capell 02].

No que diz respeito aos algoritmos de *segmentação baseada em superfície*, temos as seguintes sub-categorias principais: *region growing* [Chazelle 95] [Kalvin 96], *watersheds* [Mangan 99] [Zuckerberger 02], *hierarchical clustering* [Garland 01], [Katz 05], *iterative clustering* [Lloyd 82], *spectral clustering* [Chung 96], [Zhang 07], [Reuter 09], e *fuzzy clustering* [Katz 03]. De notar que o nosso algoritmo esta incluído na sub-categoria dos algoritmos de *region growing*. Basicamente, este consiste na decomposição da malha em regiões poligonais convexas. Cada região começa com um vértice inicial ou polígono e cresce em número de polígonos (que vão sendo agregados) até ser satisfeita uma condição pré-definida ou um critério de paragem. Isto é, a região cresce enquanto é válida uma condição pré-definida relacionada com a convexidade. A maioria dos algoritmos baseados em *region growing* têm duas etapas: *region growing* (expansão de região) e *region merging*(fusão de regiões) [Zuckerberger 02], [Kalvin 96], [Lavoué 05], [Kim 10].

1.2 Contribuições

As principais contribuições do algoritmo de segmentação de malhas proposto neste artigo são as seguintes:

- *Convexidade relaxada*. Este tipo de convexidade permite-nos formar regiões que não são estritamente convexas nem estritamente côncavas. Isto é particularmente apropriado para a segmentação de objectos de forma livre, como por exemplo, o polvo e a formiga que podemos ver na Figura 1.
- *Point membership test* (PMT). PMT é um caso particular do teste SMC (*set membership classification*), que é muito popular na modelação CSG (*constructive solid geometry*) [Tilove 80]. Este teste é neste artigo usado pela primeira vez como teste de convexidade na segmentação de malhas.
- *Função de altura com sinal*. Ao contrário de outros algoritmos que usam funções de altura sem sinal (ver, por exemplo, [Mangan 99]), nós adoptamos uma função de altura com sinal que nos permite organizar a malha em regiões de dois tipos: montes e vales. Esta função de altura com sinal é obtida pelo sinal da curvatura dada por

$$\kappa(i) = (-1)^i K \tag{1}$$

onde K é a curvatura de Frobenius e i é indice de convexidade que toma o valor 0 para vértices convexos e 1 para vértices côncavos.

O restante artigo está organizado como se segue. A secção 2 faz a descrição do conceito de convexidade relaxada e descreve, ainda, de forma breve o algoritmo MeV. As Secções 3-9 descrevem o algoritmo passo a passo. A Secção 10 apresenta os resultados mais relevantes , em particular os resultados obtidos com o *benchmark* de Princeton. Finalmente, a Secção 11 apresenta as conclusões finais.

2 SEGMENTAÇÃO RELAXADA

2.1 Conceitos Teóricos

De acordo com a asserção de Hoffman [Hoffman 97] referida na Seção 1, a visão humana delimita as regiões do objeto ao longo dos mínimos negativos das curvaturas principais, o que implica que as regiões significativas de um objeto são convexas. Isto é ilustrado na Figura 3, onde a malha se divide em regiões convexas relaxadas; nenhuma região côncava relaxada faz parte desta malha. Em suma, ao contrário das outras técnicas de segmentação que dividem uma malha em regiões convexas, côncavas e de sela, a nossa divide uma malha em regiões convexas relaxadas e regiões côncavas relaxadas, evitando assim a sobre-segmentação da malha tanto quanto que possível.

Uma região não precisa ser estritamente convexa (respectivamente, côncava) para ser classificada como um monte (respectivamente, vale). Uma região *convexa relaxada* ou monte é uma região que é delimitada por um ciclo de arestas côncavas. Por outro lado, uma região *côncava relaxada* ou vale é uma região que é delimitada por um ciclo de arestas convexas. Isto significa que uma região convexa relaxada (respectivamente, região côncava relaxada) pode conter arestas côncavas (respectivamente, arestas convexas) no seu interior, desde que não formem ciclos. Assim, uma região convexa também é convexa relaxada, mas não vice-versa; de modo semelhante, uma região côncava é também côncava relaxada, mas não vice-versa. A convexidade relaxada é particularmente importante na segmentação de ob-

jetos de forma livre; por exemplo, as pernas do polvo descrito na Figura 1 são exemplos de regiões convexas relaxadas.

Um monte tem um pico, i.e., um ponto com curvatura positiva máxima. Um vale tem um nadir ou ponto mais baixo, i.e., um ponto com curvatura negativa mínima. Começando pelos picos e nadires, podemos construir montes e vales, respectivamente, até eles satisfazerem as suas condições de paragem.

2.2 Esboço do Algoritmo

De modo a criar uma segmentação baseada nos conceitos apresentados, foi implementado o algoritmo de segmentação MeV composto pelas seguintes etapas:

1. *Cálculo da curvatura dos vértices*. Cálculo da curvatura de Frobenius em cada vértice da malha (Ver Secção 3).

2. *Cálculo da curvatura das faces*. Cálculo da curvatura de cada face, calculando a média da curvatura de Frobenius dos seus vértices (Ver Secção 4).

3. *Classificação das arestas*. Classificar cada aresta da malha como aresta convexa, aresta côncava ou aresta plana. Esta classificação é feita usando o PMT, em vez do tradicional ângulo diedral (Ver Secção 5).

4. *Classificação das faces*. Classificar cada face da malha como face convexa ou face côncava. A convexidade de cada face é determinada pela tipo de convexidade das suas arestas (Ver Secção 6).

5. *Encontrar Picos e Nadires*. Encontrar faces de curvatura máxima (picos dos montes) e faces de curvatura mínima (nadires dos vales). Estas faces são as faces iniciais das regiões da malha a criar no próximo passo (Ver Secção 7).

6. *Expansão de Regiões*. Construir regiões a partir das faces iniciais, que são faces pico e faces nadir. De recordar que uma região da malha corresponde a um conjunto conexo de faces duma malha de triângulos (Ver Secção 8).

7. *Fusão de Regiões*. Juntar regiões adjacentes se for apropriado (Ver Secção 9).

Em resumo, o algoritmo MeV criar uma segmentação geométrica, na qual as regiões são montes e vales. Cada região termina o seu processo de expansão quando a sua fronteira (i.e., arestas que a delimitam) se torna estritamente convexa ou côncava.

3 CÁLCULO da CURVATURA dos VÉRTICES

A curvatura num vértice é calculada através da norma da matriz de covariância, das normais $\{\mathbf{N}_i\}$ das faces incidentes no vértice, onde $\mathbf{N}_i = (x_i, y_i, z_i)$ e $i = 1, \ldots, n$. Para essa finalidade, temos que determinar o vetor médio das normals, que é:

$$\bar{\mathbf{N}} = (\bar{x}, \bar{y}, \bar{z}), \text{ com } \bar{x} = \sum_{i=1}^{n} x_i,\ \bar{y} = \sum_{i=1}^{n} y_i, \text{e } \bar{z} = \sum_{i=1}^{n} z_i.$$

A fórmula geral para calcular a covariância de duas variáveis $u \in \{x, y, z\}$ and $v \in \{x, y, z\}$ é

$$\sigma_{uv}^2 = \frac{1}{n} \sum_{i=1}^{n} (u_i - \bar{u})(v_i - \bar{v}) \tag{2}$$

de modo que a matriz de covariância é a seguinte:

$$C = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \tag{3}$$

Consequentemente, a norma da matriz de covariância (isto é, matriz de Frobenius) é a seguinte:

$$K = \sqrt{\sum_{\substack{u \in \{x,y,z\} \\ v \in \{x,y,z\}}} \sigma_{uv}^2} \tag{4}$$

que representa a curvatura combinatória num dado vértice rodeado por n faces. Note que, de acordo com a Eq. (4), o valor de K é sempre maior ou igual a zero, de modo que não há possibilidade de distinguir entre um vértice convexo e um vértice côncavo, se estes tiverem a mesma curvatura. Como mostramos mais à frente, este problema resolve-se através do PMT.

4 CÁLCULO da CURVATURA das FACES

Depois de calcular a curvatura de Frobenius em cada vértice, calcula-se a curvatura de cada face como sendo a média da curvatura dos seus vértices. Mas, como vimos na secção anterior, isso não nos fornece qualquer informação sobre a convexidade de qualquer vértice, aresta, face, ou região da malha. A análise da convexidade da malha inicia-se com a análise da convexidade das suas arestas através do PMT, o que é então propagado para as suas faces, podendo-se então começar a formar regiões da malha.

5 CLASSIFICAÇÂO das ARESTAS

A análise da convexidade de cada aresta é baseada no PMT.

5.1 Classificação pelo PMT

De acordo com o critério PMT, todo o ponto 3D é classificado em relação à malha do objecto da seguinte forma: o ponto está do lado de *fora* da malha; o ponto está *sobre* a malha; o ponto está do lado de *dentro* da malha.

Classificar pontos desta forma, em relação a uma malha de triângulo, reduz-se à classificação de um ponto relativamente ao plano definido pela face da malha que lhe está mais próximo. Vamos, então, usar a equação vetorial do plano α dada pelo produto interno $\overrightarrow{N} \cdot (\overrightarrow{CP}) = 0$, onde $\overrightarrow{N}$ é o vector normal à face de centro C e $\overrightarrow{CP}$ representa um vetor em α definido por C e qualquer outro ponto P pertencente a α. Assumindo que $N = (a, b, c)$, $C = (x_0, y_0, z_0)$

e $P = (x, y, z)$, obtemos facilmente a equação Cartesiana do plano como $\alpha(x, y, z) : ax + by + cz + d = 0$, onde $d = (-ax_0 - by_0 - cz_0)$. É claro que o plano $\overrightarrow{N} \cdot (\overrightarrow{CP}) = 0$ divide $\mathbb{R}^3$ em dois semi-espaços, i.e., semi-espaço positivo $\overrightarrow{N} \cdot (P - C) > 0$ e o semi-espaço negativo $\overrightarrow{N} \cdot (P - C) < 0$. A normal $\overrightarrow{N}$ aponta para o semi-espaço positivo (i.e., para fora), de modo que podemos facilmente testar se um ponto está dentro ou fora da malha usando um único triângulo nas proximidades.

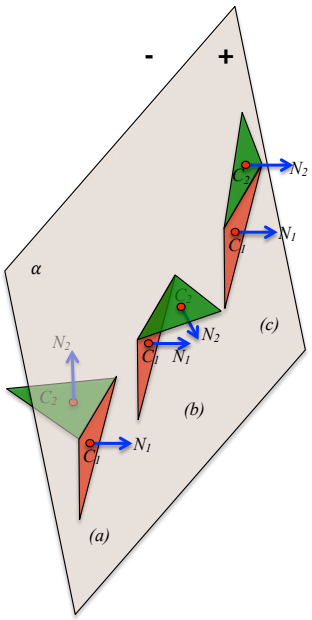

Figura 2. Análise da convexidade de duas faces vizinhas (vermelha e verde) baseada no PMT.

5.2 Análise da convexidade das aresta via PMT

O critério PMT pode ser usado para avaliar a convexidade de uma aresta, que é partilhada pelas duas faces nela incidentes (Figura 2). Seja C_1 e $\overrightarrow{N_1}$ o centro e vector normal da primeira face, e C_2 e $\overrightarrow{N_2}$ o centro e vector normal da segunda face. Então, nós temos:

- Se $\overrightarrow{N_1} \cdot (C_2 - C_1) < 0$, a segunda face situa-se no lado negativo do plano α, assim sendo, a aresta partilhada pelas faces é definida como *convexa* (Figura 2(a));
- Se $\overrightarrow{N_1} \cdot (C_2 - C_1) > 0$, a segunda face situa-se do lado positivo do plano α, assim sendo, a aresta partilhada pelas faces é definida como *côncava* (Figura 2(b)).
- Se $\overrightarrow{N_1} \cdot (C_2 - C_1) = 0$, então as faces são co-planares (Figura 2(c)), e a aresta partilhada é definida como *plana*;

Note-se que este critério pode ser usado para saber se uma aresta é ou não convexa, sem calcular o ângulo diedral entre os triângulos incidentes na aresta. Além disso, o PMT não sofre da ambiguidade do ângulo diedral na avaliação da convexidade das arestas. Após a realização de alguns testes, observou-se que o critério PMT é mais rápido do que o critério de ângulo de diedral, pois gasta menos de 50 por cento do tempo.

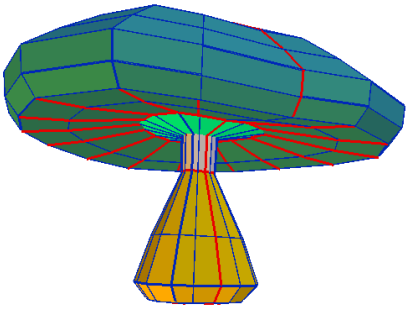

Figura 3. Cogumelo segmentado usando o algoritmo MeV, com as arestas convexas e côncavas representadas

6 CLASSIFICAÇÃO das FACES

Antes de iniciar a formação das regiões da malha, temos que avaliar e classificar cada face em relação à convexidade. Para esse efeito, adotamos a seguinte classificação para as faces:

- *Face convexa*. Uma face é *convexa* se o seu número de arestas convexas é maior ou igual ao número de arestas côncavas. Neste caso, tendo em consideração que o descritor de convexidade i igual a zero para faces convexas, obtemos o sinal da curvatura $\kappa(0) = (-1)^0 K = K$, onde K representa a média da curvatura de Frobenius da face.
- *Face côncava*. A face é *côncava* se o seu número de arestas côncavas é maior do que o número de arestas convexas. Neste caso, o descritor de convexidade i igual a 1, de modo que o sinal da curvatura $\kappa(1) = (-1)^1 K = -K$.
- *Face plana*. A face é *plana* se todos arestas delimitadoras são planas. Neste caso, o sinal da curvatura $\kappa(i) = 0$ porque $K = 0$, não importa o valor de seu descritor de convexidade i.

7 ENCONTRAR PICOS e NADIRES

Nesta etapa o objetivo é encontrar os picos dos montes e os nadires dos vales da malha de triângulos.

Intuitivamente, os picos são vértices convexos da malha, como por exemplo, os cantos convexos de um cubo. Por outro lado, os pontos de nadir são vértices côncavos da malha. Pontos planares são pontos localizados em zonas planas, isto é, pontos de curvatura nula.

No entanto, estamos é interessados em encontrar faces de pico e faces de nadir. Uma face de pico é uma face de

curvatura máxima positiva, isto é, uma face rodeada por faces com menor ou igual curvatura; uma face de nadir é uma face de curvatura mínima negativa, isto é, uma face rodeada por faces com maior ou igual curvatura. As faces iniciais dos montes são especificamente faces de pico, enquanto para os vales são as faces de nadir. Faces de pico e de nadir são agrupadas em arrays separados, para serem facilmente selecionados no etapa seguinte.

8 EXPANSÃO DE REGIÕES

A etapa de expansão de regiões do algoritmo baseia-se no conceito de convexidade relaxada. Seguindo a asserção de Hoffman, o nosso algoritmo de segmentação cria em primeiro lugar todas as regiões convexas relaxadas; depois, o algoritmo constrói as restantes regiões côncavas relaxadas.

Algoritmo 1: Construir Montes da Malha

Verificar : F: array de faces *seed* para os montes
Verificar : H: array vazio de montes
1: Ordenar F por curvatura decrescente
2: $n \leftarrow$ tamanho de F
3: **para** $i \leftarrow 0$ até $n-1$ **fazer**
4: **se** $F[i] \notin$ a uma região já criada da malha **então**
5: Criar novo monte m
6: $m \leftarrow m \cup F[i]$
7: **enquanto** $\exists$ aresta não-côncava a delimita m **fazer**
8: $f \leftarrow$ face adjacente a a
9: $m \leftarrow m \cup f$
10: **fim enquanto**
11: $H \leftarrow H \cup m$
12: **fim se**
13: **fim para**

O procedimento para criar as regiões convexas relaxadas (ou montes) de uma malha é descrito no Algoritmo 1. A construção de um monte começa com uma face inicial convexa (cf. linhas 4 a 12 no Algoritmo1). Com o objectivo de expandir a região, examinam-se as faces vizinhas para verificar se podem ser adicionadas ao (cf. linha 7) monte, repetindo-se então o processo de crescimento iterativo para cada face adicionada anteriormente. Somente faces planas e convexas podem ser adicionadas à região corrente. A condição de paragem verifica-se quando toda a fronteira do monte é formada só por arestas côncavas, como expresso pela condição do ciclo na linha 7. No Algoritmo 1, supomos que temos n montes porque temos no início N faces iniciais. Mas, se alguma face inicial é inserida num outro monte durante a formação deste, a condição na linha 4 não será satisfeita, por isso, termina tendo um número de montes que é menor que n. Por exemplo, todas as faces de um pico em torno dos vértices de um cubo irão ser incluídas num único monte, que compreende toda a malha que cobre o cubo.

Da mesma forma, o processo para formar as regiões côncavas relaxadas (ou vales) de uma malha está descrito no Algoritmo 2. Construir um vale começa com uma face de nadir. A região do vale cresce a partir de uma face côncava inicial adicionando a seguir faces vizinhas que são côncavas ou planas. O crescimento de um vale pára quando a sua fronteira é formada só por arestas convexas (cf. linha 7 no Algoritmo 2).

Algoritmo 2: Construir Vales da Malha

Verificar : F: array de faces seed para os vales
Verificar : V: array vazio de vales
1: Ordenar F por curvatura decrescente
2: $n \leftarrow$ tamanho de F
3: **para** $i \leftarrow 0$ até $n-1$ **fazer**
4: **se** $F[i] \notin$ a uma região já criada da malha **então**
5: Criar um novo vale v
6: $v \leftarrow v \cup F[i]$
7: **enquanto** $\exists$ aresta não-convexa a delimita v **fazer**
8: $f \leftarrow$ face adjacente a a
9: $v \leftarrow v \cup f$
10: **fim enquanto**
11: $V \leftarrow V \cup v$
12: **fim se**
13: **fim para**

Recorde-se que os montes são construídos antes dos vales, o que está em conformidade com a afirmação de Hoffman [Hoffman 97] , isto é, de acordo com o modo como os seres humanos percepcionam as formas do mundo envolvente. Em termos de algoritmos, isto traduz-se no fato de que qualquer região ao ser expandida vai preservando a sua convexidade relaxada. Isto significa que uma região convexa relaxada pode compreender arestas côncavas, desde que essas arestas da malha não formem um ciclo. Isso vale também para a região côncava relaxada de uma malha que possuem arestas convexas.

Além disso, a condição que paragem da formação de uma região da malha é bastante importante no nosso algoritmo pois termina a formação de um monte (e também um vale) no ciclo de inflexão que separa o monte (e também um vale) das outras regiões da malha. O ciclo de inflexão que delimita um monte é definido por arestas côncavas, enquanto o que delimita um vale apenas compreende arestas convexas. Uma consequência subtil das condições de paragem para os montes e vales é que o crescimento da região pára automaticamente, sendo o suficiente para efectuar a segmentação da superfície de objectos de forma não livre (por exemplo, peças mecânicas na parte superior da Figura 1); nenhuma etapa de fusão de regiões é necessária a não ser para objetos de forma livre. O mesmo se passa para alguns objectos de forma livre mais simples que também não necessitam do passo suplementar de fusão de regiões para alcançar as segmentações finais, como é o caso do cogumelo apresentado na Figura 3.

9 FUSÃO DE REGIÕES

A fusão de regiões é uma etapa essencialmente necessária para objetos de forma livre como, por exemplo, os seres humanos e animais. Estes objetos de forma livre

são os objetos mais difíceis de segmentar no processo de segmentação do nosso algoritmo (e na maioria dos outros métodos) porque eles apresentam muitas variações na curvatura, o que provoca o aparecimento de *sobre-segmentação* na malha. Em grande parte, os altos e baixos dos objetos de forma livre levou-nos a chegar à noção de geometria relaxada, que está no cerne do nosso algoritmo. Assim, a fusão de regiões visa reduzir o excesso de segmentação da malha.

Na etapa de fusão de regiões, decidimos usar o algoritmo proposto por Chen e Georganas [Chen 06]. A ideia principal deste algoritmo é reduzir o número de pequenas regiões (segmentos), através da junção das regiões mais pequenas com regiões maiores, que lhes são adjacentes. Basicamente, uma pequena região é fundida com a região adjacente com o qual compartilha a parte mais longa da sua fronteira.

Observa-se que a *sobre-segmentação* também pode fazer a sua aparição em objetos de forma não livre (por exemplo, peças mecânicas), isto é, com as regiões de malhas com faces co-planares. Isso acontece como consequência dos erros de arredondamento em cálculos aritméticos com vírgula flutuante. A fim de evitar este problema, é necessário usar uma tolerância para garantir a co-planaridade de triângulos adjacentes, mas esta operação é feita na etapa de anterior.

10 RESULTADOS EXPERIMENTAIS

10.1 Configuração de Testes

O nosso algoritmo de segmentação foi projetado e implementado num computador com um processador 2.4 Intel Core Duo e um sistema operativo Mac OS X, usando o OpenGL e a *User Interface Library* (GLUI), que é uma biblioteca em C++ baseada no *OpenGL Utility Toolkit* (GLUT).

10.2 Aferição (Benchmarking)

Utilizou-se o *Princeton benchmarking* [Chen 09] para segmentações, com o objectivo de comparar o nosso algoritmo com o estado-da-arte dos algoritmos de segmentação de malhas. Esta ferramenta de *benchmarking* fornece 19 categorias de modelos, em que cada categoria é composta por 20 objectos. Esta ferramenta também fornece 4300 segmentações criadas por seres humanos que servem de referencia para a avaliação dos métodos, sendo fornecidos para cada objecto, em média, 11 segmentações criadas por seres humanos.

O *Princeton benchmarking* disponibiliza comparações quantitativas das segmentações geradas pelos seres humanos e das segmentações geradas por computador produzidos pelos sete algoritmos seguintes: *k-means* [Shlafman 02], *random walks* [Lai 08], *fitting primitives* [Attene 06b], *normalized cuts* [Golovinskiy 08] *randomized cuts* [Golovinskiy 08], *core extraction* [Katz 05] e *shape diameter function* [Shapira 08]. De notar que o software de *benchmarking* não inclui os códigos destes sete algoritmos, mas apenas segmentações produzidos por eles,

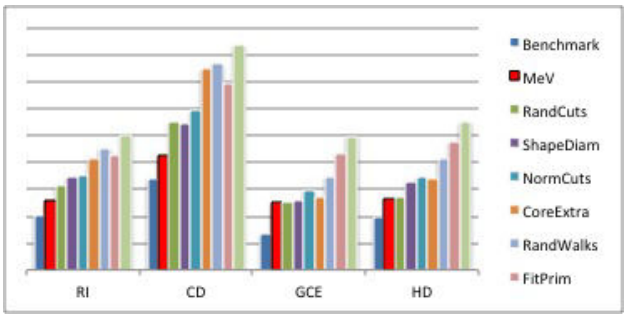

Figura 4. Resultados das métricas.

que funcionam como dados de entrada para o software de *benchmarking*. Portanto, antes de executar a avaliação dos oito algoritmos (sete do *benchmarking* mais o nosso), adicionamos as segmentações geradas pelo nosso algoritmo ao *benchmark*.

Para comparar as segmentações, a ferramenta de *benchmarking* usa as seguintes quatro métricas:

- *Rand Index* (RI). Mede a probabilidade de dois triângulos pertencerem à mesma ou a diferentes segmentações.
- *Cut Discrepancy* (CD). Intuitivamente, é um método baseado na fronteira e que mede as distâncias entre os cortes definidos pelas fronteiras das regiões (do método e os criados pelo ser humano), de modo que quanto menores forem estas distancias, melhor será a segmentação gerada pelo método .
- *Consistency Error* (CE). Esta métrica tenta determinar semelhanças e diferenças hierárquicas nas segmentações.
- *Hamming Distances* (HD). Mede as diferenças globais entre regiões de duas segmentações. A principal vantagem destas métricas é que elas permitem encontrar correspondências entre regiões produzidas por um algoritmo de segmentação e as segmentações criadas pelos seres humanos.

Em resumo, uma das métricas concentra-se em erros de fronteira (*Cut Discrepancy*), e as outras três focam-se em diferenças de região (*Hamming Distance, Rand Index*, e *Consistency Error*).

A Figura 4 mostra os resultados do nosso algoritmo em comparação com os outros sete algoritmos de benchmarking usando as quatro métricas referidas anteriormente. Note-se que as métricas são consistentes umas com as outras, no sentido de que os algoritmos de segmentação têm o mesmo desempenho relativamente a essas métricas. Como pode ser observado, o algoritmo MeV tem a melhor *média* de desempenho para todas as métricas.

Numa comparação mais detalhada dos diferentes algoritmos de segmentações, usámos a métrica RI, que mede o grau de correspondência da forma da região entre duas

segmentações do mesmo modelo (normalmente uma delas foi criada por seres humanos). A Tabela 1 mostra os resultados comparativos da segmentação para cada categoria de objetos, onde as entradas da tabela apresentam os rankings destes oito algoritmos de acordo com a métrica de RI, em que 1 é o melhor e 8 é o pior. Os resultados mostram que o nosso algoritmo está classificado como o melhor na métrica RI em relação às segmentações criadas pelos seres humanos, uma posição que ele mantém para quase todas as categorias de objetos.

O software de *benchmark* também nos permitiu visualizar um conjunto de imagens resultantes da segmentação MeV para um conjunto de 380 modelos, alguns dos quais se mostram na Figura 1.

Tabela 1. Resultados do *Rand Index*.

Rand Index	Object Category																		Average
	Human	Cup	Glasses	Airplane	Ant	Chair	Octopus	Table	Teddy	Hand	Plier	Fish	Bird	Armadillo	Bust	Mech	Bearing	Vase	FourLeg	
MeV	1	1	2	2	3	3	1	1	1	2	5	1	1	5	2	1	3	1	2	1
RandCuts	2	2	1	3	2	6	5	7	2	1	2	4	2	1	1	6	2	2	3	2
ShapeDiam	4	6	5	1	1	2	2	4	3	6	8	2	3	2	3	3	1	4	1	3
NormCuts	5	3	3	5	4	1	4	2	5	4	4	5	7	7	5	2	4	5	5	4
CoreExtra	7	4	7	7	5	5	3	6	4	3	1	3	4	8	7	7	8	3	6	5
RandWalks	8	5	8	8	6	4	6	3	6	8	6	6	8	4	6	4	7	6	8	7
FitPrim	3	7	6	4	7	8	8	5	7	7	3	8	6	3	4	5	5	7	4	6
KMeans	6	8	4	6	8	7	7	8	8	5	7	7	5	6	8	8	6	8	7	8

11 CONCLUSÕES

Neste artigo apresenta-se um novo algoritmo de segmentação de malhas triangulares, designado por MeV, cuja ideia principal é encontrar, em primeiro lugar, os pontos extremos da malha onde a curvatura atinge um valor máximo local ou um mínimo local, os quais denunciam a presença de montes e vales respectivamente. Aliás, as correspondentes faces de pico e faces de nadir são as faces iniciais de expansão de montes e vales respectivamente.

Note-se que é suficiente usar só o passo de expansão de regiões para segmentar corretamente objetos de forma não-livre, enquanto objetos de forma livre, em alguns casos, precisam da etapa extra de fusão de regiões para mitigar o problema de sobre-segmentação que possa acontecer. No entanto, em muitos dos casos, o excesso de segmentação dos objetos de forma livre já é evitado pela aplicação do conceito de convexidade relaxada na construção das regiões.

No futuro, mesmo tendo em conta que o algoritmo MeV produz segmentações mais próximas da "ground-truth"do que os atuais algoritmos do estado da arte, pretende-se aumentar ainda mais o desempenho em relação a esses algoritmos quer em termos de complexidade quer em relação à diminuição do ruído.

Referências

[Attene 06a] M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, e A. Tal. Mesh segmentation — a comparative study. Em *Proceedings of the IEEE International Conference on Shape Modeling and Applications (SMI'2006)*, páginas 7–19. IEEE Computer Society, 2006.

[Attene 06b] Marco Attene, Bianca Falcidieno, e Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22:181–193, 2006.

[Bajaj 92] C. Bajaj e T. Dey. Convex decomposition of polyhedra and robustness. *SIAM Journal on Computing*, 21(2):339–364, 1992.

[Biederman 87] Irving Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[Capell 02] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, e Zoran Popovic. Interactive skeleton-driven dynamic deformations. *ACM Transactions on Graphics*, 21(3):586–593, 2002.

[Chazelle 81] Bernard M. Chazelle. Convex decompositions of polyhedra. Em *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, STOC '81, páginas 70–79, New York, NY, USA, 1981. ACM.

[Chazelle 95] Bernard Chazelle, David P. Dobkin, Nadia Shouraboura, e Ayellet Tal. Strategies for polyhedral surface decomposition: an experimental study. Em *Proceedings of the 11th Annual Symposium on Computational Geometry*, SCG '95, páginas 297–305, New York, NY, USA, 1995. ACM Press.

[Chen 06] Lijun Chen e NicolasD. Georganas. An efficient and robust algorithm for 3d mesh segmentation. *Multimedia Tools and Applications*, 29(2):109–125, 2006.

[Chen 09] Xiaobai Chen, Aleksey Golovinskiy, e Thomas Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics*, 28(3):73:1–73:12, August 2009.

[Chung 96] Fan Chung. *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1996.

[Garland 01] Michael Garland, Andrew Willmott, e Paul S. Heckbert. Hierarchical face

clustering on polygonal surfaces. Em *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, páginas 49–58, New York, NY, USA, 2001. ACM.

[Ghosh 13] Mukulika Ghosh, Nancy M. Amato, Yanyan Lu, e Jyh-Ming Lien. Fast approximate convex decomposition using relative concavity. *Computer-Aided Design*, 45(2):494 – 504, 2013.

[Golovinskiy 08] Aleksey Golovinskiy e Thomas Funkhouser. Randomized cuts for 3D mesh analysis. *ACM Transactions on Graphics*, 27(5):145:1–145:12, Dezembro 2008.

[Hoffman 97] Donald D. Hoffman e Manish Singh. Salience of visual parts. *Cognition*, 63(1):29 – 78, 1997.

[Kalvin 96] Alan D. Kalvin e Russell H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics & Applications*, 16:64–77, May 1996.

[Katz 03] Sagi Katz e Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, 2003.

[Katz 05] Sagi Katz, George Leifman, e Ayellet Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21:649–658, 2005.

[Kim 10] Hyoungseok B. Kim e Hosook Kim. Mesh segmentation based on local geometric properties. *International Journal of Computer Science and Network Security*, 10(1), 2010.

[Lai 08] Yu-Kun Lai, Shi-Min Hu, Ralph R. Martin, e Paul L. Rosin. Fast mesh segmentation using random walks. Em *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*, SPM '08, páginas 183–191, New York, NY, USA, 2008. ACM.

[Lavoué 05] Guillaume Lavoué, Florent Dupont, e Atilla Baskurt. A new cad mesh segmentation method, based on curvature tensor analysis. *Computer Aided Design*, 37:975–987, September 2005.

[Li 01] Xuetao Li, Tong Wing Woon, Tiow Seng Tan, e Zhiyong Huang. Decomposing polygon meshes for interactive applications. Em *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, páginas 35–42, New York, NY, USA, 2001. ACM.

[Lien 04] Jyh-Ming Lien e Nancy M. Amato. Approximate convex decomposition of polyhedra. Em *ACM SIGGRAPH 2004 Posters*, SIGGRAPH '04, New York, NY, USA, 2004. ACM.

[Lloyd 82] S. Lloyd. Least square quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[Mangan 99] Alan Mangan e Ross Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, October 1999.

[Reuter 09] Martin Reuter, Silvia Biasotti, Daniela Giorgi, Giuseppe Patanè, e Michela Spagnuolo. Discrete laplace-beltrami operators for shape analysis and segmentation. *Computers and Graphics*, 33:381–390, June 2009.

[Shapira 08] L. Shapira, A. Shamir, e D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259, 2008.

[Shlafman 02] Shymon Shlafman, Ayellet Tal, e Sagi Katz. Metamorphosis of polyhedral surfaces using decomposition. *Computer Graphics Forum*, 21(3):219–228, 2002.

[Tilove 80] R.B. Tilove. Set membership classification: a unified approach to geometric intersection problems. *IEEE Transactions on Computers*, C-29(10):874–883, October 1980.

[Zhang 07] Hao Zhang, Oliver van Kaick, e Ramsay Dyer. Spectral methods for mesh processing and analysis, 2007.

[Zuckerberger 02] Emanoil Zuckerberger, Ayellet Tal, e Shymon Shlafman. Polyhedral surface decomposition with applications. *Computers & Graphics*, 26(5):733–743, 2002.