ROBERTO YUJI YKKO UEDA

DESENVOLVIMENTO DE UM SISTEMA CAD COM SUPORTE A CURVAS E SUPERFÍCIES

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

São Paulo
1998

ROBERTO YUJI YKKO UEDA

DESENVOLVIMENTO DE UM SISTEMA CAD COM SUPORTE A CURVAS E SUPERFÍCIES

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

Área de Concentração:
Engenharia Mecânica

Orientador: Prof. Dr.
Marcos de Sales Guerra Tsuzuki

São Paulo
1998

Aos meus pais e irmãos

Agradecimentos

Agradeço ao Prof. Dr. Marcos de Sales Guerra Tsuzuki pela orientação prestada desde os anos em que eu cursava a graduação. Devido ao seu empenho à tarefa de ensino e constantes incentivo e compreensão foi possível realizar este trabalho.

Agradeço aos meus amigos Fábio K. Takase e Tomaz Sasaki pela amizade concretizada no bom humor e em tantos detalhes de serviços.

Agradeço ao Prof. Dr. Paulo Eigi Miyagi os conselhos e a orientação acadêmica.

Agradeço aos meus pais e irmãos a minha própria vida e a educação dentro de uma família sadia, ingredientes essenciais para a realização de qualquer dos meus atos.

Agradeço a tantas pessoas que me proporcionaram as condições para estudar e realizar este trabalho.

Agradeço a Deus por me ter criado e capacitado para trabalhar.

SUMÁRIO

LISTA DE FIGURAS

LISTAGENS DE ALGORITMOS

RESUMO

Modelagem de Sólidos e Modelagem Geométrica são duas técnicas utilizadas para representar formas físicas. Atualmente estão em pleno desenvolvimento, possuindo várias aplicações em Engenharia: Modelagem do Produto, Projeto Auxiliado por Computador (CAD), Planejamento do Processo Auxiliado por Computador (CAPP), Manufatura Auxiliada por Computador (CAM), entre outras. Como área de pesquisa, a Modelagem de Sólidos e a Modelagem Geométrica englobam várias disciplinas incluindo Matemática, Ciência da Computação e Engenharia. Apesar de terem crescido separadamente, as necessidades tecnológicas exercem tal pressão que estas técnicas tendem a fundir-se em uma só.

Proporcionando informações básicas sobre os principais conceitos da representação B-Rep ("*Boundary Representation*") de Modelagem de Sólidos e sobre curvas e superfícies de Bézier da Modelagem Geométrica, este trabalho propõe um caminho para a fusão das duas técnicas. O objetivo é introduzir, em um modelador de sólidos poliedral, uma estrutura de dados unificadora, na qual a informação geométrica (curvas e superfícies) é entendida como um atributo dos elementos topológicos (face, aresta e vértice).

Como conseqüência do aumento da precisão geométrica dos sólidos modelados, a gama de aplicações dos modeladores poliedrais pode ser ampliada. Neste trabalho apresentamos uma proposta para adaptar algoritmos poliedrais para geração de sólidos primitivos, translação linear e rotacional, secção por plano e operações booleanas entre sólidos, de modo que possam ser executadas em harmonia com curvas e superfícies, agora representadas de forma exata. São expostas também soluções para melhorar o desempenho em termos de processamento computacional em operações booleanas e para diminuir a ocupação de memória dos modelos B-Rep.

ABSTRACT

Solid modeling and Geometric Modeling are two techniques used to represent physical forms. Today they are in full development, having many applications in Engineering: Product modeling, Computer Aided Design (CAD), Computer Aided Process Planning (CAPP), Computer Aided Manufacturing (CAM), among others. In the research area, Solid Modeling and Geometric Modeling comprise many disciplines including Mathematics, Computer Science and Engineering. Although they developed separately, technological necessities are pressuring in a direction such that both technologies are going to fuse in to one.

By providing information about basic concepts of B-Rep (Boundary Representantion) Solid Modeling and about Bézier curves and surfaces of Geometric Modeling, this work proposes a way of merging both techniques. The goal is to introduce, in a polyedral modeler, an unified data structure, in which the geometrical information (curves and surfaces) will be an attribute of topological primitives (face, edge and vertex).

Because of increasing geometric precision of modeled objects, the applications can be extended. In this work we exposed a proposal to adapt polyhedral algorithms to create basic primitives and to execute translational and rotational sweeping, spliting and booelan operations, in a way that such operations may be executed together with curves and surfaces, now precisely represented. It is also explained how to improve the performance of boolean operations as well as to reduce memory occupation of B-Rep models.

1. Introdução

1.1. O objetivo deste trabalho

Os sistemas CAD / CAM desenvolveram-se muito nos últimos 20 anos, incorporando um número crescente de informações com relação à forma física que representam. Historicamente, as técnicas de representação da forma física são divididas segundo duas grandes classes: Modelagem de Sólidos e Modelagem Geométrica.

As técnicas de Modelagem Geométrica avançaram muito com a indústria automobilística desde a década de 50 (BÉZIER, 1986), onde são utilizadas, principalmente, para definir a carroceria dos automóveis. Atualmente, possuem aplicações na indústria naval para definir o perfil de embarcações, e na indústria aeronáutica para definir as asas e a fuselagem do avião. Elas possuem como característica principal o suporte ao controle da forma da superfície.

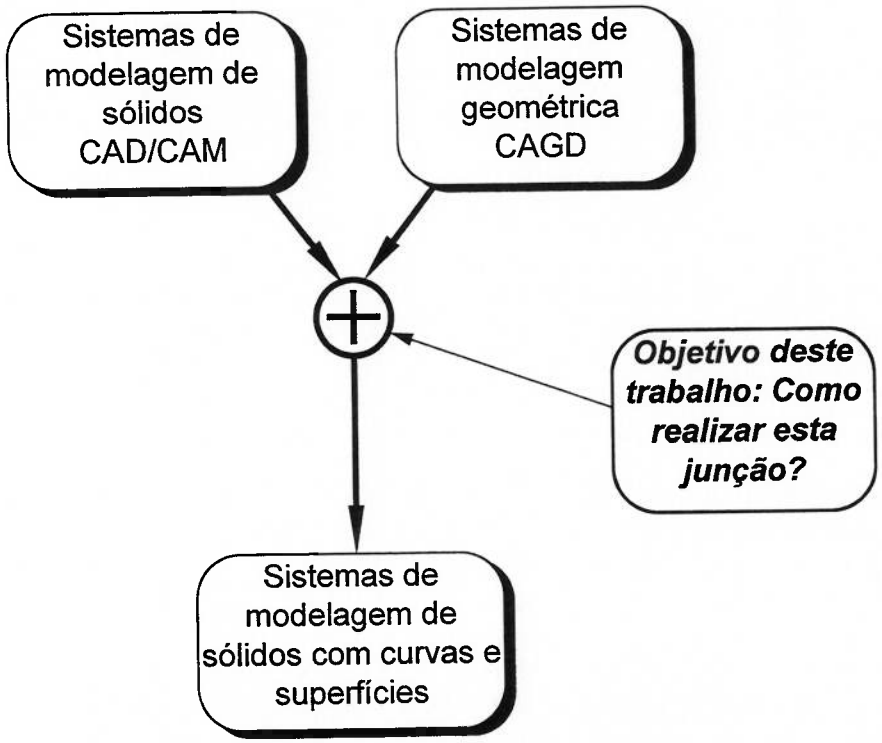


Figura 1-1 Objetivo do trabalho.

As técnicas da Modelagem de Sólidos surgiram bem depois, durante a década de 70 (ZEID 1991). Elas têm como característica principal a criação de Modelos Computacionais capazes de classificar qualquer ponto do espaço tridimensional como sendo: interno, externo ou sobre o

contorno do sólido. Podemos citar como aplicações mais importantes o suporte à determinação de propriedades de massa (como momento de inércia e volume, dentre outras propriedades) e a geração automática de malhas para sistemas de análise por elementos finitos. Entretanto, o domínio de representação era restrito a objetos poliedrais, ou, aproximações poliedrais de objetos curvos (como esferas, cilindros, cones dentre outros).

Neste trabalho estamos propondo uma representação unificadora que suporta características das técnicas de Modelagem Geométrica em objetos representados pela técnica de Modelagem de Sólidos. Esta proposta está ilustrada na Figura 1-1.

1.2. O balanço entre Modelagem de Sólidos e Modelagem Geométrica

Estas duas tecnologias foram desenvolvidas separadamente conforme salienta FARIN (1990). Costuma-se associar sólidos poliedrais à técnica de Modelagem de Sólidos e a representação de curvas e superfícies à técnica de Modelagem Geométrica. Na literatura possuímos poucas propostas para o assunto. TORIYA e CHIYOKURA (1991) descrevem duas propostas utilizadas no Modelador Designbase, desenvolvido pela Ricoh. MÄNTYLÄ (1988) comenta uma proposta semelhante adotada no modelador de sólidos GWB desenvolvido pela HUT (Helsinki University of Technology).

As necessidades tecnológicas atuais exigem o suporte a ambas técnicas simultaneamente. A representação poliedral é necessária pois vários algoritmos presentes em Modelagem de Sólidos fazem uso desta representação. Por exemplo, o cálculo de propriedades de massa de sólidos é realizado utilizando-se aproximações poliedrais, e a visualização de sólidos quando realiza o processamento de linhas e superfícies escondidas, utiliza também aproximações poliedrais.

A representação de curvas e superfícies é necessária para possuirmos acesso à geometria exata da forma. Esta informação é útil para determinar o caminho de usinagem com maior precisão, principalmente na determinação do vetor tangente em uma posição específica. Neste trabalho, propusemos um mecanismo de sincronismo entre as duas técnicas de forma a poderem coexistir sem detrimento de uma em relação à outra. Esta proposta foi implementada no USPDesigner 1.0[1]. O presente trabalho possui a seguinte estrutura:

No capítulo 2 são apresentados os conceitos básicos da técnica de Modelagem de

[1] Modelador de Sólidos desenvolvido pelo Departamento de Engenharia Mecânica da Universidade de São Paulo.

Sólidos.

No capítulo 3 é exposta a teoria básica sobre curvas e superfícies. As formas matemáticas destes elementos e sua manipulação: como definir e tratar uma curva ou superfície de Bézier; como dividir curvas de Bézier; como calcular a intersecção entre estas curvas e superfícies.

O capítulo 4 explica uma proposta para realizar a união das técnicas de Modelagem de Sólidos e Modelagem Geométrica. Indicam-se quais são as modificações necessárias para transformar o modelador de sólidos USPDesigner 1.0 em um modelador que suporte curvas e superfícies.

O capítulo 5 apresenta os resultados obtidos. Finalmente o capítulo 6 analisa os resultados, tece algumas conclusões e discute aplicações futuras.

2. Modelagem de Sólidos

Segundo MÄNTYLÄ (1988), um modelo é um objeto construído de forma artificial que facilita a observação de um objeto de interesse. Os desenhos técnicos, vastamente utilizados em engenharia, são modelos, uma vez que as pessoas capacitadas podem interpretar as informações contidas nas coleções de imagens bidimensionais e nos simbolismos para representar dimensões, tolerâncias e características de superfície, dentre outras informações contidas nos desenhos técnicos.

Muitos métodos de modelagem geométrica têm sido utilizados para representar formas tridimensionais. As técnicas mais conhecidas e utilizadas nos sistemas CAD para engenharia mecânica são a modelagem por fio de arame (*wireframe modeling*), os modelos por superfícies (*surface modeling*) e a modelagem de sólidos (*solid modeling*). Várias técnicas de modelagem de sólidos foram propostas na literatura (cfr. ZEID 1991, pág 350). As mais utilizadas são a representação B-Rep (*Boundary Representation*) e a representação CSG (*Construtive Solid Geometry*).

2.1. Modelagem por fio de arame

Um modelo de fio de arame é representado por listas de arestas que por sua vez contêm pontos. Cada ponto também é uma lista que contém suas respectivas coordenadas (valores `x`, `y`, `z`). Para um projetista acostumado ao desenho técnico, esta é a representação mais natural de um objeto, uma vez que são as retas e curvas em um desenho que definem a forma tridimensional. A grande vantagem desta representação é a sua simplicidade. Na prática esta característica reflete-se na reduzida quantidade de memória utilizada e no curto tempo de acesso e processamento computacional. A principal desvantagem da modelagem por fio de arame é o fato dela ser ambígua. A Figura 2-1 ilustra este tipo de ambigüidade, onde um modelo pode ser associado a mais que um objeto real. Outra desvantagem ocorre quando é necessário representar objetos que possuem muitas arestas, uma vez que sua visualização torna-se confusa e em alguns casos impossível de ser interpretada.

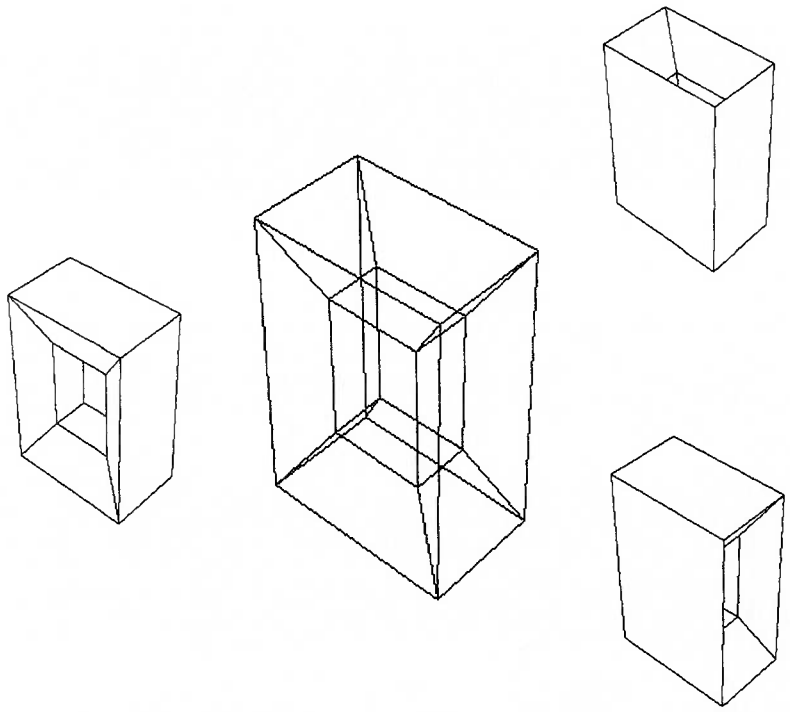


Figura 2-1 Um modelo ambíguo.

A representação por fio de arame já classifica as informações do modelo em informações geométricas e topológicas. Para compreender esta classificação, apresentam-se duas possíveis implementações em linguagem C para a representação por fio de arame. Na primeira implementação definem-se dois elementos primitivos: vértice e linha (vide Listagem 2-1). Um vértice é definido por suas três coordenadas: **x**, **y** e **z**. Uma linha é representada pelos dois vértices que a delimitam.

```
typedef struct {
      float vx, vy, vz;
} ponto ;

typedef struct {
      ponto p1;
      ponto p2;
} linha ;
```

Listagem 2-1 Primeira implementação para a Representação por Fio de Arame.

Quando um modelo possuir múltiplas linhas, esta proposta apresentará um armazenamento ineficiente devido à possibilidade de existirem várias linhas com compartilhamento de vértices. Além do mais, o compartilhamento de vértices só poderia ser detectado pela comparação numérica das coordenadas, que é uma operação computacionalmente arriscada. Uma forma de estruturar a informação, evitando a necessidade de cópias múltiplas, é ilustrada no trecho de código da Listagem 2-2, onde os vértices de uma linha são referenciados por meio de ponteiros.

```
typedef struct {
        ponto *p1;
        ponto *p2;
} linha ;
```

Listagem 2-2 Segunda implementação para a Representação por Fio de Arame.

A diferença entre estas duas estruturas, sob o ponto de vista de implementação, é mínima. Entretanto, a representação do compartilhamento dos vértices entre as arestas permite definir algoritmos de edição de modelos mais eficazes. Assim, ao selecionar-se um vértice e alterar suas coordenadas, todas as linhas que compartilham este vértice serão alteradas (vide Figura 2-2). Informalmente, a informação sobre o compartilhamento de elementos é denominada por informação topológica.

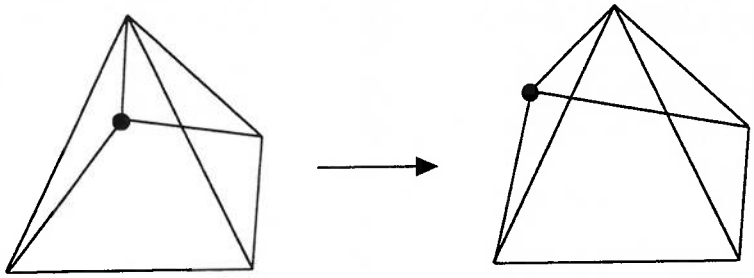


Figura 2-2 Informação topológica: linhas compartilhando um vértice.

2.2. *Modelagem por superfície*

A modelagem por superfícies, além das listas de arestas e pontos, presentes na modelagem por fio de arame, possui também uma lista de faces. A Figura 2-3 apresenta alguns exemplos desta técnica. A modelagem por superfícies é considerada uma extensão da modelagem por fio de arame. Esta técnica possibilita a definição de algoritmos para remover linhas escondidas, bem como, de algoritmos de visualização como a eliminação de superfícies escondidas.

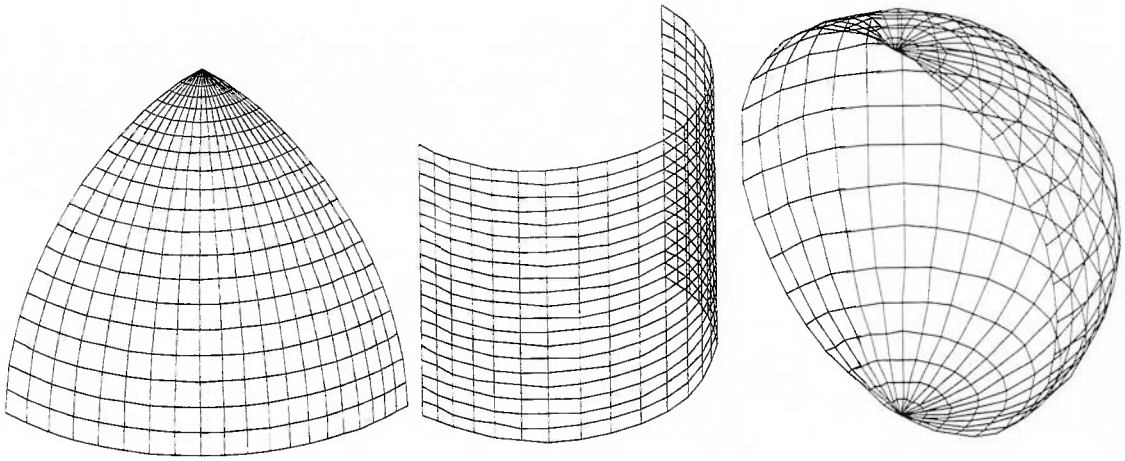


Figura 2-3 Exemplos de modelos de superfície.

2.3. Modelos sólidos

Embora alguns autores afirmem que um modelo sólido não pode ser definido segundo uma representação por superfície, é importante notar que isto não é totalmente verdadeiro. Um modelo sólido é representado por um conjunto de faces coerentemente orientadas, com vetores normais das faces apontando para o exterior do sólido, por exemplo. As informações topológicas, no caso de representação por superfície, podem ser obtidas por métodos algorítmicos. Entretanto, situações de ambigüidade podem ocorrer (vide Figura 2-4). O problema consiste na verificação da existência ou não de compartilhamento de elemento entre as faces f_1 e f_2.

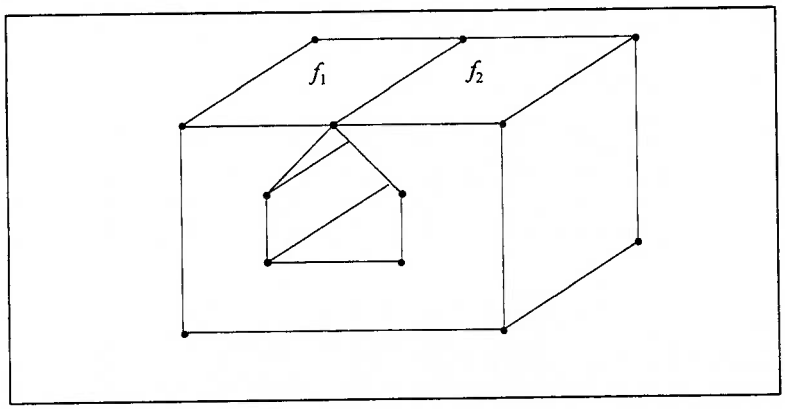


Figura 2-4 Ambigüidade em modelos por superfície.

Um modelo sólido, por conter informações sobre o compartilhamento de elemento entre as faces, é capaz de informar sem ambigüidade qual a relação existente entre estas duas faces. Já um modelo por superfície, por interpretar a topologia do objeto de forma algorítmica, não será capaz de

gerar a mesma informação sem incorrer em ambigüidade (mais de uma interpretação). A seguir discutiremos as duas formas de representação para modelagem de sólidos mais significativas: CSG e B-Rep.

2.3.1. Representação CSG

Na representação CSG, um sólido é definido pela combinação de sólidos simples. Estes sólidos simples são chamados primitivos. Um sólido é representado segundo uma árvore binária onde as suas folhas são os primitivos e os nós internos são operadores booleanos que relacionam os dois nós inferiores. A construção de modelos CSG combinando primitivos por operadores booleanos é relativamente fácil, pois é uma tarefa intuitiva.

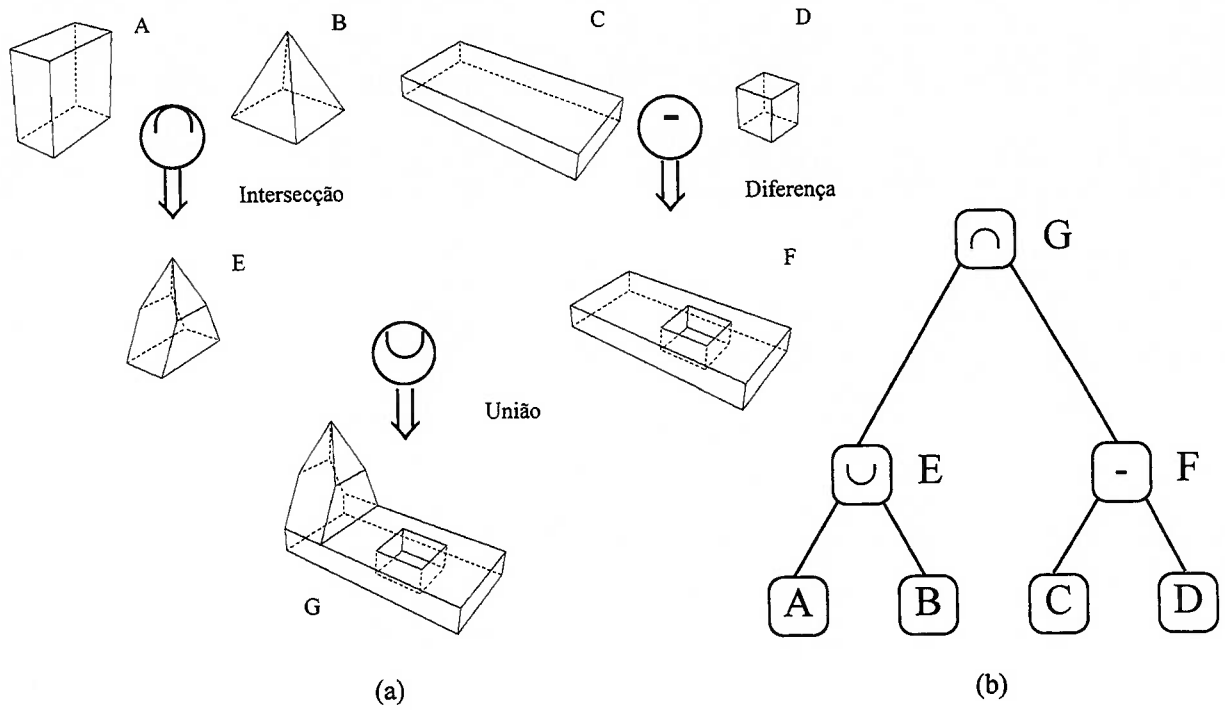


Figura 2-5 Representação CSG de um sólido.

A Figura 2-5 (a) mostra um exemplo de sólido com representação CSG. O sólido E é definido pela combinação dos sólidos A e B, por meio de uma operação de intersecção. O sólido F é definido pela diferença entre os sólidos C e D. Finalmente, o sólido G é definido pela união dos sólidos E e F. Este processo seqüencial de modelagem é representado por uma árvore binária denominada árvore CSG, ilustrada na Figura 2-5 (b).

2.3.2. Representação B-Rep

A representação B-Rep armazena detalhes de como as faces, arestas e vértices de um sólido se unem para representar um sólido. Um sólido representado em modelo B-Rep deve possuir, por exemplo, a capacidade de descrever como cada face está conectada às suas faces adjacentes, de modo a formar um volume totalmente fechado no espaço. Nos modelos CSG, a informação de adjacência entre faces pode ser obtida por técnicas numéricas que analisam as proximidades geométricas de seus componentes; contudo, estas técnicas geralmente envolvem um custo computacional considerável além de apresentarem problemas de precisão numérica. Nos modelos B-Rep estas informações estão presentes de forma explícita na estrutura de dados.

Os três tipos de elementos básicos de um sólido (face, aresta e vértice) e a informação geométrica relacionada aos mesmos formam os constituintes básicos dos modelos B-Rep. Junto com as informações geométricas, como equações do plano das faces e coordenadas de vértice, um modelo B-Rep deve também representar a relação entre as faces, arestas e vértices. Normalmente as informações geométricas dos elementos de um sólido são denominadas por geometria do modelo B-Rep, enquanto as informações sobre o compartilhamento dos elementos básicos são denominadas informalmente por topologia do modelo. Pode-se dizer que a topologia funciona como uma goma onde as informações geométricas são aglutinadas (MÄNTYLÄ, 1988); ou então que "as informações topológicas criam um vigamento no qual as informações geométricas são posicionadas" (TSUZUKI, 1991).

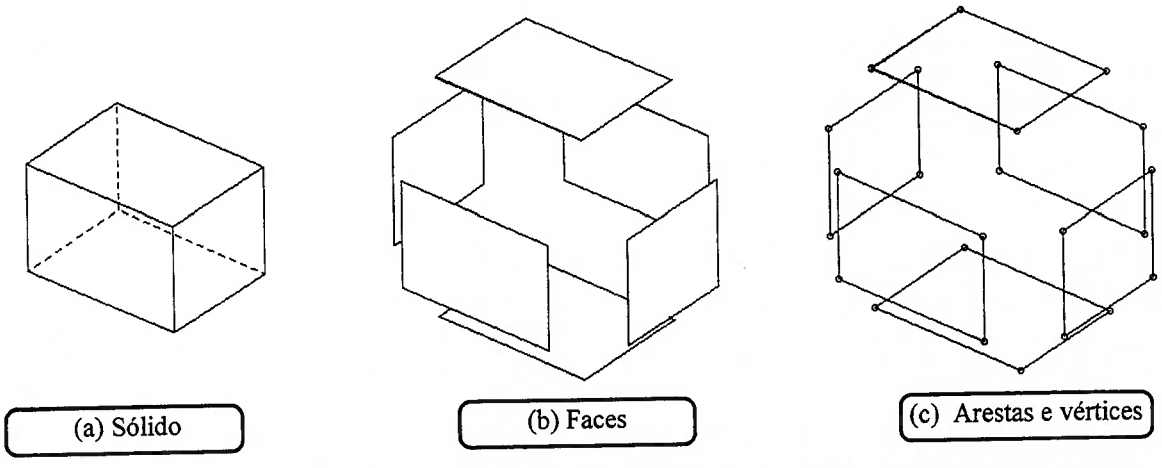


Figura 2-6 Componentes básicos de um modelo por fronteira (MÄNTYLÄ, 1988).

A Figura 2-6 ilustra os componentes básicos de um modelo B-Rep. Na figura, a superfície é dividida em conjuntos fechados de faces (a); cada face é representada através de um polígono de contorno (b), que pode ser representado em termos de arestas e vértices (c).

2.4. Um sistema de Modelagem de Sólidos B-Rep

Num sistema de modelagem de sólidos podemos definir três níveis de representação de um sólido (TSUZUKI et el. 1991). O primeiro nível, o nível mais alto de abstração, interage com o usuário através de operações para construir, modificar e armazenar os sólidos. Neste nível, o sólido é descrito por um conjunto de operações de alto nível, que permite o interfaceamento com o usuário. Entre o primeiro nível e o segundo nível de implementação, localiza-se toda a infra-estrutura matemática e algorítmica do sistema de modelagem que suportam e tornam executáveis as operações disponibilizadas ao usuário no primeiro nível. No segundo nível, o sólido é descrito por um conjunto de operadores específicos, denominados por Operadores de Euler, que serão tratados com detalhes à frente. No terceiro nível, o nível inferior de abstração, o sólido é representado por uma estrutura de dados computacional.

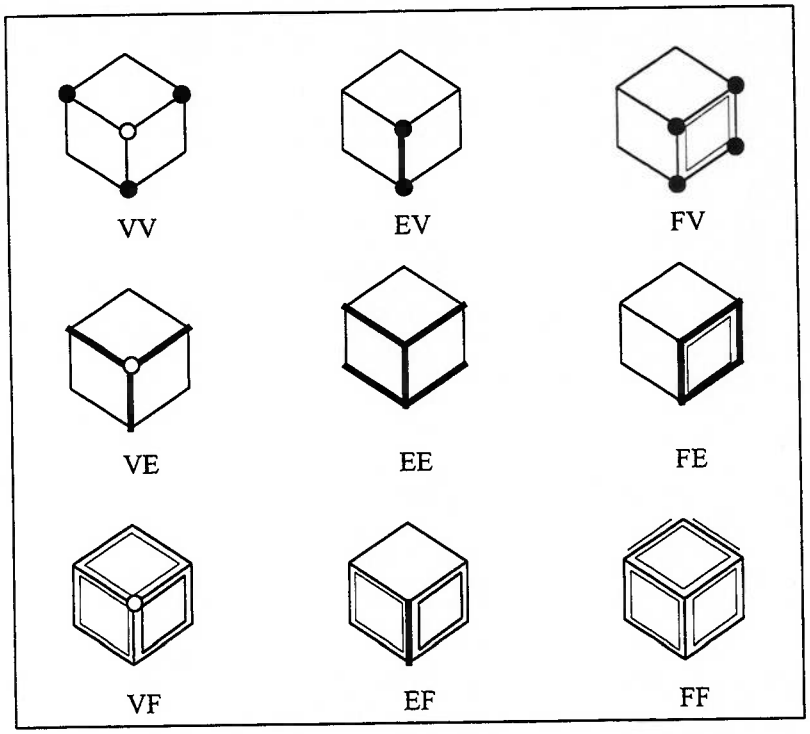


Figura 2-7 Relações de adjacência (WEILER, 1985).

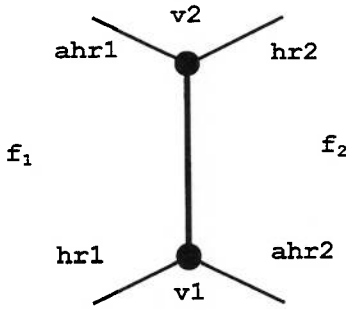
Por ser uma representação de modelos sólidos, a representação B-Rep precisa armazenar informações de como as faces, as arestas e os vértices de um sólido estão compartilhados (relações de adjacência). As nove relações de adjacência, definidas pela combinação dois a dois entre os três elementos primitivos da representação B-Rep, podem ser visualizadas na Figura 2-7. Nesta figura, a relação VV é entre um vértice e outros vértices que com este definem uma aresta, a relação EV é entre uma aresta e os dois vértices que a definem, a relação FV é entre uma face e os vértices que a definem, a relação VE é entre um vértice e as arestas limitadas por ele, a relação EE é entre uma

aresta e as outras quatro arestas que possuem a mesma face como contorno, a relação FE é entre uma face e as arestas que a limitam, a relação VF é entre um vértice e as faces que o possuem, a relação EF é entre uma aresta e as duas faces que ela separa e finalmente, a relação FF é entre uma face e as outras que a circundam.

2.4.1. Estrutura de Dados

Nesta sessão apresentam-se três estruturas de dados que foram muito utilizadas na literatura para implementar a representação B-Rep. As estruturas estão baseadas na aresta como elemento de referência: estrutura *winged-edge*, estrutura *half-edge* e estrutura unificada.

A estrutura *winged-edge* mantém as informações de adjacência por meio de ponteiros a vários elementos adjacentes à aresta de referência: duas faces, dois vértices e quatro arestas. Cada uma das quatro arestas compartilha com a aresta de referência um vértice e uma face. (vide Figura 2-8). Durante um período relativamente longo, apesar da estrutura *winged-edge* ter sido largamente utilizada por pesquisadores de modeladores de sólidos, pouca coisa foi feita para racionalizar analiticamente a sua eficiência.



(a)

Tabela da aresta

v1	v2
hr1	hr2
ahr1	ahr2
f_1	f_2

hr = aresta no sentido horário
ahr = aresta no sentido anti-horário

(b)

Figura 2-8 Estrutura winged-edge.

Observe que algumas relações de adjacência envolvem um número variável de elementos. A relação de adjacência FE, por exemplo, requer um número variável de arestas. A Figura 2-9 ilustra uma pirâmide de quatro lados, e possuímos faces com três e quatro arestas adjacentes. Nesta figura é possível observar que as arestas definem um circuito direcional ao redor da face. Por exemplo, a seqüência de arestas e_1, e_2, e_3 e e_4 define o circuito direcional das arestas

da face f_1. Também é possível observar que cada aresta participa em dois circuitos direcionais e em sentidos opostos. Por exemplo, a aresta e_1 participa dos circuitos direcionais das faces f_1 e f_5 em sentidos complementares.

Algumas relações de adjacência envolvem um número constante de elementos adjacentes, não importando a situação. A relação de adjacência EV, por exemplo, envolve exatamente dois vértices para cada aresta. Conforme foi demonstrado por WOO (1985), é conveniente que as relações de adjacência armazenadas explicitamente na representação possuam um número constante de elementos adjacentes. Este critério aplica-se a três relações de adjacência: EV, EE e EF. Combinando estas três relações de adjacência descobrimos a estrutura *winged-edge*.

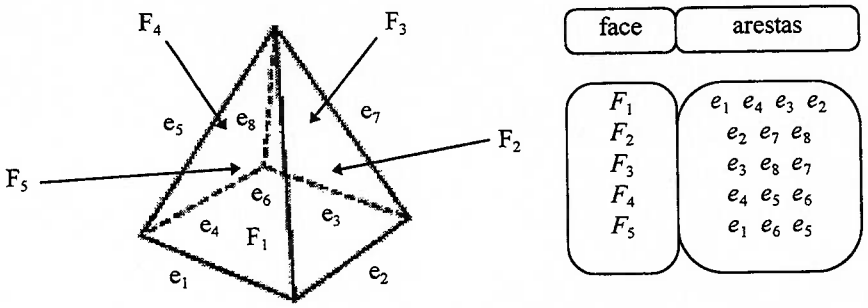


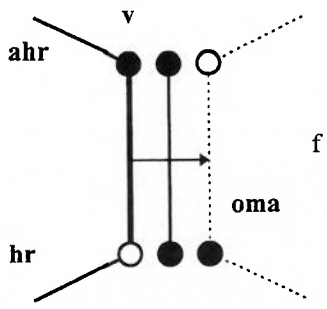
Figura 2-9 Estrutura de dados baseada na relação FE.

A estrutura *winged-edge* possui também a relação de adjacência fracional. Ao invés de armazenar todas as arestas pertencentes à relação de adjacência FE para cada face, ela armazena apenas uma aresta pertencente à relação para cada face. A relação de adjacência FE cria um conjunto de armazenamento proporcional ao número de faces e independente do número de arestas associadas às várias faces.

Conforme analisado por WEILER (1985), na estrutura *winged-edge*, as arestas assumem duas funções principais: representar o circuito direcional de faces e representar a aresta propriamente dita. A junção destas duas funcionalidades numa única estrutura causa uma deficiência computacional. Esta deficiência é clara, em particular, quando as arestas devem ser referenciadas pelo procedimento que percorre seqüencialmente o circuito direcional de arestas de cada face. A necessidade deste algoritmo surge com freqüência em operações gráficas ou geométricas aplicadas ao modelo representado.

Percorrer o circuito de arestas de uma face é uma operação unidirecional, devido à própria orientação do circuito de arestas. Na estrutura *winged-edge* cada aresta é utilizada para delimitar duas faces; portanto, cada aresta participa de dois circuitos orientados. Como os dois circuitos possuem orientações opostas entre si, cada aresta é percorrida sempre duas vezes e em direções opostas. Entretanto, é necessário verificar e determinar a direção em que cada aresta está

sendo percorrida a cada passo do percurso, um processo que aumenta consideravelmente o custo de tempo do procedimento e implicará no desenvolvimento de algoritmos complexos quando estiverem em ambiente curvo.



(a)

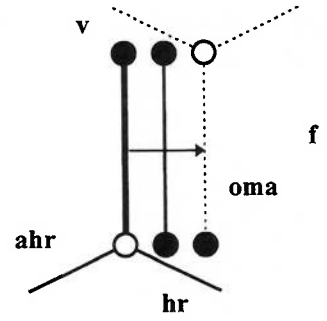
Tabela da *half-edge*

v
hr
ahr
f
oma

hr = aresta no sentido horário
ahr = aresta no sentido anti-horário
oma = outra meia-aresta

(b)

Figura 2-10 Estrutura face-aresta.



(a)

Tabela da meia-aresta

v
hr
ahr
f
oma

hr = aresta no sentido horário
ahr = aresta no sentido anti-horário
oma = outra meia-aresta

(b)

Figura 2-11 Estrutura vértice-aresta.

WEILER (1985) solucionou este problema separando as duas principais funções da aresta. Esta separação foi obtida pela divisão de cada *winged-edge* em duas metades. A conectividade entre ambas as metades é mantida por um ponteiro que referencia a metade oposta. Existem duas possibilidades para realizarmos esta separação: estrutura face-aresta (Figura 2-10) e estrutura vértice-aresta (Figura 2-11). A definição da estrutura face-aresta foi feita baseando-se na relação de adjacência FE. Em contrapartida, a estrutura vértice-aresta foi baseada na relação de adjacência VE.

Em cada estrutura *half-edge* está sendo representada a metade das informações de adjacência da estrutura *winged-edge*. Cada *half-edge* possui apenas uma orientação, e cada face possui um circuito direcional de *half-edges*.

Para evitar a duplicidade das informações associadas à aresta (por exemplo, atributo de cor) nas duas *half-edges*, KALAY (1989) propôs a criação de uma nova estrutura para a aresta que estaria associada a duas *half-edges* que definem a aresta de referência.

Até este momento, foi assumido que cada face possui apenas um contorno. Entretanto, casos práticos requerem que uma face possua mais de um contorno (como uma face com furos). Faces com mais de um contorno podem ser simuladas pela técnica de aresta-ponte (*bridge-edge*) no qual uma aresta une os contornos de uma face entre si. A aresta-ponte, portanto, possui a mesma face adjacente em ambas as laterais (YAMAGUCHI, 1985) (vide Figura 2-12 (a)).

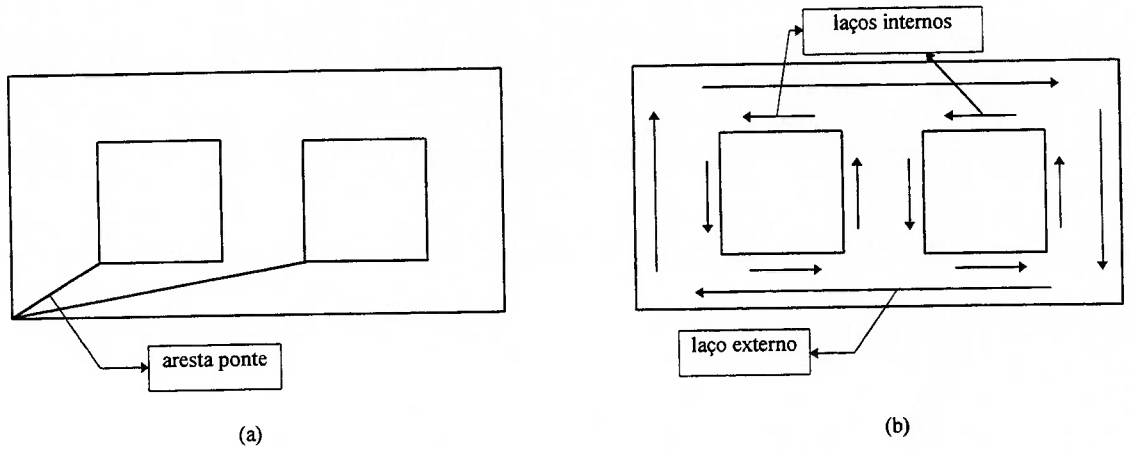


Figura 2-12 Extensão para faces com múltiplos contornos.

Entretanto, a técnica aresta-ponte não é muito eficiente porque será necessário determinar como os contornos devem ser conectados pelas arestas-ponte, o que criará a necessidade de complexos algoritmos para implementar operações de modelagem. Como exemplo, podemos citar as Operações Booleanas que provavelmente interseccionarão as arestas-ponte. Alterações na estrutura de dados *half-edge* de maneira a suportar faces com mais de um contorno não afetarão a estrutura B-Rep ao nível de aresta, mas sim, ao nível de face. Uma técnica muito comum é adicionar uma estrutura de tamanho fixo chamada laço (*loop*) que é associada a cada contorno da face. A estrutura laço simplesmente fornece à estrutura face um mecanismo para manter uma lista ligada de ponteiros para os seus múltiplos contornos. Cada face possui um laço externo e zero ou mais laços internos (vide a Figura 2-12 (b)).

Devido ao formalismo das Operações Booleanas, ao combinarmos dois sólidos por uma Operação Booleana, o resultado será sempre apenas um sólido. Entretanto, mesmo em situações especiais, como a situação exemplificada na Figura 2-13, onde o resultado da Operação Booleana aparenta apresentar dois sólidos, o resultado é considerado como sendo apenas um sólido. Entretanto, nesta situação em especial, considera-se que o sólido resultante possui dois *shells*. Em

outras palavras, para representar esta situação especial, foi criado o elemento *shell* que representa conjuntos disconexos de faces no espaço.

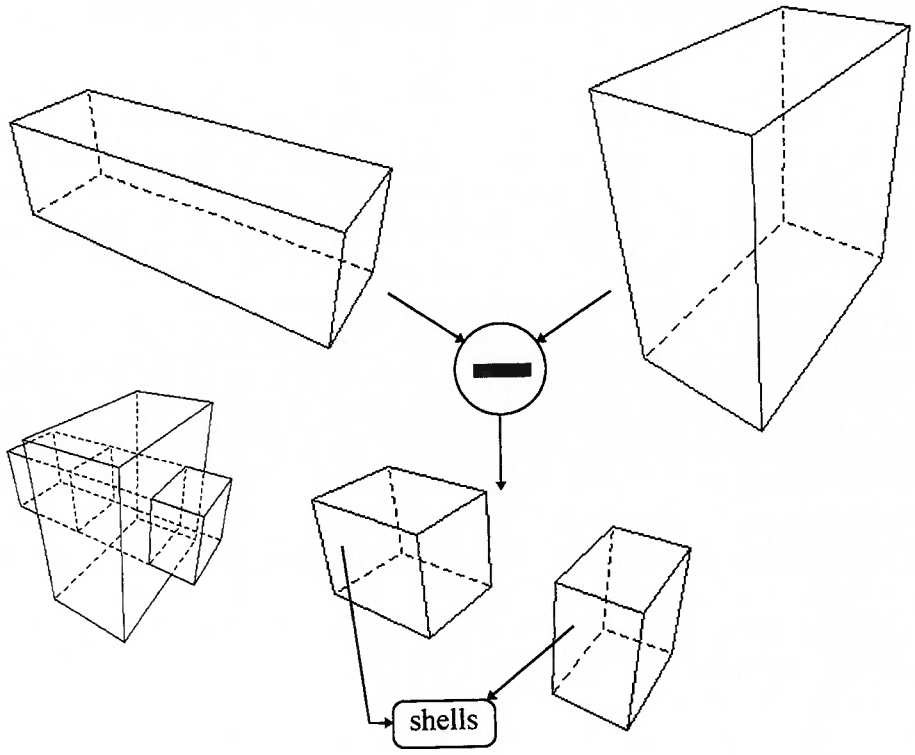


Figura 2-13 Sólido com dois shells.

A Figura 2-14 ilustra a hierarquia da estrutura *half-edge* resultante.

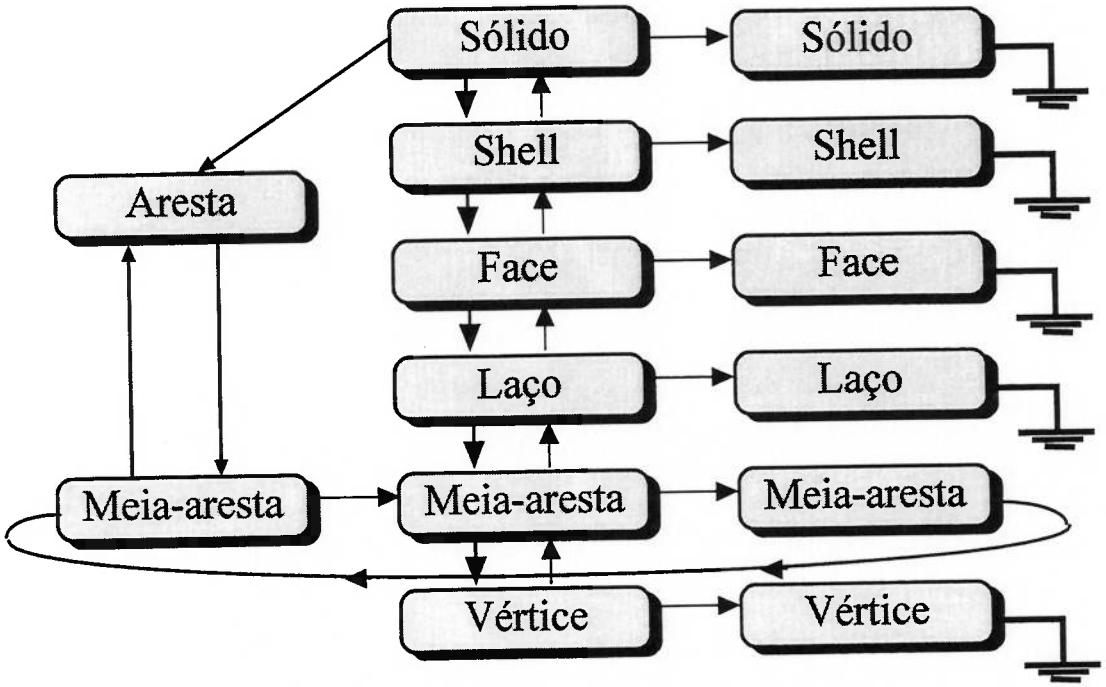


Figura 2-14 Vista hierárquica da estrutura *half-edge*.

A Figura 2-15 ilustra os elementos primitivos de um modelo sólido.

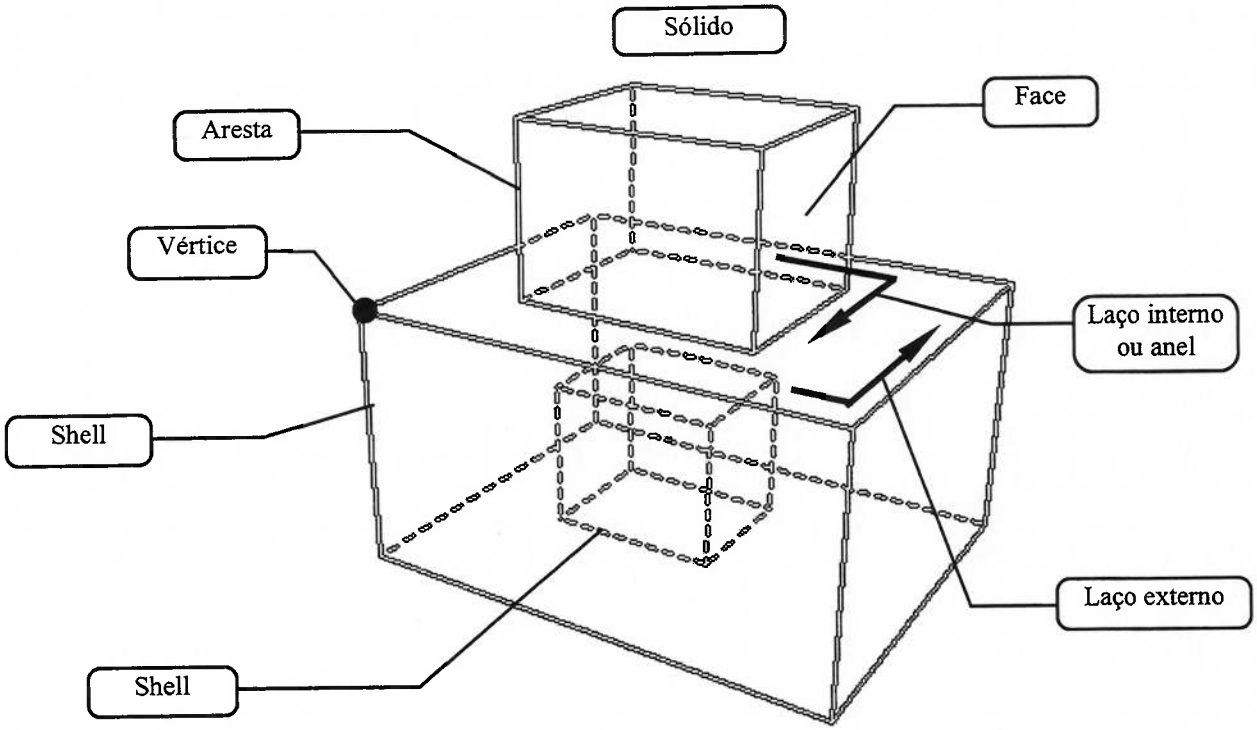


Figura 2-15 Elementos topológicos de um modelo sólido.

2.4.2. Operadores de Euler

Por conterem informações sobre as adjacências entre os elementos primitivos, as estruturas computacionais anteriormente expostas são bastante complexas e a sua manipulação exige muitos cuidados para que a consistência dos dados seja mantida. Para contornar este problema, um conjunto de operadores foi desenvolvido com o objetivo de tornar a manipulação das estruturas de dados da representação B-Rep mais intuitiva. Eles permitem que a construção do sólido possa ser executada passo a passo, escondendo todos os detalhes de implementação da estrutura de dados. Estes operadores formam o segundo nível de representação do modelador.

A equação de Euler-Poincaré diz que um sólido poliédrico é topologicamente válido se a seguinte relação entre as suas quantidades de elementos for verificada (ZEID, 1991):

$$v - e + 2f = 2(s - h) + l$$

onde v é o número de vértices do sólido, e o número de arestas, f o número de faces, s o número de shells, h o número de furos e l o número de laços. A forma mais conhecida na literatura da equação de Euler-Poincaré supõe ainda a existência de r anéis no sólido, onde $r=l-f$. Ficando a equação da seguinte forma (MÄNTYLÄ, 1988):

$$v - e + f = 2(s - h) + r$$

Vários autores demonstram que seis operadores são suficientes para construir todos os objetos. Enquanto estes seis operadores podem ser escolhidos de várias maneiras, considerações de modularidade e independência criaram apenas pequenas variações na coleção encontrada na literatura. Com a finalidade de facilitar a memorização, os Operadores de Euler estão definidos utilizando a nomenclatura da Tabela 1.

Tabela 1 Nomenclatura dos Operadores de Euler

Símbolo	Significado
M	*make*
K	*kill*
V	*vertex*
E	*edge*

Símbolo	Significado
F	*face*
S	*shell*
H	*hole*
R	*ring*

Por exemplo, o nome **MEV** deve ser traduzido por *Make Edge, Vertex* (Cria uma aresta e um vértice).

A seguir foram selecionados e estão descritos os seis operadores mais comumente utilizados na literatura (MÄNTYLÄ, 1988) e uma representação gráfica para cada operador pode ser observada na Figura 2-16.

- **MVSF** (*Make Vertex Solid Face*): este operador cria um sólido inicial com apenas uma face e um vértice;
- **MEV** (*Make Edge Vertex*): este operador adiciona a um sólido uma aresta e um vértice. A aresta é criada conectando-se um vértice já existente ao novo vértice criado;
- **MEF** (*Make Edge Face*): este operador adiciona ao sólido uma aresta e uma face. A face é criada pela divisão de uma face já existente acrescentando-se a nova aresta;
- **KEMR** (*Kill Edge Make Ring*): este operador divide o contorno de uma face em dois laços pela remoção de uma aresta-ponte;
- **KFMRH** (*Kill Face Make Ring Hole)*: nenhum dos operadores discutidos anteriormente é capaz de modificar as propriedades topológicas globais da estrutura de dados, como dividir um sólido em dois componentes ou criar um furo passante. O operador **KFMRH** possui este objetivo;
- **MSFKR** (*Make Shell Face Kill Ring*): este é outro operador que manipula informações globalmente. Ele transforma o anel de uma face em uma nova face, e todo o conjunto de faces associadas à nova face constituirá um novo *shell*;

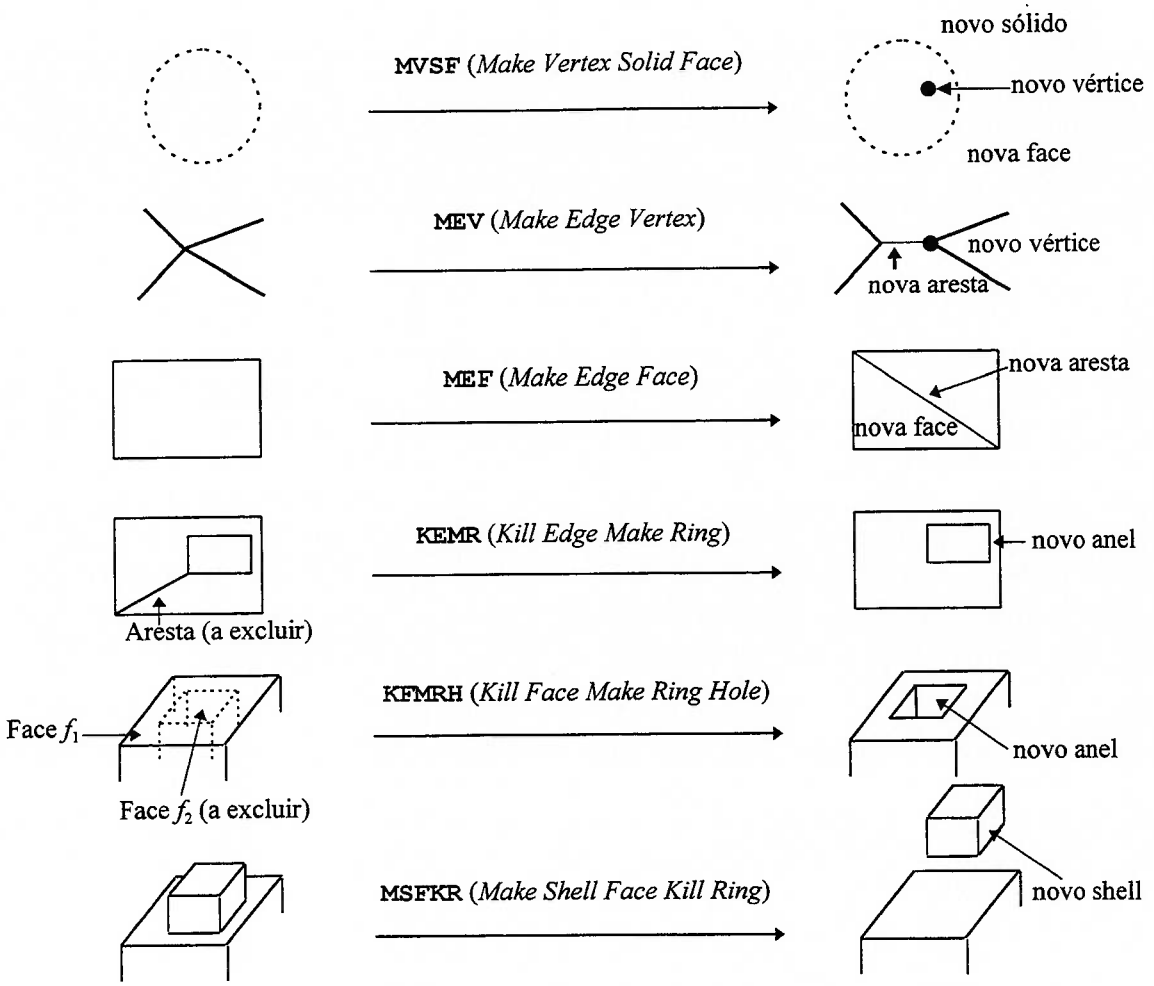


Figura 2-16 Representação gráfica dos Operadores de Euler.

Os seis operadores detalhados acima são todos construtivos; entretanto, para que o sólido possa ser alterado satisfatoriamente ele precisa também de algumas operações destrutivas. Por isto, cada operador construtivo possui um operador correspondente destrutivo. São eles:

KVSF (*Kill Vertex Solid Face*) ←→ **MVSF** (*Make Vertex Solid Face*)

KEV (*Kill Edge Vertex*) ←→ **MEV** (*Make Edge Vertex*)

KEF (*Kill Edge Face*) ←→ **MEF** (*Make Edge Face*)

MEKR (*Make Edge Kill Ring*) ←→ **KEMR** (*Kill Edge Make Ring*)

MFKRH (*Make Face Kill Ring Hole*) ←→ **KFMRH** (*Kill Face Make Ring Hole*)

KSFMR (*Kill Shell Face Make Ring*) ←→ **MSFKR** (*Make Shell Face Kill Ring*)

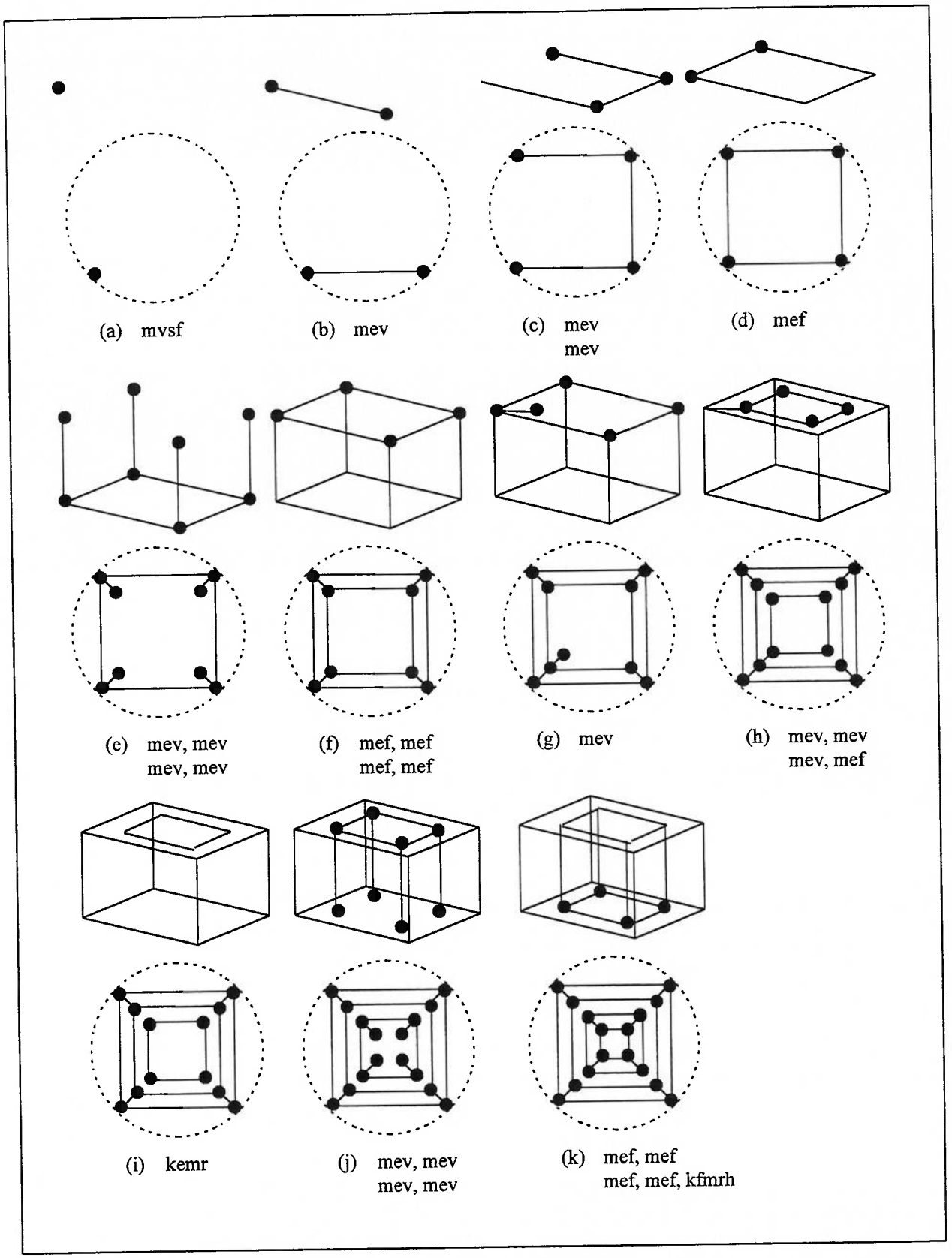


Figura 2-17 Seqüência de Operadores de Euler para criação de um cubo com furo passante (MÄNTYLÄ, 1988).

Como salientado por TSUZUKI (1991), durante o processo de construção de um sólido pela utilização de Operadores de Euler, a validade topológica do mesmo é mantida observando-se a equação de Euler-Poincaré. Entretanto, é comum o agrupamento de Operadores de Euler em uma certa seqüência para mantermos também a geometria válida. É importante observar que não há como manter a geometria válida em todos os estágios da construção. Por isto, os Operadores de Euler devem ser agrupados em seqüências que possuam algum significado.

Através de um exemplo simples, um bloco retangular com um furo passante retangular, é possível ilustrar a utilização dos Operadores de Euler (Figura 2-17).

A combinação de Operadores de Euler em seqüências com significado é realizada pela interface existente entre o nível mais alto de abstração e o segundo nível de abstração. No nível mais alto possuímos dois tipos de operadores: operadores locais e operadores globais.

2.4.3. Operadores Locais

Os operadores locais permitem que algumas características dos sólidos possam ser modificadas diretamente pelo usuário preservando as consistências topológica e geométrica do local modificado. Por não realizarem alterações em áreas externas à localidade das características na que eles se aplicam, não são realizadas verificações sobre o sólido após a realização de uma operação local. Entretanto, existe a possibilidade de se criar um sólido inválido utilizando-se operadores locais, como por exemplo, um sólido auto-interceptante (Figura 2-18). Não é uma tarefa fácil verificar se um sólido é auto-interceptante; geralmente, depende da correta utilização destas operações pelo usuário.

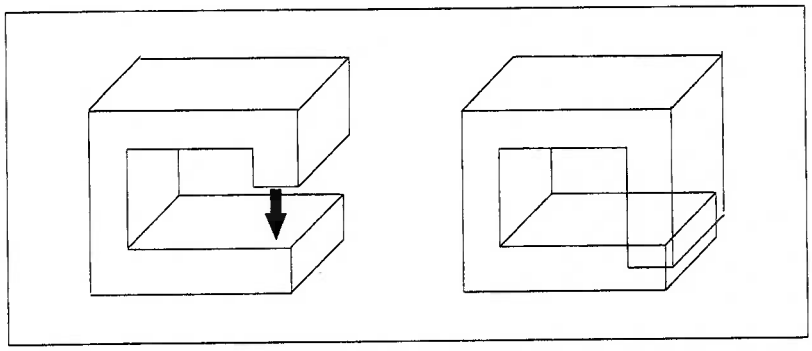


Figura 2-18 Um sólido auto-interceptante.

Um operador local pode ser considerado uma extensão das Operações de Euler. Um operador local, ao ser acionado pelo usuário, definirá uma seqüência de Operadores de Euler que a implemente. Antes de definirmos os operadores locais, apresentaremos algumas rotinas básicas de modelagem.

- Cria arco de circunferência: esta rotina define uma aproximação poligonal de um arco de circunferência, a partir de um ponto fornecido. Assim, o algoritmo que implementa esta rotina pode ser definido utilizando-se apenas o Operador **MEV** (Figura 2-19). Para criar uma circunferência, será necessário definir um arco de 360° e o último Operador de Euler **MEV** deve ser substituído por um Operador **MEF** (vide Figura 2-20).

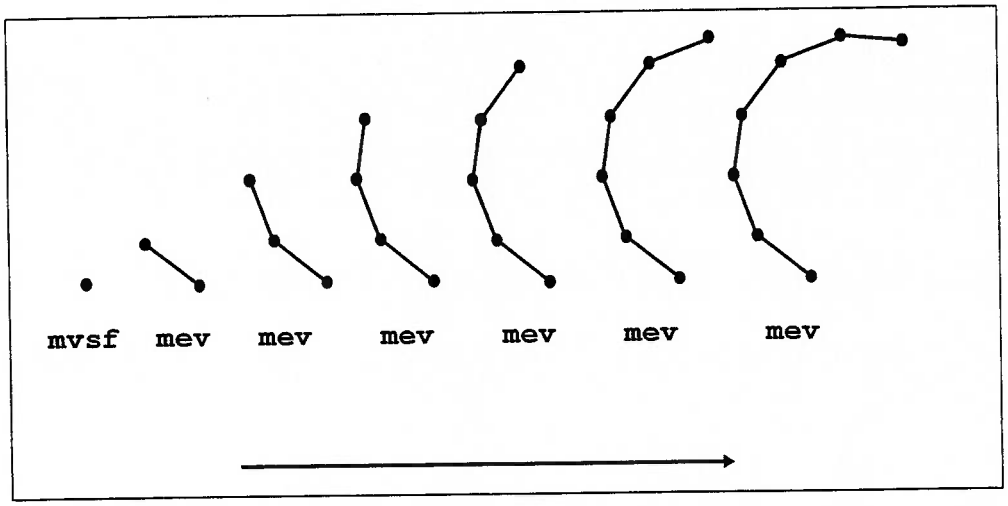


Figura 2-19 Seqüência de Operadores de Euler que implementam a criação de um arco.

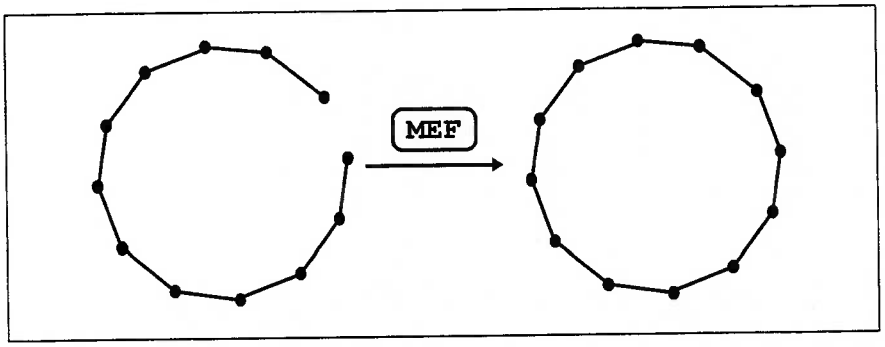


Figura 2-20 Finalização da criação de uma circunferência.

- Extrusão translacional: a extrusão translacional deve ser aplicada a uma face e será realizada segundo um sentido específico. Como exemplo de sólido primitivo criado por meio deste operador temos o cubo (Figura 2-21(c)), o cilindro (Figura 2-21(e)) e o prisma (Figura 2-21(b)). Se estiver disponível uma função que degenere uma face em um ponto, será possível criarmos sólidos primitivos como o cone (Figura 2-21(a)) e a pirâmide (Figura 2-21(d)).

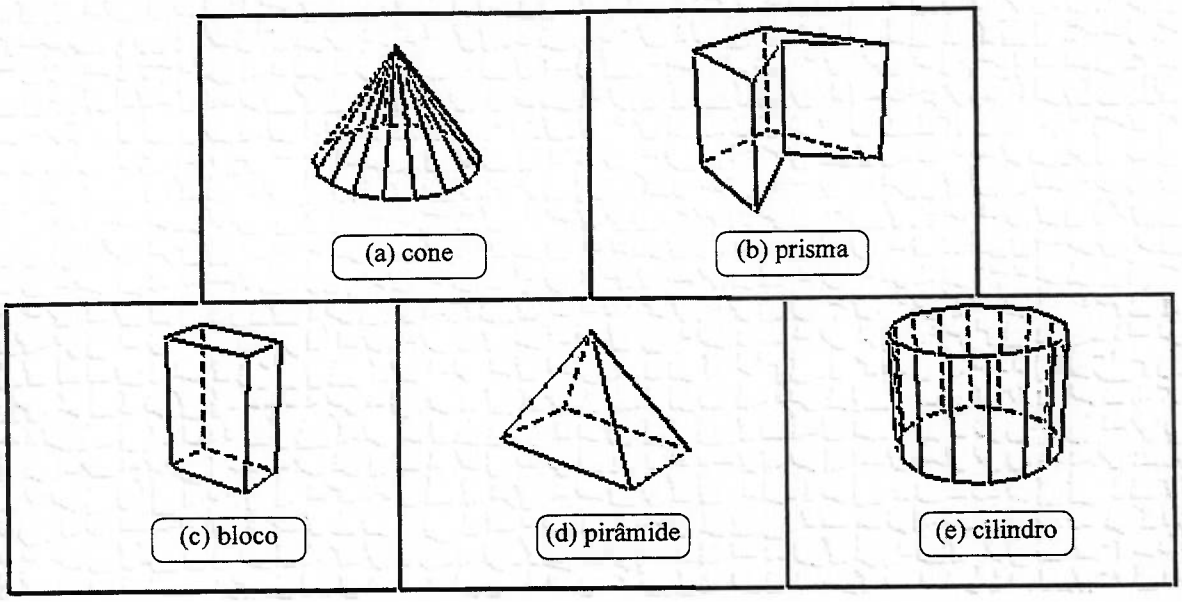


Figura 2-21 Sólidos obtidos por extrusão translacional.

- Extrusão rotacional: a operação de extrusão rotacional consiste em girar uma figura bidimensional, com contorno aberto ou fechado, ao redor de um eixo, definindo um sólido de revolução. Como exemplos de sólidos criados por meio deste operador temos a esfera, o peão de xadrez e o toróide. (Figura 2-22).

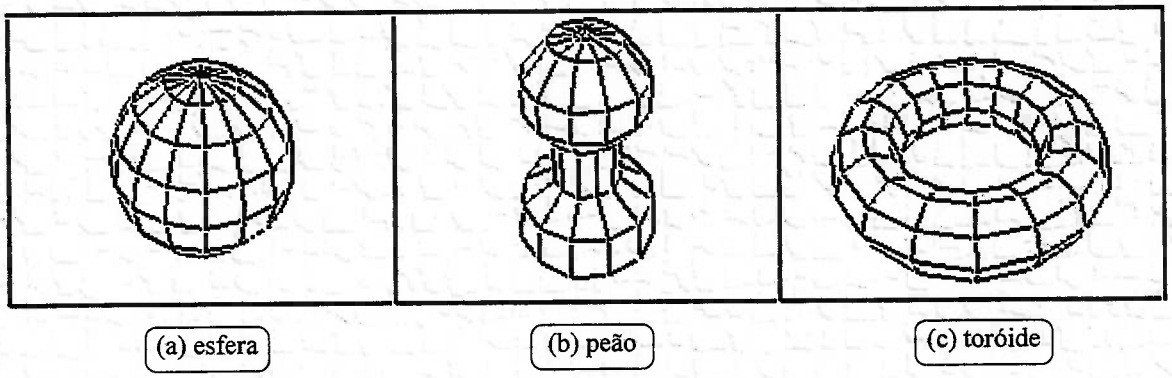


Figura 2-22 Sólidos gerados por extrusão rotacional.

- Suavização de arestas e vértices: este é outro exemplo de operador local. Esta operação é utilizada para melhorar a resistência, a manufaturabilidade, a segurança e a aparência de um sólido. CHIYOKURA e KIMURA (1988) propuseram operadores de suavização. A Figura 2-23 exibe um sólido com algumas arestas suavizadas utilizando-se este operador.

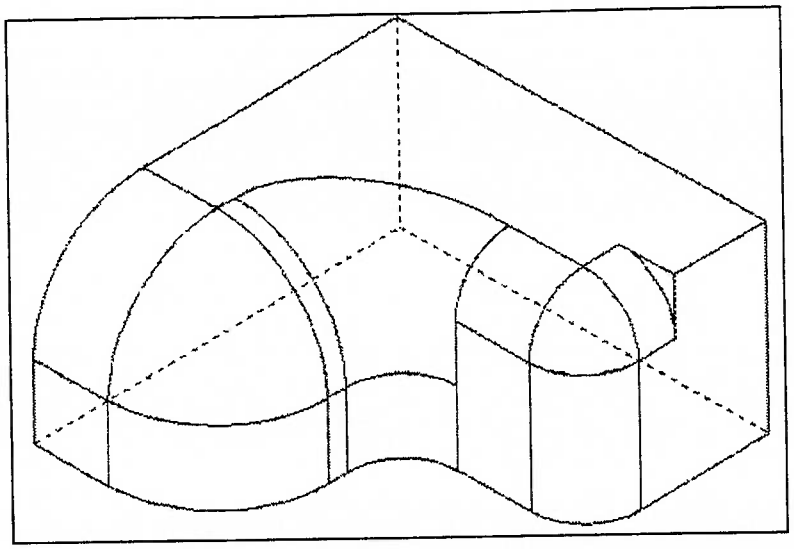


Figura 2-23 Sólido com arestas e vértices suavizados.

2.4.4. Operadores Globais (Avançados)

2.4.4.1.Secção por Plano

A operação de secção por plano é a primeira operação global que será apresentada. Esta função é muito utilizada nos sistemas CAD atuais. Dentre as suas possíveis aplicações podemos citar: geração de isolinhas com o objetivo de definir camadas para estereolitografia, definição de seções de corte para facilitar a sua visualização, dentre outras.

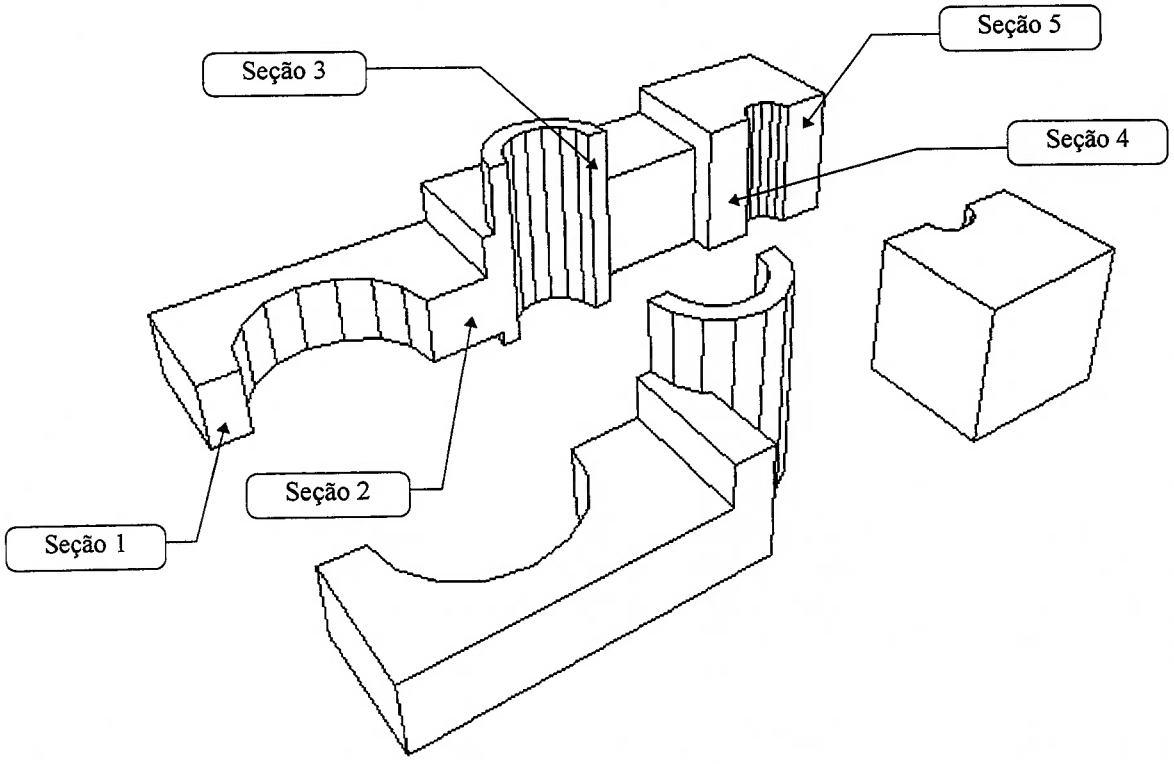


Figura 2-24 Um sólido seccionado.

Devido ao seu formalismo, ao aplicarmos este operador a um sólido, sempre obteremos como resultado dois sólidos. Portanto, é preciso prever as situações em que um dos dois sólidos resultantes corresponda a um sólido nulo (indicando que o sólido original está posicionado completamente de um lado do plano de corte), ou que o sólido resultante corresponde a um sólido com vários shells, conforme o exemplo ilustrado na Figura 2-24.

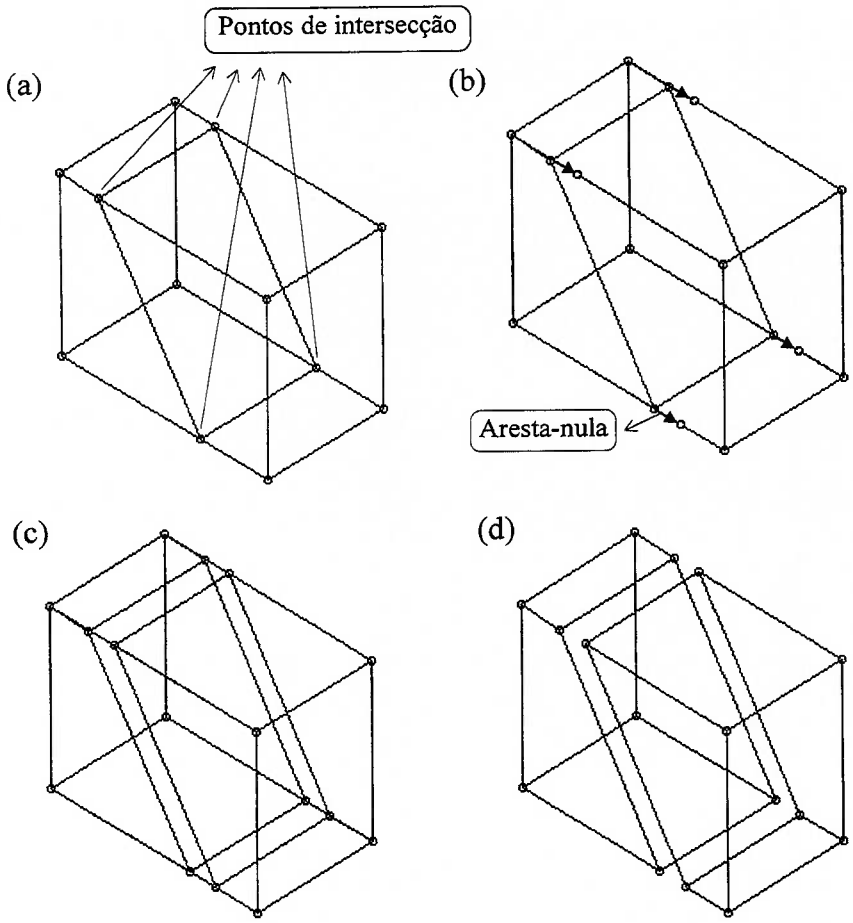


Figura 2-25 Etapas da Secção por Plano.

Em um modelador de sólidos poliédrico o algoritmo de secção por plano pode ser dividido em cinco etapas (MÄNTYLÄ, 1988):

1. Determinação dos pontos de intersecção e inserção de vértices nestes pontos: o algoritmo verifica se os vértices das arestas do sólido estão em lados opostos ao plano de corte. Se isto ocorre, insere-se um novo vértice com as coordenadas do ponto de intersecção da aresta com o plano. Este vértice é guardado em uma lista para posterior processamento. Há casos em que um dos vértices da aresta já está sobre o plano de corte. Se isto ocorrer, armazena-se este mesmo vértice. (vide a Figura 2-25 (a)).

2. Classificação dos vértices de intersecção: os vértices armazenados são classificados para processar casos especiais como arestas que estão sobre o plano de corte ou vértices sobre o plano de corte (Figura 2-25 (a)).
3. Inserção de arestas nulas: nos pontos de intersecção do sólido com plano são inseridas arestas "sem comprimento geométrico" mas com significado topológico. (vide Figura 2-25 (b)).
4. Junção e eliminação de arestas nulas: estas arestas marcam os locais onde é necessário realizar as uniões entre suas extremidades para formar as novas faces. Uma vez terminadas as junções as arestas nulas são removidas. (Figura 2-25 (c, d)).
5. Atualização das estruturas de dados: a última etapa do algoritmo atualiza as listas de arestas e vértices de cada sólido resultante (vide a Figura 2-26).

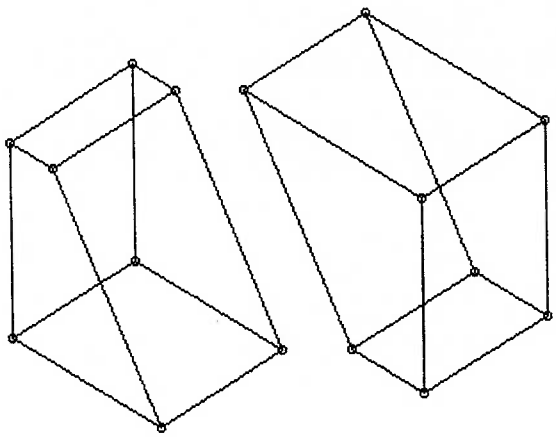


Figura 2-26 Resultado final de uma Secção por Plano.

2.4.4.2.Operações Booleanas entre Sólidos

Algoritmos para determinar a união, intersecção ou diferença de dois sólidos podem ser utilizados por modeladores de representação B-Rep de representação híbrida[1]. Eles também podem ser utilizados para converter sólidos representados por árvores CSG para uma equivalente representação B-Rep.

Conceitualmente, estes algoritmos não são difíceis, mas sua implementação requer um substancial trabalho por várias razões. Considerar as várias posições especiais de incidência dos elementos básicos em três dimensões pode ser muito trabalhoso. Ainda mais, a presença de

[1] Modeladores que suportam mais que uma representação.

superfícies curvas introduzem problemas matemáticos não triviais, e desde que a maioria dos algoritmos requer técnicas numéricas, é inevitável que surjam problemas de precisão numérica e estabilidade de cálculos.

Um grande segmento da literatura requer que as superfícies dos objetos representados por representações B-Rep sejam fechadas e *manifolds*[2] orientados contidos no espaço tridimensional. Um exemplo de um sólido não *manifold* está ilustrado na Figura 2-27 (a), resultado da união de dois braços L. O problema na aresta (PQ) é que ela é adjacente a quatro faces. É imediato notar que a aplicação de operadores booleanos sobre dois sólidos *manifold* pode criar um resultado não *manifold*. Três propostas para tratar estruturas não *manifold* foram desenvolvidas:

1. Objetos necessitam ser *manifold*, logo operações com resultados não *manifold* não são permitidas e são consideradas como erro.
2. Objetos são *manifold* topológicos, mas seu envolvimento no espaço tridimensional permite a coincidência geométrica de estruturas separadas topologicamente.
3. Objetos não *manifold* são permitidos tanto como entrada quanto como saída.

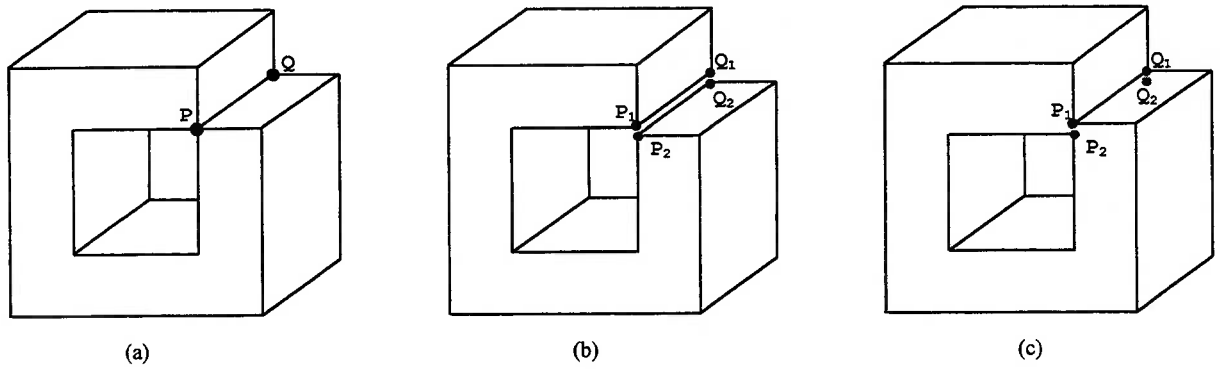


Figura 2-27 Resultado na união de dois braços L.

Na segunda proposta é necessário fornecer uma interpretação topológica para as estruturas não *manifold*. No exemplo da Figura 2-27, interpreta-se uma aresta não *manifold* como sendo duas arestas separadas que coincidiram. As duas possibilidades existentes estão ilustradas na Figura 2-27. Qual a interpretação mais natural? Por questões de robustez escolhe-se a interpretação na qual a superfície é triangularizável sem triângulos degenerados. Note que esta triangularização só é possível para o caso (b), e não para o caso (c).

[2] Intuitivamente, ao redor de todos os pontos de uma superfície *manifold* existe uma vizinhança que é homeomórfica ao plano, isto é, pode-se deformar a superfície *manifold* localmente em um plano e não será possível identificar pontos vizinhos separados.

A Figura 2-28 ilustra um exemplo dos quatro operadores booleanos que podem ser aplicados a dois sólidos.

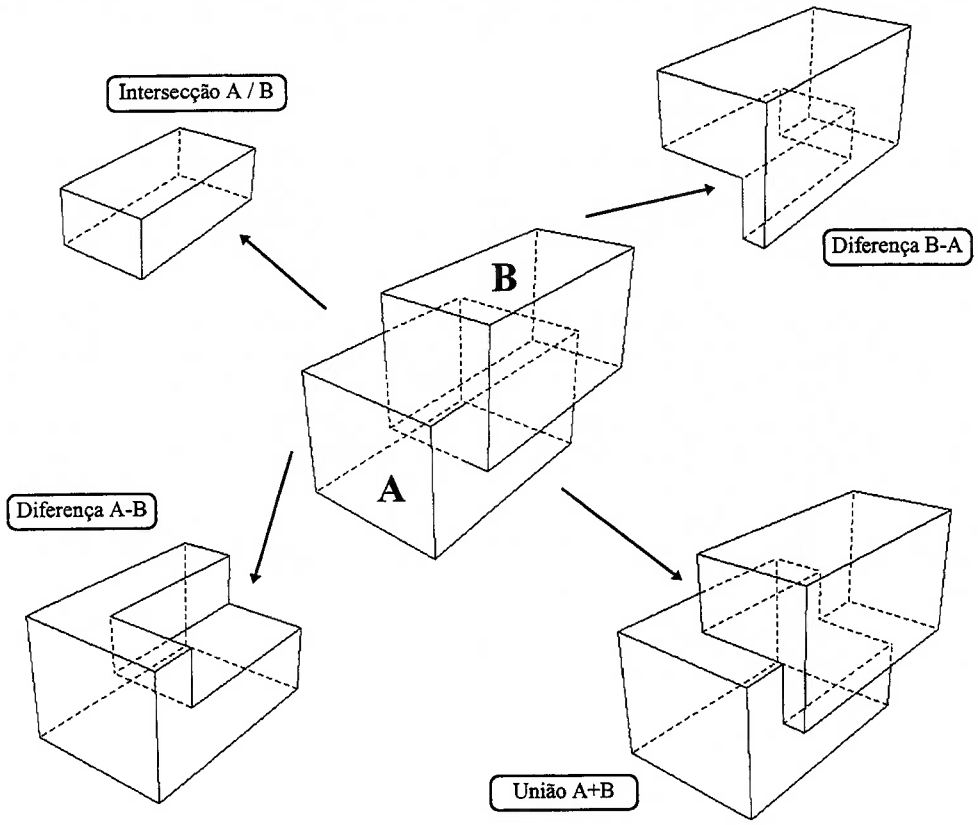


Figura 2-28 Operações booleanas.

De forma similar ao algoritmo de secção por plano o algoritmo de operações booleanas pode ser dividido em cinco etapas (MÄNTYLÄ, 1988):

1. Determinação dos pontos de intersecção: o algoritmo verifica se as arestas do sólido A interceptam as faces do sólido B e vice-versa. Se isto ocorre insere-se um novo vértice com as coordenadas do ponto de intersecção. Os vértices de intersecção são armazenados em uma lista (Figura 2-29 (a)) .
2. Classificação dos vértices de intersecção: eles são classificados de acordo com a operação booleana que estiver sendo realizada.
3. Inserção de arestas nulas: nos vértices de intersecção são inseridas arestas sem comprimento geométrico mas com significado topológico. (Figura 2-29 (b)).
4. Junção e eliminação das arestas nulas: as extremidades das arestas são unidas a fim de formar as novas faces do sólido resultante (Figura 2-29 (c, d)).
5. Atualização das estruturas de dados.

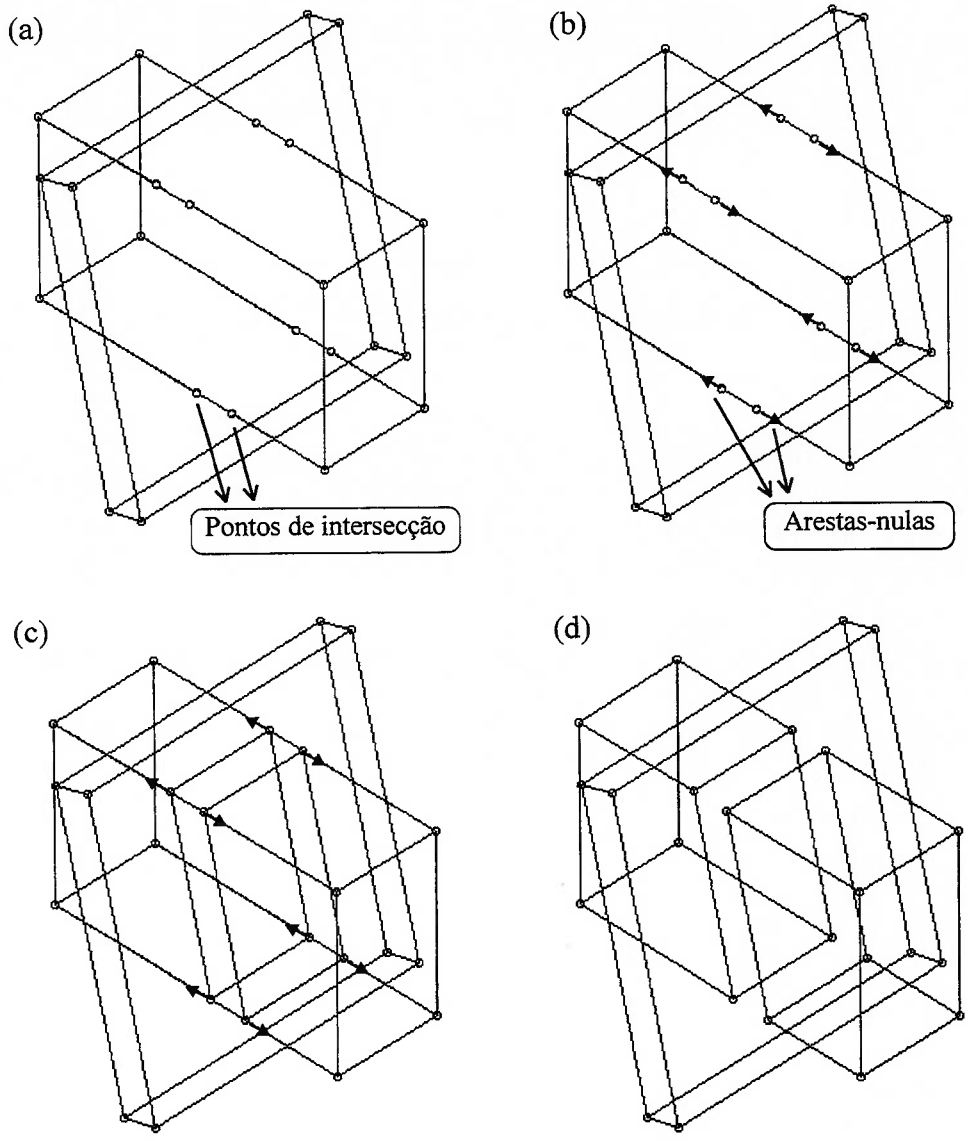


Figura 2-29 Etapas de uma Operação Booleana de união entre sólidos.

3. Modelagem geométrica

3.1. Introdução

Curvas espaciais ou tridimensionais possuem um papel importante na engenharia, projeto e manufatura de uma grande gama de produtos, como automóveis, cascos de navios, fuselagem e asas de aviões, pás de hélices propulsoras, sapatas, garrafas, edifícios, etc.

Uma curva pode ser representada por um conjunto de pontos (representação sintética). Desde que os pontos estejam apropriadamente espaçados, eles podem ser conectados por segmentos de linha, fornecendo uma representação visual adequada para a curva. Em outras aplicações, uma representação analítica (representação segundo a equação associada à curva) será mais adequada. A representação analítica possui muitas vantagens sobre a representação sintética, como precisão, armazenamento mais compacto e facilidade de cálculo de pontos intermediários.

Matematicamente, tanto a forma paramétrica quanto a não-paramétrica são utilizadas para a representação analítica de uma curva. A representação não-paramétrica pode ser tanto implícita quanto explícita. Uma representação não-paramétrica explícita é dada por:

$$z = f(x,y) \tag{3.1}$$

Desta forma, para cada par de valores (x,y) apenas um valor de z é obtido. Conseqüentemente, curvas fechadas, como circunferências, não podem ser representadas explicitamente. Uma representação não-paramétrica implícita é dada por:

$$f(x,y,z) = 0 \tag{3.2}$$

Tanto a representação da curva não-paramétrica explícita quanto a implícita são dependentes do sistema de coordenadas. Além do mais, quando pontos da curva são calculados, segundo incrementos iguais de x, y ou z, os pontos freqüentemente estarão igualmente distribuídos, segundo o comprimento da curva. Esta desigualdade na distribuição dos pontos afetará a precisão da representação gráfica. Neste trabalho vamos limitar-nos ao estudo das curvas sintéticas e mais especificamente à curva de Bézier.

3.2. Representação Paramétrica de Curvas Sintéticas

Na forma paramétrica cada ponto sobre a curva é representado por um único parâmetro. Para uma curva tridimensional, com t como parâmetro, as coordenadas cartesianas de um ponto sobre a curva são dadas por:

$$\begin{aligned} x &= x(t) \\ y &= y(t) \\ z &= z(t) \end{aligned} \tag{3.3}$$

O vetor posição de um ponto sobre a curva é dado por:

$$\mathbf{P}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} \tag{3.4}$$

A representação mais simples para uma curva paramétrica ocorre para um segmento de reta. Dados dois pontos extremos do segmento de $\mathbf{P}_1$ e $\mathbf{P}_2$, a representação paramétrica para o segmento de reta é dada por:

$$\mathbf{P}(t) = \mathbf{P}_1 + t.(\mathbf{P}_2 - \mathbf{P}_1) \qquad 0 \le t \le 1 \tag{3.5}$$

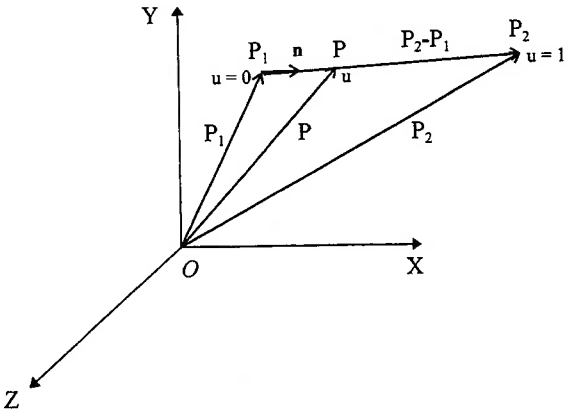


Figura 3-1 Reta com representação analítica.

Como $\mathbf{P}(t)$ é um vetor posição, cada um dos componentes de $\mathbf{P}(t)$ possui uma representação paramétrica:

$$\left.\begin{aligned} x(t) &= x_1 + t.(x_2 - x_1) \\ y(t) &= y_1 + t.(y_2 - y_1) \\ z(t) &= z_1 + t.(z_2 - z_1) \end{aligned}\right\} \qquad 0 \le t \le 1 \tag{3.6}$$

Existe uma grande variedade de formas de definir uma curva paramétrica. Cada uma delas pode ser classificada como sendo interpolação ou aproximação de pontos. No caso de uma

curva interpolada, esta deverá passar por um conjunto de pontos fornecidos P_i, em uma seqüência ordenada (vide Figura 3-2).

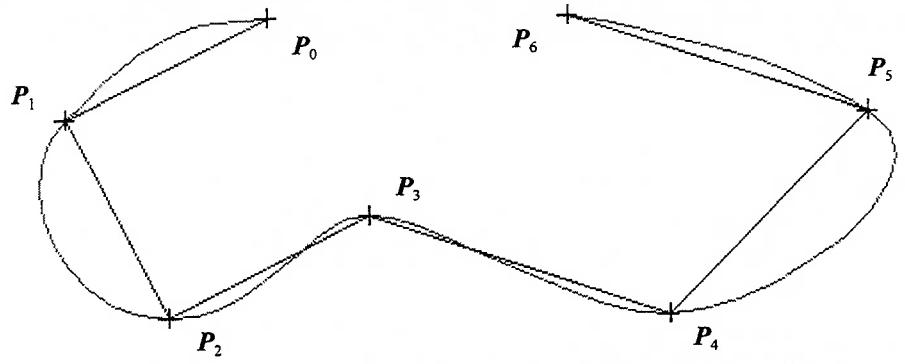


Figura 3-2 Curva obtida pela interpolação de pontos.

Para as técnicas baseadas na aproximação a curva deverá passar "próxima" aos conjunto de pontos fornecidos, que serão chamados pontos de controle (vide Figura 3-3). O significado exato de "próximo" dependerá de cada técnica em particular.

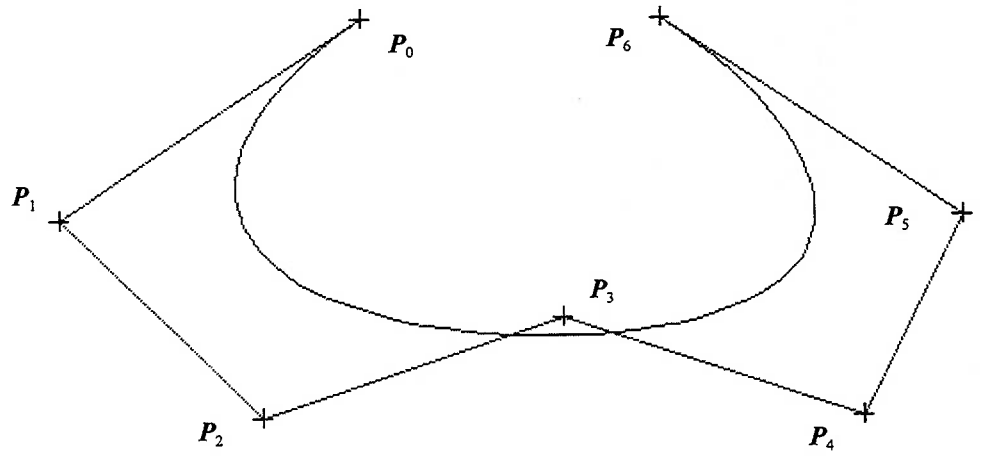


Figura 3-3 Curva obtida pela aproximação de pontos.

Em ambos os casos, ao movimentarmos um ponto do conjunto fornecido, a curva é alterada. Neste trabalho vamos estudar com maior detalhe as curvas de Bézier que são curvas aproximadoras. Um dos fatores que determina o uso das curvas em uma aplicação específica é a influência do conjunto de pontos sobre a curva. O projetista pode preferir controlar a forma da curva de modo local ou global, alterando a posição de um dos pontos. Se a mudança deste ponto resulta em uma mudança localizada da curva, nas vizinhanças do ponto, temos um controle local da curva; em caso contrário temos um controle global. A seguir detalharemos as Curvas de Hermite e as Curvas de Bézier.

3.2.1. Curva de Hermite

A interpolação de Hermite determina uma curva que passa por dois pontos fornecidos, além de que os vetores tangentes nestes pontos também são fornecidos (vide Figura 3-4).

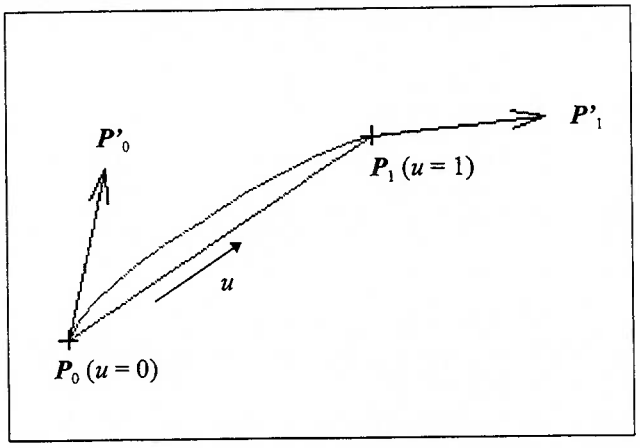


Figura 3-4 Curva de Hermite.

A equação paramétrica da curva de Hermite é dada por:

$$\mathbf{P}(u) = \sum_{i=0}^{3} \mathbf{C}_i u^i, \qquad 0 \le u \le 1 \tag{3.6}$$

onde u é o parâmetro e $\mathbf{C}_i$ são os coeficientes do polinômio. Na forma escalar esta equação é escrita como:

$$\begin{aligned} x(u) &= C_{3x}u^3 + C_{2x}u^2 + C_{1x}u + C_{0x} \\ y(u) &= C_{3y}u^3 + C_{2y}u^2 + C_{1y}u + C_{0y} \\ z(u) &= C_{3z}u^3 + C_{2z}u^2 + C_{1z}u + C_{0z} \end{aligned} \tag{3.7}$$

Na forma vetorial expandida a Equação (3.6) pode ser escrita como:

$$\mathbf{P}(u) = \mathbf{C}_3 u^3 + \mathbf{C}_2 u^2 + \mathbf{C}_1 u + \mathbf{C}_0 \tag{3.8}$$

Esta equação pode ser escrita também na forma matricial como:

$$\mathbf{P}(u) = \mathbf{U}^T \mathbf{C} \tag{3.9}$$

onde $\mathbf{U} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}^T$ e $\mathbf{C} = \begin{bmatrix} \mathbf{C}_3 & \mathbf{C}_2 & \mathbf{C}_1 & \mathbf{C}_0 \end{bmatrix}^T$. $\mathbf{C}$ é o vetor de coeficientes. O vetor tangente à curva em um determinado ponto é dado pela diferenciação da Equação (3.6) em relação a u, resultando em:

$$\mathbf{P}'(u) = \sum_{i=1}^{3} \mathbf{C}_i i u^{i-1}, \qquad 0 \le u \le 1 \tag{3.10}$$

Para encontrarmos os coeficientes $\mathbf{C}_i$, consideremos a curva de Hermite com extremidades $\mathbf{P}_0$ e $\mathbf{P}_1$ mostradas na Figura 3-4. Aplicando as condições de contorno ($\mathbf{P}_0, \mathbf{P}_0'$, em $u = 0$ e $\mathbf{P}_1$, $\mathbf{P}_1'$ em $u = 1$), temos:

$$\begin{aligned}\mathbf{P}_0 &= \mathbf{C}_0 \\ \mathbf{P}_0' &= \mathbf{C}_1 \\ \mathbf{P}_1 &= \mathbf{C}_3 + \mathbf{C}_2 + \mathbf{C}_1 + \mathbf{C}_0 \\ \mathbf{P}_1' &= 3\mathbf{C}_3 + 2\mathbf{C}_2 + \mathbf{C}_1\end{aligned} \tag{3.11}$$

Resolvendo as quatro equações para os coeficientes, obtemos:

$$\begin{aligned}\mathbf{C}_0 &= \mathbf{P}_0 \\ \mathbf{C}_1 &= \mathbf{P}_0' \\ \mathbf{C}_2 &= 3(\mathbf{P}_1 - \mathbf{P}_0) - 2\mathbf{P}_0' - \mathbf{P}_1' \\ \mathbf{C}_3 &= 2(\mathbf{P}_0 - \mathbf{P}_1) + \mathbf{P}_0' + \mathbf{P}_1'\end{aligned} \tag{3.11}$$

Substituindo a Equação (3.11) na Equação (3.8) e rearranjando os termos:

$$\begin{aligned}\mathbf{P}(u) &= (2u^3 - 3u^2 + 1)\mathbf{P}_0 + (-2u^3 + 3u^2)\mathbf{P}_1 \\ &+ (u^3 - 2u^2 + u)\mathbf{P}_0' + (u^3 - u^2)\mathbf{P}_1' \qquad 0 \le u \le 1\end{aligned} \tag{3.12}$$

$\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P'}_0$, e $\mathbf{P'}_1$ são chamados coeficientes geométricos. O vetor tangente torna-se:

$$\begin{aligned}\mathbf{P}'(u) &= (6u^2 - 6u)\mathbf{P}_0 + (-6u^2 + 6u)\mathbf{P}_1 \\ &+ (3u^2 - 4u + 1)\mathbf{P}_0' + (3u^2 - 2u)\mathbf{P}_1', \qquad 0 \le u \le 1\end{aligned} \tag{3.13}$$

A Equação (3.13) pode ser escrita na forma matricial como:

$$\mathbf{P}(u) = \mathbf{U}^T [M_H] \mathbf{Vcc}, \qquad 0 \le u \le 1 \tag{3.14}$$

onde $[M_H]$ é a matriz de Hermite e **Vcc** é vetor geométrico (ou vetor de condições de contorno). Ambos são dados por:

$$[M_H] = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.15}$$

$$\mathbf{Vcc} = \begin{bmatrix} \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_0' & \mathbf{P}_1' \end{bmatrix}^T \tag{3.16}$$

Comparando a Equação (3.9) e a Equação (3.14) pode-se observar que $\mathbf{C} = [M_H]\mathbf{Vcc}$ ou $\mathbf{Vcc} = [M_H]^{-1}\mathbf{C}$ onde:

$$[M_H]^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \tag{3.17}$$

A Equação (3.12) descreve a curva de Hermite em termos dos seus pontos extremos e dos seus vetores tangentes. Ela também mostra que a forma da curva pode ser controlada modificando-se seus pontos extremos ou seus vetores tangentes. Se os pontos extremos são fixos no

espaço, o projetista pode controlar a forma da curva mudando as magnitudes e direções dos vetores tangentes $\mathbf{P}'_0$ e $\mathbf{P}'_1$.

3.2.2. Curvas de Bézier

As curvas e superfícies de Bézier são creditadas a P. Bézier da firma automobilística francesa Regie Renault que desenvolveu e utilizou estas curvas no sistema de software chamado UNISURF que foi utilizado pelos projetistas para definir os painéis que compõem a carroceria de vários automóveis Renault. Estas curvas, chamadas curvas de Bézier, foram também desenvolvidas independentemente por P. DeCasteljau da companhia de automóveis Citröen. O sistema UNISURF de Bézier foi publicado primeiro na literatura; esta é a razão das curvas levarem agora o nome de Bézier (BARTELS, 1987 e ZEID, 1991).

As principais diferenças entre as curvas de Bézier e as curvas de Hermite são:

1. A forma da curva de Bézier é controlada somente pelos seus pontos de controle. As primeiras derivadas não são utilizadas no desenvolvimento da curva como no caso da curva de Hermite. Isto permite ao projetista uma maior sensibilidade sobre a relação pontos de controle / curva.
2. A ordem ou o grau da curva de Bézier é variável e relaciona-se com o número de pontos de controle que a definem; n+1 pontos definem uma curva de grau n que permite uma continuidade de maior ordem. Isto não ocorre para as curvas de Hermite onde o grau é sempre 3.
3. A curva de Bézier é mais suave que a curva de Hermite porque possui derivadas de maior ordem (ZEID, 1991).

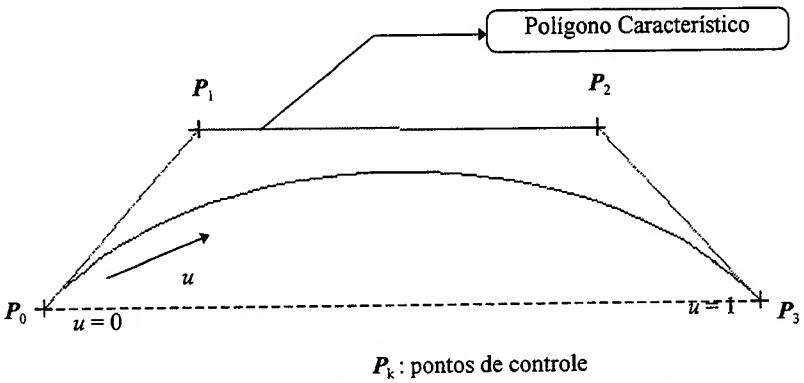


Figura 3-5 Curva de Bézier cúbica (nomenclatura) (ZEID, 1991).

A curva de Bézier é definida em termos da localização dos n+1 pontos de controle. Eles formam os vértices do que chamamos de polígono de controle ou polígono característico de Bézier como é mostrado na Figura 3-5. Somente o primeiro e o último ponto de controle pertencem à

curva. A curva é sempre tangente ao primeiro e ao último segmento do polígono característico. A curva é sempre interna ao polígono característico. Estas três observações permitem ao usuário prever a forma da curva uma vez dados os seus pontos de controle.

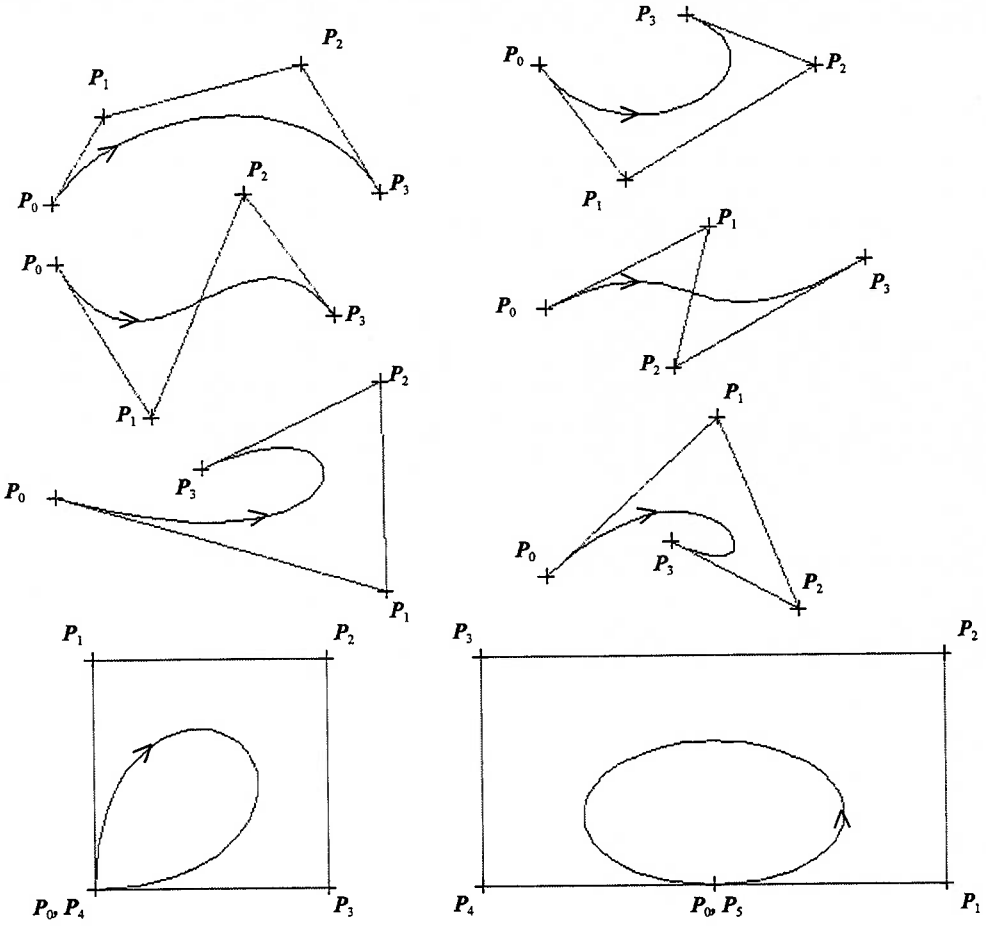


Figura 3-6 Curvas de Bézier cúbicas para vários conjuntos de pontos de controle (ZEID, 1991).

A Figura 3-6 ilustra várias curvas de Bézier cúbicas para diversos conjuntos de pontos de controle. Matematicamente, para n+1 pontos de controle, a curva de Bézier é definida pelo seguinte polinômio de grau n:

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i B_{i,n}(u), \qquad 0 \le u \le 1 \tag{3.18}$$

onde $\mathbf{P}(u)$ é um ponto da curva e $\mathbf{P}_i$ é o i-ésimo ponto de controle. $B_{i,n}(u)$ são os polinômios de Bernstein, que atuam como funções de base ou de ajuste para a curva de Bézier e são dados por:

$$B_{i,n}(u) = C(n,i)u^i(1-u)^{n-i} \tag{3.19}$$

onde $C(n,i)$ é o coeficiente binomial:

$$C(n,i) = \frac{n!}{i!(n-i)!} \tag{3.20}$$

Utilizando a Equação (3.18) e a Equação (3.19) e observando que $C(n,0) = C(n,n) = 1$, a Equação (3.18) pode ser expandida para:

$$\begin{aligned}\mathbf{P}(u) = \mathbf{P}_0(1-u)^n + \mathbf{P}_1 C(n,1)u(1-u)^{n-1} + \mathbf{P}_2 C(n,2)u^2(1-u)^{n-2} \\ +\cdots+\mathbf{P}_{n-1}C(n,n-1)u^{n-1}(1-u) + \mathbf{P}_n u^n, \quad 0 \le u \le 1\end{aligned} \tag{3.21}$$

As características da curva de Bézier são baseadas nas propriedades dos polinômios de Bernstein e podem ser resumidas da seguinte forma:

1. A curva interpola o primeiro e o último ponto de controle; isto é, ela passa por $\mathbf{P}_0$ e $\mathbf{P}_n$, substituindo u=0 e u=1 respectivamente na Equação (3.21).
2. A curva é tangente ao primeiro e ao segundo segmentos do polígono característico. Utilizando a Equação (3.18) e a Equação (3.19), a r-ésima derivada nos pontos inicial e final da curva pode ser dada respectivamente por:

$$\mathbf{P}^r(0) = \frac{n!}{(n-r)!}\sum_{i=0}^{r}(-1)^{r-i}C(r,i)\mathbf{P}_i \tag{3.22}$$

$$\mathbf{P}^r(1) = \frac{n!}{(n-r)!}\sum_{i=0}^{r}(-1)^{r-i}C(r,i)\mathbf{P}_{n-i} \tag{3.23}$$

Logo as derivadas primeiras nos pontos extremos são:

$$\mathbf{P}'(0) = n(\mathbf{P}_1 - \mathbf{P}_0) \tag{3.24}$$

$$\mathbf{P}'(1) = n(\mathbf{P}_n - \mathbf{P}_{n-1}) \tag{3.25}$$

onde ($\mathbf{P}_1$-$\mathbf{P}_0$) e ($\mathbf{P}_n$-$\mathbf{P}_{n-1}$) definem o primeiro e o último segmentos do polígono característico da curva.

3. A curva é simétrica com respeito a u e $(1-u)$. Isto significa que a seqüência de pontos de controle que definem a curva pode ser revertida sem alterar a forma da curva; isto é, revertendo a direção de parametrização não ocorre mudança na forma da curva. Isto pode ser conseguido substituindo-se $1-u=v$ na Equação (3.21) e notando que $C(n,i) = C(n,n-i)$. Isto é resultante do fato de que $B_{i,n}(u)$ e $B_{n-i,n}(u)$ são simétricos se plotados como função de u.
4. O polinômio de interpolação $B_{i,n}(u)$ tem valor máximo de $C(n, i)(i/n)^i(1-i/n)^{n-i}$, que ocorre em $u=i/n$. (Obtido da equação $d((B_{i,n})/du = 0)$. Isto implica que cada ponto de controle influencia de forma mais acentuada um determinado trecho da curva. Por exemplo para uma curva de Bézier cúbica, os pontos $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$ e $\mathbf{P}_3$ têm maior influência nos arredores de u = 0, 1/3, 2/3 e 1 respectivamente.

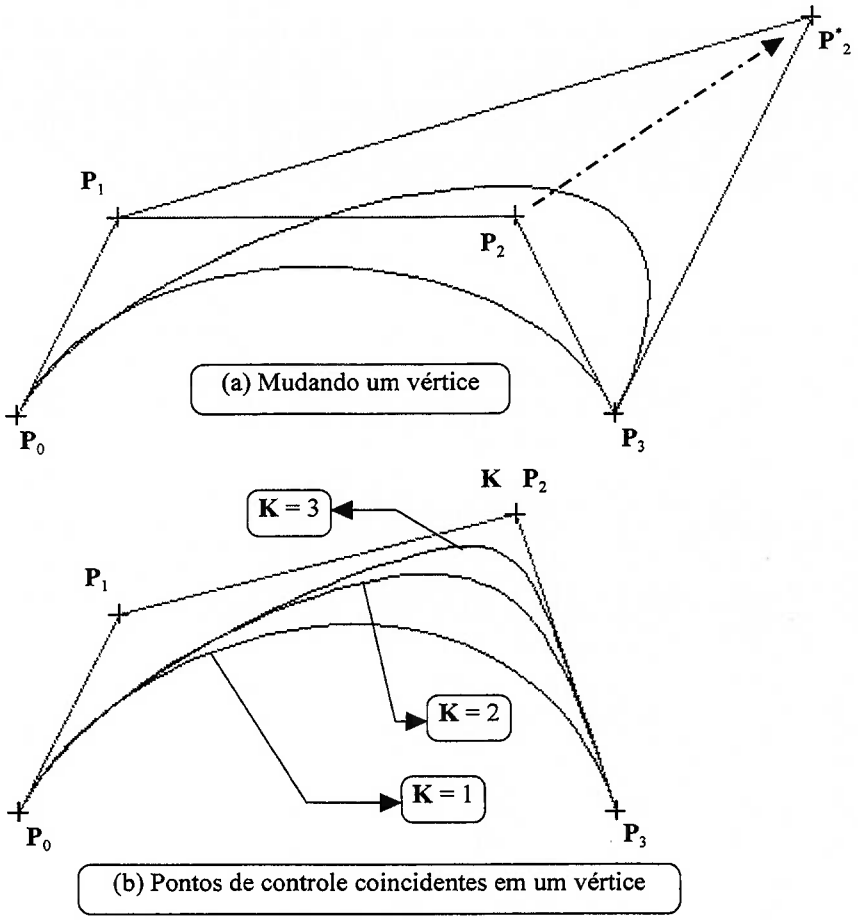


Figura 3-7 Modificações sobre uma curva de Bézier cúbica (ZEID, 1991).

5. A forma da curva pode ser modificada pelo reposicionamento de um ou mais vértices do polígono, ou mantendo o polígono fixo e definindo múltiplos pontos coincidentes em um mesmo vértice. A Figura 3-7(a) mostra um ponto $\mathbf{P}_2$ sendo movimentado para uma nova posição $\mathbf{P}^*_2$ e Figura 3-7(b) ilustra um vértice $\mathbf{P}_2$ com multiplicidade *K*. Quanto maior a multiplicidade em $\mathbf{P}_2$ mais a curva é "puxada" para este vértice.
6. Uma curva de Bézier fechada pode ser facilmente obtida fechando-se o polinômio característico. A Figura 3-6 ilustra alguns exemplos de curvas de Bézier fechadas.
7. Para qualquer valor válido de *u*, a soma das funções $B_{i,n}(u)$ associadas aos pontos de controle é igual à unidade, para qualquer grau de curva de Bézier.

Até agora somente um segmento de curva de Bézier foi considerado. Entretanto nas aplicações práticas, muitas vezes é necessário compor vários segmentos de curva. Nestas aplicações é desejável manter a continuidade entre as curvas. A Figura 3-8 mostra dois conjuntos de pontos $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ e $\mathbf{P}_4$ e $\mathbf{P}_5$, $\mathbf{P}_6$, $\mathbf{P}_7$ e $\mathbf{P}_8$. Para obter a continuidade de ordem zero é suficiente colocar os pontos extremos coincidentes, $\mathbf{P}_4$ na Figura 3-8 (a). Para obter a a continuidade de primeira ordem é preciso que as tangentes do final do primeiro segmento e do início do segundo segmento tenham a mesma

direção e sejam proporcionais segundo uma constante. No exemplo, os pontos $\mathbf{P}_3$, $\mathbf{P}_4$ e $\mathbf{P}_5$ devem ser colineares, como mostra a Figura 3-8 (b). Utilizando a Equação (3.24) e a Equação (3.25) pode-se escrever:

$$\mathbf{P}_4 - \mathbf{P}_3 = \tfrac{4}{3}(\mathbf{P}_5 - \mathbf{P}_4) \tag{3.26}$$

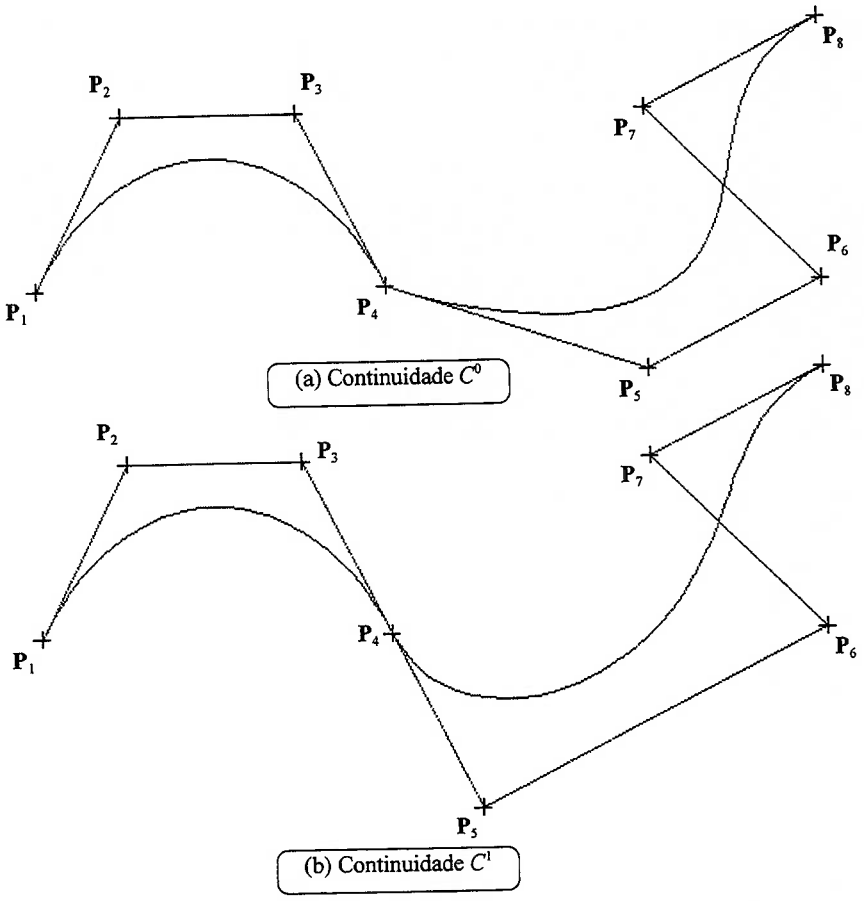


Figura 3-8 Ajuste de segmentos de curva de Bézier (ZEID, 1991).

Uma característica muito útil é a propriedade da envoltória convexa. A envoltória convexa é definida como sendo o maior polígono convexo definido pelo conjunto de pontos de controle. É possível demonstrar que a envoltória convexa sempre contém a curva de Bézier. A Figura 3-9 ilustra alguns exemplos de envoltória convexa.

As curvas de Bézier parecem ser superiores às curvas de Hermites, mas possuem algumas desvantagens. Primeiro, a curva não passa pelos pontos de controle, fato que pode ser inconveniente para alguns projetistas. Segundo, esta curva não possui controle local, mas somente controle de natureza global. Se um ponto de controle é mudado, toda a curva é modificada. Em conseqüência, o projetista não pode modificar apenas parte da curva.

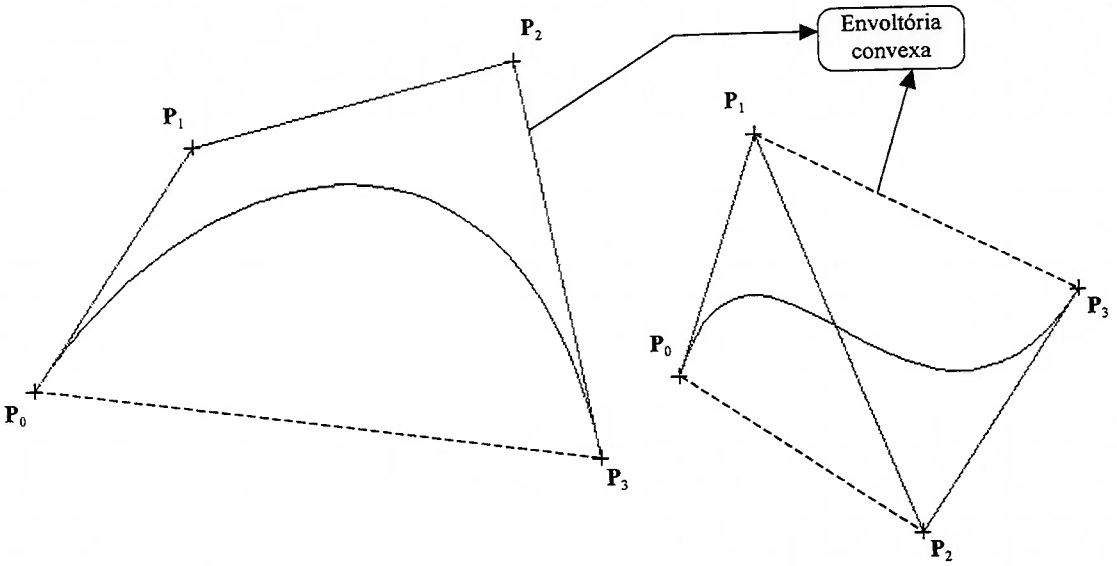


Figura 3-9 Envoltória convexa de uma curva de Bézier.

3.2.3. Geração dos pontos da Curva

Para desenvolver uma curva de Bézier ou de Hermite, utiliza-se sua equação paramétrica para gerar pontos consecutivos que são interligados por segmentos de reta. A geração dos pontos da curva pode ser realizada pela simples substituição sucessiva de valores do parâmetro u na equação paramétrica. Este processo é ineficiente porque requer a realização de muitas operações matemáticas a cada novo ponto, principalmente multiplicações. Os métodos incrementais são comprovadamente mais eficientes.

A técnica da diferença-para-frente (*forward difference technique*) para gerar uma curva com equação polinomial a intervalos iguais dos seus parâmetros é um método incremental que elimina o uso de multiplicações. Como exemplo considere a aplicação do método sobre um polinômio de terceiro grau. Se uma curva é dada por:

$$\mathbf{P}(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d}, \qquad 0 \le u \le 1 \tag{3.27}$$

e se n+1 pontos espaçados igualmente devem ser gerados para uma faixa u, então $\Delta u = 1/n$. As seguintes equações podem ser utilizadas para inicializar o método:

$$\begin{aligned} \mathbf{P}(0) &= \mathbf{d} \\ \Delta_1\mathbf{P}_0 &= \mathbf{a}(\Delta u)^3 + \mathbf{b}(\Delta u)^2 + \mathbf{c}(\Delta u) \\ \Delta_2\mathbf{P}_0 &= 6\mathbf{a}(\Delta u)^3 + 2\mathbf{b}(\Delta u)^2 \\ \Delta_3\mathbf{P}_0 &= 6\mathbf{a}(\Delta u)^3 \end{aligned} \tag{3.28}$$

Qualquer ponto $\mathbf{P}_{i+1}$ pode ser gerado a partir de $\mathbf{P}_i$ utilizando-se as equações abaixo:

$$\begin{aligned} \mathbf{P}_{i+1} &= \mathbf{P}_i + \Delta_1\mathbf{P}_i, \quad 0 \le i \le n \\ \Delta_1\mathbf{P}_{i+1} &= \Delta_1\mathbf{P}_i + \Delta_2\mathbf{P}_i \\ \Delta_2\mathbf{P}_{i+1} &= \Delta_2\mathbf{P}_i + \Delta_3\mathbf{P}_i \end{aligned} \tag{3.29}$$

Estas equações mostram que são necessárias somente três adições para gerar qualquer ponto e $3n$ adições para gerar os n pontos. Em geral, para um polinômio de grau n, n adições são requeridas para calcular cada ponto (NEWMAN, 1979).

3.3. Representação Paramétrica de Superfícies Sintéticas

Esta seção descreve duas superfícies paramétricas: superfícies de Hermite e superfícies de Bézier.

3.3.1. Superfície de Hermite Bicúbica

A superfície de Hermite é definida por quatro pontos, por oito vetores tangentes (quatro na direção u e quatro na direção v) e pelas quatro derivadas cruzadas de segunda ordem. Assim, cada ponto possui associado dois vetores tangentes e uma derivada cruzada de segunda ordem. A equação da Superfície de Hermite é dada por:

$$\mathbf{P}(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{C}_{ij}u^i v^j, \qquad 0 \le u \le 1,\ 0 \le v \le 1 \tag{3.30}$$

A forma matricial da Equação (3.30) é

$$\mathbf{P}(u,v) = \mathbf{U}^T[C]\mathbf{V}, \qquad 0 \le u \le 1,\ 0 \le v \le 1 \tag{3.31}$$

onde $\mathbf{U} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}^T$, $\mathbf{V} = \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}^T$, e a matriz de coeficientes é dada por

$$[C] = \begin{bmatrix} C_{33} & C_{32} & C_{31} & C_{30} \\ C_{23} & C_{22} & C_{21} & C_{20} \\ C_{13} & C_{12} & C_{11} & C_{10} \\ C_{03} & C_{02} & C_{01} & C_{00} \end{bmatrix} \tag{3.32}$$

De modo a determinar os coeficientes C_{ij}, aplicam-se as condições de contorno à Equação (3.31). Resolvendo os coeficientes e rearranjando os termos tem-se a seguinte equação final de um *patch* bicúbico:

$$\mathbf{P}(u,v) = \mathbf{U}^T\left[M_H\right][B]\left[M_H\right]^T\mathbf{V}, \qquad 0 \le u \le 1,\ 0 \le v \le 1 \tag{3.33}$$

onde $\left[M_H\right]$ é a matriz de Hermite (vide Equação (3.15)), e $[B]$, a matriz geométrica é dada por:

$$[B]=\left[\begin{array}{cc:cc} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{v00} & \mathbf{P}_{v01} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{v10} & \mathbf{P}_{v11} \\ \hdashline \mathbf{P}_{u00} & \mathbf{P}_{u01} & \mathbf{P}_{uv00} & \mathbf{P}_{uv01} \\ \mathbf{P}_{u10} & \mathbf{P}_{u11} & \mathbf{P}_{uv10} & \mathbf{P}_{uv11} \end{array}\right] \tag{3.34}$$

A matriz [B] é particionada para indicar o agrupamento das condições de contorno similares. Ela pode ser também descrita como:

$$[B]=\left[\begin{array}{c:c} [P] & [P_v] \\ \hdashline [P_u] & [P_{uv}] \end{array}\right] \tag{3.35}$$

onde $[P]$, $\left[P_u\right]$, $\left[P_v\right]$, e $\left[P_{uv}\right]$ são as submatrizes dos pontos, dos vetores tangentes em u, dos vetores tangentes em v, e das derivadas cruzadas de segunda ordem.

3.3.2. Superfície de Bézier

A superfície de Bézier é uma extensão da curva de Bézier em duas direções u e v. Um conjunto ordenado de pontos de controle é utilizado para construir uma superfície como ilustrada na Figura 3-10. A equação da superfície pode ser escrita como extensão da Equação (3.18); ou seja:

$$\mathbf{P}(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}\mathbf{P}_{ij}B_{i,n}(u)B_{j,m}(v), \qquad 0\le u\le 1,\ 0\le v\le 1 \tag{3.36}$$

onde $\mathbf{P}(u,v)$ é um ponto na superfície, $B_{i,n}(u)$ e $B_{j,m}(v)$ são dados pela Equação (3.19) e $\mathbf{P}_{ij}$ são os pontos de controle.

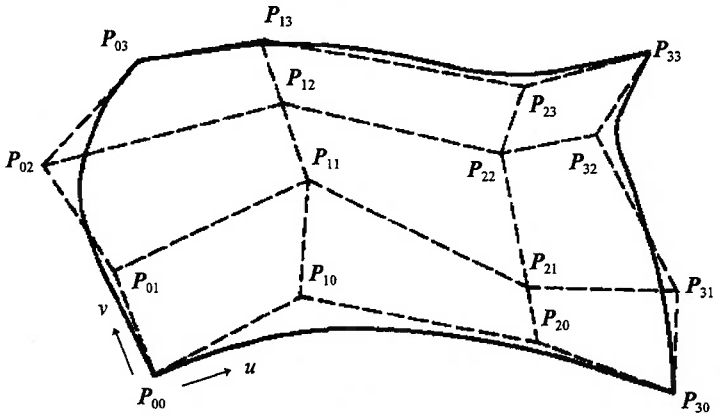


Figura 3-10 Uma superfície de Bézier (ZEID, 1991).

Os pontos estão dispostos em uma rede retangular de n+1 por m+1. Uma vez expandida, a equação (3.36) torna-se:

3.4. Algoritmos de Intersecção

Os cálculos de intersecção de curva/plano, superfície/plano e superfície/superfície são essenciais para realizar as operações globais citadas no capítulo 2. Há vários métodos para realizar estes cálculos e uma boa classificação pode ser encontrada em TORIYA (1991).

3.4.1. Intersecção Curva de Bézier / Plano

Ao seccionarmos um sólido que possui superfícies paramétricas definindo o seu contorno por um plano, é preciso determinar os pontos de intersecção entre as curvas de contorno destas superfícies e o plano de corte.

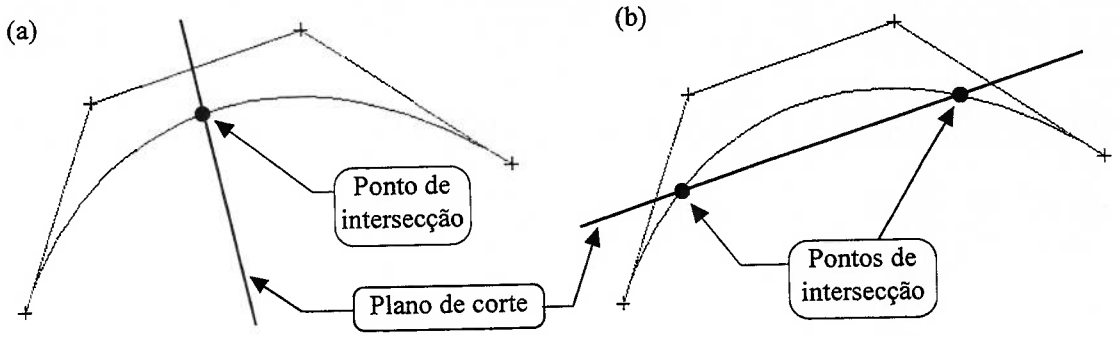


Figura 3-11 Intersecção curva / plano.

O plano divide o espaço em dois semi-espaços e sua equação é dada por:

$$ax + by + cz + d = 0 \qquad \sqrt{a^2 + b^2 + c^2} = 1 \tag{3.41}$$

A distância de um ponto do espaço (x_1, y_1, z_1) ao plano é dada por:

$$dis = ax_1 + by_1 + cz_1 + d \tag{3.42}$$

Esta distância possui um sinal que estará associado a um dos dois semi-espaços. Assim, se dois pontos da curva estiverem em semi-espaços distintos, é porque a curva intersecciona o plano. O algoritmo proposto inicia testando os pontos extremos da curva; caso eles estejam em semi-espaços distintos, então realizamos uma nova iteração, agora com o ponto médio entre os dois anteriores. Nesta situação possuímos dois trechos da curva e selecionamos o trecho que possui extremos em semi-espaços distintos. Este processo é repetido até atingir a precisão desejada.

Caso ambas as extremidades da curva estejam no mesmo semi-espaço, realizamos mais um teste com o ponto médio para verificar se alguma intersecção pode ser determinada (vide Figura 3-11).

Este algoritmo não soluciona todos os problemas. Este algoritmo foi escolhido por sua facilidade de implementação e por atender ao domínio do nosso projeto.

3.4.2. Intersecção superfície / plano (Algoritmo Marching)

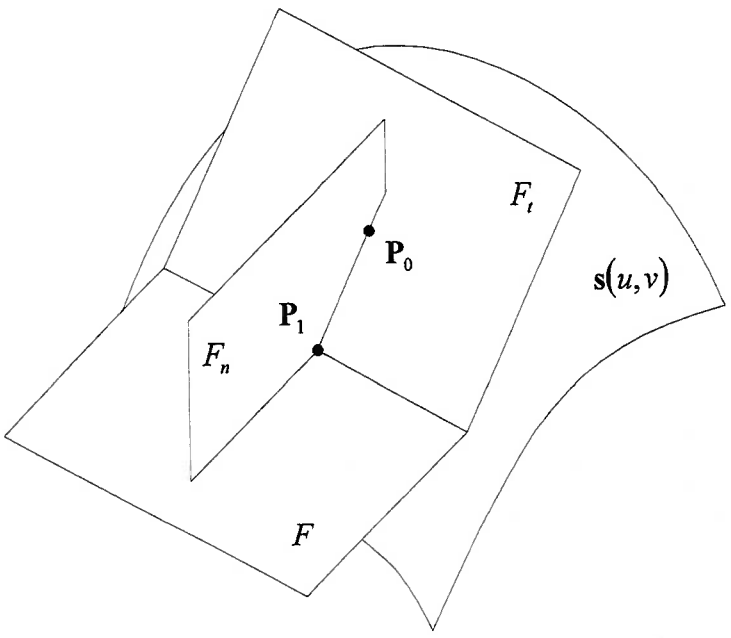
O algoritmo desenvolvido baseia no algoritmo de intersecção de Marching descrito por HOSAKA (1992). Ele consta de três partes: na primeira é necessário encontrar um ponto inicial sobre a superfície de Bézier; na segunda parte o algoritmo aproxima um ponto qualquer sobre a superfície para um ponto sobre a curva de intersecção; na terceira parte o algoritmo encontra um ponto próximo à curva de intersecção a partir de um ponto de intersecção conhecido. Os pontos são calculados recursivamente utilizando-se a segunda e terceira parte do algoritmo.

Primeira parte: Determinação de um ponto sobre a superfície de Bézier.

O algoritmo procura a intersecção de uma das bordas da superfície com o plano de intersecção utilizando o algoritmo de intersecção Curva de Bézier / plano. O primeiro ponto encontrado em alguma das bordas também está sobre a superfície.

Segunda parte: Aproximação de um ponto qualquer sobre a superfície para um ponto sobre a curva de intersecção.

Seja $\mathbf{r}$ o vetor posição no espaço de um ponto em um plano F, $\mathbf{n}_f$ o vetor normal unitário do plano F e d a distância do plano à origem do sistema; então pode-se descrever o plano por $\mathbf{r} \cdot \mathbf{n}_f = d$.



F : plano de intersecção

$\mathbf{s}(u,v)$: superfície paramétrica

$\mathbf{P}_0$: ponto inicial sobre $\mathbf{s}(u,v)$

F_t : plano tangente a $\mathbf{s}(u,v)$ em $\mathbf{P}_0$

F_n : plano auxiliar, perpendicular a F e F_t e passando por $\mathbf{P}_0$

$\mathbf{P}_1$: ponto aproximado pelo algoritmo

Figura 3-12 Algoritmo de intersecção Superfície-Plano (Primeira etapa).

Seja uma superfície dada por $\mathbf{s}(u,v)$, como os pontos do plano e da superfície são coincidentes na sua curva de intersecção, isto é $\mathbf{s}(u,v) = \mathbf{r}$, pode-se escrever:

$$\mathbf{s}(u,v) \cdot \mathbf{n}_f = d \tag{3.43}$$

Esta é a equação das curvas de intersecção. Fixando o valor de u ou v, pode-se calcular o correspondente u ou v de um ponto da intersecção resolvendo a Equação (3.43).

A Figura 3-12 ilustra o algoritmo. Seja F o plano de intersecção e $\mathbf{n}_f$ a sua normal. Dado um ponto inicial $\mathbf{P}_0 = \mathbf{s}(u_0, v_0)$ define-se o plano F_t tangente à superfície no ponto $\mathbf{P}_0$ com normal $\mathbf{n}_t$. Define-se em seguida o plano F_n, ortogonal ao plano de intersecção e que passa pelo ponto $\mathbf{P}_0$. Este plano terá uma normal dada por $\mathbf{n}_n$.

A intersecção destes três planos, F, F_t e F_n, resulta no ponto $\mathbf{P}_1$ que pode ser descrito por:

$$\mathbf{P}_1 = \frac{d_f \cdot (\mathbf{n}_t \times \mathbf{n}_n) + d_n \cdot (\mathbf{n}_f \times \mathbf{n}_t) + d_t \cdot (\mathbf{n}_n \times \mathbf{n}_f)}{\mathbf{n}_f \cdot (\mathbf{n}_t \times \mathbf{n}_n)} \tag{3.44}$$

Como este ponto está no plano tangente a $\mathbf{P}_0$ pode-se expressar o valor $\mathbf{P}_1 - \mathbf{P}_0$ utilizando-se os vetores básicos da superfície em $\mathbf{P}_0$. Assim temos:

$$\Delta\mathbf{P} = \mathbf{P}_1 - \mathbf{P}_0 = s_u \cdot du + s_v \cdot dv \tag{3.45}$$

onde $\mathbf{s}_u$ e $\mathbf{s}_v$ são as derivadas nas direções u e v (vide seção 3.3.2). Fazendo o produto vetorial com o vetor $\mathbf{s}_v \times \mathbf{n}_f$ em ambos os lados da equação, temos:

$$\begin{aligned} \Delta\mathbf{P} \cdot \mathbf{s}_v \times \mathbf{n}_f &= du \cdot \mathbf{s}_u \cdot \mathbf{s}_v \times \mathbf{n}_f + dv \cdot \mathbf{s}_v \cdot \mathbf{s}_v \times \mathbf{n}_f \\ \Rightarrow \Delta\mathbf{P} \cdot \mathbf{s}_v \times \mathbf{n}_f &= du \cdot \mathbf{s}_u \cdot \mathbf{s}_v \times \mathbf{n}_f \end{aligned} \tag{3.46}$$

$$du = \frac{\left[\Delta\mathbf{P}, \mathbf{s}_v, \mathbf{n}_f\right]}{\left[\mathbf{s}_u, \mathbf{s}_v, \mathbf{n}_f\right]} \tag{3.47}$$

Analogamente multiplicando ambos os lados da equação por $\mathbf{s}_u \times \mathbf{n}_f$, temos:

$$dv = -\frac{\left[\Delta\mathbf{P}, \mathbf{s}_u, \mathbf{n}_f\right]}{\left[\mathbf{s}_u, \mathbf{s}_v, \mathbf{n}_f\right]} \tag{3.48}$$

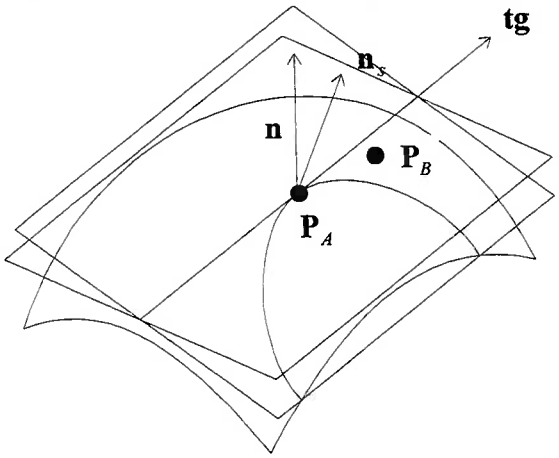
Seja $s(u_0 + du, v_0 + dv)$ um novo $\mathbf{P}_0$. Determina-se um novo ponto $\mathbf{P}_1$ e calcula-se recursivamente o procedimento descrito até que $\Delta\mathbf{P} = \mathbf{P}_1 - \mathbf{P}_0$ torne-se suficientemente pequeno.

$$s(u_0, v_0) = \mathbf{P}_0 \Rightarrow s(u_0 + du, v_0 + dv) = \mathbf{P}_1 \tag{3.49}$$

Normalmente a convergência é muito rápida e resulta em um ponto da curva de intersecção. O procedimento não converge quando o plano tangente à superfície no ponto $\mathbf{P}_0$ torna-se paralelo ao plano de intersecção.

Terceira parte: Determinação de um ponto próximo à curva de intersecção a partir de um ponto sobre a curva de intersecção.

Após determinar um ponto da curva de intersecção $\mathbf{P}_A$ com o procedimento anterior é possível traçar a curva de intersecção a partir dele. Determina-se um ponto $\mathbf{P}_B$ aproximado por $\mathbf{P}_B = s(u_0 + \Delta u, v_0 + \Delta v)$. Os valores de Δu e Δv podem ser calculados a partir da projeção dos vetores $\mathbf{s}_u$ e $\mathbf{s}_v$ sobre o vetor tangente $\mathbf{tg} = \mathbf{n} \times \mathbf{n}_s$.



$\mathbf{n}$: normal do plano de intersecção

$\mathbf{n}_s$: normal da superfície em $\mathbf{P}_A$

$\mathbf{tg} = \mathbf{n} \times \mathbf{n}_s$

$\mathbf{P}_A$: ponto sobre a curva de intersecção entre o plano e superfície

$\mathbf{P}_B$: aproximação do próximo ponto sobre a curva

Figura 3-13 Algoritmo de intersecção Superfície-Plano (Segunda etapa).

O ponto $\mathbf{P}_B$ é um bom ponto inicial $\mathbf{P}_0$ para o procedimento de aproximação descrito anteriormente. Uma vez corrigido o ponto $\mathbf{P}_B$ calcula-se um novo ponto próximo à curva de intersecção utilizando as projeções sobre o vetor tangente. Utiliza-se este procedimento até que a curva atinja uma das bordas.

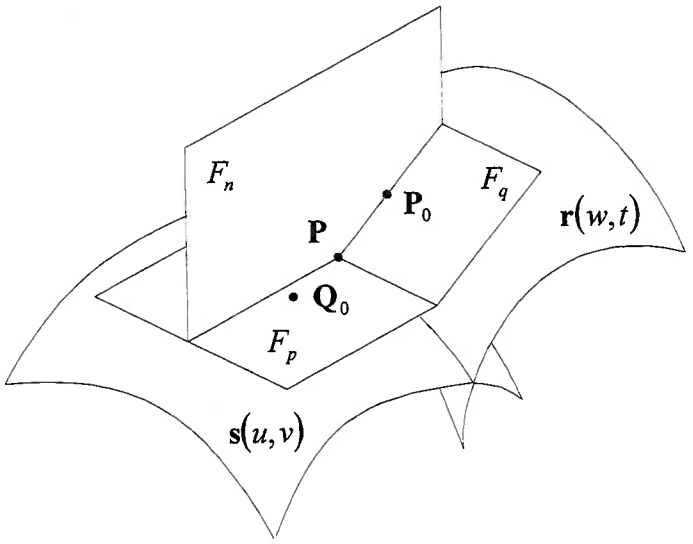
3.4.3. Intersecção superfície / superfície (Algoritmo Marching)

O algoritmo utilizado para calcular a curva de intersecção entre uma ou mais superfícies paramétricas é muito similar ao algoritmo de Intersecção superfície / plano.

Primeira parte: Determinação de um ponto sobre cada uma das superfícies de Bézier.

Quando se realiza uma operação booleana entre dois sólidos com superfícies paramétricas gera-se primeiro um sólido poliédrico. Os vértices deste sólido estão bastante próximos dos vértices do sólido resultante final. Há vértices marcados próximos à curva de intersecção. Escolhendo-se um destes vértices é possível então determinar um ponto de cada superfície que esteja próximo a este vértice.

Segunda parte: Aproximação de um ponto qualquer sobre a superfície para um ponto sobre a curva de intersecção.



$\mathbf{r}(w,t)$: superfície paramétrica $\mathbf{r}$

$\mathbf{s}(u,v)$: superfície paramétrica $\mathbf{s}$

$\mathbf{P}_0$: aproximação inicial do ponto de intersecção sobre $\mathbf{r}(w,t)$

$\mathbf{Q}_0$: aproximação inicial do ponto de intersecção sobre $\mathbf{s}(u,v)$

F_q : plano tangente a $\mathbf{r}(w,t)$ em $\mathbf{P}_0$

F_p : plano tangente a $\mathbf{s}(u,v)$ em $\mathbf{Q}_0$

F_n : plano auxiliar, perpendicular a F_q e F_p e passando por $\mathbf{P}_0$

Figura 3-14 Algoritmo de Intersecção superfície / superfície.

Consideremos duas superfícies paramétricas $\mathbf{r}(w,t)$ e $\mathbf{s}(u,v)$ ilustradas na Figura 3-14. Os pontos $\mathbf{P}_0$ e $\mathbf{Q}_0$ estão respectivamente sobre as superfícies $\mathbf{r}(w,t)$ e $\mathbf{s}(u,v)$. Assim temos:

$$\mathbf{P}_0 = \mathbf{r}(w_0,t_0) \text{ e } \mathbf{Q}_0 = \mathbf{s}(u_0,v_0) \tag{3.50}$$

Vamos denominar o plano tangente à superfície $\mathbf{r}(w,t)$ no ponto $\mathbf{P}_0$ por F_p e seu vetor normal por $\mathbf{n}_p$. Da mesma forma vamos denominar por F_q e $\mathbf{n}_q$ o plano tangente à superfície

$\mathbf{s}(u,v)$ no ponto $\mathbf{Q}_0$ e seu vetor normal. Logo:

$$\mathbf{n}_p = \frac{\mathbf{r}_w \times \mathbf{r}_t}{|\mathbf{r}_w \times \mathbf{r}_t|} \text{ e } \mathbf{n}_q = \frac{\mathbf{s}_u \times \mathbf{s}_v}{|\mathbf{s}_u \times \mathbf{s}_v|} \tag{3.51}$$

onde: $\mathbf{s}_u$ é a derivada de primeira ordem da superfície s na direção u

$\mathbf{s}_v$ é a derivada de primeira ordem da superfície s na direção v

$\mathbf{r}_w$ é a derivada de primeira ordem da superfície r na direção u

$\mathbf{r}_t$ é a derivada de primeira ordem da superfície r na direção v

Denominando por d_p e d_q as distâncias a partir da origem dos planos F_p e F_q, temos que:

$$d_p = \mathbf{n}_p \cdot \mathbf{r}(w_0, t_0) \text{ e } d_q = \mathbf{n}_q \cdot \mathbf{s}(u_0, v_0) \tag{3.51}$$

Vamos definir um plano F_n que passa por $\mathbf{P}_0$ e é ortogonal aos planos F_p e F_q. Seu vetor normal será denominado por $\mathbf{n}_n$ e sua distância à origem será d_n. Então:

$$\mathbf{n}_n = \frac{\mathbf{n}_p \times \mathbf{n}_q}{|\mathbf{n}_p \times \mathbf{n}_q|} \text{ e } d_n = \mathbf{n}_n \cdot \mathbf{r}(w_0, t_0)$$

A interseção destes três planos determina um novo ponto $\mathbf{P}$. $\Delta\mathbf{P}$ é definido como $\mathbf{P} - \mathbf{P}_0$ e $\Delta\mathbf{Q}$ como $\mathbf{P} - \mathbf{Q}_0$. Pode-se calcular $\mathbf{P}$ da seguinte forma:

$$\mathbf{P} = \frac{d_p \cdot (\mathbf{n}_q \times \mathbf{n}_n) + d_q \cdot (\mathbf{n}_n \times \mathbf{n}_p) + d_n \cdot (\mathbf{n}_p \times \mathbf{n}_q)}{\mathbf{n}_n \cdot (\mathbf{n}_p \times \mathbf{n}_q)} \tag{3.52}$$

Vamos assumir que as seguintes relações são verdadeiras, para valores de δ_w, δ_t, δ_u e δ_v suficientemente pequenos:

$$\mathbf{r}_w \delta_w + \mathbf{r}_t \delta_t = \Delta\mathbf{P} \tag{3.53}$$

$$\mathbf{s}_u \delta_u + \mathbf{s}_v \delta_v = \Delta\mathbf{Q} \tag{3.54}$$

Definem-se os seguintes valores:

$$\mathbf{r}'_w = \mathbf{r}_w \times \mathbf{n}_p \qquad \mathbf{r}'_t = \mathbf{r}_t \times \mathbf{n}_p \tag{3.55}$$

$$\mathbf{s}'_u = \mathbf{s}_u \times \mathbf{n}_q \qquad \mathbf{s}'_v = \mathbf{s}_v \times \mathbf{n}_q \tag{3.56}$$

Combinando as equações (3.53) e (3.54) respectivamente com as equações (3.55) e (3.56) pode-se deduzir que:

$$\delta_w = \frac{(\mathbf{r}'_t \cdot \Delta\mathbf{P})}{(\mathbf{r}'_t \cdot \mathbf{r}_w)}, \delta_t = \frac{(\mathbf{r}'_w \cdot \Delta\mathbf{P})}{(\mathbf{r}'_w \cdot \mathbf{r}_t)}, \delta_u = \frac{(\mathbf{s}'_v \cdot \Delta\mathbf{Q})}{(\mathbf{s}'_v \cdot \mathbf{s}_u)}, \delta_v = \frac{(\mathbf{s}'_u \cdot \Delta\mathbf{Q})}{(\mathbf{s}'_u \cdot \mathbf{s}_v)} \tag{3.57}$$

Assim os novos pontos serão:

$$\mathbf{P}_0 = \mathbf{r}(w_0 + \delta_w, t_0 + \delta_t) \tag{3.58}$$

$$\mathbf{Q}_0 = \mathbf{s}(u_0 + \delta_u, v_0 + \delta_v) \tag{3.59}$$

Definindo $\Delta PQ = \max\{|\Delta\mathbf{P}|, |\Delta\mathbf{Q}|\}$ repete-se o procedimento acima até que ΔPQ seja menor que Δ.

Terceira parte: Determinação de um ponto próximo à curva de intersecção a partir de um ponto sobre a curva de intersecção.

A determinação do próximo ponto da curva é feita de maneira parecida com a aproximação. São calculados os pontos $\mathbf{P}_0 = \mathbf{r}(w_0, t_0)$ e $\mathbf{Q}_0 = \mathbf{s}(u_0, v_0)$ depois de feita a aproximação garantindo que esses dois pontos estejam próximos. Em seguida, calcula-se os planos F_p, tangente à superfície $\mathbf{r}(w,t)$ em $\mathbf{P}_0$ e F_q, tangente à superfície $\mathbf{s}(u,v)$ em $\mathbf{Q}_0$. O vetor normal ao plano F_n, que passa por $\mathbf{P}$ e é perpendicular aos planos F_p e F_q indica a direção a ser seguida. O sentido é determinado pelo sentido seguido anteriormente e devidamente armazenado para evitar que o traçado da curva tenha seu sentido invertido caso haja inversão do sinal do produto vetorial.

O traçado continua até que a curva atinja uma das bordas de uma das superfícies.

4. Um modelador de sólidos com modelagem geométrica

Em nosso cotidiano, é comum atribuir aos cantos e arestas dos objetos (móveis, paredes, etc) o valor angular de 90 graus. O ângulo reto parece estar presente por todos os lados, entretanto, se olharmos de perto estes traços geométricos, notaremos que as arestas de uma mesa ou as quinas das paredes não são perfeitamente retas, mas possuem uma suavização.

Por esta razão, uma representação da realidade que estivesse restrita a linhas retas, abolindo ou negligenciando a entidade curva, estaria limitada e ofereceria um modelo distorcido do mundo. Isto é o que ocorre com alguns modeladores de sólidos que utilizam aproximações lineares para representar curvas, assim como superfícies poliédricas para representar formas não planares.

A necessidade da representação explícita de elementos curvos decorre, dentre outros fatores, da crescente importância que tem ganho não somente a estética, mas também a necessidade de uma representação mais precisa para acompanhar os avanços no desenvolvimento dos sistemas CAM (Usinagem por máquinas de comando numérico) e a estereolitografia (confecção de moldes tridimensionais através de *laser*). Assim, o usuário final fica satisfeito com a beleza do produto, e o projetista se beneficia de uma representação geometricamente fiel.

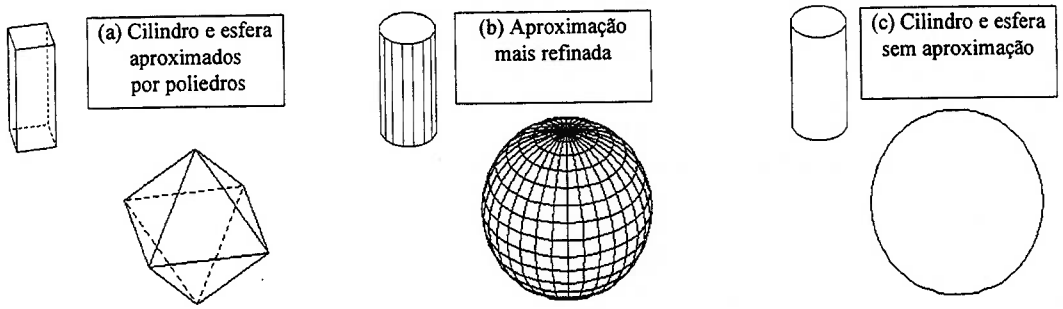


Figura 4-1 Aproximações poliedrais de sólidos.

Um modelador de sólidos *B-Rep* sem suporte para a modelagem geométrica é capaz de gerar poliedros que aproximam cilindros, esferas e toróides (vide Figura 4-1). Entretanto, os dados sobre os raios dos objetos, utilizados para gerar os segmentos que aproximam os arcos de circunferência, não são armazenados pela estrutura de dados. Uma vez gerado o sólido, as informações sobre curvas e superfícies são perdidas. Este tipo de modelador é denominado por modelador de sólidos poliedral, pois impossibilita a manipulação exata das curvas e superfícies dos objetos, que são reduzidas sempre a segmentos ou faces planas de um poliedro (vide Figura 4-1 (a)).

Quando um sólido é gerado com uma aproximação grosseira (Figura 4-1 (a)) para conseguir uma aproximação mais refinada (Figura 4-1 (b)) será necessário fornecer novamente os raios para gerar novos sólidos.

Para solucionar este problema é necessário expandir a estrutura de dados utilizada em um modelador poliedral, incluindo informações sobre curvas e superfícies e modificando todos os algoritmos que manipulem estas informações. Isto possibilita a representação exata de cilindros, esferas e outros objetos que possuem curvas ou superfícies definindo o seu contorno (vide Figura 4-1 (c)). Chamaremos por modelo sólido geométrico o modelo sólido representado segundo estas características.

4.1. A nova estrutura de dados

Neste trabalho estaremos estudando a possibilidade de representar a geometria da curva como atributo das meia-arestas, e a geometria da superfície como atributo das faces. Desta maneira, estaremos definindo uma clara separação entre informações geométricas e informações topológicas (vide a Figura 4-2).

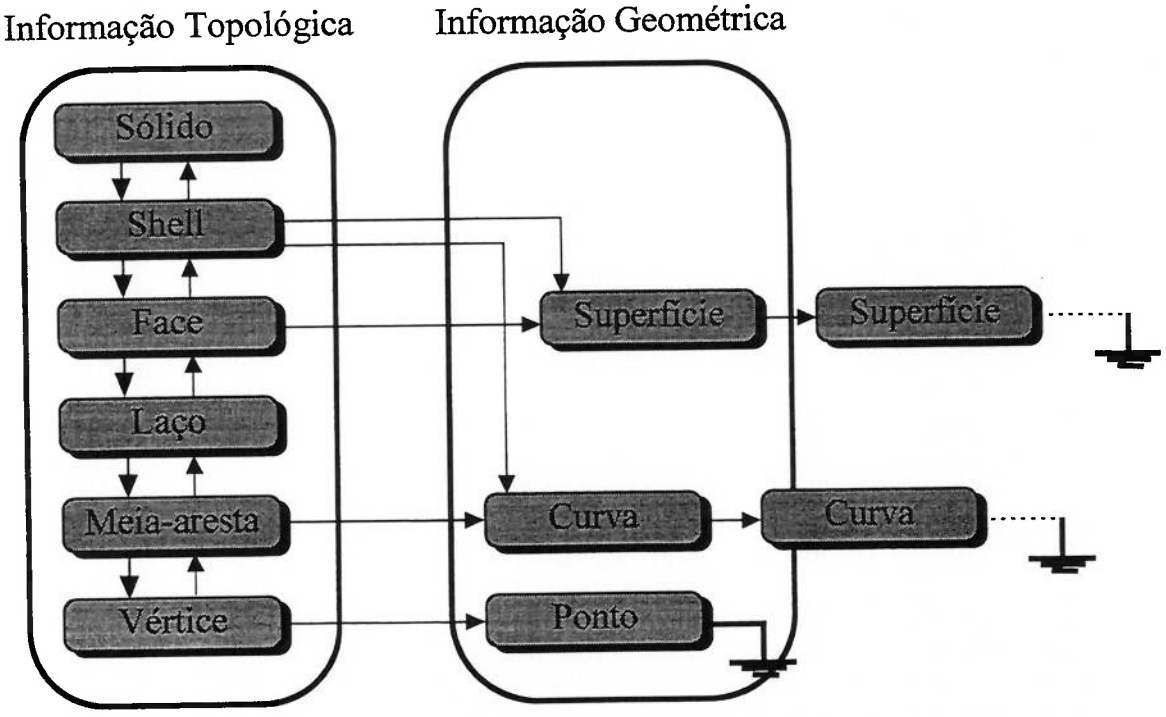


Figura 4-2 Estrutura de dados do modelador de sólidos geométrico.

Devido a esta clara separação, é possível representar superfícies que não possuem um contorno coincidente com o contorno da face associada, conforme ilustra a Figura 4-3. Neste

exemplo, um cilindro pode ser representado por uma superfície cilíndrica infinita que estará delimitada pelas arestas das faces superior e inferior, ou seja, a informação topológica impõe limites à representação de geometrias infinitas.

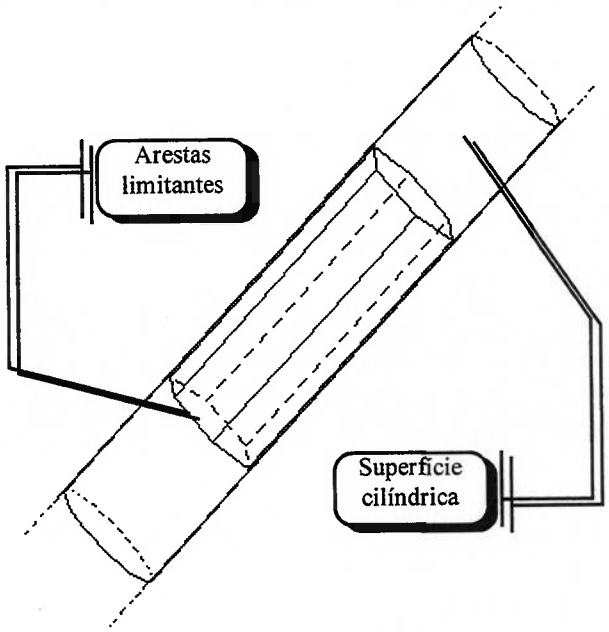


Figura 4-3 A geometria infinita e a topologia limitante.

Esta característica é muito importante porque, quando uma operação booleana é realizada, freqüentemente algumas faces do sólido resultante correspondem a apenas uma parte do original. Caso calculássemos uma nova representação para a superfície com o novo contorno definido pela operação booleana, seriam geradas imprecisões acumulativas. Desta maneira, optamos pela solução de manter a geometria original da superfície e definir o contorno da face pelas informações topológicas.

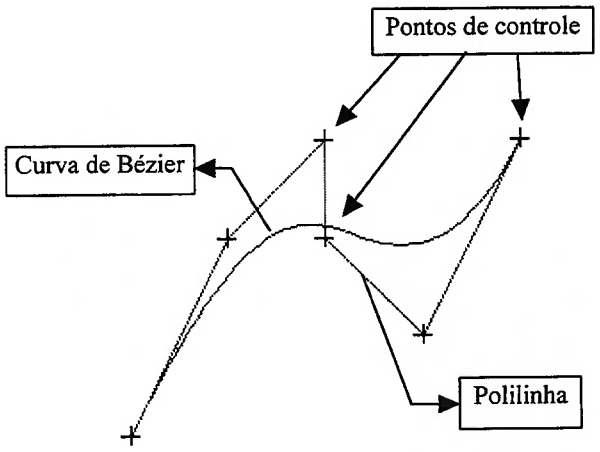


Figura 4-4 Polilinha utilizada para representar o polígono característico de uma curva de Bézier.

Segundo o nosso entender, a estrutura de dados do novo modelador apresenta poucas diferenças em relação à estrutura de dados de um modelador poliedral. As informações sobre curvas foram associadas a cada *half-edge* enquanto as informações sobre superfícies foram associadas à estrutura face.

Neste trabalho, representaremos dois tipos de curvas:

•Retas: ela é definida pelo pontos extremos da aresta associada.

•Polilinhas: são representadas por uma lista de pontos no espaço. As polilinhas definem o polígono característico das curvas de Bézier (vide a Figura 4-4).

As informações geométricas das curvas e superfícies de um sólido definem uma lista que está ligada diretamente ao elemento *shell* (vide Figura 4-5) de um sólido.

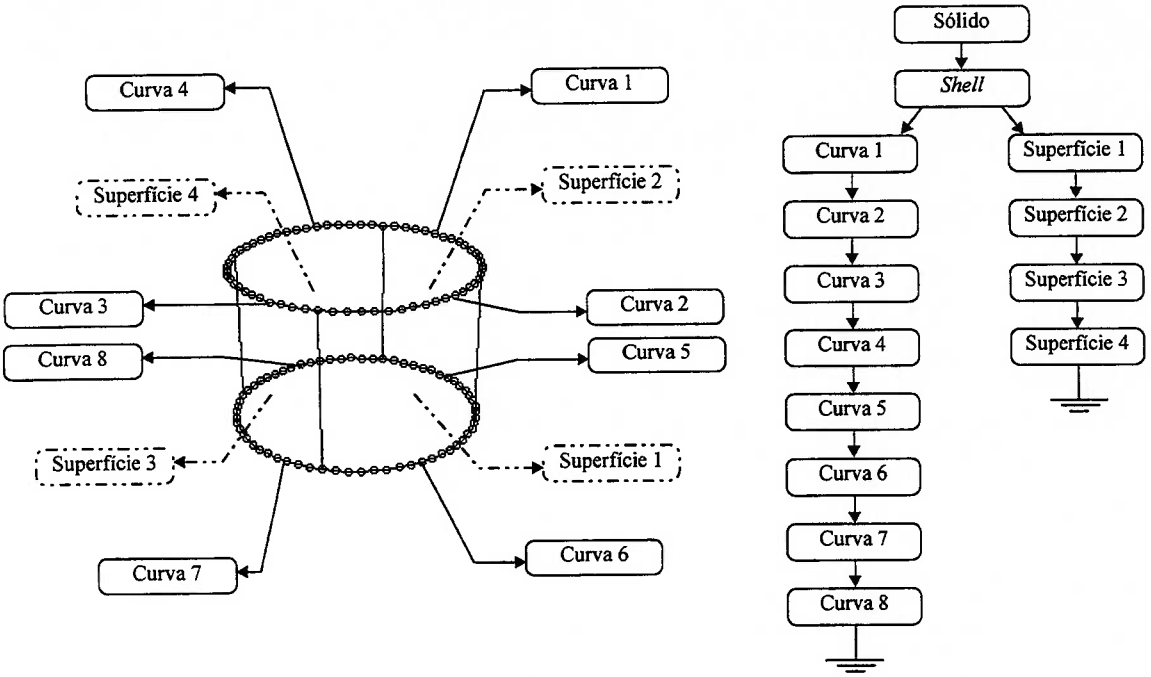


Figura 4-5 Exemplo de armazenamento de informações geométricas.

Para representar uma curva de Bézier de grau 3 basta armazenar 4 pontos de controle, onde cada ponto é uma coordenada cartesiana no espaço tridimensional. A Listagem 4-1 mostra a implementação da estrutura para armazenar um ponto de controle em uma lista. **pnt** é um vetor que contém as três coordenadas e **nxt** é um ponteiro que indica o próximo ponto de controle.

```
struct {
   vector      pnt ;          // coordenada do ponto de polilinha
   lpoint      *nxt ;         // ponteiro para o próximo ponto
} lpoint ;
```

Listagem 4-1 Estrutura que armazena um ponto da polilinha.

Além dos pontos de controle, necessitamos de uma estrutura para armazenar a informação sobre a curva propriamente dita; o tipo de curva (reta ou Bézier), o seu identificador e o ponteiro para a polilinha. A Figura 4-6 e a Listagem 4-2 descrevem a estrutura completa para armazenar uma curva.

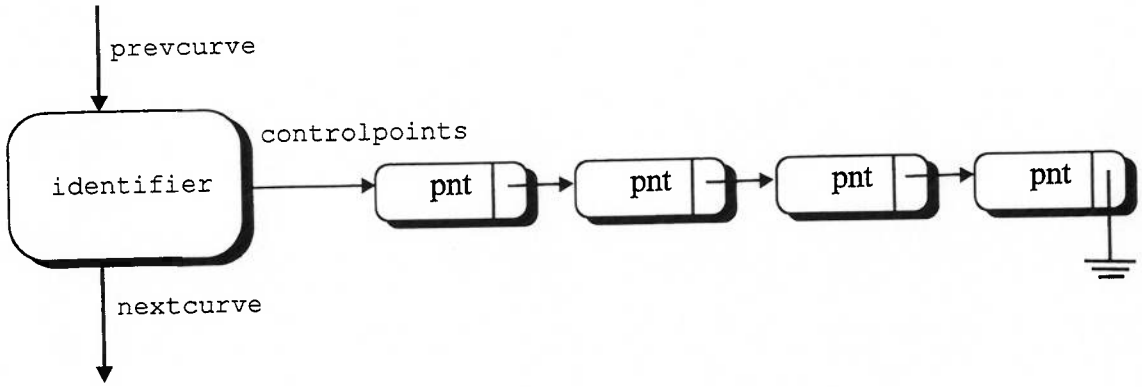


Figura 4-6 Representação das informações geométricas.

```
struct {
   Id    identifier ;       // Identificador
   lpnt  *controlpoints ;   // ponteiro para o início de polilinha
   curve *prevcurve ;       // ponteiro para a curva anterior
   curve *nextcurve ;       // ponteiro para a curva seguinte
} curve ;
```

Listagem 4-2 Implementação da estrutura para armazenar curvas.

Para representar uma superfície de Bézier de ordem 3 é necessário armazenar 16 pontos de controle em uma matriz. A estrutura completa para armazenar uma superfície de Bézier está ilustrada na Figura 4-7 e sua implementação corresponde à Listagem 4-3.

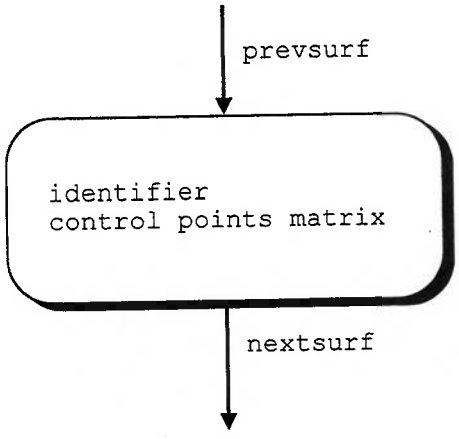


Figura 4-7 Representação da estrutura de dados para uma superfície.

```
struct {
  Id          identifier;       // Identificador da superfície
  vector      matrix [4][4] ;   // Matriz dos pontos de controle
  surface     *prevsurf ;       // ponteiro p/ superfície anterior
  surface     *nextsurf ;       // ponteiro p/ superfície seguinte
} surface ;
```

Listagem 4-3 Implementação da estrutura que armazena superfícies.

Uma vez modificada a estrutura de dados foi necessário implementar mais oito rotinas ao nível dos operadores de Euler para manipular as novas informações.

- **MCI** (*Make Curve Information*): este operador cria uma nova curva e a associa ao sólido fornecido.
- **MSI** (*Make Surface Information*): este operador é semelhante ao anterior, mas relativo às superfícies.
- **KCI** (*Kill Curve Information*): este operador remove uma curva de um sólido. Ao remover a curva devemos também desassociar a informação geométrica de todas as meia-arestas correspondentes.
- **KSI** (*Kill Surface Information*): semelhante ao anterior, mas aplicado a superfícies.
- **ACAE** (*Associate Curve to Approximation Edge*): associa a representação geométrica de uma curva a uma meia-aresta da topologia do sólido.
- **ASAF** (*Associate Surface to Approximation Face*): associa uma superfície a uma face.
- **ADDPP** (*Add Point to Polyline*): acrescenta um ponto a uma polilinha;
- **REMPP** (*Remove Point from Polyline*): remove um ponto da polilinha.

4.2. Sincronismo entre as informações topológicas e geométricas

Segundo a representação proposta, todas as características de um modelador poliedral foram mantidas. Por exemplo, através de uma varredura das arestas de um sólido é possível exibi-lo segundo uma visualização do tipo *wire-frame*. Através de uma varredura das faces de um sólido, é possível obter informações sobre a massa do sólido (momento de inércia, volume, etc...). Também, através de uma varredura das faces de um sólido, será possível gerar as informações de entrada para programas de elementos finitos.

Apesar das vantagens de fidelidade geométrica existentes em um modelo sólido geométrico, algumas aplicações requerem um modelo poliedral como entrada (visualização, cálculo do momento de inércia, etc). Nestas aplicações, caso a aproximação poliedral realizada seja grosseira, é conveniente possuirmos um mecanismo para melhorarmos a aproximação. Neste sentido é importante existir um sincronismo entre as informações topológicas e geométricas para definir em tempo real qual o nível de detalhamento ou de precisão desejado. Por exemplo, quando o modelo for armazenado em arquivo, é conveniente compactar sua estrutura e armazená-lo com o menor nível de detalhamento possível. Ou ainda, quando o sólido for exibido será conveniente linearizarmos as curvas para facilitar a sua visualização. Para uma melhor compreensão observe a Figura 4-8. Ela ilustra uma circunferência que está representada por quatro curvas de Bézier (Figura 4-8 (a)). A representação mais simples desta curva, desconsiderando a informação geométrica, é uma reta entre os pontos inicial e final (Figura 4-8 (b)). Um grau de detalhamento maior está ilustrado na Figura 4-8 (c). Note-se que neste caso, o número de aresta e vértices dobrou.

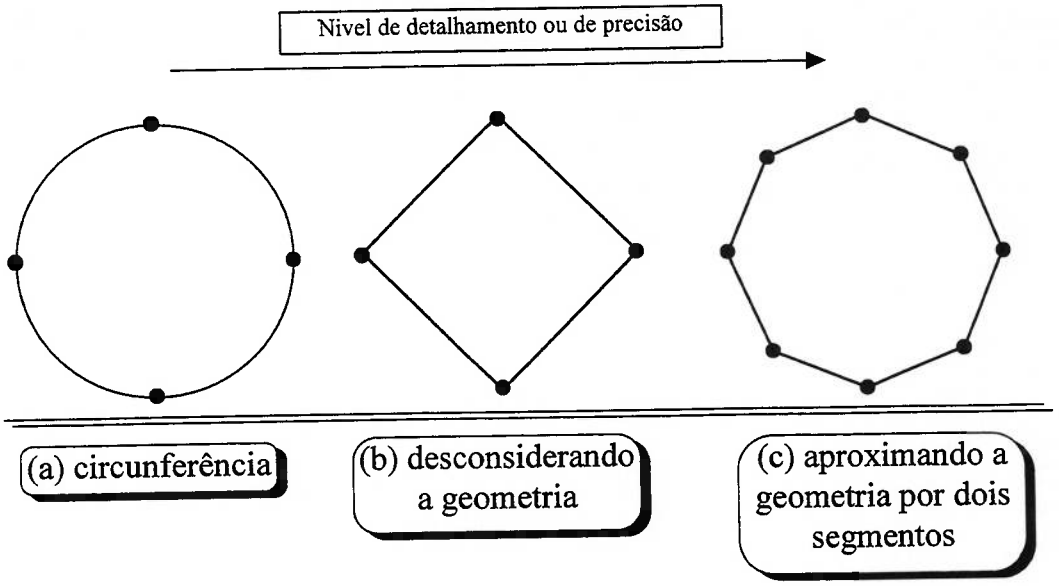


Figura 4-8 Aproximações poligonais de uma circunferência.

O sincronismo entre as informações geométrica e topológica é, em nossa proposta, realizada por uma associação entre as estruturas de dados das informações topológicas e das informações geométricas. Conforme ilustra a Figura 4-9, o arco de circunferência está sendo representado de forma exata e a aproximação linear possui um conjunto de arestas que definem o modelo poliedral. Segundo a representação que propomos toda aresta de aproximação está associada à representação exata.

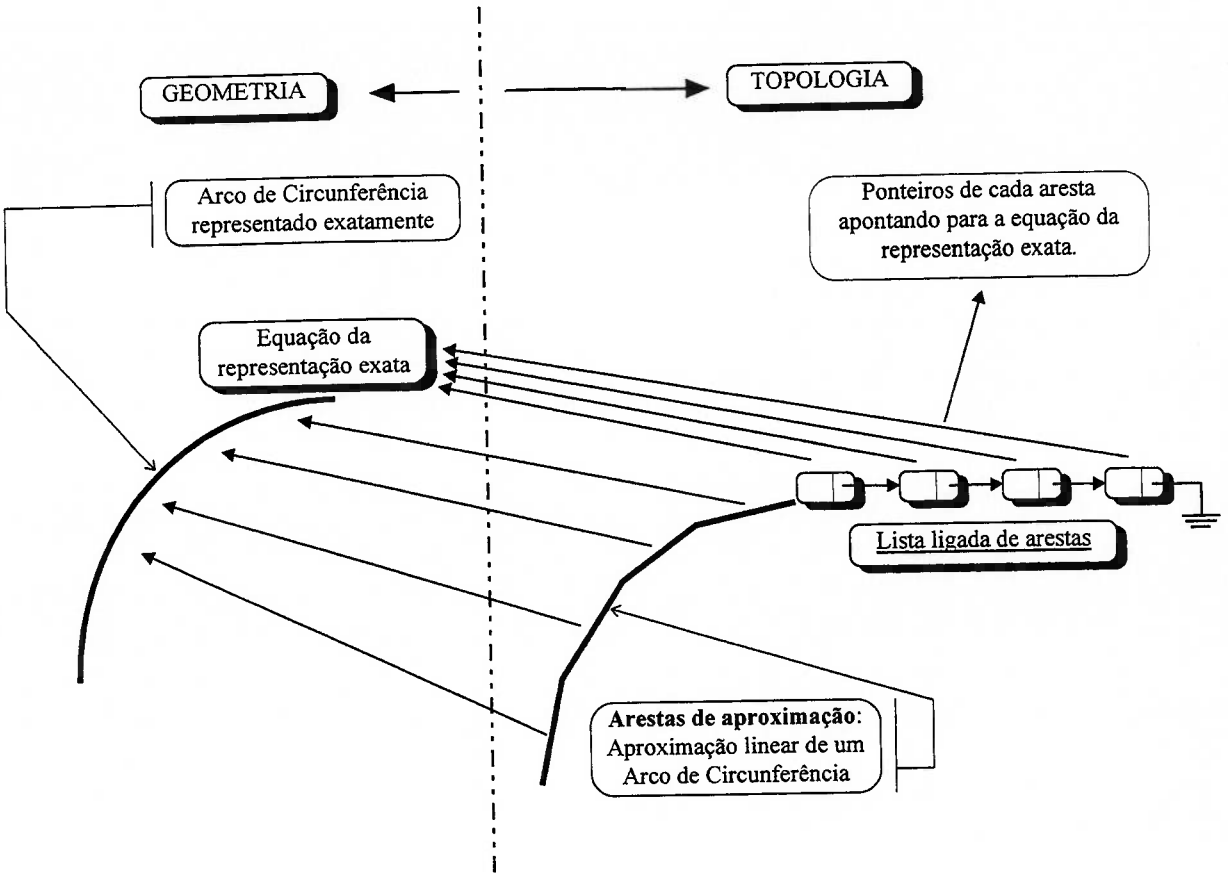


Figura 4-9 Sincronismo das informações topológica e geométrica.

Para realizar a compactação do modelo será necessário um mecanismo que remova todas as arestas de aproximação associadas à representação exata, deixando apenas uma. Se posteriormente desejarmos uma aproximação mais precisa para o arco de circunferência, basta gerar novas arestas de aproximação na proporção necessária e, novamente, associá-las à representação exata. A Figura 4-10 ilustra este mecanismo.

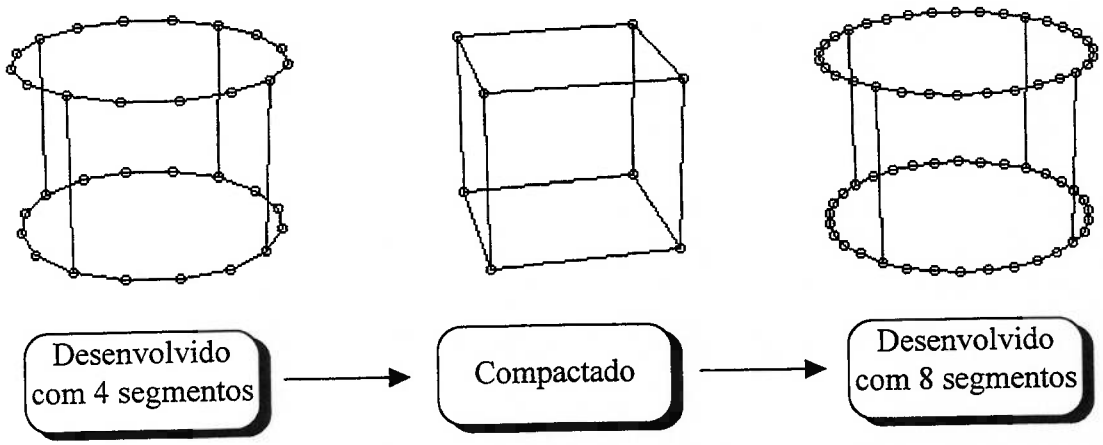


Figura 4-10 Mecanismo para aumentar o número de linhas que aproximam informações curvas.

Os passos para implementação da rotina de desenvolvimento de curvas podem ser resumidos na Listagem 4-4.

```
<Cálculo do número de segmentos de reta para aproximar a curva>;
Para <número de segmentos> faça
        <cálculo do ponto de aproximação Pi>;
        <aplicação do operador MEV para Pi e Pi+1>;
        <associação da informação geométrica à nova aresta>;
Fim
```

Listagem 4-4 Rotina de desenvolvimento geométrico das curvas de um sólido.

Para compactar uma curva, basta fornecer um dos segmentos de aproximação desta curva e o algoritmo desenvolvido utilizará operadores **KEV**'s para deletar as aproximações da curva, deixando ao final apenas um único segmento (vide Listagem 4-5).

```
Até que <haja somente uma aresta de aproximação> faça
        <dessassociação da informação geométrica de uma aresta
         de aproximação>;
        <aplicação do operador KEV sobre esta mesma aresta de
         aproximação>;
Fim
```

Listagem 4-5 Rotina de compactação geométrica de um sólido.

4.3. Adaptação dos Operadores Locais

Nesta seção detalharemos as alterações que foram feitas ao USP Designer, Modelador de Sólidos desenvolvido pelo Departamento de Engenharia Mecânica da EPUSP, de modo a suportar a nossa proposta.

- **Gerador de arco**: representado na Figura 4-11, ele foi modificado para calcular os pontos de controle de uma curva de Bézier que aproxima o arco, ao invés de utilizar as informações sobre raio, ângulos inicial e final e coordenadas do centro para gerar diretamente os segmentos de reta, esta infomação é utilizada. O objetivo é padronizar as informações curvas para facilitar sua manipulação posterior. Uma vez que os pontos de controle estão calculados basta utilizar os operadores de desenvolvimento e compactação para aumentar ou diminuir a precisão da representação do arco. O valor de K, calculado em YAMAGUCHI (1989), determina a melhor aproximação de uma curva de Bézier para uma circunferência.

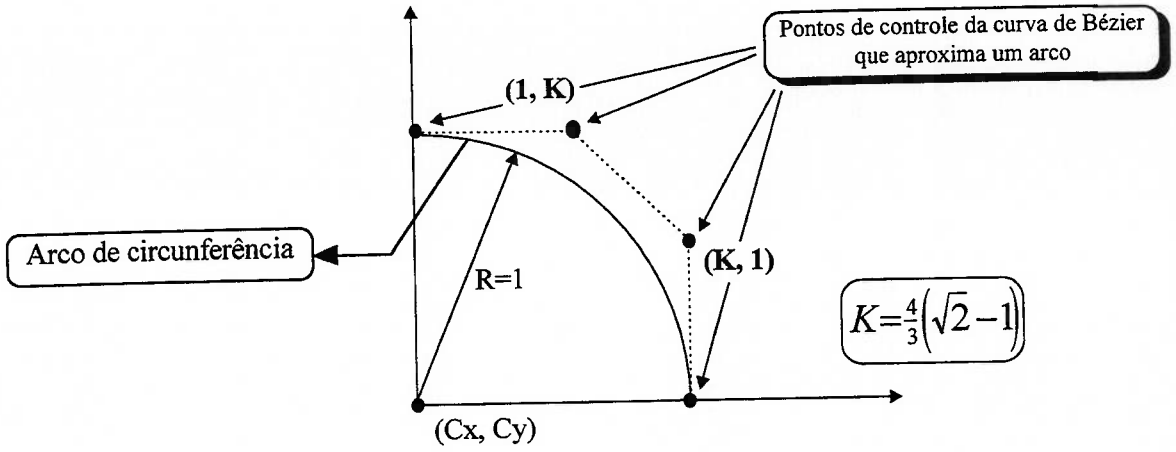


Figura 4-11 Construção de um arco de circunferência aproximado por uma curva de Bézier.

- **Extrusão translacional:** A extrusão deve ser adaptada para copiar as informações geométricas sobre as curvas para as novas arestas da sua face posterior. Esta operação está ilustrada na Figura 4-12. A lâmina que sofre a extrusão possui um contorno curvo e será necessário associar informações geométricas às novas arestas geradas. Além disso é preciso transladar os pontos de controle das curvas de Bézier.

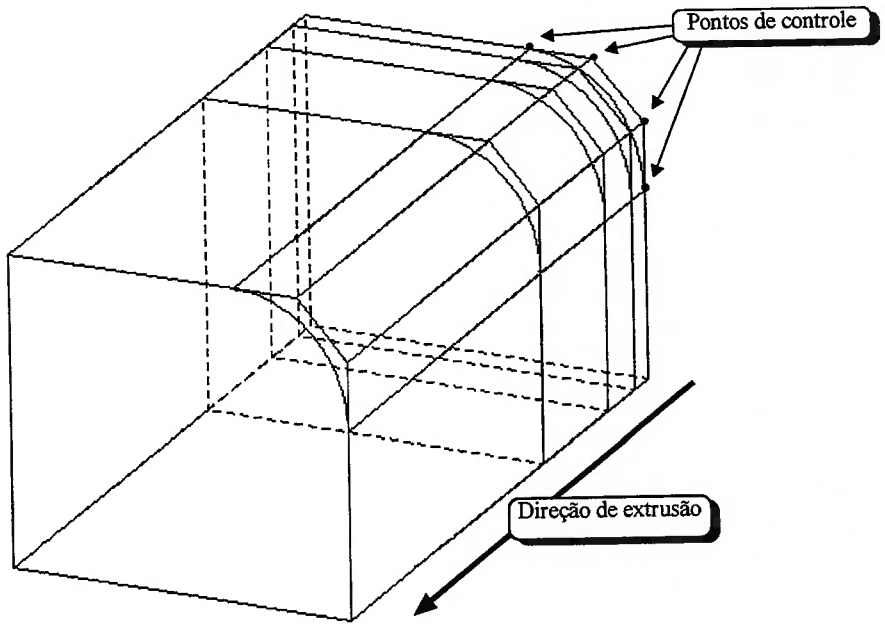


Figura 4-12 Um objeto obtido por extrusão translacional.

- **Extrusão rotacional:** foi implementado um algoritmo para extrusão rotacional similar ao algoritmo acima citado, combinado com o novo algoritmo para gerar arcos. Ao rotacionar uma face com arestas em torno de um eixo as arestas geradas de forma tangencial devem possuir informações sobre curvas.

- **Geração de Primitivos Básicos:** os sólidos primitivos que necessitam de modificações são o cilindro, o cone, a esfera e o toróide. A Figura 4-13 ilustra estes primitivos. As modificações são pequenas uma vez que a maior diferença concentra-se na geração de arcos e nas rotinas de extrusão translacional e rotacional.

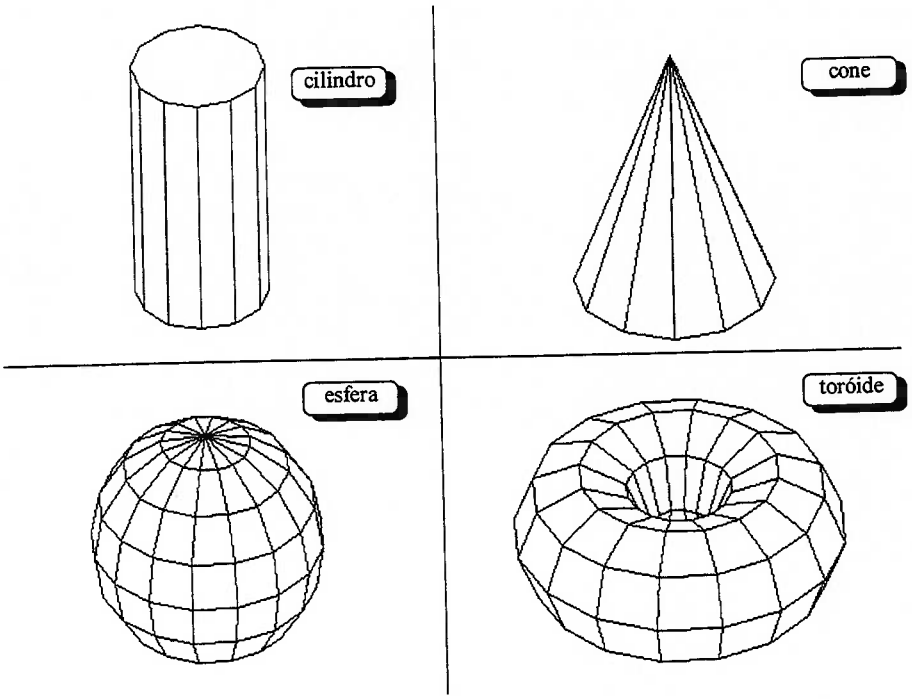


Figura 4-13 Primitivos básicos modificados.

4.4. Adaptação dos operadores globais

4.4.1. Secção por Plano

O algoritmo de secção por plano foi consideravelmente modificado para poder trabalhar com sólidos com curvas e superfícies. Analisaremos dois casos: no primeiro o plano de corte intercepta arestas que são curvas; no segundo caso não ocorre intercecção de curvas mas o plano intercepta superfícies.

1° caso: O plano de corte intercepta arestas com informação curva.

A Figura 4-14 (a) ilustra um cilindro na sua forma desenvolvida e a (b) na sua forma compactada. O plano de corte está representado em (c).

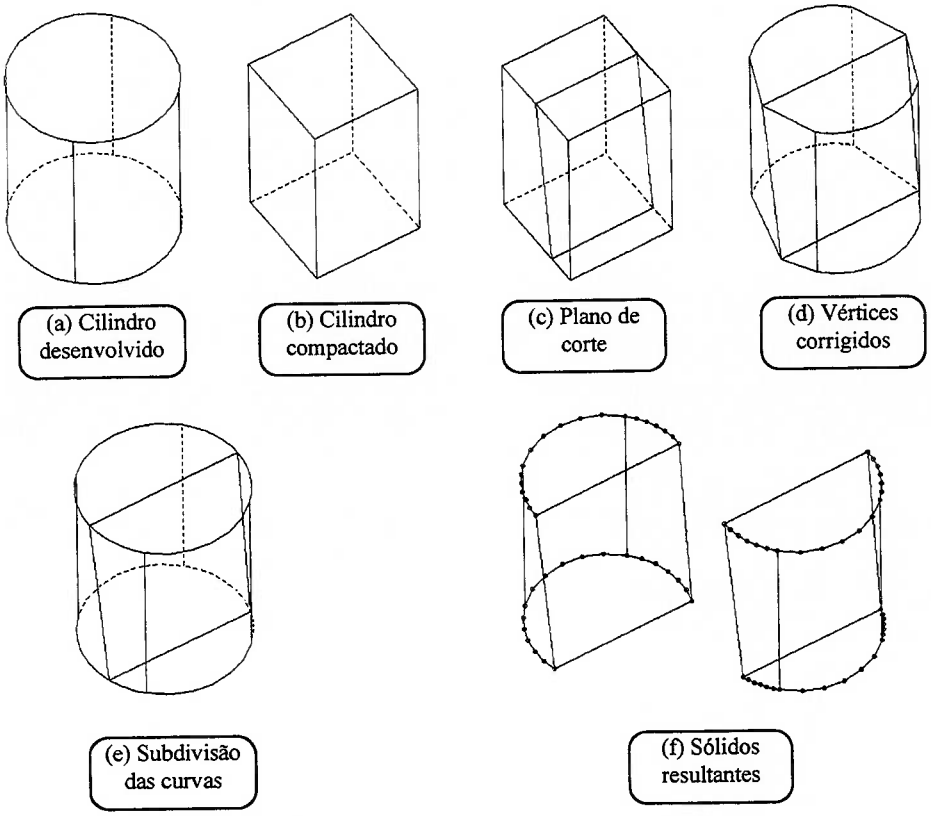


Figura 4-14 Secção por plano - caso I.

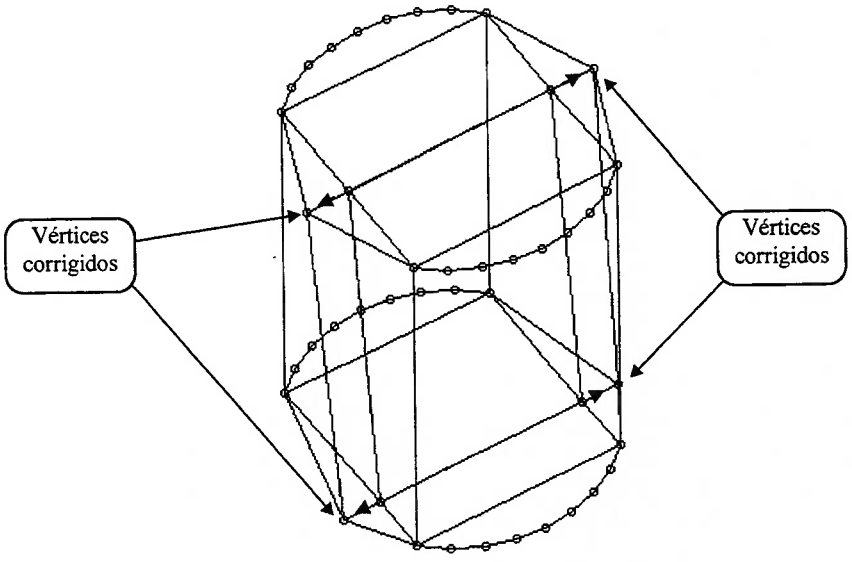


Figura 4-15 Detalhe da correção das coordenadas dos vértices de intersecção.

A primeira parte do algoritmo de secção, relaciona-se com a determinação dos vértices de intersecção. Uma vez determinados estes vértices, o novo algoritmo compacta as arestas interceptadas e processa os vértices de intersecção, verificando se estão geometricamente corretos. Na maioria dos casos será necessário corrigir sua posição geométrica. Isto é realizado utilizando-se a informação geométrica sobre a curva de Bézier correspondente. Calcula-se o ponto de intersecção entre a curva e o plano, e corrige-se as coordenadas do vértice de intersecção (Figura 4-14 (d) e Figura 4-15).

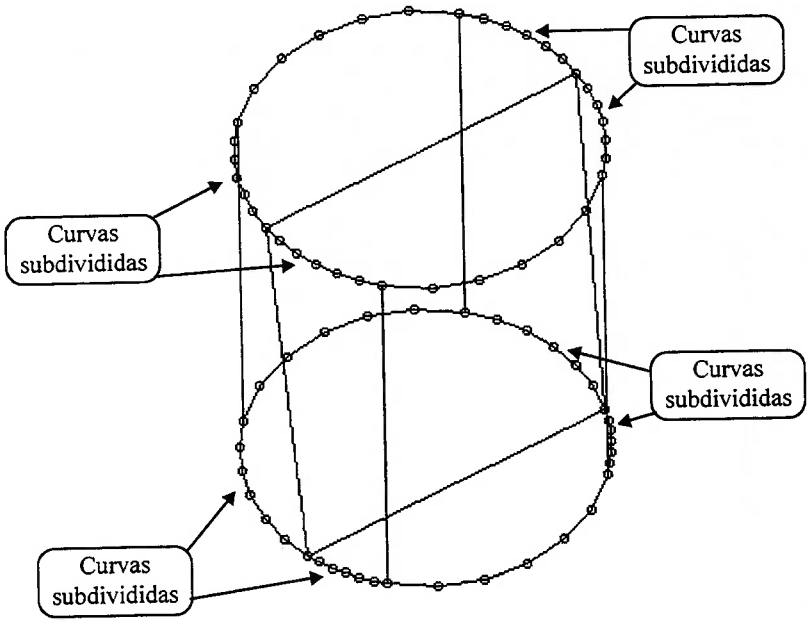


Figura 4-16 Detalhe da subdivisão das curvas.

Em seguida é preciso subdividir a curva interceptada e associar informações geométricas às novas arestas Figura 4-14(e). Na Figura 4-14(f) e na Figura 4-16 é possível notar que todas as arestas foram desenvolvidas com seis divisões.

O restante do algoritmo de secção por plano foi mantido idêntico ao utilizado no modelador poliédrico, com exceção de uma alteração realizada ao final do algoritmo e que está exemplificada no segundo caso.

2º caso: O plano de corte não intercepta arestas com informação curva.

Este caso é mais simples e a alteração no algoritmo para o novo modelador ocorre somente depois que o sólido já foi seccionado. A Figura 4-17 ilustra o exemplo de um cilindro e o plano de corte para este caso.

Uma vez realizada a operação de corte, o novo algoritmo analisa, em sua última etapa, as arestas sobre o plano de corte e verifica se alguma delas possui uma das faces adjacentes com atributo de superfície. Em caso afirmativo, calcula-se a curva de intersecção entre o plano de corte e a superfície de Bézier, utilizando-se o algoritmo Marching (vide Capítulo 3).

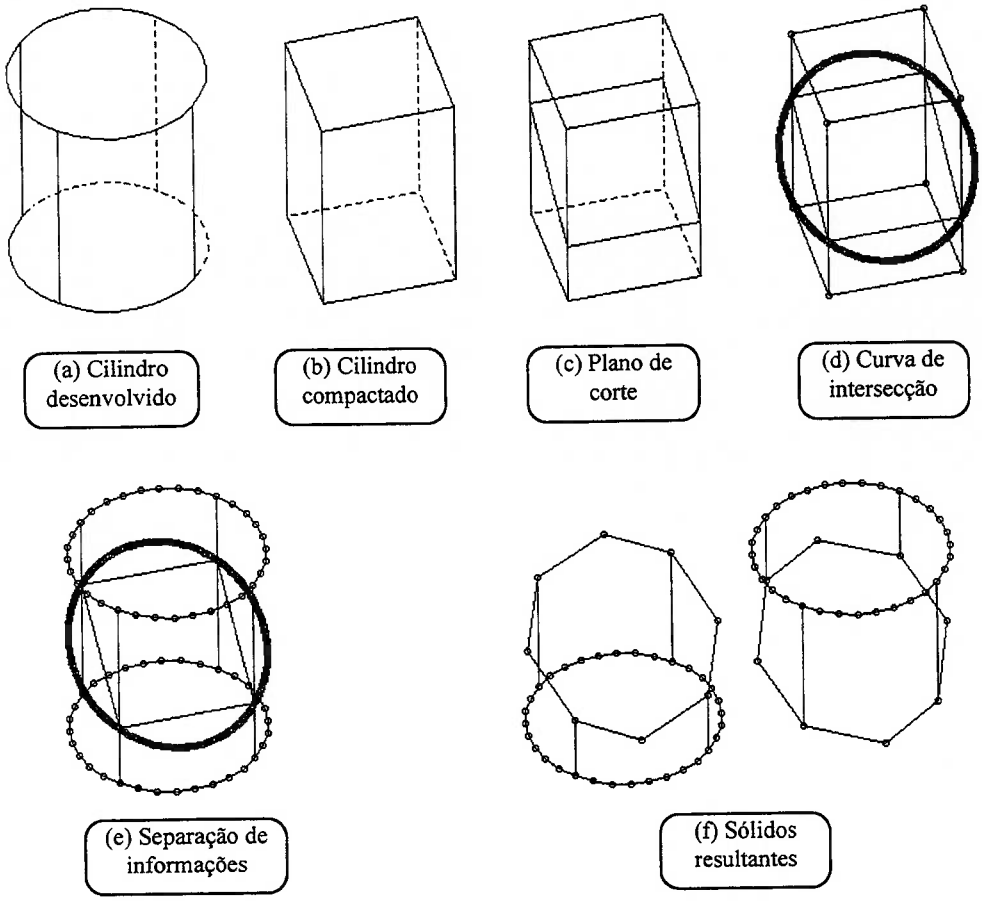


Figura 4-17 Secção por plano - caso II.

4.4.2. Operações Booleanas

Para trabalhar com curvas e superfícies o algoritmo de operações booleanas foi modificado considerando-se dois casos:

1° caso: as faces de um sólido interceptam arestas com informação curva.

A Figura 4-18 (a) ilustra dois sólidos que se interceptam: um bloco e um cilindro na sua forma desenvolvida. Na Figura 4-18 (b) o cilindro apresenta-se na sua forma compactada e os vértices de intersecção foram calculados. Em seguida o algoritmo desenvolvido verifica se há vértices de intersecção que pertencem a arestas com informação curva. Se isto ocorre o algoritmo calcula o ponto de intersecção da curva correpondente à aresta e à face interceptada e substitui as

coordenadas do vértice armazenado (Figura 4-18 (c)). A curva é subdividida e a informação curva nas arestas correpondentes é atualizada. Com as coordenadas corrigidas e as curvas subdivididas, continua-se o processamento pelo algoritmo de operações booleanas (Figura 4-18 (d)). Pode-se observar o resultado com as arestas desenvolvidas na Figura 4-18 (e)).

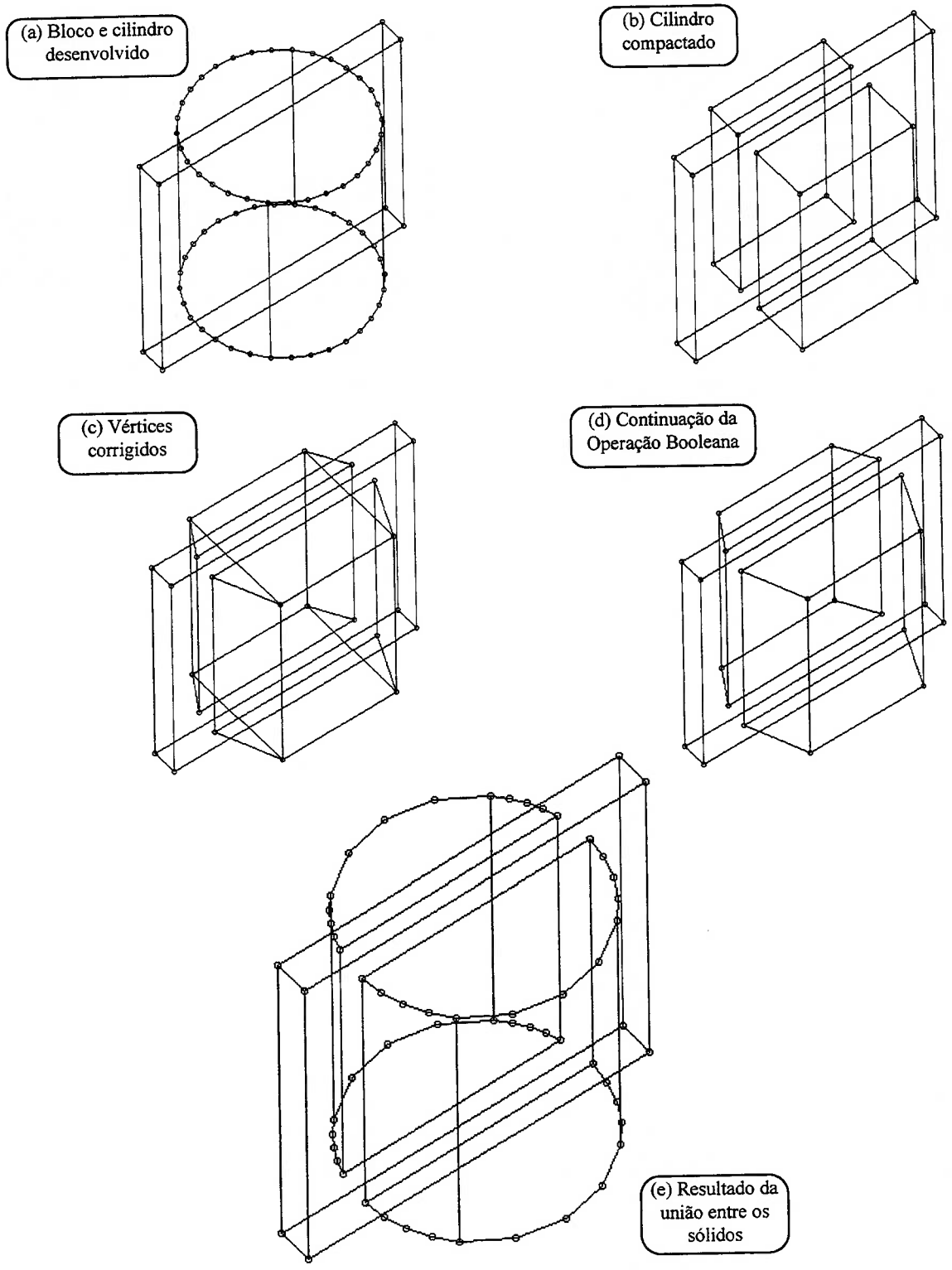


Figura 4-18 Caso I de Operação Booleana de sólidos com informação de curvas e superfícies.

2° caso: há intersecção entre os sólidos mas as arestas com informação geométrica não são interceptadas.

O exemplo da Figura 4-19 (a) mostra um bloco interceptando um cilindro na sua forma compactada. As arestas do cilindro com informação geométrica não interceptam as faces do bloco. Em conseqüência não será preciso processar as arestas com informação geométrica. Se existem arestas que não possuem informação geométrica ao final do algoritmo de operações booleanas, isto significa que existem arestas com informação geométrica incorreta (vide Figura 4-19 (b)). Verifica-se os atributos de superfície nas faces adjacentes destas arestas e calcula-se a curva de intersecção entre as superfícies encontradas, utilizando-se o algoritmo Marching descritos no Capítulo 3 (vide Figura 4-19 (c)). A Figura 4-19 (d) ilustra o resultado da união do bloco com o cilindro com as arestas desenvolvidas com vários segmentos.

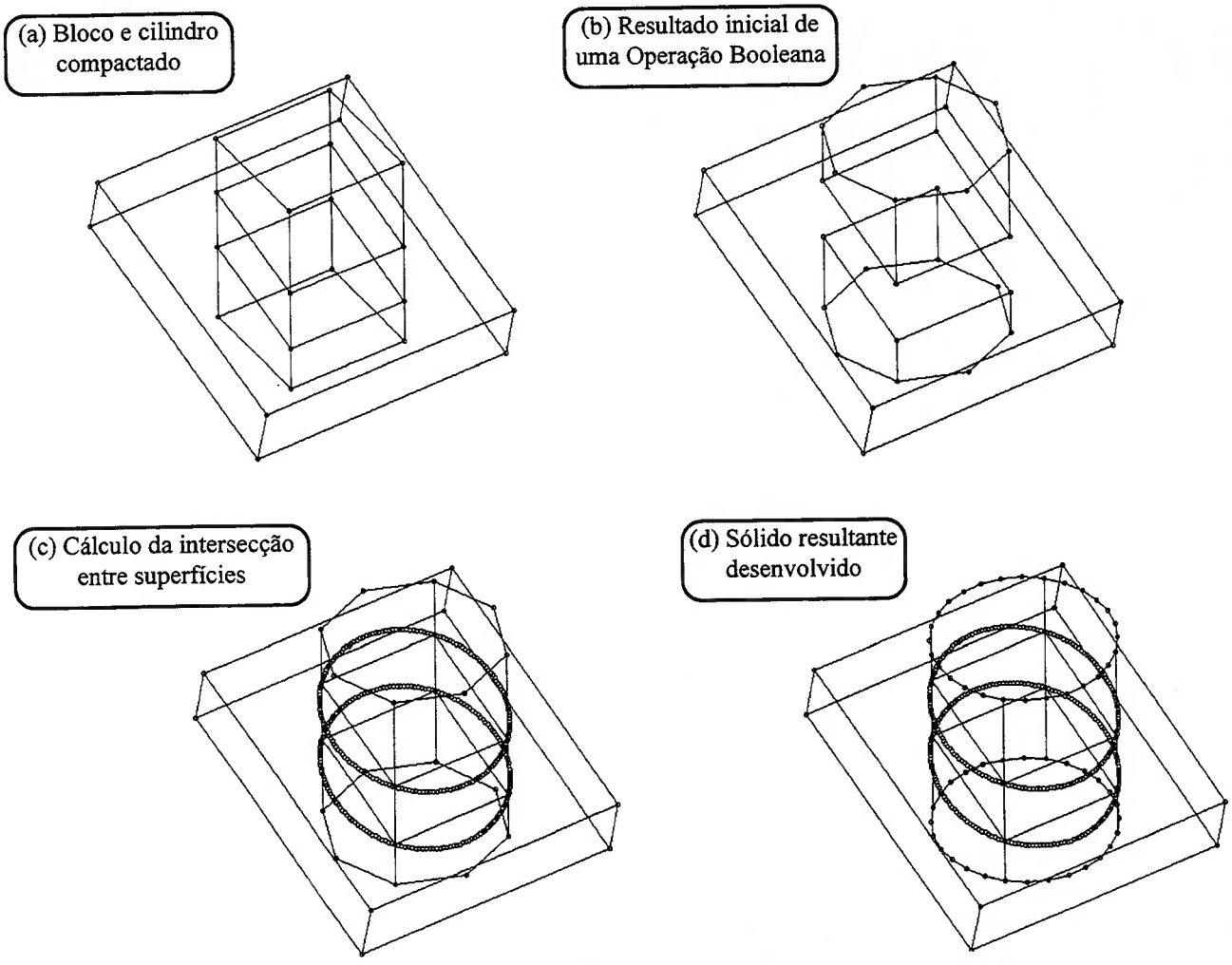


Figura 4-19 Caso II de Operações Booleanas de sólidos com curvas e superfícies.

5. Resultados

A nova estrutura de dados, explicada detalhadamente no capítulo anterior, é o principal resultado do trabalho. Neste capítulo vamos mostrar as possíveis aplicações das alterações propostas ao modelador de sólidos USPDesigner 1.0.

5.1. Adaptação dos operadores locais

5.1.1. Suavização de arestas

A Figura 5-1 mostra uma operação de suavização de aresta que ilustra a utilização de curvas e superfícies em um modelo sólido. A partir de um paralelepípedo (a), reposiciona-se os vértices da aresta que será arredondada (b, c, d, e) para formar uma "face de suavização" (f, g). Em seguida associa-se a informação geométrica às arestas laterais da face de suavização (h, i). Por último, associa-se a informação geométrica à face de suavização.

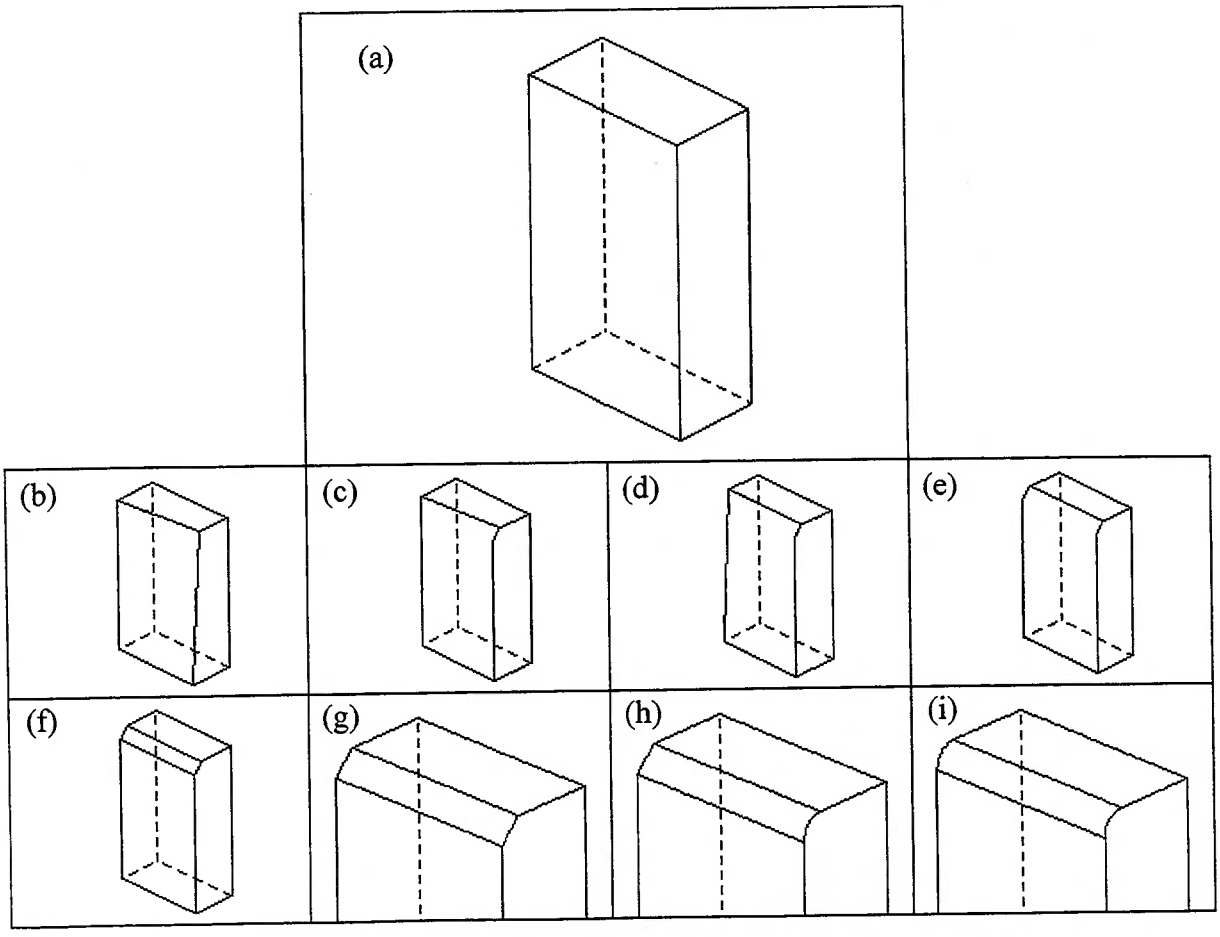


Figura 5-1 Seqüência da operação de Suavização de arestas.

As arestas com informação geométrica podem ser melhor visualizadas na Figura 5-2. Os círculos pequenos representam os vértices de aproximação da curva, que neste caso foi dividida em oito segmentos.

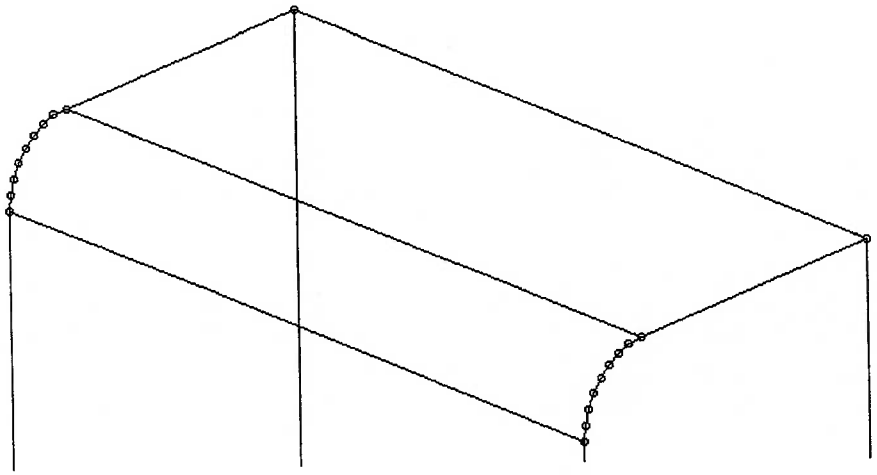


Figura 5-2 Detalhe da operação de Suavização de arestas.

5.1.2. Geração de malha sobre superfície

A superfície que caracteriza a face de suavização pode ser vista na Figura 5-3. A partir das informações sobre a superfície pode-se escolher um grau de aproximação poliédrica para a mesma. As Figura 5-3 (a), (b), (c) e (d) ilustram respectivamente malhas com 2x2, 3x3, 5x5 e 8x8 divisões.

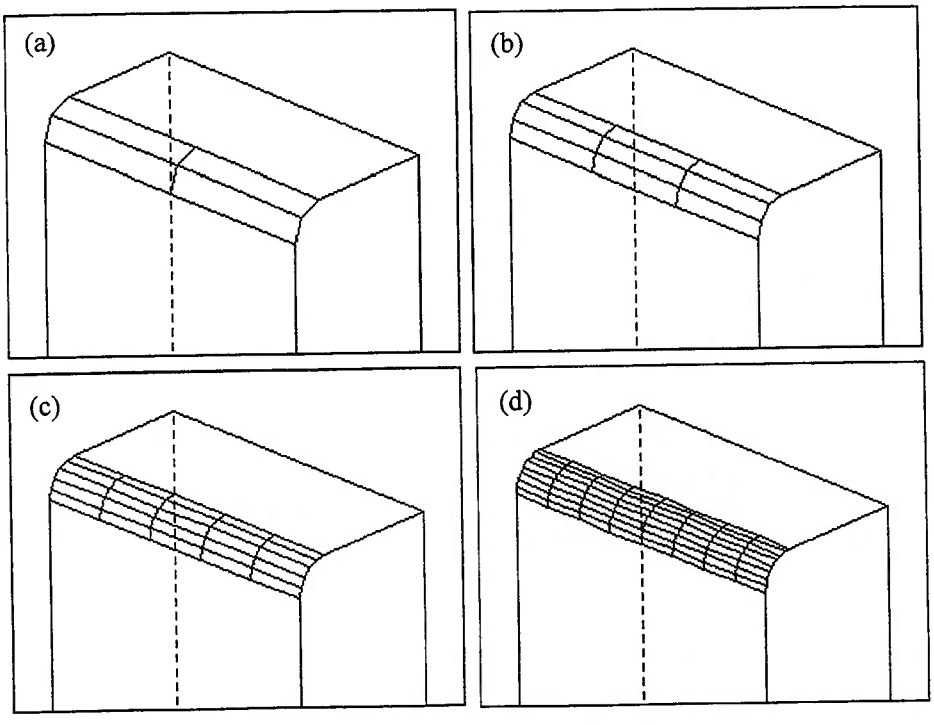


Figura 5-3 Exemplos de definição de malha de superfície.

5.1.3. Geração de primitivos básicos

Os operadores locais de extrusão translacional e rotacional são utilizados para gerar sólidos primitivos do modelador. Eles foram modificados para suportar as informações de curvas e superfícies. A Figura 5-4 (a) ilustra um cilindro compactado ou linearizado, ou seja, representado na sua forma mais simples.

Nas figuras (b) e (c) vemos a representação desenvolvida, onde cada aresta foi definida com seis divisões. As figuras (d), (e) e (f) mostram desenvolvimentos do cilindro com menos divisões.

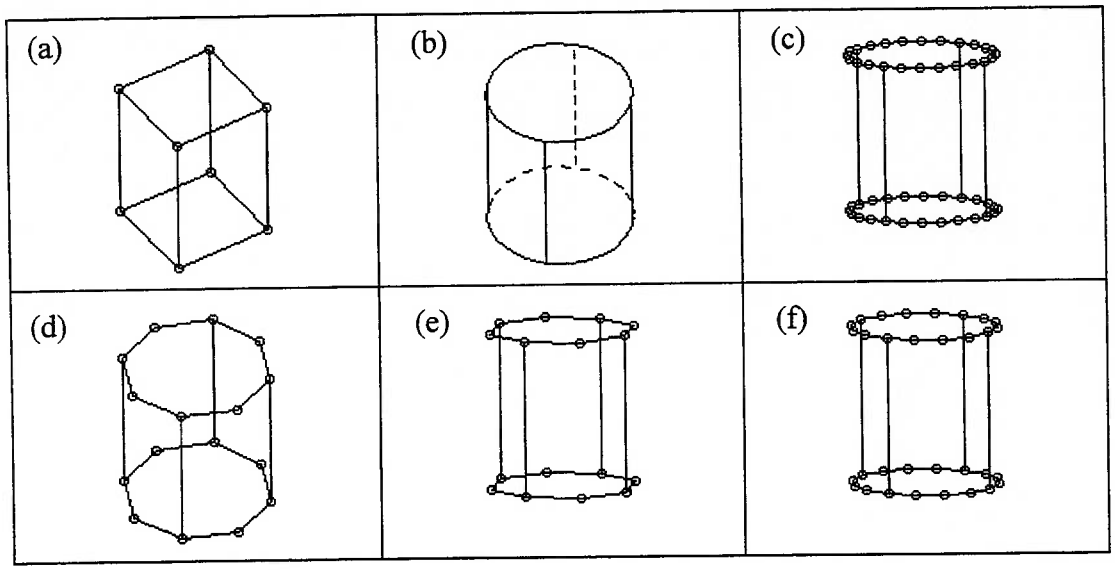


Figura 5-4 Exemplos de cilindros linearizados e desenvolvidos.

A Figura 5-5 (a) ilustra uma esfera compactada e a Figura 5-5 (b) mostra a mesma esfera desenvolvida com 8 divisões em cada aresta.

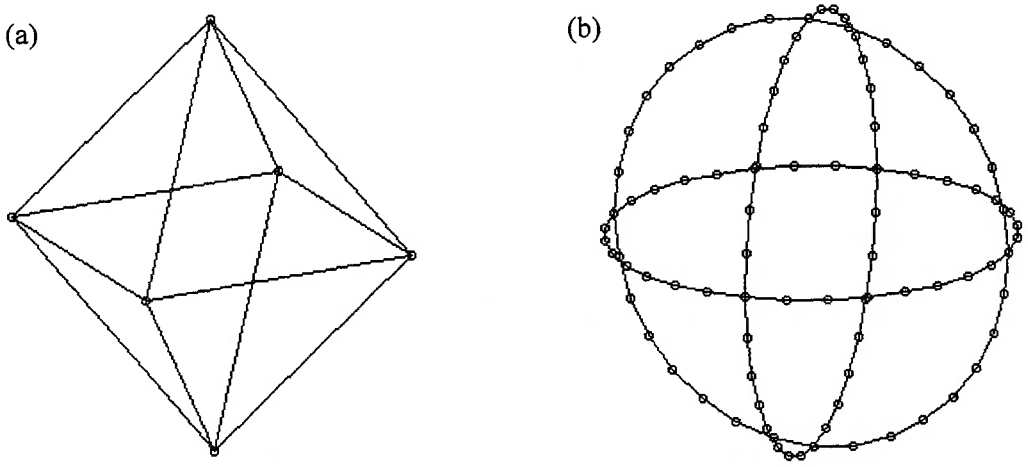


Figura 5-5 Esfera linearizada e desenvolvida.

A partir de uma circunferência, utilizando-se o algoritmo de extrusão rotacional, é possível gerar um toróide. Sua forma compactada está ilustrada na Figura 5-6 (a). Com cinco divisões por aresta ele pode ser visualizado nas Figura 5-6 (b), (c) e (d), sob ângulos diferentes.

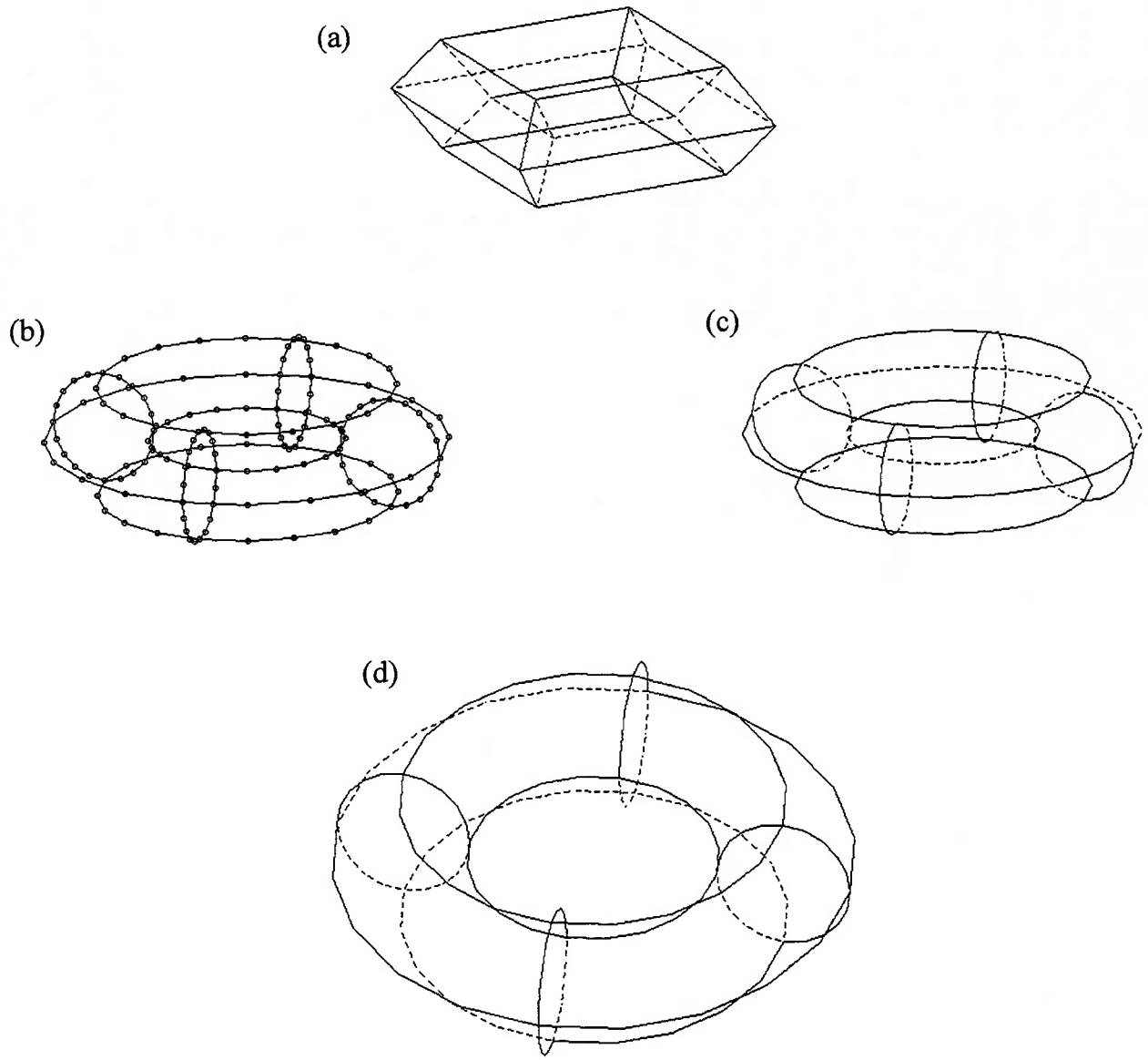


Figura 5-6 Toróide linearizado e desenvolvido.

5.2. *Adaptação dos operadores globais*

5.2.1. Secção por Plano

Um exemplo de secção por plano que utiliza os algoritmos dos dois casos expostos no capítulo 4 pode ser visto na Figura 5-7. O plano de corte secciona parte das superfícies do cilindro e também divide as curvas da base do cilindro. As figuras (a), (b) e (c) ilustram respectivamente o sólido com o plano de corte e os resultados separados da secção. As figuras (d), (e) e (f) mostram os mesmos resultados com as arestas subdivididas.

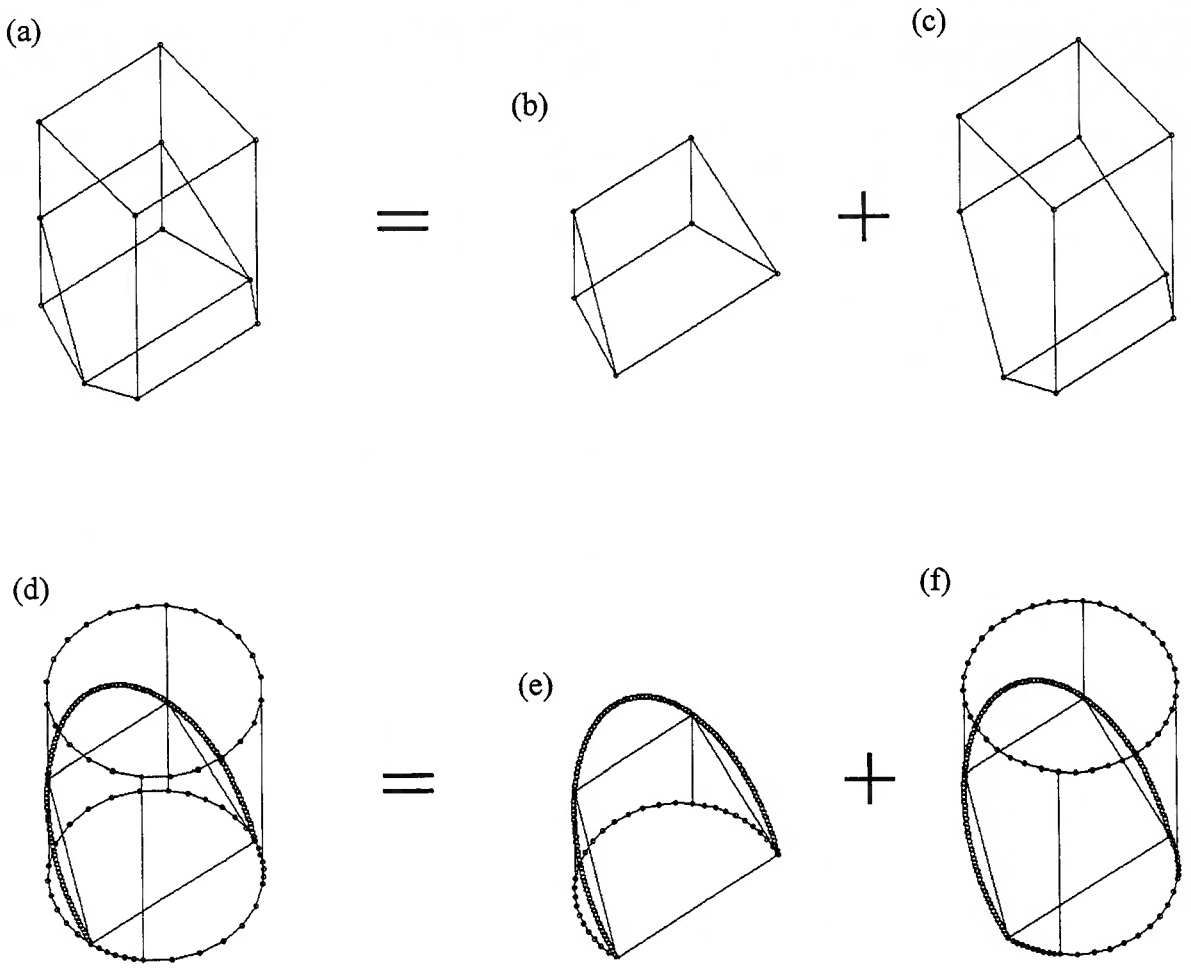


Figura 5-7 Aplicação do algoritmo proposto para Operação de Secção por Plano.

5.2.2. Operações Booleanas

A Figura 5-8 ilustra todas as opções de Operação Booleana do exemplo citado no capítulo 4.

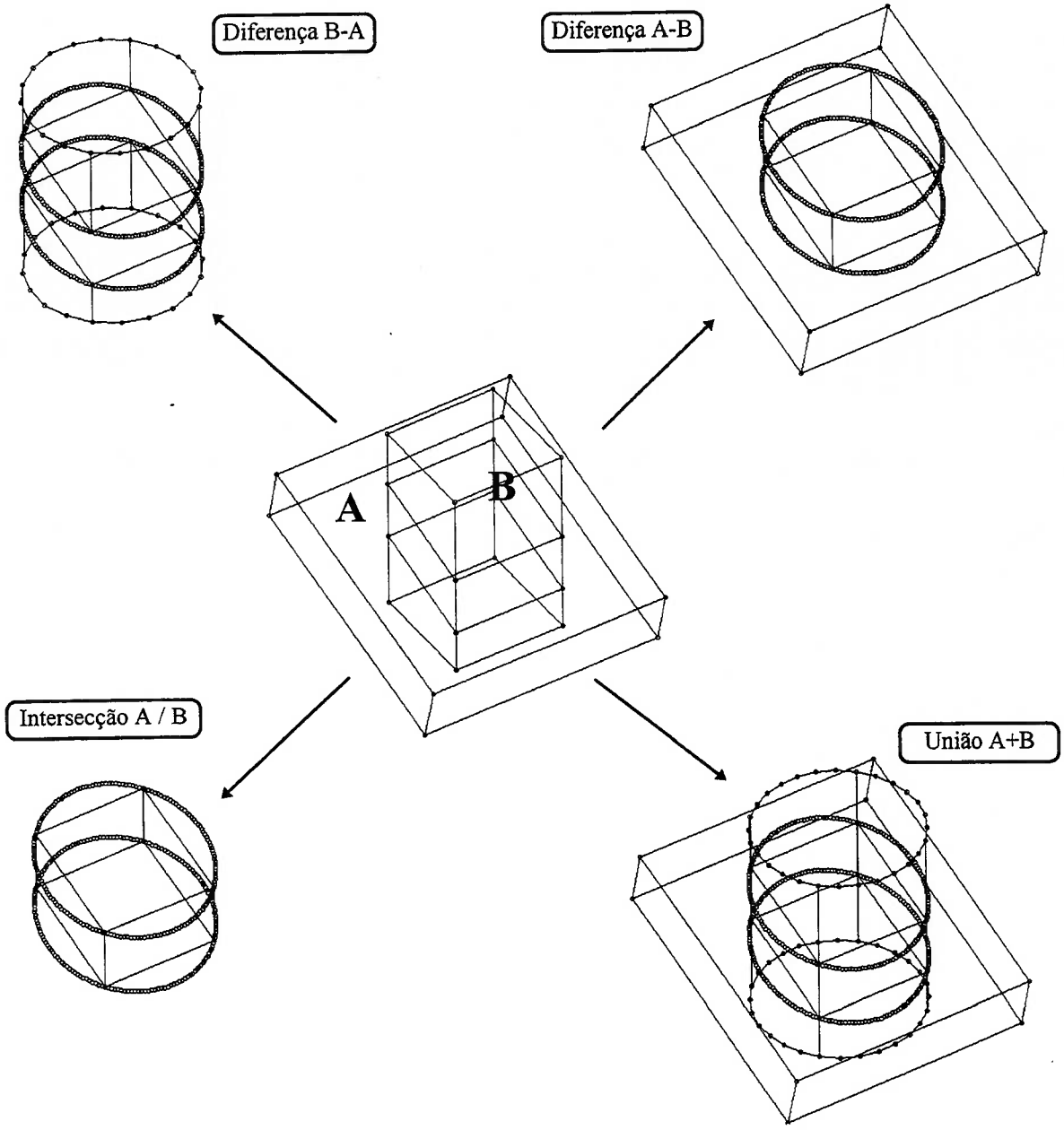


Figura 5-8 Exemplo de Operação Booleana utilizando o algoritmo proposto.

5.3. Adaptação de outras ferramentas

Os algoritmos de rotação e translação de objetos também sofreram uma ligeira modificação para poderem operar com informação de curva e superfície. A Figura 5-9 ilustra alguns exemplos destes procedimentos.

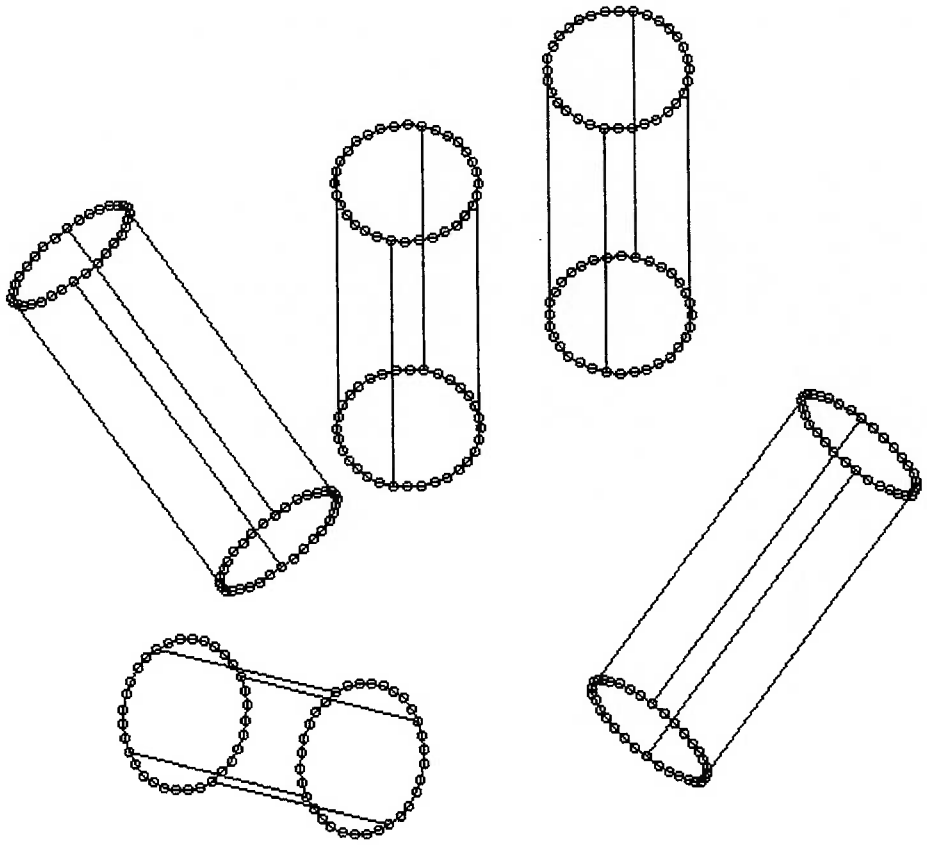
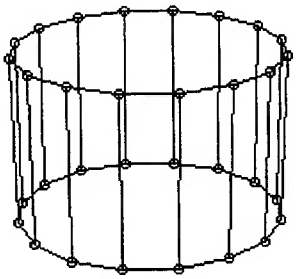


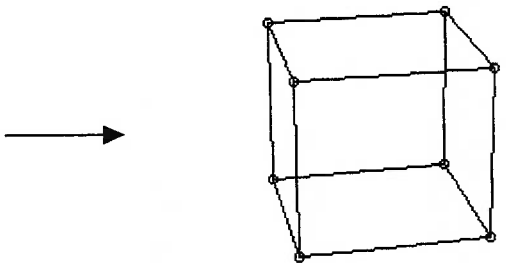
Figura 5-9 Rotação e translação de sólidos com informação geométrica associada.

6. Conclusões e discussões

A utilização da modelagem de sólidos está tendendo a desenvolver-se mais e tornar-se mais popular. Neste trabalho foi desenvolvido e apresentado um modelador de sólidos com estrutura B-Rep que incorpora a representação de curvas e superfícies de Bézier. Este sistema CAD é uma proposta para unificar duas técnicas de projeto comumente utilizadas em engenharia: a modelagem de sólidos e a modelagem geométrica. O modelador de sólidos desenvolvido, ao armazenar de forma explícita as informações geométricas sobre superfícies e curvas, possibilita que o usuário modele com mais precisão os objetos que possuem geometrias não planas. O modelador desenvolvido harmoniza as informações topológicas e geométricas.

Com relação à memória utilizada para armazenar os sólidos pode-se dizer que, apesar de incorporar novas informações, a nova estrutura ocupa menos de memória. Um exemplo é o cilindro ilustrado na Figura 6-1 (a), que está representado pela antiga estrutura de dados e ocupa a memória correspondente aos elementos listados. A Figura 6-1 (b) ilustra o mesmo sólido na forma compactada e a quantidade de elementos que este modelo possui. A quantidade de elementos topológicos na representação antiga é três a quatro vezes maior que a quantidade de elementos na nova estrutura. Entretanto as estruturas que armazenam as informações geométricas ocupam uma grande quantidade de memória, anulando parte da economia obtida com a redução dos elementos topológicos.





(a)

Topologia	Geometria
18 faces	
48 arestas	
32 vértices	96 pontos

(b)

Topologia	Geometria
6 faces	4 superfícies
12 arestas	8 curvas
8 vértices	24 pontos

Figura 6-1 Representação de um cilindro.

As operações propostas para sincronizar as informações topológicas e geométricas foram denominadas por operações de desenvolvimento e compactação do sólido. Elas são muito úteis não somente para armazenar concisamente o sólido mas também para visualizar e realizar algoritmos de varredura de arestas e faces mais rapidamente. No método proposto os algoritmos de corte por plano e de operações booleanas sofrem pequenas modificações para garantir a precisão geométrica dos resultados.

A estrutura de dados proposta comporta somente curvas e superfícies de Bézier de terceiro grau. Um próximo aperfeiçoamento desta estrutura pode ser a incorporação de curvas e superfícies cônicas e outras curvas e superfícies sintéticas.

Durante o desenvolvimento do novo modelador de sólidos foi verificada uma forte limitação da proposta feita por MÄNTYLÄ (1988) para algoritmos de implementação de operações booleanas e secção por plano no caso de utilizarmos informações geométricas. A proposta feita por MÄNTYLÄ (1988) determina vértices de intersecção, entre dois sólidos no caso de operações booleanas (Figura 2-29 (a)). Entretanto, algumas situações de intersecção não envolvem vértices e sim curvas de intersecção. Neste caso uma superfície intersecciona outra superfície definindo um laço de intersecção.

Outra limitação existente na atual implementação relaciona-se ao algoritmo de Marching. Caso as duas superfícies possuam na sua intersecção planos tangentes coplanares, o algoritmo não converge, impossibilitando a determinação da curva de intersecção.

Apesar das limitações encontradas, foi possível desenvolver um modelador de sólidos geométrico, utilizando a estrutura proposta, bem como demonstrar a sua aplicabilidade com as modificações feitas no modelador de sólidos poliedral USPDesigner 1.0.

7. Bibliografia

BARTELS, R.H.; BEATTY, J.C.; BARSKY, B.A., **An Introduction to Splines for use in Computer Graphics & Geometric Modeling**. Morgan Kaufmann Publishes, Inc, 1987.

BAUMGART, B.G., **Geometric modelling for computer vision**. Technical report, Report STAN-CS-74-463, Stanford University: Stanford Artificial Intelligence Laboratory, 1974.

BÉZIER, P., **The Mathematical Basis of the UNISURF CAD system**, Butlerworths, 1986.

CHIYOKURA, H., **Solid Modelling with DesignBase**, Addison-Wesley Publishing Company, 1988.

FARIN, Gerald E., **Curves and surfaces for computer aided geometric design: a practical guide**. 2.ed., San Diego, Academic Press, 1990.

HOSAKA, M., **Modeling of Curves and Surfaces in CAD/CAM**, Berlin, Springer-Verlag, 1992.

KALAY, Y. E., **The Hybrid Edge: A Topological Data Structure for Vertically Integrated Geometric Modelling**, Computer Aided Design, 21(3):130-140, April 1989.

MÄNTYLÄ, M., **An Introduction to Solid Modeling**, Rockeville, Computer Science Press, 1988.

NEWMAN, W.M.; SPROULL, R.F., **Principles of Interative Computer Graphics**, Second Edition, McGrawHill, 1979.

REQUICHA, A. A. G., **Representation for rigid solids: Theory, method and systems**, ACM Computing Surveys, 12(4):437-464, 1980.

TORIYA, H.; CHIYOKURA H.; **3D CAD Principles and Applications**. Berlin, Springer-Verlag, 1991.

TSUZUKI, M. S. G, **Modelagem de Sólidos: Modelos Limitantes (B-rep)**, Departamento de Engenharia Mecânica da EPUSP, 1991.

WEILER, K., **Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments**, IEEE Computer Graphics & Applications, 5(1):21-39, 1985.

YAMAGUCHI, F., **Curves and Surfaces in Computer Aided Geometric Design**, Berlin, Springer-Verlag, 1989.

ZEID, I., **CAD/CAM Theory and Practice**, Singapura, McGraw-Hill, 1991.