BEDS: Uma Estrutura de Dados para Malhas Triangulares

Frutuoso G. M. Silva Abel J. P. Gomes
Instituto de Telecomunicações
Dep. de Informática - Universidade da Beira Interior
Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã
{fsilva,agomes}@di.ubi.pt

Resumo

As malhas triangulares têm um papel fundamental na Computação Gráfica. Este artigo apresenta uma nova estrutura de dados geométrica para representar malhas triangulares, designada de Butterfly Edge Data Structure (BEDS). Esta estrutura de dados representa apenas os vértices e as arestas, sendo as faces representadas implicitamente por vértices. Esta estrutura de dados implementa uma representação C_2^9 para malhas triangulares, o que significa que tem dois acessos directos e sete indirectos para aceder a toda a informação topológica da malha. Apesar disso, permite o acesso às faces directamente através das arestas como um conjunto de três vértices. Esta estrutura de dados permite também representar malhas triangulares não-manifold.

Palavras-Chave

Malhas triangulares, estruturas de dados.

1. INTRODUÇÃO

As malhas de polígonos sempre foram muito usadas em Computação Gráfica, e em particular as malhas triangulares devido essencialmente à sua simplicidade matemática. Além disso, a cada vez maior complexidade dos modelos geométricos produzidos quer pelas tecnologias de aquisição de dados 3D (ex: *laser scanners*) quer pelos sistemas CAD, e o facto do hardware gráfico estar optimizado para a visualização de triângulos, tem levado também a que muitos modelos 3D sejam aproximados por malhas de triângulos para facilitar a sua manipulação e visualização.

É pois desejável ter uma estrutura de dados geométrica eficiente, quer em termos de memória quer em termos de tempo de processamento, particularmente para modelos com grande número de triângulos, isto é milhões de triângulos.

Muitos dos algoritmos sobre malhas manipulam principalmente vértices e arestas; as faces (ou triângulos) são usadas essencialmente na visualização. Por exemplo, alguns algoritmos de simplificação e refinamento de malhas são baseados nas operações de colapso da aresta e na subdivisão do vértice, respectivamente [Hoppe 96, Garland 97, Silva 04b]. Também, muitas das operações de edição de malhas são aplicadas aos vértices, que directamente afectam as arestas e faces a que pertencem [Lee 99, Lee 02, Silva 04a].

Neste artigo, propõe-se uma nova estrutura de dados geométrica, designada de *Butterfly Edge Data Structure* (BEDS), particularmente adequada para a manipulação de malhas triangulares gigantes, i.e. malhas onde as faces são definidas sem ambiguidades pelos seus vértices, como é o caso dos triângulos. Assim, a BEDS não representa os triângulos explicitamente. Em vez disso, o terceiro vértice de cada triângulo incidente numa aresta é colocado num vector separado de vértices da aresta.

Este artigo está organizado do seguinte modo: O trabalho relacionado aparece na Secção 2. A Secção 3 apresenta a estrutura de dados BEDS. Os resultados e comparações com outras estruturas de dados aparecem na Secção 4. Finalmente, na Secção 5 são tiradas algumas conclusões.

2. TRABALHO RELACIONADO

Muitas estruturas de dados representam uma malha triangular através de um conjunto de faces, cada uma definida por um tuplo de vértices (ex: Triangle list). Isto torna difícil a criação e desenvolvimento de algoritmos rápidos para aceder à informação topológica visto ser necessário efectuar pesquisas globais.

Outras estruturas de dados usam representações baseadas na aresta para representar malhas. Por exemplo, a estrutura *Half-edge* [Mäntylä 88] foi desenhada para tirar partido da estrutura *Winged-edge* [Baumgart 72], ou seja, os B-reps (Boundary Representations) evoluíram de estruturas não orientadas para estruturas orientadas para permitir o acesso rápido à informação topológica. As estruturas orientadas permitem a pesquisa e acesso à informação topológica eficiente mas necessitam de mais memória para representar as arestas porque cada uma é representada por duas semi-arestas orientadas.

Campagna *et al.* [Campagna 98] apresentaram a estrutura *Directed-edges* que é específica para malhas triangulares. No entanto, esta não é uma estrutura concisa pois baseia-se também no conceito da semi-aresta, onde uma aresta é definida por dois vértices mais a informação das arestas seguinte e anterior e ainda a referência à outra aresta (i.e. a outra semi-aresta).

Loop [Loop 00] introduziu a estrutura de dados *Tri-edge* para malhas triangulares. Uma malha é representada por um conjunto de vértices e triângulos, e um triângulo é definido em termos de três vértices e de três arestas. Uma aresta tem a informação sobre a que triângulo pertence e a sua posição no triângulo. Esta é uma estrutura de dados concisa mas pouco eficiente em termos de rapidez porque não permite o acesso directo à informação topológica a partir de vértices e arestas.

Kallmann e Thalmann [Kallmann 01] apresentaram uma estrutura de dados concisa, chamada *Star-Vertex*, para malhas triangulares. É baseada na informação dos vértices e da sua vizinhança. Apesar de ser uma estrutura de dados concisa, o acesso à informação topológica é uma tarefa lenta porque não existe informação explicita sobre arestas e faces.

Também para malhas triangulares Rossignac *et al.* [Rossignac 01] propuseram a estrutura de dados *Corner Table*. Esta estrutura de dados baseia-se nas relações triângulo → vértice e triângulo → triângulo. Um canto é a associação entre um triângulo com um dos seus vértices. Assim um triângulo é definido por um canto mais os cantos seguinte e anterior. Mas esta estrutura de dados foi concebida para permitir uma eficiente compressão/descompressão de malhas triangulares.

Silva e Gomes [Silva 03] apresentaram a estrutura de dados *AIF* que é uma estrutura não orientada para modelos poligonais, triangulares ou não. Esta é uma estrutura de dados concisa e rápida quando comparada com as tradicionais B-reps (ex. *Half-edge*).

Recentemente, Botsch *et al.* [Botsch 08] identificaram duas estruturas de dados especialmente adequadas para o processamento geométrico de malhas, nomeadamente a estrutura *Half-edge* e a *Directed-edges*. Mas se a estrutura *Half-edge* permite representar malhas genéricas, já a estrutura *Directed-edges* só suporta malhas triangulares. Como se verá mais adiante a estrutura de dados proposta neste artigo também é adequada ao processamento geométrico e, além disso, é mais concisa que as estruturas de dados *Half-edge* e *Directed-edges*.

3. BEDS

A *Butterfly Edge Data Structure* (BEDS) é uma representação baseada na aresta que armazena os vértices e as arestas de uma malha triangular. Ou seja, os triângulos não são armazenados explicitamente. Contudo, os triângulos podem ser facilmente gerados com base nos seus vértices que se encontram armazenados na estrutura de dados, por exemplo para efeitos de visualização da malha.

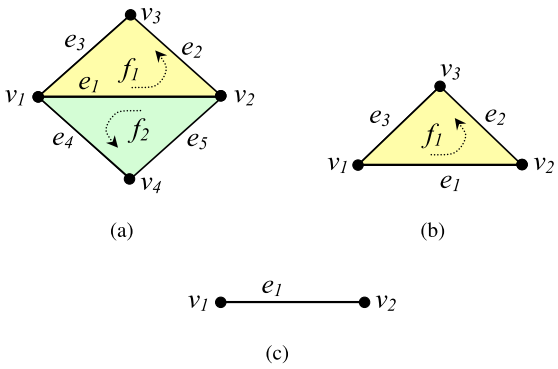


Figura 1. Tipos de arestas possíveis.

3.1. Tipos de Arestas

De uma forma genérica a "*Butterfly*" possui um corpo, a aresta, e duas asas (os triângulos) como mostra a Figura 1(a); por isso se chama estrutura de dados *Butterfly*. Como ilustra a Figura 1 existem três tipos de arestas, nomeadamente:

- *Aresta butterfly* - A aresta *butterfly* possui duas asas que são os triângulos incidentes nela, isto em situações *manifold*. Por exemplo, a aresta e_1 na Figura 1(a) é uma aresta *butterfly*, a qual é definida pelos seus vértices, v_1 e v_2 e pelos dois vértices opostos v_3 e v_4. Os vértices v_1, v_2 e v_3 definem a primeira asa f_1, enquanto v_1, v_2 e v_4 definem a segunda asa f_2.
- *Aresta butterfly com uma única asa* - Uma aresta *butterfly* particular é uma aresta que possui apenas uma única asa. Este caso é ilustrado na Figura 1(b), onde a aresta e_1 é definida apenas pelos seus vértices v_1, v_2 e por um vértice oposto v_3. Neste caso a asa é definida por v_1, v_2 e v_3. Este tipo de arestas é adequado para representar arestas da fronteira em malhas não fechadas.
- *Aresta butterfly sem asas* - Esta é também designada de aresta simples, ou seja, é uma aresta sem asas. Ela é simplesmente definida pelos seus vértices adjacentes, como ilustra a Figura 1(c). Este tipo de aresta permite representar arestas que não possuam faces incidentes, por exemplo em malhas não-*manifold*.

3.2. Esquema de Incidências

Uma malha é definida por um par $M = \{V, E\}$ onde V é um conjunto finito de vértices e E é um conjunto finito de arestas. Note-se que não existe explicitamente um conjunto de faces F na estrutura de dados BEDS.

Dizemos que um vértice v_i é adjacente a uma aresta e_j (simbolicamente, $v_i \prec e_j$) se v_i está contido na fronteira de e_j (ou seja, $Fr(e_j) \cap v_i \neq \emptyset$) e a dimensão de v_i é menor que a dimensão de e_j. Isto é equivalente a dizer que e_j é incidente em v_i (simbolicamente, $e_j \succ v_i$).

O esquema de incidência desta estrutura de dados é baseado na ideia apresentada por Brisson [Brisson 93]. No

entanto, a BEDS não é implementada como um conjunto de tuplos, mas como um conjunto de triângulos que têm o mesmo poder descritivo.

Um vértice $v \subset V$ é definido por $v = \{e_1, e_2, e_3, ..., e_n\}$, ou seja, pelo conjunto de arestas (e_i, com $i = 1, ..., n$) incidentes no vértice.

Uma aresta $e \subset E$ é definida por $e = \{\{v_1, v_2\}, \{v_3, v_4, ..., v_m\}\}$, ou seja por dois conjuntos de vértices. No primeiro conjunto de vértices temos os vértices adjacentes à aresta (v_1 e v_2). Assim cada aresta representa explicitamente a relação $V \prec E$. No segundo conjunto de vértices temos os vértices (v_j, com $j = 1, ..., m$) opostos à aresta. No caso deste segundo conjunto de vértices ser vazio, significa que a aresta e é uma aresta *butterfly* sem asas. Caso contrário, a aresta e é uma aresta *butterfly* com uma, duas ou mais asas. Assim cada vértice oposto à aresta juntamente com os vértices adjacente à aresta formam uma asa, ou seja, um triângulo é definido implicitamente. Isto significa que cada aresta também representa, ainda que de forma implícita, a relação de adjacência $V \prec F$ e a relação de incidência $F \succ E$.

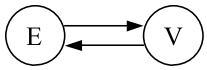


Figura 2. Diagrama da BEDS.

A Figura 2 apresenta o diagrama da BEDS, onde estão as relações explicitamente armazenadas na estrutura de dados, a relação de incidência $E \succ V$ e a relação de adjacência $V \prec E$. Com base na classificação de Weiler [Weiler 88] e de Ni e Bloor [Ni 94], este esquema representa 2 relações em 9 possíveis, ou seja, pertence à classe C_2^9.

Note-se ainda que no caso de objectos com várias componentes disjuntas, cada uma delas será representada por uma malha na estrutura de dados BEDS. Logo uma malha na BEDS não pode ter duas componentes disjuntas.

3.3. Representação da BEDS

A BEDS representa uma malha triangular através de conjuntos de vértices e arestas. Assim usou-se uma classe em C++ para codificar cada tipo de célula da seguinte forma:

```
class Point {
  double x,y,z;    //coordenadas
}
class Vertex {
  int id;              //identificador
  Vector <Edge> li; //arestas incidentes
  Point *pt;           //geometria
}
```

```
class Edge {
  int id;              //identificador
  Vertex *v1, *v2;    //vért. adjacentes
  Vector <Vertex> lv; //vért. opostos
  Vector <Point> lp; //normais
}
class TMesh {
  int id;              //identificador
  Vector <Vertex> vv;  //lista vértices
  Vector <Edge> ev;  //lista arestas
}
```

Um vértice possui a sua geometria (`pt`) e a lista de arestas incidentes (`li`). Uma aresta é definida pelos seus vértices adjacentes (`v1, v2`) e por uma lista de vértices (`lv`) opostos à aresta onde é colocado o terceiro vértice de cada face incidente na aresta. Além disso, a aresta possui ainda uma lista de pontos (`lp`) para armazenar as normais a cada face incidente na aresta. Finalmente, uma malha armazena a lista de vértices (`vv`) e de arestas (`ev`) que a definem.

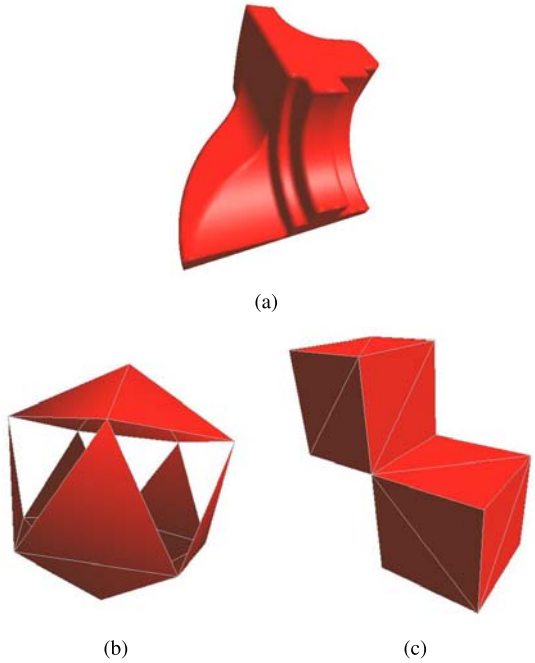


Figura 3. (a) Malha manifold; (b)-(c) Malhas não-manifold.

3.4. Suporte para Malhas Triangulares Genéricas

Como ilustra a Figura 3, a BEDS suporta malhas triangulares genéricas pois permite representar malhas *manifold* e não-*manifold*; a malha da Figura 3(b) é não-*manifold* porque os triângulos à volta de alguns vértices não formam uma vizinhança homeomorfa a 2-*manifold*; a malha da Figura 3(c) possui uma aresta partilhada por dois cubos (ou quatro triângulos), o que significa que também não é *manifold*.

A BEDS é uma estrutura de dados completa porque permite representar sem ambiguidade os objectos. Por exem-

plo, na Figura 3(c) a aresta partilhada pelos dois cubos é representada univocamente na BEDS por:

$$e = \{\{v_1, v_2\}, \{v_3, v_4, v_5, v_6\}\}$$

onde os dois primeiros vértices v_1 e v_2 são adjacentes à aresta, e o segundo conjunto de vértices v_3, v_4, v_5 e v_6 são os vértices opostos à aresta que permitem definir implicitamente as quatro faces incidentes na aresta.

Uma *dangling edge* é representada na BEDS por uma aresta sem asas; um exemplo é a aresta e_1 da Figura 4(b), a qual é representada por:

$$e_1 = \{\{v_1, v_2\}, \{\}\}$$

onde v_1 e v_2 os vértices adjacentes à aresta, e o conjunto de vértices opostos é vazio.

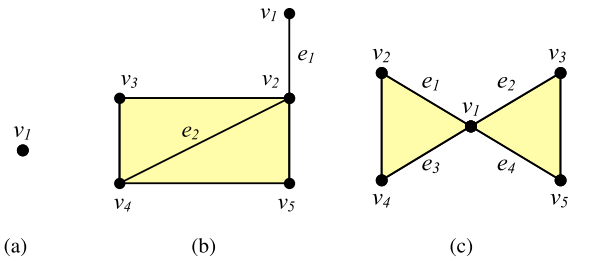


Figura 4. Outras malhas não-manifold.

Um exemplo de uma aresta com uma única asa é a aresta e_1 da Figura 4(c), a qual é definida por:

$$e_1 = \{\{v_1, v_2\}, \{v_4\}\}$$

onde v_1 e v_2 são os vértices adjacentes à aresta, e v_4 é o vértice oposto da aresta e que permite, juntamente com os vértices adjacentes, definir implicitamente a face.

Quanto à representação dos vértices podemos dizer que estes são representados sem ambiguidades através das suas arestas incidentes, visto que temos a informação sobre as asas associadas a cada aresta. Por exemplo, na Figura 4(a) o vértice isolado v_1 não possui arestas incidentes, assim é representado por:

$$v_1 = \{\}.$$

Já o vértice v_1 da Figura 4(c) é definido do seguinte modo:

$$v_1 = \{e_1, e_2, e_3, e_4\}.$$

3.5. Visualização da Malha

A existência dos vértices opostos nas arestas *butterfly* permite-nos a reconstrução das faces, porque um triângulo é definido sem ambiguidades por três vértices. Assim a visualização da malha consiste na criação dos triângulos a partir dos vértices das arestas com asas (ex: através da primitiva GL_TRIANGLE do OpenGL).

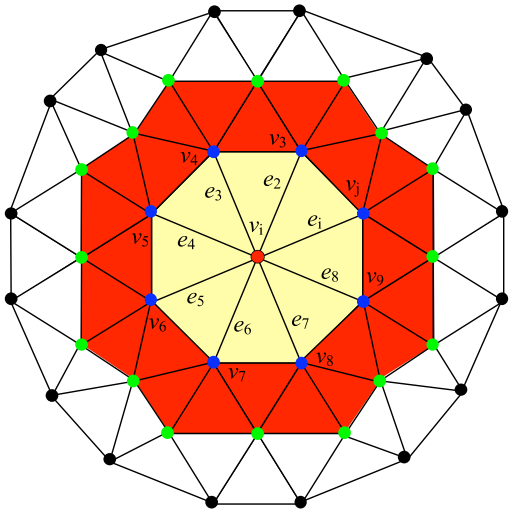


Figura 5. Visualização por níveis.

A visualização de uma malha requer assim o acesso aos vértices de cada asa que implicitamente definem as faces. No entanto, cada asa é visitada três vezes pois as faces são triangulares. De forma a visualizar cada face uma única vez é necessário um algoritmo que permita visitar cada asa também uma única vez, por exemplo marcando as asas visitadas. Com este intuito foi desenvolvido o algoritmo seguinte:

Algoritmo 1
Entrada:
(a) Uma Malha
Início

1. Seleccionar um v arbitrário e colocá-lo numa lista lv
2. Enquanto a lista lv não estiver vazia
3. Seleccionar o primeiro vértice v_i da lista lv (ex: v_i na Figura 5).
4. Aceder à lista de arestas la incidentes no vértice v_i (ex: $e_i, e_2, e_3, e_4, e_5, e_6, e_7, e_8$, na Figura 5).
5. Para cada aresta e_i da lista la
   a) Guardar o vértice v_j de e_i que é diferente de v_i e armazená-lo na lista lv.
6. Para cada aresta e_i da lista la não visitada
   a) Visualizar as suas faces incidentes f_j, j=0,1,..,n (i.e. as suas asas)
   b) Marcar as arestas como visitadas
7. Voltar ao passo nº 2
8. Para cada aresta e_i da lista de arestas da malha
   a) Se existir uma aresta e_k ainda não marcada
   a.1) Colocar um dos seus vértices adjacentes v_i na lista lv
   a.2) Voltar ao passo nº 2

Fim

Estruturas de Dados	Não-manifold	Bytes/△
Triangle list	Sim	18
Star-Vertex	Sim	10+4k
BEDS	Sim	30+2k
Tri-edge	Não	36
Directed-edges	Sim	44
AIF	Sim	42+2k
Winged-edge	Não	60

Tabela 1. Requisitos de Memória.

Estruturas de Dados	Não definidas	Classe
AIF	0	C_4^9
Winged-edge	0	$C_{2,3}^9$
BEDS	3	C_2^9
Directed-edges	3	C_3^9
Triangle list	6	C_2^9
Tri-edge	6	C_2^9
Star-Vertex	6	C_1^9

Tabela 2. Número de relações representadas explicitamente.

Veja-se que com este algoritmo a visualização é efectuada por níveis de triângulos ao longo da malha (Figura 5). Esta técnica de visualização usa a marcação das arestas e o crescimento circular ao longo da malha, garantindo assim que todas as faces são visualizadas e que cada uma é representada uma única vez.

4. RESULTADOS E COMPARAÇÕES

4.1. Requisitos em Termos de Memória

A Tabela 1 compara as diversas estruturas de dados em termos do espaço em memória necessário ao armazenamento de uma malha triangular. Com base na fórmula de Euler (V-E+F=2) uma malha bi-dimensional triangular sem buracos e com n vértices tem cerca de m faces ($m \approx 2n$) e a arestas ($3m$=$2a$).

A segunda coluna da Tabela 1 indica se as estrutura de dados suportam malhas não-*manifold* ou não. A coluna "Bytes/△" mostra o número de bytes necessários para o armazenamento de um triângulo (face). A variável k indica o grau dos vértices, ou seja, o número de arestas incidentes num dado vértice.

A *Triangle list* é a estrutura mais concisa pois apenas armazena os vértices e faces. Ela requer 18 bytes por triângulo, ou seja, $(3 \times 4)n = 12n = 6m$ bytes para as coordenadas dos vértices, e $(3 \times 4)m = 12m$ bytes para as faces (i.e., referência aos três vértices da face). Assumindo que um *float* e um ponteiro requerem 4 bytes cada. No entanto, esta estrutura de dados é normalmente usada apenas para visualização e não para processamento e edição de malhas visto requerer pesquisas globais.

A *Star-vertex* é outra estrutura de dados concisa pois requer apenas 10+4k bytes por triângulo. Esta armazena apenas os vértices e os seus vizinhos. Cada vértice vizinho é definido conjuntamente com um índice de modo a permitir percorrer a malha por uma ordem, neste caso seguindo o sentido contrário dos ponteiros do relógio. Esta estrutura suporta também modelos não-*manifold* mas onde a pesquisa de informação é uma operação pouco eficiente porque não existem explicitamente na estrutura de dados as entidades aresta e face. Consequentemente, seis relações de adjacência/incidência não estão disponíveis como indica a Tabela 2.

A estrutura *Tri-edge* requer 36 bytes por triângulo, ou seja, requer aproximadamente o mesmo que a BEDS mas ela é menos rápida e só suporta malhas *manifold*.

Estruturas de Dados	$V \rightarrow V$	$V \rightarrow E$	$V \rightarrow F$	Total
Triangle list	R	R	R	3R
Star-Vertex	D	D	I	2D+I
BEDS	I	D	I	D+2I
Tri-edge	R	R	R	3R
Directed-edges	R	R	R	3R
AIF	I	D	I	D+2I
Winged-edge	I	I	I	3I

Tabela 3. Acesso baseado no vértice.

De acordo com a fórmula de Euler, uma malha triangular na BEDS requer aproximadamente 36, 38, 40 ou 42 bytes por face, com k=3,4,5 ou 6 respectivamente. Ou seja, requer $(3 \times 4 + 4 \times k)n = (12 + 4k) = (6 + 2k)m$ bytes para as coordenadas dos vértices mais as referências às arestas incidentes no vértice, e $(4 \times 4)a = 16a = 24m$ bytes para as arestas (referências aos quatro vértices). Logo requer $(6 + 2k)m + 24m = (30 + 2k)m$ bytes por triângulo.

As restantes estruturas referidas na Tabela 1, *Direct-edges*, *AIF* e *Winged-edge* requerem mais memória que a BEDS. No entanto, as estruturas *Direct-edges* e *AIF* são também mais rápidas que a BEDS.

4.2. Rapidez de Acesso à Informação

Uma estrutura de dados B-rep é definida por um conjunto de adjacências e incidências entre entidades topológicas. Nove relações são possíveis com base em três entidades (vértice, aresta e face, ver [Weiler 85]) e podem ser usadas para avaliar a rapidez de uma estrutura de dados no acesso à informação topológica. Com estas três entidades topológicas, podem formar-se várias classes de incidência para as estruturas de dados, entre C_1^9 e C_9^9 (ver [Woo 85]). No entanto, as estruturas analisadas neste trabalho pertencem às seguintes classes: C_4^9, C_3^9, $C_{2,3}^9$, C_2^9 e C_1^9.

A Tabela 2 mostra o número de relações explicitamente representadas nas várias estruturas de dados, as quais permitem o acesso a toda a informação topológica. A coluna "Classe" mostra o número de relações que estão explicitamente representadas na estrutura, dentro das nove

Estruturas de Dados	$E \rightarrow V$	$E \rightarrow E$	$E \rightarrow F$	Total
Triangle list	R	R	R	3R
Star-Vertex	R	R	R	3R
BEDS	D	I	D	2D+I
Tri-edge	R	R	R	3R
Directed-edges	D	R	D	2D+R
AIF	D	I	D	2D+I
Winged-edge	D	I	D	2D+I

Tabela 4. Acesso baseado na aresta.

Estruturas de Dados	$F \rightarrow V$	$F \rightarrow E$	$F \rightarrow F$	Total
Triangle list	D	D	R	2D+R
Star-Vertex	R	R	R	3R
BEDS	R	R	R	3R
Tri-edge	D	D	R	2D+R
Directed-edges	R	D	I	D+I+R
AIF	I	D	I	D+2I
Winged-edge	I	I	I	3I

Tabela 5. Acesso baseado na face.

possíveis. Já a coluna "Não definidas" expressa o número de relações que não estão definidas na estrutura de dados porque as entidades pura e simplesmente não existem na estrutura de dados (ex: face na estrutura BEDS). Assim é necessário inferir primeiramente essas entidades para depois se poder aceder à informação topológica com base nelas, e por isso, requerem um custo adicional de processamento.

Quanto maior for o número de relações explicitamente representadas numa estrutura de dados, mais memória esta necessita. Mas por outro lado, mais rápido é o acesso à informação topológica.

As Tabelas 3, 4 e 5 caracterizam as diversas estruturas de dados em termos da eficiência do acesso à informação topológica para uma malha triangular com n vértices. A Tabela 3 apresenta os acessos baseados na entidade vértice, enquanto a Tabela 4 apresenta os acessos baseados na entidade aresta. Finalmente a Tabela 5 apresenta os acessos baseados na entidade face. Estas Tabelas caracterizam as relações em termos de acessos directos (D), indirectos (I) e que envolvam reconstrução (R). Assumindo que um acesso directo é mais rápido que um acesso indirecto, e que um acesso indirecto é mais rápido que um acesso que envolve reconstrução, podemos assim comparar a rapidez de cada estrutura de dados. Por exemplo, se considerarmos que um acesso directo corresponde a t, onde t é uma unidade de tempo, e um acesso indirecto corresponde pelo menos a $2t$, e que um acesso que envolva reconstrução corresponde pelo menos a $3t$, podemos assim calcular o tempo total para o acesso a toda a informação topológica para as diferentes estruturas de dados.

A Tabela 6 apresenta os tempos totais para cada estrutura

Estruturas de Dados	V+E+F	Tempo total
AIF	4D+5I	14t
Winged-edge	2D+7I	16t
BEDS	3D+3I+3R	18t
Directed-edges	3D+I+4R	19t
Star-Vertex	2D+I+6R	22t
Triangle list	2D+7R	23t
Tri-edge	2D+7R	23t

Tabela 6. Tempos totais de acesso a toda a informação topológica.

com base nos três tipos de acesso considerados. Note-se que um acesso é directo quando corresponde a uma relação que está explicitamente armazenada na estrutura de dados. Já um acesso é considerado indirecto quando recorre a uma segunda entidade para aceder à informação. Por último, um acesso que envolve reconstrução indica que é necessário primeiro criar a entidade pois ela não existe explicitamente na estrutura de dados.

A estrutura de dados mais rápida é a *AIF*, pois requer apenas $14t$ para aceder a toda a informação topológica, como se pode ver na Tabela 6. Isto deve-se ao facto de possuir mais relações explicitamente representadas na estrutura de dados. Mas por outro lado, é também menos concisa que a generalidade da estruturas de dados como se pode confirmar na Tabela 1.

A estrutura *Winged-edge* pertence à classe $C^9_{2,3}$, o que significa que possui apenas duas relações totais e três relações parciais explicitamente representadas. Por isso é mais rápida que a BEDS pois requer apenas $16t$ em vez de $18t$ requeridos pela BEDS para aceder a toda a informação topológica. Mas por isso, a *Winged-edge* necessita também de mais memória que a BEDS (ver Tabela 1).

A BEDS é a terceira estrutura de dados mais rápida, pois requer $18t$ para aceder a toda a informação topológica. No entanto, apesar de não incluir a entidade face permite também inferir a informação sobre as faces, definidas pelos seus vértices, a partir das arestas. Além disso é também a terceira estrutura de dados mais concisa.

A quarta estrutura mais rápida é a *Directed-edges*, requer $19t$, mas não é uma estrutura de dados concisa como se pode ver pela Tabela 1.

A estrutura de dados *Star-vertex* pertence à classe C^9_1, o que significa que possui apenas uma relação explicitamente representada. As restantes oito relações de adjacência e incidência estão representadas implicitamente, o que requer um mecanismo de inferência para aceder à informação envolvida nestas relações. Como não possui as entidades aresta e face explicitamente representadas, significa que seis relações de adjacência e incidência não estejam na estrutura de dados como se pode ver na Tabela 2. Consequentemente, esta estrutura não é rápida pois requer $22t$ para aceder a toda a informação topológica.

As restantes estruturas de dados, a *Triangle List* e a *Triedge* também não são rápidas pois requerem $23t$. No entanto, são ambas estruturas de dados concisas.

4.3. Resultados Práticos

Para uma avaliação de resultados foi desenvolvido um algoritmo de simplificação de malhas sobre a BEDS. Este algoritmo é baseado na operação de colapso da aresta, que neste caso colapsa uma aresta num vértice que corresponde ao ponto médio da aresta. Ou seja, o algoritmo implementado foi o algoritmo *NSA* [Silva 04b] sobre a estrutura de dados BEDS.

Malha	Original	Simplificada
Vaca	5804 F	2436 F
Coelho	69473 F	29166 F
Dragão	871414 F	365446 F

Tabela 7. Malhas usadas nos testes e representadas na Figura 6.

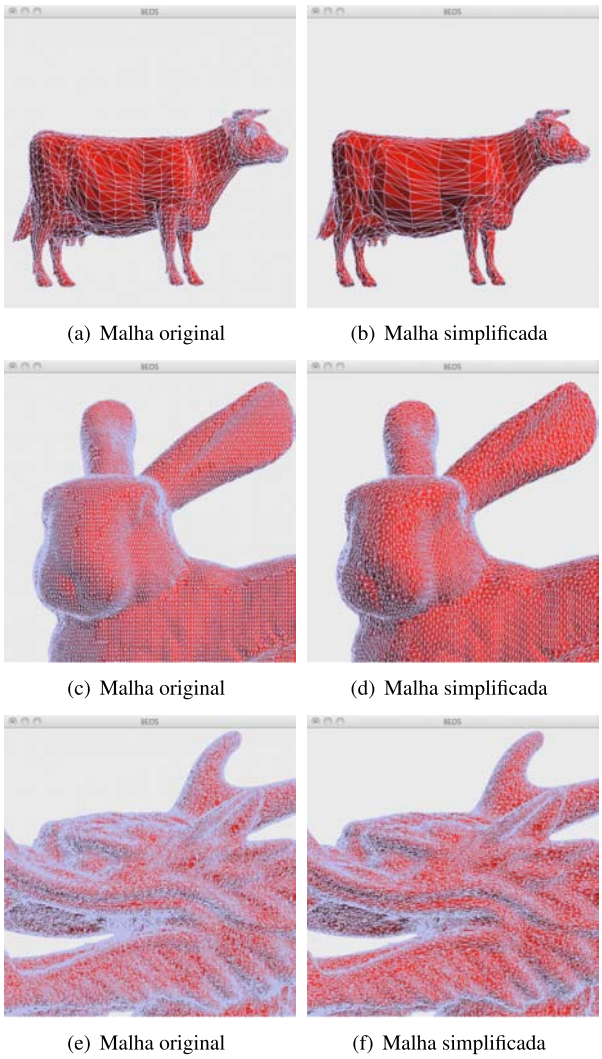
Para os testes foram usadas malhas bem conhecidas, como a vaca, o coelho e o dragão como ilustra a Figura 6. A Tabela 7 apresenta as características destas malhas, nomeadamente o número de faces das malhas originais e o número de faces das versões simplificadas criadas pelo nosso algoritmo de simplificação.

Para avaliar os requisitos de memória da estrutura de dados BEDS usámos a estrutura AIF como referência para as comparações, isto porque estas duas estruturas de dados são semelhantes em termos dos vértices. Por isso não é necessário a comparação em termos da entidade vértice pois esta requer o mesmo espaço em memória em ambas as estruturas de dados. Assim, para as malhas de teste, calculou-se a memória necessária para o seu armazenamento nas duas estruturas de dados, BEDS e *AIF*, com base no número de entidades de cada malha.

Malha	#V	#A	#F
Vaca	2904	8706	5804
Coelho	34835	104 310	69473
Dragão	437645	1309256	871414

Tabela 8. Número de entidades das malhas representadas na Figura 6.

A Tabela 8 apresenta o número de vértices ($\#V$), arestas ($\#A$) e faces ($\#F$) necessários ao armazenamento das malhas de teste na estrutura de dados *AIF*. A representação destas malhas na BEDS requer o mesmo número de vértices e o número de arestas *butterfly* corresponde também ao número de arestas, mas neste caso não existem as faces.



(a) Malha original (b) Malha simplificada

(c) Malha original (d) Malha simplificada

(e) Malha original (f) Malha simplificada

Figura 6. Três malhas usadas nos testes. Versão original e versão simplificada com o algoritmo de simplificação *NSA*.

Por exemplo, para representar a malha da vaca na estrutura *AIF*, e assumindo que um ponteiro requer 4 bytes, temos:
- Para as arestas que referenciam dois vértices e duas faces incidentes (no caso de malhas *manifold*)
$8706 \times 2 \times 2 \times 4 = 139296$ bytes
- Para as faces que referenciam três arestas
$5804 \times 3 \times 4 = 69648$ bytes
Logo a malha da vaca na estrutura *AIF* requer $139296 + 69648 = 208944$ bytes, isto sem considerar o espaço necessário aos vértices pois é igual em ambas as estruturas de dados.

Na BEDS a malha da vaca requer:
- Para as arestas *butterfly* que referenciam quatro vértices
$8706 \times 4 \times 4 = 139296$ bytes
Logo na BEDS esta malha requer apenas 139296 bytes, o que significa que requer menos 33% de espaço em memória do que na AIF.

Como se pode ver na Tabela 9, no caso da malha do coelho,

esta requer 2 502 636 bytes na estrutura *AIF* e apenas 1 668 960 bytes na estrutura de dados BEDS, ou seja, também uma redução de 33%. De forma similar, a malha do dragão requer menos 33% de espaço em memória na BEDS em relação à *AIF*, ou seja, 20 948 096 bytes na BEDS e 31 405 064 bytes na *AIF*.

Resumindo, uma malha triangular na estrutura de dados BEDS requer menos 33% do espaço em memória do que na estrutura de dados *AIF*. Este facto é principalmente relevante para malhas com grande número de vértices devido às limitações de memória.

Malha	*AIF*	BEDS
Vaca	208944 bytes	139296 bytes
Coelho	2 502 636 bytes	1 668 960 bytes
Dragão	31 405 064 bytes	20 948 096 bytes

Tabela 9. Espaço ocupado pelas malhas dos testes nas estruturas *AIF* e BEDS.

5. CONCLUSÕES

A BEDS é uma estrutura de dados que mantém informação à cerca das faces na representação dos vértices na estrutura de dados, e por isso não possui a entidade face. Por esta razão é uma estrutura de dados concisa.

Em termos de rapidez esta estrutura de dados tem um bom desempenho na classe C_2^9, ou seja tem dois acessos directos e sete acessos indirectos à informação topológica. No entanto, também permite a reconstrução das faces directamente das arestas como um conjunto de três vértices.

De entre as estruturas de dados analisadas, apenas as estruturas *AIF* e *Winged-edge* são mais rápidas que a BEDS. No entanto, a BEDS é mais concisa que estas estruturas de dados. Além disso podemos dizer que a BEDS é uma estrutura de dados genérica pois permite suportar também malhas não-*manifold*.

Em resumo, podemos dizer que a BEDS é uma alternativa para aplicações que necessitam de manipular grandes malhas triangulares. De facto, a BEDS é genérica, concisa e rápida quando comparada com as outras estruturas de dados que possuem o mesmo poder de representação.

No futuro pretende-se implementar algoritmos já implementados noutras estruturas de dados (ex: algoritmos de edição, etc.) de forma a comparar o tempo de execução de ambas as versões.

Referências

[Baumgart 72] B. G. Baumgart. Winged-edge polyhedron representation. Relatório técnico, STAN-CS-320, Stanford University, 1972.

[Botsch 08] Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Levy, Stephan Bischoff, e Christian Rossl. Geometric modeling based on polygonal meshes. Em *Eurographics Tutorial*, 2008.

[Brisson 93] E. Brisson. Representing geometric structures in d dimension: topology and order. *Discrete & Computational Geometry*, 9(4):387–426, 1993.

[Campagna 98] Swen Campagna, Leif Kobbelt, e Hans-Peter Seidel. Directed edges — A scalable representation for triangle meshes. *Journal of Graphics Tools*, 3(4):1–12, 1998.

[Garland 97] Michael Garland e Paul S. Heckbert. Surface simplification using quadric error metrics. Em *Proceeding of Siggraph*, páginas 209–216. ACM Press, 1997.

[Hoppe 96] Hugues Hoppe. Progressive meshes. Em *Proceeding of Siggraph*, páginas 99–108. ACM Press, 1996.

[Kallmann 01] Marcelo Kallmann e Daniel Thalmann. Star-vertices: A compact representation for planar meshes with adjacency information. *Journal of Graphics Tools*, 6(1):7–18, 2001.

[Lee 99] Seungyong Lee. Interactive multiresolution editing of arbitrary meshes. Em *Computer Graphics Forum*, volume 18(3), páginas 73–92, 1999.

[Lee 02] Yunjin Lee e Seungyong Lee. Geometric snakes for triangular meshes. Em *Computer Graphics Forum*, volume 21(3), páginas 229–238, 2002.

[Loop 00] Charles Loop. Managing adjacency in triangular meshes. Relatório Técnico No. MSR-TR-2000-24, Microsoft Research, 2000.

[Mäntylä 88] Martti Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, 1988.

[Ni 94] Xiujun Ni e M. Susan Bloor. Performance evaluation of boundary data structures. *IEEE Computer Graphics and Applications*, 14(6):66–77, 1994.

[Rossignac 01] Jarek Rossignac, Alla Safonova, e Andrzej Szymczak. 3d compression made simple: Edgebreaker on a corner table. Em *Proceedings of Shape Modeling International Conference*, páginas 278–283, 2001.

[Silva 03] Frutuoso G. M. Silva e Abel J. P. Gomes. AIF - a data structure for polygonal meshes. Em *Lecture Notes in Computer Science, Vol. 2669, Part III, V. Kumar, M. Gravilova, C. Tan and P. L'Ecuyer (eds.)*, páginas 478–487. Springer-Verlag, 2003.

[Silva 04a] Frutuoso G. M. Silva e Abel J. P. Gomes. Interactive editing of multiresolution meshes. Em *Proceedings of XVII Brazilian Symposium on Computer Graphics and Image Processing / II Ibero-American Symposium on Computer Graphics (SIBGRAPI/SIACG'04)*, páginas 202–209. IEEE Computer Society Press, 2004.

[Silva 04b] Frutuoso G. M. Silva e Abel J. P. Gomes. Normal-based simplification algorithm for meshes. Em *Proceedings of Theory and Practice of Computer Graphics (TPCG'04)*, páginas 211–218. IEEE Computer Society Press, 2004.

[Weiler 85] Kevin Weiler. Edge-based data structure for solid modelling in curved-surface environments. *IEEE Computer Graphics & Applications*, 5(1):21–40, 1985.

[Weiler 88] Kevin Weiler. The radial edge structure: a topological representation for non-manifold geometric boundary modelling. *Geometric Modelling for CAD applications*, páginas 3–36, 1988.

[Woo 85] Anthony C. Woo. A combinatorial analysis of boundary data structure schemata. *IEEE Computer Graphics & Applications*, 5(3):19–27, 1985.