LUIZ CRISTOVÃO GOMES COELHO

Modelagem de Cascas com Interseções Paramétricas

TESE DE DOUTORADO

Departamento de Informática
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
Rio de Janeiro, Agosto de 1998

Luiz Cristovão Gomes Coelho

Modelagem de cascas com interseções paramétricas

Tese apresentada ao Departamento de Informática da PUC-Rio como parte dos requisitos para a obtenção do título de Doutor em Informática: Ciência da Computação.

Orientador: Marcelo Gattass
Co-orientador: Luiz Henrique de Figueiredo

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 11 de agosto de 1998

À minha família

Agradecimentos

Ao Prof. Marcelo Gattass agradeço os ensinamentos de Computação Gráfica e o apoio demonstrados não apenas durante esta tese, mas em toda a minha carreira como pesquisador e profissional de Ciência da Computação.

Ao Luiz Henrique de Figueiredo, agradeço a valiosa convivência ao longo de toda a tese, as aulas de geometria computacional, programação C e LaTeX, as lições de objetividade, a participação na preparação do texto da tese, o incentivo e a confiança demonstrados nos momentos mais cruciais do trabalho, sem o que esta tese não teria sido concluída e nem teria sido tão prazerosa.

A toda minha família, em particular a minha querida esposa Beatriz e ao meu filho Arthur pelo apoio e compreensão demonstrados durante o trabalho da tese.

Ao Prof. Luiz Fernando Martha por ter me ensinado programação estruturada, pelas aulas de representação paramétrica de superfícies, pelo incentivo dado à minha área de pesquisa, e ao desenvolvimento do MG, inclusive com a cessão dos algoritmos de geração de malhas e do algoritmo de interseção entre uma reta e um retalho paramétrico.

Ao Prof. Paulo Cezar Carvalho pelas idéias, em particular pela sugestão de uso de árvores de ordenação espacial para a construção otimizada de subdivisões planares topológicas.

Ao Waldemar Celes pelo apoio, idéias e implementação da biblioteca de B-trees, além do companheirismo e amizade.

Às Profas. Leila de Floriani e Paola Magillo do Departamento de Ciência da Computação da Universidade de Gênova, Itália, pela cessão da implementação dos operadores topológicos da estrutura de dados DCEL.

Ao Ivan Fábio Motta de Menezes pelas lições de geometria vetorial, pelo apoio sólido em métodos de otimização e programação matemática e pela amizade.

Ao Maurício Riguette Mediano pelas sugestões e implementação da biblioteca de R^*-trees e pelas aulas de bancos de dados e utilização de árvores em geral.

À Profa. Clarisse Sieckenius de Souza pelos ensinamentos de ciências cognitivas e se-

mióticas, e pela orientação no trabalho de restruturação da interface do MG.

Aos amigos e companheiros Camilo Freire, Márcio Santi, Eduardo Setton, Marcelo Tílio, André Costa, Renato Borges, Carlos Levy, Peter Hohl, Paulo Sedrez, André Derraik, Renato Cerqueira, Carlos Cassino, André Clinio, Joao Luiz Campos, Joaquim Bento Cavalcante, Ovidio Goulart, Roberto de Beauclair Seixas, e a todos no TeCGraf pelo apoio irrestrito e convivência saudável.

Ao Eng. Isaías Quaresma Masetti do CENPES/Petrobras pelas aulas de engenharia Naval e apoio dado ao meu trabalho desde os tempos de mestrado.

Ao pessoal do CENPES, nas pessoas de Marco Antônio Tetkovic, Mauro Costa de Oliveira, Eduardo Vardaro, do DIPREX-SEPRON, que tiveram participação efetiva na especificação, teste e no projeto do programa MG, e ao Marcos Donato por ter sido o único testador da versão Linux.

Ao CAPES, CNPq e a Fundação Padre Leonel Franca pelo auxílio financeiro.

Resumo

Apresenta-se uma metodologia para modelagem de cascas para elementos finitos definidas em superfícies paramétricas. A metodologia consiste na criação de curvas e geração de malhas sobre os retalhos paramétricos construídos com base nestas curvas, que também são usadas para a conexão de malhas adjacentes. O modelo final é uma representação de todas as malhas combinadas em uma única estrutura de dados.

As ferramentas básicas para geração de tais malhas são uma interface para modelagem de curvas espaciais e os algoritmos geométricos para construção de mapeamentos nos domínios elementares. O problema central em modelagens compostas é o tratamento dado às malhas em superfícies que se interceptam. Um algoritmo capaz de modelar com precisão as curvas de interseção e de ajustar as duas malhas para as novas restrições geradas é apresentado neste trabalho. O algoritmo é parte de um programa completo para modelagem interativa de cascas, que tem sido usado no projeto de grandes sistemas flutuantes para explotação de petróleo em águas profundas. O uso de uma variante da estrutura de dados DCEL, que usa árvores de ordenação espacial para armazenar as entidades topológicas ao invés de listas ou vetores, permite que malhas bastante refinadas sejam reconstruídas em tempo compatível com o trabalho interativo. Estas árvores aceleram os cálculos de interseção necessários à determinação dos pontos de interpolação das curvas de *trimming*, permitindo também a reconstrução das malhas usando-se apenas consultas locais.

Abstract

We present a methodology for modeling finite-element meshes defined on parametric surface patches. The idea is to build curves and generate meshes over the parametric patches built with these curves, which also connect adjacent meshes. The final model is a representation of all meshes combined into a single data structure.

The basic tools to generate such meshes are the user interface to model space curves and the geometric algorithms to construct the elementary domain mappings. The main problem in composite modeling is how to handle mesh surfaces that intersect each other. We present an algorithm that models the intersection curves precisely and adjusts both meshes to the newly formed borders. The algorithm is part of an interactive shell modeling program, which has been used in the design of large offshore oil structures. We avoid unacceptable interaction delays by using a variant of the DCEL data structure that stores topological entities in spatial indexing trees instead of linked lists. These trees speed up the intersection computations required to determine points of the trimming curves, and also allows mesh reconstruction using only local queries.

Sumário

Lista de Figuras

Lista de Pseudo-códigos

Capítulo 1

Introdução

Modelagem de superfícies e geração de malhas são problemas importantes em *Computer Aided Geometric Design*, especialmente em aplicações de engenharia, onde o método dos elementos finitos é usado para simular o comportamento de modelos de cascas. Diversos tipos de projetos de engenharia, tais como: automóveis, sistemas flutuantes, fuselagem e asas de aviões, turbinas, compressores, etc, dependem de um modelador capaz de produzir as diferentes formas geométricas e de definir malhas de elementos finitos para representar os modelos com precisão nas simulações numéricas. Muitas destas estruturas apresentam um nível de complexidade geométrica muito alto, de forma que dificilmente podem ser modeladas manualmente ou mesmo pelo desenvolvimento de algoritmos específicos improvisados para cada problema.

Em simulações numéricas adaptativas, existe a necessidade de redefinição da malha a cada passo, sem a alteração da geometria do modelo. Este processo pode alterar a malha nas fronteiras do objeto e aí surge a necessidade de se conectar o simulador numérico com o modelador geométrico [63]. Em duas dimensões, as informações de fronteira são as curvas de bordo, bastando que se tenha uma descrição geométrica contínua destas curvas, lançando-se mão de algoritmos de reconstrução da malha. Em três dimensões, as fronteiras são superfícies necessariamente curvas em algumas modelagens, e a geração adaptativa da malha, formada por elementos sólidos ou bidimensionais, exige um modelador capaz de reconstruir as malhas automaticamente.

As diversas formas de abordagem do problema de modelagem de superfícies apresentam alguns aspectos que podem ser identificados e que são decisivos para o processo de modelagem: os objetos primitivos utilizados, as estruturas de dados que os representam, a interface com o usuário, e os algoritmos geométricos de construção, são tópicos fundamentais em modelagens de superfícies, ou de retalhos de superfícies [1].

[1] Os retalhos de superfície, que indicam superfícies limitadas por curvas de bordo, são muitas vezes referenciados neste texto e em outros trabalhos apenas como retalhos.

Uma forma intuitiva de se construir modelos complexos consiste em combinar várias superfícies geradas individualmente em uma única representação, recortando as regiões excedentes determinadas pelas interseções. Para que este tipo de modelagem seja produtiva, o problema de interseção entre dois retalhos limitados deve ser resolvido de forma eficiente e robusta. Mais até do que o problema de interseção entre dois retalhos, em modelagens de estruturas compostas por várias superfícies, é muito comum ocorrerem várias interseções entre um retalho e vários outros, de forma que tais situações também devem ser tratadas. Tanto a reconstrução de malhas quanto o cálculo de interseções entre superfícies são campos ativos de pesquisa [2–4, 32, 42, 48, 49, 51, 55, 83].

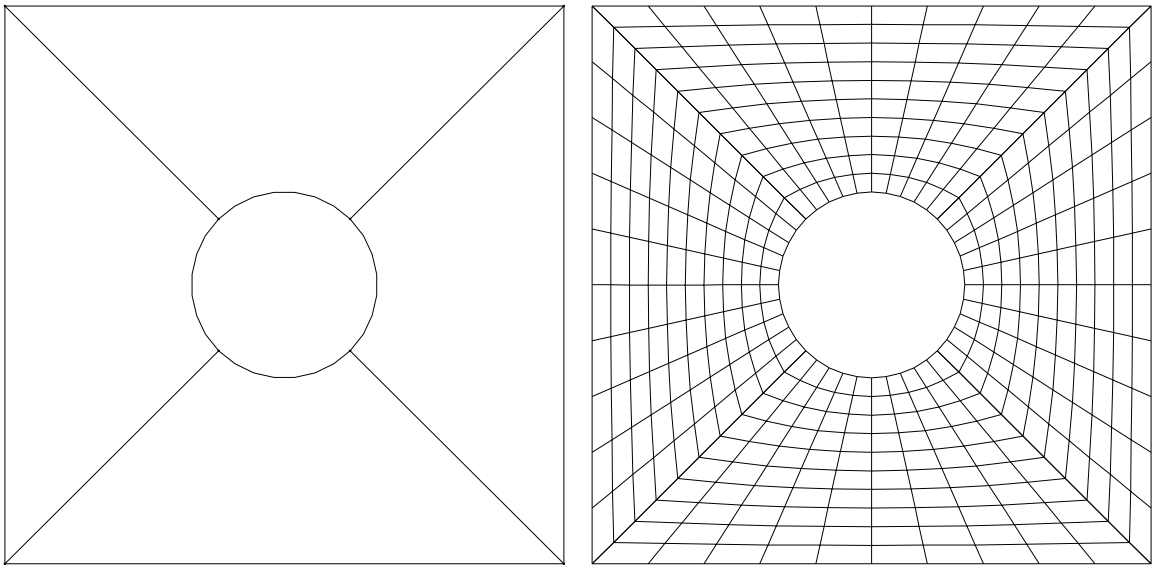
1.1 Exemplo de modelagem intuitiva

Para apresentar o tipo de modelagem que é abordado nesta tese, a Figura 1.1 mostra a construção de um modelo de placa com furo feito com as técnicas tradicionalmente usadas em duas dimensões [37, 38]. A Figura 1.1a mostra as curvas de bordo que especificam as subdivisões planares na placa. A Figura 1.1b mostra quatro malhas geradas por mapeamentos bilineares radiais feitos com as curvas de bordo. A obtenção de uma malha única que representa estes mapeamentos pode ser feita com a identificação dos vértices comuns sobre as curvas internas, e pela criação de uma única lista de vértices e de elementos. Nesta abordagem, todas as curvas devem ser especificadas antes de se aplicar a técnica de mapeamento para geração das malhas internas às regiões.

Uma alternativa para a criação deste modelo é mostrada na Figura 1.2. Tal construção é feita inicialmente pela geração de uma malha primária, que representa a placa completa, sem o furo. Em seguida, faz-se a identificação da curva que representa o furo, alterando-se localmente a malha. Os elementos internos ao furo, limitados por esta curva de recorte, são removidos e a malha está completa.

Se um novo furo deve ser definido nos modelos das Figuras 1.1 e 1.2, com as técnicas existentes [38, 64] pode-se refazer a malha usando uma triangulação de Delaunay com a geração de novos pontos no interior de toda a região mapeada, como mostra a Figura 1.3a. A Figura 1.3b mostra como a malha da Figura 1.2 seria redefinida com a modelagem alternativa por alterações locais e remoção dos elementos internos ao novo furo.

A técnica de modelagem adotada neste trabalho usa retalhos tridimensionais, onde as malhas primárias definidas sobre as superfícies são alteradas localmente pelo algoritmo de interseção desenvolvido (Capítulo 3). Este algoritmo considera tanto as representações paramétricas das duas superfícies envolvidas quanto as malhas definidas sobre elas. As regiões a serem recortadas



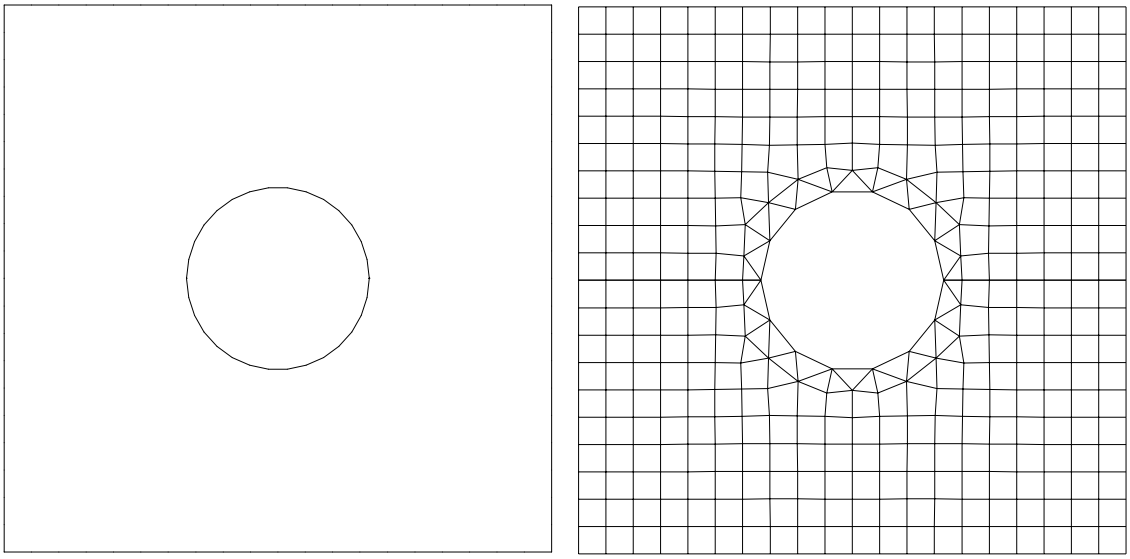
(a) Curvas de bordo das subregiões. (b) Malhas bilineares.

Figura 1.1: Modelagem tradicional da placa com furo com malhas radiais bilineares.

são identificadas posteriormente pelo usuário, por apontamento simples. As malhas mostradas nas Figuras 1.2b e 1.3b foram geradas com esta técnica. A Figura 1.4 mostra as curvas usadas na definição da malha da superfície planar da Figura 1.3b, e dos cilindros que atravessam a malha plana. As malhas das superfícies cilíndricas ortogonais à placa foram usadas nas definições das curvas de interseção e nas alterações locais da malha, tendo sido removidas para facilitar a visualização.

O exemplo da superfície planar apenas exemplifica o processo de modelagem utilizado, pois o objetivo desta tese é a construção de superfícies de cascas compostas tridimensionais com variadas representações paramétricas. Objetiva-se tratar, principalmente, os casos em que as curvas de interseção são muito difíceis de serem construídas interativamente, e também onde a redefinição das superfícies com as curvas de interseção produzem deformações indesejáveis do modelo original, como é mostrado no Capítulo 3. A Figura 1.5 mostra um encontro típico entre duas cascas que exemplifica a dificuldade de modelagem interativa da região de interseção. A parte central da placa cilíndrica, que apenas pode ser identificada após os cálculos de interseção, foi removida para facilitar a visualização da malha na superfície tubular mostrada na Figura 1.5.

Como é apresentado ao longo desta tese, esta técnica de modelagem se mostra produtiva em três dimensões pois a reparametrização das superfícies usando-se as curvas de *trimming* pode não produzir representações satisfatórias, além de ser muito difícil especificar novas regiões para aplicação das técnicas de mapeamentos elementares em muitos casos.



(a) Curvas de bordo e *trimming*.

(b) Malha após interseção e remoção do furo central.

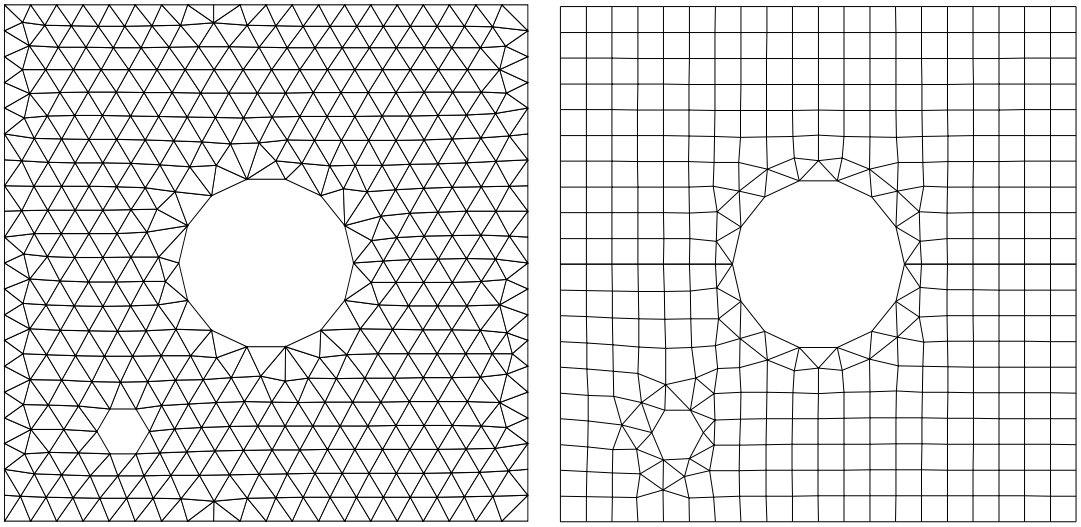
Figura 1.2: Modelagem da placa com furo usando-se a técnica proposta.

1.2 Trabalhos correlatos

Dois métodos básicos de interseção entre superfícies paramétricas têm sido usados em aplicações de engenharia: os métodos de marcha ou continuação [2–4, 77], e os métodos de decomposição [32, 42, 48]. Nos métodos de marcha, determina-se a curva de interseção caminhando-se pelas superfícies em três dimensões, estimando-se a direção do vetor tangente à curva em cada passo. Nos métodos de decomposição, subdividem-se os espaços paramétricos dos retalhos em retângulos, usando-se os *bounding boxes* relativos a estes intervalos para se testar as interseções. A solução é refinada até que uma determinada precisão seja alcançada. Esta precisão é proporcional ao tamanho das duas superfícies envolvidas, e à complexidade das curvas de interseção. Uma discussão mais detalhada sobre estes métodos é apresentada na Seção 2.3.

Os trabalhos sobre interseções paramétricas de superfícies abordam o problema do ponto de vista geométrico, ou seja, da definição precisa das curvas de interseção [3], tentando contornar as dificuldades numéricas envolvidas. Os trabalhos que apresentam técnicas de reconstrução das malhas sobre as superfícies determinam as curvas de interseção e fazem a completa redefinição da malha, muitas vezes usando técnicas de triangulações [51].

Uma outra abordagem é feita por Lo [55], que considera malhas de elementos triangulares para a representação das superfícies envolvidas, fazendo a reconstrução local destas malhas pela redefinição dos elementos na região de interseção, de forma a considerar as curvas de interseção,



(a) Triangulação de Delaunay. (b) Alteração local da malha.

Figura 1.3: Modelagens da placa com dois furos.

que são determinadas pelas interseções entre os elementos planares. Entretanto, o algoritmo de Lo não considera as possíveis representações contínuas que as superfícies que originam as malhas podem apresentar, de forma que as curvas de interseção e, conseqüentemente, as novas faces geradas, podem não corresponder fielmente às descrições originais. Este problema é discutido também na Seção 3.1.

1.3 Contribuições originais

Esta tese aborda as questões de modelagem de retalhos compostos (ou simplesmente modelagem composta) sob os aspectos de precisão geométrica com o uso de representações paramétricas para curvas e superfícies, interface com o usuário, e estruturas de dados. A metodologia de modelagem usada se assemelha à *Constructive Solid Geometry* [46], mas usa os retalhos de superfície como objetos primitivos e mantém as malhas originais sobre estes retalhos inalteradas fora das regiões em que eles se interceptam. As malhas geradas podem ser usadas em simulações numéricas por elementos finitos de casca, pois possuem boa qualidade geométrica.

As curvas espaciais são usadas como elemento básico de modelagem das superfícies. Considera-se um ambiente onde os retalhos são definidos por curvas paramétricas de bordo. De particular importância no tipo de modelagem abordada nesta tese, as superfícies de Coons [27] são definidas por curvas em todos os bordos e, em muitos casos, representam satisfatoriamente a intenção de modelagem em superfícies definidas por seções transversais, como é o caso dos

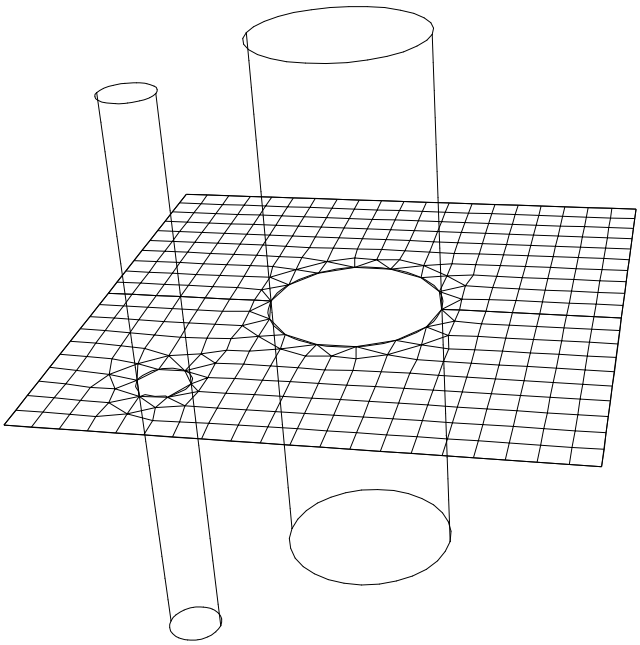


Figura 1.4: Curvas usadas na modelagem da placa com dois furos.

cascos de muitos sistemas flutuantes. O usuário define discretizações para as curvas de bordo que são usadas para interpolar uma malha para cada superfície. A malha final é a união de todas as malhas individualmente geradas sobre cada retalho. As superfícies que se interceptam têm as suas malhas automaticamente redefinidas nas regiões de interseção, e as regiões distintas em cada malha podem ser identificadas facilmente permitindo a remoção das regiões que não fazem parte da malha final.

Apresenta-se também um algoritmo totalmente automático para resolução do problema de interseção entre malhas sobre superfícies paramétricas. Uma das principais características responsáveis pela eficiência desse algoritmo é que as buscas necessárias ao cálculo das curvas de interseção e reconstrução das malhas são apoiadas por uma estrutura de dados topológica cujas principais propriedades são o uso do espaço paramétrico das superfícies para orientação das entidades e o uso de árvores de busca B-trees [26] e R^*-trees [8] para o armazenamento destas entidades ao invés de listas encadeadas. Estas árvores de indexação espacial contribuem decisivamente para a eficiência global do algoritmo. O algoritmo de interseções foi implementado e é parte de um programa completo de modelagem de cascas e geração de malhas desenvolvido ao longo da tese: o MG (*Mesh Generator*), que têm sido usado com sucesso no projeto de grandes sistemas flutuantes para explotação de petróleo pelo Centro de Pesquisas da Petrobras (CENPES).

Esta tese enfoca também alguns aspectos relativos à geometria de curvas e superfícies, e aos problemas de interface para a geração de cascas compostas. Discutem-se os problemas de interpolação, interface de construção e representação de curvas espaciais. Apresenta-se uma me-

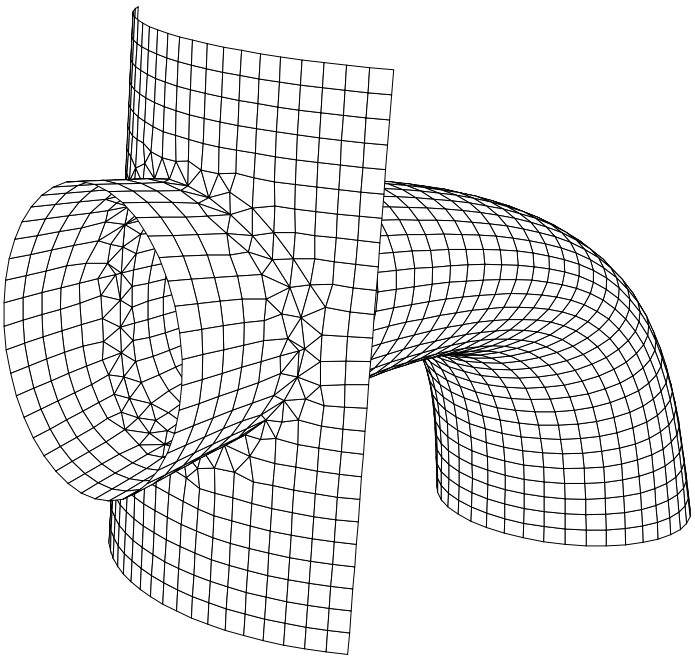


Figura 1.5: Interseção entre cascas apresentando curva não planar.

todologia para se definir *sweeps* genéricos, usando-se apenas duas curvas e suas subdivisões. Os aspectos de interface abordados nesta tese apresentam em comum o paradigma de manipulação direta, que foi adotado na implementação do MG. O modelo de interface utilizado é resultado de um trabalho de avaliação feito com testes-piloto de modelagem, apresentados pelo autor e de Souza em outro trabalho [22]. A estrutura de dados usada no MG é concisa o suficiente para representar uma grande variedade de estruturas flutuantes e peças mecânicas, projetadas pelos engenheiros do CENPES.

1.4 Organização da tese

O Capítulo 2 descreve as questões relativas à modelagem de cascas, apresentando os aspectos geométricos envolvidos na modelagem de curvas e superfícies, os problemas e soluções relativos à interface com o usuário para estas construções, e as características das estruturas de dados relacionadas ao problema. Ao final de cada seção, são mostradas as técnicas adotadas na implementação do MG. Ainda neste capítulo, é feita uma revisão dos métodos de interseção entre superfícies paramétricas que tratam o problema do ponto de vista geométrico.

O Capítulo 3 descreve o algoritmo original desenvolvido, apresentando como são determinadas as interseções entre duas superfícies, e como são reconstruídas as malhas sobre estas superfícies. É apresentada a estrutura topológica adaptada para conectar as malhas de duas su-

perfícies que se interceptam, e também como o uso das árvores de indexação espacial reduz o tempo de processamento dos algoritmos usados. Os dois apêndices desta dissertação estão relacionados diretamente com o algoritmo de interseção e mostram os operadores topológicos que mantêm consistentemente a representação bidimensional sobre as superfícies, e também como pode ser feita a construção inicial partindo-se de uma estrutura comum de elementos finitos.

O Capítulo 4 mostra alguns exemplos do funcionamento do algoritmo de interseção desenvolvido com testes entre pares de superfícies, e modelagens compostas de malhas sobre superfícies que se interceptam gerando vários componentes conexos, com ênfase para os modelos de estruturas flutuantes e peças mecânicas projetados no CENPES.

O Capítulo 5 mostra as conclusões e as observações importantes sobre o trabalho desenvolvido. Algumas propostas para novas pesquisas com base nas técnicas propostas encerram a tese.

Capítulo 2

Modelagem de cascas

A descrição da superfície completa de muitos modelos usados em simulações por elementos finitos é facilitada pelo uso de representações por partes. Esta característica se acentua em modelagens de sistemas flutuantes, tais como navios e plataformas semi-submersíveis, que são naturalmente formados por partes distintas (vide Figura 2.1). Um aspecto importante da modelagem é que os encontros entre as superfícies que compõem o modelo devem estar bem representados na malha de elementos finitos ou de contorno usada na simulação numérica de projeto. Deve-se usar então uma metodologia para se conectar as diversas superfícies distintas que formam as partes para se descrever o modelo completo com uma única representação de elementos finitos ou de contorno.

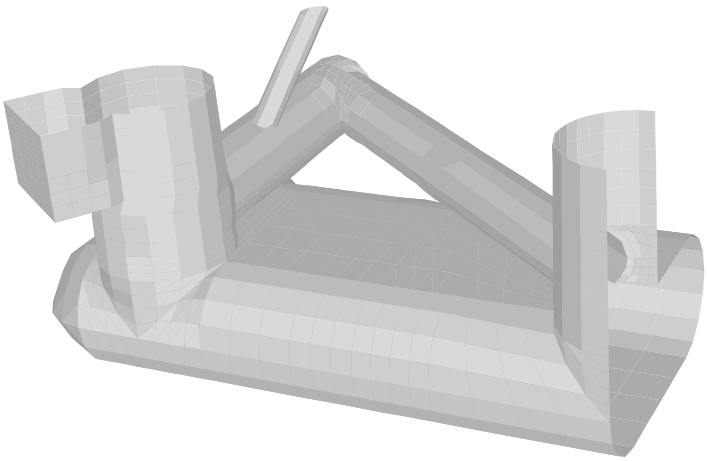


Figura 2.1: Modelo imerso de plataforma semi-submersível.

O problema de modelagem de retalhos compostos pode ser resolvido com um *software* que possua uma boa interface para construção de curvas espaciais. Com base nestas curvas, as superfícies podem ser geradas com o apoio de uma estrutura de dados capaz de representar

concisamente os retalhos individuais e suas adjacências. É particularmente importante que esta estrutura dê suporte aos algoritmos de construção e interseção entre os retalhos. Muitos aspectos estão envolvidos com o processo de modelagem de cascas compostas, como a mostrada na Figura 2.1. Dentre eles, os mais relevantes são:

1. Modelagem de curvas e superfícies;
2. Geração de malhas para elementos finitos;
3. Interface com o usuário;
4. Algoritmos geométricos (interseção e interpolação); e
5. Estruturas de dados.

Neste capítulo, descreve-se as questões de modelagem de cascas envolvidas com cada um destes aspectos. A questão da interseção e reconstrução das malhas em superfícies paramétricas é abordada detalhadamente no Capítulo 3, uma vez que a proposta lá apresentada constitui o núcleo central desta tese. Ao final de cada seção deste capítulo, são apresentadas as técnicas escolhidas para a implementação do MG.

Inicialmente, apresentam-se os tópicos geométricos e de construção interativa referentes à modelagem de curvas espaciais e superfícies (Seções 2.1 e 2.2), tendo como paradigma básico a manipulação direta [22]. A Seção 2.3 apresenta uma revisão dos métodos de determinação das interseções entre superfícies paramétricas que não tratam do problema de malhas. As estruturas de dados para representação da modelagem composta são abordadas na Seção 2.4.

2.1 Modelagem de curvas

A entidade geométrica que representa os bordos e as interseções entre as superfícies são as curvas espaciais. A utilização de curvas na definição de superfícies é uma metodologia bastante difundida entre as modelagens baseadas em domínios elementares [44], ou em modelagens *free-form* [15, 43]. As técnicas de modelagem que usam curvas para definir as superfícies e as malhas poliedrais de elementos finitos geralmente possuem discretização associada com as subdivisões das curvas geradoras. Algumas funcionalidades importantes podem ser identificadas para as curvas em modelagens de superfícies para elementos finitos:

1. Fornecer informações para a definição geométrica de cada superfície, individualmente;
2. Limitar as superfícies, definindo retalhos paramétricos e regiões distintas;
3. Representar o contato geométrico contínuo entre retalhos que se interceptam; e
4. Gerar e controlar o espaçamento dos pontos que fazem parte das malhas de todos os retalhos a ela adjacentes.

As curvas podem não apresentar uma boa descrição parametrizável a priori, como é o caso de curvas que não são continuamente diferenciáveis, normalmente resultantes de interseções entre retalhos com formulações distintas. Além disso, a modelagem interativa de curvas no espaço não é uma tarefa simples, pois a especificação das posições em 3D envolve problemas como a projeção do espaço tridimensional para o plano da tela do computador, a especificação da profundidade da projeção, os problemas de *feedback*, interfaces para seleção de entidades gráficas e transformações geométricas, dentre outros. Estes aspectos são discutidos na Seção 2.1.3.

Apesar destas dificuldades, modeladores comerciais, como o PATRAN [56] e o ANSYS [66], apresentam soluções consistentes para a modelagem de curvas espaciais. Estes modeladores não tratam, entretanto, do problema de interseção e reconstrução das malhas associadas com a descrição paramétrica das superfícies.

Algumas curvas são mais facilmente modeladas pela interseção de duas superfícies simples do que pela descrição dos pontos de interpolação. A Figura 2.2 mostra uma curva de interseção que não é continuamente diferenciável, resultante da interseção de dois retalhos com formulações distintas (as faces da região de *trimming* de um dos retalhos foram suprimidas do desenho para facilitar a visualização da curva).

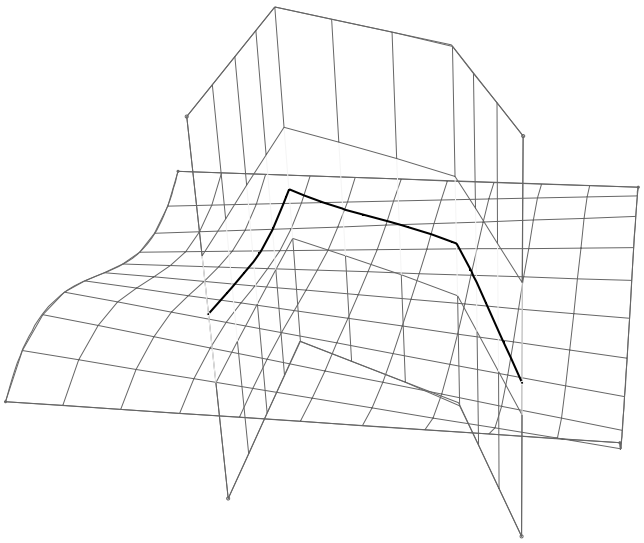


Figura 2.2: Curva de interseção entre superfícies distintas.

2.1.1 Interpolação paramétrica

O problema clássico de interpolação de curvas passando por pontos persiste como um tópico importante de pesquisas [52]. Estabelecendo uma nomenclatura bem simples para o problema,

tem-se:

$$P(t_i) = P_i, \quad 0 \leq i \leq n \ , \tag{2.1}$$

onde os P_i são os $n+1$ pontos de interpolação fornecidos (também chamados de pontos amostrais), e $0 = t_0 < t_1 < \ldots < t_n = 1$ são os parâmetros escolhidos para cada um destes pontos.

Dependendo do tipo de curva e da parametrização utilizados, obtêm-se resultados bastante distintos. Como pode ser demonstrado pela teoria de vigas (vide *duck spline* [74]), polinômios cúbicos representam muito bem B-splines que passam por pontos que funcionam como restrições ou apoios. Além de ser desejável em modelagens interativas, o controle local promove o desacoplamento do sistema de equações, necessário à determinação dos parâmetros de B-splines cúbicas. Este fato torna o cálculo bastante rápido (o sistema de equações que resolve o problema é tri-diagonal), sendo recomendável para alterações promovidas por eventos do *mouse*, tais como movimentação, inserção ou remoção de novos pontos de controle.

Parametrizações

Muitos trabalhos enfocam o problema da escolha das parametrizações para definição de B-splines interpolantes baseando-se nos objetivos da modelagem e, muitas vezes, em heurísticas específicas para cada caso. A escolha mais imediata é a parametrização *uniforme*, que considera os trechos igualmente espaçados, ou seja:

$$t_i = i/n \ . \tag{2.2}$$

Se os pontos amostrados para determinação da curva possuem espaçamento (distância euclidiana) constante, o que raramente se pode garantir, esta parametrização fornecerá bons resultados.

Como normalmente a parametrização uniforme é insatisfatória, uma boa alternativa é ponderar os parâmetros baseando-se na distância relativa entre os pontos consecutivos (*chord length*). Assim tem-se:

$$t_i - t_{i-1} = \frac{|P_i - P_{i-1}|}{\sum_{j=1}^{n} |P_j - P_{j-1}|} \ . \tag{2.3}$$

Com pontos igualmente espaçados, as equações 2.2 e 2.3 fornecem parâmetros idênticos.

Lee [52] apresenta um novo tipo de parametrização, chamada de *centrípeta*, que através de limitações impostas na definição do cálculo das variações dos parâmetros considera, além da distância, a "força centrípeta" relacionada com as curvaturas experimentadas nos pontos amostrais. Lee questiona se uma parametrização pelo comprimento do arco (*arc length*), que pode ser obtida através de iterações repetidas [1, 12, 28, 82], fornece a melhor forma por representar

a trajetória de uma partícula com velocidade constante. Lee argumenta que, quando um carro experimenta uma trajetória que passa por pontos desalinhados, não tenta naturalmente fazê-lo com velocidade constante. Na realidade, a velocidade é proporcional ao conforto que o motorista experimenta ao dirigir com segurança, sendo dependente da aceleração centrípeta. Baseando-se nestes aspectos, Lee propõe a seguinte formulação:

$$t_i - t_{i-1} = \frac{|P_i - P_{i-1}|^{1/2}}{\sum_{j=1}^{n-1} |P_j - P_{j-1}|^{1/2}} \ . \tag{2.4}$$

Esta formulação, com pontos espaçados desigualmente, fornece resultados melhores, do ponto de vista de variação suave das curvaturas, do que as parametrizações uniforme, *chord length*, e *arc length*. De novo, se os pontos forem igualmente espaçados, os resultados de 2.2, 2.3, e 2.4 são os mesmos.

Uma outra parametrização, apresentada por Foley e Nielson [40], define:

$$t_i - t_{i-1} = d_i \left[1 + \frac{3}{2} \frac{\hat{\Theta}_i d_{i-1}}{d_{i-1} + d_i} + \frac{3}{2} \frac{\hat{\Theta}_{i+1} d_{i+1}}{d_i + d_{i+1}} \right], \tag{2.5}$$

onde $d_i = |P_i - P_{i-1}|$, $\hat{\Theta}_i = \min(\pi - \Theta_i, \pi/2)$, e Θ_i é o ângulo formado por P_{i-1}, P_i, P_{i+1}. O valor $\hat{\Theta}_i$ é chamado de *ângulo exterior ajustado*; à medida em que $\hat{\Theta}_i$ cresce, a variação paramétrica aumenta de um intervalo que está entre 1 e 4 vezes o valor do comprimento da corda.

Uma propriedade que distingue a parametrização uniforme das outras três é que ela é a única invariante sob transformações afins, em sua formulação básica. Nielson [65] mostra como uma normalização no cálculo das distâncias torna as parametrizações dadas pelas Equações 2.3, 2.4 e 2.5 invariantes sob transformações afins.

Farin [35] mostra algumas comparações entre as parametrizações citadas, com exemplos bastante elucidativos, no que diz respeito à variação das curvaturas experimentadas pelas curvas quando se interpola pontos desigualmente espaçados. Com base nestes resultados, decidiu-se usar no MG curvas B-splines cúbicas por partes com continuidade C^2 (com relação ao parâmetro global que as define), e a parametrização dada pela Equação 2.5.

Condições de contorno

Supondo-se n segmentos cúbicos entre $n+1$ pontos de interpolação (vide Figura 2.3), têm-se $4n$ graus de liberdade a serem determinados no sistema de equações formado. Se a resolução do sistema for tratada como curvas de Bézier locais, devem ser obtidos quatro pontos de controle em cada um dos n trechos. As propriedades interessantes que se apresentam por herança das curvas de Bézier (fecho convexo, controle local, invariância por transformações afins, simetria, precisão linear, interpolação linear e variação reduzida) justificam a adoção deste tipo de representação.

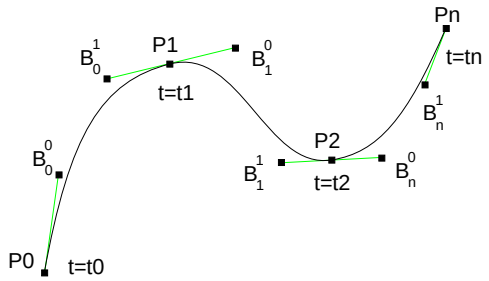


Figura 2.3: Curva interpolante com controle local por Béziers em cada trecho.

As condições de contorno do problema fornecem as restrições que tornam o sistema determinado. Os pontos de interpolação são os mesmos para cada um dos trechos adjacentes, o que fornece $2n$ restrições ao problema. Mais $2(n-1)$ restrições são obtidas com as condições de continuidade paramétrica da primeira e segunda derivadas em cada ponto de conexão. Restam dois graus de liberdade a restringir para que o sistema fique determinado. Estas duas condições adicionais podem ser associadas às curvaturas ou às tangentes nos pontos extremos, $P(t_0)$ e $P(t_n)$. Quando os vetores tangentes nos dois extremos são conhecidos, o sistema fica determinado, e este tipo de interpolação recebe o nome de *clamped ends*. Adotou-se a implementação da condição *natural* ou *relaxada*, que corresponde a curvaturas nulas nos extremos, ou seja: $P^{''}(t_0) = 0$ e $P^{''}(t_n) = 0$.

Para o tratamento de curvas e superfícies fechadas, ou que apresentam auto-interseções, este trabalho adota a filosofia de particionamento. Todas as curvas fechadas devem ser subdivididas, mantendo-se as condições de contorno nos pontos de subdivisão, de forma a garantir a continuidade C^2. Este procedimento evita os problemas topológicos que inviabilizariam o uso das estruturas de dados mostradas nas Seções 2.4 e 3.3.

2.1.2 Representação da curva interpolante

Com o problema de interpolação resolvido, deve-se usar alguma forma de representação para as curvas de Bézier locais. Para se visualizar a curva, selecionar, calcular interseções, amostrar posições com espaçamento determinado, subdividir, aglutinar, etc; pode-se usar amostragens uniformes e se trabalhar com a linha poligonal equivalente. Uma outra representação, mais consistente do ponto de vista geométrico, é obtida pela consideração da curvatura experimentada em cada trecho. Para se gerar uma linha poligonal adaptativa com a curvatura, pode-se usar o algoritmo de de Casteljau [29, 30], ou uma alternativa eficiente proposta por Figueiredo [31] (para Béziers, ver Chandler [20]). A Figura 2.4 mostra como se determina um ponto na curva com as interpolações lineares do algoritmo de de Casteljau, onde cada ponto de interpolação

intermediário é dado por: $b_i^k = (1-t)b_i^{k-1} + tb_{i+1}^{k-1}$.

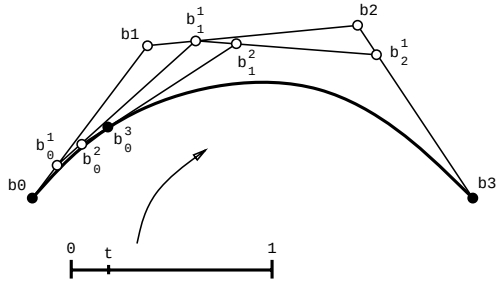


Figura 2.4: Algoritmo de de Casteljau para Bézier cúbica.

O Pseudo-código 2.1 mostra a formulação recursiva do algoritmo de de Casteljau desenvolvida por de Boor [28]. O teste *Colinear* é feito usando-se uma tolerância relativa a um ângulo pequeno. O algoritmo *BranchLeftRight* subdivide os três segmentos de reta definidos por *bez* em dois, um referente à metade esquerda (*lbez*), e o outro referente ao lado direito (*rbez*), como mostra a Figura 2.5.

Pseudo-código 2.1 O algoritmo de de Boor [28].

```
algorithm PolydeBoor ( Point bez[4], Point poly[] )
   input: Four control points bez[4]
   output: Adaptive poygonal poly[]
begin
   Point lbez[4], rbez[4]
   if Collinear ( bez ) then
       Add middle point to poly
       return
   end if
   BranchLeftRight ( bez, lbez, rbez )
   PolydeBoor ( lbez, poly[] )
   PolydeBoor ( rbez, poly[] )
end PolydeBoor
```

Para se obter uma poligonal adaptativa da curva B-spline cúbica completa basta fazer chamadas em seqüência para cada um dos n segmentos individuais, informando os quatro pontos de controle locais para o algoritmo *PolydeBoor*. A chamada recursiva com *lbez* antes de *rbez* faz com que os pontos sejam organizados na poligonal no mesmo sentido de percurso de cada trecho. A Figura 2.6a mostra a poligonal adaptativa gerada para representar uma B-spline cúbica contendo três trechos e quatro pontos de interpolação. Os pontos de controle das curvas de

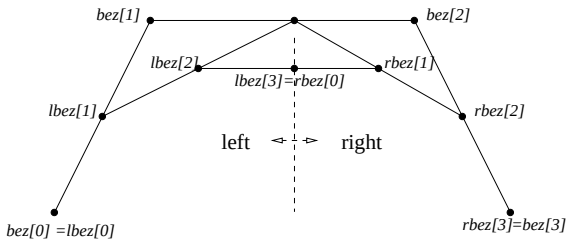
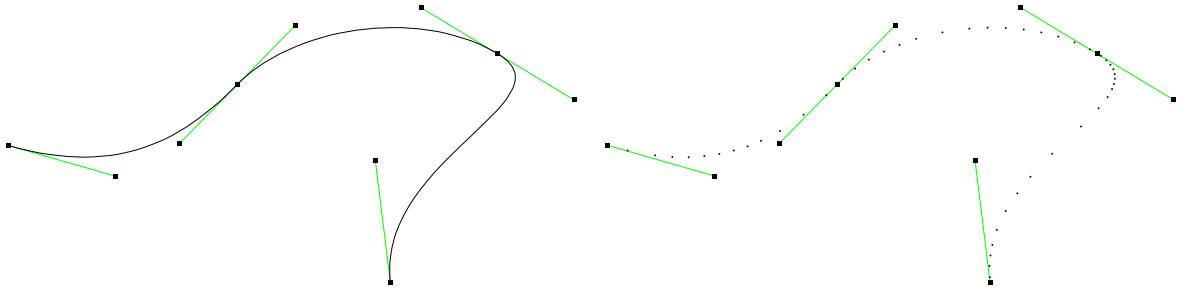


Figura 2.5: Subdivisão da busca em *PolydeBoor* feita por *BranchLeftRight*.

Bézier locais são mostrados com marcas quadradas. A Figura 2.6b mostra os pontos que definem a poligonal adaptativa, que se concentram nas regiões onde a curvatura é mais acentuada.



(a) Segmentos de reta.

(b) Pontos.

Figura 2.6: Poligonal adaptativa com a curvatura representando B-spline cúbica.

O comprimento total da poligonal adaptativa com a curvatura que representa a curva completa aproxima o comprimento desta curva, o que viabiliza a determinação de uma parametrização global do tipo *chord length* para esta poligonal. Pode-se extrair informações sobre os vetores tangente em qualquer ponto, além de se estimar as posições para a subdivisão da curva. Esta subdivisão é usada na discretização das malhas geradas sobre as superfícies construídas com base nestas curvas. Em malhas de elementos finitos são comuns as subdivisões em intervalos de mesmo comprimento ou com tamanhos variáveis, usando-se uma progressão aritmética ou geométrica para especificar a variação. A Figura 2.7 mostra a B-spline da Figura 2.6 subdividida em vinte trechos com uma progressão geométrica onde a razão entre o primeiro e o último segmentos é $1/10$.

Os tipos de curvas que podem ser criadas com este tipo de formulação são suficientes para representar satisfatoriamente as curvas de interseção e a quase totalidade das seções transversais dos sistemas flutuantes que se deseja modelar. Uma exceção são os arcos de círculo, vastamente usados na modelagem de estruturas tubulares como as colunas e os contraventamentos das plataformas semi-submersíveis, que não são bem representados com a interpolação cúbica apre-

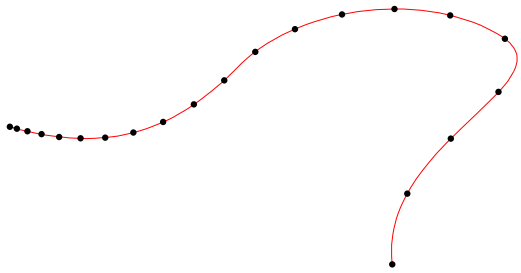


Figura 2.7: Subdivisão da B-spline interpolante com progressão geométrica.

sentada. A solução adotada neste trabalho foi a criação de três categorias de curvas: as B-splines cúbicas mostradas nesta seção, as poligonais formadas por segmentos de reta, e os arcos de círculos. Por simplicidade da representação interna, decidiu-se pela criação da classe específica para modelar poligonais formadas por segmentos de reta, que poderia ser feita com as B-splines se os pontos de controle fossem forçados a coincidir com os pontos de interpolação adjacentes.

Uma representação usando NURBS [35] resultaria em uma única formulação interna para estes três tipos de curvas. A interface, entretanto, deveria apresentar um comportamento específico para o tratamento de arcos de círculo e para as linhas poligonais, de forma a facilitar o trabalho do usuário na inserção de pontos interpolantes nestes tipos de curvas, ao invés da edição dos pontos de controle. Apesar de representarem um padrão de fato nas aplicações em *CAD*, não se adotou a representação por NURBS, que poderia unificar a representação interna de curvas.

2.1.3 Construção interativa de curvas

Os modeladores vetoriais bidimensionais [14, 53] parecem ter estabelecido padronizações, instaladas pelo uso, para a construção de curvas planares. À medida em que os pontos são fornecidos, é gerada uma curva que interpola estes pontos, e que pode ter qualquer posição ou vetor tangente alterado por manipulação direta. As transformações geométricas são feitas pela manipulação de marcas posicionadas nas fronteiras do objeto, podendo-se fazer translações, rotações, escalas ou cisalhamentos em grupos de objetos previamente selecionados. A Figura 2.8 mostra uma curva bidimensional e as marcas referentes às transformações do objeto selecionado.

Trabalhos como o de Militão e Carvalho [61] transportam modelagens bidimensionais para o espaço tridimensional através da especificação de uma cota para cada posição, fazendo as tarefas de modelagem em um plano. Uma outra estratégia é a modelagem de curvas planares e o posicionamento interativo destas curvas em 3D para a especificação de seções transversais de sólidos [41]. Estas modelagens são muitas vezes chamadas de $2\frac{1}{2}$D. Modelos feitos por rotação de uma área planar em torno de um eixo, também podem ser considerados como modelagens $2\frac{1}{2}$D. Certamente existem limitações neste tipo de abordagem, como por exemplo a construção

de objetos que se interceptam em várias direções. Para este tipo de modelagem geral, algum tipo de interface 3D se faz necessária.

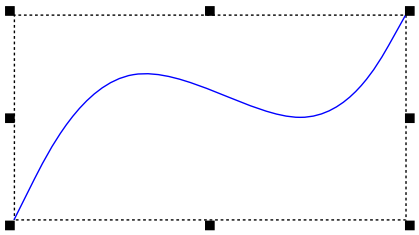


Figura 2.8: Marcas para transformações bidimensionais.

A modelagem gráfico-interativa clássica de curvas em 2D, com o *mouse* como ferramenta para a manipulação direta dos pontos de interpolação, necessita de alguns paradigmas adicionais para ser feita em 3D. O primeiro ponto é a especificação da profundidade de projeção, ou seja, sendo o *mouse* um dispositivo bidimensional de localização, como simular o movimento na cena em 3D? Alguns trabalhos [39, 87] fazem uso de cursores 3D controlados pelo movimento do *mouse*. Nestes modeladores, uma camada de *software* que considera a direção e os planos de projeções faz a tradução de 2D para 3D em um volume de visualização previamente especificado. O *feedback* usado é o desenho de três linhas ortogonais, normalmente paralelas aos eixos cartesianos.

Uma outra maneira de se especificar posições no espaço é usar um dispositivo físico de localização 3D, que se mostra bastante eficiente na modelagem de maquetes ou de modelos em escala. O dispositivo mais comum que é usado é a mesa digitalizadora 3D. Coloca-se o objeto que se deseja modelar no domínio, ajusta-se as transformações volumétricas e amostra-se pontos na superfície do sólido, podendo-se posteriormente aplicar algoritmos para determinação das superfícies sobre os pontos amostrados, como forma de modelagem para fins diversos.

Tanto os cursores quanto os dispositivos de localização 3D apresentam dificuldades para o projeto de curvas no espaço. Os dispositivos físicos 3D não são muito usados em modelagem de curvas pois, além de usualmente não estarem disponíveis, necessitam de um modelo físico como referência para a entrada de pontos. Os cursores apresentam imprecisão no movimento devido à dependência existente entre a posição do cursor na tela e os parâmetros de projeção usados. Novos artifícios são necessários para se criar facilidades na mudança dos parâmetros de projeção da cena, tais como uso do teclado, novos diálogos, ou atração para símbolos e entidades já modeladas. Nas bibliotecas atuais para desenvolvimento de interfaces gráficas 3D, como o Open Inventor [87], que oferecem estruturas de dados para modelagem de todos os objetos, inclusive aqueles relativos às metodologias de construção, nenhum método para a especificação interativa de pontos em 3D é sugerido.

Adotou-se o paradigma de manipulação direta das entidades, com a criação de um ambiente que simula o 3D. O autor e outros [24] apresentaram uma metodologia para a especificação de coordenadas em 3D com o uso do *mouse* com o auxílio de um plano de interface móvel. Sobre este plano, desenha-se um cursor com duas linhas ortogonais que se prolongam até os limites laterais, quando o usuário está fazendo a tarefa de especificação de posições em 3D. Um ponto importante no projeto da interface para a criação interativa de curvas espaciais por manipulação direta no MG é a implementação do controle dos pontos de interpolação na criação de qualquer tipo de curva. A Figura 2.9 mostra o aspecto do plano de interface durante a construção de uma curva do tipo B-spline cúbica, onde pode-se notar que, além dos pontos de interpolação, o usuário pode mover os pontos de controle de tangentes (mostrados com círculos brancos).

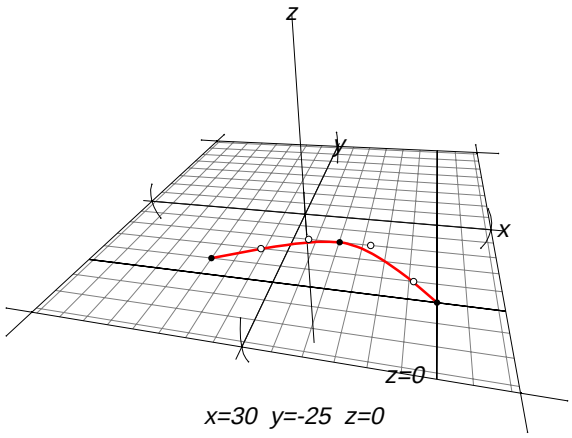


Figura 2.9: Plano móvel de interface adotado no MG.

Com o plano de interface, pode-se usar as metodologias de modelagem comumente usadas em 2D, fazendo-se a atração para a malha de pontos sobre a superfície do plano *(snap to grid)*, ou para posições já modeladas *(snap to objects)*, como ferramentas auxiliares à manipulação direta. Testes piloto feitos com a interface do MG indicam que o plano de interface foi usado com sucesso nas operações propostas [22]. A conclusão principal dos testes piloto indica que o *feedback* dos símbolos gráficos utilizados e das operações parciais de modelagem é decisivo para o sucesso de construções compostas. Os testes piloto indicam ainda que o plano de interface se mostra bastante importante na tarefa de auxílio ao usuário para as operações que podem ser feitas. Um bom exemplo é a transformação por espelhamento que é feita relativamente ao plano, onde testou-se a sua própria manipulação.

O uso do plano de interface resolve o problema de especificação da profundidade da projeção, transportando o problema de modelagem para uma superfície planar bem definida, inserida no contexto de modelagem.

2.1.4 Modelo de manipulação da projeção

Para se criar um ambiente produtivo para a construção interativa de curvas, usando-se como paradigma básico a manipulação direta, deve-se usar um controlador de mudança de projeções nos moldes do V3D [18], ou do *ArcBall* [80], também feitos por manipulação direta com o uso do *mouse*. O trabalho de Castier e outros [18], além de apresentar uma taxonomia para classificação dos tipos de manipulação da projeção, sugere o uso do padrão gráfico de desenho OpenGL [62], que faz as tarefas de *rendering* do modelo e dos símbolos auxiliares usados.

Entende-se ser natural o uso do *mouse* tanto para entrada de posições tridimensionais quanto para a especificação dos parâmetros de projeção por manipulação direta, pois qualquer interação feita de outra forma representaria uma quebra do processo de modelagem, prejudicando a tarefa global de construção. No MG, a biblioteca V3D é usada para definição dos parâmetros de projeção por manipulação direta. As funcionalidades de mudança de projeção são acionadas por um botão na interface que interrompe temporariamente a tarefa corrente, e passa a simular as movimentações de câmera, ponto de referência, etc.

2.1.5 Transformações geométricas tridimensionais

Para especificação das transformações geométricas em 3D usando-se o *mouse*, van Emmerik [84] apresenta uma metodologia com o uso de marcas posicionadas junto aos limites do objeto, semelhante ao já instalado padrão em 2D. Por manipulação direta, escolhendo-se a marca apropriada à operação desejada (escala, translação ou rotação), pode-se especificar transformações relativas a qualquer direção ou ponto de referência. A Figura 2.10 mostra o desenho das marcas relativas às transformações por rotação (marcas extremas) e translação (marca central).

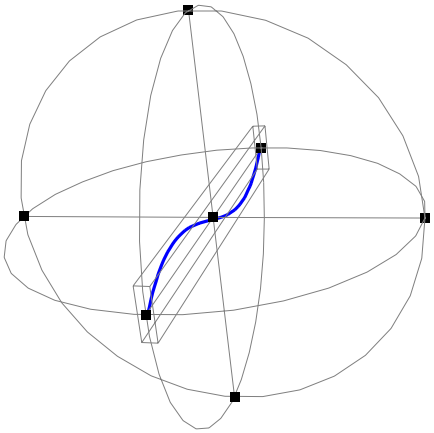


Figura 2.10: Marcas para especificação de transformações 3D em uma curva.

Existem dificuldades associadas aos parâmetros de projeção da cena para se especificar

transformações com o modelo proposto por van Emmerik. Para se especificar uma translação paralela ao eixo y, por exemplo, as vistas ideais são aquelas em que este eixo se projeta nas direções horizontal ou vertical, privilegiando a movimentação do *mouse* para esta transformação, pois a escolha do movimento é feita pela seleção da marca, no momento do pressionamento do botão do *mouse*, e pelo produto escalar mais significativo entre os vetores que representam os eixos de translação e o vetor definido por eventos consecutivos de movimentação do *mouse*.

Associando-se à geração de poligonais de controle que definem as curvas espaciais e à manipulação dos parâmetros de projeção, o modelo ideal de interface para a construção de cascas deve englobar também as funcionalidades de transformações geométricas, com o modelo de van Emmerik integrado em um modelo único consistente e que permite que se faça as tarefas de forma natural e acoplada.

As transformações 3D com o modelo de van Emmerik foram implementadas para todos as entidades gráficas presentes na estrutura de dados (vide Seção 2.4), e também para o plano de interface. Este modelo de transformações completa o ambiente de interface criado para o MG.

2.2 Modelagem de Superfícies

Esta seção apresenta a formulação de algumas superfícies que podem ser construídas a partir de curvas espaciais: bilineares, *sweeps*, Béziers, B-splines e retalhos de Coons. As superfícies do tipo *sweep* genérico apresentadas aqui foram desenvolvidas para atender a análises de tubulações marítimas que sofreram danos.

2.2.1 Superfícies bilineares

Talvez a mais simples de todas, a superfície bilinear é completamente determinada por um circuito de quatro pontos. Estes pontos são considerados como os cantos que definem o espaço paramétrico uv, com $0 \leq u, v \leq 1$. Desta forma, P_0 corresponde ao par $(u, v) = (0, 0)$, P_1 a $(0, 1)$, P_2 a $(1, 1)$, e P_3 a $(1, 0)$. Qualquer ponto no interior do retalho definido pelos quatro pontos é determinado por interpolação linear nos dois sentidos:

$$S(u, v) = P_0(1 - u)(1 - v) + P_1(1 - u)v + P_2u(1 - v) + P_3uv. \tag{2.6}$$

Se os quatro pontos usados como cantos forem coplanares, então o retalho também é plano. A Figura 2.11 mostra um parabolóide hiperbólico que é obtido com a interpolação dada pela superfície bilinear que passa pelos pontos $P_0 = (1, 0, 0)$, $P_1 = (0, 1, 0)$, $P_2 = (1, 1, 1)$, e $P_3 = (0, 0, 1)$.

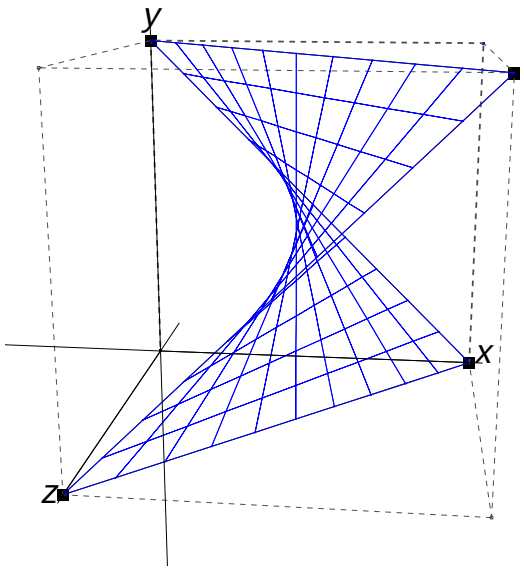


Figura 2.11: Mapeamento bilinear definindo parabolóide hiperbólico.

O MG trata o plano de interface como uma superfície bilinear, o que facilita o desenho dos símbolos auxiliares, como as *grid-lines* mostradas na Figura 2.9. Os quatro vértices que controlam a posição do plano de interface são mantidos equidistantes de uma posição central, que pode ser alterada pelo usuário por manipulação direta, segundo o modelo de van Emmerik (vide Seção 2.1.5). O espaço do objeto que se está modelando é definido pelo tamanho do plano de interface, ou seja, pelo espaçamento dos vértices que definem o plano. O MG permite que se construa qualquer superfície bilinear usando-se quatro curvas retas nos bordos, fazendo-se subdivisões compatíveis duas a duas.

2.2.2 Sweeps

As superfícies classificadas como *sweeps* de translação, chamadas de simplesmente de *Sweeps* no contexto do MG, são geradas pelo arrasto unidimensional de uma curva qualquer. Tais superfícies são chamadas de regradas. Mais precisamente, pode-se aplicar a seguinte técnica para se classificar uma superfície como sendo regrada: em torno do vetor normal à superfície em um ponto qualquer, rotaciona-se um plano limitado por um dos vetores que definem as derivadas direcionais no espaço paramétrico (d_u ou d_v); se em pelo menos uma posição deste plano a aresta limitada estiver sobre a superfície, diz-se que a superfície é regrada nesta direção. Desta forma, as superfícies bilineares são duplamente regradas.

Pode-se definir o mapeamento do espaço paramétrico (u, v) para o tridimensional (x, y, z)

em qualquer superfície da classe *Sweep* fazendo:

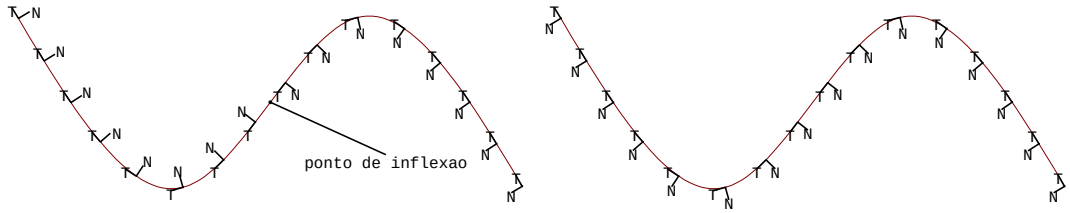
$$S(u, v) = P(u)T(v), \tag{2.7}$$

onde $P(u)$ é dado pela parametrização da curva geradora, e $T(v)$ é a transformação dada pelo *sweep* específico. Nos *Sweeps* a transformação $T(v)$ é dada por $T(v) = vD$, onde D é a direção de translação.

Uma outra categoria de *sweeps* são os de rotação, chamados de *Rsweeps* no contexto do MG, gerados pelo arrasto de uma curva em torno de um eixo (vetor) de rotação posicionado sobre um ponto definido no espaço 3D. Um ângulo determina o desenvolvimento total α_{tot}. Nos *Rsweeps*, a transformação é dada por $T(v) = M(v)$, e envolve o cálculo da matriz M que define a rotação genérica em torno do eixo passando pelo ponto de referência, e com o ângulo total α_{tot}, e $0 \leq v \leq 1$.

Nos *sweeps* genéricos, os retalhos são gerados por transformações compostas, como por exemplo translação e rotação simultâneas, aplicadas a uma curva qualquer. No MG, os *sweeps* genéricos são referenciados como *Gsweeps* e descritos por duas curvas: uma define as seções transversais em cada passo e a outra define a trajetória, ou seja, as transformações a serem aplicadas à curva de arrasto.

O triedro de Frenet-Serret [36] apresenta mudanças bruscas de orientação nos pontos de inflexão, como mostra a Figura 2.12a. Em curvas planares, este fato pode ser identificado pela mudança de direção do vetor *binormal* da curva, que aponta para fora do plano nas posições anteriores ao ponto de inflexão, e para dentro do plano nos pontos subseqüentes. Se a base definida pelo triedro de Frenet-Serret fosse usada para determinar as transformações a cada passo da matriz M na Equação 2.7, uma superfície com *saltos* seria gerada.



(a) Triedros de Frenet-Serret.

(b) Triedros usados no *Gsweep*.

Figura 2.12: Triedros definindo transformações a cada passo da curva trajetória.

Uma alternativa para este problema apresentada por Bloomenthal [11] consiste em se redefinir o triedro para que os vetores N e B, que representam a normal principal e o vetor binormal respectivamente, não mudem de direção nos pontos de inflexão, como mostra a Figura 2.12b. No

ponto inicial da curva, a base corresponde ao triedro de Frenet-Serret. Para os pontos seguintes, testa-se o sinal do produto escalar entre o vetor binormal calculado no ponto corrente e o calculado no ponto anterior; se este produto for negativo, inverte-se o vetor binormal referente ao ponto corrente. Superfícies geradas com esta metodologia têm sido usadas em simulações de gasodutos submersos avariados pela DIPREX/CENPES/Petrobras, pois as medidas necessárias à modelagem são tomadas na superfície externa do tubo (obtidas por mergulhadores ou robôs em grandes profundidades), de forma que pequenos desvios nas três dimensões originam pontos de inflexão na curva trajetória. A Figura 2.13 exemplifica uma modelagem de um tubo com eixo variável.

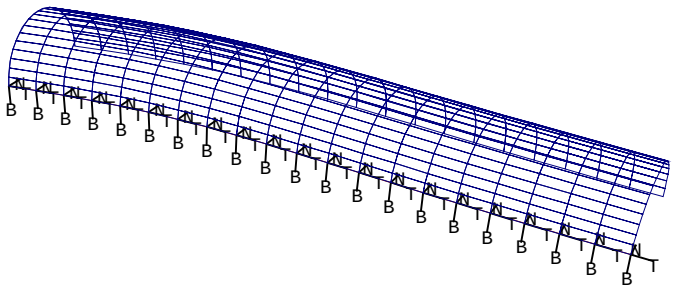


Figura 2.13: Tubo circular avariado apresentando eixo variável.

2.2.3 Superfícies de Bézier

As superfícies de Bézier são definidas por uma malha de pontos de controle que orientam as interpolações nos bordos e no interior. O algoritmo de de Casteljau, descrito na Seção 2.1.2, pode ser usado para se obter a representação de uma superfície de Bézier. Por repetidas interpolações bilineares (Equação 2.6), faz-se a construção variando-se os parâmetros u e v. Na forma matricial usando-se interpolações cúbicas pode-se expressar qualquer ponto na superfície partindo-se de uma malha de pontos $\{b_{i,j}\}_{i,j=0}^{3}$ e de posse dos parâmetros (u, v) (vide Farin [35]), por:

$$b_{r,r}^{i,j} = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} b_{i,j}^{r-1,r-1} & b_{i,j+1}^{r-1,r-1} \\ b_{i+1,j}^{r-1,r-1} & b_{i+1,j+1}^{r-1,r-1} \end{bmatrix} \begin{bmatrix} 1-v & v \end{bmatrix} \tag{2.8}$$

com $r \in \{1, 2, 3\}$, $i, j \in \{0.1\}$ e $b_{ij}^{00} = b_{ij}$. Na Figura 2.14 mostra-se a malha de pontos necessária para determinação de uma superfície bicúbica de Bézier.

O algoritmo que descreve a Bézier bicúbica com os pontos de controle mostrados na Figura 2.14 demanda que as curvas de bordo também sejam cúbicas de Bézier, o que nem sempre é viável em modelagens livres. Farin [35] apresenta uma formulação mais genérica para superfícies definidas pelo produto tensorial de curvas de diferentes graus nas direções u e v. A representação

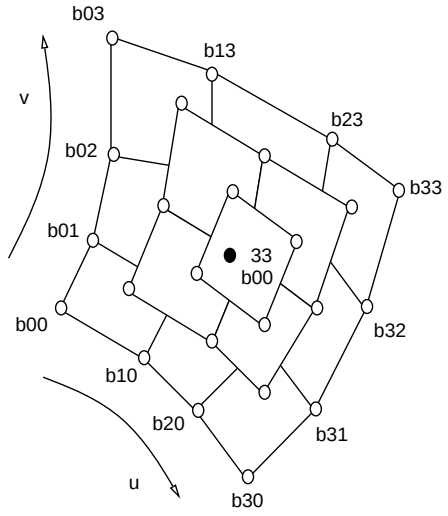


Figura 2.14: Malha de pontos de controle superfície de Bézier bicúbica [35].

por superfícies de Bézier não foi incorporada ao MG, pois optou-se pelas superfícies definidas por curvas de bordo.

2.2.4 Superfícies B-splines

Os retalhos do tipo B-splines são definidos de forma semelhante às Béziers pelo produto tensorial das curvas de bordo, associado à escolha dos *knot-vectors*. Supondo-se retalhos bicúbicos e *triple-end knot-vectors*, têm-se a seguinte formulação:

$$S(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} d_{ij} N_i^3(u) N_j^3(v), \qquad \text{onde:} \tag{2.9}$$

$$N_i^0(u) = \begin{cases} 1 & \text{se } u_{i-1} \leq u < u_i, \\ 0 & \text{para os outros casos} \end{cases} \qquad \text{e} \tag{2.10}$$

$$N_l^n(u) = \frac{u - U_{l-1}}{u_{l+n-1} - u_{l-1}} N_l^{n-1}(u) + \frac{u_{l+n} - u}{u_{l+n} - u_l} N_{l+1}^{n-1}(u) \qquad . \tag{2.11}$$

Barsky e Greenberg [5] apresentam uma metodologia para determinação dos pontos de controle de uma superfície B-spline partindo-se dos pontos de interpolação. Este tipo de abordagem permite que se construa representações contínuas para superfícies em que se conhece pontos onde se identifica as coordenadas paramétricas e tridimensionais. O que ocorre em muitas modelagens, entretanto, é que o usuário conhece apenas as coordenadas de algumas seções transversais ou pontos notáveis, não dispondo de informações sobre uma malha de pontos na superfície. Além disso, a edição tridimensional da malha de pontos de controle de uma superfície B-spline bilinear não é uma tarefa fácil, do ponto de vista de interface. Por estas razões, no MG adotou-se o uso

das superfícies cuja representação é completamente formulada pelas curvas espaciais: os retalhos de Coons mostrados na Seção 2.2.5, e os *sweeps* mostrados na Seção 2.2.2.

2.2.5 Superfícies de Coons

Diferentemente das superfícies de Bézier [10] e B-spline[1], as superfícies de Coons [27], ao invés de serem definidas por uma malha de pontos de controle, usam apenas as curvas de bordo para gerar os pontos no domínio.

Para se definir a formulação da superfície bilinear de Coons, é necessária a apresentação das *lofted surfaces*, que são geradas por duas curvas de fronteira, mapeando o domínio da forma mais simples possível, a linear, ou seja:

$$S(u,v) = (1-v)c_1(u) + vc_2(u) \ . \tag{2.12}$$

Desta forma, todas as linhas com $u = constante$ são linhas retas ligando a curva c_1 à curva c_2. Quando o domínio de interpolação é retangular, com quatro curvas nos bordos, pode-se usar duas interpolações lineares nas duas direções:

$$S_u(u,v) = (1-v)c_1(u) + vc_2(u) \tag{2.13}$$

$$S_v(u,v) = (1-u)c_3(v) + uc_4(v) \ , \tag{2.14}$$

que, se subtraídas da Equação 2.6, fornecem a equação de Coons para superfícies bilineares [74]:

$$\begin{aligned} S(u,v) =& (1-v)c_1(u) + vc_2(u) + (1-u)c_3(v) + uc_4(v) - \\ & [P_0(1-u)(1-v) + P_1(1-u)v + P_2u(1-v) + P_3uv] \ , \end{aligned} \tag{2.15}$$

onde c_1, c_2, c_3 e c_4 são as curvas de bordo e P_0, P_1, P_2 e P_3 são as posições dos quatro cantos podendo ser calculadas por $c_1(u = 0)$ ou $c_4(v = 0)$, $c_1(u = 1)$ ou $c_3(v = 0)$, $c_2(u = 1)$ ou $c_3(v = 1)$, e $c_3(u = 0)$ ou $c_4(v = 1)$, respectivamente.

De maneira análoga, pode-se definir a superfície de Coons trilinear em um retalho formado por três curvas quaisquer, usando-se três valores paramétricos, um para cada direção ou curva, da seguinte forma [74]:

$$\begin{aligned} S(u,v,w) =& \frac{1}{2}[\frac{u}{1-v}c_1(v) + \frac{w}{1-v}c_2(1-v) + \frac{v}{1-w}c_2(w) + \frac{u}{1-w}c_3(1-w) + \\ & \frac{w}{1-u}c_3(u) + \frac{v}{1-u}c_1(1-u) - wc_3(0) - uc_1(0) - vc_2(0)] \end{aligned} \tag{2.16}$$

A Figura 2.15 mostra uma superfície trilinear de Coons gerada com três cúbicas de Bézier nos bordos.

[1]Uma boa referência sobre o uso industrial de curvas e superfícies nos anos 60 é [68]

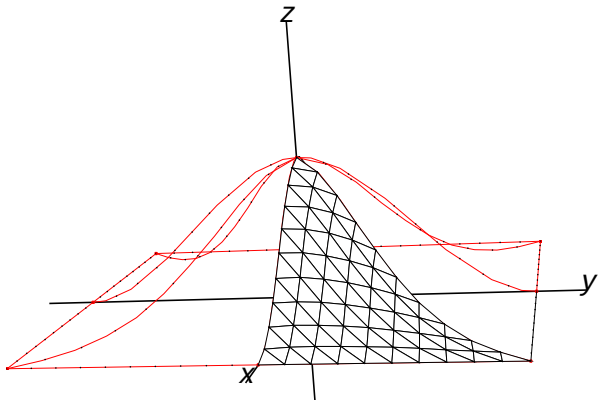
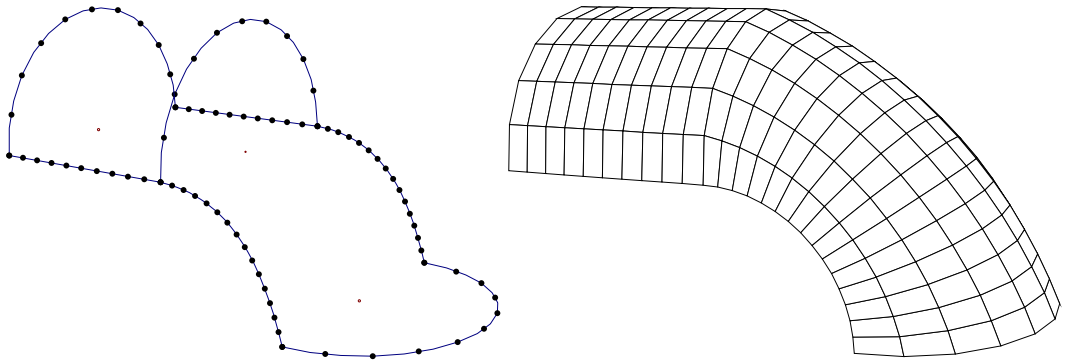


Figura 2.15: Retalho de Coons trilinear.

Se existe a necessidade de controle das curvaturas nos bordos, o que ocorre quando dois retalhos justapostos devem possuir continuidade em algum grau, deve-se ampliar a definição da formulação linear para o grau desejado. Se as duas curvas de bordo que definem o retalho em uma direção forem cúbicas, por exemplo, pode-se definir um mapeamento cúbico e com isto fazer o controle das tangentes nos bordos. A Figura 2.16b mostra dois retalhos bilineares justapostos gerados com as curvas de bordo mostradas na Figura 2.16a que poderiam possuir derivada contínua ao longo da curva de conexão, se um mapeamento bicúbico fosse utilizado. Deve ser notado que, com o uso de superfícies de Coons bilineares, as seções transversais intermediárias não são arcos de círculo como as curvas de bordo.

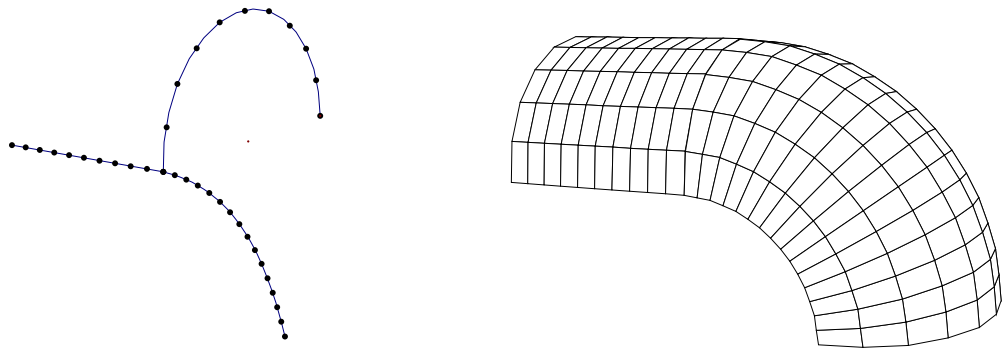


(a) Curvas de bordo. (b) Malhas.

Figura 2.16: Retalhos bilineares justapostos.

Uma alternativa para se conseguir uma modelagem com superfícies contínuas com as curvas da Figura 2.16a, consiste em usar um *Gsweep* com a metodologia descrita na Seção 2.2.2. A Figura 2.17a mostra que apenas três curvas são necessárias para a especificação dos dois *Gsweeps*

que formam as duas cascas contínuas, cujas malhas são apresentadas na Figura 2.17b. Observa-se que todas as seções transversais obtidas perpendicularmente às duas curvas geradoras correspondem ao arco que círculo que as descreve.



(a) Curvas geradoras. (b) Malha.

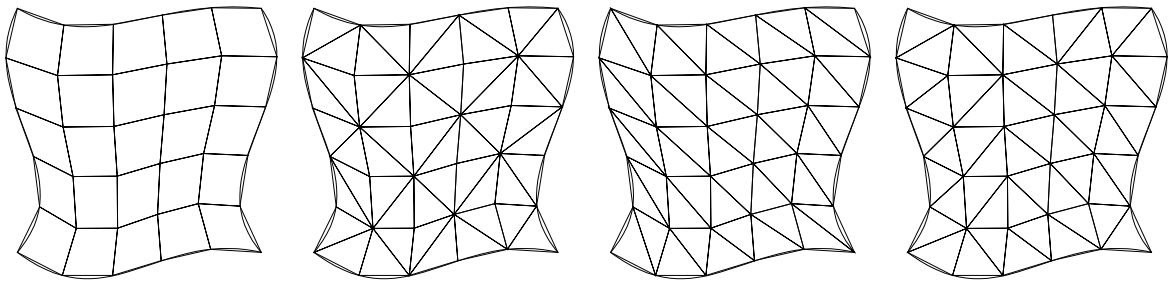
Figura 2.17: Retalhos contínuos feitos com *Gsweeps*.

2.2.6 Mapeamentos sobre superfícies

O trabalho de Haber e Abel [44] é uma referência básica para as pesquisas em mapeamento e discretização de superfícies, aplicadas à geração de malhas para elementos finitos. Haber e Abel foram os pioneiros a definir a separação entre os diversos níveis hierárquicos de representação que um modelo complexo de elementos finitos deve apresentar para que uma malha única seja consistentemente gerada. Estes níveis hierárquicos mostravam a separação que deve existir entre os diversos retalhos e também entre a descrição matemática da superfície e da malha internas a cada um dos retalhos. Haber e Abel [45] descrevem ainda como as idéias planares de uso dos mapeamentos transfinitos como projetores podem ser transportadas para três dimensões. Os projetores lineares, bilineares e trilineares correspondem, mais precisamente, à formulação das superfícies de Coons na forma discreta. Observa-se que qualquer formulação que seja usada para representar superfícies determinadas pelo produto tensorial das curvas de bordo descreve a mesma superfície, desde que a ordem de interpolação seja a mesma.

Para o caso bidimensional, as tecnologias de definição de superfícies discretas podem ser usadas como regra de geração de pontos e elementos no domínio da região a ser mapeada. Os pontos gerados pertencem ao plano, única e exclusivamente porque todos os pontos das curvas de bordo que os mapeiam também estão contidos no plano. Para exemplificar, a Figura 2.18 mostra as diversas técnicas de se mapear uma superfície planar usando um projetor bilinear para gerar os mesmos pontos, porém com diferentes definições para a topologia dos elementos. Estas técnicas

de especificação da topologia das malhas são aplicáveis em três dimensões e foram adotadas na construção das superfícies no MG.



(a) Quadrilateral. (b) *Union jack*. (c) Δ à esquerda. (d) Δ de Delaunay.

Figura 2.18: Técnicas de mapeamento planares.

A especificação dos mapeamentos para a geração de malhas de elementos finitos no domínio de superfícies que possuem descrição geométrica previamente definida consiste apenas da determinação da topologia dos elementos (comumente quadrilaterais ou triangulares), e da lista de vértices relativa ao espaço paramétrico. A Figura 2.19 mostra a influência da discretização com duas semi-esferas de raio idêntico definidas pela rotação de um arco de círculo em torno do eixo x. A diferença evidente que se observa pelo número de faces corresponde à subdivisão grosseira do arco gerador da superfície superior, e ao passo angular bastante maior.

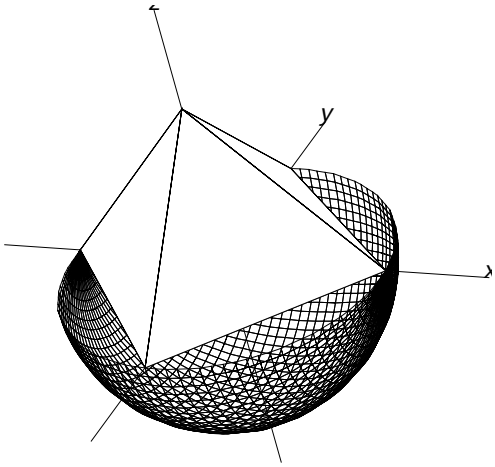


Figura 2.19: Duas semi-esferas de raio idêntico e mapeamentos distintos.

2.2.7 Construção de superfícies

A interface para construção das superfícies que se baseiam em curvas é bastante simples, se comparada com a modelagem destas curvas. Na maioria dos casos, como em superfícies de Coons [27], a especificação do circuito de curvas que define o bordo do retalho é a tarefa mais difícil do ponto de vista da interface, ficando as informações geométricas e topológicas, definidas pelo tipo de subdivisão que as curvas apresentam, implicitamente contidas nas próprias curvas ou em parâmetros adicionais informados textualmente.

As superfícies do tipo *Gsweep* possuem interface bastante semelhante, bastando que o usuário construa uma curva de arrasto e outra que defina a trajetória. Para os outros *sweeps*, escolhe-se uma curva e informa-se os parâmetros adicionais textualmente.

2.3 Interseção de superfícies

A interseção de duas superfícies paramétricas é um problema de difícil solução [70] e de grande importância em modelagem. A solução analítica é muitas vezes inviável e pouco prática, uma vez que retalhos simples podem produzir interseções difíceis de se representar [49]. Os trabalhos atuais, quase que na totalidade, investigam soluções numéricas aproximadas.

Os métodos para solução do problema de interseção em *CAGD* pertencem a duas categorias principais[2]: métodos de *marcha* e métodos de *subdivisão*. Os métodos de *marcha* ou *continuação* determinam as curvas de interseção no espaço tridimensional do objeto pela marcha na direção do seu vetor tangente [2–4, 83]. Os métodos de *subdivisão* ou *decomposição* determinam as curvas de *trimming* no espaço paramétrico bidimensional das superfícies, por subdivisão recursiva a cada passo [48].

2.3.1 Métodos de marcha

Os métodos de marcha possuem três passos básicos:

1. a obtenção de pontos iniciais de interseção,
2. a marcha na direção do vetor tangente, e
3. a ordenação do conjunto disjunto de interseções.

A obtenção de pontos iniciais de interseção pode ser feita com testes entre faces nas duas superfícies obtidas por enumeração uniforme exaustiva nos espaços paramétricos, ou por *quadtrees* usando-se os *bounding boxes* dos retalhos. Qualquer ponto determinado inicialmente

[2] [70] e [47] fazem classificações mais amplas.

em 3D deve ser movido gradualmente (*relaxed*) [3] na direção do ponto de interseção real dos retalhos usando-se os dois domínios paramétricos, de forma a aproximar uma precisão escolhida. Este passo é muitas vezes problemático quando resolvido com o algoritmo de Newton-Raphson linearizado [4], e ainda envolve as tolerâncias necessárias à identificação de posições coincidentes no cálculo das distâncias euclidianas entre os pontos.

No segundo passo do algoritmo, a determinação da direção de caminhamento envolve previsões da direção do vetor tangente [2, 21] para se caminhar em uma mesma curva de interseção. Neste passo, faz-se também a detecção de pontos nas fronteiras e de pontos conflitantes, já calculados em passos anteriores, e a identificação de pontos de bifurcação. Todas estas etapas apresentam dificuldades em potencial para uma implementação robusta de um método de marcha. Barnhill e Kersey [3] abordam cuidadosamente cada um destes aspectos e propõem soluções eficientes.

Barnhill e Kersey [3] mostram ainda como uma estrutura de dados baseada em *quadtrees* [76] pode ser usada para fazer a ordenação do conjunto disjunto de pontos obtidos nos passos anteriores.

2.3.2 Métodos de subdivisão

Nos métodos de subdivisão por refinamento progressivo, subdivide-se os dois domínios paramétricos dos retalhos em retângulos, que inicialmente correspondem ao intervalo total de desenvolvimento da superfície. Em seguida, calcula-se os dois *bounding boxes* tridimensionais destes intervalos e testa-se a interseção entre eles; não havendo interseção, interrompe-se a subdivisão. Se houver interseção, divide-se os dois retângulos em áreas menores, e repete-se o procedimento até que o tamanho dos *bounding boxes* seja tão pequeno quanto o valor de tolerância especificado.

Para exemplificar a subdivisão recursiva obtida com o refinamento progressivo dos espaços paramétricos, aborda-se o cálculo das duas curvas de interseção definidas pelas interseções entre a casca toroidal e o plano mostrados na Figura 2.20.

A Figura 2.21 mostra como evoluem as subdivisões no plano, mostradas na parte inferior, e na superfície toroidal, mostradas na parte superior, nas buscas pela identificação das curvas de *trimming*. As curvas de *trimming* finais em cada espaço paramétrico são mostradas na parte direita da Figura 2.21.

Os método de subdivisão recursiva dos espaços paramétricos fornecem em cada passo uma interseção potencial válida e que possui uma precisão bem definida (mesmo que grosseira em passos iniciais), além de não conduzir a muitos dos problemas citados nos métodos de marcha, como o relaxamento dos pontos para as duas superfícies, ou a subdivisão do espaço paramétrico

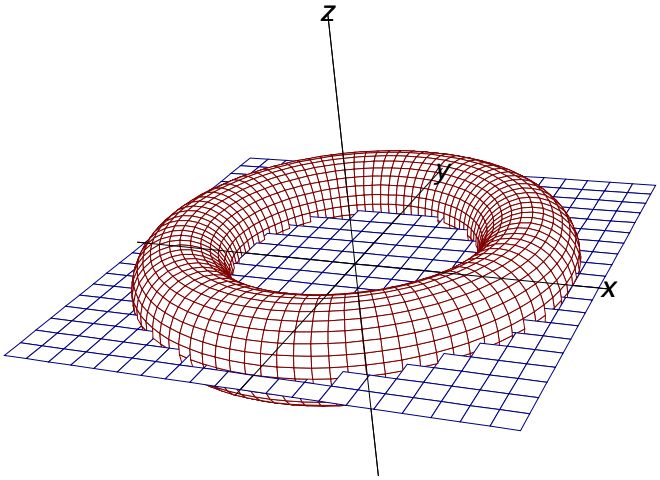


Figura 2.20: Interseção visual entre plano e superfície toroidal.

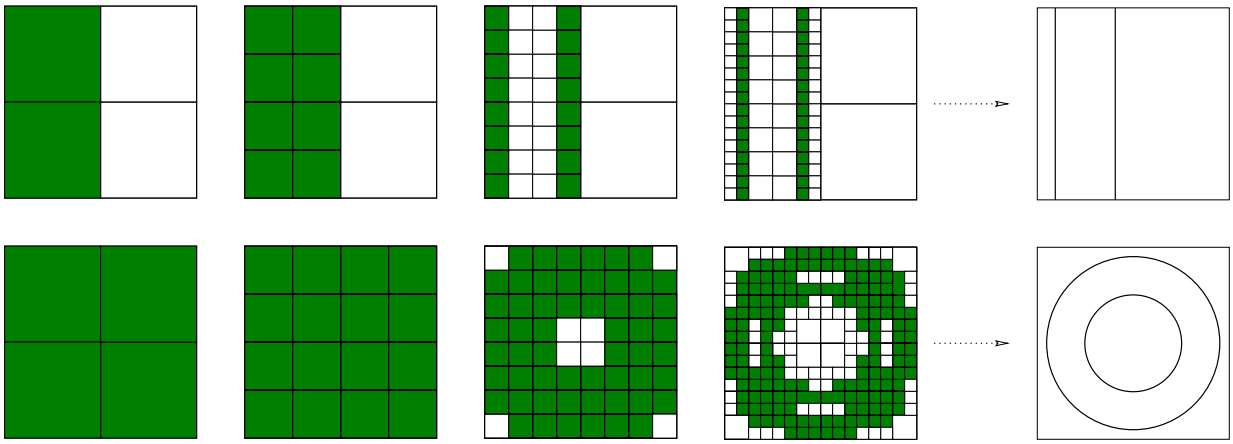


Figura 2.21: Evolução das subdivisões nos espaços paramétricos do plano (parte inferior) e da superfície toroidal (parte superior).

para o cálculo das interseções iniciais. Por outro lado, a definição das curvas de interseção não é obtida como resultado imediato das subdivisões paramétricas.

Gleicher e Kass [42] descrevem um algoritmo para interseção de superfícies baseado em aritmética intervalar [72], que usa a técnica de refinamento progressivo. Na aritmética intervalar aplicada à interseção de superfícies paramétricas, escreve-se avaliadores que recebem como entrada um retângulo no espaço paramétrico representado pelos limites u_{min}, v_{min} e u_{max}, v_{max}, e como saída fornecem um *bounding box* para o retalho correspondente em 3D.

A estrutura de dados apresentada por Gleicher e Kass é interessante pois facilita a compreensão do algoritmo. Esta estrutura de dados contém dois elementos topológicos básicos: os *nós* e as *listas*. Os *nós* são representações de intervalos no espaço paramétrico, e as *listas* contêm referências para os *nós* da outra superfície, que potencialmente interceptam o nó que a contém.

Figueiredo [32] apresenta resultados mais eficientes que os obtidos por Gleicher com o uso da aritmética afim [25], que muitas vezes fornece *bounding boxes* menores, reduzindo o espaço de amostragem e conseqüentemente os testes de interseção.

A determinação das curvas de *trimming* nos algoritmos de subdivisão pode ser feita por extração de informações presentes na estrutura de *nós* e *listas* apresentadas por Gleicher e Kass. Ao final do algoritmo, tendo sido alcançada uma precisão desejada, cada *nó* (ou seu centro) representa um ponto de interseção que pode ser conectado.

Os trabalhos discutidos nesta seção não resolvem o problema de interseção de superfícies do ponto de vista da modelagem de elementos finitos proposta, que considera as malhas inicialmente existentes nos retalhos e que precisa que os elementos redefinidos pelos cálculos de interseção apresentem boa qualidade geométrica. O Capítulo 3 apresenta este problema com detalhe, descrevendo a estrutura de dados e um novo algoritmo para reconstrução das malhas.

2.4 Estruturas de dados

Martha [59] mostra que a estrutura de dados *Arestas Radiais* [86] (*Radial Edge* ou simplesmente RED) pode ser usada em simulações numéricas por elementos finitos para representação de estruturas formadas por superfícies compostas na modelagem de sólidos. Além dos aspectos relativos à modelagem de sólidos formados por retalhos de superfície, Martha investiga a evolução de fraturas no modelo, com a redefinição da malha de elementos finitos e contorno definida sobre estas superfícies. A modelagem, no entanto, não é feita com interseções paramétricas. O processo é semi-automático, onde o usuário define linhas retas no espaço paramétrico que são usadas para subdividir retalhos ou para construir novas curvas representativas de fraturas.

Apesar de o objetivo desta tese não ser modelagem de sólidos, mas sim modelagem de cascas, a estrutura de dados RED poderia ser uma alternativa para se armazenar consistentemente as curvas e superfícies usadas na modelagem. Em termos de entidades topológicas básicas da estrutura arestas radiais, os *vértices*, as *arestas*, e as *faces* poderiam ser relacionadas diretamente com a modelagem de cascas abordada nesta tese. Os vértices estariam associados aos pontos inicial e final das curvas, sendo comuns às curvas adjacentes. As curvas abertas poderiam corresponder às arestas, e as faces aos retalhos limitados pelas curvas de bordo. Quando se trabalha com superfícies de Coons, as curvas de bordo já existem na estrutura de dados antes da criação dos retalhos. Na construção dos *sweeps*, devem ser consideradas curvas nas fronteiras dos retalhos, eventualmente criando uma curva de bordo, para que o modelo topológico permaneça consistente. A entidade sólido estaria associada à definição de regiões fechadas e consistiria de grupos de retalhos que definem uma casca simples. Entretanto, a estrutura RED requer muita memória

para a construção de um modelo completo e visa representar consistentemente uma subdivisão espacial, de forma que uma alternativa mais simples foi adotada nesta tese.

A inserção de retalhos planares para modelagem automática de subdivisões espaciais [19] mostra que o tempo médio estimado para estas construções é proporcional ao quadrado do número final de retalhos a serem inseridos. Este aspecto não seria impeditivo para o uso desta estrutura em modelagens interativas, uma vez que o número total de superfícies que formam um modelo complexo (vide Figura 2.1) não é grande. Os critérios geométricos necessários para a inserção de retalhos não planares teriam que ser, entretanto, avaliados com maior rigor, pois existem os problemas de auto-interseções, tratamento de retalhos cíclicos, etc.

Um outro aspecto é a investigação da necessidade de relacionamento espacial entre os elementos que formam as malhas sobre os retalhos e as outras entidades topológicas presentes na estrutura. Uma lista simples de vértices e faces seria suficiente se não fosse permitida nenhuma operação que alterasse a subdivisão planar, em que consistem os mapeamentos sobre as superfícies. Entretanto, a proposta de modelagem deste trabalho engloba as interseções e o recorte de partes excedentes dos retalhos, além da reconstrução das malhas, de forma que deve existir algum mecanismo de encadeamento entre malhas distintas, mesmo que seja temporário.

Com o objetivo de representar de forma consistente e concisa a modelagem de cascas compostas, e considerando-se a importância das funcionalidades das curvas neste tipo de modelagem, adotou-se no MG uma estrutura de dados mais simples do que a RED, onde as entidades topológicas estão diretamente relacionadas com os tipos geométricos usados. A Figura 2.22 mostra o diagrama modular de entidades desenvolvido [23].

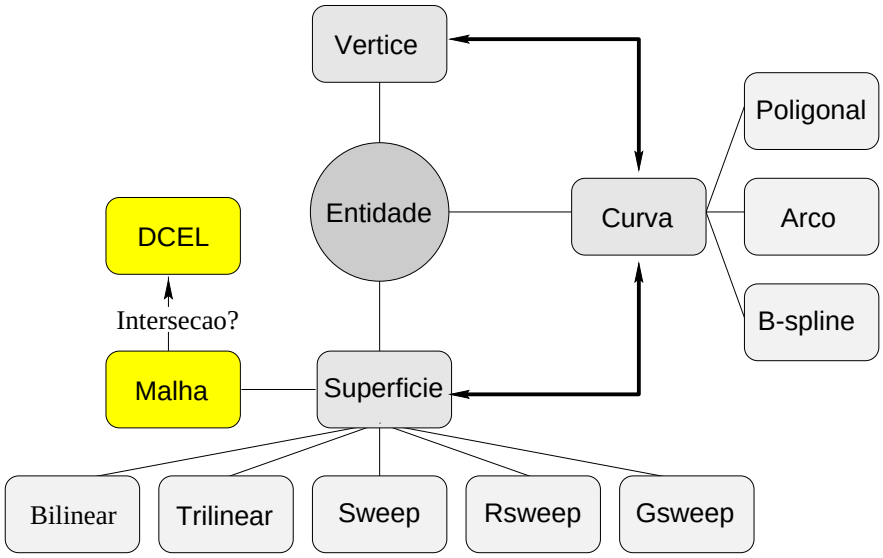


Figura 2.22: Módulos da estrutura de dados adotada.

As entidades que compõem o modelo de representação são os Vértices, as Curvas, e as Superfícies, cada uma com a sua Malha interna. Cada curva possui pontos e vetores de definição geométrica e apenas dois vértices topológicos nas duas fronteiras. Nas curvas fechadas, duas referências para o mesmo vértice são armazenadas. Como explicado na Seção 2.1.2, foram implementados três tipos de curvas: a B-spline cúbica interpolante, os Arcos de círculo, e as Poligonais formadas por segmentos de reta.

As informações de adjacências entre as entidades são mostradas na Figura 2.22 pelas linhas mais espessas que contém setas nos dois extremos. Cada vértice possui uma lista de *usos* que contêm referências para todas as curvas adjacentes. De maneira similar, cada curva possui uma lista de *usos* com referências para todas as superfícies adjacentes, além das referências para os dois vértices nas fronteiras. Para exemplificar a lista de *usos* de superfícies das curvas, a Figura 2.23 mostra uma curva com seis superfícies adjacentes ($S0$ a $S5$). As superfícies possuem referências apenas para as curvas de bordo.

A organização topológica desta estrutura possui alguma semelhança com a RED, não sendo entretanto proposta deste trabalho representar subdivisões espaciais com a identificação de regiões fechadas com a estrutura utilizada. A ordenação polar das superfícies sugerida pela Figura 2.23 (existente na RED) não é mantida na estrutura de dados do MG, podendo ocorrer ou não. Não existe também o formalismo das atualizações com os operadores topológicos usados por Weiler [86].

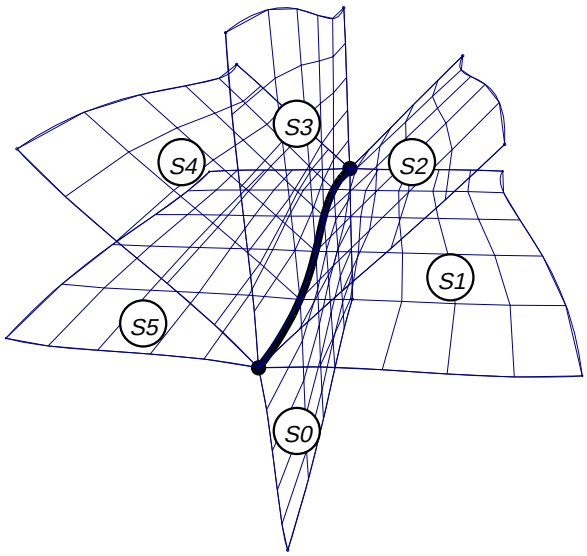


Figura 2.23: Lista de *usos* de superfícies presentes nas curvas.

As superfícies são definidas pela geometria das curvas geradoras e pela representação paramétrica usada. As classes Bilinear e Trilinear correspondem às superfícies de Coons com

quatro e três lados. As outras três classes: Sweep, Rsweep e Gsweep correspondem às superfícies por arrasto definidas na Seção 2.2.2.

As malhas sobre as superfícies que não apresentam interseções são representadas por listas simples de vértices e faces, para que se faça o desenho e seleção destas entidades. Estas malhas não apresentam qualquer tipo de conexão com as malhas das superfícies adjacentes. Esta conexão é feita quando se unifica o modelo, fazendo-se a construção de uma lista única de faces e vértices capaz de ser exportada para uma simulação por elementos finitos.

Para as superfícies que se interceptam, são construídas estruturas de dados planares usando-se o espaço paramétrico de cada uma para se ordenar as entidades topológicas (vértices, arestas e faces). O Capítulo 3 mostra como a estrutura de dados DCEL [71] foi adaptada para se resolver o problema de retalhos que se interceptam, fazendo-se a reconstrução automática das malhas envolvidas. Como mostra o Capítulo 3, nestas superfícies a estrutura de dados necessária à solução do problema de interseção é bastante maior do que a estrutura existente nas malhas primárias.

Capítulo 3

Interseção Paramétrica de Malhas

As soluções para o problema de interseção paramétrica de superfícies apresentadas na Seção 2.3 procuram resolver o problema do ponto de vista geométrico para a determinação precisa das curvas de interseção. Nos métodos de marcha, a determinação do ponto de partida das curvas de interseção pode ser feita com uma subdivisão do espaço paramétrico que acaba por definir uma *malha* de elementos em cada retalho. Esta malha, entretanto, não é explorada nos passos subseqüentes, de forma que os resultados dos cálculos feitos com os algoritmos baseados no método de marcha consistem de uma lista de curvas de interseção.

Os métodos de subdivisão apresentam como resultado dos cálculos uma subdivisão adaptativa dos dois espaços paramétricos dos retalhos envolvidos. Para que as técnicas contidas nos algoritmos de marcha ou subdivisão sejam usadas no contexto de modelagem apresentado no Capítulo 2, é necessário que se faça a completa definição de todas as curvas de interseção existentes entre os retalhos e, em seguida, se faça a definição das malhas de elementos finitos a serem usadas na simulação numérica, respeitando as curvas de interseção calculadas. Se não existir nenhum relacionamento entre as malhas e as curvas de interseção calculadas, surgem os problemas associados com o corte (*trimming*) das partes excedentes, que exige a identificação das regiões paramétricas distintas.

Uma outra dificuldade para a reconstrução das malhas com as técnicas de interseção apresentadas no Capítulo 2 está relacionada com os domínios de geração dos mapeamentos. As curvas de interseção devem funcionar como restrições para a geração dos novos elementos nas superfícies paramétricas, de forma que pode ficar difícil a especificação de domínios elementares de três ou quatro lados para se aplicar as técnicas tradicionais de mapeamentos (vide Seção 2.2.6). Na maioria dos casos, a única alternativa é a triangulação de Delaunay com restrições aplicada a todo o domínio dos retalhos, fazendo-se a inserção de pontos internos regularmente espaçados (vide Figura 1.3a).

Além destas dificuldades, a regularização dos tamanhos dos elementos gerados individualmente em cada superfície, e as questões relativas à qualidade geométrica destes elementos teriam que ser avaliadas por alguma metodologia que trate simultaneamente as duas superfícies.

Uma outra linha de trabalhos aborda o problema do ponto de vista de reconstrução das malhas existentes antes de se calcular as interseções. Souza e Gattass [81] apresentam uma metodologia baseada em grafos para construir malhas em superfícies paramétricas, estendendo o trabalho pioneiro de mapeamentos transfinitos usando curvas de contorno de Haber e Abel [44]. Os trabalhos de Sheng e Hirsch [78] e de Lau e Lo [51] seguem a linha de reconstrução completa das faces sobre as superfícies, considerando-se as curvas de *trimming*. A diferença central entre estes dois trabalhos está na proposta de triangulação, que no trabalho de Sheng e Hirsch é feita no espaço paramétrico e que no trabalho de Lau e Lo é feita pela técnica de *advancing front* [54] diretamente em 3D. Entretanto, estes trabalhos não dão ênfase ao problema de interseção de superfícies, que é um estágio de pré-processamento na construção de tais malhas.

No enfoque de elementos finitos utilizado neste trabalho, entende-se que o problema de interseção de superfícies deve ser abordado como um problema de *reconstrução de malhas*, de forma que não somente as representações paramétricas dos retalhos são consideradas, mas também as malhas sobre eles. Em outras palavras, as curvas de interseção devem pertencer a ambas as superfícies e a ambas as malhas após terem sido calculadas as interseções. Além disso, as discretizações definidas pelo usuário nas curvas de bordo devem ser respeitadas, de forma a acomodar corretamente malhas adjacentes a uma mesma curva. Desta forma, o objetivo global é construir uma malha única pela união das duas superfícies, e esta malha deve ser formada em sua maioria pelos elementos das malhas iniciais, que são modificadas apenas localmente, nas regiões de interseção. A Figura 3.1 exemplifica a reconstrução proposta.

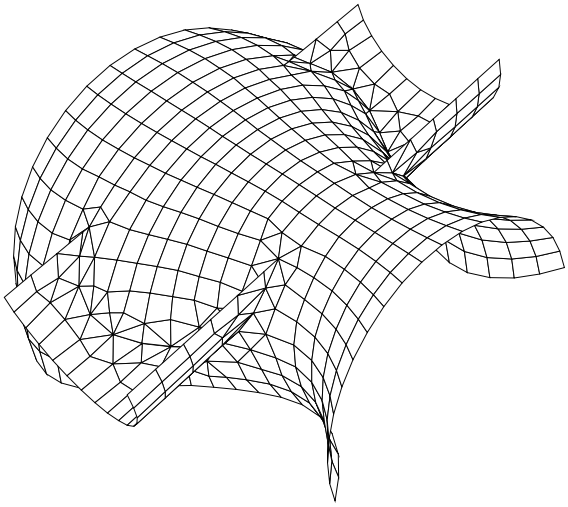


Figura 3.1: Malha composta após interseção.

As soluções anteriores para o problema de interseção de superfícies funcionam bem em muitos casos, mas não tratam consistentemente do problema de interseção de malhas como foi definido no parágrafo anterior. Uma exceção é o trabalho de Lo [55], que motivou o desenvolvimento do algoritmo apresentado neste trabalho. Lo desenvolveu um algoritmo simples para interseção de malhas formadas por faces triangulares que se adaptam às curvas de interseção. A solução de Lo não usa a descrição paramétrica contínua para as superfícies, de forma que pode ser facilmente implementada na maioria dos sistemas de modelagem. Por outro lado, nas proximidades de regiões de curvatura acentuada, os pontos de interseção calculados podem não pertencer às superfícies originais, o que é inaceitável para muitos problemas de modelagem de cascas.

Uma outra dificuldade em potencial do algoritmo de Lo é a conexão dos segmentos de reta individualmente calculados, que formam as linhas poligonais de interseção. Estes segmentos são calculados testando-se os pares de triângulos de cada superfície como mostra a Figura 3.2. Uma vez que o espaçamento dos pontos de interseção pode ser muito desigual em muitos casos, a conexão dos segmentos individuais depende de tolerâncias difíceis de se determinar,

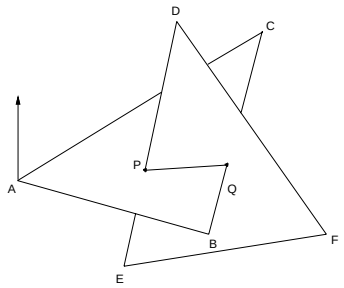


Figura 3.2: Segmentos individuais do algoritmo de Lo [55].

O algoritmo apresentado neste capítulo possui etapas similares às do algoritmo de Lo, e procura apresentar soluções eficientes para os problemas geométricos potenciais citados acima, usando em muitos casos as informações presentes nas topologias das malhas ao invés de se fazer testes de proximidade. Na seção seguinte apresentam-se as características necessárias para os algoritmos que pretendam resolver o problema de interseção de malhas proposto.

3.1 Caracterização do problema de interseção

Considera-se que a solução ideal para o problema de interseção de malhas paramétricas proposto deve ter as seguintes características:

- **Vértices nas superfícies originais:** Os vértices da malha resultante devem pertencer às respectivas superfícies paramétricas originais. Em particular:

– As curvas de interseção calculadas devem estar contidas em ambas as superfícies. Se as curvas de interseção pertencem apenas às faces das malhas mas não às superfícies, então a simulação por elementos finitos pode conduzir a resultados falsos ou inaceitáveis [9]. A forma mais simples de garantir a correção é calcular as curvas de *trimming* no espaço paramétrico e então mapeá-las para o espaço do objeto.

– A geometria da superfície resultante deve refletir com fidelidade a geometria das superfícies originais, pois estas representam a intenção do projetista. Em particular, novos retalhos paramétricos não devem ser definidos por produto tensorial das curvas de interseção, pois isto pode conduzir a geometrias bem diferentes. A Figura 3.3 mostra que as superfícies bilineares geradas com as curvas de interseção laterais ao furo não reproduzem a superfície cilíndrica original, gerando uma mossa.

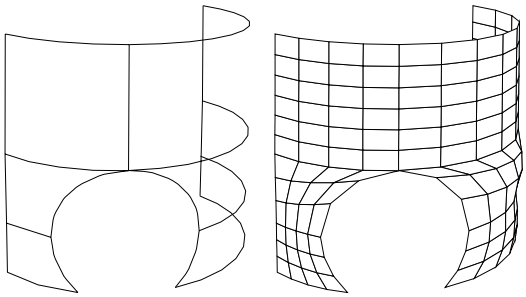


Figura 3.3: Mossa produzida pela re-parametrização.

– Pontos nos espaços paramétricos correspondendo ao mesmo ponto de interseção, quando mapeados em 3D, devem estar a uma distância menor do que uma tolerância pré-determinada. Usualmente, uma fração do tamanho do menor elemento interceptado nas duas malhas originais conduz a bons resultados.

- **Qualidade da malha:** Os elementos gerados durante a reconstrução devem apresentar boa qualidade geométrica na malha final, além de possuir tamanho médio semelhante aos das malhas originais. Se as malhas originais possuem elementos com tamanhos muito desiguais ou se existirem arestas muito próximas de uma curva de interseção, então a malha resultante pode apresentar elementos alongados com diferenças angulares acentuadas, se nenhuma correção for feita.

- **Modificações locais nas malhas:** Apenas elementos próximos à regiões interceptadas devem ser modificados durante a reconstrução. Elementos afastados das regiões de interseção devem permanecer inalterados. Da mesma forma, se técnicas de suavização forem usadas para aumentar a qualidade das malhas, então ela também deve ser local.

- **Identificação de regiões:** As novas subdivisões planares definidas no espaço paramétrico das superfícies pelas curvas de *trimming* devem ser identificadas automaticamente. Estas regiões podem ou não fazer parte do modelo final gerado; além disso, elas podem inclusive possuir atributos diferentes (cargas, materiais, condições de contorno).

- **Reconstrução automática:** As duas malhas devem ser automaticamente redefinidas para incluir as curvas de interseção e possivelmente excluir as regiões excedentes. O usuário não deve ser responsável por edições manuais das malhas resultantes.

- **Eficiência:** O tempo e a memória necessários ao cálculo das interseções deve ser idealmente linear com o número de elementos das regiões de interseção. Algoritmos quadráticos com o número total de elementos das malhas são muito lentos para grandes malhas e não são apropriados para modelagem interativa.

- **Robustez:** Um número arbitrário de curvas de interseção pode ser gerado, possuindo geometria, topologia e pontos de interpolação diferentes. Em particular, curvas fechadas devem ser tratadas corretamente, e devem originar regiões com buracos quando recortadas.

3.2 O algoritmo proposto

O algoritmo proposto para reconstruir duas superfícies mapeadas A e B possui três passos básicos, que contribuem para atender aos requisitos listados na última seção:

1. Determinação dos pontos de interseção:

   (a) Calcular e armazenar as interseções das arestas em A contra as faces em B, e

   (b) Calcular e armazenar as interseções das arestas em B contra as faces em A.

2. Determinação das curvas de interseção:

   (a) Conectar os pontos de interseção para construir a lista de linhas poligonais que representam as curvas de interseção;

   (b) Interpolar curvas paramétricas passando pelos pontos das linhas poligonais;

   (c) Calcular novos vértices com espaçamento adequado para estas malhas sobre estas curvas; e

   (d) Projetar estes novos vértices em ambas as superfícies.

3. Reconstrução das topologias:

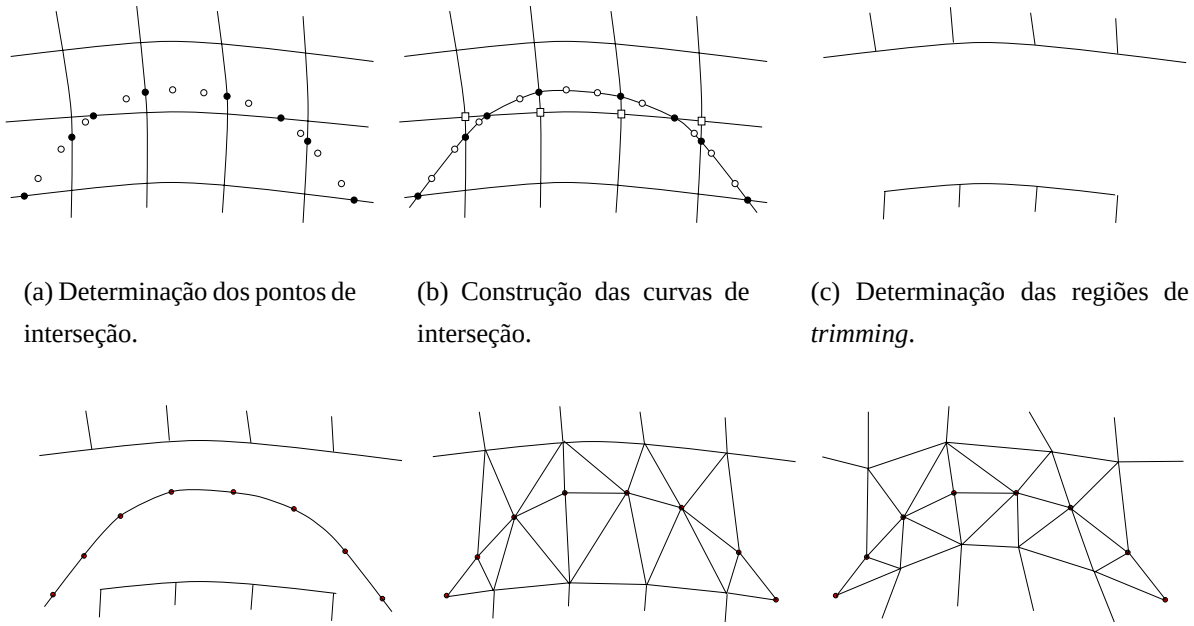
(a) Determinar as regiões de *trimming* removendo vértices e arestas com base nas linhas poligonais;

(b) Inserir novas arestas sobre as curvas de *trimming* usando os novos pontos definidos no Passo 2;

(c) Triangular as regiões de *trimming* em cada superfície; e

(d) Suavizar ambas as malhas.

A idéia geral do algoritmo de reconstrução é mostrada na Figura 3.4. A Figura 3.4a mostra os pontos de interseção. Os círculos cheios indicam as posições onde as arestas da superfície mostrada cruzam as faces da outra superfície, e os círculos vazados representam os cruzamentos das arestas da outra superfície. A Figura 3.4b mostra a curva de *trimming* obtida pelo processo de conexão determinado pelo Passo 2. A Figura 3.4c ilustra o Passo 3a, onde as arestas adjacentes aos vértices próximos, mostrados com marcas quadradas na Figura 3.4b, foram removidas. A Figura 3.4d ilustra o Passo 3b, onde as novas arestas que conectam os vértices igualmente espaçados foram inseridas na superfície mostrada. A Figura 3.4e mostra a triangulação antes da aplicação do algoritmo de suavização. A Figura 3.4f mostra o aspecto da malha após os reposicionamentos feitos com a técnica de suavização.



(a) Determinação dos pontos de interseção.

(b) Construção das curvas de interseção.

(c) Determinação das regiões de *trimming*.

(d) Inserção das curvas de *trimming*.

(e) Triangulação inicial das regiões de *trimming*.

(f) Suavização das malhas.

Figura 3.4: Uma visão geral do algoritmo.

De forma a evitar o teste de todas as arestas contra todas as faces no Passo 1, armazenam-se as entidades topológicas em árvores de indexação espacial, da forma mostrada na Seção 3.3. Uma vez que as arestas e faces podem apresentar curvaturas acentuadas em três dimensões, usa-se um procedimento numérico para determinar os pontos de interseção, que é descrito na Seção 3.4. Ao final do Passo 1, as arestas de uma malha estão ligadas com as faces que elas interceptam na outra malha, e vice-versa. Para cada par aresta/face interceptado, são armazenadas também as coordenadas paramétricas do ponto de interseção, nos espaços paramétricos respectivos.

No Passo 2b, as curvas de *trimming* são calculadas no espaço paramétrico por conexão e interpolação dos pontos calculados no Passo 1. Nos Passos 2b–d, são determinadas representações contínuas para as curvas de *trimming* no espaço paramétrico. Pontos igualmente espaçados sobre as curvas de *trimming* são então determinados e *relaxados* para as superfícies originais. Mais detalhes são mostrados na Seção 3.5.

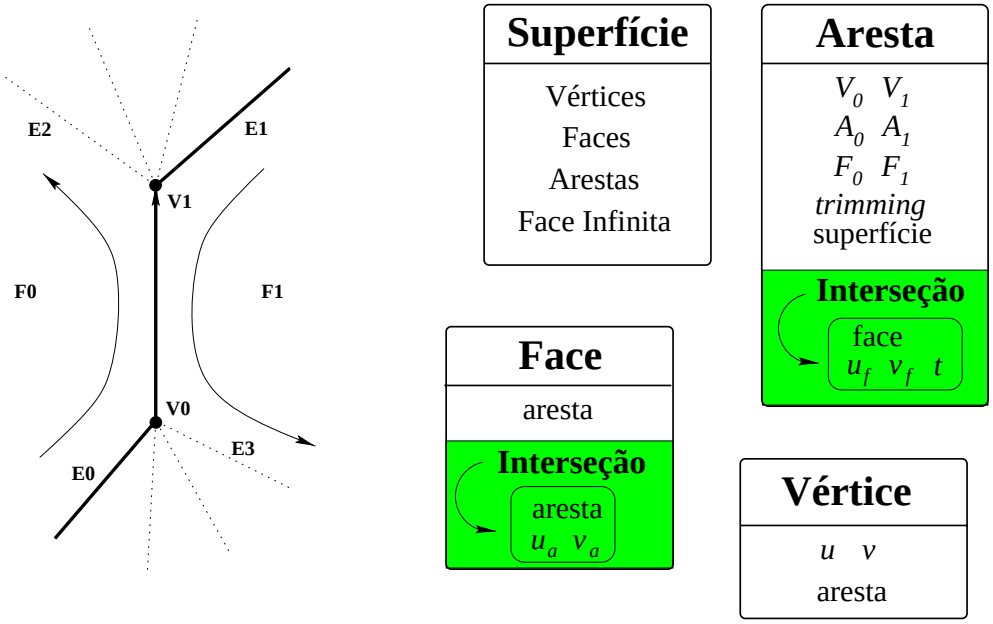
No Passo 3a, as regiões de *trimming* são identificadas. Estas regiões, na realidade sub-retalhos paramétricos, são faces da estrutura de dados (vide Seção 3.3) ampliadas pela eliminação de um conjunto de arestas. Ao final do Passo 3a existem, em cada superfície, tantas regiões quantas forem as curvas de *trimming*. O Passo 3b consiste na inserção das arestas que representam as curvas de *trimming*, o que conduz à subdivisão de cada face de *trimming* em duas. Após isso, no Passo 3c, cada região de *trimming* é triangulada pela inserção de arestas, obedecendo aos critérios geométricos definidos na Seção 3.6. Por fim, a aplicação da técnica de suavização mostrada na Seção 3.6.4 aumenta a qualidade dos elementos próximos à região de interseção.

3.3 A estrutura de dados

Em cada superfície, constrói-se uma subdivisão planar de cada malha armazenando-a em uma estrutura de dados variante da DCEL [71] (*Doubly Connected Edge List*), estendida para conectar duas topologias e trabalhar com superfícies. Além disso, as entidades topológicas (vértices, arestas e faces) são armazenadas em árvores, ao invés de listas encadeadas ou vetores. Este armazenamento reduz os tempos de buscas necessários à determinação dos pontos de interseção (Passo 1 do algoritmo) e à construção da malha inicial (vide Apêndice B).

A estrutura DCEL é bastante semelhante à *Winged Edge* [7], suprimindo-se apenas duas arestas. A Figura 3.5a mostra as informações topológicas armazenadas pelas arestas, onde as linhas mais grossas indicam as arestas diretamente presentes na estrutura. A Figura 3.5b mostra as entidades da estrutura de dados utilizada, e seu encadeamento topológico.

A entidade básica correspondente ao grafo planar em 2D usada neste trabalho é a *Superfície*. Cada superfície deve possuir uma parametrização bidimensional bem definida (su-



(a) Aresta na estrutura DCEL.

(b) A estrutura de dados.

Figura 3.5: A estrutura DCEL estendida.

perfícies de Coons bilineares ou trilineares, e os *sweeps* descritos no Capítulo 2 foram os tipos tratados), referências para as árvores de vértices, arestas e faces, e uma única referência para a *Face Infinita*[1]. Ainda com relação à parametrização, as duas superfícies envolvidas na interseção devem apresentar intervalos paramétricos iguais, de zero a um em cada eixo por exemplo, ou uma normalização deve ser feita para que a convergência do algoritmo do Passo 1 seja rápida (vide Seção 3.4). Usa-se o espaço paramétrico bidimensional para orientar as entidades geométricas da mesma forma que em subdivisões planares.

Os *Vértices* possuem descrição geométrica definida pelas coordenadas paramétricas (u, v) associadas à superfície a que pertencem. A única referência topológica existente na definição dos vértices é uma aresta incidente, da mesma forma que na *Winged Edge* padrão.

As *Arestas*, que formam a base da estrutura, são os lados de cada elemento da malha existente. As arestas são consideradas retas no espaço paramétrico da superfície, mas apresentam geometria curva em três dimensões. Os campos V_0, V_1, A_0, A_1, F_0 e F_1, mostrados na Figura 3.5, correspondem respectivamente aos vértices inicial e final, à aresta anterior na face F_0 ou próxima anti-horária em V_0, à aresta anterior na face F_1 ou próxima anti-horária em V_1, à face à esquerda e à face à direita. Além das referências topológicas, armazena-se um indicador (*trimming*) que

[1]Da mesma forma que em subdivisões planares, na DCEL a face infinita corresponde ao ciclo externo de arestas, que apresenta orientação inversa de todas as outras faces.

é ativado para as arestas que se situam sobre alguma curva de interseção. Esta referência é que viabiliza a identificação de regiões disjuntas, após os cálculos de interseção. Os campos armazenados em *Interseção* são apenas instanciados para as arestas que possuem interseção com uma face qualquer. O campo *face* é a referência para a face da outra superfície que é interceptada pela aresta. Os campos u_f e v_f correspondem às coordenadas paramétricas referentes à superfície da face armazenada em *face*.

As *Faces* são os sub-retalhos limitados pelas arestas definidas no mapeamento da superfície completa; possuindo uma referência para uma aresta na própria superfície. De maneira similar às arestas, os campos: *aresta*, u_a e v_a em *Interseção* apenas são instanciados quando se detecta uma aresta da outra superfície que intercepta a face.

O campo *Interseção* das arestas e faces é usado na determinação das curvas de *trimming* e na reconstrução das topologias; ele é a chave para o encadeamento topológico de duas DCELs, feito quando duas superfícies se interceptam. A Figura 3.6 mostra como este encadeamento topológico de arestas e faces de duas superfícies se processa. A informação geométrica de cada ponto de interseção é armazenada nos pares (u_f, v_f) e (u_a, v_a), nos campos instanciados para interseção mostrados pela Figura 3.5.

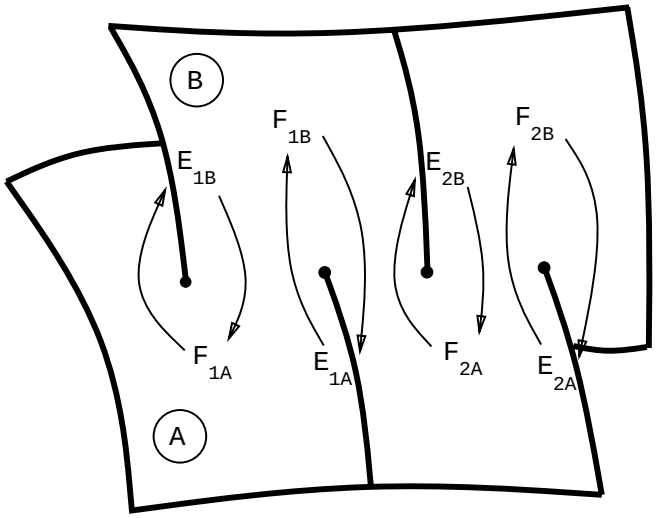


Figura 3.6: Encadeamento topológico das DCELs.

Para o armazenamento das entidades topológicas, adotou-se árvores B-trees [26] para os vértices e arestas, e R*-trees [8] para as faces. As B-trees são determinantes para que na construção das DCELs iniciais, que representam as malhas, não sejam necessárias buscas em todas as entidades existentes (vide Apêndice B). As R*-trees são fundamentais para que os pontos de interseção sejam obtidos com poucos testes entre faces e arestas (vide Seção 3.4.1). Os vértices são inseridos em uma B-tree, que usa as coordenadas paramétricas dos pontos para fazer buscas

em ordem lexicográfica. Cada vértice é batizado com um índice de forma que vértices com índices menores também possuem coordenadas paramétricas menores, de maneira lexicográfica. As arestas são orientadas do vértice de menor índice para o de maior, e são armazenadas em uma B-tree que usa estes índices como chave de busca, de forma semelhante à B-tree de vértices. As faces são inseridas em uma R*-tree usando-se o *bounding box*[2] tridimensional como chave de inserção e busca. Para exemplificar a construção das R*-trees, a Figura 3.7 mostra árvore bidimensional com número de faces por nó entre 2 e 4.

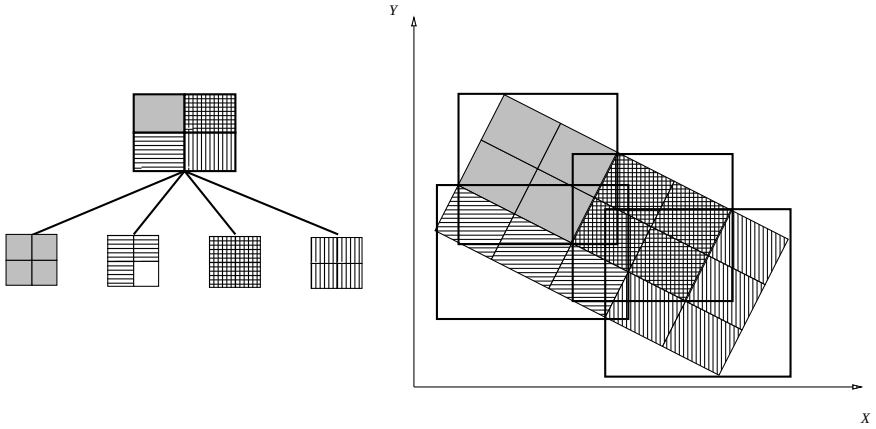


Figura 3.7: R*-tree bidimensional contendo de duas a quatro faces por nó.

A determinação dos *bounding boxes* das arestas é feita com base nos pontos de uma linha poligonal adaptativa com a curvatura, construída com a metodologia descrita na Seção 2.1.2. Para o cálculo dos *bounding boxes* das faces, usam-se os pontos obtidos de uma amostragem definida pelas arestas da face que apresentam maiores curvaturas, e portanto mais pontos amostrais.

O Apêndice A descreve o conjunto de operadores topológicos necessários a todas as alterações feitas pelo algoritmo de interseção, apresentados no Passo 3 (Seção 3.6). O Apêndice B mostra como é feita a construção da topologia inicial partindo-se de uma estrutura simples de vértices e faces (estrutura convencional usada para malhas de elementos finitos).

3.4 Determinação dos pontos de interseção

No Passo 1 do algoritmo, as arestas em uma superfície são testadas com as faces da outra superfície que potencialmente a interceptam.

[2]Existem formas de se determinar o *bounding box* preciso de um retalho sem a necessidade de subdivisão, avaliando-se apenas as posições 3D e os vetores tangentes nos pontos extremos [6].

3.4.1 Cerceamento com as R*-trees

A R*-tree de faces organizada pelos *bounding boxes* tridimensionais torna a busca pelas faces que potencialmente interceptam uma aresta em uma pesquisa limitada a alguns nós da árvore. Árvores com um mínimo de 4 e um máximo de 10 faces por nó apresentam bom rendimento [60] no caso médio para subdivisões planares, tendo sido adotados estes limites na implementação da biblioteca de R*-trees tridimensionais usada no MG. Para arestas e faces que possuem *bounding boxes* muito menores do que os da superfície completa, obtém-se um ganho significativo de desempenho nas buscas.

O *bounding box* da aresta é usado para determinar na R*-tree o conjunto de faces que podem interceptá-la. A Figura 3.8 mostra a interseção de uma aresta reta com uma superfície de Coons bilinear formada por duas retas e duas Béziers cúbicas nas fronteiras. A aresta, o seu *bounding box*, as faces superfície de Coons, e os *bounding boxes* das faces que potencialmente interceptam a aresta são desenhadas.

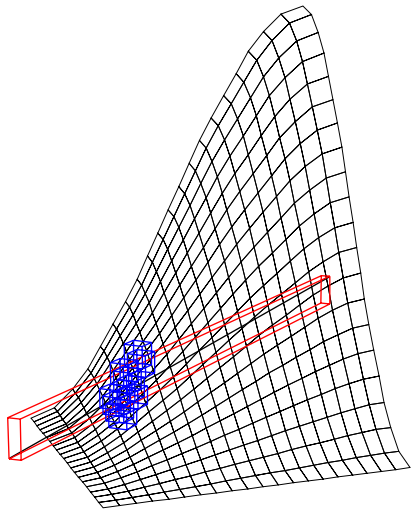


Figura 3.8: Faces de um retalho de Coons que potencialmente interceptam uma aresta.

3.4.2 Interseção entre arestas e faces

Como Faux e Pratt [36] colocam, o cálculo da curva de interseção entre duas superfícies pode ser visto como um problema de resolução de equações (geralmente não-lineares) simultâneas, ou como um problema de minimização, onde o quadrado da norma euclidiana entre os pontos nas duas superfícies é minimizado com o ajuste dos parâmetros u e v. O problema de determinação do ponto de interseção entre a aresta-curva e o pseudo-retalho, que é a face que apresenta interseção potencial com a aresta, também pode ser transformado em um problema de minimização, cuja

solução é obtida usando-se as técnicas de programação matemática [85]. Para cada par aresta-face selecionado, acha-se o ponto onde eles se interceptam pela solução de um problema de minimização: o algoritmo termina quando a distância euclidiana do ponto avaliado na aresta com o parâmetro t ao ponto na face avaliado com as coordenadas (u, v) for menor que uma dada tolerância. Esta tolerância corresponde a uma fração do menor comprimento das arestas envolvidas na interseção, que deve ser determinado a priori.

A metodologia adotada nesta tese se assemelha à técnica usada por Chen e Ozsoy [21], que aproximam a solução do problema de interseção de superfícies como um conjunto de interseções entre curva e superfície. Chen e Ozsoy formulam o problema em função das coordenadas paramétricas nos dois retalhos chegando a um sistema de quatro incógnitas (pares (u, v) em uma superfície e (w, s) na outra) e três equações, relativas aos três eixos do espaço tridimensional. A restrição adicional usada por Chen e Ozsoy é chamada de *engaging direction* e está relacionada com círculo osculador, definindo uma direção no espaço paramétrico da primeira superfície, o que reduz o problema à interseção desta *curva* com a outra superfície. No caso de arestas retas no espaço paramétrico da superfície que a contém adotado nesta tese, a direção da curva está bem determinada, não existindo a necessidade de adoção de restrições extras para resolver o problema. A Figura 3.9 mostra os elementos envolvidos no problema de interseção da aresta-curva (V_0, V_1), que está sobre uma superfície (não desenhada), com uma superfície $S(u, v)$. As coordenadas (u_a, v_a) podem ser substituídas pelo parâmetro t, que define a reta paramétrica no espaço da superfície não desenhada.

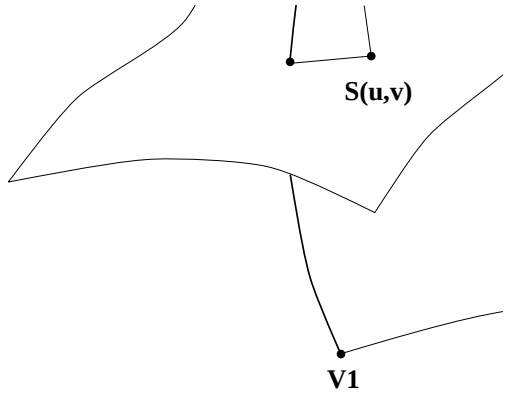


Figura 3.9: Notação usada para o problema de interseção.

O vetor distância $\vec{F}$ é, em última análise, uma função dos parâmetros u, v, e t:

$$\vec{F}(u, v, t) \;=\; \left\{ \begin{array}{c} S_x(u, v) - R_x(t) \\ S_y(u, v) - R_y(t) \\ S_z(u, v) - R_z(t) \end{array} \right\} . \tag{3.1}$$

Para resolver o problema $\vec{F}(\vec{p}) = \vec{0}$, onde $\vec{p} = (u, v, t)$, usando método de Newton-Raphson, o termo de correção $\Delta\vec{p}$ é dado por:

$$\mathbf{J}^t \Delta\vec{p} \;=\; -\vec{F}(\vec{p}), \tag{3.2}$$

onde a matriz jacobiana **J** é dada por:

$$\mathbf{J} \;=\; \nabla\vec{F}(\vec{p})^t \;=\; \begin{bmatrix} \dfrac{\partial S_x}{\partial u} & \dfrac{\partial S_y}{\partial u} & \dfrac{\partial S_z}{\partial u} \\ \dfrac{\partial S_x}{\partial v} & \dfrac{\partial S_y}{\partial v} & \dfrac{\partial S_z}{\partial v} \\ -\dfrac{\partial R_x}{\partial t} & -\dfrac{\partial R_y}{\partial t} & -\dfrac{\partial R_z}{\partial t} \end{bmatrix} \tag{3.3}$$

Uma vez que a matriz jacobiana é composta pelas tangentes direcionais no retalho de superfície e pela tangente à curva, que também se traduz em uma derivada direcional na superfície da aresta, deve-se ter cuidado com o módulo dos vetores tangente, para garantir a convergência. Se as superfícies possuem parametrizações diferentes, uma bicúbica e outra bilinear, por exemplo, então uma normalização dos vetores é necessária antes de se resolver a Equação 3.2. O mesmo procedimento deve ser observado se os intervalos paramétricos nos dois retalhos forem diferentes.

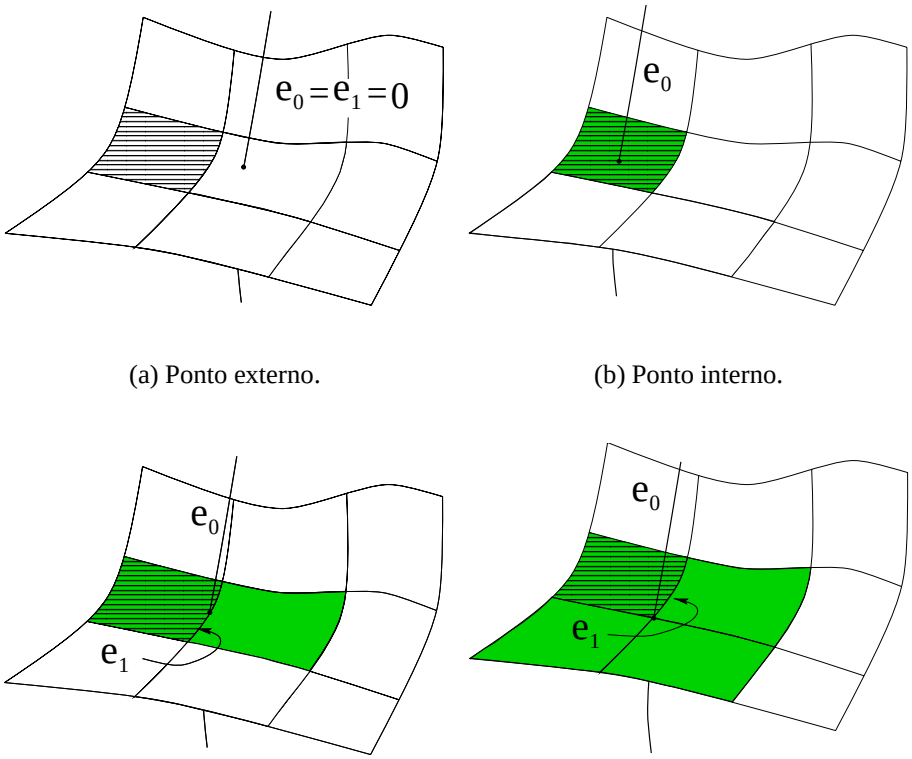
Para tornar o algoritmo suficientemente robusto de forma a tratar corretamente as singularidades que ocorrem em pontos de tangência ou paralelismo entre face e arestas, usa-se o método modificado de Newton-Raphson apresentado por Deuflard [34]. Mesmo quando o ponto de interseção entre uma aresta e face não se dá em uma posição de tangência, é comum acontecerem situações onde o sistema fica mal-condicionado em iterações intermediárias, sendo portanto fundamental o uso das pseudo-inversas de Deuflard.

3.4.3 Posições relativas no espaço paramétrico

A seção anterior descreve o algoritmo de interseção entre uma aresta e uma face, que fornece como resultado dois pares paramétricos: (u_f, v_f) na superfície da face, e (u_a, v_a) (ou t) na superfície de aresta. É necessário ainda que a posição relativa entre a face e o ponto de interseção determinado na aresta seja classificado por testes no espaço paramétrico do retalho da face. Um algoritmo de ponto em polígono (as arestas da face são retas paramétricas) usando a técnica de *single shot* verifica se o ponto de interseção é interno, externo, sobre uma fronteira, ou sobre um vértice da face em questão.

A Figura 3.10 mostra as posições relativas que um ponto de interseção entre uma aresta da superfície A (apenas a aresta é desenhada) e uma face da superfície B (desenhada com hachuras)

podem assumir. Nesta figura, as faces da superfície B que devem ser atualizadas nos campos *Interseção* (vide Figura 3.5b) são preenchidas com um tom cinza. A Figura 3.10a mostra o caso em que o ponto é externo à face testada; neste caso nenhuma atualização é feita na estrutura de dados. Quando o ponto é interno à face, como mostra a Figura 3.10b, o par aresta-face é atualizado nos campos *Interseção*. Se o ponto de interseção se localiza sobre uma aresta da superfície, como mostra a Figura 3.10c, além da aresta interceptada (na superfície A), as duas faces vizinhas à aresta na superfície B também têm seus campos em *Interseção* atualizados. Quando a interseção se dá sobre um vértice da face testada, como mostra a Figura 3.10d, todas as faces adjacentes a este vértice devem também ser atualizadas.



(a) Ponto externo.

(b) Ponto interno.

(c) Ponto em aresta.

(d) Ponto em vértice.

Figura 3.10: Posições relativas entre ponto de interseção e face.

Em qualquer destes casos, quando o campo *Interseção* tem que ser atualizado, se ele já se encontra instanciado em cálculos anteriores, nenhuma modificação é necessária. Como é mostrado na Seção 3.5, uma única referência em cada aresta e face é suficiente para conectar-se todos os pontos de interseção, de forma que não são necessárias atualizações quando a aresta ou face já tiver o campo *Interseção* preenchido.

Encerrando os testes para atualização da conexão entre as duas superfícies, para os casos

em que o ponto de interseção não é externo à face (Figuras 3.10b a d), deve-se testar se o ponto ocorre em um dos vértices da aresta. Se isso ocorrer, deve-se atualizar todas as arestas adjacentes a este vértice, o que evita que o mesmo ponto fronteiriço seja calculado pelas arestas vizinhas.

3.4.4 Troca das superfícies

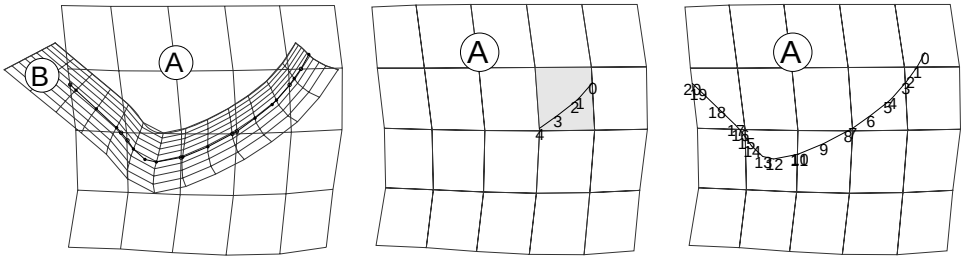
Para que a determinação dos pontos de interseção esteja completa e para que os campos necessários à construção das curvas de *trimming* estejam corretamente instanciados e preenchidos, todos os procedimentos descritos nesta seção devem ser repetidos trocando-se as superfícies, ou seja, testando as arestas da segunda superfície contra as faces da primeira. Ao final deste passo, todas as informações topológicas e geométricas necessárias para construir as curvas de interseção encontram-se armazenadas nas duas DCELs.

3.5 Determinação das curvas de interseção

O Passo 2 é feito pelo percorrimento das faces interceptadas da primeira superfície, propagando-se as interseções pelas faces adjacentes a arestas interceptadas, identificadas pelo campo *Interseção*. Em cada ponto armazenam-se referências para as arestas interceptadas em cada superfície.

3.5.1 Poligonais locais e globais

São usados dois tipos de linhas poligonais: *poligonais locais*, que conectam pontos de interseção em uma face determinada; e *poligonais globais*, que são formadas de poligonais locais e representam uma poligonal de *trimming* completa. Ambas poligonais local e global são simplesmente uma lista de pontos. Cada um destes pontos contém duas referências para arestas: a_0 e a_1. Em pontos de interseção regulares apenas o campo a_0 não é nulo, contendo uma referência para uma aresta em uma das superfícies. O campo a_1 apenas é preenchido quando o ponto de interseção representa um encontro entre duas arestas (vide Figuras 3.10c e d), uma em cada superfície. Nestes casos deve-se armazenar em a_1 uma referência para a aresta que esteja na outra superfície, diferente da superfície da aresta em a_0. No caso mostrado na Figura 3.10d, qualquer das arestas adjacentes ao vértice interceptado pode ser referenciada. A Figura 3.11a mostra a curva de interseção e os pontos de interseção de duas superfícies A e B. A Figura 3.11b mostra a poligonal local referente à face desenhada em cinza. Os pontos 0 e 4 correspondem a interseções das arestas desta face com as faces da superfície B; os pontos 1, 2, e 3 correspondem a interseções de arestas da superfície B internas à face desenhada em cinza na superfície A. A Figura 3.11c



(a) Pontos de interseção. (b) Poligonal local. (c) Poligonal global.

Figura 3.11: Poligonais local e global usadas no Passo 2 do algoritmo.

mostra a poligonal global obtida pela conexão de poligonais locais ao longo de todas as faces interceptadas da superfície A.

Cada face da superfície A é marcada à medida em que é visitada; o Passo 2a termina quando todas as faces interceptadas já foram visitadas. Desta forma, todos os componentes conexos, ou curvas de interseção, são determinados em uma única busca nas faces da superfície A.

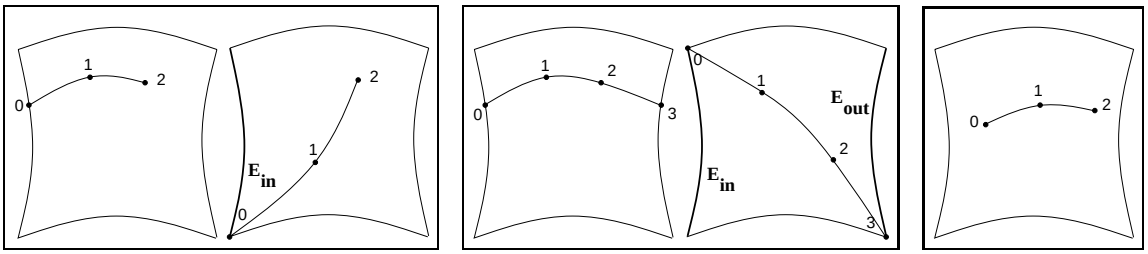
3.5.2 Casos previstos

Uma classificação dos casos previstos é necessária para apresentação do algoritmo de propagação de poligonais do Passo 2. A Figura 3.12 mostra as três possibilidades previstas para posições relativas entre poligonal local e face. As classificações que podem ser feitas com o par face/poligonal local são:

Singlecrossed: Quando uma única aresta é interceptada (vide Figura 3.12a);

Doublecrossed: Quando duas arestas são interceptadas (vide Figura 3.12b); e

Isolated: Quando nenhuma aresta é interceptada e a poligonal é isolada (Figura 3.12c).



(a) *Singlecrossed*. (b) *Doublecrossed*. (c) *Isolated*.

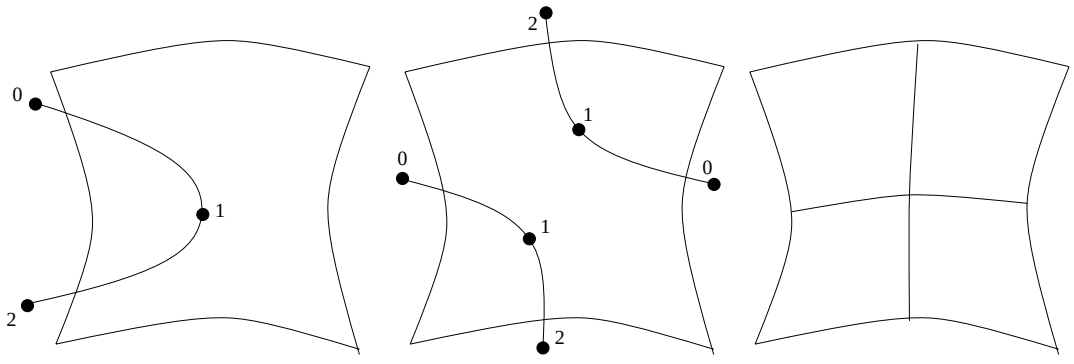
Figura 3.12: Classificações entre face e poligonal local.

O caso mais comum é o *Doublecrossed*, onde a poligonal global cruza completamente a face. Deve ser notado que a poligonal global é sempre propagada através de arestas ou vértices interceptados, e apenas pode terminar sobre um ponto de interseção interno a uma face ou sobre uma aresta, seja ela de fronteira ou não.

Para os casos em que as interseções das arestas ocorrem em um dos vértices (ponto 0 da Figura 3.12a à direita e pontos 0 e 3 da Figura 3.12b à direita), o que ocorre muito freqüentemente nas interseções mostradas no Capítulo 4, propaga-se a poligonal global por todas as faces adjacentes a este vértice. Estes casos foram identificados no Passo 1, de forma que todas as arestas adjacentes devem ter sido atualizadas em seus campos *Interseção*.

Os casos possíveis de interseções entre dois retalhos paramétricos são muito mais variados do que os mostrados na Figura 3.12. O conjunto que representa a interseção pode ser uma superfície, quando os dois retalhos coincidem, uma ou mais curvas, um ponto, o conjunto vazio, ou qualquer combinação entre estes casos. O propósito da modelagem proposta nesta tese não é o de construção de um algoritmo que detecte todas as possibilidades geométricas de interseção entre dois retalhos, uma vez que o trabalho se concentra na reconstrução das malhas existentes. Instâncias de retalhos coincidentes e pontos isolados resultantes de retalhos mutuamente tangentes não são tratados pelo algoritmo proposto, devendo ser identificados e redefinidos pelo usuário do sistema. Dá-se ênfase aos casos em que as interseções são curvas bem definidas e em que os retalhos possuem malhas compatíveis com a complexidade das curvas de interseção.

Em termos de entidades da estrutura de dados, arestas que são interceptadas mais de uma vez pela mesma curva de interseção e faces com duas curvas de interseção distintas, como mostram as Figuras 3.13a e 3.13b respectivamente, não são tratadas pelo algoritmo. Ambos os casos podem resolvidos com um aumento da discretização, como o mostrado na Figura 3.13c.



(a) Uma aresta interceptada duas vezes.

(b) Uma face cruzada por duas curvas diferentes.

(c) A nova discretização.

Figura 3.13: Interseções não tratadas e uma solução.

O aumento da discretização deve ser feito por iniciativa do usuário do sistema, redefininindo as malhas envolvidas e fazendo nova chamada para o algoritmo de interseção. Outras alternativas para se resolver a limitação de um único cruzamento por aresta, não investigadas nesta tese, seriam a subdivisão das arestas que apresentassem uma segunda interseção ou o armazenamento de uma lista de faces interceptadas.

3.5.3 Construção das poligonais globais

A construção de lista de poligonais globais é mostrada pelo Pseudo-código 3.1. O *bounding box* determinado pelos pontos de interseção calculados no Passo 1 é usado como chave de busca na R*-tree de faces da superfície A, de forma a visitar quase que somente as faces que realmente foram interceptadas. Identificada uma face interceptada, inicializa-se uma nova poligonal global que é propagada recursivamente ao longo de toda a superfície pelo algoritmo *BuildGPoly*, apresentado a seguir, marcando as faces à medida em que são visitadas. O algoritmo *BuildGPolyList* prossegue gerando novas poligonais globais (componentes conexos) enquanto houver faces interceptadas e ainda não visitadas.

Pseudo-código 3.1 O algoritmo para construção da lista de poligonais globais.

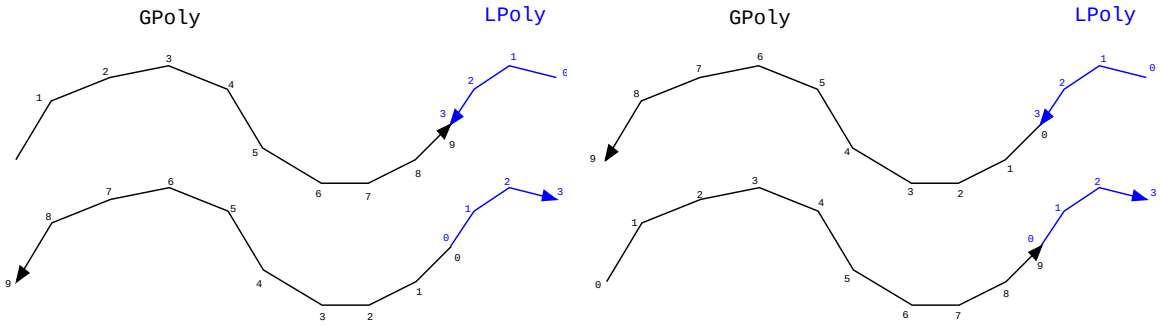
```
algorithm BuildGPolyList
   input: surface S1
   output: list of global poligonals List
begin
   Initializes List as empty
   for each Face in bounding box of Step 1 do
       if Face has intersection and Face has not been visited then
          Creates new GPoly
          BuildGPoly ( Face, GPoly )
          Adds GPoly to List
       end if
   end for
end BuildGPolyList
```

O algoritmo que propaga cada poligonal global ao longo da superfície, através das arestas interceptadas, é apresentado pelo Pseudo-código 3.2. Os parâmetros de entrada para o algoritmo são uma referência para a face onde a poligonal se inicia (*Fc*), e para a poligonal (*GPoly*) que será propagada pela superfície por chamadas recursivas. Ao final do algoritmo, a poligonal *GPoly* possui, para cada ponto de interseção, uma (a_0) ou duas (a_0 e a_1) referências para arestas das

superfícies. Estas informações serão usadas no Passo 3 do algoritmo, para definição das regiões (faces) de *trimming*, sem a necessidade de se fazer buscas para identificação de posições relativas das curvas de *trimming* com as faces em ambos os retalhos.

O teste de interrupção da recursão é feito no início do algoritmo, verificando-se se a face *Fc* se encontra marcada, ou seja, se ela já foi visitada pelo algoritmo em outro nível da recursão. Quando a face *Fc* corresponde à face infinita, a poligonal *GPoly* atingiu uma fronteira da representação e a recursão também deve ser interrompida. O algoritmo cria inicialmente a poligonal local *LPoly*, onde serão armazenadas todas as interseções da face *Fc*. O algoritmo *BuildLPoly* constrói a poligonal *LPoly* de *Fc*, retornando também referências para as possíveis arestas interceptadas: *Ein* e *Eout*, que correspondem respectivamente à primeira e segunda arestas de *Fc* com interseção. As referências *Ein* e *Eout* são usadas na classificação da face *Fc* de acordo com a Figura 3.12.

Os pontos de interseção de *LPoly* são então inseridos em uma das duas extremidades de *GPoly* pelo algoritmo *AddGPoly*. Um dos campos a_0 ou a_1 presente nas posições extremas de *LPoly* deve conter uma referência para uma aresta que coincide com os extremos de *GPoly*, indicando o ponto de conexão. A direção de ambas as poligonais são também classificadas de acordo com as adjacências. A Figura 3.14a mostra as situações em que a poligonal local deve ser inserida com sua direção invertida, e a Figura 3.14b mostra os casos em que a poligonal local deve ser inserida com a sua ordem natural.



(a) Situações de inversão.

(b) Situações de manutenção.

Figura 3.14: Direções relativas entre as poligonais local e global.

Em seguida, a face *Fc* é marcada como *visitada*, e sua classificação é feita de acordo com as referências em *Ein* e *Eout*. Se *Ein* existe (não é nula), testa-se se a interseção ocorre sobre um vértice, o que pode ser feito pela avaliação do parâmetro t (vide Figura 3.9). Se a interseção ocorre em uma posição intermediária entre os dois vértices ($0 < t < 1$), propaga-se a poligonal global pela face adjacente a *Fc* pela aresta *Ein*. Se a interseção ocorre sobre um vértice de *Ein*

Pseudo-código 3.2 O algoritmo para construção das poligonais globais.

```
algorithm BuildGPoly ( Fc, GPoly )
   input: face Fc and global polyline GPoly
   output: global polyline GPoly completed
begin
   if Fc is marked or Fc is infinity return
   Creates local polyline LPoly
   BuildLPoly ( LPoly, Fc, Ein, Eout )
   AddGPoly ( LPoly, GPoly )
   Removes LPoly
   Marks face Fc
   if Ein is not NULL then
      if intersection in Ein is over a vertex
         for each Fviz adjacent to the intersected vertex
            BuildGPoly ( GPoly, Fviz )
         end for
      else
         Fviz = face neighbor to Ein
         BuildGPoly ( GPoly, Fviz )
      endif
      if Eout is not NULL then
         if intersection in Eout is over a vertex
            for each Fviz adjacent to the intersected vertex
               BuildGPoly ( GPoly, Fviz )
            end for
         else
            Fviz = face neighbor to Eout
            BuildGPoly ( GPoly, Fviz )
         endif
      endif
   end if
end BuildGPoly
```

($t = 0$, indicando interseção em V_0, ou $t = 1$, indicando interseção em V_1), propaga-se a poligonal global por todas as faces adjacentes a este vértice. Se *Eout* existir, o que classifica a face *Fc* como *Doublecrossed*, repete-se o procedimento de propagação feito com a aresta *Ein*. Nos casos em que *Ein* e *Eout* são nulas, a poligonal é isolada no interior da face *Fc* e não precisa ser propagada pois já está completa após ter sido transportada para a poligonal global por *AddGPoly*.

3.5.4 Construção das poligonais locais

O Pseudo-código 3.3 mostra o algoritmo *BuildLPoly* que insere todos os pontos de interseção da face *Fc* na poligonal local *LPoly*. Os algoritmos *BuildLPoly* e *AddIntPoint* usam as informações topológicas calculadas no Passo 1 do algoritmo para construir a poligonal *LPoly* por percorrimento alternado nas arestas das faces em ambas as superfícies envolvidas. O algoritmo *BuildLPoly* percorre as arestas da face *Fc* a procura da primeira aresta (*Ein*) que possui interseção. Se tal aresta existir, armazena-se a sua referência na primeira posição no campo a_0 de *LPoly*. O algoritmo prossegue pela chamada ao algoritmo *AddIntPoint*, que insere os pontos de interseção internos à face *Fc*. Se a aresta armazenada em *Ein* não for nula, procura-se por uma outra possível aresta *Eout* da face *Fc* que também possua interseção, armazenando-a na última posição de *LPoly*, concluindo o algoritmo. A Figura 3.15 mostra uma situação em que *Ein* e *Eout* existem, e dois pontos internos foram inseridos por *AddIntPoint*.

Pseudo-código 3.3 O algoritmo para construção de poligonais locais.

```
algorithm BuildLPoly ( Fc, LPoly )
   input: face Fc and local polyline LPoly
   output: polyline LPoly with the intersection points of face Fc
begin
   gets Ein as the first intersected edge of face Fc
   if Ein exists inserts Ein at the first position of LPoly
   AddIntPoint ( Fc, Ein, LPoly )
   if Ein is null return
   gets Eout as the other intersected edge of face Fc
   if Eout exists inserts Eout at the last position of LPoly
end BuildLPoly
```

O algoritmo *AddIntPoint*, mostrado pelo Pseudo-código 3.4, parte da face da outra superfície usando a referência presente no campo *face* da aresta *Ein*, armazenando-a em *Fint*. Se a aresta *Ein* for nula, uma busca na R*-tree é feita pelo algoritmo *GetFaceIntersectingFace* para

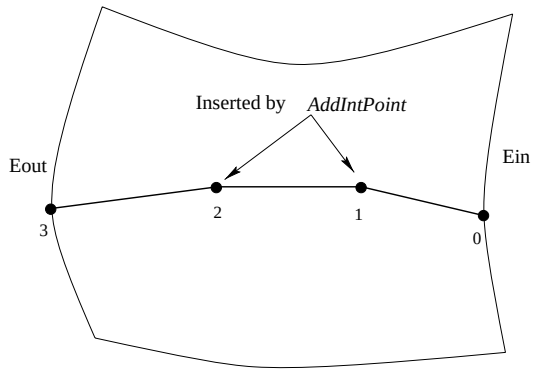


Figura 3.15: Poligonal local construída pelo algoritmo *BuildLPoly*.

achar a face *Fint*. Ressalta-se que apenas em poligonais isoladas, como as da Figura 3.12c, esta busca é necessária. Enquanto existirem faces *Fint* que apresentam interseção com a face *Fc* determina-se a aresta *Eint* que a intercepta (*GetEdgeIntersectingFace* checa as arestas de *Fint*), armazena-se a sua referência em *LPoly* e passa-se à face vizinha por esta aresta. Este *loop* termina em uma das três situações: quando a face vizinha já foi visitada, quando ela corresponde à face infinita, ou quando não existe nenhuma outra aresta interceptando a face *Fc*.

Pseudo-código 3.4 O algoritmo para inserção dos pontos internos a uma face.

```
algorithm AddIntPoint ( Fc, Ein, LPoly )
   input: face Fc, edge Ein and local polyline LPoly
   output: polyline LPoly with the intersections internal to Fc
begin
   if Ein is not null Fint = Ein->face
   else GetFaceIntersectingFace ( Fc, Fint )
   while Fint is not null then
      GetEdgeIntersectingFace ( Fint, Eint )
     if Eint is no null
       adds Eint to LPoly
       assigns to Fint the face neighbour to Fint by Eint
     else
       Fint = null
     end if
   end while
end AddIntPoint
```

3.5.5 Curvas contínuas

Para converter as poligonais globais encontradas no Passo 2a em curvas de interseção contínuas, constróem-se B-splines cúbicas por trechos que interpolam os pontos de interseção presentes em cada poligonal global. Para isto, usa-se a variação paramétrica centrípeta definida por Foley e Nielson [40] (vide Equação 2.5),que apresenta bons resultados para pontos com espaçamento muito variado, como é o caso das poligonais globais na quase totalidade dos casos. Esta interpolação é feita com a metodologia descrita ao longo da Seção 2.1.1, com a diferença de que os pontos das curvas de interseção são inseridos automaticamente.

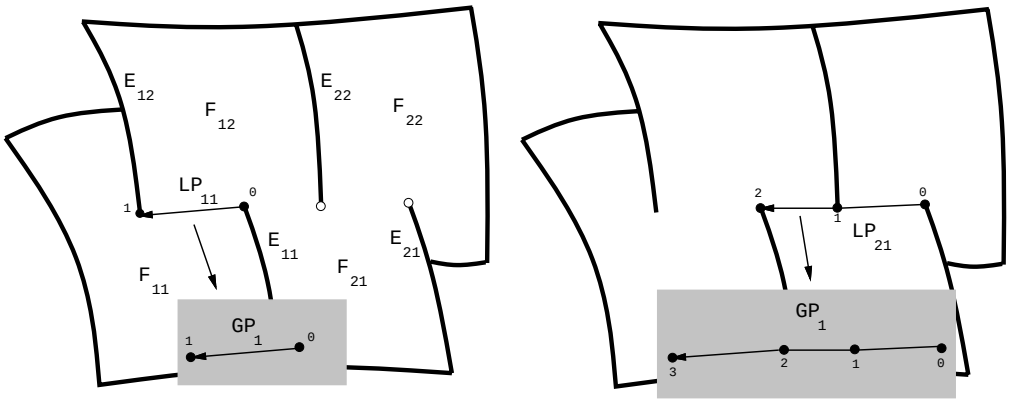
De posse destas representações contínuas para as curvas de interseção, deve-se amostrar pontos igualmente espaçados ao longo da curva (vide Seção 2.1.2), usando uma distância que corresponde a uma fração (adotou-se o valor 0.9) do tamanho médio das arestas interceptadas nas duas malhas iniciais. Estes pontos amostrados definem vértices representando as curvas de interseção na malha combinada. Mesmo considerando que os pontos de interpolação estão situados sobre ambas as superfícies segundo tolerância determinada, os pontos amostrados podem não estar, devendo ser *relaxados* de forma que as novas posições se aproximem das duas superfícies. Tal tarefa é feita por um algoritmo muito semelhante ao procedimento iterativo mostrado no Passo 1, mas de convergência muito mais rápida devido à proximidade dos pontos no início do processo. Como resultado deste algoritmo, pares de coordenadas paramétricas em ambas as superfícies são determinados para cada ponto amostrado.

3.5.6 Um exemplo

Para exemplificar os procedimentos referentes ao Passo 2 do algoritmo, aborda-se a construção da poligonal global gerada pelas superfícies A e B, mostradas na Figura 3.16. Supondo que a busca na R*-tree de faces de A forneça a Face F_{1A} para iniciar a primeira poligonal global GP_1, em *BuildGPolyList*; o algoritmo *BuildGPoly* é então chamado com F_{1A} e GP_1 como parâmetros.

Após a criação da poligonal local LP_{11}, o algoritmo *BuildGPoly* chama *BuildLPoly*, passando F_{1A} e LP_{1A} como parâmetros. A primeira aresta interceptada de F_{1A} é A_{1A}, que é armazenada na primeira posição de LP_{1A} e em *Ein*, passando-se à determinação das interseções internas com a chamada à função *AddIntPoint*, tendo F_{1A}, A_{1A} (*Ein*) e LP_{1A} como parâmetros. Em *AddIntPoint* percorrem-se as arestas da face F_{1B}, encontrando-se A_{1B} que apresenta interseção com F_{1A}. A aresta A_{1B} é então inserida na segunda posição de LP_{1A}; a face vizinha a F_{1B} por A_{1B} é a face infinita, o que encerra *AddIntPoint*.

A busca por uma outra aresta interceptada em F_{1A}, em *BuildLPoly*, resulta vazia, de forma que a poligonal local está completa ($LP_{1A} = \{A_{1A}, A_{1B}\}$), sendo inserida na poligonal global



(a) Poligonal local LP_{1A}.

(b) Poligonal local LP_{2A}.

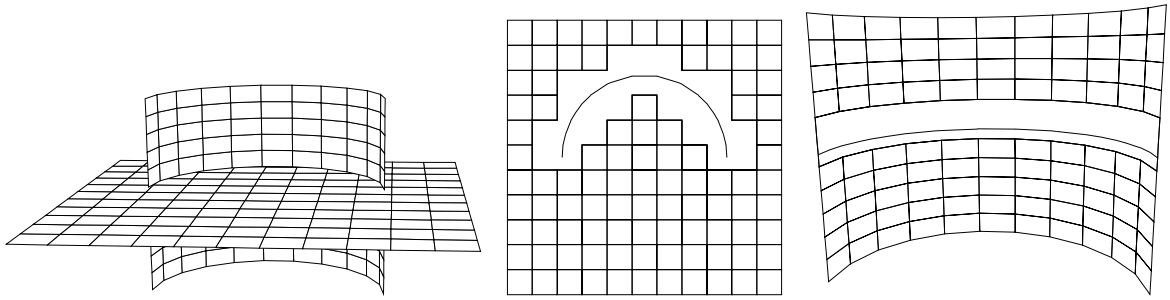
Figura 3.16: Poligonais locais e global (GP_1) no passo 2 do algoritmo.

GP_1, que se encontrava vazia. A Figura 3.16a ilustra este estágio. Após a remoção de LP_{1A} e a marcação da face F_{1A}, o algoritmo *BuildGPoly* classifica a face F_{1A} como *Singlecrossed*; o algoritmo *BuildGPoly* é então chamado recursivamente, com a face F_{2A} vizinha à aresta A_{1A} como parâmetro.

Seguem-se a criação de nova poligonal local LP_{2A} referente à face F_{2A} e o armazenamento da aresta A_{2A} na primeira posição desta poligonal, dentro do procedimento *BuildLPoly*. Em *AddIntPoint*, A_{2B} é inserida na segunda posição de LP_{2A}, passando-se à face F_{1B} onde nenhuma outra aresta, além de A_{2B}, intercepta F_{2A}. De volta à *BuildLPoly*, a última aresta A_{1A} é inserida em LP_{2A}. Quando o controle retorna ao algoritmo *BuildGPoly*, a poligonal local LP_{2A} possui três interseções: $\{A_{2A}$, A_{2B} e $A_{1A}\}$. O procedimento *AddGPoly* insere a poligonal LP_{2A} em GP_1, que passa a possuir as interseções: $\{A_{1B}, A_{1A}, A_{2B}, A_{2A}\}$, como mostra a Figura 3.16b. A face F_{21} é então marcada e classificada como do tipo mostrado na Figura 3.12b, sugerindo propagações por duas faces vizinhas. A face vizinha F_{1A} por A_{1A} já se encontra marcada, e a face vizinha por A_{2A} é a face infinita da superfície S_1, de forma que a recursão termina. O controle retorna então ao algoritmo *BuildGPolyList*, que insere a poligonal GP_1 na lista de poligonais, passando à segunda face (F_{2A}) de S_1, que por já se encontrar marcada encerra o algoritmo.

3.6 Reconstrução das topologias

Cada curva gerada no Passo 2 origina uma região de *trimming* em cada um dos retalhos, pela remoção de grupos de arestas próximas à curva de interseção.



(a) Malhas originais.

(b) Face e curva de *trimming* no plano.

(c) Face e curva de *trimming* na casca cilíndrica.

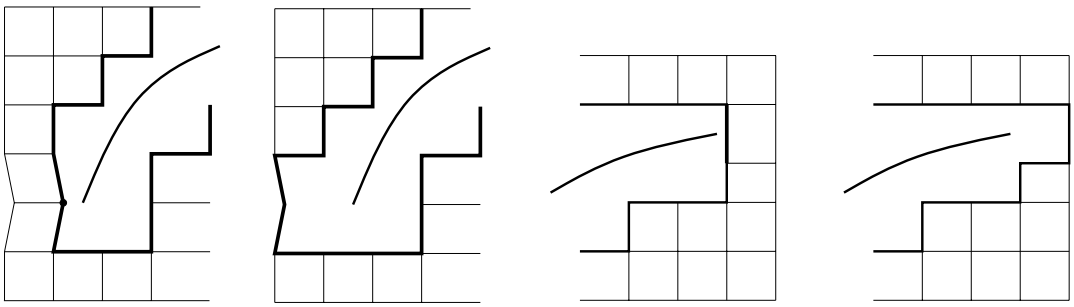
Figura 3.17: Faces de *trimming* em uma interseção plano×casca cilíndrica.

3.6.1 Obtenção das regiões de trimming

As regiões de *trimming* são na realidade faces de cada representação topológica, que são expandidas à medida em que se remove as arestas próximas aos pontos de interseção. As arestas interceptadas estão diretamente armazenadas nos campos a_0 e a_1 da cada ponto de interpolação das curvas de interseção. Para se obter as faces de *trimming*, no Passo 3a, para cada ponto de interpolação das curvas de interseção, removem-se todas as arestas conectadas ao vértice mais próximo destes pontos (mostrados com marcas quadradas na Figura 3.4b), incluindo a própria aresta interceptada. A determinação do vértice mais próximo é feita testando-se o valor do parâmetro t, que indica a posição da interseção em cada aresta. Se este valor for menor do que 0.5, são removidas as arestas conectadas ao vértice V_0 da aresta; de outra forma, remove-se as arestas conectadas ao vértice V_1.

As arestas pertencentes à fronteira, identificadas por serem adjacentes à face infinita, e aquelas geradas sobre curvas de interseção em cálculos anteriores, identificadas pelo campo *trimming*, não podem ser removidas sendo subdivididas pela inserção de um novo vértice na posição de cruzamento. A Figura 3.17 mostra as faces de *trimming* identificadas pela remoção e subdivisão de arestas no caso da interseção entre uma superfície plana e uma casca cilíndrica. Na Figura 3.17a, mostram-se as malhas originais; na Figura 3.17b a face de *trimming* na superfície plana é mostrada; e na Figura 3.17c mostra-se a face de *trimming* na casca cilíndrica.

As faces de *trimming* possuem apenas um ciclo de arestas, se geradas por curvas de interseção abertas. Se o procedimento de remoção de arestas fosse aplicado às curvas fechadas, faces auto-conectadas com um ciclo externo e um interno seriam geradas, não podendo ser representadas pela estrutura DCEL. As curvas de *trimming* fechadas são, por esta razão, tratadas de uma maneira um pouco diferente, fazendo-se a subdivisão de algumas arestas internas inter-



(a) Vértice próximo. (b) Vértice removido. (c) Aresta próxima. (d) Aresta removida.

Figura 3.18: Propagação de regiões de *trimming* nos extremos das curvas.

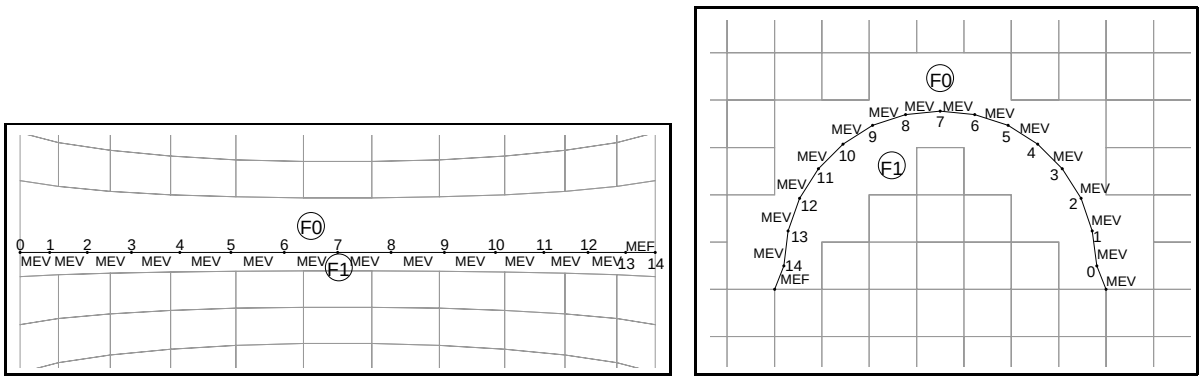
ceptadas, como é apresentado no exemplo de inserção das arestas de *trimming* da Figura 3.20 na próxima seção.

Observa-se ainda que os extremos das curvas de *trimming* devem ser testados quanto à proximidade relativa à face que ela originou, pois situações de arestas ou vértices muito próximos podem ocorrer, mesmo não estando conectados a arestas interceptadas. A Figura 3.18a ilustra uma situação de um vértice próximo ao extremo inferior esquerdo de uma curva de *trimming*. A Figura 3.18b mostra a região de *trimming* expandida pela eliminação do vértice marcado.

A outra situação prevista ocorre quando a entidade muito próxima ao extremo de uma curva de *trimming* é uma aresta. A Figura 3.18c mostra a aresta próxima desenhada com uma espessura maior. Na Figura 3.18d, a região de *trimming* foi expandida pela eliminação desta aresta.

3.6.2 Inserção das arestas de *trimming*

No Passo 3b, as arestas que representam as curvas de *trimming* são inseridas, dividindo as faces de *trimming* em duas, uma localizada à esquerda e uma à direita de cada curva, pela conexão dos pontos uniformemente espaçados amostrados sobre estas curvas. As arestas criadas sobre estas curvas recebem marcas positivas nos campos *trimming*, mostrados na Figura 3.5. Cada extremo da curva pode ou não pertencer a uma aresta de fronteira. A Figura 3.19a mostra os operadores de Euler [58] usados para inserir as arestas que representarão a curva de interseção, subdividindo a face de *trimming* mostrada na Figura 3.17c. Os pontos extremos (0 e 14) foram inseridos durante o processo de remoção e subdivisão de arestas no Passo 3a. Os segmentos seguintes da curva são inseridos com o operador *MEV*, exceto o último (ponto 13 ao 14) que é criado com o operador



(a) Casca cilíndrica.

(b) Plano.

Figura 3.19: Operadores usados para inserção das arestas de *trimming* nas faces da Figura 3.17.

MEF, que divide a face em duas (faces $F0$ e $F1$ na Figura 3.19a).

A Figura 3.19b mostra como a face de *trimming* do retalho planar é subdividida, ilustrando como os pontos extremos são conectados à face de *trimming*. Uma chamada inicial ao operador *MEV* conecta o primeiro ponto da curva de *trimming* (0) ao vértice mais próximo na face de *trimming*; seguem-se sequências de chamadas ao operador *MEV*, inserindo-se todos os segmentos da curva de *trimming*. Por fim, uma chamada ao operador *MEF* conecta o último ponto (14) ao vértice mais próximo na face de *trimming*, completando a subdivisão da face com a criação de uma nova ($F1$).

Para tratar os casos de curvas de *trimming* fechadas, no Passo 3a estas curvas são identificadas e, ao invés de se remover as arestas correspondentes à primeira posição nas duas superfícies, faz-se a inserção destas posições por subdivisão destas arestas, da mesma forma que para as arestas de contorno e de *trimming*. Este procedimento evita a criação de faces com dois ciclos, configuração que não é suportada pela DCEL. A posição central da curva também é conectada a ambos os lados da face de *trimming* de forma a criar quatro faces de *trimming* adjacentes em cada superfície. Esta segunda divisão desacopla a triangulação das duas faces adjacentes à aresta da primeira posição da curva de *trimming*, previamente subdividida. A Figura 3.20 mostra um exemplo de uma curva de *trimming* fechada sobre uma casca cilíndrica. Os operadores topológicos usados para inserir a curva de *trimming* e criar as faces $F0$ a $F4$ são também mostrados na figura.

3.6.3 Triangulação das faces de trimming

No Passo 3c, as regiões de *trimming* são trianguladas de forma a completar a reconstrução das malhas. Um algoritmo de *ear-cut* [67] subdivide recursivamente cada polígono simples, que são

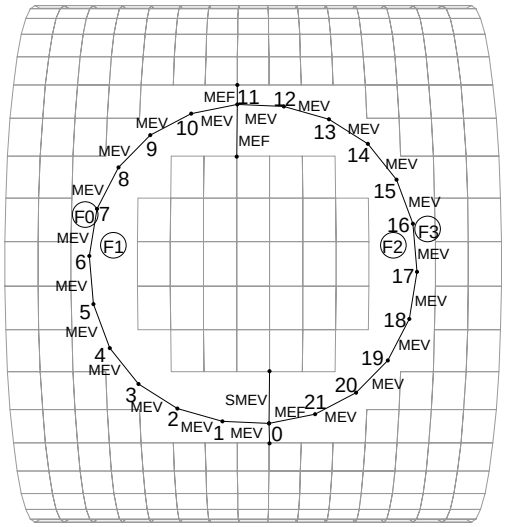


Figura 3.20: Operadores usados para inserir as arestas de uma curva de *trimming* fechada.

as faces de *trimming*. Os critérios usados no espaço paramétrico para obter a triangulação inicial são:

1. **Consistência:** As arestas que estão totalmente contidas na região paramétrica limitada pelas arestas, sem apresentar interseções com outras; e

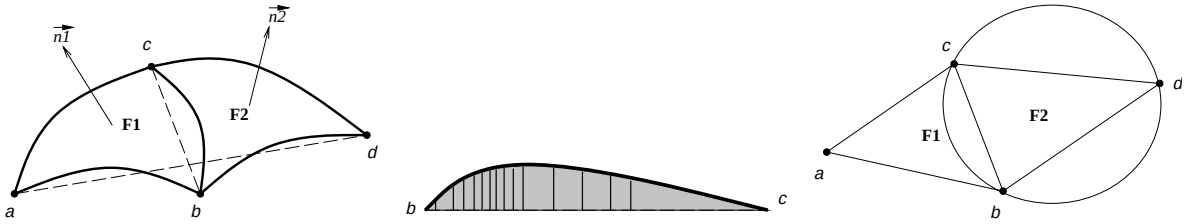
2. **Proximidade:** Arestas que conectam os dois vértices mais próximos da face de *trimming*.

O segundo estágio do algoritmo de triangulação aumenta a qualidade dos triângulos gerados fazendo troca de arestas em triângulos adjacentes (*edge-swap*). Os critérios geométricos utilizados para as trocas de arestas entre dois triângulos adjacentes (arestas *ad* e *bc* nos triângulos **F1** e **F2** da Figura 3.21a, por exemplo) são os seguintes:

1. **Curvatura:** A área do setor formado pela corda e pela curva que segue a superfície em 3D ao longo da aresta é a menor.

2. **Delaunay:** Arestas que obedecem ao critério padrão de Delaunay projetado em um plano cujo vetor normal é uma média dos vetores normais das faces adjacentes.

O critério de *Curvatura* pode ser feito com a adição de uma poligonal adaptativa à curvatura da superfície, construída para representar a aresta em 3D. A área do setor é aproximada pelos trapézios formados pela projeção dos pontos na corda, como mostra a Figura 3.21b. O critério de *Delaunay* é feito em um plano cujo vetor normal é determinado pela média dos dois vetores normais avaliados nos pontos centrais dos triângulos adjacentes, com mostra a Figura 3.21a.



(a) Faces adjacentes. (b) Critério de *Curvatura*. (c) Critério de *Delaunay*.

Figura 3.21: Critérios geométricos para avaliação de arestas.

As duas faces adjacentes são projetadas neste plano para calcular os ângulos nos vértices a e d, mostrados na Figura 3.21c.

O algoritmo de triangulação usa estes critérios na ordem em que eles foram apresentados pelas seguintes razões:

1. As regiões de *trimming* podem apresentar curvaturas acentuadas, onde critérios planares, como o de *Delaunay* por exemplo, podem não conduzir a bons resultados.

2. O critério de *Curvatura* pode não ser suficiente, como é o caso das superfícies planares.

3. Regiões de *trimming* em superfícies planares subdivididas por arestas que obedecem apenas aos critérios de *Consistência* e *Proximidade* podem ser transformadas em uma triangulação de Delaunay com restrições por um algoritmo que faz trocas de arestas [67] usando o critério de *Delaunay*.

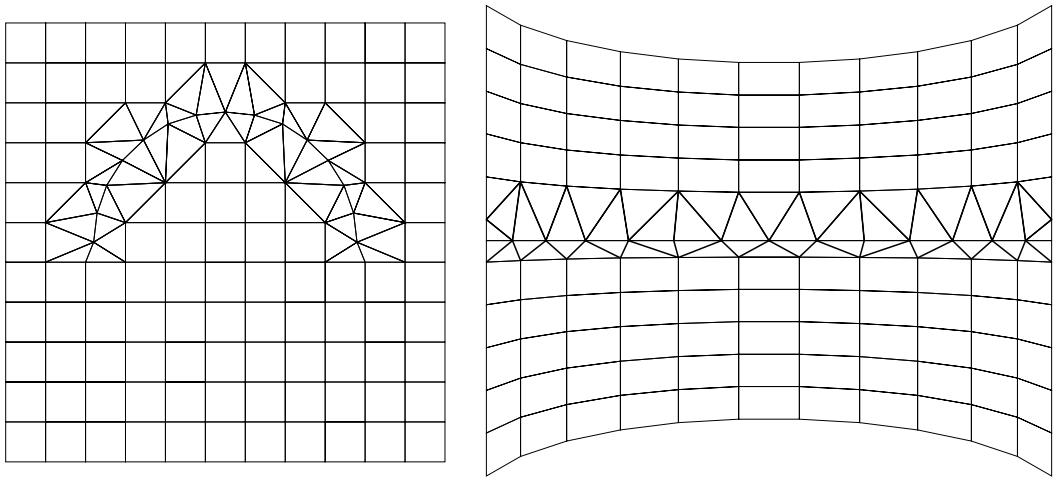
4. Em malhas de elementos finitos sobre superfícies com curvaturas acentuadas, o autor estima que a melhor modelagem é aquela em que as arestas privilegiam o critério de *Curvatura* com relação ao de *Delaunay*.

Por estas razões, os passos básicos do algoritmo de triangulação de uma face de *trimming* são:

1. Obter uma triangulação inicial empregando os critérios de *Consistência* e de *Proximidade* por meio de subdivisões recursivas da face de *trimming*.

2. Fazer trocas de arestas com as arestas criadas no Passo 1 de acordo com o critério de *Curvatura*. Se as curvaturas nas duas arestas forem muito próximas, então o critério de *Delaunay* é usado para definir o problema.

A Figura 3.22 mostra as faces da Figura 3.17 trianguladas pelo algoritmo apresentado. As arestas criadas para subdividir as faces de *trimming* são desenhadas com linhas mais grossas.



(a) No plano. (b) Na casca cilíndrica.

Figura 3.22: Triangulação das regiões de *trimming*.

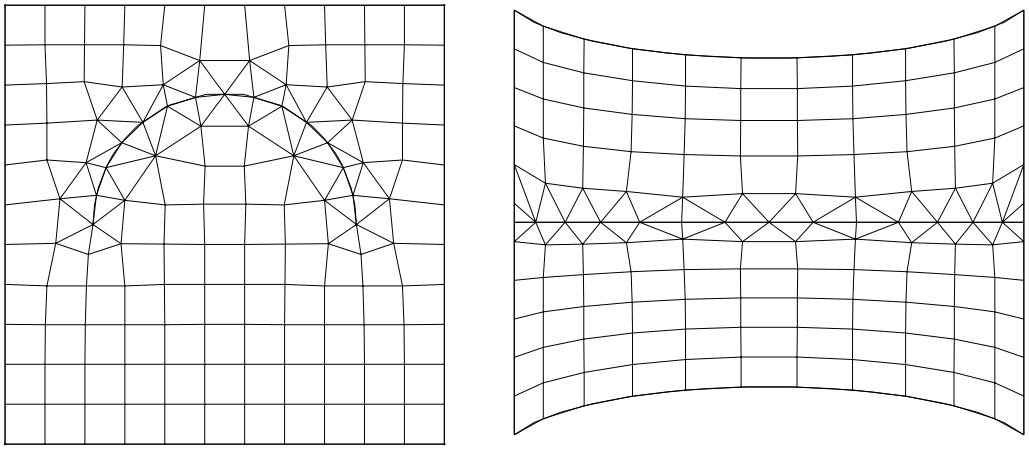
3.6.4 Suavização

A técnica de reconstrução mostrada nas seções anteriores produz triângulos em ambas as malhas com boa qualidade, como mostra a Figura 3.22. Estas malhas podem, entretanto, ser suavizadas de forma a melhorar e uniformizar a qualidade dos elementos gerados.

A técnica Laplaciana padrão de suavização em duas dimensões [88] pode ser aplicada individualmente aos vértices *internos* (que não pertencem à fronteira ou a uma curva de *trimming*) das superfícies, usando-se a média das coordenadas paramétricas dos vértices. Esta técnica pode não conduzir a bons resultados pois não leva em conta a distorção que os elementos apresentam em 3D. Se as coordenadas cartesianas tridimensionais dos vértices forem usadas de maneira a considerar a distorção tridimensional, deve-se fazer a recondução destes vértices para as superfícies originais, pois, invariavelmente, estes vértices são movidos para fora destas superfícies.

A determinação dos vértices adjacentes a um vértice qualquer na superfície é facilmente obtida com a estrutura de dados topológica DCEL. O mesmo algoritmo usado no Passo 2 (Seção 3.5.5) reprojeta os vértices em cada uma das superfícies e define as novas coordenadas com poucas iterações. A suavização global de todos os vértices da malha pode, entretanto, ter alto custo computacional, se comparada com os outros passos do algoritmo. Além disso, os vértices distantes das regiões de interseção não precisam ser movidos, pois os elementos criados pelas técnicas de mapeamento em domínios elementares geram, na maioria dos casos, elementos de boa qualidade.

Por estas razões, adotou-se a metodologia descrita por Rypl e Krysl [75], onde é fei-



(a) No plano. (b) Na casca cilíndrica.

Figura 3.23: Suavização das regiões de *trimming*.

ta uma pré-suavização puramente paramétrica antes de se fazer a média em 3D. Além disso, a movimentação tridimensional é significativa apenas para os vértices adjacentes às regiões de *trimming*, sendo que os demais permanecem quase que inalterados. Esta técnica torna compatíveis os tempos de processamento relativos entre o algoritmo de suavização e os outros passos, e apresenta malhas com boa qualidade como mostram os resultados apresentados no capítulo seguinte. Foram adotadas apenas três repetições da pré-suavização no espaço paramétrico e duas movimentações locais dos vértices das faces de *trimming* em 3D. A Figura 3.23 mostra as malhas da Figura 3.22 após a aplicação da técnica de suavização descrita.

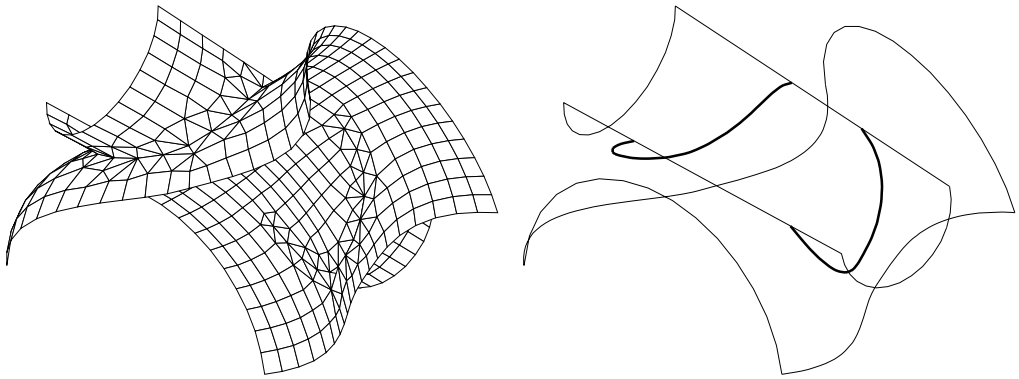
Capítulo 4

Exemplos

Para testar as técnicas apresentadas nos capítulos anteriores, apresentam-se neste capítulo alguns exemplos de modelagens com superfícies que se interceptam. Os exemplos foram criados com o MG, que usa as bibliotecas G3D [17] e CD [16] para gerar as figuras em PostScript.

4.1 Pares de superfícies

A seguir, mostram-se alguns exemplos de interseções entre pares de superfícies, cujas malhas foram reconstruídas pelo algoritmo proposto. O primeiro exemplo já foi mostrado na Figura 3.1, é reapresentado na Figura 4.1a com uma vista da parte inferior da malha, e representa a união de uma superfície bilinear de Coons com um cilindro, cujo fundo foi removido após a separação feita pelo algoritmo. A Figura 4.1b mostra as curvas que modelam as superfícies primárias e as duas curvas de interseção simétricas, desenhadas com espessura maior.



(a) Malha final. (b) Curvas.

Figura 4.1: Interseção entre casca cilíndrica e superfície de Coons bilinear.

4.1.1 Cascas cilíndricas

O próximo exemplo consiste da união de duas cascas cilíndricas, modeladas como superfície bilineares com as curvas mostradas na Figura 4.2. As Figuras 4.3 e 4.4 mostra os resultados do algoritmo para malhas com discretização crescente. Nas Figuras 4.3a e 4.3b, a discretização é pequena o suficiente para que as regiões de interseção se estendam até as fronteiras, de forma que a suavização reposiciona apenas alguns vértices internos. As Figuras 4.4a e 4.4b mostram que, em malhas com resoluções maiores, apenas alguns elementos próximos à região de interseção são afetados, e a maioria dos elementos das malhas originais permanecem inalterados. Este fato também pode ser comprovado na Figura 3.1.

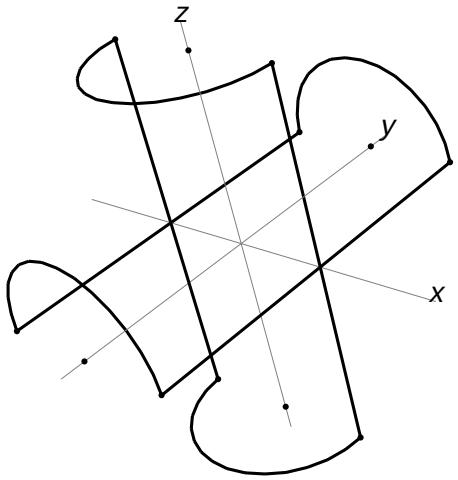
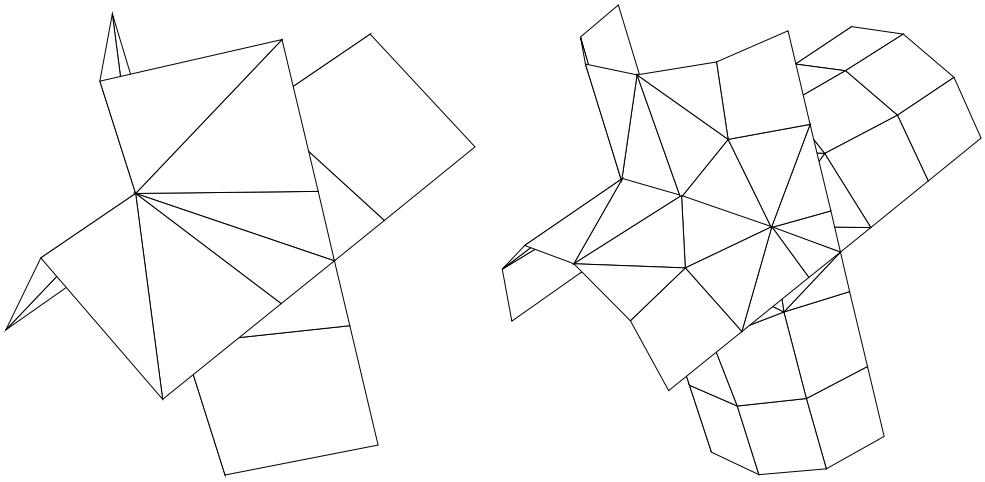


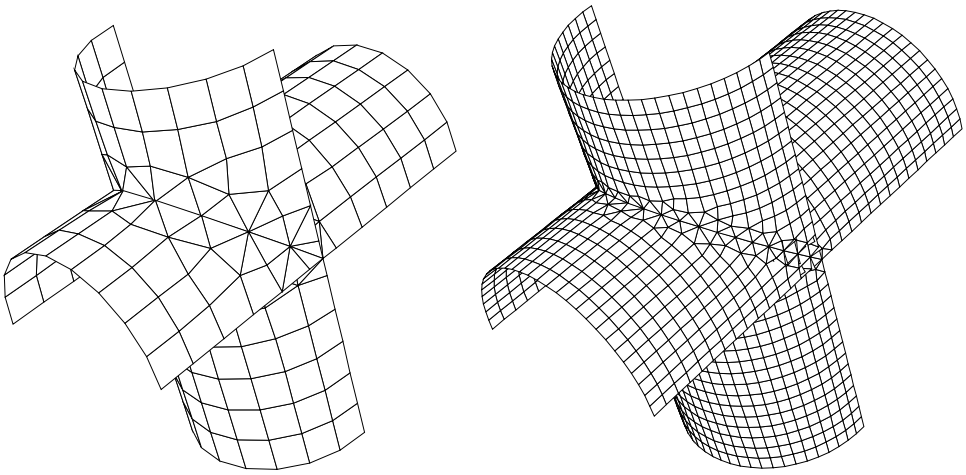
Figura 4.2: Curvas de fronteira das superfícies das Figuras 4.3 e 4.4.



(a) 2×3.

(b) 5×5.

Figura 4.3: Cascas cilíndricas com baixa resolução.



(a) 10×11. (b) 30×30.

Figura 4.4: Cascas cilíndricas com resoluções maiores.

4.1.2 Cilindro reto e curvo

A Figura 4.5 mostra a interseção das duas superfícies básicas usadas na modelagem do permutador de calor mostrado na Seção 4.2.1, como exemplo de modelagem composta. Este exemplo apresenta alguns pontos de interseção em que a matriz jacobiana é negativa (vide Seção 3.4.2), e que o uso das pseudo-inversas de Deuflard [34] foi determinante para a solução.

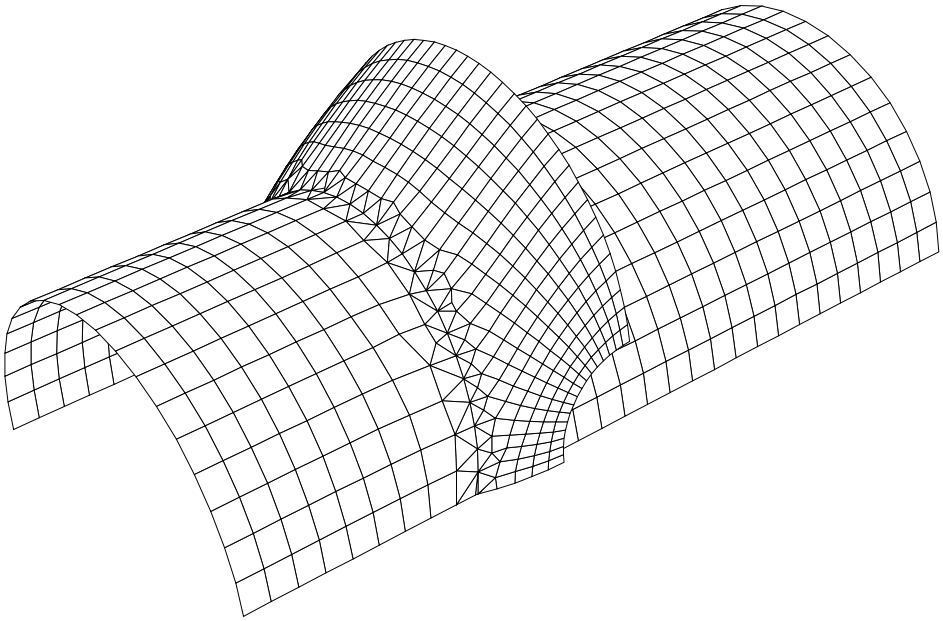


Figura 4.5: Interseção entre cilindros reto e curvo.

4.1.3 Componentes conexos

A Figura 4.6 mostra um exemplo de interseção em que dois componentes conexos são definidos pelas interseções. Os retalhos são uma casca cilíndrica e uma casca toroidal. A Figura 4.7 mostra um detalhe. Neste exemplo, três regiões distintas são identificadas em cada retalho. As Figuras 4.8a e 4.8b mostram os elementos no espaço paramétrico, com as regiões distintas desenhadas com diferentes espessuras de linha. Note que os elementos não apresentam bom aspecto no espaço paramétrico como acontece no espaço do objeto. A Figura 4.9 mostra as malhas após a remoção da região central da casca cilíndrica.

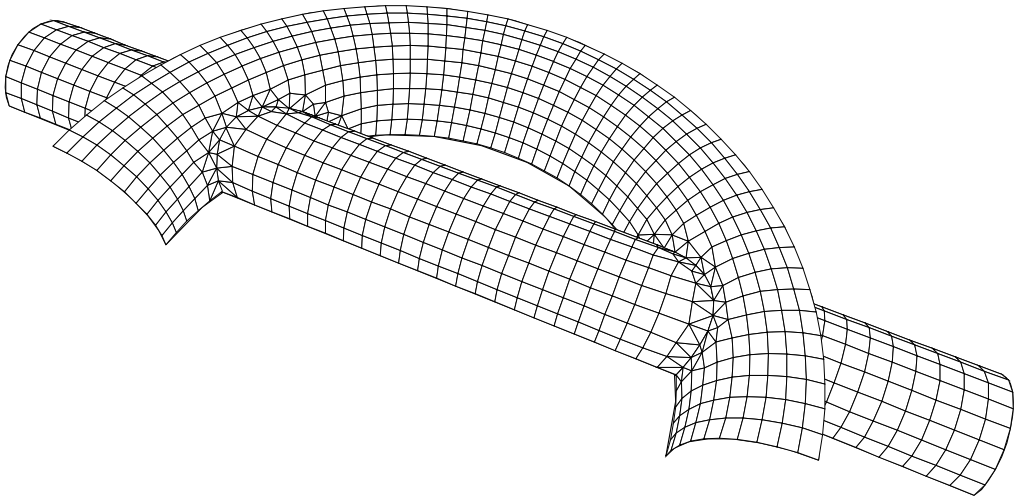


Figura 4.6: Interseção com dois componentes conexos.

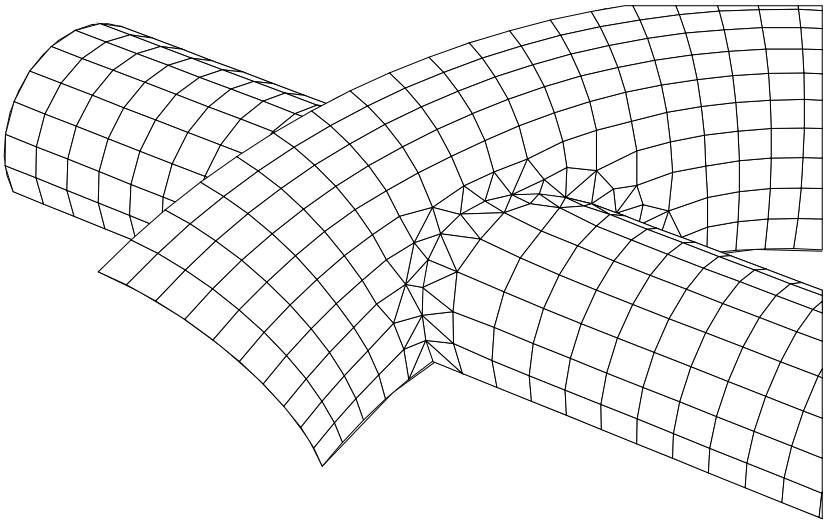


Figura 4.7: Detalhe da interseção da Figura 4.6.

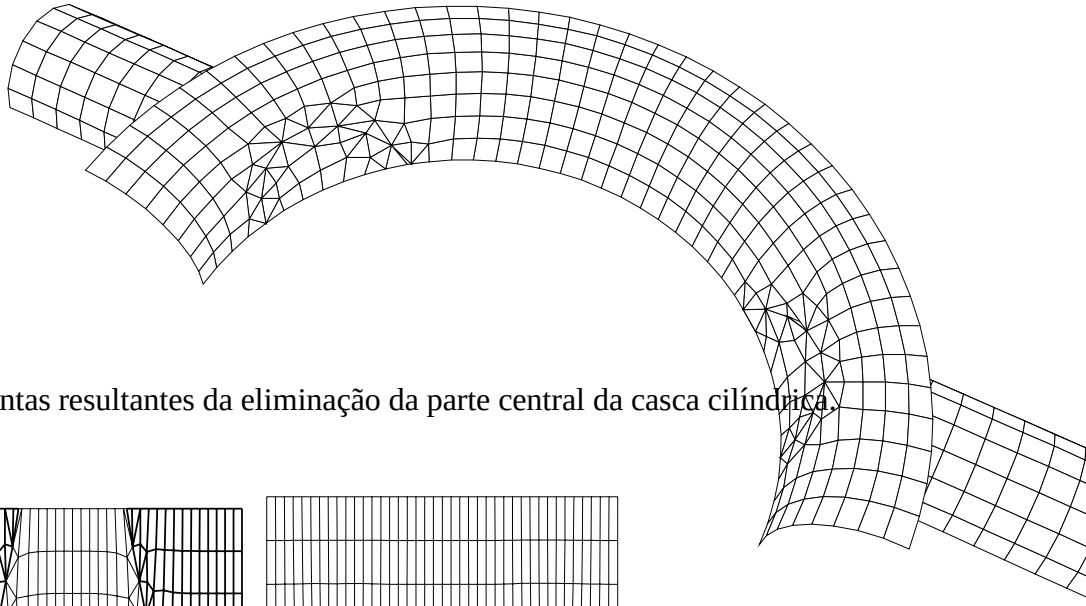
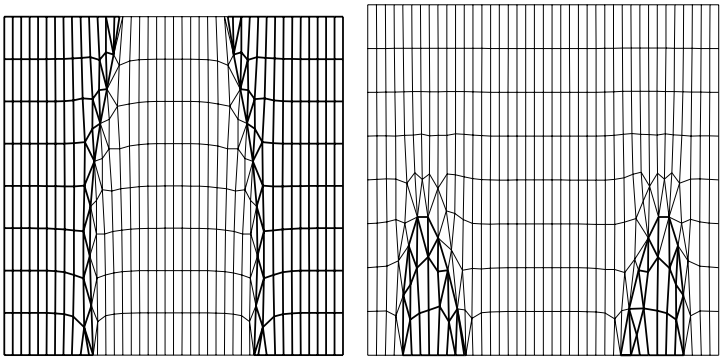


Figura 4.9: Regiões distintas resultantes da eliminação da parte central da casca cilíndrica.



(a) Casca cilíndrica. (b) Casca toroidal.

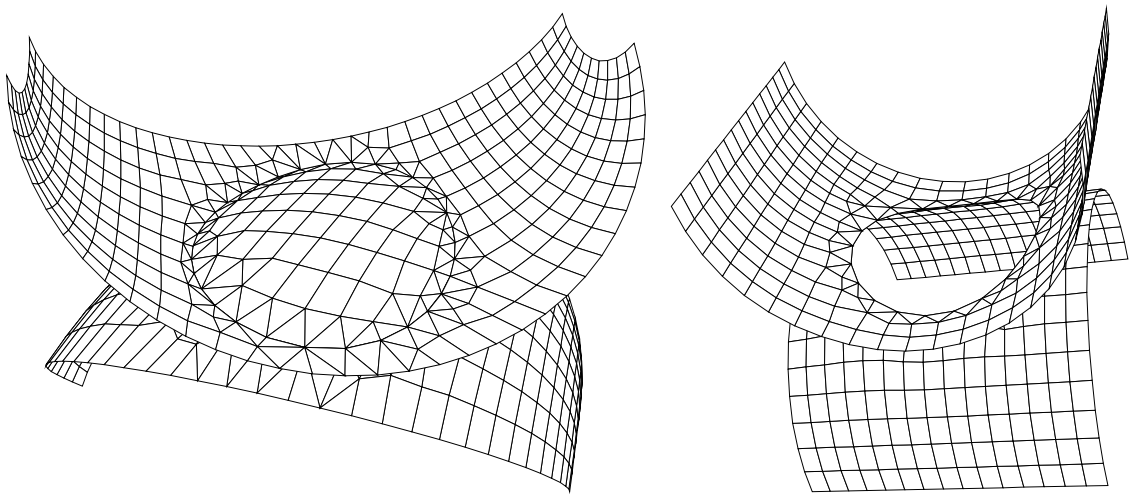
Figura 4.8: Espaços paramétricos das cascas cilíndrica e toroidal.

4.1.4 Curvas fechadas

Os exemplos mostrados nas Figuras 4.10a e 4.10b ilustram casos de reconstruções feitas sobre curvas de interseção fechadas. O exemplo da Figura 4.10a é formado por superfícies onde as quatro curvas de bordo são arcos de círculo. No exemplo mostrado na Figura 4.10b, as regiões centrais em ambos os cilindros foram removidas.

4.1.5 Superfícies com cantos

Superfícies com curvaturas acentuadas ou com tangentes discontínuas, tais como superfícies poliedrais, são obstáculos potenciais para a convergência dos procedimentos numéricos descritos na Seção 3.4. As Figuras 4.11 e 4.12 mostra que o algoritmo consegue tratar corretamente tais superfícies, e que as triangulações possuem aspecto satisfatório tanto na malha com cantos (Figu-



(a) Retalhos de Coons com 4 arcos. (b) Duas cascas cilíndricas.

Figura 4.10: Exemplos de interseções que geram curvas fechadas.

ra 4.12a) quanto na superfície planar (Figura 4.12b). As superfícies com cantos somente podem ser tratadas devido ao uso de poligonais adaptativas para o cálculo das tangentes nas curvas de bordo das superfícies. Os problemas de otimização que podem ser associados ao cálculo de interseções entre superfícies com cantos são de difícil solução [34].

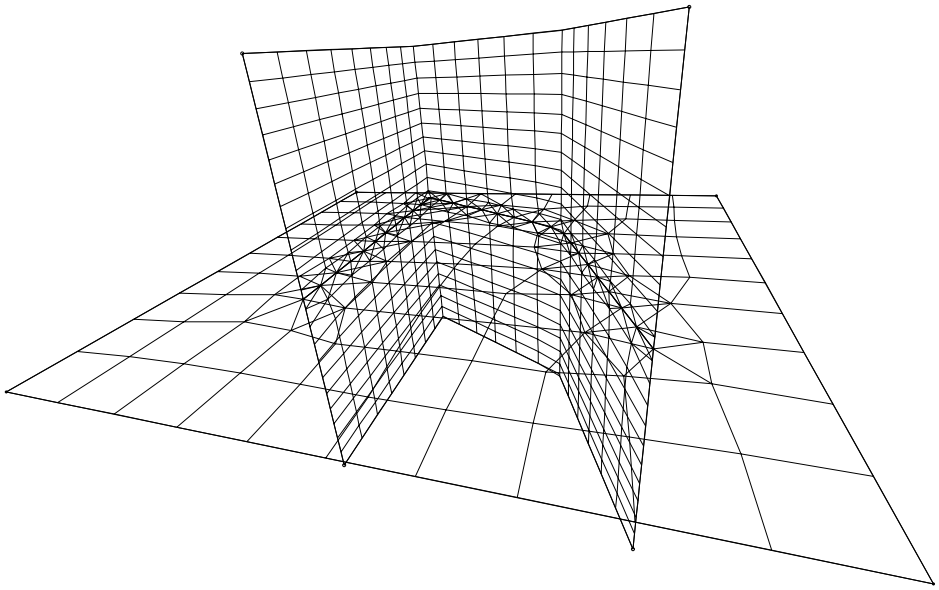
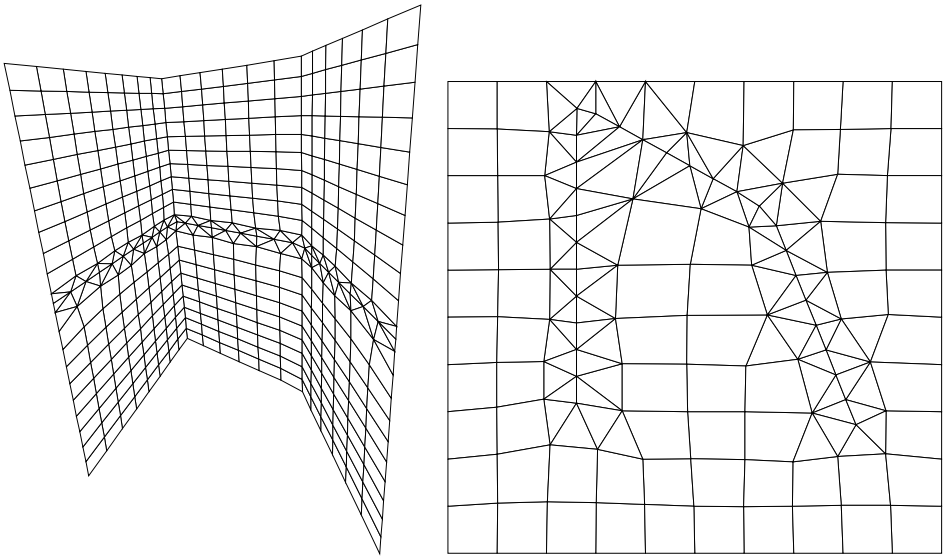


Figura 4.11: Malha de superfície planar com cantos composta pela interseção.



(a) Superfície com cantos. (b) Superfície planar.

Figura 4.12: Interseção entre plano e superfície com cantos.

4.1.6 Qualidade das malhas

Um estudo da qualidade dos elementos gerados sobre as malhas de exemplos das seções anteriores foi feito usando-se as metodologias de medição de triângulos em superfícies mostradas por Lau e Lo [51]. O medidor de qualidade α de um triângulo ABC usado por Lau e Lo é:

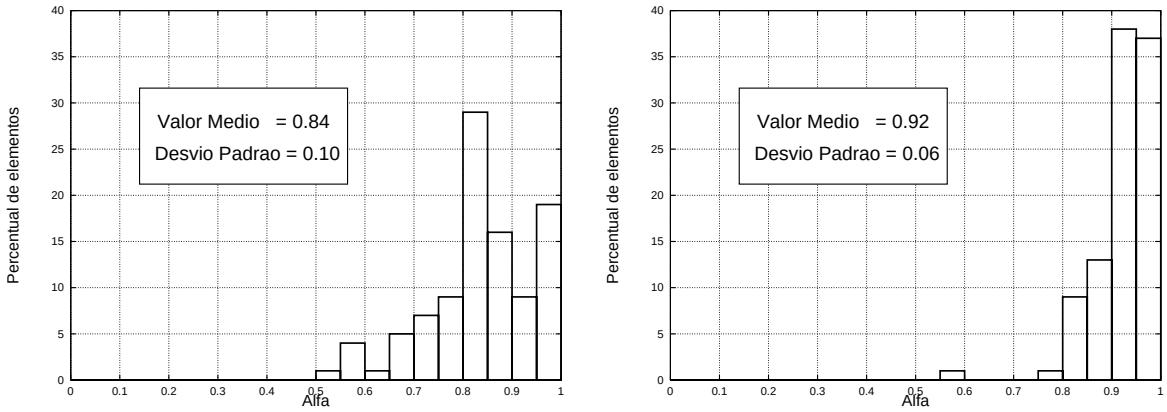
$$\alpha(\Delta ABC) = 2\sqrt{3}\frac{\|AB \times AC\|}{\|AB\|^2 + \|BC\|^2 + \|CA\|^2}. \tag{4.1}$$

O valor ótimo de α é 1.0, obtido para triângulos equiláteros. Se os três pontos do triângulo ABC forem colineares, então α vale 0.0. A Tabela 4.1 apresenta várias medições do parâmetro α ($\alpha_{méd}$, $\alpha_{máx}$ e $\alpha_{mín}$) definido pela Equação 4.1, além dos valores do desvio padrão (σ) antes e depois da suavização. Além da referência para as figuras, a Tabela 4.1 também apresenta o número total de elementos nas duas malhas, o número de triângulos gerados pela reconstrução, os tempos de processamento, obtidos em um Pentium-Pro de 200 MHz com 128 Mb de RAM rodando Linux (Kernel 2.0.34).

Para ilustrar a alteração da qualidade dos elementos triangulares promovida pelo algoritmo de suavização (vide Seção 3.6.4), a Figura 4.13a mostra o histograma relativo aos valores α de cada um dos triângulos gerados pela triangulação inicial na reconstrução das malhas sobre as superfícies cilíndricas da Figura 4.4b. A Figura 4.13b mostra o histograma avaliado após a aplicação do algoritmo de suavização. Estes histogramas foram construídos com intervalos de 20 elementos de um total de 300.

Figura	#Elem.	#Δ	Tempo (s)	Antes de *smooth*				Após *smooth*			
				$\alpha_{méd}$	$\alpha_{máx}$	$\alpha_{mín}$	σ	$\alpha_{méd}$	$\alpha_{máx}$	$\alpha_{mín}$	σ
3.1	931	260	25.36	0.81	1.00	0.20	0.15	0.90	1.00	0.55	0.09
4.3a	36	20	0.45	0.84	0.90	0.54	0.10	0.80	0.98	0.59	0.06
4.3b	50	52	0.68	0.84	1.00	0.54	0.10	0.90	1.00	0.73	0.06
4.4a	220	96	2.18	0.84	1.00	0.54	0.10	0.93	1.00	0.60	0.06
4.4b	1800	300	6.28	0.84	1.00	0.54	0.10	0.92	1.00	0.57	0.06
4.5	1550	274	6.19	0.80	1.00	0.18	0.17	0.90	1.00	0.21	0.10
4.6	728	170	4.48	0.77	1.00	0.01	0.23	0.91	1.00	0.10	0.18
4.10a	734	216	3.72	0.78	1.00	0.31	0.15	0.85	1.00	0.16	0.13
4.10b	952	284	6.45	0.76	1.00	0.16	0.20	0.79	1.00	0.43	0.12

Tabela 4.1: Qualidade e tempos obtidos para os exemplos das seções anteriores.



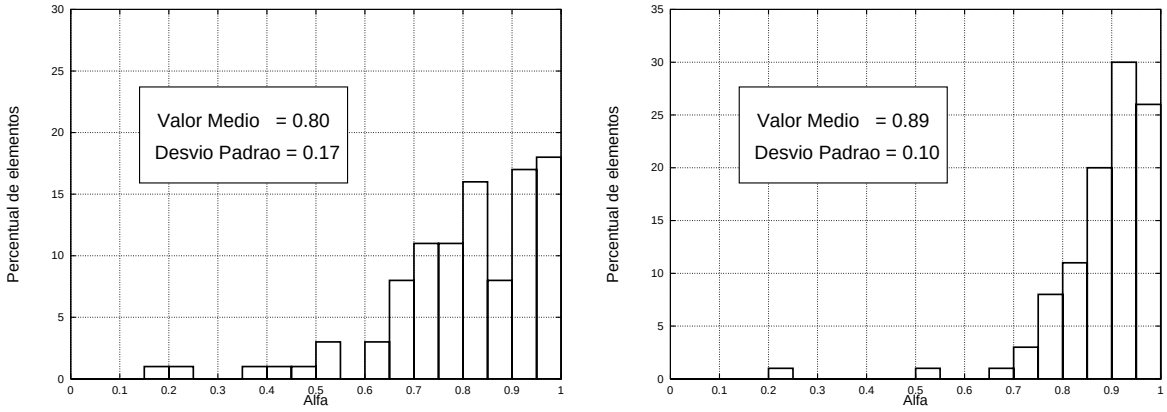
(a) Antes da suavização.

(b) Após a suavização.

Figura 4.13: Histograma das malhas da Figura 4.2d.

As Figuras 4.14a e 4.14b mostram os histogramas referentes à malha sobre os cilindros mostrados na Figura 4.5. Os histogramas da Figura 4.14 foram gerados com o mesmo agrupamento de elementos usado na Figura 4.13.

Em ambos os processos de suavização mostrados nas Figuras 4.13 e 4.14, observa-se a diminuição da dispersão e o aumento da qualidade média das malhas, embora ainda exista uma pequena percentagem de elementos com baixa qualidade. Os elementos de baixa qualidade que persistem após a atuação do algoritmo de suavização possuem vértices situados sobre curvas de bordo e de interseção, como pode ser obervado nas Figuras 4.5 e 4.9.



(a) Antes da suavização.

(b) Após a suavização.

Figura 4.14: Histograma das malhas da Figura 4.3.

4.2 Modelagens compostas

Com o MG podem ser feitas construções de cascas compostas usando-se curvas espaciais definidas pelo usuário, onde são geradas malhas com os mapeamentos primários, e redefinições com o algoritmo de interseção.

4.2.1 Permutador de calor

A Figura 4.5 mostra a interseção básica usada para a construção de um modelo de permutador de calor usado em refinarias de petróleo. A Figura 4.15 mostra a malha final da superfície combinada do permutador. A Figura 4.16 mostra as curvas geradoras de todas as superfícies individuais. Nesta figura, as curvas de interseção são desenhadas com uma espessura maior do que as curvas modeladas pelo usuário.

A peculiaridade deste exemplo está na remoção das regiões internas dos cilindros reto e curvo, identificadas pelo algoritmo de reconstrução. Todos os outros retalhos foram gerados com as curvas modeladas pelo usuário.

4.2.2 Plataforma semi-submersível

Um outro exemplo de modelagem composta é mostrado na Figura 4.17a, onde um modelo para análise de flutuação da plataforma semi-submersível P-XXVII da Petrobrás foi construído com a metodologia de geração de mapeamentos primários, seguida de recortes das regiões excedentes. Este modelo é uma representação da porção submersa para uma determinada posição do plano

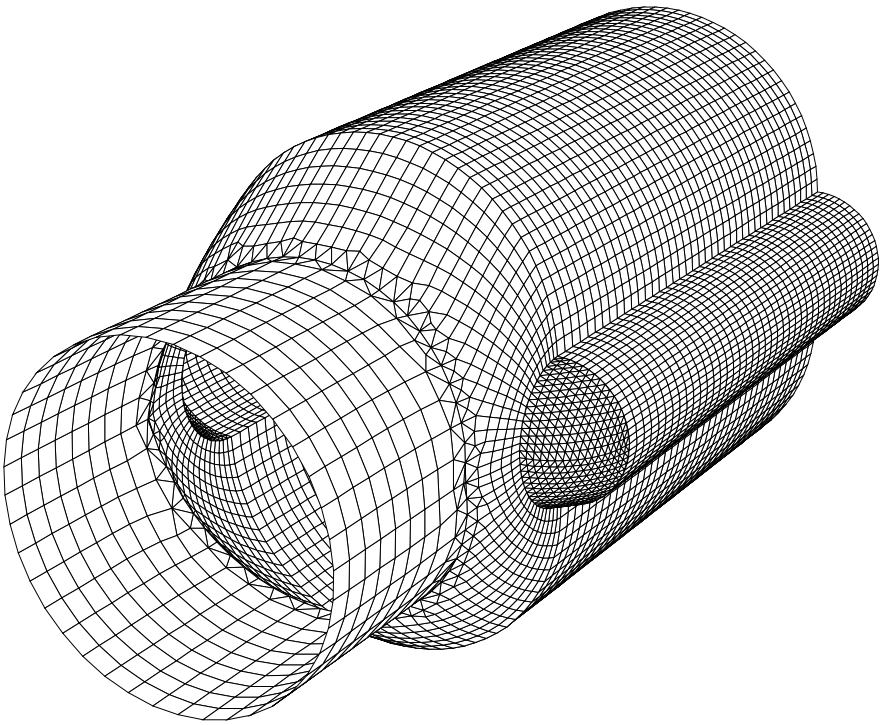


Figura 4.15: Malha final do permutador.

da linha d'água, e apresenta dupla simetria com relação aos planos XZ e YZ [1], de forma que apenas um quarto da casca completa foi construída.

O contraventamento tubular horizontal foi modelado com duas superfícies de Coons bilineares justapostas, contendo dois arcos de círculo e duas retas longitudinais. Este contraventamento é a superfície-chave do modelo pois apresenta interseções com a coluna central, que é um retalho de um tronco de cone, com o plano de topo do flutuador, que também apresenta interseções com as duas colunas verticais, com a superfície lateral do flutuador, e ainda com a superfície fictícia que representa o plano XZ de simetria. A coluna interceptada pelo contraventamento foi mostrada na Figura 3.3 para exemplificar o amassamento provocado pela reparametrização com as curvas de interseção. Nas Figuras 4.17b e 4.17c mostram-se os detalhes da malha reconstruída sobre esta coluna, mantendo-se a parametrização inicial.

As interseções foram calculadas com o algoritmo mostrado no Capítulo 3 para cada par de superfícies individualmente, e os vértices criados nas fronteiras das superfícies são considerados em novos cálculos de interseção. A Figura 4.18 mostra todas as curvas usadas para a completa modelagem da plataforma. As curvas de interseção são mostradas com espessura maior do que as curvas criadas interativamente. As regiões distintas na coluna central, no plano de topo do flutuador, na superfície lateral do flutuador, e no contraventamento horizontal, apenas puderam

[1] O eixo X é longitudinal e o eixo Z aponta para a superfície.

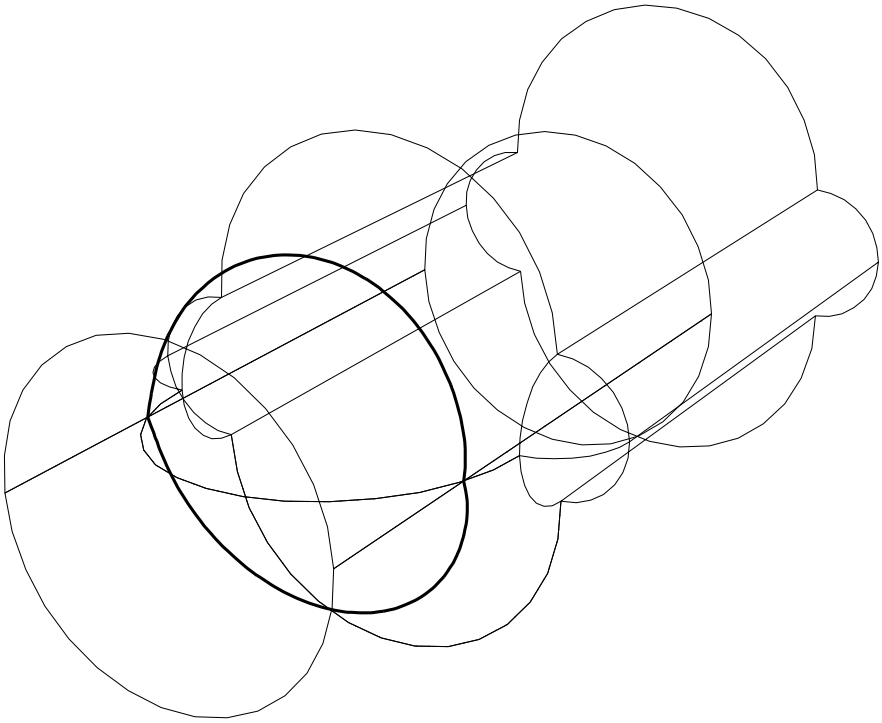
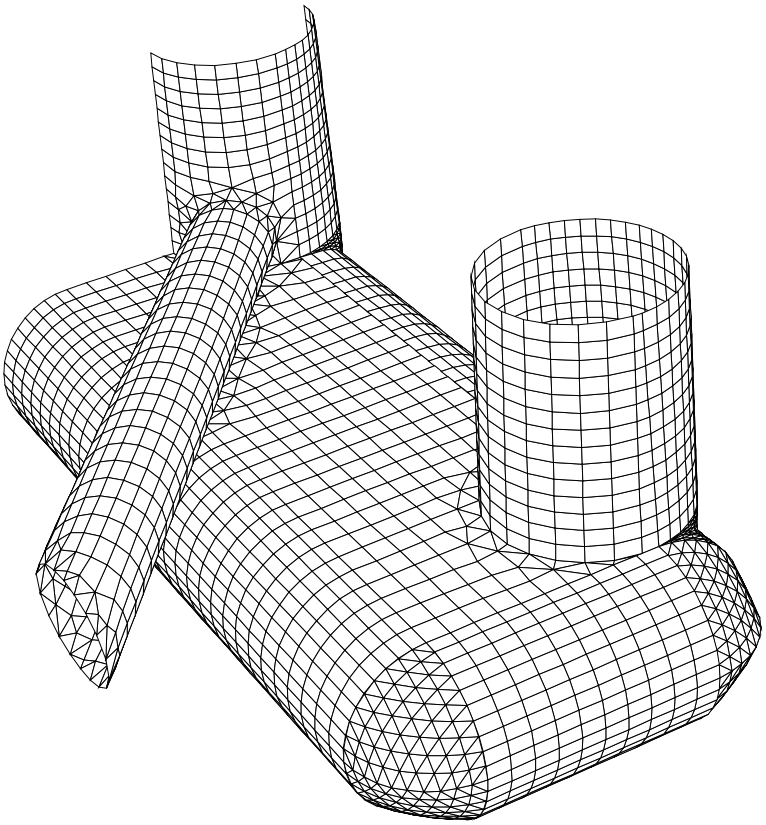


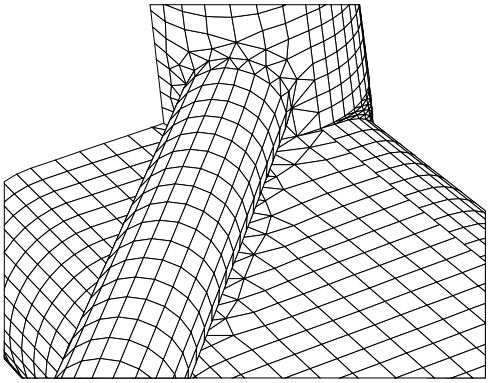
Figura 4.16: Curvas usadas na modelagem do permutador.

ser recortadas após todos os cálculos de interseções, pois são isoladas por mais de uma curva de *trimming*.

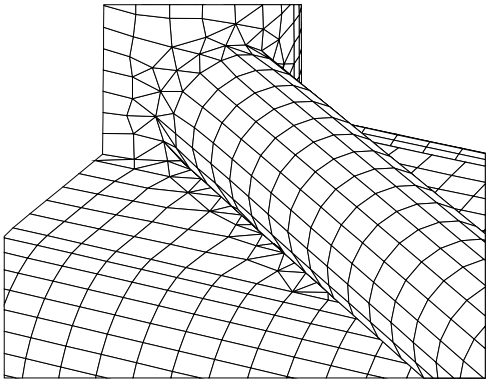
De forma a recortar as superfícies do contraventamento no plano XZ de simetria, construiu-se um retalho plano apenas para retirar as regiões situadas do outro lado deste plano. A Figura 4.19 mostra um detalhe das curvas usadas na modelagem do retalho sobre o plano XZ além da malha modificada na região do contraventamento.



(a) Vista completa do modelo.



(b) Detalhe do contraventamento.



(c) Outro lado do detalhe.

Figura 4.17: Modelo de plataforma semi-submersível.

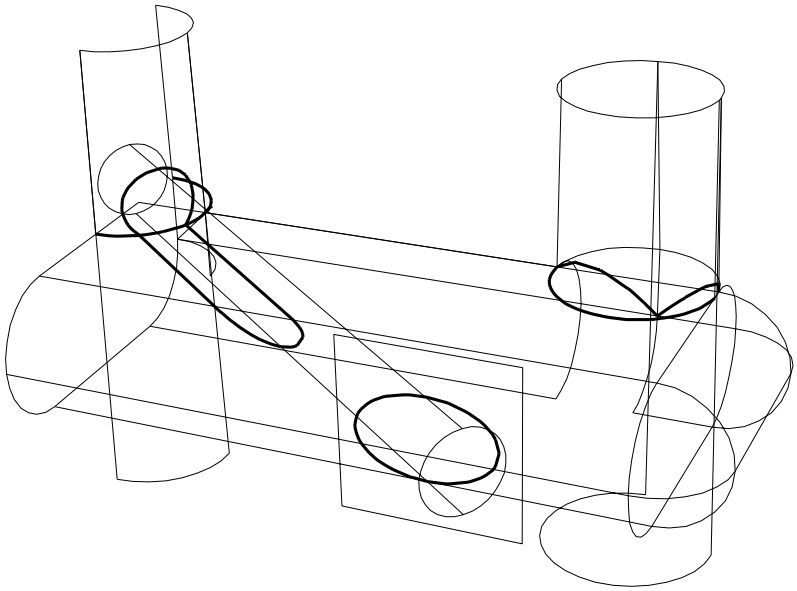


Figura 4.18: Curvas geradas para a modelagem da plataforma semi-submersível.

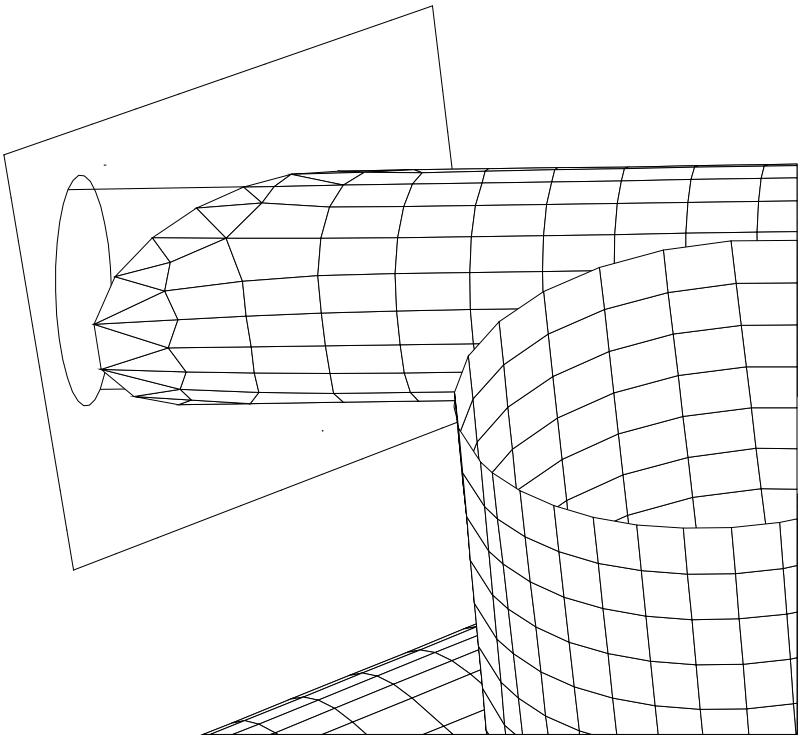


Figura 4.19: Curvas envolvidas no recorte feito no plano XZ.

Capítulo 5

Conclusões

Uma técnica de modelagem que redefine automaticamente malhas geradas sobre superfícies paramétricas apoiada por um algoritmo que determina as curvas de interseção foi desenvolvida e implementada. O modelador MG usa um conjunto de técnicas usuais de representação geométrica de curvas e superfícies, apoiado por uma interface com o paradigma de manipulação direta com o plano móvel de interface [22] e pelo novo algoritmo de interseções entre malhas com representações paramétricas para a geração de modelos de casca para análises por elementos finitos ou de contorno. Esta técnica pode ser classificada como *CSG* de malhas, pois as parametrizações iniciais são usadas ao longo de todo o processo de modelagem. Esta metodologia se mostra bastante produtiva pois reduz o esforço de modelagem à definição das curvas básicas que formam o modelo.

Pode-se caracterizar a técnica de modelagem usada segundo os critérios conhecidos de avaliação [73]:

- **Validade:** A malha única de elementos que une todos os retalhos individualmente gerados é válida, em cada estágio de modelagem, desde que os retalhos não apresentem auto-interseções, que não são tratadas neste trabalho.

- **Unicidade:** Não existe ambiguidade quanto ao modelo de malhas criado pois todos os elementos possuem orientação bem definida, identificada pela visualização do sistema de cores e símbolos adotados na interface.

- **Poder de expressão:** Uma variedade grande de modelos podem ser construídos com as metodologias propostas, como pode ser observado pelos exemplos apresentados. O *sweep* genérico com duas curvas apresentado na Seção 2.2.2, apesar de possuir formulação simples, pode ser usado para gerar variados tipos pertencentes a esta classe de superfícies, entre elas os *sweeps* de translação e de rotação com passos não uniformemente espaçados.

A manipulação direta dos pontos de controle das curvas do tipo B-splines cúbicas interpolantes permite a criação de variadas formas, capazes de representar com precisão os modelos de navios e plataformas semi-submersíveis que o CENPES tem projetado. O uso de NURBS apenas unificaria a representação interna das curvas e superfícies, não sendo portanto um limitador quanto aos tipos de superfície utilizados.

- **Operações booleanas:** O uso da estrutura de dados topológica sobre os retalhos permite que o usuário escolha por simples apontamento as regiões que compõem a malha final. Esta identificação é conseguida por um algoritmo de interseção e *trimming* de malhas que contém idéias inovadoras, constituindo a contribuição central desta tese.

- **Representação:** A representação por fronteiras usadas com o uso das poligonais adaptativas com a curvatura e da estrutura DCEL sobre as superfícies facilitam tanto os algoritmos de *rendering* quanto a integração de todos os retalhos na malha única final.

Uma estrutura de dados topológica usada para representar subdivisões planares foi estendida para viabilizar a conexão de duas malhas paramétricas, usando árvores de ordenação espacial para o armazenamento das entidades. Estas árvores espaciais reduzem a complexidade dos cálculos de interseção pelo descarte imediato das partes de uma malha que não podem interceptar a outra. A estrutura DCEL modificada, usada para representar cada retalho de superfície, viabiliza a construção das curvas de interseção sem a necessidade de buscas globais. As informações topológicas são usadas também na reconstrução das malhas visando obter elementos com boa qualidade geométrica, aplicáveis em análises por elementos finitos.

Os resultados mostram que o algoritmo é rápido se comparado com a construção das superfícies individualmente, podendo ser usado em modelagens interativas com superfícies compostas. Testes feitos com malhas de até cem mil faces sugerem que o tempo de processamento do algoritmo cresce linearmente com o número de faces envolvidas nas interseções. Isto pode ser explicado pelo pequeno número de faces que realmente são interceptadas devido às buscas eficientes possibilitadas pelas árvores de ordenação espacial.

A reconstrução das topologias, mais especificamente o algoritmo de triangulação das faces de *trimming* é o que consome mais tempo de processamento de todos os três passos (uma média de 60% do tempo total), e a sua implementação atual usa as técnicas de aceleração descritas por O'Rourke [67], que ainda apresentam complexidade $O(n^2)$, onde n é o número total de arestas nas regiões de interseção. Na prática, n é muito menor do que o número total N de arestas em ambas as malhas. Tipicamente, $n = O(\sqrt{N})$, o que é consistente com os resultados experimentais mencionados acima.

A estrutura de dados DCEL é simples e compacta, tendo se mostrado eficiente para o programa de modelagem. É importante observar a construção das representações topológicas apenas para as malhas que se interceptam, pois o uso indiscriminado da DCEL em todas as superfícies poderia tornar a modelagem de objetos compostos por muitos retalhos inviável, do ponto de vista de memória utilizada. Para aplicações que necessitem tratar todas as degenerações topológicas em interseções mais genéricas (faces com furos, arestas do tipo auto-ciclo ou arestas e vértices internos a faces), a estrutura de dados *half-edge* [58] poderia ser uma escolha mais adequada. Entretanto, a estrutura *half-edge* é bastante menos compacta e o impacto do uso de memória deveria ser avaliado cuidadosamente, testando-se malhas grandes.

O algoritmo de interseção não requer a ordenação dos pontos de interseção para definir o corte de uma face usando testes geométricos, como apresentado por Cavalcanti e outros [19]. O algoritmo também não necessita de buscas globais para o encadeamento dos segmentos individuais, necessárias no algoritmo de Lo [55], pois esta tarefa é feita à medida em que as poligonais globais são calculadas com testes topológicos nas extremidades das curvas de interseção. Curvas de *trimming* com curvaturas acentuadas e vários componentes conexos são também tratados naturalmente.

A qualidade geométrica de algumas curvas de interseção geradas pelo algoritmo foram comparadas calculando-se a distância dos pontos avaliados sobre elas com a solução analítica, e os resultados mostraram que eles estão muito próximos. Desta forma, conclui-se que a metodologia de interpolação usando a variação paramétrica definida por Nielson e Foley [40] apresentou bons resultados nos testes do algoritmo.

Um dos problemas centrais enfrentado pelos métodos de marcha [77] é a determinação da topologia das curvas de interseção. Em um passo intermediário pode-se passar de uma componente conexa a outra, se as duas curvas estão muito próximas. Com o algoritmo apresentado, o problema das componentes conexas é bem resolvido se as malhas possuem resolução suficiente para representar curvas próximas.

Devido ao elevado custo computacional do algoritmo de suavização global, tentou-se, ao longo dos testes, fazer apenas a movimentação dos vértices pertencentes às faces de *trimming*, identificadas no Passo 3a do algoritmo. Com este procedimento, o algoritmo se torna bastante mais rápido e os triângulos também apresentam boa qualidade geométrica. Entretanto, os elementos quadrilaterais adjacentes apresentavam invariavelmente distorções elevadas. Como foi apresentado recentemente [13], esforços no sentido de construir algoritmos simultaneamente locais e regulares para a suavização de malhas triangulares são inúteis, pois estas propriedades são incompatíveis. A técnica de pré-suavização paramétrica implementada faz com que os afastamentos decorrentes do reposicionamente tridimensional sejam menores em média, diminuin-

do o número de iterações necessárias ao algoritmo de reprojeção dos vértices, o que acelera a suavização. Observa-se que três repetições do algoritmo de suavização em 3D fazem com que novos reposicionamentos sejam desnecessários. A diferença média obtida entre três repeticões e trezentas é da ordem de 0.01%. Como pode-se interpretar pela observação da Figura 3.23, os elementos que possuem menor qualidade geométrica são adjacentes às fronteiras dos mapeamentos, devido à restrição física imposta. Uma técnica global de suavização pode ser avaliada em trabalhos futuros (vide Seção 5.1).

O conjunto de faces que compõem as regiões distintas, que são separadas por arestas de *trimming* e pertencentes à fronteira, podem ser facilmente identificadas mesmo usando-se uma R^*-tree em cada retalho. Partindo-se de uma face qualquer da região que se quer identificar, propaga-se a seleção através das faces adjacentes pelas arestas interiores que possuem o campo *trimming* valendo zero, ou seja, aquelas que não pertencem à fronteira ou tenham sido inseridas sobre curvas de *trimming*. Esta identificação permite a especificação de atributos diferentes dentro de uma mesma superfície e corte das partes excedentes.

O exemplo apresentado na Figura 4.17 mostra que o algoritmo pode ser usado para modelagens de superfícies compostas contendo várias interseções em um mesmo retalho com um número relativamente pequeno de operações de interseção. Este exemplo também mostra que a maioria dos elementos manteve o seu aspecto original, e aqueles adjacentes aos elementos interceptados tiveram seus vértices movidos de forma a aumentar a qualidade geométrica.

5.1 Trabalhos futuros

Com o objetivo de tornar o algoritmo de interseções mais robusto e independente das malhas envolvidas, poderia-se tentar solucionar os problemas topológicos de dois cruzamentos em uma mesma aresta e duas curvas de *trimming* em uma mesma face fazendo subdivisões automáticas a medida em que tais casos ocorram. Este refinamento local das malhas não comprometeria nem a precisão geométrica das curvas de interseção e nem o rendimento do algoritmo, uma vez que o número de ocorrências destes casos nos exemplos testados foi muito pequeno.

Para unificar a representação interna de curvas e superfícies, poderia-se adotar a representação por NURBS, usando-se a definição de pontos de controle das B-splines que interpolam os pontos definidos pelas curvas poligonais, B-splines e arcos de círculo, e pelas superfícies de Coons e *sweeps* [5], mantendo-se, entretanto, a interface diferenciada para cada tipo existente no MG. Com o uso de NURBS, a criação de novos tipos de curvas e superfícies consistiria apenas de uma nova estratégia de definição dos pontos de controle em cada caso.

De forma a acomodar a representação de subdivisões espaciais com retalhos curvos, po-

deria ser investigada a substituição da estrutura de dados do MG pela RED, pois, como pode ser observado pelos exemplos mostrados, poucos retalhos são usados na definição de modelos compostos. Entretanto, a estrutura DCEL (ou uma outra representação planar topológica) interna a cada superfície interceptada deveria ser mantida em conjunto com a referência para cada face da RED.

O algoritmo de reconstrução das topologias está sendo modificado para converter os elementos triangulares em quadrilaterais de forma a uniformizar as malhas resultantes. Alguns trabalhos sobre geração de malhas quadrilaterais têm sido considerados, apresentando soluções que variam entre o uso de padrões ou *templates* para geração de nós internos [79], e a troca ou eliminação de arestas [69, 88].

São conhecidos resultados bidimensionais onde elementos quadrilaterais em análises por elementos finitos em estado plano de tensão ou deformação produzem resultados melhores que as malhas com elementos triangulares [57]. Por outro lado, estudos com cascas enrijecidas mostram bons resultados de elementos quadráticos triangulares [9]. Antes de se fazer a alteração global da malha para a uniformização com quadriláteros ou triângulos, deve-se fazer uma análise criteriosa com simulações em cascas tridimensionais para se avaliar a qualidade *real* das malhas geradas com a metodologia de reconstrução proposta. Avaliações preliminares feitas pelos engenheiros do CENPES usando elementos de contorno têm mostrado bons resultados.

Os elementos adjacentes às fronteiras das superfícies são os que apresentam pior qualidade geométrica, de forma que seria interessante fazer um tratamento nos vértices das fronteiras interceptadas. Este tratamento deveria ser feito simultaneamente em todas as as malhas das superfícies adjacentes, reposicionando-se os vértices que subdividem as arestas de contorno usando-se a descrição geométrica da curva situada na fronteira interceptada.

Na maioria das modelagens bidimensionais [37], define-se a *geometria* pela especificação das curvas de contorno, fazendo a subdivisão planar, e, para as regiões onde os mapeamentos elementares não se aplicam, lança-se mão de algoritmos de triangulação. A técnica de modelagem tridimensional com modificações locais das malhas poderia ser testada em duas dimensões, como uma alternativa à modelagem convencional. Deveriam ser avaliados o esforço do usuário, a complexidade dos algoritmos de interseção, e a qualidade das malhas geradas, tanto do ponto de vista geométrico quanto da acurácia dos resultados das simulações.

Apêndice A

Operadores topológicos

A criação e manutenção da subdivisão paramétrica armazenada pela DCEL é consistentemente mantida por um conjunto de *Operadores de Euler*, que mantêm a fórmula de *Euler Poincaré* [46, 58] ($V - A + F = 2$). Os operadores usados no MG para tratar os retalhos com interseção foram inicialmente desenvolvidos para implementar [50] o algoritmo de triangulação de Delaunay apresentado em [33].

Este grupo de operadores construtivos e destrutivos encapsula todas as atualizações necessárias à estrutura DCEL, sendo apresentado a seguir:

- **MSVF:** O operador *Make Surface Vertex and Infinity Face* inicializa a construção criando a entidade *superfície*, gerando um vértice e a *face infinita*. As árvores são também inicializadas e a face infinita não é inserida na R-tree por ser uma entidade especial com orientação contrária às demais. O número de Euler é mantido ($1 - 0 + 0 = 1$).

- **MEV:** O operador *Make Edge and Vertex* cria uma nova aresta e um vértice que se conecta ao restante da estrutura apenas por esta aresta.

- **MEBF:** O operador *Make Edge Bound Face* cria a primeira face da representação dividindo a face infinita em duas, com a inserção de uma nova aresta. Este operador especial é acionado apenas uma vez durante toda a construção e é apresentado em separado por possuir implementação e funcionalidade totalmente distintas do tradicional *Make Edge and Face*.

- **MEF:** O operador *Make Edge and Face* cria uma aresta que conecta dois vértices existentes, dividindo uma face em duas. A face que é subdividida é passada como parâmetro para o operador MEF, que é usado para a criação de todas as faces seguintes à face inicial, criada pelo *MEBF*.

- **KEF:** O operador *Kill Edge and Face* (inverso do *MEF*) é utilizado na remoção de uma aresta e, conseqüentemente, de uma face. As arestas que apontavam para a face eliminada passam a apontar para a outra vizinha da aresta eliminada.

- **KEV:** O *Kill Edge and Vertex* (inverso do *MEV*) elimina uma aresta e um vértice, que apenas se conecta ao restante da DCEL por esta aresta.

- **KSVF:** O operador *Kill Surface Vertex and Infinity Face* é o inverso do *MSVF* sendo usado para a remoção final das entidades, esvaziando a estrutura.

- **SEMV:** O operador *Split Edge and Make Vertex* é utilizado para dividir arestas que pertencem ao contorno ou a uma curva de *trimming* interna a um retalho, no passo 3(b) do algoritmo, apresentado na Seção 3.6.

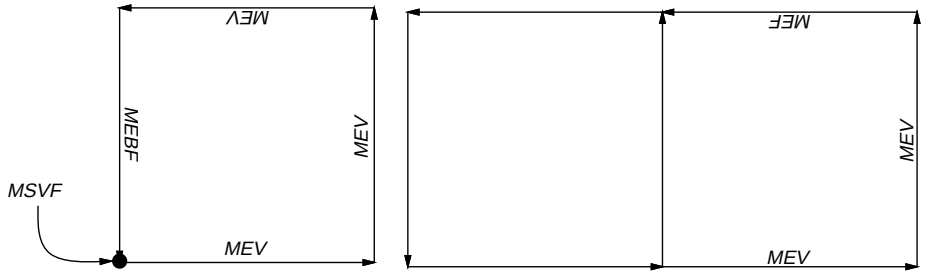
Nenhum deste operadores faz qualquer tipo de busca global. As modificações são todas locais e as buscas são feitas em um nível superior, onde são usadas as chaves apropriadas nas árvores de armazenamento apresentadas. A inserção e a remoção de entidades nas árvores também são feitas com as chaves de busca.

Apêndice B

Mapeamento Inicial das superfícies

Pode-se fazer a construção partindo-se da estrutura clássica de elementos finitos, onde as faces são identificadas por circuitos de vértices, armazenadas em uma lista ou vetor, e os vértices contêm as coordenadas geométricas e um indexador que os identifica no circuito das faces, estando usualmente armazenados em um vetor. As coordenadas dos vértices, entretanto, devem ser identificadas no espaço paramétrico das superfícies e não em 3D como são apresentadas em modelos para elementos finitos.

A determinação do operador topológico que deve ser acionado e das entidades que devem ser alteradas são resultantes das buscas feitas, antes da inserção de cada entidade durante a transformação das faces da estrutura de elementos finitos na DCEL. A Figura B.1(a) mostra os operadores necessários à criação da primeira face da representação. A Figura B.1 (b) mostra-se a sequência de operadores necessário à inserção da segunda e demais faces.



(a) Face inicial. (b) Segunda Face.

Figura B.1: Operadores para construção das faces iniciais.

Referências Bibliográficas

[1] J. Ahlberg, E. Nilson, and J. Walsh. *The Theory of Splines and their Applications*. Academic Press, 1967.

[2] L.N. Andrade and Wu, Shin–Ting. Caminhando sobre uma interseção de superfícies com passos circulares. *Anais do IX SIBGRAPI*, páginas 151–158, 1996.

[3] R. Barnhill and S. Kersey. A marching method for parametric surface/surface intersection. *Computer Aided Geometric Design*, 7:257–280, 1990.

[4] R.E. Barnhill, G. Farin, M. Jordan, and B.R. Piper. Surface/surface intersection. *Computer Aided Geometric Design*, 4:3–16, 1987.

[5] B. Barsky and D. Greenberg. Determining a set of B-spline control vertices to generate an interpolating surface. *Computer Graphics and Image Processing*, 14:203–209, 1979.

[6] W. Barth, R. Lieger, and M. Schindler. Ray tracing general parametric surfaces using interval arithmetic. *The Visual Computer*, 10:363–371, 1994.

[7] B. Baumgart. A polyedron representation for computer vision. *AFIPS Proc.*, volume 44, páginas 589–596, 1975.

[8] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The R^*-Tree: An efficient and robust access method for points and rectangles. *ACM SIGMOD Conference on Management of Data*, páginas 322–332, Maio 1990.

[9] A.C. Benjamin. *Análise do Comportamento Não Linear Físico e Geométrico de Cascas Enrijecidas*. Tese de doutorado, COPPE/UFRJ, 1991.

[10] P. Bézier. Définition numérique des courbes et surfaces I. *Automatisme*, XI:625–632, 1966.

[11] J Bloomenthal. Calculation of reference frames along a space curve. *Graphics Gems*, páginas 567–571, 1990.

[12] K. Brodlie. A review of methods for curve and function drawing. K. Brodlie, editor, *Mathematical Methods in Computer Graphics and Design*, páginas 1–38. Academic Press, 1980.

[13] J.F. Buss and R.B. Simpson. Planar mesh refinement cannot be both local and regular. *Numerische Mathematik*, 79:1–10, 1998.

[14] M.M. Carneiro. Interact: um modelo de interação para editores gráficos. Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 1995.

[15] M.S. Casale. Free-form solid modeling with trimmed surface patches. *IEEE Computer Graphics and Applications*, 7:33–43, Janeiro 1987.

[16] C.R. Cassino. *CD - Manual de Referência - versão 2.0*. Pontifícia Universidade Católica do Rio de Janeiro, Grupo de Tecnologia em Computação Gráfica - TeCGraf/PUC-Rio, Abril 1996. `http://www.puc-rio.br/tecgraf/manuais/cd.html`.

[17] C.R. Cassino. *G3D - Manual de Referência - versão 1.0*. Pontifícia Universidade Católica do Rio de Janeiro, Grupo de Tecnologia em Computação Gráfica - TeCGraf/PUC-Rio, Abril 1996.

[18] B. Castier, L.F. Martha, and M. Gattass. Uma taxonomia para manipulação interativa e visualização de objetos 3D. *Anais do VII SIBGRAPI*, páginas 149–156, 1994.

[19] P.R. Cavalcanti, P.C.P. Carvalho, and L.F. Martha. Non-manifold modeling: An approach based on spatial subdivision. *Computer-Aided Design*, 29:209–220, Março 1997.

[20] R.E. Chandler. A recursive technique for rendering parametric curves. *Computers and Graphics*, 14:477–479, 1990.

[21] J.J. Chen and T.M. Ozsoy. Predictor–corrector type of intersection algorithm for C^2 parametric surfaces. *Computer-Aided Design*, 20:347–352, 1988.

[22] L.C. Gomes Coelho and C.S. de Souza. Comunicação de problemas e soluções geométricas em uma interface 3D. *Anais do VIII SIBGRAPI*, páginas 233–240, 1995.

[23] L.C.G. Coelho, M. Gattass, and L.F. Martha. Modeling techniques to generate 3D meshes. *Anais do XVI CILAMCE*, páginas 1270–1274, 1996.

[24] L.C.G. Coelho, L.F. Martha, C.S. de Souza, and M. Gattass. Geração de malhas de superfície no espaço com manipulação direta e orientação a objetos. *Anais do XIV CILAMCE*, páginas 1275–1284, 1993.

[25] J.L.D. Comba and J. Stolfi. Affine arithmetic and its applications to computer graphics. *Anais do VI SIBGRAPI*, páginas 9–18, 1993.

[26] D. Comer. The Ubiquitous B-tree. *ACM Computing Surveys*, 11:121–131, Junho 1979.

[27] S.A. Coons. Surfaces for computer aided design. Relatório técnico, Design Division, Mech. Engin. Dept., M.I.T., Cambridge, Massachusetts, 1964.

[28] C. de Boor. *A Practical Guide to Splines*. Springer, 1978.

[29] P. de Casteljau. *Outillages méthodes calcul*. Relatório técnico, A. Citroen, Paris, 1959.

[30] P. de Casteljau. *Courbes et surfaces à poles*. Relatório técnico, A. Citroen, Paris, 1963.

[31] L.H. de Figueiredo. Adaptive sampling of parametric curves. *Graphics Gems V*, páginas 173–178, 1995.

[32] L.H. de Figueiredo. Surface intersection using affine arithmetic. *Graphics Interface '96*, páginas 161–170, 1996.

[33] L. de Floriani and E. Puppo. An on-line algorithm for constrained Delaunay triangulation. *CVGIP: Graphical Models and Image Processing*, 54:290–300, 1992.

[34] P. Deuflard. A modified Newton method for the solution of ill-conditioned systems of non-linear equations with applications to multiple shooting. *Numer. Math.*, 22:289–315, 1974.

[35] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, 1990.

[36] I. D. Faux and M. J. Pratt. *Computational Geometry for Design and Manufacture.* Ellis Horwood, 1979.

[37] W. Celes Filho. *Gerenciamento de Subdivisões Planares Hierárquicas*. Tese de doutorado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 1995.

[38] W. Celes Filho. *MTOOL - Manual do Usuário - versão 2.0*. Pontifícia Universidade Católica do Rio de Janeiro, Grupo de Tecnologia em Computação Gráfica - TeCGraf/PUC-Rio, Janeiro 1996.

[39] R. Fischer. Genesys - sistema híbrido para modelagem de sólidos. Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 1991.

[40] T.A. Foley and G.M. Nielson. Knot selection for parametric spline interpolation. L. Schumaker, editor, *Mathematical Methods in CAGD*, páginas 445–467. Academic Press, 1989.

[41] G.L. Fonseca. Editor gráfico de malhas transfinitas tridimensionais para elementos finitos. Dissertação de Mestrado, Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro, 1989.

[42] M. Gleicher and M. Kass. An interval refinement technique for surface intersection. *Graphics Interface '92*, páginas 242–249, Maio 1992.

[43] U. Gudukbay and B. Ozguç. Free-form solid modeling using deformations. *Computers & Graphics*, 14:491–500, 1990.

[44] R. Haber and J.F. Abel. Discrete transfinite mappings for the description and meshing of three-dimensional surfaces using interactive computer graphics. *International Journal for Numerical Methods in Engineering*, 18:41–66, 1982.

[45] R. Haber, M.S. Shephard, J.F. Abel, R.H. Gallagher, and D.P. Greenberg. A general two-dimensional, graphical finite element preprocessor utilizing discrete transfinite mappings. *International Journal for Numerical Methods in Engineering*, 17:1015–1044, 1981.

[46] C.M. Hoffmann. *Geometric and Solid Modeling – An Introduction*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989.

[47] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A K Peters, 1993.

[48] E. Houghton, E. Emnett, R. Factor, and L. Sabharwal. Implementation of a divide-and-conquer-method for the intersection of parametric surfaces. *Computer Aided Geometric Design*, 2:173–184, 1985.

[49] S. Katz and T. Sederberg. Genus of the intersection curve of two rational surface patches. *Computer Aided Geometric Design*, 5:253–258, 1988.

[50] G. Lanzilli, A. Martino, and I. Saccardo. DELAUNAY: un programma per la costruzione incrementale di una triangolazione vincolata di Delaunay. Relatório técnico, Facoltà di Ingegneria, Università degli Studi di Roma "La Sapienza", 1992.

[51] T.S. Lau and S.H. Lo. Finite element mesh generation over analytical curved surfaces. *Computers & Structures*, 59:301–309, 1996.

[52] E.T.Y. Lee. Choosing nodes in parametric curve interpolation. *IEEE Computer Graphics and Applications*, 21:363–370, Julho 1989.

[53] M.A. Linton, P.A. Calder, and J.M. Vlissides. Interviews: A [C++] graphical interface toolkit. Relatório técnico CSL-TR-88-358, Stanford U, Julho 1988.

[54] S.H. Lo. A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21:1403–1426, 1985.

[55] S.H. Lo. Automatic mesh generation over intersecting surfaces. *International Journal for Numerical Methods in Engineering*, 38:943–954, 1995.

[56] R.H. MacNeal. *MSC/PATRAN 6 & 7 FEA USER'S MANUAL*. MacNeal Schwendler Corporation, Janeiro 1996. `http://www.macsch.com:80/bookstore/patran.html`.

[57] A. Malanthara and W. Gerstle. Comparative study of unstructured meshes made of triangles and quadrilaterals. *6th International Meshing Roundtable'97*, páginas 437–447, Outubro 1997.

[58] M. Mäntylä. *An Introduction to Solid Modeling*. Science Press, Rockville, Maryland, 1988.

[59] L.F. Martha. *Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Fracture Simulations in Three-Dimensions*. Tese de doutorado, Cornell University, 1989. Ithaca, N.Y.

[60] M.R. Mediano, M. Gattass, and M.A. Casanova. HPS-tree: Um método de acesso para armazenar mapas longos com multi-resolução geométrica e topológica. *Anais do IX SIBGRAPI*, páginas 219–226, 1996.

[61] J.G. Militão and P.C.P. de Carvalho. Uma metodologia para posicionamento de prédios em terrenos acidentados. *Anais do IX SIBGRAPI*, páginas 227–234, 1996.

[62] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide*. Addison Wesley, 1993.

[63] J.B. Cavalcante Neto. Simulação auto-adaptativa baseada em enumeração espacial recursiva de modelos bidimensionais de elementos finitos. Dissertação de Mestrado, Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro, 1994.

[64] J.B. Cavalcante Neto, M.T.M. Carvalho, and L.F. Martha. Combinação das técnicas de quadtree e Delaunay para geração automética de malhas de elementos finitos. *Anais do VI SIBGRAPI*, páginas 285–291, 1993.

[65] G.M. Nielson and T.A. Foley. A survey of applications of an affine invariant norm. L. Schumaker, editor, *Mathematical Methods in CAGD*, páginas 445–467. Academic Press, 1989.

[66] A. O'Leary. *ANSYS Modeling and Meshing Guide – Release 5.4*. SAS IP Incorporated, Maio 1998. `http://www.ansys.com/ServSupp/Library/library.html`.

[67] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, New York, 1993.

[68] G. Peters. Interactive computer graphics application of the parametric bi-cubic surface to engineering design problems. R. Barnhill and R. Riesenfeld, editors, *Computer Aided Geometric Design*. Academic Press, 1974.

[69] D.O. Potyondy, P.A. Wawrzynek, and A.R. Ingraffea. An algorithm to generate quadrilateral or triangular element surface meshes in arbitrary domains with applications to crack propagation. *International Journal for Numerical Methods in Engineering*, 38:2677–2701, 1995.

[70] M. Pratt and A. Geisow. Surface/surface intersection problems. J. Gregory, editor, *The Mathematics of Surfaces*. Clarendon Press, 1986.

[71] F.P. Preparata and M.I. Shamos. *Computational Geometry - an introduction*. Springer Verlag, New York, 1990.

[72] H. Ratschek and J. Rokne. *Computer Methods for the Range of Functions*. Ellis Horwood Ltd., 1984.

[73] A. A. G. Requicha. Representations for rigid solids: Theory methods, and systems. *ACM Computing Surveys*, 12:437–464, Dezembro 1980.

[74] D.F. Rogers and J.A. Adams. *Mathematical Elements for Computer Graphics*. McGraw Hill, 1990.

[75] D. Rypl and P. Krysl. Triangulation of 3D surfaces. *Engineering with Computers*, 13:87–98, 1997.

[76] H. Samet. Neighbor finding techniques for images represented by quadtrees. *Computer Graphics and Image Proccessing*, 18:37–57, 1982.

[77] P. Schramm. Intersection problems of parametric surfaces in CAGD. *Computing*, 53:355–364, 1994.

[78] X. Sheng and B.E. Hirsh. Triangulation of trimmed surfaces in parametric space. *Computer-Aided Design*, 24:437–444, Agosto 1992.

[79] K. Shimada and T. Itoh. Automatic conversion of 2D triangular mesh into quadrilateral mesh. *International Conference on Computational Engineering Science*, páginas 350–355, 1995. `http://www.trl.ibm.co.jp/projects/s7340/meshing/t2q/t2qE.htm`.

[80] K. Shoemake. ARCBALL: A user interface for specifying three-dimensional orientation using a mouse. *Graphics Interface '92*, páginas 151–156, Maio 1992.

[81] L.T. Souza and M. Gattass. A new scheme for mesh generation and mesh refinement using graph theory. *Computers & Structures*, 46:1073–1084, 1993.

[82] H. Späth. *Two Dimensional Spline Interpolation Algorithms*. A K Peters, 1993. ISBN 1-56881-017-2.

[83] Tz. E. Stoyanov. Marçoing along surface/surface intersection curves with an adaptive step length. *Computer Aided Geometric Design*, 9:485–489, 1992.

[84] M.J.G.M. van Emmerik. A direct manipulation technique for specifying 3D object transformations with a 2D input device. *Computer Graphics Forum*, 9:355–362, 1990.

[85] G.N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design: With Applications*. McGraw Hill Text, 1984.

[86] K. Weiler. *Topological Structure for Geometric Modeling*. Tese de doutorado, Rensselaer Polythecnic Institute, 1986. Troy, N,Y.

[87] J. Wernecke. *The Inventor Mentor*. Addison Wesley, 1994.

[88] J.Z. Zhu, O.C. Zienkiewicz, E. Hinton, and J. Wu. A new approach to the development of automatic quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32:849–866, 1991.

Modelagem de cascas com interseções paramétricas

Luiz Cristovão Gomes Coelho

Tese apresentada ao Departamento de Informática da PUC-Rio, no dia 11 de agosto de 1998, como parte dos requisitos para a obtenção do título de Doutor em Ciências em Informática, tendo sido aprovada pela Banca Examinadora, da qual participaram os seguintes professores:

Prof. Marcelo Gattass orientador
PUC-Rio

Prof. Luiz Henrique de Figueiredo co-orientador
LNCC

Prof. Luiz Fernando Martha
PUC-Rio

Profa. Wu, Shin–Ting
UNICAMP

Prof. Paulo Cezar Pinto Carvalho
IMPA

Prof. Marcelo Dreux
PUC-Rio

Prof. Roberto de Beauclair Seixas
LNCC

Visto e permitida a impressão.
Rio de Janeiro, de de 1998.

Coordenador dos Programas de Pós-Graduação do Centro Técnico Científico