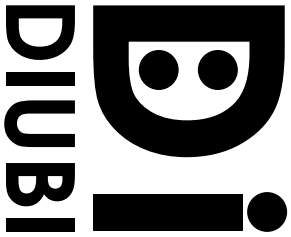
Universidade da Beira Interior
Departamento de Informática

Estruturas Poligonais Deformáveis com Resolução Variável

Frutuoso Gomes Mendes da Silva

Tese submetida para obtenção do grau de
Doutor em Engenharia Informática

Abril 2005



Departamento de Informática
Universidade da Beira Interior

Estruturas Poligonais Deformáveis com Resolução Variável

Frutuoso Gomes Mendes da Silva

Tese realizada sob a orientação do
Prof. Doutor Abel João Padrão Gomes

Abril de 2005

À Graça e aos meus filhos,

Patrícia e Pedro

Abstract

This thesis focuses on data structures and algorithms for multiresolution polygonal meshes.

Polygonal meshes have gained an increasing importance in computer graphics due to recent development and use of data acquisition technologies like the 3D scanners. However, meshes generated by these 3D acquisition devices normally possess redundant information. This is due to high resolutions allowed for those devices, which are used in a uniform manner independently of the shape features of the objects. Thus, it is sometimes necessary to simplify a mesh to reduce the amount of data to be processed.

In this context, this thesis introduces a concise and unambiguous data structure, called AIF data structure to accommodate general polygonal meshes, i.e. meshes that are not necessarily triangular and manifold. In addition, its descriptive power enables fast access to topological information about adjacency and incidence relationships.

Also an AIF-based multiresolution scheme is described in this thesis. For that, a new simplification algorithm, called NSA algorithm, for meshes has been proposed using the edge collapse operation. The main novelty of this algorithm is to use the same criterion to select and validate the edge to be collapsed. As usual, the refinement of the mesh is based on the vertex split operation.

Finally, a new methodology for the interactive editing of meshes is introduced. It is based on the concept of sub-mesh. This way, a mesh can be structured and viewed as a hierarchy of sub-meshes, what facilitates its editing. This is important because any scanner-generated mesh is always represented as a whole, having no means of distinguishing a region from another; for example, a leg from a head of a bunny mesh.

Resumo

O trabalho descrito nesta tese centra-se no desenvolvimento e codificação de estruturas de dados e algoritmos para a representação e processamento de malhas poligonais multiresolução.

As malhas poligonais são cada vez mais usadas na computação gráfica devido ao recente e crescente desenvolvimento das tecnologias de aquisição de dados 3D como, por exemplo, os *scanners* tridimensionais. No entanto, as malhas geradas por estes dispositivos possuem normalmente informação redundante. Isto deve-se ao facto da cada vez maior resolução dos sistemas de aquisição de dados, a qual é aplicada uniforme e independentemente das características de forma dos objectos. Por isso, é muitas vezes necessário simplificar as malhas de modo a reduzir a quantidade de informação a processar.

Neste contexto, esta tese introduz uma estrutura de dados concisa, designada por AIF para representar malhas poligonais genéricas, ou seja, malhas não necessariamente triangulares e *manifold*. Além disso, o seu poder descritivo permite acessos rápidos à informação topológica referente a adjacências e incidências.

É descrito ainda um esquema multiresolução baseado na estrutura de dados AIF. Para isso, foi proposto um novo algoritmo de simplificação de malhas, designado por algoritmo NSA, que usa a operação de contracção de arestas para simplificar a malha. A principal novidade deste algoritmo é usar o mesmo critério para a escolha e validação da aresta a contrair. O refinamento da malha é efectuado recorrendo à subdivisão do vértice, que é a operação inversa da contracção de arestas.

Finalmente, introduziu-se uma nova metodologia para a edição interactiva de malhas poligonais. Esta metodologia baseia-se no conceito de sub-malha. Desta forma, uma malha pode ser estruturada e vista como uma hierarquia de sub-malhas, o que circunscreve e facilita a sua edição. Este facto é importante porque qualquer malha gerada por um *scanner* 3D é sempre vista como um todo, na qual não é possível distinguir uma região de outra; por exemplo, não é possível distinguir uma perna de um coelho relativamente à sua cabeça na malha que o representa.

Agradecimentos

A todos os que directa ou indirectamente tornaram possível a realização desta tese, e em especial ao meu orientador, Prof. Doutor Abel João Padrão Gomes, pela confiança e apoio prestados ao longo destes anos.

Aos meus colegas de gabinete do Departamento de Informática, bem como ao meu colega Francisco Morgado, pelo debate de ideias e apoio manifestado durante a realização deste trabalho.

À UBI e ao Departamento de Informática, ao Grupo de Redes e Multimédia do IT e ao PRODEP III (Acção 5.3 - Formação Avançada de Docentes do Ensino Superior) pelos apoios concedidos, sem os quais este trabalho teria sido muito mais difícil de concretizar.

Às seguintes empresas Avalon Archive, Cyberware, Polygon Technology GmbH, Stanford University e 3D Cafe, que gentil e gratuitamente disponibilizam um conjunto de modelos para fins não comerciais, no qual estão incluídos todos os modelos utilizados na realização do trabalho descrito nesta tese.

Quero por fim agradecer à minha esposa pelo apoio e compreensão manifestados ao longo destes anos contribuindo assim para o meu equilíbrio emocional.

Terminologia

Aqui são apresentados os termos e abreviaturas usados no decorrer desta tese.

ACM - Association for Computing Machinery

AIF - Adjacency and Incidence Framework.

B-rep - Boundary representation.

BSP - Bi-Star Planarity Algorithm.

CAD - Computer Aided Design.

CSG - Constructive Solid Geometry

GL - Graphics Library.

IEEE - Institute of Electrical and Electronics Engineers

JPEG - Joint Photographic Experts Group.

LOD - Level-Of-Detail.

NSA - Normal-based Simplification Algorithm.

SMF - Shape Mesh File.

Surfels - Surface Elements as Rendering Primitives.

VRML - Virtual Reality Modelling Language.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este capítulo apresenta a motivação e os objectivos que estiveram na origem deste trabalho, bem como os principais resultados obtidos durante a sua elaboração. A estrutura da tese aparece descrita no final do capítulo.

1.1 Motivação e Objectivos

Uma das principais razões que levou à realização deste trabalho foi o facto de a análise multiresolução ser cada vez mais utilizada na área da computação gráfica, nomeadamente aplicada às malhas poligonais. Normalmente a análise multiresolução está associada à matemática, pois permite descrever uma função para diferentes resoluções. Basicamente, uma função é aproximada por outra função que é conhecida como onduleta (*wavelet*) e por um conjunto de coeficientes que permitem recuperar a função original.

A utilização da análise multiresolução em computação gráfica iniciou-se na área da compressão de imagem com a utilização de onduletas. Hoje as onduletas são usadas no formato JPEG [47]. Recentemente a análise multiresolução começou a ser utilizada também noutros campos da computação gráfica, como a visualização, a compressão, a transmissão e edição de malhas poligonais.

O trabalho de investigação desenvolvido nesta tese centra-se precisamente na análise multiresolução aplicada a malhas poligonais. Mais concretamente, apresentam-se mecanismos de multiresolução aplicados à visualização e edição de malhas poligonais embora possam ser aplicados a muitas outras áreas como, por exemplo, na compressão e transmissão de malhas.

A análise multiresolução poderá no futuro vir a ser uma alternativa aos modelos LOD (*Level Of Detail*) que são hoje usados em animação, ambientes virtuais e jogos de computador. Ao contrário dos modelos LOD que possuem diferentes níveis discretos de detalhe em simultâneo armazenados em memória, a multiresolução permite ter

modelos com diferentes níveis contínuos de detalhe mantendo apenas um único modelo armazenado em memória. Neste caso, há que ter algoritmos de simplificação e refinamento para se obter um modelo mais grosseiro ou mais detalhado, respectivamente.

1.2 Contribuições da Tese

O trabalho desta tese enquadra-se nas malhas poligonais multiresolução, também designadas de malhas de resolução variável. No desenvolvimento do trabalho de investigação foi possível identificar três aspectos distintos da análise multiresolução aplicada a malhas poligonais. O primeiro aspecto refere-se às estruturas de dados geométricas usadas para representar malhas poligonais. O segundo aspecto tem que ver com os algoritmos de simplificação e refinamento de malhas poligonais. O terceiro e último aspecto refere-se à edição interactiva de malhas poligonais. Assim, pode dizer-se que as principais contribuições da tese são as seguintes:

- A introdução de uma nova estrutura de dados para representar malhas, designada por estrutura de dados AIF (*Adjacency and Incidence Framework*). É uma estrutura de dados concisa e que permite o acesso rápido à informação topológica, sendo por isso também adequada à multiresolução. Além disso, é uma estrutura de dados genérica no sentido em que suporta a representação de malhas não necessariamente triangulares e topologicamente equivalentes a variedades (*manifolds*) bidimensionais. É ainda uma estrutura de dados simples de usar e manipular, pois só usa um único operador de pesquisa para aceder a *toda* a informação topológica de adjacências e incidências.

- A introdução de um novo algoritmo de simplificação de malhas poligonais baseado na operação de contracção de arestas. Como se verá, este algoritmo é o mais rápido dentro da sua classe (a contracção de arestas), o que em parte se deve ao facto de usar o mesmo critério para a escolha e validação da aresta a contrair. Além disso, permite reduzir o número de faces de uma malha para cerca de metade numa única simplificação.

- A descrição de uma nova metodologia para a edição interactiva de malhas poligonais. A principal ideia subjacente é a divisão de uma malha em sub-malhas, sub-malhas estas que podem depois ser editadas e manipuladas independentemente sem afectar o resto da malha. Esta metodologia pode ser aplicada a malhas multiresolução. A edição de malhas multiresolução permite *simplificar*, *editar* e depois *refinar* a malha para a sua resolução original. Deste modo podem editar-se malhas com grande número de células, visto que a edição é efectuada numa versão simplificada da malha.

1.3 Artigos Publicados

Esta tese contém material que já foi apresentado na forma de artigos científicos em conferências nacionais e internacionais de computação gráfica, alguns dos quais publicados pela ACM, IEEE e Springer-Verlag, nomeadamente:

- Frutuoso G. M. Silva and Abel J. P. Gomes. Adjacency and Incidence Framework - A data structure for efficient and fast management of multiresolution meshes. In *Proceedings of International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE'03)*, ACM Press, pp.159-166, 2003.
`http://doi.acm.org/10.1145/604471.604503`

- Frutuoso G. M. Silva and Abel J. P. Gomes. AIF - A data structure for polygonal meshes. In *Proceedings of 2nd International Workshop on Computer Graphics and Geometric Modeling (CGGM'03)*, included in *Proceedings of Computational Science and Its Applications (ICCSA'03)*, Lecture Notes in Computer Science, Vol. 2669, Part III, V. Kumar, M. Gravilova, C. Tan and P. L'Ecuyer (eds.), Springer-Verlag Press, pp.478-487, 2003.
`http://www.springerlink.com/openurl.asp?genre=article`
`&issn=0302-9743&volume=2669&spage=478`

- Frutuoso G. M. Silva and Abel J. P. Gomes. Interactive editing of arbitrary sub-meshes. In *Proceedings of Computer Graphics International (CGI'03)*, IEEE Computer Society Press, pp.218-221, 2003.
`http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=27295&page=2`

- Frutuoso G. M. Silva and Abel J. P. Gomes. AIF - Uma estrutura de dados B-rep para modelos Poligonais. *Actas do 12º Encontro Português de Computação Gráfica (EPCG'03)*, pp.1-7, 2003.
`http://virtual.inesc.pt/12epcg/papers/01.html`

- Frutuoso G. M. Silva and Abel J. P. Gomes. Normal-based simplification algorithm for meshes. In *Proceedings of Theory and Practice of Computer Graphics (TPCG'04)*, IEEE Computer Society Press, pp.211-218, 2004.
`http://csdl.computer.org/comp/proceedings/tpcg/2004/2137/00/`
`21370211abs.htm`

- Frutuoso G. M. Silva and Abel J. P. Gomes. Interactive editing of multiresolution meshes. In *Proceedings of XVII Brazilian Symposium on Computer Graphics and Image Processing / II Ibero-American Symposium on Computer Graphics (SIBGRAPI/SIACG'04)*, IEEE Computer Society Press, pp.202-209, 2004.
`http://csdl.computer.org/comp/proceedings/sibgrapi/2004/2227/00/`
`22270202abs.htm`

- Frutuoso G. M. Silva and Abel J. P. Gomes. A B-rep data structure for polygonal meshes. *VIRTUAL - The Portuguese Journal of Computer Graphics*, Special edition - Advances in Computer Graphics in Portugal, Adérito Marcos and Miguel Salles Dias (eds.), Novembro 2004.
`http://virtual.inesc.pt/aicg04/papers/01.html`

1.4 Estrutura da Tese

Esta tese é composta por mais cinco capítulos para além do actual, nomeadamente:

No Capítulo 2 faz-se um levantamento do estado da arte respeitante a malhas poligonais em três vertentes fundamentais: (i) estruturas de dados geométricas, (ii) análise multiresolução e (iii) edição de malhas poligonais.

No Capítulo 3 descreve-se a estrutura de dados AIF (*Adjacency and Incidence Framework*) que foi propositadamente desenhada para representar malhas poligonais. Como se verá, esta estrutura de dados também poderá ser usada como uma estrutura de dados B-rep *non-manifold*.

No Capítulo 4 descreve-se um novo algoritmo de simplificação de malhas poligonais baseado na operação de contracção de arestas, que permite criar uma sequência de malhas para diferentes níveis de resolução, designado por algoritmo NSA (*Normal-based Simplification Algorithm*).

No Capítulo 5 descreve-se uma nova metodologia para a edição interactiva de malhas poligonais com base no conceito de sub-malha. Esta nova metodologia é aplicável quer a malhas usuais quer a malhas multiresolução.

A tese termina com o Capítulo 6, no qual se apresenta as conclusões mais significativas sobre os resultados obtidos, assim como se aponta novas direcções para o trabalho futuro.

Capítulo 2

Malhas Poligonais: O Estado da Arte

Este capítulo faz uma introdução ao tema da tese e um resumo do estado da arte em relação às malhas poligonais, nomeadamente no que diz respeito às estruturas de dados geométricas usadas para as representar, à análise multiresolução e à edição interactiva de malhas poligonais.

2.1 Introdução

As malhas poligonais têm tido uma utilização crescente em computação gráfica, em parte devido à sua versatilidade na representação de objectos tridimensionais. As malhas são hoje usadas numa grande variedade de aplicações, entre as quais se contam as aplicações de CAD, visualização, ambientes virtuais, animação e jogos de computador. A isto não é estranho o facto de os polígonos e malhas poligonais serem as principais primitivas suportadas pelo hardware gráfico e pelo software de visualização.

Nos últimos anos muita investigação em malhas poligonais foi levada a cabo, em particular nos seguintes tópicos: representação de malhas [23, 72], simplificação de malhas [31, 32, 30, 41, 19, 34, 37, 68, 126], simplificação de malhas de grandes dimensões [46, 65, 67], refinamento de malhas dependente do contexto [40, 39, 52, 24, 81], edição de malhas multiresolução [129, 61, 6] e, ainda, compressão e transmissão de malhas [33, 114, 116, 45].

Os recentes avanços na tecnologia de aquisição de dados 3D permitem produzir facilmente uma grande variedade de malhas poligonais a partir de objectos reais. No entanto, estas malhas possuem normalmente uma enorme quantidade de polígonos, o que dificulta depois a sua eficiente manipulação. O desempenho dos algoritmos que manipulam malhas poligonais depende em primeiro lugar da quantidade de células da malha. Para obviar este problema recorre-se à simplificação das malhas, embora

a) 69473 faces. b) 36767 faces. c) 11229 faces.

Figura 2.1: Malha de um coelho para diferentes tamanhos com três níveis de detalhe.

com alguma perda de qualidade da malha. No entanto, muitas vezes essa perda de qualidade não é perceptível ao utilizador, ou seja, não é perceptível para o olho humano.

A manipulação eficiente de malhas de grandes dimensões é conseguida através do uso de múltiplas representações com resolução variável para cada modelo (modelos LOD), ou seja, modelos com diferentes níveis discretos de detalhe. Neste caso, o nível de detalhe apropriado é escolhido com base em factores como a distância ao observador, a visibilidade, a importância, etc. Por exemplo, a Figura 2.1 apresenta várias malhas de um coelho com três tamanhos e três níveis de resolução diferentes. Quando a distância ao observador aumenta, o tamanho da malha diminui, pelo que é conveniente diminuir a resolução da malha já que a perda de detalhe torna-se imperceptível, podendo ser utilizadas versões simplificadas como ilustra a Figura 2.1 com as malhas mais claras na diagonal.

Os modelos LOD são ainda hoje a forma mais utilizada para a manipulação de malhas poligonais de grandes dimensões. No entanto, a análise multiresolução veio introduzir uma representação alternativa aos modelos LOD. A análise multiresolução permite representar um modelo com diferentes níveis contínuos de detalhe, embora exista uma

única malha em memória em cada instante. Depois, com o auxílio de algoritmos de simplificação e refinamento faz-se a passagem de um nível de detalhe para o seguinte e para o anterior, consoante as necessidades da aplicação. Contudo, o desempenho destes algoritmos ainda limita a sua utilização em aplicações de tempo real que necessitam de altas taxas de refrescamento (*frames rates*).

O desempenho dos algoritmos de simplificação e refinamento depende não só da sua complexidade, mas também da estrutura de dados utilizada, visto ser ela que disponibiliza a informação aos algoritmos. Assim, as questões da análise multiresolução aplicada a malhas poligonais passam não só pelos algoritmos mas também pelas estruturas de dados de suporte. Por outro lado, o desempenho dos algoritmos depende sempre da densidade de polígonos da malha, ou seja, quanto maior é a malha, maior é o tempo necessário à sua simplificação.

As malhas de grandes dimensões (ou seja, com grande número de células) são hoje em dia alvo de grande actividade de investigação e desenvolvimento, uma vez que se produzem malhas poligonais com resoluções cada vez mais elevadas, ou seja, com milhões de células. Estas malhas são também designadas por malhas gigantes.

Da necessidade de visualização de malhas de grande dimensões surgiu outra abordagem para a visualização de malhas baseada na geometria de pontos [91, 66], designada por *point-based rendering*. Esta nova abordagem é também conhecida por representação baseada em *surfels* (*surface elements*) [84, 115]. O que acontece na abordagem baseada na geometria de pontos é que, em vez da discretização clássica de uma superfície em triângulos ou polígonos, a discretização é feita com *surfels* que são pontos sem uma explícita conectividade entre si. Neste caso, a cada ponto associa-se, por exemplo, uma esfera para efeitos de visualização. Para mais informação sobre esta abordagem veja-se [130, 1, 83, 82].

Os problemas inerentes ao processamento de malhas gigantes fizeram também surgir outro tipo de algoritmos, chamados *out-of-core*. Estes algoritmos tratam a malha por zonas e não como um todo. De facto, em alguns casos, não é possível carregar toda a malha em simultâneo na memória devido a limitações físicas dos computadores em termos de armazenamento. Estes algoritmos baseiam-se na subdivisão do espaço envolvente [65, 67, 29] ou segmentação da malha [39, 86]. Os algoritmos *out-of-core* têm hoje grande aplicação em computação gráfica [98] como, por exemplo, na compressão [45] e na simplificação [65] de malhas gigantes. Mais informação sobre este tipo de algoritmos pode ser consultada em [98, 46].

2.2 Estruturas de Dados Geométricas

Diversas estruturas de dados podem ser usadas para definir e representar objectos geométricos em computação gráfica. Estas estruturas de dados podem ser classificadas em três categorias principais: estruturas baseadas na representação de fronteira

(B-rep) [5, 77, 124], na representação por decomposição celular (por exemplo, as *octrees* [78]), e na representação baseada em árvores CSG [88].

Na representação de malhas poligonais, são normalmente usadas estruturas de dados B-rep, mas também estruturas de dados específicas.

2.2.1 Estruturas de Dados B-rep

Numa estrutura de dados B-rep, apenas a superfície que define a fronteira do objecto é representada explicitamente. Ou seja, apenas a fronteira que separa o interior do exterior do objecto é armazenada. A fronteira pode ser definida por faces poligonais ou superfícies paramétricas [42]. No caso das malhas, a estrutura de dados B-rep armazena faces poligonais.

Há estruturas de dados B-rep orientadas e não-orientadas. Dois exemplos de estruturas de dados B-rep não-orientadas são os seguintes:

Estrutura de Dados Baseada em *Cell-Tuples* [11]. Um *cell-tuple* é uma sequência ordenada de células de diferentes dimensões que partilham relações de adjacência e incidência. Por exemplo, (v_1, e_1, f_1) é um *cell-tuple* onde o vértice v_1 é adjacente à aresta e_1, a qual por sua vez é adjacente à face f_1. Analogamente, pode dizer-se que f_1 é incidente em e_1 e e_1 é incidente em v_1. Muitas estruturas de dados B-rep não permitem desenvolver algoritmos rápidos e eficientes com malhas com grande número de células. Por exemplo, procurar uma célula que satisfaça uma relação topológica específica na estrutura de dados *cell-tuple* é uma tarefa lenta, em particular para malhas de grande dimensão, pois requer o processamento de *todos* os *cell-tuples*.

Estrutura de Dados Baseada na Lista de Triângulos. Esta estrutura de dados suporta apenas faces triangulares. Uma malha é representada por um conjunto de triângulos e pelo conjunto de vértices que compõem a malha. Neste caso é difícil desenvolver algoritmos eficientes para aceder à informação topológica. Por exemplo, para encontrar o conjunto de faces incidentes num determinado vértice é necessário percorrer *todas* as faces armazenadas na estrutura de dados, o que requer mais tempo do que seria de esperar. Esta estrutura de dados é vocacionada para visualização visto que o hardware gráfico está optimizado para o processamento de triângulos.

As faces são normalmente orientadas, isto é têm dois lados, o lado da frente e o lado de trás. Por convenção, o lado da frente da face é aquele que é visível num B-rep quando é visto do lado de fora do objecto. Normalmente usa-se a regra da mão direita para definir a orientação de uma face [27]. De forma a efectuar a visualização das faces estas devem estar correctamente orientadas, sendo a sua visualização baseada na normal a cada face. A normal a uma face pode ser calculada através do produto externo entre duas arestas consecutivas do contorno da face. A normal é um vector

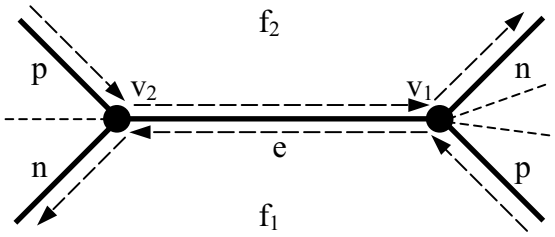


Figura 2.2: Representação *Winged-Edge*.

perpendicular à face que aponta para o lado de fora do objecto. Assim, a partir da orientação de uma face é possível saber de que lado se está a observar o objecto, ou seja, qual é o lado exterior e interior do objecto. Esta informação serve também para efectuar a iluminação do objecto.

Uma das maneiras de "acelerar"os algoritmos é usar estruturas de dados orientadas, apesar de isto requerer mais memória para armazenar as células orientadas. De alguma maneira as células orientadas são redundantes. As estruturas de dados B-rep orientadas mais utilizadas são aquelas baseadas nas representações *winged-edge* [5], *half-edge* [77] e *radial edge* [124].

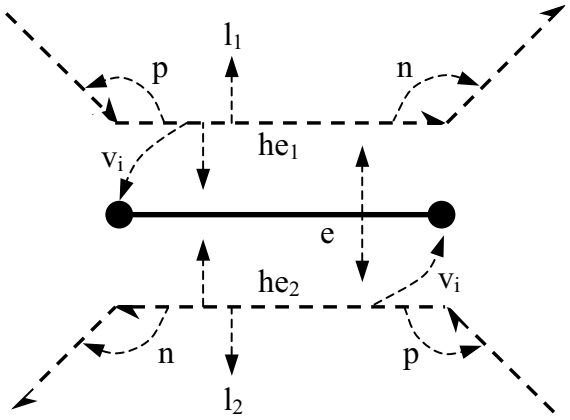


Figura 2.3: Representação *Half-Edge*.

Estrutura de Dados Baseada na Representação *Winged-Edge*. É uma estrutura de dados orientada como ilustra a Figura 2.2. Cada aresta e possui apontadores para os seus vértices adjacentes v_1 e v_2, apontadores para as faces f_1 e f_2 nela incidentes, e apontadores para as arestas seguinte n e anterior p na fronteira de cada uma das duas faces incidentes. Tem-se ainda uma tabela de vértices em que cada vértice tem um apontador para uma aresta que lhe é incidente, e ainda outra tabela de faces onde cada face tem um apontador para uma aresta que lhe é adjacente.

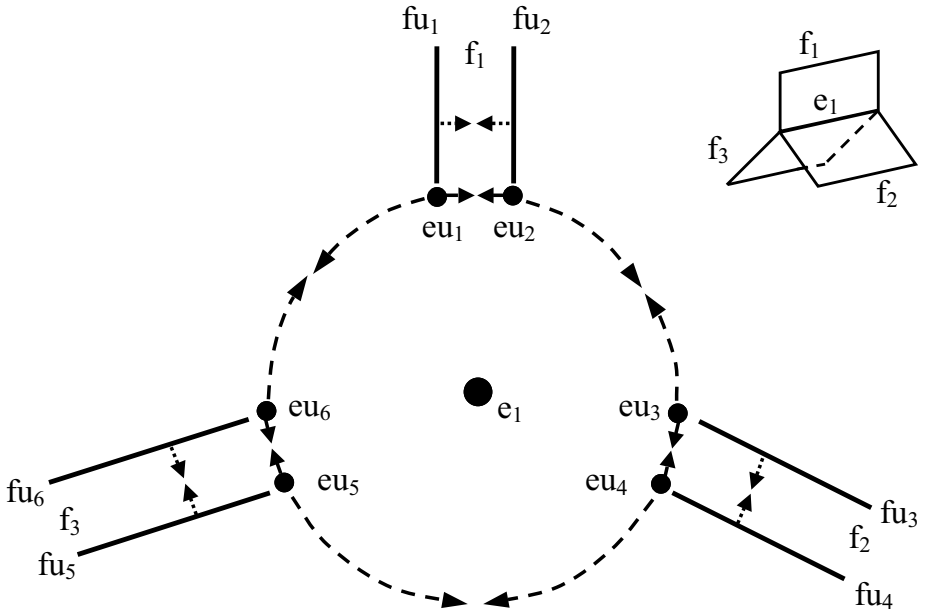


Figura 2.4: Representação *Radial Edge* de três faces incidentes numa aresta.

Estrutura de Dados Baseada na Representação *Half-Edge*. É uma variante da estrutura de dados *Winged-Edge* em que cada aresta é constituída por duas arestas orientadas em sentidos opostos (*half-edges*). Cada aresta orientada está associada a uma das faces incidentes na aresta. É também uma estrutura de dados orientada como se pode ver na Figura 2.3. Nesta estrutura de dados, cada *half-edge* contém um apontador para o seu vértice de origem, outro para a outra *half-edge* gémea que define a aresta, dois apontadores para as *half-edges* seguinte e anterior segundo a orientação estabelecida e, ainda, um apontador para o ciclo de *half-edges* em torno de uma face. Cada vértice tem um apontador para a *half-edge* da qual é a origem. Do mesmo modo, cada face tem um apontador para uma *half-edge* do ciclo de *half-edges* que a delimita.

Por exemplo, no caso da Figura 2.3, a *half-edge* he_1 aponta para o seu vértice de origem v_i, para a outra *half-edge* he_2 da aresta e, para as *half-edges* seguinte n e anterior p, e ainda para o ciclo a que pertence l_1.

Estrutura de Dados Baseada na Representação *Radial Edge*. Esta estrutura de dados é uma generalização da estrutura *Winged-Edge*, pois suporta a representação de objectos geométricos *non-manifold* [124]. É uma estrutura orientada onde cada face tem dois ciclos de arestas, um para cada lado da face, o que requer grandes recursos em termos de memória. Por exemplo, a Figura 2.4 mostra a representação *radial edge* para três faces f_1, f_2 e f_3 incidentes na aresta e_1. Como se depreende da Figura 2.4, as três faces possuem seis ciclos de arestas $(fu_1, fu_2, fu_3, fu_4, fu_5, fu_6)$, bem como seis arestas orientada $(eu_1, eu_2, eu_3, eu_4, eu_5, eu_6)$ para uma única aresta e_1.

Como se verá no Capítulo 3, a estrutura de dados AIF proposta nesta tese também

suporta objectos *non-manifold*, mas tem a vantagem de ser muito mais concisa e de permitir acessos mais rápidos à informação topológica do que a estrutura de dados *Radial Edge*.

Mais recentemente, Lee e Lee [62] propuseram uma nova estrutura de dados, designada por *Partial Entity Structure*, que resulta de um melhoramento da estrutura de dados *Radial Edge*. Esta estrutura de dados reduz o espaço de armazenamento para metade em relação à estrutura baseada na representação *Radial Edge*.

2.2.2 Estruturas de Dados para Malhas Poligonais

Com a crescente utilização das malhas poligonais em computação gráfica têm surgido nos últimos anos algumas estruturas de dados específicas para malhas poligonais, em particular para malhas triangulares.

Kallmann e Thalmann [49] desenvolveram uma estrutura de dados concisa para malhas poligonais, designada de *Star-Vertex*. Esta estrutura é baseada na informação sobre as incidências no vértice. Mas apesar da sua concisão, o acesso à informação topológica (adjacências e incidências) é uma tarefa lenta, pois nem sequer é mantida explicitamente informação sobre arestas e faces na estrutura de dados. Assim, é necessário inferir essas informações *a priori*.

Algumas estruturas de dados foram concebidas para representar apenas malhas simpliciais. Por exemplo, as estruturas de dados *Directed Edges* [14] e *Tri-Edges* [71] suportam apenas malhas triangulares, ou seja, malhas simpliciais de dimensão 2.

Mais genericamente, a estrutura de dados *Progressive Simplicial Complexes* (PSC) [85] suporta a representação de complexos simpliciais orientáveis (ou seja, não necessariamente orientados) de dimensão n. No entanto, a ausência de complexos orientados na estrutura PSC coloca algumas dificuldades à sua visualização. De facto, ao contrário das malhas progressivas proposta por Hoppe [38], a estrutura PSC evita o armazenamento das normais à superfície nos vértices. Em vez disso, faz uso de campos de suavização para diferentes materiais à semelhança do que permite as tecnologias *Wavefront*.

Floriani *et al.* [26] apresentaram uma nova estrutura de dados para malhas triangulares *non-manifold* que codifica explicitamente os vértices (V) e os triângulos (T). Esta estrutura de dados armazena as seguintes relações de adjacência e incidência $V \rightarrow V$, $V \rightarrow T$, $T \rightarrow V$ e $T \rightarrow T$. É uma estrutura concisa, mas o acesso às relações topológicas que envolvem as arestas é lento porque as arestas não estão explicitamente representadas na estrutura de dados.

Botsch *et al.* [8] apresentaram um novo núcleo geométrico baseado numa estrutura de dados *half-edge* optimizada a que chamaram OpenMesh. Este núcleo geométrico suporta, além das primitivas geométricas habituais, ainda algumas primitivas avançadas como malhas progressivas [38] e superfícies subdivididas [97, 121]. Este núcleo geomé-

trico encontra-se disponível em `http://www.openmesh.org` de onde pode ser descarregado para utilização gratuita nos termos da *GNU Library General Public License* (`http://www.gnu.org/`).

Mais recentemente, têm surgido mais algumas estruturas de dados multiresolução específicas para malhas poligonais. Mais detalhes sobre estas estruturas de dados podem ser encontrados em Cignoni *et al.* [18] e Floriani *et al.* [25].

Abordagens	Classificação
Onduletas	$Detalhado \rightarrow Grosseiro$
Algoritmos de Subdivisão	$Grosseiro \rightarrow Detalhado$
Algoritmos de Simplificação	$Detalhado \rightarrow Grosseiro$

Tabela 2.1: Diferentes abordagens da análise multiresolução em computação gráfica.

2.3 Análise Multiresolução

A análise multiresolução está inevitavelmente associada ao estudo das onduletas (*wavelets*) em matemática, ao processamento de imagem, bem como a outras áreas científicas. A quantidade de informação disponível em vários meios sobre onduletas é tão vasta que, por exemplo, quando se faz uma pesquisa web sobre livros com a palavra *wavelets* aparecem centenas deles, entre os quais se contam as referências [48, 13, 110].

Em computação gráfica, o principal interesse da utilização da análise multiresolução reside na possibilidade de ter diversos níveis de detalhe ou resolução num único modelo geométrico, ou seja, de ter uma representação hierárquica do modelo. Veja-se [20] para um resumo sobre os avanços da multiresolução na computação gráfica.

A Tabela 2.1 apresenta uma classificação das diferentes abordagens da análise multiresolução em computação gráfica em termos da representação hierárquica da granularidade dos modelos. Por exemplo, num algoritmo de subdivisão é gerado um modelo mais detalhado de um objecto geométrico do que o modelo original. Ao invés, no caso dos algoritmos baseados em onduletas e dos algoritmos de simplificação está associada uma granularidade mais grosseira ao modelo gerado do que a do objecto original. As diferentes abordagens são apresentadas nas secções seguintes dando especial ênfase à multiresolução aplicada a malhas poligonais, pois são estas que interessa apresentar no âmbito desta tese.

2.3.1 Onduletas

As onduletas são uma ferramenta matemática da análise multiresolução. As onduletas permitem decompor hierarquicamente uma função em duas partes: uma função

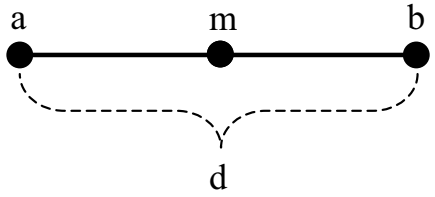


Figura 2.5: Exemplo de aplicação da onduleta de *Haar*.

aproximada mais simples (ou grosseira) e os coeficientes de detalhe necessários para recuperar a função original.

As onduletas são utilizadas em diversas áreas, sendo talvez as de processamento de sinal e processamento de imagem as mais divulgadas. Nestas áreas, as onduletas são utilizadas para decompor o sinal em duas gamas de frequências: baixas e altas. No processamento de imagem, este processo de decomposição é designado de filtragem, ou seja a aplicação de filtros baixos e altos [3].

Para uma melhor compreensão das onduletas apresenta-se de seguida a onduleta de *Haar* e um exemplo simples da sua aplicação à compressão de imagem.

Onduleta de *Haar*. Considerem-se dois pontos vizinhos a e b como ilustra a Figura 2.5. Os pontos a e b possuem uma correlação de proximidade, pelo que pode tirar-se partido disso. De facto, a partir de a e b pode calcular-se a sua média m e a sua diferença d. Tem-se então

$$\begin{aligned} m &= (a+b)/2 \\ d &= b-a \end{aligned}$$

Reescrevendo a e b em função de m e d, tem-se

$$\begin{aligned} a &= m - d/2 \\ b &= m + d/2 \end{aligned}$$

Logo, pode representar-se os pontos a e b através de uma aproximação m e com o detalhe d, pois a partir de m e d é possível recuperar os valores a e b.

Resolução	Imagem
4	[10 8 7 3]
2	[9 5 1 2]
1	[7 1 2 2]

Tabela 2.2: Onduleta de *Haar* aplicada a uma imagem.

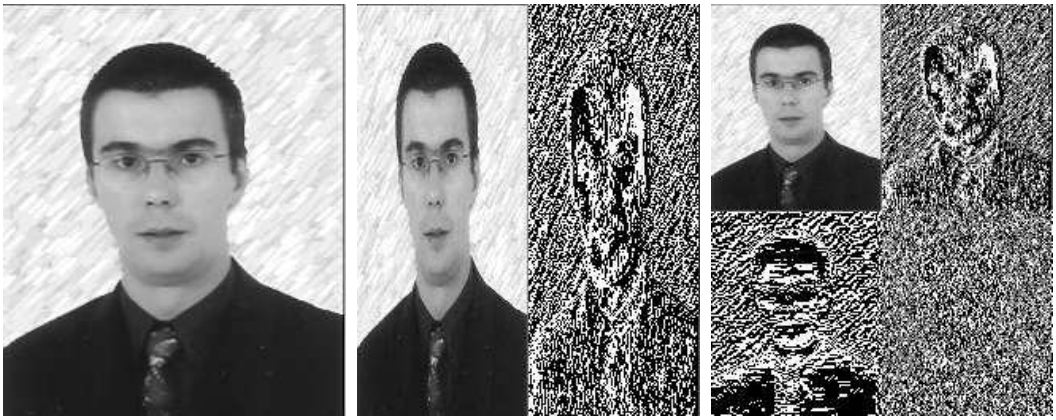
Onduleta de *Haar* na Compressão de Imagem. Veja-se então um exemplo de aplicação da onduleta de *Haar* na compressão de uma imagem. Considere-se uma imagem com uma resolução de quatro pixéis como se mostra na Tabela 2.2. Então uma imagem com resolução 2 pode ser criada a partir da imagem com resolução 4 [10 8 7 3] calculando a média e a diferença relativamente à média para cada par de pontos. Assim, tem-se os valores [9 5 1 2] para a resolução 2, onde os dois primeiros valores (9 e 5) correspondem às médias e os dois últimos (1 e 2) ao detalhe associado. Estes valores são calculados do seguinte modo

$$\begin{array}{lll} m = (10+8)/2 = 9 & \text{e} & d = 9-8 = 1 \\ m = (7+3)/2 = 5 & \text{e} & d = 5-3 = 2 \end{array}$$

Da mesma forma pode calcular-se a imagem para a resolução 1, cujos valores [7 1 2 2] são calculados de forma semelhante

$$m = (9+5)/2 = 7 \quad \text{e} \quad d = 7-5 = 2$$

Em resumo, esta imagem com quatro pixéis [10 8 7 3] pode ser representada por uma outra aproximada com dois pontos [9 5] e com o detalhe associado [1 2], e pode ainda ser representada por uma outra aproximação com apenas um ponto [7] e com o detalhe [1 2 2]. Como já foi referido o detalhe serve para poder recuperar a resolução original.

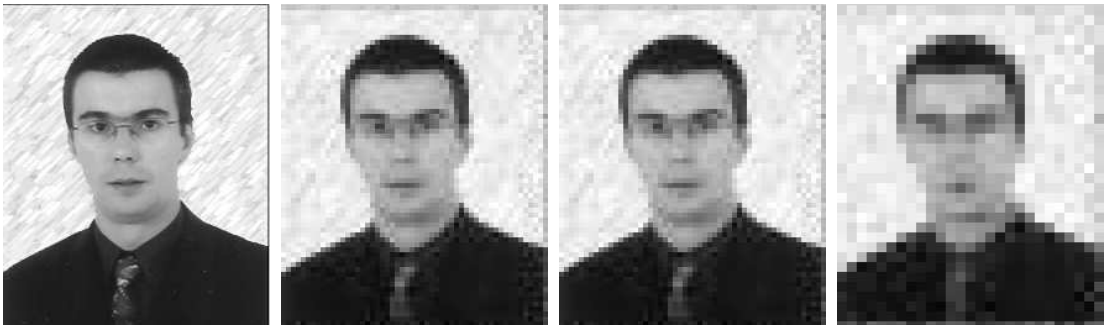


a) Imagem original b) 1ª compressão c) 2ª compressão

Figura 2.6: Aplicação da onduleta de *Haar* na compressão de uma imagem.

Por exemplo, na Figura 2.6 pode ver-se a aplicação da onduleta de *Haar* a uma imagem, onde na Figura 2.6(a) se tem a imagem original e na Figura 2.6(b) a imagem comprimida depois de aplicada a onduleta de *Haar* uma vez, segundo a direcção horizontal, e na Figura 2.6(c) a imagem comprimida depois de aplicada a onduleta de *Haar* pela segunda vez, agora segundo a direcção vertical. Assim, no lado esquerdo da Figura 2.6(b) podemos ver a imagem comprimida em 50% e que é normalmente o que se designa por onduleta. No lado direito da Figura 2.6(b) tem-se os coeficientes da onduleta que permitem recuperar a resolução anterior. Note-se que a criação de uma

nova imagem para outro nível de resolução faz-se por aplicação da onduleta segundo a horizontal e depois segundo a vertical, ou vice versa.



a) Original b) Nível 1 c) Nível 2 d) Nível 3

Figura 2.7: Resultado da compressão de uma imagem usando a onduleta de *Haar*.

A Figura 2.7 mostra o resultado da aplicação da onduleta de *Haar* na criação de três níveis de detalhe para a mesma imagem, onde se pode constatar a perda de resolução na passagem para um nível superior de compressão.

Onduletas em computação gráfica. Em computação gráfica, pode usar-se também a teoria das onduletas para simplificar malhas, através da aplicação de uma transformação **A** que cria uma malha grosseira e ao mesmo tempo aplicando outra transformação **B** para captar o detalhe que é perdido devido à primeira transformação (Figura 2.8). Mais informações sobre onduletas em computação gráfica podem ser encontradas em [28, 94, 108, 109, 117, 118].

2.3.2 Algoritmos de Subdivisão

Em computação gráfica, os algoritmos de subdivisão são usados para criar superfícies subdivididas ou malhas poligonais com diversos níveis de detalhe. A partir de um modelo grosseiro gera-se um modelo mais detalhado por subdivisão sucessiva das células da malha, como ilustra a Figura 2.9 para o caso de uma malha triangular. Na Figura 2.9 tem-se a malha inicial M_0 e dois passos do algoritmo de subdivisão que geram M_1 e M_2, respectivamente. Claro que a subdivisão obedece a regras precisas que permitem distinguir um esquema de subdivisão de outro.

As superfícies subdivididas têm importantes propriedades de modelação: suportam malhas com topologias arbitrárias; garantem uma certa continuidade da superfície (continuidade C^0); e permitem o desenvolvimento de algoritmos eficientes que são simples de codificar.

A subdivisão permite definir uma curva ou superfície como o limite de uma sequência de refinamentos (subdivisões) sucessivos de um modelo inicial grosseiro. Existem vários esquemas de subdivisão para superfícies (e também para curvas).

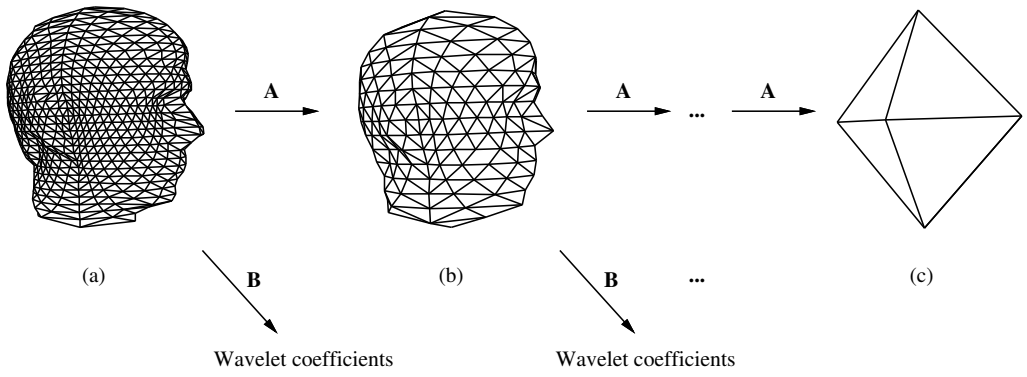


Figura 2.8: Decomposição de uma malha poligonal (retirada de [72]).

Esquemas de Subdivisão para Malhas Os algoritmos de subdivisão adicionam novas células a uma malha inicial de modo a produzir uma nova malha mais detalhada. A criação de novas células faz-se à custa da inserção de novos vértices por subdivisão de arestas ou por subdivisão de faces. Por exemplo, a Figura 2.10 ilustra alguns dos esquemas de subdivisão normalmente usados para faces triangulares e quadrangulares. Por exemplo, os esquemas representados na Figura 2.10 (a) e (d) referem-se à subdivisão de arestas, enquanto que os esquemas representados na Figura 2.10 (c) e (f) referem-se à subdivisão de faces por inserção de um vértice. Quanto aos esquemas representados na Figura 2.10 (b) e (e) são uma combinação dos esquemas de subdivisão de arestas e subdivisão de faces.

Os primeiros esquemas de subdivisão para malhas irregulares foram os esquemas de Catmull-Clark [15] e de Doo-Sabin [21]. Mais tarde surgiu o esquema de Loop [70] para malhas triangulares arbitrárias. Note-se que todos estes esquemas permitem criar diversas aproximações à malha. A diferença entre eles reside no cálculo dos novos vértices.

Dyn *et al.* [22] apresentaram um novo esquema designado de *butterfly* para interpolar

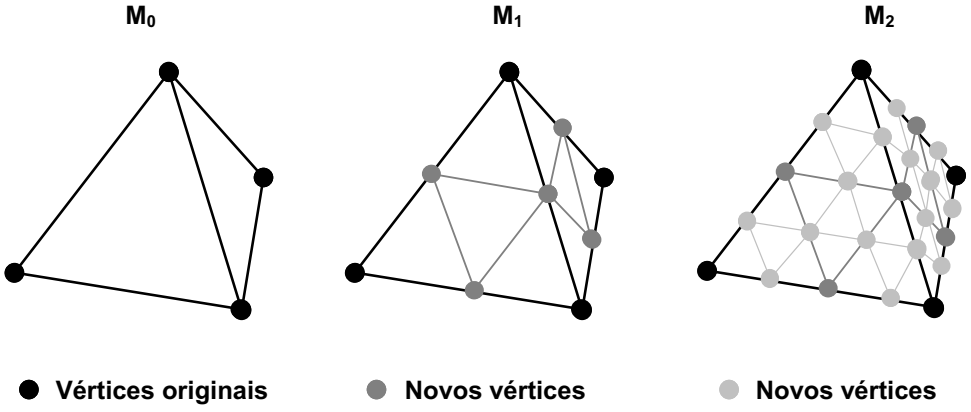


Figura 2.9: Subdivisão de uma malha poligonal - tetraedro.

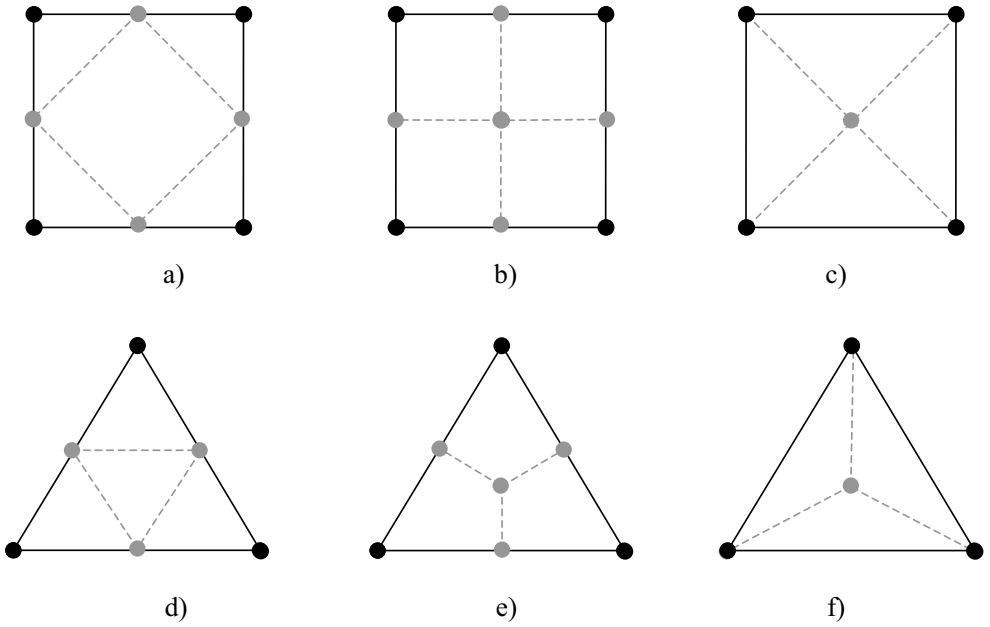


Figura 2.10: Três tipos de subdivisão para faces triangulares e quadrangulares.

superfícies subdivididas triangulares com continuidade C^1. Kobbelt [54] apresentou também um esquema com continuidade C^1 mas para malhas quadrangulares com topologia arbitrária. Esquemas alternativos têm surgido na literatura e estão descritos em [128, 121].

Nas Figuras 2.11 e 2.12(a) apresentam-se os coeficientes dos esquemas de subdivisão para malhas quadrangulares. A Figura 2.11 apresenta os coeficientes usados no cálculo da geometria dos novos vértices segundo o esquema Catmull-Clark, enquanto que a Figura 2.12(a) apresenta os coeficientes usados no cálculo da geometria dos novos vértices segundo o esquema Doo-Sabin.

Para malhas triangulares pode usar-se o esquema de Loop e o esquema *butterfly*.

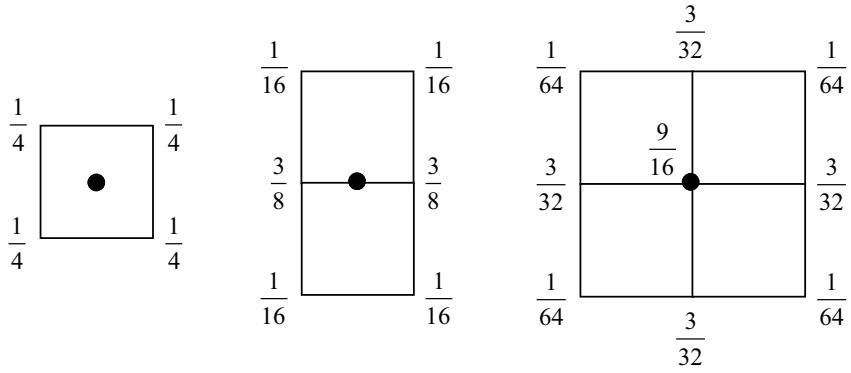
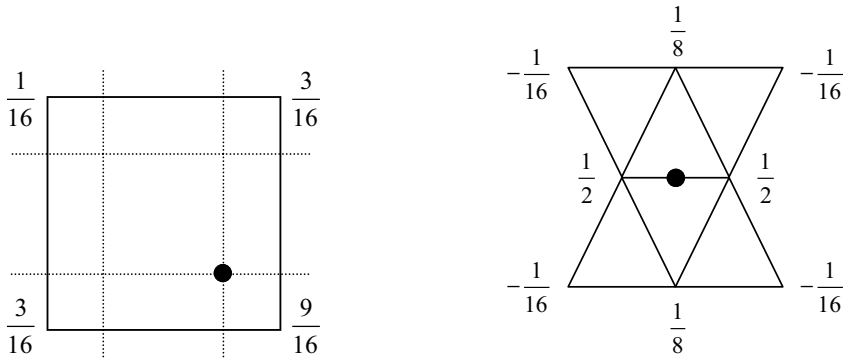


Figura 2.11: Coeficientes do esquema de Catmull-Clark.



a) Doo-Sabin b) *Butterfly*

Figura 2.12: Coeficientes dos esquemas de Doo-Sabin e *butterfly*.

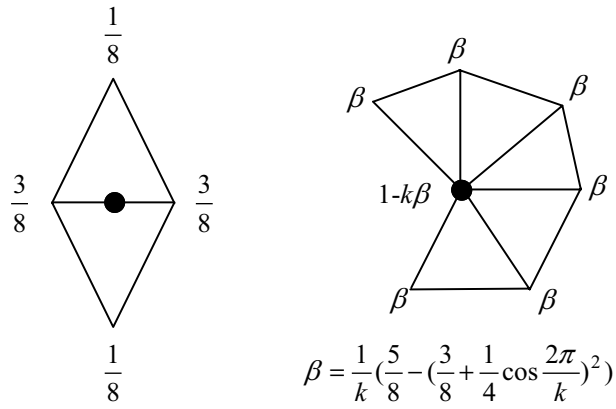
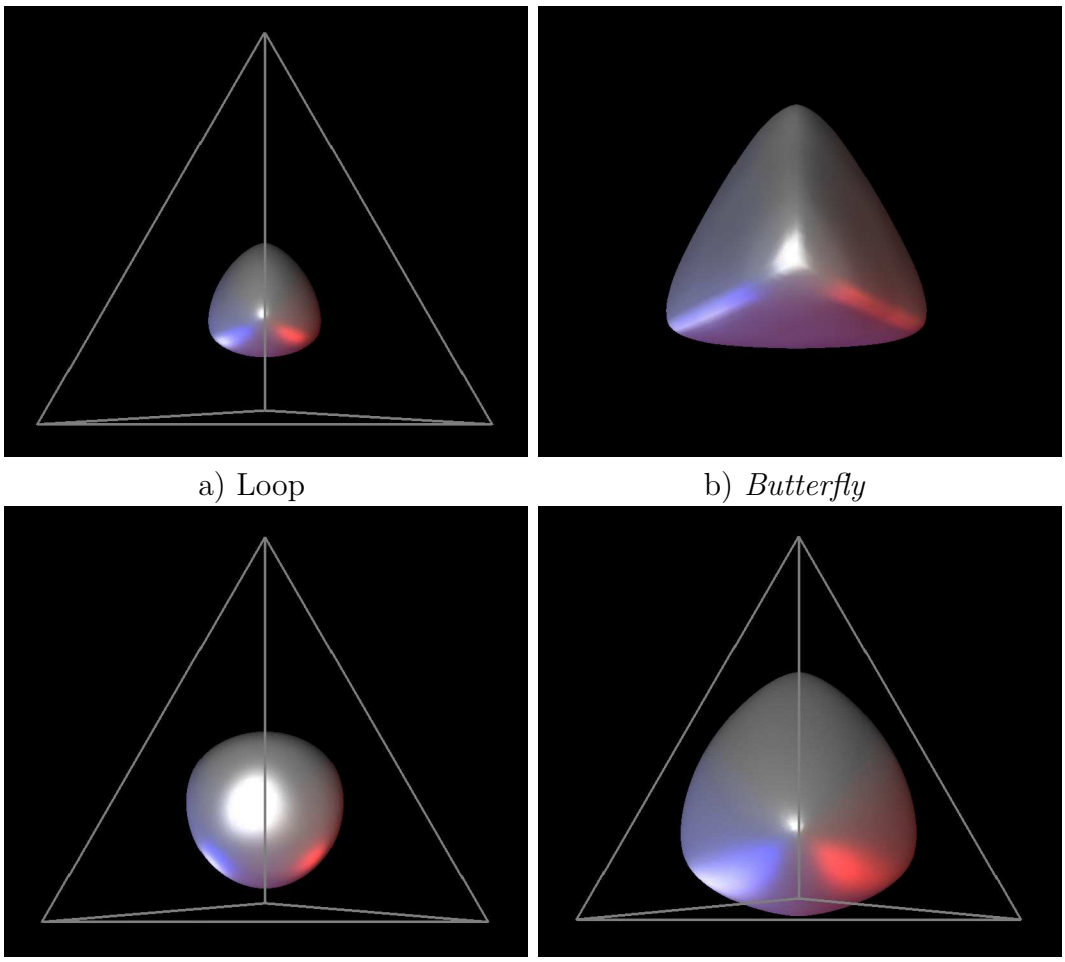


Figura 2.13: Coeficientes do esquema de Loop.

Estes esquemas são topologicamente semelhantes, mas geometricamente distintos. Em termos topológicos, estes dois esquemas usam a subdivisão de arestas (Figura 2.10(d)). A principal diferença entre os dois esquemas é a forma com calculam a geometria dos novos vértices. Assim, no caso do esquema *butterfly*, o cálculo de cada novo vértice é feito recorrendo aos coeficientes dos vértices indicados na Figura 2.12(b). No esquema de Loop, o cálculo de um novo vértice é feito de acordo com os coeficientes dos vértices indicados na Figura 2.13. Assim, como são geometricamente distintos, os resultados produzidos também são distintos.

A Figura 2.14 evidencia as diferenças práticas entre os diversos esquemas de subdivisão apresentados anteriormente. Como se pode constatar, o resultado de cada esquema de subdivisão faz-se notar na forma final do objecto, podendo este ser maior ou menor e mais ou menos suave.

Um resumo mais alargado dos esquemas de subdivisão e dos últimos desenvolvimentos na área pode ser consultado em [92].



a) Loop b) *Butterfly*

c) Catmull-Clark d) Doo-Sabin

Figura 2.14: Comparação dos esquemas de subdivisão para o caso de um tetraedro (Retiradas de [128]).

2.3.3 Algoritmos de Simplificação

Os algoritmos de simplificação são outra das formas usadas na criação de representações multiresolução, ou seja, representações com uma estrutura hierárquica de modelos com diversos níveis de detalhe. Neste caso parte-se do modelo detalhado para obter um modelo mais grosseiro após vários passos de simplificação.

Simplificar uma malha poligonal M_i consiste na geração de outra malha M_j com menor número de células, onde a malha M_j obedece a um critério que tem associado um erro máximo admissível. Em alguns casos, o erro é imposto implicitamente pelo número de células da malha M_j.

Os algoritmos de simplificação podem ser classificados em três categorias, consoante o tipo de operação usada para a simplificação da malha. Assim, temos as seguintes categorias:

- **Remoção de células** (*Cell decimation*).
  Basicamente, estes algoritmos seleccionam uma célula (vértice, aresta ou face) para remover, após o que removem também todas as suas células adjacentes, o que implica a reconstrução da malha na zona de onde foram removidas as células. Embora estes algoritmos reduzam o número de células da malha, a topologia do modelo é preservada. A sua principal vantagem é que os vértices que permanecem na malha pertencem à malha original, o que de alguma forma permite ter a informação da malha original. Este facto permite ainda a utilização dos atributos (cor, textura, etc.) dos vértices que se mantêm inalterados.
  Os algoritmos descritos em [95, 53, 19, 56] pertencem a esta categoria.

- **Confluência de vértices** (*Vertex clustering*).
  Os algoritmos desta classe utilizam um volume que envolve a malha, que depois é subdividido em pequenos sub-volumes. A subdivisão do volume envolvente pode ser do tipo *octree*. Na subdivisão *octree*, um volume é recursiva e hierarquicamente subdividido em oito sub-volumes.
  Os vértices que se encontrarem dentro de um mesmo sub-volume passam a ser representados apenas pelo vértice representativo (ou *cluster*), sendo então criada uma nova malha com base apenas nos vértices representativos. O vértice representativo pode ser o centro de massa do sub-volume ou outro ponto qualquer do sub-volume escolhido por um critério pré-definido.
  Normalmente, estes algoritmos produzem aproximações muito grosseiras do modelo porque não preservam nem a topologia nem a geometria do modelo, e a qualidade das malhas depende sempre da dimensão do sub-volume criado.
  Alguns dos algoritmos desta categoria são descritos em [89, 12, 65].

- **Contracção de arestas** (*Edge collapsing*).
  Estes algoritmos simplificam a malha iterativamente pela contracção de arestas em vértices. A escolha da aresta a contrair é uma das características intrínsecas de cada algoritmo que determina como as simplificações são geradas. Estes algoritmos mantêm a topologia do modelo e os vértices do modelo simplificado também podem pertencer ao modelo original.
  A operação de contracção de arestas tem como desvantagem poder causar localmente auto-intersecções ou inflexões na malha, pelo que se torna necessário identificar cuidadosamente estas situações.
  Os algoritmos descritos em [41, 38, 31, 68] pertencem a esta categoria.

O novo algoritmo de simplificação proposto nesta tese, e que é descrito no Capítulo 4, insere-se na categoria dos algoritmos baseados na contracção de arestas. Veja-se então os critérios de simplificação, ou seja, de escolha da aresta a contrair que permitem distinguir os diferentes algoritmos.

Critérios de Simplificação Utilizados na Contracção de Arestas. A operação de contracção de arestas é uma operação estandardizada. A principal diferença entre os

algoritmos que a usam é o critério de selecção da aresta a contrair. Diferentes critérios implicam diferentes tempos de processamento e malhas com diferentes qualidades.

Hoppe [38] introduziu um algoritmo para a construção de malhas progressivas onde o critério para a escolha da aresta a contrair é baseado na minimização de uma função de energia. Uma malha progressiva é basicamente uma malha multiresolução que pode ser simplificada e refinada através das operações de contracção de arestas e subdivisão de vértices, respectivamente. No Capítulo 4 é apresentado um esquema multiresolução também baseado nestas duas operações.

O algoritmo proposto por Garland e Heckbert [31] segue um critério geométrico para a escolha da aresta a contrair que é baseado na minimização do erro associado a cada vértice. Este erro é definido como uma soma de quadrados da distância a um conjunto de planos à volta da aresta a contrair, nomeadamente a cada um dos seus vértices adjacentes. Este algoritmo permite ainda a confluência de pares de vértices mesmo que não exista uma aresta entre ambos. Claro que nesta situação não é preservada a topologia da malha. No entanto, este algoritmo tem a vantagem de produzir excelentes aproximações rapidamente.

Lindstrom e Turk [68] apresentaram um algoritmo que se baseia na minimização de uma soma ponderada que inclui dois termos: um referente à preservação do volume da malha e o outro que representa uma optimização da fronteira da malha. De um modo geral, o algoritmo selecciona a posição do vértice para cada aresta contraída de modo a minimizar a variação do volume da malha. Além disso é feita também a minimização da variação das áreas junto da fronteira da malha, por minimização da variação das áreas dos triângulos.

Guskov *et al.* [36] introduziram as malhas normais. Uma malha normal é um malha multiresolução, em que cada malha da hierarquia é criada a partir de um deslocamento segundo a direcção normal em relação à malha anterior. Cada uma destas malhas da hierarquia é criada recorrendo ao algoritmo de simplificação apresentado por Garland e Heckbert.

Mais recentemente, Kho e Garland [51] apresentaram uma nova abordagem para a simplificação de malhas controlada pelo utilizador. O utilizador impõe restrições geométricas em áreas específicas de forma a manter as características da superfície naquela zona. Isto significa que a simplificação é apenas aplicada nas áreas onde não existem restrições. O algoritmo de simplificação usado nesta abordagem é mais uma vez o de Garland e Heckbert.

De um modo geral, todos os algoritmos de simplificação de malhas resultam de um compromisso entre o desempenho do algoritmo e a qualidade das malhas obtidas. Para mais detalhes sobre os algoritmos de simplificação de malhas veja-se as referências [16, 68, 87, 30, 44, 74, 73].

2.4 Edição de Malhas

2.4.1 Edição de Malhas Poligonais

A edição interactiva de malhas poligonais é um tópico emergente na computação gráfica. A edição e deformação interactivas de malhas é cada vez mais importante em diversas áreas, que vão desde a modelação geométrica à animação e aos ambientes virtuais. Ao contrário das superfícies paramétricas que são deformáveis por deslocação dos seus pontos de controlo, com as malhas pode obter-se um controlo mais intuitivo das deformações editando directamente os vértices ou aplicando transformações geométricas locais ou globais à malha.

Lee [63] apresentou um novo método para a edição interactiva de malhas poligonais arbitrárias. O utilizador especifica uma área rectangular a editar, a qual é depois mapeada num plano onde o utilizador pode editar os vértices da malha. Este método tem como restrição o facto de a área a editar ter de ser homeomorfa a um disco bidimensional. Este método não permite editar os vértices directamente. Ao utilizador apenas é dada a possibilidade de manipular imagens dos vértices na área mapeada de modo similar a pontos de controlo de uma superfície paramétrica.

Kanai *et al.* [50] apresentaram um novo esquema de modelação de malhas, designado por fusão de malhas. Este esquema é baseado na metamorfose ou combinação de duas partes de malhas diferentes numa única malha. É estabelecida uma correspondência inicial entre as duas malhas através do mapeamento de um conjunto de pontos da malha origem na malha destino. Depois é gerada uma transição suave por interpolação entre os pontos da malha original e os pontos da malha destino.

Suzuki *et al.* [112] propuseram um método para deslocar uma parte de uma malha triangular com alteração da sua posição e orientação. Este método é baseado numa redefinição adaptativa (*re-meshing*) da malha sempre que é detectada uma deformação exagerada das faces devido à deslocação de uma parte da malha. Nesses casos, faz-se uma redefinição adaptativa modificando adequadamente a malha, apagando ou subdividindo faces através de operações topológicas locais. No entanto, algumas direcções de deslocamento podem causar irregularidades na malha que são difíceis de corrigir. Além disso, como a redefinição adaptativa da malha é definida por interpolação simples podem perder-se características da malha ou introduzir ondulações indesejáveis na malha.

Lee e Lee [64] apresentaram um método baseado em contornos activos, a que chamaram *geometric snakes*, para identificar características de forma em malhas triangulares. Uma *geometric snake* é um contorno activo que se desloca suavemente da sua posição inicial especificada pelo utilizador para cima de uma característica da malha enquanto minimiza uma função de energia. Uma *geometric snake* é semelhante aos contornos activos usados no processamento de imagem [7].

Madi e Walton [76] apresentaram um novo método para a selecção e modificação

interactiva de malhas poligonais. Este método tira partido de uma nova estrutura de dados, chamada *Triangle Loop Data Structure* (TLDS), e de uma tabela de ponteiros para os vértices que permite o acesso rápido às células vizinhas de um dado vértice, tornando assim rápidas e eficientes as operações de edição de malhas.

Recentemente, Botsch e Kobbelt [10] apresentaram uma ferramenta para a modelação de malhas triangulares não estruturadas, a qual é baseada na optimização de restrições impostas à malha. Primeiro o utilizador especifica um conjunto de restrições sobre a fronteira da zona a editar, as quais permitem definir uma função de base que pode ser ajustada convenientemente para produzir os efeitos desejados na edição. Por exemplo, as condições de continuidade de C^0 até C^2. A edição da malha é depois efectuada recorrendo a uma ansa com seis graus de liberdade, havendo ainda a possibilidade de variar a escala dimensional (ou seja, nove graus de liberdade), sendo a deformação controlada pelas funções de base pré-definidas.

Sumner e Popovic [111] apresentaram ainda recentemente um novo método para copiar automaticamente a deformação de uma malha para outra malha. Este método não requer que as malhas tenham o mesmo número de células ou a mesma conectividade. No entanto, o método ajusta-se melhor a malhas que tenham algumas semelhanças pois a transferência de deformações entre malhas muito distintas é um problema ainda em aberto.

Sorkine *et al.* [107] apresentaram também recentemente uma representação Laplaciana para malhas. Com base nesta representação desenvolveram algumas operações de edição de malhas como, por exemplo, a edição de uma região de interesse baseada na manipulação de uma ansa, a transferência e combinação de detalhes entre duas malhas e o corte-e-costura (*cut-and-paste*) entre duas malhas.

2.4.2 Edição de Malhas Multiresolução

As técnicas de multiresolução por seu lado oferecem duas vantagens importantes na edição de malhas. Primeiro, os algoritmos que operam sobre representações multiresolução podem obter aumentos de desempenho significativos. Segundo, uma representação multiresolução permite separar o detalhe do modelo da sua forma de base. Estas duas vantagens permitem desenvolver novas metodologias de edição de malhas.

No contexto da multiresolução, Zorin *et al.* [129] propuseram a primeira ferramenta de edição para malhas semi-regulares. Esta ferramenta combina algoritmos de subdivisão e suavização para editar malhas multiresolução de uma forma interactiva.

Kobbelt *et al.* [57] generalizaram as técnicas anteriores de multiresolução para malhas triangulares arbitrárias. No entanto, a representação multiresolução é criada *a priori* porque o desempenho dos algoritmos não permite fazê-lo interactivamente. Além disso, introduziram uma hierarquia geométrica do modelo refinado para o modelo grosseiro, usando o algoritmo do chapéu de chuva (*umbrella algorithm*).

Kobbelt *et al.* [58] estenderam a hierarquia multiresolução a malhas triangulares não estruturadas, tendo apresentado um novo método baseado nas técnicas de iluminação de Phong para calcular os coeficientes de detalhe. Uma abordagem semelhante foi proposta por Guskov *et al.* [35], na qual são usadas técnicas básicas de processamento de sinal aplicadas às malhas triangulares com conectividade irregular.

Kobbelt *et al.* [55] apresentaram uma nova abordagem para a deformação de malhas multiresolução, na qual o detalhe é representado implicitamente pela diferença entre malhas independentes. Propuseram ainda uma representação dinâmica da malha que ajusta a conectividade durante a operação de deformação de modo a preservar a qualidade da malha. Esta operação causa distorções que podem ser reduzidas dinamicamente através da redefinição local da malha.

Biermann *et al.* [6] propuseram um conjunto de operações intuitivas de corte-e-costura (*cut-and-paste*) para superfícies multiresolução para edição interactiva. No entanto, esta abordagem é difícil de generalizar a costura (*paste*) de regiões que não sejam topologicamente equivalentes a um disco bidimensional.

Mais recentemente, Botsch e Kobbelt [9] apresentaram uma nova representação para modelos multiresolução usando elementos de volume entre as diversas resoluções como forma de codificar o detalhe. Mantendo estes elementos de volume localmente constantes durante as operações de edição é possível obter um comportamento mais adequado da superfície e dos seus pormenores.

Outros esquemas para a edição e deformação de malhas têm sido propostos na literatura, alguns dos quais usando representações multiresolução. Contudo, as operações de edição são normalmente baseadas na manipulação directa de vértices ou pontos de controlo [4, 43, 96], na deformação axial [60], na deformação com *lattices* [75], ou na deformação com *wires* [106].

No Capítulo 5 apresenta-se uma nova metodologia para a edição interactiva de malhas poligonais usando sub-malhas [102]. Esta metodologia é também utilizada com uma representação multiresolução para as malhas [104].

2.5 Sumário

Este capítulo faz um levantamento do estado da arte mais relevante no que respeita às malhas multiresolução. Neste levantamento teve-se em conta três linhas orientadoras fundamentais que nortearam o trabalho de investigação que conduziu à escrita desta tese, nomeadamente:

- Estruturas de dados para malhas poligonais. Apresentaram-se as estruturas de dados específicas para malhas poligonais e as tradicionais estruturas de dados B-rep. Com este enquadramento pretendeu-se situar o desenvolvimento da estrutura de dados AIF no âmbito desta tese. Esta estrutura de dados é descrita

no Capítulo 3.

- Análise multiresolução. Relativamente a este ponto foram apresentadas as três abordagens principais: as onduletas, os algoritmos de subdivisão e os algoritmos de simplificação de malhas. Deste modo, criou-se o enquadramento que levou ao desenvolvimento do algoritmo de simplificação descrito no Capítulo 4.

- A edição interactiva de malhas. Por fim fez-se o ponto da situação actual dos algoritmos de edição interactiva de malhas. Assim, criou-se o enquadramento do trabalho realizado sobre a edição interactiva de malhas descrito no Capítulo 5 da tese.

Capítulo 3

Estrutura de Dados AIF

Este capítulo descreve uma nova estrutura de dados, designada de AIF (*Adjacency and Incidence Framework*), que foi desenhada para representar malhas poligonais genéricas. As malhas não necessitam de ser *manifold* nem triangulares. A estrutura de dados AIF é topologicamente orientável, mas não orientada. Isto significa que a estrutura AIF não possui células orientadas. Consequentemente, é uma estrutura de dados mais concisa que as tradicionais estruturas B-rep e permite acessos mais rápidos à informação topológica (adjacências e incidências) do que a generalidade das estruturas de dados usadas para representar malhas poligonais.

3.1 Relações de Adjacência e Incidência

Uma malha poligonal é um conjunto de células (vértices, arestas e faces) relacionadas topologicamente entre si. Estas relações topológicas são conhecidas por relações de adjacência e incidência.

Diz-se que uma célula x é *adjacente* a uma célula y (simbolicamente $x \prec y$) se x está contida na fronteira de y (ou equivalentemente, $fr(y) \cap x \neq \emptyset$) e a dimensão de x é menor que a dimensão de y; equivalentemente, diz-se que y é *incidente* em x (simbolicamente, $y \succ x$). Por exemplo, diz-se que um vértice de uma aresta lhe é adjacente, mas podem existir várias arestas incidentes no mesmo vértice. A relação de adjacência (incidência) é transitiva, isto é, se $v \prec e$ e $e \prec f$ então $v \prec f$.

No que respeita a malhas poligonais, há duas relações básicas de adjacência, nomeadamente $V \prec E$ e $E \prec F$; as relações inversas $E \succ V$ e $F \succ E$ são as relações de incidência. Estas quatro relações podem ser combinadas para obter as nove relações de adjacência introduzidas por Weiler [123]. Por exemplo, as relações $V \rightarrow F$ e $E \rightarrow E$ podem ser obtidas do seguinte modo:

$$V \rightarrow F = (V \rightarrow E) \circ (E \rightarrow F) = (E \succ V) \circ (F \succ E)$$
$$E \rightarrow E = (E \rightarrow V) \circ (V \rightarrow E) = (V \prec E) \circ (E \succ V)$$

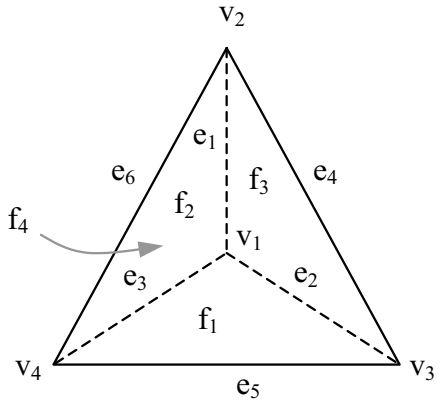


Figura 3.1: Malha *manifold*.

As quatro relações topológicas referidas no parágrafo anterior constituem uma representação na classe C_4^9 (veja-se Ni e Bloor [79] para mais detalhes). Como se verá na próxima secção, esta representação é a usada pela estrutura de dados AIF. É a melhor representação da classe C_4^9 em termos do número de acessos à informação topológica, pois requer um número mínimo de acessos directos e indirectos à estrutura de dados para aceder às quatro relações explícitas e às restantes cinco relações implícitas, respectivamente. Um acesso directo requer uma simples chamada ao operador de pesquisa de informação (que será descrito mais adiante), enquanto que um acesso indirecto requer duas ou mais chamadas ao operador de pesquisa.

O esquema de incidência de uma malha na estrutura de dados AIF [100, 101, 103] pode ser descrito em termos de um conjunto de tuplos $T = \{v_i, e_j, f_k\}$, onde f_k é uma face incidente numa aresta e_j (simbolicamente, $f_k \succ e_j$) e e_j é incidente num vértice v_i ($e_j \succ v_i$); analogamente, v_i é adjacente a e_j ($v_i \prec e_j$) e e_j é adjacente a f_k ($e_j \prec f_k$). Por exemplo, o tetraedro da Figura 3.1 possui o seguinte esquema de incidência:

$$(v_1, e_1, f_2)(v_2, e_1, f_2)(v_3, e_2, f_1)(v_4, e_3, f_1)$$
$$(v_1, e_1, f_3)(v_2, e_1, f_3)(v_3, e_2, f_3)(v_4, e_3, f_2)$$
$$(v_1, e_2, f_3)(v_2, e_6, f_2)(v_3, e_4, f_3)(v_4, e_5, f_1)$$
$$(v_1, e_2, f_1)(v_2, e_6, f_4)(v_3, e_4, f_4)(v_4, e_5, f_4)$$
$$(v_1, e_3, f_2)(v_2, e_4, f_3)(v_3, e_5, f_1)(v_4, e_6, f_2)$$
$$(v_1, e_3, f_1)(v_2, e_4, f_4)(v_3, e_5, f_4)(v_4, e_6, f_4)$$

O esquema de incidência de uma malha pode ser considerado ele próprio como uma estrutura de dados, a qual é conhecida como a estrutura de dados *cell-tuple* que foi introduzida por Brisson [11]. A estrutura de dados AIF tem o mesmo poder de descrição das adjacências e incidências que a estrutura de dados *cell-tuple*, mas é mais concisa e de acesso mais rápido à informação topológica, como se poderá ver pelos resultados experimentais apresentados mais adiante. Isto deve-se ao facto da estrutura de dados AIF consistir num conjunto de células (e não num conjunto de tuplos), onde cada célula está representada explicitamente com uma ou duas relações topológicas do seguinte modo:

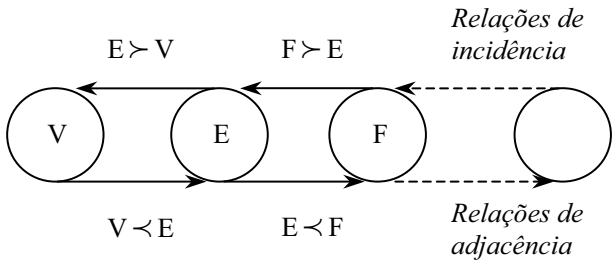


Figura 3.2: Diagrama da estrutura de dados AIF.

- Um vértice é definido em termos das suas aresta incidentes, ou seja, a relação $E \succ V$;

- Uma aresta é definida em termos dos seus vértices adjacentes e das suas faces incidentes, ou seja, as relações $V \prec E$ e $F \succ E$;

- Uma face é definida em termos das suas arestas adjacentes, ou seja, a relação $E \prec F$.

3.2 Estrutura de Dados AIF

Uma malha bidimensional na estrutura de dados AIF é definida em termos de um conjunto $M = \{V, E, F\}$, onde V é um conjunto finito de vértices, E é um conjunto finito de arestas, e F é um conjunto finito de faces não necessariamente simplesmente conexas.

Um *vértice* $v \in V$ é definido por um conjunto de arestas e_i $(i = 1...k)$ incidentes em v, isto é, $v = \{e_1, e_2, e_3, ..., e_k\}$. Esta é a forma como a relação $E \succ V$ está representada na estrutura de dados AIF.

Uma *aresta* $e \in E$ é definida por um par de conjuntos, o primeiro para os vértices adjacentes à aresta, e o segundo para as faces incidentes na aresta, isto é, $e = \{\{v_1, v_2\}, \{f_1, f_2, ..., f_k\}\}$, onde v_1 e v_2 são os vértices adjacentes à aresta, e $f_1, f_2, ..., f_k$ são as faces incidentes na aresta. Assim, temos também as relações $V \prec E$ e $F \succ E$ representadas na estrutura de dados AIF.

Uma *face* $f \in F$ é definida em termos do conjunto de arestas e_i $(i = 1...k)$ que lhe são adjacentes, ou seja, $f = \{e_1, e_2, e_3, ..., e_k\}$. Com esta definição temos também a relação $E \prec F$ representada na estrutura de dados AIF.

Com estas quatro relações básicas de adjacência e incidência explicitamente representadas na estrutura de dados AIF podemos representar malhas poligonais genéricas, independente de serem *manifold* ou *non-manifold*.

A estrutura de dados AIF acomoda pois uma representação óptima na classe C_4^9 para

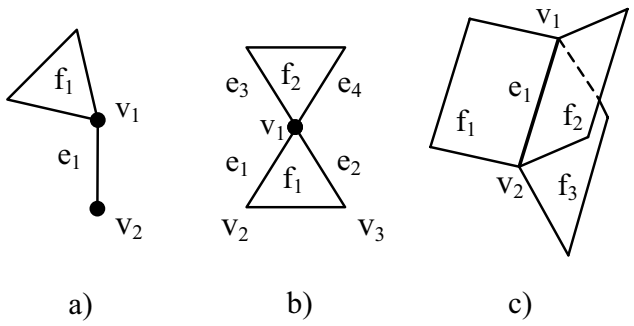


Figura 3.3: Objectos *non-manifold*.

malhas bidimensionais, mas pode ser facilmente generalizada para $C_{2n}^{(n+1)^2}$ quando se pretende representar objectos de dimensão n (Figura 3.2). A estrutura de dados AIF mantém a informação explícita de $2n$ relações de adjacência e incidência entre células, mas permite inferir informação topológica adicional de modo a atravessar qualquer malha duma forma eficiente.

3.2.1 Malhas *Non-manifold*

A estrutura de dados AIF suporta malhas *manifold* e *non-manifold*, bem como permite distinguir entre objectos *manifold* e *non-manifold*.

A Figura 3.3 apresenta três situações ou casos fundamentais de *non-manifoldness* em malhas bidimensionais. No caso (a) há uma aresta e_1 sem faces incidentes, isto é, uma aresta remanescente. A sua representação na estrutura de dados AIF é a seguinte:

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

Por outro lado, todas as arestas do objecto (b) têm uma única face incidente nelas. Assim, por exemplo, a representação da aresta e_1 do objecto (b) é a seguinte:

$$e_1 = \{\{v_1, v_2\}, \{f_1\}\}$$

A aresta e_1 do objecto (c) já tem três faces incidentes, f_1, f_2, f_3, e a sua representação é feita através dos seus vértices adjacentes e das faces incidentes, ou seja:

$$e_1 = \{\{v_1, v_2\}, \{f_1, f_2, f_3\}\}$$

Assim, a estrutura de dados AIF suporta malhas *non-manifold* porque é baseada na informação sobre adjacências e incidências entre as células (vértices, arestas e faces). Como consequência, pode distinguir-se entre um objecto *manifold* e um *non-manifold* inquirindo a estrutura de dados através do operador de pesquisa que é apresentado na Secção 3.5. Em particular, pode identificar-se as seguintes situações:

- *Um vértice isolado.* Um vértice isolado não tem arestas incidentes.

- *Uma aresta remanescente.* Uma aresta remanescente não tem faces incidentes.

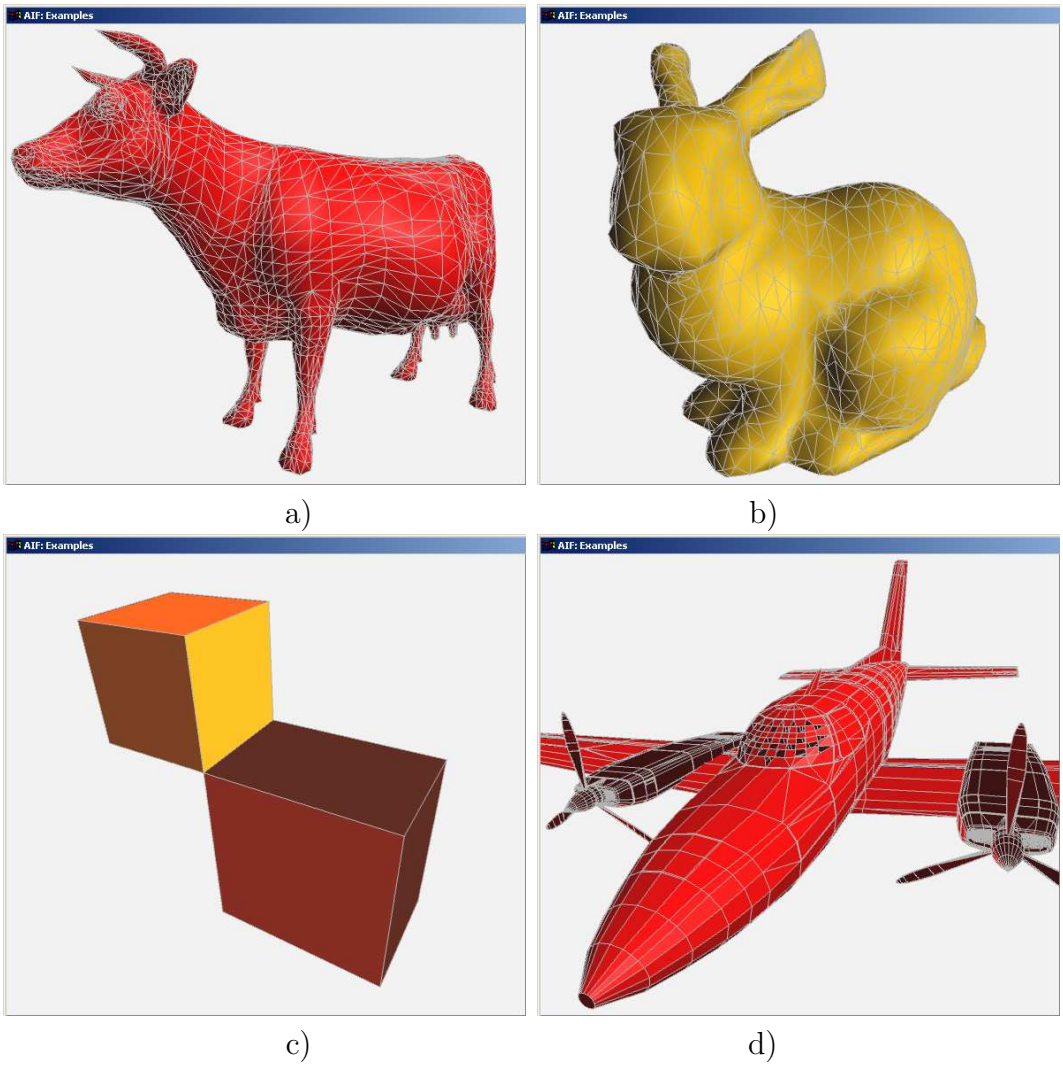


Figura 3.4: Malhas AIF, *manifold* e *non-manifold*.

- *Uma aresta de fronteira.* Uma aresta de fronteira tem uma única face incidente.
- *Um vértice de corte.* Um vértice de corte tem pelo menos $4 + 2n$ $(n = 0, ..., k)$ arestas incidentes. Um vértice de corte divide uma malha em duas ou mais malhas disjuntas; por exemplo, o vértice v_1 na figure 3.3(b) é um vértice de corte.
- *Um furo numa face.* Um furo é identificado pela existência de mais do que um ciclo de arestas numa dada face, o que pode ser feito por inferência do operador de pesquisa já referido.

Na Figura 3.4(d) o modelo do avião é *non-manifold* porque faltam algumas faces na zona do cockpit (zonas mais escuras). Note-se que uma aresta pode ter um número qualquer de faces incidentes como é o caso do objecto *non-manifold* representado na Figura 3.4(c), no qual existe uma aresta com quatro faces incidentes.

A estrutura de dados AIF suporta também malhas poligonais, quer sejam triangulares (Figura 3.4(a)(b)) ou não (Figura 3.4(c)(d)).

3.3 AIF - Codificação

A estrutura de dados AIF representa malhas bidimensionais constituídas por três conjuntos de células: vértices, arestas e faces. Cada tipo de célula foi codificado através de uma classe em C++, mas também se poderia codificar todos os tipos de células com uma única classe, como é necessário para malhas de dimensão n. Isto é possível porque qualquer célula de dimensão n pode ser representada como um conjunto de células incidentes de dimensão $(n+1)$ e um conjunto de células adjacentes de dimensão $(n-1)$.

A estrutura de dados AIF foi então codificada do seguinte modo:

```
class Point {
  double x,y,z;     // coordenadas dos pontos
}

class Vertex {
  int id;           // identificador do vértice
  evector li;       // vector de arestas incidentes no vértice
  Point *pt;        // geometria do vértice
}

class Edge {
  int id;           // identificador da aresta
  Vertex *v1, *v2;  // vértices adjacentes à aresta
  fvector li;       // vector de faces incidentes na aresta
}

class Face {
  int id;           // identificador da face
  evector la;       // vector de arestas adjacentes à face
  Point *nf;        // normal à face
}

class Mesh {
  int id;           // identificador da malha
  vvector vv;       // vector de vértices da malha
  evector ev;       // vector de arestas da malha
  fvector fv;       // vector de faces da malha
}
```

No protótipo desenvolvido todas as listas foram codificadas através de vectores dinâmicos. Como já foi referido anteriormente, a estrutura de dados AIF não é *topologicamente* orientada, pois não possui células orientadas. Ela é *geometricamente* orientada pela normal `nf` à face na classe `Face`. As normais às faces são determinadas por indução topológica duma orientação consistente nas faces, ou seja, todas as faces orientadas no sentido dos ponteiros do relógio ou no sentido contrário. Para isso desenvolveu-se um mecanismo para induzir a orientação na malha, como se descreve de seguida.

3.4 Mecanismo de Orientação

Para a visualização gráfica de uma malha é necessário que a estrutura de dados possua dados sobre a orientação, ou então se possa inferi-los de alguma forma. O que acontece é que a estrutura de dados AIF não é *orientada* do ponto de vista topológico, pois não possui células orientadas. No entanto, a estrutura de dados AIF é uma estrutura de dados *orientável*. Isto significa que uma orientação pode ser induzida topologicamente na malha de uma forma consistente para todas as faces.

O mecanismo de orientação desenvolvido começa por orientar uma qualquer face da malha com auxílio do operador de pesquisa descrito na secção seguinte. Isto é, selecciona-se uma aresta adjacente à face e percorre-se a sua fronteira num dos sentidos possíveis. Depois de terminar o processo de orientação de uma face, a normal da face é calculada e armazenada na estrutura de dados na classe `Face` no campo `nf`. Se no final se verificar que o sentido escolhido foi o sentido errado, então basta inverter a direcção de todas as normais calculadas. De seguida determinam-se as faces vizinhas ($F \rightarrow F$) da face inicialmente orientada, e para cada uma delas induz-se a mesma orientação. Aplicando este processo de orientação a todas as faces da malha obtém-se uma orientação consistente para toda a malha.

As faces vizinhas da face orientada são determinadas pelo operador de pesquisa de informação topológica. Isto requer a travessia da fronteira da face por aplicação alternada do operador $\blacktriangleright\!\lhd_0$ e $\blacktriangleright\!\lhd_1$ às células (arestas e vértices) que a delimitam. O operador $\blacktriangleright\!\lhd_0$ devolve o próximo vértice da fronteira da face orientada, e o operador $\blacktriangleright\!\lhd_1$ permite saber qual a próxima aresta adjacente à face. Isto significa que se tem de usar o operador de pesquisa tantas vezes quantas o número de células que delimitam a face, ou seja, que lhe são adjacentes.

No caso de malhas *non-manifold*, algumas das faces podem ficar por orientar depois de terminar o processo de orientação descrito anteriormente. Por exemplo, na Figura 3.4(d) faltam algumas faces na zona do cockpit o que impossibilita que algumas das faces venham a ser orientadas, pois para isso são necessárias as faces vizinhas. Nestes casos, tem de se aplicar o processo de orientação às faces que ficaram por orientar. Mas esta tarefa tem de ser efectuada com algum cuidado de modo a manter a consistência da orientação e permitir uma correcta visualização da malha.

O processo de orientação de malhas foi optimizado. Para isso sempre que se carrega a malha em memória é efectuada logo a sua orientação pelo mecanismo descrito anteriormente. Neste processo, e por forma a minimizar o número de chamadas ao mecanismo de orientação e assim aumentar o desempenho da estrutura de dados AIF, a lista de arestas de cada uma das faces é ordenada por forma a manter a orientação da malha consistente. Este facto permite posteriormente redefinir a normal a qualquer face sem ter de recorrer sistematicamente ao mecanismo de orientação de toda a malha, ou seja, passamos a ter a estrutura de dados AIF orientada aumentando o seu desempenho.

Esta estratégia foi desenvolvida com sucesso para malhas triangulares mas poderá ser também aplicada a malhas não triangulares. Assim, quando existem alterações geométricas e topológicas locais na malha torna-se mais eficiente calcular novas normais às faces.

3.5 Operador Topológico de Pesquisa

O tempo é um factor crítico nas aplicações interactivas (por exemplo na animação e nos jogos) que fazem uso de malhas poligonais. São necessários algoritmos de pesquisa rápidos para identificar localmente as células adjacentes ou incidentes noutras células de modo a rapidamente processar as mais variadas operações geométricas sobre malhas.

A estrutura de dados AIF usa um único operador de pesquisa, chamado operador máscara, para aceder a *toda* a informação sobre adjacências e incidências. O operador máscara é definido da seguinte forma:

- $\bowtie\!\triangleleft_d : V \times E \times F \rightarrow C$, com $C = V \cup E \cup F$, sendo V o conjunto dos vértices, E o conjunto de arestas, e F o conjunto de faces, tal que $\bowtie\!\triangleleft_d(v_i, e_j, f_k) = \{c_l^d\}$, tem como resultado um conjunto de células de dimensão d.

Os argumentos do operador máscara são células do conjunto $V \times E \times F$. Estes argumentos estabelecem relações de adjacência e incidência entre eles. A célula $NULL$ como argumento de dimensão $n = d$ significa que todas as células de dimensão n que satisfaçam as condições de adjacência/incidência expressas pelos argumentos são devolvidas como resultado. Caso contrário, se $n \neq d$ e a célula de dimensão n é também $NULL$, então nenhuma restrição de adjacência/incidência é imposta ao resultado. No caso de $n = d$ e a célula de dimensão n é diferente de $NULL$, o operador máscara devolve todas as células de dimensão n como anteriormente, à excepção da célula argumento de dimensão n. Se $n \neq d$ e a célula de dimensão n é diferente de $NULL$, então a célula de dimensão n coloca uma restrição adicional de adjacência/incidência ao resultado.

A chamada do operador máscara é feita pelo menos com um argumento não nulo, podendo ter dois ou mesmo os três argumentos não nulos. Qualquer combinação de

argumentos é possível desde que haja pelo menos um não nulo. No entanto, no caso da chamada do operador de índice zero com apenas o primeiro argumento diferente de $NULL$, significa que este devolverá os vértices vizinhos do vértice argumento. Assim, se pretendermos saber os vértices do vértice, as arestas da aresta ou as faces da face o operador devolve-nos os vértices, arestas ou faces vizinhas do argumento respectivamente.

Considere-se novamente a malha do tetraedro representado na Figura 3.1 para ilustrar o funcionamento do operador máscara em conjunção com a estrutura de dados AIF. Assim tem-se:

1. $\blacktriangleright\!\triangleleft_1(v_1,\text{NULL},\text{NULL}) = \{e_1, e_2, e_3\}$ devolve directamente todas as arestas incidentes em v_1.

2. $\blacktriangleright\!\triangleleft_2(v_1,\text{NULL},\text{NULL}) = \{f_1, f_2, f_3\}$ devolve indirectamente todas as faces incidentes em v_1. Requer a chamada intermédia do operador $\blacktriangleright\!\triangleleft_1(v_1,\text{NULL},\text{NULL})$ para devolver todas as arestas e_1, e_2, e_3 incidentes em v_1. Depois o operador máscara $\blacktriangleright\!\triangleleft_2(\text{NULL}, e_i,\text{NULL})$ é chamado para cada aresta e_i $(i = 1, 2, 3)$ de modo a calcular as faces incidentes em e_i e v_1.

3. $\blacktriangleright\!\triangleleft_0(\text{NULL}, e_1,\text{NULL}) = \{v_1, v_2\}$ devolve directamente os vértices da aresta e_1.

4. $\blacktriangleright\!\triangleleft_2(\text{NULL}, e_1,\text{NULL}) = \{f_2, f_3\}$ devolve directamente as faces incidentes na aresta e_1.

5. $\blacktriangleright\!\triangleleft_0(\text{NULL},\text{NULL}, f_1) = \{v_1, v_3, v_4\}$ devolve indirectamente todos os vértices da face f_1. Requer uma chamada intermédia ao operador $\blacktriangleright\!\triangleleft_1(\text{NULL},\text{NULL}, f_1)$ para primeiro determinar as arestas e_i adjacentes a f_1. Depois o operador máscara $\blacktriangleright\!\triangleleft_0(\text{NULL}, e_i,\text{NULL})$ é chamado para cada aresta e_i de modo a calcular os vértices adjacentes à aresta e_i e à face f_1.

6. $\blacktriangleright\!\triangleleft_1(\text{NULL},\text{NULL}, f_1) = \{e_2, e_3, e_5\}$ devolve directamente todas as arestas adjacentes à face f_1.

7. $\blacktriangleright\!\triangleleft_2(v_3, e_1,\text{NULL}) = \{f_3\}$ devolve directamente as faces incidentes em e_1 e v_3.

8. $\blacktriangleright\!\triangleleft_1(v_1, e_2, f_1) = \{e_3\}$ devolve directamente todas as arestas adjacentes à face f_1 e incidentes em v_1, e diferentes de e_2.

9. $\blacktriangleright\!\triangleleft_0(v_4,\text{NULL}, f_1) = \{v_1, v_3\}$ devolve indirectamente todos os vértices adjacentes à face f_1 diferentes de v_4. Requer a chamada intermédia do operador máscara $\blacktriangleright\!\triangleleft_1(\text{NULL},\text{NULL}, f_1)$ para devolver primeiro todas as arestas e_i adjacentes a f_1. Depois o operador máscara $\blacktriangleright\!\triangleleft_0(v_4, e_i,\text{NULL})$ é chamado para cada aresta e_i de modo a calcular os vértices adjacentes a e_i e f_1 diferentes de v_4.

10. $\blacktriangleright\!\triangleleft_0(v_1,\text{NULL},\text{NULL}) = \{v_2, v_4, v_3\}$ devolve indirectamente todos os vértices adjacentes a v_1. Requer a chamada intermédia do operador $\blacktriangleright\!\triangleleft_1(v_1,\text{NULL},\text{NULL})$ para devolver primeiro todas as arestas e_i incidentes em v_1. Depois o operador

máscara $\blacktriangleright\!\!\triangleleft_0(v_1, e_i, \text{NULL})$ é chamado para cada aresta e_i de modo a calcular os vértices adjacentes a e_i e diferentes de v_1.

Note-se que o operador máscara $\blacktriangleright\!\!\triangleleft_d$ não necessita de manipular todos as células da estrutura de dados AIF. O operador máscara manipula a malha localmente usando as relações de adjacência e incidência explicitamente armazenadas na estrutura de dados AIF. Assim, o seu desempenho é independente da quantidade de células da malha. Este aspecto é muito importante na manipulação de malhas com grande número de células, em particular em operações interactivas.

3.6 Operador de Pesquisa - Codificação

O operador máscara é usado para aceder à informação sobre adjacências e incidências entre as células armazenadas na estrutura de dados AIF. Recorde-se que as relações $V \prec E$, $E \prec F$, $E \succ V$ e $F \succ E$ estão armazenadas explicitamente na estrutura de dados AIF; $V \prec E$ está representada pelos vértices `v1` e `v2` na classe `Edge`, $E \prec F$ está representada pela lista `la` na classe `Face`, $E \succ V$ está representada pela lista `li` na classe `Vertex`, e $F \succ E$ está representada pela lista `li` na classe `Edge`.

As restantes cinco relações topológicas são inferidas percorrendo a estrutura de dados como se descreve de seguida. As relações $V \prec F$ e $F \succ V$ são inferidas pelo operador máscara através dos seguintes algoritmos:

```
MaskOperator(2,v,NULL,NULL) //relação V->F
//Parâmetros: índice, vértice, aresta, face
Begin
  For each e incident at v do
     For each f incident on e do
        Add f to result
   return set of faces
End

MaskOperator(0,NULL,NULL,f) //relação F->V
//Parâmetros: índice, vértice, aresta, face
Begin
  For each e adjacent to f do
     For each v adjacent to e do
        Add v to result
   return set of vertices
End
```

As restantes três relações de vizinhança entre células da mesma dimensão, $V \rightarrow V$, $E \rightarrow E$ e $F \rightarrow F$ são inferidas pelo operador máscara através dos seguintes algoritmos:

```
MaskOperator(0,v,NULL,NULL) //Relação V->V
//Parâmetros: índice, vértice, aresta, face
Begin
  For each e incident at v do
     For each vi adjacent to e do
        If (vi <> v)
           Add vi to result
   return set of vertices
End

MaskOperator(1,NULL,e,NULL) //Relação E->E
//Parâmetros: índice, vértice, aresta, face
Begin
  For each v adjacent to e do
     For each ei incident at v do
        If (ei <> e)
           Add ei to result
   return set of edges
End

MAskOperator(2,NULL,NULL,f) //relação F->F
//Parâmetros: índice, vértice, aresta, face
Begin
  For each e adjacent to f do
     For each fi incident on e do
        If (fi <> f)
           Add fi to result
   return set of faces
End
```

3.7 Comparações

Esta secção apresenta resultados comparativos entre a estrutura de dados AIF e outras estruturas de dados, quer estas estruturas sejam específicas para a representação de malhas poligonais, quer sejam as tradicionais estruturas B-rep.

3.7.1 Espaço em Memória

A Tabela 3.1 explicita as diferenças entre as diversas estruturas de dados em termos da memória necessária ao armazenamento de uma malha triangular com n vértices. A coluna 'Tipo de malha' identifica o tipo de malha suportada pela estrutura de dados, ou seja, triangular ($\triangle$) ou poligonal ($\square$), enquanto a coluna '*Non-manifold*' indica se a

Estruturas de dados	Tipo de Malha	*Non-manifold*	Bytes/△
Triangle List	△	yes	18
Star-Vertex	□	yes	10+4k
Progressive meshes	△	no	33
Tri-edge	△	no	35
AIF	□	yes	29+2k
PSC	△	yes	37+2k
Directed-edges	△	yes	44
Half-edge	□	no	46
FastMesh	△	no	53
Radial Edge	□	yes	56
Winged-edge	□	no	60

Tabela 3.1: Comparação do espaço em memória para diferentes estruturas de dados.

estrutura de dados suporta malhas *non-manifold* ou não. A última coluna 'Bytes/△' apresenta o número de bytes necessários para o armazenamento de uma face triangular na estrutura de dados. A variável k designa o grau do vértice, ou seja, o número de arestas incidentes no vértice.

A estrutura de dados *Triangle List* (lista de triângulos) é uma estrutura concisa pois apenas armazena os vértices e as faces. É usada principalmente para visualização visto que as placas gráficas estão optimizadas para a visualização de polígonos, em particular triângulos.

A estrutura de dados *Star-Vertex* apenas armazena vértices. Cada vértice armazena também os seus vértices vizinhos. A cada vértice vizinho está associado um índice que permite efectuar a navegação à volta do vértice segundo uma ordem pré-definida. É uma estrutura de dados também concisa que apenas necessita de $10 + 4k$ bytes por triângulo. Além disso, esta estrutura de dados suporta malhas genéricas, mas os acessos à informação sobre adjacências/incidências são lentos, pois não inclui arestas e faces explicitamente. A ausência de arestas e faces implica que seis relações de adjacência/incidência não estejam definidas (veja-se na coluna 'Não definidas' da Tabela 3.2).

As estruturas de dados *Progressive Meshes* [38] e *Tri-edges* [71] são também bastante concisas, visto que necessitam apenas de 33 e 35 bytes por triângulo, respectivamente. No entanto, estas estruturas de dados apenas suportam malhas triangulares *manifold*.

Por outro lado, uma malha triangular com n vértices na estrutura de dados AIF requer aproximadamente 35, 37, 39 ou 41 bytes por triângulo para $k = 3, 4, 5$ e 6, respectivamente. De facto, de acordo com a fórmula de Euler para uma malha bidimensional triangular, uma malha com n vértices tem cerca de m faces ($m = 2n$) e a arestas ($2m = 3a$). Assim, assumindo que um número real e um ponteiro ocupam 4 bytes cada, a memória necessária para uma malha triangular na estrutura de dados

AIF é:

- $(3 \times 4 + 4 \times k)n = (12 + 4k)n = (6 + 2k)m$ bytes para os vértices - coordenadas dos vértices e ponteiros para as arestas incidentes;
- $(2 \times 4 + 2 \times 4)a = 16a \approx 11m$ bytes para as arestas - ponteiros para os vértices adjacentes e ponteiros para as faces incidentes;
- $(3 \times 4)m = 12m$ bytes para as faces - ponteiros para as arestas adjacentes.

Note-se que a estrutura de dados AIF não inclui células orientadas (por exemplo, *half-edges*). Por isso é mais concisa que as estruturas B-rep tradicionais (*Winged-edge* [5]) e as suas variantes orientadas (*Half-edge* [77], *Directed-edges* [14], *FastMesh* [80], e a *Radial Edge* [124]). Por fim, a estrutura de dados *PSC* [85] suporta apenas malhas triangulares e requer mais memória do que a estrutura de dados AIF.

Resumindo, algumas das estruturas de dados analisadas ou necessitam de mais memória do que a estrutura de dados AIF, ou então suportam apenas malhas triangulares *manifold*. Pelo contrário, a estrutura de dados AIF é mais genérica e concisa.

3.7.2 Acesso à Informação Topológica

A Tabela 3.2 classifica as diversas estruturas de dados em termos do acesso à informação sobre adjacências e incidências para uma malha triangular. O acesso a esta informação está relacionado com o número de relações que estão explícita e implicitamente armazenadas na estrutura de dados. Isto é, quantos acessos são necessários para devolver *toda* a informação sobre adjacências e incidências da estrutura de dados. Tendo em conta que para malhas bidimensionais há três entidades topológicas (vértice, aresta e face), pode dizer-se que existem nove relações de adjacência e incidência [123].

Na Tabela 3.2 a coluna 'Classe' indica o número de relações de adjacência e incidência explicitamente representadas na estrutura de dados. Por exemplo, uma estrutura de dados C_4^9 tem quatro relações explicitamente representadas em nove possíveis. Assim, temos acesso directo a quatro relações e às células a elas associadas sem nenhum processamento adicional. As restantes cinco relações estão representadas implicitamente, visto que todas as estruturas de dados são completas, ou seja, representam malhas sem ambiguidade. Isto significa que temos de ter mecanismos de inferência para aceder a estas cinco relações de adjacência e incidência. A estes mecanismos de inferência que requerem mais do que um acesso para obter a informação topológica chamamos acessos indirectos.

Acontece que os acessos indirectos exigem um processamento adicional porque não só algumas relações topológicas não estão explicitamente representadas, mas porque algumas células topológicas não existem nas estruturas de dados, como se indica na coluna 'Não definidas' da Tabela 3.2. Por exemplo, a estrutura de dados *Star-Vertex* não

Estruturas de Dados	Não-definidas	Classe
AIF	0	C_4^9
PSC	0	C_4^9
Winged-edge	0	C_2^9
Radial Edge	0	C_2^9
Half-edge	0	C_2^9
FastMesh	3	C_2^9
Directed-edges	3	C_2^9
Triangle List	6	C_2^9
Star-Vertex	6	C_2^9
Progressive meshes	6	C_2^9
Tri-edge	6	C_2^9

Tabela 3.2: Classificação de estruturas de dados segundo as relações topológicas.

possui arestas e faces representadas explicitamente; como consequência, seis relações de adjacência/incidência não estão definidas (Tabela 3.2).

Observando outra vez a Tabela 3.2, pode ver-se que as estruturas de dados AIF e *PSC* são as mais rápidas em termos do número de acessos. Claro, que a rapidez depende do número de relações de adjacência e incidência explicitamente armazenadas na estrutura de dados, mas também é verdade que quantas mais relações se armazenam mais memória é necessária. Ambas as estruturas de dados AIF e *PSC* pertencem à classe C_4^9. De acordo com Ni e Bloor [79], pode dizer-se que ambas as estruturas de dados são óptimas nesta classe em termos do acesso à informação topológica, pois requerem um número mínimo de acessos para aceder a toda a informação sobre adjacências e incidências. Contudo, a estrutura de dados *PSC* apenas suporta malhas triangulares e necessita de mais memória do que a estrutura de dados AIF (ver Tabela 3.1).

As outras estruturas de dados da Tabela 3.2 pertencem à classe C_2^9 (dois acessos directos e sete acessos indirectos para obter toda a informação topológica). Por exemplo, a estrutura de dados *FastMesh* não possui a entidade vértice. Assim, as três relações de adjacência e incidência envolvendo vértices não estão definidas, podendo no entanto ser inferidas com um custo adicional em termos de tempo. As restantes seis relações envolvendo arestas e faces já podem ser obtidas mais facilmente. Contudo apenas duas estão explicitamente representadas na estrutura de dados, o que corresponde apenas a dois acessos directos, sendo os restantes acessos indirectos.

3.7.3 Desempenho da Estrutura AIF

O desempenho da estrutura de dados AIF foi avaliado num PC equipado com um processador Pentium 4, com 768MB de memória, e uma placa gráfica GeForce 4 com 64MB, e com o sistema operativo Windows 2000. A Tabela 3.3 mostra os resultados

			Tempo de Criação		Tempo de Visualização	
Malhas	#V	#F	Leitura	Orientação	GL_P	GL_L
Vaca	2904	5804	0.140	0.060	0.010	0.010
Avião	2745	3946	0.180	0.040	0.010	0.001
Coelho	34835	69473	2.474	0.982	0.170	0.030
Cavalo	48485	96966	2.554	1.292	0.210	0.035
Venus	50002	100000	3.425	1.382	0.240	0.040
Dragão	437645	871414	21.080	24.536	2.033	0.330

Tabela 3.3: Tempos de criação e visualização (em segundos).

para diferentes modelos armazenados na estrutura de dados AIF e representados na Figura 3.5.

A Tabela 3.3 congrega informação sobre os tempos de criação e visualização de uma malha. Assim, a criação de uma malha na estrutura de dados AIF a partir de um ficheiro de texto, por exemplo no formato *.SMF (*Shape Mesh File*), inclui:

1. Leitura do ficheiro do disco (coluna 'Leitura' da Tabela 3.3);
2. Orientar a malha usando o mecanismo de orientação (coluna 'Orientação' da Tabela 3.3).

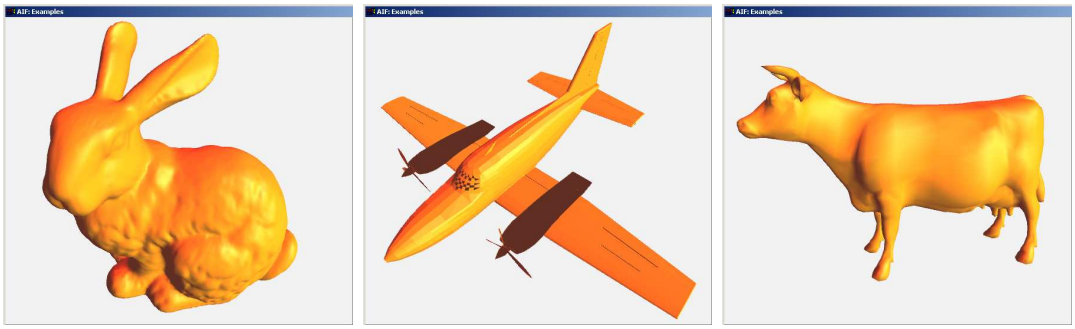
A visualização de uma malha pode ser feita de duas formas:

1. Criar a primitiva GL e visualizá-la (coluna 'GL_P' da Tabela 3.3);
2. Criar uma lista GL com a primitiva e visualizá-la (coluna 'GL_L' da Tabela 3.3).

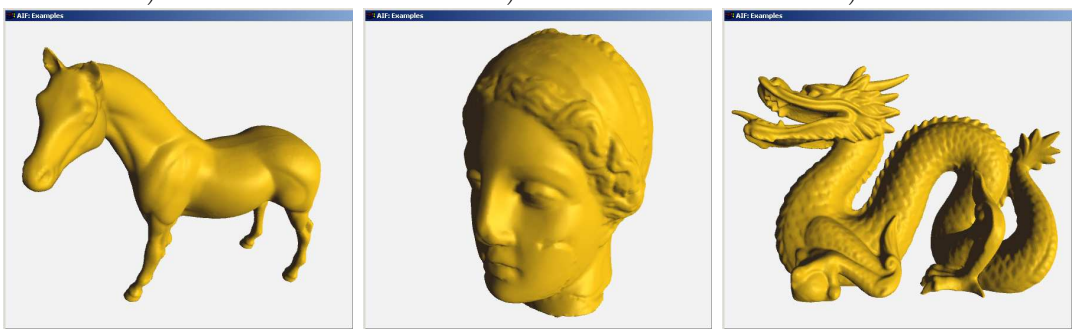
Por observação da Tabela 3.3 pode constatar-se que a visualização de malhas *manifold* (por exemplo, as malhas do coelho, do cavalo e de Venus da Figura 3.5), e de, malhas poligonais *non-manifold* (por exemplo, a malha do avião da Figura 3.5) é rápida mesmo quando não se usam listas GL (coluna 'GL_P' da Tabela 3.3). A utilização de listas GL reduz significativamente o tempo de visualização (coluna 'GL_L' da Tabela 3.3), mas com um custo adicional no tempo de criação das listas GL. Contudo, para malhas com grande número de células, como é o caso da malha do dragão (Figura 3.5(f)), os tempos aumentam proporcionalmente ao número de células da malha.

No que diz respeito à visualização, a estrutura de dados AIF suporta diferentes tipos de primitivas gráficas como, por exemplo:

- triângulos (GL_TRIANGLES - Figura 3.6(a))
- fitas de triângulos (GL_TRIANGLE_STRIP - Figura 3.6(b))



a) Coelho. b) Avião. c) Vaca.



d) Cavalo. e) Venus. f) Dragão.

Figura 3.5: Diversas malhas na estrutura de dados AIF.

- triângulos incidentes num vértice (GL_TRIANGLE_FAN - Figura 3.6(c))
- polígonos (GL_POLYGON - Figura 3.6(d))

Por exemplo, a malha da vaca na Figura 3.6(c) é mais suave porque se usou as normais nos vértices, em vez das normais nas faces como na Figura 3.6(a).

De notar que no caso da Figura 3.6(b) não se usou nenhum algoritmo optimizado para a representação das fitas de triângulos. No entanto, existem vários algoritmos para esse efeito, dos quais se destacam o algoritmo FTSG [125] que é referenciado na literatura como sendo o mais rápido. Contudo recentemente surgiu também o algoritmo STRIP [119] que é também rápido e eficiente.

3.8 Sumário

Neste capítulo apresentou-se uma nova estrutura de dados para malhas poligonais, designada por AIF. Foram também apresentados alguns testes comparativos, em termos de memória e rapidez no acesso à informação topológica, entre a estrutura de dados AIF e outras estruturas de dados: as estruturas de dados específicas para malhas e as estruturas B-rep tradicionais.

Constatou-se que a estrutura de dados AIF é mais "rápida"no acesso à informação

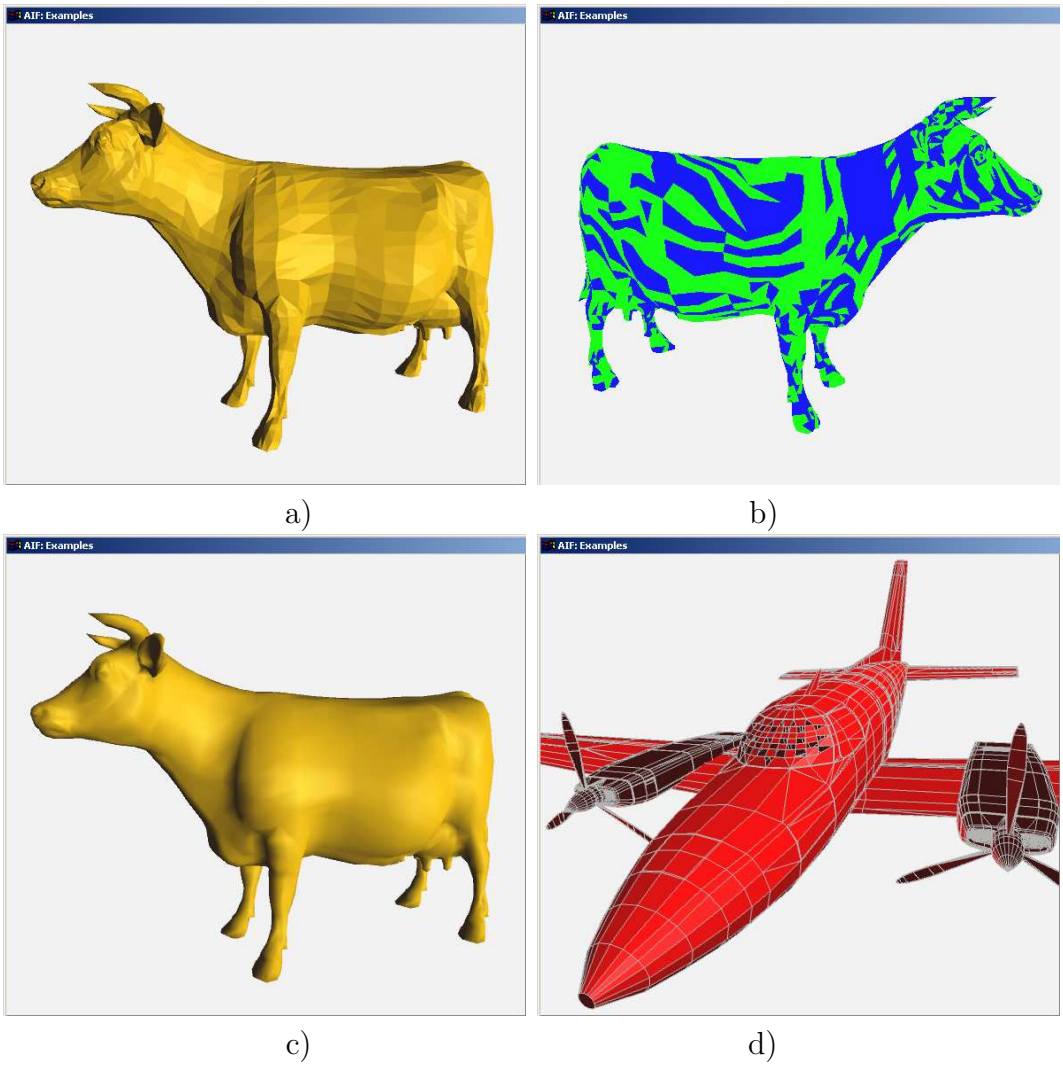


Figura 3.6: Visualização com diferentes tipos de primitivas gráficas.

topológica do que a generalidade das estruturas de dados. Além disso, é uma estrutura de dados genérica que suporta malhas poligonais, *manifold* e *non-manifold*, não necessariamente triangulares.

Note-se que a estrutura de dados AIF, apesar de ser uma estrutura de dados para representação de malhas poligonais, não deixa de ser também uma estrutura B-rep, pois permite representar objectos geométricos pela sua fronteira, não sendo necessário que as células sejam simplesmente conexas, embora este último aspecto não tenha sido completamente explorado no âmbito desta tese. No entanto, a estrutura de dados é claramente mais concisa do que as estruturas de dados B-rep tradicionais porque não possui células orientadas.

Outra contribuição de especial relevo é a utilização de um único operador —operador máscara— para a extracção de informação topológica a partir da estrutura de dados.

Capítulo 4

Algoritmos de Simplificação de Malhas

As malhas poligonais utilizadas em vários domínios da computação gráfica têm cada vez maior número de células, o que torna muitas vezes difícil a sua manipulação. Os algoritmos de simplificação de malhas resolvem de algum modo este problema pois permitem substituir uma malha com grande densidade de polígonos por uma outra menos densa sem que o olho humano note a diferença.

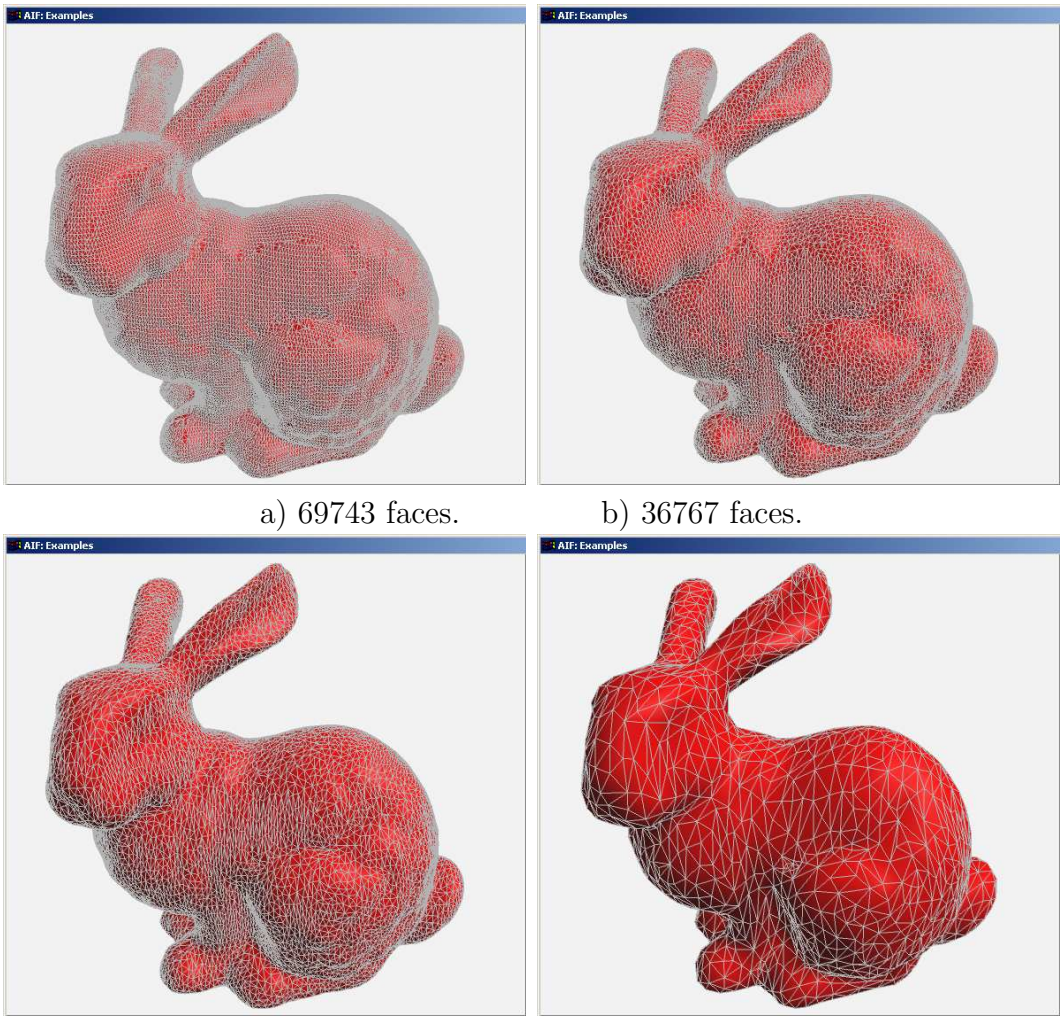
Este facto é particularmente importante não só para modelar cenas ou parte delas quando a distância ao observador aumenta, mas também para efeitos de armazenamento e transmissão de modelos geométricos pela Internet. Para tanto, basta ter presente os jogos de computador em rede.

O objectivo deste capítulo é, pois, apresentar um novo algoritmo para a simplificação de malhas poligonais, designado de NSA (*Normal-based Simplification Algorithm*), bem como o seu antecessor BSP (*Bi-Star Planarity algorithm*).

4.1 Simplificação de Malhas

Os modelos gerados pelos sistemas de aquisição de dados 3D, também designados por *scanners*, são cada vez mais usados na computação gráfica. A resolução destes *scanners* 3D tem aumentado bastante, o que faz com que a quantidade de informação por modelo seja cada vez maior. Actualmente, há mesmo *scanners* capazes de produzir malhas com resoluções muito para além da capacidade de discernimento do olho humano.

O problema é que se é necessária maior resolução para melhor definir os detalhes do modelo, já nas zonas com menor complexidade de forma (zonas mais planas) essa resolução é desnecessária. No entanto, os *scanners* 3D geram modelos com o mesmo número de pontos quer nas zonas com maior variação de forma, quer nas zonas com menor variação de forma, criando assim muita informação desnecessária.



a) 69743 faces. b) 36767 faces.

c) 19967 faces. d) 4099 faces.

Figura 4.1: Modelo com diferentes níveis de detalhe.

A manipulação eficiente de modelos com grande número de células em aplicações gráficas interactivas iniciou-se com o uso de modelos com diferentes níveis de detalhe discretos (LOD) como, por exemplo, acontece no formato VRML [120]. Um modelo com diferentes níveis de detalhe permite ter diferentes representações de um mesmo objecto geométrico com diferentes resoluções, como mostra a Figura 4.1 para uma malha de um coelho.

Um modelo LOD consiste numa sequência fixa de malhas independentes com diferente número de células, cada uma das quais representa o objecto com uma resolução diferente. Torna-se, assim, necessário armazenar separada e simultaneamente em memória todas as malhas de cada objecto. Infelizmente, dadas as limitações de memória dos computadores actuais, o número de malhas armazenadas para cada objecto é pequeno. Este facto faz com que possam existir variações abruptas entre níveis de detalhe consecutivos.

Algoritmos de Simplificação	Tempo de execução para um modelo com cerca de 70 000 faces
Hoppe [38]	51 minutos
Guéziec[34]	8 minutos
Lindstrom and Turk [68]	2.5 minutos
Garland and Heckbert [31]	15 segundos
Bi-Star Planarity [99]	9 segundos

Tabela 4.1: Tempos de execução de diferentes algoritmos.

A criação de diversas representações de um objecto (ou seja, de um modelo LOD) é muitas vezes efectuada recorrendo a algoritmos de simplificação de malhas a partir de uma malha inicial.

Os algoritmos de simplificação de malhas permitem aproximar uma malha por outra com menos informação, de tal modo que a diferença entre a malha original e a malha simplificada não é visualmente perceptível. Portanto, a simplificação de uma malha M_i consiste em gerar outra malha M_j com menor número de células (vértices, arestas e faces). A malha resultante obedece a um critério que normalmente é baseado num erro máximo admissível. O erro pode ser ainda definido implicitamente através da especificação do número de células pretendidas para a malha simplificada.

4.2 Algoritmo BSP

Existem vários algoritmos para a simplificação de malhas que partem de uma malha fina (malha original) e geram uma malha simplificada. Como se viu no Capítulo 2, há três estratégias principais para simplificar uma malha: (i) por remoção iterativa de células (vértices, arestas ou faces), (ii) por confluência de vértices ou ainda (iii) por contracção de arestas.

O algoritmo BSP (*Bi-Star Planarity algorithm*) usa a operação de contracção de arestas para a simplificação da malha com base na avaliação da complanaridade das faces em redor da aresta a contrair. A principal diferença entre os algoritmos que usam esta operação é o critério de escolha da aresta a contrair. O critério tem influência quer no desempenho do algoritmo, quer na qualidade das malhas geradas.

A Tabela 4.1 apresenta os tempos de execução de vários algoritmos de simplificação que usam também a operação de contracção de arestas, incluindo o algoritmo BSP. O algoritmo proposto por Hoppe [38] baseia-se na minimização de uma função de energia. O algoritmo proposto por Guéziec [34] baseia-se na manutenção do volume da malha dentro de uma dada tolerância. O algoritmo proposto por Garland e Heckbert [31] usa um critério geométrico baseado na estimação da distância a um conjunto de planos. Por último, o algoritmo de Lindstrom e Turk [68] baseia-se no uso de restrições como

a conservação do volume e da área. Mas, contrariamente aos outros algoritmos, o algoritmo de Lindstrom e Turk faz a simplificação de uma malha a partir da malha corrente, ou seja, não guarda nem usa informação sobre a malha original para tomar decisões.

O algoritmo BSP baseia-se na complanaridade das faces à volta dos vértices da aresta a contrair e, também, usa apenas informação da malha corrente. O critério de escolha da aresta revelou-se adequado pois permitiu desenvolver um algoritmo de simplificação rápido. É um critério puramente geométrico, sendo a aresta a contrair escolhida da seguinte forma:

1. Selecciona-se uma aresta e;
2. Se pelo menos um dos vértices da aresta e foi anteriormente criado por uma operação de contracção no mesmo passo da simplificação, então volta-se ao passo 1 e escolhe-se outra aresta;
3. Se as faces incidentes em e não são aproximadamente complanares a menos de um erro ϵ, então volta-se ao passo 1 e escolhe-se outra aresta;
4. Para cada vértice v da aresta e, se as faces incidentes em v não forem aproximadamente complanares a menos de um erro ϵ, então volta-se ao passo 1 e escolhe-se outra aresta;
5. É feita a contracção da aresta e num vértice, que é o ponto médio da aresta e, após o que se remove as faces que lhe são incidentes.

A operação de contracção de uma aresta é descrita na Figura 4.2. A contracção da aresta e começa pela coalescência das faces f_{11} e f_{12} numa única face f_1, bem como das faces f_{21} e f_{22} na face f_2. A coalescência de duas faces é feita pela remoção da aresta comum às duas faces assim como uma desta duas faces. No caso das faces f_{11} e f_{12}, a aresta a remover é e_1^*, ao passo que no caso das faces f_{21} e f_{22} a aresta a remover é e_2^*. Por fim, é feita a contracção da aresta e num dos seus pontos que, no algoritmo BSP é o seu ponto médio. Este ponto transforma-se depois num vértice. Esta operação remove, além da aresta e, um dos seus vértices que no caso da Figura 4.2 é o vértice v^*.

O algoritmo BSP apesar de ser mais rápido que os outros algoritmos da sua classe (ver Tabela 4.1) e de preservar a forma global do objecto (ver Figura 4.3), a qualidade das malhas geradas não é satisfatória porque produz triângulos irregulares, ou seja, triângulos estreitos e alongados.

A qualidade da malha pode ser avaliada por diferentes métodos (ver [127, 90, 2, 17] para mais detalhes). No trabalho aqui apresentado, a qualidade da malha foi avaliada usando a ferramenta geométrica designada por *Metro* [17]. Esta ferramenta permite calcular os erros máximo e médio de uma malha simplificada em relação à malha original. No algoritmo BSP, é possível melhorar a qualidade da malha escolhendo um

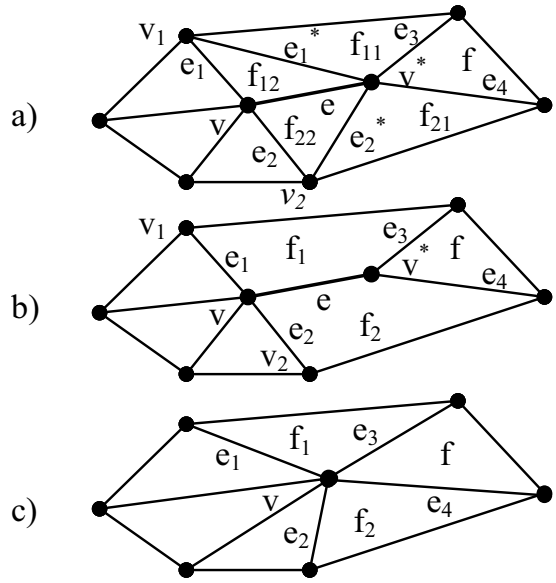


Figura 4.2: Simplificação por contracção de uma aresta (e).

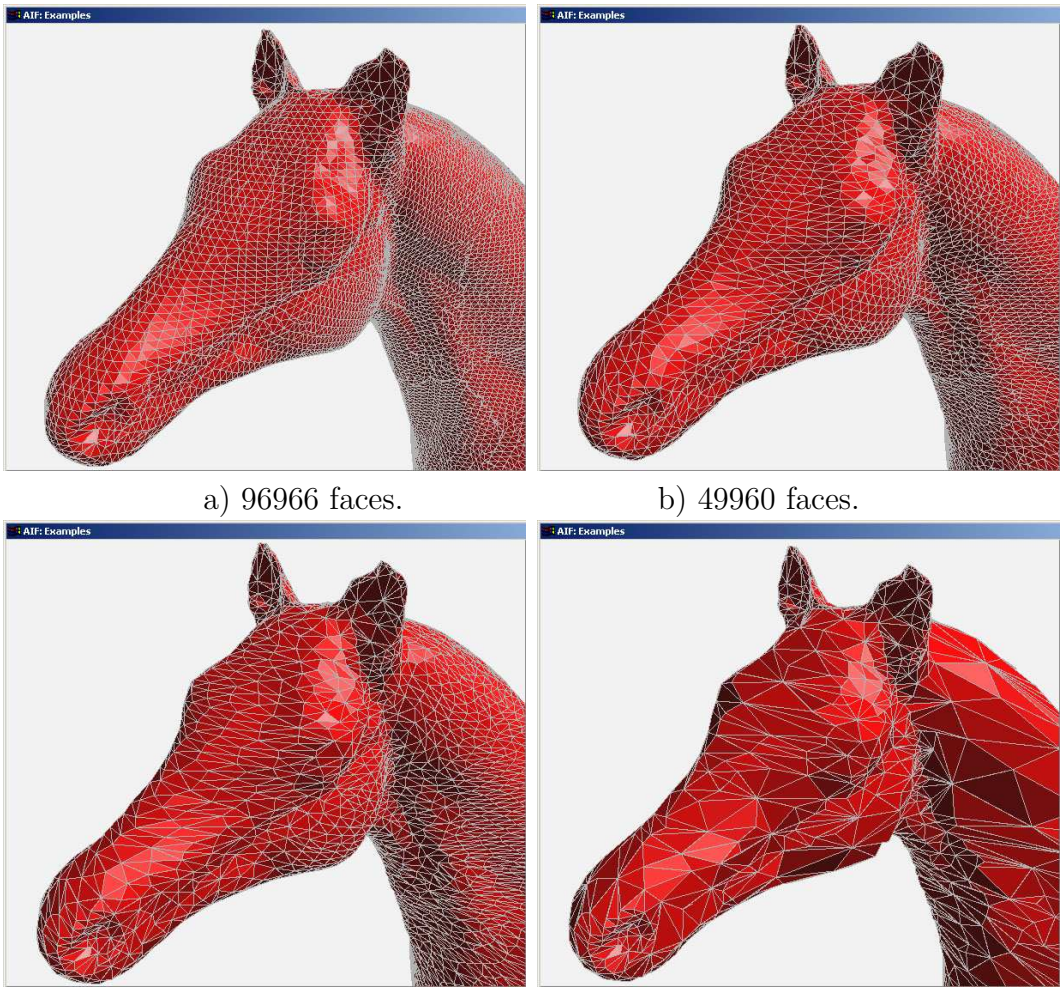
ponto que minimize o erro da malha na operação de contracção de arestas, em vez de usar sempre o ponto médio da aresta, mas isso iria reflectir-se num acréscimo de tempo de execução do algoritmo.

O maior problema do algoritmo BSP é, pois, a qualidade das malhas geradas logo após as primeiras simplificações. De facto, como se pode constatar pela Figura 4.3, o algoritmo pode gerar triângulos muito irregulares, ou seja, triângulos muito compridos e estreitos, o que não é aconselhável para efeitos de visualização. O fenómeno da irregularidade do comprimento dos lados dos triângulos acentua-se em cada simplificação da malha. Na Figura 4.3, este fenómeno não é muito notório em (b) mas começa a ser perceptível em (c) e mais ainda em (d).

4.3 Algoritmo NSA

O algoritmo NSA (*Normal-based Simplification Algorithm*) é outro algoritmo de simplificação de malhas baseado na contracção de arestas que surgiu como forma de solucionar o principal problema do algoritmo BSP: a qualidade das malhas geradas.

A operação de contracção consiste na contracção da aresta e dos seus vértices num único vértice, como mostram as Figuras 4.2 e 4.4. No algoritmo NSA, este vértice também é o ponto médio da aresta, mas nada impede que seja escolhido outro ponto da aresta. A operação de contracção (*edge collapse*) de uma aresta conduz à remoção de um vértice, três arestas e duas faces (Figura 4.4). De notar que a operação de contracção de arestas é invertível, sendo esta operação inversa designada por subdivisão do vértice (*vertex split*).



a) 96966 faces. b) 49960 faces.

c) 27872 faces. d) 10286 faces.

Figura 4.3: Diferentes níveis de detalhe criados com o algoritmo BSP.

A subdivisão de um vértice em dois requer que lhe seja passada alguma informação adicional como, por exemplo, as coordenadas do novo vértice e quais os outros vértices que permitirão definir as duas novas faces (Figura 4.4). Normalmente esta informação é armazenada durante a operação de contracção, o que permite depois refinar a malha de volta à sua resolução original como acontece nas malhas progressivas apresentadas por Hoppe [38].

Como se verá no Capítulo 5, a operação de contracção de arestas (*edge collapse*) e a operação inversa (*vertex split*) permitem definir uma representação multiresolução para malhas poligonais.

O algoritmo NSA é baseado na variação das normais às faces em redor da aresta a contrair [105]. Além disso, contrariamente ao que acontece com os outros algoritmos existentes na literatura, usa o mesmo critério quer para a escolha quer para a validação da aresta a contrair. De facto, a maioria dos algoritmos usa um critério para a simplificação e outro critério para a validação da aresta a contrair.

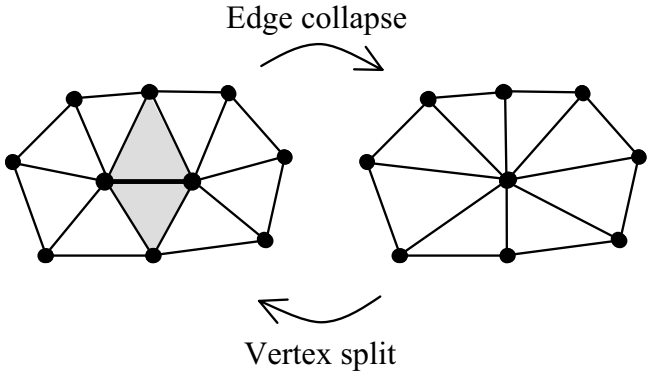


Figura 4.4: Contracção de uma aresta e subdivisão de um vértice.

4.3.1 Critério de Simplificação

A contracção de uma aresta no algoritmo NSA só acontece quando a variação das normais às faces em redor da aresta a contrair estiver contida numa dada tolerância ε. O valor de ε define a variação admissível para o ângulo entre as normais à face antes e depois da contracção da aresta. Este valor é especificado pelo utilizador num intervalo pré-definido. Quanto maiores forem os valores de ε maior será a simplificação da malha.

Este é um critério geométrico que requer que a região à volta da aresta a contrair seja aproximadamente complanar. Mas o contrário já não é verdade, ou seja, a complanaridade não assegura que exista uma variação mínima das normais às faces. Por exemplo, a Figura 4.5 mostra dois casos de faces complanares, mas só as faces da Figura 4.5(a) respeitam o critério visto que temos duas faces com a mesma orientação, enquanto que na Figura 4.5(b) temos duas faces com orientações opostas.

Note-se que o algoritmo NSA requer a quasi-complanaridade da região a simplificar, o que acaba por validar simultaneamente as operações de escolha e contracção de arestas, porque a operação de contracção só é permitida quando a variação das normais às faces for menor que um dado ε.

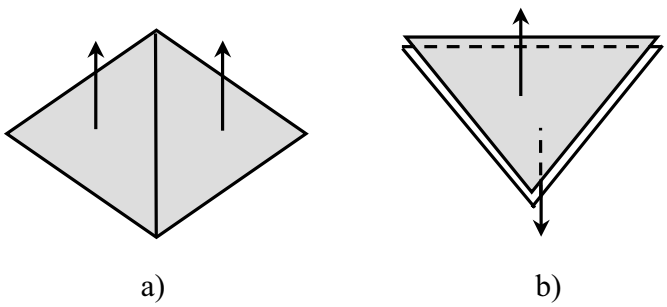


Figura 4.5: Exemplo de faces planas.

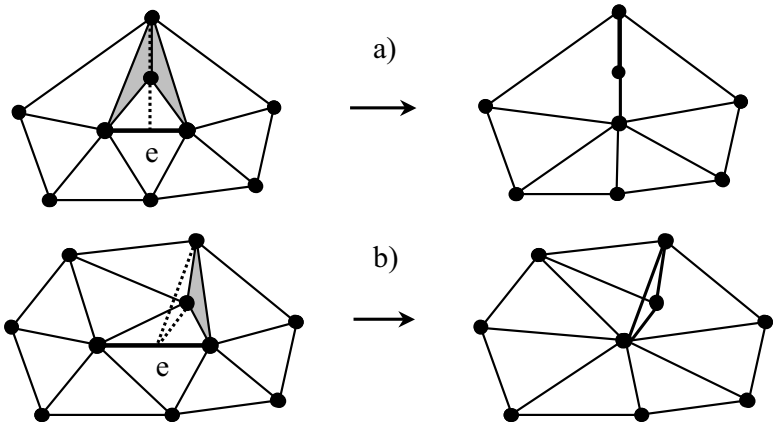


Figura 4.6: Validação da operação de contracção de arestas.

De facto, a operação de contracção de arestas pode causar inversões locais na superfície da malha, a menos que haja uma forma de controlar a sobreposição de faces (Figura 4.6(a)) ou a criação de concavidades (Figura 4.6(b)) na malha, que podem aparecer depois da contracção de uma aresta. Por exemplo, na Figura 4.6(a) a contracção da aresta e provoca a sobreposição das faces sombreadas como se indica pelo segmento a traço interrompido. Similarmente, a face sombreada na Figura 4.6(b) passa a estar na posição indicada pela face a traço interrompido, o que significa que muda de orientação depois da contracção da aresta e. Em ambos os casos o algoritmo NSA não permite a contracção da aresta e porque a variação das normais seria maior do que ε, e assim sendo faz simultaneamente a validação da operação de contracção de arestas.

Para a validação da operação de contracção de arestas muitos autores recorrem a heurísticas. Por exemplo, Hoppe et al. [41] usam um valor máximo para o ângulo entre arestas incidentes num vértice, em vez de verificarem se na malha passou a haver sobreposição de faces ou concavidades depois da operação de contracção. Esta estratégia previne não só os casos indesejados, mas também o aparecimento de triângulos finos. Outros algoritmos usam também a normal antes e depois da operação de contracção para validar a aresta a contrair [31], mas usam outro critério para a escolha da aresta a contrair, ou seja, outro critério de simplificação.

Resumindo, o algoritmo NSA simplifica a malha apenas em zonas aproximadamente complanares, o que faz com que produza pequenas variações nas normais às faces. Logo o critério de escolha da aresta a contrair permite simultaneamente validar a operação de contracção de arestas.

4.3.2 Resultados do Algoritmo NSA

O algoritmo NSA produz simplificações com boa qualidade que preservam a forma global do objecto e as suas fronteiras como se pode ver na Figura 4.7. A forma é

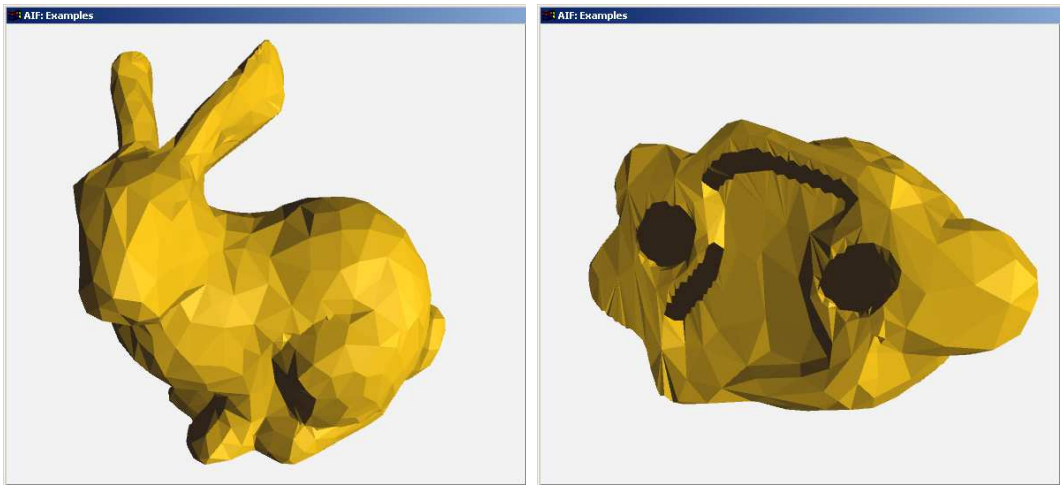


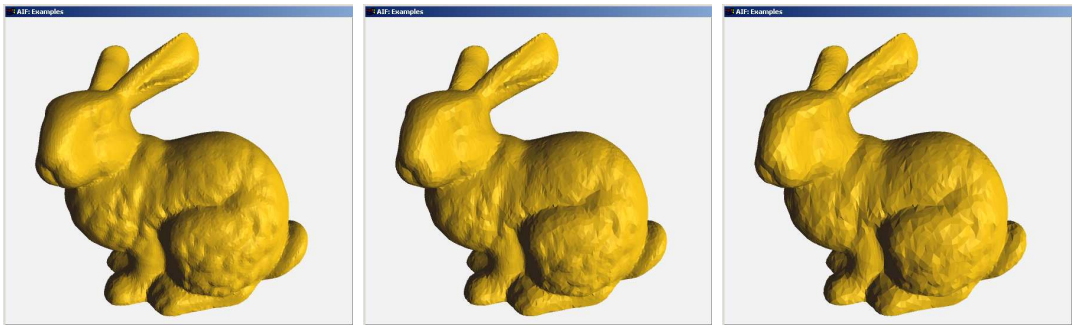
Figura 4.7: Preservação da forma e fronteira da malha (2051 faces).

preservada quer a malha seja fechada ou não. No caso das malhas abertas (ou com fronteira) a fronteira da malha é preservada visto não ser permitida a contracção de uma aresta que contenha arestas vizinhas pertencentes à fronteira da malha. Isto é, uma aresta da fronteira da malha contém apenas uma face incidente, o que é uma situação simples de identificar através da estrutura de dados AIF. Desta forma é possível manter a fronteira da malha para qualquer resolução.

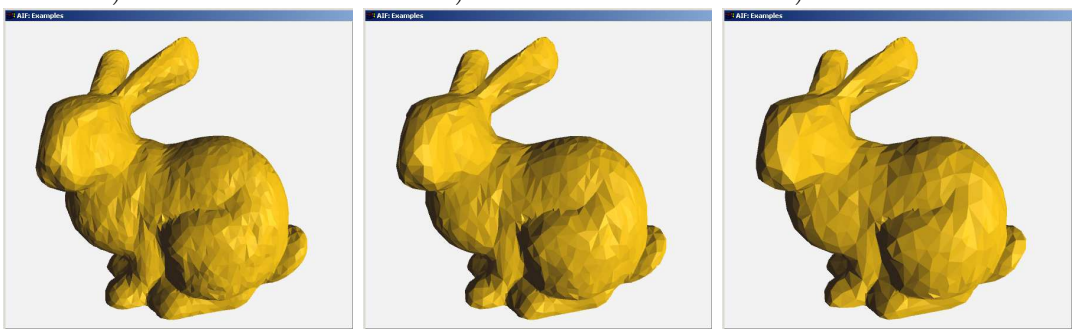
Para uma dada tolerância ε, o algoritmo NSA produz um número finito de simplificações do modelo. Por exemplo, a Figura 4.8 mostra as primeiras seis simplificações criadas pelo algoritmo NSA para a malha do coelho com $\varepsilon = 0.025$. Estas são as simplificações mais significativas, ou seja, onde a redução do número de faces de uma simplificação para a seguinte é superior a 1000 faces. Contudo, o algoritmo pode criar ainda mais algumas simplificações, mas a redução do número de faces começa a ser cada vez menos significativa. No limite, haverá uma simplificação a partir da qual não é criada mais nenhuma outra para o mesmo valor de ε. Na Figura 4.9 são apresentadas para a malha do coelho algumas das simplificações menos significativas, bem como a última simplificação com apenas 1263 faces a partir da qual não é criada mais nenhuma outra para o mesmo valor de $\varepsilon = 0.025$ (Figura 4.9(c)).

Nas primeiras simplificações o algoritmo NSA reduz o número de faces da malha em mais de 45%, como é o caso da malha do coelho na Figura 4.8. Note-se que em alguns casos chega a atingir mesmo os 49%, como é o caso da malha do cavalo e da malha de Venus que aparecem no final deste capítulo.

De modo a ter uma simplificação mais uniforme de toda a malha e evitar o aparecimento de triângulos irregulares foi ainda imposta a seguinte restrição ao algoritmo NSA: em cada operação de contracção de arestas, todas as aresta incidentes no novo vértice ficam impossibilitadas de ser alvo de uma operação de contracção durante a simplificação corrente. Isto evita a sobre-simplificação da malha numa mesma região, o que conduz a uma aceleração do processo de simplificação. Isto evita também uma



a) 36767 faces. b) 19967 faces. c) 11229 faces.

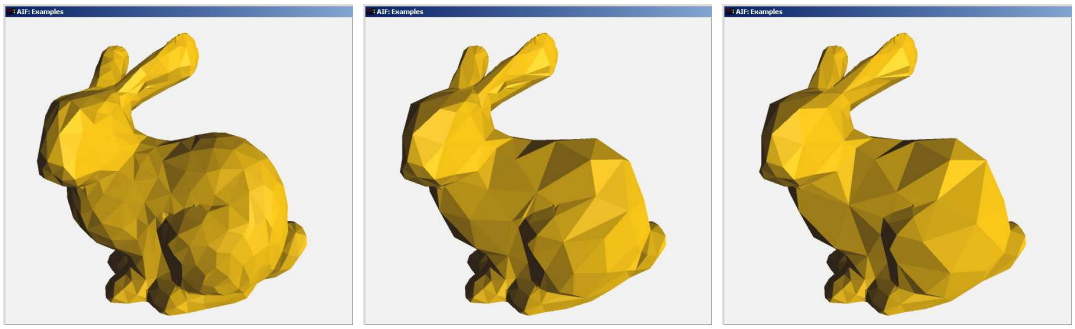


d) 6629 faces. e) 4099 faces. f) 2785 faces.

Figura 4.8: Simplificações mais significativas da malha do coelho com $\varepsilon = 0.025$.

grande alteração de forma de cada zona da malha, o que se traduz pela manutenção da qualidade da malha.

O refinamento da malha também é possível aplicando a operação inversa (*vertex split*) aos vértices criados em cada simplificação. Neste caso, e durante a operação de contracção, é armazenada pelo menos a posição de um dos vértices originais, permitindo voltar à aproximação anterior. De modo a distinguir os vértices criados em cada simplificação é usado um vector de vectores, onde cada entrada contém os vértices de cada simplificação.



a) 2051 faces. b) 1367 faces. c) 1263 faces.

Figura 4.9: Simplificações menos significativas da malha do coelho com $\varepsilon = 0.025$.

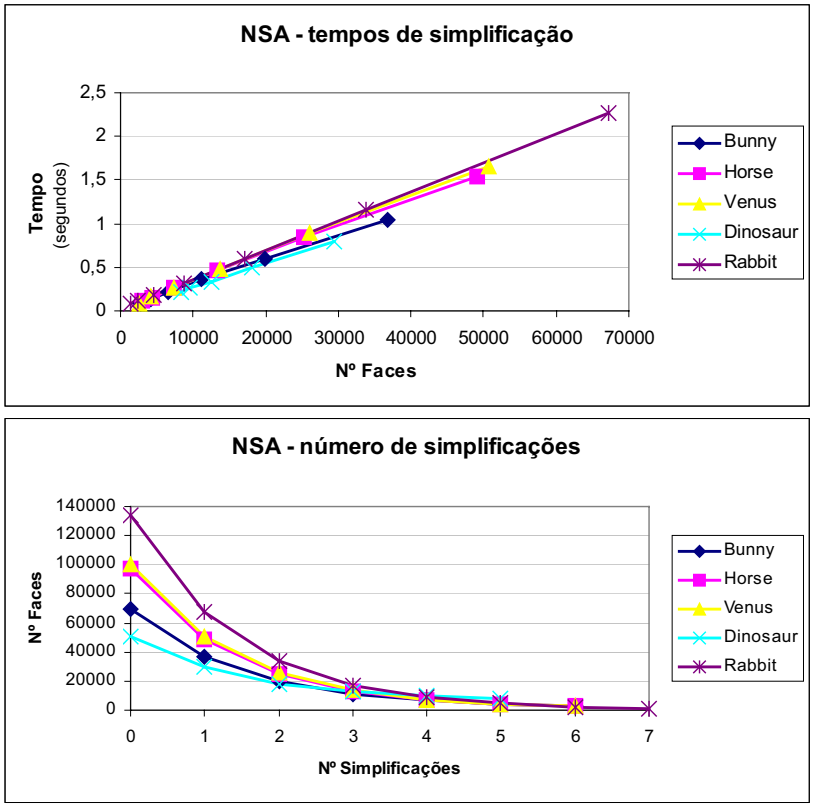


Figura 4.10: Resultados do algoritmo NSA para diferentes malhas.

O desempenho do algoritmo NSA depende principalmente do tamanho do modelo, mas também do número de aproximações criadas para um dado modelo, como mostra a Figura 4.10. Como seria de esperar (Figura 4.10 NSA - tempos de simplificação) as malhas de maiores dimensões têm tempos de simplificação maiores. Na prática existe um número finito de aproximações para cada modelo e para cada valor de ε. Isto é, para um dado ε, qualquer modelo tende para um número mínimo de faces a partir do qual já não é possível simplificá-lo mais. Veja-se a Figura 4.10 (NSA - número de simplificações) para diferentes modelos com o mesmo valor de ε, onde o algoritmo NSA criou cinco aproximações mais significativas para a malha do dinossauro (Dinosaur), sete para o coelho (Rabbit) e seis para outro coelho (Bunny), cavalo (Horse) e Venus.

A Figura 4.11 mostra os tempos de simplificação e refinamento para a malha do coelho (Bunny) para dois valores diferentes do erro, $\varepsilon = 0.025$ e $\varepsilon = 0.1$. Pode constatar-se que os diferentes valores de ε produzem aproximações com diferente número de faces. Quanto maior for o erro, menor número de faces terá a malha. No entanto, os tempos de simplificação e refinamento não se alteram muito, mesmo para dois valores distintos de ε.

Nas Figuras 4.14 e 4.15, que se encontram no final do capítulo, são apresentados mais alguns resultados do algoritmo NSA para diversas malhas com diferente número de faces, onde se pode ver a malha original, a primeira e a última simplificações mais

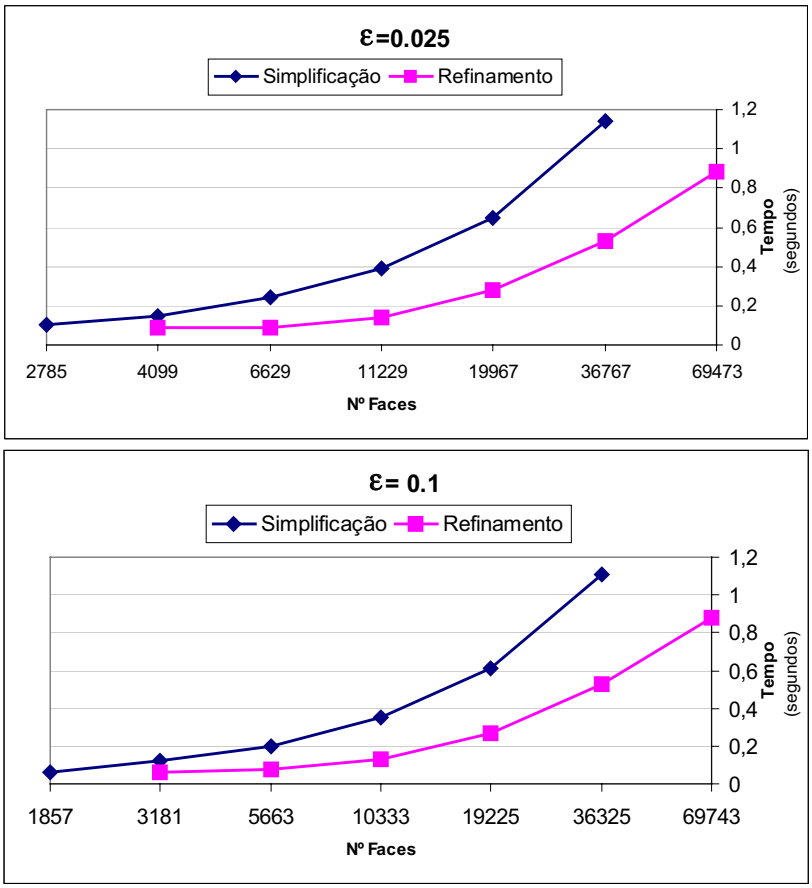


Figura 4.11: Tempos de simplificação e refinamento para a malha do coelho (Bunny).

significativas para cada modelo com $\varepsilon = 0.025$.

No caso das malhas de maiores dimensões (Figura 4.15), como é o caso das malhas do elefante, do dragão e do Buda, que têm entre 290000 e 1100000 faces, o algoritmo NSA tende a criar mais aproximações do que para malhas pequenas (Figura 4.14). No entanto, essa diferença depende mais das características da malha (se tem muitas variações de forma ou não) do que da própria dimensão da malha. Por exemplo, na Figura 4.15 o número de aproximações para a malha do elefante foi de oito enquanto para a malha do dragão e do Buda foi de apenas nove. Como se pode ver a quantidade de faces da malha tem pouca influência no número de aproximações, já as características de forma da malha são determinantes.

4.3.3 Algoritmo NSA - Comparações

Os testes comparativos foram efectuados num PC equipado com um processador Pentium 4 a 1.6GHz, 768MB de RAM, uma placa gráfica GeForce 4 com 64MB e sobre o sistema operativo Windows 2000.

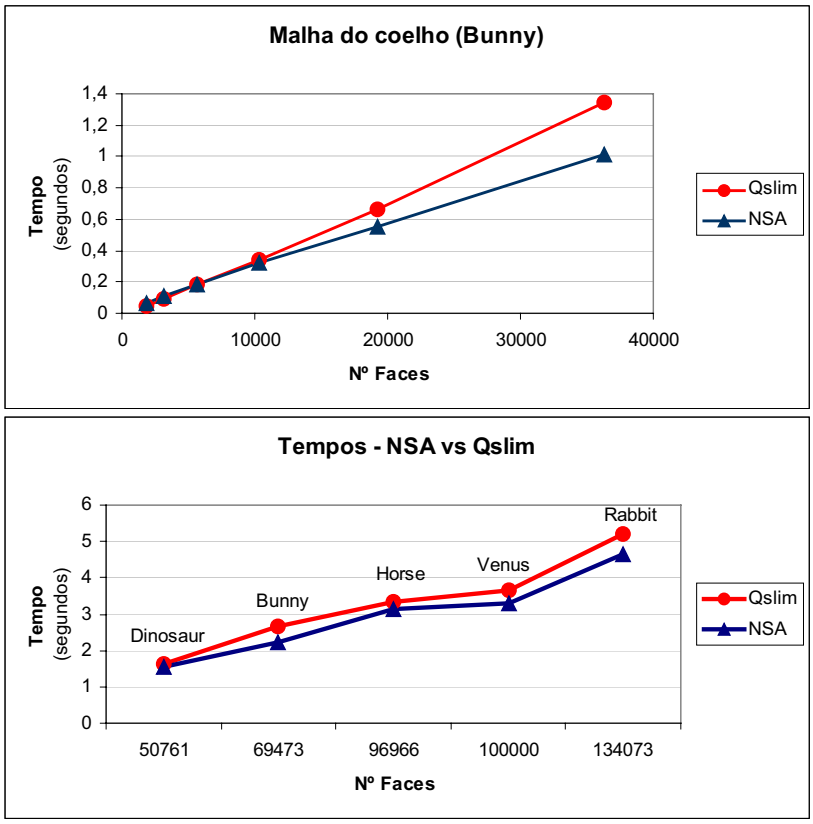


Figura 4.12: Tempos de simplificação NSA vs Qslim.

O algoritmo NSA foi desenvolvido sobre a estrutura de dados AIF. Como se viu no Capítulo 3, esta estrutura de dados permite o acesso bastante rápido à informação topológica, o que possibilitou o desenvolvimento de um algoritmo rápido.

A maior vantagem do algoritmo NSA é que o seu tempo de processamento é inferior ao dos outros algoritmos da sua classe de acordo com a literatura [16]. Pode dizer-se que isto se deve principalmente a três factores: (i) a usar um critério meramente geométrico e simples de calcular, (ii) de se contrair as arestas sempre no seu ponto médio, evitando cálculos adicionais, e (iii) devido à estrutura de dados AIF de suporte.

A melhor forma de analisar os resultados do algoritmo NSA é através da comparação com outros algoritmos. Assim, usou-se o algoritmo QSlim proposto por Garland e Heckbert [31] que, segundo Lindstrom e Turk [68], é o algoritmo mais rápido da sua classe. Além disso, o algoritmo QSlim é *freeware* e está disponível em `http://www.cs.cmu.edu/~garland`, o que permitiu correr ambos os algoritmos na mesma máquina para os mesmos modelos, ou seja, permitiu fazer uma comparação efectiva entre eles.

A Figura 4.12 apresenta os tempos de simplificação dos algoritmos NSA e QSlim para o modelo do coelho. Apresenta ainda os tempos totais de simplificação para a criação das mesmas simplificações para diferentes modelos por parte dos dois algoritmos, NSA

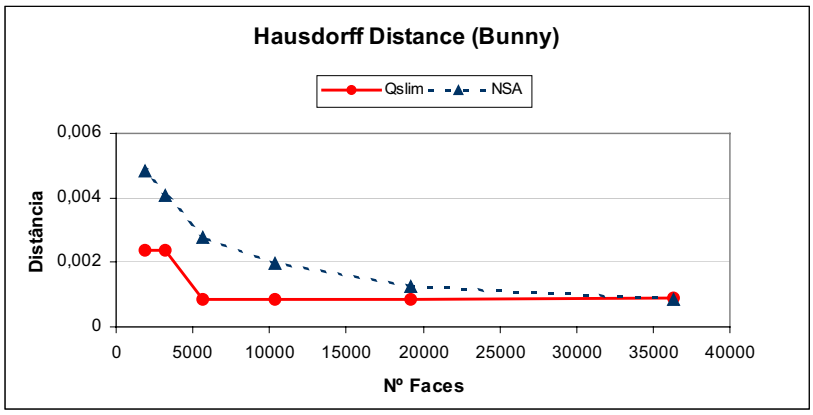


Figura 4.13: Qualidade da malha NSA vs Qslim.

e QSlim. Em geral, como mostra a Figura 4.12 o algoritmo NSA é mais rápido que o algoritmo QSlim para os diversos modelos.

4.3.4 Qualidade da Malha

A avaliação da qualidade da malha foi, mais uma vez, feita recorrendo à ferramenta geométrica *Metro* [17]. Esta ferramenta permite não só calcular os erros máximo e médio, mas também a distância de *Hausdorff* entre a malha simplificada (B) e a malha original (A) o que permite medir a qualidade da malha. A distância *Hausdorff* é dada por:

$$H(A,B) = max\ \{\ h(A,B), h(B,A)\ \}$$

onde $h(A,B)$ é dada por:

$$h(A,B) = \max_{a\in A}\ \{\ \min_{b\in B}\ \{\ d(a,b)\ \}$$

com $d(a,b)$ a distância euclidiana entre os pontos a e b. Note-se que em alguns casos $h(A,B) \neq h(B,A)$.

A Figura 4.13 mostra, para a malha do coelho (Bunny), uma comparação entre a qualidade da malha gerada pelo algoritmo NSA e pelo algoritmo QSlim usando a distância *Hausdorff* como factor de comparação. A distância *Hausdorff*, e claro a qualidade da malha, é similar em ambos os algoritmos para as primeiras aproximações, mas já é significativamente diferente nas últimas simplificações. Nas últimas simplificações, o algoritmo QSlim consegue criar malhas com melhor qualidade que as criadas pelo algoritmo NSA. Isto deve-se ao facto do algoritmo NSA só usar a aproximação corrente para criar a aproximação seguinte, não usando informação da malha original ou aproximações anteriores, como acontece no caso do algoritmo QSlim. Além disso, o

algoritmo NSA efectua a contracção de uma aresta sempre no seu ponto médio, e nem sempre esse será o ponto que minimiza o erro introduzido, enquanto que o algoritmo QSlim escolhe sempre o ponto que minimiza o erro, o que resulta numa maior qualidade da malha. Se se escolhesse o ponto que minimiza o erro no algoritmo NSA, este facto iria reflectir-se num acréscimo de tempo de execução do algoritmo.

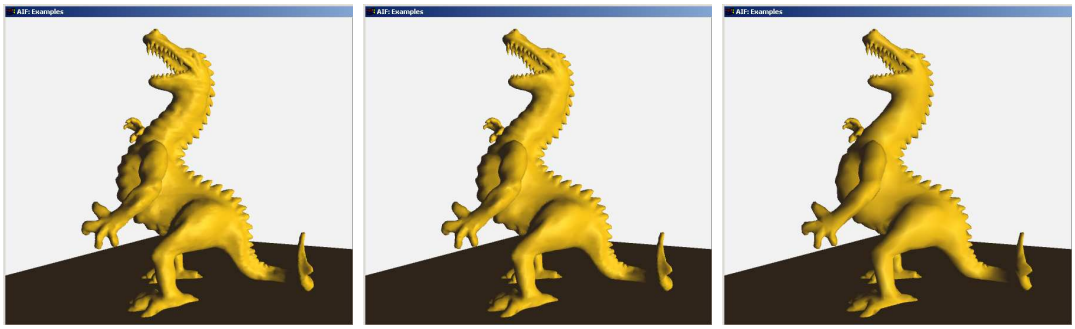
4.4 Sumário

A complexidade e os detalhes de forma dos modelos geométricos tem vindo a aumentar muito devido principalmente à crescente sofisticação dos sistemas de modelação e às tecnologias de aquisição de dados 3D. Normalmente estes modelos geométricos são malhas poligonais que têm muitas vezes um tal número de células que torna difícil o seu armazenamento e visualização em computador.

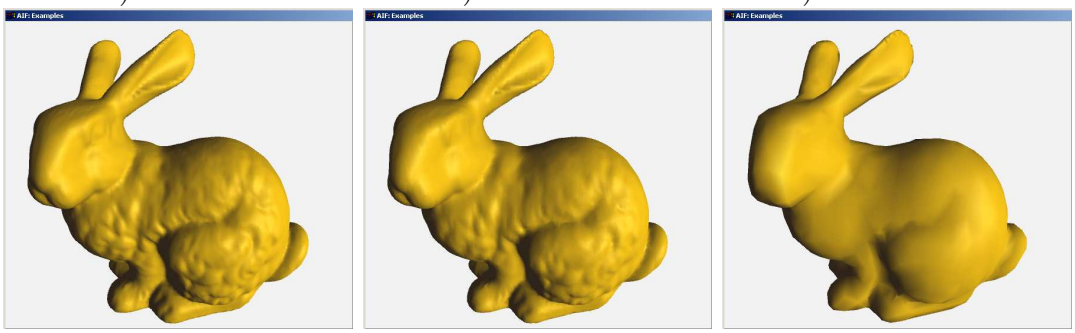
Para resolver estes problemas surgiram os algoritmos de simplificação de malhas. Assim, neste capítulo descreveram-se dois algoritmos de simplificação de malhas desenvolvidos no âmbito desta tese, o BSP e o seu sucessor NSA.

O algoritmo NSA é o algoritmo de simplificação de malhas mais rápido dentro da sua classe (contracção de arestas) pelas razões apresentadas anteriormente. Além disso, consegue reduzir o número de faces de uma malha até 49% numa única simplificação. Pode também dizer-se que o algoritmo NSA tem uma boa relação entre a sua "rapidez"e a qualidade das malhas geradas. Aliás, a prática tornou claro que existe sempre um compromisso entre a rapidez do algoritmo de simplificação e qualidade dos resultados.

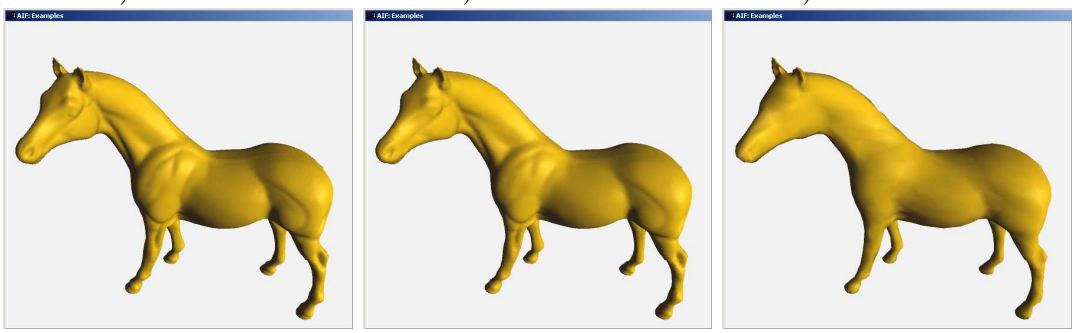
Resumindo, o algoritmo NSA produz bons resultados em termos do seu tempo de execução, da qualidade da malha gerada e da preservação da forma do objecto, conforme é necessário para a maioria das aplicações.



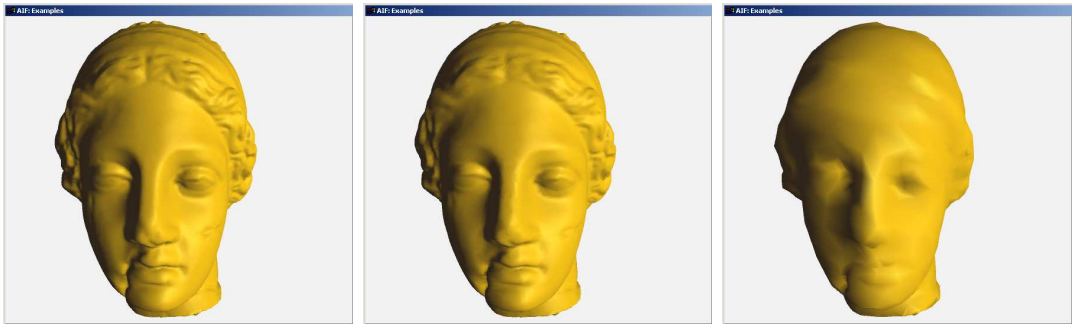
a) 50761 faces. b) 29375 faces. c) 8309 faces.



a) 69473 faces. b) 36767 faces. c) 2785 faces.

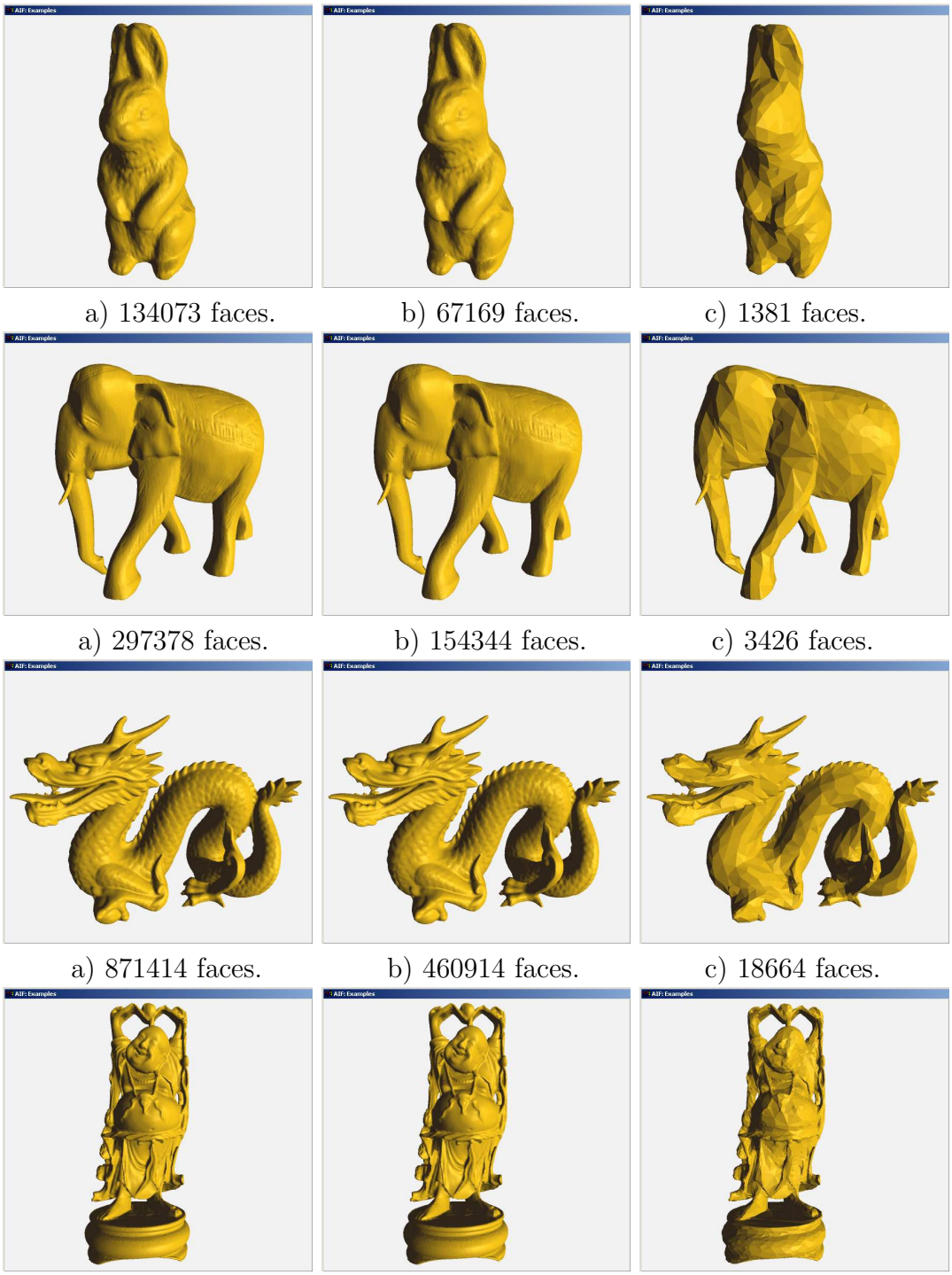


a) 96966 faces. b) 49106 faces. c) 2904 faces.



a) 100000 faces. b) 50784 faces. c) 2546 faces.

Figura 4.14: Resultados do algoritmo NSA para malhas de pequenas dimensões, ou seja, com o número de faces ≤ 100000 e $\varepsilon = 0.025$.



a) 134073 faces. b) 67169 faces. c) 1381 faces.

a) 297378 faces. b) 154344 faces. c) 3426 faces.

a) 871414 faces. b) 460914 faces. c) 18664 faces.

a) 1085634 faces. b) 584116 faces. c) 37052 faces.

Figura 4.15: Resultados do algoritmo NSA para malhas de grandes dimensões, ou seja, com o número de faces > 100000 e $\varepsilon = 0.025$.

Capítulo 5

Edição de Malhas Poligonais

Este capítulo descreve uma nova metodologia para a edição interactiva de malhas poligonais. Esta metodologia baseia-se na subdivisão de uma malha em sub-malhas, às quais se pode depois aplicar transformações geométricas. Isto permite circunscrever as operações de edição a regiões de interesse da malha. Significa isto que há, de certo modo, um controlo local sobre a deformação da malha. Esta metodologia de edição pode ser aplicada quer a malhas poligonais usuais, quer a malhas multiresolução. No caso de malhas multiresolução, pode editar-se mais facilmente malhas com grande número de células porque é sempre possível simplificá-la, editá-la e depois refiná-la de volta à sua resolução original.

5.1 Malhas Poligonais

As malhas poligonais são cada vez mais usadas para representar objectos tridimensionais em computação gráfica, sendo muitas vezes geradas por *scanners* tridimensionais. No entanto, editar ou deformar uma malha gerada por um *scanner* 3D não é uma tarefa fácil, pois a malha é vista como um todo, não sendo possível distinguir uma região (sub-malha) de outra. Assim, não é possível distinguir a que região pertence um determinado vértice. Por exemplo, não é possível dizer se um dado vértice pertence à perna esquerda ou à orelha direita de um coelho.

As ferramentas interactivas de edição de malhas são por isso cada vez mais importantes para uma grande variedade de aplicações, desde da modelação geométrica à animação ou aos ambientes virtuais.

Na edição de uma malha pode ter-se um controlo mais intuitivo da deformação da malha através da alteração da posição dos vértices ou aplicando transformações geométricas às sub-malhas (regiões da malha definidas pelo utilizador) no sentido que vão de encontro às expectativas do utilizador quanto à deformação da malha. Por contraposição, para deformar uma superfície paramétrica basta alterar a posição de

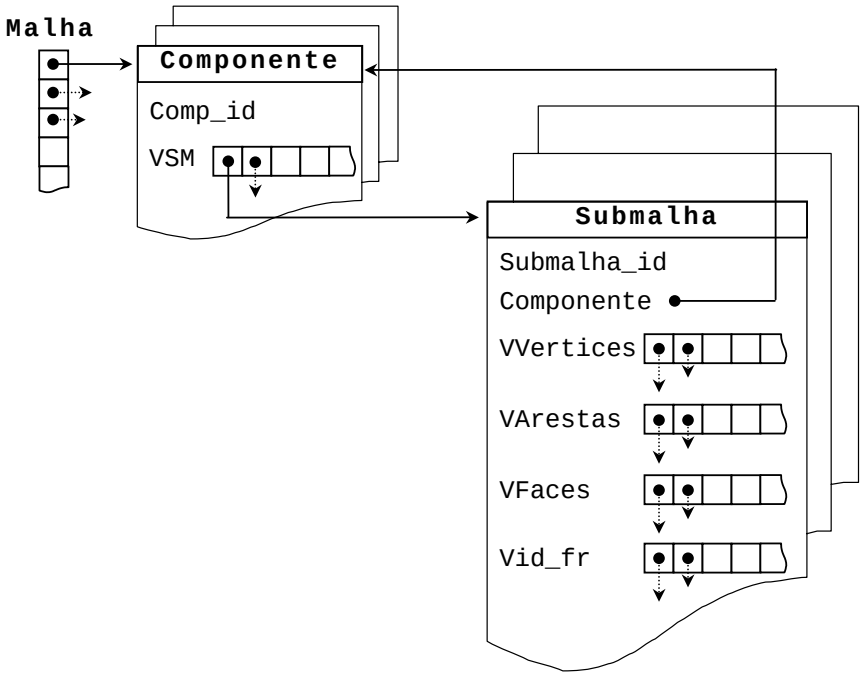


Figura 5.1: Estrutura de dados AIF modificada para suportar sub-malhas.

pelo menos um dos seus pontos de controlo. No entanto, nem sempre é intuitiva a deformação que a superfície terá após a edição dos pontos de controlo.

Para que o utilizador possa definir regiões de interesse na malha (ou sub-malhas) de forma a tornar mais fácil a sua edição houve a necessidade de modificar a estrutura de dados AIF, como se descreve na secção seguinte.

5.2 Estrutura de Dados AIF Modificada

A estrutura de dados AIF, apresentada no Capítulo 3, foi modificada de modo a suportar componentes e sub-malhas de acordo com o esquema da Figura 5.1. Uma sub-malha é um sub-conjunto bidimensional de células interligadas de uma malha. Cada sub-malha é definida interactivamente pelo utilizador.

Uma componente de uma malha M é a sub-malha maximal de M, ou seja, é uma malha que contém todas as células de M interligadas. Assim, uma malha (`Malha`) pode ser vista como um vector de ponteiros para componentes. Mas se uma malha possuir várias componentes a sua intersecção é o conjunto vazio.

Uma componente (`Componente`) é uma estrutura que contém um identificador (`Comp_id`) e um vector (`VSM`) de ponteiros para sub-malhas. Neste caso, se uma componente possuir várias sub-malhas, a sua intersecção pode ser o conjunto vazio ou uma sub-malha, pois é possível definir sub-malhas dentro de sub-malhas.

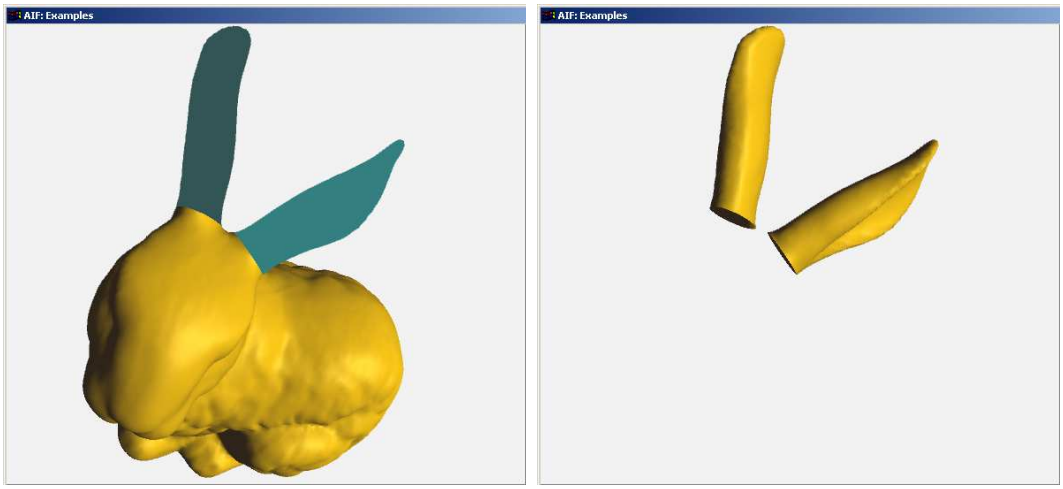


Figura 5.2: Subdivisão da malha do coelho em sub-malhas (orelhas).

Uma sub-malha (`Submalha`) é uma estrutura que, para além do identificador (`Submalha_id`) da sub-malha e do ponteiro para a componente (`Componente`) a que pertence, possui três vectores de ponteiros: um para os vértices (`VVertices`), outro para as arestas (`VArestas`) e o outro para as faces (`VFaces`). Além disso, possui ainda um vector (`Vid_fr`) com as arestas que delimitam a sub-malha (ou seja, a sua fronteira).

Portanto, a estrutura de dados AIF modificada suporta malhas com uma ou mais componentes, onde cada componente pode ter várias sub-malhas. Por exemplo, na Figura 5.2 tem-se a malha do coelho com apenas uma componente. Essa componente possui três sub-malhas: duas para as orelhas do coelho e outra para o resto do corpo. Isto não significa que a estrutura de dados AIF modificada não suporta uma hierarquia de sub-malhas, ou seja, sub-malhas que possuem outras sub-malhas.

A malha do coelho da Figura 5.2 pode incluir, para além das sub-malhas das orelhas, uma sub-malha para a cabeça que possui as sub-malhas das orelhas. Isto permite editar a sub-malha da cabeça como um todo mas não elimina a possibilidade de editar separadamente cada uma das sub-malhas das orelhas. Esta hierarquia de sub-malhas é muito útil na edição de malhas e constitui, talvez, uma das principais contribuições desta tese no que se refere à edição de malhas. Na secção 5.4 serão vistos alguns exemplos de edição de malhas usando hierarquia de sub-malhas.

5.3 Sub-malhas

Uma malha convencional é normalmente um objecto no qual, do ponto de vista da estrutura de dados, não é possível distinguir sub-objectos. De facto, geralmente a estrutura de dados não tem forma de distinguir um vértice da orelha do coelho de um vértice do corpo do coelho. Isto torna difícil a edição e manipulação de malhas.

Figura 5.3: Malha de uma vaca e malha com a cabeça da vaca.

Pelo contrário, as sub-malhas facilitam a manipulação de malhas por aplicação de técnicas conhecidas de edição a regiões da malha. Isto deve-se ao facto de uma sub-malha ser uma parte da malha que pode ser manipulada independentemente do resto da malha, pois sabe-se sempre quais são as células que lhe pertencem.

A principal ideia para a edição interactiva de uma malha é pois a sua partição em sub-malhas, permitindo depois editá-las através de transformações geométricas independentemente do resto da malha.

As sub-malhas permitem editar *directamente* qualquer malha triangular ou poligonal. As operações de edição estão confinadas à sub-malha, podendo ou não afectar as células exteriores à sub-malha. Neste caso, as operações de edição utilizadas são transformações geométricas como a variação de escala, a rotação, a variação de escala progressiva (*tapering*) e a rotação progressiva (*twisting*).

As sub-malhas permitem também a criação de outras malhas. De facto, pode gravar-se qualquer sub-malha em ficheiro passando esta a ser uma malha autónoma que passa a ser re-utilizável em qualquer altura. Por exemplo, na Figura 5.3 vê-se a malha de uma vaca e a malha com a cabeça da vaca que foi criada a partir da sub-malha da cabeça da vaca. Na malha da vaca existem 5804 faces já na malha com a cabeça existem apenas 2126 faces.

5.3.1 Criação de Sub-malhas

A criação de uma sub-malha é feita através da selecção interactiva de um colar de arestas ou faces (ou seja, definindo a sua fronteira) como mostra a Figura 5.4(a) para a cabeça da vaca. Este processo é aceitável no caso de malhas com um número pequeno de células, mas no caso de malhas com um número grande de células torna-se moroso. Alternativamente, pode definir-se uma sub-malha através de uma sequência de arestas

a) b)

Figura 5.4: Criação de sub-malhas.

que aproxima a intersecção de um plano (designado de plano de corte) com a malha. Por exemplo, a Figura 5.4(b) mostra a definição da sub-malha da cabeça da vaca usando este método, onde se pode ver o plano de corte e a fronteira calculada através de uma sequência de arestas que aproximam o plano de corte.

A criação de sub-malhas pode também ser feita em malhas multiresolução. Assim, é possível primeiro criar uma versão simplificada da malha na qual é mais fácil definir sub-malhas. Uma malha multiresolução deve simplificar-se para facilitar a criação e edição de sub-malhas, sendo depois possível refiná-la para a sua resolução original. Por exemplo, na Figura 5.5(a) é difícil definir interactivamente um colar de faces para delimitar a sub-malha da cabeça do coelho. Mas já é fácil fazer isso na versão simplificada da malha da Figura 5.5(b), pois esta possui um número reduzido de faces. A Figura 5.5 mostra a malha multiresolução do coelho, onde se vê a malha original (Figura 5.5(a)), a malha simplificada (Figura 5.5(b)), a criação de uma sub-malha para a cabeça do coelho na malha simplificada (Figura 5.5(c)), e a malha refinada com a sua resolução original e já com a correspondente sub-malha actualizada (Figura 5.5(d)).

No caso das malhas multiresolução é necessário ter cuidados especiais no processo de refinamento de forma a manter a consistência da fronteira das sub-malhas, ou seja, manter sempre a fronteira fechada. De facto, quando uma malha é refinada através da operação de subdivisão de um vértice (*vertex split*) como um todo, pode acontecer que o novo vértice a inserir na malha pertença ou não à fronteira da sub-malha. Isto é, quando se faz a subdivisão de um vértice da fronteira da sub-malha três situações podem ocorrer para a inserção do novo vértice: este fica na fronteira, ou na sub-malha, ou na outra parte da malha, como ilustra a Figura 5.6.

Considere-se que o vértice a subdividir é o vértice v_i da Figura 5.6(a). A fronteira é representada pelas arestas e vértices mais grossos. Ela separa uma sub-malha definida pelas arestas a traço contínuo da outra definida pelas arestas a traço interrompido. O

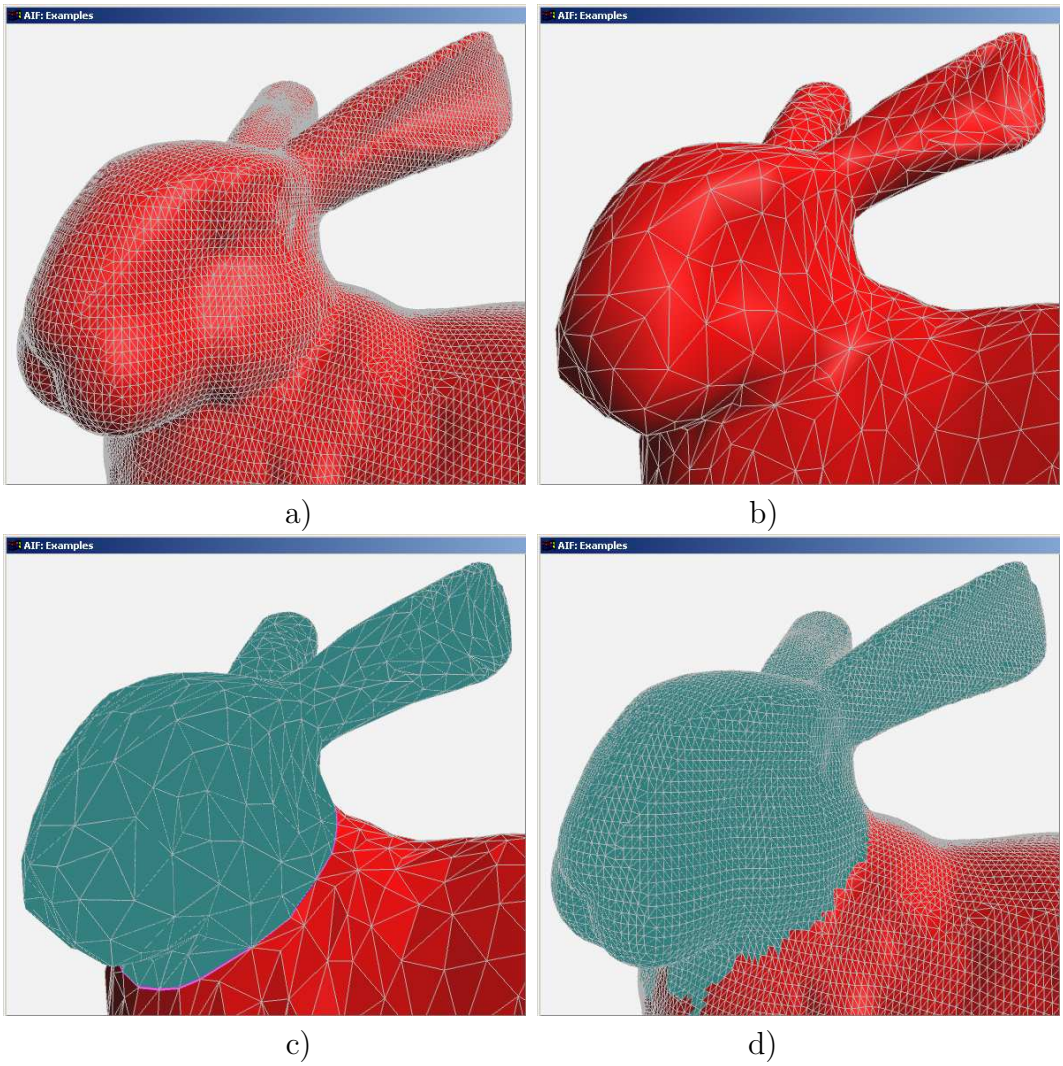


Figura 5.5: Criação de sub-malhas numa malha multiresolução.

novo vértice v_j a inserir pertencerá à fronteira das sub-malhas (Figura 5.6(b)), ou pertencerá à sub-malha representada a traço interrompido (Figura 5.6(c)) ou pertencerá à outra sub-malha (Figura 5.6(d)). A classificação do vértice v_j depende da classificação do vértice que foi subdividido v_i (ou seja, do vértice que lhe deu origem), mas também da classificação dos seus vértices vizinhos.

O novo vértice v_j pertencerá à fronteira se tiver vértices vizinhos que pertençam a ambas as sub-malhas. Caso contrário, pertencerá à sub-malha à qual pertencem os seus vértices vizinhos. Esta classificação é efectuada também para as arestas e faces por classificação dos seus vértices e arestas vizinhas, respectivamente. Assim, é possível definir sub-malhas para qualquer nível de resolução. Por exemplo, quando não é fácil definir uma sub-malha na resolução original devido ao elevado número de faces, como é o caso da Figura 5.5.

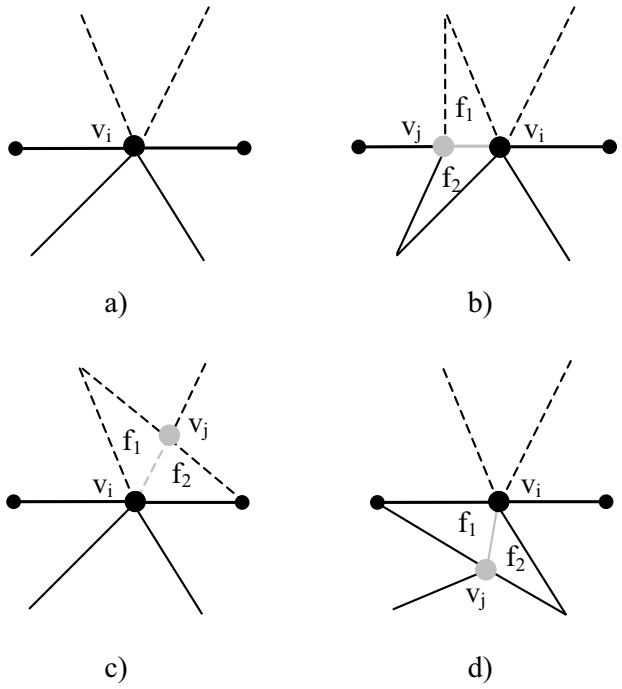
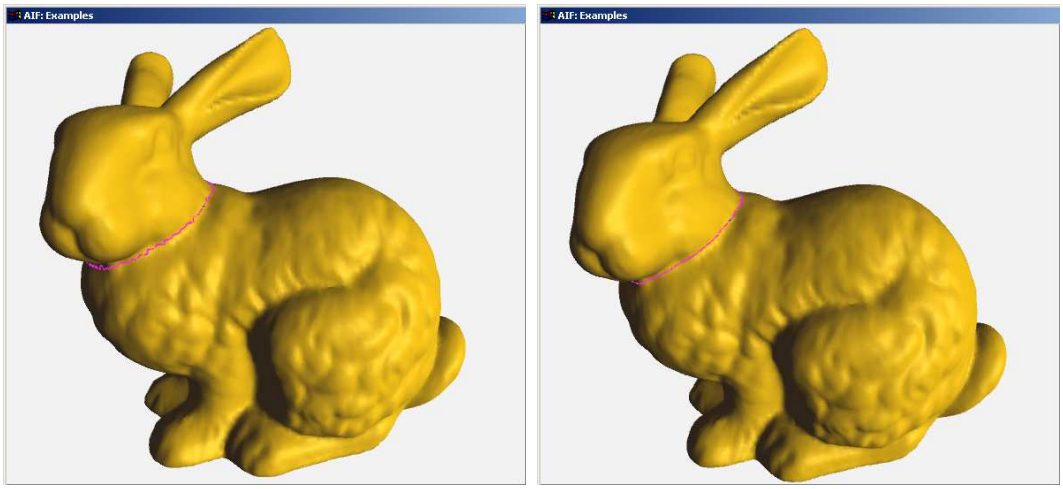


Figura 5.6: Operação de subdivisão do vértice (v_i) da fronteira de uma sub-malha.

5.3.2 Regularização da Fronteira de uma Sub-malha

Como se viu anteriormente, a construção de uma sub-malha faz-se por delimitação da sua fronteira, ou seja, por um contorno fechado de arestas. Contudo os contornos definidos na malha são muitas vezes irregulares devido à própria geometria da malha, como se pode ver pela fronteira da cabeça do coelho na Figura 5.7(a). De forma a poder ter um contorno (ou fronteira) regular, isto é que pertença a um plano (Figura 5.7(b)), há a necessidade de ajustar a geometria da malha na zona de fronteira, através da projecção dos vértices do contorno num plano, mas mantendo a sua topologia inalterada.

A regularização do contorno começa pela eliminação de cristas do contorno. Diz-se que o contorno tem uma crista quando quaisquer três vértices consecutivos, v_i, v_{i+1} e v_{i+2} do contorno pertencem à mesma face e as arestas definidas por v_i, v_{i+1} e v_{i+1}, v_{i+2} intersectam o plano de corte. Por exemplo, na Figura 5.8(a) o contorno definido pelos vértices de maior tamanho tem uma crista definida pelos vértices v_i, v_{i+1} e v_{i+2}, onde as arestas v_i, v_{i+1} e v_{i+1}, v_{i+2} intersectam o plano de corte representado a traço interrompido. A eliminação desta crista faz-se simplesmente removendo o vértice intermédio (v_{i+1}) do contorno como se vê na Figura 5.8(b). Este vértice v_{i+1} deixa assim de pertencer à sub-malha para passar a pertencer à malha que a contém. De notar que o aparecimento de cristas é um facto intrínseco ao próprio processo de selecção do contorno que aproxima o plano de corte, pois são os vértices mais próximos do plano de corte que são seleccionados.



a) b)

Figura 5.7: Re-definição da fronteira de uma sub-malha.

A regularização do contorno, depois de eliminadas as cristas, faz-se através da projecção de todos os vértices do contorno no plano de corte. Contudo tem de se determinar uma direcção de projecção em cada vértice de forma a minimizar a deformação localmente em torno do contorno da malha. No entanto, numa malha poligonal não é possível definir uma direcção tangente única em cada vértice como, por exemplo, numa superfície implícita em que existe uma única direcção tangente em cada ponto da superfície.

Logo, para cada vértice v do contorno calcula-se a intersecção das suas arestas incidentes com o plano de corte representado a traço interrompido na Figura 5.9(b), e determina-se o ponto médio das intersecções calculadas (Figura 5.9(c), pontos mais claros sobre o plano de corte), sendo estes a nova posição do vértice (v' na Figura 5.9(d)). Desta forma pode calcular-se um contorno regular definido com base num plano, por exemplo o plano de corte que permite definir a sub-malha. Este ajuste local da geometria da malha é feito mantendo a topologia da malha e minimizando a

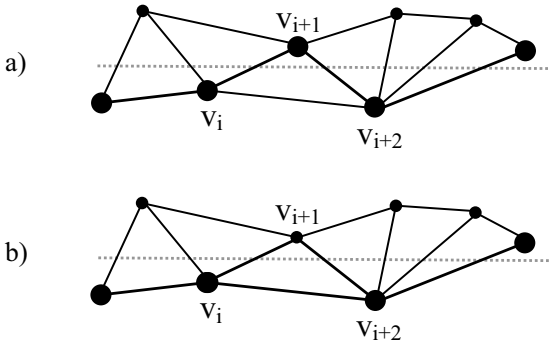


Figura 5.8: Eliminação de cristas do contorno que delimita uma sub-malha.

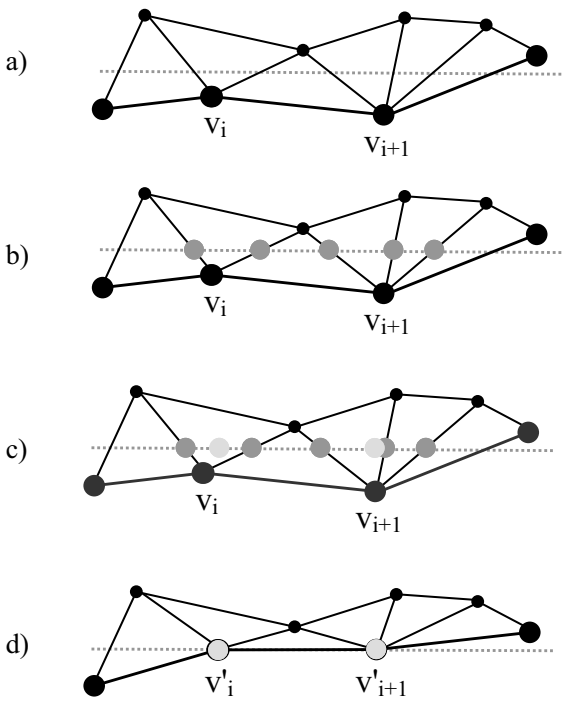


Figura 5.9: Planificação do contorno com base no plano de corte.

deformação daquela região.

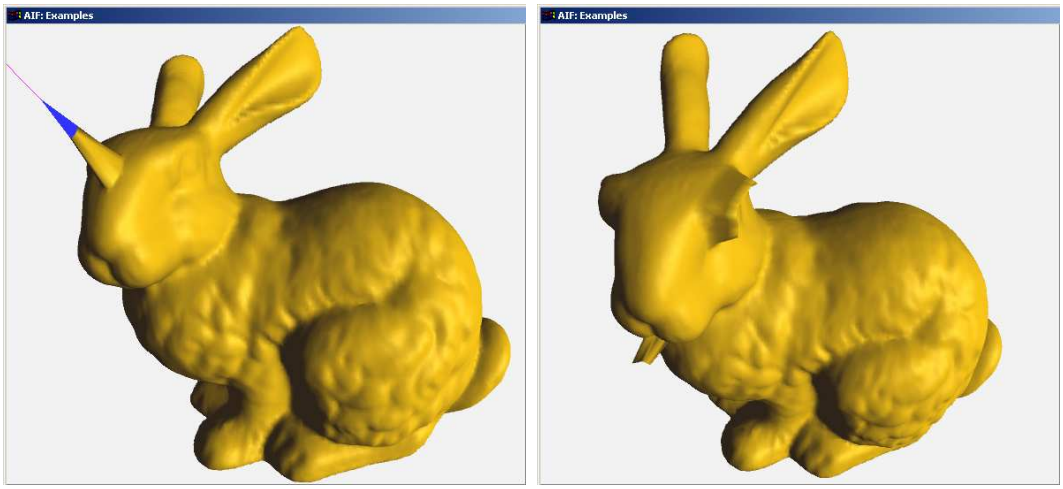
A criação de fronteiras regulares permite minimizar as deformações na zona de fronteira aquando da aplicação de transformações geométricas às sub-malhas, especialmente no caso de rotações que podem originar a sobreposição de vértices da fronteira da sub-malha. Neste caso, é mais fácil de corrigir a deformação se a fronteira for regular.

5.3.3 Propagação da Deformação na Zona de Fronteira

As operações de edição podem deformar consideravelmente a zona de fronteira de uma sub-malha (ou seja, os colares de faces adjacentes à fronteira). Por isso é necessário muitas vezes suavizar essa região, e isso pode ser feito de duas maneiras:

- Durante a operação de edição da sub-malha. Neste caso, propaga-se a deformação a alguns níveis de faces (colares de faces) para além da fronteira da sub-malha, isto é, propagando o efeito da deformação à malha que a contém;
- Aplicar um operador de suavização depois da operação de edição. Neste caso é feita uma correcção da geometria de cada vértice com base na geometria dos seus vértices vizinhos.

Em qualquer operação de edição tem de se definir o número de colares de faces que serão afectados pela deformação, ou seja o nível de propagação. A definição do nível



a) b)

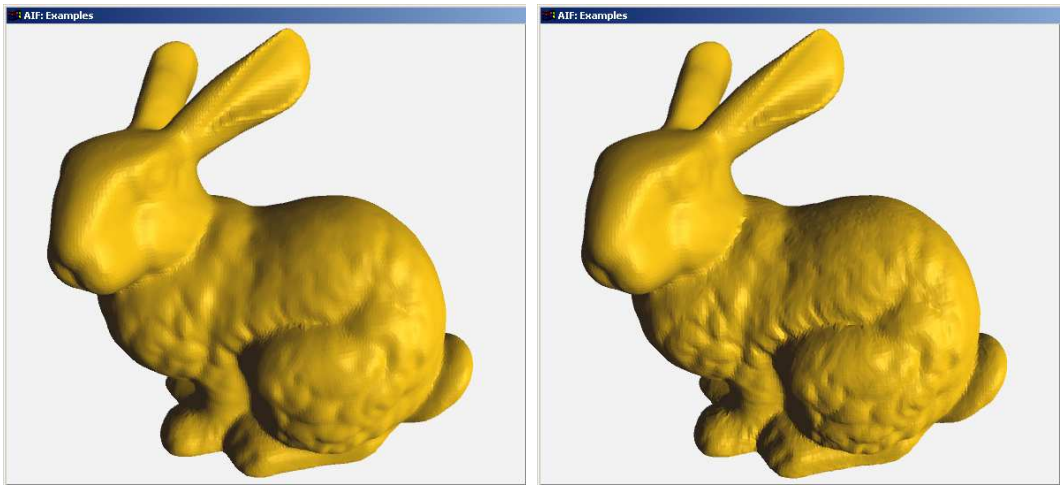
Figura 5.10: Edição de vértices.

de propagação tem com objectivo suavizar a deformação no colar de faces na zona de fronteira. Esta suavização é conseguida através da propagação da deformação aos vários colares de faces de forma progressiva. Por exemplo, se a uma sub-malha for aplicada um rotação de 20 graus com nível de propagação igual a 3, isto significa que o primeiro colar de faces depois da fronteira pode ser afectado por uma rotação de 15 graus, o segundo colar de faces por uma rotação de 10 graus e o terceiro e último colar pode ser afectado por uma rotação de apenas 5 graus. Desta forma é possível suavizar a transição da malha para a sub-malha.

É o utilizador que controla a extensão da deformação especificando *a priori* o nível de propagação. No entanto, é sempre possível confinar a deformação a uma sub-malha de modo a que nada aconteça para além da sua fronteira, ou seja, o nível de propagação é igual a zero.

Considere-se que n é o nível de propagação. Se $n = 0$, então não existirá propagação da deformação para além do(s) vértice(s) da fronteira da sub-malha, ou seja, a deformação afectará apenas a sub-malha. Caso contrário, a deformação é propagada por n colares de faces da malha que contém a sub-malha a editar, de modo a minimizar o efeito da deformação na zona de fronteira.

Por exemplo, o nível de propagação na deformação da malhas da cabeça do coelho na Figura 5.10(a) é igual a 4, e é igual a 3 para o caso dos olhos na Figura 5.10(b). No caso da língua do coelho na Figura 5.10(b) esta parece fina porque não houve propagação da deformação, ou seja, o nível de propagação é igual a 0 e assim apenas os vértices seleccionados foram afectados. Neste caso não foi definida uma sub-malha, tendo sido apenas editado um conjunto de vértices em simultâneo. Na edição de um vértice ou conjunto de vértices pode também propagar-se a deformação aos vértices vizinhos, bastando para isso especificar o nível de propagação.



a) Suavização da malha do coelho. b) Suavização da sub-malha da cabeça.

Figura 5.11: Operador Laplaciano de suavização.

Pode aplicar-se também um operador de suavização da malha depois da sua edição, como é o caso do operador Laplaciano proposto por Taubin em [113]. Taubin usou este operador para suavizar uma malha como um todo, mas neste trabalho aplicou-se também este operador para a suavização de sub-malhas.

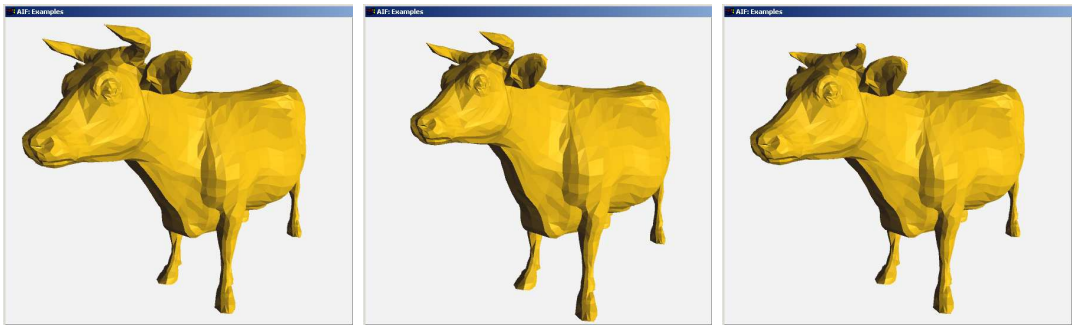
A Figura 5.11(a) mostra a aplicação do operador Laplaciano de suavização a toda a malha do coelho, enquanto que na Figura 5.11(b) a suavização só foi aplicada à sub-malha da cabeça. Esta operação de suavização pode ser considerada como mais uma operação de edição, pois é bastante útil para a suavização de malhas (ou sub-malhas) sempre que aparecem irregularidades na malha devido às operações de edição ou mesmo devido ao processo de refinamento no caso de malhas multiresolução.

Existem outros operadores de suavização que poderão também ser aplicados às sub-malhas como, por exemplo, o proposto por Lam *et al.* [59] que permite uma melhor preservação das arestas do que o operador Laplaciano ou, ainda, o proposto por Schneider e Kobbelt [93] que garante continuidade geométrica (G^1) ao longo da fronteira.

5.4 Edição Baseada em Operações Geométricas

A edição de malhas poligonais pode ser efectuada através de operações sobre vértices e operações sobre sub-malhas. No primeiro caso pode alterar-se a localização espacial de um vértice segundo uma direcção pré-definida pelo utilizador através de um vector tridimensional. A deformação pode afectar um vértice apenas ou um conjunto de vértices em simultâneo.

A Figura 5.10(a) mostra a deformação da malha do coelho por edição de um único



a) b) c)

Figura 5.12: Edição de sub-malhas com/sem sub-malhas.

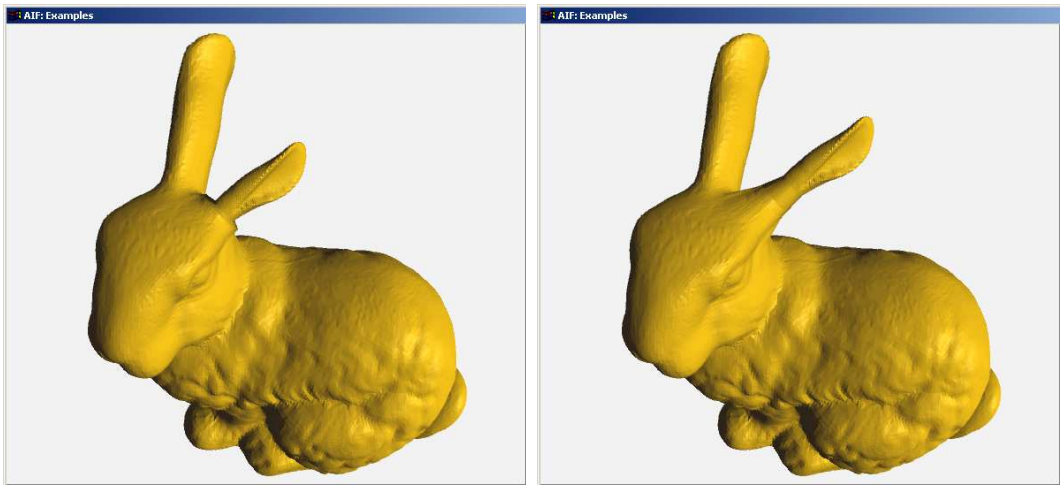
vértice, enquanto que na Figura 5.10(b) a deformação é aplicada a conjuntos de vértices, nomeadamente na zona dos olhos e da boca do coelho.

A deformação local depende não só da transformação aplicada aos vértices, ou seja, do deslocamento segundo a direcção estabelecida, mas também do nível de propagação imposto à deformação. Por exemplo, no caso da Figura 5.10(a) a deformação foi propagada por quatro colares. Já para a Figura 5.10(b), no caso da boca do coelho não houve propagação da deformação.

As transformações geométricas são normalmente aplicadas a um objecto como um todo. Neste trabalho estas transformações podem ser também aplicadas às sub-malhas individualmente, em particular operações como a variação de escala e a rotação [27, 122], ou ainda as suas variantes progressivas introduzidas por Barr [4]. A ideia subjacente às operações progressivas é a de alterar a transformação em cada ponto onde é aplicada. Este tipo de transformações foi adaptado com sucesso para a edição de sub-malhas.

A edição interactiva de sub-malhas através de transformações geométricas permite ao utilizador observar directamente o resultado de uma deformação. Diga-se que o efeito de uma transformação geométrica pode ser sempre desfeito por aplicação da operação inversa. Por exemplo, o efeito da rotação de uma sub-malha de 20 graus pode ser desfeito aplicando uma rotação de -20 graus. Do mesmo modo, uma variação de escala de 0.5 pode ser anulada com uma variação de escala de 2.0.

Como já foi sugerido anteriormente, a hierarquia de sub-malhas permite que o efeito da edição de uma malha seja aplicado também às suas sub-malhas. Por exemplo, a Figura 5.12(b) mostra a edição da sub-malha da cabeça da vaca, que foi afectada por uma variação de escala, a qual contém as sub-malhas dos cornos da vaca que sofreram a mesma transformação como seria de esperar. Já no caso da Figura 5.12(c) é feita a edição das sub-malhas dos cornos, a qual não afecta o resto da cabeça da vaca. Na Figura 5.12(a) pode ver-se a malha original.



a) b)

Figura 5.13: Variação de escala sem/com propagação da deformação.

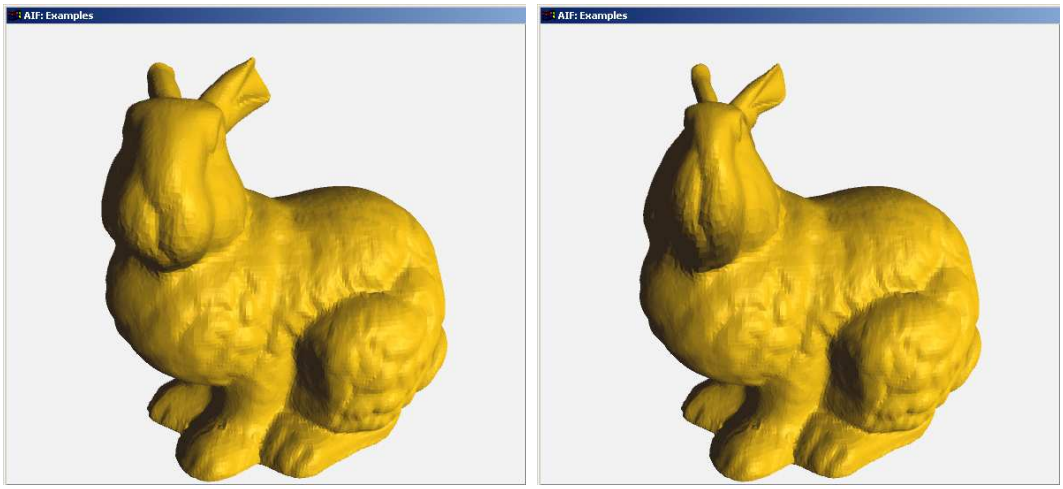
5.4.1 Transformações Geométricas Aplicadas a Sub-malhas

De seguida apresenta-se a edição de sub-malhas para as seguintes operações: variação de escala, rotação, variação de escala progressiva e rotação progressiva. No entanto, podem estender-se as operações de edição de sub-malhas a outros operadores como, por exemplo, a deformação por *lattices*[75] ou usando o operador *twister*[69].

Variação de escala. Esta operação permite alterar o tamanho de uma sub-malha. A variação de escala pode ser efectuada com propagação da deformação ou não. A propagação tende a suavizar o efeito da operação nos colares de faces junto à fronteira da sub-malha. No entanto, quando a variação de escala é efectuada sem propagação da deformação é necessário ajustar de forma automática a posição da sub-malha à sua fronteira. Esta necessidade advém do facto de os vértices da fronteira da sub-malha permanecerem na mesma posição enquanto que os vértices vizinhos que pertencem à sub-malha afastam-se da fronteira. Assim, a alteração de escala de uma sub-malha sem propagação (ou seja, com nível de propagação igual a zero) tem de ajustar a sua posição em relação ao plano que aproxima a sua fronteira na malha.

Esta operação é ilustrada na Figura 5.13 onde se vê o resultado da variação de escala aplicada à sub-malha da orelha esquerda do coelho. Na Figura 5.13(a) a variação de escala é efectuada sem propagação da deformação, onde o ajuste é feito no primeiro colar de faces da sub-malha, já na Figura 5.13(b) pode ver-se a mesma operação mas com propagação da deformação (nível de propagação igual a 3) onde o ajuste passou a ser feito não na sub-malha mas sim na malha.

Variação de escala progressiva. Esta operação é semelhante à variação de escala mas o factor de escala não é constante para todos os colares de vértices. No entanto,



a) b)

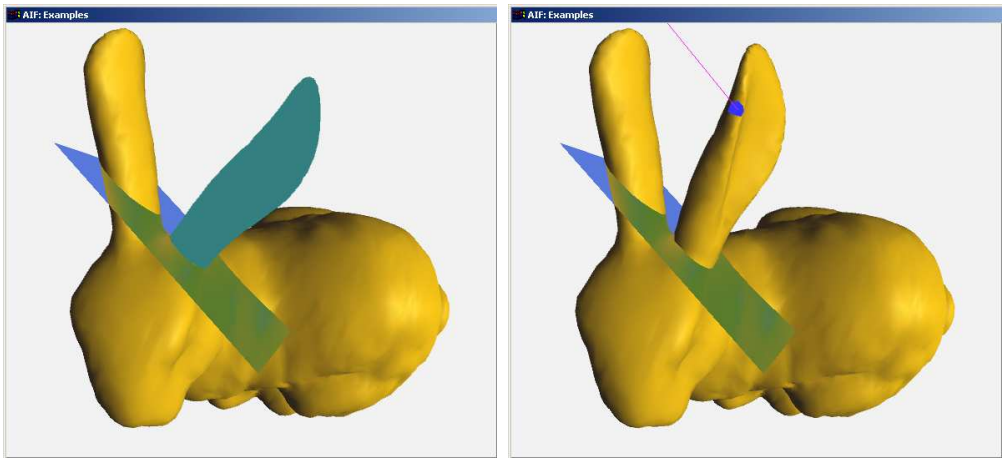
Figura 5.14: Variação de escala progressiva sem/com propagação da deformação.

pode aplicar-se esta operação apenas para alguns colares de vértices ao longo dos quais o factor de escala vai variando, mantendo o factor de escala constante para os restantes colares de vértices. Isto é, fez-se uma combinação da operação de variação de escala progressiva com a variação de escala simples. Assim, se o nível de propagação é zero o factor de escala varia progressivamente para cada colar de faces, ou seja, temos a variação de escala progressiva usual como se pode ver na Figura 5.14(a) onde foi efectuada a variação de escala para a sub-malha da cabeça do coelho. Caso contrário, o factor de escala é progressivo para os colares de faces até ao nível de propagação especificado, a partir do qual passa a ser constante como mostra a Figura 5.14(b) onde também foi feita a variação de escala da cabeça do coelho mas aqui com o nível de propagação igual a 20.

Como esta operação faz uma variação de escala progressiva não existe a necessidade de propagar a deformação à malha ou ajustar a sub-malha, pois o primeiro nível de vértices da sub-malha (ou seja, a sua fronteira) mantém-se na mesma posição e apenas os restantes níveis de vértices são afectados progressivamente.

Rotação. A rotação de uma sub-malha implica a especificação de um eixo de rotação e do valor do ângulo a rodar. O eixo de rotação é definido por um vector tridimensional, que é inicialmente definido como uma normal a um vértice seleccionado pelo utilizador que poderá depois ser ajustado interactivamente. Note-se que a normal ao vértice é calculada com base na média das normais das faces incidentes nesse vértice.

A rotação de uma sub-malha pode originar alguns efeitos indesejados, especialmente no colar fronteiriço entre a sub-malha e a malha. De facto, é necessário cuidados especiais na rotação de uma sub-malha de modo a prevenir as auto-intersecções ou sobreposições de geometria na zona de fronteira. Para evitar este problema existem duas soluções. A primeira solução é propagar a deformação da sub-malha por alguns



a) Sub-malha original. b) Sub-malha rodada.

Figura 5.15: Rotação usando um plano como restrição.

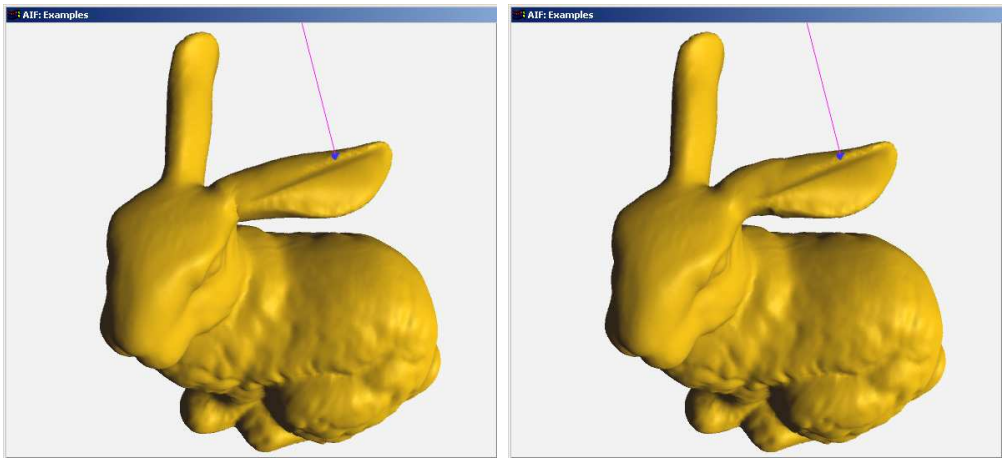
níveis à malha (sub-malha) que a contém. Neste caso, faz-se uma propagação progressiva para os colares de faces até ao nível de propagação especificada. No entanto, não se consegue eliminar por completo o aparecimento de deformações indesejadas sem redefinir localmente a malha.

Claro que quando se roda uma sub-malha em relação ao eixo que é perpendicular ao plano da sua fronteira, a deformação produzida na fronteira pode ser considerável para grandes ângulos de rotação. Mas não é possível corrigir a deformação sem alterar a topologia da malha localmente. No entanto, esta operação de alteração da topologia é mais simples para o caso de contornos regulares do que para o caso de contornos irregulares.

A segunda solução passa pela imposição de restrições ao ângulo de rotação. Por exemplo, através da utilização de um plano auxiliar, os vértices de um nível, acima ou abaixo do plano não poderão passar para o lado oposto desse plano. O plano inicial é definido pela fronteira da sub-malha evitando assim que o nível de vértices vizinhos da fronteira na sub-malha, que está acima do plano, possa passar para baixo deste plano. Esta estratégia pode aplicar-se sucessivamente aos níveis seguintes.

Por exemplo, a Figura 5.15 mostra a rotação da orelha esquerda do coelho (sub-malha da orelha) usando um plano como restrição, pois a geometria da sub-malha junto da fronteira têm tendência a criar auto-intersecções ou sobreposições. O plano serve de barreira a possíveis deformações indesejadas, não deixando que os vértices mudem de posição em relação ao plano. Portanto, um vértice acima ou abaixo do plano não pode atravessar o plano depois da rotação.

Rotação progressiva. A rotação progressiva (*twisting*) é uma rotação onde o ângulo de rotação não é constante, vai variando progressivamente para cada colar de vértices. Esta operação faz uma transição suave na fronteira da sub-malha com a malha que



a) b)

Figura 5.16: Rotação progressiva de uma sub-malha (orelha direita).

a contém como se pode ver na Figura 5.16. De modo similar à variação de escala progressiva, se o nível de propagação é zero, o ângulo de rotação varia progressivamente para todos os colares de faces (Figura 5.16(a)), ou seja, tem-se a rotação progressiva usual. Se o nível de propagação é diferente de zero então o ângulo de rotação é progressivo para os colares de faces até ao nível de propagação especificado, a partir do qual passa a ser constante como mostra a Figura 5.16(b) (nível de propagação igual a 10).

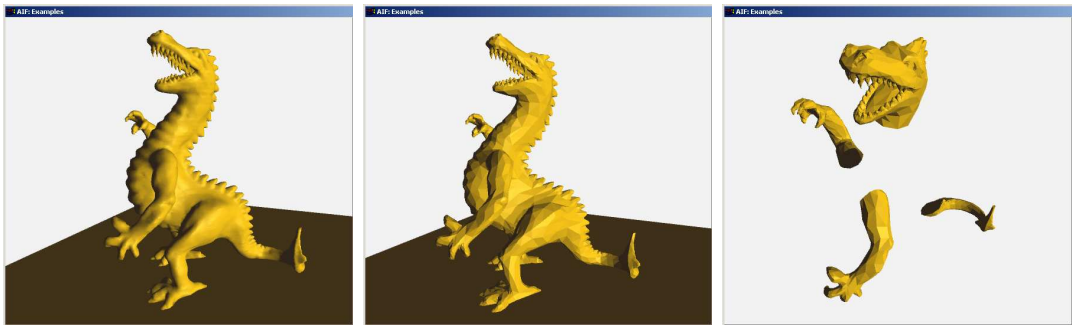
Similarmente à variação de escala progressiva, se o nível de propagação for diferente de zero, então temos uma combinação da rotação progressiva com a rotação. Isto é, uma rotação progressiva é aplicada aos colares de faces até ao nível de propagação especificado, enquanto que para os restantes colares de faces é aplicada a rotação.

Tanto no caso da rotação progressiva como no caso da variação de escala progressiva não existe deformação para além da fronteira da sub-malha.

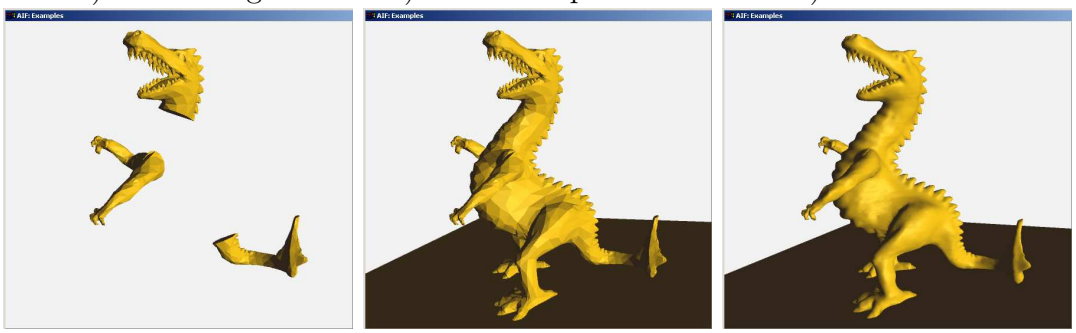
5.4.2 Edição de Malhas Multiresolução

A edição interactiva de uma malha nem sempre é fácil devido à grande densidade de polígonos. Contudo o recurso a malhas multiresolução pode contornar este problema visto que se pode editar uma malha simplificada e depois propagar a deformação à malha original. Assim a malha inicial é primeiro simplificada para um nível de resolução desejado, na qual o utilizador pode definir as sub-malhas necessárias à sua edição. Depois as sub-malhas são editadas aplicando as transformações desejadas, e finalmente a malha é refinada e suavizada se necessário.

A Figura 5.17 ilustra todo o processo de edição da malha multiresolução para o caso da malha do dinossauro. A sequência de imagens descreve: (a) a malha original; (b) a malha simplificada pelo algoritmo NSA descrito no Capítulo 4; (c) as sub-malhas



a) Malha original. b) Malha simplificada. c) Sub-malhas.



d) Sub-malhas editadas. e) Malha editada. f) Malha refinada.

Figura 5.17: Processo de edição de malhas multiresolução.

entretanto criadas; (d) as sub-malhas editadas através de operações de edição descritas em 5.4.1, nomeadamente: variação de escala da cabeça e cauda e rotação do membros superiores do dinossauro; (e) a malha simplificada e editada; e, finalmente, (f) a malha refinada já com as alterações feitas.

A Figura 5.18, no final deste capítulo, mostra mais alguns exemplos da edição de malhas multiresolução para outros modelos. O nariz de Venus foi deformado através da edição de alguns vértices, as orelhas do coelho foram rodadas progressivamente, a tromba e os chifres do elefante foram rodados e deformados por uma variação de escala progressiva, e no dragão foi reduzida a sua cabeça e a parte traseira recorrendo a uma variação de escala progressiva. Note que a malha do dragão possui mais de 2.5 milhões de células (vértices, arestas e faces), facto este que torna difícil a sua edição na sua resolução original. Como sugerem as imagens da Figura 5.18 não é difícil alterar a forma de uma malha localmente usando sub-malhas, tal como é necessário em muitas aplicações como, por exemplo, na animação ou nos jogos.

5.5 Sumário

Neste capítulo introduziu-se uma nova metodologia para a edição interactiva de malha poligonais, com ou sem multiresolução, através do uso de uma hierarquia de sub-malhas. A utilização directa de sub-malhas acaba por ser, talvez, uma das principais con-

tribuições desta tese no que se refere à edição interactiva de malhas poligonais.

A utilização de sub-malhas permite ao utilizador manipular e editar interactivamente regiões (ou sub-malhas) da malha sem alterar o resto da malha. A estas é possível depois aplicar as transformações geométricas descritas anteriormente. No entanto, nada impede a aplicação de outros tipos de transformações geométricas às sub-malhas.

A utilização de malhas multiresolução facilita a edição interactiva de malhas com grande número de células, pois permite simplificar, editar e refinar a malha. A grande vantagem é que a edição da malha pode ser feita numa versão simplificada, sendo depois propagados os efeitos da edição à malha original por refinamento, usando a hierarquia multiresolução.

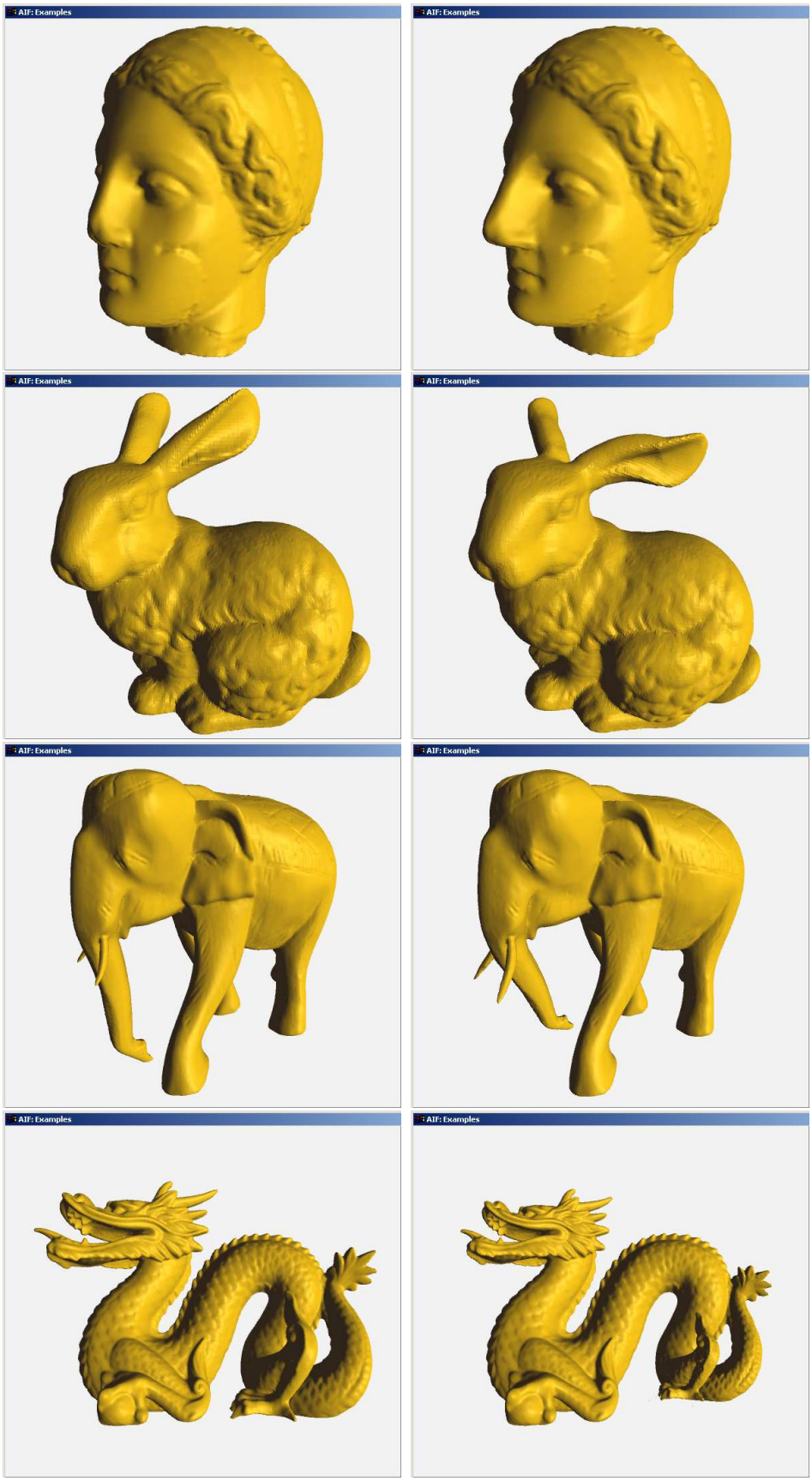


Figura 5.18: Edição de malhas multiresolução: Venus, coelho, elefante e dragão.

Capítulo 6

Conclusões e Trabalho Futuro

As conclusões mais relevantes do trabalho de investigação que conduziu à elaboração desta tese, assim como as perspectivas que daí resultam para trabalho futuro, são de seguida enunciadas.

6.1 Conclusões

As conclusões decorrentes do trabalho realizado no âmbito desta tese são aquelas que de algum modo já foram esboçadas nos Capítulos 3, 4 e 5.

No Capítulo 3 apresentou-se uma nova estrutura de dados, designada por AIF, capaz de representar malhas poligonais *manifold* e *non-manifold*, contrariamente a muitas outras estrutura de dados que permitem apenas representar malhas triangulares e *manifold*.

A estrutura de dados AIF é uma estrutura de dados não orientada, ou seja, não possui células orientadas. É por isso que é mais concisa do que as tradicionais estruturas de dados B-rep e do que muitas das estruturas de dados específicas para malhas. No entanto, apesar de não ser orientada, a estrutura de dados AIF possui um mecanismo de orientação por indução que permite a correcta visualização de uma malha. Além disso, a estrutura de dados AIF possui vértices, arestas e faces explicitamente representados, o que permite um acesso mais rápido a toda a informação sobre adjacências e incidências, contrariamente a muitas estruturas de dados que representam apenas uma ou duas destas entidades.

Outra contribuição importante é a introdução de um único operador para pesquisar *toda* a informação topológica na estrutura de dados AIF: o operador máscara. A estrutura de dados AIF e o operador máscara formam, pois, um núcleo geométrico surpreendentemente simples, mas capaz de descrever quer malhas quer objectos geométricos mais gerais. Por exemplo, este núcleo pode ser usado como um núcleo B-rep, sobre o qual a implementação dos respectivos operadores de Euler se afigura uma tarefa

trivial.

No Capítulo 4 apresentou-se um algoritmo de simplificação de malhas, designado por NSA, que usa a operação de contracção de arestas. Este algoritmo tem a particularidade de usar o mesmo critério para a escolha e validação da aresta a contrair, o que não acontece noutros algoritmos de simplificação.

O algoritmo NSA é o mais rápido dentro da sua classe conforme provam os resultados apresentados. Além disso, consegue reduzir o número de faces de uma malha até 49% numa única simplificação. No entanto, a qualidade da malha gerada pelo algoritmo NSA é ligeiramente inferior à do algoritmo QSlim para as últimas simplificações. A criação de malhas multiresolução é também baseada no algoritmo NSA que usa a operação de contracção de arestas em conjunção com a operação inversa, a subdivisão de vértices.

No Capítulo 5 apresentou-se uma nova metodologia para a edição interactiva de malhas poligonais, com ou sem multiresolução, usando o conceito de sub-malha. Esta metodologia permite editar uma sub-malha (uma região da malha) independentemente do resto da malha através da aplicação de transformações geométricas. Esta metodologia constitui pois, uma contribuição importante no que se refere à edição interactiva de malhas, que pode muito bem ser integrada em sistemas de animação computacional. Note-se que esta metodologia é também aplicável a malhas multiresolução, o que permite editar malhas com grande número de células. Para isso, a malha é inicialmente simplificada para uma resolução adequada, editada e depois refinada para a sua resolução original.

6.2 Trabalho Futuro

A breve trecho, há a intenção de disponibilizar gratuitamente na Internet o núcleo geométrico baseado na estrutura de dados AIF de forma a disseminar a sua utilização e divulgação. O núcleo geométrico deverá incluir os operadores de Euler, bem como um conjunto de primitivas gráficas estandardizadas, podendo assim servir de base ao desenvolvimento de aplicações de modelação, edição, visualização, etc.

Com base na estrutura de dados AIF seria ainda importante uma análise comparativa com outras estruturas de dados por forma a quantificar a influência de uma estrutura de dados no desempenho de um algoritmo. Isto é, usar um ou mais algoritmos para serem codificados igualmente em diferentes estruturas de dados, incluindo a estrutura de dados AIF, de modo a analisar o desempenho das estruturas de dados. O que acontece muitas vezes é que os resultados dos algoritmos são muitas vezes comparados mesmo quando codificados em diferentes estruturas de dados, e assim não é possível saber qual a verdadeira influência da estrutura de dados no desempenho de um algoritmo.

Na edição de malhas poligonais pretende-se desenvolver ainda novas operações de edição com sub-malhas, nomeadamente edição com *lattices* [75] e edição com *wires*

[106]. Além disso, é necessário também desenvolver mais mecanismos de prevenção das distorções e auto-intersecções da malha devidas às operações de edição como, por exemplo, o que foi apresentado para o caso da rotação de sub-malhas.

No que se refere ao esquema de multiresolução desenvolvido, e que usa o algoritmo de simplificação de malhas NSA, o que se pretende desenvolver no futuro é um protótipo de um sistema de animação que possa integrar o esquema multiresolução para a representação dos objectos para diferentes níveis de detalhe. Isto é, tentar usar o esquema de multiresolução para criar os diferentes níveis de detalhe para os objectos em cena consoante a sua distância ao observador virtual, em vez de usar modelos LOD como é usual.

Bibliografia

[1] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, and Claudio T. Silva. Computing and rendering point set surfaces,. *IEEE Transaction on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[2] Nicolas Aspert, Diego Santa-Cruz, and Touradj Ebrahimi. Mesh: Measuring errors between surfaces using the hausdorff distance. In *Proceedings of the IEEE International Conference in Multimedia and Expo (ICME)*, pages 705–708, 2002.

[3] M. Barlaud. *Wavelets in Image Communication*. Elsevier, 1994.

[4] Alan H. Barr. Global and local deformations of solid primitives. In *Proceedings of Siggraph*, volume 18(30), pages 21–30. ACM Press, 1984.

[5] B. G. Baumgart. Winged-edge polyhedron representation. Technical report, STAN-CS-320, Stanford University, 1972.

[6] Henning Biermann, Ioana Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics*, 21(3):312–321, 2002.

[7] Andrew Blake and Michael Isard. *Active Contours*. Springer-Verlag, 1999.

[8] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt. Openmesh – a generic and efficient polygon mesh data structure. In *Proceedings of OpenSG Symposium*, 2002.

[9] Mario Botsch and Leif Kobbelt. Multiresolution surface representation based on displacement volumes. In *Proceedings of Eurographics*, pages 483–491, 2003.

[10] Mario Botsch and Leif Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics (TOG)*, 23(3):630–634, 2004.

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

[12] Dmitry Brodsky and Benjamin Watson. Model simplification through refinement. In *Proceedings of Graphics Interface*, pages 221–228, 2000.

[13] C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms - A Primer*. Prentice Hall, 1998.

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

[15] E. Catmull and J Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10:350–355, 1978.

[16] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, 22(1):37–54, 1998.

[17] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[18] Paolo Cignoni, Leila De Floriani, Peter Lindstrom, Valerio Pascucci, Jarek Rossignac, and Claudio Silva. Multi-resolution modeling, visualization and streaming of volume meshes. In *Eurographics - Tutorial #2*, pages 1–116. INRIA and the Eurographics Association, ISSN 1017-4656, 2004.

[19] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. Simplification envelopes. In *Proceedings of Siggraph*, pages 119–128. ACM, 1996.

[20] Neil A. Dodgson, Michael S. Floater, and Malcolm A. Sabin. *Advances in Multiresolution for Geometric Modelling*. Springer-Verlag, 2004.

[21] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.

[22] Nira Dyn, David Levin, and John A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, 1990.

[23] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of Siggraph*, volume 29, pages 173–182. ACM Press, 1995.

[24] Jihad El-Sana and Amitabh Varshney. Generalized view-dependent simplification. In *Proceedings of Eurographics*, volume 18(3), pages 83–94, 1999.

[25] Leila De Floriani, Leif Kobbelt, and Enrico Puppo. A survey on data structures for level-of-detail models. In *Advanced in Multiresolution for Geometric Modelling*, pages 49–74. Dodgson, Floater, Sabin (Eds.), Springer, 2004.

[26] Leila De Floriani, Paola Magillo, Enrico Puppo, and Davide Sobrero. A multi-resolution topological representation for non-manifold meshes. In *Proceedings of the seventh ACM symposium on Solid Modeling and Applications*, pages 159–170, 2002.

[27] James D. Foley, Andries van Dam, and John F. Hughes Steve K. Feiner. *Computer Graphics: Principles and Pratices.* Addison-Wesley Publishing Company, 1996.

[28] Alain Fournier. Wavelets and their applications in computer graphics. In *Siggraph Course Notes*, 1995.

[29] Michael Garland. Multiresolution modeling: Survey & future opportunities. In *Eurographics - State of the Art Reports*, pages 111–131. ACM, 1999.

[30] Michael Garland. *Quadric-Based Polygonal Surface Simplification.* PhD thesis, Carnegie Mellon University, Pittsburgh, May 1999.

[31] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceeding of Siggraph*, pages 209–216. ACM Press, 1997.

[32] Michael Garland and Paul S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of IEEE Visualization*, 1998.

[33] Craig Gotsman, Stefan Gumhold, and Leif Kobbelt. Simplification and compression of 3d meshes. In *Tutorials on Multiresolution in Geometric Modelling - (Munich Summer School Lecture Notes)*, pages 319–361. A. Iske, E. Quak, M. Floater (Eds.), Springer, 2002.

[34] André Guéziec. Surface simplification with variable tolerance. In *Second Annual Intl. Symp. on Medical Robotics and Computer Assisted Surgery (MRCAS '95)*, pages 132–139, 1995.

[35] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution signal processing for meshes. In *Proceedings of Siggraph*, pages 325–334. ACM Press, 1999.

[36] Igor Guskov, Kiril Vidimce, Wim Sweldens, and Peter Schröder. Normal meshes. In *Proceedings of Siggraph*, pages 95–102. ACM Press, 2000.

[37] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. In *Siggraph - Course Notes 25.* ACM Press, 1997.

[38] Hugues Hoppe. Progressive meshes. In *Proceeding of Siggraph*, pages 99–108. ACM Press, 1996.

[39] Hugues Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceeding of IEEE Visualization*, pages 35–42, 1998.

[40] Hugues Hoppe. View-dependent refinement of progressive meshes. Technical report, MSR-TR-98-02, Microsoft Research, 1998.

[41] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Proceedings of Siggraph*, pages 19–26. ACM Press, 1993.

[42] Josef Hoschek and Dieter Lasser. *Fundamentals of Computer Aided Geometric Design*. A K Peters, Wellesley, Massachusetts, 1993.

[43] William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form deformations. *Computer Graphics*, 26(2):177–184, 1992.

[44] Andreas Hubeli and Markus Gross. A survey of surface representations for geometric modeling. Technical report, #335, Swiss Federal Institute of Technology Zurich, 2000.

[45] Martin Isenburg and Stefan Gumhold. Out-of-core compression for gigantic polygon meshes. In *Proceedings of Siggraph*, pages 935–942. ACM Press, 2003.

[46] Martin Isenburg, Peter Lindstrom, Stefan Gumhold, and Jack Snoeyink. Large mesh simplification using processing sequences. In *Proceedings of IEEE Visualization*, pages 465–472, 2003.

[47] JPEG. Joint photographic experts group. http://www.jpeg.org/.

[48] Gerald Kaiser. *A Friendly Guide to Wavelets*. Birkhauser, 1994.

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

[50] Takashi Kanai, Hiromasa Suzuki, Jun Mitani, and Fumihiko Kimura. Interactive mesh fusion based on local 3d metamorphosis. In *Proceedings of Graphics Interface*, pages 148–156, 1999.

[51] Youngihn Kho and Michael Garland. User-guided simplification. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, pages 123–126. ACM Press, 2003.

[52] Junho Kim and Seungyong Lee. Truly selective refinement of progressive meshes. In *Proceedings of Graphics Interface*, pages 101–110, 2001.

[53] Reinhard Klein, Gunther Liebich, and Wolfgang Straßer. Mesh reduction with error control. In *Proccedings of IEEE Visualization*, pages 311–318. IEEE Computer Society, 1996.

[54] Leif Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. *Computer Graphics Forum*, 15(3):409–420, 1996.

[55] Leif Kobbelt, Thilo Bareuther, and Hans-Peter Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. In *Proceeding of Eurographics*, volume 19(3), pages C249–C260, 2000.

[56] Leif Kobbelt, Swen Campagna, and Hans-Peter Seidel. A general framework for mesh decimation. In *Graphics Interface*, pages 43–50, 1998.

[57] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of Siggraph*, volume 32, pages 105–114. ACM Press, 1998.

[58] Leif Kobbelt, Jens Vorsatz, and Hans-Peter Seidel. Multiresolution hierarchies on unstructured triangle meshes. *Computational Geometry Journal: Theory and Applications*, 14:5–24, 1999.

[59] Roberto Lam, Robert Loke, and Hans du Buf. Alisamento e dizimação de malhas triangulares. In *Actas do 10º Encontro Português de Computação Gráfica*, pages 1–9, 2001.

[60] Francis Lazarus, Sabine Coquillart, and Pierre Jancène. Interactive axial deformations. Rapport de recherche nº 1891, INRIA, 1993.

[61] Aaron W. F. Lee, David Dobkin, Wim Sweldens, and Peter Schröder. Multiresolution mesh morphing. In *Proceedings of Siggraph*, pages 343–350. ACM Press, 1999.

[62] Sang Hun Lee and Kunwoo Lee. Partial entity structure: A fast and compact non-manifold boundary representation based on partial topological entities. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, pages 159–170, 2001.

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

[64] Yunjin Lee and Seungyong Lee. Geometric snakes for triangular meshes. In *Computer Graphics Forum*, volume 21(3), pages 229–238, 2002.

[65] Peter Lindstrom. Out-of-core simplification of large polygonal models. In *Proceedings of Siggraph*, pages 259–262. ACM Press, 2000.

[66] Peter Lindstrom. Out-of-core construction and visualization of multiresolution surfaces. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, pages 93–102, 2003.

[67] Peter Lindstrom and Cláudio T. Silva. A memory insensitive technique for large model simplification. In *Proceedings of IEEE Visualization*, pages 121–126. IEEE Computer Society, 2001.

[68] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *Proceedings of IEEE Visualization*, pages 279–286. IEEE Computer Society, 1998.

[69] Ignacio Llamas, Byungmoon Kim, Joshua Gargus, Jarek Rossignac, and Chris D. Shaw. Twister: a space-warp operator for the two-handed editing of 3d shapes. *ACM Transactions on Graphics*, 22(3):663–668, 2003.

[70] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Utah University, 1987.

[71] Charles Loop. Managing adjacency in triangular meshes. Technical Report No. MSR-TR-2000-24, Microsoft Research, 2000.

[72] Michael Lounsbery, Tony D. DeRose, and Joe Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, 1997.

[73] David Luebke, Martin Reddy, Jonathan D. Cohen, Amitabh Varshney, Benjamin Watson, and Robert Huebner. *Level Of Detail For 3D Graphics*. Morgan Kaufmann, 2002.

[74] David P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001.

[75] Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of Siggraph*, pages 181–188. ACM Press, 1996.

[76] Mohsen Madi and Desmond Walton. Interactive selection and modification of polyhedral 3d objects. In *Proceedings of the International Conference on Computer Vision and Graphics*, volume 2, pages 510–517, Poland, 2002.

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

[78] D. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.

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

[80] Renato Pajarola. Fastmesh: Efficient view-dependent meshing. In *Proceedings of Pacific Graphics*, pages 22–30. ACM Press, 2001.

[81] Renato Pajarolai and Christopher DeCoro. Efficient implementation of real-time view-dependent multiresolution meshing. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):353–368, 2004.

[82] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled models. In *Computer Graphics Forum*, volume 22(3), pages 281–289, 2003.

[83] Mark Pauly, Richard Keiser, Leif Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. In *Proceedings of Siggraph*, pages 641–650. ACM Press, 2003.

[84] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surface elements as rendering primitives. In *Proceedings of Siggraph*, pages 335–342. ACM Press, 2000.

[85] Jovan Popovic and Hugues Hoppe. Progressive simplicial complexes. In *Proceedings of Siggraph*, pages 217–224. ACM Press, 1997.

[86] Chris Prince. Progressive meshes for large models of arbitrary topology. Master's thesis, University of Washington, 2000.

[87] Enrico Puppo and Roberto Scopigno. Simplification, LOD and multiresolution - principles and applications. In *Eurographics Tutorial Notes*, 1997.

[88] A.A.G. Requicha. Representation for rigid solids: theory, methods and systems. *ACM Computing Surveys*, 12(4):437–464, 1980.

[89] Jarek Rossignac and Paul Borrel. Multi-resolution 3D approximations for rendering complex scenes. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 455–465, 1993.

[90] Michael Roy, Frédéric Nicolier, Sebti Foufou, Frédéric Truchetet, Andreas Koschan, and Mongi Abidi. Assessment of mesh simplification algorithm quality. In *Proceedings of SPIE Electronic Imaging*, volume 4661, pages 128–137, 2002.

[91] Szymon Rusinkiewicz and Marc Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of Siggraph*, pages 343–352. ACM Press, 2000.

[92] Malcolm Sabin. Recent progress in subdivision: a survey. In *Advanced in Multiresolution for Geometric Modelling*, pages 203–230. Dodgson, Floater, Sabin (Eds.), Springer, 2004.

[93] Robert Schneider and Leif Kobbelt. Geometric fairing of irregular meshes for free-form surface design. *Computer Aided Geometric Design*, 18(4):359–379, 2001.

[94] Peter Schröder and Wim Sweldens. Wavelets in computer graphics. In *Siggraph Course Notes*. ACM Press, 1996.

[95] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In *Proceedings of Siggraph*, pages 65–70. ACM Press, 1992.

[96] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *Proceedings of Siggraph*, pages 151–160. ACM Press, 1986.

[97] Thomas W. Sederberg, Jianmin Zheng, David Sewell, and Malcolm Sabin. Non-uniform recursive subdivision surfaces. In *Proceedings of Siggraph*, pages 387–394. ACM Press, 1998.

[98] Claudio T. Silva, Yi-Jen Chiang, Jihad El-Sana, and Peter Lindstrom. Out-of-core algorithms for scientific visualization and computer graphics. In *Proceedings of IEEE Visualization, Tutorial #4*. IEEE Society Press, 2002.

[99] Frutuoso G. M. Silva and Abel J. P. Gomes. Adjacency and incidence framework - a data structure for efficient and fast management of multiresolution meshes. In *Proceedings of International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE'03)*, pages 159–166, Melbourne, Australia, 2003. ACM Press.

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

[101] Frutuoso G. M. Silva and Abel J. P. Gomes. Aif - uma estrutura de dados b-rep para modelos poligonais. In *Actas do 12º Encontro Português de Computação Gráfica (EPCG'03)*, pages 1–7, 2003.

[102] Frutuoso G. M. Silva and Abel J. P. Gomes. Interactive editing of arbitrary sub-meshes. In *Proceedings of Computer Graphice International (CGI'03)*, pages 218–221. IEEE Computer Society Press, 2003.

[103] Frutuoso G. M. Silva and Abel J. P. Gomes. A b-rep data structure for polygonal meshes. *VIRTUAL - The Portuguese Journal of Computer Graphics*, Special edition - Advances in Computer Graphics in Portugal, Adérito Marcos and Miguel Salles Dias (eds.), Novembro 2004.

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

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

[106] Karan Singh and Eugene Fiume. Wires: A geometric deformation technique. In *Proceedings of Siggraph*, pages 405–414. ACM Press, 1998.

[107] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 179–188. Eurographics Association, 2004.

[108] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for computer graphics: A primer part 1. *IEEE Computer Graphics and Applications*, 15(3):76–84, 1995.

[109] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. Wavelets for computer graphics: A primer part 2. *IEEE Computer Graphics and Applications*, 15(4):75–85, 1995.

[110] Eric J. Stollnitz, Tony D. Derose, David H. Salesin, and Anthony D. DeRose. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, 1996.

[111] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):399–405, 2004.

[112] Hiromasa Suzuki, Yusuke Sakurai, Takashi Kanai, and Fumihiko Kimura. Interactive mesh dragging with an adaptive remeshing technique. *The Visual Computer*, 16(3/4):159–176, 2000.

[113] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of Siggraph*, pages 351–358. ACM Press, 1995.

[114] Gabriel Taubin and Jarek Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2):84–115, 1998.

[115] Ireneusz Tobor, Christophe Schlick, and Laurent Grisoni. Rendering by surfels. In *Proceeding of GRAPHICON*, pages 193–204, 2000.

[116] Costa Touma and Craig Gotsman. Triangle mesh compression. In Kellogg Booth W. Davis and Alain Fournier, editors, *Proceedings of Graphics Interface*, pages 26–34. Morgan Haufmann Publishers, 1998.

[117] Sébastien Valette and Rémy Prost. Wavelet-based multiresolution analysis of irregular surface. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):113–122, 2004.

[118] Sébastien Valette and Rémy Prost. Wavelet-based progressive compression scheme for triangular meshes: Wavemesh. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):123–129, 2004.

[119] Oliver Matias van Kaick, Murilo Vicente Gonçalves da Silva, and Hélio Pedrini. Efficient generation of triangle strips from triangulated meshes. In *Proceedings of WSCG*, volume 12(1-3), pages 475–481, 2004.

[120] VRML. The virtual reality modeling language (vrml) - iso/iec 14772, 1997. http://www.web3d.org/x3d/specifications/vrml/ISO_IEC_14772-All/.

[121] Joe Warren and Henrik Weimer. *Subdivision Methods For Geometric Design - A constructive approach*. Morgan Kaufman, 2002.

[122] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques - Theory and Practice*. Addison-Wesley, 1992.

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

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

[125] Xinyu Xiang, Martin Held, and Joseph S. B. Mitchell. Fast and effective stripification of poligonal surface models. In *ACM Symposium on Interactive 3D Graphics*, pages 71–78. ACM Press, 1999.

[126] Jingqi Yan, Pengfei Shi, and David Zhang. Mesh simplification with hierarchical shape analysis and iterative edge contraction. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):142–151, 2004.

[127] Steve Zelinka and Michael Garland. Permission grids: Pratical, error-bounded simplification. *ACM Transaction on Graphics*, 21(2):207–229, 2002.

[128] Denis Zorin and Peter Schröder. Subdivision for modeling and animation. In *Siggraph Course Notes*. ACM Press, 2000.

[129] Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. In *Proceedings of Siggraph*, pages 259–268. ACM Press, 1997.

[130] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3d: An interactive system for point-based surface editing. In *Proceedings of Siggraph*, pages 322–329. ACM Press, 2002.