IMPLEMENTAÇÃO DE UM ALGORITMO VETORIZADO PARA A GERAÇÃO DE SUPERFÍCIES TRIDIMENSIONAIS

André Luiz Battaiola

INPE - Instituto Nacional de Pesquisas Espaciais
Caixa Postal 515
12201 São José dos Campos, SP, Brasil
andre@dpi.inpe.br

Abstract. Nowadays, with the great increasing of the computers processing power, weather forecast could be possible and it's very important for the comunity. Normally, the meteorologist works with great data sets, so the graphics representation of these data associated with satelite or radar images are important resources to study the climate behavior. In that way, systems of meteorological data visualization have proved that they are important tools for the meteorologists. The author, based in the knowledge obtained working in the INPE project to implement the system of meteorological data visualization MicroMAGICS and in the reasearch work developed in the Space Science and Engineering Centre in Madison USA, where he studied techniques to generate tridimensional graphics, analyse in this paper an algorithm to generate tridimensional surfaces that can be used to visualize many phisical fields, in special, the fields related to the climate.

Introdução

Em sistemas do tipo CAD ou CAE, a geração de um gráfico é feita de forma interativa pelo usuário que já tem uma concepção pré-estabelecida deste gráfico. Ao contrário, nos sistemas de visualização de dados científicos, normalmente não se conhece a priori a forma do gráfico que será gerado, pois ela é feita com base em medidas de uma determinada grandeza física. Assim, normalmente não se conhece a forma de um campo elétrico antes da sua geração.

A geração de uma superfície tridimensional com base em medidas de uma grandeza física é um recurso muito util em várias áreas de atuação, como por exemplo, meteorologia [Hibbard-Santek (1988)], [Hibbard-Santek (1989-D)], geologia, hidrologia [Hibbard-Santek (1989-A)], medicina [Lorensen et al. (1987)], etc. Em todas estas áreas, normalmente se dispõe de dados medidos no espaço e se deseja obter um gráfico da sua variação.

O exemplo mais simples para ilustrar a aplicação deste recurso é a geração de uma nuvem. Vamos imaginar que uma grade tridimensional envolve uma nuvem no espaço e que a cada ponto da grade foi associado um valor relativo de umidade. Assumindo-se que um ponto dentro da nuvem deve possuir valor de 85% de umidade, é facil de se determinar os pontos dentro e fora da nuvem. Analisando-se a configuração dos pontos que definem os cubos da grade, o algoritmo dos cubos marchantes [Lorensen et al. (1987)] permite a geração da superfície que delimita a nuvem.

Observe-se que este algoritmo nao é adequado para trabalhar com várias grandezas físicas associadas simultaneamente a um ponto de grade, ou seja, o algoritmo nao é capaz de produzir uma única superfície que represente, por exemplo, o comportamento de pressão, temperatura e umidade.

No SSEC, os pesquisadores William L. Hibbard e David Santek desenvolveram o sistema tridimensional de visualização de dados meteorológicos VIS-5D [Hibbard-Santek (1990)], [Hibbard (1986)], [Hibbard-Santek (1989-B)], [Hibbard-Santek (1989-C)]. Este sistema utiliza o método dos cubos marchantes para a geração de superfícies. Como o gerador de superfícies é um módulo básico do sistema VIS-5D, é facil de se entender a importância de um rápido algoritmo de cubos marchantes em um sistema altamente interativo como este. O cerne deste trabalho é o desenvolvimento deste algoritmo rápido.

Descrição Sucinta do Algoritmo dos Cubos Marchantes

Para melhor explicar o funcionamento deste algoritmo, vamos exemplificar (fig. 1) uma possível configuração de um cubo de uma grade tridimensional que contém um volume qualquer. Cada vértice do cubo foi enumerado e os marcados estão fora do volume. Examinando-se os vértices dentro e fora do volume, pode se definir um polígono que se aproxime da forma da superfície do volume dentro deste cubo.

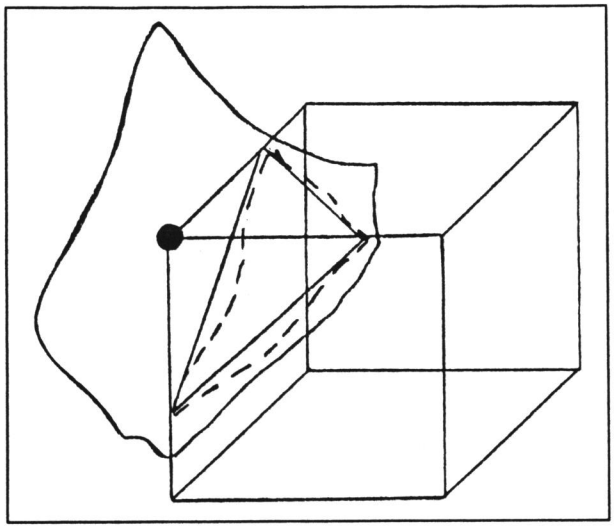


Figura 1 - Polígono que representa uma porção de uma superfície.

Como um cubo tem oito vértices, existem 256 configurações para os seus vértices. No entanto, eliminando-se configurações repetidas através de rotações as 256 configurações se reduzem a 30 configurações básicas. Para cada configuração básica são associados polígonos que melhor representam a superfície do volume dentro do cubo (fig. 2).

Caso não se diferencie as configurações negativas (fig. 3), os casos básicos se reduzem a 15. No entanto, é possível se utilizar o produto vetorial para o cálculo das normais. Neste caso, de forma a facilitar o cálculo do sentido do vetor normal a superfície é melhor se manter a diferenciação entre os dois tipos de configurações.

Analisando-se cada uma das 256 configurações do cubo, pode se construir uma tabela na qual se define o formato dos polígonos para cada uma das configurações. Esta tabela de configurações (**TC**) será a base para o cálculo dos polígonos que concatenados definem a superfície.

A forma de concatenar os polígonos será abordada adiante. No entanto, uma forma simples porém, dependendo do pacote gráfico que se estiver utilizando, não muito eficiente de se gerar a superfície é a cada polígono determinado, dividí-lo em triângulos, calcular as normais aos triângulos e com ela calcular os parâmetros (cor, sombreamento, visibilidade, etc) que definem a imagem do triângulo e desenha-lo de imediato.

Dois Tipos de Abordagem para o Algoritmo

Trabalhar com a TC com os polígonos já decompostos em triângulos ou não, são duas formas de abordar o algoritmo que no ambiente em que foi desenvolvido o sistema fazem uma diferença importante na eficiência e nos resultados do programa implementado.

Certamente trabalhar com triângulos parece inicialmente mais interessante e pode-se citar no mínimo uma razão para isto: é muito mais simples se manipular um triângulo, que é o polígono básico, do que polígonos de número de lados genéricos.

A divisão dos polígonos em triângulos traz algumas ambiguidades, pois não existe apenas uma maneira de se dividir em triângulos um polígono com número de lados maior do que 3. No entanto, isto não chega a comprometer a qualidade dos gráficos gerados. Note-se que o próprio algoritmo já contém ambiguidades que aparecem quando se vai conectar os triângulos para a geração da superfície. Algumas destas ambiguidades podem ser tratadas como casos especiais, como exemplificado adiante.

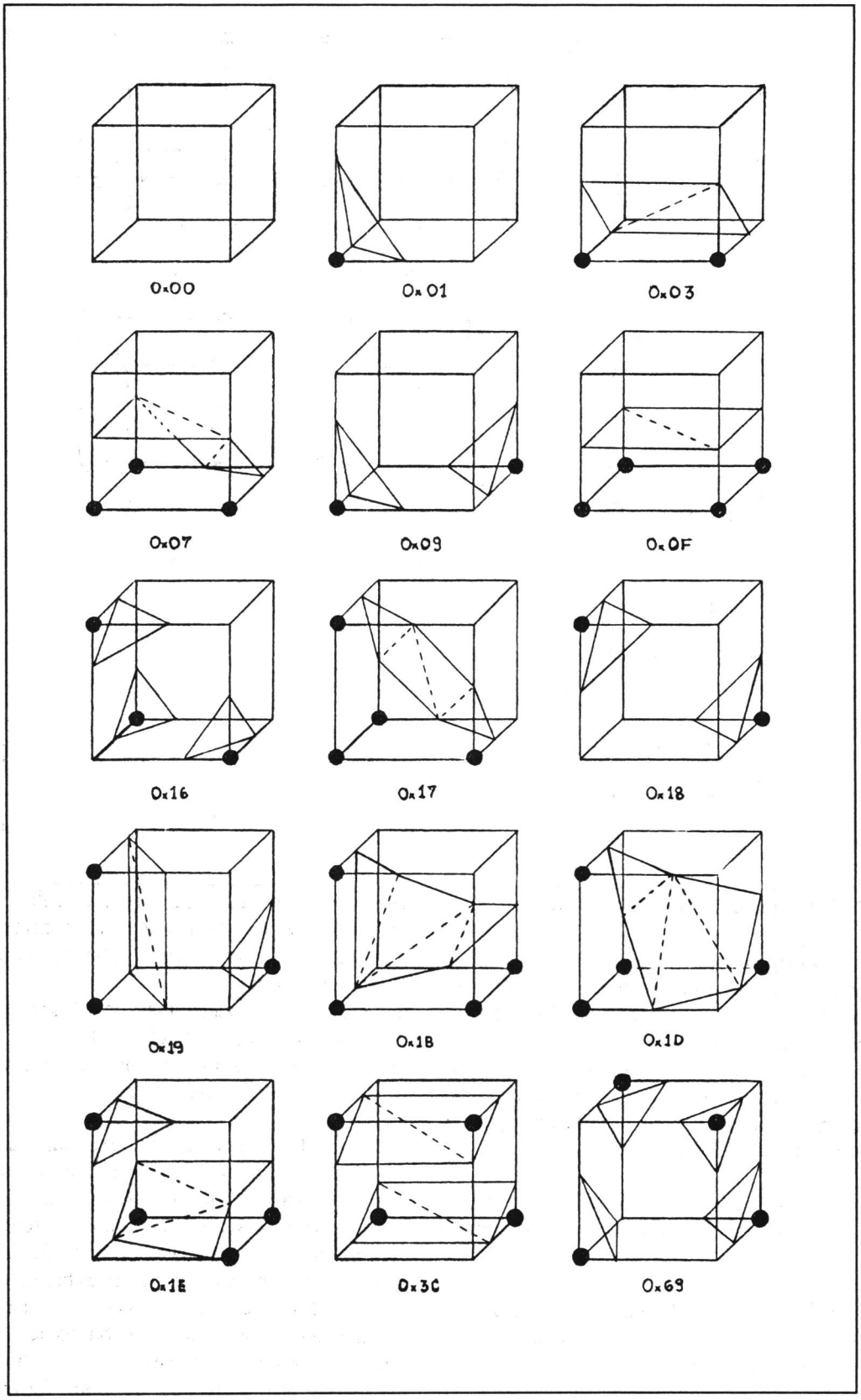
O fator que influência na forma de se definir a TC é o pacote gráfico utilizado na implementação do algoritmo. No caso, utilizou-se o XFDI [XFDI] que requer que toda superfície a ser gerada esteja definida como uma malha de triângulos. Para se produzi-la é melhor, do ponto de vista de eficiência, partir-se de polígonos não divididos a priori em triângulos.

Caso nao se trabalhe com malha de triângulos, ou seja, caso se processe um triângulo por vez, é melhor se utilizar a TC já dividida em triângulos. Observe que o algoritmo foi implementado com os dois tipos de abordagem.

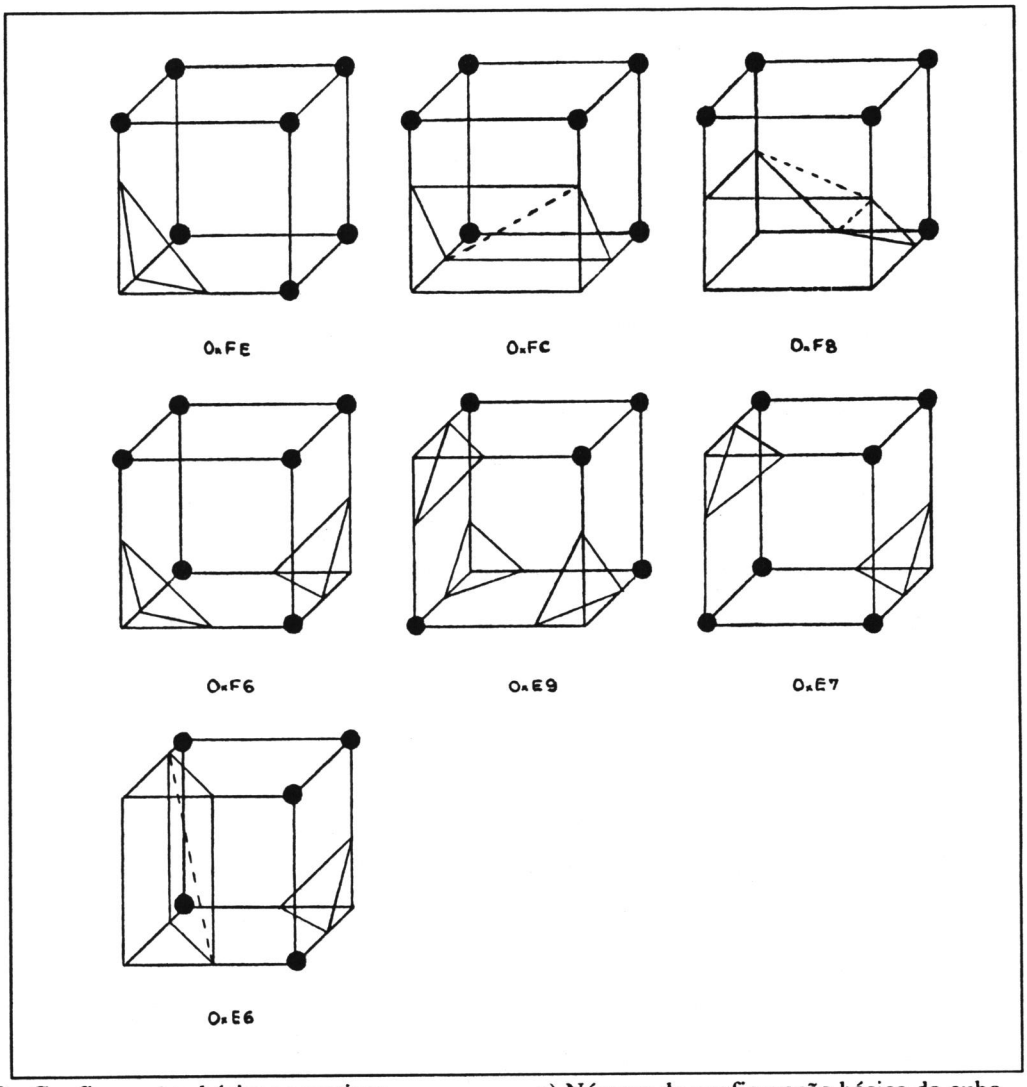
Implementação

O algoritmo dos cubos marchantes foi implementado de acordo com o esboço abaixo. Cada um de seus módulos serão discutidos nos próximos itens. Observe-se que este algoritmo gera polígonos que serão divididos em triângulos na fase de concatenação.

```
CubosMarchantes ( )
{  Le_Tabela_de_Configurações( );
   Calcula_Configuração_de_Todos_Cubos( );
   Calcula_Vertices-Poligonos_de_Cada_Cubo( );
   Calcula_Normais( );
   Concatena_Poligonos( );
}
```



Fígura 2 - Configurações básicas.



Fígura 3 - Configurações básicas negativas.

A Tabela de Configurações

Inicialmente será discutido neste item a forma de geração da TC. Ela é gerada em dois passos. Primeiramente, analisa-se cada uma das 256 configurações e identifica-se a configuração básica em que cada cubo se enquadra e a forma de rotação capaz de colocá-lo em sua configuração básica. Os seguintes dados estão na Tabela de Rotações (**TR**):

a) Numero da configuração básica do vértice do cubo.
b) Vértices do cubo rotacionados para a configuração básica.

Observe-se que esta tabela não é utilizada pelo algoritmo, ela é apenas a base para a construção da TC. A TC contém os seguintes dados:

a) Número da configuração básica do cubo.
b) Número de polígonos para a dada configuração.
c) Número de lados de cada polígono da configuração em hexadecimal.
d) Índices em hexadecimal dos lados do cubo que contém vértices do(s) polígono(s).
e) Índices dos lados do cubo que identificam a sequência de vértices do(s) polígono(s).

Como eficiência é sempre o requisito fundamental, a TC deve ser definida de forma a diminuir ao máximo o processamento na rotina. Assim, ao invés de se trabalhar com os vértices do cubo, se preferiu trabalhar com os lados. A cada lado do cubo foi associado um índice (fig. 4). A vantagem de se trabalhar com os lados do cubo é que ao se processar um cubo, verifica-se inicialmente quais lados do cubo contém vértices dos polígonos, posteriormente estes vértices são calculados e associados aos polígonos de uma forma simples e rápida. Este assunto não será detalhado neste trabalho.

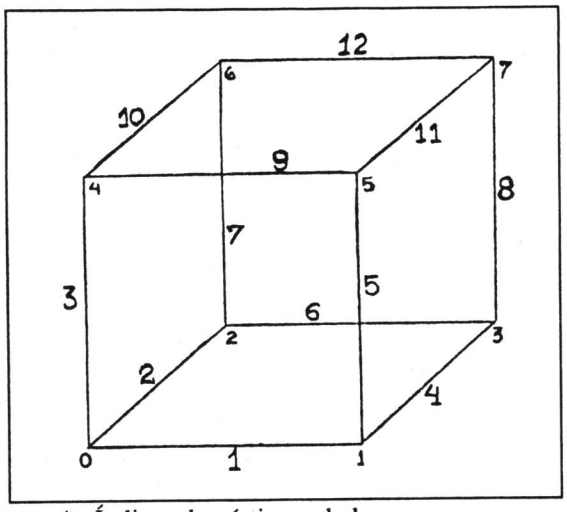


Figura 4 - Índices de vértices e lados.

Observe-se que a TC é gerada por um programa aparte que é basicamente um laço de interações que analisa cada configuração de cubo de acordo com a sua configuração básica. Cada dois vértices do cubo que definem um lado são rotacionados e em seguida calcula-se o índice do lado rotacionado.

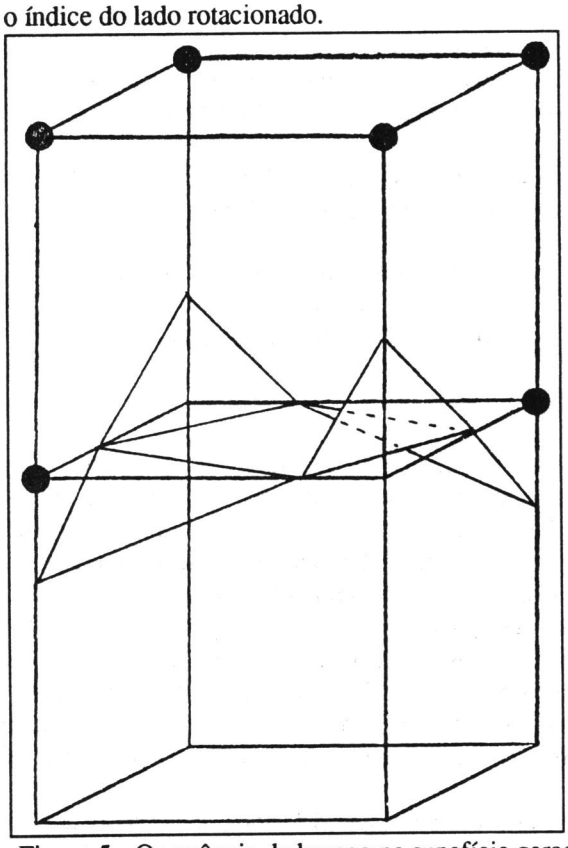


Figura 5 - Ocorrência de buraco na supefície gerada.

A TC também deve conter as configurações correspondentes aos casos especiais, ou seja, ambiguidades que surgem quando determinados tipo de configurações ocorrem em cubos adjacentes. O exemplo mais simples destas ambiguidades ocorre quando se tem um cubo inferior com configuração 0x90 e o cubo imediatamente superior com configuração 0xF9 Considerando-se que os pontos marcados do cubo na figura 5 indicam pontos externos a superfície e os não marcados indicam os internos, um observador que vê a superfície de cima notará nela um buraco que corresponde exatamente ao quadrilátero formado na união dos triângulos. Normalmente, observa-se este tipo de problema em regiões de variação suave de uma superfície ligeiramente plana.

Para se evitar este problema deve se definir uma nova configuração que será associada a configuração 0xF9. O que se pretende é se redefinir os dois triângulos do caso 0xF9 para um hexágono, o que elimina o buraco. Assim, a TC deve conter estas configurações extras.

Cálculo das Configurações

Como eficiência é um requisito fundamental para esta rotina, o seu algoritmo deve ser o mais eficiente dentro do conjunto de algoritmos que melhor aproveitam os recursos da máquina em que a rotina será processada. No caso, os recursos da máquina são processamento vetorial e paralelo.

Em máquinas com recursos vetoriais, onde um certo número de componentes de um vetor é processado ao mesmo tempo, deve se tentar implementar a rotina utilizando-se o maior número possível de operações vetoriais. O primeiro passo para se vetorizar uma rotina é identificar operações similares que podem ser processadas de forma independente. O cálculo da configuração dos pontos de cada cubo, por exemplo, é independente de um cubo para outro e portanto, naturalmente vetorizável em função do número do cubo.

De forma a permitir a vetorização do cálculo da configuração de cada cubo, duas operações auxiliares são necessárias inicialmente. A primeira é a geração de um vetor (**pcubo**) que contém o identificador do primeiro vértice de cada cubo. Assim, no caso de uma grade de dimensão 3x3x3 (fig. 6), tem se pcubo[0] = 0, pcubo[1] = 1, pcubo[2] = 3, pcubo[3] = 4, pcubo[4] = 9, pcubo[5] = 10, pcubo[6] = 12, pcubo[7] = 13.

Observe-se que a enumeração dos cubos de uma grade depende da forma pela qual se marcha na grade. No caso da rotina implementada se escolheu o seguinte caminho de marcha: primeiro na direção do eixo "y", segundo na do eixo "x" e terceiro na do eixo "z". A razão desta escolha é porque os valores dos pontos de grade (**VPGs**) são obtidos de um vetor (**VTGRID**) que foi definido inicialmente como uma matriz tridimensional em uma rotina Fortan e como se é sabido, o compilador Fortran alinha uma matriz por coluna. Assim, para se otimizar os cálculos de

endereçamento dos VPGs, se adotou este modo de percurso.

Para se calcular a configuração de um cubo é necessário se comparar os VPGs associados com os vértices de um cubo com um valor de referência (**isovalor**). Como a superfície a ser gerada cortará os lados de um cubo cujos vértices tem VPG acima e abaixo do isovalor, este parâmetro é o que indica a forma das superfícies passíveis de serem geradas em uma determinada grade. Assim, na segunda operação, se utiliza um vetor auxiliar (**vtaux**) que está alinhado com VTGRID. Cada componente de vtaux pode assumir três valores: **0** se o ponto de grade tem valor menor do que isovalor, **1** se o ponto de grade tem valor igual ou maior do que isovalor e **0x1000** se o VPG for invalido.

Observe-se que VPG invalido indica que um determinado ponto de grade não tem dado associado. É comum, por exemplo, dados provenientes de radar meteorológicos estarem incompletos, ou então, por falta de precisão serem descartados.

Por último se calcula as configurações que são armazenadas no vetor denominado **vtconf**. Observe-se que os parâmetros valor_plano_z?_?? são associados aos vértices de cada cubo conforme definido a seguir.

```
valor_plano_z0_00 -> vtaux[ pcubo[ii] ]
valor_plano_z0_10 -> vtaux[ pcubo[ii] + Ydim ]
valor_plano_z0_01 -> vtaux[ pcubo[ii] + 1 ]
valor_plano_z0_11 -> vtaux[ pcubo[ii] + Ydim + 1]
valor_plano_z1_00 -> vtaux[ pcubo[ii] + kxy ]
valor_plano_z1_10 -> vtaux[ pcubo[ii] + Ydim + kxy ]
valor_plano_z1_01 -> vtaux[ pcubo[ii] + 1 + kxy ]
valor_plano_z1_11 -> vtaux[ pcubo[ii] + 1 + Ydim +
    kxy ]
```

Onde **Xdim** e **Ydim** são as dimensões da grade nos eixos "x" e "y" e **kxy** é o produto de Xdim e Ydim. Observe-se que a dimensão da grade no eixo "z" está refletido no valor de pcubo. Assim, o cálculo das configurações é somente uma conjugação de operações de *shift* e *and* sobre vtaux, cujas componentes são endereçadas por pcubo. Ressalte-se que todas as três configurações envolvidas no cálculo das configurações são vetorizáveis.

Cálculo dos Vértices

O cálculo dos vértices dos triângulos já não é uma operação naturalmente vetorizável. Isto porque na grade, um vértice pode ser compartilhado por um cubo superior e um inferior ou por um cubo lateral a outro. Assim o cálculo dos vértices não é independente em função dos cubos.

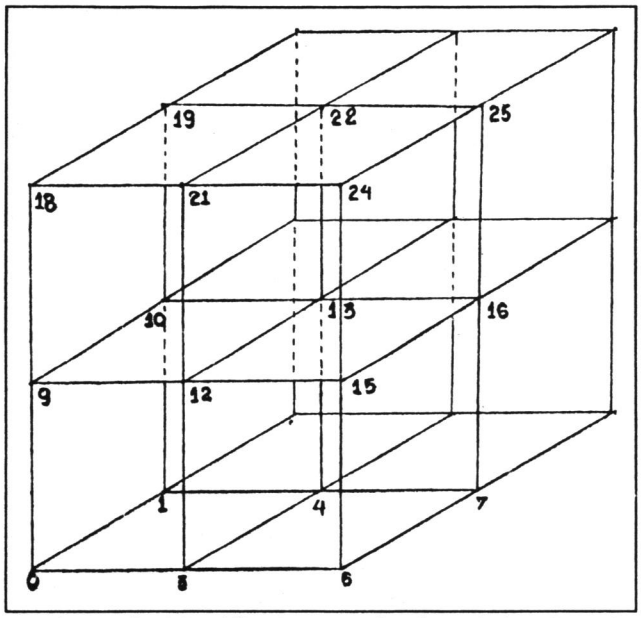


Figura 6 - Identificador do primeiro vértice de cada cubo.

O cálculo dos vértices dos triângulos é feita sequencialmente ao longo dos cubos enumerados de acordo com o fragmento de código abaixo:

```
Calcula_Vertices_dos_Poligonos_de_Cada_Cubo()
{   for (ii=0; ii < num_cubos; ii++)
    {
        Acha_LadosCubo_com_Vert_Pol();
        Calcula_Vertices();
        Atualiza_Estrutura_de_Dados();
    }
}
```

Para se descobrir os lados do cubo que contém vértices dos polígonos basta apenas consultar a TC.

É necessário se evitar o cálculo repetido de vértices. Assim foram definidos quatro conjuntos auxiliares de planos. Dois deles estão associados ao plano "xy" e serão considerados o "chão" (**pXY_0**) e o "teto" (**pXY_1**) dos cubos que estão em um mesmo plano horizontal (fig 7). Os outros dois estão associados ao plano "yz" e serão considerados a "lateral esquerda" (**pYZ_0**) e a "lateral direita" (**pYZ_1**) dos cubos (fig. 8).

Como um cubo tem 12 lados, é definido um vetor auxiliar (**vav**) de dimensão 12 no qual será armazenado os "índices" dos vértices calculados para o cubo corrente. Assim, quando se calcula um novo vértice, o índice associado a ele é inserido em vav e em um dos planos auxiliares. Os vértices no topo dos cubos vão para pXY_1 e os no fundo para pXY_0. Similarmente,

os vértices na lateral esquerda e direita do cubo vão, respectivamente, para pYZ_0 e pYZ_1.

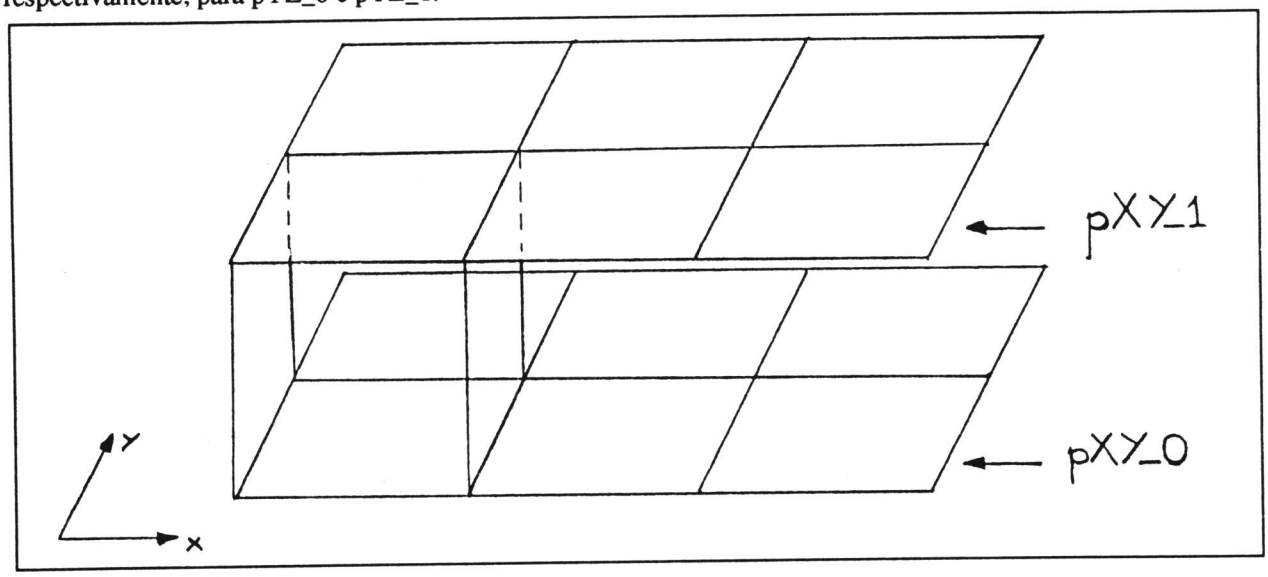


Figura 7 - Plano horizontal.

Quando se processa um vértice que já foi calculado anteriormente, o indice do vértice é simplesmente retirado de um dos planos auxiliares e inserido em vav.

Com todos os vértices calculados e os respectivos índices inseridos em vav, se atualiza VX, VY, VZ, pXY_0, pXY_1, pYZ_0, pYZ_1, Pol_f_Vert e Vert_f_Pol. **VX, VY e VZ** contém as coordenadas dos vértices. **Pol_f_Vert** armazena os índices dos polígonos em função dos índices dos vértices. **Vert_f_Pol** armazena os índices dos vértices em função dos índices dos polígonos. Pol_f_Vert e Vert_f_Pol serão utilizados no cálculo das normais e da malha de triângulos.

De forma a se otimizar o processamento, quando se passa para o processamento do plano horizontal superior, inverte-se os planos auxiliares: o antigo "teto" vira o novo "chão" e o antigo "chão" é limpo para se tornar o próximo "teto". Assim, não se calcula nenhum ponto na parte inferior dos cubos, desde que pelo menos uma camada completa de cubos em um plano horizontal já tenha sido processado.

A explicação da forma de utilização dos planos auxiliares "yz" é a mesma da utilização dos planos auxiliares "xy".

Quando calcula-se os vértices com os polígonos divididos a priori em triângulos, é necessário apenas se modificar o módulo de atualização da estrutura de dados.

Cálculo das Normais

Para o cálculo da normais à superfície em cada vértice é possível se utilizar dois tipos de métodos: gradiente (**MGR**) e produto vetorial (**MPV**).

Pelo primeiro método, em uma grade regular, a normal em um vértice é calculado utilizando-se os VPGs através da seguinte fórmula:

$$Nx(x,y,z) = VPG(x+1,y,z) - VPG(x-1,y,z)$$
$$Ny(x,y,z) = VPG(x,y+1,z) - VPG(x,y-1,z)$$
$$Nz(x,y,z) = VPG(x,y,z+1) - VPG(x,y,z+1)$$

Observe-se que após este cálculo, as normais são normalizadas.

Quando os VPGs apresentam uma variação relativamente uniforme, o método do gradiente funciona bem, permitindo a geração de superfícies com boa qualidade visual, por exemplo, este método é excelente para o cálculo das normais de uma esfera. Testes realizados demonstraram que é possível se conseguir uma precisão média de até 5 graus no cálculo das normais de uma esfera utilizando-se este método, o que permite a geração de uma esfera de alta qualidade visual, ou seja, não se distingue linhas de quebra na esfera. Note-se também que em termos computacionais, este método é altamente eficiente e simples de ser implementado em forma vetorial.

No entanto, quando os VPGs variam abruptamente, o que significa que a superfície gerada é bem irregular, este método não é apropriado porque a imprecisão no cálculo da normal é significativo. No caso de superfícies irregulares não é possível se determinar a normal teórica e compará-la com a normal calculada, como é possível se fazer com uma esfera. Logo, não é possível se fazer um teste de precisão da normal e, portanto, a qualidade visual da imagem é o único parâmetro de avaliação desta precisão e, consequentemente, do método empregado.

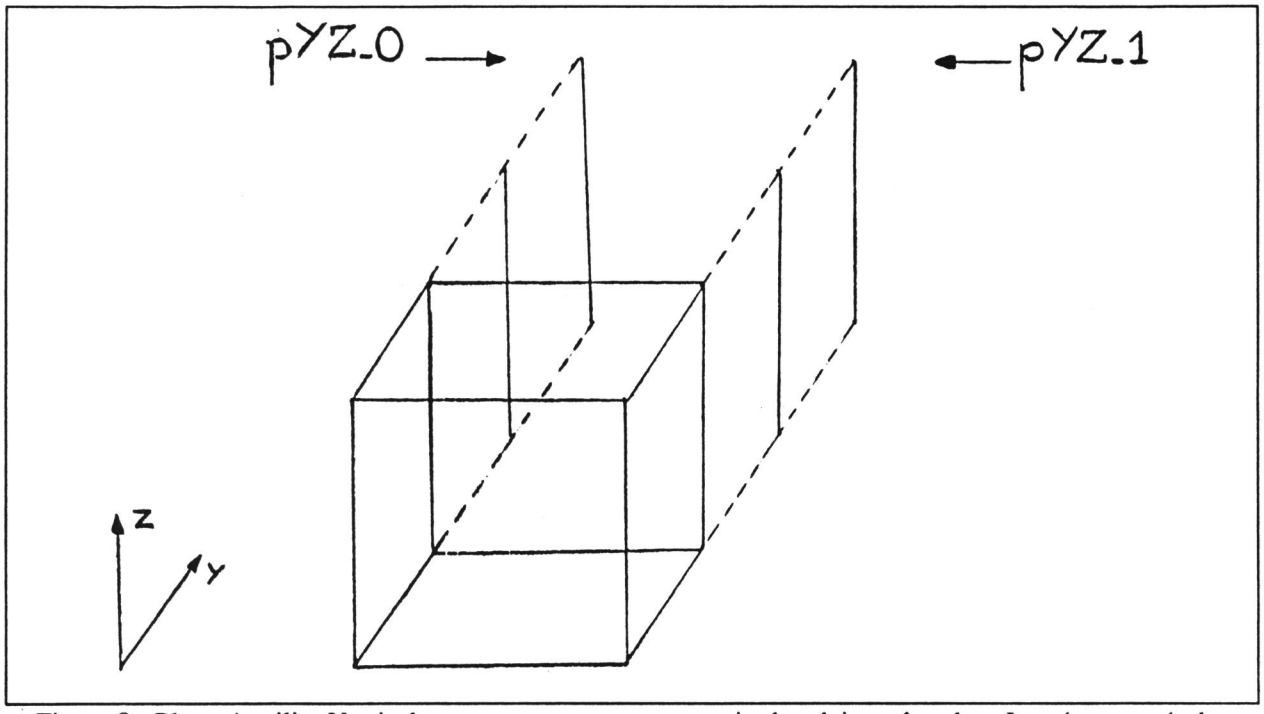


Figura 8 - Plano Auxiliar Vertical.

O MPV permite a geração de superfícies com maior qualidade visual. O processo é também simples, no entanto, mais custoso em termos computacionais do que o método do gradiente. A idéia básica deste método é que dado um vértice, calcula-se uma normal média a partir das normais de todos os triângulos conectados ao vértice e a normal de cada triângulo é calculada pelo produto vetorial.

Existe um detalhe de fundamental importância para o aumento da qualidade da imagem quando da utilização deste método. É logico que ao se calcular a normal média deve se dar um peso as normais que entram no cálculo e uma maneira natural de se atribuir este peso é através da área do triângulo. Logo é possível se fazer a normal direta ou inversamente proporcional à área do triângulo.

Muitos testes demonstraram que a normal inversamente proporcional à área do triângulo permite a geração de uma superfície com muito melhor qualidade do que a diretamente proporcional. Uma explicação pode ser dada através do esboço na figura 9 em que três retângulos formam uma espécie de "cotovelo". Observe-se que sendo a normal dos retângulos inversamente proporcional à área deles, as normais dos vértices compartilhados estão mais próximas da normal do retângulo menor e, portanto, se consegue uma transição mais suave da imagem neste cotovelo. Facilmente se nota que diminuindo a área do retângulo menor até ela se tornar uma linha, as normais dos dois vértices nesta linha deveriam apontar na direção da normal média das normais dos dois retângulos. Isto é compatível com a direção das normais médias calculadas quando o retângulo menor tem área diferente de zero.

O problema no uso deste método é exatamente, quando ocorrem casos em que a área do triângulo é zero. Isto pode ocorrer no processamento de grades em que há pontos que tem valor igual ao valor de referência (isovalor). Neste caso pode se calcular um mesmo ponto para os três vértices de um triângulo. Observe-se que somar um valor constante as coordenadas dos vértices calculados para impedir que eles caiam no vértice do cubo não funciona porque traz uma imprecisão significativa no cálculo das normais.

Limpar a estrutura de dados de forma a evitar vértices coincidentes se mostrou inviável dado ao grande impacto que teria sobre a velocidade de processamento do algoritmo. Note-se que este é um problema normalmente raro de ocorrer quando se processa dados reais e, portanto, a sua solução não deve ter impacto sobre o processamento dos casos em que este problema não ocorre.

Assim, uma solução razoavelmente simples para este problema foi a combinação do MPV com o MGR. Quando a área do triângulo é menor do que um valor mínimo, o que indica que a área do triângulo é muito pequena ou zero, é calculado o valor do gradiente para cada um dos seus três vértices. De forma a tornar o valor da normal calculada pelo gradiente de alguma forma proporcional à área do triângulo, normaliza-se os vetores e divide os pelo valor da área mínima.

Observe-se que o valor mínimo de área está relacionado com a precisão de *floating point* da

máquina (**pFP**). Assim, se realizou alguns testes e verificou-se que um bom critério para a definição da área mínima (**AM**) é "minimo(1.0e-4,pFP)".

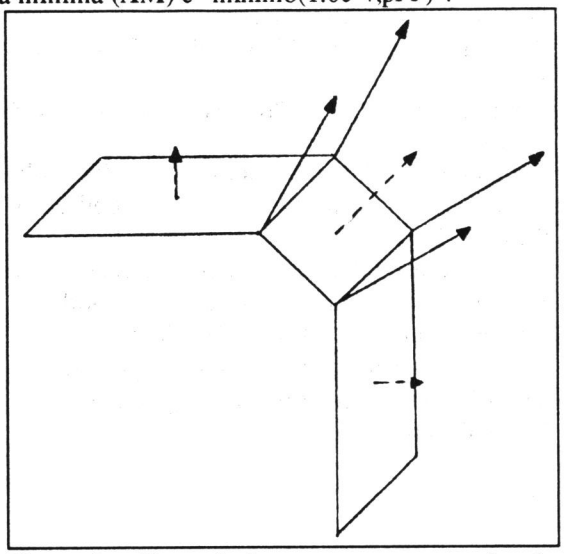


Figura 9 - Normais inversamente proporcionais às áreas.

No cálculo das normais se destacam quatro operações principais:

a) Determina-se os vetores diferença para o cálculo do produto vetorial, ou seja, calcula-se os vetores "A", "B", "C", etc, que definem triângulos consecutivos dentro dos polígonos. Calcula-se o produto vetorial entre os vetores "A" e "B", "B" e "C", etc.

b) Armazena-se no vetor n[i] o produto vetorial dos vetores V1[i] e V2[i] e normaliza-se n[i], o que corresponde a torná-lo proporcional à área do triângulo. No caso de área zero, desconsidera-se o cálculo.

c) Calcula-se o índice (**k**) do vértice que corresponde ao "v-ésimo" vértice do polígono "p" e atualiza o vetor das normais: N[k] += n[i].

d) No final do processamento, se N[k] igual a zero, calcula-se o gradiente "g" do vértice e atualiza o vetor das normais: N[k] += g.

e) Terminado as operações anteriores, se aplica a matriz TN (explicada abaixo) sobre as normais N[i] e posteriormente elas são normalizadas.

As operações "a", "b" e "e" são vetorizáveis.

Observe-se que não é possível se vetorizar um laço que contenha a chamada de rotinas escritas pelo implementador. Somente rotinas intrínsicas ao compilador (raiz quadrada, seno, cosseno, etc) são passíveis de vetorização. Assim, dentro de um laço vetorizado são chamadas apenas *macros*.

Seja **T** uma matriz de transformações escalares a ser aplicada aos pontos que definem uma superfície. A partir de T, calcula-se **TN,** que é a matriz de transformações a ser aplicada às normais.

Avaliação do Algoritmo Implementado

Para se testar a eficiência do algoritmo implementado em comparação com a versão antiga, se montou duas grades. Os dados de grade de uma delas representam um plano horizontal e a outra representa quatro esferas que se interceptam no espaço.

As máquinas onde se realizou os testes foi uma super estação de trabalho Stellar/Stardent e um supercomputador Cray II.

No caso da Stellar, a velocidade de processamento da representação plano foi 4 vezes mais rápida que a versão anterior do algoritmo e da representação esfera 3 vezes maior. No caso do Cray II, a velocidade de processamento da representação plano foi 3 vezes maior e a da representação esfera foi 2,5 vezes maior.

Para um sistema interativo em que o gerador de superfícies é um módulo básico, estes dados de aumento de peformance são altamente relevantes.

Ressalte-se que em ambas as máquinas, os programas foram apenas vetorizados. Se descartou paralelizar trechos dos programas por uma simples razão: "a velocidade de execução dos programas paralelizados não aumentou de um fator igual ao número de processadores da máquina". Tanto o Cray como a Stellar tem quatro processadores e a paralelização dos programas permitiu que se aumentasse a velocidade deles de um fator em torno de 1.6.

Este pequeno aumento de performance está relacionado ao fato de que os trechos que preferencialmente se paraleliza em um programa são os "laços vetorizados". Isto porque as operações nestes laços são independentes entre si, o que é essencial para que elas sejam executadas em paralelo. Assim, um processador se mantém responsável pela execução do programa e dividi com os outros o processamento dos laços vetorizados.

Como paralelizar laços vetorizados implica em otimizar uma porção já otimizada do programa, o tempo de execução do programa é limitado pelo tempo de execução das suas partes não otimizadas. Assim, como as partes não paralelizadas são executadas pelo processador principal, não se consegue multiplicar a velocidade de execução do programa pelo número de processadores da máquina.

Uma forma de se contornar este problema é associar uma cópia do programa a cada processador, de forma que cada um gere uma superfície. Isto é muito útil para se produzir uma sequência de gráficos de um mesmo campo meteorológico em intervalos de tempo diferentes ou então para se gerar superfícies associadas a diferentes campos.

A configuração das esferas usada nos testes representa um caso mais complexo e portanto mais próximo dos casos reais. No caso de uma grade de dimensões 70x70x70, esta configuração envolve 19989 polígonos ou 39647 triângulos processados em 1,34 segundos. Logo, a versão Cray do algoritmo é capaz de processar por segundo em torno de 15.000 polígonos ou 30.000 triângulos ou 250.000 cubos.

Bibliografia

[Battaiola (1991)] Análise de Conceitos relacionados a Implementação de Sistemas de Visualização Tridimensional de Dados Meteorológicos
André Luiz Battaiola
Tese de doutorado apresentada no Departamento de Engenharia Elétrica - Escola Politécnica - USP - 1991

[Camara et al. (1989)] MicroMAGICS: Meteorological Graphics on Microcomputers
G. C. Neto, E. Nishimura, A. L. Battaiola, C. H. S. Ning, N. L. Vijaykumar, L. Massa (INPE) e J. Daabeck (ECMWF)
Anais Sibgrapi 1989

[Foley et al. (1990)] Computer Graphics - Principles and Pratice, Second Edition
J. Foley, A. van Dam, S. Feiner, J. Hughes
Addison Wesley, 1990

[Hibbard (1986)] Computer-Generated Imagery for 4-D Meteorological Data
William L. Hibbard
Bulletin of the American Meteorological Society, vol. 67, num. 11, Novembro 1986

[Hibbard-Santek (1988)] Visualizing Weather Data
William L. Hibbard, David Santek
Workshop on Graphics in Meteorology, ECMWF, 1988

[Hibbard-Santek (1989-A)] 4-D Dysplay of Geohydrological Model Results
David A. Santek, William L. Hibbard
Proceedings, 5th Interactive. Info. and Processing Systems for Met., Ocean., and Hydro., 1989

[Hibbard-Santek (1989-B)] Visualizing Large Data Sets in the Earth Sciences
William L. Hibbard, D. Santek
Computer, vol. 22, num. 8, 1989

[Hibbard-Santek (1989-C)] Interactivity is the Key
David A. Santek, William L. Hibbard
Chapel Hilll Workshop on Volume Visualization, University of North Carolina, 1989

[Hibbard-Santek (1989-D)] Visualizing Large Data Sets in the Earth Sciences
William L. Hibbard, David A. Santek
Computer - IEEE, Agosto 1989

[Hibbard-Santek (1990)] The VIS-5D System for Easy Interactive Visualization
William Hibbard, Dave Santek
Proceedings of the First IEEE Conference on Visualization, 1990

[Hibbard et al. (1991)] Progress with the VIS-5D System
William Hibbard, Brian Paul, Andre L. Battaiola
Artigo aceito pela comissão de publicação do Siggraph 91

[Lorensen et al. (1987)] Marching Cubes: A High Resolution 3D Surface Construction Algorithm
William E. Lorensen, Harvey E. Cline
Computer Graphics, vol. 21, num. 4, julho de 1987.

[XFDI] XFDI Users and Programming Guide
Stellar Computer Inc.