UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Um Algoritmo para Interpolação de Formas entre Objetos Modelados por Superfícies Spline

por

Manuel Menezes de Oliveira Neto

Dissertação submetida como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Anatólio Laschuk
Orientador

Porto Alegre, Fevereiro de 1992.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Oliveira Neto, Manuel Menezes de

Um Algoritmo para Interpolação de Formas entre Objetos Modelados por Superfícies Spline / Manuel Menezes de Oliveira Neto.—Porto Alegre: CPGCC da UFRGS, 1992.

127 p.: il.

Dissertação (mestrado)—Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1992. Orientador: Laschuk, Anatólio

Dissertação: Computação Gráfica
Interpolação de Formas, Animação Modelada, Superfícies Spline

Dedico este trabalho a três filósofos:

Minha mãe, a quem devo tudo. Dela recebi as mais nobres lições de amor, inteligência, humildade, determinação e fortaleza;

Meu pai, cuja inteligência e determinação estão sempre presentes, embora tenhamos nos separado ainda nos primeiros instantes da caminhada;

Ninha, pessoa de amor infinito. Ela me ensinou a contar.

Eles sim, merecem títulos de mestres.

Dois *pixels* distintos determinam exatamente uma reta; três *voxels*, um plano.

AGRADECIMENTOS

Gostaria de agradecer a todas as pessoas que, de algum modo, contribuíram para a realização deste trabalho. De maneira especial, ao professor Laschuk, pela confiança, orientação e pela habilidade com que, ensinando sobre curvas e supefícies, me apresentou uma conjunção perfeita: Computação Gráfica - profissão e *hobby*.

Também gostaria de expressar meus agradecimentos ao professor Marcos Sebastiani pelas valiosas sugestões à formalização dos resultados; à minha irmã Ana e ao professor Elian Machado, pelo incentivo e pelo exemplo; ao amigo Rui Bastos, pela parceria extraordinária; aos amigos Tadeu, George, Gladimir, Marcelow, McShadow e Kitajima, pelo apoio e estímulos de sempre; ao Casacurta, pela crítica implacável que me levou a rever as primeiras soluções, e ao Osório, pelo grande auxílio na digitalização de algumas das melhores imagens deste trabalho.

"O que é escrito, ordenado, factual nunca é suficiente para abarcar toda a verdade: a vida sempre transborda de qualquer cálice."

Boris Pasternak

SUMÁRIO

LISTA DE FIGURAS **13**
LISTA DE TABELAS **14**
LISTA DE PROCEDIMENTOS **14**
GLOSSÁRIO **16**
RESUMO **18**
ABSTRACT **20**

1 INTRODUÇÃO **22**
1.1 O estado da arte 23
1.2 Estética e formalismo 24
1.3 Motivação e objetivos 24
1.4 Estrutura da dissertação 25

2 SUBDIVISÃO DE SUPERFÍCIES BÉZIER **27**
2.1 Introdução 27
2.2 Curvas e superfícies Bézier 27
2.3 O Algoritmo de *de Casteljau* 28
2.3.1 Subdivisão de superfícies Bézier 31
2.4 Resumo 32

3 O ALGORITMO INTERPOLADOR DE FORMAS **33**
3.1 Introdução 33
3.2 Alguns conceitos topológicos 33
3.2.1 Espaços métricos, espaços conexos e conjuntos fechados 33
3.2.2 Homotopia, homeomorfismo e homeotopia 35
3.2.3 Decomposição celular e subdivisão celular 37
3.2.4 Relações de incidência e representações isomorfas 38
3.3 O algoritmo ideal 39
3.4 Uma versão topológica para o algoritmo ideal 40

3.4.1 M e M' homeomorfos 41
3.4.2 M e M' não homeomorfos 41
3.5 Uma versão geométrica para o algoritmo ideal 43
3.5.1 Uma representação padrão 44
3.5.2 A função interpoladora 45
3.5.3 A versão geométrica propriamente dita 47
3.5.3.1 Identificação das relações de incidência 48
3.5.3.2 Equacionamento do número de *patches* 62
3.5.3.3 Escolha dos *centros de malha* 64
3.5.3.4 Equacionamento do número de linhas 65
3.5.3.5 Equacionamento do número de colunas 66
3.5.3.6 Casos especiais 71
3.5.3.7 Regra para subdivisão de arestas 72
3.5.3.8 Estabelecimento das correspondências entre pares de *patches* 73
3.5.4 Ordem de complexidade do algoritmo 75
3.5.4.1 Conversão dos objetos para a representação padrão 75
3.5.4.2 Identificação das relações de incidência 76
3.5.4.3 Equacionamento do número de *patches* 76
3.5.4.4 Estabelecimento de correspondências entre pares de *patches* 76
3.5.4.5 Estabelecimento de correspondências entre coeficientes geométricos . 77
3.5.4.6 Custo total do algoritmo 77
3.5.5 Visualização de transformações descontínuas 77
3.6 Resumo 77

4 INTERPOL3D **80**
4.1 Introdução 80
4.2 Características 80
4.3 Estrutura do protótipo 81
4.3.1 Universo de uma animação 82
4.3.2 Interface para especificação do número de quadros 82

4.3.3 Interface para associação dos pares de objetos a interpolar 83
4.3.4 Interface para controle de velocidade 85
4.3.5 Cálculo das posições 87
4.4 Obtenção dos valores de ΔS e Δq 90
4.4.1 Obtenção do polinômio a partir dos coeficientes geométricos 90
4.4.1.1 Cálculo da área sob a curva (ΔS) 92
4.4.2 Cálculo da área do elemento de área Δq 94
4.5 Resultados 95
4.6 Resumo 98

5 ANÁLISE DOS RESULTADOS **99**
5.1 Introdução 99
5.2 O algoritmo Super-Construtor 99
5.3 Interpolação de formas no sistema SARA 100
5.4 O Algoritmo Interpolador de Formas x as técnicas analisadas 102
5.5 A modelagem dos objetos e os tempos de interpolação 104
5.6 Resumo 107

6 CONCLUSÕES **109**
6.1 O Algoritmo Interpolador de Formas 110
6.2 Estruturas de dados e procedimentos 111
6.3 Protótipo 112
6.4 Resultados 112
6.5 Possíveis extensões ao trabalho 113
6.5.1 Uso do **AIF** com superfícies racionais 113
6.5.2 Utilização de subdivisões adaptativas 113
6.5.3 Utilização de texturas 115

ANEXO A-1 ANIMAÇÃO POR COMPUTADOR **117**
A-1.1 O conceito de animação 117
A-1.2 O que é animação por computador 117

A-1.3 Classificações dos sistemas de animação 119
A-1.3.1 Complexidade e poder de expressão dos sistemas de animação . . . 119
A-1.3.2 Modo de produção de seqüências animadas 120
A-1.3.3 Animação por quadros-chaves x animação modelada 120
A-1.4 Linguagens para animação 121
A-1.5 Métodos de controle de animação 122
A-1.6 Classificação do protótipo INTERPOL3D segundo os critérios apresentados 123
BIBLIOGRAFIA **124**

LISTA DE FIGURAS

Figura 1.1 Interpolação de forma entre uma *esfera* e uma rainha de xadrez. 23

Figura 2.1 Curva Bézier. 28

Figura 2.2 Superfície Bézier. 29

Figura 2.3 Algoritmo de *de Casteljau*. 30

Figura 2.4 Esquema de *de Casteljau*. 30

Figura 2.5 *Patch* Bézier bicúbico (a) antes da subdivisão; (b) após subdividi-lo em relação à dimensão paramétrica v. 32

Figura 3.1 Espaços: (a) conexo (b) não conexo. 34

Figura 3.2 Caminho $f : [0, 1] \rightarrow M$. 34

Figura 3.3 Conjuntos: (a) fechado (b) aberto. 35

Figura 3.4 Homotopia: superfície plana $\rightarrow$ superfície cilíndrica. 36

Figura 3.5 Homeomorfismo e homeotopia (esfera e cubo) 36

Figura 3.6 Espaços homeomorfos não homeotópicos no R^3: (a) toro; (b) toro anelado. 37

Figura 3.7 Decomposição celular: (a) superfície cilíndrica; (b) subdivisão em quadriláteros; (c) decomposição celular; (d) reconstrução da superfície 37

Figura 3.8 Subdivisão celular da superfície 3.4 (c) 38

Figura 3.9 Decomposições celulares isomorfas 39

Figura 3.10 Subdivisão de um espaço conexo (plano delimitado) em *linhas* e *colunas*. 41

Figura 3.11 Decomposições celulares de duas superfícies cilíndricas. 42

Figura 3.12 Representações isomorfas para M e M'. 42

Figura 3.13 Decomposições celulares: (a) superfície cilíndrica; (b) toro. . . . 43

Figura 3.14 Equacionamento do número de células. 44

Figura 3.15 *Patches* bicúbicos: (a) não degenerado; (b) com o bordo direito degenerado em um ponto; (c) degenerado em uma linha; (d) degenerado em um ponto. 47

Figura 3.16 (a) superfície cilíndrica modelada por 4 *patches*; (b) decomposição celular corespondente. As setas representam as vizinhanças entre os *patches*. 48

Figura 3.17 Árvore binária de pesquisa 3D. 49

Figura 3.18 Abordagem logarítmica *versus* abordagem quadrática. 50

Figura 3.19 Tipos Objeto_log e *Patch_x, Patch_y, Patch_z* e Info_patch. . . . 53

Figura 3.20 *Patches* representados em uma árvore binária de pesquisa 3D. . 54

Figura 3.21 Estrutura de dados para a versão logarítmica: lista encadeada com estrutura recursiva de aninhamento associada a árvores binárias de pesquisa 3D. 55

Figura 3.22 Tipos Objeto_q e *Patch_q*. 60

Figura 3.23 Estrutura de dados para a versão quadrática: lista encadeada com estrutura recursiva de aninhamento. 61

Figura 3.24 Representação da figura 3.8 sem rotulação de arestas. 63

Figura 3.25 Equacionamento do número de *patches* para as representações da figura 3.24. 63

Figura 3.26 Equacionamento do número de linhas. Descrição dos objetos: (a) antes; (b) durante e (c) após. 65

Figura 3.27 Equacionamento do número de colunas. Descrição dos objetos (a) antes; (b) durante e (c) após. 66

Figura 3.28 Sentido de realização das subdivisões. 70

Figura 3.29 Subdivisões realizadas pelo procedimento 3.16 para $c = 3$, $nc = 5$. 70

Figura 3.30 Exemplo de caso especial. 71

Figura 3.31 Caso especial: as áreas sombreadas correspondem a discos que foram extraídos das superfícies cilíndricas originais. 71

Figura 3.32 Equacionamento por inclusão de *patches* degenerados. 72

Figura 3.33 Decomposições celulares de uma superfície cilíndrica e de um toro: (a) antes e (b) depois do equacionamento do número de *patches*. 73

Figura 4.1 Transformação de um peão em lâmpada. 81

Figura 4.2 Estrutura do protótipo INTERPOL3D. 82

Figura 4.3 Universo de uma animação INTERPOL3D. 83

Figura 4.4 Interface para especificação do número de quadros. 84

Figura 4.5 Interface para estabelecimento de correspondências 85

Figura 4.6 Interface para definição do comportamento da velociade de interpolação. 86

Figura 4.7 Áreas correspondentes aos segmentos da curva Bézier. 88

Figura 4.8 Delimitação das áreas correspondentes a cada quadro. 89

Figura 4.9 **M.A.T.E.** (a) Interpolação de uma esfera em rainha: esfera. . . 96

Figura 4.10 **M.A.T.E.** (b) 33% da transformação realizada. 96

Figura 4.11 **M.A.T.E.** (c) 66% da transformação realizada. 97

Figura 4.12 **M.A.T.E.** (d) 100% da transformação realizada. 97

Figura 5.1 Transfomação entre um paralelepípedo e um obelisco. Extraído de [BET 89]. 100

Figura 5.2 Interpolação de um octaedro em um icosaedro. Extraído de [BET 89]. 100

Figura 5.3 Pré-processamento das arestas de O1 e O2. Extraído de [THA 85]. 101

Figura 5.4 Transformação de um bispo em uma rainha. 104

Figura 5.5 Transformação de uma torre em um rei. 105

Figura 5.6 Objetos modelados por *sweeping* rotacional. 106

Figura 5.7 Variação do tempo de pré-processamento necessário para transformar um peão nos demais objetos da tabela 5.3. 108

Figura 6.1 Subdivisão de *patches* para $u = 1/2$: (b) primeira subdivisão; (c) segunda subdivisão. 114

Figura 6.2 Subdivisão de *patches* para: (b) $u = 1/3$ e (c) $u = 1/2$. 114

LISTA DE TABELAS

Tabela 3.1 Relação dos subobjetos do bule de Utah, com seus respectivos números de *patches* [CRO 87] 51

Tabela 5.1 Vantagens do Algoritmo Interpolador de Formas sobre as técnicas analisadas. 103

Tabela 5.2 Objetos com seus respectivos números de *patches*. 106

Tabela 5.3 Tempos (em centésimos de segundo) envolvidos na fase de pré-processamento. 107

LISTA DE PROCEDIMENTOS

Procedimento 3.1 Algoritmo Ideal 39

Procedimento 3.2 Identificação das relações de incidência ($n \log n$) 52

Procedimento 3.3 Caminhamento pré-fixado na dimensão X 56

Procedimento 3.4 Caminhamento pré-fixado na dimensão Y 56

Procedimento 3.5 Caminhamento pré-fixado na dimensão Z 56

Procedimento 3.6 Encontra vizinhos 57

Procedimento 3.7 Pesquisa binária em árvore 3D 57

Procedimento 3.8 Pesquisa na árvore 3D na dimensão X 58

Procedimento 3.9 Pesquisa na árvore 3D na dimensão Y 58

Procedimento 3.10 Pesquisa na árvore 3D na dimensão Z 58

Procedimento 3.11 Acerta vizinhança 59

Procedimento 3.12 Identificação das relações de incidência - versão quadrática 61

Procedimento 3.13 Determinação do *centro de malha* (CM) 64

Procedimento 3.14 Equaciona número de *patches* 67

Procedimento 3.15 Equacionamento vertical 67

Procedimento 3.16 Equacionamento horizontal 68

Procedimento 3.17 Subdivide *patches* acima do centro de malha 69

Procedimento 3.18 Estabelecimento de correspondências entre *patches* . . . 74

Procedimento 4.1 Cálculo de ΔS 93

Procedimento 4.2 Cálculo de Δq 95

Procedimento 5.1 Equacionamento do número de vértices 101

GLOSSÁRIO

Frame buffer - Memória destinada a armazenar descrições das imagens a serem exibidas em um monitor de vídeo. Comumente referida como *memória de vídeo*.

Animação baseada em quadros-chaves - Técnica de animação bidimensional, na qual quadros intermediários são obtidos a partir da interpolação de quadros-chaves.

Inbetweening - Denominação do processo de interpolação utilizado para gerar quadros intermediários em uma animação baseada em quadros-chaves.

Patch - Porção delimitada de uma superfície.

Patch degenerado - *Patch* contendo pelo menos um de seus bordos degenerados, ou seja, representado por um ponto.

Pixel (*picture element*) - Cada um dos elementos da matriz de pontos de um monitor de vídeo.

Preview - Modo de apresentação de seqüências animadas, em que as cenas são exibidas em representação aramada. Constituem uma maneira rápida de verificar se as ações pretendidas para os personagens foram efetivamente realizadas.

Representação aramada - Representação de objetos por um conjunto de arestas. *Wireframe*.

Rendering - Conjunto de técnicas que visam incorporar realismo a imagens geradas por computador.

Scroll animation - Técnica utilizada para exibição de seqüências animadas que consiste em dividir a memória de vídeo (*frame buffer*) em diversas áreas onde são armazenadas as imagens. A partir de então, o controlador de vídeo exibe-as, uma a uma, observando certa ordem pré-estabelecida.

Storyboard - Seqüência de ilustrações que descreve, de modo sucinto, as cenas mais importantes de um filme, organizadas cronologicamente.

Sweeping - Técnica de modelagem na qual a descrição de um objeto é obtida pelo deslocamento de uma curva no espaço. Tal curva pode sofrer alterações em sua forma ao longo do deslocamento.

Voxel (*volume element*) - Análogo tridimensional do *pixel*.

RESUMO

Este trabalho apresenta um algoritmo para interpolação de formas entre objetos modelados por superfícies *spline*. Cada um dos objetos envolvidos pode apresentar geometria bastante complexa, desde que sua superfície corresponda à realização de um espaço topológico conexo.

O **Algoritmo Interpolador de Formas** possibilita a geração, de modo automático, das diversas formas intermediárias, sem necessidade de intervenção por parte do usuário. Este precisa apenas especificar os objetos inicial e final. Pode-se, de maneira bastante simples, associar-lhe uma técnica para controle da velocidade de interpolação.

O critério de correção do **Algoritmo Interpolador de Formas** baseia-se em elementos da Topologia Combinatória. De modo a torná-lo evidente, a descrição do algoritmo é realizada em três etapas. Num primeiro instante, apresenta-se um *algoritmo ideal* capaz de realizar transformações entre espaços topológicos. O passo seguinte corresponde a traduzí-lo para uma linguagem de topologia. Por último, a versão topológica é descrita em termos geométricos.

A versão geométrica consiste em: converter os objetos a serem interpolados para uma representação padrão (Bézier bicúbica); identificar as relações de incidência entre os diversos *patches* que compõem as descrições de cada um dos objetos; eleger, para cada um destes, um *patch* de referência; equacionar, relativamente aos *patches* de referência, os números de *patches* acima, abaixo, à direita e à esquerda, em ambas representações; estabelecer uma correspondência biunívoca entre os *patches* dos objetos, preservando suas relações de incidência; definir, para cada par de *patches* associados, uma correspondência biunívoca entre seus coeficientes geométricos e interpolá-los, gerando, deste modo, as diversas formas intermediárias.

A escolha de superfícies Bézier como base para descrição do algoritmo justifica-se pela facilidade de subdividí-las preservando a forma original e simplici-

dade de conversão a partir de outros tipos. O equacionamento do número de *patches* é obtido por meio de subdivisões dos mesmos, utilizando o algoritmo de *de Casteljau*.

Um protótipo de sistema de animação que implementa uma versão simplificada do algoritmo é descrito em detalhes. A partir desse protótipo, foram produzidas várias seqüências de interpolação que ilustram a eficácia do algoritmo proposto. Essas encontram-se distribuídas ao longo do texto.

O trabalho também apresenta uma análise da complexidade do algoritmo e compara-o com outros encontrados na literatura. Neste ponto, ressaltam-se as vantagens do **Algoritmo Interpolador de Formas** sobre aqueles, destacando-se que: as formas intermediárias representam uma transição coerente entre os objetos inicial e final; não é exigida qualquer intervenção por parte do usuário e superfícies curvas podem ser manipuladas diretamente.

Palavras-chaves: Interpolação de Formas, Animação Modelada, Superfícies *Spline*.

ABSTRACT

This work presents an algorithm for shape interpolation between objects modeled by spline surfaces. It allows that each object envolved can have a complex shape, since it be a realization of a conex topologic space.

The **Algoritmo Interpolador de Formas** allows the generation, in an automatic fashion, of all intermediate shapes involved in an interpolation process without intervention by the user. The user has only to specify the initial and final objects. It is possible to incorporate an approach for interpolation velocity control.

The **Algoritmo Interpolador de Formas**' correction criterion lies on Combinatorial Topology's formalisms. To make it clear, the algorithm's description is carried out in three steps. In the first one, an *ideal algorithm* able to transform topological spaces is presented. In the next step, this ideal algorithm is translated into a topological language. Finally, the topological version is translated into a geometrical one.

The geometric version consists of: convert the objects to bicubic Bézier form; identify the neighborhood of the objects' patches; choose, in each object, a reference patch; equalize, relatively to the reference patches, the number of patches above, under, at right and at left in both representations; establish correspondences between the patches of both objects; define for each pair of associated patches correspondences between their control points and interpolate the associated control points.

Bézier patches are used as standard form because their facilities for subdivision and conversion from and to other representations. Subbdivisions of Bézier patches are performed using de Casteljau's algorithm.

A prototype of an animation system that implements a simplified version of the algorithm is discussed in detail. Many sequences that ilustrate shape interpolations examples were obtained using this prototype. These sequences can be found along the text.

This work also analizes the complexity of the proposed algorithm and compares it with others found in the literature. The **Algoritmo Interpolador de Formas**' advantages are showed: intermediate shapes correspond to soft transitions between the initial and final objects; the user has only to specify the objects to be interpolated and the algorithm can manipulate curved surfaces.

Key-words: Shape Interpolation, Modeled Animation, Spline Surfaces.

1 INTRODUÇÃO

Até bem poucos anos, reconhecia-se na indústria cinematográfica dois tipos de produtos bastante distintos: filmes convencionais e desenhos animados. Mais recentemente, observa-se uma fusão entre estes, com resultados surpreendentemente enriquecedores para ambas as partes. A Computação Gráfica desempenha, neste contexto, um papel fundamental, garantindo a unificação dos dois mundos. Cenas, antes impossíveis de serem obtidas apenas com o uso de técnicas tradicionais, podem, agora, ser modeladas, coloridas e animadas, garantindo efeitos impressionantes. Estes se incorporaram à linguagem moderna e têm no cinema e na publicidade suas principais formas de expressão. Busca-se imitar e estender a realidade. Assim, ao mesmo tempo em que se utilizam modelos físicos de iluminação, síntese de texturas, animações baseadas na Física, também se busca a produção de efeitos virtuais. Cenários reais podem ser palco de metamorfoses entre objetos; cenas modeladas por computador podem acolher atores humanos. Tudo de tal forma aparentemente real que, algumas vezes, o expectador não consegue distinguir entre o sintético e o natural.

O ato de produzir transformações entre figuras é intrínseco à própria natureza do desenho animado, sendo um recurso valioso e imprescindível à disposição da criatividade do animador. A sua automatização corresponde a uma das primeiras dificuldades encontradas pelos sistemas de animação bidimensionais assistidas por computador [BUR 71]. Nestes casos, foram alcançados resultados significativos, ainda que com necessidade de considerável intervenção por parte do usuário. Contudo, existem restrições decorrentes da perda de informação que acontece na passagem objeto 3D→ projeção 2D. Estas ocorrem, por exemplo, quando se faz necessário interpolar linhas visíveis de um objeto em linhas escondidas (não visíveis) de outros. Catmull [CAT 78] discute estas restrições.

No contexto tridimensional, não se conhece, da literatura, algoritmo capaz de gerar, de modo automático, interpolações entre formas geometricamente complexas ou descritas por superfícies curvas. Visando preencher tal lacuna, este

trabalho apresenta um algoritmo apto a realizar transformações entre objetos modelados por superfícies *spline*. Para tanto, faz-se necessário apenas especificar os objetos a serem interpolados, os quais podem ser compostos por diferentes números de *patches*; como resultado, são produzidas as diversas formas intermediárias envolvidas (figura 1.1).

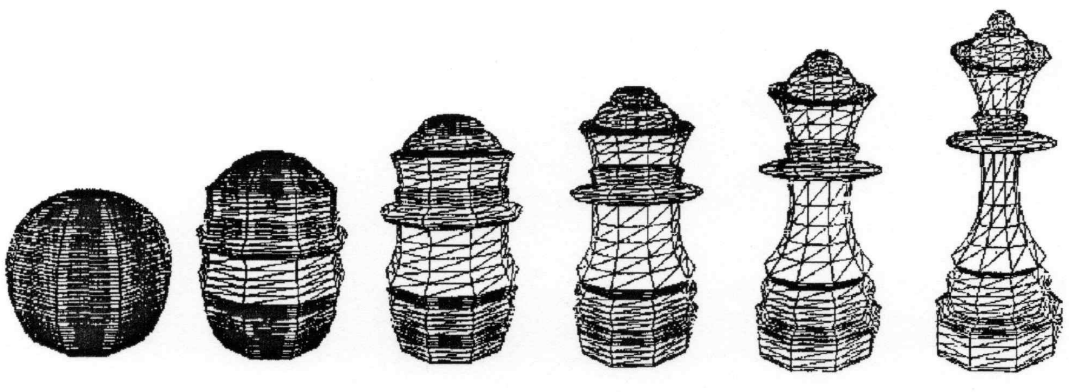

Figura 1.1: Interpolação de forma entre uma *esfera* e uma rainha de xadrez.

1.1 O estado da arte

A literatura apresenta duas estratégias para interpolação de formas, as quais, no capítulo 5, serão discutidas e comparadas com o método proposto. Estas correspondem às abordagens de Bethel e Uselton [BET 89], restrita a transformações entre poliedros, e de Lopes [LOP 88], onde é feita uma generalização a três dimensões das idéias de Burtnyk e Wein [BUR 71]. A técnica de Lopes [LOP 88] manipula objetos descritos por arestas e exige do usuário uma forte interferência no processo. Ambas abordagens apresentam limitações quanto à impossibilidade de manipular superfícies curvas, necessidade de intervenção por parte do usuário (seja estabelecendo correspondências entre arestas, no primeiro caso; seja construindo descrições específicas para os objetos, de modo a favorecer uma dada transformação, no segundo caso) e dificuldade para modelagem dos objetos.

Digital Productions, empresa de Los Angeles, especializada em Computação Gráfica, produziu seqüências animadas que ilustram interpolações de formas e afirma poder realizá-las entre, *praticamente*, quaisquer objetos [DRE 89]. Embora seus resultados se constituam em material de extrema importância para uma análise crítica, tanto do método proposto quanto do por ela aplicado, as tentativas de obter detalhes sobre a abordagem utilizada não lograram sucesso.

1.2 Estética e formalismo

Não obstante os extraordinários resultados alcançados com o emprego de diversas técnicas desenvolvidas em Computação Gráfica, fica a questão levantada pelo matemático contemporâneo Philip Davis [DAV 88]: *quais são os elementos de universalidade que conferem à Ciência da Computação* - e, particularmente, à Computação Gráfica, - *o direito de ser denominada ciência, e não meramente um ofício*? *Seriam matemáticos esses elementos*?

Se durante os anos 70, alguns efeitos interessantes foram produzidos valendo-se de *truques* inteligentes [BLI 78], hoje, buscam-se resultados que sejam, não apenas esteticamente agradáveis ou de aparência real, mas que possam ser descritos por algum formalismo. De acordo com essa tendência, o algoritmo aqui apresentado é expresso em termos de elementos da Topologia Combinatória, de modo a evidenciar seu critério de correção. Além disso, extensões podem ser mais claramente sugeridas se apoiadas em descrições menos informais.

1.3 Motivação e objetivos

Em virtude das limitações presentes nas propostas de Bethel [BET 89] e Lopes [LOP 88] e dada a importância de um recurso para interpolação de formas em um ambiente profissional para animação, este trabalho objetivou a construção de um algoritmo com as seguintes características:

- ser automático, no sentido de exigir do usuário apenas a indicação dos objetos a interpolar;
- permitir a direta manipulação de superfícies curvas;
- realizar interpolações entre objetos modelados por diferentes números de *patches*;
- ser descrito por algum tipo de formalismo, de modo a se poder, facilmente, evidenciar sua validade e critério de correção e
- permitir, de maneira simples, a incorporação de técnicas para controle da velocidade de interpolação, através de uma linguagem gráfica.

A escolha de *patches spline* como base para modelagem dos objetos justifica-se pelo fato de permitirem a representação de superfícies curvas e por terem se tornado um padrão na área de modelagem geométrica.

1.4 Estrutura da dissertação

Os demais capítulos deste trabalho encontram-se organizados da seguinte maneira: o capítulo 2 revisa a utilização do algoritmo de *de Casteljau* como técnica para subdivisão de *patches* Bézier; o capítulo 3 apresenta o **Algoritmo Interpolador de Formas** (**AIF**), realizando uma transição gradual entre sua versão topológica e outra, eminentemente geométrica. Tal apresentação visa explicitar o formalismo subjacente à estratégia proposta. A versão geométrica, assim obtida, é descrita com detalhes em cada uma de suas fases e são discutidas situações de exceção (casos especiais). O capítulo encerra com uma análise da complexidade envolvida e com algumas considerações sobre visualização de transformações não definidas continuamente. O capítulo 4 descreve **INTERPOL3D** - um protótipo de sistema de animação que implementa uma versão simplificada do algoritmo proposto. Esta simplificação, entretanto, não compromete suas potencialidades, permitindo a obtenção de resultados bastante expressivos. O capítulo 5 discute os resultados alcançados, analisando duas outras técnicas presentes na literatura, sendo enfatizadas as vantagens do **Algoritmo Interpolador de Formas** sobre essas. Também são

apresentados tempos obtidos com a utilização do protótipo, confirmando sua expectativa de complexidade. O capítulo 6 conclui o trabalho sugerindo possíveis extensões. Por fim, um anexo resume e esclarece alguns dos principais conceitos relacionados à produção de seqüências animadas com auxílio de computador.

2 SUBDIVISÃO DE SUPERFÍCIES BÉZIER

2.1 Introdução

Este capítulo revisa a utilização do algoritmo de *de Casteljau* como técnica para subdivisão de *patches* Bézier, enfatizando que as superfícies resultantes, enquanto justapostas, preservam as formas dos *patches* originais.

Equações paramétricas constituem o meio mais utilizado para representação de objetos em modelagem geométrica [MOR 85]. Superfícies bicúbicas são as formas mais freqüentemente utilizadas, devido ao fato de que superfícies de menor grau apresentam pouca flexibilidade sobre o controle de suas formas, enquanto que superfícies de maior grau podem introduzir torções indesejáveis, além de requisitarem maior esforço computacional [FOL 90]. Devido ao papel particularmente importante que desempenham no contexto de modelagem geométrica, todas as menções a curvas e superfícies, neste trabalho, referem-se a curvas cúbicas e superfícies bicúbicas, respectivamente.

2.2 Curvas e superfícies Bézier

Uma curva Bézier paramétrica $C_B(v)$ (figura 2.1) apresenta a seguinte forma matricial [MOR 85]:

$$C_B(v) = \left[\begin{array}{cccc} (1-v)^3 & 3v(1-v)^2 & 3v^2(1-v) & v^3 \end{array} \right] \left[\begin{array}{cccc} G_1 & G_2 & G_3 & G_4 \end{array} \right]^T \qquad (2.1)$$

com $v \in [0,1]$ e G_i representando o i-ésimo coeficiente geométrico (ponto de controle) da curva $C_B(v)$.

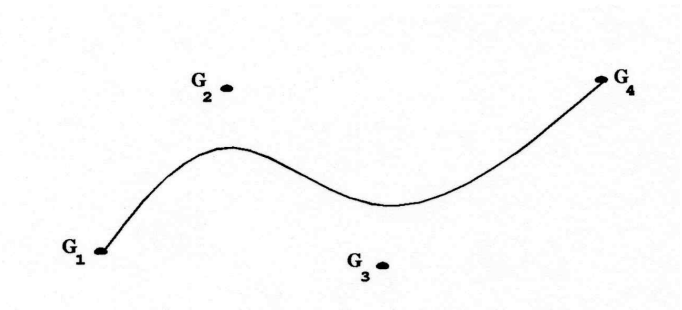

Figura 2.1: Curva Bézier.

Por sua vez, uma superfície Bézier (figura 2.2) é expressa por [MOR 85]:

$$S_B(v,u) = \left[\begin{array}{cccc} (1-v)^3 & 3v(1-v)^2 & 3v^2(1-v) & v^3 \end{array} \right] P \left[\begin{array}{c} (1-u)^3 \\ 3u(1-u)^2 \\ 3u^2(1-u) \\ u^3 \end{array} \right] \tag{2.2}$$

com v e $u \in [0,1]$ e

$$P = \left[\begin{array}{cccc} G_{11} & G_{12} & G_{13} & G_{14} \\ G_{21} & G_{22} & G_{23} & G_{24} \\ G_{31} & G_{32} & G_{33} & G_{34} \\ G_{41} & G_{42} & G_{43} & G_{44} \end{array} \right] \tag{2.3}$$

sendo P a matriz dos coeficientes geométricos da superfície.

2.3 O Algoritmo de *de Casteljau*

O algoritmo de *de Casteljau* é uma construção geométrica, que se baseia em interpolações lineares (bi-lineares) dos coeficientes que definem as curvas (superfícies) [FAR 90]. Para o caso de curvas, o algoritmo consiste em construir recursivamente, através de interpolações lineares, um polinômio de mesmo grau que a curva em questão. Assim, sejam G_1, G_2, G_3 e G_4 os coeficientes geométricos de uma curva Bézier. Tem-se (figura 2.3):

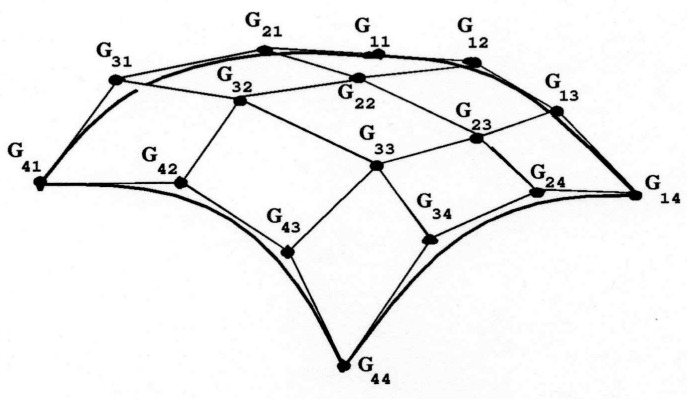

Figura 2.2: Superfície Bézier.

$$G_1^1(v) = (1-v)G_1 + vG_2 \tag{2.4}$$

$$G_2^1(v) = (1-v)G_2 + vG_3 \tag{2.5}$$

$$G_3^1(v) = (1-v)G_3 + vG_4 \tag{2.6}$$

$$G_1^2(v) = (1-v)G_1^1 + vG_2^1 \tag{2.7}$$

$$G_2^2(v) = (1-v)G_2^1 + vG_3^1 \tag{2.8}$$

$$G_1^3(v) = (1-v)G_1^2 + vG_2^2 \tag{2.9}$$

Substituindo (2.4), (2.5) e (2.6) em (2.7) e (2.8) e estes por sua vez em (2.9), obtém-se a expressão de uma curva Bézier:

$$G_1^3(v) = (1-v)^3G_1 + 3v(1-v)^2G_2 + 3v^2(1-v)G_3 + v^3G_4 \tag{2.10}$$

O algoritmo preserva os coeficientes intermediários $G_i^k(v)$, os quais podem ser convenientemente reescritos em forma triangular - o esquema de *de Casteljau* [FAR 90] (figura 2.4), onde a corresponde aos coeficientes originais da curva; b, aos coeficientes que definem o segmento compreendido entre G_1 e $G_1^3(v)$; e c, aos coeficientes que definem o segmento compreendido entre $G_1^3(v)$ e G_4 (figura 2.3).

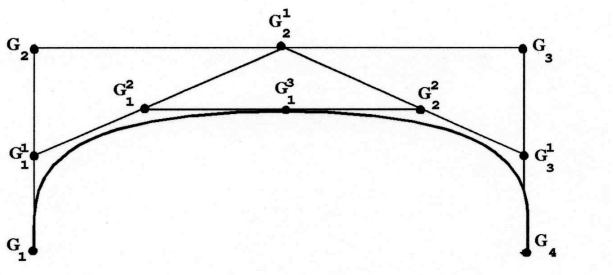

Figura 2.3: Algoritmo de *de Casteljau*.

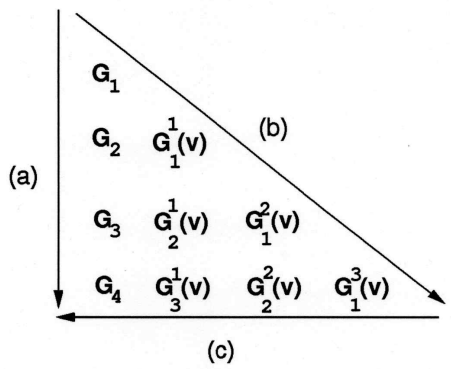

Figura 2.4: Esquema de *de Casteljau*.

2.3.1 Subdivisão de superfícies Bézier

Dada uma curva Bézier $C^0(v_0)$, com $v_0 \in [a, c]$, pode-se, por meio do algoritmo de *de Casteljau*, obter-se o ponto sobre a curva para o qual $v_0 = b, b \in [a, c]$, assim como seus respectivos coeficientes geométricos intermediários - esquema de *de Casteljau*. Isto corresponde a subdividí-la em 2 segmentos $C_1^1(v_0)$ e $C_2^1(v_0)$, tais que $C_1^1(v_0) = C^0(v_1), v_1 \in [a, b]$ e $C_2^1(v_0) = C^0(v_2), v_2 \in [b, c]$. Seja $G = [G_1\ G_2\ G_3\ G_4]^T$ a matriz dos coeficientes geométricos da curva $C^0(v_0), v_0 \in [0, 1]$. $G(C_1^1)$ e $G(C_2^1)$, os coeficientes dos novos segmentos $C_1^1(v_0)$ e $C_2^1(v_0)$ são dados por (figuras 2.3 e 2.4):

$$G(C_1^1) = \begin{bmatrix} G_1 & G_1^1(v_0) & G_1^2(v_0) & G_1^3(v_0) \end{bmatrix}^T \tag{2.11}$$

$$G(C_2^1) = \begin{bmatrix} G_1^3(v_0) & G_2^2(v_0) & G_3^1(v_0) & G_4 \end{bmatrix}^T \tag{2.12}$$

Para o caso de $v_0 = \frac{1}{2}$, tem-se:

$$G(C_1^1) = (\frac{1}{8}) \begin{bmatrix} 8 & 0 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 2 & 4 & 2 & 0 \\ 1 & 3 & 3 & 1 \end{bmatrix} G \tag{2.13}$$

$$G(C_2^1) = (\frac{1}{8}) \begin{bmatrix} 1 & 3 & 3 & 1 \\ 0 & 2 & 4 & 2 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 8 \end{bmatrix} G \tag{2.14}$$

A subdivisão de uma superfície Bézier é obtida subdividindo-se, da maneira descrita acima, as 4 isoparamétricas da dimensão escolhida (figura 2.5).

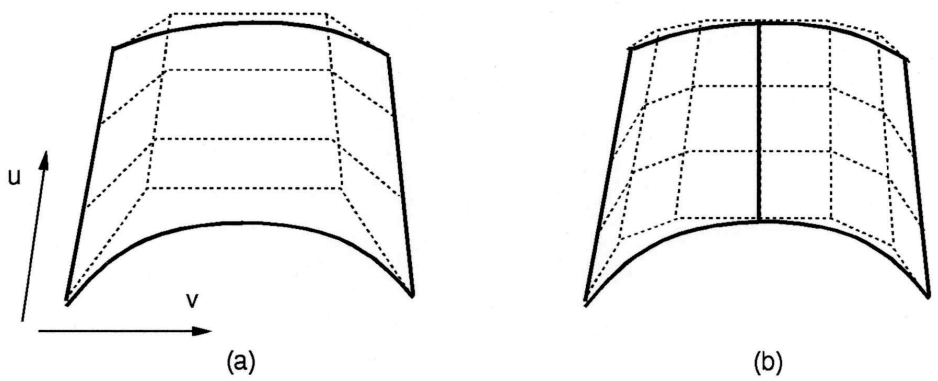

Figura 2.5: *Patch* Bézier bicúbico (a) antes da subdivisão; (b) após subdividi-lo em relação à dimensão paramétrica v.

2.4 Resumo

Este capítulo revisou a utilização do algoritmo de *de Casteljau* como técnica para subdivisão de superfícies Bézier, enfatizando que os *patches* resultantes, enquanto justapostos, preservam a forma da superfície original.

3 O ALGORITMO INTERPOLADOR DE FORMAS

3.1 Introdução

Este capítulo apresenta um algoritmo para interpolação de formas entre objetos modelados por superfícies *spline*. A construção do algoritmo será, por questões de clareza, realizada em três etapas. Partir-se-á de definições topológicas abstratas para se chegar a uma versão basicamente geométrica. Essas passagens, pretendidas suaves, intencionam evidenciar o formalismo topológico subjacente ao algoritmo e que lhe serve como critério de correção. Assim, num primeiro instante, serão apresentados alguns conceitos de modo a familiarizar o leitor com pouco ou nenhum conhecimento prévio em Topologia. A seguir, será descrito um *algoritmo ideal* capaz de realizar transformações entre espaços topológicos. A seção seguinte traduz o *algoritmo ideal* em termos de uma linguagem da Topologia Combinatória. Por último, será feita a tradução da linguagem topológica para uma linguagem geométrica factível de implementação em um computador digital. Nesta última fase, serão feitas considerações sobre a ordem de complexidade do algoritmo apresentado, bem como sobre sua utilização num contexto de geração de seqüências animadas, no qual as transformações pretendidas não estejam definidas continuamente.

3.2 Alguns conceitos topológicos

3.2.1 Espaços métricos, espaços conexos e conjuntos fechados

Um *espaço métrico* [1] é um par (M, d), onde M é um conjunto e d , a sua métrica, uma função $d : MxM \rightarrow R$ que associa a cada par ordenado de elementos $x, y \in M$ um número real $d(x, y)$, chamado a distância de x a y , de modo que sejam satisfeitas as seguintes condições para quaisquer x, y e $z \in M$ [LIM 77]:

d1) $d(x, x) = 0$;

[1] Todo espaço métrico é um espaço topológico, embora nem todo espaço topológico seja metrizável.

d2) se $x \neq y$ então $d(x, y) > 0$;

d3) $d(x, y) = d(y, x)$;

d4) $d(x, z) \leq d(x, y) + d(y, z)$

Um espaço métrico é dito *conexo* se ele não pode ser decomposto na união de dois conjuntos abertos não-vazios e disjuntos (figura 3.1). Um *caminho* em um espaço métrico M é uma aplicação contínua $f : [0, 1] \rightarrow M$, onde chama-se $f(0)$ a origem e $f(1)$ a extremidade do caminho f (figura 3.2); diz-se que M é *conexo por caminhos* se para cada par de pontos (x, y) de M existe um caminho $f : [0, 1] \rightarrow M$ contido em M, tal que $f(0) = x$ e $f(1) = y$ [LIM 77].

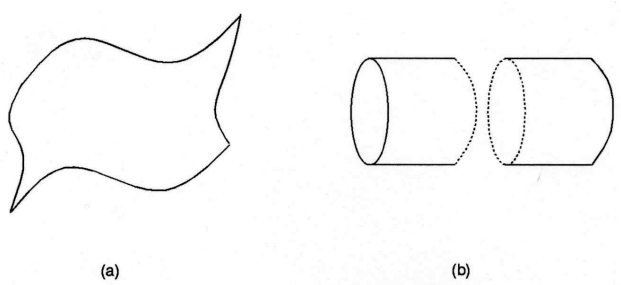

Figura 3.1: Espaços: (a) conexo (b) não conexo.

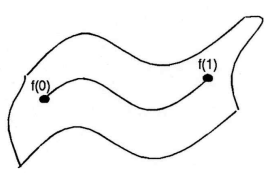

Figura 3.2: Caminho $f : [0, 1] \rightarrow M$.

Um conjunto é dito *fechado* (figura 3.3(a)) se a ele pertencem, além de seus pontos interiores, também os pontos da sua fronteira. Por exemplo, o círculo $\{(x, y) \ / \ x, y \in R \text{ e } x^2 + y^2 \leq 1\}$.

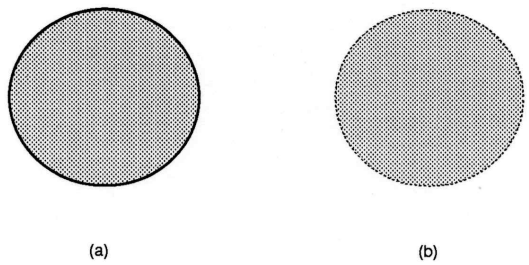

Figura 3.3: Conjuntos: (a) fechado (b) aberto.

3.2.2 Homotopia, homeomorfismo e homeotopia

Sejam M e M' dois espaços métricos. Duas funções contínuas $f, g: M \rightarrow M'$ são ditas *homotópicas* se existe uma aplicação contínua $F : Mx[0,1] \rightarrow M'$, tal que $F(x,0) = f(x)$ e $F(x,1) = g(x)$ para cada ponto x de M. Considerando o segundo parâmetro da função F como um parâmetro de tempo, uma homotopia F representa, intuitivamente, uma deformação contínua da aplicação f para a aplicação g , com a restrição de F a Mxt_0 sendo o estágio da deformação no instante t_0. Assim, sejam, por exemplo, f a função que define uma superfície plana delimitada (figura 3.4(a)) e g a função que define uma superfície cilíndrica (figura 3.4(d)). Uma aplicação F, como descrita, permite a deformação da superfície plana na superfície cilíndrica. Observe-se, entretanto, que F, embora esteja definida continuamente, não possui inversa. Isto decorre da identificação, em (d), de pontos pertencentes aos bordos da superfície (a) (representados pelas arestas pontilhadas), caracterizando F como uma aplicação não injetiva.

Uma aplicação contínua bijetiva $f : M \rightarrow M'$ com inversa $f^{-1} : M' \rightarrow M$, também contínua, é denominada um *homeomorfismo* de M sobre M'. Diz-se, nestes casos, que os espaços M e M' são homeomorfos, ou ainda, que apresentam a mesma estrutura topológica [LIM 77]. Dois espaços homeomorfos podem ser continuamente deformados tanto no sentido $f : M \rightarrow M'$, como no sentido inverso $f^{-1} : M' \rightarrow M$ [LIM 77] (figura 3.5). Note-se que um homeomorfismo exige ape-

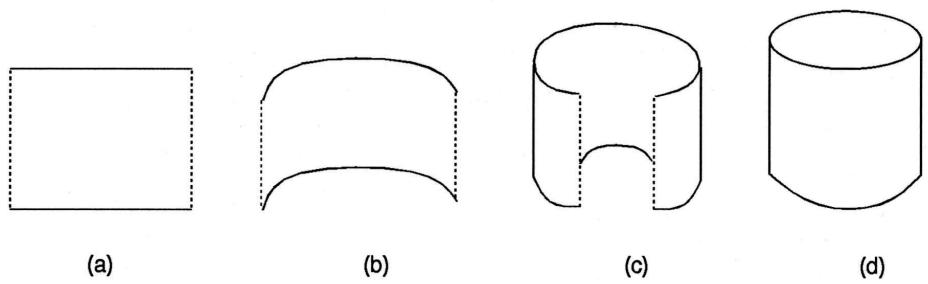

Figura 3.4: Homotopia: superfície plana → superfície cilíndrica.

nas uma definição que associe a cada ponto $x \in M$, seu correspondente $x' \in M'$, não sendo necessário o conhecimento do caminho que leva x em x'. Entretanto, no contexto deste trabalho, homeomorfismos estarão sempre associados a caminhos que levem x em x'.

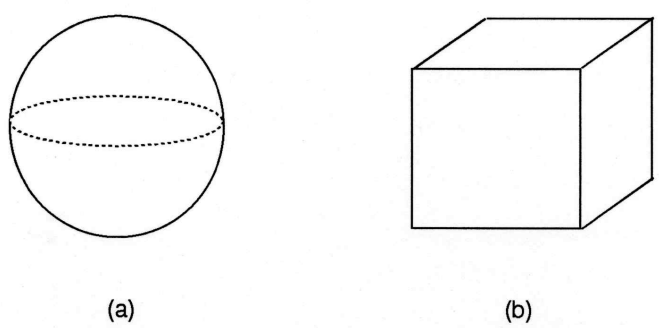

Figura 3.5: Homeomorfismo e homeotopia (esfera e cubo)

Uma transformação homotópica é dita uma *homeotopia* quando cada uma das superfícies intermediárias é, simultaneamente, homeomorfa a f e a g. Tem-se, nestes casos, uma situação de homeomorfismo para a qual se conhece o caminho que leva cada ponto $x \in M$ em seu correspondente $x' \in M'$ e que ocorre sem interpenetração da superfície em transformação. Observe-se que esta última característica nem sempre é satisfeita para o caso dos homeomorfismos (figura 3.6). Tais como estes, homeotopias correspondem a transformações bijetoras e bicontínuas.

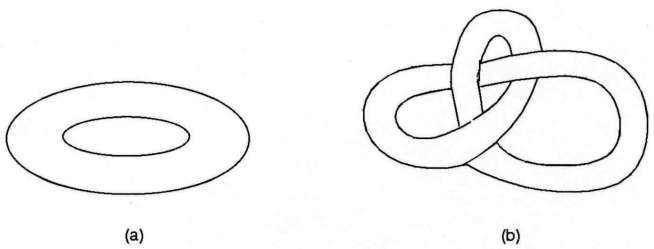

Figura 3.6: Espaços homeomorfos não homeotópicos no R^3: (a) toro; (b) toro anelado.

3.2.3 Decomposição celular e subdivisão celular

Seja M uma superfície. Uma *decomposição celular* de M é uma representação planar em que a superfície é subdividida em regiões homeomorfas a polígonos, sendo suas arestas convenientemente rotuladas e orientadas. A superfície original pode ser reconstituída *colando-se* as arestas de mesmos rótulos, respeitando-se suas orientações (figura 3.7). Cada uma das regiões assim obtidas é dita uma célula ou simplexo.

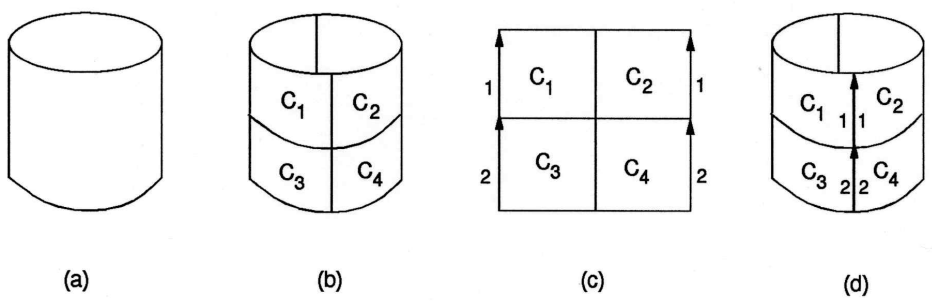

Figura 3.7: Decomposição celular: (a) superfície cilíndrica; (b) subdivisão em quadriláteros; (c) decomposição celular; (d) reconstrução da superfície

A figura 3.7 apresenta: (a) uma superfície cilíndrica; (b) uma possível subdivisão da superfície em quadriláteros; (c) sua decomposição celular e (d) a reconstrução da superfície original a partir de (c).

As formas e o número de células envolvidas em uma dada decomposição pode variar bastante. Deste modo, é possível ter-se, para a superfície representada na figura 3.7, a configuração alternativa mostrada na figura 3.8. A representação da célula C_1 (figura 3.7(c)) é equivalente à constituída por c'_1, c'_2, c'_5 e c'_6 (figura 3.8). Percebe-se, portanto, que dada uma célula C, é sempre possível decompô-la em n sub-células, $n \geq 2$. À decomposição de C em sub-células, denomina-se *subdivisão celular*.

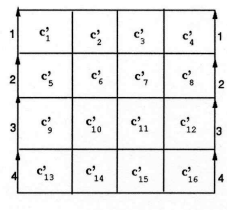

Figura 3.8: Subdivisão celular da superfície 3.4 (c)

3.2.4 Relações de incidência e representações isomorfas

Seja P um polígono qualquer e sejam $v_1, v_2, \cdots, v_n$ os seus vértices. Diz-se que v_i é vértice incidente a uma aresta e_j de P, se v_i é vértice de e_j . Neste caso, tem-se também que e_j é aresta incidente ao vértice v_i . Diz-se ainda que e_j é aresta incidente à face de P (f), se e_j é aresta de P. E, neste caso, que f é face incidente à aresta e_j . Duas decomposições celulares D e D' são ditas *isomorfas* se é possível estabelecer uma correspondência biunívoca I entre as faces, as arestas e os vértices de D e D', tal que preserve as relações de incidência. Em outras palavras, se f é face de D, e_i e e_j são arestas incidentes à f, v_k é vértice incidente às arestas e_i e e_j e se f' é face de D', e'_i e e'_j são arestas incidentes à f', v'_k é vértice incidente às arestas e'_i e e'_j, então se os pares $(f, f'), (e_i, e'_i)$ e (v_k, v'_k) encontram-se definidos por I, tem-se que o par (e_j, e'_j) também se encontra definido por I (figura 3.9).

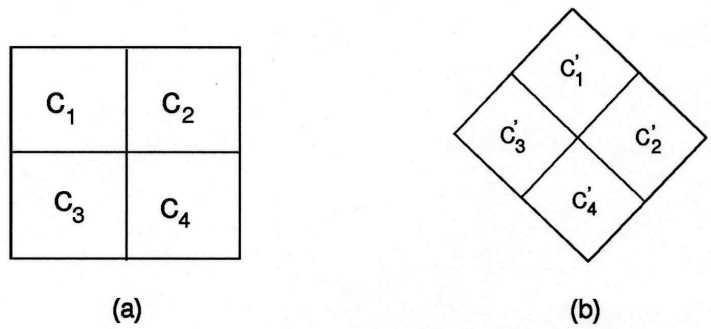

Figura 3.9: Decomposições celulares isomorfas

3.3 O algoritmo ideal

Um *algoritmo ideal* capaz de deformar continuamente um espaço topológico M em um outro espaço topológico M' pode ser expresso pelo procedimento a seguir:

procedimento *transformação* (M, M': espaços topologicos)
inicio
  caso $f : M \to M'$ **seja**
  homeotopia: aplique a homeotopia $f : M \to M'$;
    pare;
  homeomorfismo: defina o homeomorfismo $f : M \to M'$;
    pare;
  homotopia: aplique a homotopia $f : M \to M'$;
    pare;
  outros: informe que não e possivel deformar M em M';
  fim_caso;
fim
Procedimento 3.1 Algoritmo ideal.

Um algoritmo ideal deve realizar, preferencialmente, transformações homeotópicas. Caso isso não seja possível, definir homeomorfismos ou, em último caso, efetuar

transformações homotópicas. Para os casos em que não seja possível realizar a deformação, o usuário deve ser informado pelo algoritmo.

3.4 Uma versão topológica para o algoritmo ideal

Considere os seguintes teoremas da Topologia Combinatória:

TEOREMA 3.1 Toda superfície definida no R^3 possui uma representação planar denominada decomposição celular [ALE 35];

TEOREMA 3.2 Se M e M' são homeomorfos, então é possível obter-se representações isomorfas para M e M', por meio de decomposições e subdivisões celulares [ALE 35].

Note-se que a demonstração do **teorema 3.2** não é construtiva, *i.e.*, não provê um método para obter tais decomposições, embora lhes garanta a existência.

Sejam M e M' dois espaços conexos. Pode-se, então, realizar a seguinte tradução do *algoritmo ideal* para uma linguagem topológica:

1. Obter decomposições celulares para M e M';
2. Caso M e M' sejam homeomorfos, promover subdivisões celulares de modo a tornar suas representações isomorfas; caso contrário, realizar subdivisões celulares de modo a equacionar os números de *linhas* e *colunas* de células em ambas representações (figura 3.10);
3. Estabelecer uma correspondência biunívoca entre as células de M e M', observando suas relações de incidência e
4. Aplicar transformações $f_i : c_i \rightarrow c'_i$ que levem cada célula c_i de M em sua célula correspondente c'_i de M'.

A verificação de que o algoritmo funciona sempre que existir uma aplicação contínua $f : M \rightarrow M'$ pode ser dividida em duas partes:

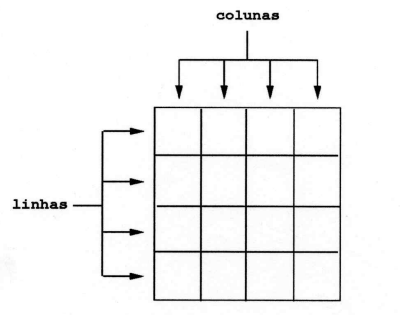

Figura 3.10: Subdivisão de um espaço conexo (plano delimitado) em *linhas* e *colunas*.

- quando M e M' forem homeomorfos e
- quando M e M' forem não homeomorfos.

3.4.1 M e M' homeomorfos

Considere as decomposições celulares de M e M' apresentadas na figura 3.11. Pelo **teorema 3.2**, pode-se obter representações isomorfas para M e M' (figura 3.12). Os pares de células (a'_i, b'_i) são constituídos por conjuntos fechados e disjuntos. Como existe uma tranformação afim, e portanto contínua, $f_i : a'_i \rightarrow b'_i$ em R^3 , que mapeia cada célula $a'_i \in M$ em sua correspondente $b'_i \in M'$, com $f_i \mid a'_i \cap a'_j = f_j \mid a'_i \cap a'_j$ (f_i e f_j apresentam mesmos valores quando restritas aos pontos das arestas comuns às celulas a_i e a_j), então $\cup f_i$ é contínua e, portanto, existe uma transformação contínua de M em M'. Tem-se, ainda, que $f_i^{-1} : b'_i \rightarrow a'_i$ e $\cup f_i^{-1}$ também são contínuas. Assim, para transformar M em M', basta aplicar $g = \cup f_i$.

3.4.2 M e M' não homeomorfos

Quando M e M' não forem homeomorfos, nem sempre será possível realizar a deformação no sentido pretendido. Assim, por exemplo, pode-se definir uma transformação homotópica de uma superfície plana em uma superfície cilíndrica

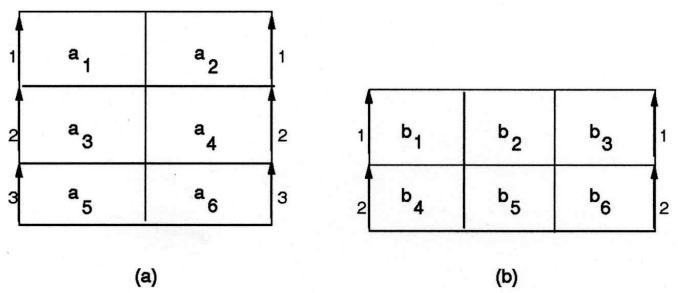

Figura 3.11: Decomposições celulares de duas superfícies cilíndricas.

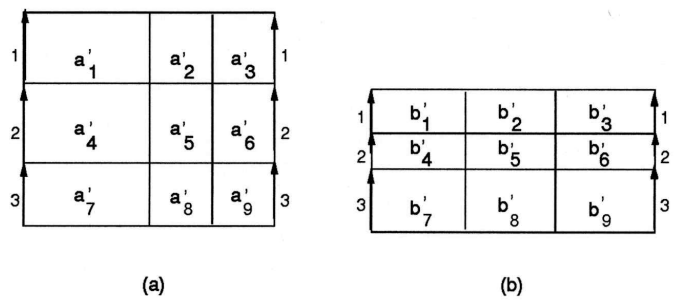

Figura 3.12: Representações isomorfas para M e M'.

(figura 3.4) ou de uma esfera em um toro, não sendo possível, entretanto, definir suas inversas.

Muito embora não se possa construir representações isomorfas entre espaços não homeomorfos, pode-se realizar decomposições celulares de modo a obter representações com iguais números de células. Considere as representações de uma superfície cilíndrica M e de um toro M', como apresentadas na figura 3.13. Neste exemplo, existe homotopia no sentido superfície cilíndrica$\rightarrow$ toro. Subdivisões celulares podem ser aplicadas a M e a M' de modo a produzir representações com iguais números de *linhas* e *colunas*, como as mostradas na figura 3.14.

Novamente, a'_i e b'_i são conjuntos fechados e disjuntos. Como existe uma transformação afim, e portanto contínua, $f_i : a'_i \rightarrow b'_i$ em R^3 com $f_i \mid a'_i \cap a'_j = f_j \mid a'_i \cap a'_j$, então $\cup f_i$ é contínua. E, mais uma vez, para transformar M em M', basta aplicar $g = \cup f_i$. Note-se, entretanto, que $f_i^{-1} : b'_i \rightarrow a'_i$ não se encontra definida, uma vez que os pontos situados sobre as arestas 3 e 4 não têm uma única imagem pela inversa.

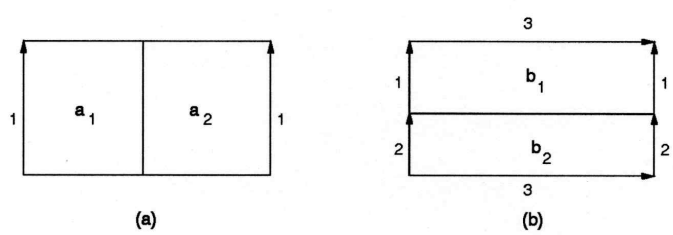

Figura 3.13: Decomposições celulares: (a) superfície cilíndrica; (b) toro.

3.5 Uma versão geométrica para o algoritmo ideal

Sejam O e O' dois objetos conexos descritos por superfícies paramétricas no R^3 e sejam P e P', respectivamente, os conjuntos de *patches* que os definem. Pelo **teorema 3.1**, existem D e D', as decomposições celulares de O e O', respec-

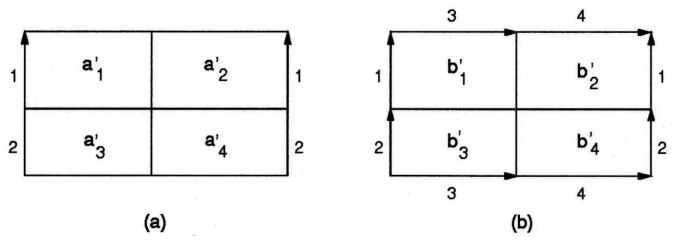

Figura 3.14: Equacionamento do número de células.

tivamente. Pode-se, portanto, fazer com que os conjuntos de células e os conjuntos de *patches* que definem um mesmo objeto coincidam, ou seja, $D = \{p \ / \ p \in P\}$ e $D' = \{p' \ / \ p' \in P'\}$. Como cada *patch* é um conjunto fechado (formado por todos os seus pontos interiores mais os pontos de fronteira), de modo a se estabelecer uma analogia completa com a versão topológica do algoritmo, resta definir um critério para subdivisão de *patches*, análogo à subdivisão celular, com a restrição de não alterar a geometria original do *patch* subdividido e, por extensão, a geometria do objeto como um todo.

3.5.1 Uma representação padrão

A escolha adequada de uma forma padrão para representar os diversos *patches* dos objetos e que leve em conta as características do algoritmo é fundamental. Principalmente, porque se pretende permitir transformações entre objetos modelados por diferentes tipos de superfícies, por exemplo, Hermite, *B-spline*, entre outras.

Dentre as representações paramétricas, as superfícies Bézier são as que apresentam maior facilidade de subdivisão [BAR 87] [FOL 90]. Bartels [BAR 87] cita um trabalho de Catmull de 1975 no qual este propõe um método para subdivisão de curvas e superfícies Bézier que se utiliza apenas de somas e *shifts*.

Superfícies *B-pline* não uniformes racionais correspondem àquelas com maior poder de expressão, permitindo a exata representação de superfícies quádricas e de revolução. Entretanto, por não apresentarem matrizes de base explícitas (no caso de superfícies não uniformes tais matrizes são definidas recursivamente), suas subdivisões são obtidas com a utilização dos algoritmos recursivos Oslo ou de Böhm [BAR 87], computacionalmente mais caros que o algoritmo de *de Casteljau*. Ainda como decorrência de não possuírem matrizes de base explícitas, não podem ser convertidas diretamente nas demais representações, exceto Bézier, o que pode ser obtido com a inclusaão de nós com alguma multiplicidade. Por sua vez, superfícies Bézier apresentam suas pares racionais as quais também permitem representações precisas de superfícies quádricas e de revolução.

Analisando um *patch* Hermite, observa-se que seus coeficiente geométricos compõem-se de pontos de controle e vetores tangentes [FAR 90]. Já os coeficientes geométricos de *patches B-spline* são constituídos apenas por pontos de controle. Entretanto, o número destes pode variar [BAR 87]. Por sua vez, *patches* Bézier apresentam sempre 16 pontos de controle como coeficientes geométricos e podem ser obtidos a partir de, por exemplo, *patches* Hermite ou *B-spline* de mesmo grau (a recíproca também é verdadeira), preservando integralmente a geometria da superfície representada [MOR 85].

Assim, devido à maior facilidade de subdivisão [FOL 90] , uniformidade de representação (16 pontos de controle) e possibilidade de conversão a partir de diferentes representações, elegeu-se o formato Bézier como padrão a ser utilizado na versão geométrica do algoritmo.

3.5.2 A função interpoladora

Sejam O e O' objetos conexos definidos por composições de *patches* Bézier. Se O e O' são homeomorfos, então pode-se construir representações isomorfas para O e O' e, portanto, existem $g : O \rightarrow O'$ e $g^{-1} : O' \rightarrow O$, ambas contínuas. Por sua vez, caso exista somente homotopia entre O e O', tem-se apenas $g : O \rightarrow O'$, ou $g^{-1} : O' \rightarrow O$, contínua.

A aplicação $g : O \rightarrow O'$ que transforma o objeto O no objeto O', pode ser definida como $g = \cup f_i$, onde $f_i : p_i \rightarrow p'_i$, p_i corresponde ao i-ésimo *patch* do objeto O e p'_i, ao i-ésimo *patch* do objeto O'. Como cada *patch* Bézier é representado por 16 coeficientes geométricos, pode-se redefinir cada uma das aplicações f_i como $f_i : C_i \rightarrow C'_i$, onde C_i (3.1) é a matriz 4x4 dos coeficientes geométricos de p_i e C'_i (3.2), sua análoga em p'_i. O estabelecimento de uma relação $R : C_i \rightarrow C'_i$, que associa pares ordenados $(c_{lk}, c'_{lk}), 1 \leq l, k \leq 4$, é imediato. Deste modo, as aplicações $f_i : p_i \rightarrow p'_i$ podem ainda ser indicadas por $f_i : \cup h_{ilk}$, onde $h_{ilk} : c_{ilk} \rightarrow c'_{ilk}$, c_{ilk} é o elemento da linha l, coluna k de C_i e c'_{ilk} , seu análogo em C'_i.

Note-se que, mesmo nos casos em que $c_{ilk} = c_{ijw}$ e $c'_{ilk} \neq c'_{ijw}$, as aplicações $f_i : p_i \rightarrow p'_i$, se definidas ($p_i$s não degenerados - figura 3.15), serão contínuas. Observe-se que, c_{ilk}s, enquanto coeficientes geométricos sujeitos a uma ponderação, consituem apenas uma forma bastante econômica para representar os diversos pontos de uma superfície complexa, não correspondendo, necessariamente, a pontos desta superfície. Portanto, uma aplicação contínua $f_i : p_i \rightarrow p'_i$ pode ser constituída pela união de relações h_{ilk}, desde que p_i não seja um *patch* degenerado. Observe-se ainda que os caminhos que levam cada ponto $x \in p_i$ em seu correspondente $x' \in p'_i$ ficam perfeitamente determinados por meio da interpolação dos coeficientes geométricos de p_i e p'_i.

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \tag{3.1}$$

$$C' = \begin{bmatrix} c'_{11} & c'_{12} & c'_{13} & c'_{14} \\ c'_{21} & c'_{22} & c'_{23} & c'_{24} \\ c'_{31} & c'_{32} & c'_{33} & c'_{34} \\ c'_{41} & c'_{42} & c'_{43} & c'_{44} \end{bmatrix} \tag{3.2}$$

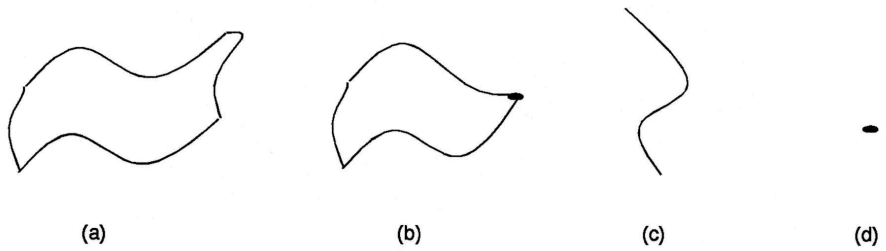

Figura 3.15: *Patches* bicúbicos: (a) não degenerado; (b) com o bordo direito degenerado em um ponto; (c) degenerado em uma linha; (d) degenerado em um ponto.

3.5.3 A versão geométrica propriamente dita

Pode-se, agora, definir um algoritmo para interpolação de formas entre objetos modelados por *patches spline*, que se compõe dos seguintes passos:

1. converter os objetos a interpolar para a representação padrão (Bézier);
2. identificar as relações de incidência para os vários *patches* que compõem as descrições de cada um dos objetos;
3. caso os objetos sejam homeomorfos, promover subdivisões em seus *patches*, utilizando o algortimo de *de Casteljau*, de modo a obter representações isomorfas; caso não sejam, as subdivisões devem equacionar os números de *linhas* e *colunas* dos *patches* que compõem as representações dos objetos;
4. estabelecer uma correspondência biunívoca entre os *patches* dos dois objetos, preservando suas relações de incidência. Para cada par de *patches* associados, estabelecer uma correspondência biunívoca entre seus coeficientes geométricos e
5. interpolar, simultaneamente, cada um dos pares de coeficientes geométricos associados no item 4, obtendo, assim, as diversas formas intermediárias.

O item 1, conversão entre representações, corresponde a assunto exaustivamente abordado na literatura [FAR 90] [BAR 87] [MOR 85] [FOL 90] e, portanto, não será discutido neste trabalho. A seguir, serão detalhadas a etapas 2 a 5.

3.5.3.1 Identificação das relações de incidência

Identificar as relações de incidência consiste em reconhecer os vizinhos (cima, baixo, direita e esquerda) de cada um dos *patches* que compõem os objetos (figura 3.16).

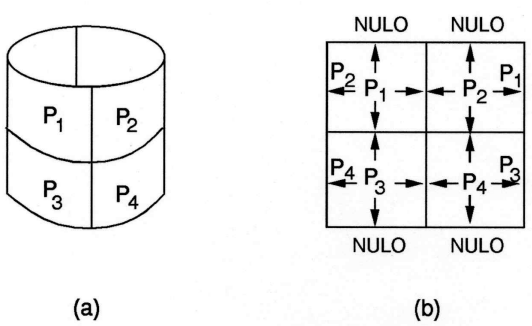

Figura 3.16: (a) superfície cilíndrica modelada por 4 *patches*; (b) decomposição celular corespondente. As setas representam as vizinhanças entre os *patches*.

Sejam n e n', respectivamente, os números de *patches* Bézier de O e O' após suas conversões para a forma padrão e suponha-se que $n > n'$. Numa abordagem *força bruta*, as arestas que representam os bordos de cada *patch* precisam ser verificadas contra as de todos os demais *patches* do objeto. Neste caso, ter-se-ia, para essa etapa do algoritmo, um custo de $O(n^2)$ ou, mais precisamente, $n^2 - n$, com espaço de armazenamento requerido de n ($O(n)$).

O emprego de estruturas de dados adequadas pode reduzir bastante o tempo de pesquisa de um algoritmo. Contudo, a utilização de tais estruturas freqüentemente só se justifica quando o algoritmo executa sobre um conjunto de dados suficientemente grande e estável [PRE 85]. De outra forma, os custos adicionais de armazenamento, inicialização e atualização das estruturas de dados, somados ao tempo de pesquisa, podem tornar seu uso desaconselhável.

Nesta fase do **AIF**, procura-se identificar *patches* que apresentem bordos em comum, ou seja, bordos que se encontrem definidos pelos mesmos (quatro) coeficientes geométricos (coordenadas X, Y e Z), caracterizando, assim, uma pesquisa associativa. Uma pesquisa é dita *associativa* quando a pergunta a qual ela procura responder tem suas condições estabelecidas em relação a um conjunto de múltiplas chaves [BEN 75] (p.e., identificar em um conjunto de pontos $p(x, y, z)$, aqueles para os quais $x > 2$, $0 < y < 12$ e $z > 5$).

Árvores de pesquisa binária multidimensionais (árvores k-D) [BEN 75] constituem estruturas de dados ótimas para realização de pesquisas associativas. Em uma árvore de pesquisa binária k-dimensional, k chaves se alternam ao longo de grupos de k níveis (figura 3.17)). A utilização de uma árvore k-D para um objeto constituído por n registros (*patches*) apresenta os seguintes custos [BEN 75]:

- construção da árvore, $O(n \log n)$;
- consulta a um nodo, $O(\log n)$;
- exclusão da raiz, $O(n^{(k-1)/k})$;
- exclusão de um nodo não raiz, $O(\log n)$ e
- reorganização da árvore de modo a proporcionar consultas ótimas, $O(n \log n)$.

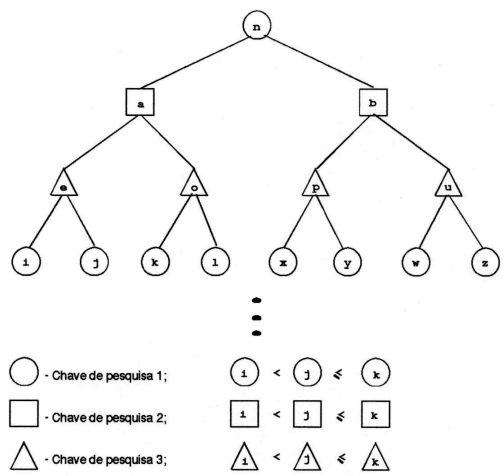

Figura 3.17: Árvore binária de pesquisa 3D.

Para o caso do **AIF**, o uso de uma árvore 3-D implica a sua construção, mais a realização de $4n$ consultas (uma para cada bordo dos n *patches*), totalizando um custo de $5n \log n$ ($O(n \log n)$) por objeto. O espaço de armazenamento necessário é $\cong 1,2n$ ($O(n)$) (figura 3.20).

Comparando as expressões de custo das abordagens multidimensional e força bruta, encontra-se o valor de n, a partir do qual a utilização de árvores 3-D se torna atraente (figura 3.18):

$$\begin{aligned} n^2 - n &= 5n \log n \\ n &\cong 24 \end{aligned} \tag{3.3}$$

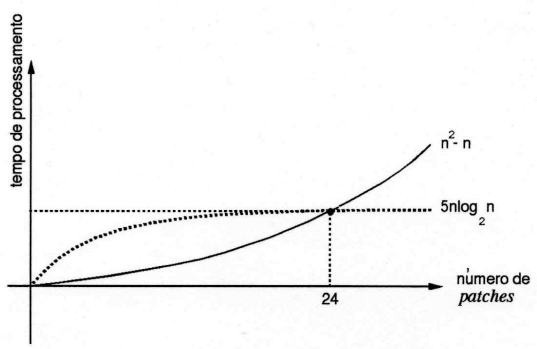

Figura 3.18: Abordagem logarítmica *versus* abordagem quadrática.

Deste modo, para objetos cujas componentes conexas sejam constituídas por conjuntos de n *patches*, tais que $n \geq 24$, a utilização de árvores 3-D é compensadora. Do contrário, a abordagem força bruta será mais econômica!

Uma consideração importante é o fato de que objetos geometricamente complexos são, com freqüência, construídos a partir de subobjetos. Assim, por exemplo, o famoso bule de Utah apresenta-se constituído por seis componentes conexas [CRO 87] apresentadas na tabela 3.1.

Tabela 3.1: Relação dos subobjetos do bule de Utah, com seus respectivos números de *patches* [CRO 87]

subobjetos	*nro de patches*
tampa	8
aro	4
bojo	8
bico	4
alça	4
fundo	4

A implementação da versão logarítimica (procedimentos 3.2 a 3.11) pode ser feita a partir dos tipos **objeto_log**, **patch_x**, **patch_y**, **patch_z** e **info_patch** (figura 3.19). A estrutura de dados requerida corresponde a uma lista encadeada com estrutura recursiva de aninhamento [TRE 85] associada a árvores de pesquisa binária 3-D (figuras 3.20 e 3.21).

tipo objeto_log:

identificação do objeto: identificador; { IDL }
primeiro_subobjeto: ponteiro para objeto_log; { SUBOBL - prim. da lista de subobjetos }
primeiro_*patch*: ponteiro para *patch_x*; { PATL - raiz da arvore 3-D de *patches* }
proximo_objeto: ponteiro para objeto_log; { PROXL - prox. na lista de objetos }

tipo patch_x:

PYesq: ponteiro para *patch_y*; { ponteiro p/ *patch* cujo Xmax < Xmin do *patch* representado }
Xmin: real; { Xm - menor abscissa do *patch* }
PYdir: ponteiro para *patch_y*; { ponteiro p/ *patch* cujo Xmax ≥ Xmin do *patch* representado }

tipo patch_y:

PZabaixo: ponteiro para *patch_z*; { ponteiro p/ *patch* cujo Ymax < Ymin do *patch* representado }

Ymin: real; { Ym - menor ordenada do *patch* }

PZacima: ponteiro para *patch_z*; { ponteiro p/ *patch* cujo Ymax ≥ Ymin do *patch* representado }

tipo patch_z:

PXatras: ponteiro para *patch_x*; { ponteiro p/ *patch* cujo Zmax < Zmin do *patch* representado }

Zmin: real; { Zm - menor z do *patch* }

patch_3D: info_*patch*; { ponteiro para os dados do *patch* representado }

PXa_frente: ponteiro para *patch_x*; { ponteiro p/ *patch* cujo Zmax ≥ Zmin do *patch* representado }

tipo info_patch:

X: ponteiro para matriz[4,4] de reais; { coordenadas X }

Y: ponteiro para matriz[4,4] de reais; { coordenadas Y }

Z: ponteiro para matriz[4,4] de reais; { coordnadas Z }

*patch*_correspondente : ponteiro para info_*patch*; { PCL - *patch* de O' no qual sera interp. }

*patch*_acima : ponteiro para info_*patch*; { PAL - vizinho acima }

*patch*_abaixo : ponteiro para info_*patch*; { PBL - vizinho abaixo }

*patch*_a_direita : ponteiro para info_*patch*; { PDL - vizinho direita }

*patch*_a_esquerda : ponteiro para info_*patch*; { PEL - vizinho esquerda }

procedimento *identifica_relações_incidência_log* (O:objeto_log)

inicio { realiza caminhamento pre-fixado }

aprofunda_x (O, $O \rightarrow$ primeiro_*patch*);

fim

Procedimento 3.2 Identificação das relações de incidência ($n \log n$).

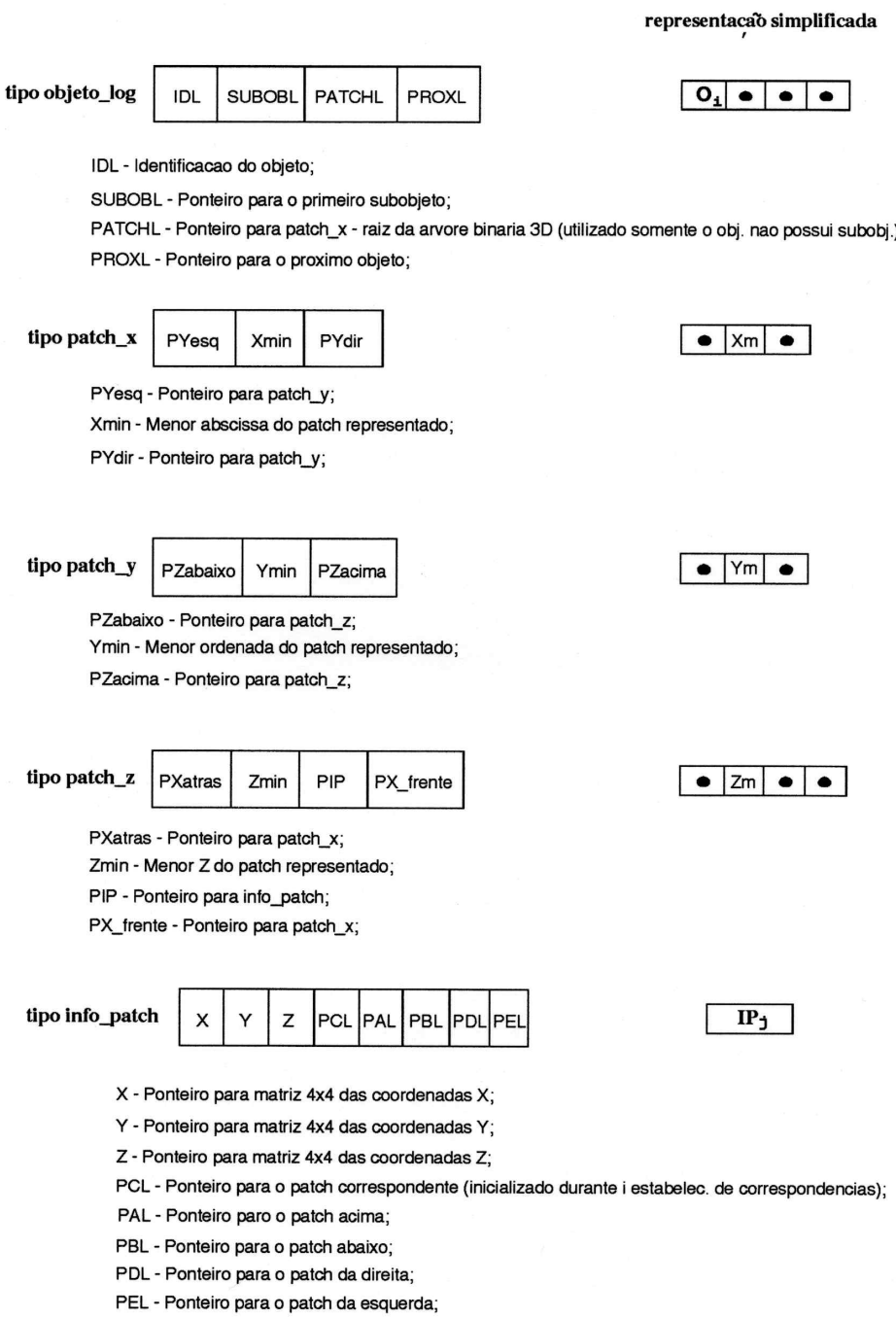

Figura 3.19: Tipos Objeto_log e *Patch_x, Patch_y, Patch_z* e Info_patch.

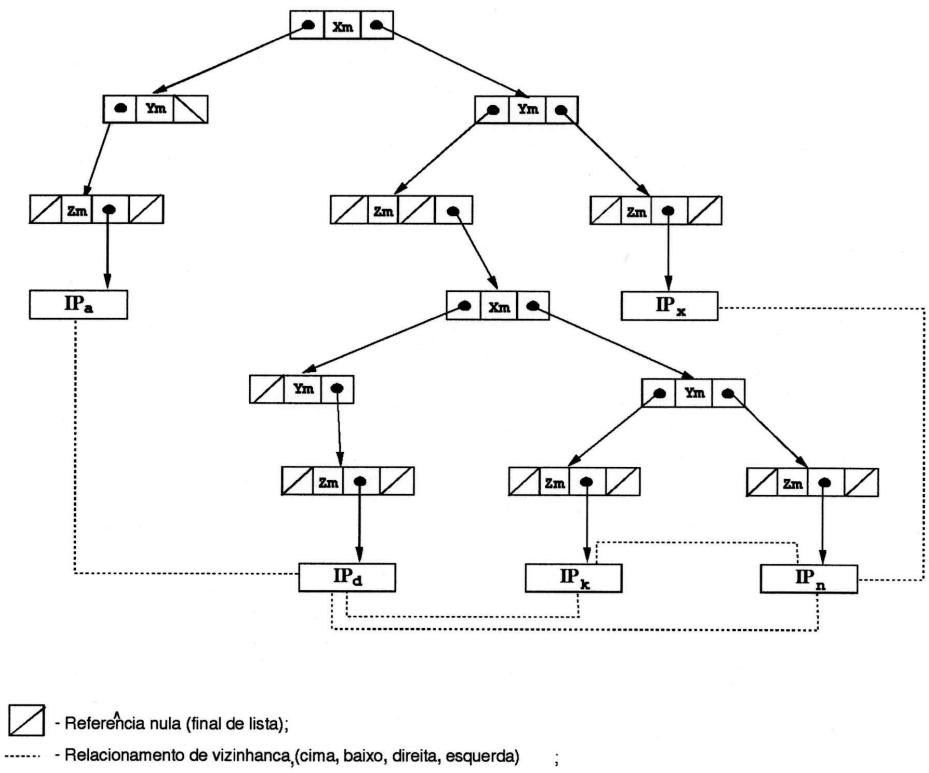

Figura 3.20: *Patches* representados em uma árvore binária de pesquisa 3D.

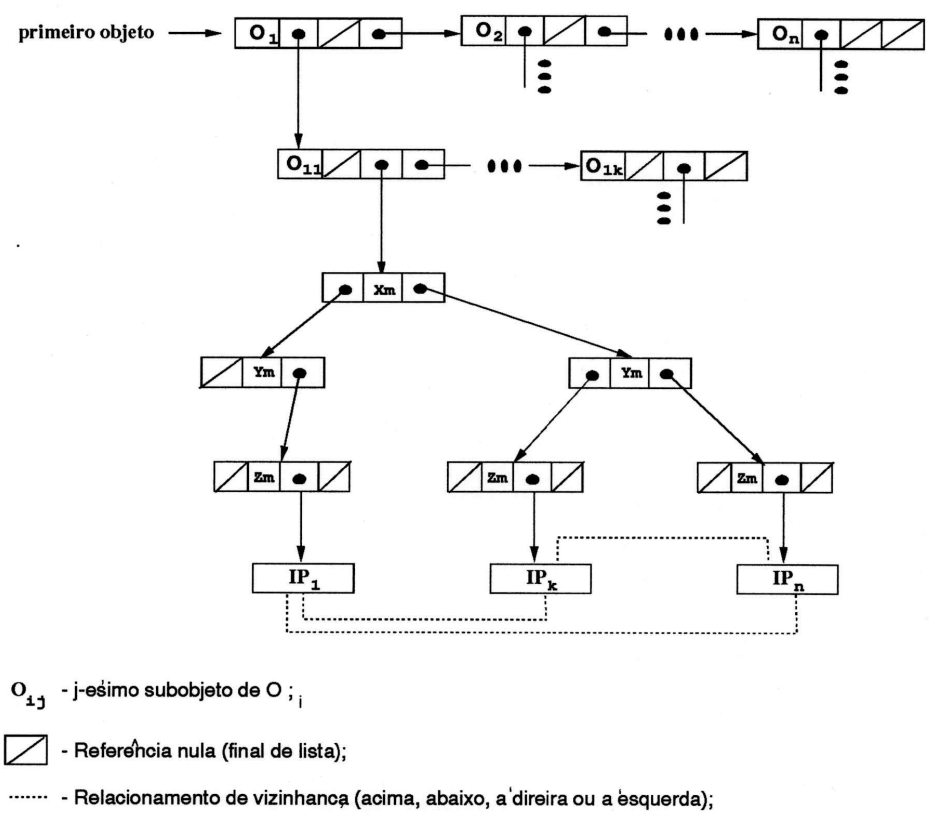

Figura 3.21: Estrutura de dados para a versão logarítmica: lista encadeada com estrutura recursiva de aninhamento associada a árvores binárias de pesquisa 3D.

procedimento *aprofunda_x* (*O*:objeto_log; pat_x: ponteiro para *patch_x*)
inicio
  se (pat_x ≠ **NULO**) **então**
  inicio
    aprofunda_y (*O*, pat_x→ PYesq);
    aprofunda_y (*O*, pat_x→PYdir);
  fim;
fim
Procedimento 3.3 Caminhamento pré-fixado na dimensão X.

procedimento *aprofunda_y* (*O*:objeto_log; pat_y: ponteiro para *patch_y*)
inicio
  se (pat_y ≠ **NULO**) **então**
  inicio
    aprofunda_z (*O*, pat_y→PZabaixo);
    aprofunda_z (*O*, pat_y→PZacima);
  fim;
fim
Procedimento 3.4 Caminhamento pré-fixado na dimensão Y.

procedimento *aprofunda_z* (*O*:objeto_log; pat_z: ponteiro para *patch_z*)
inicio
se (pat_z→PXatras ≠ **NULO ou** pat_z→PXa_frente **NULO**) **então**
inicio
aprofunda_x (*O*, pat_z→PXatras);
aprofunda_x (*O*, pat_z→PXa_frente);
fim
senão
encontra_vizinhos (*O*, pat_z);
fim
Procedimento 3.5 Caminhamento pré-fixado na dimensão Z.

procedimento *encontra_vizinhos* (*O*:objeto_log; pat_z: ponteiro para *patch_z*)
inicio
  i : inteiro;
  vizinho : ponteiro para *info_patch*;
  borda : vetor[4] = [cima, baixo, direita, esquerda];
  para i **variando de** 1 **ate** 4 **faça**
  inicio
    pesquisa_vizinho (*O*, pat_z→*patch*_3D, vizinho, bordo[i]);
    se (vizinho ≠ **NULO**) **então**
      acerta_vizinhança (pat_z→*patch*_3D, vizinho, bordo[i]);
  fim;
fim
Procedimento 3.6 Encontra vizinhos de um dado *patch*.

procedimento *pesquisa_vizinho* (*O*:objeto_log; *patch_ref*, vizinho: ponteiro para *info_patch*; bordo_ref : bordo)
inicio { pesquisa binaria em arvore 3D }
  decide_sobre_x (*O* → primeiro_*patch*, *patch_ref*, vizinho, bordo_ref);
fim
Procedimento 3.7 Pesquisa binária em árvore 3D.

procedimento *decide_sobre_x* (px : ponteiro para *patch_x*; *patch_ref*, vizinho: ponteiro para *info_patch*; bordo_ref : bordo)
inicio
  se (px ≠ **NULO**) **então**
  inicio
  se (*maxima_abscissa_de* (*patch_ref*, bordo_ref) < px.Xmin) **então**
      decide_sobre_y (px→PYesq, *patch_ref*, vizinho, bordo_ref)
    senão
      decide_sobre_y (px→PYdir, *patch_ref*, vizinho, bordo_ref)
  fim

senão

vizinho = **NULO** ;

fim

Procedimento 3.8 Pesquisa na árvore 3D na dimensão X.

procedimento *decide_sobre_y* (py : ponteiro para *patch_y*;
patch_ref, vizinho: ponteiro para *info_patch*;
bordo_ref : bordo)

inicio

se (*maxima_ordenada_de* (*patch_ref*, bordo_ref) < py.Ymin) **então**

decide_sobre_z (py→PZabaixo, *patch_ref*, vizinho, bordo_ref)

senão

decide_sobre_z (py→PZacima, *patch_ref*, vizinho, bordo_ref);

fim

Procedimento 3.9 Pesquisa na árvore 3D na dimensão Y.

procedimento *decide_sobre_z* (pz : ponteiro para *patch_z*;
patch_ref, vizinho: ponteiro para *info_patch*;
bordo_ref : bordo)

inicio

se (*maximo_z_de* (*patch_ref*, bordo_ref) < pz.Zmin) **então**

decide_sobre_x (pz→PXatras, *patch_ref*, vizinho, bordo_ref)

senão

se (pz→*patch*_3D ≠ **NULO e**

bordo_em_comum (*patch_ref*, pz→ *patch*_3D, bordo_ref) **e**

patch_ref ≠ pz→*patch*_3D) **então**

vizinho = pz→*patch*_3D

senão

decide_sobre_x (pz→PXa_frente, *patch_ref*, vizinho, bordo_ref);

fim

Procedimento 3.10 Pesquisa na árvore 3D na dimensão Z.

procedimento *acerta_vizinhaça*(*patch_ref*, vizinho : ponteiro para *info_patch*;
bordo_ref : bordo)
inicio
caso (bordo_ref) **seja**
cima: *patch_ref*→*patch*_acima = vizinho;
baixo: *patch_ref*→*patch*_abaixo = vizinho;
direita: *patch_ref*→*patch*_a_direita = vizinho;
esquerda:*patch_ref*→*patch*_a_esquerda = vizinho;
fimcaso;
fim
Procedimento 3.11 Acerta vizinhança.

A função *bordo_em_comum* (procedimento 3.10) retorna o valor lógico *verdadeiro* caso o bordo de referência (bordo_ref) de *patch_ref* também seja bordo do *patch* apontado por pz→*patch*_3D.

A implementação da versão quadrática (procedimento 3.12) pode ser feita utilizando os tipos **objeto_q** e **patch_q** (figura 3.23) e, como estrutura de dados, uma lista encadeada com estrutura recursiva de aninhamento [TRE 85] (figuras 3.22 e 3.23).

tipo objeto_q:
identificação do objeto: identificador; { ID }
primeiro_subobjeto: ponteiro para objeto_q; { SUBOB - prim. da lista de subobjetos }
primeiro_*patch*: ponteiro para *patch_q*; { PATCH - prim. da lista de *patches* do objeto }
proximo_objeto: ponteiro para objeto_q; { PROX - prox. na lista de objetos }

tipo patch_q:
X: ponteiro para matriz[4,4] de reais; { coordenadas X }
Y: ponteiro para matriz[4,4] de reais; { coordenadas Y }
Z: ponteiro para matriz[4,4] de reais; { coordenadas Z }

*patch*_correspondente : ponteiro para *patch_q*; { PC - *patch* de O' no qual este *patch* sera interpolado }

*patch*_acima : ponteiro para *patch_q*; { PA - vizinho acima }

*patch*_abaixo : ponteiro para *patch_q*; { PB - vizinho abaixo }

*patch*_a_direita : ponteiro para *patch_q*; { PD - vizinho direita }

*patch*_a_esquerda : ponteiro para *patch_q*; { PE - vizinho esquerda }

proximo_*patch* : ponteiro para *patch_q*; { PP - prox. na lista de *patches* }

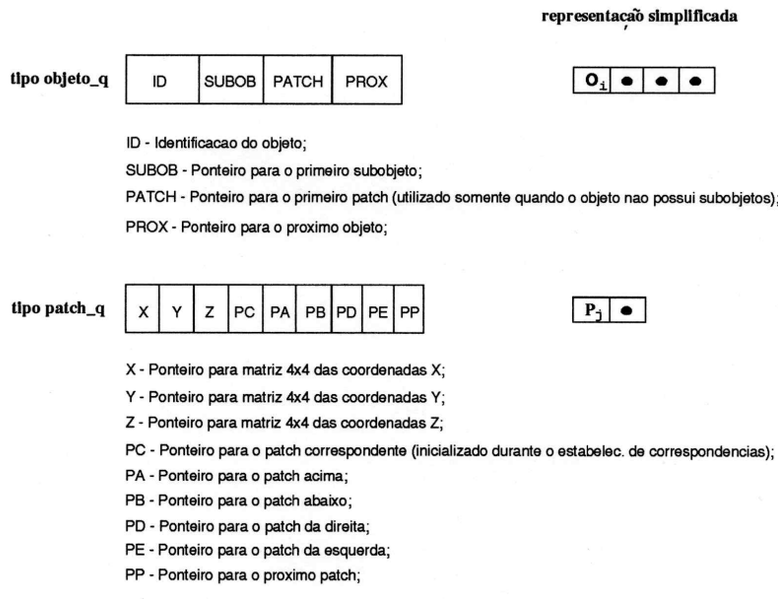

Figura 3.22: Tipos Objeto_q e *Patch_q*.

procedimento *identifica_relações_de_incidência* (O:objeto_q)

inicio

patche_ref, patch_aux: *patch_q*;

incidencia: conjunto{cima, baixo, direita, esquerda};

patch_ref := *primeiro_patch_do_objeto* (O);

enquanto (*patch_ref* $\neq$ **NULO**) **faça**

inicio

patch_aux := *proximo_de* (*patch_ref*);

enquanto (*patch_aux* $\neq$ **NULO**) **faça**

inicio

se (*possuem_aresta_em_comum* (*patch_ref,patch_aux*, incidencia)) **então**

define_incidência_comum (*patch_ref,patch_aux*, incidencia);

patch_aux := *proximo_de* (*patch_aux*);

fim;

patch_ref := *proximo_de* (*patch_ref*);

fim;

fim

Procedimento 3.12 Identificação das relações de incidência - versão quadrática.

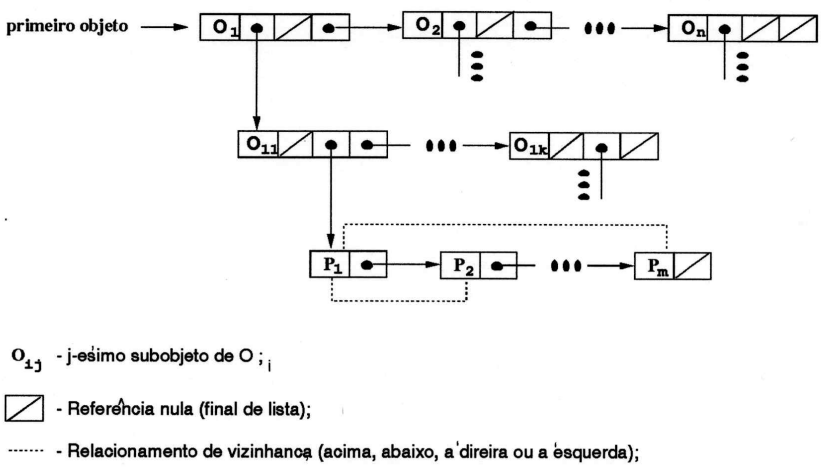

Figura 3.23: Estrutura de dados para a versão quadrática: lista encadeada com estrutura recursiva de aninhamento.

A chamada de *possuem_aresta_em_comum* (procedimento 3.11) retorna o valor lógico *verdadeiro* se *patch_ref* e *patch_aux* compartilharem alguma aresta. Neste caso, *incidência* conterá a indicação dos bordos comuns a *patch_ref* e *patch_aux*, relativamente a *patch_ref*. Já o procedimento *define_incidência_comum* atualiza os ponteiros de vizinhança entre os *patches* (*patch*_acima, *patch*_abaixo, *patch*_a_direita, *patch*_a_esquerda - tipo **patch_q**) de acordo com a informação retornada em *incidência*.

As demais fases do **AIF** serão descritas de modo independente da abordagem selecionada para o procedimento de identificação das relações de incidência

(logrítimica ou quadrática) e, conseqüentemente, das estruturas de dados a elas associadas.

Observe-se que a manutenção de um procedimento de ordem quadrática, apesar da existência de um outro, logarítmico, justifica-se pelo fato de muitos objetos estruturados serem constituídos por subobjetos que apresentam menos de vinte e quatro *patches* (p.e., bule de Utah, tabela 3.1). Para estes casos, a versão quadrática se mostra mais vantajosa. A decisão em favor de uma das abordagens fica, entretanto, a critério do implementador.

3.5.3.2 Equacionamento do número de *patches*

Sejam O e O' dois objetos a serem interpolados. Durante a etapa de equacionamento do número de *patches*, pode-se encontrar as seguintes situações:

1. O e O' são homeomorfos;
2. O e O' não são homeomorfos entre si, sendo, entretanto, ambos homeomorfos a certos espaços modelos (p.e., disco, esfera, toro, bitoro, etc.);
3. O e O' não são homeomorfos, sendo pelo menos um deles não homeomorfo a nenhum dos espaços modelos citados no ítem 2.

De modo a evitar repetições tediosas, os casos 1 e 2 serão abordados em conjunto. Para tanto, abolir-se-á, momentaneamente, as rotulações e orientações de arestas, de modo a se obter modelos mais abstratos. Assim, por exemplo, para os objetos descritos na figura 3.11, ter-se-ão as representações mostradas na figura 3.24. O ítem 3 constitui um caso especial e será tratado na seção 3.5.3.6 (casos especiais).

Observe-se que, embora ambos objetos apresentem o mesmo número de *patches*, suas representações divergem quanto aos números de linhas e colunas. Estes podem ser equacionados, por exemplo, como mostrado na figura 3.25. A partir de tal equacionamento é possível definir $g = \cup f_i$, com $f_i : a'_i \rightarrow b'_i$.

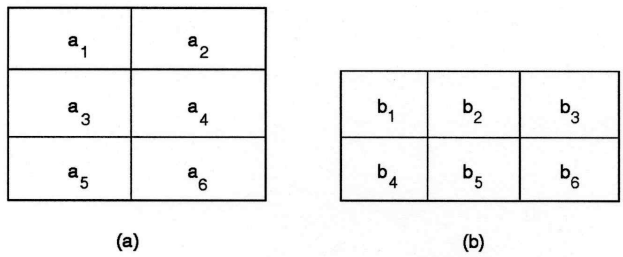

Figura 3.24: Representação da figura 3.8 sem rotulação de arestas.

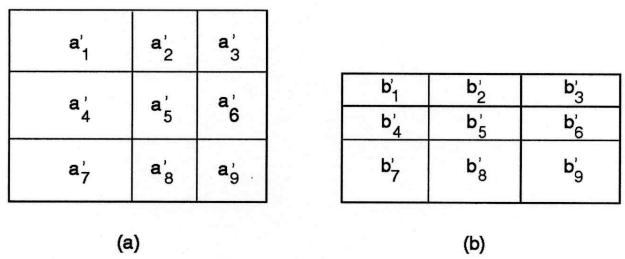

Figura 3.25: Equacionamento do número de *patches* para as representações da figura 3.24.

O procedimento para equacionar o número de *patches* pode ser dividido em duas etapas: um equacionamento do número de linhas (equacionamento vertical) e um equacionamento do número de colunas (equacionamento horizontal). O equacionamento vertical (horizontal) tem por referência o número de *patches* do objeto que apresenta o maior número de linhas (colunas).

3.5.3.3 Escolha dos *centros de malha*

Antes de iniciar efetivamente o equacionamento, elege-se, em cada objeto, um *patch* de referência. Este deve ser, preferencialmente, central, de modo a que as eventuais subdivisões necessárias possam ser regularmente distribuídas, a partir do *patch* de referência, ao longo do objeto, conforme será descrito nas duas próximas seções. A um *patch* assim escolhido, convencionou-se chamá-lo *centro de malha* e indicá-lo por CM.

função *determina_CM* (O: objeto): *patch*
inicio
altura, linha_CM, coluna_CM : inteiro;
$\bar{x}$, Z : real;
$\bar{x} := 0$; { $\bar{x}$ da linha_CM }
coordenada_z := ∞;
altura := *altura_objeto* (O); { em *patches* }
linha_CM := *inteiro_mais_proximo* (altura/2 + 0.4);
para cada *patch* p_i na linha_CM **faça**
$\bar{x} := \bar{x}$ + *somatorio_das_coordenadas_x_de* (p_i)/16;
coluna_CM := *coluna* $\in$ *linha_CM_mais_proximo_ao_ponto_XZ*($\bar{x}$,Z);
retorne (*patch* (linha_CM, coluna_CM));
fim

Procedimento 3.13 Determinação do *centro de malha* (CM).

O procedimento 3.13 ilustra uma possível função para determinar centros de malha. Outras podem, entretando, ser utilizadas. O uso de Z com valor ∞ evita que as superfícies de objetos que apresentem simetria radial em relação aos eixos X

ou Z, sofram interpenetrações desnecessárias durante a transformação. Tal valor de coordenada é relativo ao sistema de referência onde será realizada a transformação.

3.5.3.4 Equacionamento do número de linhas

Sejam, portanto, O e O' dois objetos e sejam CM e CM' seus respectivos centros de malha. O equacionamento do número de linhas consiste em: identificar, nas colunas dos centros de malha, quantos *patches* encontram-se acima e abaixo de CM e CM'; equacionar suas quantidades e propagar as subdivisões para os lados [OLI 91a].

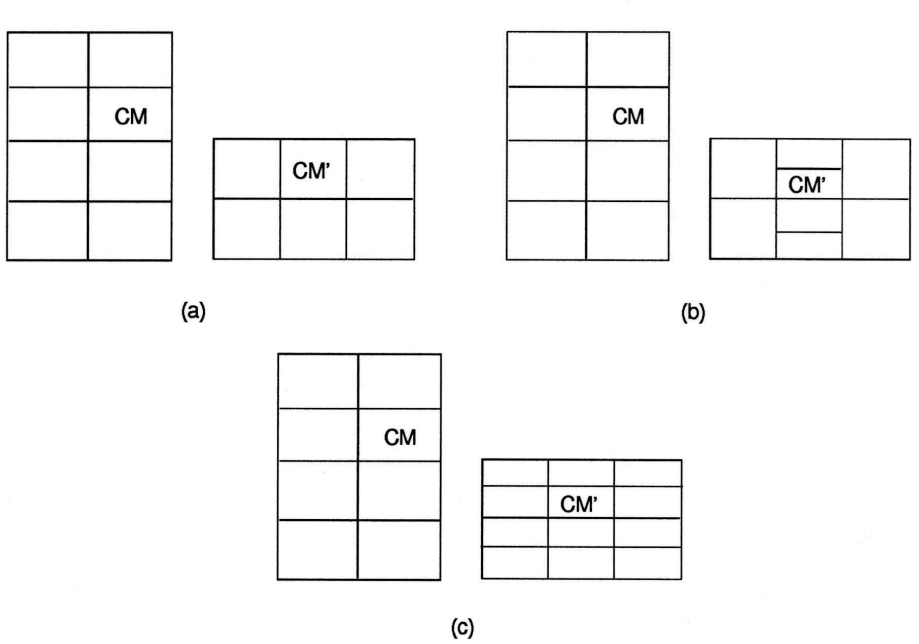

Figura 3.26: Equacionamento do número de linhas. Descrição dos objetos: (a) antes; (b) durante e (c) após.

Vê-se, no exemplo da figura 3.26, que para O tem-se um *patch* acima e dois abaixo de CM. Já em O', existe apenas um *patch* abaixo de CM'. Por meio de subdivisões adequadas, obtém-se o equacionamento do número de linhas nas colunas que contém CM e CM' (figura 3.26(b)). A seguir, todas as subdivisões são propagadas para os lados (figura 3.26(c)).

3.5.3.5 Equacionamento do número de colunas

De modo análogo ao apresentado na seção anterior, identificam-se, para as linhas que contêm os centros de malha, quantos *patches* encontram-se à esquerda e quantos à direita de CM e CM'. Após o equacionamento destes , o passo seguinte consiste em propagar as subdivisões efetuadas para cima e para baixo. Considerando as representações de O e O' resultantes da etapa anterior, ter-se-á a situação mostrada na figura 3.27 [OLI 91a].

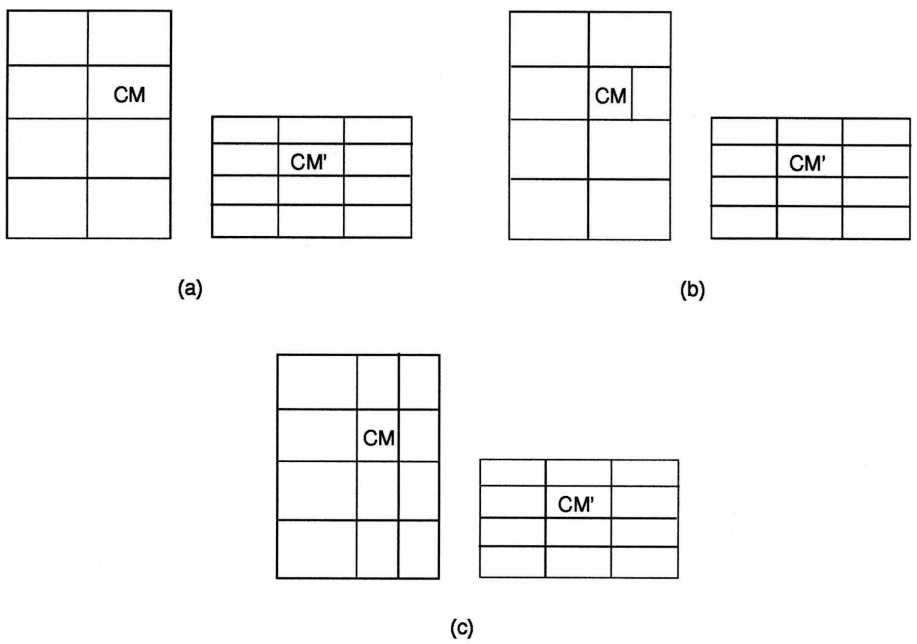

Figura 3.27: Equacionamento do número de colunas. Descrição dos objetos (a) antes; (b) durante e (c) após.

Vê-se que O apresenta apenas um *patch* à esquerda de CM, ao passo que em O' existe um *patch* à esquerda e outro à direita de CM'. Por meio de subdivisões adequadas, atinge-se o equacionamento nas linhas que contém CM e CM' (figura 3.27(b)). A seguir, realiza-se a propagação das subdivisões para cima e para baixo (figura 3.27(c)).

procedimento *equaciona_numero_de_patches* (O, O': objeto)

inicio

CM, CM' : *patch*;

CM := *determina_CM*(O);

CM':= *determina_CM*(O');

equac_vertical(O, O',CM,CM');

equac_horizontal(O, O',CM,CM');

fim

Procedimento 3.14 Equaciona número de *patches*.

procedimento *equac_vertical* (O, O':objeto; CM,CM': centro malha)

inicio

acima_CM, abaixo_CM, acima_CM', abaixo_CM‘ : inteiro;

acima_CM := *nro_patches_acima_de_CM* (O, CM);

acima_CM' := *nro_patches_acima_de_CM* (O', CM');

abaixo_CM := *nro_patches_abaixo_de_CM* (O, CM);

abaixo_CM':= *nro_patches_abaixo_de_CM* (O', CM');

se (acima_CM $>$ acima_CM') **então**

subdivide_acima_CM(O', CM', acima_CM $-$ acima_CM')

senão se(acima_CM $<$ acima_CM') **então**

subdivide_acima_CM(O, CM, acima_CM' $-$ acima_CM);

se (abaixo_CM $>$ abaixo_CM') **então**

subdivide_abaixo_CM(O', CM', abaixo_CM $-$ abaixo_CM')

senão se (abaixo_CM $<$ abaixo_CM') **então**

subdivide_abaixo_CM(O, CM, abaixo_CM' $-$ abaixo_CM);

fim

Procedimento 3.15 Equacionamento vertical.

procedimento *equac_horizontal* (O, O':objeto,CM,CM':centro malha)
inicio
esq_CM, dir_CM, esq_CM', dir_CM' : inteiro;
esq_CM := *nro_patches_esq_de_CM* (O, CM);
esq_CM':= *nro_patches_esq_de_CM* (O', CM');
dir_CM := *nro_patches_dir_de_CM* (O, CM);
dir_CM':= *nro_patches_dir_de_CM* (O', CM');
se (esq_CM > esq_CM') **então**
subdivide_esq_CM (O', CM', esq_CM − esq_CM')
senão se (esq_CM < esq_CM') **então**
subdivide_esq_CM (O, CM, esq_CM' − esq_CM);
se (dir_CM > dir_CM') **então**
subdivide_dir_CM (O', CM', dir_CM − dir_CM')
senão se (dir_CM < dir_CM') **então**
subdivide_dir_CM (O, CM, dir_CM' − dir_CM);
fim
Procedimento 3.16 Equacionamento horizontal.

procedimento *subdivide_acima_CM* (O: objeto; CM: *patch*; nro_subd: inteiro)
inicio
subd_realizadas, acima_CM, incremento, inc_altura: inteiro;
patch_ref, *patch_aux*: *patch*;
subd_realizadas = 0; *patch_ref* = CM;
se (*patch_ref*→acima ≠ **NULO**) **então**
patch_ref = *patch_ref*→acima; { poupa CM }
enquanto (nro_subd > subd_realilzadas) **faça**
inicio
acima_CM = *nro_patches_acima_CM* (O, CM);
se (acima_CM > (nro_subd - subd_realizadas)) **então**
inc_altura = nro_subd - subd_realizadas + 1
senão
inc_altura = 2; { *patch_aux* permanece como *sub_patch*

```
                    de baixo apos sua divisão }
    enquanto ( nro_subd > subd_realizadas e patch_aux ≠ NULO) faça
    inicio
      subdivide_patch (O, patch_aux, VERTICAL);
      propaga_subdivisões_para_os_lados (O, patch_aux);
      subd_realizadas = subd_realizadas + 1;
      incremento = 0;
      enquanto (incremento < inc_altura e patch_aux ≠ NULO ) faça
      inicio
        patch_aux = patch_aux→ acima;
        incremento = incremento + 1;
      fim;
    fim;
    se (patch_aux = NULO e nro_subd > subd_realizadas ) então
       patch_aux = patch_ref;
  fim;
fim
```

Procedimento 3.17: Subdivide *patches* acima do centro de malha.

Os procedimentos *subdivide_abaixo_CM*, *subdivide_dir_CM* e *subdivide_esq_CM* são análogos a *subdivide_acima_CM*. O procedimento *subdivide_patch* implementa o algoritmo de *de Casteljau*, ao passo que *propaga subdivisões para os lados* consiste de várias chamadas a *subdivide_patch*.

Sejam c, b, d e e, respectivamente, os números de *patches* localizados acima, abaixo, à direita e à esquerda de CM. As subdvisões, em quaisquer destas quatro direções, são regularmente distribuídas a partir do centro de malha (figura 3.28). Assim, seja, por exemplo, nc o número de subdvisões a serem realizadas acima de CM. O procedimento 3.16 garante que tais subdivisões sejam regularmente distribuídas (figura 3.29).

Observe-se que o procedimento descrito não considera as dimensões dos *patches* envolvidos. Entretanto, as subdivisões são regularmente distribuídas a partir

dos centros de malhas, garantindo que as transformações se realizem de maneira bastante suave. O uso de um critério para distribuir as subdivisões, que leve em conta as dimensões dos *patches*, possibilita a obtenção de resultados ainda mais agradáveis. Entretanto, isto implica um maior custo computacional, decorrente do cálculo da área de cada um dos *patches* padrões.

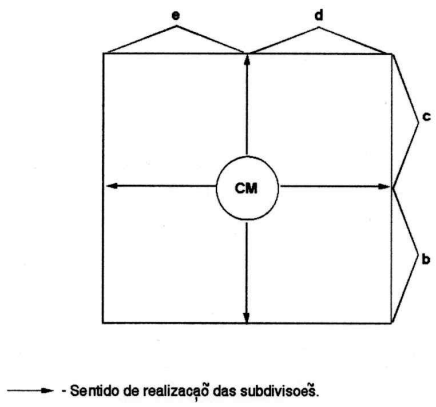

Figura 3.28: Sentido de realização das subdivisões.

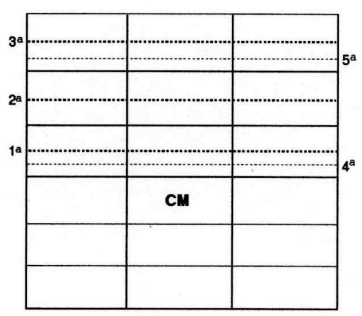

Figura 3.29: Subdivisões realizadas pelo procedimento 3.16 para $c = 3$, $nc = 5$.

A partir do equacionamento do número de *patches* assim obtido, pode-se estabelecer uma correspondência biunívoca, entre pares de *patches* de O e O',

respeitando-se suas relações de incidência, bem entre seus respectivos coeficientes geométricos. Após tal definição, resta interpolar os coeficientes associados.

3.5.3.6 Casos especiais

Casos especiais ocorrem quando os objetos apresentam uma descontinuidade na justaposição dos *patches* que os definem, de tal modo que se torna bastante difícil obter representações isomorfas, apenas subdividindo seus *patches* (figuras 3.30 e 3.31).

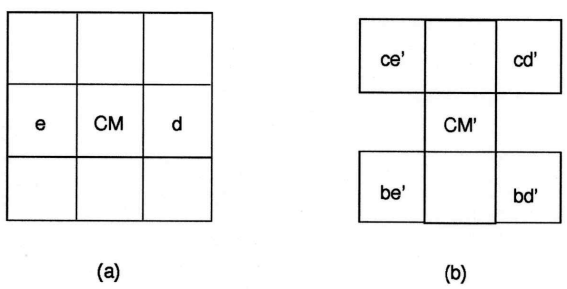

Figura 3.30: Exemplo de caso especial.

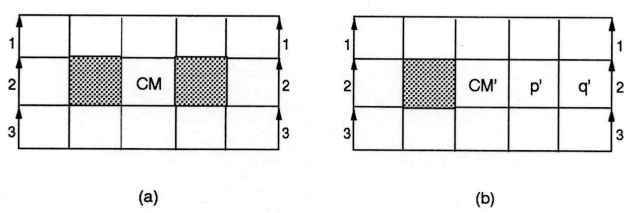

Figura 3.31: Caso especial: as áreas sombreadas correspondem a discos que foram extraídos das superfícies cilíndricas originais.

As representações da figura 3.30 admitem uma homotopia $f : (a) \rightarrow (b)$, com os *patches* e e d (figura 3.30(a)) sendo mapeados, respectivamente, para as

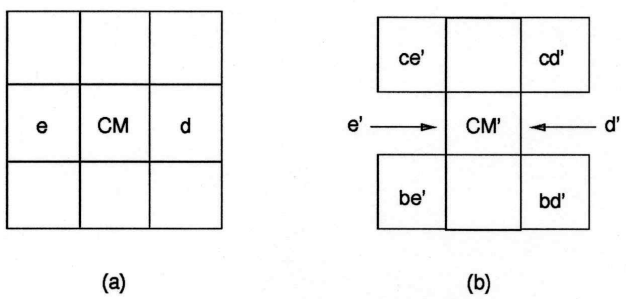

Figura 3.32: Equacionamento por inclusão de *patches* degenerados.

arestas esquerda e direita de CM' (figura 3.30(b)). De modo a tratar uniformemente todos os casos, equaciona-se o número de *patches* por meio da inclusão de novos *patches* degenerados nas arestas esquerda e direita de CM', conforme mostrado na figura 3.32. Note-se que, como os *patches* inseridos foram degenerados nas fronteiras de outros pré-existentes, tem-se que: as geometrias dos objetos permaneceram inalteradas; a homotopia $f_i : (a) \rightarrow (b)$ continua existindo; os números de *patches* foram equacionados e os *patches ce'* e *cd'* passam a apresentar vizinhança abaixo, ao passo que *be'* e *bd'*, vizinhança acima. A partir de então, pode-se estabelecer uma correspondência biunívoca entre pares de *patches* de O e O' e interpolar os coeficientes geométricos associados.

O caso mostrado na figura 3.31 apresenta solução semelhante à ilustrada na figura 3.32. No primeiro, existe a homotopia $f_i : (b) \rightarrow (a)$ (figura 3.31), no qual o *patch* localizado à direita de CM' (p') poderá ser degenerado na aresta esquerda de q' ou, na aresta direita de CM'.

3.5.3.7 Regra para subdivisão de arestas

Durante as duas últimas seções, suprimiu-se as ilustrações de rótulos e orientações das arestas que representam os bordos dos *patches* envolvidos, de modo a produzir representações menos carregadas e poder-se concentrar apenas nos aspectos concernentes ao método. Retorna-se, agora, ao uso de rótulos e orientações, introduzindo a seguinte regra para subdivisão de arestas:

REGRA 3.1 Caso a aresta a ser subdividida apresente rótulo, sua orientação é preservada em cada um dos segmentos gerados, os quais recebem rótulos distintos, de modo a evidenciar que representam entidades diferentes.

Portanto, ao se subdividir uma aresta, a mesma é substituída pelas respectivas sub-arestas recém-criadas, as quais preservam a orientação da aresta original. Assim, sejam O e O' um cilindro e um toro, respectivamente, os objetos ilustrados na figura 3.26. Suas representações, antes e após a etapa de equacionamento, são mostradas na figura 3.33.

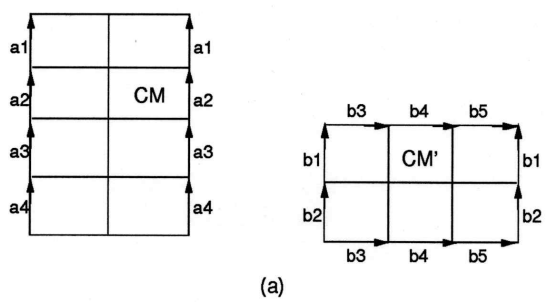

Figura 3.33: Decomposições celulares de uma superfície cilíndrica e de um toro: (a) antes e (b) depois do equacionamento do número de *patches*.

3.5.3.8 Estabelecimento das correspondências entre pares de *patches*

Uma vez realizado o equacionamento do número de *patches*, a função $f_i : p_i \rightarrow p'_i$, que associa a cada *patch* p_i do objeto O um *patch* p'_i de O', pode ser definida por meio do seguinte procedimento recursivo, com chamada inicial *correspondência*(CM,CM',REFERÊNCIA_VAZIA).

procedimento *correspondência* (p_i, p'_i : *patch*; bordo_referência: bordo)
inicio
  se ($p'_i \neq$ **NULO**) **então**
  inicio
    $p_i \rightarrow$*patch*_correspondente := p'_i; $p'_i \rightarrow$*patch*_correspondente := p_i;
    insere_na_lista_de_visitados (p_i);
    se ($p_i \rightarrow$cima $\neq$ **NULO e** *não_visitado* ($p_i \rightarrow$cima)) **então**
      correspondência ($p_i \rightarrow$cima, $p'_i \rightarrow$ cima, cima);
    se ($p_i \rightarrow$baixo $\neq$ **NULO e** *não_visitado* ($p_i \rightarrow$baixo) **então**
      correspondência ($p_i \rightarrow$baixo, $p'_i \rightarrow$baixo, baixo);
    se ($p_i \rightarrow$esquerda $\neq$ **NULO e** *não_visitado* ($p_i \rightarrow$esquerda) **então**
      correspondência ($p_i \rightarrow$esquerda, $p'_i \rightarrow$esquerda, esquerda);
    se ($p_i \rightarrow$direita $\neq$ **NULO e** *não_visitado* ($p_i \rightarrow$direita)) **então**
      correspondência ($p_i \rightarrow$direita, $p'_i \rightarrow$direita, direita);
  fim
  senão
  inicio
    caso (bordo_referência)
      cima: p'_i = *inclui_patch_degenerado* ($p_i \rightarrow$baixo.corresp, cima);
      baixo: p'_i = *inclui_patch_degenerado* ($p_i \rightarrow$cima.corresp, baixo);
      esquerda: p'_i = *inclui_patch_degenerado* ($p_i \rightarrow$direita.corresp, esq);
      direita: p'_i = *inclui_patch_degenerado* ($p \rightarrow$esquerda.corresp, dir);
    fimcaso;
    correspondência (p_i, p'_i, REFERÊNCIA_VAZIA);
  fim;
fim

Procedimento 3.18 Estabelecimento de correspondências entre *patches*.

O termo REFERÊNCIA_VAZIA sugere que quando p_i e p'_i forem ambos diferentes de NULO, as ações realizadas pelo procedimento independem do valor do parâmetro *bordo_referência*. O procedimento *não_visitado*(p_k) retorna o valor lógico *verdadeiro* caso p_k ainda não esteja presente na *lista_de_patches_visitados*. A função

inclui_patch_degenerado(patch_ref,bordo), insere um *patch* degenerado no bordo *bordo* de *patch_ref*, devolvendo a identificação do *patch* inserido.

A apresentação deste procedimento encerra a descrição do **Algoritmo Interpolador de Formas**, uma vez que a interpolação dos coeficientes associados dispensa esclarecimentos. No capítulo 5 (análise dos resultados), serão feitas considerações acerca da utilização de trajetórias lineares x não-lineares na obtenção de diferentes efeitos de interpolação.

3.5.4 Ordem de complexidade do algoritmo

A aplicação $g : O \rightarrow O'$ que transforma O em O' foi descrita por meio dos seguintes passos:

1. converter os objetos para a representação padrão;
2. identificar as relações de incidência entre os *patches* de cada um dos objetos;
3. determinar CM e CM', respectivamente, os centros de malha de O e O';
4. equacionar, relativamente a CM e CM', os números de *patches* nas representações de O e O';
5. estabelecer correspondências entre pares de *patches* (p_i, p'_i) de O e O';
6. associar pares de coeficientes geométricos (c_{ij}, c'_{ij}) de p_i e p'_i e
7. interpolar os pares de coeficientes geométricos (c_{ij}, c'_{ij}) associados.

A seguir, serão analisadas cada uma das etapas identificadas acima, relativamente às suas complexidades.

3.5.4.1 Conversão dos objetos para a representação padrão

Sejam n e n', respectivamente, os números de *patches* padrões (Bézier) dos objetos O e O' antes da etapa de equacionamento. Como a conversão de cada

patch é obtida em tempo constante (produto matricial [MOR 85]), o custo desta fase é da ordem de $n+n'$, ou seja, linear em relação à soma dos números de *patches* originais.

3.5.4.2 Identificação das relações de incidência

Bastante explorada na seção 3.5.3.1, esta fase apresenta complexidade de $O(n \log n)$.

3.5.4.3 Equacionamento do número de *patches*

Sejam k_s e k_i ($k_s = f_s(n, n')$ e $k_i = f_i(n, n')$) constantes não negativas, $s = k_s nn'$ a quantidade de subdivisões a realizar e $i = k_i nn'$ o número de *patches* degenerados a inserir. O custo para determinação de CM e CM' (procedimento 3.3) varia de modo linear em relação a n e n', respectivamente. Como a subdivisão de uma superfície Bézier (seção 2.3.1) é obtida em tempo constante e também o é a inclusão de um novo *patch*, o custo da fase de equacionamento é da ordem de $s+i$, ou seja, proporcional ao produto nn'. Mais precisamente, $(k_s + k_i)nn'$, onde, com freqüência, $(k_s + k_i) < 1$.

3.5.4.4 Estabelecimento de correspondências entre pares de *patches*

Seja $N = (n+n'+s+i)/2$ o número de *patches* que compõem cada uma das descrições de O e O' após o equacionamento. Como o procedimento 3.18 sempre pára (N é finito) e todos os *patches* de O e O' são visitados (O e O' são conexos), o procedimento *correspondência* apresenta complexidade da ordem de $s+i$, ou seja, proporcional ao produto nn'.

3.5.4.5 Estabelecimento de correspondências entre coeficientes geométricos

O custo envolvido no estabelecimento das correspondências entre todos os coeficientes geométricos de O e todos de O' é $16N$ (cada *patch* é constituído por 16 pontos de controle), ou seja, proporcional ao produto nn'.

3.5.4.6 Custo total do algoritmo

O custo total do algoritmo é influenciado pela fase de *equacionamento do número de patches*, apresentando custo de $(k_s + k_i)nn'$, ou seja $O(nn')$.

3.5.5 Visualização de transformações descontínuas

O que fazer quando se deseja obter uma transformação não definida continuamente, por exemplo, de uma superfície cilíndrica em uma superfície plana ? De acordo com a seção 3.2.2, não existe uma função g que realize tal transformação. Caso se deseje produzir uma seqüência de imagens que a ilustre, faz-se necessário definir $g = \cup r_i$, sendo cada $r_i : p_i \rightarrow p'_i$ uma relação que mapeia um *patch* $p_i \in O$ em seu correspondente $p'_i \in O'$. Deste modo, relaxando as exigências de continuidade $r_i \mid p_i \cap p_j = r_j \mid p_i \cap p_j$ e $r_i(p_i) = r_j(p_j)$ (caso $p_i \equiv p_j$), para $i \neq j$, as diversas formas intermediárias que ilustram transformações descontínuas podem ser obtidas por meio de interpolação dos coefientes geométricos de p_i e p'_i. Este recurso é válido, uma vez que uma descontinuidade matemática não implica, necessariamente, uma descontinuidade visual entre quadros consecutivos e, portanto, pode ser explorada esteticamente.

3.6 Resumo

Este capítulo apresentou um algoritmo de $O(nn')$ para interpolação de formas entre objetos modelados por superfícies *spline*. Foram descritas três versões: uma hipotética (*algoritmo ideal*), uma topológica e outra geométrica. Esta estratégia

de exposição visou tornar claro o formalismo topológico que garante a validade e serve como critério de correção para o algoritmo proposto. A passagem da versão topológica para a geométrica pode ser resumida pelas seguintes analogias: (célula, *patch* Bézier) e (subdivisão celular, algoritmo de *de Casteljau*). A escolha de superfícies Bézier como base para descrição do algoritmo deveu-se aos seguintes pontos:

- necessidade de se utilizar uma representação uniforme;
- possibilidade de subdivisão preservando a forma original e
- facilidade para conversão a partir de outros tipos de *patches*.

O algoritmo apresentado consiste em:

1. converter os objetos a serem interpolados para a representação padrão (Bézier bicúbica);
2. identificar as relações de incidência entre os diversos *patches* que compõem as descrições de cada um dos objetos;
3. obter, para cada objeto, seu centro de malha;
4. equacionar, relativamente a estes, os números de linhas e colunas em ambas representações;
5. estabelecer uma correspondência biunívoca entre os *patches* dos objetos, preservando suas relações de incidência;
6. definir, para cada par de *patches* associados, uma correspondência biunívoca entre seus coeficientes geométricos e
7. interpolar os coeficientes geométricos associados, gerando, deste modo, as diversas formas intermediárias.

Apesar do **AIF** não levar em consideração as dimensões dos *patches* envolvidos, seu critério para distribuição das subdivisões produz resultados bastante homogêneos.

Também foram discutidos casos especiais, apresentada uma análise da complexidade do algoritmo e feitas considerações acerca da visualização de transformações não definidas continuamente.

4 INTERPOL3D

4.1 Introdução

Este capítulo descreve um protótipo de sistema de animação modelada, INTERPOL3D [OLI 91b], que implementa uma versão simplificada do **Algoritmo Interpolador de Formas**. Aqui, serão descritas suas características e interfaces e serão comentados os resultados obtidos com sua implementação.

4.2 Características

A construção do protótipo INTERPOL3D objetivou exclusivamente a implementação do **Algoritmo Interpolador de Formas** e, portanto, algumas operações freqüentemente encontradas em sistemas de animação convencionais, como translações, escala e rotações, não se encontram disponíveis.

INTERPOL3D é parte integrante do ambiente para modelagem, visualização e animação tridimensional BEZIER4D [OLI 91c]. Permite gerar interpolações de formas e cores entre objetos modelados por superfícies Bézier, os quais podem ser constituídos por diferentes números de *patches*. Uma vez que os objetos já se encontram descritos no formato padrão (seção 3.5.1), INTERPOL3D implementa os passos 2 a 5 do Algoritmo Interpolador de Formas (seção 3.5.3). O controle de animação é explícito (veja anexo A), realizado por meio de interfaces gráficas (figuras 4.4 a 4.6). Estas permitem definir pares de objetos a serem interpolados e associar, a cada um destes, uma curva característica do comportamento da velocidade de interpolação.

Sem comprometer os objetivos de validação do algoritmo, INTERPOL3D restringe suas entradas a objetos constituídos por espaços topologicamente equivalentes a um disco ou a uma superfície cilíndrica. A partir destes, pode-se ilustrar tanto exemplos de transformações homotópicas (figura 3.4), quanto homeotópicas (figura 4.1). Embora não seja possível representar superfícies fechadas (variedade-

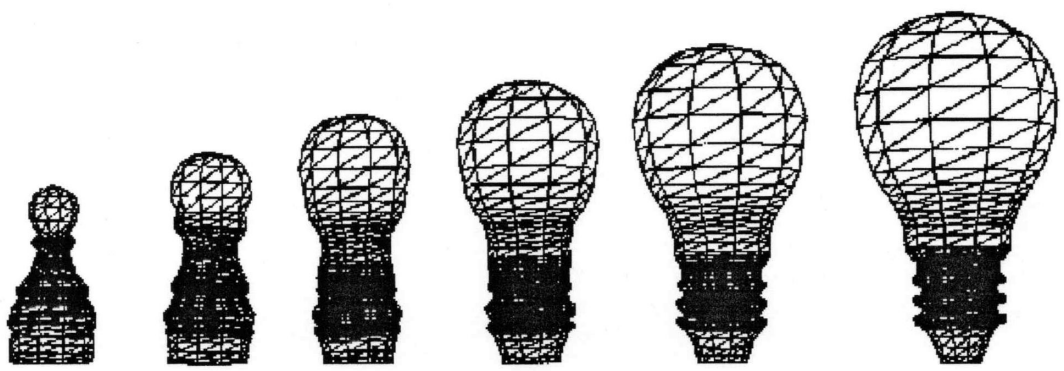

Figura 4.1: Transformação de um peão em lâmpada.

2 [MAN 88]), isto não constitui um fator restritivo à produção de seqüências animadas. *Representações* visualmente idênticas às representações variedade-2 podem ser obtidas fazendo-se com que os bordos de superfícies cilíndricas convirjam para pontos [OLI 91b] (figuras 1.1 e 4.1).

4.3 Estrutura do protótipo

INTERPOL3D constitui-se de um núcleo, representado pelo **Algoritmo Interpolador de Formas** (versão quadrática), e de três interfaces:

- especificação do número de quadros [1];
- associação de pares de objetos a interpolar e
- controle de velocidade.

Como entradas, recebe um conjunto de objetos modelados por superfícies Bézier (universo da animação) e o número de quadros a serem gerados. Sua saída corresponde à seqüência de descrições das cenas tridimensionais intermediárias (figura 4.2).

[1] O termo *quadro* não parece adequado para expressar um instante de uma animação tridimensional, visto que esta independe da posição do observador. Entretanto, será mantido em virtude de já se encontrar bastante difundido na comunidade de Computação Gráfica.

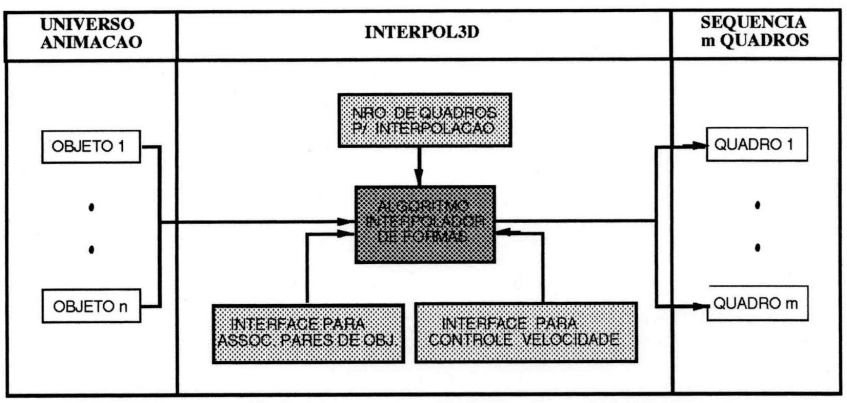

Figura 4.2: Estrutura do protótipo INTERPOL3D.

A seguir, será apresentado o universo de uma animação e serão detalhadas suas interfaces.

4.3.1 Universo de uma animação

O universo de uma animação INTERPOL3D corresponde a um universo no ambiente BEZIER4D [OLI 91c], sendo composto por objetos, os quais podem ser, recursivamente, compostos por subobjetos, até que o usuário decida pela sua concretização como um conjunto de *patches* (figura 4.3).

4.3.2 Interface para especificação do número de quadros

Esta interface permite ao usuário definir o número de quadros em que a interpolação se processará, escolher entre saída em vídeo (*preview* aramado) ou em arquivo e, neste caso, com ou sem interpolação de cores (componentes R, G e B). As duas últimas opções: saída em vídeo/arquivo e interpolação de cores *on/off*, exibem os valores dos atributos atualmente selecionados, fazendo uso de um menu dinâmico por enumeração [BAG 89] (figura 4.4). A saída em arquivo observa o

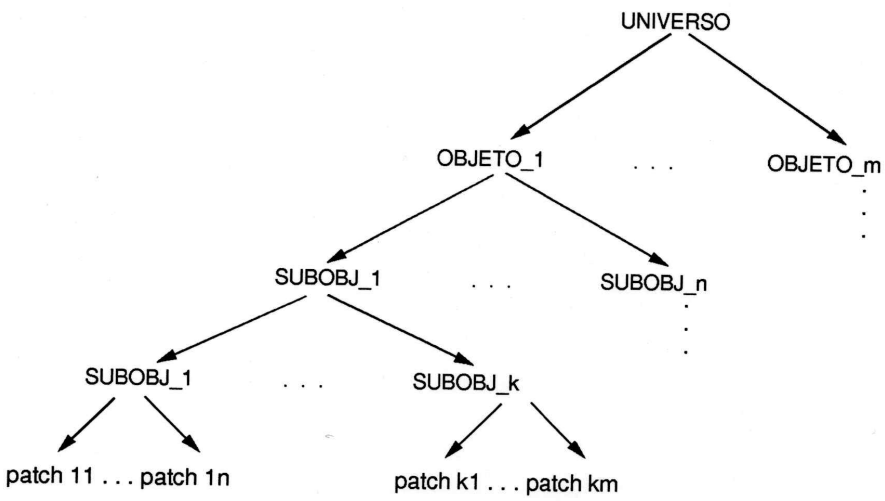

Figura 4.3: Universo de uma animação INTERPOL3D.

formato padrão BEZIER4D [OLI 91c]. Desta maneira, pode-se utilizar os recursos do ambiente tanto para obtenção de *rendering* das cenas, como para discretização das superfícies em triângulos, de modo a prover entradas para outros programas (p.e., outros sistemas de animação e visualização).

4.3.3 Interface para associação dos pares de objetos a interpolar

Tal interface permite ao usuário definir pares de objetos a serem interpolados. Consiste de um menu dinâmico (figura 4.5), onde são mostradas duas colunas, cada uma contendo a relação de todos os objetos presentes no universo. A primeira, corresponde aos candidatos a elementos de partida para interpolação, enquanto que a segunda, aos candidatos a elementos de chegada da interpolacão. Entretanto, somente objetos ditos *terminais*, ou seja, não constituídos por subobjetos (p.e., subobjeto k, figura 4.3) podem ser interpolados. Esta convenção procura evitar que se realizem transformações nas quais, as superfícies que representam cada um dos objetos não caracterizem exemplos de espaços conexos.

Figura 4.4: Interface para especificação do número de quadros.

Objetos estruturados são ditos *não terminais* (p.e., OBJETO_1, figura 4.3) e correspondem àqueles cujos retângulos apresentam um **x** (p.e., bule, figura 4.5). Estes não se encontram disponíveis para associação. Seus nomes são, entretanto, mantidos, uma vez que viabilizam o uso da notação " ../ ", mais econômica na maioria dos casos.

Objetos selecionados na coluna da esquerda são ditos *objetos origem*, ao passo que os selecionados na coluna da direita são denominados *objetos alvo*. A hierarquia entre os objetos é indicada por meio de barras invertidas (figura 4.5).

Visto que um objeto origem pode ser associado a, no máximo, um objeto alvo, ao passo que este pode estar associado a vários objetos origem (significando que ao final da interpolação ter-se-á várias instâncias do mesmo alvo), as correspondências entre pares de objetos são efetuadas da seguinte maneira:

1. O usuário seleciona, por apontamento, um objeto alvo. Neste instante, a cor a ele associada preenche o retângulo situado no alto da segunda coluna (figura 4.5), indicando que aquele se encontra selecionado;

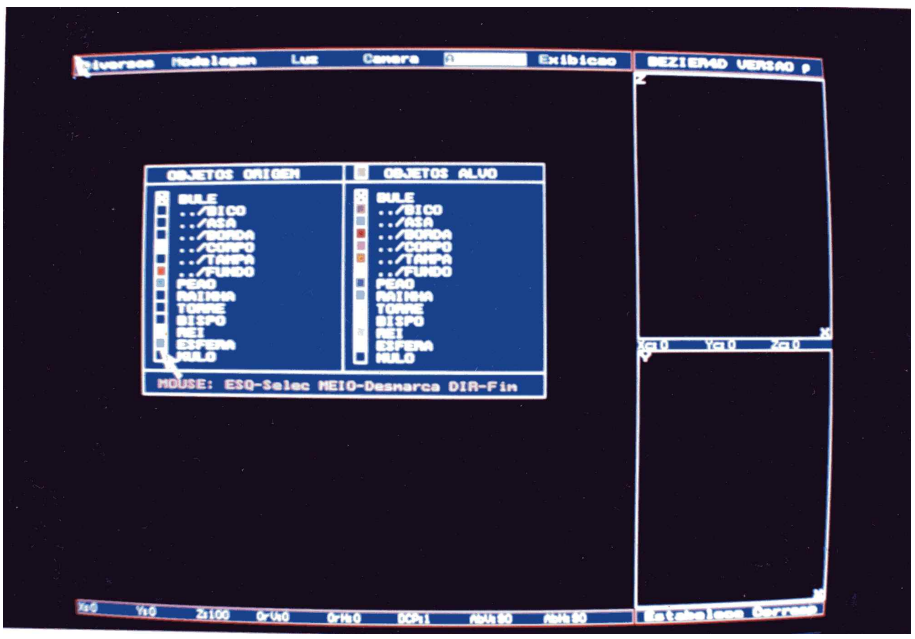

Figura 4.5: Interface para estabelecimento de correspondências

2. A seguir, o usuário deve associá-lo a um objeto na coluna da esquerda (objeto origem). Este, então, recebe a cor correspondente ao objeto no qual deve ser interpolado.

Os passos 1 e 2 podem ser repetidos até que todas as correspondências desejadas tenham sido estabelecidas.

Embora não sejam permitidas associações entre objetos estruturados, suas interpolações podem ser obtidas interpolando-se cada um de seus subobjetos terminais. Elementos presentes no universo e para os quais não se tenha estabelecido uma correspondência, permanecerão inalterados ao longo da animação, possibilitando a montagem de cenários nos quais as transformações serão realizadas (figuras 4.9 a 4.12).

4.3.4 Interface para controle de velocidade

Aos pares de objetos a serem interpolados, pode-se associar uma descrição de como se comportará a velocidade ao longo do processo de interpolação. Caso

aquela não seja especificada, a velocidade é assumida constante. Tal informação é expressa por meio de um gráfico (velocidade x quadros), no qual o usuário indica o comportamento da velocidade por meio de uma curva Bézier cúbica (figura 4.6). Embora esta apresente, no máximo, um ponto de inflexão, foi considerada suficiente para fins de implementação do protótipo, visto permitir uma variedade bastante grande de efeitos. Se desejado, a interface pode ser facilmente estendida, de modo a utilizar composições de curvas Bézier (mantendo a homogeneidade do ambiente) ou, ainda, outras curvas paramétricas.

O controle sobre a variação de velocidade é realizado definindo-se, interativamente, uma curva característica para cada um dos pares de objetos associados. Simultaneamente à definição da curva, a interface exibe o número do quadro em que a transformação será concluída (figura 4.6). Tal informação corresponde à abscissa do último coeficiente geométrico da curva Bézier, relativamente ao comprimento do eixo dos quadros, onde esse comprimento encontra-se associado ao número especificado para o total de cenas. Isto permite ao usuário a possibilidade de indicar que pares de objetos tenham suas interpolações concluídas em diferentes instantes da animação.

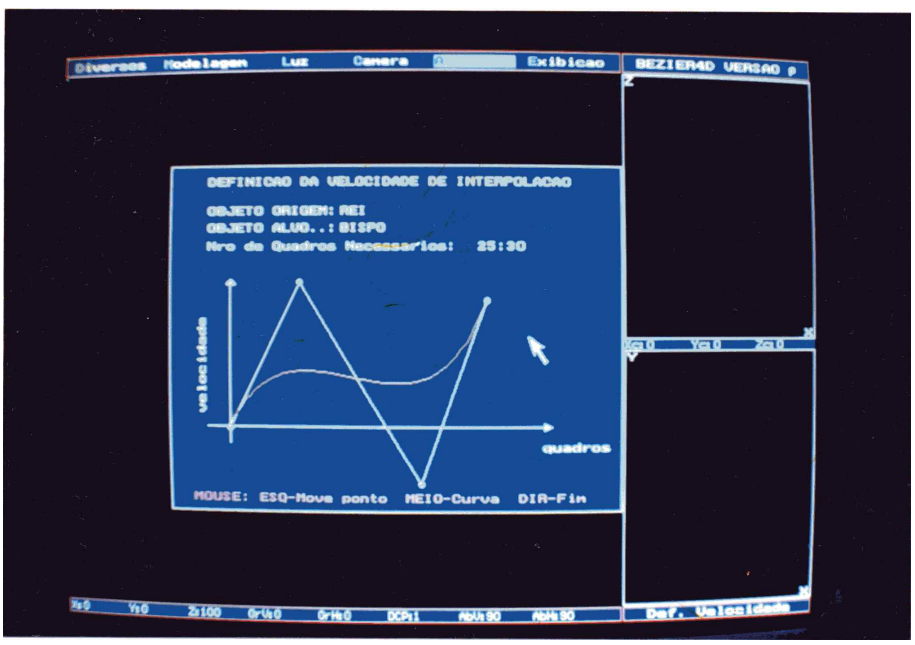

Figura 4.6: Interface para definição do comportamento da velociade de interpolação.

O sistema estabelece como restrição que os coeficientes geométricos da curva apresentem abscissas crescentes, de modo que a curva represente uma função velocidade.

4.3.5 Cálculo das posições

Uma vez definidas as curvas de velocidade, calcula-se, para cada par de objetos a interpolar, $\Delta S = \sum_{i=0}^{a} \Delta s_i - \sum_{j=0}^{b} \Delta s_j$, onde: Δs_i corresponde à área do i-ésimo segmento de curva Bézier apresentando ordenadas positivas e Δs_j corresponde à área do j-ésimo segmento de curva Bézier apresentando ordenadas negativas (figura 4.7). O cálculo destas áreas pode ser realizado encontrando-se a representação algébrica (polinômio) da curva a partir de sua representação geométrica (pontos de controle) [MOR 85]. De posse do polinômio, os valores de áreas são obtidos por meio de integração paramétrica [GRA 41]. O eixo dos quadros é, então, dividido em $n-1$ segmentos iguais, cada um correspondendo a um dos quadros a ser gerado (figura 4.8). O primeiro quadro corresponde à cena original modelada pelo usuário. Como, em última instância, o algoritmo interpola pontos de controle de *patches* correspondentes, o processo de variação de velocidade é realizado da seguinte forma:

1. Calcula-se, para cada quadro, sua contribuição relativa **rq** para a área total ΔS referente ao par a ser interpolado; $rq = s\Delta q/\Delta S$, onde Δq representa a área do elemento de área correspondente ao quadro q (figura 4.7), e $s = 1$ se o elemento de área encontra-se acima do eixo dos quadros. Do contrário, $s = -1$. Caso o elemento de área contenha alguma das raízes da equação $f(x) = 0$ (curva de velocidade), sua contribuição relativa é dada pela soma das contribuições de cada um de seus sub-elementos localizados integralmente acima ou abaixo do eixo;

2. A cada quadro, os coeficientes geométricos do *patch* p_i (objeto origem) são incrementados de $rq\Delta p_i$, onde $\Delta p_i(\Delta p_{ix}, \Delta p_{iy}, \Delta p_{iz})$ corresponde às matrizes

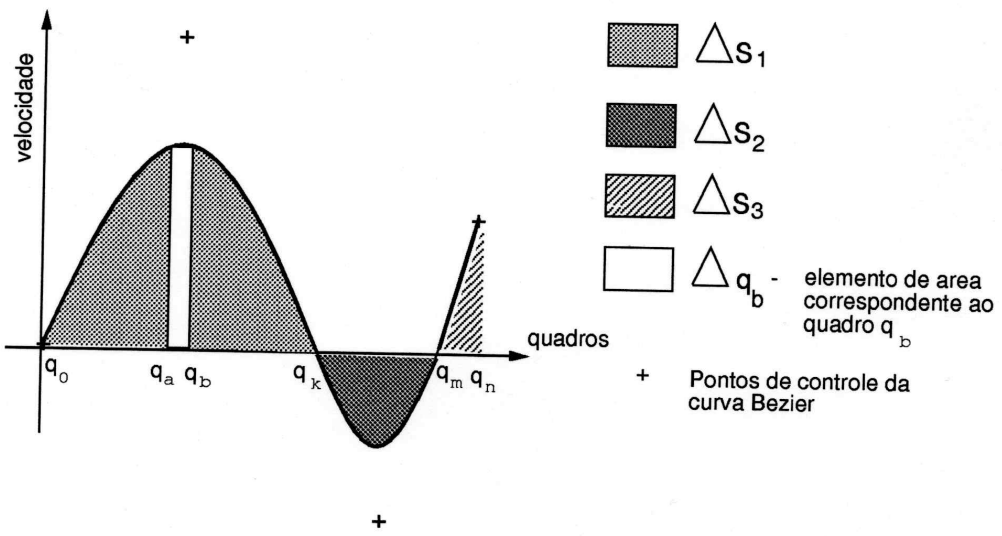

Figura 4.7: Áreas correspondentes aos segmentos da curva Bézier.

4x4 das diferenças entre os valores de coordenadas $(\Delta x, \Delta y, \Delta z)$ de $p_{ialvo(x,y,z)} - p_{iorigem(x,y,z)}$.

Assim, a cada novo quadro, os coeficientes geométricos de cada *patch* p_i de um objeto origem são incrementados de $(\delta x_{ij}, \delta y_{ij}, \delta z_{ij})$, proporcionais a Δp_i e à área relativa do elemento de área correspondente ao quadro. A contribuição relativa de um quadro q para ΔS pode variar entre os diversos pares de objetos a transformar, uma vez que cada transformação pode apresentar uma curva particular de velocidade e, portanto, um ΔS e Δqs específicos.

Observe-se que, ao longo da interpolação, a velocidade pode assumir valores negativos (figura 4.7). Esta característica pode ser utilizada para inverter o *sentido* da interpolação durante o processo. De modo a simplificar sua exposição, considere-se apenas o seguinte par de pontos a ser interpolado: $p_{origem}(0,0,0)$ e $p_{alvo}(10,10,10)$. Deste modo, o elemento de Δp_i correspondente à diferença entre os pontos é dado por $\delta p_i = (10,10,10)$. Considere-se a curva apresentada na figura 4.7

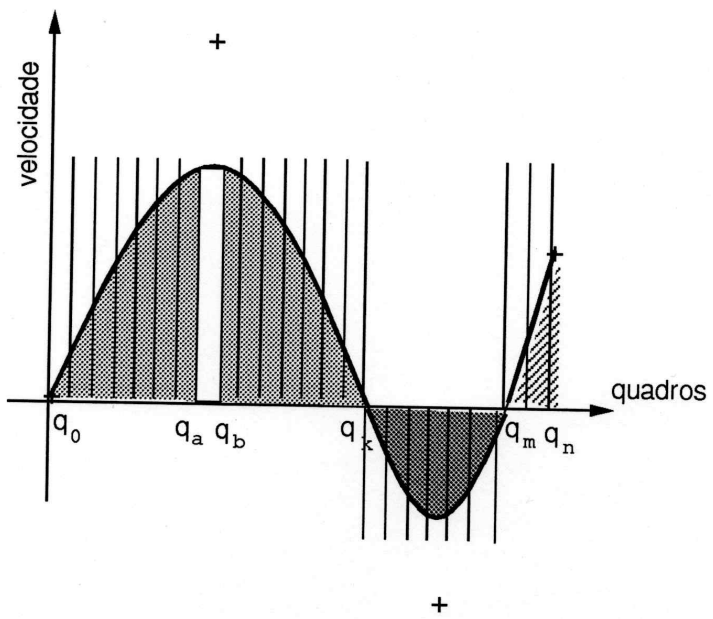

Figura 4.8: Delimitação das áreas correspondentes a cada quadro.

e suponha que $\Delta s_1 = 11, \Delta s_2 = 2$ e $\Delta s_3 = 1$. Neste caso, serão os seguintes os valores de coordenadas associados a instantes da interpolação:

$q_i : P_i = P_0 + (\sum_{j=1}^{i} s_j \Delta q_j)\delta p_i / \Delta S$

$q_0 : P_0 = P_0 + (0)10/10 = (0, 0, 0)$
$q_k : P_k = P_0 + (11)10/10 = (11, 11, 11)$
$q_m : P_m = P_0 + (9)10/10 = (9, 9, 9)$
$q_n : P_n = P_0 + (10)10/10 = (10, 10, 10)$

A interpolação de cores é realizada de forma análoga. Utilizando o procedimento descrito, interpola-se $cor_{origem}(R, G, B)$ e $cor_{alvo}(R, G, B)$, obtendo-se as diversas cores intermediárias *sincronizadas* com as formas correspondentes.

4.4 Obtenção dos valores de ΔS e Δq

Esta seção descreve a técnica implementada para obtenção dos valores de área delimitados pelos diversos segmentos da curva Bézier. O procedimento para cálculo de ΔS consiste em:

- obter o polinômio que descreve a curva, a partir de seus coeficientes geométricos [MOR 85];
- identificar os valores do parâmetro para os quais a curva corta o eixo dos quadros e
- a partir desses, por meio de integração paramétrica [GRA 41], obter-se os valores de área delimitados pelos diversos segmentos da curva.

Já o cálculo de Δq consiste de:

- obter, a partir do polinômio encontrado durante o cálculo de ΔS, pela aplicação do método de Newton [CAR 69], os valores v_{i-1} e v_i do parâmetro v, para os quais $x_{i-1} = q_{i-1}$ e $x_i = q_i$ (figura 4.8);
- a partir de v_{i-1} e v_i, por meio de integração paramétrica, obter a área do elemento de área correspondente, sempre observando a existência de raízes para $y(v) = 0$ (4.12) no intervalo (x_{i-1},x_i).

4.4.1 Obtenção do polinômio a partir dos coeficientes geométricos

Seja $C(v)$ uma curva Bézier

$$C(v) = VMG \tag{4.1}$$

onde,

$$V = \left[\begin{array}{cccc} v^3 & v^2 & v & 1 \end{array} \right] \tag{4.2}$$

$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 3 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.3}$$

$$G = \begin{bmatrix} G_1 & G_2 & G_3 & G_4 \end{bmatrix}^T \tag{4.4}$$

representando-a como um polinômio cúbico, tem-se:

$$C(v) = a_3 v^3 + a_2 v^2 + a_1 v + a_0 \tag{4.5}$$

ou

$$C(v) = VA \tag{4.6}$$

com

$$A = \begin{bmatrix} a_3 & a_2 & a_1 & a_0 \end{bmatrix}^T \tag{4.7}$$

Igualando 4.1 e 4.6:

$$VA = VMG \tag{4.8}$$

e, portanto

$$A = MG \tag{4.9}$$

Substituindo A, M e G:

$$\begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 3 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

$$\begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} -G_1 + 3G_2 - 3G_3 + G_4 \\ 3G_1 - 6G_2 + 3G_3 \\ -3G_1 + 3G_2 \\ G_1 \end{bmatrix} \tag{4.10}$$

Sejam

$$x(v) = a_{3x}v^3 + a_{2x}v^2 + a_{1x}v + a_{0x} \tag{4.11}$$

$$y(v) = a_{3y}v^3 + a_{2y}v^2 + a_{1y}v + a_{0y} \tag{4.12}$$

componentes paramétricas da curva $C(v)$. Assim, tem-se:

$$\begin{aligned} x(v) &= (-G_1(x) + 3G_2(x) - 3G_3(x) + G_4(x))v^3 + (3G_1(x) - 6G_2(x) + 3G_3(x))v^2 + \\ &\quad (-3G_1(x) + 3G_2(x))v + G_1(x) \end{aligned}$$

$$\begin{aligned} y(v) &= (-G_1(y) + 3G_2(y) - 3G_3(y) + G_4(y))v^3 + (3G_1(y) - 6G_2(y) + 3G_3(y))v^2 + \\ &\quad (-3G_1(y) + 3G_2(y))v + G_1(y) \end{aligned}$$

onde $G_i(x)$ e $G_i(y)$ correspondem respectivamente às coordenadas x e y do coeficiente geométrico G_i.

4.4.1.1 Cálculo da área sob a curva (ΔS)

Sejam r_1, r_2 e r_3 os valores do parâmetro v, para os quais a curva intercepta o eixo dos quadros, obtidos resolvendo-se a equação $y(v) = 0$ (4.12). Para a solução de $y(v) = 0$, utilizou-se o método de Newton [CAR 69] com tolerância de 10^{-7}, sendo os valores de r_1, r_2 e r_3 calculados, tendo como valores iniciais, respectivamente, $0, 1/2$ e 1. Raízes repetidas ou fora do intervalo $[0, 1]$ (intervalo do parâmetro) foram ignoradas. O cálculo de ΔS é ilustrado pelo procedimento 4.1.

função *Calcula_*ΔS (pts_ctrl: pontos_de_controle)
inicio
$\Delta S = 0$;
$v_0 = 0$;
nro_raizes = 0;
para $\imath$ **variando de** 1 **ate** 3 **faça**
inicio
prim_aproximacao = $(\imath - 1)$*0.5;
r_{nro_raizes} = *raiz_polinomio_cubico* (pts_ctrl, prim_aproximacao);
se raiz_nao_repetida(r_{nro_raizes})) **então**
nro_raizes = nro_raizes + 1;
fim;
para $\imath$ **variando de** 0 **ate** (nro_raizes - 1) **faça**
se $r_i \in (v_0, 1)$ **então**
inicio
$\Delta S = \Delta S + \int_{v_0}^{r_i} y(v)\, dx$;
$v_0 = r_i$;
fim;
$\Delta S = \Delta S + \int_{v_0}^{1} y(v)\, dx$;
retorne (ΔS);
fim

Procedimento 4.1 Cálculo de ΔS.

onde

$$dx = (3a_{3x}v^2 + 2a_{2x}v + a_{1x})\, dv \tag{4.13}$$

e a integral definida $\int_{v_0}^{v} y(v)\, dx$ é dada por

$$\int_{v_0}^{v} y(v)\, dx = \int_{v_0}^{v} (a_{3y}v^3 + a_{2y}v^2 + a_{1y}v + a_{0y})(3a_{3x}v^2 + 2a_{2x}v + a_{1x})\, dv$$

$$= \int_{v_0}^{v} 3a_{3y}a_{3x}v^5\, dv + \int_{v_0}^{v} (2a_{3y}a_{2x} + 3a_{2y}a_{3x})v^4\, dv +$$

$$\int_{v_0}^{v}(a_{3y}a_{1x}+2a_{2y}a_{2x}+3a_{1y}a_{3x})v^3\,dv+$$
$$\int_{v_0}^{v}(a_{2y}a_{1x}+2a_{1y}a_{2x}+3a_{0y}a_{3x})v^2\,dv+$$
$$\int_{v_0}^{v}(a_{1y}a_{1x}+2a_{0y}a_{2x})v\,dv+\int_{v_0}^{v}a_{0y}a_{1x}\,dv$$

$$\begin{aligned}=\;& 3a_{3y}a_{3x}\int_{v_0}^{v}v^5\,dv+(2a_{3y}a_{2x}+3a_{2y}a_{3x})\int_{v_0}^{v}v^4\,dv+\\ & (a_{3y}a_{1x}+2a_{2y}a_{2x}+3a_{1y}a_{3x})\int_{v_0}^{v}v^3\,dv+\\ & (a_{2y}a_{1x}+2a_{1y}a_{2x}+3a_{0y}a_{3x})\int_{v_0}^{v}v^2\,dv+\\ & (a_{1y}a_{1x}+2a_{0y}a_{2x})\int_{v_0}^{v}v\,dv+a_{0y}a_{1x}\int_{v_0}^{v}dv\end{aligned}$$

$$\begin{aligned}=\;& (1/2)a_{3y}a_{3x}\,v^6\mid_{v_0}^{v}+(1/5)(2a_{3y}a_{2x}+3a_{2y}a_{3x})\,v^4\mid_{v_0}^{v}+\\ & (1/4)(a_{3y}a_{1x}+2a_{2y}a_{2x}+3a_{1y}a_{3x})\,v^4\mid_{v_0}^{v}+\\ & (1/3)(a_{2y}a_{1x}+2a_{1y}a_{2x}+3a_{0y}a_{3x})\,v^3\mid_{v_0}^{v}+\\ & (1/2)(a_{1y}a_{1x}+2a_{0y}a_{2x})\,v^2\mid_{v_0}^{v}+a_{0y}a_{1x}\,v\mid_{v_0}^{v}\end{aligned}$$

A partir dos pontos de controle informados, são obtidos os coeficientes do polinômio, os quais são utilizados, junto com uma aproximação inicial, no cálculo de uma raiz para $y(v)=0$ (4.12). A função *raiz_polinomio_cubico* implementa o método de Newton como descrito. O valor por ela devolvido estará sempre compreendido no intervalo $[0,1]$. Caso o valor calculado encontre-se fora desse, será retornado o extremo (0 ou 1) mais próximo do valor encontrado.

4.4.2 Cálculo da área do elemento de área Δq

Seja $[v_i, v_f]$ o intervalo do parâmetro v correspondente ao quadro q_i. O valor de v_i (v_f) é obtido resolvendo-se, também pelo método de Newton, a equação (4.11) para $x(v)=x(q_{i-1})$ ($x(v)=x(q_i)$), onde $x(q_{i-1})$ e $x(q_i)$ representam, respectivamente, as abscissas, relativas ao eixo dos quadros, correspondentes ao início

e ao fim do i-ésimo quadro (figura 4.8). O procedimento 4.2, com chamada Calcula_$\Delta q(v_i, v_f)$, descreve o cálculo de Δq.

função *Calcula_*Δq (v_0, v_f: parâmetro v; nro_raizes: int; r_i: raizes): real
inicio
  $\Delta q = 0$;
  para $\imath$ **variando de** 0 **ate** (nro_raizes - 1) **faça**
  se $r_i \in (v_0, v_f)$ **então**
  inicio
    $\Delta q = \Delta q + \int_{v_0}^{r_i} y(v)\, dx$;
    $v_0 = r_i$;
  fim;
  $\Delta q = \Delta q + \int_{v_0}^{v_f} y(v)\, dx$;
  retorne (Δq);
fim
Procedimento 4.2 Cálculo de Δq.

4.5 Resultados

Os resultados obtidos com a implementação do **Algoritmo Interpolador de Formas** são bastante satisfatórios como pode ser observado nas figuras 4.9 a 4.12. Estas ilustram a interpolação de uma *esfera* em uma rainha de xadrez, criada especialmente para a realização do filme **M.A.T.E.** [MAT 91], produzido no **CPGCC-UFRGS**. As superfícies fechadas foram aproximadas por superfícies topologicamente equivalentes a cilindros. A *esfera* (figura 4.9) foi modelada com 16 *patches*, enquanto que, na rainha, foram empregados 24 *patches*.

Além destes resultados, foram produzidas algumas animações do tipo *scroll-animation* em estações Proceda 4270-PC. Essas ilustram a aplicação das técnicas descritas para controle da velocidade de interpolação, bem como transformações entre geometrias bastante diferentes (p.e., uma lâmpada em um peão de xadrez e este, por sua vez, em um guarda-chuva aberto).

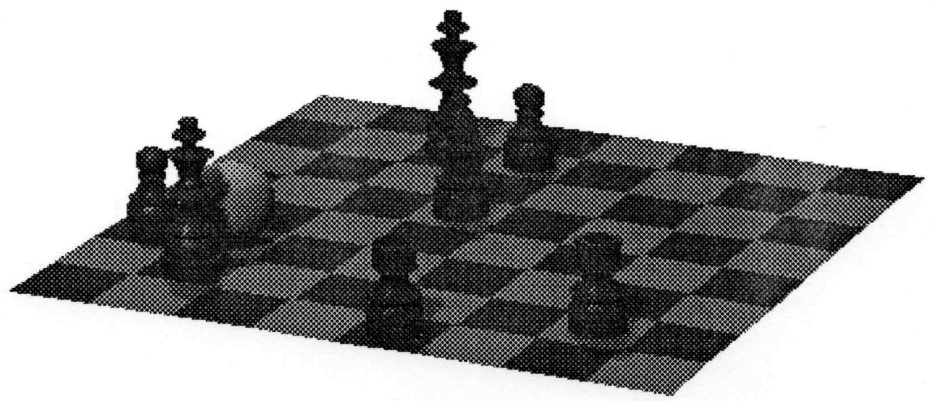

Figura 4.9: **M.A.T.E.** (a) Interpolação de uma esfera em rainha: esfera.

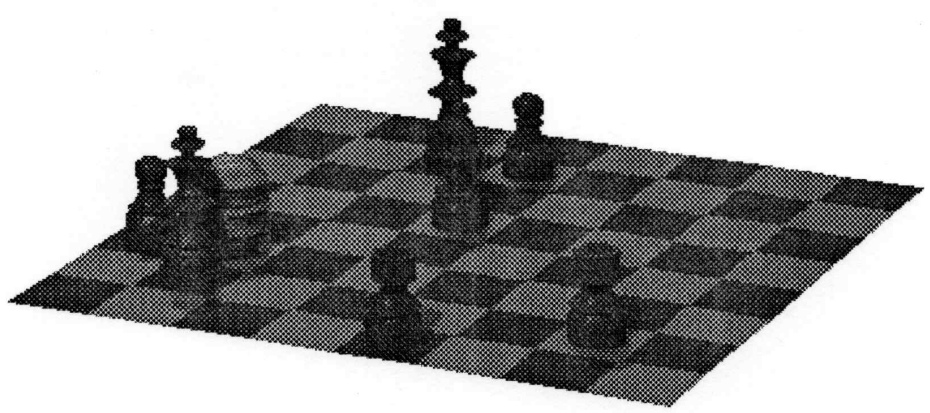

Figura 4.10: **M.A.T.E.** (b) 33% da transformação realizada.

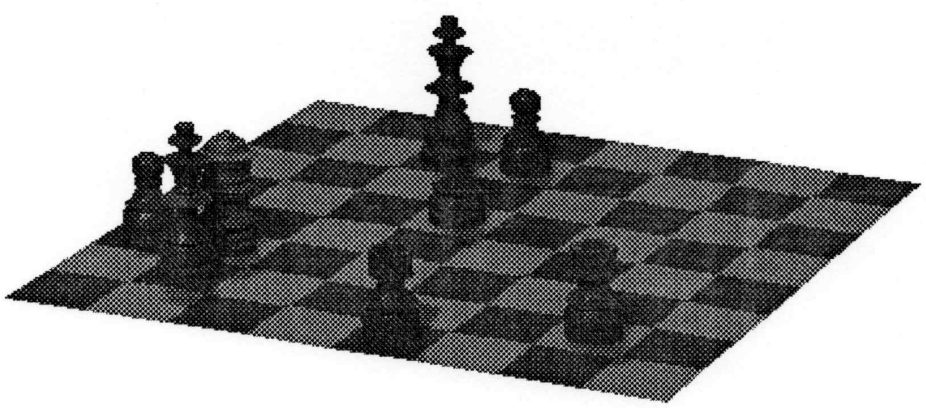

Figura 4.11: **M.A.T.E.** (c) 66% da transformação realizada.

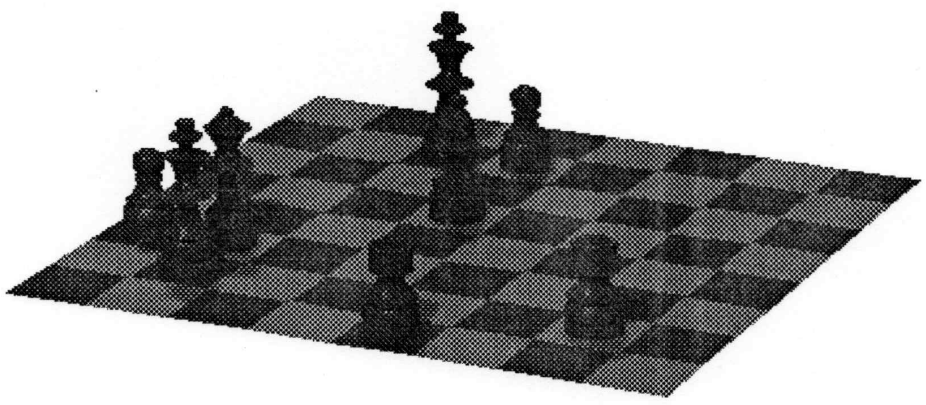

Figura 4.12: **M.A.T.E.** (d) 100% da transformação realizada.

Também ficou comprovada a liberação da necessidade, por parte do usuário, de especificar quaisquer correspondências entre *patches* dos objetos a serem interpolados. O algoritmo se encarrega, automaticamente, do equacionamento do número de *patches* e da definição de suas correspondências. Deste modo, o usuário, em situação bastante cômoda, precisa apenas indicar quais objetos devem ser interpolados.

4.6 Resumo

Este capítulo descreveu um protótipo de sistema de animação que implementa uma versão simplificada do **Algoritmo Interpolador de Formas**. Apesar de restringir a classe de entradas válidas, INTERPOL3D ilustra as potencialidades do algoritmo proposto. Dentre estas, destacam-se:

- a possibilidade de interpolar objetos modelados por diferentes números de *patches* ;
- o fato de liberar o usuário da necessidade de especificar correspondências entre *patches* de ambos objetos e
- as imagens intermediárias geradas representarem transições *coerentes* entre os objetos envolvidos.

Além disso, mostra-se como pode ser facilmente agregado ao algoritmo um esquema para controle sobre a variação da velocidade de interpolação.

5 ANÁLISE DOS RESULTADOS

5.1 Introdução

Este capítulo analisa o algoritmo proposto em relação a duas estratégias para interpolação de formas encontradas na literatura: o algoritmo Super-Construtor [BET 89] e uma abordagem implementada no sistema de animação SARA [LOP 88]. Em seguida, são feitas considerações sobre a qualidade dos resultados obtidos e é apresentada uma descrição dos tempos observados com a utilização do protótipo INTERPOL3D.

5.2 O algoritmo Super-Construtor

O algoritmo proposto por Bethel [BET 89] retringe-se a transformações entre poliedros. Consiste em equacionar seus números de faces, através da duplicação de alguns vértices (inserindo, deste modo, faces degeneradas), e interpolar os pares de vértices correspondentes. Isto faz com que algumas faces, antes degeneradas, *surjam*, enquanto outras, eventualmente, *desapareçam* (veja figura 5.1). O algoritmo Super-Construtor, como implementado por Bethel [BET 89], apresenta complexidade de ordem quadrática em relação ao número de faces do poliedro mais complexo e requer que o usuário estabeleça uma correspondência entre uma face do poliedro origem e uma face do poliedro alvo. A partir destas, procede-se o estabelecimento das correspondências entre as demais, observando-se critérios de vizinhança entre elas, até que todas as faces do objeto localmente mais simples tenham sido utilizadas. Neste momento, tal objeto recebe inserções de faces degeneradas em vértices e o processo continua até que todas as faces apresentem correspondentes.

Os resultados obtidos são modestos e limitados a geometrias bastante simples (figura 5.1). À medida que o número de faces aumenta, as formas intermediárias tendem a se tornar incompreensíveis (figura 5.2). Ao se realizar uma transformação entre um poliedro definido por poucas faces e outro, por um número

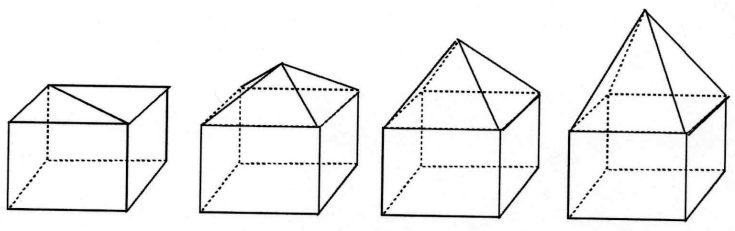

Figura 5.1: Transfomação entre um paralelepípedo e um obelisco. Extraído de [BET 89].

relativamente maior, observar-se-á o surgimento de um grande número de pequenas faces a partir dos vértices do poliedro mais simples. Por outro lado, suas faces originais permanecerão, por algum tempo, consideravelmente maiores que as recém-surgidas, ocasionando uma transformação não-homogênea. Além disso, as novas faces tendem a ser inseridas em uns poucos vértices. Dependendo das geometrias dos poliedros originais, isto pode ocasionar interpenetrações com uma freqüência bastante elevada. Estas constatações podem ser observadas na seqüência de imagens extraída de [BET 89], que ilustra a transformação de um octaedro (8 faces) em um icosaedro (20 faces) (figura 5.2).

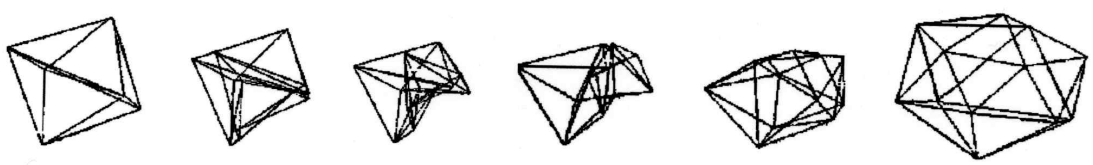

Figura 5.2: Interpolação de um octaedro em um icosaedro. Extraído de [BET 89].

5.3 Interpolação de formas no sistema SARA

A técnica utilizada para interpolação de formas no sistema de animação SARA [LOP 88] corresponde a uma generalização, a três dimensões, das idéias de Burtnyk e Wein [BUR 71], desenvolvidas originalmente para um contexto bidimensional (animação baseada em quadros-chaves). Segundo Burtnyk [BUR 71], figuras descritas por conjuntos de arestas podem ser continuamente transformados

interpolando-se adequadamente seus vértices. Entretanto, isto pressupõe que ambas descrições apresentem o mesmo número de vértices. Do contrário, faz-se necessária uma etapa de pré-processamento, representada a seguir [THA 85]:

procedimento *equaciona_nro_vertices* (O1, O2 : objetos; NV1,NV2: nro_vertices)
inicio
  se (NV1 > NV2) **então** { caso NV2 > NV1, o algoritmo e o
                mesmo com NV1,NV2, O1 e O2 permutados }
  inicio
    v = (NV1 - 1) div (NV2 - 1);
    p = (NV1 - 1) mod (NV2 - 1);
    adicione v vertices as p primeiras arestas de O2 e $(v-1)$ as demais;
  fim;
fim

Procedimento 5.1 Equacionamento do número de vértices.

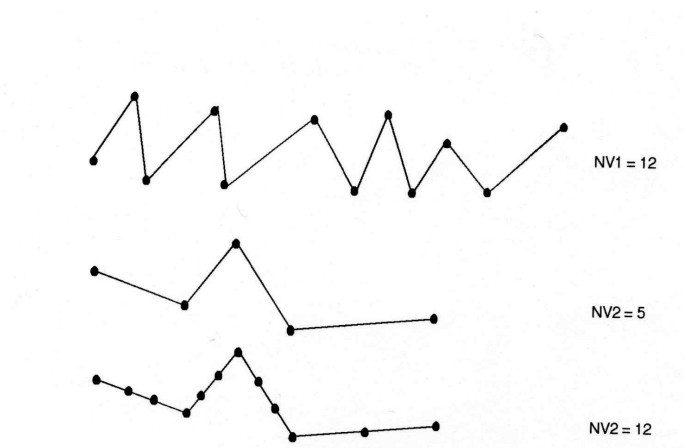

Figura 5.3: Pré-processamento das arestas de O1 e O2. Extraído de [THA 85].

A figura 5.3 ilustra o uso do procedimento 5.1. Nesta abordagem, a distribuição dos novos vértices ao longo das arestas já existentes é determinante na definição das formas intermediárias geradas, uma vez que o método interpola o i-ésimo vértice da primeira figura no i-ésimo vértice da segunda [LOP 88]. Entretanto, o algoritmo não faz nenhuma consideração acerca das geometrias envolvidas. Os vértices inseridos são, de tal modo, distribuídos ao longo das várias arestas que

ocasionam o surgimento de regiões com altas ou baixas densidades de novos vértices, independentemente das dimensões das respectivas arestas (figura 5.3). Por outro lado, não é utilizada nenhuma técnica para, pelo menos, distribuir, de maneira mais homogênea, ao longo das diversas arestas, as quantidades de vértices a inserir.

Em constituindo uma generalização tridimensional da técnica exposta, a estratégia implementada por Lopes [LOP 88] sofre das mesmas limitações, ampliadas em uma dimensão. Contudo, não é apresentada nenhuma descrição relativa à etapa de pré-processamento que esclareça como são realizadas as associações entre pares de arestas no espaço tridimensional. O seguinte trecho foi transcrito de [LOP 88]: "Na presente implementação cabe ao animador a correta modelagem dos atores por malhas de polígonos de forma a garantir que a estrutura geométrica intermediária seja gerada preservando a consistência da malha. A interpolação de forma genérica e automática entre duas quaisquer descrições por malhas de polígonos é um objetivo a atingir, não sendo um caso trivial de resolver, e não sendo conhecido algoritmo publicado solucionando o problema".

5.4 O Algoritmo Interpolador de Formas x as técnicas analisadas

O **AIF** apresenta uma série de vantagens sobre as técnicas analisadas. Dentre elas, destacam-se (tabela 5.1):

- não requer do usuário qualquer intervenção no processo, exceto a indicação dos objetos a interpolar;
- possibilidade de interpolar objetos modelados diretamente por *patches* curvos e
- existência de vários programas modeladores que utilizam descrições *spline*.

A necessidade de intervenção por parte do usuário, bastante presente na abordagem de Lopes [LOP 88], constitui uma característica indesejável; objetos são

Tabela 5.1: Vantagens do Algoritmo Interpolador de Formas sobre as técnicas analisadas.

Outras técnicas	*Algoritmo Interpolador de Formas*
impossibilidade de manipular superfícies curvas. Estas devem ser discretizadas em polígonos, aumentando as exigências de tempo de processamento e memória;	objetos podem ser modelados por *patches* curvos;
necessidade de intervenção por parte do usuário, fortemente caracterizada na abordagem de Lopes [LOP 88], onde a mesma descrição de um objeto pode não ser adequada para diferentes transformações;	o usuário precisa apenas indicar quais objetos devem ser interpolados;
dificuldade na modelagem dos objetos, feita, basicamente, de maneira artesanal.	existência de um grande número de modeladores que utilizam *patches spline*.

construídos de modo a servirem bem a uma transformação específica, podendo a mesma descrição não ser adequada para diferentes transformações.

A impossibilidade de manipular superfícies curvas, no caso das outras abordagens, exige que aquelas sejam discretizadas em um grande número de polígonos, aumentando as exigências de tempo de processamento e memória.

Outro ponto a ser destacado é o fato de que tanto na abordagem de Bethel [BET 89], quanto na de Lopes [LOP 88], a modelagem dos objetos assume um caráter fortemente artesanal. Isto é ainda mais crítico na técnica implementada por Lopes [LOP 88], visto que se propõe a interpolar formas mais complexas que os simples poliedros de Bethel.

Embora a implementação do **AIF** não tenha levado em conta a dimensão dos *patches* envolvidos, seu critério para distribuição das subdivisões produz resultados bastante homogêneos (seção 3.5.3.4). Deste modo, possibilita gerar interpolações nas quais as formas intermediárias apresentam grande coerência entre si e entre os

objetos originais (figuras 5.4 e 5.5). Tal coerência decorre, enormemente, do uso de trajetórias lineares para interpolação dos coeficientes geométricos. A utilização de trajetórias não lineares, embora permita a obtenção de uma maior variedade de caminhos para interpolação, pode levar a formas intermediárias com pouca semelhança em relação aos objetos extremos.

Considerações sobre a qualidade dos resultados obtidos com o **AIF** são, essencialmente, subjetivas, uma vez que esses não apresentam contrapartida na realidade. Contudo, a forte coerência observável entre as formas intermediárias e extremas admite classificar suas transformações como *intuitivamente esperadas* (figuras 1.1, 4.1, 4.9 a 4.12, 5.4 e 5.5). Deste modo, pode-se considerar o objetivo de construir um algoritmo para interpolação entre formas geometricamente complexas, satisfatoriamente alcançado.

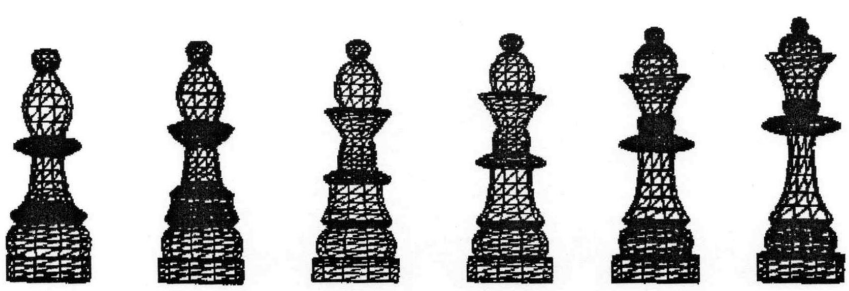

Figura 5.4: Transformação de um bispo em uma rainha.

5.5 A modelagem dos objetos e os tempos de interpolação

A modelagem de objetos representados por superfícies (não confundir com *boundary representation* - modelagem de sólidos [MAN 88]) pode ser realizada, basicamente, dos seguintes modos: *sweeping* rotacional e/ou translacional, composição de *patches* e parametrização de primitivas. O *sweeping* rotacional permite a obtenção de objetos com geometrias bastante complexas a partir da definição de um

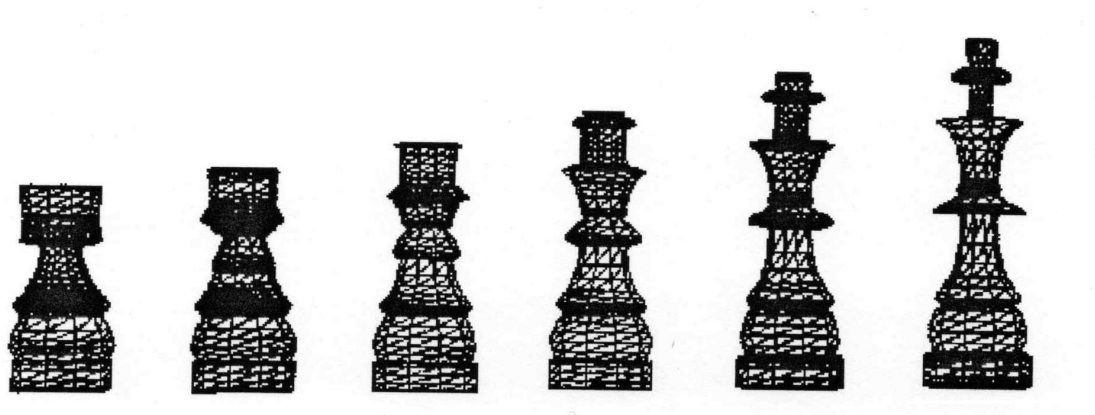

Figura 5.5: Transformação de uma torre em um rei.

perfil a ser revolucionado em torno de um eixo. Devido ao seu poder de expressão, simplicidade conceitual e por permitir gerar objetos aceitos como entradas válidas pelo protótipo INTERPOL3D, a técnica de *sweeping* rotacional foi escolhida para construção dos objetos a serem utilizados nos testes do algoritmo proposto. Observe-se, entretanto, que a escolha deveu-se apenas a uma questão de conveniência e não, a uma limitação do algoritmo. Os objetos modelados (figura 5.6), bem como seus respectivos números de *patches* (Bézier), encontram-se listados na tabela 5.2.

Peças de xadrez foram escolhidas para realização dos testes, devido aos seguintes motivos:

- apresentarem geometrias complexas;
- serem esteticamente interessantes;
- pertencerem a um mesmo contexto capaz de prover objetos formados por diferentes números de *patches* e
- serem facilmente gerados por *sweeping* rotacional.

A tabela 5.3 mostra a soma dos tempos envolvidos nas etapas de identificação das relações de incidência, equacionamento do número de *patches* e estabelecimento das correspondências entre *patches* e coeficientes geométricos das duas

Tabela 5.2: Objetos com seus respectivos números de *patches*.

objetos	*nro de patches*
esfera	16
peão	16
bispo	18
torre	22
rainha	24
rei	26

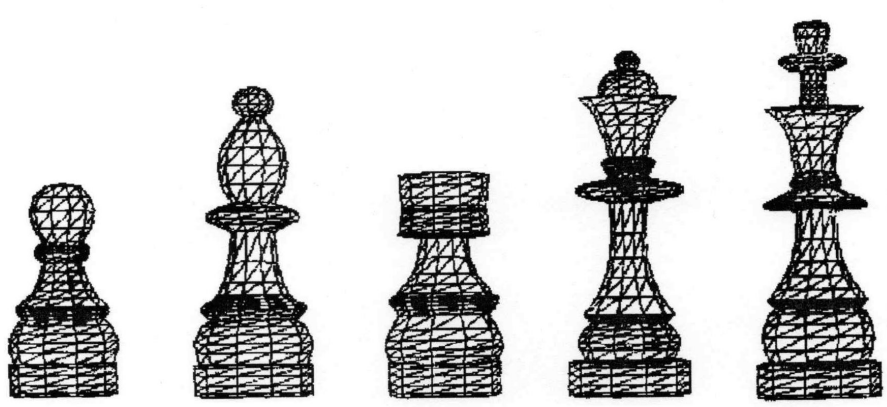

Figura 5.6: Objetos modelados por *sweeping* rotacional.

Tabela 5.3: Tempos (em centésimos de segundo) envolvidos na fase de pré-processamento.

	esfera	peão	bispo	torre	rainha
peão	370	-	-	-	-
bispo	406	406	-	-	-
torre	521	521	571	-	-
rainha	593	593	642	741	-
rei	668	668	714	818	879

representações. A etapa de identificação das relações de incidência foi implementada utilizando a abordagem quadrática deste procedimento. Esta, entretanto, apresenta menor custo computacional para a classe de objetos escolhida (número de *patches* menor que vinte e quatro - seção 3.5.3.1). Estes resultados foram observados em um microcomputador compatível com IBM PC AT 386 com co-processador aritmético Intel 80387. O código referente ao protótipo INTERPOL3D foi escrito em Turbo C versão 2.0 e os tempos medidos utilizando a função **clock()** disponível na linguagem.

Os tempos observados independem do sentido especificado para interpolação (*i.e.*, peão→rainha ou rainha→peão, por exemplo). O gráfico da figura 5.7 ilustra o comportamento da curva que descreve a variação do tempo de pré-processamento necessário para interpolar um peão nos demais objetos. Os valores correspondem aos dados estabelecidos na tabela 5.3.

5.6 Resumo

Este capítulo analisou o algoritmo proposto relativamente a dois métodos para interpolação de formas encontrados na literatura: o algoritmo Super-Construtor [BET 89] e uma abordagem que generaliza o trabalho de Burtnyk e Wein [BUR 71] à três dimensões [LOP 88]. Foram evidenciadas, nas duas técnicas, deficiências quanto à possibilidade de manipular superfícies curvas, necessidade de intervenção por parte do usuário, além de restrições quanto às facilidade de modelagem. O algoritmo proposto por Bethel [BET 89], apesar de restrito a poliedros, tende a

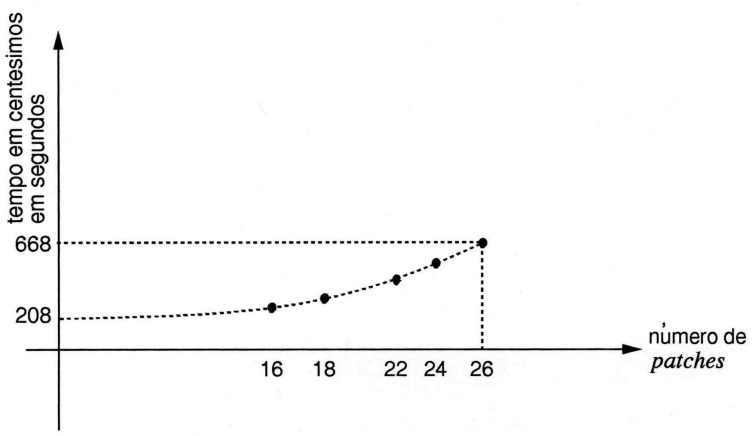

Figura 5.7: Variação do tempo de pré-processamento necessário para transformar um peão nos demais objetos da tabela 5.3.

gerar transformações não-homogêneas quando esses apresentem uma diferença relativa considerável entre seus números de faces. A abordagem de Lopes [LOP 88] exige que os objetos sejam construídos levando em conta os alvos de suas interpolações. Isto pode se tornar impraticável nos casos de objetos suficientemente complexos.

Como características e vantagens do **AIF**, foram mencionadas:

- as formas intermediárias geradas apresentam grande coerência entre si e em relação aos objetos originais;
- o usuário não precisa interferir no processo (p.e., modelando objetos de maneira específica, com intuito de favorecer determinada transformação);
- superfícies curvas podem ser manipuladas diretamente e
- o **AIF** utiliza, como entradas, objetos modelados por *patches spline*, para os quais existe um grande número de programas modeladores.

6 CONCLUSÕES

A utilização crescente, por parte das indústrias cinematográfica e publicitária, de efeitos especiais obtidos com o uso de técnicas de Computação Gráfica, vem exigindo não só um aumento no grau de realismo das imagens geradas, como também o desenvolvimento de métodos para obtenção de efeitos virtuais. Dentre estes, destacam-se as metamorfoses entre objetos, que constituem um recurso valioso à disposição da criatividade do animador.

Interpolações de formas foram tratadas, inicialmente, em contextos bidimensionais, sendo os objetos descritos por conjuntos de arestas [BUR 71]. Contudo, existem restrições decorrentes da perda de informação que acontece na passagem objeto 3D→projeção 2D. Tais restrições ocorrem, por exemplo, quando se faz necessário interpolar linhas visíveis de um objeto em linhas escondidas (não visíveis) de outros. Catmull [CAT 78] discute estes problemas.

No contexto tridimensional, como não ocorre perda de informação, interpolações de formas podem ser obtidas sem necessidade de intervenção por parte do usuário. Entretanto, a literatura não registra nenhuma abordagem que trate este problema de maneira automática. Também não é conhecido algoritmo capaz de manipular superfícies curvas diretamente.

Digital Productions, empresa especializada em Computação Gráfica, produziu seqüências animadas que ilustram transformações entre objetos e afirma poder realizá-las entre, *praticamente*, quaisquer objetos [DRE 89]. Contudo, nenhuma das tentativas de estabelecer contatos com essa empresa obtiveram sucesso. Deste modo, não foi possível a realização de uma análise do método por ela utilizado; tampouco, uma determinação do seu grau de automatismo. Aparentemente, tal algoritmo manipula superfícies curvas. Entretanto, apenas a observação da seqüência animada [DRE 89] é insuficiente para que se percebam evidências irrefutáveis.

As demais abordagens conhecidas correspondem às descritas por Bethel [BET 89] e por Lopes [LOP 88]. A primeira restringe-se a transformações entre poliedros e seus

resultados são bastante modestos. A segunda é uma generalização, à três dimensões, das idéias apresentadas por Burtnyk [BUR 71], desenvolvidas originalmente para um contexto bidimensional. Nenhuma dessas é capaz de manipular superfícies curvas diretamente; exigem que o usuário conheça o funcionamento do algoritmo utilizado, de modo a construir descrições para os objetos que favoreçam a realização das transformações e a modelagem dos objetos constitui tarefa fortemente artesanal.

Com o intuito de obter um procedimento capaz de suprir as deficiências apresentadas pelas abordagens de Bethel [BET 89] e de Lopes [LOP 88], objetivou-se a construção de um algoritmo com as seguintes características:

- ser automático, no sentido de exigir do usuário apenas a indicação dos objetos a interpolar;
- realizar interpolações entre objetos modelados por diferentes números de *patches*;
- permitir a direta manipulação de superfícies curvas;
- ser descrito por algum tipo de formalismo, de modo a se poder, facilmente, evidenciar sua validade e critério de correção;
- permitir, de maneira simples, a incorporação de técnicas para controle da velocidade de interpolação e
- utilizar objetos constuídos por processos não artesanais.

6.1 O Algoritmo Interpolador de Formas

Atendendo às exigências estabelecidas, obteve-se o **Algoritmo Interpolador de Formas**. Sua validade e critério de correção baseiam-se em elementos da Topologia Combinatória. O algoritmo apresentado consiste em:

1. converter os objetos a serem interpolados para a representação padrão (Bézier bicúbica);
2. identificar as relações de incidência entre os diversos *patches* que compõem as descrições de cada um dos objetos;

3. obter, para cada objeto, seu centro de malha;
4. equacionar, relativamente a estes, os números de linhas e colunas em ambas representações;
5. estabelecer uma correspondência biunívoca entre os *patches* dos objetos, preservando suas relações de incidência;
6. definir, para cada par de *patches* associados, uma correspondência biunívoca entre seus coeficientes geométricos e
7. interpolar os coeficientes geométricos associados, gerando, deste modo, as diversas formas intermediárias.

A escolha de superfícies Bézier como base para descrição do algoritmo deveu-se aos seguintes pontos:

- necessidade de se utilizar uma representação uniforme;
- facilidade de subdivisão preservando a forma original e
- facilidade para conversão a partir de outros tipos de *patches*.

Apesar do **AIF** não levar em conta as dimensões dos *patches* envolvidos, seu critério para distribuição das subdivisões produz resultados bastante homogêneos. Observou-se que, aliado a esse, o uso de trajetórias lineares para interpolação dos coeficientes geométricos produz formas intermediárias apresentando alto grau de coerência entre si e entre os objetos originais. A utilização de trajetórias não lineares, embora permita a obtenção de uma maior variedade de caminhos para interpolação, pode levar a formas intermediárias com pouca semelhança em relação aos objetos extremos.

6.2 Estruturas de dados e procedimentos

Ao longo do capítulo 3, foram apresentadas as estruturas de dados e os procedimentos necessários à implementação do **AIF**. De modo mais específico,

foram discutidas duas versões para o procedimento de *identificação das relações de incidência*: uma logarítmica $O(n \log n)$ e outra quadrática $O(n^2)$. Associadas a cada uma destas, foram definidas as estruturas de dados mais adequadas. A primeira faz uso de árvores binárias de pesquisa tridimensionais [BEN 75] e apresenta vantagens sobre a segunda quando os objetos a interpolar (suas componentes conexas) são constituídas por vinte e quatro ou mais *patches*. Do contrário, a versão quadrática será mais econômica. O algoritmo completo apresenta complexidade $O(nn')$.

6.3 Protótipo

A implementação do protótipo INTERPOL3D confirmou as expectativas quanto à eficácia do **AIF**, com excelentes resultados. Apesar das restrições estabelecidas em sua especificação, todas as potencialidades e objetivos definidos para o algoritmo puderam ser observados.

INTERPOL3D se utiliza de controle explícito definido por meio de uma linguagem gráfica. Uma curva Bézier cúbica é utilizada para permitir controle sobre a velocidade de interpolação.

6.4 Resultados

Exemplos de alguns dos resultados obtidos com a utilização do **AIF** podem ser encontrados ao longo deste trabalho, atestando a eficácia do algoritmo proposto. Outros resultados significativos incluem uma seqüência que ilustra a transformação de uma esfera em rainha, produzida para realização do filme **M.A.T.E.** [MAT 91]; a transformação de um peão em rainha, publicada na capa dos anais do IV Simpósio Brasileiro de Computação Gráfica e Porcessamento de Imagens, bem como no jornal Folha de São Paulo, caderno *ciência* (12 de julho de 1991), além de várias seqüências exibidas no modo *scroll animation* em estações Proceda 4270-PC.

6.5 Possíveis extensões ao trabalho

6.5.1 Uso do **AIF** com superfícies racionais

Superfícies racionais vêm se firmando como padrão em Computação Gráfica. Isto se deve, basicamente, ao fato de proporcionarem representações exatas para superfícies quádricas e de revolução [FAR 90] [BAR 87]. Embora o protótipo tenha sido implementado em um contexto de superfícies uniformes não-racionais, o Algoritmo Interpolador de Formas, como descrito no capítulo 3, é perfeitamente adequado à implementação com superfícies racionais. Neste caso, deve ser utilizada uma versão racional do algoritmo de *de Casteljau*, que, por razões de estabilidade numérica, segundo Farin [FAR 90], consiste em: subdividir a pré-imagem 4D do *patch* racional (de maneira análoga à descrita no capítulo 2 para o caso não-racional) e projetar os coeficientes intermediários $[w_i x \ w_i y \ w_i z \ w_i]^T$ no hiperplano $w = 1$. Os valores de (x, y, z) corresponderão aos coeficientes geométricos dos poliedros de controle para os segmentos esquerdo e direito da superfícies subdividida, sendo w_i o peso associado ao i-ésimo coeficiente. A partir destes, as superfícies são obtidas por meio de expressão semelhante à apresentada na seção 2.2, exceto pelo fato de existirem pesos associados aos diversos coeficientes geométricos.

6.5.2 Utilização de subdivisões adaptativas

A subdivisão de *patches* Bézier para valores dos parâmetros u e $v = 1/2$ (como implementado no protótipo), produz efeitos surpreendentemente interessantes (embora se trate de um conceito subjetivo), mesmo quando o número de subdivisões a realizar é superior ao número de *patches* a subdividir. A figura 6.1 ilustra o caso de um *patch* que deve sofrer duas subdivisões. Uma situação como a mostrada na figura 6.2, entretanto, parece ser mais adequada para estes casos. Porém, uma tal subdivisão adaptativa implica a utilização do algoritmo de *de Casteljau* em sua forma recursiva (sem simplificações realizadas a partir do conhecimento do valor de parâmetro a ser utilizado para subdivisão - seção 2.3.1), aumentando o tempo de processamento requerido pelo algoritmo. Observe-se que situação mostrada na

figura 6.1 produz novos *patches* com diferentes tamanhos. Entretanto, uma próxima subdivisão, se necessária, seria aplicada ao "sub-*patch*" a_2 (figura 6.1(c)), atingindo, deste modo, uma situação ideal. Estas se caracterizam por produzirem *patches* com mesmas dimensões e ocorrem quando o número de divisões aplicadas ao *patch* pode ser expresso na forma $2^n - 1$, onde n é inteiro não negativo.

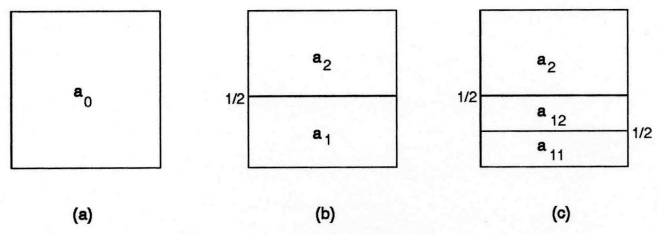

Figura 6.1: Subdivisão de *patches* para $u = 1/2$: (b) primeira subdivisão; (c) segunda subdivisão.

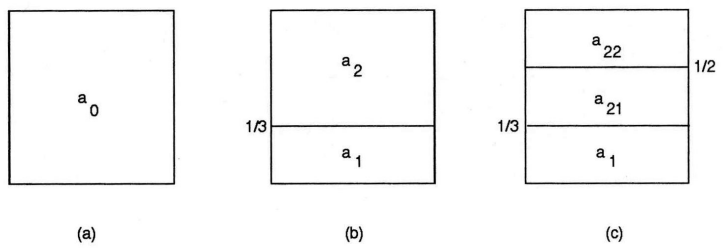

Figura 6.2: Subdivisão de *patches* para: (b) $u = 1/3$ e (c) $u = 1/2$.

Um procedimento de subdivisão adaptativa pode ser, entretanto, bastante útil num contexto de superfícies não-uniformes. Nestes casos, os custos adicionais de utilização de algoritmos recursivos, como Oslo ou Böhm [BAR 87], podem ser compensados pela qualidade das imagens geradas.

6.5.3 Utilização de texturas

A simulação de texturas [WAL 91] contribui bastante para o aumento da percepção de realismo em imagens sintetizadas por computador, visto que os objetos passam a exibir características próprias dos materiais de que são constituídos.

Pensar sobre transformações é pensar, não apenas em diferentes formas, mas também nos diferentes materiais que os constituem. A realização de transformações acompanhadas das respectivas mudanças nas texturas dos objetos permite a obtenção de efeitos bem mais interessantes do que os conseguidos interpolando-se apenas formas e cores. Isto pode ser facilmente implementado interpolando-se as informações de textura no espaço RGB, de maneira semelhante à descrita para cores (seção 4.3.5). No caso de ser utilizado mapeamento de saliências (*bump mapping*) [BLI 78], os vetores deslocamento normais à superfície também devem ser interpolados.

Após o equacionamento (seção 3.5.3.2), ter-se-á o mesmo número de *patches* para as representações de ambos objetos. Assim, seja O^k a k-ésima forma intermediária resultante da interpolação de O em O' e seja p_k um *patch* de O^k. Sejam T_i e T'_i, as informações de texturas associadas, respectivamente, aos *patches* p_i e p'_i, correspondentes de p_k em O e O'. A obtenção da cor da textura associada ao ponto $pt_k(x, y, z)$ pertencente a p_k, pode ser realizada do seguinte modo:

- encontrar os valores de u e v (variáveis paramétricas) associadas ao ponto $pt_k(x, y, z)$. Isto pode ser extremamente simplificado, por exemplo, discretizando as superfícies em triângulos, onde cada vértice armazena seus valores correspondentes de u e v. Um algoritmo de linha de varredura (*scan-line*) utilizando-se de um z-*buffer* [ROG 85] pode ser utilizado para realização do *rendering*;
- caso as informações de texturas estejam representadas em mapas bidimensionais, a cor associada ao ponto $pt_k(x, y, z)$ pode ser expressa por :

$$\begin{aligned} COR_{RGB}(pt_k(x,y,z)) &= COR_{RGB}(T'_i(u,v))\frac{(n-k)}{n} + \\ & \quad (COR_{RGB}(T_i(u,v)) - COR_{RGB}(T'_i(u,v)))\frac{k}{n} \end{aligned}$$

onde n representa o número total de quadros da transformação;

- caso se utilizem texturas sólidas [PER 85], tem-se:

$$\begin{aligned} COR_{RGB}(pt_k(x,y,z)) &= COR_{RGB}(T_i'(x_i',y_i',z_i'))\frac{(n-k)}{n} + \\ & (COR_{RGB}((T_i(x_i,y_i,z_i)) - COR_{RGB}(T_i'(x_i',y_i',z_i')))\frac{k}{n} \end{aligned}$$

onde x_i', y_i', z_i' e x_i, y_i, z_i correspondem às coordenadas dos pontos pt_i e pt_i', sobre p_i e p_i', respectivamente, obtidos a partir da aplicação dos valores de u e v à expressão (2.2);

- para a interpolação de, por exemplo, uma textura sólida em outra descrita por um mapa bidimensional tem-se:

$$\begin{aligned} COR_{RGB}(pt_k(x,y,z)) &= COR_{RGB}(T_i'(u,v))\frac{(n-k)}{n} + \\ & (COR_{RGB}((T_i(x_i,y_i,z_i)) - COR_{RGB}(T_i'(u,v)))\frac{k}{n} \end{aligned}$$

ANEXO A-1 ANIMAÇÃO POR COMPUTADOR

A-1.1 O conceito de animação

O termo *animação* refere-se ao efeito de produzir em um objeto a aparência de vida. Encontra-se na literatura, uma vasta conceituação que contempla tanto definições genéricas como a de Foley [FOL 90]: animação é qualquer modificação que apresente algum efeito visual, como, por exemplo, variação de posição, forma, cor, textura, transparência e estrutura de um objeto e/ou alterações na posição, orientação e foco da câmera, mudança na iluminação ou ainda na técnica de *rendering*; quanto definições mais específicas [THA 85]: animação é a técnica na qual a ilusão de movimento é criada pela exibição de uma série de quadros, cada um apresentando uma pequena alteração em relação ao anterior, projetados a uma certa taxa, comumente 24 quadros por segundo, no caso de cinema, ou 30 quadros por segundo, no caso de vídeo.

A-1.2 O que é animação por computador

O processo convencional para produção de seqüências animadas envolve uma série de etapas [CAT 78] [THA 85] [FOL 90] que podem ser resumidamente descritas como:

- concepção da estória;
- produção do *storyboard*;
- gravação da trilha sonora;
- associação de instantes da animação a sons significativos;
- desenho dos quadros-chaves;
- geração dos quadros intermediários;
- cópias dos desenhos para folhas de acetato e

- coloração e filmagem dos quadros.

Computadores podem desempenhar uma variedade de diferentes papéis na produção de desenhos animados e, portanto, o termo animação por computador é impreciso e algumas vezes sua interpretação pode ser confusa [THA 85]. Animação por computador denota a realização, com uso de computador, de quaisquer combinações das tarefas acima relacionadas. Animação auxiliada por computador, termo embora muitas vezes confundido com aquele, refere-se especificamente aos sistemas de animação baseados em quadros-chaves (Nível 2, seção 7.3.1) [THA 85]. Eis algumas atividades que podem ser executadas com auxílio de um computador [THA 85]:

- na criação de desenhos:
  - **a.** desenhos podem ser digitalizados;
  - **b.** desenhos podem ser criados por meio de um editor gráfico interativo e
  - **c.** objetos complexos podem ser gerados via programação;
- na criação de movimentos:
  - **a.** quadros intermediários ou movimentos complexos podem ser gerados automaticamente.
- para colorir os quadros:
  - **a.** imagens realísticas podem ser geradas e
  - **b.** desenhos podem ser pintados utilizando um sistema interativo;
- na filmagem:
  - **a.** câmeras de vídeo podem ser controladas por computador e
  - **b.** câmeras sintéticas podem ser programadas;
- na fase de pós-produção:
  - **a.** edição e sincronização podem ser controladas por computador.

A-1.3 Classificações dos sistemas de animação

Apesar dos critérios de classificação apresentados por Pueyo [PUE 88], prefere-se, neste trabalho, classificar os sistemas de animacão segundo três critérios principais:

- complexidade e poder de expressão;
- modo de produção das seqüências animadas e
- animações por quadros-chaves *versus* animações modeladas.

A-1.3.1 Complexidade e poder de expressão dos sistemas de animação

Thalmann [THA 85] apresenta a definição de cinco níveis segundo os quais sistemas de animação por computador podem ser classificados. Cada nível apresenta as seguintes características:

Nível 1 sistemas utilizados para, interativamente, criar, colorir, armazenar, recuperar e modificar desenhos. Este nível é constituído por editores gráficos;

Nível 2 sistemas capazes de gerar quadros intermediários e mover objetos ao longo de uma trajetória;

Nível 3 sistemas que oferecem ao animador operações que podem ser aplicadas aos objetos, como, por exemplo, translações e rotações, além de operações aplicáveis às câmeras sintéticas, como, por exemplo, *zoom*, movimentos horizontais e/ou verticais;

Nível 4 sistemas que permitem a definição de atores - objetos que possuem suas próprias animações e

Nível 5 sistemas extensíveis que podem aprender durante sua utilização. Sistemas com esta característica ainda não se encontram disponíveis.

A-1.3.2 Modo de produção de seqüências animadas

Quanto ao modo de produção, pode-se classificar os sistemas de animação em: sistemas de quadro a quadro e sistemas de tempo real. Nos primeiros, cada quadro da animação é gerado para posterior exibição, o que pode ser feito utilizando-se abordagens como *scroll animation*, *real-time playback* ou gravação em filme.

A técnica de *scroll animation* consiste em dividir a memória de vídeo (*frame buffer*) em diversas áreas onde são armazenadas as imagens a serem exibidas. A partir de então, o controlador de vídeo deve ser informado de que porções daquela memória devem ser exibidas a cada vez e em que ordem. O inconveniente deste método é a limitação física da memória de vídeo que restringe o número de quadros que podem ser armazenados simultaneamente.

Real-time playback consiste em armazenar em disco as imagens geradas, que são, posteriormente, lidas e exibidas. Neste caso, praticamente não existem limitações quanto ao número de quadros que podem ser utilizados em uma animação. Animações obtidas com o emprego desta técnica apresentam, entretanto, qualidade infeior às obtidas com o uso de *scroll animation*.

Animações em tempo real são limitadas pela capacidade do computador utilizado e requerem *hardware* especializado como arquiteturas paralelas e processadores gráficos. Cada imagem de uma seqüência em tempo real deve ser gerada e exibida em menos de 1/15 segundo. Do contrário, não se obtém a ilusão de movimento contínuo [THA 85].

A-1.3.3 Animação por quadros-chaves x animação modelada

Interpolação de quadros-chaves constitui a técnica básica do processo de animação convencional. Conhecido como *inbetweening*, corresponde à geração de quadros intermediários para uma seqüência de imagens em movimento/transformação, dados um primeiro e último quadros. A essência deste processo corresponde à determinação de que partes ou pontos do primeiro quadro devem ser mapeados

em quais partes ou pontos do último quadro [REE 81]. Apesar de sua simplicidade conceitual, os processos de interpolação apresentam dificuldades no que diz respeito ao estabelecimento de correspondências entre as partes a serem interpoladas, especificação de acelerações/desacelerações dos movimentos e descontinuidades nas imagens de quadros vizinhos [REE 81].

O processo de animação 2D auxiliado por computador apresenta várias restrições. Nele, a geração automática de quadros intermediários, a partir de quadros-chaves, é dificultada pelo fato de que os desenhos são projeções 2D de objetos 3D. Deste modo, perde-se informação necessária para o processo de interpolação [CAT 78]. Outra deficiência é a necessidade do usuário especificar as correspondências entre linhas dos dois quadros, isto é, a necessidade de serem definidas correspondências entre linhas visíveis e outras, possivelmente não visíveis, de modo que as transições possam se processar corretamente [CAT 78]. A dificuldade de um programa determinar automaticamente tais correspondências deve-se à perda de informação que acontece na passagem objeto 3D $\rightarrow$ projeção 2D.

Animações modeladas correspondem à criação e manipulação de objetos no espaço tridimensional, podendo produzir efeitos bastante realísticos. Filmes como: *The Abyss, The Last Starfighter, Young Sherlock Holmes* [FOL 90], *Return of Jedi* e *TRON* [THA 85], utilizaram-se de efeitos especiais obtidos por meio de animações 3D. Sistemas de animação modelada situam-se, geralmente, nos níveis 3 ou 4 (seção 7.3.1).

A-1.4 Linguagens para animação

Existem três tipos principais de linguagens para descrição de animações [FOL 90]:

- notações de listas lineares;
- linguagens de propósito geral nas quais são embutidas diretivas de animação e
- linguagens gráficas.

Nas linguagens com notações de listas lineares, cada evento é descrito através de uma sintaxe como a seguinte: <**qi**>, <**qf**>, <**ação**> <**objeto**> <**parâmetros**>, onde <**qi**> especifica o número do quadro em que a ação deve ser iniciada, <**qf**> indica o número do quadro em que a ação deve ser concluída, <**ação**> corresponde à ação a ser executada sobre o objeto <**objeto**> e <**parâmetros**> corresponde à lista de parâmetros necessários a <**ação**>. Assim, por exemplo: 6, 15, TRANSLADE TORRE 0, 0, 30 significa que a partir do sexto quadro e até o décimo quinto, o objeto TORRE deve ser transladado de três unidades a cada quadro em relação ao eixo z.

Linguagens de propósito geral podem ser utilizadas como linguagens hospedeiras, nas quais são embutidas diretivas de animação. Isto implica a existência de um pré-processador capaz de converter as diretivas em chamadas a subrotinas pré-definidas ou em comandos da própria linguagem hospedeira.

Linguagens gráficas procuram descrever animações de um modo visualmente mais rico. A principal idéia contida em tais linguagens é a substituicão do paradigma textual por um paradigma visual [FOL 90]. Sua importância decorre enormemente do fato de que, por meio de linguagens textuais, podem ser utilizadas construções de alto nível. Estas podem encobrir, numa notação compacta, eventos complexos, dificultando a compreensão do animador sobre o que realmente está sendo descrito.

A-1.5 Métodos de controle de animação

Existem vários métodos para controle de animações, os quais, de algum modo, são independentes das linguagens utilizadas, no sentido de que a maioria dos métodos podem ser adaptados para uso com os vários tipos de linguagens [FOL 90]. O controle de uma animação diz respeito a como são especificadas as ações e interações entre os diversos objetos que a compõem. Foley [FOL 90] apresenta sete métodos de controle de animação:

- controle explícito;

- controle procedural;
- atores;
- reprodução de movimentos reais;
- controles cinemáticos e dinâmicos;
- sistemas baseados em restrições e
- animação baseada em Física.

O controle explícito é o mais simples dentre todos. Corresponde a especificar como se processará a animação por meio da descrição dos métodos de interpolação associados aos quadros intermediários. Tal descrição pode ser informada explícita (via linguagem) ou interativamente, via dispositivo de entrada (p.e., *mouse*). O controle procedural requer que o animador programe o comportamento dos objetos e suas interações. Atores se utilizam de conceitos da teoria de orientação a objetos, como objetos e mensagens. *Tracking live action* corresponde a registrar imagens dos movimentos de pessoas ou animais, extraindo daí informações para interpolação de quadros intermediários. Os três últimos métodos baseiam-se na simulação de leis físicas.

A-1.6 Classificação do protótipo INTERPOL3D segundo os critérios apresentados

Quanto à complexidade e poder de expressão, INTERPOL3D não se enquadra rigorosamente em nenhum dos 5 níveis descritos, uma vez que objetivou exclusivamente a obtenção de efeitos de interpolação de formas. Entretanto, situa-se entre os níveis 2 e 3, podendo facilmente incorporar as demais características que definem o nível 3.

As seqüências animadas são produzidas quadro a quadro (limitação imposta pelo *hardware* utilizado) e suas apresentações baseiam-se no uso da técnica de *scroll animation*. O protótipo utiliza-se de animação modelada e de uma linguagem gráfica com controle explícito.

BIBLIOGRAFIA

[ALE 35] ALEXANDROFF, Paul; HOPF, Heinz. Topologische invariazsätze und anschließende begriffsbildungen. In: - . **Topologie**. Berlin: Springer-Verlag, 1935. chap. 3, p.313-408.

[BAG 89] BAGGIO, A.; ZORZO, A.: ROSA, F. GIM: gerador de interfaces orientadas a menus. **In**: SEMINÁRIO INTERNO DE MICROELETRÔNICA, 4., 1989, Tramandaí. **Anais ...** Porto Alegre: CPGCC, 1991. 137p. p.30-33.

[BAR 87] BARTELS, R.; BEATTY, B.; BARSKY, B. **An Introduction to splines for use in computer graphics & geometric modellig**. San Mateo: Morgan Kaufmann Publishers, 1987. 476p.

[BEN 75] BENTLEY, J. Multidimensional binary search trees used for associative searching. **Communications of the ACM**, New York, v.18, n.9, p.509-517, Sept. 1975.

[BET 89] BETHEL, Wesley ; USELTON, Samuel. Shape distortion in computer-assisted keyframe animation. **In:** STATE-OF-THE-ART IN COMPUTER ANIMATION, 1989, Tokyo. **Proceedings ...** Tokyo: Springer-Verlag, 1989. 224p. p.215-227.

[BLI 78] BLINN, James. Simulation of wrinkled surfaces. **In:** SIGGRAPH 78 CONFERENCE, 11., August 23-25, 1978, Atlanta. **Proceedings ...** New York: ACM, 1978. p.286-292.

[BUR 71] BURTNYK, Nestor; WEIN, Marceli. Computer-generated keyframe animation. **Journal of Society for Motion Picture and Television Engineers**, Washington, v.80, n.3, 149-153, Mar. 1971.

[CAR 69] CARNAHAN, B.; LUTHER, H; WILKES, J. Newton's methods. In: - . **Applied numerical methods**. New York: John Wiley & Sons, 1969. chap.3, p.171.

[CAT 78] CATMULL, Edwin. The Problems of computer-assisted animation. **In:** SIGGRAPH 78 CONFERENCE, 11., August 23-25, 1978, Atlanta. **Proceedings ...** New York: ACM, 1978. p.348-353.

[CRO 87] CROW, Frank. The Origins of the teapot. **IEEE Computer Graphics & Applications**, Washington, v.7, n.1, Jan. 1987, p.8-14.

[DAV 88] DAVIS, Philip and HERSH, Reuben. **O Sonho de Descartes**. Rio de Janeiro: Editora Francisco Alves, 1988. 335p.

[DRE 89] Dreaming machine: the visual components, 1989 (gravação de vídeo).

[FAR 90] FARIN, Gerald. **Curves and surfaces for computer aided geometric design**: a practical guide. 2. ed. San Diego: Academic Press, 1990. 444p.

[FOL 90] FOLEY, James et al. **Computer graphics**: principles and practice. 2. ed. Reading: Addison Wesley, 1990. 1174p.

[GRA 41] GRANVILLE, W.; SMITH, P.; LONGLEY, W. The Definite integral. In: - . **Elements of the differential and integral calculus**. Boston: The Atheneum, 1941. chap. 14, p.242.

[LIM 77] LIMA, Elon. **Espaços métricos**. São Paulo: Edgard Blücher, 1977. 299p.

[LOP 88] LOPES, Pedro C. F. **SARA**: um sistema de animação por computador. Lisboa: Instituto Superior Técnico, Universidade Técnica de Lisboa, 1988. Diss. mestr.

[MAN 88] MÄNTYLÄ, Martti. **An introduction to solid modeling**. Rockville: Computer Science, 1988. 401p.

[MAT 91] UFRGS, Instituto de Informática, Curso de Pós-graduação em Ciência da Computação. M.A.T.E. Porto Alegre, 1991 (gravação de vídeo).

[MOR 85] MORTENSON, Michael. **Geometric modeling**. New York: John Wiley, 1985. 763p.

[OLI 91a] OLIVEIRA NETO, Manuel M.; LASCHUK, Anatólio. Um algoritmo para interpolação de formas entre objetos descritos por superfícies spline. **In:** CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 11. , 1991, Santos. **Anais ...** Santos: SBC, 1991. 372p. p.18-32.

[OLI 91b] OLIVEIRA NETO, Manuel M.; LASCHUK, Anatólio. INTERPOL3D - sistema de animação por interpolação de formas de objetos descritos por superfícies Bézier. **In:** SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, 4. , 1991, São Paulo. **Anais ...** São Paulo: SIBGRAPI, 1991. 263p. p.101-113.

[OLI 91c] OLIVEIRA NETO, Manuel M.; BASTOS, Rui M. R. BEZIER4D - ambiente para modelagem, visualização e animação tridimensional. **In:** SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, 4. , 1991, São Paulo. **Anais ...** São Paulo: SIBGRAPI, 1991. 263p. p.143-154.

[PER 85] PERLIN, k. An Image synthesizer. **In**: SIGGRAPH 85 CONFERENCE, July 22-26, 1985, San Francisco. **Proceedings ...** New York, ACM, 1985. p.287-296.

[PRE 85] PREPARATA, Franco; SHAMOS, Michel. **Computational geometry**: an introduction. New York: Springer-Verlag, 1985. 390p.

[PUE 88] PUEYO, Xavier and TOST, Daniela. A Survey of computer animation. **Computer Graphics Forum**, Amsterdam, v.7, n.4, Dec.1988. p.281-300.

[REE 81] REEVES, William. Inbetweening for computer animation utilizing moving point constraints. **In:** SIGGRAPH 81 CONFERENCE, 15., 1981, Dallas, Texas. **Proceedings ...** New York: ACM, 1981. p.263-269.

[ROG 85] ROGERS, David. **Procedural elements for computer graphics**. Singapore: McGraw- Hill, 1985. 433p.

[THA 85] THALMANN, Nadia; THALMANN, Daniel. **Computer animation: theory and practice**. Tokyo: Springer-Verlag, 1985. 240p.

[TRE 85] TREMBLAY, Jean; SORENSEN, Paul. **An Introduction to data structures with applications**. Singapore: McGraw-Hill, 1984. 861p.

[WAL 91] WALTER, Marcelo. **A Obtenção de texturas na síntese de imagens realísticas num ambiente limitado**. Porto Alegre: CPGCC - UFRGS, 1991. Diss. mestr.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

"Um algoritmo para interpolação de formas entre objetos modelados por superfícies SPLINE".

Dissertação apresentada aos Srs.

Prof. Dr. Léo Pini Magalhães (UNICAMP)

Prof. Anatólio Laschuk

Profa. Carla Maria Dal Sasso Freitas

Prof. Dr. Marcos Sebastiani (Inst. de Matemática-UFRGS)

Visto e permitida a impressão

Porto Alegre, 19/09/92

Prof. Anatólio Laschuk
Orientador

Prof. Dr. Ricardo Augusto da L. Reis
Coordenador do Curso de Pós-Graduação
em Ciência da Computação

SMSG/CCB \DIVCPGCC\ASSINA.BAN