LOCALIZAÇÃO EXATA DE PONTOS EM MAPAS ESFÉRICOS USANDO UMA ADAPTAÇÃO DA ÁRVORE DE PARTIÇÃO BINÁRIA DO ESPAÇO

BSP Esférica

João B. Mendes,[1] Cristiano D. Ferreira,[1] and Marcus V. A. Andrade[1]

[1]*Departamento de Informática*
Universidade Federal de Viçosa, MG, Brasil
{jbm, cdalmas, marcus}@dpi.ufv.br

Abstract This paper describes an adaptation of BSP (Binary Space Partitioning) to represent the spherical surface partitioning defined by a spherical map. This structure is used in a point location algorithm, also described here. All algorithms are exact (roundoff free) based on exact computation paradigm.

Keywords: Spherical maps, point location, binary space partitioning

1. Introdução

Sistemas de Informações Geográficas (SIG) é uma importante área de aplicação da geometria computacional que aborda problemas envolvendo diversos tipos de mapas (mapas rodoviários, mapas hidrográficos, etc.) em que deseja realizar a localização de pontos, operação de sobreposição, determinação do menor caminho entre dois pontos, etc Force, 1996; Maguire et al., 1991.

Neste trabalho, o objetivo é abordar o problema de localização em mapas, em particular, nos restringindo a mapas esféricos - mapas cuja superfície da esfera é dividida em três tipos de elementos: vértices (pontos), arestas (círculos ou arcos de círculos) e faces (regiões abertas) Andrade, 1999. Vale ressaltar que nestes mapas, os círculos podem ter raios de tamanho arbitrário (i.e., podem ser máximos ou não). Veja figura 1.

A localização de pontos em mapas é um problema clássico da geometria computacional com aplicação em diversas áreas Sarnak and Tarjan, 1986; Asano, 1986; Iacono, 2001; Edelsbrunner et al., 1986. Este problema consiste, basicamente, em determinar qual elemento de um determinado mapa contém um dado ponto p. No caso particular dos mapas esféricos, este elemento pode

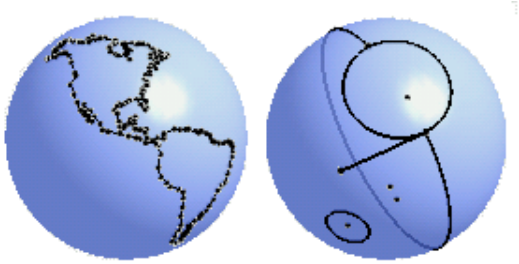

Figure 1. Exemplos de mapas esféricos.

ser um vértice, uma aresta ou uma face. Existe uma solução proposta por Andrade Andrade et al., 2002 que é baseada num processo "incremental" e possui complexidade *O(n)*, onde *n* representa o número de elementos do mapa. Uma vez que no contexto SIG's os mapas esféricos possuem entre 10^3 e 10^5 elementos e a operação de localização de um objeto (ou de vários objetos) pode ser realizada diversas vezes sobre o mesmo mapa então este algoritmo pode produzir tempo de resposta consideravelmente longo comprometendo sua utilização.

Uma questão importante a ser considerada em problemas geométricos tais como o de localização são as dificuldades geradas pelos erros de arredondamento Halperin and Shelton, 1997; Hoffmann, 1989; Yap and Dubé, 1995; Yap, 1993. A princípio, esta questão poderia ser vista como um excesso de capricho nas aplicações SIGt's, pois os seus dados são, normalmente, aproximados e as soluções aproximadas são satisfatórias em várias situações práticas. Mas, infelizmente os erros de arredondamento podem levar os programas a situações críticas impedindo que seja produzido algum resultado; isto é, o programa pode abortar.

Neste trabalho vamos apresentar um algoritmo exato, baseado no *paradigma da computação exata* Yap, 1993; Yap and Dubé, 1995, para localização de pontos em mapas esféricos. O algoritmo utiliza uma adaptação de árvores *BSP* (*Binary Space Partitioning*) com o intuito de tornar o processo de pesquisa mais eficiente, sendo esta adaptação a principal contribuição deste trabalho.

2. Representação da geometria e da topologia

Do ponto de vista geométrico, um mapa esférico é uma divisão da esfera $\mathbb{S}^2$ em três *elementos*: *vértices* (pontos), *arestas* (círculos ou arcos de círculos) e *faces* (regiões abertas). A representação da geometria e da topologia de um mapa esférico se baseia no modelo proposto por Andrade e Stolfi Andrade and Stolfi, 2001; Andrade, 1999. A seguir é apresentado um breve resumo

deste modelo que utiliza inúmeros conceitos de coordenadas homogêneas e geometria projetiva orientada Stolfi, 1991.

Pontos e círculos sobre $\mathbb{S}^2$

Um ponto $p \in \mathbb{S}^2$ é representado utilizando coordenadas homogêneas $[w, x, y, z]$. Note que, para todo ponto, $x^2 + y^2 + z^2 = w^2$.

Por definição, uma aresta de um mapa esférico pode ser um círculo completo ou um arco de círculo e, por sua vez, um *círculo orientado* na esfera corresponde à interseção entre um plano orientado e $\mathbb{S}^2$. Esta correspondência é biunívoca, isto é, para todo círculo existe um único plano que o define e vice-versa. Desta forma, um círculo pode ser representado pelos coeficientes homogêneos do plano que o define.

Dado um plano $\alpha = \langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$, o círculo c gerado pelo plano de suporte α é denotado por $c = scrc(\alpha)$ e representado por $(\!(\alpha_0, \alpha_1, \alpha_2, \alpha_3)\!)$.

Dois pontos distintos p e q sobre um círculo c dividem este círculo em duas partes conexas. Por definição, denominamos de *arco esférico* o conjunto de pontos que vão de p até o ponto q, ao longo de c, de acordo com o sentido de c. Veja figura 2. Este arco é representado por $A = (p, \overset{\frown}{c}, q)$.

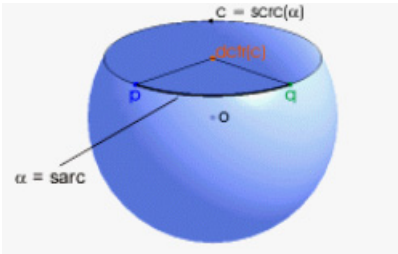

Figure 2. O arco $(p, \overset{\frown}{c}, q)$.

Como nosso objetivo é desenvolver um algoritmo exato para localização de pontos num mapa esférico então vamos nos restringir a mapas cujos vértices (pontos) e arestas (arcos) possuem coordenadas (coeficientes) homogêneas racionais e que portanto, podem ser representados exatamente utilizando números inteiros. Na prática, esta restrição não é significativa pois, como demonstrado em Andrade, 1999, qualquer ponto $p \in \mathbb{S}^2$ pode ser aproximado por um ponto q com coordenadas homogêneas inteiras, tal que a distância entre p e q seja tão pequena quanto se queira. O mesmo ocorre com as arestas (arcos) sobre $\mathbb{S}^2$.

No processamento de mapas, em diversas situações é necessário representar os pontos de interseção entre as arestas. Visto que a interseção entre duas arestas (i.e., dois círculos) racionais são pontos cujas coordenadas podem não

ser racionais então estes pontos não podem ser representados exatamente pelas suas coordenadas.

Para representar estes pontos adotamos a seguinte estratégia: dados os círculos a e b que se interceptam, sejam α e β os planos de suporte destes círculos e seja l a reta de interseção entre estes planos. Note que os pontos de interseção entre a e b correspondem aos pontos de interseção entre a reta l e a esfera $\mathbb{S}^2$. Veja Figura 3. Então, estes pontos de interseção podem ser representados pela reta de interseção entre os planos de suporte dos círculos. Visto que estes círculos (i.e., os seus planos de suporte) têm coeficientes racionais então a reta l pode ser representada exatamente utilizando *coeficientes de Plücker* Andrade, 1999; Stolfi, 1991 que, neste caso, também são racionais (equivalemente, inteiros).

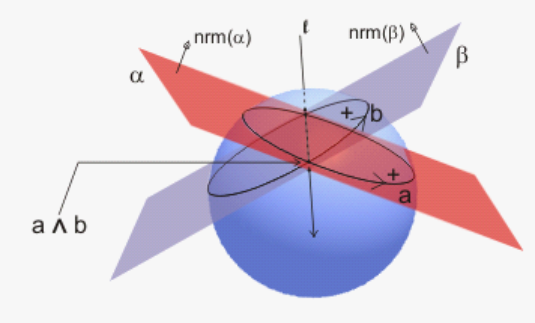

Figure 3. Interseção canônica entre dois círculos.

Dada a reta l, distinguimos três pontos sobre esta reta: $ext(l)$ e $ent(l)$ são, respectivamente, os pontos onde a reta sai e entra em $\mathbb{S}^2$ e o ponto médio entre estes dois pontos é chamado de $mid(l)$.

Como a interseção entre dois círculos pode ser composta por dois pontos, então para eliminar esta ambigüidade definimos a *interseção canônica* entre os círculos a e b como sendo o ponto $ext(l)$.

Concluindo, os pontos de interseção entre as arestas (racionais) de um mapa esférico podem ser representadas exatamente pelo seguinte conjunto de pontos: $\{ext(l) \mid l$ é uma reta racional$\}$.

Representação da topologia dos mapas esféricos

A topologia dos mapas esféricos é representada utilizando a estrutura SMC (*Spherical Maps by Corners*) Andrade, 1999 que pode ser interpretada como uma extensão da estrutura *half-edge* Mantyla, 1998; Weiler, 1986 incluindo a possibilidade de representar vértices isolados, arestas ovais (arestas sem vértices extremos) e faces com múltiplas bordas.

3. BSP e mapas esféricos

Uma estrutura muito utilizada em geometria computacional e computação gráfica para estabelecer a organização de informações espaciais é a árvore BSP (*Binary Space Partitioning*) Thibault and Naylor, 1987; Fuchs et al., 1980; Schumacker et al., 1969 que define uma partição binária do espaço. Informalmente, esta partição do espaço $\mathbb{R}^3$ pode ser descrita da seguinte forma: dada uma lista L de objetos em $\mathbb{R}^3$, seja um plano α que intercepta o interior desta região. Particione os objetos da lista L em relação ao plano α produzindo três listas L_+, L_- e L_0 que contêm os objetos (ou parte deles) que estão respectivamente do lado positivo, do lado negativo ou sobre o plano α. Cada uma destas listas é novamente particionada até que um determinado critério de parada seja alcançado. Esta operação determina uma ordem dos novos subespaços em relação ao plano divisor.

Formalmente, seja L uma lista de objetos no espaço $\mathbb{R}^n$. Uma BSP de dimensão n, denotada por BSP_n, é uma árvore onde a cada nó t é associado um hiperplano divisor $\alpha(t)$, de dimensão $\mathbb{R}^{n-1}$, e três sub-árvores $B_+(t)$, $B_-(t)$ e $B_0(t)$ que estão associadas, respectivamente, às listas L_+, L_- e L_0. Assim, dado um nó t da árvore, seja $\alpha(t)$ o hiperplano separador associado ao nó t, e seja $P(t)$ o domínio espacial associado à t. Uma BSP_n é definida da seguinte forma:

$$\text{(i)}\ \text{para o nó raiz } t,\ P(t) = \mathbb{R}^n;$$

$$\text{(ii)}\ P(B_+(t)) = P(t) \cap \alpha_+(t);$$

$$\text{(iii)}\ P(B_-(t)) = P(t) \cap \alpha_-(t);$$

$$\text{(iv)}\ P(B_0(t)) = P(t) \cap \alpha_0(t);$$

onde $\alpha_+(t)$, $\alpha_-(t)$ e $\alpha_0(t)$ são, respectivamente, os sub-espaços do lado positivo, do lado negativo e sobre o hiperplano $\alpha(t)$.

Observe que a árvore $B_0(t)$, na verdade é uma BSP_{n-1}.

Esta estratégia de partição do espaço pode ser adaptada para se estabelecer uma partição da superfície da esfera $\mathbb{S}^2$, sendo que neste caso, as folhas da árvore representam regiões dadas por $P(t) \cap \mathbb{S}^2$ onde t é um nó folha da BSP_3. Esta combinação da BSP_n com uma esfera de dimensão $\mathbb{S}^{n-1}$ será denotada por $SBSP_n$. Vale notar que certas folhas da $SBSP_n$ podem representar "regiões vazias" e outras folhas podem representar um conjunto de regiões desconexas sobre $\mathbb{S}^{n-1}$. Uma $SBSP_n$ é dita *conexa* quando cada nó folha da árvore está associado a uma única região conexa da superfície da esfera $\mathbb{S}^{n-1}$.

Visto que um mapa esférico é uma partição de $\mathbb{S}^2$ então podemos utilizar uma $SBSP_3$ para representar a partição definida pelo mapa esférico. Veja figura 4.

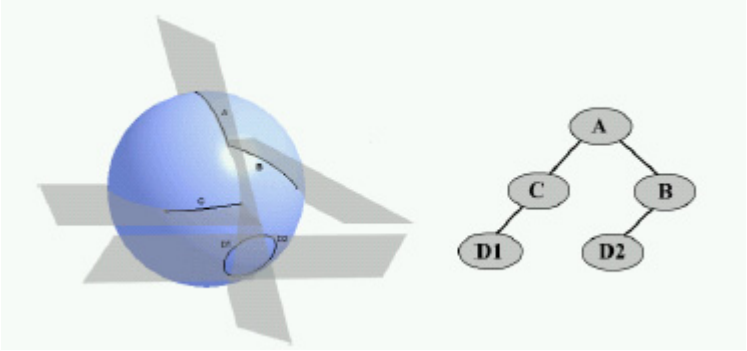

Figure 4. Partição da esfera e a árvore correspondente.

O objetivo desta associação é tornar mais eficiente o processo de localização de pontos num mapa, pois, uma vez obtida essa associação, o processo de localização de um ponto consiste basicamente em percorrer a árvore como descrito na seção 5.

Para facilitar a obtenção das relações topológicas do mapa, a $SBSP_3$ será utilizada em conjunto com a estrutura SMC que representa a topologia do mapa esférico. Desta forma, em cada folha t da $SBSP_3$, onde $P(t) \cap \mathbb{S}^2 \neq \phi$, haverá a indicação de qual elemento do mapa está associado àquela folha.

Vale ressaltar que esta representação pode levar a situações onde uma folha t da árvore esteja associada a mais de um elemento do mapa. Assim sendo, diremos que uma $SBSP_3$ é *categórica* quando, para toda folha t, a região representada por esta folha está associada a um único elemento do mapa ou quando a região é vazia.

Para o processo de localização, o ideal é utilizar uma $SBSP_3$ categórica.

4. Construção de uma $SBSP_3$

Dado um mapa esférico M, seja L a lista de vértices, arestas e faces de M. A $SBSP_3$ associada a M é construída pelo algoritmo *buildSBSPTree3* utilizando, como planos separadores, os planos de suporte dos arcos (arestas) e planos passando pelas retas que definem os vértices.

Na verdade, o algoritmo constrói uma $SBSP_3$ associada a um mapa esférico M' correspondente ao refinamento do mapa M. O refinamento M' é obtido inserindo a aresta oval associada ao plano separador utilizado em cada partição da lista L e todo elemento de M' referencia o respectivo elemento de M que o contém (que o originou). Este refinamento é utilizado para auxiliar a classificação dos elementos da lista L em relação ao plano separador.

Dada a lista L com os vértices, arestas e faces de M, seja e um elemento de L que não é uma face do mapa. Se e for uma aresta, seja α o plano de

suporte do círculo associado a e; caso contrário, e é um vértice isolado e, neste caso, α é um plano passando pela reta que define e. Insira em M' a oval definida pelo plano α - inicialmente, M' é um mapa trivial (com apenas uma face). O próximo passo é classificar os vértices e as arestas de L em relação ao plano α inserindo esses elementos (ou parte deles) nas respectivas listas L_+, L_- e L_0. Vale notar que as arestas que são interceptadas pelo plano α são particionadas e divididas em três partes que são distribuídas pelas listas L_+, L_- e L_0. Num segundo passo, as faces de L são classificadas em relação ao plano α. A inserção da oval referente ao plano α no mapa M' é realizada utilizando uma variação do algoritmo de inserção de uma aresta num mapa descrito em Andrade Andrade, 1999.

Este processo é repetido recursivamente nas listas L_+, L_- e, por último, L_0 enquanto cada lista possuir mais de um elemento (vértices ou arestas). Quando houver apenas faces na lista então é criado um nó folha t na árvore e é associado a este nó a lista de faces do mapa M' que possuem interseção com a região $P(t) \cap \mathbb{S}^2$. Assim, ao final do processamento, todos os elementos de M' estarão associados a pelo menos uma folha da árvore.

É importante ressaltar que a versão do algoritmo descrita acima não gera uma $SBSP_3$ categórica, mas no entanto, na seção 6 é descrita uma estratégia para estender o algoritmo de modo que seja obtida uma $SBSP_3^*$ categórica.

Para classificar os elementos (vértices e arestas) de L em relação ao plano separador é utilizado o algoritmo *ClassifyElement* que recebe como parâmetros o elemento e o plano e retorna a posição do elemento em relação a este plano.

A classificação de um vértice (ponto) consiste em determinar, de maneira exata, a posição do ponto em relação ao plano. Esta operação é realizada pela função $SideOf$, descrita em Andrade Andrade, 1999.

A classificação de uma aresta é dividida em duas partes dependendo da aresta ser uma oval ou um arco. No caso de uma oval s, seja a o círculo de suporte da oval e seja b o círculo gerado pelo plano α. Neste caso, se os círculos a e b não se interceptam então basta gerar um ponto sobre o círculo a e classificar este ponto em relação a α utilizando a função *SideOf*. Caso contrário, se houver interseção, então é retornada uma indicação de que a aresta é interceptada pelo plano de partição e precisa ser particionada.

Por outro lado, se a aresta é o arco $A = (p, \overset{\frown}{c}, q)$ então seja b o círculo gerado pelo plano α. Primeiramente, é verificado se os círculos b e c se interceptam e se esta interseção pertence ao arco A. Caso isto não se verifique, novamente basta retornar a posição de um dos extremos de A em relação α. No entanto, se existe interseção e ela pertence a A então o algoritmo também retorna a indicação de que a aresta é interceptada pelo plano.

A partição de uma aresta em relação ao plano separador determinada no processo descrito acima é realizada pelo algoritmo *SplitArc*. Este algoritmo recebe como parâmetros a aresta $A = (p_0, \overset{\frown}{c}, p_1)$, o plano particionador α e

as listas L_+, L_- e L_0; daí, é calculado o ponto q correspondente à interseção entre c e o círculo $scrc(\alpha)$ e os arcos $(p_0, \overset{\frown}{c}, q)$ e $(q, \overset{\frown}{c}, p_1)$ são inseridos nas respectivas listas L_+ e L_- e o ponto q é inserido em L_0. É importante ressaltar que neste processo é estabelecida (e armazenada) uma associação entre os elementos criados e o elemento original do mapa M para que o processo de localização possa indicar o (efetivo) elemento que contém o ponto.

Para classificar as faces de L em relação ao plano α, inicialmente são obtidas aquelas faces que possuem em sua fronteira pelo menos uma aresta situada sobre α e que estão no lado positivo deste plano. Estas faces devm ser inseridas na lista L_+. Para obter essas faces, seja c o círculo gerado pelo plano separador α; então, percorra os elementos (vértices e arestas) de M' que estão sobre o círculo c e, para cada um destes elementos, tome a face do mapa adjacente ao respectivo elemento e a insira na lista L_+. Para obter as faces situadas no lado negativo de α, repita esse procedimento percorrendo os elementos no sentido inverso de c inserindo as faces obtidas na lista L_-. Finalmente, verifique se existe alguma face em L que ainda não foi classificada. Se houver, isto significa que a face não é interceptada por α e, portanto, para classificá-la basta gerar um ponto sobre sua fronteira e classificá-lo em relação a α.

A definição da $SBSP_3$ estabelece que cada nodo interno gera três subárvores $B_+(t)$, $B_-(t)$ e $B_0(t)$, sendo que $B_0(t)$ é uma $SBSP_2$, ou seja, é a representação da partição binária do plano de suporte de um círculo. Na verdade, o objetivo da $SBSP_2$ é classificar os elementos do mapa que estão sobre um plano de partição. A construção da $SBSP_2$ segue uma estratégia semelhante à construção da $SBSP_3$, porém, neste caso é estabelecida uma partição do plano (do círculo de suporte) utilizando segmentos de retas, sendo que estas retas correspondem às retas de suporte utilizadas para representar os pontos do

5. Algoritmo de localização

Uma vez construída a $SBSP_3$, o processo de localização de um ponto p no mapa pode ser realizado de maneira relativamente simples, comparando a posição de p em relação ao plano separador em cada nó t da árvore e prosseguindo a pesquisa em $B_+(t)$, $B_-(t)$ ou $B_0(t)$. Este processo é repetido até que um nó folha t seja alcançado. Daí, podemos concluir que o ponto está localizado na região $P(t) \cap \mathbb{S}^2$ e portanto, basta determinar qual elemento do mapa esférico está associado àquela região.

Caso a $SBSP_3$ seja categórica então ao se alcançar uma folha t o processo de localização se encerra e basta retornar o elemento do mapa associado ao nó t. No entanto, considerando a versão atual do algoritmo que gera uma árvore que não necessariamente é categórica, há uma importante questão a ser contornada: a folha pode estar associada a mais de uma face. Neste caso, o algoritmo de localização tem que determinar qual destas faces contém o ponto.

Para resolver esta questão, elaboramos um algoritmo *InsideFace* que dado um ponto p e uma face f do mapa M' retorna *true* ou *false* indicando respectivamente se o ponto está dentro ou fora da face. Este algoritmo utiliza uma combinação dos algoritmos *ClosestFaceExit* e *WhereTo*, definidos em Andrade et al., 2002; Andrade, 1999, e consiste em: dado o ponto p e uma face f, seja q um ponto na borda desta face e seja c um arco ligando p a q; utilize uma variação do algoritmo *ClosestFaceExit* para determinar o ponto de interseção entre o arco c e as bordas de f que é o mais próximo de p no sentido definido pelo caminho que vai de p para q. Agora, utilize o algoritmo *WhereTo* neste ponto de interseção considerando o caminho no sentido de q para p para determinar se, neste ponto, o caminho está "entrandot'ť ou "saindot'ť da face f; daí, podemos concluir que p está respectivamente dentro ou fora da face f.

6. Trabalhos futuros

Como citado anteriormente, a versão atual do algoritmo gera uma $SBSP_3$ que não é categórica. O nosso próximo objetivo é definir um método para tornar a árvore categórica. Na verdade, já temos uma proposta para resolver esta questão e resta mostrar que esta estratégia é suficiente. A idéia básica consiste em inserir planos auxiliares para separar as faces associadas a uma mesma folha da árvore. Em outras palavras, suponha que uma folha da $SBSP_3$ esteja associada a uma lista de faces $f_1, \cdots, f_n$. Dada uma face f nesta lista, sejam $v_1, \cdots, v_m$ os vértices de uma borda da face f. Então, tome três vértices v_i, v_j e v_k e sejam l_i, l_j e l_k retas tais que $v_i = ext(l_i)$, $v_j = ext(l_j)$ e $v_k = ext(l_k)$. Gere um plano passando pelos pontos $mid(l_i)$, $mid(l_j)$ e $mid(l_k)$ - este pontos são todos racionais e portanto o plano também o é. A inserção deste plano na $SBSP_3$ pode isolar a face f ou então particionar esta face (e outras) faces da lista.

O objetivo é mostrar que a repetição deste processo, em algum momento, isola a face f das outras faces da lista. Portanto, repetindo-se este processo para as outras faces que ainda não estão isoladas permite a geração de uma $SBSP_3$ categórica.

Além disso, pretendemos também realizar um conjunto de testes para comparar a eficiência deste algoritmo em relação ao algoritmo incremental.

Agradecimentos

Este trabalho foi parcialmente financiado com recursos do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, entidade governamental brasileira promotora do desenvolvimento científico e tecnológico através do projeto TerraUFV número 552435/2002-3.

Andrade, M. (1999). *Representação e Manipulação Exatas de Mapas Esféricos*. PhD thesis, Instituto de Computação - UNICAMP. (in Portuguese).

Andrade, M., Barros, W., and Stolfi, J. (2002). An exact algorithm for point location on spherical maps. *IV Simpósio Brasileiro de Geoinformática*, pages 99–107.

Andrade, M. and Stolfi, J. (2001). Exact algorithms for circles on the sphere. *International Journal of Computational Geometry e Applications*, 11(3):267–290.

Asano, T. (1986). A new point-location algorithm and its practical efficiency:comparison with existing algorithms. *ACM Transactions on Graphics (TOG)*, 3:86–109.

Edelsbrunner, H., Guibas, L., and Stolfi, J. (1986). Optimal point location in a monotone subdivision. *SIAM Jornal on Computing*, 15:317–340.

Force, C. I. T. (1996). Applications challenges to computational geometry. Technical Report TR-521-96, Princeton University.

Fuchs, H., Kedem, Z., and Naylor, B. (1980). On visible surface generation by a priori tree structures. *7Th. annual conference on Computer Graphics and interactive techniques*, pages 124–133.

Halperin, D. and Shelton, C. (1997). A perturbation scheme for spherical arrangements with application to molecular modeling. *13Th. Annual Symposium on Computational Geometry*, pages 183–192.

Hoffmann, C. (1989). The problems of accuracy and robustness in geometric computation. *IEEE Computer*, 3(22):31–42.

Iacono, J. (2001). Optimal planar point location. *12th annual ACM-SIAM symposium on Discrete algorithms*, pages 340–341.

Maguire, D., Goodchild, M., and Rhind, D. (1991). *Geographical Information Systems - Principles and applications*, volume 2. John Wiley & Sons.

Mantyla, M. (1998). *An Introdution to Solid Modeling*. Computer Science.

Sarnak, N. and Tarjan, R. (1986). Planar point location using search trees. *Communications of the ACM*, 29:669–679.

Schumacker, R., R. Brand, M. G., and Sharp, W. (1969). Study for applying computer-generated images to visual simulation. Technical Report AFHRL-TR-69-14, US Air Force Human Resources Laboratory, Brooks Air Force Base, San Antonio (USA).

Stolfi, J. (1991). *Oriented Projetive Geometry - A framework for geometric computations*. Academic Press.

Thibault, W. and Naylor, B. (1987). Set operations on polyhedra using binary space partitioning trees. *Comput. Graph.*, 21(4):153–162. Proc. SIGGRAPH '87.

Weiler, K. (1986). *Topological Structures for Geometric Modeling*. PhD thesis, Rensselaer Polytechnic Institute.

Yap, C. (1993). Towards exact geometric computation. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 405–419.

Yap, C. and Dubé, T. (1995). The exact computation paradigm. In Du, D.-Z., , and Hwang, F. K., editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 452–492. World Scientific Press, Singapore, 2nd edition.