Artigo

SIBGRAPI'91
IV Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens

CRIAÇÃO E MANUTENÇÃO DE SUBDIVISÕES PLANARES

Paulo Roma Cavalcanti
(IMPA/CNPq)

Paulo Cezar P. Carvalho
(IMPA/CNPq)

Luiz Fernando Martha
(IMPA/CNPq)

Página em branco na versão original impressa.

CRIAÇÃO E MANUTENÇÃO DE SUBDIVISÕES PLANARES

Paulo Roma Cavalcanti, Paulo Cezar P. Carvalho e Luiz Fernando Martha
Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina, 110
22460 - Rio de Janeiro, RJ

ABSTRACT: This paper deals with the problem of interactively creating and maintaining a subdivision of the two-dimensional Euclidean space. The main goal is to create an editor capable of handling planar subdivisions, having a well-defined interface and creating an abstraction layer which hides the geometrical and topological problems that occur when building and maintaining such subdivisions.

1. INTRODUÇÃO

O presente trabalho trata do problema de criar e manter uma subdivisão do espaço Euclidiano bi-dimensional de forma interativa. O objetivo principal é a criação de um editor de diagramas planares constituído por uma biblioteca de funções, com uma *interface* bem definida, criando uma camada de abstração que esconde os problemas topológicos e geométricos envolvidos na criação e manutenção de uma subdivisão planar arbitrária (com arestas curvas).

A criação de uma subdivisão planar consiste em, dado um conjunto de segmentos de curvas (ou simplesmente segmentos), obter o grafo plano determinado por eles. Para tal é necessário calcular os pontos de interseção entre estes segmentos, criando *vértices* em cada um destes pontos, *arestas* entre cada par de vértices ligados por um segmento e *faces*, sempre que um conjunto de arestas formar uma área fechada.

O problema de criar uma subdivisão planar deve ser abordado sobre três aspectos: o da *correção* dos algoritmos, inseridos em um mundo ideal e supondo-se aritmética real com precisão infinita; o da *robustez* dos algoritmos, onde o problema é produzir um resultado aceitável, mesmo lidando com dados imprecisos e trabalhando com aritmética de ponto flutuante; e o da *eficiência* dos algoritmos adotados.

O EDP - Editor de Diagramas Planares é capaz de criar e manter um diagrama planar interativamente, permitindo a inserção de novos segmentos em tempo real. Para lidar com a diversidade de geometrias de segmentos necessária às aplicações, foi identificado um conjunto de funções que permitem a geração de bibliotecas de geometrias manipuláveis pelo EDP. A introdução de uma nova geometria para os segmentos pode ser feita através da criação da biblioteca correspondente.

2. ESTRUTURAS DE DADOS

Para que seja possível criar e manter de forma eficiente um diagrama planar qualquer, formado por vértices, faces e arestas, é necessário que seja utilizada uma estrutura de dados que permita obter todas as relações de adjacência entre estes entes topológicos, de forma localizada e baseada apenas na informação topológica armazenada na estrutura.

Escolheu-se uma estrutura de dados do tipo *half-edge* conforme [MANT88], juntamente com os operadores de Euler, que são usados para alterá-la, mantendo-se sempre a consistência topológica do

modelo. Esta estrutura de dados foi criada para representar a topologia de variedades (*manifolds*) bidimensionais em três dimensões. Sua utilização em duas dimensões faz com que se interprete a subdivisão planar como o mergulho do grafo que representa a topologia de uma variedade na superfície de uma esfera de raio infinito. Por isso sempre existe uma face externa (infinita) na subdivisão planar.

A estrutura *half-edge* vê cada face da subdivisão planar como limitada por um conjunto de *loops* ou ciclos, que por sua vez são sequências de *half-edges*. A cada aresta (*edge*) estão associados dois *half-edges*, orientados em sentidos opostos (fig. 1).

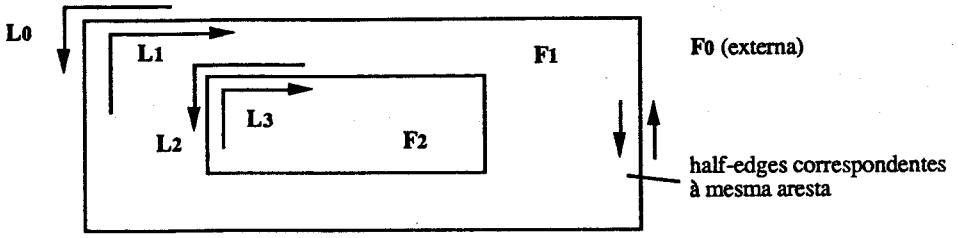

Figura 1 - Faces, loops e half-edges.

Em cada estrutura que define entes topológicos foram incluídos ponteiros para atributos geométricos ou definidos pelo usuário. No caso da aresta, os atributos geométricos são os parâmetros que definem a sua geometria; no caso da face são as coordenadas do menor retângulo que a engloba.

3. INSERÇÃO DE SEGMENTOS

A principal rotina do EDP é a que permite inserir no diagrama um novo segmento S. As extremidades inicial e final de S serão denotadas por S_i e S_f, respectivamente. Um segmento inteiramente contido em uma face (incluindo sua fronteira), será chamado de *segmento simples*.

Quando um segmento é inserido podem ocorrer quatro tipos de situação: o segmento não intercepta nenhuma aresta do diagrama; o segmento intercepta uma aresta em um vértice; o segmento intercepta uma aresta num ponto que não está identificado com nenhum vértice; e o segmento possui um ponto em comum com alguma aresta. O segmento pode interceptar várias arestas diferentes, e cada aresta pode ser interceptada em vários pontos. No entanto exige-se que o segmento seja uma curva simples (sem auto-interseção).

Para inserção de segmentos, é necessário descobrir quais arestas do diagrama são cortadas e as coordenadas de cada ponto de interseção. Isto pode ser feito por partes, pois se o vetor T_i (tangente a S em S_i) está contido em uma face F_1, a primeira interseção, a partir de S_i, ocorrerá obrigatoriamente com uma aresta pertencente a F_1.

Primeiro deve-se testar se S intercepta cada aresta de F_1, retornando, neste caso, a interseção mais próxima de S_i. Caso a interseção não coincida com um vértice do diagrama, a aresta cortada por S terá que ser dividida topologicamente. Desta forma ter-se-á sempre um vértice V_i no ponto de interseção.

O segmento S deve ser quebrado em duas partes, uma de S_i até V_i (S_1) e a outra de V_i até S_f (S_2). S_{1f} e S_{2i} passam a estar associados a V_i. S_1 é um segmento simples por construção e o processo deve ser repetido para S_2, até que S_2 também seja um segmento simples (fig.2).

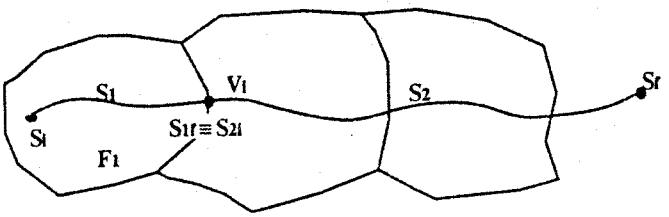

Figura 2- Decomposição de um segmento em segmentos simples.

4. INSERÇÃO DE SEGMENTOS SIMPLES

Suponha que se deseja inserir um segmento simples com extremidades S_i e S_f. Suponha também que todas as faces do diagrama estão orientadas no sentido horário, isto é, a fronteira externa é orientada no sentido horário e as internas (se houver alguma) no sentido anti-horário. Por uma questão de uniformidade será considerado que a face externa (infinita) não possui fronteira externa.

O primeiro passo para incluir S é verificar se S_i coincide geometricamente com algum vértice do diagrama para que possa ser associado a ele. Caso não exista tal vértice, S_i pode estar sobre uma aresta, que deverá ser dividida (surgindo um novo vértice), ou no interior de uma face, devendo neste caso ser criado um loop com um único vértice nesta face. Em ambos os casos, o vértice será associado a S_i.

Topologicamente podem ocorrer três situações como consequência da inclusão de um segmento simples: ser fechada uma nova face, caso S_f toque uma aresta pertencente ao mesmo loop que contém S_i (fig.3a); ser eliminado um loop, caso S_f toque uma aresta pertencente a um loop diferente daquele que contém Si (fig.3b); ser simplesmente criada uma aresta, caso Sf não toque nenhuma aresta do diagrama (fig.3c). Cada um destes casos implica no uso do operador de Euler apropriado [MANT 88].

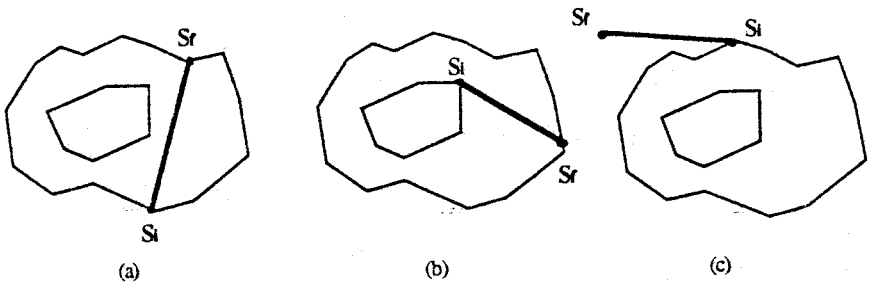

Figura 3 - Casos possíveis na inclusão de um segmento simples

Para determinar em qual face o segmento S se encontra, será considerado o vetor tangente T_i a S em S_i. Existem dois casos triviais: quando S_i está associado a um vértice isolado (que não pertence a nenhuma aresta), ou quando pertence a uma única aresta. Em ambos os casos S pertence à face que contém S_i.

Se S_i pertence a mais de uma aresta, então S está contido na face delimitada por duas arestas (que podem ter sido criadas pela divisão de uma aresta quando o vértice S_i foi criado), cujas tangentes em S_i são dois vetores u e v, um precedendo e o outro sucedendo T_i no sentido anti-horário (fig.4). A determinação

destes vetores permite estabelecer a face a qual o segmento S pertence, além de fornecerem também os argumentos para os operadores de Euler a serem empregados.

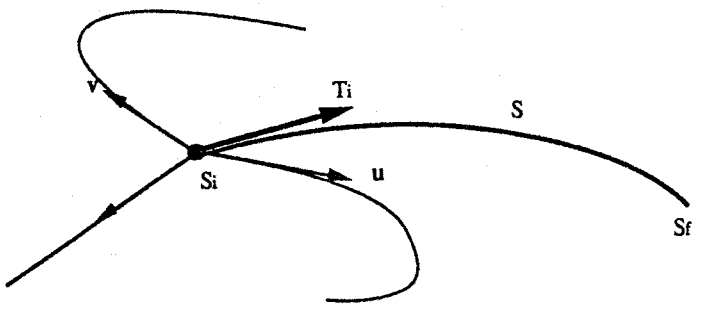

Figura 4 - Determinação da face que contém T_i.

Quando uma nova face é formada, ela é sempre criada a partir de alguma outra face do diagrama, chamada de Fo. A nova face pode conter inteiramente alguns loops de Fo, que são as suas fronteiras internas. Deve-se determinar quais loops de Fo passam a pertencer à nova face. Isto é feito varrendo-se a lista de loops de Fo (com exceção daquele que forma sua fronteira externa e daquele que delimita a nova face) e verificando se qualquer um de seus vértices está dentro da área delimitada pela fronteira externa da nova face. Caso esteja, ele será movido para a nova face (isto é, passará a pertencer a ela, fig.5).

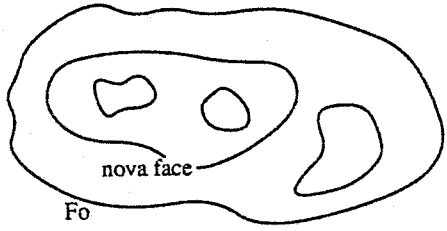

Figura 5 - Distribuição de loops.

5. CÁLCULO DE INTERSEÇÕES

Um dos algoritmos mais importantes na construção de um diagrama planar é o que calcula as interseções de um segmento com as arestas do diagrama. Basicamente existem três casos a considerar: uma das extremidades do segmento toca uma aresta; o segmento intercepta uma aresta; ou uma parte do segmento coincide com uma aresta. Para obter algoritmos robustos não basta tratar do problema matemático de encontrar a interseção de duas curvas, devido a dois problemas adicionais: o erro introduzido no cálculo dos pontos de interseção quando se utiliza aritmética de ponto flutuante; e a imprecisão inerente a todo processo interativo, quando o usuário não deseja (ou não consegue) especificar posições geométricas com precisão absoluta. Para lidar com estes dois problemas, a fim de obter uma implementação robusta, é necessário introduzir o conceito de tolerância [HOFF89].

O mecanismo utilizado considera que o diagrama foi gerado corretamente e que o novo segmento deve se adaptar a ele. Cada vértice e cada aresta possui uma campo de atração cuja abrangência depende de uma dada tolerância. Este campo de atração fornece espessura às arestas.

No EDP, quando um segmento é criado interativamente, ele sofre um pré-processamento. Se uma de suas extremidades cair dentro do campo de atração de um vértice, suas coordenadas passam a ter os valores das coordenadas do vértice. Caso caia dentro do campo de atração de uma aresta, suas coordenadas passam a ter os valores das coordenadas do ponto determinado pela projeção da extremidade sobre a aresta (o usuário já vê o segmento modificado). Em um segundo passo, o segmento pode ser dividido em dois, caso o seu interior seja atraído por um vértice.

É vital que a implementação do campo de atração seja feita consistentemente. Caso contrário, a alteração na geometria do segmento pode fazer com que um segmento que não interceptaria nenhuma aresta passe a interceptar, dependendo da configuração já existente e do valor da tolerância utilizada.

6. ESTIMATIVA PARA O TEMPO DE INCLUSÃO DE UM NOVO SEGMENTO

O primeiro passo no processo de inclusão de um novo segmento, que consiste em associar a extremidade inicial do segmento a um vértice já existente (ou, caso não exista tal vértice, criando-o primeiro), é o mais demorado. Uma vez associado o vértice inicial, o tempo de inclusão é proporcional ao número de arestas de cada face cortada.

O primeiro passo implica em determinar qual vértice, qual aresta ou qual face contém S_i. O algoritmo implementado para verificar a qual face um dado ponto pertence percorre a lista das faces, verificando se o ponto se encontra dentro da área F_{ext}, delimitada pela fronteira externa da face corrente F_c (inicialmente a primeira face da lista). Caso esteja, só precisam ser consideradas, daí para frente, as faces que estejam completamente contidas em F_c. Para verificar isto, basta tomar um vértice qualquer, de cada face subsequente, e verificar se ele está contido em F_{ext}. Se uma face que passou no teste anterior também contiver o ponto, esta será a nova face corrente F_c. O processo termina quando se tiver chegado ao final da lista das faces. F_c indica então qual face contém o ponto.

O processo descrito acima pode ser acelerado utilizando o retângulo que engloba cada face. De todo modo, obtém-se um algoritmo cujo tempo de execução é proporcional ao número de elementos topológicos (vértices, arestas e faces) do diagrama. Como o número de faces(F), vértices (V) e arestas (A) em uma divisão planar estão relacionados linearmente pela equação de Euler (V - A + F = 2), a complexidade do algoritmo pode ser expressa por O(A).

Existem, na literatura, algoritmos que permitem a inserção de um novo segmento simples em tempo ($O(\log^2 A)$) [MEHL84]. Tais algoritmos, no entanto, exigem estruturas de dados mais sofisticadas, que serão implementadas em um futuro trabalho.

7. ARMAZENAMENTO DE UM DIAGRAMA PLANAR

Uma vez criado um diagrama planar, é natural que seja necessário salvá-lo para futuras atualizações ou utilizações. O grande problema nesta tarefa é que uma estrutura de dados topológica é implementada tipicamente com o emprego de ponteiros.

Uma forma elegante de armazenar a estrutura em disco é através do uso do algoritmo de inversão [MANT88]. Este algoritmo desfaz completamente a estrutura, eliminando todas as suas arestas e todos os seus vértices isolados. Em disco são gravados, então, os operadores de Euler inversos daqueles utilizados na destruição da estrutura. A execução destes operadores na ordem inversa recria o diagrama original. Deve-se notar, porém, que, para a completa recriação do diagrama original, é necessário salvar também os atributos geométricos e os atributos do usuário associados a cada ente topológico.

Como os operadores inversos aos que destroem a estrutura devem ser executados na ordem inversa, é aconselhável armazená-los já de acordo com esta ordenação, ao invés de ler o arquivo de trás para frente. Para tal, é utilizada uma lista encadeada com um nó para cada operador inverso. Cada rotina que implementa cada operador deve incluir um nó na lista, com a descrição do operador inverso e seus respectivos parâmetros, sempre que estiver sendo executado o algoritmo de inversão.

O processo se resume, então, em percorrer a lista na ordem apropriada, gravando o registro de descrição do operador e os registros de atributos geométricos e do usuário, se os respectivos ponteiros não forem nulos. Para recriar o diagrama basta ler o arquivo, executando cada operador e ligando os entes topológicos criados aos seus atributos.

Com este processo é possível armazenar todas as informações presentes em qualquer diagrama em um arquivo seqüencial. O único aspecto inconveniente é que o diagrama é destruído, devendo ser reconstruído caso seja necessário reutilizá-lo após o salvamento.

8. O EDITOR DE DIAGRAMAS PLANARES (EDP).

O EDP tem por objetivo permitir a construção interativa de qualquer diagrama planar. Ele possui funções para: adicionar novos segmentos; eliminar arestas ; destruir um diagrama; salvar e recuperar um diagrama; obter informações sobre os entes topológicos de um diagrama (seus identificadores e valores dos atributos do usuário); selecionar a geometria dos segmentos; ampliar partes do diagrama (zoom).

O EDP foi escrito em C, utiliza o sistema gráfico GKS/puc, o sistema de *interface* com usuário IntGraf, e o executável ocupa cerca de 260 Kbytes. Em ambientes DOS 4.0, com 640 Kbytes de memória, é possível criar um diagrama com cerca de 600 faces e 600 vértices (com arestas poligonais).

8.1 - Arquitetura do EDP.

A metodologia utilizada na implementação do EDP foi a de construir camadas de *software* com funcionalidades bem definidas: uma camada responsável pelo tratamento da topologia, através dos operadores de Euler (esta camada desconhece completamente que tipo de geometria está sendo utilizada); uma camada responsável pela manipulação de entidades geométricas (para a implementação desta camada foi definido um conjunto de funções responsáveis pelo tratamento de cada geometria específica); e por fim uma camada responsável pela interação com o usuário .

O manipulador de geometria oferece três funções básicas e três funções de apoio:

```
insert_edge(segmento,tipo,var vértice)   find_vertex(face,ponto)
move_cur_vert(ponto,var vértice)         find_edge(face,var ponto,var par)
delete_edge(aresta,indicador )           find_owner_face(ponto)
```

A função **insert_edge** inclui um segmento, com origem em um vértice já existente; `tipo` indica se o segmento deve ser rejeitado quando ele interceptar uma aresta do diagrama. O parâmetro `vértice` é atualizado a cada segmento inserido.

A função **move_cur_vert** faz o parâmetro `vértice` apontar para um vértice coincidente geometricamente com o ponto fornecido ou, caso não exista tal vértice, cria um novo vértice no ponto especificado.

A função **delete_edge** elimina uma dada aresta; `indicador` indica se deve ser efetuada a união dos retângulos que delimitam as faces que a compartilham, caso a aresta pertença a mais de uma face (pois uma delas desaparecerá). O único caso em que esta união não deve ser feita é durante o algoritmo de inversão.

A função **find_vertex** retorna um ponteiro para o vértice, pertencente a uma dada face que coincide geometricamente com um dado ponto. Se face for NULL, procura em todas as faces. Retorna NULL se não existir tal vértice.

A função **find_edge** retorna um ponteiro para a aresta, pertencente a uma dada face, que contenha um dado ponto. Se face for NULL, procura em todas as faces. Retorna NULL se não existir tal aresta; `ponto` é atualizado para indicar as coordenadas do ponto da aresta para o qual ele foi atraído; `par` é uma estrutura (union) que contém informações adicionais (por exemplo o valor paramétrico da aresta no ponto).

A função **find_owner_face** retorna um ponteiro para a face do diagrama que contém um dado ponto.

8.2 Tratamento de Geometria de Forma Genérica.

O manipulador de geometria foi projetado de maneira a permitir que a forma geométrica dos segmentos seja arbitrária. Para passar a suportar uma nova geometria, deve ser escrito apenas um conjunto (pré-definido) de funções. Para isto, cada aresta tem um ponteiro para uma lista de ponteiros para função, ou seja, elas trazem consigo a indicação de que rotinas específicas implementam cada função capaz de manipular a sua geometria.

Para capacitar o manipulador de geometria a tratar um determinado tipo de geometria, as seguintes funções devem ser escritas.

1) Dada uma aresta, retornar o tipo da sua geometria.
2) Dado um par de arestas, A_1 e A_2, que foram produzidas como resultado da divisão topológica de uma aresta A, distribuir os atributos geométricos de A entre A_1 e A_2.
3) Dado um par de arestas, A_1 e A_2, que tinham a mesma geometria, eram incidentes a um mesmo vértice e que foram unidas topologicamente formando uma única aresta A, juntar os atributos geométricos de A_1 e A_2 para formarem o atributo geométrico de A.
4) Dada uma aresta A, encontrar o vetor tangente a A, pelo menos nos dois pontos coincidentes geometricamente com os seus vértices.
5) Dada uma aresta A, retornar uma medida da distância de uma de suas extremidades a qualquer um de seus pontos.
6) Dadas duas arestas, A_1 e A_2, determinar se elas se interceptam e, em caso afirmativo, retornar o ponto de interseção (caso exista mais de um) mais próximo do vértice inicial de A_1.
7) Dada uma aresta, retornar as coordenadas de qualquer um de seus pontos que não seja coincidente, geometricamente, com a posição do seu vértice inicial.
8) Dada uma aresta e um ponto de referência R, calcular a área delimitada por dois segmentos de reta, com origem em R e fim em cada vértice da aresta, e a própria aresta.
9) Dada uma aresta A, retornar um ponteiro para uma área de memória que contém as coordenadas referentes à discretização de A, em um certo número de pontos.

10) Dada uma aresta, desenhá-la.
11) Dado um ponteiro para os atributos geométricos, liberar a área de memória correspondente.
12) Dado um arquivo e um ponteiro para os atributos geométricos, gravar todos os atributos e uma indicação de quantos registros foram gravados.
13) Dado um arquivo e a indicação de quantos registros devem ser lidos, ler todos os atributos geométricos e retornar um ponteiro para a área que os contém.

Na realidade, o uso da função número 6 é evitado, pois ela compromete a modularidade do sistema. Acrescentar uma nova geometria teria que levar em conta a interação com as já existentes. Para lidar com interseção, a abordagem escolhida foi a de discretizar as arestas (através da função número 9).

A função número 7 é necessária para o tratamento de auto-ciclos.

A função número 8 serve para determinar a orientação de um loop. Calculando a área (com sinal) de cada uma de suas arestas em relação a um ponto de referência fixo, por exemplo, um de seus vértices, descobre-se se ele está orientado no sentido horário ou anti-horário, de acordo com o sinal da área . Um método similar pode ser utilizado para determinar se um dado ponto está dentro ou fora de uma face.

8.3 Tratamento de Atributos do Usuário.

Existem funções, no manipulador de geometria, encarregadas de inicializar, com valores *defaults*, os atributos do usuário de cada ente topológico criado. Elas alocam uma área de memória para os atributos, e fazem com que o ponteiro para atributo do usuário do ente criado contenha o seu endereço.

Quando uma aresta é dividida topologicamente, a situação é um pouco diferente, pois, em geral, deseja-se que a nova aresta possua atributos idênticos ao da aresta original. Para este caso, existe uma função encarregada de alocar uma área de memória para os atributos da nova aresta e preenchê-la com valores idênticos aos da aresta original.

Há, ainda, funções análogas às funções 11, 12 e 13, capazes de manipular atributos do usuário.

9. ARESTAS POLIGONAIS

Embora o EDP tenha sido projetado para lidar com arestas quaisquer, um caso particular importante de geometria de arestas é aquele dado por linhas poligonais, pois este é o tipo de geometria mais característico de aplicações ligadas a área de mapeamento geográfico e geológico. Além disso, qualquer tipo de curva pode ser aproximada por uma poligonal.

Uma poligonal é um conjunto (finito) conexo, ordenado e sem duplicatas de segmentos de reta. Ela fica perfeitamente caracterizada dada a sequência ordenada dos pontos pertencentes a dois segmentos (pontos intermediários) mais os seus pontos inicial e final (chamados de extremidades). Para armazenar uma sequência ordenada de pontos, basta uma estrutura (chamada Lista_p) que contenha o número de pontos e um ponteiro para uma lista com as coordenadas dos pontos.

No EDP, as extremidades de uma poligonal estão sempre associadas a dois vértices de uma aresta (sendo necessário distinguir o vértice inicial do vértice final da aresta). Os pontos intermediários, no entanto, são armazenados separadamente. O ponteiro para o atributo geométrico da aresta aponta para a Lista_p, com os seus pontos intermediários ordenados do vértice inicial para o final. No caso de uma aresta reta, este ponteiro é nulo.

Cada ponto intermediário de uma aresta poligonal pode ser imaginado como um falso vértice, que também possui um campo de atração para efeito de tolerância. O algoritmo que calcula o ponto de interseção

entre duas poligonais verifica se cada segmento de reta da primeira poligonal intercepta algum segmento da segunda. Caso o ponto de interseção caia dentro do campo de atração de um vértice (ou falso vértice) de qualquer uma das duas arestas, são devolvidas as coordenadas do vértice como sendo as coordenadas do ponto de interseção.

Para garantir que, ao ser atraída, a poligonal não passa a interceptar uma aresta cuja interseção não foi detectada, considera-se que apenas um segmento de reta da poligonal é modificado por vez, sempre alterando o seu ponto final e mantendo o ponto inicial fixo. Desta forma, basta considerar a projeção dos vértices de cada aresta sobre o segmento de reta em questão e retornar a interseção mais próxima do ponto inicial.

10. APLICAÇÕES

A capacidade de criar subdivisões planares, de forma interativa ou não, possui várias aplicações práticas. Sistemas geográficos e de elementos finitos desempenham estas tarefas, porém tipicamente usando estruturas de dados mais primitivas que as usadas neste trabalho.

A principal vantagem em utilizar o método apresentado vem a ser que, ao final do processo, haverá uma estrutura de dados topológica que possibilita encontrar todas as relações de adjacência de forma eficiente. Com uma biblioteca de funções de consulta apropriada, rapidamente se extrai qualquer uma destas relações. Por exemplo, dado um vértice, quais arestas são incidentes a ele, ou dada uma face, quais faces compartilham cada uma de suas arestas. Considerando apenas faces, vértices e arestas, existem nove relações de adjacência diferentes [WEIL85]. Com estruturas de dados convencionais, algumas destas relações não podem ser encontradas de forma eficiente.

Num sistema geográfico típico, para se obter uma estrutura de dados onde algumas das relações de adjacência estão presentes de forma explícita, são necessárias três etapas: digitalizar um mapa; executar um procedimento que marque determinados pontos especiais como sendo pontos de entroncamento de segmentos e, a partir dos entroncamentos e dos segmentos, gerar as faces. Pior ainda, o mapa deve ser digitalizado sempre em um mesmo sentido e arestas que não pertençam à face externa são digitalizadas duas vezes, uma em cada sentido. Em alguns sistemas, até os pontos de entroncamento devem ser indicados em tempo de digitalização.

Com o EDP, as três etapas são executadas simultaneamente e em tempo real. Não há qualquer ordem imposta para a digitalização. Quando é fechada uma face, o EDP preenche a sua área com uma determinada cor, o que serve como *feedback* para o usuário. Ele também possui um modo de operação que recusa um segmento quando este intercepta uma aresta, evitando a geração de faces indesejadas.

Uma outra aplicação onde a utilização de estruturas de dados topológicas para subdivisões planares tem se mostrado bastante eficiente é no desenvolvimento de sistemas bi-dimensionais de elementos finitos. Wawrzynek [WAWR87] utilizou, com excelentes resultados, a estrutura *winged edge* [BAUM75] no desenvolvimento de um sistema integrado (envolvendo pré-processamento, análise numérica e pós-processamento) de elementos finitos para simulação de propagação de trincas em duas dimensões.

A utilização de estruturas de dados topológicas para subdivisões planares também possibilita um ambiente adequado para a geração interativa de malhas de elementos finitos bi-dimensionais de boa qualidade [CARV90, CAMP91].

11. CONCLUSÕES

A utilização de estruturas de dados topológicas propicia uma nova forma de trabalho que, espera-se, venha a ser incorporada nos sistemas de CAD futuros, automatizando tarefas que, nos sistemas atuais, são executadas manualmente ou ineficientemente.

A principal contribuição deste trabalho é a criação de uma biblioteca, com uma *interface* bem definida, que permite que programas de aplicação possam criar e manter diagramas planares arbitrários. Além disso, é possível adicionar novas geometrias para as arestas dos diagramas, bastando que se escreva uma biblioteca com funções pré-definidas capazes de tratar a nova geometria.

A extensão natural deste trabalho, ora em fase de desnvolvimento, envolve a criação de subdivisões volumétricas do espaço Euclidiano tri-dimensional. Para isto, é necessário adotar estruturas de dados *non-manifold* [WEIL86]. Aplicações típicas estariam nas áreas de geologia e elementos finitos [MART89].

Bibliografia.

[BAUM75] Baumgart, B. G. - *A Polihedron Representation for Computer Vision.* AFIPS Proc., Vol 44, pp 589-596, 1975.

[CAMP91] Campos, Jorge Alberto de Prado - *Geração de Malhas de Elementos Finitos Bi-dimensionais Baseada em Estruturas de Dados Topológicas.* Tese de Mestrado, PUC-Rio, Dep. Eng. Civil. Março 1991.

[CARV90] Carvalho, P. C. P., Gattass, M. & Martha, L.F. - *A Software Tool which Allows Interactive Creation of Planar Subdivisions and Applications to Educational Programs.* Proceedings of the International Conference on Computer Aided Training in Science and Technology Barcelona, July 1990. Edited by: E. Oñate et. al., Pinerage Press, 1990, pp 201-207.

[HOFF89] Hoffmann, Christoph M. - *Geometric & Solid Modeling: An Introduction.* Morgan Kaufmann Publishers, Inc - 1989.

[MANT88] Mäntylä, Martti - *Solid Modeling.* Computer Science Press 1988.

[MART89] Martha, Luiz Fernando - *Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Fracture Simulation in Three Dimensions.* Tese de Doutorado. Cornell University, Ithaca, N4, 1989.

[MEHL84] Mehlhorn, K. *Data Structures and Algorithms 3: Multi-dimensional searching and Computational Geometry.* Springer-Verlag, 1984.

[WAWR87] Wawrzynek, Paul A. & Ingraffea, Anthony R.- *An Edge-Based Data Structure for Two-Dimensional Finite Element Analisys.* Engineering with Computers, Vol. 3, pp 13-20, 1987.

[WEIL85] Weiler, Kevin - *Edge-Based Data Structures for Solid Modeling in Curved Environments.* IEEE, Computer Graphics Applications, Vol. 5, pp 21-40, 1985.

[WEIL86] Weiler, Kevin - *Topological Structures for Geometric Modeling.* Tese de Doutorado do Rensselaer Polytechnic Institute, Troy, NY. Agosto de 1986.