

DETECÇÃO SEMI-AUTOMÁTICA DE VEGETAÇÕES A PARTIR DE IMAGENS DE SATÉLITE

Rafael Moreira Savelli

Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio
Grupo de Tecnologia em Computação Gráfica - TECGRAF
Rua Marquês de São Vicente, 225 - Rio de Janeiro, RJ - 22453-900
savelli@tecgraf.puc-rio.br

Roberto de Beauclair Seixas

Instituto Nacional de Matemática Pura e Aplicada - IMPA
Laboratório de Visualização e Computação Gráfica - VISGRAF
Estrada Dona Castorina, 110 - Rio de Janeiro, RJ - 22460-320
rbs@impa.br

Abstract

This work presents how we could construct featured 3D terrains using information gathered from their satellites images. Those information are keys for an image-database search. These searches use wavelets for minimize disk space and improve searching speed. In order to compose the elements with trees or bushes, for example, authors use the common billboard concepts embedded with transparent technique named alpha-channel.

1. Introdução

Ambientes virtuais e visualizações 3D têm contribuído muito para o avanço da humanidade. Em especial no campo das forças armadas. Esses recursos da computação gráfica permitem que uma certa região possa ser visualizada e estudada sem ter que, necessariamente, fazer uma visita ou simulação física neste ambiente. Isso poupa, no mínimo, tempo e dinheiro com treinamentos e exercícios. Entretanto, para isso acontecer é necessário que essas representações sejam bem feitas e com o máximo de realismo possível. Assim, o modelo se aproximará bastante ao terreno real e o exercício no computador poderá substituir o realizado fisicamente.

Assim, o que esse trabalho propõe é justamente como melhorar esse realismo utilizando somente informações contidas em imagens de satélites, detectando e posicionando automaticamente os elementos do terreno, aumentando o realismo de visualizações em três-dimensões.

2. Modelagem do terreno

Os terrenos analisados aqui possuem acidentes naturais como montanhas, rios, lagos e vegetações. Assim, deseja-se representar todos esses itens com o máximo de realismo, pois estamos considerando que esses acidentes naturais são relevantes para o estudo e análise do terreno.

Para representar montanhas, uma camada (*layer*) foi adicionada ao terreno, denotando as curvas hipsométricas. Em outras palavras, dado um par de pontos (x,y) do terreno, a camada

retorna um valor para z , denotando a altura naquele ponto fornecido. Com essa camada definida para todos os pontos do terreno 2D, conseguimos uma malha de triângulos (*wireframe*).

Para representar rios e lagos, uma textura com essas informações é aplicada em cima da malha de triângulos, mostrando através de cores a posição desses elementos. Essa textura no nosso caso foi à própria imagem de satélite da região.

A figura 1 mostra como compusemos o terreno final através da junção da malha de triângulos (gerada com o mapa de alturas) com a textura aplicada. [1]

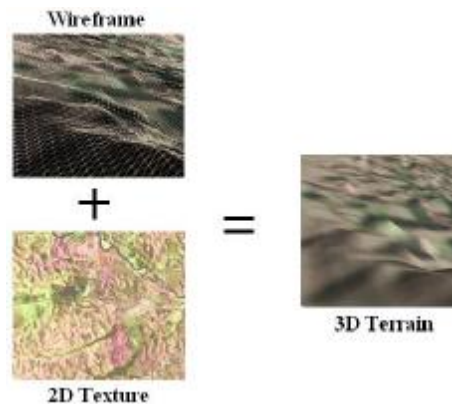


Figura 1: A composição do terreno 3D

Como podemos ver, algoritmos de visualização de terrenos são complexos. Eles tem que carregar informações de altura e de textura para dar um visual agradável ao usuário. Assim, na última década, esse assunto tem recebido uma grande atenção de pesquisadores e interessados na área da computação gráfica no intuito de melhorar cada vez mais a interatividade. Como consequência, diversas estratégias foram criadas e destacou-se, dentre elas, aquela criada por Lindstrom e Pascucci [2,3].

Até agora conseguimos um terreno com somente montanhas e lagos. Nada fora criado até então para representar as vegetações. Para estas, uma técnica chamada *billboard* foi inserida e sua explicação decorre nos próximos itens.

3. Billboard

Imagine ter que modelar em três dimensões uma árvore ou um arbusto. Esta tarefa não é fácil, pois envolve uma geometria muito complicada. Além disso, vegetações normalmente ocupam grandes áreas do terreno o que pode acarretar em uma performance ruim do computador com a réplica de várias árvores com um número enorme de polígonos. Por esse motivo, optamos por utilizar *billboards* ao invés da tradicional modelagem de objetos. O *billboard* é uma geometria bem mais simples (normalmente um quadrado) que porta uma textura do que se deseja representar. Além disso, essa estrutura deve estar sempre orientada para o observador para produzir tal efeito, reposicionando-se sempre que o usuário se mover. Essa propriedade pode ser facilmente obtida através de transformações simples representada pela rotação no eixo z . [4] Um esquema do que representa um *billboard* pode ser observado na figura 2 a seguir:

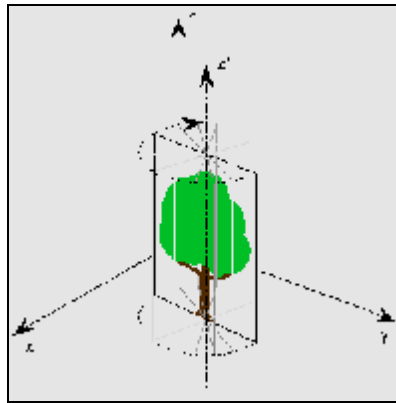


Figure 2: A billboard and its axis.

Para aumentar ainda mais o realismo, adicionamos uma camada de *alpha-channel* na imagem da textura aplicada a esse *billboard*. Com isso, obtivemos uma imagem com transparência nas bordas. Outro recurso que nós pesquisamos foi o uso de *blur*. [5] Acontece que na prática esse recurso não funcionou muito bem.

4. Digital satellite image

A imagem de satélite digital é simplesmente uma foto tirada do nosso planeta pelos seus satélites orbitais. Este trabalho utiliza imagens de satélite para fazer uma segmentação de seus pixels em busca de padronizações que caracterizem uma vegetação. Essa imagem também precisa ser georeferenciada para que essas vegetações possam ser posicionadas corretamente no terreno.

Assim, foi implementado um algoritmo de pre-classificação no qual chamamos de “buildDB” (mostrado na figura 3). Essa aplicação implementa o algoritmo *split-and-merge* na qual procura por regiões homogêneas através da divisão recursiva da imagem de satélite em quatro partes iguais. O critério de parada acontece quando um dado *threshold* é atingido, considerando aquela divisão uma região dita homogênea. [6]

Ainda nessa figura 3 temos um típico resultado de um *split-and-merge* feito em cima de uma imagem de satélite da região de *Macaé, RJ*. A aplicação foi desenvolvida com as bibliotecas IUP (Portable User Interface) [7], CD (Canvas Draw) [8] e IM (Imaging Tool) [9]. Todas essas bibliotecas foram criadas e são mantidas até hoje pelo laboratório de computação gráfica chamado TeCGraf/PUC-Rio. [10]

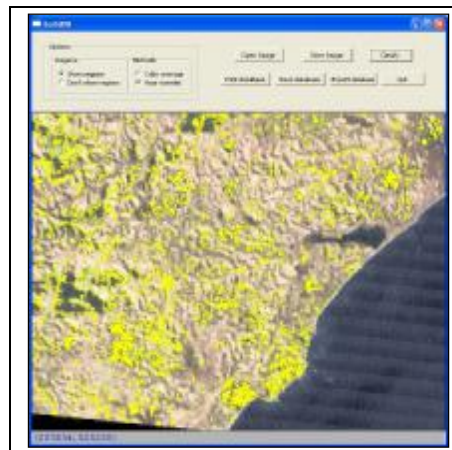


Figura 3: Split-and-merge na região de Macaé, RJ. Foram obtidas 803713 regiões homogêneas

Duas abordagens foram consideradas para obtenção da região homogênea: a média simples e por *haar-wavelet*. A primeira abordagem pára quando a média de cada componente RGB satisfaz a seguinte relação:

$$\text{média} = \text{soma} / \text{tamanho_região}$$

Onde soma representa a soma de todos os valores dos *pixels* presentes dentro de **tamanho_região**. Assim, quando essas médias R, G, B atingem um determinado limite (*threshold*) o algoritmo interrompe a divisão recursiva e considera aquela região como sendo homogênea.

O processo é semelhante para o algoritmo de wavelet descrito por *Haar* conforme descrito no tópico 6.

O resultado do *split-and-merge*, portanto, é uma imagem segmentada em regiões com propriedades similares. Essas regiões foram classificadas e catalogadas em um banco de dados. Além disso, algumas dessas regiões tiveram um *billboard* associado a elas de modo representá-las no terreno 3D.

Depois de várias tentativas, nós concluímos que o *split-and-merge* funcionou muito bem para imagens de satélite com resoluções iguais ou próximas a 1:25.000.

5. Banco de dados

Conforme dito no item anterior, as regiões obtidas do *split-and-merge* são classificadas e salvas num banco de dados. Claro que algumas dessas regiões não serão relevantes para nós, visto que estamos somente interessados em regiões que denotam algum tipo de vegetação (pois os outros itens como rios e montanhas já estão presentes na textura e no mapa de alturas). Essas regiões que não interessam a nós foram descartadas e não entraram no banco de dados.

Para cada região, precisamos salvar também as posições em que elas aparecem na textura bem como o *billboard* que melhor irá representá-la.

Quando a aplicação de visualização 3D é inicializada, ela obtém as informações de vegetação diretamente do banco de dados e posiciona corretamente no terreno 3D.

6. Wavelets

Com o intuito de identificar um elemento a partir da imagem de satélite um algoritmo de segmentação teve que ser aplicado. Entretanto, decidimos aplicar também uma outra técnica diferente daquela descrita por média simples para servir de critério de parada para o *split-and-merge*. Chama-se *wavelet*.

Wavelets são úteis em várias aplicações e em diversas áreas do conhecimento. Uma dessas aplicações estão relacionadas com processamento de sinal e imagem. No processamento de imagens o *wavelet* ainda pode ser utilizado como uma forma alternativa de se economizar espaço em disco através da compressão de uma imagem.

Hoje, existem variantes de *wavelets* como Daublet, Gabor, Laplacian, Haar e outros. Nós escolhemos o algoritmo de *Haar* por duas razões: primeiro por sua simplicidade de implementação e segundo porque funcionou muito bem nos nossos testes com imagens de satélites distintas.

A seguir, segue o algoritmo de *Haar-wavelet* [11].

Imagine que você disponha de uma imagem RGB com dimensões 4x4 *pixels* apenas. Cada componente de cor R, G e B podem assumir valores que vão de 0 à 255. Como ilustração,

mostraremos o que acontece com a componente vermelha dessa imagem aplicando o algoritmo de *wavelet*.

| | | | | |
|------|-----|-----|----|----|
| RED: | 0 | 1 | 2 | 3 |
| | 254 | 137 | 56 | 13 |

Assim, para cada par do vetor, calcula-se a média entre eles e coloca-se o resultado em um novo vetor do mesmo tamanho da imagem e inicialmente vazio. Assim:

$$\left\lfloor \frac{RED[0] + RED[1]}{2} \right\rfloor = \left\lfloor \frac{254 + 137}{2} \right\rfloor = 195 \quad \text{e} \quad \left\lfloor \frac{RED[2] + RED[3]}{2} \right\rfloor = \left\lfloor \frac{56 + 13}{2} \right\rfloor = 34$$

Com essas operações, o novo vetor se torna:

| | | | | |
|-------|-----|----|---|---|
| NOVO: | 0 | 1 | 2 | 3 |
| | 195 | 34 | | |

O próximo passo é preencher os campos 2 e 3 do vetor novo e isso se faz calculando apenas as diferenças conforme mostrado abaixo:

$$NOVO[2] = RED[0] - NOVO[0] = 254 - 195 = 59 \quad \text{And} \\ NOVO[3] = RED[2] - NOVO[1] = 56 - 34 = 22$$

Resultando em,

| | | | | |
|-------|-----|----|----|----|
| NOVO: | 0 | 1 | 2 | 3 |
| | 195 | 34 | 59 | 22 |

Até esse ponto, o algoritmo de *Haar* terminou somente uma iteração. Agora, é preciso repetir esses passos descritos em cima para a primeira metade do vetor (posições 0 e 1). Ao se fazer isso, o novo vetor toma o aspecto:

| | | | | |
|-------|-----|----|----|----|
| NOVO: | 0 | 1 | 2 | 3 |
| | 114 | 81 | 59 | 22 |

Para esse exemplo de componente vermelha da imagem o algoritmo de *wavelet* está terminado. Entretanto é preciso fazer essa etapa para todas as outras componentes.

7. Juntando as informações

A idéia principal envolvida nesse trabalho é simplificada como descrita na figura 3 a seguir:

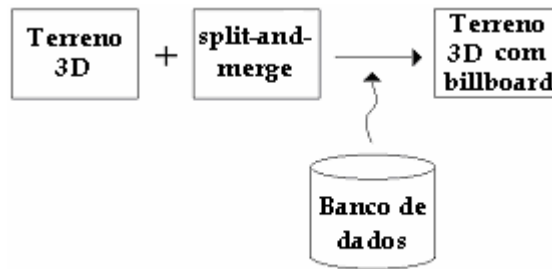


Figura 3: O processo completo para construção do terreno.

Do presente momento até agora vimos que, dado um terreno 3D e sua imagem de satélite, conseguimos extrair a vegetação deste último e inserimos no terreno 3D. O resultado visual foi muito bom conforme podemos ver no nosso caso de estudo.

8. Caso de estudo e resultados

Os algoritmos de construção do terreno com vegetação foi aplicado basicamente a dois terrenos diferentes com imagem de satélites também diferentes. Tratam-se de regiões do Brasil conhecidas como *Macaé, RJ* e *Itaoca, ES*. Essas regiões são visualizadas em um ambiente de visualização onde o usuário tem a impressão de estar observando a região por meio de um voo de helicóptero. As imagens de satélites foram obtidas gratuitamente do site da Embrapa através do site oficial da instituição. [12] Os resultados finais são mostrados a seguir onde, das figuras 5 a 7, trata-se de *Itaoca, ES*. Já as figuras de 8 a 10 tratam-se de *Macaé, RJ*.

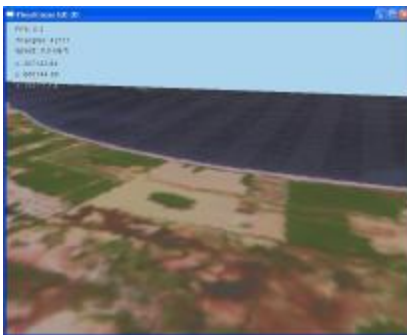


Figura 5: terreno vazio (*Itaoca, ES*)



Figura 6: Média simples (*Itaoca, ES*)

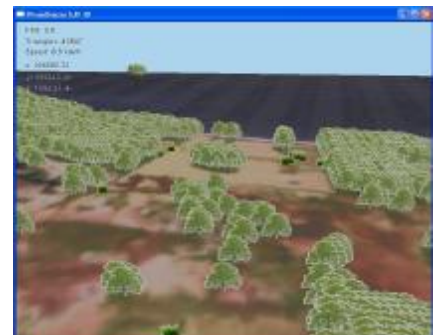


Figura 7: Haar-wavelet (*Itaoca, ES*)



Figura 8: terreno vazio (*Macaé, RJ*)

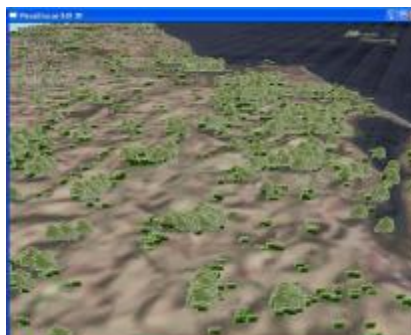


Figura 9: Média simples (*Macaé, RJ*)

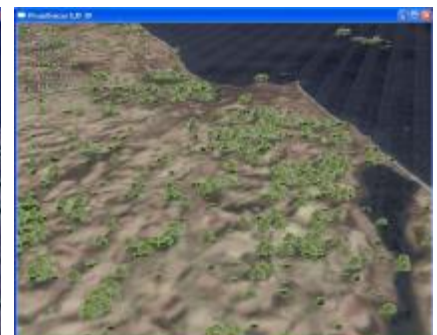


Figura 10: Haar-wavelet (*Macaé, RJ*)

9. Conclusão

No processo de classificação, o *wavelet* mostrou que pode fazer um bom trabalho para a

classificação de vegetações. De um modo geral, ele até supera um pouco o método de médias. Assim, obtivemos um excelente ganho de realismo visto que a adição de *billboards* com camadas *alpha-channel* deram os aspectos esperados.

Note que a aplicabilidade desse tipo de visualização é enorme e no campo militar podemos citar algumas como: exercício de reconhecimento aéreo e instruções de camuflagem sobre vegetações.

10. Trabalhos Futuros

Como pudemos ver nesse trabalho, os *billboards* acrescentaram realismo à visualização, entretanto algumas características intrínsecas do *billboard* geram também outros novos problemas. Como por exemplo, podemos citar o caso em que o helicóptero se encontra bem em cima de um *billboard* de modo que ao tentar visualizá-lo, nada se verá (ou quase nada). Isso ocorre porque os *billboard* somente giram em torno do eixo Z, não podendo ser girados no caso perpendicular. Para resolvermos esse problema, poderíamos propor uma mudança suave entre um billboard e outro paralelo ao terreno, mostrando, por exemplo, somente dados da copa de árvores (já que esses últimos escondem o tronco quando visto de cima).

Outro problema diz respeito a qualidade da imagem utilizada para as texturas dos billboards. Em geral elas são de baixa resolução (para pouparem memórias), mas se o helicóptero se aproximar muito desses *billboards*, a imagem vista pode não ser satisfatória. Nesse caso, uma solução possível seria utilizar *multi-textura*, trocando-a por outras mais detalhadas dependendo da visão do usuário (chamados *view-dependent*)

11. Referências

- [1] Poyart, E.; Frederick, P.; Seixas, R.B.; Gattass, M.; [Simple Real-Time Flight Over Arbitrary-Sized Terrains](#), pre-print submitted to Workshop Brasileiro de GeoInformática, 2002.
- [2] Lindstrom, P.; Pascucci, V. Visualization of Large Terrains Made Easy. Proceedings of IEEE Visualization 2001, San Diego, California, October, 2001, pp. 363-370.
- [3] Lindstrom, P.; Pascucci, V. Terrain Simplification Simplified: A General Framework for View-Dependent Out-of-Core Visualization. IEEE Transactions on Visualization and Computer Graphics, May 8, 2002.
- [4] Advanced Graphics Programming Techniques Using OpenGL, SIGGRAPH `98 Course.
- [5] OpenGL Programming Guide: The Official Guide to Learning OpenGL.
- [6] Digital Image Processing Algorithms and Applications, 282–297, 2000.
- [7] IUP, <http://www.tecgraf.puc-rio.br/iup> (10/03/2004).
- [8] CD, <http://www.tecgraf.puc-rio.br/cd> (10/03/2004).
- [9] IM, <http://www.tecgraf.puc-rio.br/im> (10/03/2004).
- [10] TeCGraf/PUC-Rio: www.tecgraf.puc-rio.br.

- [11] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Communications on Pure and Applied Mathematics*, 44:141–183, 1991.
- [12] Brazilian Digital Satellite Images: <http://www.cdbrasil.cnpm.embrapa.br/>