2
O Modelador Geométrico MG

Apresenta-se aqui um modelador de elementos finitos existente, o MG. Trata-se de um pré-processador desenvolvido para a geração de malhas sólidas de elementos finitos. É feita uma breve análise do ambiente de modelagem e da estrutura de classes do modelador, que é todo implementado segundo o conceito de programação orientada a objetos (POO), frisando-se as estruturas de dados topológicas utilizadas e a existência das ferramentas de interseção e detecção automática de regiões comentadas anteriormente, que serão de extrema importância na implementação do algoritmo de operações booleanas neste modelador, como será descrito nos próximos capítulos.

2.1.
Características gerais do modelador

No modelador MG, a base para a geração das superfícies (cascas) e dos volumes é a construção de curvas. A manipulação das entidades gráficas (vértices e curvas) é feita diretamente na área de desenho (*canvas*) e estas operações são orientadas por um plano de interface que faz a transformação do espaço bidimensional para o espaço tridimensional (Figura 2.1). A interface por manipulação direta permite maior liberdade na criação da geometria do objeto, contudo existe também a possibilidade de se entrar com os valores exatos das coordenadas ou dos parâmetros das transformações através de itens de menu.

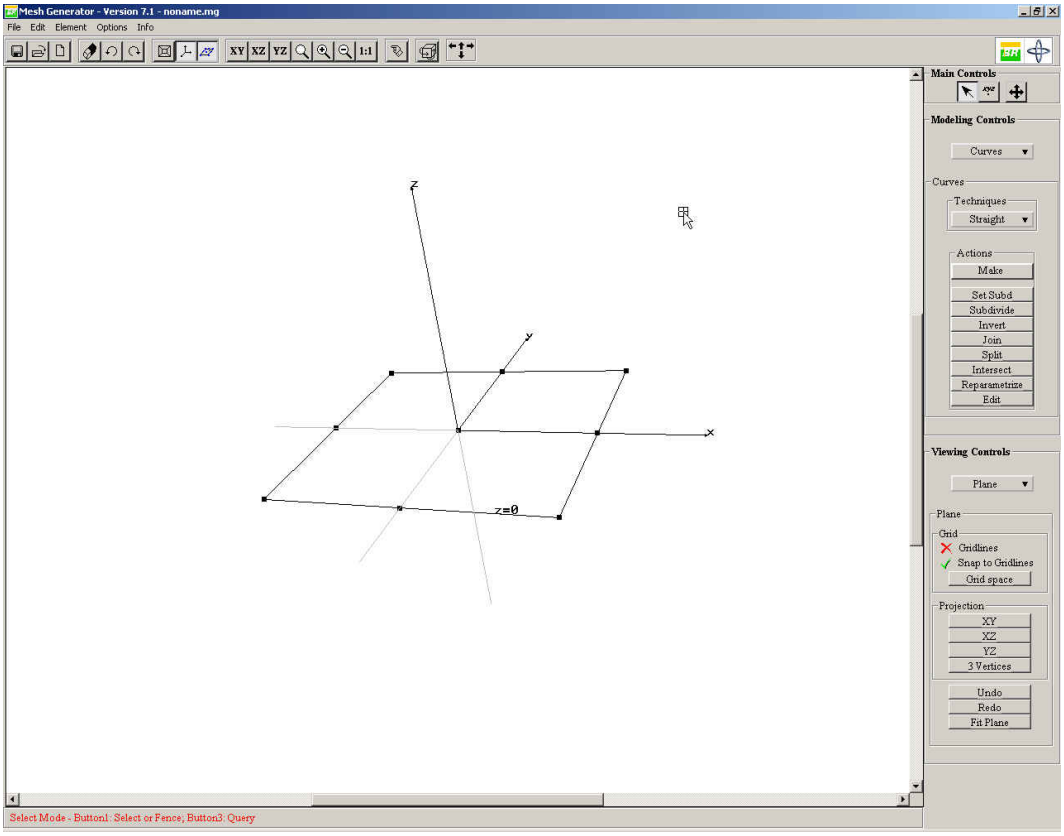


Figura 2.1 – Interface do MG e plano de interface.

Este trabalho traz uma inovação para o usuário do MG, que passa a usufruir de mais uma opção de interface para a criação de objetos. Independentemente da maneira como esta nova ferramenta está internamente implementada no modelador, buscou-se criar uma interface para as operações booleanas que se assemelhe às interfaces dos sistemas de modelagem CSG, permitindo que o usuário possa gerar sólidos unicamente através da combinação de outros sólidos pré-existentes, sem a necessidade da manipulação direta das curvas e superfícies do modelo.

O MG cria automaticamente um arquivo para armazenar as diversas etapas de edição do modelo, ou seja, ou arquivo de *backup* das informações do modelo, ou arquivo de *check point*. Quando o usuário chama a função de *undo* do programa, este arquivo é usado para fazer as alterações necessárias à estrutura de dados. Arquivos de *check point* não possuem utilidade somente no caso de o usuário desejar desfazer um passo de modelagem, mas também no caso de uma falha do sistema (execução de uma operação ilegal, invasão de memória) ou interrupção de energia. Através de parâmetros adicionais na linha de comando utilizada para a execução do programa, pode-se recuperar o estado imediatamente anterior à interrupção.

Nas simulações por elementos finitos modeladas com o MG, várias técnicas para geração de malhas não-estruturadas de elementos finitos foram e vêm sendo desenvolvidas, com algoritmos baseados em triangulação de Delaunay [20], em decomposição espacial recursiva [21] e em métodos de avanço de fronteira [22,23,24]. O modelador MG utiliza um algoritmo para geração de malhas sólidas de tetraedros em domínios arbitrários, apresentado por Cavalcante Neto [25], que também combina técnicas de avanço de fronteira com decomposição espacial recursiva. Algoritmos para geração de malhas estruturadas também estão presentes no MG, tornando-o capaz de gerar malhas com os mais diversos tipos de elementos finitos conhecidos na literatura.

É desejável que um modelador seja capaz de especificar atributos para os mais diversos tipos de problemas de engenharia, ou seja, que os atributos sejam configuráveis com relação ao tipo de análise desejada. Lira [6] incorporou ao MG o sistema de gerenciamento e configuração de atributos ESAM (*Extensible System Attributes Management*). Isto possibilitou o uso deste modelador nos mais diversos problemas de engenharia usando elementos finitos. Em seu trabalho, Lira também reorganizou internamente o código do programa MG de forma a torná-lo totalmente orientado a objetos e implementou as ferramentas de interseção de superfícies paramétricas e detecção automática de multi-regiões que serão analisadas mais detalhadamente em seções posteriores.

2.2. Modos de interface do MG

O MG incorpora os seguintes modos de interface:

- Seleção
- Mudança de Projeções
- Edição do Plano de Interface
- Transformações por Manipulação Direta
- Criação:
    1. de Curvas
    2. de Vértices
    3. de Malhas
    4. de Volumes
    5. de Sólidos
    6. de Grupos
- Visualização:

1. de Curvas
2. de Vértices

- de Malhas
- de Volumes
- de Sólidos

O modo de interação corrente determina qual o conjunto de operações disponíveis para o usuário. Alguns destes modos de interação serão brevemente discutidos a seguir.

2.2.1. Seleção

O modo de seleção é o modo *default* do programa. No MG as entidades gráficas são tridimensionais projetadas na tela. O programa seleciona a entidade que estiver à frente na projeção.

As entidades podem usufruir de dois estados: *selecionado* e *não selecionado*. O MG permite a seleção de múltiplas entidades. A seleção de uma ou mais entidades provoca a alteração do estado de seleção de todas as outras entidades para *não selecionado*. A seleção de várias entidades também pode ser realizada por meio de *fence*, ou seja, pela definição de uma janela de forma que todas as entidades integralmente contidas dentro dos limites da janela sejam selecionadas.

A *tolerância* é um aspecto importante na seleção de entidades. Dois pontos estão *próximos* quando a distância entre os dois pontos for menor que a tolerância, que pode ser definida pelo usuário. No modo de seleção, trabalha-se o tempo todo no espaço projetado da tela, então a proximidade também é calculada neste espaço projetado.

É importante frisar também a questão da *prioridade* de seleção. O programa adota a ordem de prioridade vértice, região, curva, malha 2D e malha 3D. Isto significa que se, por exemplo, o usuário apontar uma posição onde estejam presentes um vértice e uma curva dentro da tolerância de seleção, o MG seleciona o vértice.

Vértices de curvas distintas que são colocados próximos sofrem *atração*. Quando um vértice é posicionado no espaço, é feita uma pesquisa global, testando-se a proximidade de acordo com a tolerância corrente. Se algum vértice pré-existente do modelo estiver dentro do limite da tolerância, o sistema passa a

utilizar a referência deste vértice, alterando as coordenadas inicialmente fornecidas pelo usuário (Figura 2.2).

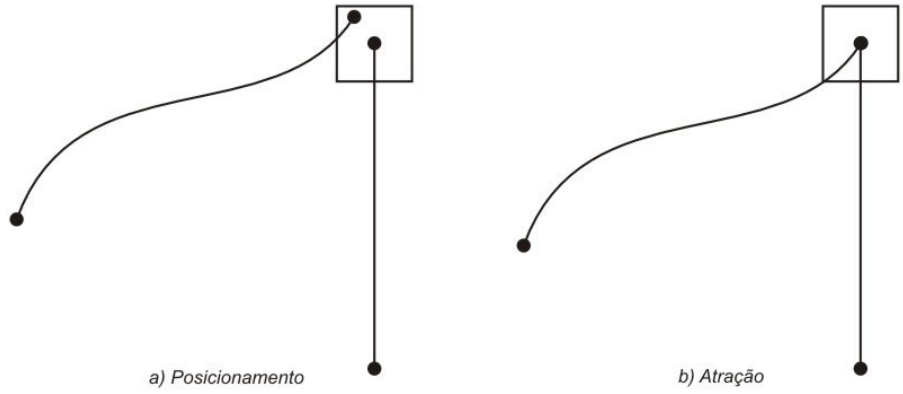


Figura 2.2 – Atração de vértices: a) posicionamento; b) atração [7].

2.2.2. Mudança de Projeções

O MG utiliza uma biblioteca que usa o modelo de câmera, que possui apenas o ponto de referência (*vrp*), a posição da câmera (*eye*) e o vetor que indica a direção vertical de projeção (*vup*). O *vrp* é colocado automaticamente pelo MG no centro do modelo e o usuário não tem acesso a este parâmetro. A posição inicial da câmera encontra-se a uma distância razoável do modelo e pode ser alterada pelo usuário. A Figura 2.3 mostra a posição inicial dos parâmetros de visualização, a esfera imaginária sobre a qual são feitas as movimentações e a vista esquemática da imagem que seria gerada na tela do computador.

Podem-se fazer translações da câmera, rotações da mesma na superfície da esfera ou em torno do seu eixo radial, alteração da distância da câmera ao ponto de referência, escala no modelo (aumentando ou diminuindo a projeção mas sem alterar nenhum parâmetro de visualização), dentre outras funções de visualização.

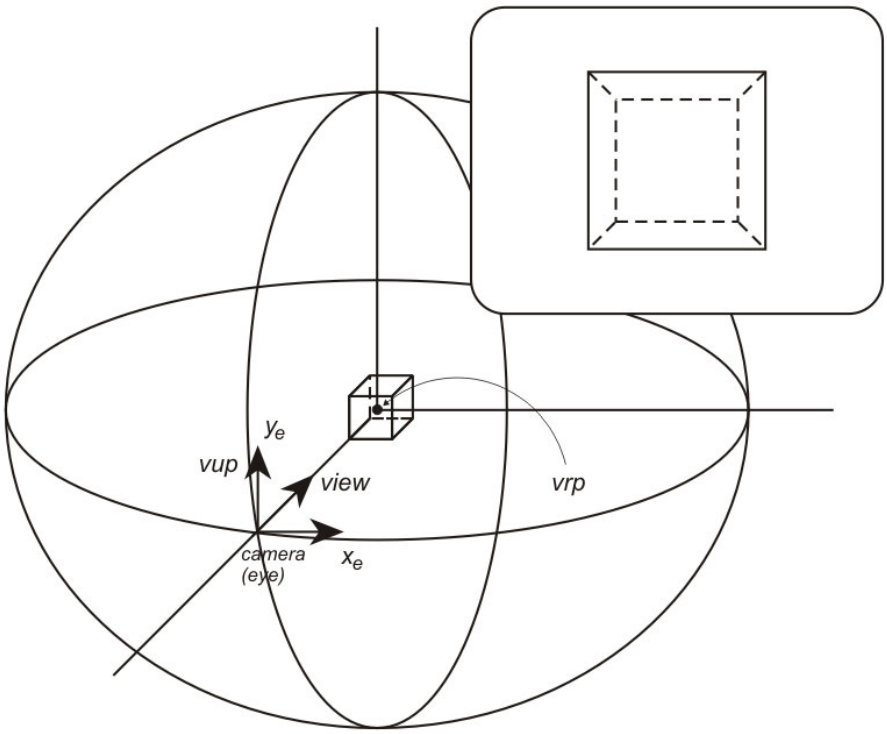


Figura 2.3 – Parâmetros de visualização e posição inicial da câmera (visão esquemática do objeto em destaque) [7].

2.2.3. Edição do plano de interface

O elemento gráfico de interface mais importante do MG é o plano de interface. Escalas, rotações e translações deste plano são as operações disponíveis por manipulação direta na área de desenho.

O plano de interface pode ser colocado nas posições padrões $z = 0$, $y = 0$ e $x = 0$, se for desejado. O usuário também possui controle sobre o desenho do *grid*, atração para os pontos do *grid* e espaçamento do *grid*. Pode-se adaptar o plano de interface aos limites do modelo, fazendo-se uma escala centrada na origem corrente do plano. O usuário também pode apontar três vértices consecutivos do modelo para que o plano de interface seja posicionado no plano geométrico definido por estes três vértices, possuindo origem no baricentro destas três posições.

2.2.4.
Transformações por manipulação direta

Este modo de interface visa a realização de transformações (translações e rotações) de entidades gráficas selecionadas por manipulação direta. Ao se selecionar este modo, eixos auxiliares são desenhados com origem no centro das entidades selecionadas para facilitar a execução das transformações pelo usuário (Figura 2.4). O centro de rotação é sempre a origem dos eixos. Podem-se fazer transformações apenas com o sistema de eixos para colocá-lo em uma posição diferente da inicial ou com todas as entidades selecionadas. É possível se alterar os valores escalares das translações e rotações através de campos editáveis na interface.

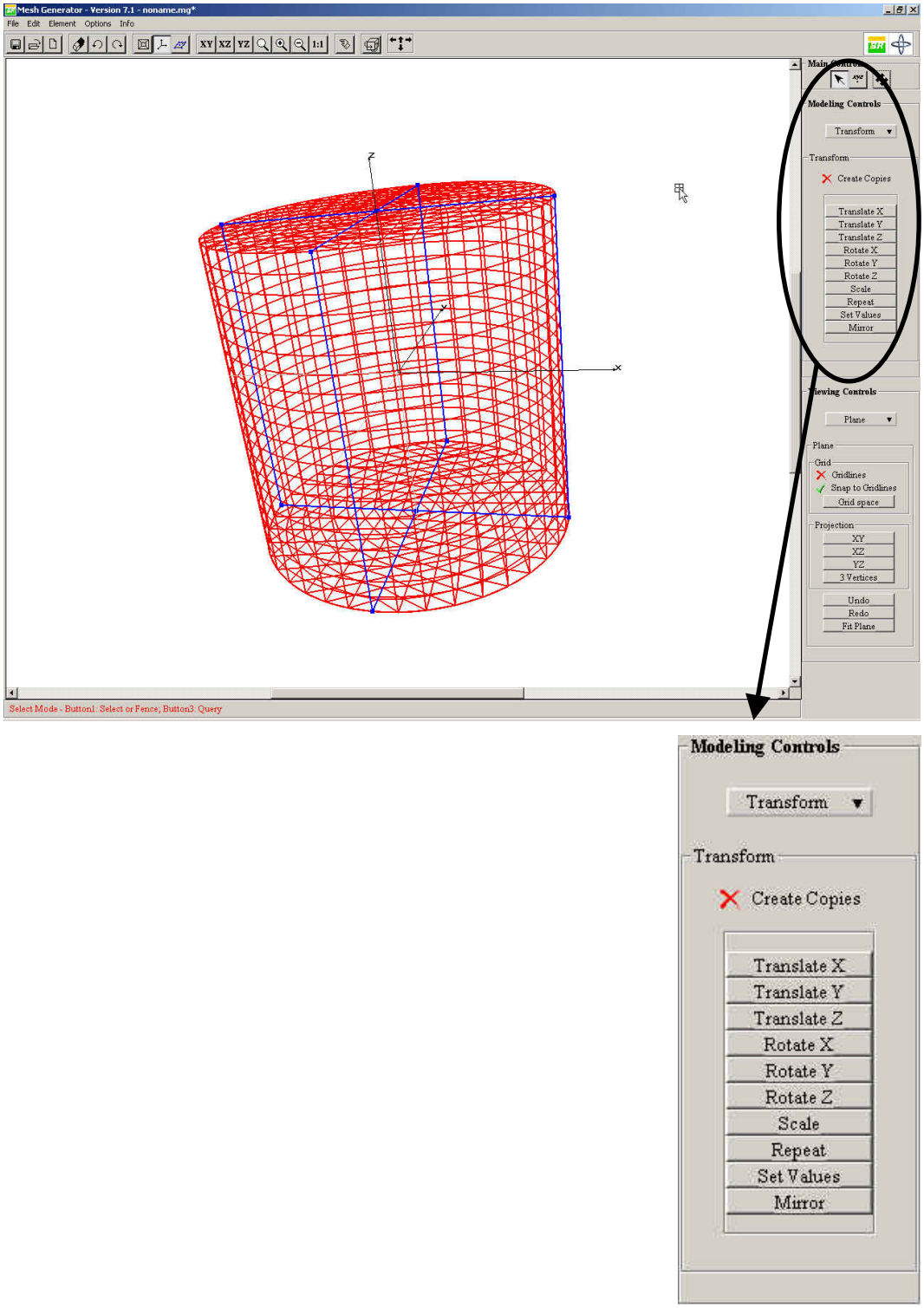


Figura 2.4 – Modo de transformação por manipulação direta.

2.2.5. Criação

A criação de vértices e curvas depende da especificação das coordenadas de vértices, que dependem da posição do plano de interface e do estado de habilitação de alguns elementos de interface que controlam a atração de pontos.

Uma das etapas mais importantes na criação de um vértice geométrico é a transformação das coordenadas bidimensionais (espaço da tela) para coordenadas tridimensionais (espaço tridimensional do objeto). Este processo é feito internamente pelo programa em duas etapas: cálculo da interseção da linha que parte do ponto que o usuário definiu na janela, possuindo a direção de projeção, com o plano de interface; e obtenção do ponto no espaço do modelo pela determinação da terceira coordenada a partir da equação implícita do plano e resolução de um sistema linear de equações com esse ponto [7].

Se o *snap* do plano de interface estiver ativo, é feita a atração para o ponto mais próximo do *grid*, respeitando-se o espaçamento corrente. Pode-se escolher também a opção de *snap* para os vértices já existentes no modelo. Esta opção faz com que seja feito um teste de *pick* nos vértices do modelo antes da transformação para o espaço do objeto. Se algum vértice estiver próximo (de acordo com a tolerância corrente) ao cursor, é feita a atração do cursor para a posição deste vértice, e suas coordenadas são utilizadas no posicionamento.

A criação de curvas, superfícies, malhas bidimensionais e tridimensionais, volumes, sólidos e grupos será analisada mais detalhadamente na próxima seção, que descreve a forma de representação interna destas entidades no modelador.

2.3. Modelagem geométrica no MG

Na seção anterior foi possível observar as principais características do MG de um ponto-de-vista de usuário. As funcionalidades básicas e a interface com o usuário foram apresentadas, buscando-se familiarizar o leitor com o ambiente de modelagem em que este trabalho está inserido.

Nesta seção, é apresentada inicialmente uma análise mais detalhada das entidades geométricas que podem ser criadas no MG, como curvas, superfícies, malhas 2D e malhas 3D. Algumas definições e a maneira como estas entidades são criadas no MG são mostradas, para que se possa ter uma noção do domínio representacional deste modelador.

Em seguida, uma visão mais interna do modelador será exposta. A maneira como as informações topológicas e geométricas são descritas e tratadas e a estrutura de dados utilizada para armazenar estas informações segundo uma representação B-Rep serão brevemente estudadas. A organização de classes do MG também será mostrada, enfatizando-se a importância do conceito de orientação a objetos no paradigma de modelagem utilizado.

Por fim, são apresentadas duas ferramentas essenciais num sistema de modelagem e que foram introduzidas no MG por Lira [6]: a interseção de superfícies paramétricas e a detecção automática de regiões. No escopo deste trabalho, estas duas ferramentas são imprescindíveis para a implementação correta e eficiente das operações booleanas, como será visto posteriormente.

2.3.1. Representação paramétrica de superfícies

Segundo Hoffmann [9], os dois métodos mais comuns para se representar superfícies são as formas implícitas e as equações paramétricas. Uma superfície é descrita pela sua forma implícita pela equação $f(x,y,z) = 0$, onde x, y e z formam o sistema de eixos no espaço Euclidiano. A equação $S(u,v) = (x(u,v),y(u,v),z(u,v))$, onde u e v formam o sistema de eixos no espaço paramétrico da superfície, define uma representação paramétrica da superfície (Figura 2.5).

O MG adota a forma paramétrica para tirar proveito de algumas de suas potencialidades e por esta forma ser a mais adequada na representação de objetos formados por seções transversais. A descrição paramétrica de superfícies com geometria arbitrária é utilizada na geração de malhas triangulares nestas superfícies através de algoritmo proposto por Miranda [26]. A superfície 3D é mapeada para uma superfície 2D (representação paramétrica da superfície), realizando-se então a triangulação bidimensional, corrigindo-se as distâncias e os ângulos distorcidos na transformação da malha do espaço 3D para o espaço paramétrico. Em seguida, a malha resultante é reconduzida ao espaço 3D. O algoritmo de interseção originalmente implementado no MG também tirava proveito da representação paramétrica das superfícies, sendo baseado na interseção entre malhas sobre as superfícies paramétricas[6,8].

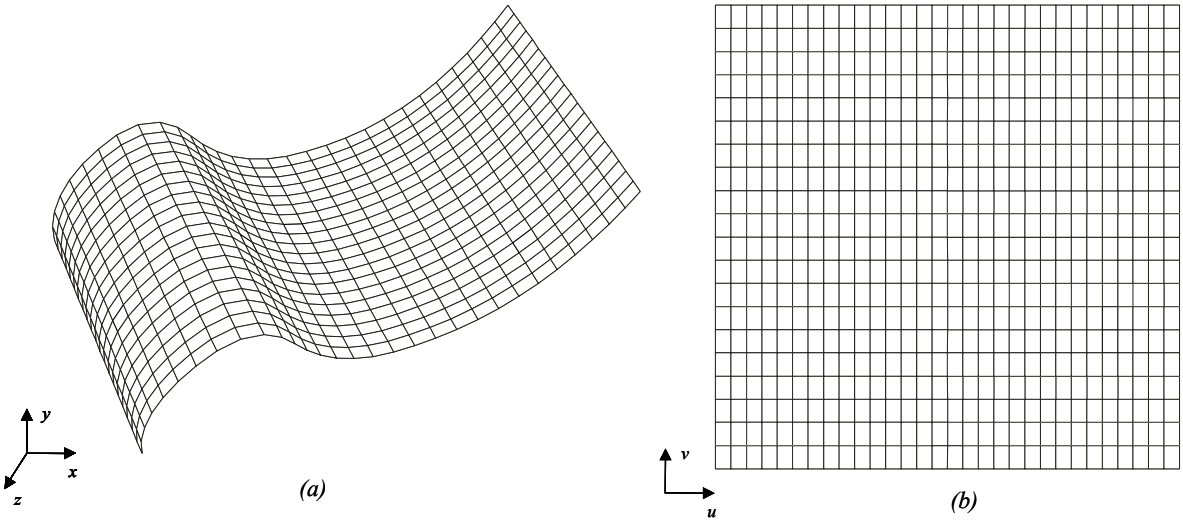


Figura 2.5 – Representações de uma superfície: a) espaço Euclidiano; b) espaço paramétrico [6].

2.3.2. Descrição geométrica de curvas e superfícies usando NURBS

A partir do trabalho de Lira [6], o MG passou a incorporar as representações de curvas e superfícies conhecidas como NURBS (*Non Uniform Rational B-Splines*) [27,28]. Segundo Lira, algumas vantagens deste tipo de representação são:

- NURBS provê uma base matemática única para representação das formas analíticas, tais como seções cônicas e superfícies quadráticas, bem como superfícies com formas quaisquer (por exemplo, cascos de navios ou carenagem de carros);
- a modelagem usando NURBS é intuitiva: quase todas as ferramentas e algoritmos geométricos possuem interpretação de fácil compreensão;
- as superfícies e curvas NURBS são invariantes quando submetidas a transformações geométricas afins (translação, rotação, projeções, etc.);
- as superfícies e curvas NURBS são generalizações de superfícies e curvas *B-Splines* e *Béziers*.

Um exemplo de superfície NURBS pode ser visto na Figura 2.6. A definição formal e as propriedades de curvas e superfícies NURBS podem ser encontradas em [27] e [28]. Para os objetivos deste trabalho, não é necessária uma descrição matemática completa deste tipo de geometria.

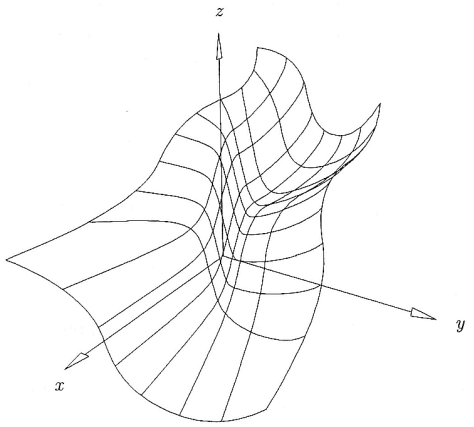


Figura 2.6 – Exemplo de superfície NURBS [6].

Diversas bibliotecas computacionais implementam as representações de curvas e superfícies do tipo NURBS, como por exemplo Nlib [29], *SOLID*S++ [30] e NURBS++ [31]. Grande parte destas bibliotecas é utilizada em empresas ou em programas comerciais de modelagem, o que garante uma certa confiabilidade aos seus usuários. No trabalho de Lira [6], a biblioteca NURBS++ foi incorporada ao MG. Esta biblioteca é implementada em linguagem C++, seguindo o conceito de programação orientada a objetos. Isto representa uma grande vantagem, pois permite a expansão da estrutura de classes. A disponibilização do código fonte também torna possível a implementação de novas funcionalidades na biblioteca. Algumas destas funcionalidades foram implementadas por Lira em seu trabalho.

A biblioteca NURBS++ possui duas classes básicas, uma contendo informações necessárias para a definição de uma curva NURBS e outra contendo informações para a definição de uma superfície NURBS. As Figuras 2.7 e 2.8 resumem a descrição destas duas classes.

Um objeto da classe que representa uma curva NURBS armazena o grau de interpolação da curva, o vetor de *knots* e os seus pontos de controle (ver [27] e [28] para uma descrição completa destes elementos). Existem algumas maneiras diferentes de se criar objetos desta classe. Uma delas é através de um construtor (método de uma classe que cria um objeto) padrão, fornecendo-se as informações descritas acima. Outras maneiras permitem a criação de diversos tipos de curvas conhecidas, como arcos, retas e *splines*, a partir de informações básicas e mais intuitivas em modelagem, como por exemplo e centro, o ponto inicial e o ponto final de um arco.

Um objeto da classe que representa uma superfície NURBS armazena os graus de interpolação da superfície nas duas direções paramétricas *u* e *v*, o vetor de *knots* em ambas as direções e os seus pontos de controle, armazenados de forma matricial. Os construtores desta classe são semelhantes aos descritos para a classe de curvas NURBS, ou seja, pode-se utilizar um construtor padrão ou outros que instanciam parâmetros de curvas mais conhecidas, como *sweep*, *Gordon* e revolução.

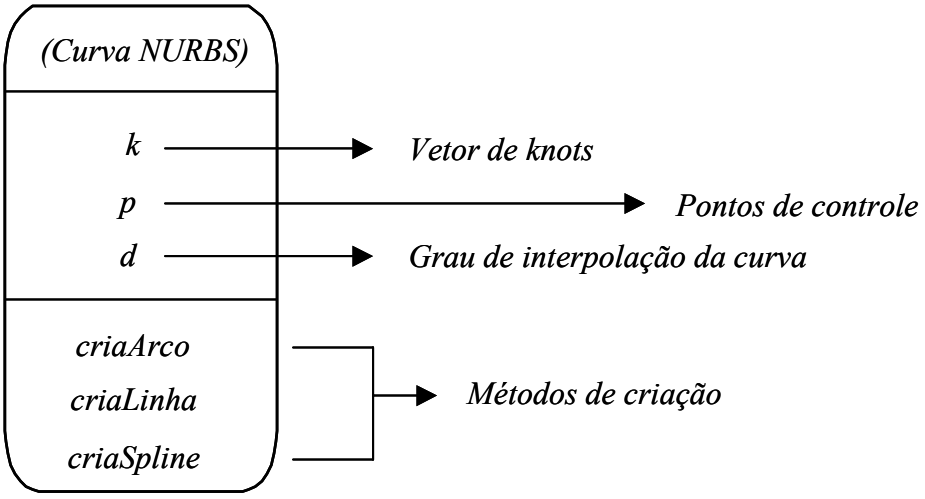


Figura 2.7 – Representação de curvas na biblioteca NURBS++ [6].

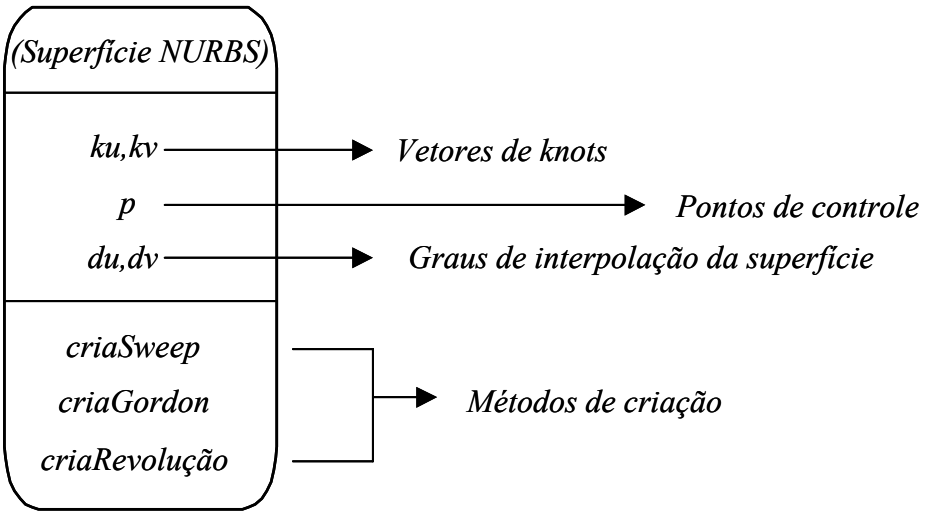


Figura 2.8 – Representação de superfícies na biblioteca NURBS++ [6].

O uso de NURBS no MG foi motivado sobretudo pela sua capacidade de representar o espaço paramétrico das superfícies. Em seu trabalho, Lira [6] incorporou a biblioteca NURBS++ no MG através da criação de uma hierarquia

de classes para representar os tipos de curvas e superfícies utilizados pelo modelador. Foram criadas classes que gerenciam a interface entre o modelador MG e a biblioteca NURBS++, que possuem referências para o objeto NURBS correspondente. Através deste, pode-se acessar os dados das curvas ou superfícies armazenados nas classes correspondentes na biblioteca NURBS++. As classes derivadas destas representam diretamente os tipos de curvas e superfícies implementados por Lira no MG. Essas são as classes que podem ser instanciadas pelo usuário através da interface do MG. As Figuras 2.9 e 2.10 ilustram a organização destas classes.

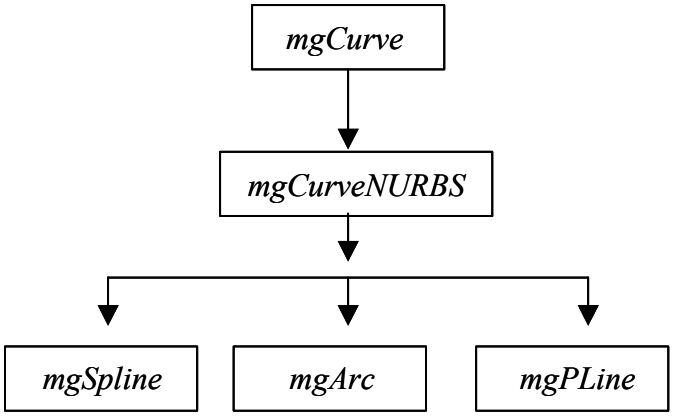


Figura 2.9 – Organização das classes de curvas no modelador MG [6].

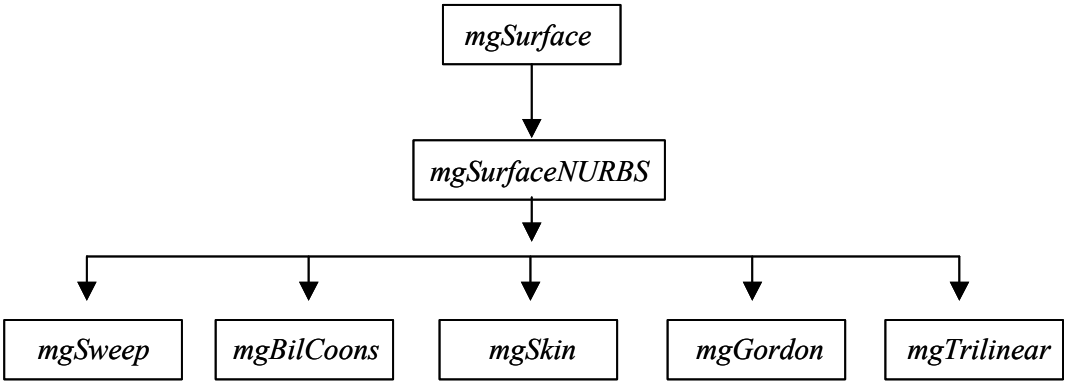


Figura 2.10 – Organização das classes de superfícies no modelador MG [6].

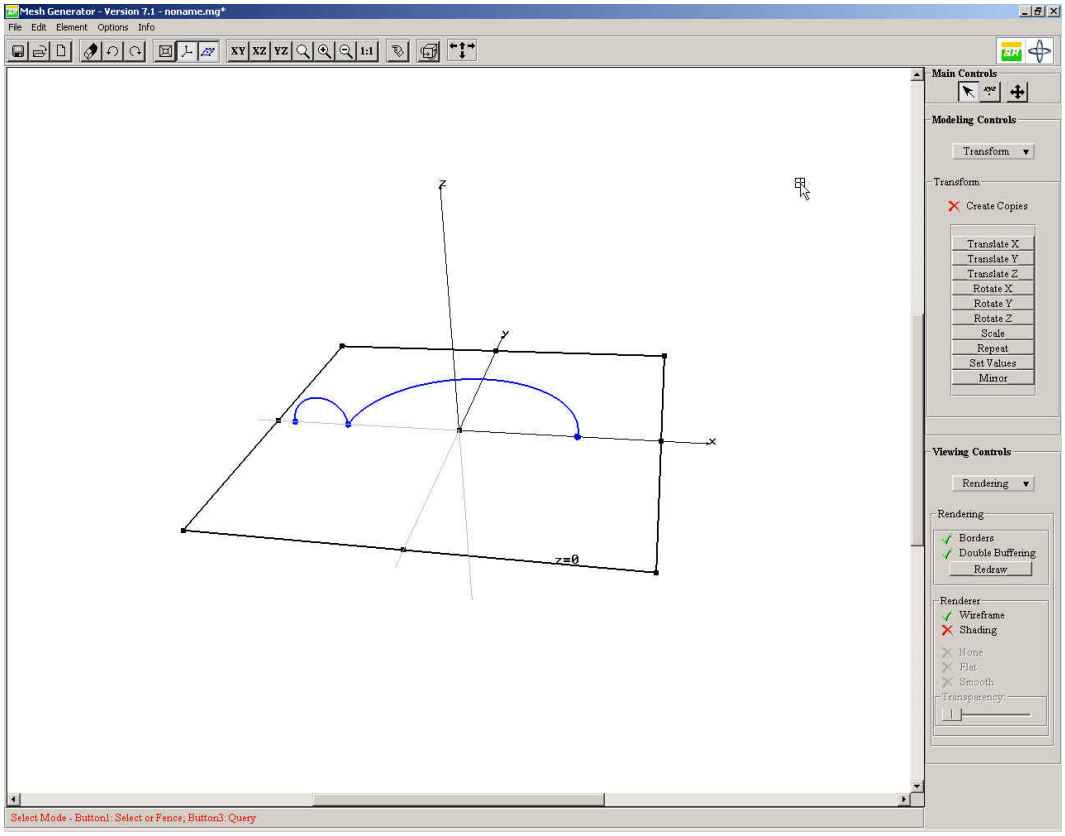
2.3.3. Geração de curvas e superfícies no MG

Para que uma nova curva possa ser criada no ambiente de modelagem do MG, o usuário deve escolher a *forma* da curva que deseja inserir, dentre três opções possíveis: linha poligonal, arco ou *B-Spline*. Estando o programa no modo de interface de criação de curvas, basta ao usuário posicionar o cursor na

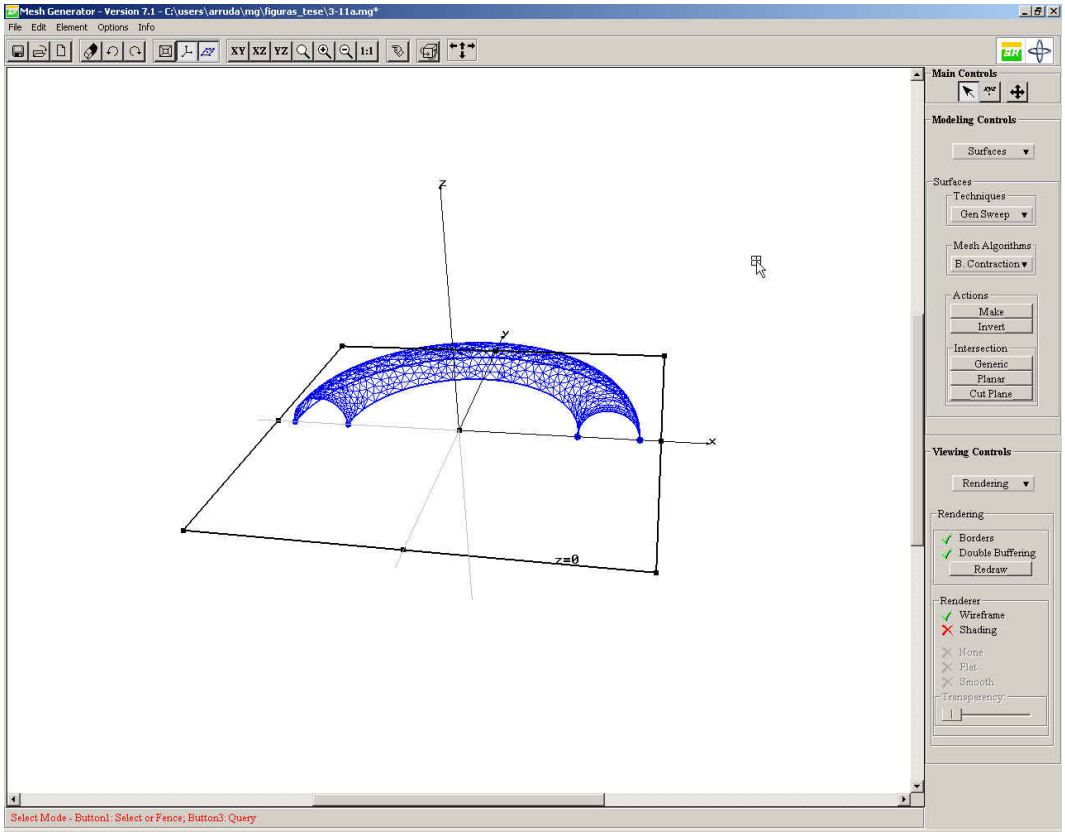
localização desejada dentro da área de desenho da interface para fornecer os pontos necessários para a descrição geométrica da curva. As questões referentes à *atração* e *tolerância*, já discutidas, são aplicáveis neste modo de interface.

Ao final da edição, os pontos são passados para a biblioteca NURBS++ que gera a curva NURBS correspondente. A estrutura de dados do MG armazena o ponteiro para esta curva juntamente com a representação de uma poligonal equivalente [32] adaptativa associada a ela. Esta poligonal é utilizada, por exemplo, em eventos de seleção e desenho destas curvas.

A criação de superfícies é feita utilizando-se as curvas existentes. O usuário escolhe um método de construção de superfícies, pré-seleciona as curvas que definem completamente a superfície desejada e cria a mesma pressionando um botão na interface. Mais uma vez, o objeto NURBS correspondente é criado na biblioteca NURBS++ e uma referência para este objeto é armazenada no MG juntamente com uma discretização da superfície (malha de elementos finitos), utilizada em eventos de seleção e desenho das superfícies. A Figura 2.11 mostra a criação de um toro usando a técnica de *sweep*.



(a)



(b)

Figura 2.11 – Criação de um toro através da técnica de *sweep*: a) curvas bases; b) superfície gerada.

A seguir são apresentadas resumidamente algumas formulações e características dos tipos de superfícies existentes no MG [6]. Nos livros de Piegl e Tiller [28] e Farin [33] encontram-se informações mais detalhadas sobre as descrições matemáticas destas superfícies.

Superfícies *bilineares* são completamente definidas por um circuito de quatro pontos. Uma representação NURBS dessa superfície é obtida pela interpolação linear entre os segmentos formados por estes pontos.

Superfícies *skin*, também conhecidas como *loft*, são aquelas obtidas pela interpolação de um conjunto de curvas ao longo de uma direção. Essas curvas representam as seções transversais da superfície gerada e não são necessariamente planas. Na direção dos bordos livres destas superfícies, uma interpolação linear é feita pela biblioteca NURBS++, gerando uma superfície linear ao longo dessa direção. As curvas geradoras fornecidas são respeitadas na construção da superfície. Assim, a superfície criada tem estas curvas como suas seções transversais (Figura 2.12). Este tipo de superfície é importante, pois serve como base para a geração de outros tipos de superfície, como será visto adiante.

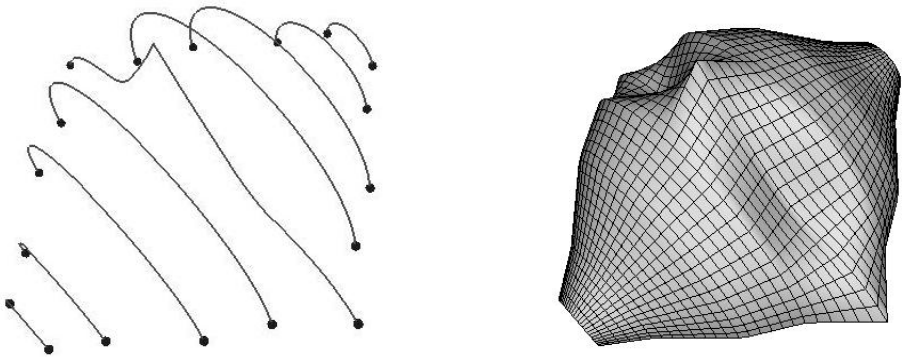


Figura 2.12 – Superfície gerada por *skin*.

Superfícies de *coons* são extensões das superfícies bilineares. Este tipo de representação usa os pontos definidos ao longo dos segmentos do contorno como pontos de controle da superfície. Não existe limitação quanto ao grau de interpolação das curvas do contorno que definem as superfície de *coons*. No MG, a construção deste tipo de superfície é realizada a partir da definição das quatro curvas de bordo que limitam a superfície. Essas curvas podem ser de qualquer tipo, não necessariamente retas, como pode ser visto na Figura 2.13.

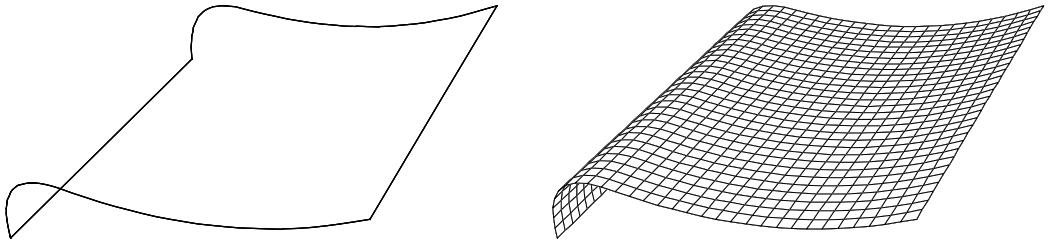


Figura 2.13 – Superfície *coons* [6].

Superfícies geradas por *sweeps* são obtidas pelo arrasto de uma seção curva ao longo de uma trajetória, também curva. No MG, estas superfícies podem ser criadas de três formas diferentes: através de um *sweep translacional*, indicando-se a curva base para o *sweep* e a direção de propagação; através de um *sweep rotacional*, rotacionando-se a curva base em torno de uma direção; ou através de um *sweep genérico*, onde a curva base é arrastada ao longo de uma outra curva que define a trajetória do *sweep* (Figura 2.14).

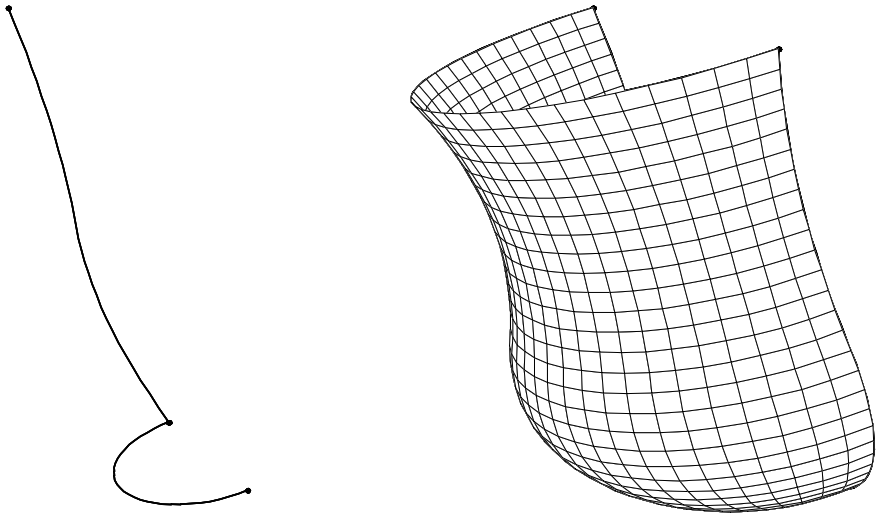


Figura 2.14 – Superfície gerada por sweep genérico [6].

Superfícies de *Gordon* são aquelas geradas por uma rede bidirecional de curvas espaciais (Figura 2.15). No MG, a construção deste tipo de superfície é realizada a partir de uma seqüência de curvas espaciais definidas pelo usuário. Deve-se indicar quais são as curvas geradoras em cada uma das direções paramétricas da superfície. Elas funcionam como seções transversais, em ambas as direções, da superfície gerada. Este tipo de superfície é uma generalização das superfícies de *coons*. Enquanto as superfícies de *coons* são geradas exatamente por uma rede de duas curvas em cada direção, as superfícies de *Gordon* não possuem tal limitação.

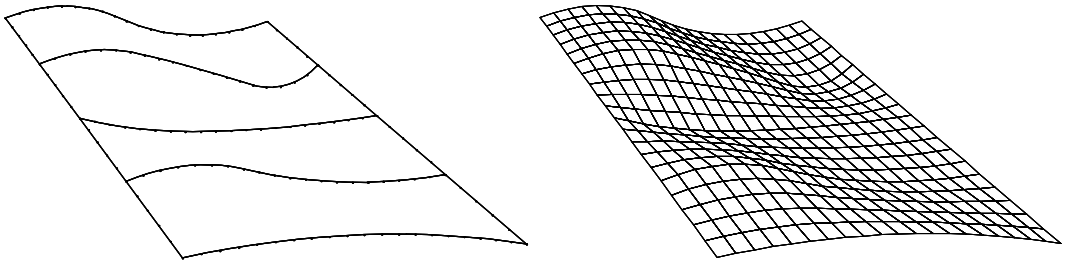
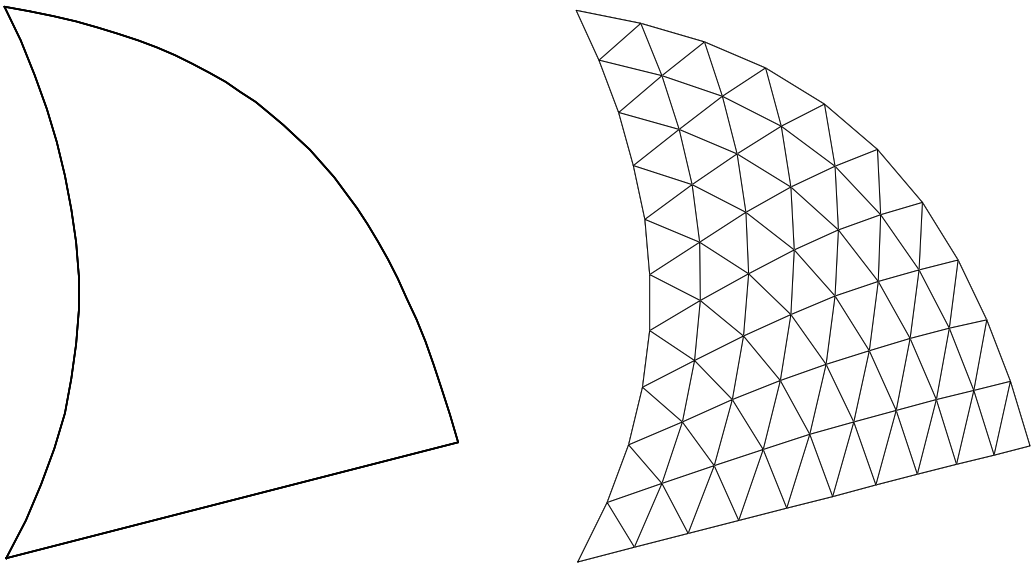


Figura 2.15 – Superfície de *Gordon* [6].

Superfícies *triangulares* são definidas por um circuito de três curvas. A representação dessas superfícies utiliza os pontos localizados ao longo dos segmentos do contorno como seus pontos de controle (Figura 2.16). Em seu trabalho, Lira [6] buscou solucionar um problema que dificultava a representação deste tipo de superfície no ambiente de modelagem do MG. Como não se conhece uma formulação matemática NURBS para representar superfícies triangulares, Lira expandiu a biblioteca NURBS++ de forma que superfícies triangulares pudessem ser aproximadas por superfícies cúbicas do tipo *Bézier* [34]. Contudo, esta implementação apresenta restrições, pois nem todas as curvas geométricas usadas na definição de uma superfície triangular podem ser representadas por curvas *Bézier*. A construção deste tipo de superfície no MG é realizada a partir da definição das três curvas de bordo que limitam a superfície.



(a) *(b)*

Figura 2.16 – Superfície triangular [6].

2.3.4.
Geração de sólidos no MG

No MG, as técnicas utilizadas para modelagem de sólidos confundem-se com o próprio mapeamento associado a cada sólido (malha 3D de elementos finitos). A geração do sólido é feita pela geração da malha que o representa [26].

Quatro técnicas são utilizadas no MG na geração de sólidos. Três delas utilizam técnicas de *sweep*, arrastando-se uma seção transversal bidimensional (retalho de superfície) ao longo de uma trajetória no espaço. A outra descreve uma metodologia para geração de malhas sólidas, não-estruturadas, em domínios arbitrários. Recomenda-se a leitura dos trabalhos de Miranda [26,35] para um maior detalhamento destas técnicas.

Vale ressaltar que no MG, dois sólidos adjacentes devem ter pelo menos um retalho de superfície comum a ambos, localizado na interface entre eles. Essa necessidade exige que retalhos de superfícies sejam criados automaticamente pelo MG para delimitar o contorno dos sólidos gerados e garantindo a consistência topológica nesta etapa da modelagem.

O *sweep translacional*, também chamado de *extrusão*, é uma técnica em que o arrasto da seção transversal do sólido é feito ao longo de uma linha reta no espaço tridimensional (Figura 2.17). Os elementos finitos gerados podem ser pentaédricos ou hexaédricos.

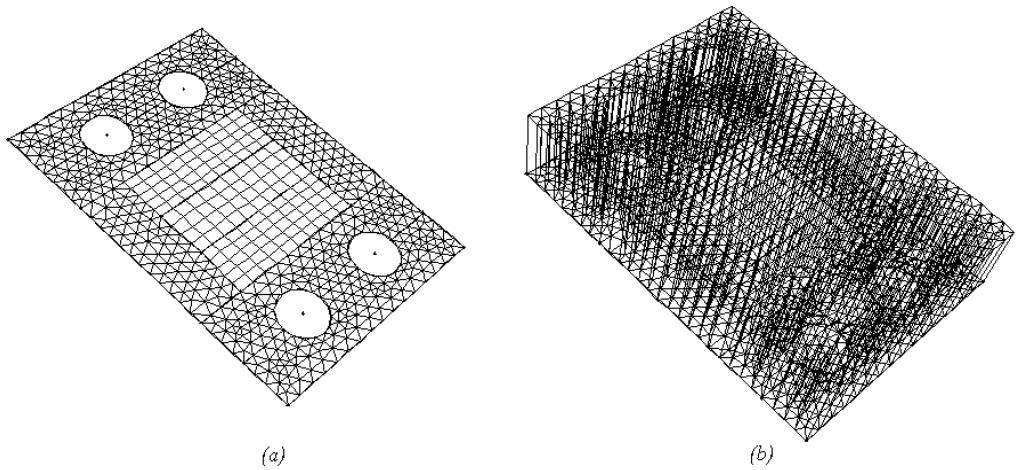


Figura 2.17 – Mapeamento de sólido por extrusão [6].

O *sweep curvo* é uma técnica semelhante à anterior, com a diferença que o arrasto da seção transversal é feito ao longo de uma curva qualquer no espaço tridimensional, que define a trajetória do *sweep* (Figura 2.18). A malha associada

à superfície pode ser formada por elementos triangulares ou quadrilaterais. Os elementos finitos sólidos podem ser pentaédricos ou hexaédricos.

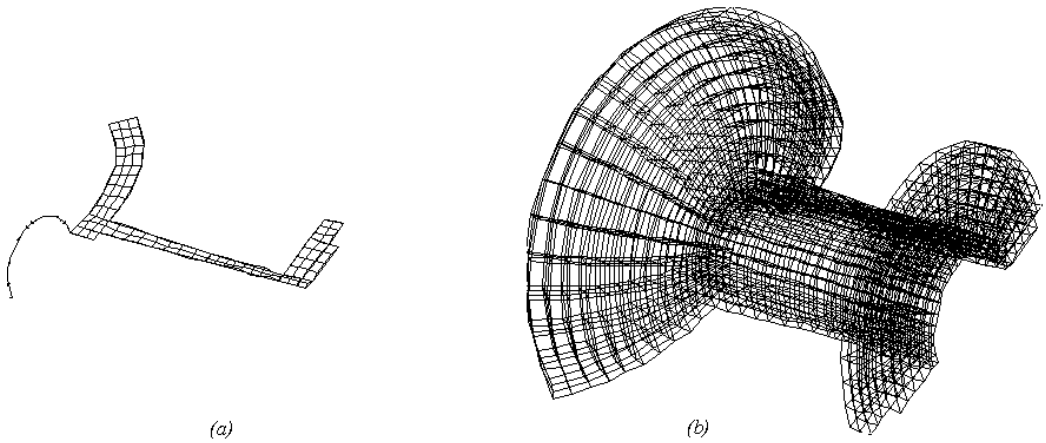


Figura 2.18 – Mapeamento de sólido por sweep curvo [6].

O *mapeamento transfinito tridimensional* consiste em uma técnica para a construção automática de malhas sólidas a partir das seções transversais de um modelo a ser discretizado em elementos finitos (Figura 2.19). Essas seções transversais devem estar ligadas por uma curva que permita identificar os pontos bases de cada seção transversal (pontos onde as curvas e seções transversais se interceptam) e o número de seções intermediárias entre duas seções, informações necessárias para o algoritmo de geração deste tipo de malha sólida. Mais uma vez, os elementos sólidos gerados podem ser pentaédricos ou hexaédricos.

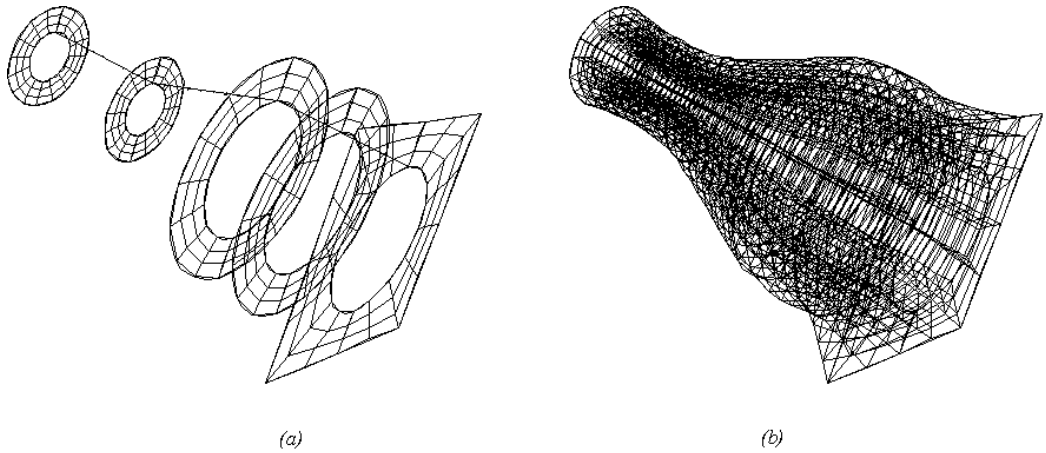


Figura 2.19 – Sólido gerado por mapeamento transfinito tridimensional [6].

A geração de *malhas sólidas em domínios arbitrários* utiliza um algoritmo para geração de malhas volumétricas não-estruturadas de tetraedros para

domínios arbitrários [25,36]. As superfícies que delimitam os sólidos gerados constituem a entrada de dados para o algoritmo responsável pela geração de tais sólidos. As malhas associadas a cada superfície são convertidas em uma representação única de elementos finitos e é verificado o fechamento de uma região com estas superfícies. Se uma região fechada for detectada, é gerada a malha tetraédrica na região correspondente (Figura 2.20). A próxima seção faz um detalhamento do algoritmo de geração de malhas de tetraedros.

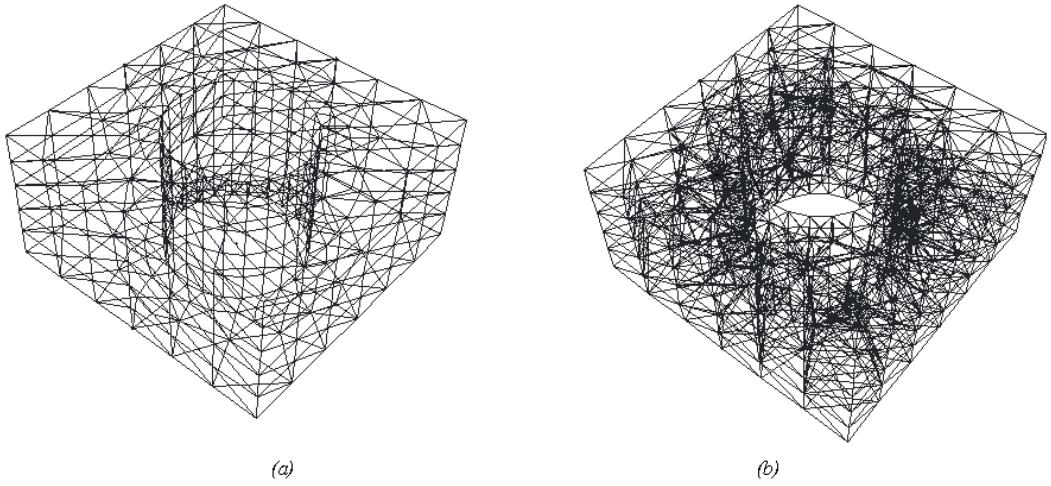


Figura 2.20 – Geração de sólido com domínio arbitrário [6].

2.3.5. Geração de Malhas Não-Estruturadas no MG

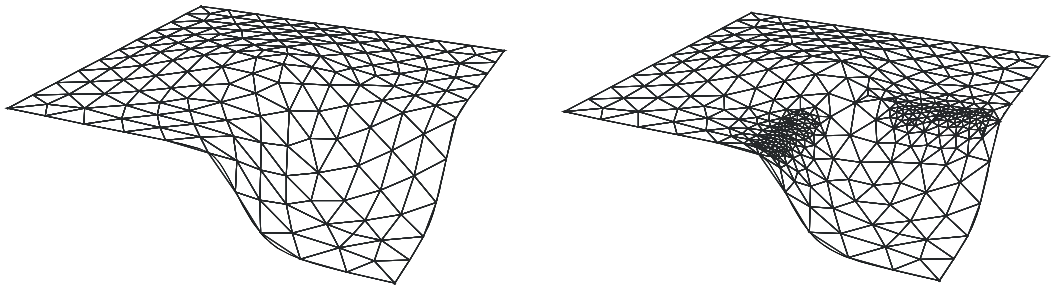
Os algoritmos utilizados no MG para geração de malhas não-estruturadas volumétricas e de superfícies combinam técnicas de avanço de fronteira [37,38] com decomposição espacial recursiva [39,40]. Estes algoritmos usam um procedimento que procura melhorar localmente a qualidade dos elementos gerados. Além disso, esses algoritmos apresentam boa transição entre as malhas de regiões com diferentes tamanhos de elementos.

Tais algoritmos devem gerar malhas que sejam conformes com as discretizações existentes no domínio, para que se possam gerar localmente novas malhas em processos adaptativos. Além disso, eles devem tratar casos onde as superfícies possuem curvaturas acentuadas, refinando a malha localmente nas regiões onde ocorrem estas curvaturas e evitando assim que elementos vizinhos formem locais pontiagudos.

2.3.5.1.
Geração de malhas em superfícies

O algoritmo implementado no MG para geração de malhas não-estruturadas em superfícies foi desenvolvido por Miranda [26]. Utilizando o espaço paramétrico da superfície, a superfície 3D é mapeada para uma superfície 2D e então a triangulação é realizada com correções nas distorções geométricas. Em seguida, a malha resultante dessa triangulação é reconduzida para o espaço 3D. A Figura 2.21 ilustra uma malha em superfície 3D.

Os dados de entrada para geração de malhas de superfícies utilizando este algoritmo são uma lista de nós definidos por suas coordenadas paramétricas e uma lista com o número de arestas em cada *loop* do modelo.



(a) *(b)*

Figura 2.21 – Exemplo de malha em uma superfície 3D: a) sem considerar distorções; b) considerando distorções.

2.3.5.2.
Geração de malhas volumétricas

No MG, um algoritmo desenvolvido por Cavalcante Neto [25,36] é utilizado para gerar malhas volumétricas não-estruturadas através de uma técnica de avanço de fronteira combinada com uma técnica de decomposição espacial recursiva, como no caso das malhas superficiais.

O algoritmo permite a representação de geometrias quaisquer, incluindo furos, cavidades e trincas, e está dividido em três passos. O primeiro é a geração de uma *octree*, utilizada para desenvolver diretrizes locais usadas para definir o tamanho dos elementos a serem gerados a partir do tamanho dos elementos triangulares na malha de contorno fornecida como dado de entrada. O segundo passo é o avanço de fronteira, que por sua vez é dividido em três etapas: geração de elementos baseada na geometria, geração de elementos

baseada na topologia e *backtracking*, usado para eliminar algumas faces dos elementos que estão impedindo o algoritmo de completar a malha.

O último passo consiste numa melhoria local da malha gerada pelo procedimento de avanço de fronteira. Ele é dividido em duas etapas: a primeira é uma técnica de suavização convencional pela realocação de nós baseado na média das coordenadas nodais, com testes de validação. A segunda etapa consiste no procedimento de *backtracking*, que remove faces de elementos de forma ruim para criar uma região onde elementos com melhor forma possam ser gerados.

2.3.6. Criação de grupos

A criação de grupos é uma ferramenta nova no MG, implementada durante o desenvolvimento deste trabalho. Além de outras funcionalidades, a existência de grupos no MG é essencial para a implementação das operações booleanas neste modelador, para que se possa atingir o nível de generalização e abrangência desejados.

De uma forma geral, um *grupo* é um conjunto de entidades topológicas formando um sub-domínio do modelo em estudo. Grupos podem conter vértices, arestas, retalhos de superfícies 2D (*patches 2D*) e regiões (*patches 3D*). Como não há nenhuma restrição quanto à seleção das entidades que farão parte de um grupo, um mesmo grupo pode conter várias entidades com dimensões diferentes, inclusive entidades pendentes ou totalmente “soltas” no espaço, como faces que não pertencem ao contorno de nenhuma região, arestas que não pertencem ao contorno de nenhuma face ou vértices sem arestas incidentes. A partir de uma entidade topológica pertencente a um grupo podem-se pesquisar as suas relações de adjacência para se obter todos os elementos adjacentes a esta entidade.

No MG, os grupos são representados por meio de uma classe *Group*. Os objetos desta classe contêm uma lista encadeada que guarda as entidades topológicas pertencentes àquele grupo. Cada grupo possui uma identificação (*label*) e não há restrições quanto à presença de uma mesma entidade topológica em dois ou mais grupos distintos. Uma outra classe, *GroupItf*, é responsável pelo gerenciamento da interface. Ela faz a comunicação entre a o modelador e o usuário na criação e manipulação de grupos. Esta classe é responsável pela criação de uma árvore na interface do programa para exibir de

forma organizada os grupos e suas entidades. Mais detalhes sobre estas classes e seus métodos serão apresentados no capítulo que descreve a implementação das operações booleanas no MG.

A criação ou remoção de um grupo no MG é feita através de sub-menus na interface do programa. No momento da criação, o usuário escolhe um *label* para o grupo. A seleção das entidades que farão parte de um grupo é realizada da forma convencional, no modo de seleção. Pode-se selecionar uma entidade de cada vez e acrescentá-la ao grupo escolhido, ou adicionar todas as entidades que farão parte daquele grupo de uma só vez. As Figuras 2.22 e 2.23 ilustram algumas destas funcionalidades.

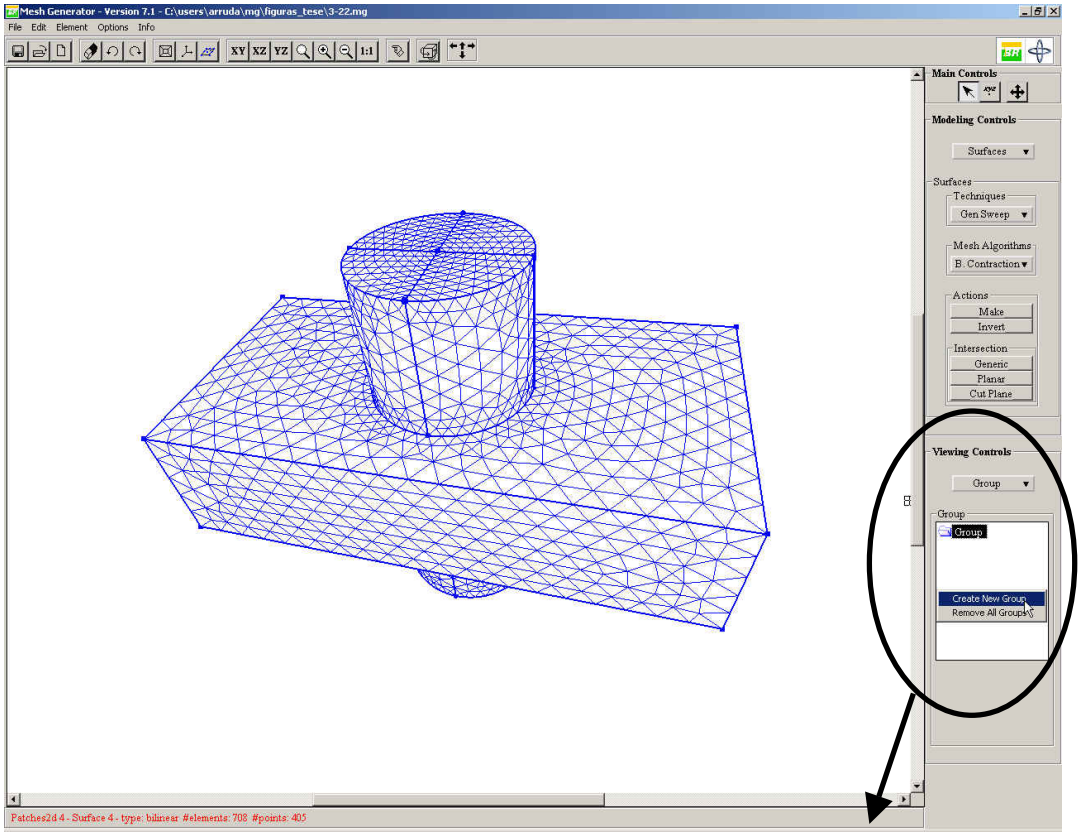
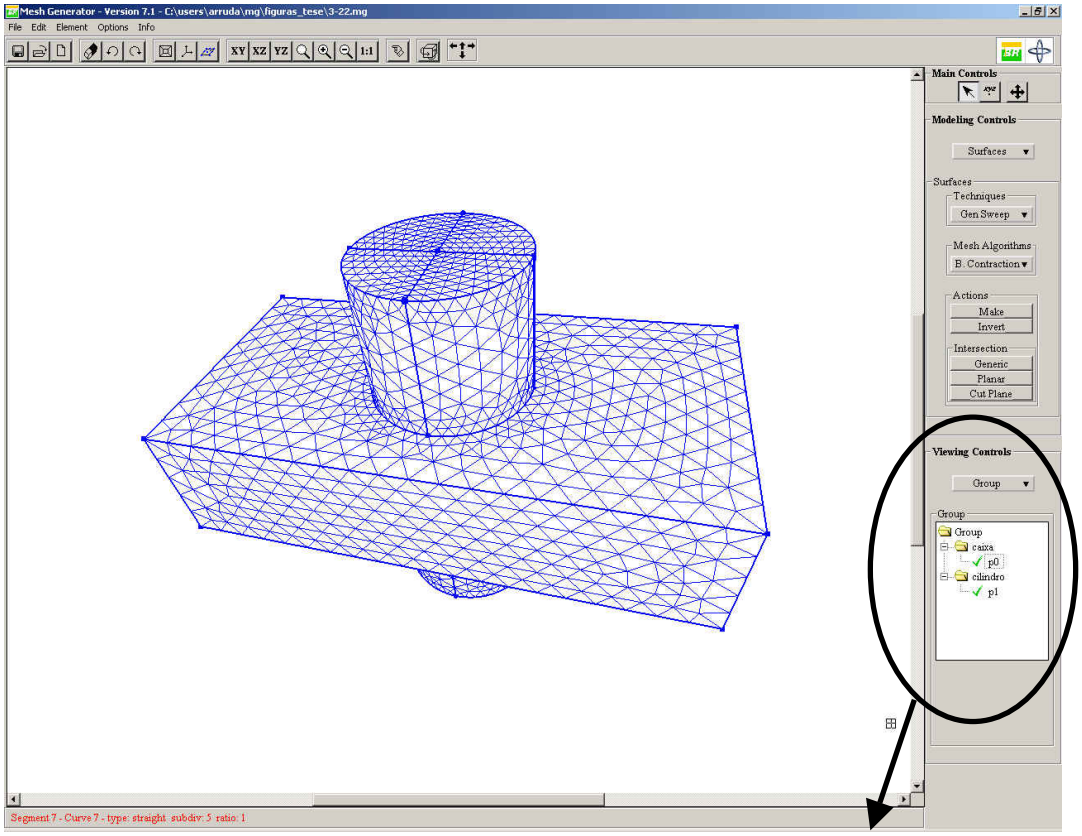


Figura 2.22 – Criação de um novo grupo no MG.



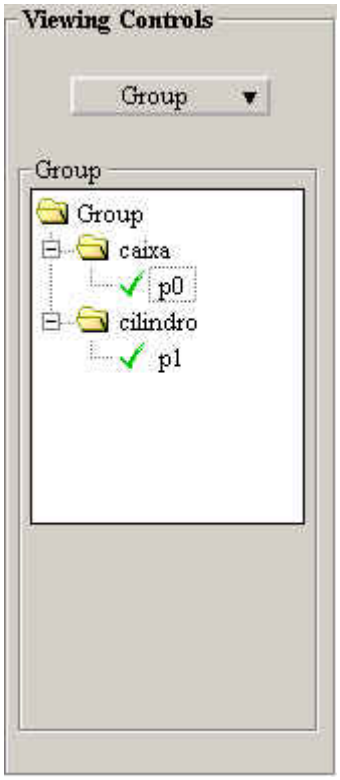


Figura 2.23 – Árvore de grupos com suas entidades topológicas.

2.3.7. Atributos

O ESAM, sistema de configuração e gerenciamento de atributos utilizado pelo MG, utiliza uma abordagem conhecida como modelagem baseada em geometria, onde os atributos são associados às entidades geométricas do modelo e a discretização dessa geometria (malha de elementos finitos) herda, automaticamente, estes atributos [6]. Esta abordagem oferece várias vantagens, como um melhor suporte para a automação do processo de modelagem, incluindo geração automática de malha e análise adaptativa, e a simplificação da geração do modelo para a simulação.

Uma descrição completa do sistema ESAM pode ser vista em [41,42], inclusive com as especificações necessárias à sua utilização.

2.3.8. Estrutura de dados híbrida

A partir do trabalho de Lira [6], o MG passou a adotar um enfoque híbrido de modelagem, baseado na combinação da representação CGC (*Complete Geometric Complex*) com a estrutura de dados do modelador, onde a representação CGC não é mantida durante todo o processo de modelagem. Um modelo CGC passou a poder ser criado em qualquer instante, quando solicitado pelo usuário para realizar a detecção automática de regiões. A criação do modelo CGC apenas nesta etapa de modelagem é devida à dificuldade de se alcançar um grau razoável de eficiência na interface com o usuário se a consistência entre geometria e topologia for forçada após cada tarefa realizada pelo usuário.

Cavalcanti [16,19] propôs um enfoque *non-manifold* para modelagem de multi-regiões. Uma metodologia geral para a criação e manipulação de uma subdivisão espacial em células com forma e geometria arbitrária foi desenvolvida. A subdivisão espacial pode ser criada através da inserção, uma a uma, de retalhos de superfícies planas, permitindo a inserção de novos retalhos em tempo real. O objeto resultante desta decomposição é classificado como um CGC. A representação CGC utilizada no MG é implementada como uma biblioteca de classes C++. Nesta representação, foi adotada a estrutura de dados *Radial Edge* (RED), proposta por Weiler [17,18] e já discutida neste trabalho. As classes da biblioteca CGC provêm operadores de alto nível que manipulam uma subdivisão espacial. Eles recebem como dados de entrada

informações geométricas dos retalhos das superfícies que são inseridas na subdivisão espacial. Estas informações geométricas são automaticamente transferidas para as informações topológicas requeridas pelos operadores *manifold* e *non-manifold* de Weiler.

Lira [6] estendeu a implementação original da CGC, que considerava apenas retalhos de superfícies planares. A partir do seu trabalho, superfícies com geometrias curvas usando NURBS passaram a ser consideradas.

O MG possui então duas representações separadas do mesmo modelo. Uma representação é armazenada na estrutura de dados do modelador. A outra é uma conversão temporária da estrutura de dados do modelador em uma representação CGC. Nessa conversão, a topologia do modelo é determinada tal que diferentes regiões possam ser reconhecidas. A CGC pode ser vista como um “costurador topológico” que gera informações de adjacência de um modelo que é consistente com a sua geometria [6]. As entidades topológicas identificadas pela representação CGC são passadas de volta para a representação MG, incluindo as regiões detectadas automaticamente.

Pode-se resumir a metodologia da modelagem híbrida adotada no MG da seguinte forma [6]:

1. O usuário gera retalhos de superfícies simples que podem se interceptar;
2. O modelador MG determina as interseções usando o algoritmo para interseção de superfícies, gerando novas curvas, segmentos de curvas e retalhos de superfícies;
3. O usuário seleciona os retalhos de superfícies que irão ser usados no modelo final, removendo partes indesejáveis;
4. O módulo CGC identifica as regiões fechadas e retorna essas informações para a estrutura de dados do MG;
5. O modelador MG calcula a malha final combinando as malhas dos retalhos e sólidos individuais.

A informação básica passada da estrutura de dados do modelador para a estrutura de dados da CGC consiste em um conjunto de retalhos de superfícies definidos pelo usuário através da interface gráfica do MG ou resultantes da interseção de superfícies. Na representação CGC, os retalhos de superfícies interceptam-se apenas em suas fronteiras.

Na conversão de volta da estrutura de dados da CGC para a estrutura de dados do MG, percorre-se todas as regiões e faces geradas pelo módulo CGC e transforma-as em entidades da representação MG.

Vale ressaltar que, no passo 3 da metodologia de modelagem descrita, a remoção de partes indesejáveis do modelo era uma tarefa que deveria ser realizada explicitamente pelo usuário. No entanto, após a inclusão das operações booleanas no MG como ferramenta adicional, foco deste trabalho, esta remoção pode ser feita implicitamente através da aplicação de uma ou mais operações booleanas sobre as entidades presentes no modelo. Isto pode ser útil, por exemplo, quando as entidades que se deseja remover são difíceis de serem selecionadas pelo usuário na área de desenho da interface (*canvas*).

2.3.9. A estrutura de dados do modelador MG

A estrutura de dados do MG distingue entidades topológicas de geométricas. Optou-se por não utilizar diretamente a *Radial Edge* como estrutura de dados do MG para não haver a necessidade de manter a consistência entre geometria e topologia em todos os passos de modelagem. Apesar disso, a estrutura de dados do MG consegue representar um modelo *non-manifold* com multi-regiões mantendo as relações de adjacência necessárias. A Figura 2.24 ilustra a organização de classes do modelador MG.

A classe *Entity* é a classe base, da qual se derivam duas outras classes, *Geometry* e *Topology*, responsáveis, respectivamente, pela representação geométrica e topológica do modelo.

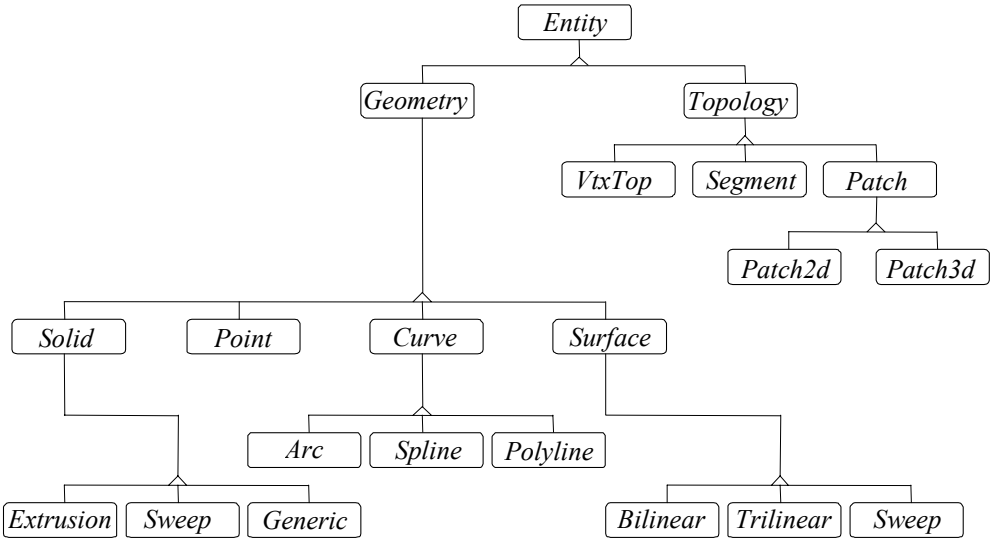


Figura 2.24 – Organização de classes do modelador MG [6].

Cada classe representando uma entidade topológica possui uma classe correspondente que representa a entidade geométrica equivalente. Ou seja, todas as entidades possuem duas instâncias, uma topológica e outra geométrica. A seguir são expostas estas entidades, as classes que as representam e algumas características destas classes:

- *Vértices* / *Pontos*: são representados por uma classe topológica *Vtxtop* e uma classe geométrica *Point*. Um objeto da classe *Vtxtop* armazena um ponteiro para o objeto da classe *Point* correspondente, uma lista de *usos* associados a este vértice (o conceito de *uso* utilizado no MG é diferente daquele introduzido pela *Radial Edge* e será definido mais tarde) e uma lista de segmentos (objetos da classe *Segment*, definida a seguir) incidentes neste vértice. Um objeto da classe *Point* armazena as coordenadas espaciais do ponto e um ponteiro para o objeto da classe *Vtxtop* correspondente. As Figuras 2.25 e 2.26 mostram as relações topológicas e geométricas dos objetos destas duas classes.

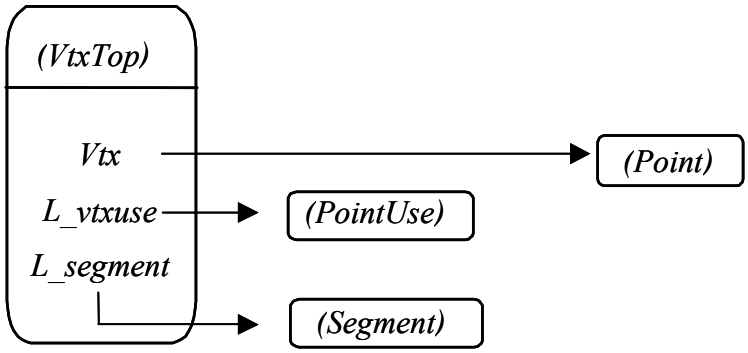


Figura 2.25 – Relações dos objetos da classe *Vtxtop* [6].

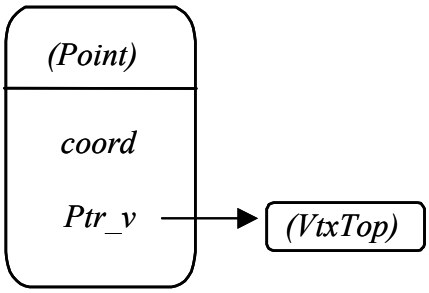


Figura 2.26 – Relações dos objetos da classe *Point* [6].

- *Segmentos* / *Curvas*: são representados por uma classe topológica *Segment* e uma classe geométrica *Curve*. Um objeto da classe *Segment* armazena uma lista de *usos* deste segmento (vale a ressalva feita anteriormente quanto ao conceito de *uso* adotado), uma lista de retalhos de superfície (objetos da classe *Patch2d*) de cujos contornos este segmento faz parte, um ponteiro para o objeto Curve correspondente, ponteiros para os objetos da classe *Vtxtop* que representam os vértices inicial e final deste segmento e dois parâmetros (números reais) que representam os limites extremos do segmento no espaço paramétrico da curva geométrica que o contém. Um objeto da classe *Curve*, por sua vez, armazena uma referência para o objeto NURBS que define a geometria da curva e uma lista de segmentos (objetos da classe *Segment*) que estão ao longo desta curva. As Figuras 2.27 e 2.28 mostram as relações topológicas e geométricas dos objetos destas duas classes.

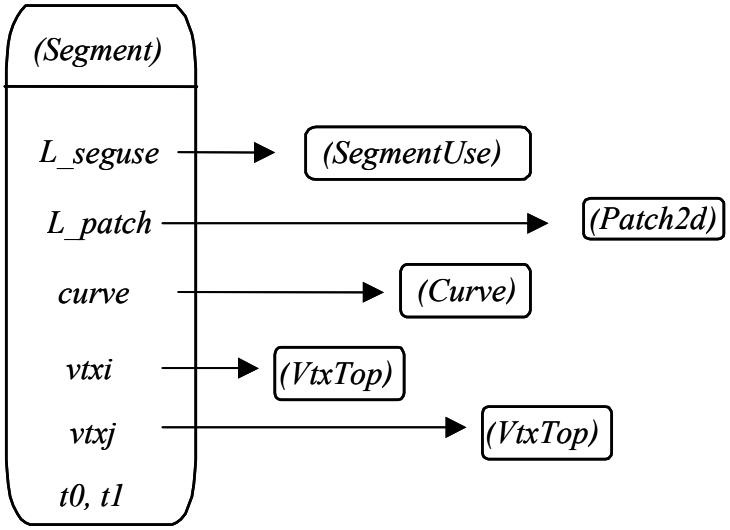


Figura 2.27 – Relações dos objetos da classe *Segment* [6].

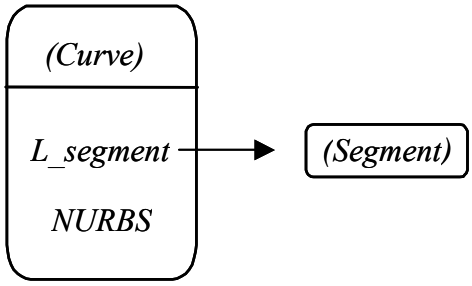


Figura 2.28 – Relações dos objetos da classe Curve [6].

- *Retalhos de superfície* / *Superfícies*: são representados por uma classe topológica *Patch2d* e por uma classe geométrica *Surface*. Um objeto da classe *Patch2d* armazena uma lista de segmentos da sua fronteira (objetos da classe *Segment*), uma lista de regiões sólidas adjacentes (objetos da classe *Patch3d*), um ponteiro para o objeto da classe *Surface* que representa a superfície geométrica que contém este retalho e uma malha de elementos finitos de superfície que eventualmente é gerada sobre ela. Um objeto da classe *Surface* armazena uma lista de retalhos de superfície existentes sobre a superfície geométrica (objetos da classe *Patch2d*), uma lista de curvas geradoras desta superfície (objetos da classe *Curve*), referências sobre seus métodos de criação (*bilinear*, *trilinear* ou *sweep*) e uma referência para o objeto NURBS que define a geometria da superfície. As Figuras 2.29 e 2.30 mostram as relações topológicas e geométricas dos objetos destas duas classes.

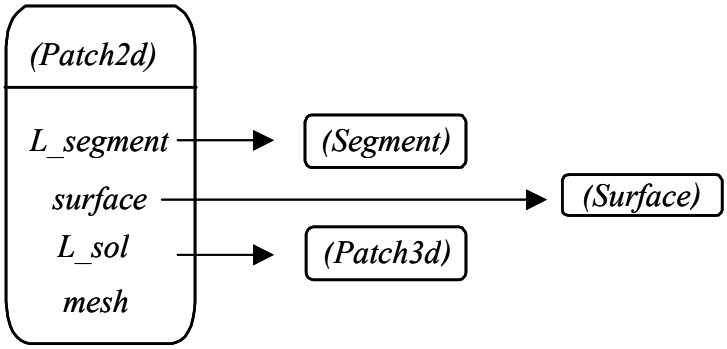


Figura 2.29 – Relações dos objetos da classe *Patch2d* [6].

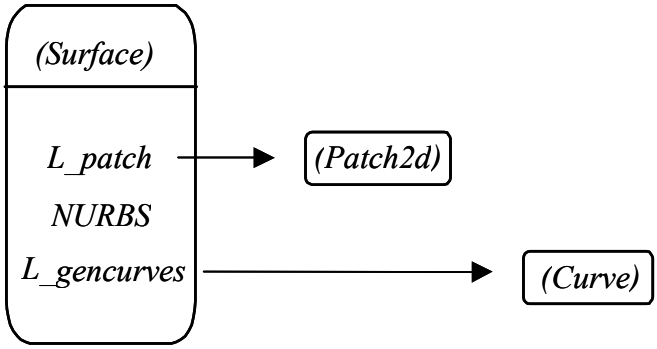


Figura 2.30 – Relações dos objetos da classe *Surface* [6].

- *Regiões* / *Sólidos*: são representados por uma classe topológica *Patch3d* e por uma classe geométrica *Solid*. Um objeto da classe *Patch3d* armazena uma lista de retalhos de superfície que formam o seu contorno (objetos da classe *Patch2d*), um ponteiro para o sólido geométrico correspondente (objeto da classe *Solid*) e uma malha de elementos finitos que eventualmente pode ser gerada sobre ele. Um objeto da classe *Solid* armazena uma lista de regiões ao longo deste sólido (objetos da classe *Patch3d*), listas de curvas e superfícies geradoras deste sólido (objetos das classes *Curve* e *Surface*, respectivamente) e informações sobre o seu método de criação. As Figuras 2.31 e 2.32 mostram as relações topológicas e geométricas dos objetos destas duas classes.

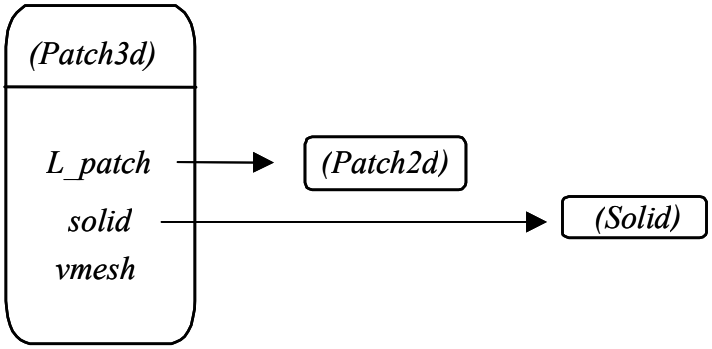


Figura 2.31 – Relações dos objetos da classe *Patch3d* [6].

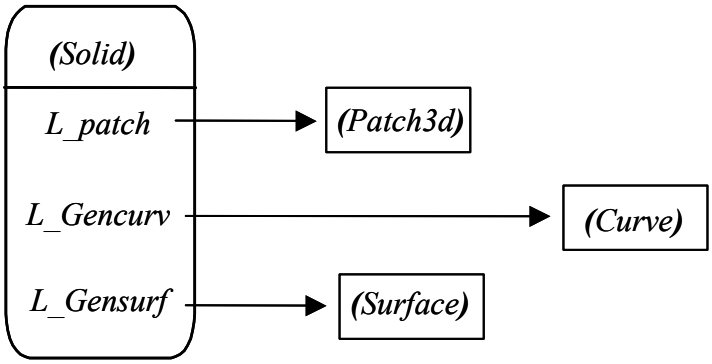


Figura 2.32 – Relações dos objetos da classe *Solid* [6].

A metodologia utilizada no MG na geração de malhas adaptativas é baseada no refinamento de cada entidade topológica no seu espaço paramétrico. Por exemplo, o refinamento dos segmentos do contorno de um retalho de superfície é necessário para a geração de malhas neste retalho. Neste contexto entra o conceito de *uso* na estrutura de dados do MG. O *uso* representa a informação geométrica de uma determinada entidade no espaço paramétrico de uma superfície. As entidades *usos* armazenam as coordenadas paramétricas [6]. Por exemplo, um vértice que está sobre duas superfícies adjacentes tem dois *usos*. A Figura 2.33 ilustra este exemplo. No caso de segmentos, cada *uso* de um segmento possui as coordenadas paramétricas dos seus pontos extremos. As classes que representam estes dois tipo de uso são descritas a seguir:

- *VertexUse*: um objeto desta classe possui uma referência para o seu objeto *VtxTop* correspondente e uma outra referência para o objeto *Surface* correspondente. Além disso, possui duas variáveis que armazenam as coordenadas paramétricas do ponto na superfície. Vale lembrar que um objeto da classe *VtxTop* (vértice topológico) possui uma lista de usos associados a ele (objetos da classe *PointUse*).

- *SegmentUse*: um objeto desta classe possui uma referência para o seu objeto *Segment* correspondente e uma outra referência para o objeto *Surface* correspondente. Além disso, possui um vetor com as coordenadas paramétricas do segmento na superfície correspondente. Vale lembrar que um objeto da classe *Segment* possui uma lista de *usos* associados a ele (objetos da classe *SegmentUse*).

As Figuras 2.34 e 2.35 mostram as relações dos objetos destas duas classes.

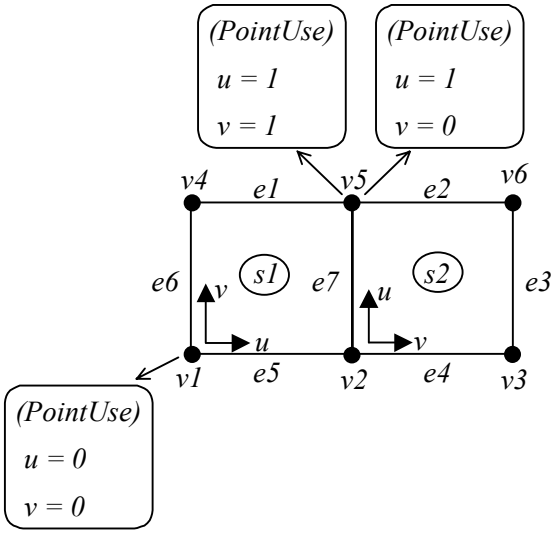


Figura 2.33 – Usos de vértices em duas superfícies adjacentes [6].

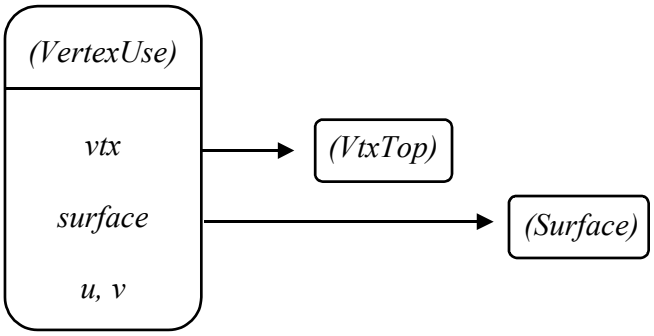


Figura 2.34 – Relações dos objetos da classe *VertexUse* [6].

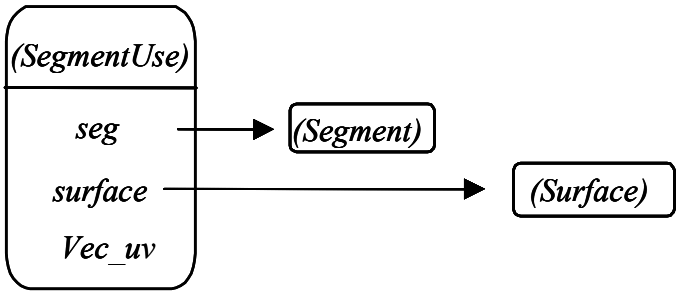


Figura 2.35 – Relações dos objetos da classe *SegmentUse* [6].

2.4. Interseção de superfícies e detecção automática de regiões

Num modelador geométrico que busca promover a simulação 3D de problemas reais de engenharia usando o MEF, a interseção de superfícies e a detecção automática de múltiplas regiões fechadas tornam-se duas ferramentas essenciais. Um número significativo de modelos das mais diversas áreas de engenharia, como estrutural, naval ou mecânica deixariam de poder ser representados, ou teriam de ser representados somente de forma aproximada caso o sistema de modelagem não comportasse tais facilidades.

Um aspecto importante também está relacionado com a definição das malhas sobre cada uma das superfícies do modelo. Dessa forma, o problema de interseção de superfícies também envolve o problema de reconstrução de malhas.

Em sua versão original, o MG já incorporava um algoritmo, proposto por Coelho [8], que resolvia o problema de interseção entre malhas de superfícies. As buscas requeridas para a determinação das curvas de interseção e a reconstrução das malhas eram suportadas por uma estrutura de dados topológica cujas principais características eram a simplicidade e o armazenamento das entidades topológicas em estruturas espaciais *B-trees* [43] e *R*-trees* [44], em vez de listas encadeadas. O uso destas estruturas espaciais aumentava a eficiência do algoritmo.

O algoritmo era baseado em um esquema para interseção de superfícies paramétricas onde as malhas de superfícies existentes eram usadas como suporte para a definição das curvas de interseção. A estrutura de dados auxiliar, definida no espaço paramétrico de cada superfície, permitia a construção das curvas de *trimming* (curvas resultantes da interseção entre superfícies e que possuem representação nos espaços paramétricos destas e no espaço tridimensional) sem a necessidade de se realizar buscas globais.

O algoritmo proposto por Coelho para determinar a interseção entre as malhas das superfícies *A* e *B* tem três passos básicos [6]:

1. Determinação dos pontos de interseção:
   1a) Cálculo e armazenamento das interseções das arestas em A contra as faces em B;
   1b) Cálculo e armazenamento das interseções das arestas em B contras as faces em A.
2. Determinação das curvas de *trimming*:
   2a) Conexão dos pontos de interseção em linhas poligonais representando as curvas de *trimming*;
   2b) Interpolação das curvas paramétricas passando pelos pontos da linha poligonal;
   2c) Determinação de novos pontos com espaçamento adequado nessas curvas;
   2d) Projeção destes novos pontos em cada superfície.
3. Reconstrução da topologia:
   3a) Determinação das regiões de *trimming* removendo vértices e arestas baseadas na linha poligonal;
   3b) Inserção de novas arestas sobre as curvas de *trimming* usando os novos pontos definidos no passo 2c;
   3c) Triangulação das regiões de *trimming* em ambas as superfícies;
   3d) Suavização das malhas.

Este algoritmo pode ser aplicado em uma grande quantidade de situações envolvendo problemas de engenharia, como pode ser visto na Figura 2.36. No entanto, existem alguns casos patológicos que não eram tratados por esta implementação original do algoritmo, tais como a interseção entre superfícies onde uma delas já foi interceptada anteriormente e a nova curva de *trimming* intercepta uma já existente.

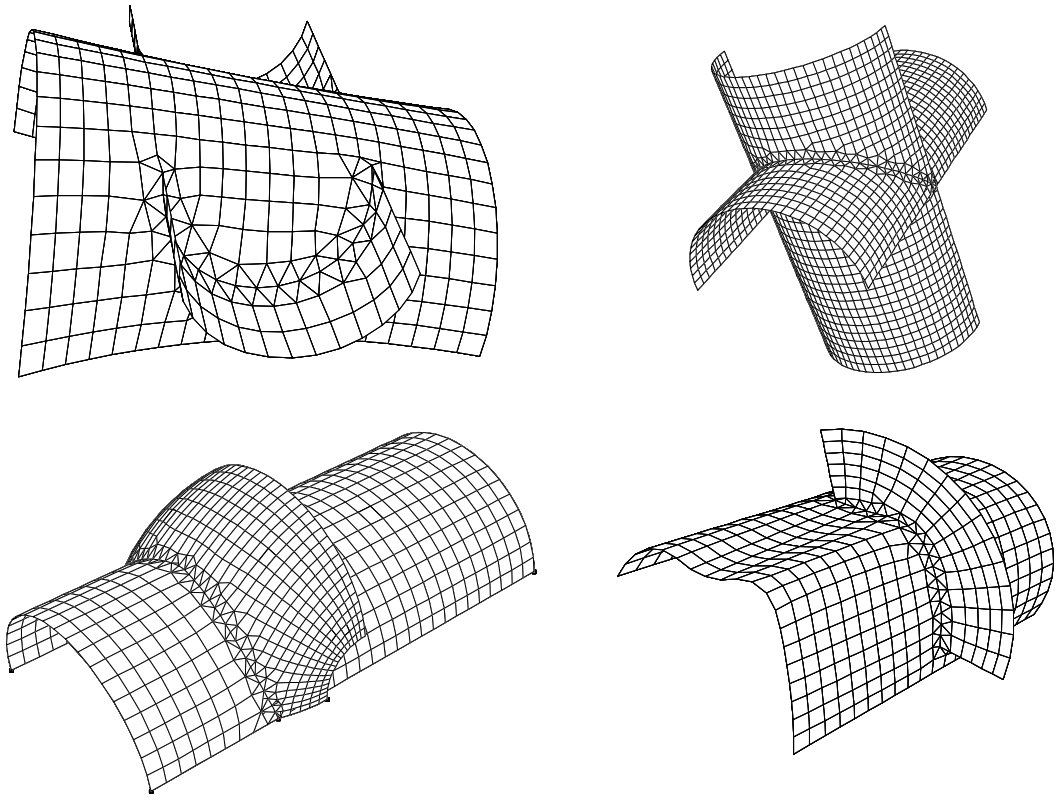


Figura 2.36 – Exemplos de interseções de superfícies [6].

Baseado nestes aspectos, Lira [6] propôs em seu trabalho alterações na implementação original do algoritmo com o objetivo de fornecer mais robustez e confiabilidade ao mesmo. Deve-se observar que no trabalho de Lira as idéias centrais do algoritmo permaneceram inalteradas, estando as principais modificações no algoritmo relacionadas com a sua implementação.

Os casos especiais tratados no trabalho de Lira referem-se às seguintes situações: *interseção aresta*/*aresta*, quando uma aresta de uma superfície intercepta uma aresta da outra superfície; *interseção aresta*/*vértice*, quando a aresta de uma superfície intercepta um vértice da outra; *curvas interceptando curvas já existentes*, quando a curva de *trimming* intercepta uma outra já existente; *interseção de superfícies não-retangulares*, como por exemplo superfícies triangulares; e *reconstrução das malhas em superfícies incidentes às curvas de interseção*. As Figuras 2.37, 2.38, 2.39 e 2.40 ilustram exemplos desses casos patológicos. Para uma descrição completa do algoritmo de interseção originalmente implementado no MG e do tratamento dos casos patológicos, inclusive com detalhes sobre as estruturas de dados utilizadas, recomenda-se a leitura dos trabalhos de Coelho [8] e Lira [6].

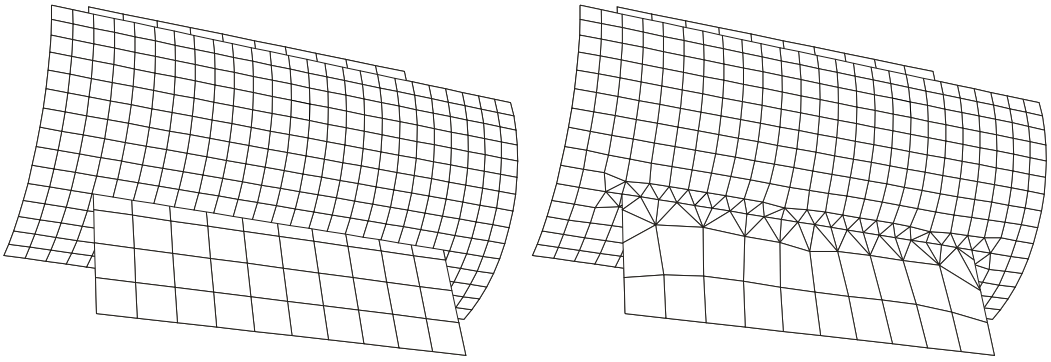


Figura 2.37 – Exemplo de interseção de malhas dos casos especiais aresta / aresta e aresta / vértice [8] e [6].

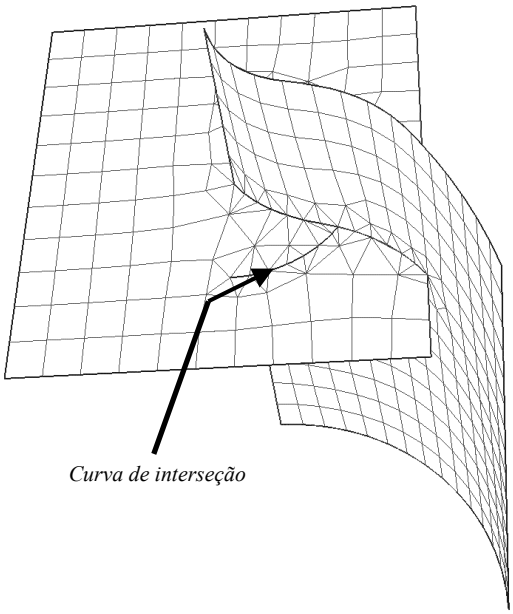


Figura 2.38 – Exemplo de interseção onde uma curva de *trimming* intercepta outra [6].

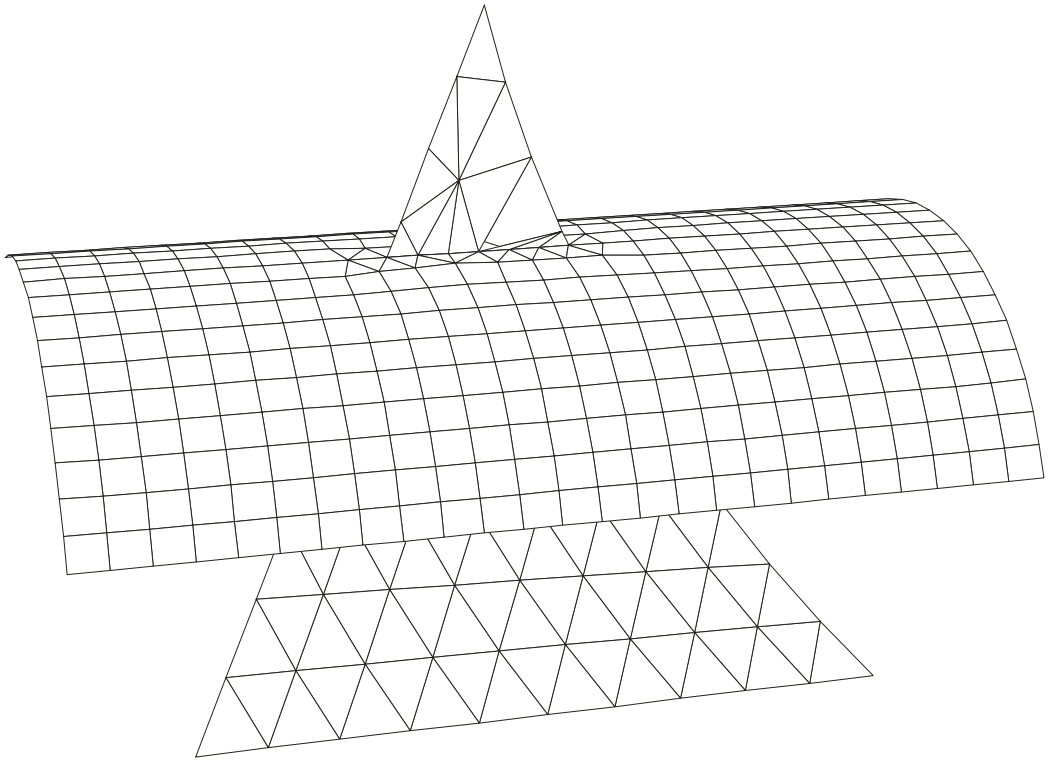


Figura 2.39 – Interseção de superfícies não-retangulares [6].

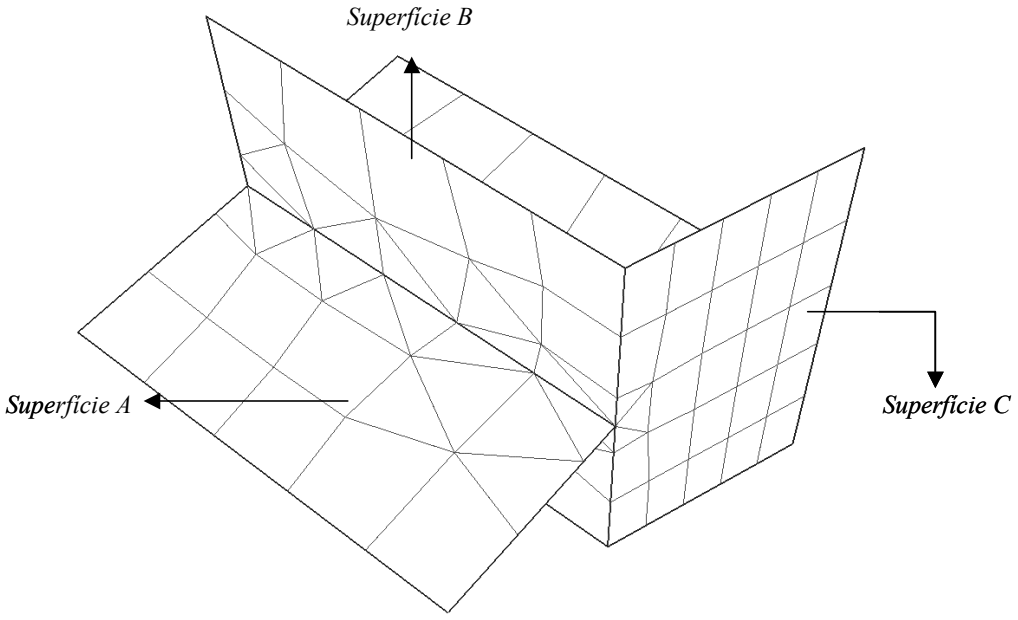


Figura 2.40 – Reconstrução da malha em superfícies incidentes às curvas de interseção [6].

Recentemente, o algoritmo de interseção do MG foi modificado por Lira de forma que todas as interseções entre retalhos de superfícies sejam calculadas através da biblioteca CGC. Os parâmetros de entrada do algoritmo são dois retalhos de superfícies (*patches2d*) e suas respectivas malhas superficiais. Os passos do novo algoritmo são os seguintes:

1. Refinamento das malhas, forçando que todos os elementos sejam triangulares. Isto garante que todos os elementos sejam faces planares. É importante observar que cada elemento possui uma referência para o retalho de superfície (*patch2d*) ao qual ele pertence. Isto permite que se recupere posteriormente a superfície associada a um determinado elemento.

2. Insere-se cada um dos elementos na RED (usando-se a biblioteca CGC). Cada um desses elementos é considerado como uma face planar. À medida em que os elementos são inseridos, a CGC vai computando automaticamente as interseções entre essas faces [19]. Ao final desta etapa, tem-se um modelo planar totalmente costurado topologicamente e consistente. Vale notar que novos vértices, arestas e faces podem ser criados nas regiões próximas às interseções. Dois importantes detalhes devem ser observados:
   - Novas arestas só serão criadas sobre a poligonal que representa a linha de interseção. Com isso, é possível detectar quais são os pontos de interseção obtidos;
   - Devido à representação topológica do modelo planar, é possível identificar os componentes conexos da interseção. Ou seja, pode-se verificar a existência de uma ou mais poligonais contendo os pontos de interseção e se essas poligonais são abertas ou fechadas.

3. Com as respostas obtidas no item 2, é possível computar todas as poligonais que representam as curvas de interseção entre as malhas planares utilizadas. Esses pontos de interseção obtidos servem como uma excelente aproximação para o cálculo de pontos de interseção entre as superfícies.

4. Usando um algoritmo de minimização pelo método de *Newton-Raphson*, calculam-se os pontos de interseção das superfícies a partir dos pontos obtidos no item 3. Isto é feito para cada ponto de interseção. Como os pontos obtidos no passo 3 estão próximos das interseções das superfícies (inclusive, no caso de superfícies planares eles são exatamente os pontos de interseção), o algoritmo converge rapidamente e encontra a resposta desejada. Alguns casos especiais

são tratados, principalmente em pontos próximos ao contorno das superfícies.

5. Os pontos de interseção obtidos no item 4 são interpolados, usando-se curvas NURBS, gerando as curvas de interseção.

6. Com essas curvas (segmentos), faz-se a compatibilidade topológica no MG, gerando novos *patches2d* (se necessário) a partir dessas curvas de interseção obtidas.

7. As malhas de elementos finitos associadas aos *patches2d* (antigos ou novos) são re-geradas usando-se o algoritmo para geração de malhas de superfícies desenvolvido por Miranda [26].

Um caso especial que ainda não havia sido tratado até a conclusão deste trabalho é aquele em que duas superfícies se sobrepõem (*overlapping*). Por se tratar de uma situação bastante peculiar no caso de superfícies curvas, não havia sido dada ênfase para o tratamento de patologias deste tipo. Contudo, quando se trabalha com superfícies planares, é mais comum surgirem situações onde isto ocorre. Apesar do algoritmo para operações booleanas proposto neste trabalho considerar casos especiais como este, a geração de modelos no MG para validar o algoritmo proposto nestes casos torna-se difícil, já que o algoritmo de interseção de superfícies, cuja chamada é requisitada antes da aplicação das operações booleanas, na maioria das vezes falha quando as superfícies se sobrepõem.

A interseção entre superfícies e curvas foi recentemente implementada no MG. Na verdade, este procedimento já era realizado como uma etapa da interseção de superfícies, mas não estava disponível para a sua utilização direta. Isto quer dizer que não havia suporte para se calcular explicitamente a interseção entre uma curva qualquer e uma superfície. A partir de sua última versão, o MG passou a contar com mais esta ferramenta, também crucial para a implementação correta do algoritmo de operações booleanas.

Um último tipo de interseção também está implementado no MG: a interseção entre duas curvas quaisquer. Obviamente, um algoritmo para resolver este problema é mais simples do que aquele apresentado para a interseção de superfícies. Na verdade, como a interseção de superfícies no MG é feita utilizando-se as malhas dos retalhos de superfície como suporte para a

determinação das curvas de interseção, este procedimento também já era realizado como uma etapa da interseção de superfícies.

O procedimento para a detecção de múltiplas regiões fechadas já foi descrito quando se apresentou a estrutura de dados híbrida do modelador MG. A representação CGC, que adota a estrutura de dados *Radial Edge* proposta por Weiler [17], é utilizada como uma forma de representação temporária da topologia do modelo quando o reconhecimento de regiões é requisitado pelo usuário. Vale ressaltar que as interseções de superfícies são realizadas antes do reconhecimento de regiões, para que na representação CGC os retalhos de superfície se interceptem apenas nas suas fronteiras. No final do processo, cada região na CGC gera um sólido na representação MG. A Figura 1.3 mostra o modelo explodido de parte de uma plataforma usada na exploração de petróleo em águas profundas, onde as regiões foram detectadas automaticamente pela técnica descrita.

A ênfase dada neste trabalho a estas duas ferramentas de modelagem é devida à importância que elas exercem na implementação do algoritmo de operações booleanas no MG. Como será visto nos próximos capítulos, para que o algoritmo possa ser aplicado corretamente sobre um conjunto de entidades topológicas pertencentes a um modelo, é necessário que todas as interseções entre superfícies, entre superfícies e curvas e entre curvas já tenham sido computadas. Esta condição forma a base para um critério de classificação das entidades topológicas de um *grupo* perante as entidades topológicas de outro *grupo*. Além disso, o reconhecimento automático de multi-regiões se faz necessário para que um conjunto conexo e fechado de retalhos de superfície possa ser tratado como um sólido, o que também é crucial para a classificação das entidades topológicas. Além disso, após a aplicação das operações booleanas sobre as entidades selecionadas, o modelador deve ser capaz de reconhecer eventuais regiões formadas com as entidades remanescentes.