ALGORITMOS PARA EXIBIÇÃO DE IMAGENS COLORIDAS

João Luiz Dihl Comba

Cláudio Esperança

Ronaldo C. Marinho Persiano

COPPE-Sistemas, UFRJ
Caixa Postal 68511
Rio de Janeiro - RJ - 21945

RESUMO: A exibição de imagens coloridas é uma das tarefas mais importantes em ambientes de sínteses de imagens. Algoritmos como "Ray-Tracing" geram normalmente espectros cromáticos muito além do representável nos dispositivos usualmente utilizados. Faz-se necessário, portanto, uma análise deste espectro, procurando produzir uma imagem que mantenha a maior representatividade em relação à imagem inicial. São discutidas diversas abordagens ao problema, discutindo-se vantagens e desvantagens de cada proposta.

1. INTRODUÇÃO

Um dos problemas mais interessantes de síntese de imagens refere-se à exibição de imagens coloridas. É comum trabalharmos com dispositivos que possuem uma tabela de cores de tamanho reduzido, especialmente se compararmos com a gama de cores a representar. No nosso caso (SUN 3/60, 3/260), temos uma tabela de cores capaz de representar 256 cores em um dado instante, dentre 16 milhões de cores possíveis.

Com esta limitação, o processo de exibição torna-se não trivial, dependendo de uma prévio processo de análise. Duas fases mostram-se importantes neste processo:

- Quantização: Escolha da tabela de cores que melhor represente a população da imagem
- Seleção: Selecionar para cada pixel da imagem a cor da tabela que o melhor representa

Estas etapas são descritas com mais detalhes nas seções seguintes.

2. PROCESSO DE ANÁLISE DA IMAGEM COLORIDA

2.1. Quantização

Neste processo devemos escolher a tabela de cores com que iremos trabalhar. Seja:

- C a tabela de cores,
- K a dimensão desta tabela,
- *RGB (Red-Green-Blue)* o espaço de cores representáveis,
- I *(Largura, Altura)* como sendo a imagem que se deseja analisar.

O Problema consiste em gerar a partir de uma imagem I uma tabela de cores C de dimensão K, onde cada Cor possue componentes *RGB*.

Três algoritmos de quantização foram implementados: Análise da População (Heckbert [3]), Corte Pela Mediana (Heckbert [3]) e Corte Pela Mediana com Média Espacial.

O Algoritmo da População ("Popularity Algorithm") parte da seguinte premissa: gerar a tabela de cores com representantes das regiões mais densas do espaço de cores da imagem. Isto é, a tabela de cores deve conter as K cores de maior frequência na imagem. Para tanto, deve-se:

- Montar uma tabela associando a cada cor da imagem a frequência de ocorrência com que ela aparece na imagem
- Gerar um Histograma de Frequência (Classificação da Tabela anterior pela frequência)

Embora apresente alguns bons resultados, este algoritmo não é adequado para imagens com espectros cromáticos muito amplos. Isto ocorre porque regiões da imagem com cores muito diferentes não possuem em geral altas frequências de ocorrência, não sendo por isso incluídas na tabela de cores.

O Algoritmo de Corte pela Mediana ("Median Cut Algorithm") já apresenta uma proposta mais elaborada. Da mesma forma como o algoritmo da População, um histograma de frequência é montado. Usando este histograma, o espaço RGB é subdividido em *K* caixas retangulares, usando a estratégia de Corte pela Mediana. Esta estratégia procura dividir as cores da imagem em subdivisões, colocando dentro de cada uma um subconjunto de cores da imagem. Para que tenhamos subdivisões que representem a imagem de forma mais distribuída, o critério de corte é definido pela mediana da componente (RGB) de maior variação na imagem. Esta estratégia pode ser explicada com mais detalhes no seguinte pseudocódigo:

```
CortePelaMediana (HistFrequencia)

  Seja
    HF [1 .. N] o Histograma de Frequencia com N cores diferentes

  Define-se uma subdivisão por:
    Subdivisão S:
      I1 (* Indice do Histograma onde comecam as cores de S *)
      I2 (* Indice do Histograma onde terminam as cores de S *)
      NCorDif   (* Numero de Cores Diferentes na subdivisao *)
      NCorTotal (* Numero total de ocorrencias de cores da subdivisao *)
      MenorCor  (* Valores minimos RGB da subdivisao *)
      MaiorCor  (* Valores maximos RGB da subdivisao *)

  (* A primeira subdivisao contem todas as cores *)
    CriaSubdivisao (1, N)
    NumeroSubdivisoes = 1
    REPEAT
      S = EscolhaSubdivisaoComMaisCores ;
      IF S.NcorDif = 1 THEN (* Ja Subdividiu o possivel *)
        EXIT ;
      IndiceRGB = ComponenteRGBMaiorVariacao (S)
      OrdenaTabelaFrequencia (IndiceRGB, S.I1, S.I2)
      Mediana = CalculaMediana (S)
      CriaSubdivisao (S.I1, Mediana)
      CriaSubdivisao (Mediana, S.I2)
      Incrementa (NumeroSubdivisoes)
    UNTIL NumeroSubdivisoes = K ;
```

Após a geração das *K* subdivisões, deve-se escolher um representante para cada uma delas. Para tanto, usa-se a média das cores existentes em cada subdivisão, ponderadas por suas frequências de ocorrências:

$$Representante^{RGB} = \frac{\sum_{i=SI1}^{SI2} HF[i].Cor^{RGB} * HF[i].Frequencia}{\sum_{i=SI1}^{SI2} HF[i].Frequencia}$$

Em imagens complexas este algoritmo produz resultados sensivelmente melhores que o algoritmo da População. Entretanto, cores com baixa frequência e distantes das cores da tabela são mal representadas. Isto ocorre porque este algoritmo analisa somente as frequências das ocorrências das cores da imagem, sem levar em consideração como as cores estão distribuídas na imagem. Com isto duas imagens completamente diferentes, mas possuindo a mesma população de cores, irão gerar a mesma tabela de cores.

A idéia básica para melhorar o algoritmo de Corte pela Mediana consiste em torná-lo orientado à imagem, isto é, levando em conta informações do espaço da imagem. Para tanto, para cada ponto da imagem é preciso fazer uma análise de sua vizinhança, utilizando um determinado critério para embutir tal informação.

Inicialmente devemos calcular, para cada pixel da imagem, uma nova cor, resultante da média ponderada dos vizinhos ao pixel (*Cor Induzida*) . A tabela 1 mostra a matriz de pesos associados aos vizinhos de um pixel.

1	2	4	2	1
2	4	8	4	2
4	8	p		

Tabela 1 - Pesos Associados aos Vizinhos de p

A *Cor Induzida* é dada pela seguinte fórmula:

$$CorInduzida^{RGB}(p) = \frac{\sum_{i=1}^{12} Imagem^{RGB}(Vizinho(p,i)) * Peso\ (i)}{\sum_{i=1}^{12} Peso\ (i)}$$

Vizinho (p, i) retorna as coordenadas do i-ésimo vizinho de p na imagem

A *Cor Induzida* é, portanto, uma estimativa da cor do pixel na imagem induzida por sua vizinhança. Para verificar o quanto esta estimativa difere da *Cor da Imagem* aplica-se uma métrica no espaco RGB (ex. Métrica Euclideana, Métrica de "Manhattan"), obtendo assim uma relação de distância entre as cores. É fácil verificar que para áreas homogêneas da imagem esta distância será reduzida, pois a *Cor Induzida* será muito próxima da *Cor da Imagem*. Em contrapartida, em regiões da imagem que contenham diversas tonalidades e com brusca mudança entre elas (em "Ray Tracing" ocorrem nas áreas de reflexão especular) teremos valores altos de distâncias.

É fato que regiões homogêneas da imagem precisam menos cores para representá-las do que regiões heterogêneas. Usando esta observação, podemos criar um mecanismo que permita que cores pertencentes à regiões heterogêneas da imagem possuam maior poder de contribuição para a geração da tabela de cores. Isto é feito alterando-se a geração da tabela de frequência de ocorrências.

Anteriormente, a frequência de cada cor correspondia ao número de ocorrências que ela aparecia na imagem (ocorrência unitária). Agora, iremos colocar um novo tipo de ocorrência, proporcional a distância entre a *Cor da Imagem* e a *Cor Induzida*, de tal forma que cores em regiões homogêneas terão reduzidos números de ocorrências, enquanto que cores em regiões heterogêneas terão altos números de ocorrências.

O novo número de ocorrências é dado por:

$$NovaOcorrencia = Distancia\ (CorInduzida, CorImagem)$$

onde
Distancia (cor1, cor2) = Max ($|cor1^{RGB} - cor2^{RGB}|$)

Esta nova proposta, a que chamamos de Média Espacial, mostrou-se muito eficiente para melhorar a qualidade do algoritmo Simples de Corte pela Mediana. A maior diferença nos resultados produzidos pelos algoritmos Simples e Com Média Espacial são encontrados, no caso de Imagens de "Ray Tracing", em regiões onde ocorreram reflexões e refrações (principalmente em sólidos de grande curvatura, como elipsóides). Nestas regiões, observou-se que uma melhor aproximação foi produzida pelo algoritmo de Corte pela Mediana com Média Espacial.

2.2. Seleção

Neste ponto temos definida nossa tabela de cores. Para gerarmos a imagem, devemos para cada ponto encontrar na tabela de cores a cor que o represente melhor. Uma solução consiste em representá-la pela cor da tabela de cores que minimiza a seguinte função:

DistanciaMinima = Min (Distancia (CorImagem, CoresdaTabela))

A maneira mais imediata de executar esta tarefa consiste em se fazer, para cada pixel da imagem, uma busca extensiva na tabela de cores, procurando encontrar a cor da tabela mais próxima da cor em questão. Este método é bastante ineficiente, visto que teremos *(Largura * Altura * K)* testes de distância. Para uma imagem de 400 * 400 pontos, teremos 40.960.000 testes. Analisando o problema, notamos que muitos testes consideram cores com poucas chances de serem ótimas. Torna-se claro que este é um típico caso de usarmos técnicas de subdivisão espacial para reduzir o conjunto de canditatos a busca exaustiva.

O uso de subdivisão consiste, neste caso, em duas etapas:

- Subdivir o Espaço RGB em $n <= K$ partições,
- Localizar para cada ponto da Imagem a partição em que este está localizado, usando a cor representante da partição como aproximação do ponto.

A subdivisão ideal é aquela que divide o espaço em K partições, e cujo representante é a cor mais próxima de todas as cores existentes dentro do espaço subdividido. As partições de VORONOI produzem esta subdivisão ideal, entretanto o cálculo dos poliedros que definem as partições, bem como a consequente localização de pontos nestes poliedros, é bastante custosa computacionalmente. Faz-se necessário, portanto, estudar-se outras técnicas de subdivisão.

Outra alternativa de subdivisão possível consiste em dividir as K cores da tabela seguindo a mesma estratégia de subdivisão adotada anteriormente (Corte Pela Mediana). Com isto teríamos partições retangulares, e o problema de encontrar a cor mais próxima seria apenas um problema de localizar a cor dentro da subdivisão retangular a qual ela pertencesse. Esta solução em primeira análise pode parecer razoável, entretanto, em certos casos podem produzir resultados altamente indesejáveis. A figura 1 mostra um caso destes, onde a Cor será erradamente aproximada para a cor C3, quando deveria ser aproximada para C4, pois a distância d1 entre a Cor e C4 é menor que a distância d2 entre a Cor e C3.

Outra alternativa pensada foi a de criar subdivisões com mais cores, ao invés de apenas uma como anteriormente. Assim, após o processo de localização teríamos um número n de cores, e a cor mais próxima seria aquela que tivesse a menor distância (neste caso bem menos testes de distância seriam feitos). Embora produza melhores resultados que a estratégia anterior, ainda assim podem acontecer casos como o apresentado na Figura 1.

Outro método de subdivisão que resolve com exatidão este problema, sem no entanto gerar a subdivisão ideal, usa subdivisões regulares. A idéia consiste em subdividir o espaço RGB em um malha regular M de células de tamanho L , e montar para cada uma das células desta malha uma lista das cores que possuem chances de serem mais próximas de um dos pontos dentro da célula. Este é um problema essencialmente geométrico, cuja solução é a seguinte:

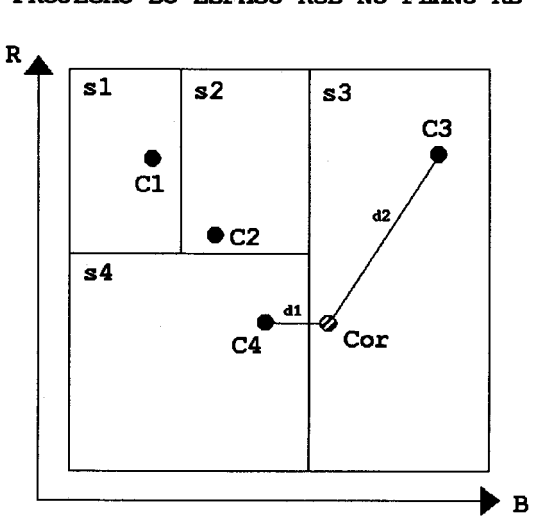

Figura 1 - Problemas na subdivisão da tabela de cores

- Encontrar para cada célula P da malha a cor da tabela de Cores com menor distância até o centro de P (Cor Mais Próxima)
- Encontrar o Lugar Geométrico dos Pontos Cuja Distância D a um ponto dentro da célula P seja menor ou igual a Distancia D' da Cor Mais Próxima a P
- Colocar na lista de possíveis cores associada à célula da malha todas as cores da tabela de cores que pertencerem e este lugar geométrico

A figura 2 mostra o Lugar Geométrico dos pontos obtido.

Calcular se um ponto está dentro deste Lugar Geométrico torna-se ineficiente. Usa-se um Conjunto Abrangente de Pontos, que engloba o Lugar Geométrico anterior, e cujo teste torna-se bem mais simples (Figura 3).Embora não resulte numa lista mímima, não causa problemas de eficiência nem introduz erros.

Esta estratégia é bastante eficiente, visto que as listas associadas a cada célula podem ser calculadas somente quando existirem cores dentro da célula, o que não é em geral o caso para muitas células da malha.

Resultados ainda melhores serão obtidos quando aplicarmos técnicas de "Dithering" (Ver Pins [5]). A idéia consiste em pertubar cada ponto da imagem de forma que padrões irregulares da imagem sejam reduzidos. Devemos executar os seguintes passos para exibir a imagem:

- Gerar para cada *Cor da Imagem* uma *Cor Perturbada*
- Encontrar, usando uma técnica de Seleção, a Cor da tabela que melhor aproxima *Cor Perturbada*

A *Cor Perturbada* gerada pelo algoritmo de "Dithering" é composta de três componentes: *Cor Induzida, Escala e Ruído*. A *Cor Induzida* resulta da ponderação das cores vizinhas a P (calculada da

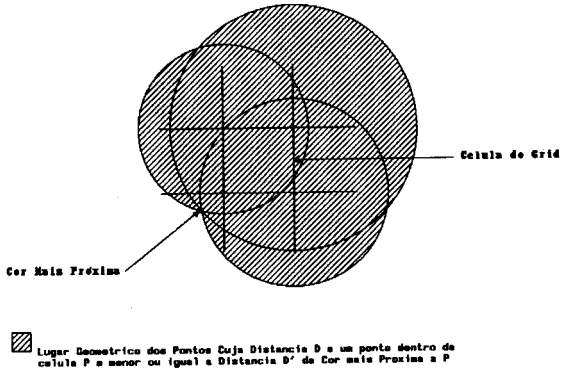

Figura 2 - Lugar Geométrico dos Pontos

mesma forma que no processo de quantização). As funções de *Escala* e *Ruído* são usadas no intuito de evitar eventuais padrões irregulares. Estas funções são as mesmas especificadas por Pins [5], baseadas em critérios empíricos.

A *Escala* é calculada a partir do número de elementos da lista associada à célula da Malha onde a *Cor da Imagem* se encontra:

$$Escala^{RGB}(CorImagem^{RGB}) = \max(3, TamanhoListaCelula(CorImagem^{RGB}))$$

O ruído é uma função randômica, cuja amplitude é dada pela média das distâncias das cores da lista até a cor média da lista:

$$|Ruido^{RGB}| \leq DistanciaMediaCoresNaLista\ (CorImagem^{RGB})$$

A *Cor Perturbada* é dada por:

$$CorPerturbada^{RGB}(CorImagem^{RGB}) = CorImagem^{RGB} + \frac{CorInduzida^{RGB}(CorImagem^{RGB})}{Escala(CorImagem^{RGB})} + Ruido^{RGB}(CorImagem^{RGB})$$

3. RESULTADOS

Para medir a execução dos diversos algoritmos apresentados foi usada a função avaliadora proposta por Pins [5]:

$$Avaliacao\ (I(Largura, Altura)) = \sum_{i=1}^{LarguraAltura} \sum_{j=1} \frac{DistEuclideana(CorOriginal, CorProduzida)}{Largura*Altura}$$

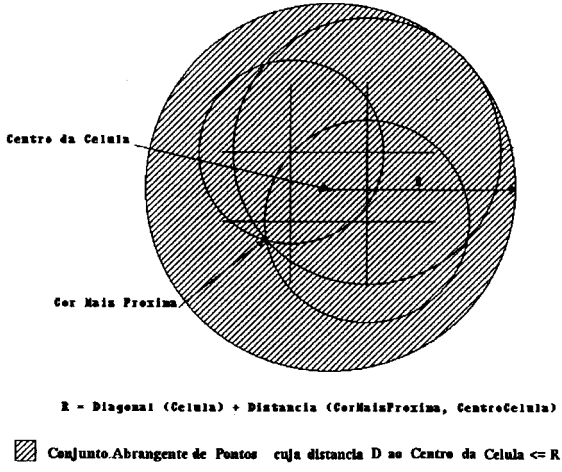

Figura 3 - Conjunto Abrangente de Pontos

Os resultados obtidos foram os seguintes:

Imagem 400 * 400 Pontos - 8031 Cores		
Quantização	Seleção	Avaliação
Algoritmo da População	Subdivisão pela Mediana	8.428
Corte Pela Mediana	Malha, Sem "Dithering"	6.268
Corte Pela Mediana	Malha, Com "Dithering"	7.251
Corte Pela Mediana com Média Espacial	Malha, Sem "Dithering"	5.776
Corte Pela Mediana com Média Espacial	Malha, Com "Dithering"	6.543

Esta medida de avaliação revela resultados não tão bons para o uso de "Dithering" na Seleção. Isto pode ser explicado pelo fato desta medida de avaliação somente analisar as cores da imagem, e não como elas estão dispostas na imagem (diferentemente de uma simples análise visual da imagem). Na análise visual percebe-se que com o uso de "Dithering" os padrões irregulares da imagem são melhor compensados. Uma medida de avaliação mais confiável será, portanto, aquela que levar em consideração a distribuição espacial da imagem.

4. CONCLUSÃO

Neste artigo foram apresentados diversas técnicas para exibição de imagens coloridas em dispositivos com tabelas de cores reduzidas. Através da análise de diversas alternativas, procurou-se buscar a melhor forma de solucionarmos o problema.

Para cada caso podemos analisar quais das alternativas propostas é a mais adequada. Em casos complexos, o uso de técnicas de subdivisão mostrou-se indispensável para a obtenção de imagens com alta fidelidade, tanto no processo de Quantização, como no processo de Seleção.

BIBLIOGRAFIA

[1] Rogers, David F. "Procedural Elements for Computer Graphics," *McGraw Hill Book Company*, 1985.

[2] Floyd, R. W. e Steinberg, L. "An adaptative algorithm for spatial gray scale," *Int. Symposium Dig. Tech. Papers*, 36, 1975.

[3] HeckBert, P. "Color Image Quantization for Frame Buffer Display," *Computer Graphics*, 16(3):297-305, Julho 1982

[4] D. E. Knuth, "Digital halftones by dot difusion," *ACM Transactions on Graphics*, 6(4):245-273, October 1987.

[5] Pins, M e Hild, Hemann. "Variations on a Dither Algorithm," *Eurographics 89*, 381-392, Elsevier Association, 1989.

[6] Samet, Hanan. "The Design and Analysis of Spatial Data Structures," *Addison-Wesley, 90*

Este trabalho foi parcialmente financiado pela CAPES, CNPQ, MCT, FINEP e FIPEC.