UNIVERSIDADE DO OESTE DE SANTA CATARINA
DEPARTAMENTO DE CIÊNCIAS EXATAS
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

COMPARAÇÃO ENTRE ALGORITMOS PARA QUANTIZAÇÃO DE IMAGENS

MONOGRAFIA SUBMETIDA À
UNIVERSIDADE DO OESTE DE SANTA CATARINA PARA A OBTENÇÃO DO
GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

SAULO POPOV ZAMBIASI

CHAPECÓ (SC), NOVEMBRO DE 1998.

COMPARAÇÃO ENTRE ALGORITMOS PARA QUANTIZAÇÃO DE IMAGENS

SAULO POPOV ZAMBIASI

ESTA MONOGRAFIA FOI JULGADA PARA OBTENÇÃO DO TÍTULO DE

BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

ÁREA DE COMPUTAÇÃO GRÁFICA E APROVADA PELO CURSO DE CIÊNCIA DA COMPUTAÇÃO

ORIENTADOR: Prof. Marcos Vinícius Rayol Sobreiro

COORDENADOR DO CURSO: Prof. David Daniel e Silva

BANCA EXAMINADORA

PRESIDENTE: Prof. Marcos Vinícius Rayol Sobreiro

Prof. David Daniel e Silva

Prof. Roberto Carlos Pavan

Prof. Itamar Leite de Oliveira

Agradecimentos

Agradeço a minha família por me ajudar e sempre me incentivar nos estudos. Realmente, eu não teria conseguido chegar até aqui, se não fosse pelo apoio deles.

Gostaria de agradecer aos meus amigos que souberam entender minha dedicação aos estudos e compreender pelos finais de semana que precisei estudar ao invés de estar com eles.

Agradeço ao meu orientador, Marcos Sobreiro, pela dedicação para comigo e pelo auxílio em tornar possível o término deste trabalho.

Agradeço também ao coordenador do meu curso, David D. Silva e a todos professores que me ajudaram e me incentivaram durante todo o curso.

Agradeço ao professor Jonas Gomes por me mostrar o que é a computação gráfica e pelo incentivo que me deu para seguir este caminho.

Agradeço também aos meus queridos colegas de curso que sempre estiveram ao meu lado e se tornaram uma família para mim. Foram algumas noites sem dormir que passamos juntos estudando e fazendo trabalhos. Vou sentir falta dessa época.

Sumário

Índice de Figuras

Índice de Fórmulas

Índice de Tabelas

Resumo

Neste trabalho, é feito um estudo sobre alguns algoritmos de quantização de imagens, em que quantizar é representar uma imagem que tenha M cores por outra imagem que tenha N cores, onde N < M.

Estes algoritmos são implementados e os resultados visuais e numéricos são comparados. Para isso, foram escolhidos o algoritmo de quantização por corte mediano, o algoritmo de quantização por octree e o algoritmo de quantização por aglomerados duplos.

Os algoritmos de quantização apresentados no trabalho são também implementados em linguagem C, e os resultados da execução do programa compilado são usados para a comparação.

As comparações que são feitas no trabalho são: comparação visual, comparação numérica e comparação de tempos de execução. São, também, apontados aspectos importantes sobre cada algoritmo.

Abstract

In this work, a study is made on some algorithms of image quantization, in which quantization is represent an image which has M colors for other image which has N color, where N < M.

These algorithms are implemented and the visual and numeric results are compared then. For that, there were chosen the quantization algorithm by median cut, the quantization algorithm by octree and the quantization by pairwise clustering.

The quantization algorithms presented in the work were also implemented in C laguage, and the execution results of the compiled program were used for the comparison.

The comparisons in this work are: visual comparison, numerical comparison and time execution comparison. In this work are, also, indicated importants aspects about each algorithm.

1. Introdução

Neste trabalho é dada uma introdução à imagem digital e seus elementos básicos. É explicado o problema de quantização de imagens e algumas possíveis estratégias para resolvê-lo. São estudados, implementados e analisados alguns algoritmos de quantização de imagens de forma a fazer um estudo comparativo entre os mesmos quanto a performance e qualidade dos resultados. Por fim, são apresentados aspectos importantes, apontando-se as vantagens e desvantagens dos mesmos.

Para obter uma comparação expressiva, é importante apresentar alguns algoritmos mais populares para quantização de imagens e de implementação bastante simples e alguns menos populares mas de resultados mais expressivos.

Apenas alguns dos trabalhos na área de quantização de imagens que foram estudados são apresentados no trabalho. Dentre eles, foram escolhidos os algoritmos de quantização por corte mediano, quantização por octree e quantização por aglomerados duplos.

1.1 Objetivo

Este trabalho objetiva, através da análise das técnicas de quantização de imagens escolhidas, fazer uma comparação das mesmas, apresentar os resultados, os pontos positivos e negativos referentes a cada uma delas e fornecer uma visão geral sobre a quantização de imagens.

1.2 Justificativa e Problematização

Ao apresentar uma imagem em um dispositivo de saída, um monitor de vídeo por exemplo, procura-se fazê-lo da melhor forma possível. Para isto, utiliza-

se a maior quantidade possível de cores para a representação da imagem. Uma vez que o olho humano pode distinguir até 10 milhões de cores diferentes, é suficiente representar uma imagem com 16,77 milhões de cores através do cubo RGB, que contém 256 níveis de cores em cada componente, vermelho, verde e azul (GERVAUTZ & PURGATHOFER, 1990, p.287).

Em casos onde é necessário diminuir o tamanho da memória necessária para armazenar a imagem com o objetivo de transmiti-la através de um canal de comunicação, ou quando precisamos exibir uma imagem em um dispositivo gráfico que possui um [1]gamute de cor menor do que a imagem necessita, é necessário um processo de seleção das melhores cores para a representação dessa imagem. A este processo de seleção dá-se o nome de Quantização de Imagens (SOBREIRO, 1998, p. 8).

De forma simples e objetiva, pode-se dizer que quantizar uma imagem é representar uma imagem que tenha M cores por outra imagem que tenha N cores, onde N < M. Quando se faz este mapeamento para um conjunto de cores menor, deve-se selecionar as cores da melhor forma possível para minimizar a distorção perceptível. Essa tarefa não é fácil e nem muito bem definida devido a fatores tais como erros de quantização (diferenças numéricas entre a imagem original e a imagem quantizada), espaço de cor (o sistema de coordenadas nos quais as cores e os erros de quantização são medidos), ambiente de visão, conteúdo da imagem, considerações estéticas e a experiência de quem está observando as imagens (SOBREIRO, 1998, p. 1).

1.3 Revisão Bibliográfica

A maior parte da bibliografia sobre quantização de imagens está disposta em artigos, havendo também algumas poucas abordagens em livros de Computação gráfica. No Brasil um trabalho expressivo sobre quantização de imagens está sendo realizado no Laboratório VISGRAF no IMPA.

[1] Gamute: conjunto de cores, do espaço de cor, presentes na imagem.

Em (GOMES & VELHO, 1994) é descrito que a quantização se refere ao processo de discretização de cor. Para codificar a imagem no computador, deve-se trabalhar com modelos de imagem onde a função imagem toma valores em um subconjunto discreto do espaço de cor, representada em forma de ponto flutuante. Apesar de que a representação por ponto flutuante é uma discretização do conjunto dos números reais, podemos considerar como "*continuum*" um espaço cujas coordenadas são especificadas em ponto flutuante. Isto é válido, pois o erro introduzido se traduz em problemas perceptuais ou comparativos apenas quando utilizamos um número muito reduzido de bits na representação das cores de uma imagem.

Um efetivo e eficiente algoritmo para Agrupamento de dados Multidimensional é demonstrado por (WAN, 1988). Este algoritmo é designado para minimizar a proporção soma-quadrada-do-erro (sum-of-square-error). Este algoritmo é aplicado com sucesso em algoritmos de quantização de imagens.

Em (GERVAUTZ, 1990) é demonstrada uma técnica de quantização de imagens através do método de quantização por "*octree*", árvore de oito filhos. O método produz imagens com boa qualidade e de simples implementação.

É proposto por (GOMES, VELHO & SOBREIRO, 1997) um novo método de quantização de imagem colorida por aglomerados duplos. Este algoritmo consiste em um processo de relaxação que calcula uma seqüência de níveis de quantização através de operações locais de agrupamento de aglomerados duplos.

(ORCHARD, 1991) mostra algumas vantagens do projeto hierárquico, em que consegue uma paleta de cores aproximada de uma solução ótima.

Em (BALASUBRAMANIAN, 1994) é proposta uma técnica eficiente para vetores de quantização chamada *"Sequential Scalar Quantization"* (SSQ).

Em (BALASUBRAMANIAN, 1995) é feita uma análise teórica de SSQ e os resultados obtidos permitiram comparar estes métodos com outras estratégias de quantização. A SSQ não só foi aplicada com sucesso para resolver problemas de quantização de imagens, como também mostrou uma potencial utilidade para

qualquer aplicação de vetor de quantização em que o custo computacional é considerado importante e pode ser tolerada uma moderada baixa de performance.

1.4 Procedimentos Metodológicos

Este trabalho foi desenvolvido conforme os seguintes procedimentos:

Escolha dos algoritmos de quantização de imagens

Foram escolhidos 3 algoritmos. Esta escolha foi feita devido à diferença básica de cada um deles.

O algoritmo de quantização por corte mediano é classificado como um algoritmo de subdivisão espacial, o algoritmo de quantização por octree classifica-se como uma análise de aglomerados e o algoritmo de quantização por aglomerados duplos foi escolhido devido ao fato do mesmo ser uma técnica de quantização, que está sendo desenvolvida no Brasil.

Estudo dos algoritmos escolhidos

Inicialmente, é dada uma introdução sobre o que é a imagem digital e seus fundamentos, depois, os algoritmos são estudados de maneira a fornecer um embasamento teórico suficiente para que os mesmos possam ser entendidos e implementados.

Escolha de uma linguagem de programação para a implementação;

Para a implementação dos algoritmos foi escolhida a linguagem de programação C, devido a sua maleabilidade e facilidades em trabalhar com cálculos.

O programa foi desenvolvido em um computador pentium MMX 200, com 16 megabytes de memória RAM. Foi utilizado o compilador e ambiente de programação Visual C++ 5.0 da Microsoft e o Sistema Operacional Microsoft Windows95.

Mesmo sendo propício o ambiente para a programação de uma interface gráfica de janelas, para o programa foi utilizada uma interface em modo texto com recursos de 32 bits e acesso à memória protegida para tornar mais fácil a [2]alocação de grandes volumes de dados na memória.

Implementação dos algoritmos em questão

Os algoritmos são implementados na linguagem C, utilizando os recursos que cada algoritmo necessita.

É importante observar que os algoritmos implementados não foram baseados em outras implementações. Estes foram implementados, pelo autor do trabalho, em cima do embasamento teórico de cada um dos algoritmos apresentados no trabalho.

Análise dos resultados obtidos.

Após finalizada a implementação, é possível pegar os resultados da execução de cada algoritmo, extraindo o erro de quantização gerado por pixel, e uma imagem resultante quantizada a partir de uma imagem de teste quantizada inicialmente para 24 bits. Estes resultados são analisados e comparados entre os algoritmos apresentados no trabalho.

[2] Alocação: reserva de espaço em memória para armazenamento de um valor.

É importante salientar que o trabalho possui duas etapas paralelas, sendo estas o trabalho escrito e a implementação dos algoritmos. A implementação justifica-se devido à necessidade de comparação dos algoritmos.

2. Imagem Digital

Na quantização de imagens, trabalha-se diretamente com a imagem digital, de forma que ela está presente em todas as áreas de computação gráfica, seja como produto final na visualização, ou como forma de interação entre o usuário e a máquina em modelagem. Desta forma, torna-se indispensável uma compreensão do significado de imagem nesses contextos.

2.1 Paradigma de Abstração

Para representar uma imagem em um computador será utilizado um modelo matemático. Para isso, será criada uma hierarquia de abstrações e, para cada nível de abstração, serão aplicados os modelos matemáticos mais adequados. Para esse fim será utilizado o modelo matemático dos quatro universos (GOMES & VELHO, 1994, p. 7-8).

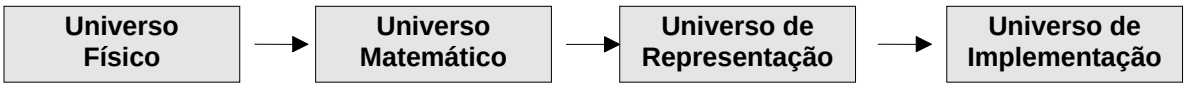


Figura 2-1: Os quatro universos de abstração.

2.1.1 O Universo Físico

A imagem é o resultado de estímulos luminosos dos objetos do mundo real capturado por algum dispositivo como uma câmera de vídeo, a retina do olho, uma máquina fotográfica, sendo que estes dispositivos decodificam os estímulos luminosos recebidos (GOMES & VELHO, 1994, p.131). Enfim, a imagem pode ser considerada tudo o que podemos ver no mundo real e que chega ao nosso cérebro como tal (SOBREIRO, 1998, p.5).

2.1.2 O Universo Matemático

No universo matemático, as imagens são descritas de forma a possibilitar a definição dos modelos abstratos de imagem. Para se definir um modelo matemático para uma cena real, a imagem é descrita na função de uma superfície bidimensional e toma-se valores em um espaço de cor.

Sendo assim, uma *imagem contínua* é uma aplicação $\mathrm{i}: U \to C$, onde $U \subset \mathbf{R}^3$ é uma superfície e C é um espaço vetorial. A função i é a *função imagem*, isto é, o *gamute de cores* da imagem, e U é um subconjunto de C, chamado de *suporte da imagem* (GOMES & VELHO, 1994, p133).

2.1.3 O Universo de Representação

Para codificar a imagem no computador deve-se também trabalhar com modelos de imagem onde a função imagem i toma valores em um subconjunto discreto do espaço de cor C. Esse processo de discretização do espaço de cor de uma imagem é chamado de *quantização* (GOMES & VELHO, 1994, p.134).

O caso mais utilizado de discretização espacial de uma imagem consiste em tomar o domínio como sendo um retângulo.

$$U = [a,b] \times [c,d] = \left\{ (x,y) \in \mathbf{R}^2 ; a \le x \le b \text{ e } c \le y \le d \right\},$$

e discretiza-se esse retângulo usando os pontos de um reticulado bidimensional $\Delta = (\Delta_x, \Delta_y)$,

$$\Delta = \left\{ (x_j, y_k) \in U ; x_j = j \cdot \Delta x, \quad y_k = k \cdot \Delta y \quad j,k \in \mathbf{Z}, \quad \Delta x, \Delta y \in \mathbf{R} \right\}$$

conforme mostra a figura 2.2. Cada pixel (x_j, y_k) da imagem pode então ser representado por coordenadas inteiras (j, k). Portanto, a imagem pode ser representada de forma conveniente no *formato matricial.* Nessa representação ela está associada a uma matriz A de ordem $m \times n$, $A = (a_{jk}) = (i(x_j, y_k))$.

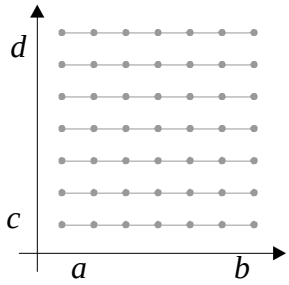


Figura 2-2: Reticulado uniforme da representação matricial da imagem.

Cada elemento a_{jk}, j=1,...,m e k=1,...,n da matriz representa o valor da função imagem *f* no ponto de coordenadas (x_j, y_k) do reticulado, sendo pois um vetor do espaço de cor representando a cor do pixel de coordenadas (j, k). Se a imagem for monocromática, A é uma matriz real, onde cada elemento é um número real que representa o valor da luminâncial do pixel.

E, por fim, chama-se *resolução de cor* ao número de bits utilizados para armazenar o vetor de cor a_{jk} de cada pixel da imagem(GOMES & VELHO, 1994, p.135-136).

2.1.4 O Universo de Implementação

Finalmente, para codificar uma imagem em uma linguagem de programação, é necessário a criação de uma estrutura de dados que seja compatível com os modelos matemáticos e representativos definidos.

Como exemplo, pode-se tomar uma estrutura de imagem que contenha 3 vetores de cores *RGB* (*Red, Green, Blue*) e duas variáveis para definir largura e altura da imagem.

```
struct color{
     float red;     // vermelho
     float green;   // verde
     float blue;    // azul
}

struct image{
     int width;     // largura
     int height;    // altura
     color *data;   // imagem
```

}

2.2 Os Elementos da Imagem Digital

Uma vez que uma imagem digital é uma imagem $i:U \to \mathbf{R}^3$ onde o suporte *U* e o espaço de cores estão discretizados, os elementos da imagem consistem, essencialmente, das coordenadas dos pixels e da informação de cada pixel, sendo que estão diretamente relacionados com a resolução espacial e resolução de cor da imagem. O conjunto finito $i(U)$ do espaço de cor discretizado é chamado de *gamute de cores* de i (SOBREIRO, 1998, p.6-7).

2.3 Histograma de Freqüência de Cor

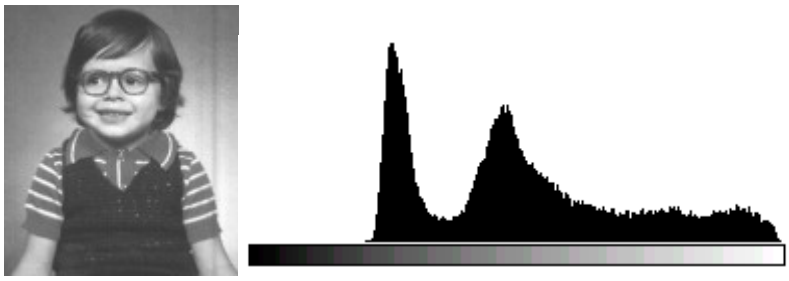


Figura 2-3: Histograma de uma imagem em tons de cinza.

As informações da distribuição de probabilidade de cada cor em determinada imagem, é de grande importância para muitas aplicações. Neste *histograma de cor,* é dada uma aproximação da quantidade de cada cor que aparece na imagem. Na figura 2.3, tem-se uma imagem em tons de cinza (do preto até o branco) e pode-se notar, através do *histograma de cor*, a ausência de cores próximas ao preto, uma alta quantidade de cores cinza-escuro e uma maior concentração de cinza-médio. Também tem-se uma certa quantidade de cores próximo ao branco e cinza-claro.

Quando se tem imagens coloridas, os *histogramas de cores* de cada cor presente na imagem podem ser analisados separadamente ou cria-se um *histograma tridimensional* (SOBREIRO, 1998, p.7).

2.3.1 Construção do Histograma de Freqüência

Para a construção do histograma de freqüência é utilizada uma estrutura de dados contendo o valor da cor, que é composta pelas três componentes de cores, o vermelho, o verde e o azul, com valores *float* do 0 á 255. Isto é suficiente para abranger até aproximadamente 16 milhões de cores. Os valores das cores são *float* devido aos cálculos. Ao final, os valores são convertidos para unsigned char. Também é utilizada, no histograma de freqüência, uma variável referente a freqüência de cada cor na imagem e uma variável para armazenar o erro de quantização da cor.

O histograma de freqüência é, então, construído, lendo-se todos os pixels da imagem um a um. Se a cor não existir no histograma, então ela é inserida e é setado o valor 1 para a freqüência. Se a cor já existir, então é apenas somado em um a freqüência da cor correspondente.

```
struct Hist{
     float red, green, blue;
     float freq;
     float erro;
} *histograma;
```

É importante salientar que o histograma de freqüência é criado a partir da imagem original pré-quantizada uniformemente para 15 bits.

3. Discretização de Cor

Para estudar com detalhes os diversos problemas existentes na discretização de cor, também chamada pelo nome de *quantização*, precisa-se caracterizar de forma mais precisa o conceito de quantização. Chama-se de *quantização* a uma transformação sobrejetiva $q: C \rightarrow C'$, de um sólido de cor C no qual as cores são representadas usando M bits, para um sólido de cor *C'* que utiliza uma codificação do vetor de cor com *N* bits, sendo *M>N*. Denomina-se *q* de *transformação de quantização*, ou simplesmente *quantização de N bits*. O espaço de *N* bits *C'* é chamado de *espaço de quantização* (GOMES & VELHO, 1994, p.142).

As duas razões básicas para se quantizar uma imagem são a exibição, em que o gamute de cores do dispositivo gráfico onde queremos apresentar a figura é menor que o gamute de cores da figura e a compressão de Imagens, que geralmente é necessária, quando queremos transmitir a imagem através de um canal de comunicação (SOBREIRO, 1998, p. 8).

3.1 Célula de Quantização

A quantização é um processo onde o espaço de cor é particionado, gerando células, chamadas de *células de quantização*. Todas as cores que estão contidas em uma célula são mapeadas para uma cor chamada de *nível de quantização*.

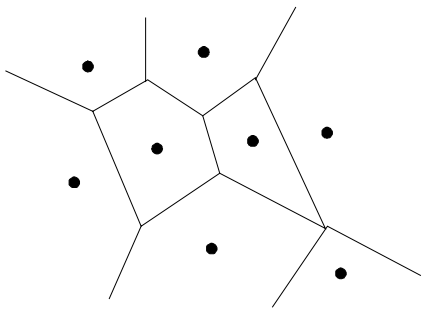


Figura 3-1: células de quantização bidimensional

3.2 Quantização e a Geometria das Células

Como foi visto, a quantização de imagens é um processo de subdivisão do espaço da imagem, criando-se subespaços sem interseção, conhecidos como células de quantização. Estas células podem ser classificadas de duas maneiras, uniforme e não-uniforme.

Existem duas maneiras de tratar o problema de quantização, quando se tem espaços de cor com mais de uma dimensão, sendo elas de forma escalar e vetorial.

3.2.1 Quantização Escalar

Na quantização escalar, quando trabalha-se com espaços de cor com n dimensões, a quantização é feita separadamente em cada componente c_i de cada vetor de cor $c = \{c_1, c_2, \ldots, c_n\}$. Neste caso, o mapa de quantização uni-dimensional *q*, e o mapa de quantização $Q: C \to C'$ é definido por

$$Q(c) = \left(q(c_1), q(c_2), \ldots, q(c_n)\right).$$

3.2.2 Quantização Vetorial

Na quantização vetorial, considera-se todos os componentes c_i ao mesmo tempo.

Quando quantiza-se um espaço de cor usando um método de quantização escalar, não levamos em consideração a correlação espacial das cores presentes no gamute da imagem que está sendo quantizada. Os métodos de quantização vetoriais eliminam este problema, pois escolhe-se as células de quantização,

levando em consideração todas as componentes de cor presentes na imagem, considerando assim a correlação espacial destas cores (SOBREIRO, 1998, p. 9).

3.2.3 Quantização Uniforme

A *quantização uniforme* é simplesmente tomar células congruentes e, em cada célula, tomar o centróide da mesma com seu nível de quantização. Esta é a mais simples maneira de se obter uma divisão do espaço. No caso de quantização escalar com *L* níveis, as células de quantização são intervalos $\left(c_{i-1}, c_i\right]$ de igual comprimento, isto é , $c_i - c_{i-1} =$constante e em cada célula o valor de quantização é dado pela média

$$q_i = \frac{c_i + c_{i-1}}{2}, 1 \le i \le L .$$

Na Figura 3.2 temos uma quantização escalar de R^2, obtido a partir de uma quantização uniforme em cada um dos eixos coordenados, com dois tipos de geometria de células de quantização uniforme bidimensional.

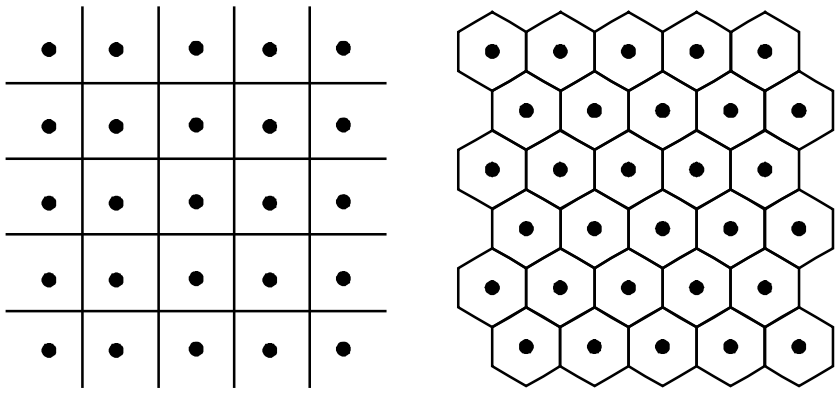


Figura 3-2: Células de quantização uniforme bidimensional e seus respectivos níveis de quantização.

A quantização uniforme é um método ótimo quando temos uma distribuição uniforme das cores pelo espaço, mas apesar de ser muito fácil de ser obtida, nem sempre ela é a mais recomendada pois, na maioria dos casos, não temos uma distribuição uniforme das cores, podendo haver células de quantização que não

terão utilidade alguma, pois pode não haver cor alguma no gamute de imagem que pertença a esta célula como na figura 3.3 (SOBREIRO, 1998, p.12).

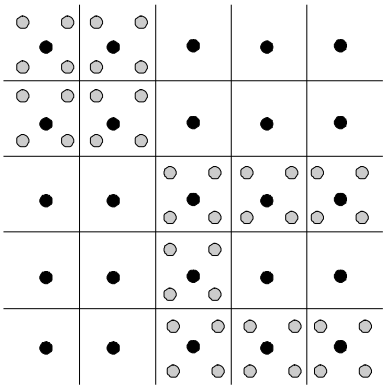


Figura 3-3: Células de quantização uniformes com distribuição não uniforme de cores pelo espaço.

3.2.3.1 Quantização Uniforme para 15 bits

Antes de aplicar os algoritmos de quantização, usualmente as imagens são quantizadas para 15 bits. Isto se dá devido ao fato de que diminui expressivamente as cores que o histograma irá conter, diminuindo, assim, o tempo de execução do algoritmo.

O resultado visual da imagem quantizada inicialmente para 15 bits não é muito afetado visualmente, pois o olho humano é muito ruim e não percebe muito bem essa diminuição de cores.

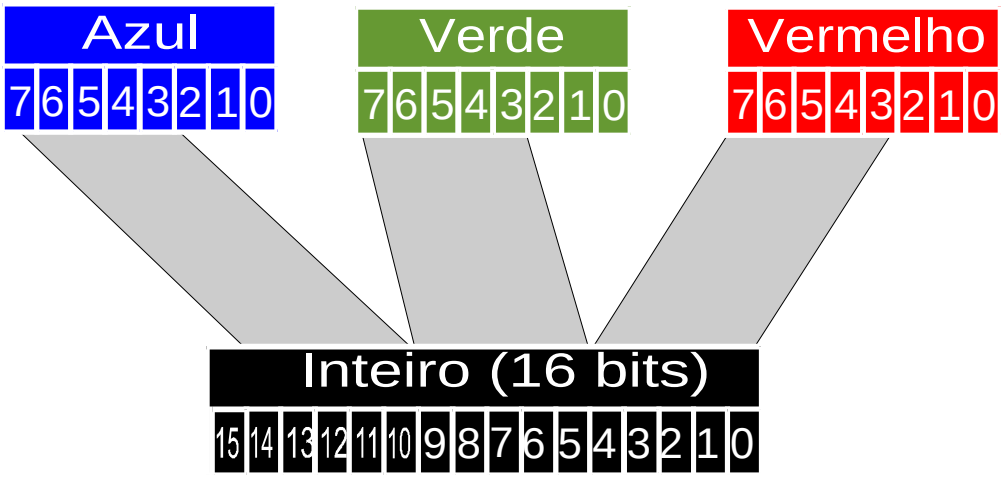


Figura 3-4: Quantização uniforme para 15 bits.

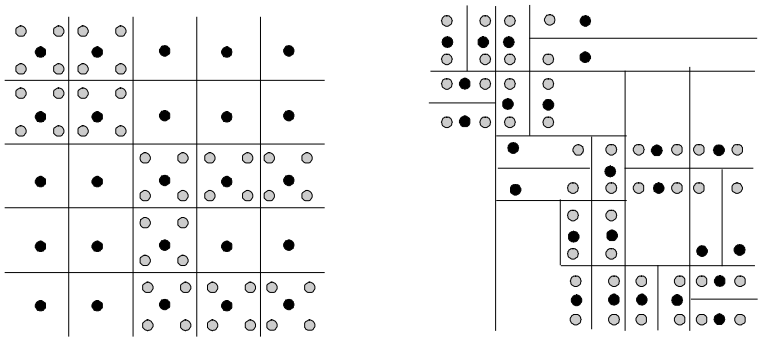
Para a quantização para 15 bits, os pixels da imagem são lidos e inseridos em um vetor de inteiros com 2 bytes, como mostra a figura 3.4. Isso é possível, desprezando os 3 bits menos significativos. Os 5 bits mais significativos são inseridos no inteiro de 2 bytes na seguinte ordem. 5 bits para o vermelho, 5 bits para o verde e 5 para o azul.

A construção do histograma de freqüência torna-se muito rápida, com a leitura esse vetor. Para ter-se cada cor, aplica-se a operação inversa.

3.2.4 Quantização Não-Uniforme

Numa distribuição de cores de uma imagem não uniforme, teremos regiões com maior concentração de cores do que em outras. Sendo assim, estas regiões devem ser subdivididas em um maior número de células com intuito de diminuirmos o erro de quantização (SOBREIRO, 1998, p.12).

Quando temos uma quantização não uniforme, também chamada *adaptativa*, tem-se a escolha da geometria das células, considerando as informações contidas no histograma de freqüência de cor das imagens.



(A) (B)

Figura 3-5: Distribuição de cores de uma imagem não uniforme com células não-adaptativas (A) e células adaptativas (B).

Toma-se como exemplo uma distribuição de cores de uma imagem não uniforme. Na figura 3.5 (A), pode-se observar algumas células de quantização em que não temos cores presentes no gamute da imagem e, na figura 3.5 (B), pode-

se notar que as células são adaptadas para que nenhuma célula de quantização fique vazia. Essa adaptação é para diminuir o máximo possível o erro de quantização.

3.2.5 Quantização e Mapa de Cor

É muito comum usarmos um *mapa de cor* para obtermos as cores dos pixels de uma imagem. Mais precisamente, suponha que temos uma imagem $f:U \subset \mathbf{R}^2 \to C$, tomamos valores em algum espaço de cor *C*. Definimos um mapa de cor $\varphi:[0,1] \subset \mathbf{R} \to C$, e os valores de cores da imagem são tomados como um conjunto dos valores do mapa de cor $\varphi([0,1]) \subset C$ conforme na figura 3.5.

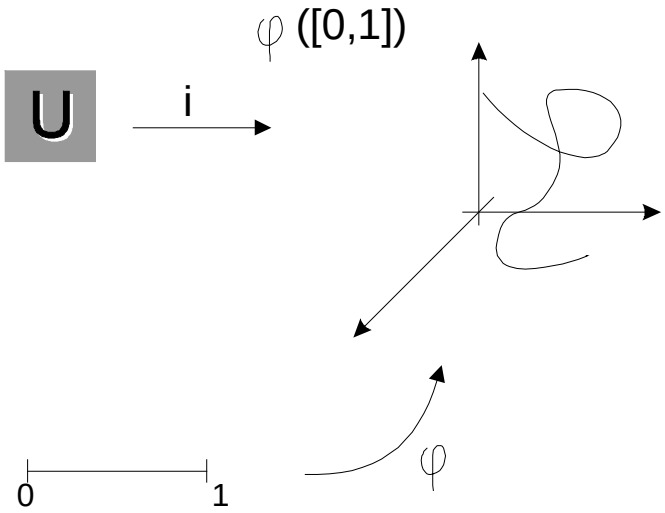


Figura 3-6: Mapa de cor de uma imagem.

Discretizamos o intervalo unitário [0,1] em *n* subintervalos definidos por alguma partição $0 = t_1 < t_2 < \ldots < t_n = 1$. Escolhendo um ponto $x_j \in [t_j, t_{j+1}]$, para $j = 1, \ldots, n-1$, obtemos uma quantização do conjunto $\varphi([0,1])$ em *n* níveis $\varphi(t_1), \varphi(t_2), \ldots, \varphi(t_n)$. Esta discretização do mapa de cor é chamada *palete*. A quantização do mapa de cor implica na quantização da imagem.

Quando nos referimos à quantização uniforme de uma imagem colorida, isto significa uma quantização uniforme de sua palete, obtida como acima, através

da subdivisão do intervalo [0,1] em *n* subintervalos uniformes (SOBREIRO, 1998, p.13).

3.3 Erro de Quantização

A determinação ótima das células de quantização e dos níveis de quantização em cada célula depende do critério utilizado para medir o erro de quantização, bem como da distribuição de cor na imagem. Esse problema é examinado com mais detalhe.

Se *q* é a transformação de quantização e *c* uma cor a ser quantizada, então

$$c = q(c) + e_q ,$$

onde e_q é o erro introduzido no processo de quantização, também chamado de *ruído de quantização.* O quadrado e_q^2 do ruído de quantização pode ser considerado como uma medida *d(c, q(c))* entre a cor *c* e seu valor quantizado *q(c)*. De modo mais preciso, toma-se uma métrica *d* no espaço de cor *C* e considera-se o espaço de quantização *C'* como um subconjunto de *C*. A *medida de distorção* da quantização *q(c)* da cor *c* é dada então pelo erro médio quadrático

$$E\left(c, q(c)\right) = \int_{-\infty}^{+\infty} p(c)\, d\left(c, q(c)\right) dc ,$$

Equação 3-1: Erro médio quadrático.

onde *p* é a função de distribuição de probabilidade de cor em *C.* O uso dessa equação para medir a distorção em uma imagem quantizada é quase intuitiva: ela pondera o erro de quantização, levando em conta a probabilidade de ocorrência de cada cor no espaço sendo quantizado (GOMES & VELHO, 1994, p. 147-148).

3.4 Quantização como um problema de Otimização

Pode-se dizer que uma quantização é ótima quando o erro gerado pelo processo é mínimo. Este erro pode ser medido conforme a equação 3.1.

Considerando que deseja-se uma quantização de *N* níveis, tem-se uma partição do espaço de cor em *N* células $K_1, K_2, \ldots K_N$. Indicando por q_k o nível de quantização da célula K_k, desta forma, a equação (3.1) pode ser escrita da seguinte maneira

$$E\left(c,q(c)\right)=\sum_{1\le j\le N}\int_{K_j}p(c)d\left(c,q_j\right)dc\,.$$

Equação 3-2: Erro médio quadrático com N níveis de quantização.

Ou ainda, levando em consideração que esta imagem possui um conjunto finito de cores em cada célula,

$$E\left(c,q(c)\right)=\sum_{1\le j\le N}\sum_{c\in k_j}p(c)d\left(c,q(c)\right).$$

Equação 3-3: Erro médio quadrático com K elementos do espaço de cor.

Devido a grande variedade de partições com *K* elementos do espaço de cor, temos então um problema difícil do ponto de vista computacional. Desta forma, tem-se um *problema de decisão*, ou seja, a cada instância do problema quer-se saber se uma dada configuração de *N* partição do espaço de cor minimiza o erro de quantização dado pela equação 3.3. Este problema não possui uma solução em tempo polinomial. Em geral, os métodos de otimização utilizados para resolver o problema de quantização não resolvem de maneira ótima o problema, podendo se enquadrar em:

- resolvem apenas uma restrição do problema;
- utilizam algum tipo de heurística;
- encontram apenas uma solução ótima aproximada.

Como o espaço de cor em questão já é discretizado para 24-bits, tem-se então um problema de aglomerados como problema de otimização na quantização, ou seja, procura-se aglomerados de cor que minimizam o erro de quantização.

3.4.1 Quantização e Análise de Aglomerados

A análise de aglomerados é usada na busca de alguma estrutura em um conjunto discreto de dados.

O problema de aglomerados de cores consiste em encontrar conjuntos de cores similares, isto é, o quão próximas estas cores estão no espaço de cor, como mostra a figura 3.7. Desta forma, pode-se escrever o problema como sendo um problema de otimização em análise de aglomerado: a solução ótima deve minimizar o erro de quantização (equação 3.1) em todas as partições de *N*-elementos do espaço de cor.

No caso de quantização de imagens, deve-se identificar *N* aglomerados de cores da imagem original que sejam similares de acordo com as medidas de quantização de forma a construir as células de quantização. Substitui-se então, as cores dos aglomerados pelo seu respectivo nível de quantização introduzindo uma distorção mínima.

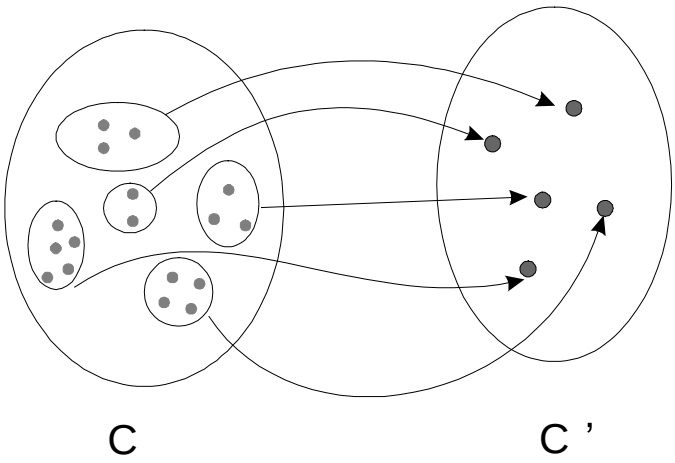


Figura 3-7: Aglomerados de cores sendo substituídos por sua cor representante.

3.5 Método Geral de Quantização

O processo de quantização de um espaço de cor consiste de duas etapas:

- Determinar as células de quantização;
- Determinar o nível de quantização em cada célula.

É então definida a função de quantização *q* na qual associa o nível de quantização às cores correspondentes, sendo que a função *q* é constante em cada célula de quantização. Tendo a transformação de quantização *q* do espaço de cor, pega-se cada cor *c* do pixel na imagem original, identifica-se a célula de quantização a que ela pertence, e substitui a cor *c* pelo valor de quantização, *q(c)*, da célula (GOMES & VELHO, 1994, p.147).

Os diferentes métodos de quantização existentes refletem a descrição acima, e funcionam de três modos distintos:

- Determinam-se inicialmente as células de quantização, e em seguida calcula-se o nível de quantização de cada célula;
- Determinam-se inicialmente os níveis de quantização, e em seguida determinam-se as cores que devem ser quantizadas para cada nível;
- Determina-se de forma interdependente e simultânea as células e os níveis de quantização.

3.6 Imagem de Teste

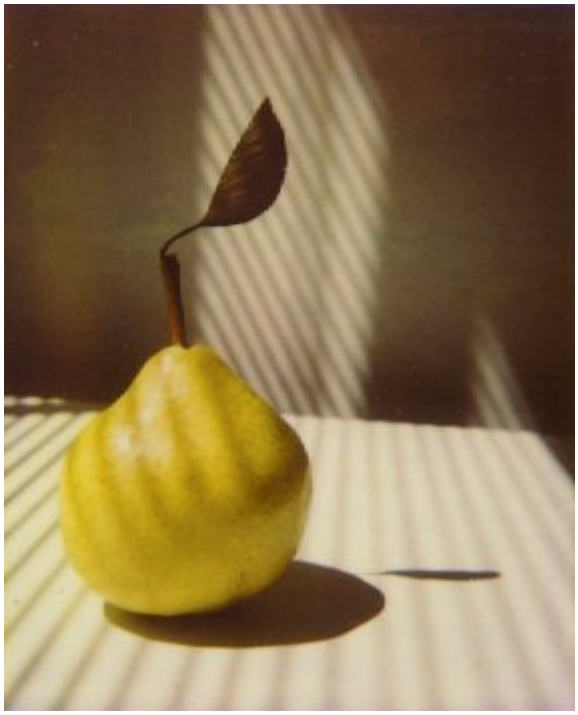


Figura 3-8: Imagem digital colorida quantizada com 24 bits: pêra.

Para comparar os métodos de quantização apresentados, é utilizada a fotografia da pêra, quantizada inicialmente para 24 bits, não tendo assim níveis de quantização perceptíveis.

4. Quantização por Corte Mediano

A quantização por corte mediano é classificada como uma quantização por subdivisão recursiva e classifica-se como um método de quantização adaptativa de cor. A primeira etapa é a obtenção de uma estimativa das propriedades estatísticas relevantes da imagem e a segunda etapa é a aplicação do método geral de quantização, particionando-se o espaço de cor com base nos dados obtidos na fase anterior. Este método foi desenvolvido por P. S. Heckbert em 1982 e é apresentado por Jonas Gomes e Luiz Velho em (GOMES & VELHO, 1995, p153-157).

Assim, inicialmente, é construído um histograma de freqüência da imagem, que fornece uma aproximação da função de distribuição de probabilidade das cores na imagem. Para reduzir o tamanho da memória ocupada pelo histograma, é comum fazer-se previamente uma quantização uniforme da imagem. Em geral utilizam-se 5 bits para cada uma das três componentes, vermelho, verde e azul (GOMES & VELHO, 1994, p.150).

4.1 Principio do Método de Quantização por Corte Mediano

Inicialmente, são determinadas as células de quantização, através de um processo de subdivisão recursiva do espaço de cor, para em seguida calcular a função de quantização em cada célula.

Tem-se inicialmente uma região do espaço de cor com o gamute de cores da imagem. Em cada iteração, essa região é subdividida em duas novas sub-regiões do espaço na qual baseia-se em características estatísticas sobre a distribuição de cores nos diversos pixels da imagem. O processo de recursão continua até que não existam mais cores da imagem original contidas em algum conjunto da subdivisão, ou até que se obtenha o número desejado de células de quantização, que corresponde ao número dos níveis.

Estando as células de quantização determinadas, é realizada a função de quantização, que é obtida através da escolha de um nível de quantização em cada célula.

4.2 Mediana de um Conjunto

Tendo um conjunto finito e ordenado de pontos do espaço

$$C = \left\{ c_1 \le c_2 \le ... \le c_{n-1} \le c_n \right\},$$

a mediana *mc* desse conjunto é definida por $c_{(n+1)/2}$ se *n* é impar e pela média dos dois elementos intermediários se *n* é par. A mediana divide o conjunto de dados *C* em duas partes com iguais números de elementos.

A extensão do algoritmo de quantização por equalização de histograma descrito acima, para imagens a cores, é conhecido na literatura pelo nome de *algoritmo do corte mediano*. Este é um dos algoritmos de quantização mais populares dentro da comunidade de computação gráfica pela sua facilidade de implementação, eficiência computacional, bons resultados perceptuais conseguidos na quantização de imagens de 24 bits para 8 bits (GOMES & VELHO, 1994, p.154).

4.3 Algoritmo do Corte Mediano

Seja *K* o número de níveis de quantização desejado. Toma-se um paralelepípedo

$$V = \left\{ \left[r_0, r_1 \right] \times \left[g_0, g_1 \right] \times \left[b_0, b_1 \right] \right\},$$

Equação 4-4: Paralelepípedo de cores

de volume mínimo que contém todas as cores no gamute da imagem a ser quantizada. Em seguida toma-se a componente do espaço de cor em cuja direção

o paralelepípedo *V* possui a aresta de maior comprimento. Supondo-se que essa é a componente verde *g*. Então, ordena-se as cores no gamute da imagem pela componente *g*, e encontra-se a mediana m_g do conjunto de cores com base nessa ordenação. Divide-se assim a região *V* em duas sub-regiões

$$V_1 = \left\{(r,g,b) \in C; g \le m_g\right\}, \qquad \text{e} \qquad V_2 = \left\{(r,g,b) \in C; g \ge m_g\right\}.$$

Equação 4-5: Sub-vetores de cor

Aplica-se então o mesmo método de subdivisão a cada uma das regiões V_1 e V_2. O processo de subdivisão continua recursivamente até que as duas regiões obtidas não contenham mais cores do gamute da imagem, ou até quando o número desejado, *K*, de células de quantização já tiver sido obtido.

Após a subdivisão do espaço de cor no número de células desejado, determina-se o nível de quantização de cada célula. Para se obter o valor de quantização de um pixel da imagem, deve-se localizar a célula que contém a cor desse pixel e fazer a quantização para o nível correspondente a essa célula.

4.4 Um exemplo de Quantização por Corte Mediano

Cor	Freqüência
c1	2
c2	3
c3	2
c4	1
c5	2
c6	1
c7	1
c8	1
c9	2

Cor	Freqüência
c1	2
c9	2
c8	1
c2	3
c3	2
c4	1
c7	1
c5	2
c6	1

Tabela 4-1: Cores e freqüências; não ordenadas (esquerda) e ordenadas (direita)

Neste exemplo, é utilizado um espaço de cor bidimensional. Neste, tem-se um conjunto de 9 cores distintas cujas freqüências são dadas pela tabela a esquerda acima.

A direção mais longa do retângulo de cores é a vertical. Ordenando-se as cores pela ordenada *y*, obtém-se a tabela a direita acima.

Verifica-se que a mediana do conjunto está na posição ocupada pela cor C2, como mostra na Figura 4.1(B).

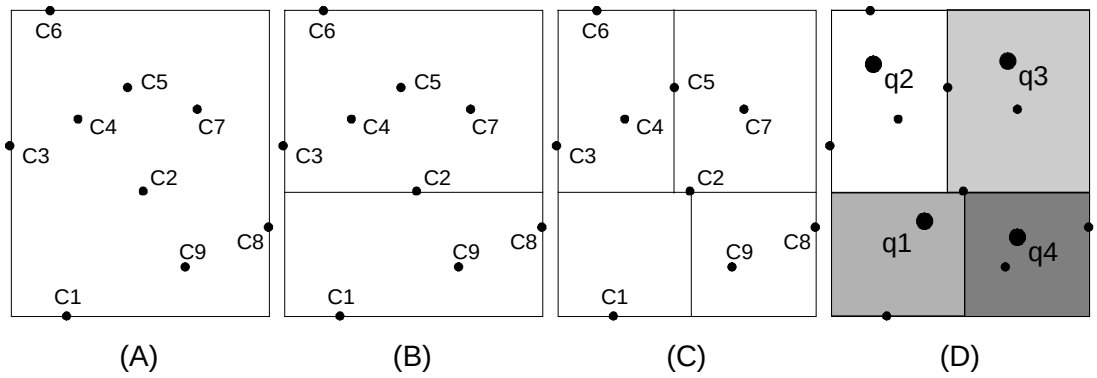


Figura 4-1: Subdivisão recursiva pelo corte mediano (A), (B) e (C); níveis de quantização (D).

A partir disso, tem-se dois retângulos de cores resultantes da subdivisão onde a aresta mais longa de cada um é a horizontal. Ordenando as cores pela componente horizontal *x*, obtém-se as duas tabelas de freqüência abaixo:

Cor	Freqüência
c1	2
c2	3
c9	2
c8	1

(A)

Cor	Freqüência
c3	2
c6	1
c4	1
c5	2
c2	3
c7	1

(B)

Tabela 4-2: Subdivisão em relação a médiana

Na Tabela 4.2 (A), a mediana é a cor *C2*, e na tabela 4.2 (B), a mediana é a cor *C5*.

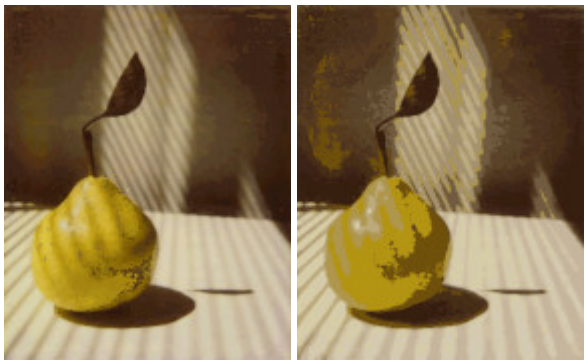
Após determinadas as quatro células de quantização, o nível de quantização em cada célula é calculado tomando a média das cores do gamute em cada célula. As cores que estão na borda da célula devem ser quantizadas

para o nível de quantização mais próximo. A função de quantização é apresentada pela tabela abaixo

c (cor)	q(c)
c1,c2	q1
c3,c4,c6	q2
c5,c7	q3
c8,c9	q4

Tabela 4-3: Células de quantização com suas cores

Podem ser usados outros critérios para subdividir o espaço de cor. Uma mudança simples do algoritmo consiste em escolher um particionamento, observando a variância entre as cores de cada sub-região e o nível de quantização associado.



(A) **(B)**

Figura 4-2: Quantização por Corte Mediano: (A) 256 cores; (B) 16 cores.

4.5 Implementação

A implementação do algoritmo do corte mediano é bastante simples. O histograma de freqüência é subdividido até que se tenha uma quantidade de subdivisões igual a quantidade de cores na qual se quer quantizar.

Para isso, foi utilizado um vetor contendo uma variável de cor (Red, Green, Blue), uma variável contendo a freqüência das cores que estão nessa célula, o ponteiro inicial e final.

Inicialmente, nesse vetor, só existe um elemento, na qual seu ponteiro inicial aponta para o primeiro elemento do histograma e seu vetor final aponta para o final do histograma.

O histograma é ordenado (nesta implementação foi utilizado o método de ordenação por seleção) pelo vetor de maior comprimento. O elemento inicial é então dividido em duas partes, usando o cálculo da mediana.

Este procedimento é repetido até que se tenha o número de elementos igual a quantidade de cores na qual se quer quantizar a imagem.

O vetor de quantização é lido seqüencialmente e é tirada a média das cores que estão entre o ponteiro inicial e o ponteiro final. As freqüências das cores que estão nessa célula são somadas.

Tendo isso, torna-se uma tarefa simples mapear as cores da imagem para sua célula de quantização correspondente.

O mapeamento é feito lendo as cores contidas no histograma de freqüência uma a uma, para cada uma delas; é lido o vetor de quantização sequencialmente até que a cor esteja entre a posição que aponta o ponteiro inicial e o ponteiro final da célula. Esta cor é então substituída pela cor correspondente na célula de quantização, que é a média das cores que ali estão contidas.

5. Quantização por Octree

O método de quantização por *octree* é apresentado como um método para preencher uma tabela de cor que produz imagens de similar qualidade dos métodos existentes. Porém, este método requer menos memória e tempo de execução. Este algoritmo é apresentado por Michael Gervaultz e Werner Purgathofer em (GERVAUTZ & PURGATHOFER, 1990, p.287).

5.1 O Princípio do Método de quantização por Octree

No método de quantização por octree, a imagem é lida seqüencialmente. As primeiras *K* cores diferentes são usadas como entrada inicial na tabela de cor. Quando outra cor é adicionada, que significa que a parte processada da imagem já tem K + 1 cores diferentes, algumas cores que estão próximas são agrupadas e substituídas pela média das cores presentes nos respectivos sub-cubos de cor. Este passo é repetido até que toda a imagem tenha sido lida.

5.2 A Octree

Para se trabalhar com este método, torna-se necessário o uso de uma estrutura de dados que permite uma rápida detecção de cores que sejam próximas em um espaço de cor. O cubo RGB na figura 5.1 pode facilmente ser administrado por uma octree.

É suficiente usar uma octree de profundidade oito para representar todas as cores possíveis. Sendo que cada nó pode ter até oito filhos em uma árvore de profundidade oito, tem-se oito elevado a oito, isto é, 16777216 cores.

Conforme caminhamos a níveis mais profundos na árvore octree, temos sub-cubos menores. Sendo assim, quanto maior a profundidade de um nó, menor

é o sub-cubo de cor representado por ele. Conseqüentemente, a profundidade de um nó é uma medida para a distância máxima de suas cores.

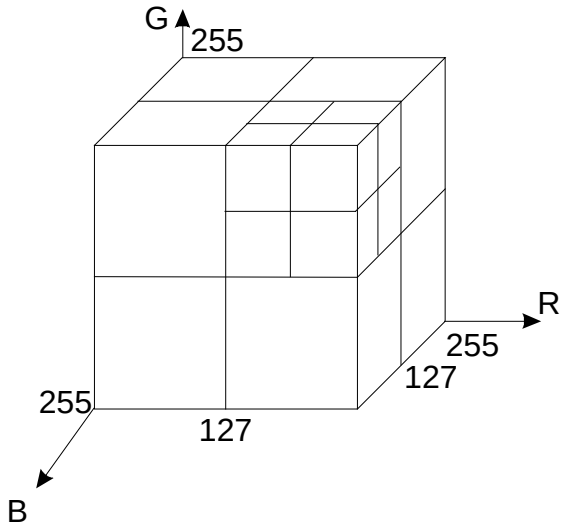


Figura 5-1: Cubo RGB para o algoritmo de octree.

5.3 O Algoritmo

O algoritmo da técnica de quantização por *octree* é feito em três partes:

- Determinação das cores representantes;
- Preenchimento da tabela de cor;
- Mapeamento das cores originas nas cores representantes.

5.3.1 Determinação das cores representantes

A *octree* é construída apenas onde é necessário, ou seja, se não tivermos ao menos uma cor contida em certo cubo, não teremos sub-cubos. De início, a *octree* está vazia. Sempre que uma cor da imagem é inserida, é gerada uma folha, na qual representa a cor.

Existem três casos possíveis quando inserimos uma cor na octree:

1º Caso: Quando o elo da árvore octree selecionado é um ponteiro vazio.

Neste caso, simplesmente cria-se um novo nó folha e armazena a nova cor neste nó.

Imagine que temos uma cor RGB(100, 100, 190) e quer-se inserir esta cor em uma octree e esta cor faz parte de um sub-cubo que ainda não está presente na octree. Cria-se um novo nó que representa o sub-cubo, onde está a cor, e insere-se a cor, incrementando o contador *NP* de cores contidas no nó (Figura 5.2).

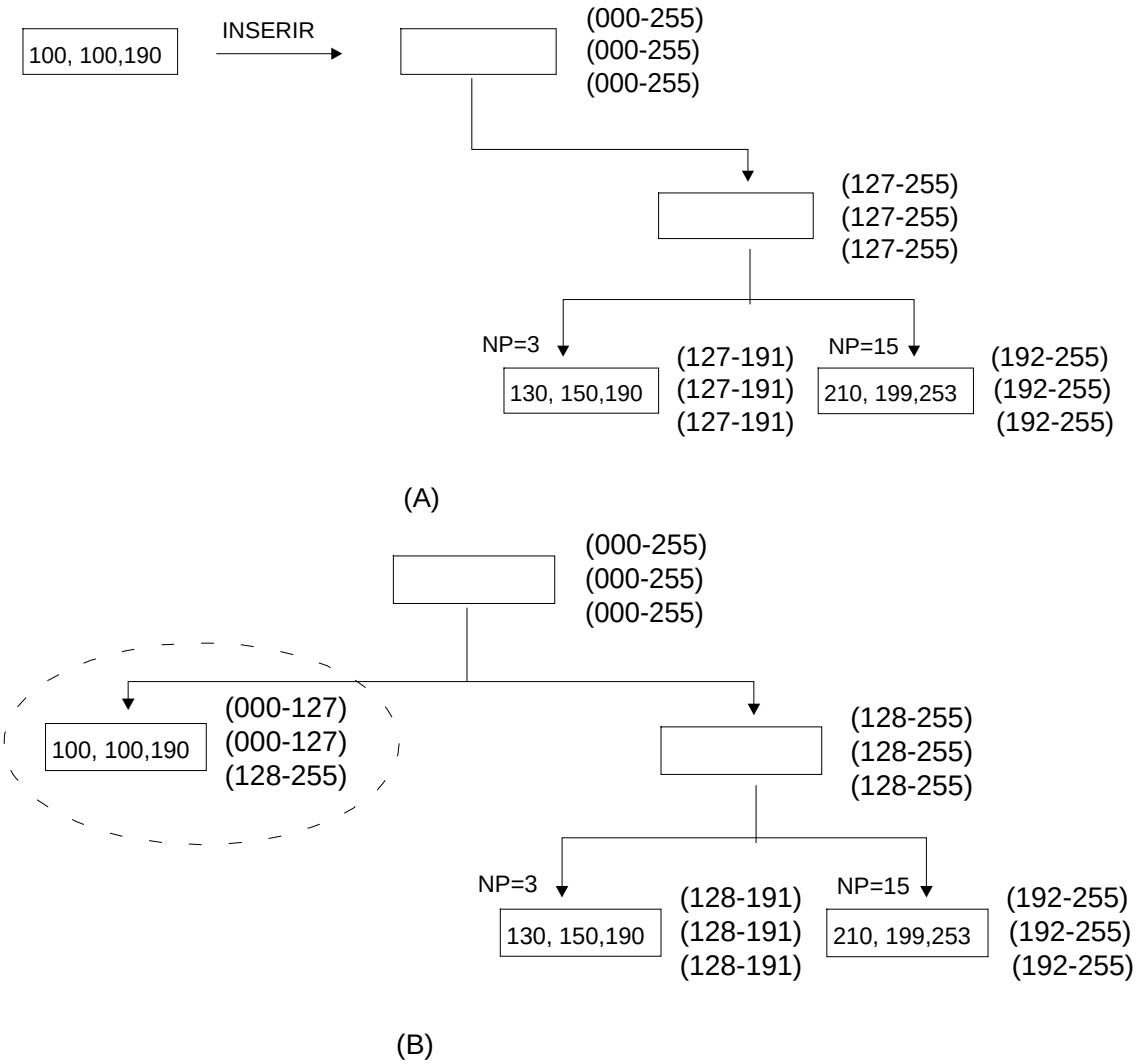


Figura 5-2: Inserção de uma nova cor em uma árvore octree, 1º caso.

2º Caso: Quando o nó folha achado é igual ao que se quer inserir.

Neste caso, o valor que quer-se inserir já existe, sendo assim, o novo valor não será inserido na árvore, mas o contador NP será incrementado de um.

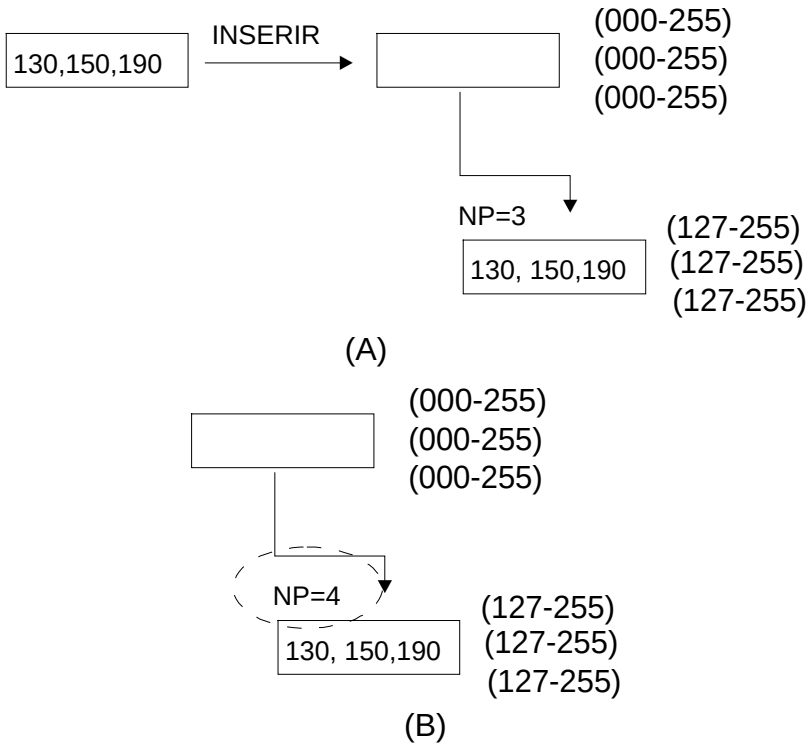
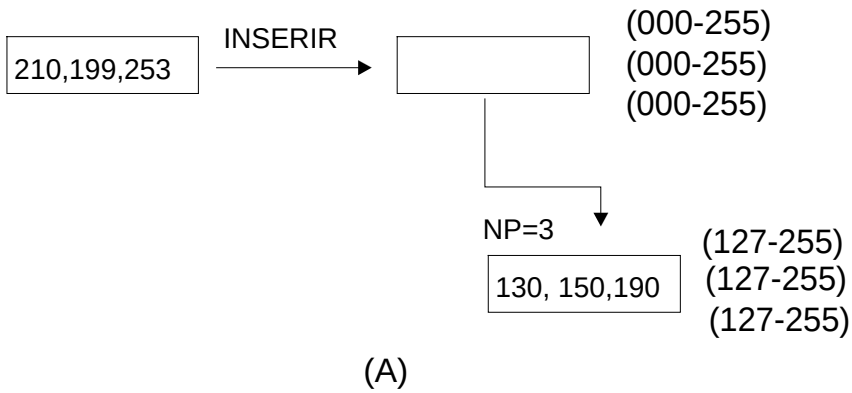


Figura 5-3: Inserção de uma nova cor em uma árvore octree, 2º caso.

3º Caso: Quando o nó folha achado é diferente da cor que se quer inserir.

Neste caso, as cores estão no mesmo sub-cubo. Então é criado um novo nó que servirá de nó intermediário, e abaixo deste, insere-se, como filhos, a cor já existente e a cor que se quer inserir.



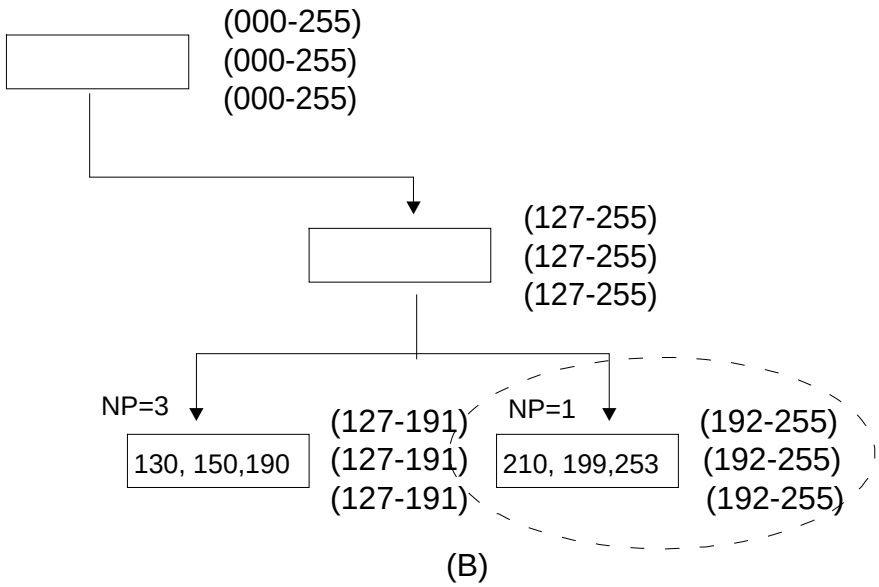


Figura 5-4: Inserção de uma nova cor em uma árvore octree, 3º caso.

Desta maneira, é criada uma *octree* incompleta, na qual muitos nós filhos estão perdidos. De fato, não é necessário preencher esta *octree* com todas as cores porque toda vez que o número K de cores é alcançado, as cores semelhantes podem ser fundidas de forma que nunca tenha-se mais que K cores. Esta ação chama-se *diminuição de octree*.

Sempre que o número de folhas excede K, a *octree* é reduzida. A redução inicia-se na parte mais baixa da octree, sempre substituindo algumas folhas pelo seu predecessor.

Ao reduzir a octree, os seguintes critérios são relevantes:

- De todos os nós redutíveis, devem ser escolhidos os que tem a maior profundidade dentro da primeira octree. Isto porque eles representam as cores que estão mais próximas.

- Se existe mais que um nó de maior profundidade, um critério adicional poderia ser usado para uma ótima seleção. Por exemplo, reduzir o nó que representa as cores em menor quantidade até o momento. Deste modo, a soma do erro será mantida pequena.

Para a construção da octree de cores, a imagem inteira terá que ser lida e todas as cores da imagem precisam ser inseridas na octree.

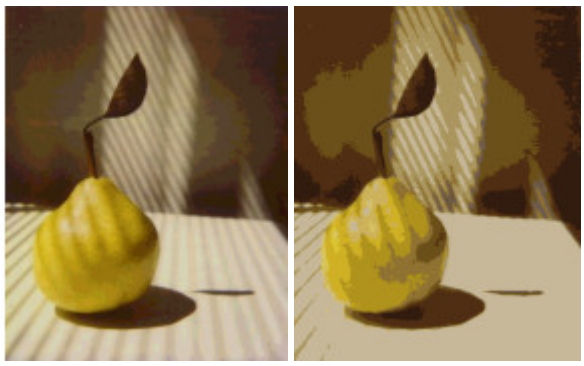
5.3.2 Preenchimento da tabela de cor

Ao final, as K folhas da árvore octree contém as cores para a tabela de cores. Elas podem ser alocadas na tabela de cor, examinando-se recursivamente a octree. Durante esta passagem recursiva por todos os nós folhas da octree, a freqüência de cada cor é também armazenada.

5.3.3 Mapeamento das cores originais nas cores representantes

O mapeamento das cores originais em suas cores representantes pode agora ser facilmente gerenciado com a octree. Ao tentar encontrar qualquer cor original na octree reduzida, sempre se chegará à um nó folha em algum nível da árvore. Este nó contém uma cor muito parecida com a que se procura, sendo esta a sua cor representante.

Se a imagem original usa menos que K cores, não irá haver redução na octree e a tabela de cor irá conter exatamente todas as cores que a imagem contém.



(A) **(B)**

Figura 5-5: Quantização por Octree: (A) 256 cores; (B) 16 cores.

O resultado visual da quantização por octree é de similar qualidade do método de corte mediano.

Na figura 5.5 é apresentado o resultado visual da quantização por octree para 256 e 16 cores.

5.4 Implementação

A implementação é bastante simples e é bem explicado sobre a seqüência lógica do algoritmo neste capítulo. O importante é apresentar a estrutura de dados na qual o algoritmo é implementado.

Para a implementação deste algoritmo é utilizada uma estrutura de dados que representa o cubo RGB.

```
/* Informacoes do cubo RGB */
typedef struct _CuboRGB{
     No     *raiz;
     u_long idx_cores;
     u_long maximo_cores;
     No     **lista;
} CuboRGB;
```

Essa estrutura contém um ponteiro para o nó raiz da árvore de oito filhos (octree), uma variável que contém a informação do número de cores na qual se quer quantizar, o número de folhas que a árvore contém (cada folha representa uma cor) e uma lista de apontadores para as folhas da árvore. Esta lista de apontadores aumenta a velocidade do algoritmo e facilita na procura das cores.

```
/* Informacoes de um no da octree */
typedef struct _No{
     struct _No *pai, *filho[8];
     float r, g, b;
     byte er, eg, eb;
     byte dr, dg, db;
     byte nivel;
     u_long n_cores;
} No;
```

Cada nó da árvore, incluindo o raiz, tem uma estrutura de dados que contém um apontador para o nó pai, um vetor contendo oito apontadores para os nós filhos, se tiver, a cor representada por três variáveis *floats* r, g e b, a área de abrangência do sub-cubo, o nível da árvore na qual se encontra o nó e a quantidade de cores que estão presentes na imagem.

6. Quantização por Aglomerados Duplos

Nesta técnica Usa-se uma estratégia de otimização local do erro de quantização, gerando níveis de quantização que estão perto do ótimo. O algoritmo gera resultados melhores que os demais e possui fácil implementação, porém, é um pouco mais lento que os demais. Este algoritmo é apresentado por Jonas Gomes, Luiz Velho e Marcos Sobreiro em (GOMES, VELHO & SOBREIRO, 1997).

6.1 Quantização de um Aglomerado de Cores

Para encontrarmos o nível de quantização ótimo associado a um aglomerado de cores, tem-se o seguinte teorema:

Teorema: Sendo $K = \{c_1, c_2, \ldots, c_m\}$ um aglomerado com *M* cores de um conjunto de cores $C \subset \mathbf{R}^n$ do gamute de uma imagem, o nível de quantização ótimo *c* do aglomerado *K* é dado por

$$c = \frac{1}{\sum_i F_i} \sum_{j=1}^{M} F_j c_j \ ,$$

Equação 6-1: Nível de quantização ótimo

onde $F_i = F(C_i)$ é a freqüência de ocorrência da cor c_i na imagem. Além disso, o erro de quantização global no aglomerado é dado por

$$E(K) = \frac{1}{\left(\sum_k F_k\right)^2} \sum_{j=1}^{M} F_j \left\| \sum_{i=1}^{M} F_i (c_i - c_j) \right\|^2 .$$

Equação 6-2: Erro de quantização

A primeira parte do teorema diz que o ótimo nível de quantização de um aglomerado é dado pelo seu centróide.

Demonstração: Considerando o quadrado de cor da métrica euclidiana, $d\left(c_i, c_j\right) = \left\| c - c_j \right\|^2$ com função distância no espaço de cor, usando a equação (3.3), e aproximando a distribuição de probabilidade de ocorrência de uma cor na imagem pelo seu histograma de freqüência, o erro de quantização no aglomerado *K* é dado por

$$E(c) = \sum_{j=1}^{M} F_j \left\| c - c_j \right\|^2 .$$

Equação 6-3: Erro de quantização com o quadrado da métrica euclidiana

Isso nos dá o erro de quantização associado ao nível de quantização ótimo *c* do aglomerado que é obtido do mínimo da função *E*.

O gradiente de *E* é dado por

$$grad(E) = \sum_{j=1}^{M} 2F_j \left(c - c_j \right).$$

Equação 6-4: Gradiente do erro.

Desta forma obtemos seu ponto crítico *c*

$$c = \frac{1}{\sum_{i=1}^{M} F_i} \sum_j F_j c_j .$$

Equação 6-5: Ponto mínimo.

Como a função *E* é convexa, *c* é na verdade seu ponto de mínimo.

Substituindo o ponto de mínimo *c* de (6.5) na equação (6.3), obtemos o erro de quantização dado na equação (6.2). Isso conclui a demonstração do Teorema.

O Corolário a seguir é um caso particular do Teorema para o caso de um aglomerado de apenas duas cores.

Corolário: Se temos um aglomerado de somente duas cores $K = \{c_i, c_j\}$, o seu nível de quantização ótimo, usando a métrica quadrática do espaço euclidiano, é dado por

$$c = \frac{F_i}{F_i + F_j} c_i + \frac{F_j}{F_i + F_j} c_j .$$

Equação 6-6: Nível de quantização ótimo de um aglomerado.

O erro de quantização associado é dado por

$$E(c_i, c_j) = \frac{F_i F_j^2 + F_j F_i^2}{(F_i + F_j)^2} \|c_i - c_j\|^2$$

Equação 6-7: Erro de quantização associado ao nível de quantização ótimo de um aglomerado

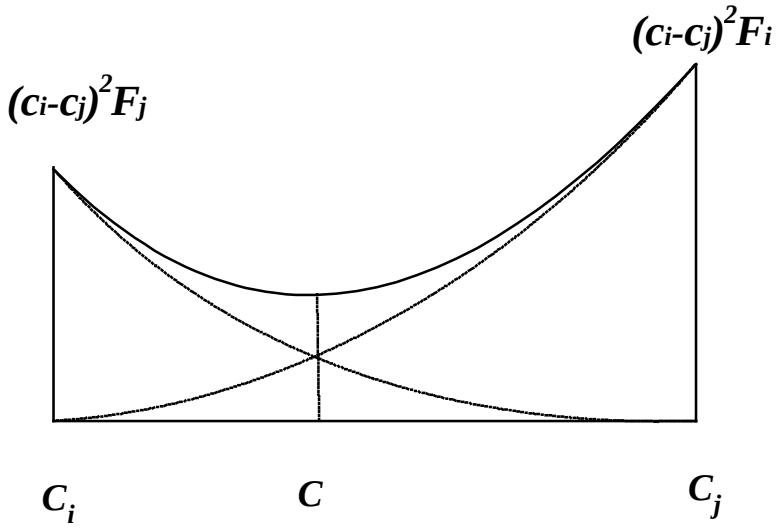


Figura 6-1: Gráfico do erro de quantização.

Uma idéia desse resultado pode ser observada na interpretação geométrica desse Corolário mostrada na Figura 6.1. Da equação 6-3 temos que o erro de quantização é dado por uma função polinomial de segundo grau

$$E_q(c) = F_i (c - c_i)^2 + F_j (c - c_j)^2 .$$

Equação 6-8: Função polinomial de segundo grau da Equação 6-3

Essa função é um arco de parábola obtido da soma dos dois arcos de parábola $F_i\left(c-c_i\right)^2$ e $F_j\left(c-c_j\right)^2$, que são mostrados como linhas pontilhadas na Figura 6.1.

Note que o nível de quantização *c* é dado pelo único ponto de mínimo da parábola. Quando $F_i = F_j$ o nível de quantização é o ponto médio do segmento $\overline{c_i c_j}$ dado por

$$c = \frac{c_i + c_j}{2}.$$

Equação 6-9: Nível de quantização

6.2 Quantização por Aglomerados Duplos

Através dos resultados do Corolário da seção anterior, tem-se um algoritmo para quantização de imagens coloridas. O método consiste de um processo de relaxação que calcula uma seqüência de níveis de quantização através de operações locais de agrupamento de aglomerados duplos.

Tendo como entrada do algoritmo o conjunto finito *C* de *M* cores contidas no gamute da imagem a ser quantizada, $C = \left(c_1, c_2, ..., c_M\right)$. Cada cor c_i tem uma freqüência de ocorrência na imagem $F_i = F\left(c_i\right)$. Associa-se um erro de quantização acumulado $E_A\left(c_i\right)$ à cada cor c_i. Inicialmente este erro é zero.

O método de quantização de imagens é composto dos seguintes passos:

1. Calcule o histograma de freqüência da imagem.

2. Usando a equação (6.7), calcule o erro de quantização $E\left(c_i, c_j\right)$ resultante da combinação de cada par de cor $\left\{c_i, c_j\right\}$ do conjunto de cores de entrada, como mostra a tabela 6.1.

	C1	**C2**	**C3**	**Cn**
C1				
C2	E(C1, C2)			
C3	E(C1, C3)	E(C2, C3)		

Tabela 6-1: Tabela dos erros de quantização resultante da combinação de cada par de cores

3. Encontre o aglomerado duplo $K_0 = \{c_i, c_j\}$ do conjunto de cores de entrada que minimiza o erro de quantização $E(c_i, c_j)$ calculado no passo anterior.

4. Usando a equação (6.6) calcule o nível de quantização c_{k_0} do aglomerado duplo $K_0 = \{c_i, c_j\}$ encontrado no passo anterior.

5. Substitua o aglomerado duplo $K_0 = \{c_i, c_j\}$ pelo seu nível de quantização c_{k_0}. Isso resultará em um conjunto de cores quantizadas *C'* com *M - 1* cores. A freqüência de ocorrência $F(c_{k_0})$ da cor c_{k_0} é dada pela soma das freqüências das cores c_i e c_j: $F(c_{k_0}) = F_i + F_j$. O erro de quantização acumulado $E_A(c_{k_0})$ da cor c_{k_0} é dado por .

$$E_A(c_{k_0}) = E(c_i, c_j) + E_A(c_i) + E_A(c_j)$$

Equação 6-10: Erro de quantização acumulado.

6. Calcule o erro de quantização para todos os aglomerados de cores $\{c_i, c_j\}$, do conjunto de cores quantizadas *C'.*

7. Use o conjunto de cores quantizadas *C*' como entrada para o passo 3 do algoritmo. Repita os passos 3 a 7 até que o número desejado de níveis de quantização seja obtido.

Este processo de relaxação acima nos fornece os níveis de quantização. A partir desses níveis, calculamos as células de quantização de forma que

$$q(c) = c_i' \Leftrightarrow d(c, c_i') \leq d(c, c_j')$$

Equação 6-11:Função das células de qunatização

para todo $1 \leq j \leq M$ com $j \neq i$.

6.2.1 Melhoria no Algoritmo

Ao agrupar as cores de uma imagem por pares na quantização, perde-se a correlação espacial das cores no gamute da imagem. Pode-se usar como solução, para minimizar esta perda, recalcular o nível de quantização em cada célula utilizando a equação (6.1) após o cálculo das células de quantização.

6.3 Implementação

Nesta parte tem-se a descrição da implementação do algoritmo de quantização por aglomerados duplos.

6.3.1 Estruturas de Dados

O algoritmo usa duas estruturas de dados principais indexadas pelo número de cores. A estrutura "**Real E[][]**" para juntar a matriz de erro $E_{i,j}$, e o vetor "**C[]**":

```
struct c[]{
   Color val;
   Real freq;
   Real err;
}
```

onde **val** é o valor da cor, **freq** é a freqüência da cor na imagem, e **err** é o erro de quantização acumulado da cor ***c***.

A junção do erro de quantização (6.7) associado à dois agrupamentos de cores $\{c_i, c_j\}$, é armazenada na entrada da matriz $E_{i,j}$. O *E* da fórmula (6.7) é uma matriz simétrica. Desde que não se executem operações da matriz com *E*,

precisa-se armazenar apenas os elementos $E_{i,j}$ para *i* < *j*, que constituem uma matriz triangular menor.

6.3.2 Pseudo-Código e Operações

A codificação dos algoritmos é simplesmente:

```
imagem Quantizar (imagem, m_niveis){
     c = calcule_histograma(image)
     E = calcule_matrix_erro(c)
     enquanto (numero_de_cores_em(E) > m_niveis){
          (ci, cj) = selecionar_par_de_cores (E)
          ck = junte_par_de_cores(ci, cj)
          trocar_cores(ci, cj) por ck
     }
     aplique_quantizacao(imagem, c)
}
```

A função **calcule_histograma** calcula a freqüência de cada cor na imagem de entrada e seleciona o conjunto de cores com freqüência **freq > 0**. Para reduzir os cálculos computacionais envolvidos, a imagem é uniformemente quantizada para 15-bits antes de calcular o histograma.

A função **calcule_matriz_erro** calcula $E_{i,j}$ na matriz usando a equação (6.7).

A função **selecionar_par_de_cores** simplesmente seleciona o par de cores $\left(c_i, c_j\right)$ com o mínimo erro de quantização $E_{i,j}$ na matriz ***E***.

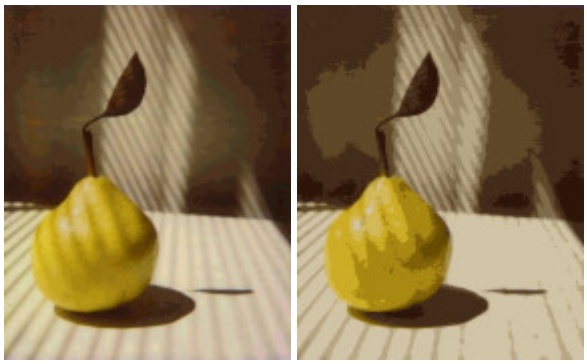
A função **junte_par_de_cores** calcula a nova cor quantizada $\boldsymbol{c_k}$ para substituir o aglomerado $\left\{c_i, c_j\right\}$, usando a equação (6.6).

A função **trocar_cores** substitui os dois aglomerados de cores $\left\{c_i, c_j\right\}$ pela cor quantizada $\boldsymbol{c_k}$. Isto é feito no terceiro estágio: Primeiro $F_k = F_i + F_j$ e $E_K = E_{i,j} + E_i + E_j$ são calculados; Em segundo lugar as entradas para as cores

c_i e c_j são excluídas do vetor ***c*** e da matriz ***E***; E terceiro, a nova entrada para a cor c_k é adicionada para ambos **c** e **E**.

A função **trocar_cores** é a parte central do algoritmo, e esta operação pode ser iniciada em termos de simples operação com a matriz **E**: O fato de excluir a cor c_j equivale a excluir a *j-ésima* linha e coluna de **E**. Se a matriz **E** é de ordem ***n*** (***n*** cores são processadas inicialmente), adicionando uma nova cor c_k equivale a adicionar o *(n + 1)-ésima* linha até o fim da matriz **E**. Essas linhas armazenam os erros das cores pares formados pela nova cor c_k com todas as outras cores $c_l, l = 1..n$.

Na figura 6.2 é apresentado um resultado visual da quantização da imagem de teste quantizada para 256 e 16 cores



(A) **(B)**

Figura 6-2: Quantização por Aglomerados Duplos: (A) 256 cores; (B) 16 cores.

7. Conclusão

Nesta parte do trabalho tem-se a comparação dos algoritmos de quantização estudados, o algoritmo de quantização por corte mediano, quantização por octree e quantização por aglomerados duplos. Também são apontados aspectos importantes de cada algoritmo inserido no contexto da técnica e forma de implementação.

7.1 Comparação Entre Algoritmos de Quantização de Imagens

Os tipos de comparações feitas com os algoritmos apresentados são: comparação visual, onde são analisadas as imagens como são apresentadas; comparação numérica, que são comparadas quanto ao erro de quantização gerado e comparação dos tempos de execução, em que as imagens são comparadas tendo em vista o tempo de execução dos algoritmos de quantização na forma de programa de computador, sendo este o resultado da implementação.

7.1.1 Comparação Visual

Para fazer a comparação visual dos algoritmos, pega-se a imagem de teste e esta é quantizada para 256 e 16 cores, utilizando cada uma das técnicas de quantização de imagens apresentadas. Tem-se como resultado as imagens a seguir.

É importante salientar que a comparação visual das imagens depende de cada observador, sendo esta comparação uma opinião pessoal.


Quantização por Corte Mediano


Quantização por Octree


Quantização por Aglomerados Duplos

Figura 7-1: Quantização da imagem da pêra para 256 cores (esquerda) e para 16 cores (direita)

A Quantização por Aglomerados Duplos mostrou, tanto em 256 como em 16 cores, um melhor resultado visual que as outras técnicas de quantização de imagens.

A técnica de quantização por octree mostrou um melhor resultado visual que a quantização por corte mediano na quantização para 256 bits, mas na quantização para 16 bits, a quantização por corte mediano teve um resultado melhor.

Na quantização por octree para 16 bits, ocorre a perda de algumas cores, e as que restaram tomam conta da imagem. Isso se dá pelo fato de que a

quantização por octree procura substituir o histograma original por um histograma com uma distribuição uniforme das intensidades das cores.

7.1.2 Comparação Numérica

O critério utilizado para a comparação numérica dos algoritmos apresentados é o erro de quantização.

O erro de quantização do pixel é a distância do pixel à sua cor quantizada, sendo assim, pode-se contar para cada imagem a média dos erros de quantização como sendo a soma dos erros de quantização dos pixels dividido pela quantidade de pixels que existe na imagem.

Algoritmo de Quantização	Erro de Quantização por Pixel
Quantização por Corte Mediano	91.4262
Quantização por Octree	89.5336
Quantização por Aglomerados Duplos	21.4209

Tabela 7-1: Erro de quantização por pixel gerado pela quantização da imagem da pêra para 256 cores.

Algoritmo de Quantização	Erro de Quantização por Pixel
Quantização por Corte Mediano	141.5762
Quantização por Octree	598.2884
Quantização por Aglomerados Duplos	128.9642

Tabela 7-2: Erro de quantização por pixel gerado pela quantização da imagem da pêra para 16 cores.

Tanto no caso da quantização para 256 cores, quanto no caso da quantização para 16 cores, novamente o algoritmo de quantização por aglomerados duplos mostrou-se superior aos outros, minimizando o erro de quantização gerado.

No caso do algoritmo de quantização por octree, como foi observado na comparação visual, mostrou um erro de quantização maior que os outros algoritmos no caso da quantização para 16 cores, mas seu erro de quantização foi menor que o algoritmo de quantização por corte mediano, no caso da quantização para 256 cores.

Com isso, pode-se concluir que o algoritmo de quantização por aglomerados duplos é sempre superior em termos de resultados do que os outros algoritmos apresentados. Também pode-se concluir que o algoritmo de quantização por octree é menos eficiente para uma quantização para um número de cores pequeno como no caso da quantização para 16 cores.

É importante salientar que todos os algoritmos são inicialmente quantizados uniformemente para 15 bits (5 bits para o vermelho, 5 bits para o verde e 5 bits para o azul), o que aumenta o erro de quantização mas diminui expressivamente os tempos de execução dos mesmos. Essa pré-quantização não é muito perceptível visualmente.

7.1.3 Comparação dos Tempos de Execução

É importante observar que os tempos de execução são comparados conforme o resultado da implementação dos algoritmos. A implementação pode não ter uma otimização adequada para ter-se um cálculo de tempo expressivo de cada algoritmo, mas pode ser utilizada para uma comparação entre os mesmos.

Isto é devido ao fato de que a preocupação principal na implementação é procurar manter fidelidade à técnica estudada e não a procura de uma implementação otimizada em relação aos tempos de execução.

O algoritmo que mostrou um tempo de execução melhor, mais rápido, foi o algoritmo de quantização por octree, tanto no caso da quantização para 16 cores como para 256 cores, sendo que o tempo de execução para 16 cores foi menor.

O algoritmo de quantização por corte mediano mostrou-se ,em média, 3 vezes mais lento que o algoritmo de quantização por octree, mas tanto no caso da quantização para 16 cores como no caso da quantização para 256 cores, mostrou-se com um tempo de execução constante.

O algoritmo de quantização por aglomerados duplos mostrou-se o mais lento em todos os casos, tendo um tempo de execução de até 5 vezes mais lento que o algoritmo de quantização por corte mediano.

7.2 Pontos Observados

Nesta parte do trabalho é feita uma breve análise, apresentando alguns pontos observados durante o estudo dos algoritmos apresentados e da implementação em linguagem de programação de cada um deles.

7.2.1 Algoritmo de Quantização por Corte Mediano

O algoritmo de quantização por corte mediano mostrou-se de fácil implementação e tempos de execução constantes.

Os resultados visuais e numéricos são bons tanto no caso de quantização para 256 cores como no caso da quantização para 16 cores.

Este algoritmo ainda pode ser adaptado e modificado para se obter resultados mais expressivos.

7.2.2 Algoritmo de Quantização por Octree

O algoritmo de quantização por octree mostrou-se bastante eficiente em termos de tempo de execução devido ao fato que torna rápida a construção da árvore e o mapeamento das cores aos seus níveis de quantização, uma vez que este trabalho não passa de uma procura recursiva em uma árvore de oito caminhos e a árvore tem apenas até oito níveis de profundidade.

O algoritmo só não mostrou-se muito eficiente no caso em que quer-se quantizar a imagem para poucas cores, como no caso da quantização para 16

cores. Porém, em casos onde o tempo de execução é um fator crucial, a velocidade do algoritmo pode compensar o resultado visual do algoritmo.

Também observa-se que o algoritmo só perde quando é quantizado para um número bastante pequeno de cores, sendo mais eficiente na quantização para 256 cores do que o algoritmo do corte mediano, tanto na minimização do erro de quantização como no tempo de execução.

É importante também salientar que pela função de redução quando é extrapolado o número de cores para a qual se quer quantizar algumas folhas são reduzidas da árvore, assumindo seu nó pai como folha, contendo a média dos valores das filhas. Este procedimento, pode levar a uma imagem com um número de cores menor do que se espera quando mais de duas folhas são reduzidas na última iteração.

7.2.3 Algoritmo de Quantização por Aglomerados Duplos

O algoritmo de quantização por aglomerados duplos mostrou-se superior em relação aos outros algoritmos apresentados, tanto na quantização para 16 cores como na quantização para 256 cores, na minimização do erro de quantização por pixel e no resultado visual. Porém, este algoritmo mostrou-se bastante lento em relação ao tempo de execução.

Isto se dá pelo fato de que para cada iteração do algoritmo, cada cor é comparada com todas as outras para se encontrar a combinação de duas cores que resulta no menor erro de quantização.

Este algoritmo pode não ser muito bom quando o tempo de execução é um dos fatores cruciais para a aplicação, mas pode ser usado eficientemente quando o tempo é um fator não muito importante.

Este algoritmo necessita de uma enorme quantidade de memória, se comparado com os outros algoritmos. Isto se deve ao fato de que o algoritmo de quantização por aglomerados duplos precisa de uma matriz de erros de n x n,

sendo que n é o número de cores que existem na imagem, e quanto maior o número de cores a quantizar, maior será a matriz de erro e maior será a quantidade de memória a alocar.

O algoritmo de quantização por aglomerados duplos, por ter se mostrado bastante eficiente no resultado visual e na minimização do erro de quantização, precisa ser trabalhado e modificado para que se possa diminuir o tempo de execução do mesmo, de forma a competir com os outros algoritmos.

REFERÊNCIAS BIBLIOGRAFICAS

(1) BALASUBRAMANIAN, R., BOUMAN, C. A. & ALLEBACH, J Sequential Scalar Quantization of Color Images. Journal of Electronic Imaging, vol 3, nº 1 p.45-59, 1994.

(2) BALASUBRAMANIAN, R., BOUMAN, C. A . & ALLEBACH, J Sequential Scalar Quantization of Vectores: Na analysis. IEEE Transactions on Image Processing, vol. 4, no. 9, p 1282-1295, 1995

(3) BOUMAN, Charles; ORCHARD, Michael. Color quantization of images. IEEE Transactions on Signal Processing, vol. 39, nº 12, p. 2677-2690, 1991.

(4) GERVAUTZ, Michael, PURGATHOFER, A simple method for color quantization: Octree quantization. Glassner, ªS. (ed), *Graphics Gems.*Cambridge, MA:Academic Press Professional. p 287-293. 1990.

(5) GOMES, Jonas, VELHO, Luiz. Computação gráfica: imagem. Rio de Janeiro : IMPA/SBM, 1994. 424p.

(6) GOMES, Jonas, VELHO, Luiz, SOBREIRO, Marcos. Color image quantization by pairwise clustering. Rio de Janeiro. SIBGRAPI 97, 1997.

(7) ORCHARD, Michael, BOUMAN, Charles, Color Quantization of Images. IEEE Trans. Vol 39, no. 12, p 2677-2690, 1991.

(8) PERRY, Greg. Programação orientada para objeto com turbo C++. Rio de Janeiro : Berkeley, tradução Rolv Riegger. 1994.

(9) SCHILDT, Herbert. C: Completo e total. São Paulo, Makron, McGraw-Hill, tradução Marcos R. A . Moraes,1990.

(10) SOBREIRO, Marcos V. R., Quantização de imagens. Rio de Janeiro, 1998. Tese (Mestrado em Computação Gráfica) - PUC do Rio de Janeiro.

(11) WAN, S, WONG, S & PRUSINKIEWICZ, P. An Algorithm for Multidimensional Data Clustering. ACM Transactions on Mathematical Software, vol 14, no 2, 1998, p153-162.

(12) ZIVIANI, Nivio. Projeto de algoritmos: com implementações em Pascal e C. São Paulo: Pioneira, 1993.