# Modelagem de Terrenos por Superfícies Triangulares de Bezier

MILTON TERUAKI SUETSUGU SAKUDE

Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica - ITA
Centro Técnico Aeroespacial
12228 - 900 São José dos Campos, SP, Brasil
ita@brfapesp.bitnet

**Abstract.** A terrain modeling generally involves a triangulation and a surface representation. With this approach, this paper presents a divide and conquer algorithm for Delaunay triangulation, a surface modeling using cubic Bezier triangular patches and some algorithms for contourn maps.

## 1 Introdução

Modelar digitalmente um terreno significa reconstruí -lo por um modelo de aproximação ou de interpolação de dados adquiridos de levantamentos ou mapas topográficos, de aerofotogrametria e de imagens de satélites.

Por se tratar de uma grande quantidade de dados, o processo necessita da determinação dos pontos vizinhos (adjacências dos pontos) por algoritmos de triangulação. A triangulação 2D é suficiente ([De Floriane (1987)], [Lee-Schachter (1980)]), embora a 3D possa ser usada [Choi et al. (1988)]. Buscando obter triângulos mais equiangulares quanto possível, usa-se um critério de otimização [Choi et al. (1988)], tais como, o de menor diagonal ou o maior ângulo em um quadrilátero e o de Delaunay. A triangulação de Delaunay forma triângulos, cuja circunscrição não contém nenhum outro ponto em seu interior, denominados triângulos de Delaunay.

Geralmente um terreno é modelado pela junção de pequenos retalhos de superfícies definidos algebricamente. Os modelos de interpolação predominam sobre os de aproximação. Um dos requisitos importantes de um modelo é evitar *overshoots* elevados, decorrentes do processo de interpolação não linear. Outro requisito é a facilidade de cálculos para se obter resultados ou propriedades ,tais como, isolinhas, mapas altimétricas, volume e área.

Neste trabalho usa-se um algoritmo de triangulação de Delaunay com a abordagem de dividir e conquistar. A modelagem de terrenos é feita usando superfícies triangulares cúbicas de Bezier.

Na seção 2 serão apresentados os fundamentos teóricos sobre a superfície triangular de Bezier usados neste trabalho. Na seção 3 será apresentado o algoritmo de triangulação utilizado; na seção 4, o modelo de superfície adotado; a seguir, na seção 5, alguns algoritmos para determinação de isolinhas e na seção 6, alguns resultados e considerações finais.

## 2 Superfície Triangular de Bezier

### 2.1 Coordenadas baricêntricas

A superfície triangular de Bezier deriva da utilização de coordenadas baricêntricas em superfícies planas. Em uma superfície plana definida por três pontos $\mathbf{p}_1$, $\mathbf{p}_2$ e $\mathbf{p}_3$ (figura 1), um ponto $\mathbf{p}$ pertencente a este plano pode ser representado por:

$$\mathbf{p} = u\mathbf{p}_1 + v\mathbf{p}_2 + w\mathbf{p}_3$$

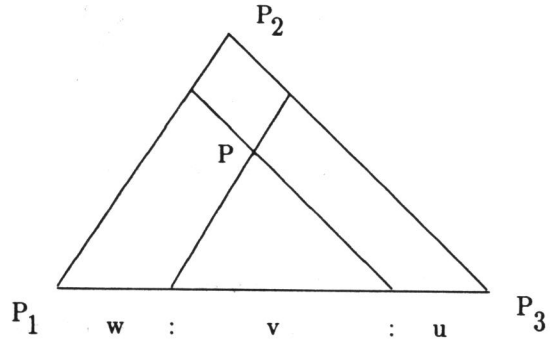



Figura 1: Coordenadas baricêntricas.

Se $u + v + w = 1$ para $u, v, w \geq 0$, o ponto $\mathbf{p}$ pertence ao interior do triângulo formado pelos pontos $\mathbf{p}_1$, $\mathbf{p}_2$ e $\mathbf{p}_3$ .

*Notações*: o índice em negrito $\mathbf{i}$ denotará o vetor $(i, j, k)$; serão usadas as abreviações $\mathbf{e}_1 = (1, 0, 0)$, $\mathbf{e}_2 = (0, 1, 0)$ e $\mathbf{e}_3 = (0, 0, 1)$.

### 2.2 Forma Bezier-Bernstein

Uma superfície Bezier pode ser expressa usando o polinômio de Bernstein [Farin (1988)]:

$$\mathbf{b}(\mathbf{u}) = \sum_{|\mathbf{i}|=n} \mathbf{b}_{\mathbf{i}} B_{\mathbf{i}}^{n}(\mathbf{u})$$

onde:
$|\mathbf{i}| = i + j + k = n$
$\mathbf{u} = (u, v, w)$; $u + v + w = 1$; $u, v, w \geq 0$;
$\mathbf{b}_{\mathbf{i}} = \mathbf{b}_{ijk}$ são os pontos de controle (figura 2);
e
$B_{\mathbf{i}}^{n}(\mathbf{u}) = \frac{n!}{i!j!k!} u^i v^j w^k$ é o polinômio de Bernstein.

A superfície triangular de Bezier interpola os pontos extermos e é limitada por curvas de Bezier de grau n (figura 2).

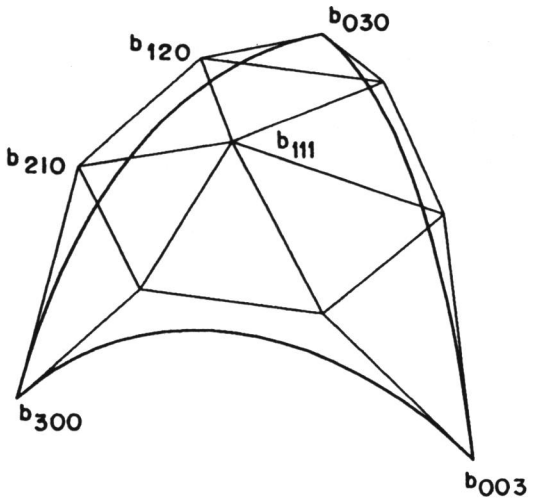



Figura 2: Superfície de Bezier.

## 2.3 Algoritmo de Casteljau

Uma das formas eficientes de avaliar a expressão anterior é usar o algoritmo recursivo de Casteljau [Farin (1988)]:

$$\mathbf{b}_{\mathbf{i}}^{r}(\mathbf{u}) = u\mathbf{b}_{\mathbf{i}+\mathbf{e}_1}^{r-1}(\mathbf{u}) + v\mathbf{b}_{\mathbf{i}+\mathbf{e}_2}^{r-1}(\mathbf{u}) + w\mathbf{b}_{\mathbf{i}+\mathbf{e}_3}^{r-1}(\mathbf{u})$$

onde:
$|\mathbf{i}| = n - r$ para $r = 1 \cdots n$ ;
$\mathbf{b}_{\mathbf{i}}^{0} = \mathbf{b}_{\mathbf{i}}$
e índices negativos significam $\mathbf{b}_{\mathbf{i}}^{r} = 0$.

## 2.4 Subdivisão

Uma das características importantes da formulação da superfície de Bezier é a existência de algoritmos de subdivisão que produzem novas superfícies de Bezier.

Em se tratando de superfícies triangulares de Bezier, existem várias maneiras de subdividir um retalho de superfície [Filip (1986)], [Goldman (1983)]. Estes algoritmos se baseam no algoritmo de Casteljau. O algoritmo de Casteljau subdivide a superfície de Bezier em três novas superfícies (figura 3b), quando se toma valores de parâmetros $0 < (u, v, w) < 1$. Quando um desses parâmetros é nulo, obtém-se duas novas superfícies (figura 3a).

A aplicação destes tipos de subdivisão sucessivamente tem o incoveniente de produzir superfícies alongadas (triângulos pontiagudos). A solução deste problema é utilizar a subdivisão em 4 novas superfícies (figura 3c) aplicando a subdivisão por dois.

## 2.5 Elevação de grau

Elevar o grau de uma superfície de Bezier significa representá-la por uma superfície de grau maior $(n + 1)$. N+1 pontos de controle adicionais devem ser gerados.

Os pontos de controle $\mathbf{b}_{\mathbf{i}}^{(1)}$ da nova superfície são obtidos de [Farin (1988)]:

$$\mathbf{b}_{\mathbf{i}}^{(1)}(\mathbf{u}) = \frac{1}{n+1}(i\mathbf{b}_{\mathbf{i}-\mathbf{e}_1} + j\mathbf{b}_{\mathbf{i}-\mathbf{e}_2} + k\mathbf{b}_{\mathbf{i}-\mathbf{e}_3})$$

onde: $|\mathbf{i}| = i + j + k = n + 1$

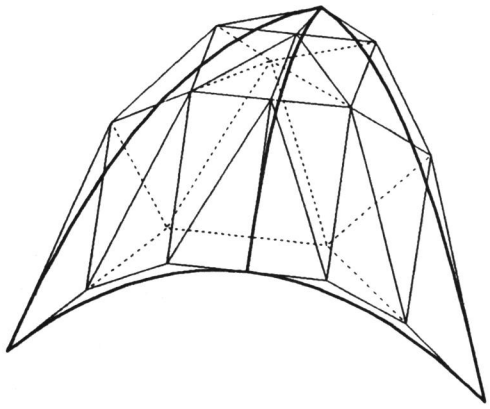
## 2.6 Continuidade $C^1$

Para se juntar duas superfícies triangulares de Bezier com continuidade $C^1$ são necessárias três condições [Farin (1988)]:

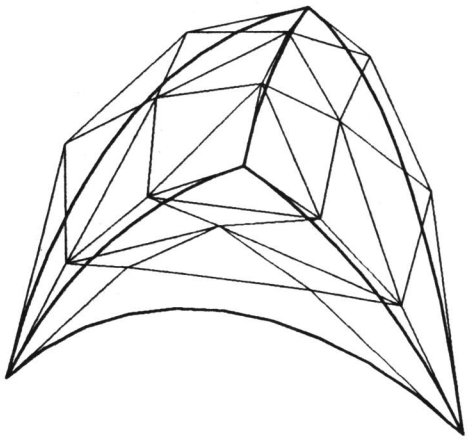
- coincidências dos pontos de controle das fronteiras;
- os pares de triângulos adjacentes sejam coplanares (figura 4);
- os domínios das superfícies (triângulos) sejam um mapeamento afim.

## 2.7 Superfície Triangular Não Paramétrica

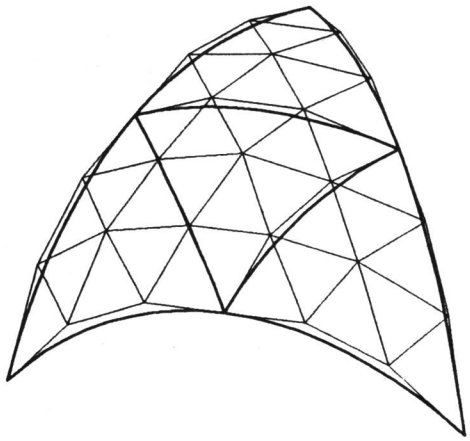
Como o domínio das superfícies paramétricas (entradas) são os parâmetros $u$ e $v$, indagações a partir do espaço da imagem $(x, y, z)$, muitas vêzes, não são fáceis de responder. Saber o valor de $z$ da superfície, dado os valores de $x$ e $y$, é um exemplo deste tipo de problema. A solução deste problema envolve o cálculo de raízes (sistemas de equações não lineares).



a. Subdivisão em 2 retalhos.



b. Subdivisão em 3 retalhos.



c. Subdivisão em 4 retalhos.

Figura 3: Subdivisão da Superfície de Bezier.

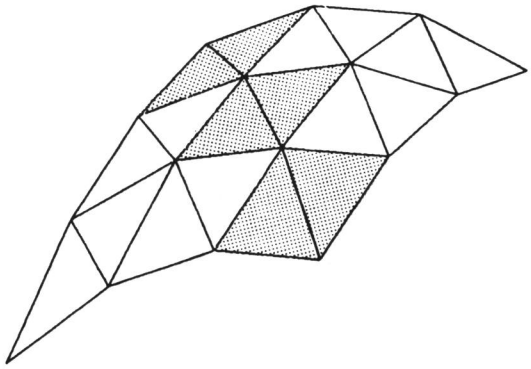


Figura 4: Condições para continuidade $C^1$.

No caso da superfície triangular de Bezier, tais problemas podem ser simplificados quando se tomam pontos em $x$ e $y$ regularmente espaçados como ilustrado na figura 5. Formas de Bezier definidas deste modo foram denominadas *não paramétricas* em [Farin (1988)] (pois tem uma parametrização simples).

Sobre as ordenadas $x$ e $y$ é válida a propriedade de precisão linear do polinômio de Bernstein [Farin (1988)]. Isto significa que sobre as ordenadas $x$ e $y$ pode se trabalhar com a superfície de Bezier de ordem 1. A prova é por elevação de grau: partindo de uma superfície de ordem 1, aplica- se o algoritmo de elevação de grau sucessivamente n-1 vêzes no espaço $x\ y$.

## 3 Triangulação de Delaunay

O algoritmo implementado baseou-se na abordagem de dividir e conquistar proposto por Lee e Schachter [Lee-Schachter(1980)], que usa como critério de otimização o de Delaunay. O resultado da triangulação é um grafo contendo triângulos de Delaunay e tendo como fronteira um polígono convexo.

### 3.1 Estrutura de Dados

Para se obter uma boa eficiência em um algoritmo é importante a utilização de uma estrutura de dados adequada.

A estrutura de dados que melhor representa o resultado de uma triangulação é um grafo. Muitos

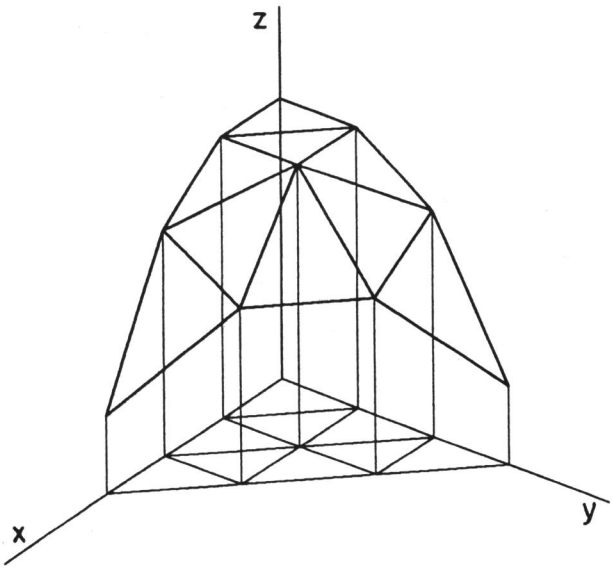



Figura 5: Superfície *não paramétrica*.

autores [De Floriane (1987)], [Choi et al. (1988)] utilizam algumas estruturas de dados auxiliares, tais como pilhas e listas, na implementação do algoritmo de triangulação. O interessante no trabalho de Lee e Schachter é a utilização de apenas uma estrutura de dados, um grafo com listas de adjacências duplamente ligadas e circulares. Pilhas, listas, árvores são casos particulares de grafos. O uso destas estruturas pode ser feito por algoritmos sob o grafo.

O acesso à adjacência de um vértice é feita por operadores *First, Succ e Pred.* O operador *Succ* $(v_0, v_1)$ fornece o próximo vértice adjacente de $v_0$ depois de $v_1$, no sentido anti-horário. O *Pred*$(v_0, v_1)$ funciona de forma semelhante, só que no sentido horário. O operador *First*$(v)$ fornece o primeiro vértice adjacente a $v$ da lista de adjacência, que para os pontos de fronteira de triangulação é o ponto da fronteira depois de $v$ quando se percorre a fronteira no sentido anti-horário. Esta disposição possibilita percorrer os pontos de fronteira do grafo.

Nesta implementação foi utilizada a representação de grafos por multilistas de adjacência [Horowitz (1987)].

### 3.2 Algoritmo de Triangulação

Segue abaixo o algoritmo recursivo de forma resumida:

```
Algorithm Triangulation (S)

    if #S > 2
       Divide (S, S1, S2)
       Triangulation (S1)
       Triangulation (S2)
       Merge (S1, S2)
    else if #S=2
      InsertEdge(S.p1,S.p2)

end algorithm
```

O algoritmo *Divide* divide o conjunto de ponto $S$ em subconjuntos $S_1$ e $S_2$, usando como limiar o ponto médio do *box* envolvente, ora em relação ao eixo $x$, ora em relação ao eixo $y$, conforme a dimensão maior do *box*. Em sua implementação foi utilizado um algoritmo semelhante ao *Partition* do *QuickSort* [Aho et al. (1983)].

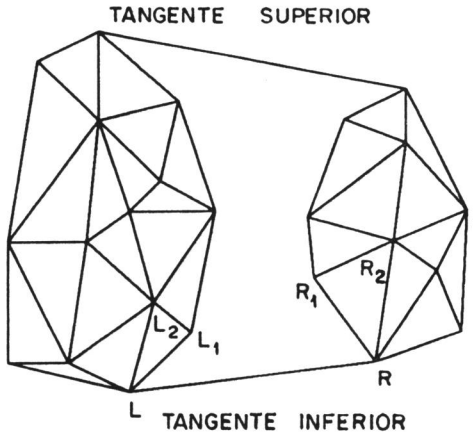



Figura 6: Algoritmo *Merge*.

O algoritmo *Merge* envolve duas partes:

- encontrar os limites das fronteiras para união (tangente superior e inferior, ver figura 6);
- união dos dois conjuntos com triângulos de Delaunay

As tangentes têm a seguinte propriedade: deixam de um dos lados todos os pontos do conjunto $S$ (figura 6). Para encontrar as tangentes, basta testar os pontos da fronteira.

No algoritmo *LowerTangent* a seguir, os dois pontos de mínimos da fronteira são escolhidos inicialmente para ser a tangente inferior. É verificado se os seus vértices adjacentes da fronteira estão de um mesmo lado (esquerdo), senão escolhe-se um desses vértices, de maneira conveniente, para formar a tangente. Este processo é repetido até encontar uma aresta (a tangente) com a propriedade acima.

```
Algorithm LowerTangent(p1,p2)

    p1s=First(p1)      p1p=Pred(p1,p1s)
    p2s=First(p2)      p2p=Pred(p2,p2s)
    do
         not_found=false
         if p1s is right of (p1,p2)
              not_found=true
              p1p=p1      p1=p1s
              p1s=Succ(p1,p1p)
         end if
         if p1p is right of (p1,p2)
              not_found=true
              p1s=p1      p1=p1p
              p1p=Pred(p1,p1s)
         end if
         if p2s is right of (p1,p2)
              not_found=true
              p2p=p2      p2=p2s
              p2s=Succ(p2,p2p)
         end if
         if p2p is right of (p1,p2)
              not_found=true
              p2s=p2      p2=p2p
              p2p=Pred(p2,p2s)
         end if
    while not_found

end algorithm
```

A união dos dois grafos é feita com triângulos de Delaunay, partindo da tangente inferior e prosseguindo em direção à tangente superior. No momento da junção, os triângulos não Delaunay próximos à fronteiras de união são removidos e substituídos por outros de Delaunay.

Lee e Schachter propuseram o seguinte algoritmo para fazer a união:

```
Algorithm  Merge(S1,S2)

    L =Min(S1)       R =Min(S2)
    T1=Max(S1)       T2=Max(S2)
    LowerTangent(L,R)
    UpperTangent(T1,T2)
    while (L,R) ≠ (T1,T2)
         InsertEdge(L,R)
         A=B=false
         R1=Pred(R,L)
         if R1 is right of (R,L)
              R2=Pred(R,R1)
              while InCircle(R2,L,R,R1)
                   DeleteEdge(R,R1)
                   R1=R2      R2=Pred(R,R1)
              end while
         else A=true
         L1=Succ(L,R)
         if L1 is right of (R,L)
              L2=Succ(L,L1)
              while InCircle(L2,R,L,L1)
                   DeleteEdge(L,L1)
                   L1=L2      L2=Succ(L,L1)
              end while
         else B=true
         if A
              L=L1
         else if B
                   R=R1
              else if InCircle(L1,L,R,R1)
                        L=L1
                   else R=R1
    end while
    InsertEdge(L,R)

end algorithm
```

O primeiro *loop while* interno remove todos os triângulos não Delaunay decorrente da inserção do ponto $R$ (figura 6) na triangulação de $S_1$. Para tanto, são analisados apenas os triângulos adjacentes à $L$ através do teste de circunscrição *InCircle*. O algoritmo *InCircle*$(p, p_1, p_2, p_3)$ verifica se o ponto $p$ está dentro da circunferência formada pelos pontos $p_1, p_2$ e $p_3$. O segundo *loop while* interno repete o processo na triangulação de $S_2$. Passado por estas duas etapas, os triângulos $LRL_1$ e $LRR_1$ são Delaunay, respectivamente, à triangulação de $S_1$ e de $S_2$. A decisão de qual deles é o de Delaunay, considerando as duas triangulações, é feita no último trecho (que começa com *if A*). A próxima aresta $LR$ pertence ao triângulo de Delaunay resultante desta decisão. As variáveis $A$ e $B$ são usadas para evitar testes *InCircle* desnecessários, que ocorrem na terminação do processo de união.

### 3.3 Desempenho do Algoritmo

Seja $N$ o número de pontos a serem triangulados. Seja $t(N)$ o tempo requerido para construir uma triangulação e $t_M$ o tempo de processar o *Merge*:

$t(N) = 2t(N/2) + t_M(N/2, N/2)$

$t(1) = 0$

onde $2t(N/2)$ representa o tempo para a triangulação dos subconjuntos $S_1$ e $S_2$.

O processo de determinação da tangente tem desempenho $O(N)$, pois tem um *loop* e cada ponto da fronteira testado é analisado apenas uma vez. O algoritmo de união é também $O(N)$, pois uma vez que

um ponto é analisado, ele não volta a ser analisado, ou seja, troca-se no máximo $N$ arestas. Logo, o tempo para fazer o Merge é $O(N)$.

Como $t_M(N/2, N/2) = O(N)$ e o processamento é realizado em um esquema de árvore binária, tem-se $t(N) = O(NlogN)$. O desempenho teórico do algoritmo de triangulação é $O(NlogN)$.

## 4 Modelo de Superfície Adotado

Se forem utilizados os triângulos resultantes da triangulação para a representação da superfície de um terreno, tem-se um modelo de interpolação linear dos pontos de entrada. Na maioria dos casos, deseja-se uma superfície mais suave. A abordagem usada neste trabalho, para tanto, é substituir estes triângulos por superfícies triangulares de Bezier de grau 3. Uma superfície de grau 2 não permite a interpolação de 3 pontos com normais arbitrárias, por isso ela foi rejeitada.

### 4.1 Determinação dos pontos de controle da superfície de Bezier

Uma superfície de Bezier de grau 3 tem 10 pontos de controle (figura 2) e interpola os 3 pontos extremos (vértices) da superfície. Logo, faltam 7 pontos a serem determinados. Os seis pontos adjacentes aos pontos extrenos definem as condições de tangência destes pontos. Se as normais dos pontos extremos forem definidas, os valores destes 6 pontos podem ser determinados. O vetor normal de um ponto extremo foi definido como a média das normais de todos os triângulos adjacentes (resultantes da triangulação). A ídéia é a mesma do algoritmo de *Gouroud Shading*.

Visando construir uma superfície *não paramétrica*, um ponto que define tangência foi definido como sendo o ponto do plano tangente ao ponto extremo, que fica a um terço do ponto extremo na projeção da aresta do triângulo sobre o plano $xy$ (figura 5).

Falta determinar (estimar) o ponto central $\mathbf{b}_{111}$. A determinação deste ponto não deve ser arbitrária para poder representar superfícies de ordens menores. Foi então usado o seguinte esquema (figura 7):

- construir duas superfícies triangulares de Bezier de grau 2, usando um esquema semelhante ao anterior, agora com a projeção do ponto de tangência no meio da aresta do triângulo;
- usando a elevação de grau nas duas superfícies, obter os pontos $\mathbf{b}'_{111}$ e $\mathbf{b}''_{111}$ (média dos 3 pontos de tangência);
- tomar o ponto $\mathbf{b}_{111}$ como sendo o ponto médio de $\mathbf{b}'_{111}$ e $\mathbf{b}''_{111}$

Escrevendo os pontos de tangência da superfície de grau 2 em função dos pontos da superfície de grau 3 obtém-se:

$$\mathbf{b}_{111} = \tfrac{1}{4}(\mathbf{b}_{012} + \mathbf{b}_{021} + \mathbf{b}_{120} + \mathbf{b}_{210} + \mathbf{b}_{102} + \mathbf{b}_{201}) - \tfrac{1}{6}(\mathbf{b}_{003} + \mathbf{b}_{030} + \mathbf{b}_{300})$$

A superfície assim construída pode-se reduzir ao grau 2, caso em que as superfícies estimadoras de $\mathbf{b}_{111}$ se coincidem, ou ainda, reduzir ao grau 1, caso em que os pontos extremos têm as mesmas normais.

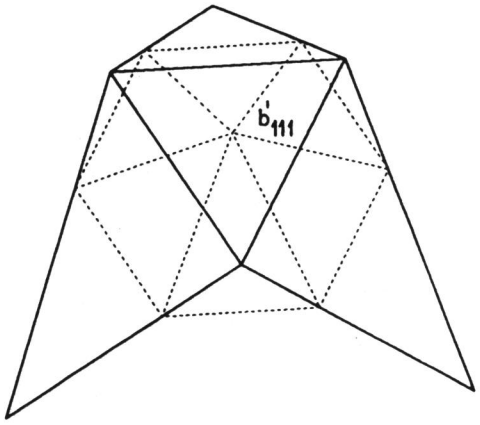



Figura 7: Superfície estimadora de $b_{111}$.

### 4.2 Superfícies com Continuidade $C^1$

Ao longo das junções das superfícies, o esquema anterior produz concordância $C^0$ (apenas nos pontos extremos, garante-se a condição $C^1$).

O ponto $\mathbf{b}_{111}$ poderia ser obtido impondo a condição de continuidade $C^1$ ao longo das junções. Analisando este problema, verifica-se que sua solução é de implementação complexa e requer ainda condições adicionais.

Neste trabalho foi usada uma solução mais simples baseado no esquema proposto em [Farin (1983)], ou seja: usar o algoritmo de subdivisão de Casteljau para produzir 3 retalhos (figura 8) e impor a condição de coplanaridade nas superfícies adjacentes usando o critério dos mínimos quadrados.

A figura 9 mostra a intersecção de um plano vertical que passa pelos pontos $a$ e $b$ pertencentes aos triângulos adjacentes achurados (figura 8). O ponto $p_a$ é o ponto do plano do triângulo que contém $b$ com ordenadas $x$ e $y$ do ponto $a$. O ponto $p_b$ tem uma definição semelhante. O critério dos mínimos qua-

drados é aplicado nos pontos $a, p_a, b$ e $p_b$ para achar os novos pontos $a'$ e $b'$ que alinham os dois triângulos adjacentes. Após efetuados os ajustes de coplanaridade nos três triângulos, é necessário ajustar os pontos indicados por $c$ e $d$ na figura 8, de modo a manter a condição de continuidade nas junções internas.

Este esquema possibilita a análise local, isto é, basta analisar apenas as superfícies adjacentes para impor as condições de continuidade $C^1$.

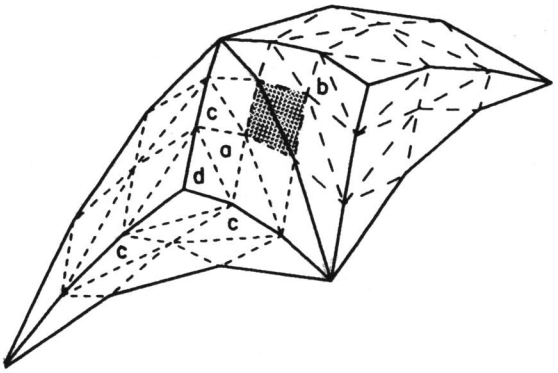



Figura 8: Superfícies com concordância $C^1$.

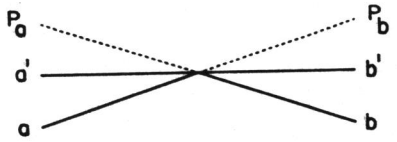



Figura 9: Ajuste de coplanaridade.

## 5 Determinação de Isolinhas

### 5.1 Curvas de Nível

Determinar uma curva de nível é resolver a equação $S(u, v).z = Z$ para todas as superfícies que compõe o modelo do terreno. Este é um problema de determinação de raízes de uma equação com duas variáveis. Sua solução envolve a redução desta equação para uma de uma variável, a obtenção de uma amostragem de pontos representativos da curva de nível e a conecção destes pontos.

A abordagem usada neste trabalho é a de dividir e conquistar, com esquema semelhante aos trabalhos de Houghton [Houghton et al. (1985)], Lasser [Lasser (1986)] e Sakude [Sakude (1989)] (figura 10):

- dividir a superfície em 4 retalhos, usando o esquema descrito na secção 2.4;
- calcular a intersecção das curvas dos contornos do retalho com o plano $Z$ constante.

O processo de subdivisão tem as seguintes finalidades:

- dividir o domínio dos parâmetros para produzir uma amostragem de pontos solução;
- separar as raízes;
- conectar os pontos da curva de nível.

A subdivisão é feita recursiva e adaptivamente com os seguintes critérios de parada: mínimo segmento da curva de nível; planaridade da superfície de Bezier e mínimo tamanho de superfície (superfície pontual).

O problema de raízes se reduziu a determinar uma raiz de um polinômio do terceiro grau. Para tanto, foi utilizado o método de Newton-Raphson. O método de Casteljau para curvas foi usado para fornecer tanto o valor do ponto quanto o valor da tangente, necessários na expressão de Newton-Raphson. Quando se avalia o valor do ponto por Casteljau, o valor da tangente também pode ser avaliado [Farin (1988)], o que diminui o tempo de processamento.

Os valores de máximo e mínimo $z$ dos pontos de controle das superfícies são armazenados para rejeitar trivialmente superfícies que não fazem intersecção com o plano $Z$ constante ou efetuar análises mais complexas, caso contrário.

Os retalhos resultantes da subdivisão são mantidos em uma *árvore* para poder processar os demais valores de $Z$ listados, evitando repetidos cálculos de subdivisão.

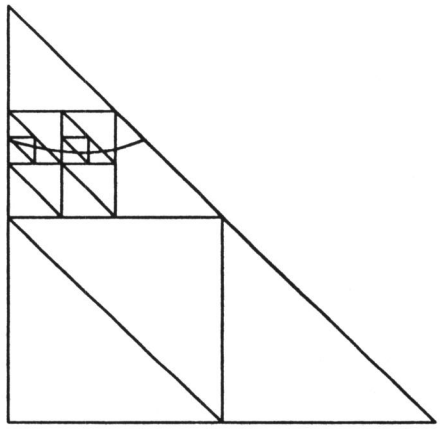


Figura 10: Algoritmo de dividir e conquistar para determinação de curvas de nível.

### 5.2 Grade Regular

Uma grade regular é uma disposição matricial de pontos onde os valores $x$ e $y$ estão espaçados uniformemente. A geração de uma grade regular implica em resolver o seguinte problema: dado $X$ e $Y$, determinar o $Z$ da superfície. Deve-se determinar os parâmetros $u$ e $v$, tais que:

$S(u,v).x = X$

$S(u,v).y = Y$

Seria um problema de resolução de sistemas de equações não lineares (raízes). No entanto, devido a utilização de superfícies triangulares *não paramétricas*, existe uma solução geométrica simples: obter o parâmetro $u$ (ou $v$) pela intersecção da reta paralela a um dos lados e que passa por $X, Y$ com uma aresta do triângulo no plano $xy$ (ver figura 1).

Para obter todos os valores amostrais da grade regular pertencente ao domínio de uma superfície de Bezier, foi utilizado um algoritmo baseado no de varredura de linha para preenchimento de polígonos.

## 6 Resultados e Conclusões

A figura 11 mostra um resultado do algoritmo da triangulação de Delaunay. A figura 12 mostra um gráfico de desempenho deste algoritmo: os pontos foram gerados randomicamente e o computador usado foi o *Sun sparc 2*. Verifica-se que a performance do algoritmo é linear em função do número de pontos e que o algorítimo é bastante rápido (85 s para 100 mil pontos (0.8/1000) )

A figura 13 mostra as curvas de nível obtidas do modelo de um terreno em linhas contínuas e as curvas de níveis da qual originaram os pontos de entrada do modelo, em tracejadas. A figura 14 mostra uma grade regular do mesmo terreno.

Uma das aplicações do algoritmo de geração de grade regular é a confecção de mapas altimétricos coloridos, bastando, para tanto, associar uma cor à ordenada $z$ e incrementar $x$ e $y$ para plotagem ponto a ponto, de modo a preencher regiões.

O modelo de terreno por superfícies triangulares de Bezier apresentado interpola os pontos fornecidos e apresenta uma boa suavização.

## 7 Referências


AHO, A., HOPCROFT, J. E. and ULLMAN, J. D., *Data Structure and Algorithms*, Reading, Addison-Wesley, 1983.

CHOI, B. K., SHIN, H. Y., YON, Y. I. and LEE, J. W., Triangulation of Scattered Data in 3D Space, *Computer Aided Design*, **20**(5): 239-248, June, 1988.

De FLORIANE, L., Surface Representation Based on Triangular Model, *The Visual Computer*, **3**(1):27-50, February, 1987.

FARIN, G., *Curves and Surface for Computer Aided Geometric Design A Practical Guide*, San Diego, Academic Press, 1988.

FARIN, G., Smooth Interpolation to Scattered 3D Data, in: BARNHILL, R. E. and BOHM, W., *Surface in Computer Aided Design*, pp.: 272-282, 1983.

FILIP, D. J., Adaptative Subdivision Algorithms for a Set of Bezier Triangles, *Computer Aided Design*, 7(2):74-78, March, 1986.

GOLDMAN, R. N., Subdivision Algorithms for Bezier Triangles, *Computer Aided Design*, **15**(3):27-36, 1983.

HOROWITZ, E., *Fundamentos de Estrutura de Dados*, Rio, Editora Campus, 1987.

HOUGHTON, E. G., EMNETT, R. F., FACTOR, J. D and SABHARWAL, C. L. Implementation of a Divide-and-Conquer method for Intersection of Parametric Surfaces, *Computer Aided Geometric Design*, bf 2:173-183, 1985.

LASSER, R. D. Intersection of Parametric Surfaces in the Bernistein-Bezier Representation, *Computer Aided Design* , **18**(4), 1986.

LEE, D. T. and SCHACHTER, B. J., Two Algoritm for Constructing a Delaunay Triangulation, *International Journal of Computer and Information Sciences*, 9(3):219-242, 1980.

SAKUDE, M. T. S. Determinação de Curvas de Nível em Terrenos Representados por Superfícies Bicúbicas, **Sibgrapi 89** , Águas de Lindóia, 524-532, 1989.

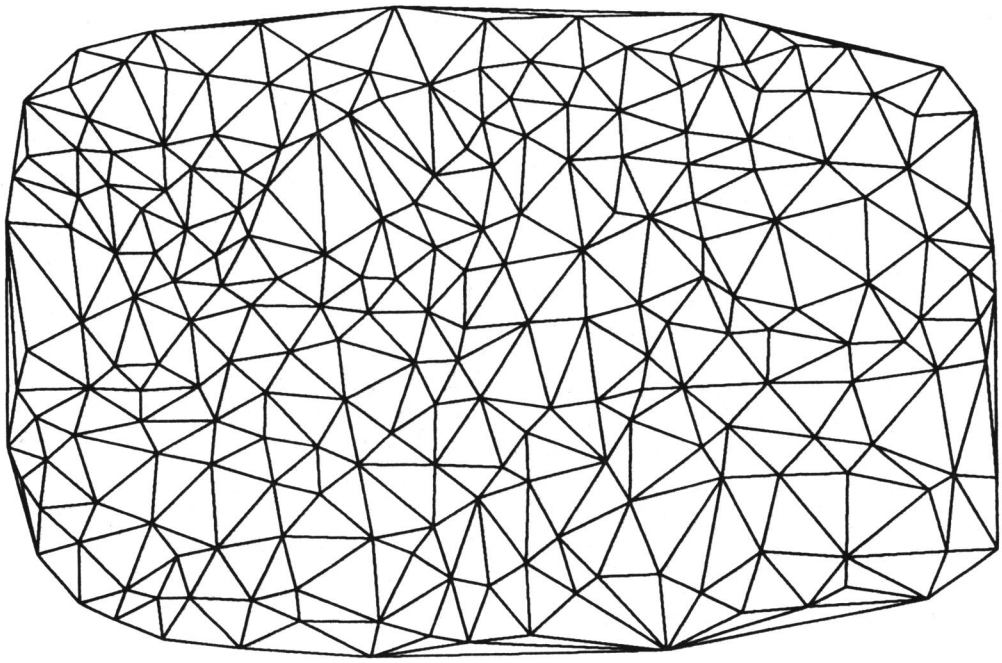


Figura 11: Triangulação de Delaunay.

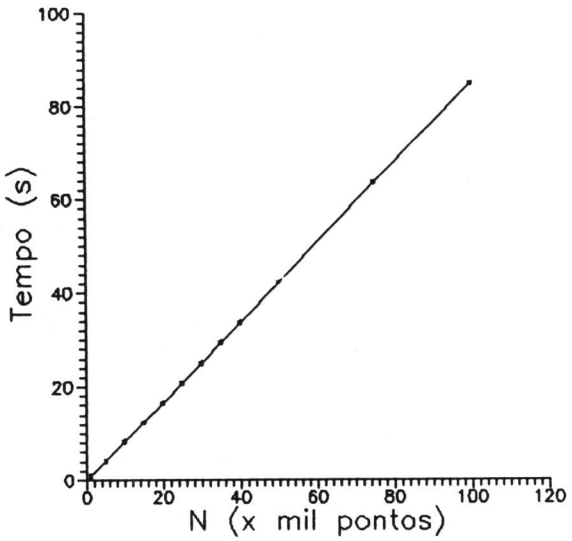



Figura 12: Desempenho do algoritmo de triangulação.

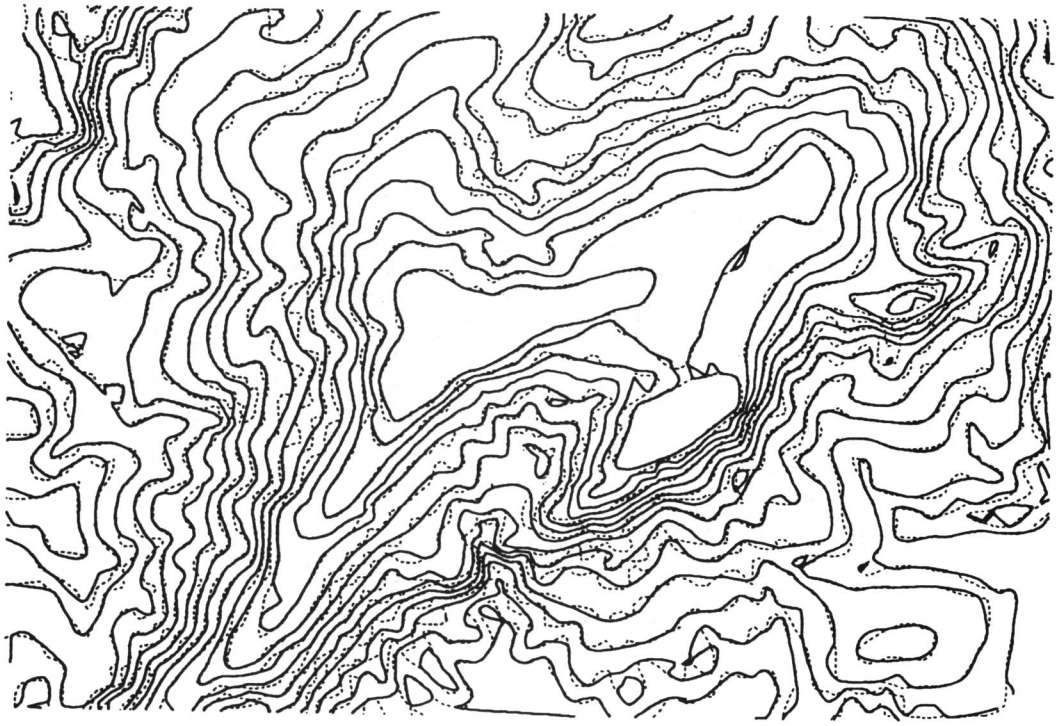


Figura 13: Curvas de nível obtidas pelo algoritmo (linha contínua) e curvas de nível de um mapa digitalizado (tracejado).

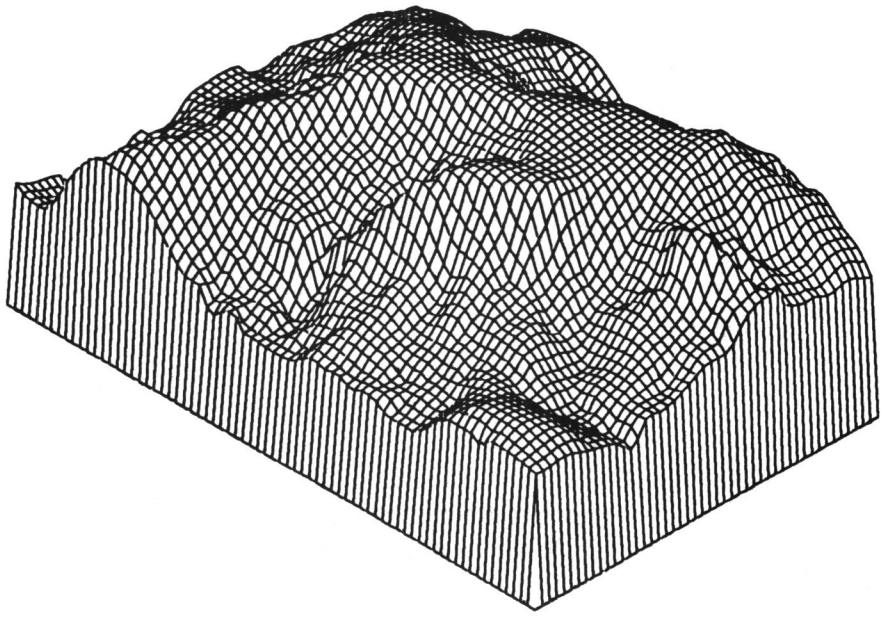


Figura 14: Grade regular do terreno.