IsSuP:
Sistema para Cálculo e Visualização de Superfícies Isopotenciais

Artemis Moroni

IsSuP:
Sistema para Cálculo e Visualização de Superfícies Isopotenciais

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pela Sra. Artemis Moroni e aprovada pela Comissão Julgadora.

Campinas, 1 de agosto de 1996

C. L. Lucchesi

Prof. Cláudio Leonardo Lucchesi
Orientador

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

IsSuP:
Sistema para Cálculo e Visualização de Superfícies Isopotenciais[1]

Artemis Moroni[2]

Instituto de Computação
UNICAMP

Banca Examinadora:

- Cláudio Leonardo Lucchesi (Orientador)[3]
- Clésio Luiz Tozzi (Suplente)[4]
- Léo Pini Magalhães[4]
- Tomasz Kowaltowski[3]

---

[1]Dissertação apresentada ao Instituto de Computação como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

[2]A autora é Bacharel em Ciência da Computação pela UNICAMP e pesquisadora no CTI.

[3]Professor do Instituto de Computação - UNICAMP.

[4]Professor da Faculdade de Engenharia Elétrica - UNICAMP.

Tese de Mestrado defendida e aprovada em 01 de agosto de 1996 pela Banca Examinadora composta pelos Professores Doutores

Prof. Dr. Léo Pini Magalhães

Prof. Dr. Tomasz Kowaltowski

Prof. Dr. Cláudio Leonardo Lucchesi

A meu marido Ívan e a meus filhos Andréia e Ian

Agradecimentos

Ao CTI, pelo apoio recebido.

A todos os colegas e principalmente aos professores do IC e da FEE, pela sua inocência e sua boa fé em aceitar este trabalho.

Aos Professores José Luiz Boldrini, do IME, e Jorge Stolfi do IC por terem dissolvido algumas dúvidas e acrescentado várias outras.

Ao Professor Roberto Lotufo, ao Renato Luppi e ao Alexandre Xavier, que me ajudaram na geração das imagens.

Aos administradores de sistema Alexandre Duarte, Gilberto Ohashi, Antonio Figueiredo e Roland Scialom que pacienciosamente me suportaram.

Ao desenhista Roberto de Oliveira e à secretária Maria José Pirola, meticulosos e carinhosos.

Em particular, ao Luiz Otávio Queiroz, por ter ouvido todas as minhas queixas e reclamações e por ter sido o único que me cobrou o término deste trabalho.

À minha família e a meus filhos, que sempre estiveram tão presentes.

A meus pais, ícones dos meus lados conflitantes.

Aos Professores Cláudio Leonardo Lucchesi, que sempre me aconselhou a ir com calma, e Léo Pini Magalhães, que sempre me apressou.

E a um fantástico cubo com um canto amarelo, brinde da IBM, que durante semanas eu girei em minhas mãos.

Depois de muitas observações da natureza que fiz com meus cinco sentidos e mais o telescópio, disse eu um dia a Sidra: "Zombam de nós; a natureza não existe, tudo é arte. É por uma arte admirável que todos os planetas dançam regularmente em torno do sol, enquanto o sol gira sobre si mesmo. É claro que alguém, tão sábio como a Sociedade Real de Londres, arranjou as coisas de modo que o quadrado das revoluções de cada planeta seja sempre proporcional à raiz cúbica de sua distância ao seu centro; e é preciso ser feiticeiro para adivinhá-lo.

Voltaire

(Os ouvidos do Conde de Chesterfield)

Resumo

A visualização e a simulação de processos físicos e o modelamento de fenômenos naturais vêm ocupando um espaço cada vez maior na ciência de computação. Abordagens recentes, que provêem soluções para essas aplicações são geralmente agrupadas sob o termo *modelamento baseado em sistemas físicos*. Neste trabalho, é descrita uma aplicação de sistemas de partículas para o modelamento de superfícies isopotenciais 3D geradas por cargas elétricas, através do Método Multipólos Rápido (MMR). O método sugere uma abordagem para acelerar a velocidade do cálculo das interações entre as partículas para uma classe específica de problemas, onde as interações são definidas usando funções potenciais.

Para representar o espaço que contém as partículas elétricas é usada a estrutura de dados *octree*. As relações de vizinhança na octree ocupam um importante papel no MMR. O sistema resultante IsSuP - Isopotential Surfaces using Particles, um sistema flexível para a avaliação e o modelamento das superfícies isopotenciais 3D, é descrito. Para visualizar as superfícies é utilizada a biblioteca V3DTOOLS para Visualização e Processamento de Volumes do sistema Khoros. Resultados numéricos e imagens são apresentadas.

Palavras chave: modelamento, sistemas de partículas, visualização científica, octree.

Abstract

The visualization and simulation of physical processes and the modelling of natural phenomena is playing an increasing role in computer science. Recent approaches, which provide solutions for these applications, are generally referenced under the term *physically based modelling*. An application of particle systems for modelling 3D isopotential surfaces generated by electrical charges using the Fast Multipole Method (FMM) is described. This method suggests an approach to accelerate the evaluation speed of the interactions among the particles for a specific class of problems, where the interactions are defined using potential functions.

The octree is used to represent the space which contains the electrical charges. The neighborhood relation in the octree plays an important role in the FMM algorithm. The resulting system, IsSuP - Isopotential Surfaces using Particles, a flexible system for the evaluation and modelling 3D isopotential surfaces generated by means of electrical charges, is presented. The V3DTOOLS Toolbox for Volume Visualization and Processing for the Khoros System is used to visualize the surfaces. Numerical results and images are shown.

Keywords: modelling, particle systems, scientific visualization, octree.

Conteúdo

Capítulo 1

Introdução

Este trabalho trata da implementação do Método Multipólos Rápido para cálculo e visualização de superfícies isopotenciais tridimensionais geradas por partículas elétricas. Este método foi inicialmente descrito por Greengard [Gre89], que apresenta um algoritmo para a avaliação rápida do potencial e campos de força em sistemas de partículas de larga escala.

A avaliação de interações gravitacionais ou elétricas (Coulomb) nesses sistemas é uma parte integral da simulação numérica em diversos processos. Em muitas situações, para ser de interesse físico, a simulação deve envolver milhares de partículas (ou mais), e os campos têm que ser avaliados para variadas configurações. Infelizmente, uma quantidade de trabalho da ordem $O(N^2)$ é requerida para avaliar as interações de todos os pares em um sistema de N partículas, a menos que alguma aproximação seja feita. Em conseqüência, simulações em larga escala são extremamente caras em alguns casos e proibitivas em outros.

O algoritmo descrito por Greengard para avaliação rápida do potencial e campos de força para sistemas de partículas de larga escala, o Método Multipólos Rápido (*Fast Multipole Method*), requer uma quantidade de trabalho proporcional a N dentro de um erro aproximado. Tanto a construção para a versão bidimensional quanto para a tridimensional foram descritas por Greengard. A partir desta publicação, Müller e Müller descrevem em seu artigo *An Object-Oriented Implementation of the Fast Multipole Method* [Mül90] uma implementação orientada para objeto do Método Multipólos Rápido (MMR), para o caso bidimensional.

Aqui, neste trabalho, é descrita a implementação para o caso tridimensional. Não foi adotada a abordagem orientada para objeto por não se ter, à época do início do desenvolvimento, as ferramentas computacionais necessárias. No Capítulo 1 são apresentados os conceitos de Modelagem Geométrica e Visualização Científica, bem como alguns tópicos específicos dessas áreas tais como Modelagem Baseada em Sistemas Físicos e Sistemas de Partículas, visando apresentar brevemente o contexto em que se insere o problema da Modelagem de Superfícies Isopotenciais no Espaço Tridimensional. No Capítulo 2 é apresentado o Método Multipólos Rápido aplicado à modelagem de superfícies isopotenciais 3D e a Teoria Potencial que o apoia. Esse método é baseado na estrutura de dados *octree*, que é descrita no Capítulo 3 bem como os procedimentos de Pesquisa de Vizinhança para essa estrutura de dados. O Capítulo 4 apresenta a arquitetura do sistema implementado, IsSuP - *Isopotential Surfaces Using Particles*. O Capítulo 5 mostra resultados obtidos e imagens das superfícies modeladas. No Capítulo 6 é apresentada a Conclusão.

O algoritmo é adotado em sua íntegra conforme proposto em [Gre89]. A principal contribuição desse trabalho, além da realização da implementação do algoritmo proposto,

está no desenvolvimento da Biblioteca de Rotinas para Pesquisa de Vizinhança em *octrees*, apresentada no Capítulo 3, e sua aplicação ao MMR. Tais rotinas foram desenvolvidas a partir das tabelas publicadas no artigo *Neighbor Finding in Images Represented by Octrees*, por Samet [Sam89], também mostradas no livro *Applications of Spatial Data Structures* [Sam90] do mesmo autor. Assim, nossa atenção se concentrará na estruturação da informação. No Capítulo 5 são apresentados resultados numéricos da avaliação das superfícies isopotenciais utilizando-se o MMR e a lei de Coulomb para os mesmos conjuntos de dados, para confronto. São também apresentadas imagens das superficies aproximadas obtidas através das rotinas da biblioteca V3DTOOLS do sistema Khoros, que é comentado no final desse capítulo.

1.1 Modelagem Geométrica

Quando criamos um modelo de um objeto, criamos um substituto daquele objeto, uma representação, incorrendo numa forma mais conveniente e fácil de usar e analisar. Se o modelo é bom, ele responderá aos nossos questionamentos da mesma maneira que o original o faria. Modelos quantitativos, por exemplo, são encontrados nas ciências físicas, sociais e na engenharia. São usualmente expressos como sistemas de equações, permitindo a experimentação alterando-se os valores das variáveis independentes, coeficientes e expoentes. Assim, modelos simplificam a estrutura atual ou o comportamento da entidade modelada [Fol96].

No processo de modelagem, tentamos nos limitar somente às informações essenciais aos nossos objetivos, ignorando outros aspectos. Modelos não necessariamente precisam conter informação geométrica. Abstrações como modelos organizacionais não

contêm informações espaciais; mesmo assim, tais modelos podem ser representados geometricamente por um diagrama. Um histograma pode, por exemplo, mostrar os resultados de uma avaliação clínica.

Os métodos da Modelagem Geométrica são usados para construir uma descrição matemática precisa da forma de um objeto real ou para simular algum processo. A construção em si mesma é usualmente uma operação apoiada pelo computador, onde o modelo é armazenado e analisado. O uso de um computador é, de fato, central a todo o processo de Modelagem Geométrica. Sem a potencialidade computacional, nós seríamos incapazes de construir e analisar modelos complexos e de importância prática.

Entre os aspectos que um moddo geométrico pode representar estão os seguintes:

- distribuição espacial e forma dos componentes, ou seja, a geometria da entidade, e outros atributos afetando a aparência dos componentes, tais como cor;
- conectividade dos componentes, ou a estrutura ou topologia da entidade. A informação de conectividade pode ser especificada abstratamente, por exemplo numa matriz de adjacências para redes ou numa estrutura de árvores para hierarquias;
- valores de dados específicos da aplicação, tais como características elétricas ou textos descritivos.

Um modelo é criado porque é um substituto econômico e conveniente para o processo ou objeto real, resultando num objeto mais fácil de usar e analisar. Além das vantagens da análise, o modelo também é uma maneira útil, senão necessária, de veicular informações. Para muitas aplicações, o modelo geométrico de um objeto físico necessita ser completado com a descrição das propriedades de reflexão da superfície, textura ou cor; ou informações das propriedades elásticas do material do objeto. Podemos determinar o

detalhe requerido em um modelo pelo uso e operações aos quais tencionamos submetê-lo. Se o modelo é suficientemente rico em detalhes descritivos, podemos realizar operações sobre ele que levem aos mesmos resultados que as operações aplicadas ao próprio objeto [Mor85].

Modelos geométricos podem ter uma estrutura hierárquica induzida por um processo de construção *bottom-up*: os componentes são usados como sub-blocos de entidades de nível mais alto, que por sua vez são usadas como sub-blocos de outras entidades. Como nos grandes sistemas de software, raramente as hierarquias são extritamente *bottom-up* ou *top-down*; o que interessa é a hierarquia final, não exatamente o processo de construção. Hierarquias de objetos são comuns porque poucas entidades são monolíticas (indivisíveis); uma vez que uma entidade é decomposta em uma coleção de partes, existe uma hierarquia de pelo menos dois níveis [Fol96].

Ainda, apesar de modelagem geométrica ser freqüentemente associada com modelagem de sólidos, é isto e mais, porque se aplica a formas e funções de qualquer dimensão.

1.1.1 Desenvolvimento Histórico

Princípios de modelagem geométrica são encontrados já nos primeiros sistemas de computação gráfica para CAD e manufatura. O sistema Sketchpad, de Ivan Sutherland (1963) está entre os pioneiros da área. A modelagem da época tendia a enfatizar os aspectos aparência e apresentação sob a ótica de projeto, destacando-se a exceção dos modelos de elementos finitos usados na análise estrutural. Do ponto de vista da manufatura,

a história começa com a simples modelagem "subtrativa" dos processos de controle numérico.

Os computadores foram introduzidos na manufatura para calcular e controlar os movimentos das máquinas de corte, o que requereu uma nova maneira de compreender e extrair informações da forma dos desenhos de engenharia. Isto não era possível até que linguagens especiais foram desenvolvidas para traduzir informações sobre a forma de desenhos para um formato computacional. Estas informações descritivas eram então transformadas em instruções para ferramentas de máquinas controladas por computador, o que marcou o início da revolução das máquinas de comando numérico na indústria.

Nos meados dos anos sessenta D. T. Ross (1967) desenvolveu no MIT um avançado compilador para uma linguagem para computação gráfica. Nessa mesma época, S. A. Coons (1963, 1965), também no MIT e J. C. Ferguson (1964), na Boeing, iniciaram importante trabalho em superfícies esculpidas. Ainda, a General Motors desenvolveu seu sistema DAC-1. Outras companhias, entre elas Douglas, Lockheed e McDonnell também realizaram desenvolvimentos significativos [Mor85].

As primeiras limitações das linguagens de programação de controle numérico estimularam trabalhos em matemática e aplicações em parametrização de superfícies e modelagem de sólidos. Isto foi reforçado por uma crescente necessidade das indústrias automobilística e aeronáutica. Em Detroit havia forte interesse na geometria de estilos: formas de corpos, estética e iluminação. Em Seattle, St. Louis e no sul da Califórnia, departamentos de engenharia aeronáutica procuravam por melhores ferramentas para estrutura e modelagem aerodinâmica. Nesta época, novos limites foram rompidos na área de geometria paramétrica, incluindo as formas bicúbicas de Coon e as superfícies especiais de Bézier.

Enquanto isso, pesquisadores das áreas de computação gráfica e CAD (incluindo arquitetura) iniciaram trabalhos nas chamadas áreas de *wireframe* e esquemas poligonais. Estes sistemas eram inicialmente bidimensionais e considerados ferramentas para esboços ou métodos para reduzir e interpretar informações digitalizadas. Um modelo *wireframe* é composto de linhas e curvas definindo as arestas de um objeto e é usualmente construído interativamente, de modo muito parecido ao de um desenho de engenharia. Cada linha ou elemento curvo é construído separada e independentemente, sendo armazenado no computador como uma lista de linhas e curvas.

Os sistemas de modelagem *wireframe* atuais são capazes de construir representações tridimensionais e têm procedimentos interativos razoavelmente poderosos. De qualquer maneira, todos os sistemas *wireframe* puros exibem deficiências bem conhecidas e severas, pois os modelos tridimensionais são freqüentemente ambíguos. Permitem também facilmente criar objetos sem sentido, desde que esses sistemas normalmente não têm testes lógicos com este objetivo. Uma outra deficiência é a falta de contorno ou informações de perfil para as superfícies inferidas entre as linhas e curvas *wireframe.*

Os esquemas de modelagem baseados em polígonos foram inicialmente desenvolvidos para criar figuras ou *renderings.* Há uma considerável área cinza entre sistemas baseados em polígonos e *wireframes*, e a distinção nem sempre é clara. A estrutura de dados poligonal é simples, consistindo de listas de referências topológicas cruzadas de vértices, arestas e faces. Os algoritmos para geração e manipulação dos quadros são freqüentemente muito sofisticados, em parte porque esquemas de polígonos foram extensivamente desenvolvidos e usados como ferramentas de pesquisa para

tecnologias de displays gráficos visando animação e superfície escondida ou técnicas de "visibilidade".

Técnicas de superfícies esculpidas foram desenvolvidas e substituíram as técnicas anteriores das indústrias naval, automobilística e aeronáutica. As técnicas de superfície de Bézier e funções bicúbicas paramétricas de Coons e Ferguson foram bem sucedidas, pois sua ênfase na representação acurada de superfícies e em interpolação atende a uma variedade de critérios de projeto.

A "verdadeira" modelagem de sólidos é relativamente nova e pretende superar as limitações de outros esquemas baseados na representação e análise de objetos tridimensionais. Seu objetivo é criar representações geométricas de objetos completas e não-ambíguas. A modelagem de sólidos compreende várias diferentes abordagens: representações de fronteiras, geometria construtiva de sólidos (CSG), representações de varredura (*sweep*), *half-spaces*, e outras. Visa atender a aplicações gerais, o que a separa de outros tipos de abordagens, que são orientados para propósitos específicos mas nem sempre dão informações suficientes para determinar todas as propriedades geométricas dos objetos definidos. A modelagem de sólidos enfatiza a aplicabilidade geral dos modelos, ou seja, representações que permitam responder algoritmicamente questões geométricas arbitrárias [Mor85].

1.1.2 Modelagem Baseada em Sistemas Físicos

Fenômenos naturais nem sempre são eficientemente representados por modelos geométricos. Neblina, p. ex., é composta de pequeninas gotas d'água, mas usar um modelo no qual cada gota seja individualmente colocada está fora de questão. Além disso, esse

modelo de gota d'água não representa acuradamente nossa percepção do neblina; a neblina é visualmente percebida como uma mancha, não como milhões de pequenas gotas d'água. A percepção visual da neblina é baseada em como a neblina altera a luz que chega aos nossos olhos, e não na forma e colocação das gotas d'água. Para modelar este efeito perceptivo eficientemente é necessário um modelo diferente dos mencionados na seção anterior. Da mesma maneira, a forma de uma folha em uma árvore pode ser modelada com polígonos e o caule com um tubo *spline*, mas colocar explicitamente cada galho, ramo e folha de uma árvore pode ser na prática impossível.

A modelagem baseada em sistemas físicos é uma área relativamente nova da computação gráfica que incorpora características físicas nos modelos, permitindo simulações numéricas do seu comportamento. Tornou-se, desta forma, um termo genérico para uma variedade de técnicas que definem princípios físicos de comportamento, utilizando o computador para calcular os detalhes. Algumas técnicas são largamente aplicáveis: modelos procedurais, fractais, modelos baseados em gramáticas e sistemas de partículas têm sido usados para modelar uma imensa variedade de objetos. Para cada técnica de modelagem, métodos dedicados de *rendering* podem ser necessários. Um deles, *volume rendering*, que consiste na atribuição de um valor para cada ponto no espaço 3D, não é totalmente novo, desde que campos escalares têm sido usados na física e na matemática há centenas de anos, mas apenas recentemente tentativas de apresentar tais campos têm sido realizadas [Fol96]. Sofisticadas técnicas de *rendering* podem ser amarradas à descrição de modelos, desde que o *rendering* pode ser visto como a simulação de um modelo de interação entre luz e matéria. Assim, essa abordagem às vezes burla a distinção natural entre modelagem,*rendering* e animação [Bar92].

A modelagem baseada em sistemas físicos combina elementos da matemática aplicada, análise numérica, física, engenharia mecânica, mecânica computacional, etc. Uma importante decorrência desses modelos é como controlá-los: se o comportamento do modelo é inerente a ele, o modelo se comportará da maneira que **ele** quer, e não da maneira que **nós** queremos. Matematicamente, tais modelos são traduzidos em equações diferenciais, as quais são tipicamente colocadas como problemas de valor inicial, onde o usuário tem controle somente sobre as condições iniciais e é difícil determinar o comportamento posterior. Uma escolha comum é usar técnicas baseadas em restrições, que permitem aos usuários especificar valores para várias propriedades dos modelos. Barzel sugere uma biblioteca de protótipos para modelagem de corpos rígidos contendo:

- movimentos newtonianos básicos de corpos rígidos, em resposta a forças e torques;
- capacidade para medir o trabalho realizado por cada força e torque e balanceá-lo em relação à energia cinética dos corpos;
- recursos para vários tipos de forças para aplicar aos corpos para permitir controle baseado em restrições;
- facilidades para manipular descontinuidades em um modelo;
- facilidade de ser estendida, incrementando os módulos acima sugeridos ou adicionando outros módulos.

1.1.3 Sistemas de Partículas

Objetos tais como nuvens, fumaça, neblina, água e fogo são denominados objetos *fuzzy*. Objetos *fuzzy* não têm superfícies lisas, bem definidas e brilhantes, pelo contrário, suas superfícies são irregulares, complexas e pobremente definidas. Eles não são objetos rígidos

nem seus movimentos podem ser descritos pelas transformações afins, comuns em computação gráfica.

Um dos métodos para modelagem de objetos *fuzzy* é através de sistemas de partículas. A representação através de sistemas de partículas difere em três aspectos básicos das representações normalmente usadas em síntese de imagens. Primeiro, um objeto é representado não por um conjunto de primitivas de superfície, tais como polígonos ou *patches*, que definem sua fronteira, mas como nuvens de primitivas de partículas que definem seu volume. Segundo, um sistema de partículas não é uma entidade estática. As partículas mudam de forma e movimento dinamicamente. Novas partículas "nascem" e outras "morrem" no decorrer do tempo. Terceiro, um objeto representado por uma partícula não é determinístico, desde que seu conteúdo e forma não estão completamente especificados.

Na modelagem de objetos *fuzzy*, a abordagem utilizando sistemas de partículas oferece algumas vantagens importantes, em relação às técnicas clássicas orientadas para superfície. Primeiro, uma partícula (um ponto no espaço tridimensional) é muito mais simples de ser gerada do que um polígono, a mais simples das representações de superfície. Conseqüentemente, pode-se processar mais primitivas básicas e produzir uma imagem mais complexa. Uma segunda vantagem é que a definição do modelo é procedural e controlada por números randômicos. Portanto, obter um modelo altamente definido não necessariamente requer uma quantidade muito maior de esforço humano, como é freqüentemente o caso de sistemas baseados em superfície. Por ser procedural um sistema de partículas pode ajustar seu nível de detalhamento para acomodar um conjunto específico de parâmetros de visualização. Da mesma forma que com superfícies fractais, a ampliação de uma determinada região pode revelar mais e mais detalhes. E finalmente, sistemas de

partículas modelam objetos vivos, i.e., que mudam de forma em um determinado período de tempo. É difícil representar dinâmicas assim complexas com técnicas de modelagem baseadas em superfícies.

A modelagem de objetos como coleções de partículas não é uma idéia nova. No final dos anos 70, os primeiros vídeogames mostravam espaçonaves explodindo através de muitos pontos brilhantes que preenchiam a tela. "Fontes pontuais" têm sido usadas como tipos de dados gráficos em muitos sistemas de modelagem tridimensionais, por exemplo, os primeiros simuladores de Evans e Sutherland, apesar de haver poucas referências a eles na literatura.

Neste trabalho, sistemas de partículas são utilizados para modelagem e visualização de superfícies isopotenciais. Neste sentido, discorreremos um pouco mais sobre este modelo, apesar de nem todos os aspectos abordados a seguir terem sido utilizados nesta implementação, mas para ilustrar o processo e possíveis desdobramentos desta aplicação.

Um sistema de partículas é uma coleção de muitas partículas que juntas representam um objeto *fuzzy*. Em um período de tempo, partículas são geradas em um sistema, movem-se, mudam e morrem dentro do sistema.

Numa seqüência de movimento, para o cálculo de cada quadro, os seguintes passos são executados:

- novas partículas são geradas dentro do sistema;
- a cada nova partícula são associados os seu atributos individuais;
- quaisquer partículas que tenham existido dentro do sistema, passado o seu tempo de vida são extintas;

- as partículas remanescentes são movidas e transformadas de acordo com seus atributos;
- uma imagem das partículas vivas é gerada no *frame buffer*.

O sistema de partículas pode ser programado para executar um conjunto de instruções a qualquer momento. Para controlar a forma, aparência e a dinâmica das partículas dentro do sistema de partículas, o projetista do modelo tem acesso a todo um conjunto de parâmetros. Em geral, a cada parâmetro está associado um intervalo no qual um valor de partícula deve constar. Por ser procedural, esta abordagem pode incorporar qualquer modelo computacional que descreve a aparência ou dinâmica do objeto. Por exemplo, os movimentos e transformações das partículas podem ser amarrados à solução de um sistema de equações diferenciais parciais, ou atributos de partículas podem ser instanciados com base em estatísticas mecânicas. Podemos, portanto tirar vantagem de modelos que foram desenvolvidos em outras disciplinas científicas ou de engenharia [Ree83].

1.2 *Volume Rendering*

Volume Rendering é uma técnica usada para mostrar as características do interior de uma região sólida através de um conjunto de imagens 2D. Em um exemplo típico, o sólido é uma peça mecânica que foi aquecida; a temperatura foi calculada em cada ponto no interior através de um meio físico ou matemático e há interesse em se visualizar esta temperatura. *Volume rendering* não é de fato uma técnica de modelagem, desde que a forma e as características da peça a ser visualizada estão ambas disponíveis, a priori; mas a conversão

desses dados para informação em um mapa de *pixels* é uma forma de modelagem; nominalmente, a modelagem de uma transformação 3D para 2D.

Um outro exemplo poderia ser a densidade de células humanas ou animais calculada em cada ponto de uma grade 3D através de tomografia computadorizada. A visualização desta informação indicaria os limites de vários tipos de células (conforme indicado por variações de densidade). As superfícies definindo essas fronteiras podem ser inferidas de amostras de informação de modo a sugerir o sólido.

Um número associado a cada ponto em um volume é chamado de "valor" naquele ponto. A coleção de todos os valores é chamado de "campo escalar" sobre o volume. O conjunto de todos os pontos com um dado valor escalar é chamado de "superfície de nível". Se o campo escalar é suficientemente contínuo, este conjunto de pontos realmente forma uma superfície. *Volume rendering* é o processo de se visualizar campos escalares. É importante se ter em mente que a informação visualizada pode não ser a ideal.

Várias abordagens têm sido desenvolvidas, podendo ser subdivididas em duas categorias: aquelas que calculam superfícies de nível e aquelas que mostram integrais de densidade ao longo dos raios. As duas podem ser combinadas associando-se a densidade somente a certas superfícies de nível e executando-se então o *ray traycing* do resultado. Havendo animação, uma terceira categoria de visualização fica disponível: uma série de slices 2D da informação é calculada e mostrada seqüencialmente, usando cor ou brilho para indicar o valor escalar em cada ponto dos slices. Havendo controle interativo da direção e nível dos slices, essa possibilidade pode resultar numa boa aproximação da estrutura interior do campo escalar.

Apesar disso, às vezes é útil visualizar a informação no conjunto, ao invés de por *slices*. Um primeiro passo nessa direção foi realizado no algoritmo *marching-cubes*, onde

valores escalares são atribuidos a cada ponto de uma grade no 3-espaço. Um nível particular de superfície pode ser aproximado determinando-se todas as intersecções da superfície com arestas da grade. São determinados os pares de pontos adjacentes cujos pontos circundam os valores desejados, i.e., se o valor de um vértice é maior do que o nível escolhido, o valor do outro é menor. A localização de uma intersecção de uma superfície de nível com a aresta é estimada por interpolação linear. A coleção de todos os pedaços de superfície assim definidos constitui uma superfície. A cada subpolígono da superfície pode ser assinalado um vetor normal para ser usado no *shading*. A superfície de nível resultante pode ser apresentada com técnicas convencionais. Esta estratégia é usada em imagens médicas para mostrar a forma dos limites entre os diferentes tipos de materiais e em outras aplicações [Fol96].

1.3 Visualização Científica

Como a modelagem geométrica, visualização científica é em si mesma uma área interdisciplinar envolvendo interface homem-máquina, ciências cognitivas, processamento de imagens, projeto e processamento de sinais, resultando em ambientes e ferramentas para a pesquisa e investigação científicas. Visualização científica diz respeito à exploração de dados e informações graficamente, como um meio de se obter compreensão e *insight* sobre os dados. Este é o objetivo fundamental de muita investigação científica. De fato, toda a compreensão em ciência, tecnologia e mesmo arte reside em nossa capacidade de visualizar. Nós mesmos freqüentemente usamos a expressão "eu vejo" para significar "eu compreendo".

A visualização científica é um processo gráfico análogo à análise numérica, e é freqüentemente referenciada como análise de informações visuais. Sistemas de visualização científica são combinações de ferramentas de software com ambientes poderosos de computação gráfica 3D que permitem que objetos geométricos ou conceitos sejam visualizados por qualquer pessoa. O software provê uma interface amigável para o usuário. O hardware deve ser capaz de manipular objetos 3D complexos, geometricamente descritos em ambientes que permitam movimento, cor e com algum nível de "realismo" para uma melhor comunicação do objeto da computação. Freqüentemente os conjuntos de dados são muito grandes, resultando em problemas de escala e de se estabelecer correlações e inter-relacionamentos entre as diferentes partes da informação.

A visualização é também um meio de se ganhar uma rápida compreensão de processos. Isto poderia ser feito usando métodos clássicos, mas tomaria muito mais tempo. A diferença entre visualização científica e computação gráfica é que esta última preocupa-se primariamente com a comunicação de informações e resultados que já foram entendidos, enquanto que na visualização científica o objetivo é **entender** a informação. Em um grande número de casos a visualização é aplicada à análise e representação de grandes volumes de informações multidimensionais permitindo ao usuário tirar conclusões e extrair resultados rápidos e significativos.

Ferramentas e técnicas nesta área dizem respeito à análise e representação de informações, eventualmente provendo recursos para mudança das informações com respeito ao tempo [Ear92]. Animações são úteis para observar relacionamentos espaciais, estruturas e dinâmica de processos. Ambientes de visualização implicam na adequação do software às necessidades e requisitos do cientista, integração da computação com a visualização,

gerenciamento de grandes conjuntos de informações, manipulação de informações multidimensionais e interação em tempo-real [Ear94].

Nesse trabalho, foi usado para visualizar os resultados obtidos o ambiente Khoros. Khoros é um ambiente aberto de domínio público para processamento de informações, visualização e desenvolvimento de software; integrando múltiplos modos de interface de usuário, geradores de código, auxílio instrucional, visualização e processamento de informações. O resultado é uma ferramenta orientada para pesquisa e desenvolvimento.

A seguir, no Capítulo 2, é apresentado o Método Multipólos Rápido para modelagem de Isosuperfícies Isopotenciais no espaço tridimensional, bem como a teoria que o apoia.

Capítulo 2

Sistemas de Partículas aplicados à Eletrostática

A visualização e simulação de processos físicos e a modelagem de fenômenos naturais têm ocupado um espaço crescente na Ciência da Computação. O computador tem sido usado para visualizar deformações em corpos sob tensão ou o comportamento de fluidos e outros materiais líquidos em situações específicas. Os métodos comuns da Computação Gráfica incluindo modelagem de sólidos e técnicas de *rendering* nem sempre são adequados para tais aplicações especiais. Novas abordagens surgem aplicadas à modelagem baseada em sistemas físicos.

Uma abordagem específica proposta por Greengard [Gre89] é a idéia de modelagem baseada em sistemas de partículas. Diferente da modelagem de sólidos, as primitivas usadas para propósitos de modelagem correspondem a entidades físicas discretas, como átomos ou moléculas. A idéia principal é descrever o **comportamento** de

corpos complexos por partículas e regras simples associadas a estas partículas. Estas regras podem corresponder, p. ex., a leis de Newton para modelar o comportamento físico; podem descrever um comportamento social ou estar associadas a outros tipos de sistemas.

Neste capítulo será descrita uma aplicação de sistemas de partículas na área de eletrostática, para a avaliação de campos potenciais elétricos. Inicialmente, na Seção 2.1, serão introduzidas as equações necessárias para descrever as interações entre as partículas. O Método Multipólos Rápido, descrito na Seção 2.2, sugere uma abordagem para acelerar os cálculos para uma classe específica de problemas, onde as interações são expressas por funções potenciais. A seguir, na Seção 2.3, é apresentada a Teoria Potencial no espaço tridimensional, e nas seções seguintes, as transformações matemáticas para o cálculo das expansões para a avaliação do potencial em uma dada região. Para tanto, são calculados os coeficientes das Expansões Multipólos, definidas na Seção 2.4, e usando esses, são avaliados os coeficientes das Expansões Locais, conforme mostrado na Seção 2.5. Essas expansões são usadas no cálculo do potencial para uma região específica.

O texto a seguir está baseado em [Mül90] e está sendo apresentado de forma ao trabalho ficar auto-contido.

2.1 A Teoria Potencial

Dado um conjunto de partículas elétricas no espaço, deseja-se calcular a força atuando sobre cada uma dessas partículas. Para tanto, será inicialmente apresentado o cálculo do potencial para uma dada posição no espaço.

A força entre dois pontos de carga é dada pela lei de Coulomb:

(2.1)

$$F = k\frac{Q_1 \quad Q_2}{R^2}$$

onde Q_1 e Q_2 são as cargas, R é a distância, e k é um fator de normalização. Nesse caso, conforme sugerido por Müller em [Mül90], k foi igualado a 1.

Sabendo-se que a força entre as cargas atua ao longo da direção que as une, a Equação de Coulomb pode ser representada também na forma vetorial. Seja R_{12} o vetor representado pelo segmento de linha de Q_1 a Q_2, e F_2 a força sobre Q_2. Então:

(2.2)

$$F_2 = k\frac{Q_1 Q_2}{R_{12}^2}u_{12}$$

onde u_{12} é o vetor unidade na direção R_{12}, e portanto será omitido.

Fixando-se a carga Q e movendo-se uma carga Q_t na redondeza, notamos a existência de uma força atuando sobre esta segunda carga. O campo elétrico E pode ser definido como o vetor força sobre uma carga de teste Q_t unitária positiva. Assim,

(2.3)

$$E = \frac{F_t}{Q_t}$$

Um último conceito de interesse é o de potencial de campo elétrico. O potencial Φ em um ponto P é o trabalho de mover uma carga unitária por uma distância d_L em um campo elétrico.De forma análoga ao cálculo de E, o potencial Φ é dado por:

$$\Phi = -\int E.d_L \tag{2.4}$$

Isto nos leva a que:

$$\Phi = k\frac{Q}{R} \tag{2.5}$$

Colocando na forma derivada e introduzindo-se a notação de gradiente V, temos que:

$$E = -\nabla\Phi \tag{2.6}$$

e Φ é chamado o potencial de E [Hay67].

A Equação 2.6 estabelece que o campo elétrico pode ser calculado pelo diferencial do potencial Φ. Para calcular a força atuando sobre uma partícula em uma posição do espaço é suficiente determinar o potencial nesta posição e calcular E e F a partir do potencial.

Uma abordagem natural para calcular o potencial em uma dada posição com respeito a cada partícula no espaço é somar os potenciais com respeito a cada uma individualmente. Deve-se notar que esta abordagem requer uma quantidade de trabalho da

ordem N, onde N é o número total de partículas elétricas. Para calcular o potencial em todas as posições de partículas no espaço este método resulta num algoritmo da ordem de N^2. Muitas abordagens tem sido estudadas com o objetivo de se obter um algoritmo que reduza o custo à ordem de N, e um deles é o algoritmo de Greengard, o Método Multipólos Rápido (MMR).

2.2 O Método Multipólos Rápido (MMR)

A idéia básica em todas as abordagens para otimizar o algoritmo de complexidade N^2 para cálculo dos potenciais é reduzir o número de objetos que participam do cálculo, por exemplo agrupando as partículas em buquês - ou *particle-in-cell* - e calculando a interação de uma partícula com um buquê ao invés de com cada partícula individual.

O potencial de um buquê de partículas pode ser expresso por uma função matemática. A qualquer momento, o potencial em uma posição do espaço com respeito a um certo grupo pode ser determinado avaliando-se a função correspondente. As funções de vários buquês distintos podem ser calculadas em uma posição específica e somadas para obter o potencial para um determinado ponto. Esta é uma maneira mais rápida de determinar o potencial do que o método à força bruta, desde que menos objetos estão envolvidos nos cálculos. O agrupamento de partículas, expressando o potencial desses grupos como expansões e usando uma estrutura de dados hierárquica para subdivisão da região, são as idéias fundamentais do algoritmo MMR. Esta abordagem resulta num algoritmo da ordem O(N) para cálculos no espaço.

Métodos *particle-in-cell* têm sido estudados e usados com sucesso, mais notavelmente em física plásmica. Assumindo-se que o potencial satisfaz a Equação de Poisson, uma malha regular é suposta sobre o espaço computacional e o método procede:

1. interpolando a densidade da fonte nos pontos da malha;
2. usando um "resolvedor de Poisson rápido" para obter valores na malha;
3. calculando a força do potencial e interpolando para a posição das partículas.

A complexidade de tais métodos é da ordem $O(N + M \log M)$, onde M é o número de pontos da malha. O número de pontos da malha é usualmente escolhido proporcional ao número de partículas, mas com uma pequena constante de proporcionalidade de modo que $M << N$. Desta forma, observa-se na prática que o custo computacional é proporcional a N.

Para aumentar a precisão dos cálculos nos métodos *particle-in-cell*, interações sobre conjuntos pequenos podem ser manipuladas pelo cálculo direto, enquanto interações sobre o campo distante são obtidas da malha, dando origem aos assim chamados métodos *particle-particle/particle-mesh*. Enquanto tais algoritmos ainda dependem, para sua performance eficiente, de uma distribuição de partículas razoavelmente uniforme, teoricamente eles permitem que uma alta precisão seja obtida. Como regra, quando a precisão requerida é relativamente baixa, e as partículas são distribuídas mais ou menos uniformemente em uma região retangular, tais métodos são satisfatórios. Entretanto, quando a precisão requerida é alta, o tempo de CPU tende a se tornar excessivo. O limite do erro na aplicação do método é tratado por Greengard no Lema 3.2.3.

Grosso modo, o algoritmo pode ser dividido em duas partes, o Pré-cálculo e a Avaliação, descritas a seguir.

2.2.1 O Pré-cálculo

O objetivo dessa etapa é preparar a estrutura de dados para os cálculos a serem realizados na próxima etapa. No pré-cálculo o espaço computacional é subdividido de maneira a acomodar adequadamente os buquês de partículas. As partículas podem ser agrupadas levando-se em conta as propriedades das expansões empregadas para cálculos potenciais. As expansões são as expansões multipólos e local, descritas nas Seções 2.4 e 2.5, e descrevem o potencial para um dado conjunto de partículas.

Uma abordagem imediata é agrupar as partículas com respeito à sua posição no espaço, o que requer um esquema de subdivisão apropriado e estruturas de dados que reflitam esse esquema, como uma *octree*. Ainda, a estrutura hierárquica permite a escolha de um nível adequado de subdivisão.

2.2.2 A Avaliação

Essa etapa é muito mais complexa que a anterior, e trata do cálculo das expansões que serão usadas para a avaliação do potencial em cada região. Por "cálculo das expansões" entende-se o cálculo dos coeficientes que serão usados nas equações das expansões.

A expansão multipólos nas células folhas deve ser determinada, descrevendo o efeito de todas as partículas contidas na célula na região **exterior** à célula. Esta região é o centro da célula mãe. Após o cálculo das expansões multipólos nas células folhas, elas devem ser transladadas para o centro de suas antecessoras e somadas. A seguir, cada célula no próximo nível de refinamento faz o mesmo com suas expansões multipólos, isto é, as

células mães transladam suas expansões multipólos para suas antecessoras. Quando a raiz da árvore é alcançada, ela conterá a expansão descrevendo o efeito de todas as partículas armazenadas na árvore. A implementação da operação de translação para as células antecessoras usa o próprio mecanismo de retorno da árvore.

Após todas as expansões multipólos terem sido determinadas, as expansões locais representando o efeito das células não vizinhas sobre uma dada célula na árvore é calculado. Células não vizinhas são as duas camadas de células de mesmo nível que estão à distância de duas células da célula pesquisada, conforme apresentado na Fig. 2.1. Pretende-se então avaliar o quanto o campo elétrico das partículas fora da célula "influe" na célula. Para cada célula na árvore, as expansões multipólos de todas as células não vizinhas são convertidas em uma expansão local e somadas. Esta expansão local é transladada para cada filha. As filhas convertem as expansões locais das células não vizinhas em expansões locais, adicionando-as à expansão local recebida de sua mãe. Isto é feito recursivamente até que as folhas sejam encontradas. Como resultado, a expansão local de uma folha representa o efeito de todas as partículas das células não vizinhas sobre aquela folha.

Finalmente, para avaliar o potencial em uma certa posição no espaço, a folha que contém aquela região deve ser encontrada. A expansão local deve ser calculada naquela posição e deve ser adicionada ao resultado do cálculo direto com as partículas das células vizinhas. Este valor será usado para a modelagem das superfícies isopotenciais. Células vizinhas são as duas camadas de células que circundam a folha.

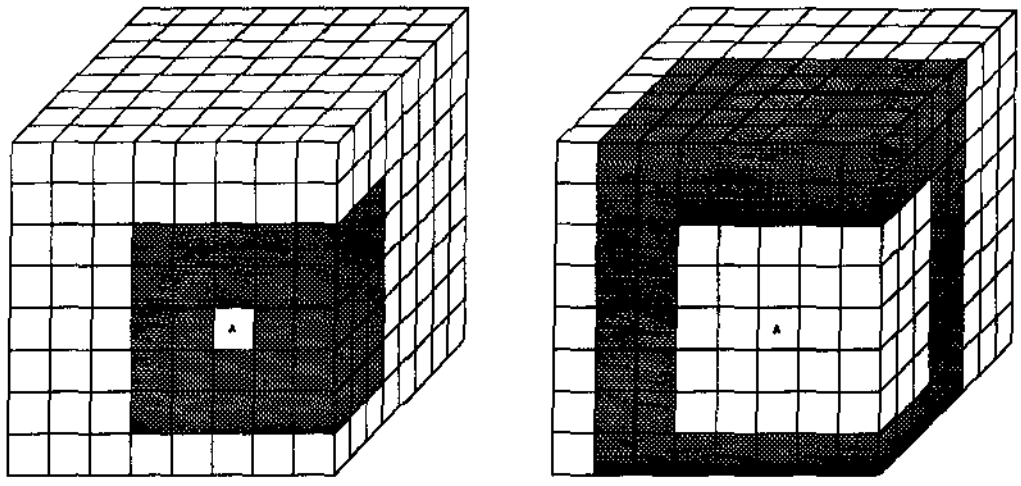


Fig. 2.1 - Células Vizinhas e Não vizinhas de A

A descrição das equações usadas para a avaliação das expansões é apresentada a seguir. Para a avaliação do MMR, é importante que tanto a determinação do conjunto de células vizinhas e não vizinhas bem como o seu acesso seja eficiente, pois para o cálculo das expansões locais, a partir do nível 2 na árvore (nível 0 = raiz), para cada célula são acessadas todas as células não vizinhas, até o penúltimo nível na árvore. Ao nível das folhas, são necessárias as informações de todas as células vizinhas de cada célula.

Na seção seguinte é apresentada a teoria potencial para o MMR. Esta leitura não é indispensável para a compreensão do resto do trabalho, mas necessária para fundamentação das estruturas de dados utilizadas, descritas no Capítulo 4. No Capítulo 3 serão descritos os algoritmos para pesquisa de células vizinhas usados na implementação do MMR.

2.3 Obtenção do Potencial pelo MMR [Mül90]

A Equação de Laplace

$$\nabla^2\Phi = 0 \tag{2.7}$$

permite calcular o potencial na região do espaço que não contém cargas.

O potencial para um grupo de partículas elétricas pode ser expresso como uma função no espaço. Para tanto é preciso encontrar uma solução da Equação de Laplace para três dimensões com respeito a determinadas condições de fronteira. A Equação de Laplace em três dimensões é dada por:

$$\nabla^2\Phi = \frac{\delta^2\Phi}{\delta x^2} + \frac{\delta^2\Phi}{\delta y^2} + \frac{\delta^2\Phi}{\delta z^2} = 0 \tag{2.8}$$

Funções harmônicas são funções que satisfazem a Equação de Laplace. Polinômios de Legendre são soluções especiais para a Equação de Laplace. Elas podem ser expressas pela seguinte fórmula:

$$P_n(x) = \frac{1}{2^n . n!} \cdot \frac{d^n}{dx^n} \cdot (x^2 - 1)^n, \tag{2.9}$$

$$n = 0, 1, 2, \ldots$$

A seguinte fórmula de recursão (em n) é mais adequada para uma geração algoritmica dos polinômios de Legendre:

$$P_n(x) = \frac{(2n-1).x.P_{n-1}(x) - (n-1).P_{n-2}(x)}{n}, \tag{2.10}$$

$n = 2, 3, 4, \ldots$

com $P_0(x) = 1$ e $P_1(x) = x$.

Generalizando, obtemos:

$$P_n^m(x) = (-1)^m.(1-x^2)^{\frac{m}{2}}.\frac{d^m}{dx^m}.P_n(x), \tag{2.11}$$

$n, m = 0, 1, 2, \ldots; m <= n$

e

$$P_n^m(x) = 0, \tag{2.12}$$

$m > n$.

Duas fórmulas recursivas simplificam a geração das funções de Legendre associadas:

$$P_n^m(x) = (n-m+2).(n+m-1).P_n^{m-2}(x) - \frac{2.(m-1).x.P_n^{m-1}(x)}{\sqrt{x^2-1}} \tag{2.13}$$

e

$$P_n^m(x) = (n-m+2).(n+m-1).P_n^{m-2}(x) - \frac{2.(m-1).x.P_n^{m-1}(x)}{\sqrt{x^2-1}} \tag{2.14}$$

onde

$$P_n^0(x) = P_n(x) \tag{2.15}$$

Para obter uma solução geral para a Equação de Laplace a transformação em coordenadas esféricas é útil. Para tanto

$$P_{cart}(x, y, z) = P_{esf}(r, \theta, \phi) \tag{2.16}$$

onde

$$r = \sqrt{x^2 + y^2 + z^2} \tag{2.17}$$

$$\theta = \begin{cases} \frac{\pi}{2}, r = 0 \\ a\cos\left(\frac{z}{r}\right), c.c. \end{cases} \tag{2.18}$$

$$\phi = \begin{cases} sign(y).\frac{\pi}{2}, x = 0 \\ sign(y).\left(\frac{\pi}{2} - sign(x).a\tan\left(\frac{x}{y}\right)\right), |x| < |y|, x \neq 0 \\ sign(y).\left(\pi - a\tan\left(\frac{y}{x}\right)\right), |x| < 0, |x| > |y| \\ a\tan\left(\frac{y}{x}\right), x > 0, x > |y| \end{cases} \tag{2.19}$$

Então a seguinte fórmula fornece uma solução para a Equação de Laplace:

$$\Phi(r,\theta,\phi) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(L_n^m.r^n + \frac{M_n^m}{r^{n+1}}\right).Y_n^m(\theta,\phi) \tag{2.20}$$

onde

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}}.P_n^{|m|}.\cos(\theta).e^{im\phi} \tag{2.21}$$

L e *M* referem-se aos momentos da expansão enquanto *Y* é chamado de harmônica esférica de grau *n*.

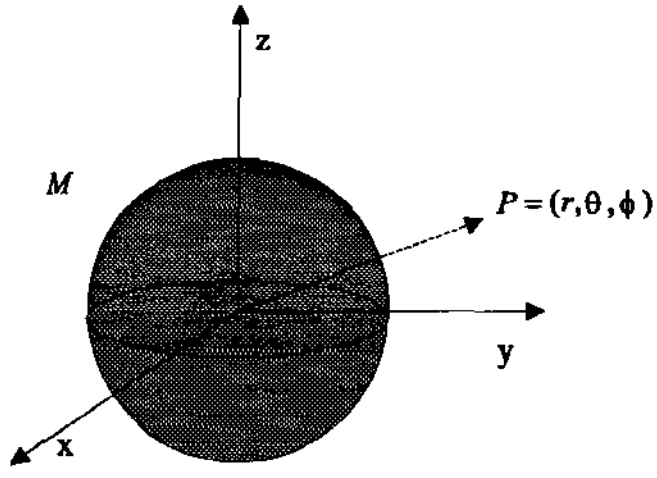


Fig. 2.2 - A Expansão Multipólos

2.4 A Expansão Multipólos

Agora é possível dar uma definição para a expansão multipólos. Uma expansão multipólos descreve o potencial para um dado conjunto de partículas elétricas. Seja um conjunto de w partículas de forças $\{q_i,\ i = 1, 2, \ldots, w\}$ e localizações $\{Q_i\ (\rho_i, \alpha_i, \beta_i),\ i = 1, 2, \ldots\}$. Então o potencial para um ponto $P = (r, \theta, \phi)$, $r > \rho_i$, conforme ilustrado na Fig. 2.2, é dado por:

$$\Phi(P) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{M_j^k}{r^{j+1}} . Y_j^k(\theta, \phi) \tag{2.22}$$

onde

$$M_u^v = \sum_{i=1}^{w} q_i . \rho_i^u . Y_u^{-v}(\alpha_i, \beta_i), \tag{2.23}$$

$$u, v = 0, 1, 2, \ldots$$

Suponha uma dada expansão multipólos centrada no ponto $O = (\rho, \alpha, \beta)$. A operação para transladar o potencial para um ponto $P = (r, \theta, \phi)$ é dada por:

$$\Phi(P) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{M_j^k}{S^{j+1}} . Y_j^k(\omega, \tau) \tag{2.24}$$

onde $O - P = (s, \omega, \tau)$. Então a expansão multipólos correspondente centrada na origem é definida como:

$$\Phi(P) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{\overline{M}_j^k}{r^{j+1}} . Y_j^k(\theta, \phi) \tag{2.25}$$

onde

$$\overline{M}_u^v = \sum_{n=0}^{u} \sum_{m=-n}^{n} \frac{M_{u-n}^{v-m} . F_m^{v-m} . A_n^m . A_{u-n}^{v-m} . \rho^n . Y_n^{-m}(\alpha, \beta)}{A_u^v} \tag{2.26}$$

e $u = 0, 1, 2, \ldots$ e $v = -u, -u+1, \ldots, u$. Ainda, F e A são definidos como:

$$F_j^k = \begin{cases} (-1)^{min(|j|,|k|)}, j.k < 0 \\ 1, c.c. \end{cases} \tag{2.27}$$

e

$$A_j^k = \frac{(-1)^j}{\sqrt{(j-k)!.(j+k)!}} \tag{2.28}$$

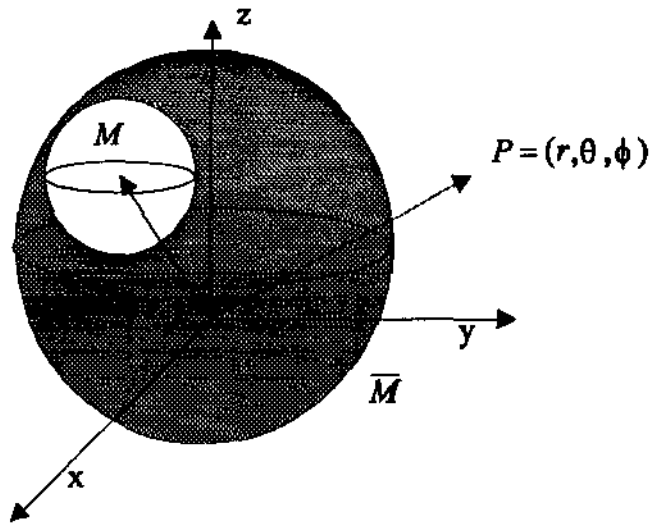


Fig. 2.3 - A Translação da Expansão Multipólos

Esta operação permite a combinação de diferentes expansões multipólos em uma única expansão. Novamente, isto pode ser obtido pela translação de todas as expansões multipólos e superposição das expansões transladadas. Isto é calculado transladando-se todas as expansões multipólos para a nova origem e somando-as, conforme ilustrado na Fig. 2.3.

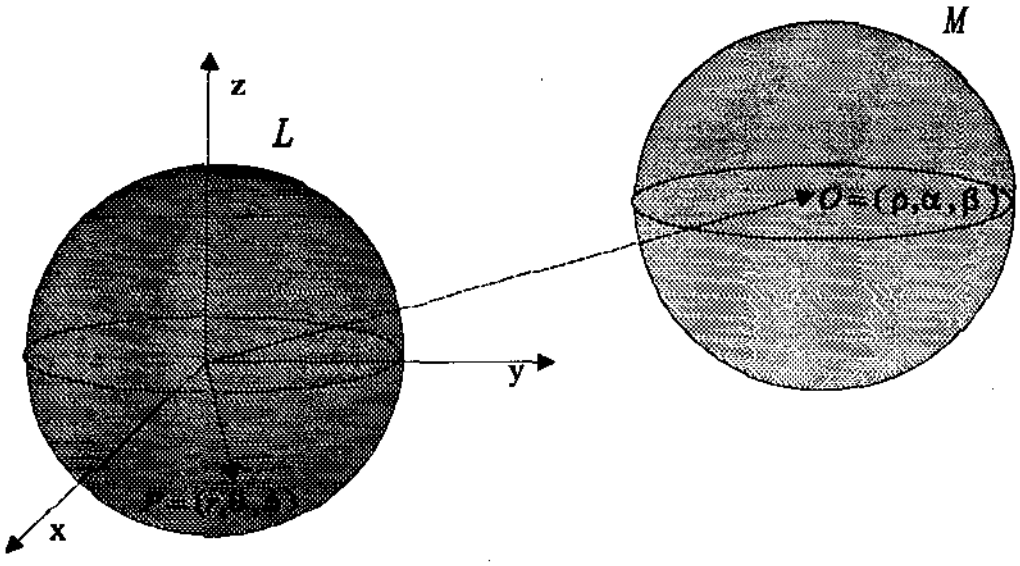


Fig. 2.4 - Conversão da Expansão Multipólos em Local

2.5 A Expansão Local

A avaliação de uma expansão local é válida fora da esfera que contém todas as partículas. Por outro lado, a contrapartida da expansão mulipólos, a expansão local é válida somente dentro de uma região de analiticidadc. Uma expansão multipólos pode ser transformada em uma expansão local a qual será válida em uma esfera com o mesmo raio que a expansão multipólos. Conseqüentemente ambas as esferas devem ser bem separadas, isto é, elas não devem se intersectar, conforme Fig. 2.4.

Seja uma dada expansão em torno do ponto $O = (\rho, \alpha, \beta)$. O potencial para um ponto $P = (r, \theta, \phi)$ é dado por:

(2.29)

$$\Phi(P) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{M_j^k}{s^{j+1}} . Y_j^k(\omega, \tau)$$

onde $O - P = (s, \omega, \tau)$. A expansão local correspondente centrada na origem é definida como:

(2.30)

$$\Phi(P) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k . Y_j^k(\theta, \phi) . r^j$$

onde

(2.31)

$$L_u^v = \sum_{n=0}^{u} \sum_{m=-n}^{n} \frac{M_n^m . G_{u,v}^m . A_n^m . A_u^v . Y_{u+n}^{m-v}(\alpha, \beta)}{A_{u+n}^{m-v} . \rho^{u+n+1}}$$

onde $u = 0, 1, 2 \ldots$ e $v = -u, -u + 1, \ldots, u$. G é definido como

(2.32)

$$G_{i,h}^k = \begin{cases} (-1)^i . (-1)^{min(|h|,|k|)}, h.k > 0 \\ (-1)^i, c.c. \end{cases}$$

A translação de uma expansão local é análoga à translação de uma expansão multipólos, conforme ilustrado na Fig. 2.5.

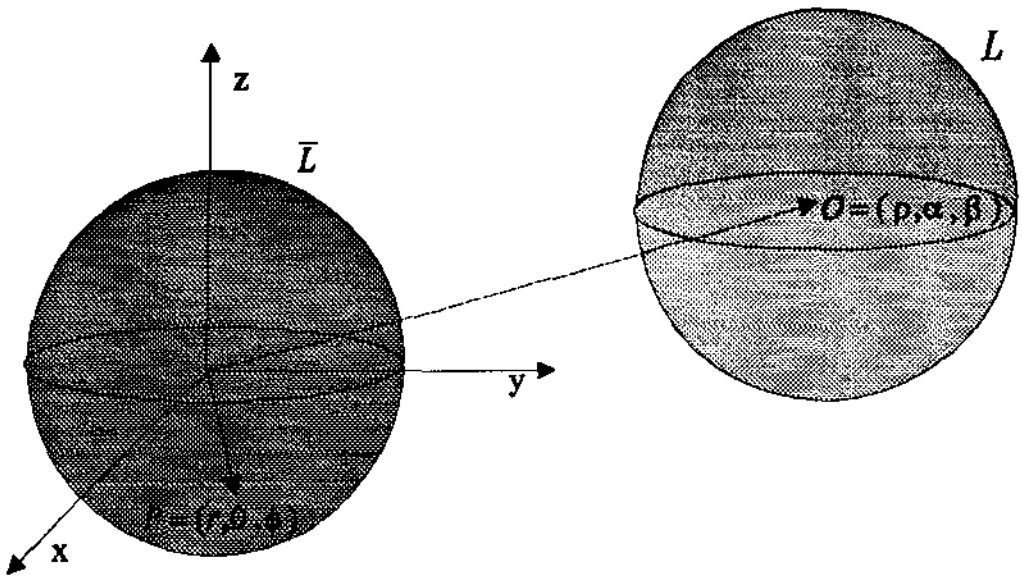


Fig. 2.5 - A Translação da Expansão Local

Seja uma dada expansão local centrada no ponto $O = (\rho, \alpha, \beta)$. O potencial para um ponto $P = (r, \theta, \phi)$ é dado por:

(2.33)

$$\Phi(P) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k . Y_j^k(\omega, \tau) . s^j$$

onde $P - O = (s, \omega, \tau)$. A expansão local correspondente centrada na origem é então definida como:

(2.34)

$$\Phi(P) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \overline{L}_j^k . Y_j^k(\theta, \phi) . r^j$$

onde

$$\overline{L}_u^v = \sum_{n=u}^{\infty} \sum_{m=-n}^{n} \frac{L_n^m . H_{n-u,m-v}^m . A_{n-u}^{m-v} . A_u^v . Y_{n-u}^{m-v}(\alpha, \beta) . \rho^{n-j}}{A_n^m} \tag{2.35}$$

onde $u = 0, 1, 2, \ldots$ e $v = -u, -u + 1, \ldots, u$. Ainda, H é definido como:

$$H_{i,h}^k = \begin{cases} (-1)^i . (-1)^h, h.k < 0 ou k = 0 \\ (-1)^i . (-1)^{k-h}, h.k > 0 e |k| < |h| \\ (-1)^i, c.c. \end{cases} \tag{2.36}$$

A seguir, no Capítulo 3, serão apresentados os esquemas e descritos os procedimentos usados para obtenção das células vizinhas e não vizinhas na *octree*.

Capítulo 3

Estruturas Hierárquicas para Subdivisão do Espaço

Nesse capítulo será mostrada a estrutura de dados usada para descrever o espaço computacional no **Sistema IsSuP - Isopotential Surfaces Using Particles**, apresentado no Capítulo 4. O espaço computacional é a representação do espaço onde estão "situadas" as particulas. A estrutura de dados adotada é a *octree*. Serão descritos também os procedimentos para pesquisa de vizinhança nessa estrutura.

A pesquisa de vizinhança na *octree* representa um aspecto fundamental do MMR e é usada na avaliação dos coeficientes das expansões multipólos e local, descritas no capitulo anterior. A expansão local será usada no cálculo do potencial nas regiões onde é desejado esse valor. Essas regiões são os pontos de uma grade regular tridimensional na *octree*. Também essas regiões são acessadas através dos procedimentos de pesquisa de vizinhança.

Na Seção 3.1 é apresentado o conceito de *octree*. Nas seções seguintes, 3.2 e 3.3, são introduzidos aspectos da pesquisa de vizinhança nessa estrutura. Na Seção 3.4, a *octree* é associada a um 3-cubo, que é também definido, e supre o mecanismo que será usado nos procedimentos de pesquisa de vizinhança. As seções 3.5 e 3.6 descrevem esses procedimentos, que serão usados no cálculo do potencial através do MMR.

3.1 Octrees

Octrees são estruturas de dados hierárquicas baseadas na decomposição recursiva do espaço em oito subvolumes, onde a raiz da *octree* refere-se ao volume completo. *Octrees* são uma generalização natural das *quadtrees* e têm sido usadas em aplicações 3-D, sendo particularmente apropriadas para representar volumes em visualização científica, onde os pontos (coordenadas) freqüentemente definem uma decomposição espacial em regiões cúbicas regulares ou irregulares, disjuntas, espaciais. Se os nós da *octree* forem usados para armazenar informações relativas a todos os subvolumes contidos, é possível explorar o volume sem se examinar cada coordenada.

Assim, a *octree* facilita a subdivisão adaptativa do espaço. O elemento inicial de uma *octree* é um cubo, um subconjunto do espaço tridimensional. O cubo - neste caso ***célula antecessora*** ou ***mãe*** - pode ser subdividido em oito cubos iguais - ***células filhas*** - recursivamente, até que um nível desejável de refinamento seja alcançado. Diz-se ainda que as células filhas de um mesmo nó são células ***irmãs***.

Na implementação do Método Multipólos Rápido para cálculo e visualização de superfícies isopotenciais, a *octree* foi usada para representar o espaço onde estão contidas as partículas elétricas. A raiz da árvore contém o espaço total e cada nó representa uma

célula em um certo grau de refinamento, onde as folhas representam as células no mais alto grau. O critério para a subdivisão de uma célula no MMR é o número máximo de partículas em uma folha. Os termos ***célula*** e ***nó*** serão usados indiscriminadamente neste texto para designar um nó genérico da árvore, sendo que o termo ***folha*** será usado para designar um nó terminal.

Se o número de partículas em um nó for maior do que um valor pré-estabelecido, o nó é então subdividido em oito outros nós. Os nós contém informações para o cálculo das expansões tais como as coordenadas da origem e do centro da célula, os valores dos coeficientes das expansões multipólos e local nesta região, o número de partículas e o caminho da raiz até a célula na árvore. Os nós não-terminais contém os apontadores para suas sub-árvores. Os nós terminais ou folhas contêm a lista de partículas contidas na folha.

3.2 Relações de Vizinhança na *Octree*

As relações de vizinhança numa *octree* são um aspecto muito importante na avaliação do algoritmo MMR. Uma expansão multipólos (Seção 2.4) pode ser calculada somente em uma posição suficientemente distante do centro da expansão, e a conversão de uma expansão multipólos em uma expansão local é válida apenas se as origens de ambas as expansões tiverem uma distância apropriada uma da outra. Isto pode ser expresso em termos de relações de vizinhança numa *octree*.

São definidas como ***vizinhas*** de uma célula X as células tendo uma face, aresta ou vértice em comum com A no mesmo nível de refinamento. Essas células são mais especificamente chamadas de ***vizinhas de primeira ordem***. Apresentamos abaixo na Fig.

3.1 o que seriam, no plano, as vizinhas de primeira ordem. No espaço, essas células seriam os cubos nas posições equivalentes. Usaremos extensivamente nesse capítulo a analogia com o plano, já que a visualização é mais simples e equivalente.

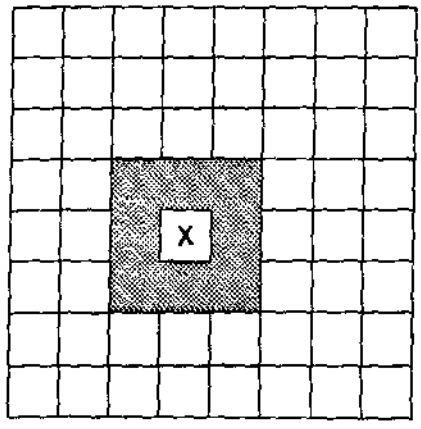


Fig. 3.1 - Células vizinhas de primeira ordem em relação a X, vistas no plano

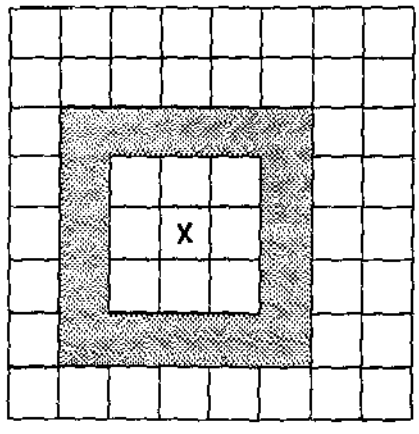


Fig. 3.2 - Células vizinhas de segunda ordem em relação a X, vistas no plano

Ainda, no algoritmo MMR para três dimensões, são também definidas como vizinhas as células que são vizinhas destas células (com exceção da própria célula *X*), as quais são chamadas de ***vizinhas de segunda ordem***, conforme mostrado na Fig. 3.2, também no plano.

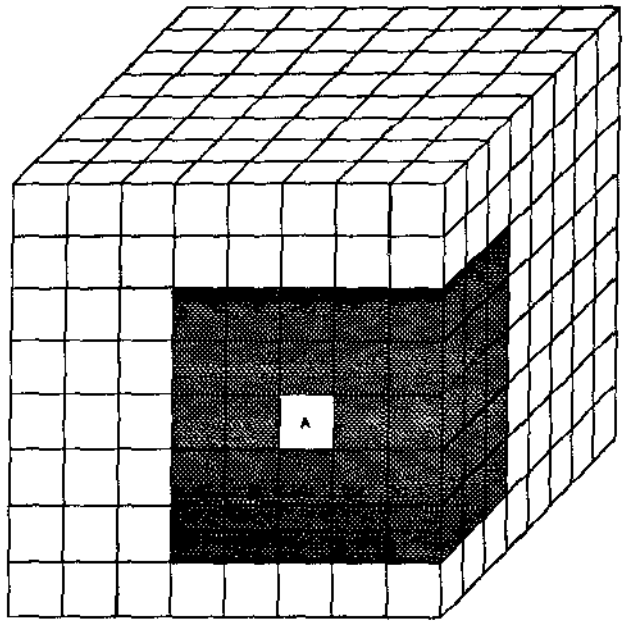


Fig. 3.3 - Células Vizinhas de A numa Octree

Acima, na Fig. 3.3 são mostradas as células vizinhas de uma outra célula *A*, no espaço.

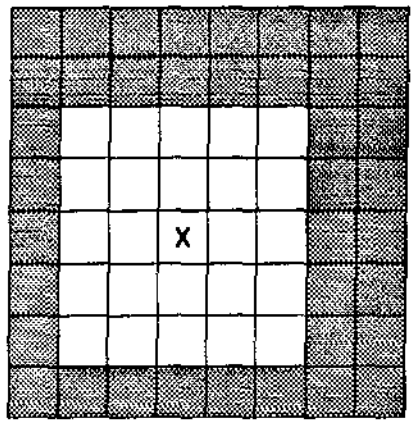


Fig. 3.4 - Células não vizinhas em relação a X, vistas no plano

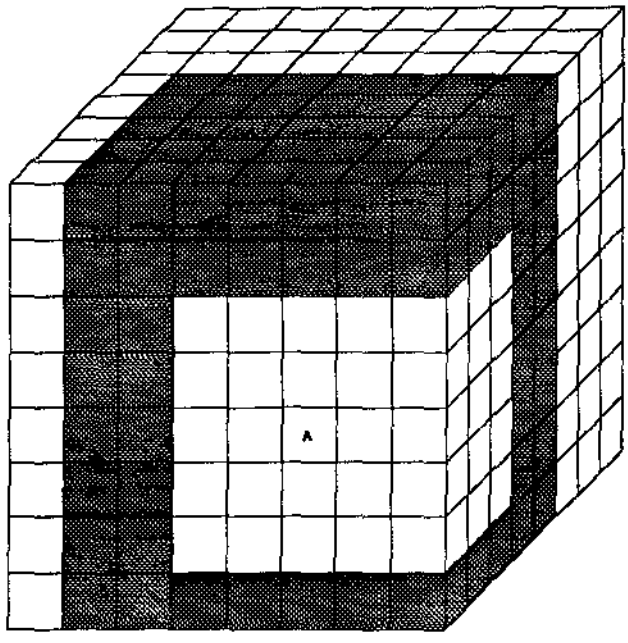


Fig. 3.5 - Células Não vizinhas de A numa Octree

Uma célula não vizinha é uma célula que não é vizinha de uma outra célula do mesmo nível. São ditas ***não vizinhas de primeira ordem*** as células vizinhas às vizinhas de segunda ordem e que não são vizinhas de *A*. Ainda, as ***não vizinhas de segunda ordem*** são as vizinhas das não vizinhas de primeira ordem e que não são vizinhas de *A*. Em outras palavras, são chamadas não vizinhas as duas camadas de células que envolvem as células vizinhas num reticulado regular. A Fig. 3.4 mostra no plano as não vizinhas de *X* e a Fig. 3.5 mostra no espaço as não vizinhas de *A*.

Seja, na Fig. 3.6, C_1 o "cubo" que contém as células vizinhas de A. Temos então um cubo de lado <= 5 circundando *A*. Se denotarmos por V_1 o volume de C_1, temos que o número máximo de células vizinhas $NViz = V_1 - 1 = 5^3 - 1 = 124$.

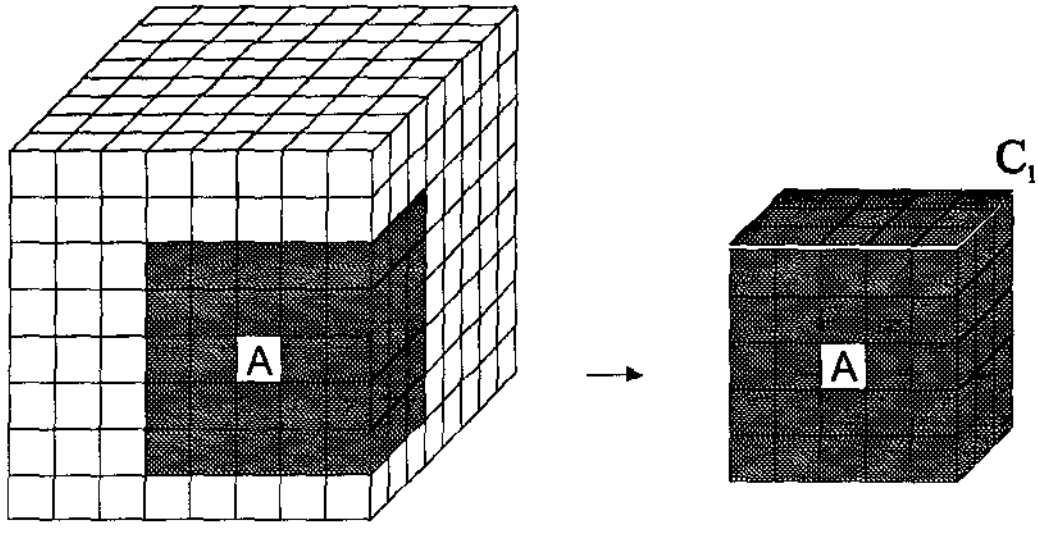


Fig. 3.6 - O cubo C_1

Analogamente, seja, na Fig. 3.7, C_2 o cubo que contém as células não vizinhas de A. Temos então um cubo de lado <= 9 circundando A. Se denotarmos por V_2 o volume de C_2, temos que o número máximo de células não vizinhas $NNviz = V_2 - V_1 = 9^3 - 5^3 = 604$.

Assim, numa *octree* cada célula tem um máximo de 124 vizinhas e 604 não vizinhas. Note que, devido ao elevado número de células não vizinhas na *octree* é importante que a sua determinação e acesso seja eficiente, pois para a avaliação do MMR elas são calculadas para todas as células na *octree*.

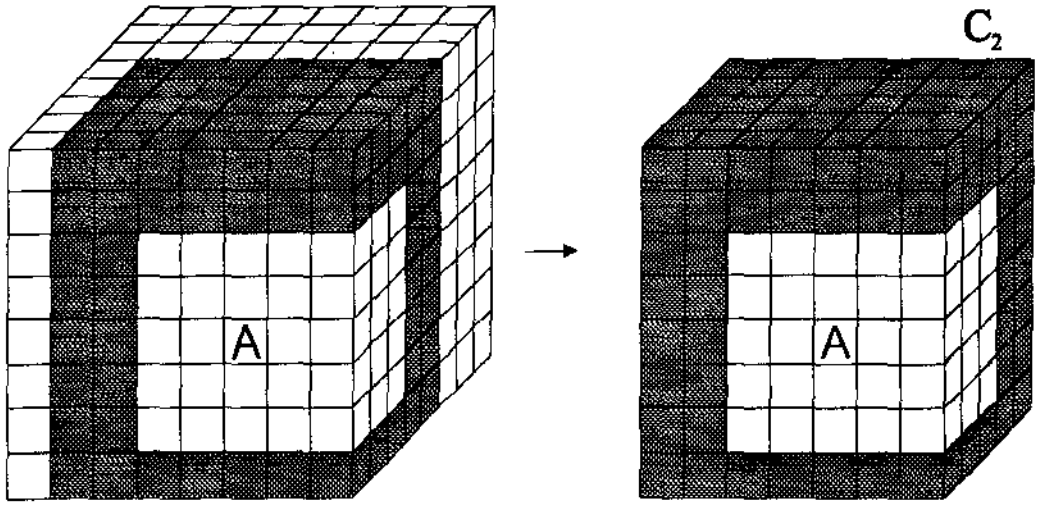


Fig. 3.7 - O cubo C_2

3.3 Algoritmos de Pesquisa de Vizinhas

Os algoritmos existentes de pesquisa de vizinhas podem ser divididos em duas principais abordagens, *bottom-up* e *top-down*, supondo-se que não haja na árvore conexões entre células do mesmo nível.

Os ***algoritmos top-down*** atravessam a árvore da raiz até a célula, usando a trilha da célula para alcançá-la. Iniciando na raiz, a filha que está no caminho da célula procurada é selecionada, e isto é repetido até que a célula buscada seja atingida. O algoritmo tem a desvantagem de que o caminho até a célula deve ser conhecido. Para cada célula vizinha, a árvore deve ser atravessada a partir do topo novamente, o que pode ser computacionalmente caro.

Numa abordagem diferente, pode-se dizer que o ***algoritmo bottom-up*** (num abuso de linguagem) usa posições relativas ao invés de posições absolutas. A idéia básica é "escalar" a *octree* até que uma antecessora comum seja encontrada, e então "descer" novamente em busca da célula vizinha. Certamente sempre se pode subir até a raiz da árvore e então iniciar a descida. De qualquer modo, o objetivo é encontrar a antecessora comum mais próxima, o que minimiza o número de nós que devem ser visitados. A partir daí, refaz-se o caminho usado para localizar esta célula, exceto que os movimentos são feitos "ao contrário".

A Fig. 3.8 mostra a representação gráfica de uma *quadtree*. Cada quadrado no desenho corresponde a uma célula (ou nó) na *quadtree*. Para orientação na *quadtree*, associa-se a cada canto da célula uma direção. Assim, suponhamos que se queira encontrar a vizinha à Oeste da célula *N*. A célula antecessora comum mais próxima é a primeira célula que é alcançada através de sua filha NE ou SE, ou seja, a primeira célula

antecessora da qual *N* não é uma descendente à Oeste. A partir daí, o caminho é percorrido "ao contrário" em movimento descendente (em direção às folhas), tomando-se sempre as direções opostas das encontradas no movimento ascendente. No caso da vizinha à Oeste, as imagens reflexas de NO e SO são NE e SE, respectivamente. Então, a vizinha à direita da célula *N* na Fig. 3.3 é a célula *K* [Sam84].

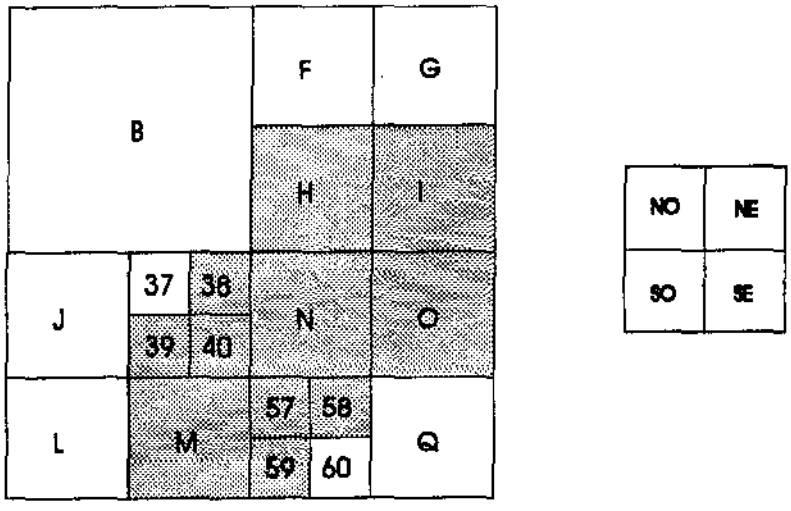


Fig. 3.8 - Representação e Orientação na *Quadtree*

A Fig. 3.9 mostra a representação esquemática da *quadtree*. Se a célula antecessora comum estiver próxima às folhas, o caminho de busca pode ser mais longo do que o caminho *top-down* da raiz até a folha. Por outro lado, o caminho de busca será muito mais curto quando a célula conectora estiver próxima da célula origem.

A razão entre os caminhos mais longo e mais curto decresce com a quantidade de filhos em um nó. Quanto mais filhos um nó pode ter, mais freqüentemente pode ocorrer que

o caminho de busca seja mais curto que o caminho da raiz até o vizinho. De qualquer modo, a abordagem *bottom-up* é em geral mais rápida do que a abordagem *top-down*, quando os vizinhos são requeridos para todos os nós [Mül90].

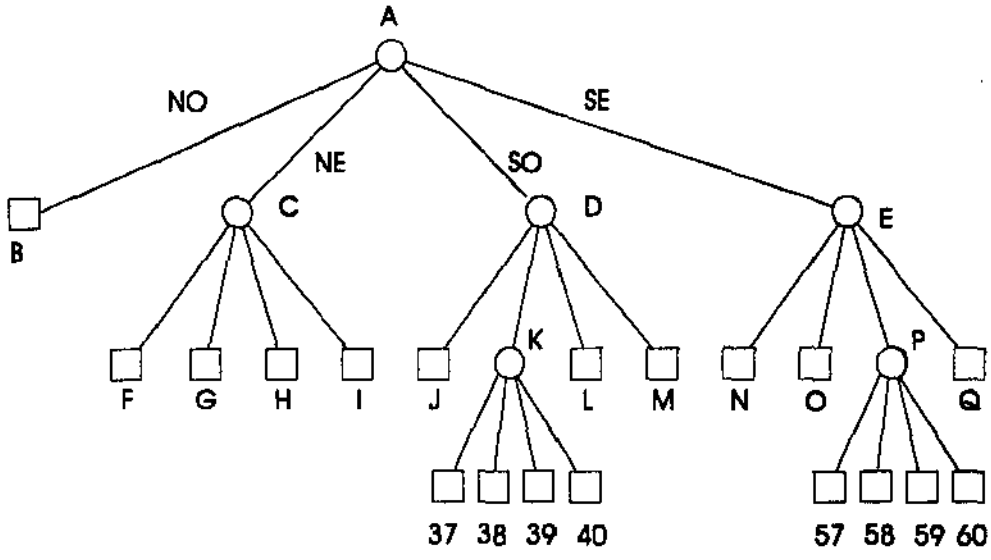


Fig. 3.9 - Representação Esquemática da *Quadtree*

3.4 A *Octree* vista como um 3-Cubo

Na literatura [Sam89], em geral a pesquisa de vizinhança em *octrees* é tratada através de uma tabela de predicados que define relacionamentos tais como adjacente, oposto, face-em-comum, aresta-em-comum. O método descrito a seguir permite a nível lógico a escolha do

menor caminho (*bottom up/top down*) para o acesso a um nó vizinho, usando apenas operações de complementação booleana.

Na abordagem *bottom-up* para pesquisa de vizinhos foi mencionado o conceito de direção no exemplo sobre a *quadtree* (NE, NO, SE, SO). Na verdade, é necessário algum esquema de orientação para se percorrer a árvore, e nada mais natural do que usar os conceitos direita/esquerda na árvore binária ou uma analogia com os pontos cardeais no caso da *quadtree*.

No caso da *octree* isto torna-se muito mais complicado, pois sua representação espacial é um cubo. Assim, fica difícil associar um esquema de orientação intuitivo tal como os mencionados acima. Se empregarmos um raciocínio análogo ao da *quadtree* à *octree*, e associarmos uma direção a cada vértice no cubo, obteremos um total de oito direções, conforme a Fig. 3.10.

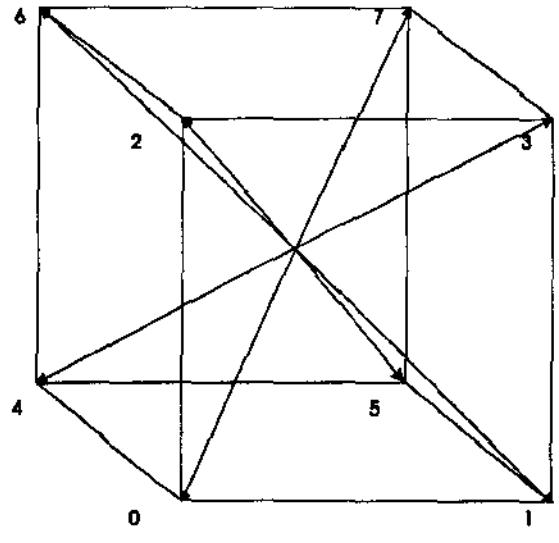


Fig. 3.10 - As direções no cubo

Buscando-se um esquema conhecido que permitisse organizar as direções na *octree* (cubo) de alguma maneira, optou-se por visualizá-la como um 3-cubo unário. Por definição [Prc73], um 0-cubo unário é um vértice; um 1-cubo é um segmento; um 2-cubo é um quadrado e um 3-cubo é o próprio cubo. Para $n \geq 4$, é possível a representação de um ***n-cubo*** através de um grafo não-direcionado, cujos vértices são os vértices do n-cubo e cujas arestas conectam os vértices que diferem em apenas uma coordenada (vértices adjacentes). Por exemplo, (1, 1, 0, 1) e (1, 0, 0, 1) são adjacentes porque eles diferem somente na coordenada x_3 no conjunto de vértices $V = \{ v \mid v = (x_4, x_3, x_2, x_1)\}$. Os diagramas dos n-cubos para $n = 1, 2, 3$ são mostrados na Fig. 3.11.

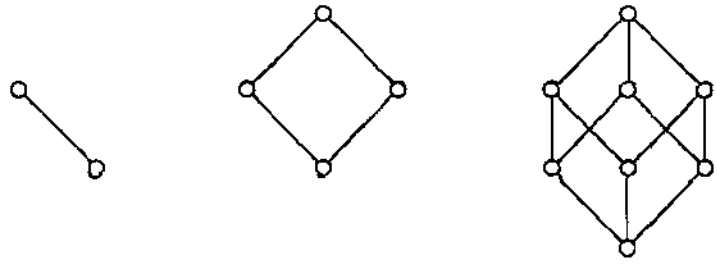


Fig. 3.11 - Os diagramas do 1-cubo, do 2-cubo e do 3-cubo

É exatamente a propriedade acima, ou seja, de dois vértices diferirem em uma única coordenada, que nos será útil na pesquisa de vizinhos ao esquematizarmos a *octree* como um 3-cubo. Na Fig. 3.12, é apresentado um cubo de lado 1 e vértices v_i, $i \in [0..7]$, onde i é o octal correspondente à seqüência de dígitos binários $z_i\, y_i\, x_i$. Os octantes da *octree*

recebem a numeração do vértice em comum com o 3-cubo, e esse esquema é usado para orientação durante o percurso da *octree*, conforme mostrado na Fig. 3.13.

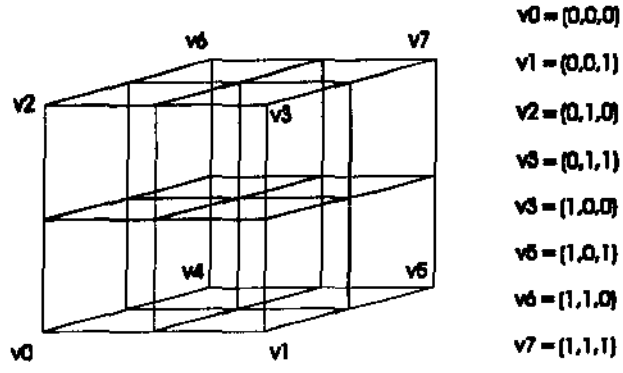


Fig. 3.12- A Octree vista como um 3-cubo

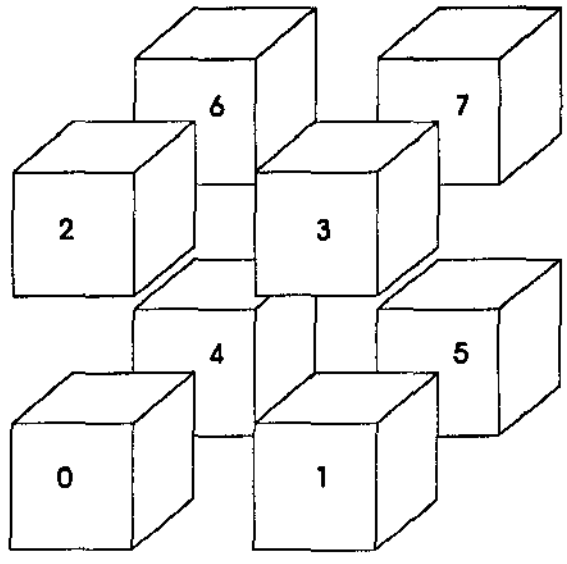


Fig. 3.13 - Octantes numerados de acordo com o 3-cubo

3.5 Algoritmos para Pesquisa de Vizinhança na Octree

A adoção de um esquema como o 3-cubo para a *octree* é bastante conveniente devido ao fato de que nesta representação os vértices adjacentes diferem em apenas uma coordenada. Baseado nesta propriedade, foram elaborados os algoritmos para obtenção dos vizinhos e não vizinhos de uma dada célula [Mor92].

Na definição da estrutura de dados, a cada célula foi associada um campo ***trilha***, que indica o caminho percorrido a partir da raiz para se atingir o nó. O campo trilha é um vetor que representa a seqüência de octantes percorridos até a célula. Por definição, a raiz está no nível 0.

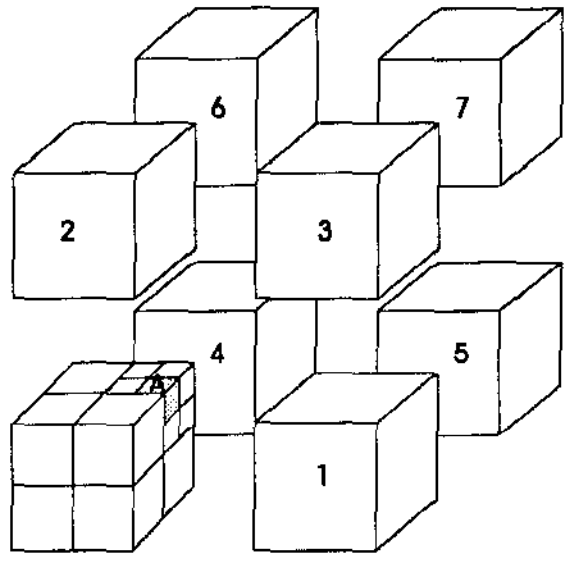


Fig. 3.14 - O caminho até a célula A

No exemplo da Fig. 3.14, a trilha associada à célula *A* é [0073] pois o caminho da raiz até a célula é obtido tomando-se sucessivamente os octantes 0, 7, 3 dos octantes anteriores:

trilha_A = [0073];

onde

```
trilha_A[0] = 0;          /*    raiz    */
trilha_A[1] = 0;
trilha_A[2] = 7;
trilha_A[3] = 3;
```

O índice do vetor trilha indica o nível do octante na *octree*; assim o octante ao nível 1 no caminho até a célula *A* é o octante 0; o octante ao nível 2 é o octante 7, e o octante ao nível 3 é o octante 3 que corresponde à própria célula *A*.Neste texto, nos referiremos freqüentemente a uma célula através de sua trilha, ou seja, a célula [0073] indica a célula que é alcançada pela trilha [0073].

Serão apresentados a seguir os conceitos de células vizinhas imediatas, e a partir dele, os conceitos de células vizinhas de face, aresta e vértice.

3.5.1 Vizinhas Imediatas

Por ***vizinhas imediatas*** definem-se as células que têm uma face em comum com uma dada célula e não são suas irmãs. Uma célula tem no mínimo zero vizinhos imediatos e no

máximo três. Tais células são também chamadas de ***vizinhas imediatas em relação a uma componente***, x, y ou z. Na figura a seguir, as células *B, C e D* são vizinhas imediatas da célula *A*.

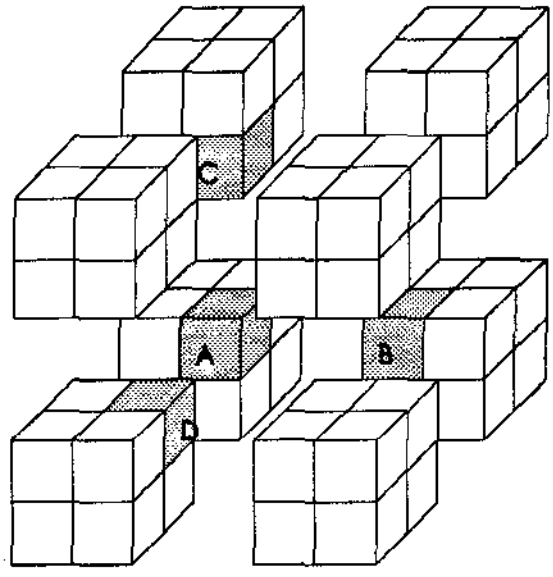


Fig. 3.15 - As células B, C e D são vizinhas imediatas da célula A

Note no desenho da Fig. 3.16 que as células marcadas têm uma face em comum com a célula A, mas não são suas vizinhas imediatas pois são células irmãs, de acordo com a definição da Seção 3.1.

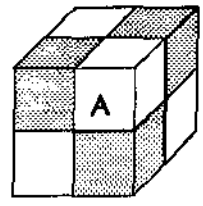


Fig. 3.16 - As células A e suas irmãs com face comum

Podemos dizer que a célula *B* é a vizinha imediata em relação a x; a célula *C* em relação a *y* e a célula *D* em relação a *z*, se adotarmos uma referência conveniente. Podemos também associar a *x, y e z* os conceitos direita/esquerda; abaixo/acima e frente/trás, respectivamente. Podemos ainda supor que cada componente terá apenas dois valores, 0 ou 1. Assim, a trilha [073] pode ser visualizada da seguinte maneira, onde n representa o nível da célula no octree:

n	*z*	*y*	*x*	t_A
0	0	0	0	0
1	0	0	0	0
2	1	1	1	7
3	0	1	1	3

Tab. 3.1 - Decomposição das direções em x, y e z.

Adotando-se os sinais '*' para "frente" e '+' para "trás", podemos reescrever a tabela acima da seguinte maneira, onde n é nível da célula na octree; t_A é a trilha da célula *A*, e *x, y e z* as componentes do eixo cartesiano:

n	z	y	x	t_A
0	*	↓	←	0
1	*	↓	←	0
2	+	↑	→	7
3	*	↑	→	3

Tab. 3.2 - Decomposição das direções.

Então, a partir da raiz no nível $n = 0$, a célula A é encontrada tomando-se o octante à frente, abaixo e à esquerda, que corresponde ao octante 0, no nível 1; a seguir o sub-octante atrás, acima e à direita, que corresponde ao octante 7 e está no nível 2. Em relação a esse octante, toma-se o sub-octante à frente, acima e à direita, que corresponde ao octante 3 no nível 3.

3.5.2 Obtenção das Vizinhas Imediatas

Vimos no exemplo da Fig. 3.14 que a trilha associada à célula A é t_A = [0073] pois o caminho da raiz até a célula é obtido tomando-se sucessivamente os sub-octantes 0, 7, 3 dos octantes que os contêm.

Temos então que o caminho percorrido em relação à componente x é [0, 0,1,1], ou seja, em relação à componente x, a célula foi encontrada através do ***sentido*** 1 (→).

n	z	y	x	t_n
0	0	0	0	0
1	0	0	0	0
2	1	1	1	7
3	0	1	1	3

Tab. 3.3 - A componente x

De maneira análoga à sugerida na Seção 3.3, "escalamos" a árvore enquanto a componente x é igual a 1, o que significa que o sentido através do qual a célula foi encontrada não mudou. Neste caso chegamos ao nível 1, onde o sentido é "0". O sentido mudou, o que indica que chegamos à célula antecessora comum. A partir daí, refazemos o caminho, invertendo o sentido das células percorridas, e obtemos o ***caminho invertido***, que nos levará à célula vizinha em relação àquela componente.

n		x		x'	
0		0		0	
1		0		1	
2	↑	1		0	↓
3		1		0	

Fig. 3.17 - O caminho inverso

Substituindo na tabela 3.3 a seqüência x = [0, 1, 1], a partir do nível 1, pela seqüência invertida x' = [1, 0, 0] obtemos a trilha t_B = [0162]. A partir daí, "descemos"

na árvore pelo caminho reflexo que é obtido invertendo os valores da componente x da direção e obtemos a trilha t_B = [0162], que é a trilha que leva à célula B.

n	z	y	x'	t_B
0	0	0	0	0
1	0	0	1	1
2	1	1	0	6
3	0	1	0	2

Tab. 3.4 - A componente x

Diz-se então que B é a ***vizinha imediata*** da célula A em relação à componente x. Diz-se também que B é a ***vizinha de face de A em relação a x.***

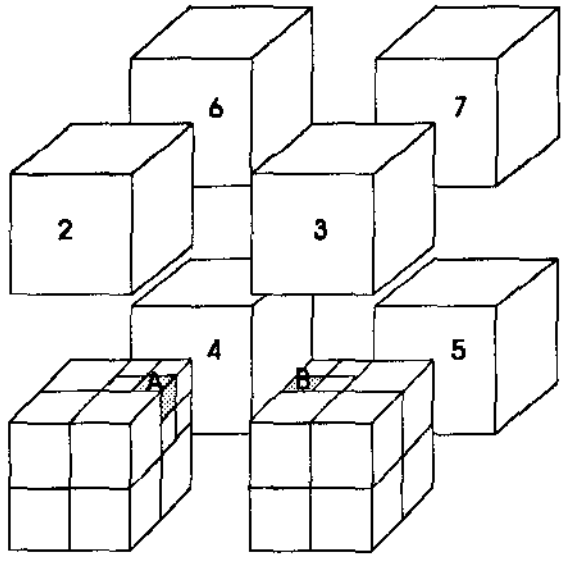


Fig. 3.18 - As células A e B, sua vizinha imediata em relação a x

As células vizinhas imediatas são sempre obtidas através de uma das componentes da direção. Assim, alternando-se as componentes e aplicando o mesmo raciocínio, obtemos as vizinhas imediatas em relação a y e a z.

Suponhamos, num outro caso, que a célula em estudo seja a célula E, onde t_E = [0033] e que se deseja calcular o vizinho imediato em relação à coordenada z. Na Tab. 3.5 é apresentada a trilha t_E, e suas componentes nas direções x, y e z. Note que a célula E foi encontrada através da direção 0 (à frente) em relação à componente z.

n	z	y	x	t_E
0	0	0	0	**0**
1	0	0	0	**0**
2	0	1	1	**3**
3	0	0	1	**3**

Tab. 3.5 - A componente x

Então, se "escalarmos" a árvore a partir do nível 2, procurando por um valor diferente daquele através da qual a célula foi encontrada, chegaremos até a raiz sem tê-lo encontrado.

n		x	
0		0	
1		0	
2	↑	0	
3		0	

Tab. 3.6 - Os valores da componente z

De fato, a célula não tem vizinho imediato em relação à componente *z*. Verifique, na figura abaixo, a posição da célula *E*. A sua vizinha imediata em relação a *z*, se existisse, seria a célula à sua frente. Como ela é uma célula na superfície, então não tem vizinha imediata em relação a *z*, conforme mostrado na figura abaixo, embora tenha em relação a *x* (à direita) e a *y* (acima).

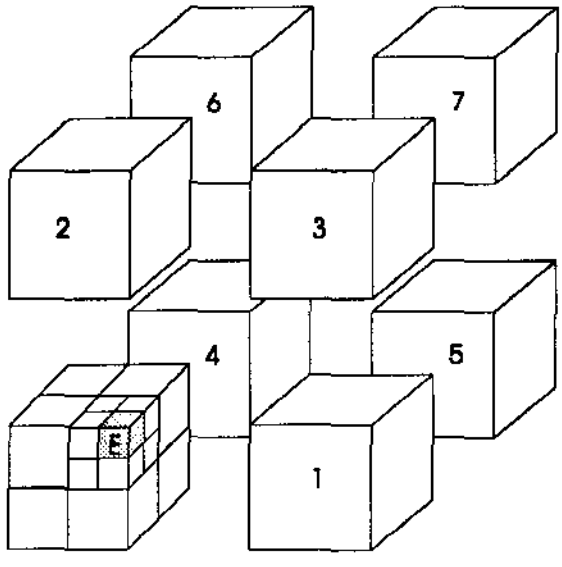


Fig. 3.19 - A célula E

3.5.3 Vizinhas Relativas a uma Face

Na seção anterior vimos como obter uma célula vizinha imediata através de uma dada componente da direção. No exemplo, a célula *B* de trilha [0162] foi obtida através da componente *x* da direção. Mas note na Fig. 3.20 abaixo que não apenas a célula *B* é vizinha de *A*, mas todas as células à esquerda na célula [016] são também vizinhas de *A*.

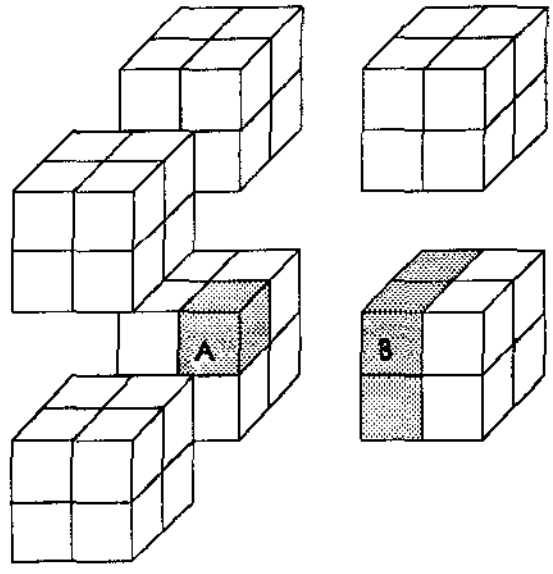


Fig. 3.20 - As vizinhas de A relativas à componente x

São elas as células [0160], [0162], [0164], [0166], lembrando sempre que a nossa referência é a Fig. 3.14. Essas células são obtidas através de todas as combinações das coordenadas *y* e *z* no nível de refinamento a que pertence a célula. No exemplo, o nível é 3.

n	z	y	x	t_A
0	0	0	0	0
1	0	0	1	1
2	1	1	0	6
3	z'	y'	0	7

Tab. 3.7 - z'e y'são as componentes a serem combinadas

As combinações das componentes são mostradas na Tab. 3.8. T_V denota a trilha da célula vizinha. No caso, $T_V[1] = 1$ e $T_V[2]=6$.

z'	y'	x	$t_V[3]$
0	0	0	0
0	1	0	2
1	0	0	4
1	1	0	6

Tab. 3.8 -As combinações das componentes z'e y'

Assim, se tomarmos os valores acima para $T_V[3]$, obteremos as células previamente mencionadas.

De modo análogo, são vizinhas imediatas as células C = [0251] e D = [0037], em relação às componentes y e z da direção; são vizinhas de face em relação à componente y as células [0250], [0251], [0254], e [0255] e em relação à componente z as células [0034], [0035], [0036], [0037], conforme Fig. 3.21.

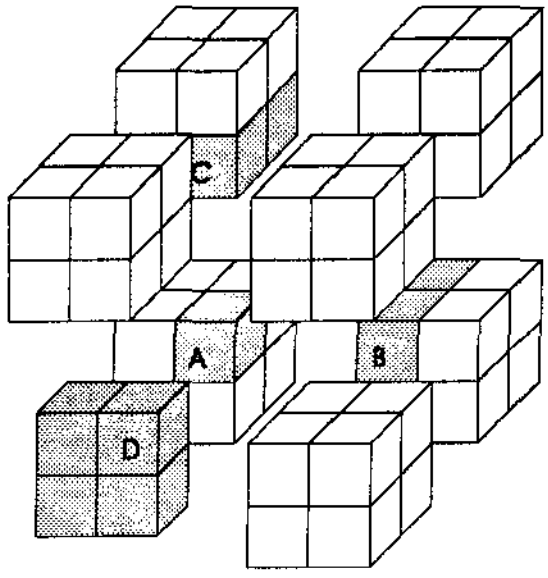


Fig. 3.21 - Vizinhas relativas a uma face da célula A

As células *B*, *C* e *D* são ditas ***células imediatas de face***. Define-se ainda, por ***vizinhas relativas a uma face***, todas as células que podem ser **obtidas** através de uma face. Note que, entre as vizinhas relativas a face, quando elas existem, apenas **uma** delas tem ***face em comum*** com a célula de trabalho: duas tem ***arestas em comum*** e uma tem um ***vértice em comum***. Além disso, uma célula tem no mínimo zero e no máximo doze vizinhas relativas a face.

3.5.4 Vizinhas Relativas a uma Aresta

De modo análogo, por ***vizinhas imediatas de aresta***, define-se as células que têm uma aresta em comum com uma dada célula. Para se obter uma vizinha imediata de aresta,

basta se tomar uma vizinha imediata de uma outra vizinha imediata, usando-se uma componente diferente da direção. Assim, no exemplo da Fig. 3.13, a célula D = [037] foi obtida a partir da célula A pela componente z da direção.

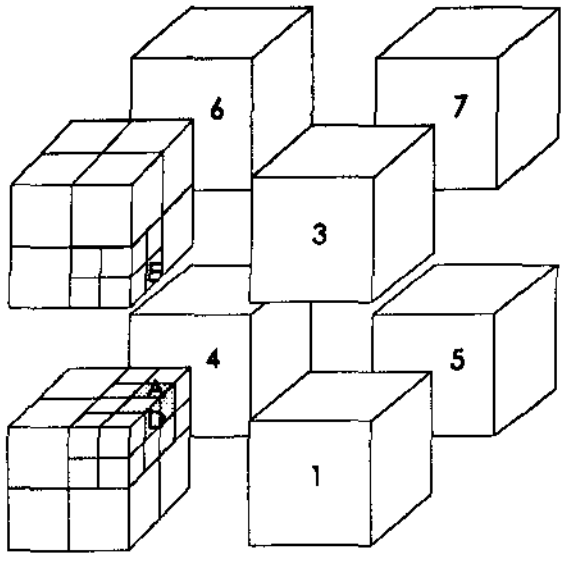


Fig. 3.22 - A Vizinha Imediata de Face de A e suas Vizinhas Imediatas de Face D e de Aresta E

Se calcularmos uma vizinha imediata de B, a partir, por exemplo da componente y, obtemos a célula E = [0215], que é vizinha imediata de aresta da célula A; ou seja, tem uma aresta em comum com a célula A. Note que poderíamos chegar à célula E através da célula C (Fig. 3.21), usando a componente y e depois a componente z. De qualquer maneira, para se chegar da célula A à célula E usam-se as componentes y e z da direção, em qualquer ordem.

Mas da mesma forma que no caso das vizinhas relativas a face, não apenas a célula *D* é vizinha da célula *A*, mas ambas as células inferiores posteriores do octante [021] também o são, conforme Fig. 3.23.

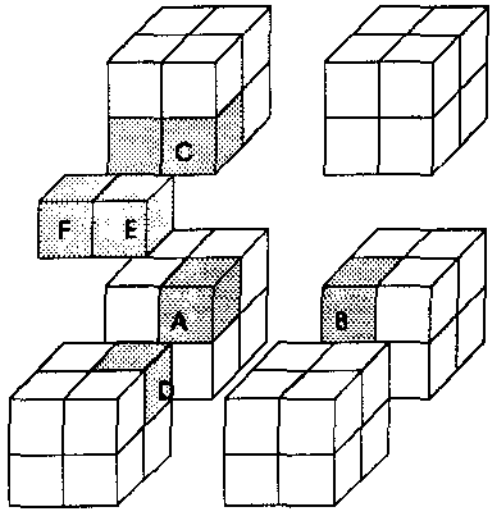


Fig. 3.23 - As Vizinhas Relativas a uma Aresta, E e F

Para se obter a trilha da outra célula, inverte-se no nível da célula (3) o valor da coordenada da componente que não foi usada, no caso a componente *x*, obtendo-se a célula cuja trilha é [0214].

n	*z*	*y*	*x*	t_n
0	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	z'	0	0	?

Tab. 3.9 - z' é a componente a ser combinada

z´	y	x	t_c[3]
0	0	0	0
1	0	0	4

Tab. 3.10 -As combinações das componentes x, y e z´

São também vizinhas imediatas de aresta as células [0126] e [0340], bem como sua irmãs [0124] e [0344]. As vizinhas imediatas de aresta são mostradas na Fig. 3.24.

Define-se portanto ***vizinhas relativas a uma aresta*** todas as células **obtidas** através de uma vizinha de aresta. Uma célula tem no mínimo zero e no máximo seis células vizinhas relativas a aresta.

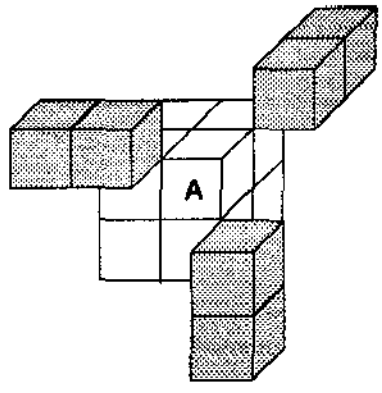


Fig. 3.24 - Vizinhas relativas a aresta

3.5.5 Vizinha Relativa a um Vértice

A *célula vizinha relativa a um vértice* é obtida através das componentes *x*, *y* e *z* da direção e é única quando existe. Dado que para se obter uma célula vizinha imediata de uma aresta usam-se duas componentes, basta portanto calcular uma vizinha imediata de uma vizinha de aresta através da terceira componente. Em outras palavras, se usarmos as componentes *x* e *y* para o cáluclo de uma vizinhas imediata de aresta, usaremos a componente restante *z* para o cálculo da vizinha imediata de vértice. Note que, como as três componentes da direção são usadas no cálculo da vizinha de vértice, qualquer vizinha de aresta pode ser usada para obtê-la. No exemplo, a célula vizinha de aresta da célula A é a célula [0304].

A célula vizinha de vértice tem em comum com a célula pesquisada um único vértice.

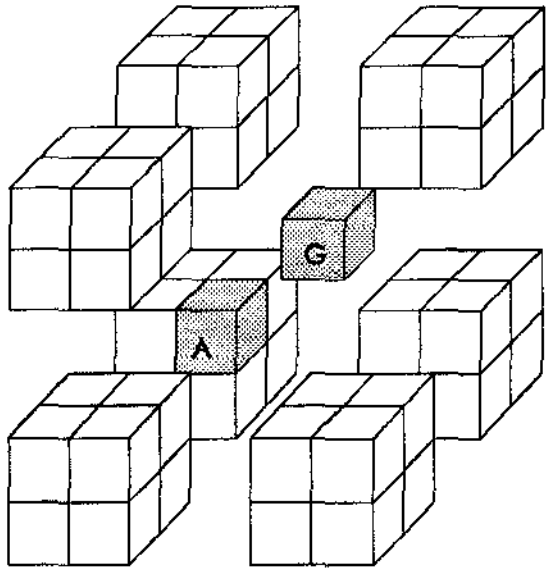


Fig. 3.25 - A célula A e sua vizinha de vértice G

3.5.6 O Total de Vizinhas Obtidas

Os métodos descritos acima nos possibilitam obter 12 células vizinhas de face, 6 vizinhas de aresta e 1 célula vizinha de vértice, completando um total de 19 células vizinhas. Se adicionarmos a este número as 7 células irmãs, obteremos então as 26 células vizinhas que envolvem a célula de trabalho, conforme mencionado na Seção 3.2. Desta forma obtemos o cubo de lado 3 que envolve a célula pesquisada, conforme mostrado abaixo.

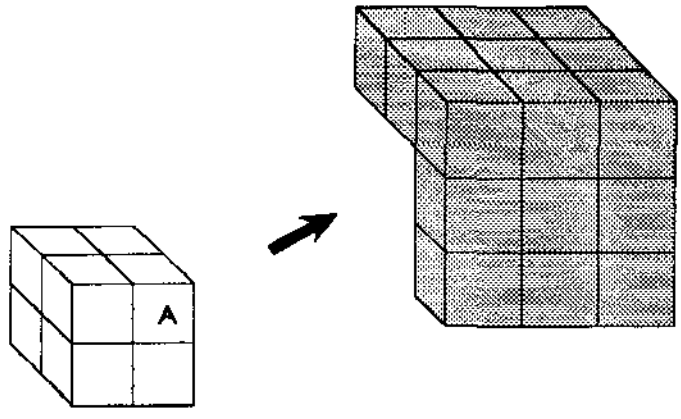


Fig. 3.26 - A célula A, suas irmãs e células vizinhas

3.6 A Pesquisa de Células Vizinhas de Segunda Ordem

Vimos na seção anterior como obter as vizinhas de face, aresta e vértice, que juntamente com as células irmãs compõem o conjunto de células vizinhas de primeira ordem. Mas no algoritmo MMR, são também necessárias as vizinhas de segunda ordem, ou seja, as células

externas do cubo de lado 5 mencionado na Seção 3.2. Para tanto, são necessárias duas etapas, brevemente descritas a seguir.

3.6.1 As Vizinhas de Segunda Ordem Irmãs das Células Vizinhas

Esta etapa é imediata. Usando-se os métodos para obtenção de vizinhas de face, aresta e vértice, basta inserir na lista de células não vizinhas as irmãs destas células que não são vizinhas, pois estão todas à distância de duas células da célula pesquisada. Dessa maneira são obtidas 27 ($4^3 - 3^3$) células vizinhas de segunda ordem, mostradas na Fig. 3.27.

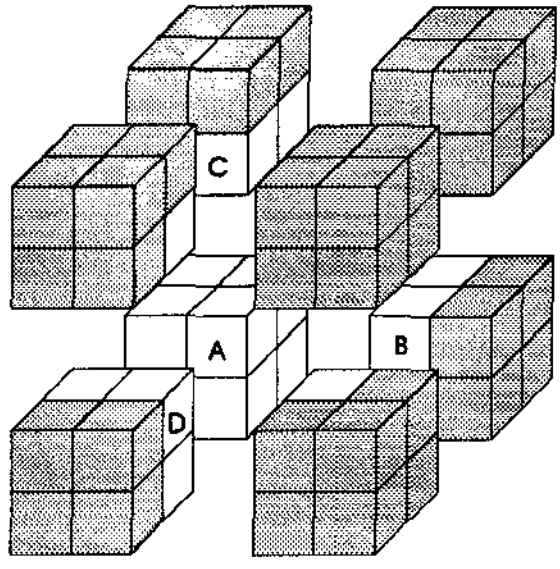


Fig. 3.27 - As irmãs das células vizinhas

3.6.2 Obtenção das Vizinhas de Segunda Ordem Através da Irmã Oposta

É definida como ***irmã oposta*** a célula irmã que tem um único vértice em comum com a célula pesquisada. Em outras palavras, está orientada em sentido oposto em relação às três componentes da direção no cubo que as contém. Na Fig. 3.28 é mostrada a irmã oposta da célula *A*, a célula *Z*, bem como suas vizinhas. No exemplo, a irmã oposta da célula *A* = [073] é a célula *Z* = [074].

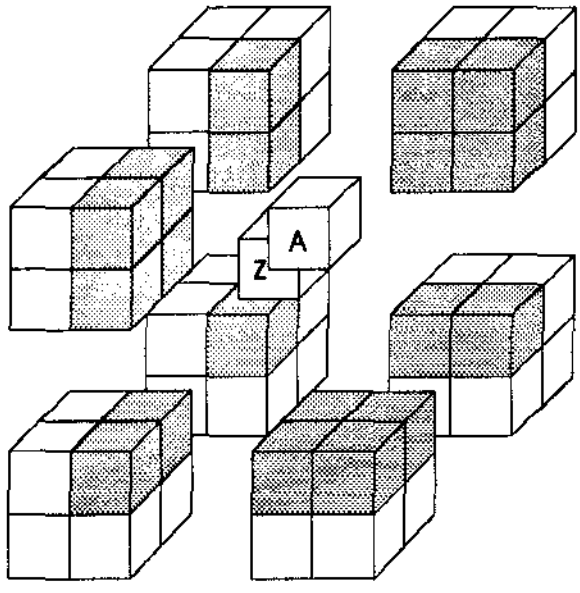


Fig. 3.28 - Células Vizinhas da Célula Oposta

Note no desenho da Fig. 3.29 que estas células não provêem todo o conjunto de células não vizinhas esperado. O objetivo é obter um cubo de lado 5, cujas células externas representam o conjunto das células vizinhas de segunda ordem das células pesquisadas. Para tanto, é preciso calcular o conjunto de ***células adjacentes*** às células recém-obtidas.

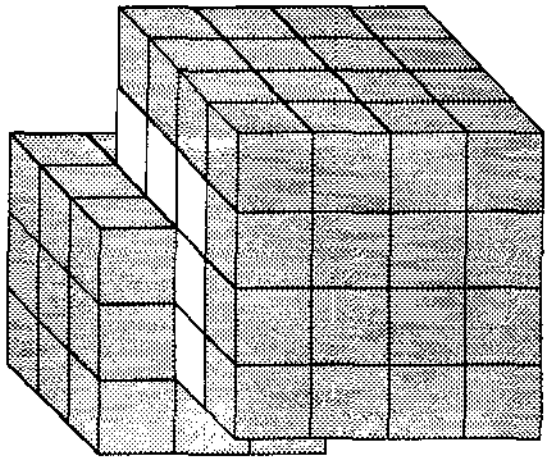


Fig. 3.29 - Células já obtidas

3.6.3 Células Adjacentes

Acima foi mostrado como obter as células vizinhas da irmã oposta de A. Por ***células adjacentes*** define-se um conjunto de células que são adjacentes em relação à face, aresta e vértice às células mencionadas. Na verdade, são as vizinhas de um cubo maior, de dimensão 4, e os métodos para obtê-las são análogos aos já descritos.

É obtido então o cubo de lado 5, cujas células externas são as vizinhas de segunda ordem da célula *A*, conforme ilustrado abaixo.

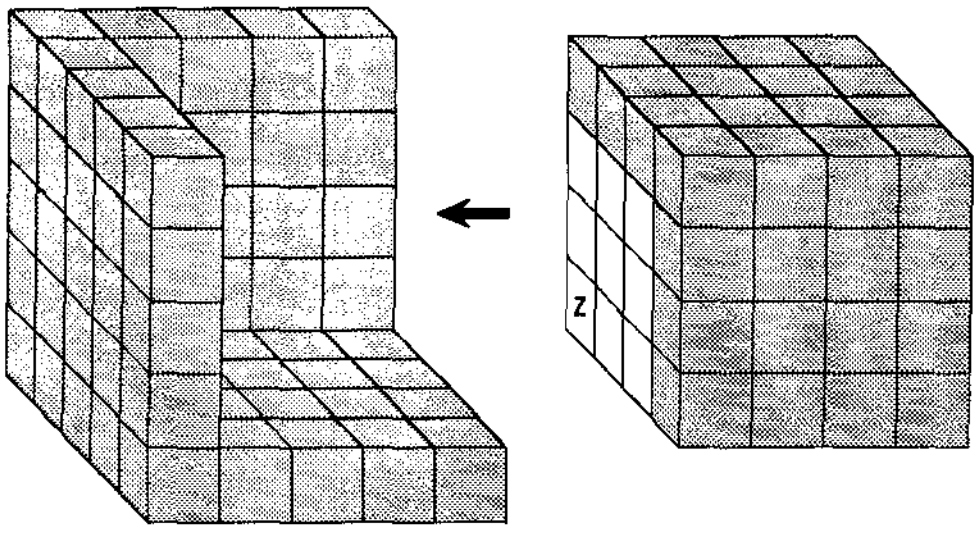


Fig. 3.30 - As Vizinhas de Segunda Ordem de A

Os mesmos esquemas acima podem ser, numa ordem diferente, aplicados para obter os ***não vizinhos***.

A seguir, no Capítulo 4 será descrita a arquitetura interna do sistema, bem como os diagramas das estruturas de dados e os algoritmos usados para o cálculo das expansões no MMR.

Capítulo 4

Descrição do Sistema IsSuP

Neste capítulo será apresentada a descrição interna do sistema **IsSuP** - ***Isopotential Surfaces using Particles***, que é composto de dois módulos separados. O primeiro módulo, preparatório, trata simplesmente da geração das partículas enquanto que o segundo, muito mais complexo, do cálculo das expansões multipólos. As expansões são usadas para a avaliação dos potenciais nas regiões desejadas do espaço computacional, resultando nos arquivos de dados (*slice files*) que serão usados pelo sistema Khoros para visualização dos resultados.

A arquitetura geral do sistema é apresentada na Fig. 4.1.

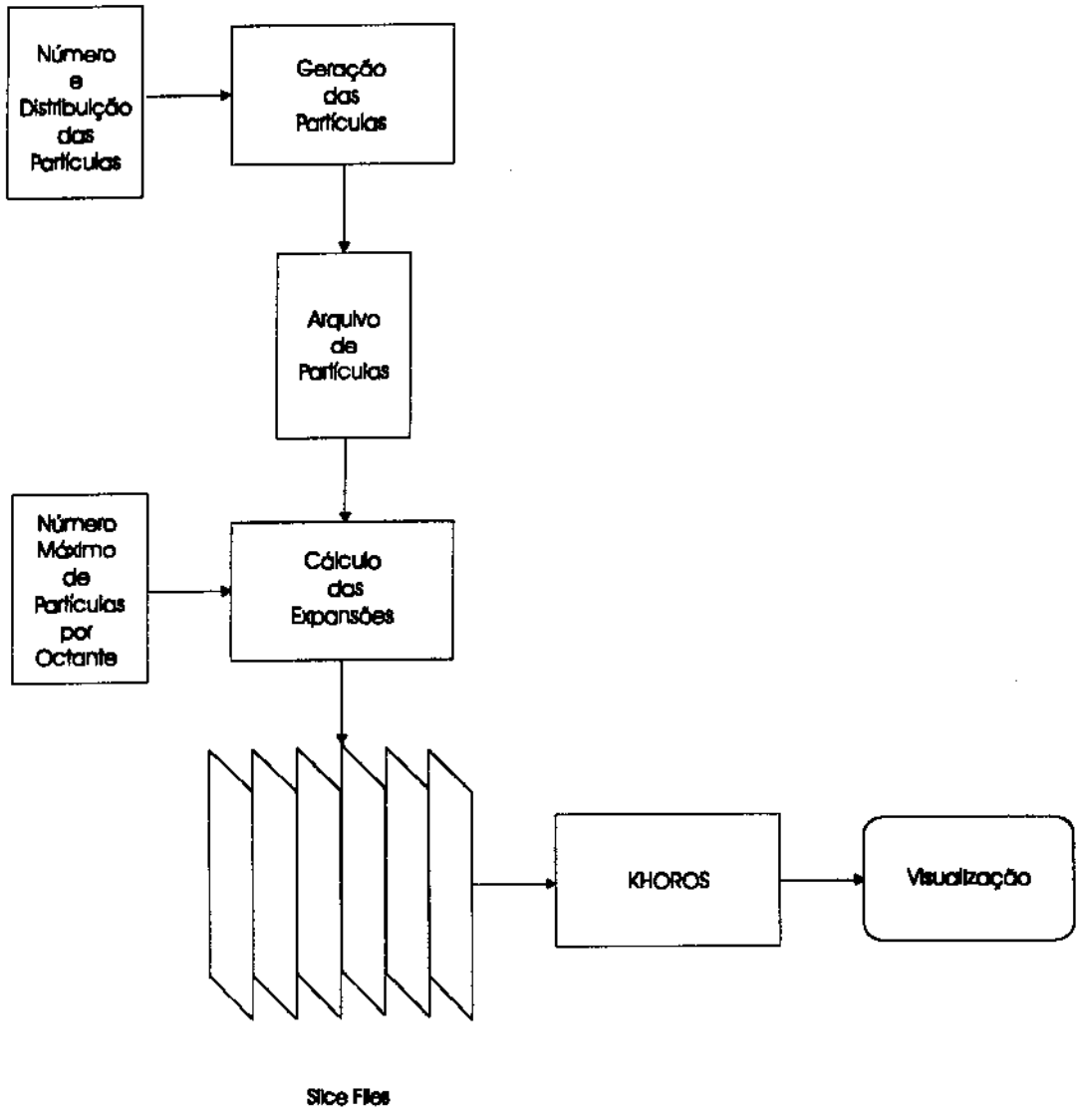


Fig. 4.1 - O sistema IsSuP

4.1 O Módulo de Geração das Partículas

A distribuição das partículas no espaço depende da aplicação a que elas se destinam. As partículas podem estar uniformemente distribuídas em uma caixa computacional, ou sobre uma superfície ou região de forma complicada. Neste sistema, as partículas são cargas elétricas puntuais contidas em um cubo regular de aresta 1. Para tanto, foram geradas randomicamente as coordenadas x, y e z e o campo c no intervalo [0, 1], onde x, y e z são as coordenadas cartesianas de cada partícula no espaço e c a carga da partícula. Variações foram experimentadas alterando-se a distribuição das partículas dentro da caixa computacional.

Entrada

- *Número de partículas (n);*
- *Distribuição das partículas nos octantes;*

Saída

- *Arquivo contendo as coordenadas cartesianas e as cargas das partículas geradas;*

Processamento

- *Gerar randomicamente* x, y, z *e* c n *vezes.*

4.2 O Módulo de Cálculo das Expansões

Este é o módulo principal do sistema, que efetua o cálculo das expansões multipólos e também o mais complexo. Este módulo foi desenvolvido de acordo com o MMR e contém três submódulos principais, associados à fase de criação da estrutura de dados, pré-cálculo e avaliação.

O primeiro submódulo lê o arquivo de partículas e cria a estrutura de dados associada ao espaço computacional necessária para o cálculo das expansões multipólos; o segundo calcula os coeficientes das expansões e o terceiro avalia o valor do potencial nas regiões desejadas. Nessa última fase são gerados os arquivos que serão lidos pelo Khoros para visualização das superfícies. Esse arquivos são chamados de *slice files*. Ainda, dois outros submódulos contém as rotinas necessárias para a avaliação das funções matemáticas e de vizinhança.

A arquitetura geral deste módulo é mostrada na Fig. 4.2 e, a seguir, é descrito em pseudo-código o processamento geral do módulo.

Entrada

- *Arquivo de Partículas*
- *Número máximo de partículas por octante*

Saída

- *Arquivos (Slice Files para geração das imagens)*

Processamento

- *Ler o arquivo de partículas e gerar a octree associada ao espaço computacional;*
- *Calcular os coeficientes das expansões multipólos e local para as subárvores;*
- *Avaliar o valor dos campos potenciais nas regiões desejadas.*

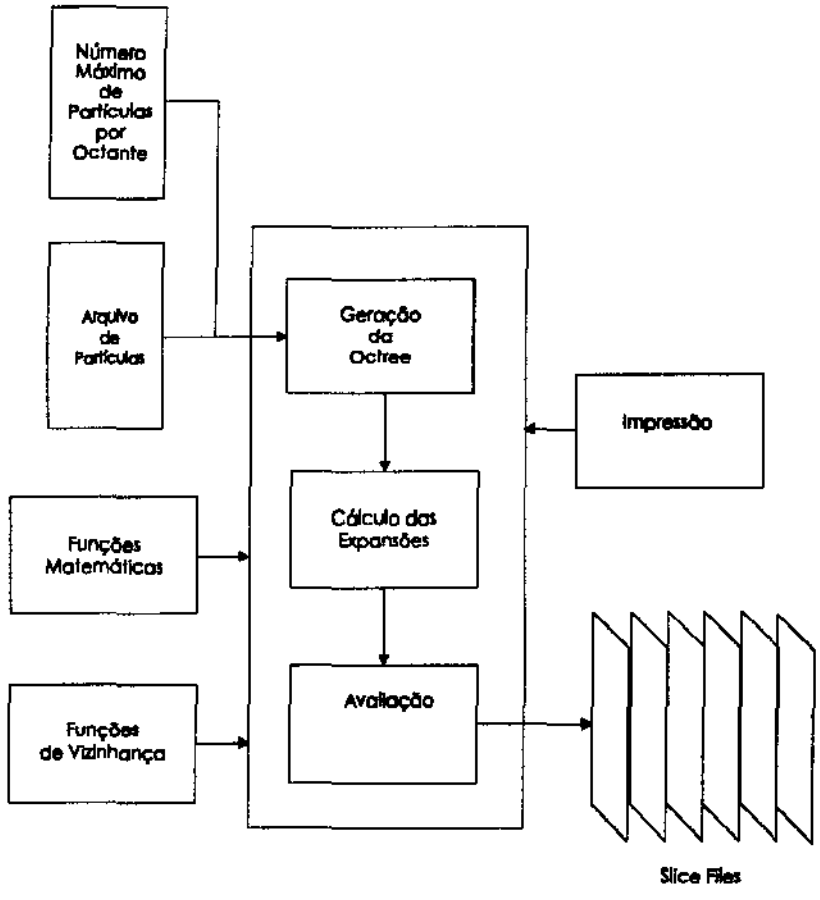


Fig. 4.2 - O Módulo de Cálculo das Expansões

4.2.1 A Geração da Octree

Neste módulo é criada a *octree* que representa o espaço computacional onde estão alocadas as partículas. Inicialmente é criado um *nó raiz* que representa o cubo. Este nó é do tipo *FOLHA*, sua origem tem as coordenadas cartesianas (0, 0, 0) e lados de dimensão 1. O número máximo de partículas por folha é dado por *max_p_folha*. Quando o número de partículas ultrapassa *max_p_folha* o nó é transformado em um nó do tipo *MÃE* e todas as partículas são alocadas nas novas folhas. Ao final deste processo, determina-se a altura da árvore e faz-se a subdivisão de alguns nós de forma que todas as folhas tenham a mesma altura, obtendo assim uma *octree* regular. A descrição de um nó genérico ou célula na octree é dada no Apêndice A.

Entrada

- *Arquivo de Partículas*

Processamento

- *Criar árvore (nó raiz);*
- *Enquanto não for fim de arquivo faça*
  - *Ler partícula;*
  - *Inserir partícula na árvore.*

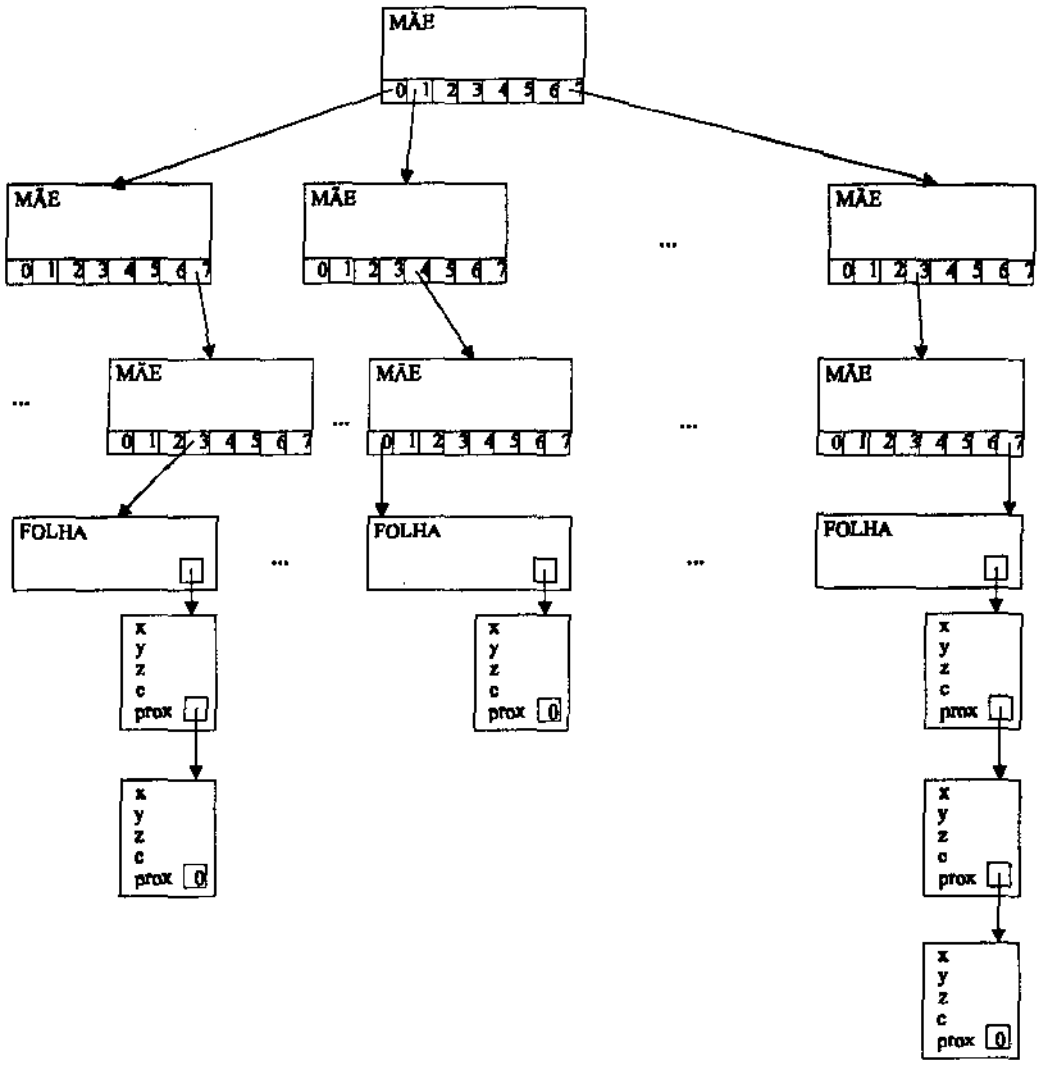


Fig. 4.4 - Diagrama Esquemático da Octree.

4.2.2 O Cálculo das Expansões

Neste módulo são calculados os valores dos coeficientes das expansões multipólos e local, que serão usados posteriormente na avaliação das superfícies isopotenciais. O cálculo é efetuado em duas etapas. Na primeira etapa (*bottom-up*) são calculados os valores das expansões multipólos para cada nó na árvore, e na segunda etapa (*top-down*), os valores das expansões locais.

4.2.2.1 O Cálculo das Expansões Multipólos

Inicialmente, já tendo sido criada a estrutura de dados da octree e alocadas as partículas, são calculados os coeficientes das expansões multipólos em todos os níveis de refinamento.

Preliminarmente, o cálculo das expansões multipólos é feito no sentido folha → raiz (*upward*). A árvore é percorrida até as folhas e os coeficientes Y e M de cada folha são calculados usando a Equação 2.21 e a Equação 2.23.

Após, são avaliados os coeficientes resultantes da translação da expansão, usando-se a Equação 2.26; e os valores são somados aos coeficientes da célula mãe. Isto é feito até se atingir a raiz. Nesse ponto tem-se então calculado o valor dos coeficientes para a avaliação das expansões multipólos para cada célula do espaço computacional.

4.2.2.2 O Cálculo das Expansões Locais

Uma vez obtidos os coeficientes das expansões multipólos para cada célula na árvore, eles são usados para o cálculo das expansões locais. Isto é feito no sentido raiz → folha (*downward*).

A expansão local descreve o campo devido a todas as partículas que não estão contidas na célula corrente nem nos seus vizinhos mais próximos. Conforme dito na Seção 2.5, uma expansão multipólos pode ser transformada em uma expansão local, que será válida em uma esfera com o mesmo raio da expansão multipólos. Conseqüentemente estas duas esferas devem ser bem separadas, i.e., não é permitido que elas se intersectem. Para tanto é usado o conceito de células não-vizinhas. A partir do nível 2 na octree (nível 0 = raiz) são avaliados os coeficientes das expansões locais de cada célula. Para este cálculo são usados os coeficientes das expansões multipólos das células não-vizinhas na Equação 2.31 e somados.

Uma vez calculados os coeficientes da expansão local, eles são usados para calcular os coeficientes das expansões locais das células-filhas, formando a expansão inicial para as células no próximo nível.Nesse ponto, todas as células da octree têm avaliados os coeficientes de suas expansões local e global.

Processamento

- *1a.- Etapa - Upward*
- *para todas as células na árvore*
  - *se a célula é FOLHA calcular o valor dos coeficientes da expansão multipólos usando a Equação 2.23;*

  transladar o valor da expansão multipólos na região da célula para o centro da célula-mãe usando a Equação 2.26;
- *para cada célula a partir do nível 2*
  - *obter as células não-vizinhas da célula;*

- *calcular o valor dos coeficientes L da expansão multipólos usando a Equação 2.31;*
- *se a célula é do tipo MÃE*

  transladar a expansão local para cada uma das filhas da célula usando a Eq. 2.35.

4.2.3 A Avaliação

Neste módulo são avaliados os valores das expansões para cada região do espaço computacional. O espaço é percorrido como num reticulado tridimensional, em relação aos eixos cartesianos *z*, *y* e *x*. Assim, o cubo computacional é fatiado em planos *(x, y)* ao longo do eixo *z*. O números de slices é especificado pelo usuário, e determina a precisão com que o espaço é percorrido.

Em cada ponto do reticulado, é localizada a folha que o contém. O valor da expansão local é avaliado nesse ponto, usando a Equação 2.34, e adicionado ao resultado do cálculo direto com as células vizinhas, calculado pela Equação 2.5, com $k = 1$. Para o cálculo das expansões multipólos e das expansões locais, são usadas as rotinas do módulo de funções matemáticas e do módulo de pesquisa de vizinhança.

Saída

- *Slice files*

Processamento

- *para cada "slice"*

- ◊ *calcular a altura do plano que conterá o slice na caixa computacional;*
- ◊ *para cada ponto do plano*
  - *calcular a folha que contém o ponto;*
  - *calcular as células vizinhas da folha que contém o ponto;*
  - *calcular o valor das expansões locais na região do ponto;*
  - *calcular a lei de Coulomb na região do ponto com as células vizinhas e somar ao valor da expansão local;*
  - *normalizar e gravar o valor no arquivo correspondente.*

Verificou-se, através dos resultados obtidos nos testes, que o maior valor potencial encontrado era sempre menor que *2.5*L_max*, onde *L_max* corresponde ao maior coeficiente *L* obtido. Assim, adotou-se como critério de normalização,

$$ip = int(p/(2.5 * L\_max) * N + 0.5),$$

onde *[0..N-1]* é o intervalo inteiro em que os valores potenciais encontrados serão normalizados e *ip* pertence a este intervalo.

4.3 O Módulo de Pesquisa de Vizinhança

Este módulo implementa os métodos para a obtenção de células vizinhas e não-vizinhas descritos no Capítulo 3. Para a avaliação do MMR é muito importante que tanto a determinação das células vizinhas quanto o acesso a elas na árvore sejam eficientes. Para a avaliação das expansões locais, a partir do nível 2 (nível 0 = raiz), para cada célula são acessadas todas as células não-vizinhas, até o penúltimo nível. Isto, se executado

rigorosamente, pode resultar em até 604 células (9^3 - 5^3). No último nível a informação de todas as células vizinhas são necessárias, o que pode resultar em até 125 células.

Na representação da *octree*, um campo *tr* contém a trilha para se acessar o nó. A trilha é a seqüência de octantes a serem atravessados, a partir da raiz. A trilha é tão longa quanto o nível do nó na árvore. Através do campo *tr* pode-se facilmente determinar se dois nós são "parentes" na *octree* e descobrir o melhor caminho para acessá-lo, a partir da raiz (abordagem *top-down*) ou a partir da célula cujos vizinhos estão sendo avaliados (abordagem *bottom-up*).

Na Fig. 4.5, os caminhos para as células *A*, *B* e *C* são *[0073], [0072] e [0162].* As células *A* e *B* tem em comum o caminho *[07]*, e o menor caminho de *A* até *B* é um nível acima e um nível abaixo, desde que eles têm uma antecessora comum. Mas o menor caminho de *A* até *C* é o caminho a partir da raiz, porque *A* e *C* não estão relacionados.

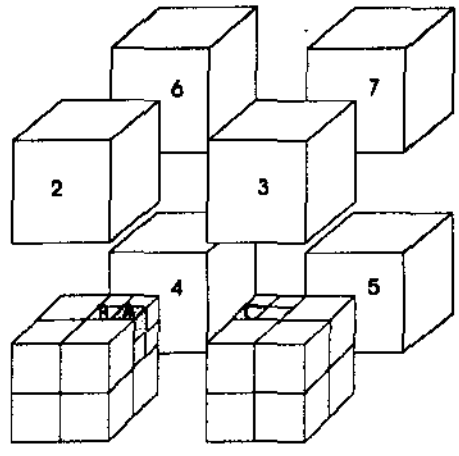


Fig. 4.5 - As células A, B e C na Octree

Esta e outras otimizações foram implementadas. Assim, por exemplo, no procedimento em que se buscam as vizinhas relativas a face (Seção 3.5.2), uma vez determinado o conjunto a ser acessado, a busca na árvore é feita a uma delas e através dela são obtidas as outras, que são suas irmãs.

O procedimento geral para se obter as células vizinhas é o que segue. Note que a definição de células vizinhas refere-se às duas camadas de célula que envolvem a célula de trabalho.

Processamento

- *Em relação à célula cel*
  - ◊ *Obter as células irmãs;*
  - ◊ *Obter as vizinhas de face em relação ao eixo x e suas irmãs;*
  - ◊ *Obter as vizinhas de face em relação ao eixo y e suas irmãs;*
  - ◊ *Obter as vizinhas de face em relação ao eixo z e suas irmãs;*
  - ◊ *Obter as vizinhas de aresta em relação aos eixos x e y e suas irmãs;*
  - ◊ *Obter as vizinhas de aresta em relação aos eixos x e z e suas irmãs;*
  - ◊ *Obter as vizinhas de aresta em relação aos eixos y e z e suas irmãs;*
  - ◊ *Obter a vizinha de vértice em relação aos eixos x, y e z e suas irmãs;*
  - ◊ *Calcular a irmã oposta;*
  - ◊ *Obter as células vizinhas da irmã oposta;*
  - ◊ *Obter as células adjacentes.*

Ao final deste procedimento, obtém-se o cubo de lado cinco mostrado na Fig. 3.1.

O procedimento para a obtenção das células não-vizinhas é análogo ao descrito acima, apenas referente à célula-mãe. Se aplicarmos o mesmo procedimento ao nível da célula mãe, obteremos o resultado descrito na Fig. 4.6. Unindo os conjuntos de células resultantes dos dois procedimentos, teremos um conjunto em que falta obter e outro obtido a mais.

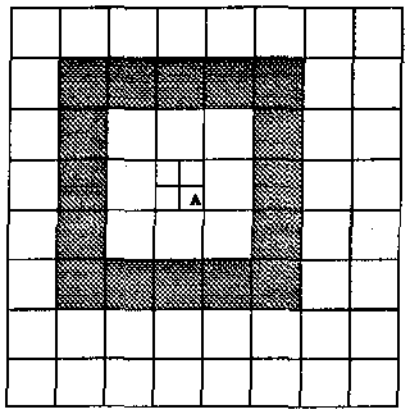


Fig. 4.6 - Células Não Vizinhas ao nível da Célula Mãe num corte da Octree

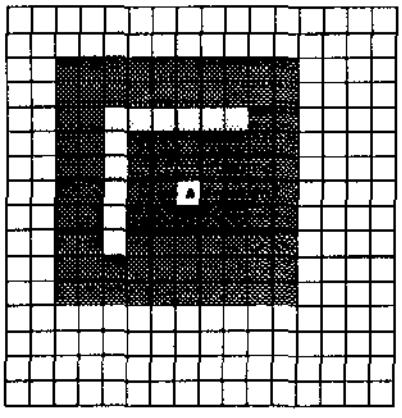


Fig. 4.7 - Células Não Vizinhas Especiais num corte da Octree

Assim, na avaliação da expansão local para uma dada célula, são usadas as expansões locais das filhas das células não-vizinhas da célula-mãe e as não-vizinhas especiais. O procedimento geral para a obtenção das células não-vizinhas é o que segue:

Processamento

- *Em relação à célula cel*
  - ◊ *Em relação à sua célula-mãe*

    Obter as células vizinhas e suas irmãs;

    Obter as não-vizinhas especiais.

4.4 O Módulo de Funções Matemáticas

Este módulo implementa as funções matemáticas necessárias para o cálculo das expansões. Essas funções referem-se às operações básicas com números complexos, não disponíveis na biblioteca standard do C, e função para transformação de coordenadas cartesianas em coordenadas esféricas. São elas:

Operações real, complexo

- *soma*
- *multiplicação*

Operações complexo, complexo

- *soma*
- *multiplicação*
- *divisão*

Funções que tem como argumento números complexos:

- *norma;*
- *potência;*
- *logaritmo;*
- *exponencial;*

As funções potência, logaritmo e exponencial foram implementadas usando os algoritmos 106 [Joh62], 243 [Col64] e 46 [Her61] da ACM, respectivamente. Todas as operações mencionadas acima retornam como resultado números complexos, com exceção da função norma, que retorna um valor real.

Outras funções matemáticas:

- *fatorial;*
- *esférica (transforma coordenadas cartesianas em coordenadas esféricas).*

4.5 O Módulo de Impressão

Contém uma biblioteca de rotinas para impressão de resultados durante a avaliação do sistema.

4.6 O Módulo de Coulomb

Dado um conjunto de partículas, esse módulo calcula os arquivos *slices* usando a lei de Coulomb. Os resultados desse módulo são usados como controle para depuração do programa.

Entrada

- *Arquivo de partículas;*
- *Número de slices;*

Saída

- *Arquivos slices contendo resultados calculados segundo a lei de Coulomb;*

Processamento

- *para cada "slice"*
  - ◊ *calcular a altura do plano que conterá o slice na caixa computacional;*
  - ◊ *para cada ponto do plano*

    calcular o campo potencial na região do ponto de acordo com a lei de Coulomb;

    gravar o *valor no arquivo correspondente.*

4.7 O Algoritmo Adaptativo

O algoritmo MMR pode ser visto como um processo recursivo de subdivisão do espaço computacional em malhas cada vez mais finas. No algoritmo adaptativo não é usado o mesmo número de níveis de refinamento em todo espaço computacional, o que pode resultar em um grande número de células vazias nos níveis mais altos. Ao invés, é fixado um valor inteiro max_p_folha, e a cada nível são subdivididas apenas as caixas que contêm mais do que max_p_folha partículas. A figura 4.9 ilustra numa quadtree uma possibilidade de subdivisão adaptativa para o caso bidimensional.

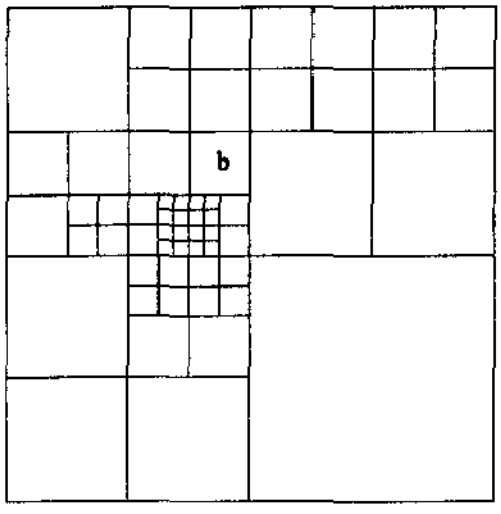


Fig. 4.8 - A Célula b na Quadtree

Nesse trabalho foi adotada a abordagem homogênea. Apesar do sistema ter sido originalmente desenvolvido para o caso adaptativo, não foi localizada a descrição equivalente do tratamento das variantes de vizinhança para o caso tridimensional. Algumas experiências empíricas foram tentadas, mas nesse caso os resultados divergiram grandemente dos obtidos através da lei de Coulomb. Desde que o sistema pode ser facilmente acomodado para um ou outro caso, optou-se nesse protótipo por manter-se a abordagem homogênea.

A seguir, no próximo capítulo, serão apresentados alguns resultados obtidos com o sistema IsSuP.

Capítulo 5

Resultados Obtidos

Os *slice files* gerados pelo sistema IsSuP são lidos como imagens no ambiente Khoros e o volume avaliado usando a V3DTOOLS. biblioteca para visualização e processamento de volumes, que é acessada pelo Cantata, ambiente de programação de linguagem visual.

5.1 Volumes Visualizados

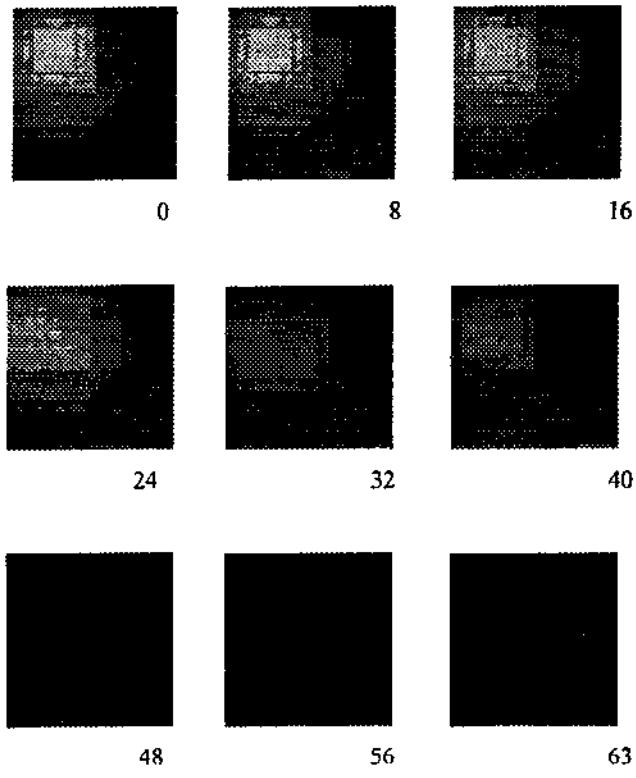
As imagens abaixo mostram uma aproximação às isopotenciais associadas aos valores 3, 4 e 5 no intervalo [0..7]. Os volumes foram gerados usando 64 *slice files*, cada um associado a uma imagem de resolução 64 x 64. As imagens 0, 8, 16, 24, 32, 40, 48, 56 e 63 são mostradas a seguir, e referem-se a um conjunto de 10.000 partículas distribuidas 20% no octante 0 e 80% em todo o espaço computacional.

Fig. 5.1 - As Isosuperfícies >= 3

Fig. 5.2 As Isosuperfícies >= 4

Fig. 5.3 - As Isosuperfícies >= 5

5.2 Imagens associadas aos Slice Files



As imagens acima referem-se aos slice files 0, 8, 16, 24, 32, 40, 48, 53, 63 gerados a partir da avaliação de 10000 partículas. Cada imagem é associada a 64 x 64 valores potenciais.

5.3 Experimento com o Khoros

Os arquivos *slice* lidos como imagens pelo sistema Khoros e o volume avaliado usando as rotinas da biblioteca V3DTOOLS. As imagens a seguir mostram uma aproximação dos valores associados à superficie isopotencial 1, 2, 3, 4 e 5 no intervalo [0..7]. Estas imagens referem-se a um experimento realizado com um conjunto de 10000 partículas distribuidas 40% no octante 0 e 60% no octante 7. Note que as figuras apresentam um cubo aparentemente "corroído". Na Fig. 5.5 os "cantos exteriores corroídos" referem-se às regiões isopotenciais 1 e 0. O canto corroído maior, às regiões 3 e 4. O "centro vazio" do volume à região 5. Neste experimento não fôram encontradas regiões associadas aos valores 6 e 7, o que significa que o valor máximo usado na parametrização pode ser reduzido. As imagens, de baixa resolução, foram gerada usando 64 arquivos *slice*, cada um de resolução (64 x 64). Imagens mais apuradas podem ser obtidas usando-se *slices* de maior resolução.

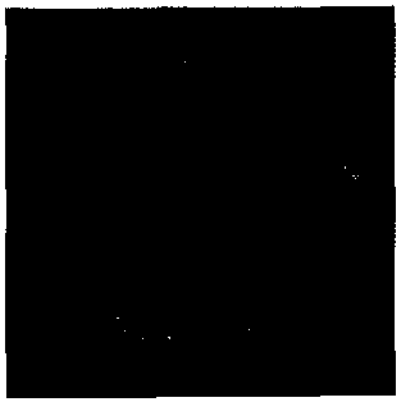


Fig. 5.4 - Isosuperfície = 1

Fig. 5.5 - Isosuperfície = 2

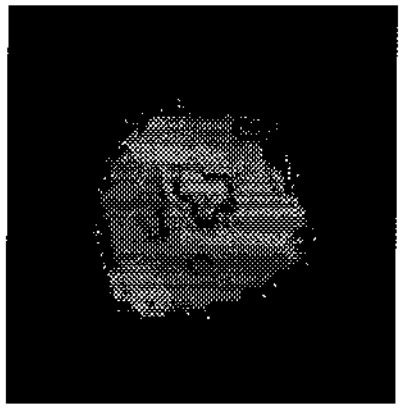


Fig. 5.6 - Isosuperficie = 3

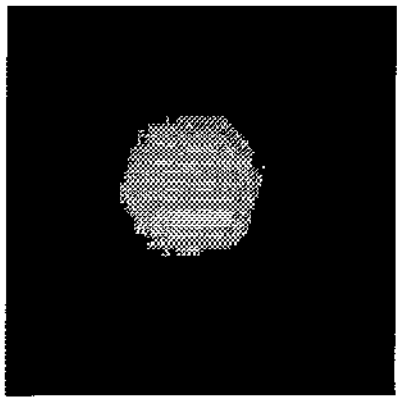


Fig. 5.7 - Isosuperficie = 4

Fig. 5.8 - Isosuperfície = 5

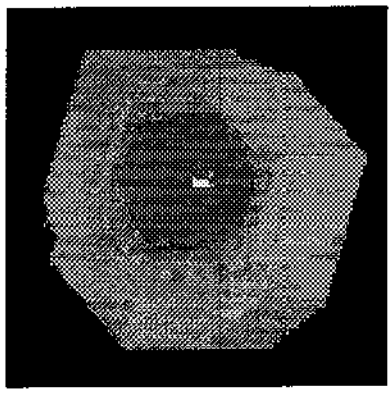


Fig. 5.9 - Diferença entre as Isosuperfícies 2 e 4

5.4 Avaliação

5.4.1 Tempo

O maior arquivo com que se conseguiu testar o Método de Coulomb continha 10^6 partículas. Para arquivos com mais de 10^6 partículas a memória do sistema não era suficiente. Para um arquivo com tal número de dados o tempo de desempenho do MMR foi três vezes menor que o tempo de desempenho do Método de Coulomb.

Método	*Tempo (Hz)*
MMR	65779
Coulomb	205196

Tab. 5.1 - Comparação entre os tempos para arquivo com 10^6 partículas

5.4.2 Resultados Numéricos

A seguir são apresentados alguns resultados associados ao arquivo acima mencionado, de 10^6 partículas. Cada uma das tabelas mostra os valores calculados nas regiões dos pontos de coordenadas (x, y, z), pelos métodos MMR e de Coulomb.

Resultados

x	0.0	0.33	0.66	1.0
y	0.0	0.0	0.0	0.0
z	0.0	0.0	0.0	0.0
MMR	7.091682e+05	8.112706e+05	8.105104e+05	7.096177e+05
Coulomb	5.947848e+05	7.014420e+05	7.007734e+05	5.952618e+05

Tab. 5.2 - y = 0.0 e z = 0.0

x	0.0	0.33	0.66	1.0
y	0.33	0.33	0.33	0.33
z	0.33	0.33	0.33	0.33
MMR	9.122867e+05	1.059659e+06	1.061908e+06	9.119852e+05
Coulomb	8.590520e+05	1.103179e+06	1.105277e+06	8.589769e+05

Tab. 5.3 - y = 0.33 e z = 0.33

Resultados

x	0.0	0.33	0.66	1.0
y	0.66	0.66	0.66	0.66
z	0.66	0.66	0.66	0.66
MMR	9.113383e+05	1.061091e+06	1.061224e+06	9.114696e+05
Coulomb	8.580571e+05	1.104775e+06	1.104660e+06	8.585612e+05

Tab. 5.4 - y = 0.66 e z = 0.66

x	0.0	0.33	0.66	1.0
y	1.0	1.0	1.0	1.0
z	1.0	1.0	1.0	1.0
MMR	7.096389e+05	8.100473e+05	8.102028e+05	7.089438e+05
Coulomb	5.951390e+05	7.003327e+05	6.998679e+05	5.946036e+05

Tab. 5.5 - y = 0.33 e z = 0.33

Capítulo 6

Conclusão

A avaliação de interações gravitacionais ou por Coulomb em sistemas de partículas de larga escala é uma parte integral da simulação numérica em diversos processos. Em muitas situações, para ser de interesse físico, a simulação deve envolver milhares de partículas (ou mais), e os campos têm que ser avaliados para variadas configurações. O algoritmo descrito por Greengard para avaliação rápida do potencial e campos de força para sistemas de partículas de larga escala, o Método Multipólos Rápido (*Fast Multipole Method*), requer uma quantidade de trabalho proporcional a N dentro de um erro aproximado. Tanto a construção para a versão bidimensional quanto para a tridimensional foram descritas.

Müller e Müller descreveram em seu trabalho *An Object-Oriented Implementation of the Fast Multipole Method* uma implementação do Método Multipólos Rápido (MMR) para o caso bidimensional. A principal motivação deste trabalho foi a implementação do MMR para o caso tridimensional, resultando no sistema IsSuP.

IsSuP é um sistema flexível para a avaliação de superfícies isopotenciais geradas por partículas elétricas no espaço. Essa flexibilidade é obtida a partir dos algoritmos de pesquisa de vizinhança em octrees, aspecto extremamente importante no MMR, apresentados em [Mor92], e também contribuição deste trabalho. Pode-se considerar ainda como resultado, a Biblioteca de Rotinas para Pesquisa de Vizinhança em octrees, genérica e extensível. O sistema IsSuP permite a escolha do número de partículas por nó, reduzindo e aumentando a profundidade da árvore e o tempo de cálculo conforme desejado. Resultados mais refinados podem ser obtidos adequando-se parâmetros do sistema à precisão desejada, o que permite uma avaliação mais rápida ou mais apurada. Experimentos foram realizados comparando-se os resultados apresentados pelo sistema com resultados obtidos usando-se o Método de Coulomb, obtendo-se boa proximidade [$Mor95_a$, $Mor\ 95_b$]. Uma comparação dos tempos de desempenho entre os dois métodos apresentou um tempo três vezes menor com a aplicação do MMR para arquivo com 10^6 partículas.

Os resultados apresentados no Capítulo 5, a visualização das isosuperfícies aproximadas, foram obtidas através da biblioteca de rotinas do sistema Khoros que permite variadas operações sobre os volumes e imagens resultantes, tais como rotação e subtração das isosuperfícies ou animação das imagens *(slice files)*. Exemplos são mostrados, bem como os resultados numéricos obtidos pela avaliação dos campos potenciais usando-se Coulomb e o MMR para diferentes distribuições de partículas.

Uma continuação natural deste trabalho seria, numa primeira etapa, reprogramar o sistema numa abordagem orientada para objetos, e a seguir a adaptação do sistema IsSuP para um supercomputador. Devido à arquitetura da estrutura de dados que suporta

o sistema, a octree, diferentes conjuntos de partículas associados aos sub-nós da árvore poderiam ser paralelamente calculados em diferentes processadores. No MMR, o resultado final da avaliação é calculado em relação aos resultados parciais dos sub-nós. Devido à limitação dos equipamentos disponíveis, de memória e de utilização, não foi possível calcular imagens de melhor precisão.

Apêndice A

```
typedef struct no
  {
   struct no *ant;
   COORD origem, centro;
   char tipo_no;
   int niv, num_part;
   TRILHA tr;
   COORD Mi[NUM_TERMOS + 1][NUM_TERMOS + 1] ,
   L[NUM_TERMOS + 1][NUM_TERMOS + 1];
   union tipo_no_u
      {
       NO_MAE nm;
       NO_FOLHA nf;
      } no_u;
   } NO, *AP_NO;
onde
typedef struct no_folha
  {
   AP_LISTA_PARTICULAS l_particulas;
  } NO_FOLHA, *AP_NO_FOLHA;
typedef struct  lista_particulas
  {
   PARTICULA part;
   struct lista_particulas *prox;
```

```
} LISTA_PARTICULAS, *AP_LISTA_PARTICULAS;

typedef struct no_mae
  {
   struct no *l_filhas[NUM_FILHAS];
  }
```

com:

ant = apontador para a célula antecessora;

origem = origem da caixa em coordenadas cartesianas;

centro = centro da caixa em coordenadas polares;

tipo_no = indica se o nó é do tipo FOLHA (0) ou MÃE (1);

niv = nível do nó;

num_part = número de partículas na região abrangida pelo nó;

tr = trilha da raiz até a caixa;

Mi = valor dos coeficientes da expansão multipólos no centro da caixa;

L = valor dos coeficientes da expansão local no centro da caixa;

tipo_no_u = contém o conjunto de campos específico de cada tipo de nó.

No caso do nó ser do tipo FOLHA (0), o único subcampo é o campo:

l_particulas = contém a lista de particulas localizadas na folha.

O nó do tipo MÃE (1) contém o subcampo:

l_filhas[NUM_FILHAS] : vetor onde cada elemento aponta para uma sub-árvore da árvore.

Bibliografia

[Bar92]

Barzel, R. *Physically-Based Modeling for Computer Graphics*. Academic Press, Inc., 1992.

[Her61]

Herndon, J. R. Exponential of a Complex Number. *Comm. ACM 4*, pg. 178, Abr. 1961.

[Joh62]

Johnson, M. L. & Sangren, W. Complex Number to a Real Power. *Comm. ACM 5*, Jul. 1962.

[Col64]

Collens, D. S. Logarithm of a Complex Number. *Comm. ACM 7*, pg. 660, Nov. 1964.

[Ear92]

Earnshaw, R. A. & Wiseman, N. *An Introductory Guide to Scientific Visualization*. Springer- Verlag, 1992.

[Ear94]

Earnshaw, R. A. et al. *Scientific Visualization*. Academic Press, 1994.

[Fol96]

Foley, J. D., A. van Dam, S.K.Feiner & J. F. Hughes. *Computer Graphics Principles and Practice*. Addison-Wesley Publishing Company, 1996.

[Gre89]

Greengard, L. *The Rapid Evaluation of Potential Fields in Particle Systems*. The MIT Press - Cambridge, Massachussets - London, England, 1989.

[Hay67]

Hayt Jr., William H. *Engineering Electromagnetics*. International Student Edition - McGraw-Hill, Inc. Library of Congress Catalog Card Number 66-24475, 1967.

[Mor85]

Mortenson, M. E. *Geometric Modelling*. John Wiley & Sons, 1985.

[Mor92]

Moroni, A., Magalhães, L. P. & Lucchesi, C. L. Modelagem Baseada em Sistemas Físicos: Pesquisa de Vizinhança em Octrees para Implementação do Método Multipólos Rápido. *Anais do SIBGRAPI V* (SBC - INPE, Brasil), pg. 115-124, 1992

[Mor95$_a$]

Moroni, A., Lucchesi, C. L. & Magalhães, L. P. Modelling 3D-Isosurfaces using Particle Systems. *GRAPHICOM'95 - Tutorials and Main Program Reports* (GRAFO Computer Graphics Society - State Education Center - St. Petersburg), Vol. 1 - pg. 193-197, 1995.

[Mor95b]

Moroni, A., Magalhães, L. P. & Lucchesi, C. L. IsSuP - Modelagem e Visualização de Superfícies Isopotenciais. *Anais do SIBGRAPI VIII* (SBC - UFSCar, Brasil) pg. 301-302, 1995.

[Mül90]

Müller, L. & Müller, W. *An Object-Oriented Implementation of the Fast Multipole Method.* Fraunhofer Gesellschaft - Arbetsgruppe Graphisque Datenverarbeitung (Wilhelminenstr, 7 - D-6100 Darmstadt - West Germany) Report No: TR-90030, 1990.

[Ree83]

Reeves, W. T. Particle Systems - A Technique for Modeling a Class of Fuzzy Objects. *SIGGRAPH 83 Computer Graphics* - Vol. 17, Number 3, Jul., 1983.

[Pre73]

Preparata & Yeh. *Introduction to Discrete Structures*. Addison-Wesley Publishing Company, 1973.

[Sam84]

Samet, H. The Quadtree and related Hierarchical Data Structures. *ACM Computer Surveys* 16, No. 2, pg. 187-260, 1984.

[Sam89]

Samet, H. Neighbor Finding in Images Represented by Octrees. *Computer Vision, Graphics and Image Processing*, 46, pg. 367-386, 1989.

[Sam90]
Samet, H. *Applications of Spatial Data Structures*, Addison-Wesley Publishing Company, 1990.