MSD - Modelador de Sólidos Didático

MARCOS DE S. G. TSUZUKI

EPUSP–Escola Politécnica da USP
Departamento de Engenharia Mecânica - Mecatrônica
Av. Prof. Mello Moraes, 2231
mtsuzuki@bruspvm.bitnet

Abstract. The CAD group of EPUSP-Mecânica/Mecatrônica developed a Solid Modeling System for educational and research purposes. This system was named MSD (Modelador de Sólidos Didático - Educational Solid Modeler). This article is divided in two parts. In the first part, it will be described the functions supported by the system. These functions are classified in ten groups: creation of primitives, hierarchical structure, transformation, display, analysis, Boolean Operators, Split, Animation, Undo and Others. In the second part, several details are going to be explained. Specially concerning the Hierarchical Structure, the Boolean Operators and the Undo Operation. Specifically, it will be discussed how to solve the problem found in Mäntylä's approach [Mäntylä (1988)] and how the "shell" element will influence the implementation of the Boolean Operators. Finally, it will be shown some aplications of the MSD (CAPP and Robotic Kinematics Simulator).

1 Introdução

O grupo de CAD da EPUSP-Mecânica/Mecatrônica desenvolveu um Sistema de Modelagem de Sólidos para fins didáticos e de pesquisa, que foi batizado com o nome Modelador de Sólidos Didático ou simplesmente MSD. Neste trabalho, inicialmente, descreveremos o MSD segundo a visão do usuário e numa segunda parte explicaremos detalhes da implementação do sistema que esperamos que sejam úteis para outras pessoas interessadas. Os detalhes estarão especialmente relacionados à estrutura hierárquica e aos Operadores Booleanos (conseqüências de se acrescentar o elemento "shell" à estrutura de dados). Também apresentaremos duas aplicações que foram desenvolvidas baseando-se no MSD.

O MSD foi projetado segundo três níveis de abstração. No nível inferior, representando o "hardware" de nosso sistema, foi utilizada a estrutura aresta-dividida que é uma variação da estrutura "winged-edge". Um sólido no MSD é representado por meio de uma estrutura hierárquica que possui os elementos primitivos: sólido, "shell", face, laço, aresta e vértice.

O nível intermediário contém a infraestrutura matemática e algorítmica. Neste nível foram utilizados dois conceitos básicos: Operadores de Euler para manipular informações topológicas do nível inferior e Procedimentos Geométricos Robustos para manipular as informações geométricas. O conceito principal deste nível é simplificar a manipulação das informações topológicas e eliminar instabilidades de cálculo. O nível superior consiste das funções disponíveis ao usuário. Nas próximas seções apresentaremos detalhes de algumas funções do usuário.

2 Funções do MSD

O usuário interage com o MSD por meio de uma linguagem de comandos. Estes comandos podem ser agrupados em dez grupos principais: criação de primitivos, estrutura hierárquica, transformação, exibição, propriedades de cálculo integral, operadores booleanos, corte, animação, "undo" e outros.

3 Animação

O MSD utiliza a técnica de "frames" chaves para criar uma animação. Os frames intermediários entre dois frames consecutivos são criados por meio de interpolações lineares. A animação é criada em três passos principais: definição da estrutura de movimento, filtro da estrutura vetorial dos frames, e carregamento da estrutura da animação. A estrutura de movimentos armazena as transformações que serão aplicadas aos eixos de transformação. Cada frame possui a sua própria estrutura de movimento. A ordem em que os movimentos são fornecidos é muito importante, pois as transformações serão aplicadas nesta ordem fornecida. A cada iteração, durante a geração dos frames intermediários, as linhas escondidas são eliminadas. O frame é filtrado para evitar que arestas muito pequenas ou coincidentes sejam exibidas. Após a criação de todos os frames intermediários, a estrutura da animação é carregada e executada.

4 Estrutura Hierárquica

A estrutura hierárquica permitiu-nos implementar o mecanismo de herança associado à sólidos. Em nosso caso particular, necessitamos estudar mecanismos de cadeia aberta (p. ex., um robô). Este tipo de problema pode ser solucionado por meio de um mecanismo de herança de transformações, isto é, ao aplicarmos uma transformação a um grupo, todos os descendentes deste grupo também sofrem a transformação. Para implementar a estrutura hierárquica necessitamos apenas de um ponteiro para o ascendente.

Em robótica é muito utilizado o conceito de Coordenadas de Junta, isto é, associa-se a um grupo um sistema de coordenadas e quando um ascendente deste grupo sofre alguma transformação o sistema de coordenadas também será transformado. Para implementar o conceito de Coordenadas de Junta, foi criada a estrutura **axis** (vide figura 1). É possível definir dois tipos de eixos: eixo de rotação e translação. O tipo do eixo está codificado no campo **amode**. As estruturas que implementam o eixo e o grupo estão relacionadas entre si.

```
typedef struct grouptp GTYPE ;
typedef struct axistp  ATYPE ;

struct axistp {
   Id     axisno    ;
   char   type      ;
   GTYPE *group     ;
   char   amode     ;
   vector acenter   ;
   vector avector   ;
} ;

struct grouptp {
   Id     groupno   ;
   GTYPE *parent    ;
   ATYPE *gaxs      ;
   char   gcolor    ;
} ;
```

Figura 1: Estrutura hierárquica.

5 Operadores Booleanos

Revisando os algoritmos propostos na literatura [Chiyokura (1988), Mäntylä (1988)], encontramos a impossibilidade de processar sólidos com vários "shells". Em parte, esta insuficiência ocorre porque o "shell" não está representado explicitamente na estrutura de dados. Um sólido pode ser composto por vários

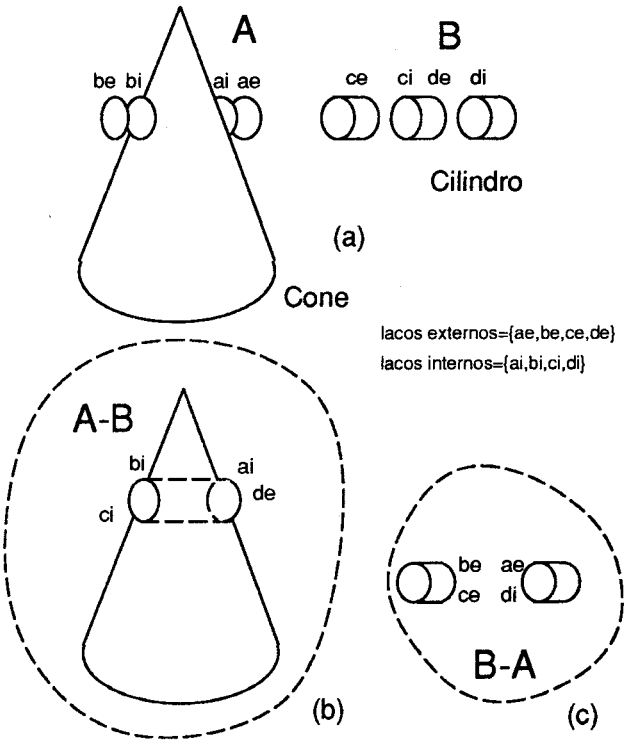

Figura 2: Exemplo de uma Operação Booleana entre um cone e um cilindro. Em (a) as faces de separação entre o cone e o cilindro foram criadas. Cada face de separação possui dois laços (um interno e outro externo). Em (b) o sólido resultante da operação $A - B$ é ilustrado. Em (c) o sólido resultante da operação $B - A$ é ilustrado.

"shells" e um "shell" é composto por várias faces. Especificamente, o algoritmo para implementar as Operações Booleanas entre dois sólidos proposto por Mäntylä [Mäntylä (1988)] possui quatro passos. No primeiro passo, as intersecções entre os dois sólidos são determinadas. Em seguida, os vários casos de intersecção são classificados em dois grupos: vértice-vértice e vértice-face. Neste mesmo passo, a vizinhança de cada vértice é classificada em externa e interna, e são inseridas arestas nulas para separar as vizinhanças externas das vizinhanças internas. No terceiro passo, as arestas nulas são unidas entre si e removidas para que uma face de separação seja criada. A face de separação sempre possuirá dois laços: um laço interno e um laço externo. A figura 2(a) ilustra o terceiro passo da diferença entre um cilindro e um cone. E no quarto passo, as partes convenientes são unidas, gerando o resultado da Operação Booleana (as figuras 2(b) e 2(c) ilustram o quarto passo da diferença entre o cilindro e o cone).

Com a inserção do elemento "shell", apenas o

```
<identifica quais shells foram manipulados
                              pelo algoritmo>
if (op == DIFERENCA)
     <inverte os shells do solido B>
<cria novos shells a partir das faces de
                                  separacao>
<transfere os novos shells para o solido
                                  resultado>
<transfere os shells restantes para o
                           solido resultado>
```

Figura 3: Operador Booleano (quarto passo).

primeiro e quarto passos devem sofrer alterações. O principal conceito é identificar dois grupos de "shells": "shells" que interseccionam o outro sólido e "shells" que não interseccionam o outro sólido. Os "shells" que interseccionam o outro sólido são tratados corretamente pelo algoritmo; entretanto, os "shells" que não interseccionam o outro sólido devem ser classificados e processados. Portanto, no primeiro passo é necessário classificar os "shells" de um sólido em relação ao outro sólido. Na prática, necessitamos classificar apenas um vértice de cada "shell" em relação ao outro sólido, pois estamos interessados em classificar apenas os "shells" que não interseccionam o outro sólido. Para classificar um ponto em relação a um sólido é necessário determinar se o ponto está interno ou externo a cada "shell" do sólido. O resultado desta classificação deve ser armazenado em uma estrutura intermediária que será utilizada no quarto passo.

O quarto passo será completamente alterado, o algoritmo da figura 3 ilustra estas alterações. Até o início do quarto passo nenhum "shell" além daqueles presentes nos sólidos iniciais foi criado; portanto, é possível identificar quais "shells" interseccionam o outro sólido e quais "shells" não interseccionam. A presença das faces de separação identifica os "shells" em que ocorre intersecção com o outro sólido. Caso seja o Operador Booleano Diferença, o sólido B é invertido. Em seguida criam-se novos "shells" pela separação dos laços internos e externos das faces de separação, para isto pode-se utilizar o Operador de Euler **msfkr** (Make Shell Face Kill Ring). Nesta separação são criadas laços de separação.

Em seguida os "shells" manipulados são transferidos para o sólido resultante. Esta operação equivale a retirar um "shell" de um sólido e inseri-lo no sólido resultante. Após a transferência dos "shells", um laço de separação proveniente do sólido A deve ser aglutinado a um laço de separação proveniente do sólido B. Para realizar a aglutinação dos laços é necessário verificar se estamos manuseando laços do mesmo "shell" ou de "shells" distintos. Para isto, devemos aplicar, respectivamente, os Operadores de Euler **kfmrh** (Kill Face Make Ring Hole) e **ksfmr** (Kill Shell Face Make Ring).

Tabela 1: Tabela com a classificação de "shells".

	$AinB$	$AoutB$	$BinA$	$BoutA$
inter.	sim	não	sim	não
união	não	sim	não	sim
difer.	não	sim	não	sim

Neste momento, todos os "shells" que possuem intersecção com o outro sólido já estão devidamente processados, e, finalmente, devemos transferir os "shells" que não interseccionam o outro sólido. A intersecção deverá ocorrer analisando-se a classificação realizada no primeiro passo, conforme a tabela 1.

A função *corte* é uma simplificação dos Operadores Booleanos, a principal diferença é que no primeiro passo os "shells" devem ser classificados em relação ao plano de corte (acima e abaixo).

6 "Undo"

Mäntylä identificou que o "Undo", armazenamento e carregamento da estrutura de dados do Modelador de Sólidos B-Rep possuem como solução o mesmo artifício: Operadores de Euler. Quando novas estruturas são acrescentadas ao Modelador é necessário definir novos Operadores de Euler (p. ex.: Operadores de Euler para a Estrutura Hierárquica). Segundo este contexto, aconselhamos que para cada novo Operador de Euler sejam criadas duas rotinas para executá-lo (Operador de Euler de alto nível e Operador de Euler de baixo nível), uma rotina para armazená-lo e uma rotina para carregá-lo.

Desta maneira, as funções que realizam o "Undo" e carregam e armazenam a estrutura de dados podem se tornar independentes dos Operadores de Euler em si. Basta que exista uma matriz definida conforme a listagem da figura 4, aonde associa-se ao código do Operador de Euler, a rotina de alto nível que o executa, a rotina que o imprime e a rotina que o carrega.

7 Aplicativos

Foram desenvolvidas algumas aplicações baseadas no MSD. Nesta seção descrevemos os sistemas que foram desenvolvidos e estão operacionais.

```
typedef struct eulercod EuCode ;

struct eulercod {
  OpCode ecode ;
  int (*xeop)(EulerOp *op) ;
  void (*peop)(FILE *f, EulerOp *op) ;
  int (*reop)(FILE *f, EulerOp *op) ;
} ;

EuCode ematrix[] = {
  MVSF , xmvsf , pmvsf , rmvsf ,
  KVSF , xkvsf , pkvsf , rkvsf ,
  MEV  , xmev  , pmev  , rmev  ,
  KEV  , xkev  , pkev  , rkev
}
```

Figura 4: Estrutura de codificação dos Operadores de Euler.

7.1 CAPP automático

Foi desenvolvido um sistema CAPP [M.S.G.Tsuzuki et al. (1991)] que realiza a ponte entre o processo de projeto e o processo de manufatura. O sistema é baseado no conceito de "features", que são regiões de interesse sobre a superfície de um sólido (p. ex.: furos, rasgos, etc. (vide figura 5). A cada feature é possível associarmos um processo de fabricação. A partir de um modelo sólido do produto (representação B-Rep), o sistema realiza o reconhecimento e o escalonamento das features presentes no sólido e fornece como resultado a sua folha de processos. O reconhecimento das features é realizado segundo um método baseado em comparação de grafos.

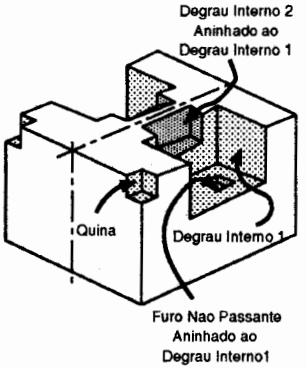

Figura 5: CAPP automático.

7.2 Simulador cinemático de robôs

Foi proposto um método para realizar a cinemática inversa de manipuladores redundantes [R.Matone et al. (1992)]. Para realizar a cinemática inversa é necessário possuir o modelo do robô a ser analisado e a trajetória que a garra deve executar. Com estes dados, é possível determinar quais os deslocamentos angulares e lineares devem ser realizados pelas juntas do robô. Para ilustrar o método proposto, foram realizadas simulações com o robô ABB IRBL6 com trilho. Para facilitar a interpretação dos resultados da simulação, os algoritmos foram interfaceados com o MSD.

8 Conclusão

Estamos utilizando o MSD para fins didáticos e de pesquisa. Atualmente, estamos incorporando uma interface homem-máquina mais amigável, baseada em janelas, botões e menus. Para um futuro próximo desejamos incorporar a representação explícita de superfícies de segundo grau (esferas, cilindros e toróides).

Também estamos realizando pesquisas para definir uma representação consistente para "features" paramétricas.

9 Referências

M. Mäntylä, "An Introduction to Solid Modeling", *Computer Science Press*, **1988**.

H. Chiyokura, "Solid Modeling with DESIGNBASE", *Addison-Wesley Publishing Company*, **1988**.

M. S. G. Tsuzuki, P. E. Miyagi, L. A. Moscato, "Automatic Feature Recognition", *Anais do COMPUGRAPHICS'91 - First International Conference on Computational Graphics and Visualization Techniques*, Sesimbra, Portugal, pp. 92-101, **1991**.

M. S. G. Tsuzuki, P. E. Miyagi, "Reconhecimento Automático de features baseado na representação B-Rep", *Anais do XI Congresso Brasileiro de Engenharia Mecânica*, São Paulo, Brasil, pp. 607-610, **1991**.

R. Matone, M. S. G. Tsuzuki, E. L. L. Cabral, P. E. Miyagi, L. A. Moscato, "An integrated system for inverse kinematics using a new numerical method", *Anais do IMACS-SICE RM2S'92 - International Symposium on Robotic, Mechatronics and Manufacturing Systems*, Kobe Japão, pp. 707-712, **1992**.