Quantização de Imagens

Marcos Vinícius Rayol Sobreiro

22 de setembro de 1998

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Estaremos durante este trabalho estudando o problema de quantização de imagens.

O que é quantizar uma imagem? De modo simples e objetivo, quantizar uma imagem é representar uma imagem que tem M cores por uma outra imagem que tenha N cores, onde $N < M$. Geralmente estamos tratando do seguinte problema, temos uma imagem de 24-bits, ou seja, 16 milhões de cores e queremos representá-la com 256 cores. Este processo de escolher as 256 melhores cores para representar as cores presentes na imagem de 24-bits e depois mapear cada cor da imagem original nas cores escolhidas é chamado de quantização da imagem.

Assim, o principal objetivo da quantização de uma imagem colorida é fazer o mapeamento de um conjunto de cores usados na imagem original para um conjunto de cores muito menor usado na imagem quantizada, que deve ser escolhido da melhor maneira de modo a minimizar a distorção visual perceptível. Entretanto, minimizar a distorção visual não é uma tarefa muito simples e nem mesmo muito bem definida, devido a papéis complexos de fatores tais como erros de quantização (discrepâncias 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.

Quantização de imagens coloridas é necessário quando mostramos imagens coloridas de tons contínuos em monitores onde não temos 24-bits para respresentar estas cores. Apesar de hoje em dia as placas de vídeo estarem mais baratas e a maioria delas já nos possibilitem mostrarmos imagens de 24-bits nos monitores, o estudo de quantização de imagens ainda é muito importante, visto a quantidade de novas publicações que estão sempre surgindo na literatura.

Mesmo quando todos os computadores puderem contar com placas de vídeo capazes de representar cores de 24-bits, a quantização de imagens coloridas ainda terá seu valor prático, assim podemos aliviar um espaço considerável nos "frame buffers"para tarefas como animação, transparência, aplicações de janelas, e outras funções gráficas. Além disso, a quantização de imagens possibilita a diminuição do tamanho físico ocupado por uma imagem, possibilitando diminuição do espaço espaço necessário na armazenagem de imagens. Com imagens menores, podemos diminuir a largura de banda para sua transmissão que são os

gargalos em muitas aplicações, particularmente quando a computação gráfica é integrada nos sistemas multimídia ou quando a tecnologia de HDTV (High Definition Television) se tornar comum.

Neste trabalho daremos uma introdução à imagem digital e seus elementos básicos. Explicitaremos o problema de quantização de imagens e as possíveis estratégias para resolvê-lo. Mostraremos vários algoritmos para quantização de imagens. A ênfase deste trabalho será em algoritmos de quantização de imagens que se utilizam de técnicas de aglomerados, onde apresentaremos uma nova proposta de algoritmo para quantização de imagens. Este estudo termina com uma comparação entre os diversos algoritmos estudados.

Existem muitos trabalhos na área de quantização de imagens. Nem todos os trabalhos estudados estão sendo apresentados na tese, somente os mais relevantes serão estudados. Uma completa referência de todos os trabalhos da área pode ser encontrada no final desta tese.

Um breve resumo de cada capítulo pode ser visto a seguir.

Capítulo 2 - Discretização de Cor e Imagem Neste capítulo vamos definir o conceito de imagem digital e seus principais elementos. O problema de quantização de imagem será definido e analisaremos as estratégias existentes para realizarmos a quantização de uma imagem.

Capítulo 3 - Quantização por Seleção Direta Uma estratégia de quantização por seleção direta será o alvo de estudo deste capítulo. O algoritmo da populosidade, talvez o primeiro algoritmo de quantização de imagem que se tem notícia, será apresentado, apresentando seus prós e contras.

Capítulo 4 - Quantização por Subdivisão Espacial Os mais importantes algoritmos de quantização que se utilizam da estratégia de subdivisão espacial serão estudados neste capítulo. A maioria dos algoritmos existentes se utilizam desta estratégia para quantizar imagens. Dentre os métodos mais importantes que estaremos estudando neste trabalha, podemos citar o algoritmo do corte mediano, o método de quantização mais utilizado.

Capítulo 5 - Quantização por Métodos de Otimização Neste capítulo apresentaremos a idéia de algoritmos de quantização que se utilizam de métodos de otimização. Na prática todo algoritmo de quantização tenta minimizar alguma coisa, quase sempre o erro de quantização. Assim, estudaremos neste capítulo algumas técnicas de minimização aplicadas à quantização de imagens.

Capítulo 6 - Quantização por Aglomerados Os algoritmos de quantização de imagens mais recentes geralmente se baseiam em técnicas de aglomerados. Neste capítulo faremos uma breve introdução à análise de aglomerados e estudaremos alguns algoritmos de quantização de imagens que se utilizam desta estratégia.

Capítulo 7 - Quantização por Aglomerados Duplos Baseado em uma técnica de aglomerados estamos propondo neste capítulo um novo algoritmo de quantização de imagens. O algoritmo proposto apresenta resultados muito bons do ponto de vista perceptual e numérico apesar de ainda ser mais lento do que a maioria dos algoritmos.

Capítulo 8 - Conclusões Vamos concluir este trabalho com um estudo comparativo entre os diversos algoritmos estudados. Neste capítulo daremos algumas outras utilizações para o algoritmo de quantização por aglomerados duplos. Também serão apresentadas algumas idéias no sentido de diminuir o tempo de execução do algoritmo e melhorar ainda mais seus resultados.

Capítulo 2

Discretização de Cor e Imagem

Neste capítulo descrevemos o conceito de imagem digital e alguns de seus elementos necessários para o desenvolvimento do estudo de quantização de imagens.

2.1 Imagem Digital

O objetivo final da maioria das aplicações de Computação Gráfica é a geração de uma imagem. Assim sendo, o estudo da imagem digital e de sua manipulação no computador (processamento de imagem) é de grande importância dentro da área.

Para representar e manipular imagens no computador temos que definir um modelo matemático apropriado para tal finalidade. Usaremos o paradigma dos Quatro Universos para nossos estudos sobre imagem.

2.1.1 Paradigma dos Quatro Universos

Este paradigma de abstração (Gomes & Velho, 1995a) consiste em estabelecer quatro universos (conjuntos), como mostrados na Figura 2.1.

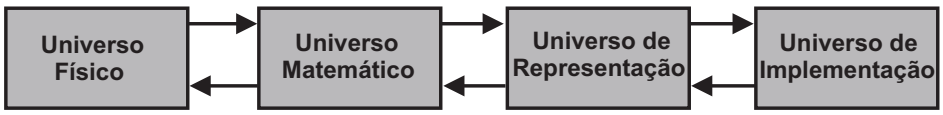


Figura 2.1: Os quatro universos de abstração.

O Universo Físico (F) contém os objetos do mundo real os quais queremos estudar. Uma descrição abstrata dos objetos do mundo físico está no Universo Matemático (M). O Universo de Representação (R) é constituído das representações discretas associadas aos objetos do Universo Matemático. Finalmente, no Universo de Implementação (I), os objetos já discretizados do Universo de Representação são mapeados em estruturas de dados possibilitando sua representação no computador.

Modelo Físico de Imagem

Tudo que podemos ver no mundo real pode ser considerado uma imagem. Assim, devemos entender uma imagem no Universo Físico como o significado intuitivo da propria palavra imagem. Como exemplo de imagens podemos citar uma imagem da retina do olho humano, uma imagem capturada por uma câmera de TV, uma fotografia, enfim, toda cena do mundo real que vemos chega ao nosso cérebro como uma imagem.

Modelo Matemático de Imagem

Tomando uma fotografia como exemplo, percebemos esta imagem como sendo impulsos luminosos trazendo informações de cor que chegam aos nossos olhos partindo de cada ponto do espaço desta fotografia. Assim, um modelo matemático natural para uma imagem é uma função definida em uma superfície bidimensional e que toma valores em um espaço de cor.

Uma imagem consiste de uma aplicação que associa a cada ponto do plano uma informação de cor

$$f : U \subset \mathbb{R}^2 \Leftrightarrow\!\!\rightarrow C^n.$$

A função f é chamada de *função imagem.* Chamamos de *suporte geométrico da imagem* o conjunto U. O conjunto de valores de f, que são as cores presentes na imagem formam um subconjunto de C conhecido como *gamute de cores* da imagem. Para imagens coloridas, temos um espaço de representação de cor tricromático, em geral com as bases primárias R, G e B - do inglês Red (vermelho), Green (verde) e Blue (azul). Neste caso, $n = 3$. Quando $n = 1$, temos as imagens que apresentam somente uma cor - *monocromáticas.*

Modelo de Representação de Imagem

Quando representamos uma imagem $f : U \subset \mathbb{R}^2 \Leftrightarrow\!\!\rightarrow C^n$ devemos levar em consideração sua *representação espacial* - que é a representação do conjunto suporte de U - e sua *representação de cor* - que é a representação do espaço de cor de f.

Representação Espacial O caso de representação espacial de uma imagem que estamos interessados é a amostragem pontual uniforme. Considere o conjunto suporte como sendo o retângulo

$$f : U = [a, b] \times [c, d] = \{(x, y) \in \mathbb{R}^2; a \leq x \leq b \quad e \quad c \leq y \leq d\}.$$

Discretizamos esse retângulo usando os pontos de um reticulado bidimensional $\triangle = (\triangle_x, \triangle_y)$,

$$\triangle = \{(x_j, y_k) \in U; \quad x_j = j \cdot \triangle x, \quad y_k = k \cdot \triangle y \quad j, k \in \mathbb{Z}, \quad \triangle x, \triangle y \in \mathbb{R}\},$$

conforme mostrado na Figura 2.2. A imagem é representada pelos valores $f(x_j, y_k)$ da função imagem nos vértices do reticulado. Cada uma destas amostras é chamada de *pixel.* Cada

pixel (x_j, y_k) da imagem pode portanto 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 $m \times n$, $A = (a_{jk}) = (f(x_j, y_k))$.

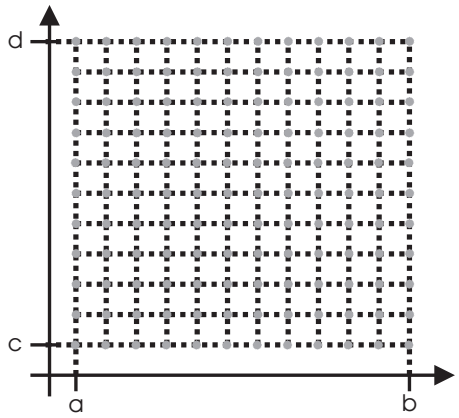


Figura 2.2: Reticulado uniforme da representação matricial da imagem.

Cada elemento a_{jk}, $j = 1, \ldots, m$ e $k = 1, \ldots, n$ 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 luminância do pixel.

Representação de Cor O espaço de cor é representado pelo espaço $\mathbb{R}^3$, portanto o problema de representação de cor é o problema de representação de números reais. Logo, devemos saber quantos bits devemos utilizar para representar uma cor. Chamamos de *resolução de cor* da imagem ao número de bits que utilizamos para representação de cor no computador. Como já mencionado, esse processo de discretização do espaço de cor de uma imagem é chamado de *quantização*.

Do ponto de vista computacional a discretização de cor está relacionada diretamente com o problema de discretização do espaço $\mathbb{R}^3$ e temos opções de usar aritmética de ponto flutuante com 32 ou 64 bits. No entanto do ponto de vista de exibição de imagens a questão é mais delicada. O problema de discretização do espaço de cor para exibição de imagem é conhecido como *quantização de cor*.

2.1.2 Elementos da Imagem Digital

Uma imagem digital é uma imagem $f : U \Leftrightarrow\!\!\!\rightarrow \mathbb{R}^3$ onde o suporte U e o espaço de cores estão discretizados. Esta imagem fica caracterizada pelos seguintes fatores:

- resolução espacial (número de pixels);
- número de componentes de cor (dimensão do espaço de cor);

- resolução de cor;

Chamamos de *gamute de cores* de f ao conjunto de cores $f(U)$ do espaço de cor discretizado, que é finito.

2.1.3 Histograma de Frequência de Cor

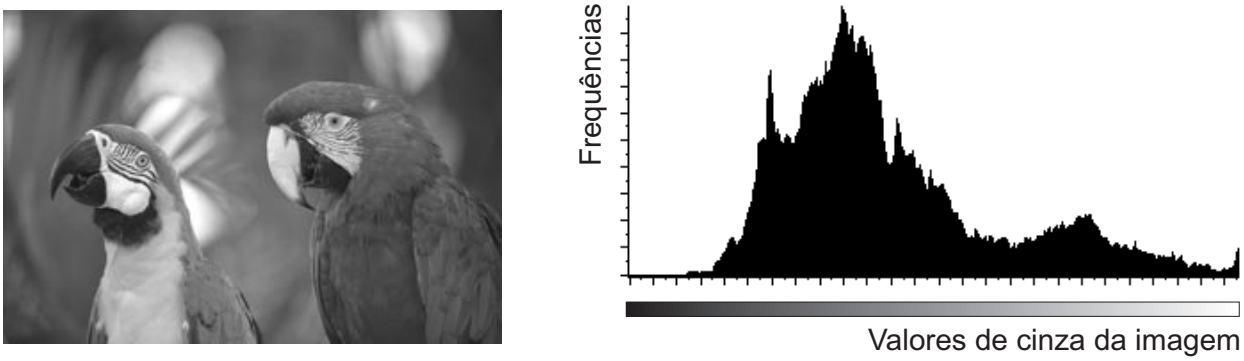


Figura 2.3: Histograma de uma imagem em tons de cinza.

Para muitas aplicações com imagens estaremos interessados em saber qual é a distribuição de probabilidade de cor de uma imagem.

Em geral, é muito difícil saber a distribuição de probabilidade de cor de uma imagem. Uma aproximação dessa distribuição é dada pelo *histograma de cor*. Nesse histograma associamos a cada intensidade de cor c presente na imagem sua frequência de ocorrência, isto é, o número de pixels na imagem que têm a cor c. A Figura 2.3 mostra a imagem das araras e seu histograma. O eixo horizontal mostra a gradação dos 256 níveis de cinza (de preto até o branco) e o eixo vertical o número de vezes que cada cor ocorre da imagem (frequência).

Observando o histograma podemos notar que esta imagem das araras quase não possui cores próximas do preto (baixa intensidade). A maioria dos pixels se concentram em tons de cinza médios. Os pixels de alta intensidade (próximo ao branco) também estão presentes na imagem porém em menor número que os pixels de tons médios de cinza.

No caso de imagens coloridas, podemos computar separadamente o histograma para cada componente ou criar um histograma tridimensional.

2.2 Definição do Problema de Quantização de Imagens

A quantização de uma imagem digital consiste em quantizar o gamute de cores da imagem, o que acarreta na quantização da informação de cor de cada pixel da imagem. Mais precisamente, se $f : U \to \mathbb{R}^3$, é uma imagem digital, o resultado da quantização de $f(x, y)$ é uma imagem $f' : U \to R_k$, tal que $f'(x, y) = q(f(x, y))$, onde q é a transformação de quantização. Desse modo, a quantização altera a resolução de cor da imagem.

Um caso comum de quantização ocorre quando queremos representar um conjunto finito R_M de M cores, por um conjunto R_N com N cores, sendo $M > N$. Isso corresponde por exemplo a quantizar um conjunto de cores representado por m bits para um conjunto de cores representado por n bits, com $m > n$. Nesse caso temos uma transformação de quantização $q : R_M \to R_N$.

Por que quantizar uma imagem? Basicamente existem duas razões: exibição e compressão.

Exibição de Imagem Para exibirmos uma imagem em um dispositivo gráfico é necessário que este suporte a resolução de cor da imagem. Ou seja, o gamute de cor do dispositivo deve ser maior ou igual ao gamute de cor da imagem. Se isto não ocorre, devemos quantizar esta imagem para podermos exibí-la neste dispositivo.

Compressão de Imagem Com a quantização de uma imagem, reduzimos o número de bits usados para armazenar seu gamute de cor. Isso reduz a quantidade total de memória necessária para armazenar a imagem e a quantidade de dados necessária para transmitir a imagem através de um canal de comunicação.

2.2.1 Células de Quantização e Níveis de Quantização

Uma quantização particiona o espaço de cor da imagem em subconjuntos, em cada um desses subconjuntos a função de quantização assume um único valor. Genericamente, considere um mapa de quantização $q : C \to C'$. Para cada cor quantizada $c'_i \in C'$ corresponde um subconjunto de cor $C_i \subset C$, consistindo de todas as cores em C que são mapeadas em c'_i

$$C_i = q^{-1}(c'_i) = \{c \in C : q(c) = c'_i\}.$$

A família (finita) de conjuntos C_i formam uma partição do espaço de cor C. Cada conjunto C_i é chamado de *célula de quantização*.

Em cada célula de quantização a função de quantização assume valores constantes c'_i. Chamamos de *nível de quantização* ou *valor de quantização* a esse valor representativo de cada célula de quantização.

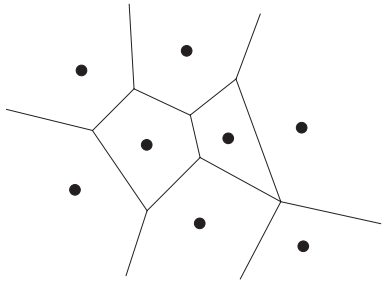


Figura 2.4: Células de quantização bi-dimensional e seus respectivos níveis de quantização.

A Figura 2.4 mostra um exemplo de células de quantização de um espaço de cor bi-dimensional com seus respectivos níveis de quantização.

2.3 Quantização e a Geometria das Células

Como vimos pela definição do problema de quantização de imagens (seção 2.2), esse processo se caracteriza pela subdivisão do espaço de cor em subespaços sem interseção e que a união de todos estes subespaços é o espaço de cor original. Assim, podemos dizer que todo método de quantização de imagens nada mais é do que um processo de subdivisão espacial.

Estes subespaços gerados pelo processo de quantização são conhecidos como células de quantização. Segundo sua geometria, estas células podem ser classificadas em duas categorias:

- uniforme
- não-uniforme.

2.3.1 Quantização Escalar e Vetorial

Quando os espaços de cor C (imagem original) e C' (imagem quantizada) têm dimensão 1, o processo de quantização é chamado de quantização uni-dimensional.

Quando o espaço de cor tem dimensão n, e a quantização de cada vetor de cor $c = \{c_1, c_2, \ldots, c_n\}$ é feita quantizando-se cada componente c_i separadamente, temos uma quantização escalar. Neste caso, temos um mapa de quantização uni-dimensional q, e o mapa de quantização $Q : C \to C'$ é definido por

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

Quando a quantização não é escalar, ou seja, consideramos todos os componentes c_i ao mesmo tempo, temos uma *quantização vetorial*.

Quando quantizamos um espaço de cor usando um método de quantização escalar não levamos em consideração a correlação espacial das cores presentes no gamute da imagem sendo quantizada. Métodos de quantização vetoriais eliminam este problema pois escolhemos as células de quantização levando em consideração todas as componentes de cor presentes na imagem, considerando assim a correlação espacial destas cores.

Na Figura 2.5 (A) temos um espaço de cor bi-dimensional onde as cores são representadas por círculos pretos. Usando um método de quantização escalar, obtemos a quantização mostrada nesta mesma figura, onde os níveis de quantização são representados por círculos cinza. Usando uma técnica de quantização vetorial obtemos a quantização mostrada na Figura 2.5 (B). Note que esta quantização tem o mesmo erro de quantização da quantização da mostrada Figura 2.5 (A) porém temos um número bem menor de células de quantização. Geramos 10 células usando quantização vetorial, ao invés das 16 células geradas usando quantização escalar uniforme. Nas Figuras 2.5 (C) e (D) temos dois exemplos de quantização escalar que geram o mesmo número de células obtidos na quantização vetorial. Note que o

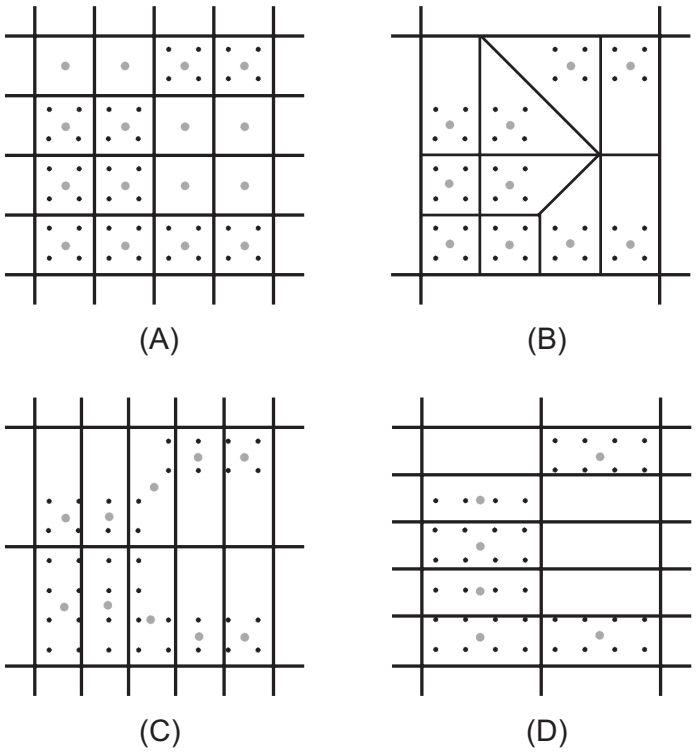


Figura 2.5: Quantização escalar X Quantização vetorial.

erro de quantização é bem maior nestas duas quantizações que na quantização obtida usando um método vetorial. Por que isto acontece? Isto se deve ao fato de que na quantização vetorial levamos em conta a correlação espacial das cores, fato que não ocorre na quantização escalar.

2.3.2 Quantização Uniforme

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

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

Na Figura 2.6 (A) temos uma quantização escalar de $\mathbb{R}^2$ obtida a partir de uma quantização uniforme em cada um dos eixos coordenados. Na Figura 2.6 (B) mostramos uma outra geometria de célula de quantização uniforme bidimensional.

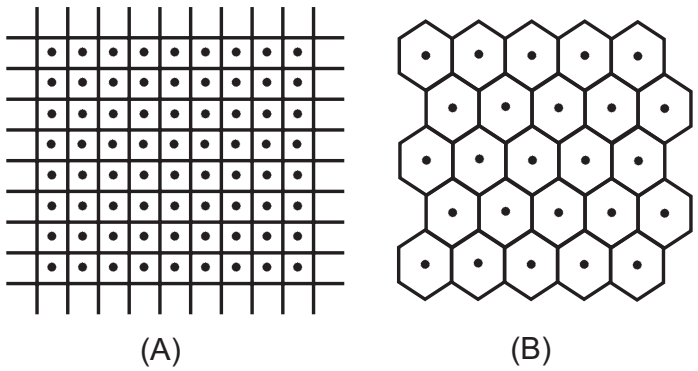


Figura 2.6: Células de quantização uniforme bidimensional e seus respectivos níveis de quantização.

2.3.3 Quantização Não-uniforme

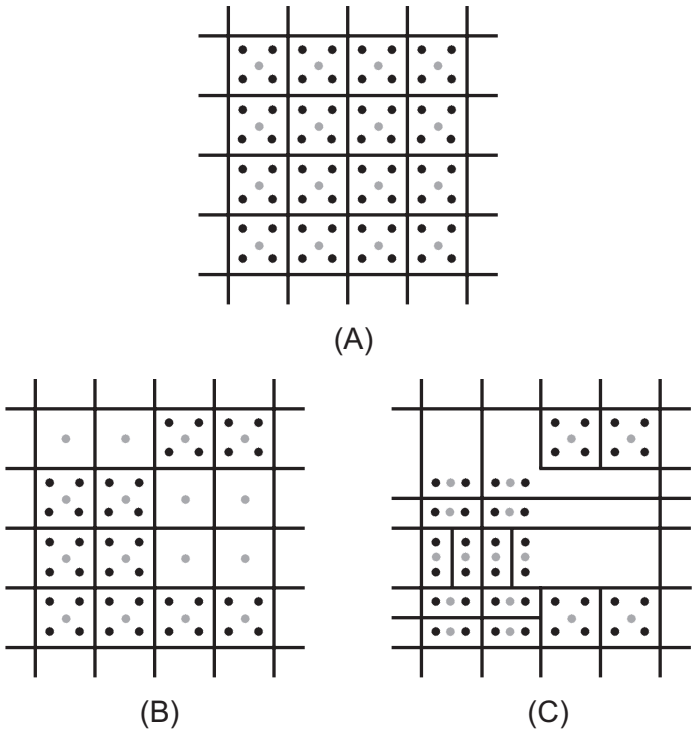


Figura 2.7: Células de quantização adaptadas à distribuição das cores na imagem.

A Figura 2.7 mostra alguns exemplos de quantização bi-dimensional para 16 níveis. As bolas pretas representam as cores no gamute da imagem que se deseja quantizar. As bolas cinzas são os níveis de quantização de cada célula. Usaremos esta figura para motivar quantização não-uniforme.

A quantização uniforme é um método ótimo quando temos uma distribuição uniforme das cores pelo espaço de cor, como pode ser observado na Figura 2.7 (A). Observe a Figura 2.7 (B), nela temos uma distribuição de cores não uniforme. Como podemos notar, existem células de quantização em que não temos cores presentes no gamute da imagem. Assim, apesar da quantização uniforme ser muito fácil de ser obtida, nem sempre ela é a mais recomendada. Basta observarmos que com esse método podem haver células de quantização que não terão utilidade alguma pois pode não haver cor alguma no gamute da imagem que pertença a esta célula.

Se a distribuição de cores de uma imagem não é 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 o intuito de diminuirmos o erro de quantização. Podemos observar pela Figura 2.7 (C) que continuamos com 16 células de quantização e que o erro de quantização (distância entre as cores de uma célula e seu nível de quantização) diminuiu.

Logo, um método de quantização que não se utiliza da subdivisão do espaço em células congruentes é chamado de *quantização não-uniforme*. A quantização não-uniforme é dita *adaptativa* quando a geometria das células é escolhida de acordo com características específicas da distribuição de cor na imagem. Nestes métodos geralmente levamos em conta informações que nos são fornecidas pelos histogramas de cor da imagem, que nos informam sobre a distribuição de probabilidade de cor na imagem.

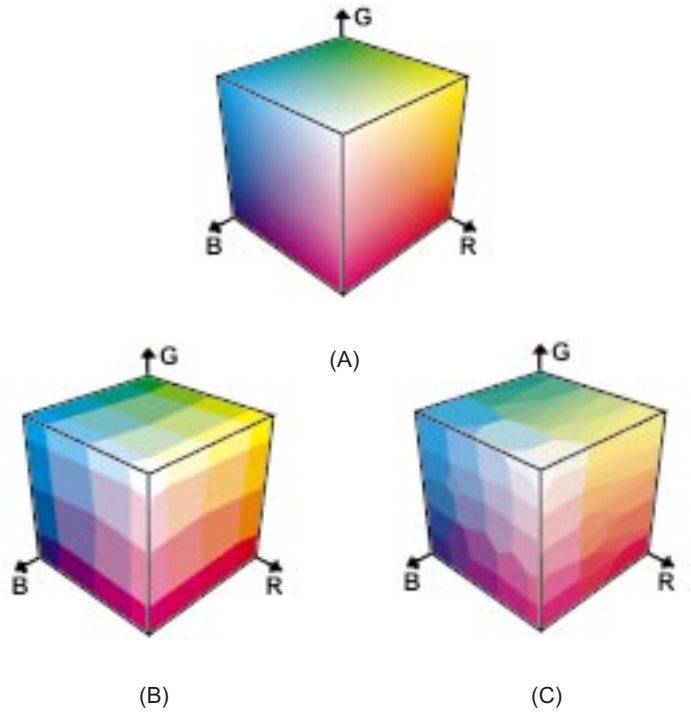


Figura 2.8: Células de quantização na quantização do cubo RGB.

Na Figura 2.8 (A) mostramos o cubo RGB. Este mesmo cubo quantizado por um método uniforme pode ser visto na Figura 2.8 (B). Uma quantização não-uniforme adaptativa do

cubo RGB é apresentada na Figura 2.8 (C).

2.3.4 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 $f : U \subset \mathbb{R}^2 \to C$ tomando valores em algum espaço de cor C. Definimos um mapa de cor $\varphi : [0,1] \subset \mathbb{R} \to C$, e os valores das cores da imagem são tomados como um conjunto dos valores do mapa de cor $\varphi([0,1]) \subset C$ (veja Figura 2.9).

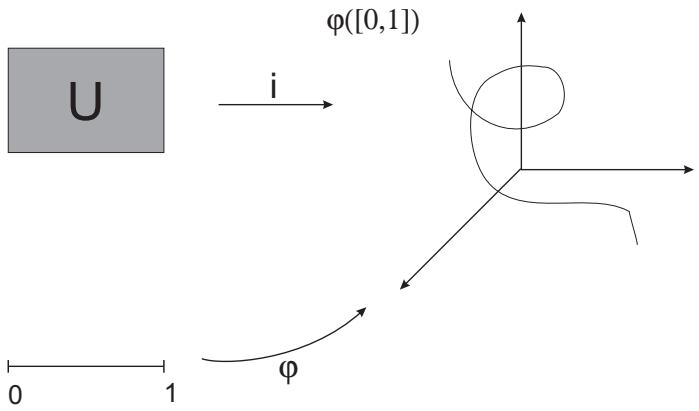


Figura 2.9: Mapa de cor de uma imagem.

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 $j = 1, \ldots, n \Leftrightarrow 1$, 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.

2.4 Erro de Quantização

A determinação ótima das células de quantização, e do nível de quantização para cada célula, depende do critério usado para medirmos o erro de quantização e da distribuição de cores na imagem. Se q é o mapa de quantização e c é a cor que será quantizada, podemos escrever

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

onde e_q é o *erro de quantização* ou *distorção*.

A distorção é causada quando substituímos a cor c pelo seu valor de quantização $q(c)$. Para medirmos a distorção usamos uma medida de distorção $d(c, q(c))$. Existem várias possibilidades para a escolha da medida de distorção d em C. Ao escolhermos uma, devemos levar em conta critérios perceptuais bem como eficiência computacional em seu cálculo. É comum usarmos uma pseudométrica ao invés de uma métrica, ou até mesmo alguma função positiva que, de alguma forma, nos forneça informação de "proximidade"no espaço de cor. Uma possível escolha é o quadrado da distância Euclidiana, $d(c_1, c_2) = \langle c_2 \Leftrightarrow c_1, c_2 \Leftrightarrow c_1 \rangle$.

A quantização de uma imagem implica na quantização de cor de cada um de seus pixels. Logo, uma medida da distorção deve levar em conta não somente a distorção causada pela quantização de cada cor do espaço de cor da imagem, mas também a frequência de ocorrência dessa cor na imagem. Uma boa medida é dada pelo erro médio quadrático

$$E(c, q(c)) = \int_C p(c) d(c, q(c)) dc, \tag{2.1}$$

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.

2.5 Quantização como um Problema de Otimização

O que é o ideal de um processo de quantização? Podemos dizer que uma quantização é ótima quando o erro gerado pelo processo é mínimo. Como vimos anteriormente, existe um erro associado ao processo de quantização. Este erro pode ser medido pela equação

$$E(c, q(c)) = \int_{-\infty}^{+\infty} p(c) d(c, q(c)) dc. \tag{2.2}$$

Se desejamos uma quantização em N níveis, teremos uma partição do espaço de cor em N células K_1, K_2, ... , K_N. Indicando por q_k o nível de quantização da célula K_k, a equação acima pode ser escrita na forma

$$E(c, q(c)) = \sum_{1 \leq j \leq N} \int_{K_j} p(c) d(c, q_j) dc. \tag{2.3}$$

Ou ainda, levando em consideração que temos um conjunto finito de cores em cada célula,

$$E(c, q(c)) = \sum_{1 \leq j \leq N} \sum_{c \in K_j} p(c) d(c, q_j). \tag{2.4}$$

O problema de quantização deveria portanto ser resolvido, de forma ideal, minimizando a distorção de quantização dada pela equação 2.4, sobre todas as possíveis N partições com

K elementos do espaço de cor com um número finito de elementos. A grande variedade de partições com K elementos do espaço de cor, torna esse problema bastante difícil do ponto de vista computacional. Este é considerado um *problema de decisão*, ou seja, a cada instância do problema queremos saber se uma dada configuração de N partição do espaço de cor minimiza o erro de quantização dado pela equação 2.4. Este tipo de problema é considerado um problema NP-completo. Um problema NP-completo é aquele que não existe uma solução em tempo polinomial para resolvê-lo, uma vez encontrada uma solução polinomial para um determinado tipo de problema NP-completo todos os demais problemas dessa classe também terão um solução polinomial. Assim, devido à complexidade do problema, em geral os métodos de otimização utilizados para resolver o problema de quantização não resolvem o problema em sua plenitude. Esses métodos de solução de tal problema em geral se enquadram em uma das categorias abaixo:

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

O problema de quantização uni-dimensional é resolvido completamente de forma ótima como será visto mais adiante na seção 5.1.

Como nosso espaço de cor já é discretizado para 24-bits, o problema de otimização na quantização nada mais é do que um problema de análise de aglomerados, ou seja, estamos procurando aglomerados de cores que minimizam o erro de quantização. Na seção seguinte entraremos um pouco mais a fundo na relação entre quantização e análise de aglomerados.

2.5.1 Quantização e Análise de Aglomerados

O problema de Aglomerados

Análise de aglomerados é uma técnica importante usada na busca de alguma estrutura em um conjunto discreto de dados. Como no caso de imagens já temos o espaço de cor discretizado para 24-bits, podemos pensar em usar algumas dessas técnicas para quantizar imagens. Dado um conjunto finito de dados, S, o problema de aglomerado em S consiste em encontrarmos uma coleção de "centros"de aglomerados que podem caracterizar apropriadamente classes relevantes de S. Na análise de aglomerados clássica, essas classes devem formar uma partição de S, de tal forma que o grau de associação seja considerado forte para os dados contidos dentro da partição e fraco para os dados contidos em outras partições.

Quantização como um Problema de Aglomerados

Considerando o espaço de cor como sendo um conjunto de dados, onde cada dado representa uma cor. O problema de aglomerado de cores consiste em encontrarmos conjuntos de cores similares. Como o conceito de similaridade de cor está ligado a quão próximas estas cores estão no espaço de cor, podemos naturalmente escrever o problema de quantização como

sendo um problema de otimização em análise de aglomerado: a solução ótima deve minimizar o erro de quantização (equação (2.2)) em todas as possíveis partições de N-elementos do espaço de cor.

Assim, no caso de quantização de imagens, devemos identificar N aglomerados de cores da imagem original que sejam similares de acordo com as medidas de quantização. Estes aglomerados vão constituir as células de quantização. Para cada aglomerado temos um nível de quantização associado. Como as cores em cada aglomerado são similares, podemos substituí-las pelo seu respectivo nível de quantização introduzindo uma distorção mínima.

A solução direta do problema de otimização em aglomerados implica em uma busca no espaço de todos as possíveis configurações de aglomerados para encontrarmos a que nos fornece o mínimo global. A complexidade combinatória desse tipo de solução a torna intratável. Por esta razão, métodos de aglomerados recaem em heurísticas para encontrarmos a solução.

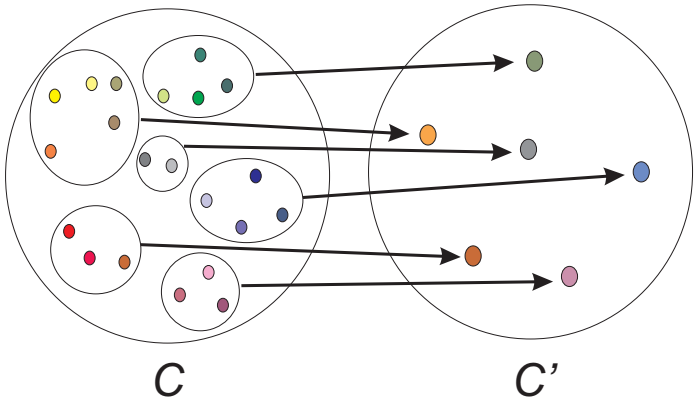


Figura 2.10: Aglomerados de cores sendo substituídos por sua cor representativa.

A Figura 2.10 nos mostra aglomerados de cores no espaço de cor C que são substituídos pelos seus respectivos níveis de quantização no espaço de cor quantizado C'.

2.6 Estratégias de Quantização

Pelo que vimos até agora, um método de quantização consiste basicamente de duas etapas:

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

Assim, a função de quantização q é definida de modo a associar às cores de uma célula de quantização, ao nível de quantização correspondente. A função q é portanto constante em cada célula de quantização. Um vez conhecida a função de mapeamento q, o processo de quantização se torna simples: para cada cor c do pixel da imagem, verifica-se a qual célula de quantização ela pertence e então substitui-se a cor c deste pixel pela cor $q(c)$ correspondente ao nível de quantização desta célula.

Todos os métodos de quantização existentes refletem a descrição acima, e funcionam de três modos distintos.

Quantização por Seleção Direta Os métodos de seleção direta escolhem, inicialmente, com base em propriedades estatísticas da imagem, os níveis de quantização a serem utilizados e a partir daí determinam a função de quantização de forma a minimizar o erro de quantização.

Quantização por Subdivisão Espacial Neste modelo, primeiro determinamos as células de quantização para em seguida calcular o nível de quantização associado a cada célula.

Temos dois tipos de quantização por subdivisão espacial: quantização uniforme e quantização adaptativa.

Quantização Híbrida Estes métodos misturam as duas estratégias de quantização, hora partindo do nível de quantização para encontrar suas células de quantização e hora fazendo o inverso, tudo isso em um mesmo algoritmo.

Um exemplo desse tipo de estratégia seria utilizarmos um método de quantização por subdivisão espacial para estimarmos uma solução inicial que será usado em um método de aglomeramento.

Todos os métodos de quantização podem ser considerados como um método de otimização pois todos eles tentam minimizar o erro de quantização na imagem quantizada. Assim, podemos quantizar tanto utilizando técnicas de otimização, como por exemplo a minimização de um funcional, como utilizando técnicas de aglomeramento.

2.7 Imagem de Teste

Figura 2.11: Imagem digital colorida quantizada com 24 bits: araras.

Usaremos a imagem das araras mostrada na Figura 2.11 para compararmos os vários métodos de quantização que apresentaremos. Esta é uma fotografia digitalizada que foi inicialmente quantizada para 24 bits (8 bits por canal) não tendo assim contornos de quantização perceptíveis.

Capítulo 3

Quantização por Seleção Direta

Como vimos anteriormente, os métodos de quantização que se utilizam de técnicas de seleção direta são aqueles em que primeiro escolhemos os níveis de quantização para depois, a partir desses níveis, gerarmos as células de quantização.

Este tipo de estratégia não é muito utilizada para desenvolvermos algoritmos de quantização pois os resultados obtidos não são muito bons, dependendo muito da distribuição de probabilidade de cores na imagem.

O método por seleção direta mais simples seria escolher K níveis de quantização aleatoriamente e depois, a partir destes níveis escolhidos, obtermos suas respectivas células de quantização. Como podemos observar, este método não é nada bom visto que não levamos em consideração nenhuma informação sobre a distribuição de probabilidade de cores na imagem. Com este método podemos obter imagens quantizadas que são muito distantes das imagens originais.

O Algoritmo da Populosidade é o mais famoso método que se utiliza da técnica de seleção direta. Na próxima seção deste capítulo falaremos um pouco mais sobre este algoritmo.

3.1 Algoritmo da Populosidade

O algoritmo da populosidade foi desenvolvido independentemente por dois grupos distintos em 1978: Tom Boyle e Andy Lippman no Architechture Machine Group do MIT e Ephraim Cohen do New York Institute of Technology. As idéias de Boyle e Lippman foram implementadas por Paul Heckbert em seu trabalho de final de graduação no MIT (Heckbert, 1980).

Este algoritmo assume que o mapamento de quantização pode ser gerado encontrando-se as regiões mais densamente povoadas na distribuição de cores da imagem original. O algoritmo da populosidade simplesmente seleciona as K cores de maior frequência do histograma, usando-as como os níveis de quantização do mapeamento. Com os níveis de quantização já selecionados encontramos suas respectivas células de quantização.

O algoritmo da populosidade funciona bem para muitas imagens, mas seu desempenho é ruim em imagens com um grande número de cores, ou quando se deseja quantizar para um número pequeno de cores (menos de 50 cores). Geralmente omite cores em regiões pouco

populosas do espaço de cor (um "highlight"em uma imagem pode desaparecer completamente nesse processo de quantização, uma vez que ele ocupa apenas um pequeno número de pixels na imagem). Um aperfeiçoamento desse algoritmo, proposto por (Heckbert, 1980) consiste em atribuírmos pesos maiores às regiões onde queremos enfatizar nas imagens. Por exemplo, se quizermos que um "highlight"esteja presente na imagem quantizada, devemos atribuir um alto peso para esta região, desta forma, quando gerarmos o histograma de cor da imagem, as cores presentes no "highlight"aparecerão como se fossem bastante frequentes na imagem, sendo escolhida como um dos K níveis de quantização. Esta solução tem o problema de necessitar de intervenção de alguém no processo, ou seja, alguém deve marcar as regiões onde se deseja dar ênfase. Logo, esta alternativa não é automática.

A Figura 3.1 (esquerda) mostra a imagem das araras quantizada para 8 bits com o algoritmo da populosidade. Na Figura 3.1 (direita) mostramos a mesma imagem quantizada para 4 bits com o mesmo algoritmo. Observando as duas imagens quantizadas podemos ver que, para poucas cores, a imagem quantizada se apresenta muito distante da imagem original.

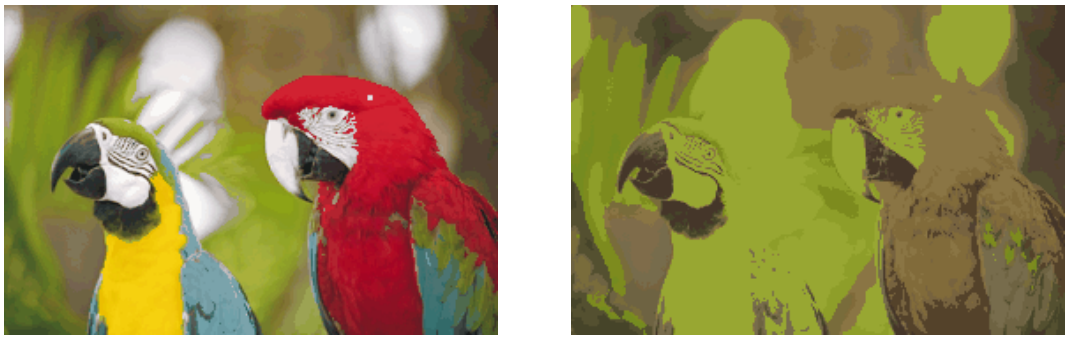


Figura 3.1: Algoritmo da Populosidade: (esquerda) quantização para 8 bits, (direita) quantização para 4 bits.

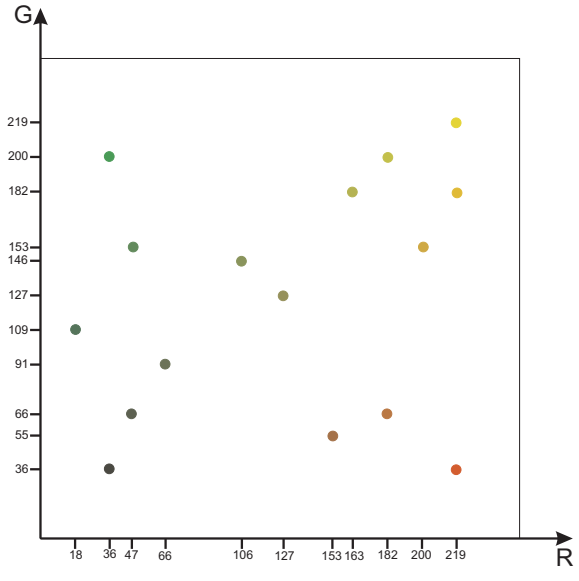
Capítulo 4

Quantização por Subdivisão Espacial

Esta estratégia de quantização de imagens consiste em subdividir o espaço de cor em células de quantização e, a partir de cada célula, encontrar seu nível de quantização. Muitos algoritmos de quantização de imagens se utilizam desta técnica.

Alguns algoritmos são bastante intuitivos, somente dividindo o espaço de cor sem levar em consideração informações fornecidas pela distribuição de cores na imagem. Outros tentam minimizar o erro de quantização, usando para isso várias metodologias.

Neste capítulo apresentaremos os principais algoritmos de quantização por subdivisão espacial. Utilizaremos a distribuição de cores no espaço bidimensional (plano RG) mostrada na Figura 4.1 para ilustrar os algoritmos de quantização que serão estudados.



	R, G	F_i
	18, 109	3
	36, 36	4
	36, 200	1
	47, 66	2
	47, 153	1
	66, 91	2
	106, 146	3
	127, 127	2
	153, 55	2
	163, 182	2
	182, 66	3
	182, 200	5
	200, 153	4
	219, 36	3
	219, 182	2
	219, 219	3

Figura 4.1: Distribuição de cores em um espaço de cor bidimensional (plano RG).

4.1 Quantização Escalar Uniforme

A forma mais intuitiva de se fazer uma subdivisão do espaço de cor consiste em dividí-lo em células congruentes tomando como valor de quantização seu centro. A este método damos o nome de *quantização escalar uniforme*.

Este método não nos fornece bons resultados, podendo ter até células de quantização onde não teremos nenhuma cor presente no gamute da imagem original. Levando em consideração a forma de como o olho humano percebe as cores, podemos tirar partido disso para gerar imagens quantizadas um pouco melhores. Assim, podemos privilegiar a componente G visto que o olho humano é mais sensível às variações de verde. Podemos também fazer menos subdivisões na componente B pois o olho humano é menos sensível a variações de azul.

Assim, o método de quantização escalar uniforme pode ser resumido como se segue:

1. Escolha quantas subdivisões se deseja fazer em cada uma das componentes RGB de cor de forma que o produto do número de subdivisões em cada uma das componentes seja igual ao número total de níveis de quantização que se deseja na imagem quantizada.

2. Divida cada um dos eixos R, G e B igualmente de acordo com o número de subdivisões escolhido para cada componente de cor RGB.

3. Faça o produto cartesiano dessas subdivisões em todas as componentes gerando as células de quantização.

4. Tome o centro de cada célula como sendo o seu nível de quantização.

5. Mapeie cada cor em seu nível de quantização mais próximo.

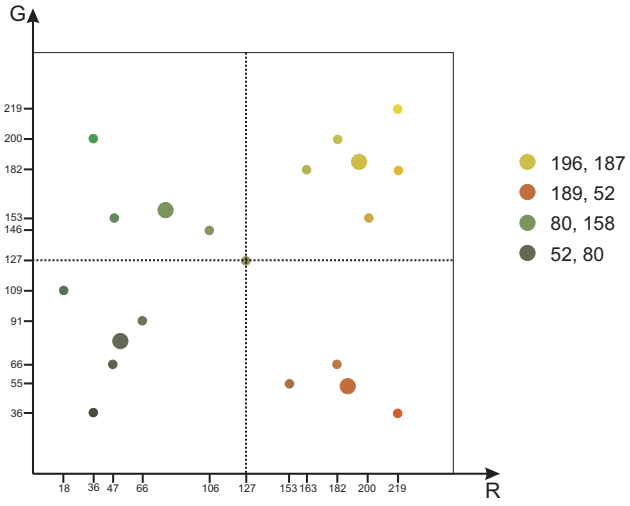


Figura 4.2: Quantização escalar uniforme para 4 níveis no plano RG.

Na Figura 4.2 temos o resultado de uma quantização escalar uniforme para 4 níveis no plano RG. Escolhemos dividir cada componente de cor 2 vezes. O nível de quantização de cada célula é representado pelo círculo maior.

Figura 4.3: Quantização Uniforme: (esquerda) quantização para 8 bits, (direita) quantização para 4 bits.

A Figura 4.3 (esquerda) mostra a imagem das araras quantizada para 8 bits usando uma quantização uniforme com 8 subdivisões em R, 8 subdivisões em G e 4 subdivisões em B. Na Figura 4.3 (direita) mostramos a mesma imagem quantizada para 4 bits usando 2 subdivisões em R, 4 subdivisões em G e 2 subdivisões em B.

4.2 Algoritmo do Corte Mediano

Introduzido na literatura em 1982 por Paul Heckbert como uma alternativa ao algoritmo da Populosidade, este é o algoritmo de quantização mais conhecido da comunidade de computação gráfica. Além de apresentar bons resultados perceptuais, é de fácil implementação e eficiente computacionalmente. Existem inúmeras variações do algoritmo original proposto por (Heckbert, 1980).

O algoritmo do corte mediano subdivide repetidamente o cubo de cor RGB em paralelepípedos cada vez menores. Esta divisão é feita pela mediana da componente RGB de maior comprimento. Para explicarmos este algoritmo primeiro devemos introduzir o conceito de *mediana de um conjunto*.

4.2.1 Mediana de um Conjunto

Dado um conjunto finito e ordenado de pontos do espaço

$$C = \{c_1 \leq c_2 \leq \ldots \leq c_{n-1} \leq c_n\},$$

a *mediana* m_C desse conjunto é definida pelo elemento do meio $c_{(n+1)/2}$ se n é ímpar, e pela média dos dois elementos intermediários, se n é par. A mediana é uma medida de

localização estatística que divide o conjunto de dados C em duas partes com igual número de elementos. Observe que, ao contrário da média, a mediana não é influenciada pela magnitude dos elementos do conjunto. Um fato importante a ser mencionado é que no cálculo da mediana devemos levar em consideração a frequência de cada elemento c_i do conjunto C. Desse modo, a construção do histograma de frequência associado ao conjunto de dados é uma etapa importante no cálculo da mediana.

O algoritmo do corte mediano procura equalizar o histograma de uma imagem, ou seja, cada cor presente na imagem terá um número de pixels aproximadamente iguais. O ideal seria uma imagem onde todas as cores ocorrecem em um mesmo número de pixels. O processo de quantização por equalização de histograma para imagens monocromáticas, consiste em se fazer subdivisões sucessivas do intervalo de intensidade da imagem utilizando a mediana do conjunto de intensidade em cada processo de subdivisão.

Estendendo-se essa idéia descrita acima para imagens coloridas temos a base do *algoritmo do corte mediano*. De forma simples, o método consiste em utilizar o algoritmo de quantização por equalização de histograma em cada uma das componentes de cor do gamute da imagem. Descreveremos detalhadamente o algoritmo para o espaço de cor RGB.

4.2.2 O Algoritmo

Indicamos por K o número de níveis de quantização desejado.

Devemos inicialmente encontrar o mínimo e o máximo valor de cada componente de cor. Chamaremos r_{min}, g_{min} e b_{min} aos valores dos mínimos das componentes R (vermelho), G (verde) e B (azul) respectivamente. Os valores dos máximos das componentes R, G e B são dados por r_{max}, g_{max} e b_{max} respectivamente. Estes valores de mínimo e máximo de R, G e B nos fornecem um paralelepípedo

$$V = \{[r_{min}, r_{max}] \times [g_{min}, g_{max}] \times [b_{min}, b_{max}]\},$$

de volume mínimo que contém todas as cores no gamute da imagem a ser quantizada. Para a distribuição de cor bidimensional mostrada na Figura 4.1, o menor quadrilátero (pois só temos 2 dimensões) de cor que contém todas as cores presentes nesta distribuição pode ser visto na Figura 4.4. Em seguida, tomamos a componente do espaço de cor em cuja direção o paralelepípedo V possui a aresta de maior comprimento. No caso da nossa distribuição bidimensional, a aresta de maior comprimento é a componente vermelha r. Vamos supor, para o caso de uma cor no RGB, que essa aresta também é a componente vermelha r. Fazemos então uma ordenação das cores no gamute da imagem pela componente r, e calculamos a mediana m_r do conjunto de cores com base nessa ordenação. Dividimos assim a região V em duas sub-regiões

$$V_1 = \{(r, g, b) \in C; r \leq m_r\},$$

e

$$V_2 = \{(r, g, b) \in C; r > m_r\}.$$

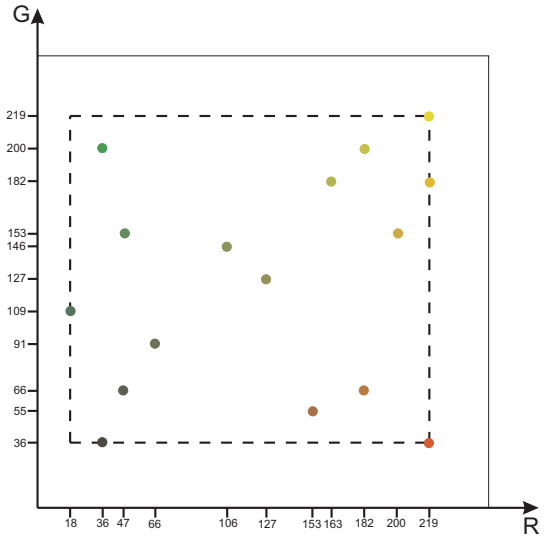


Figura 4.4: Menor quadrilátero que contém todas as cores presentes na distribuição de cor.

Aplicamos então o método de subdivisão a cada uma das regiões V_1 e V_2. Continuamos o processo de subdivisão, recursivamente, até que uma das duas condições seguintes sejam satisfeitas: as duas sub-regiões V_1 e V_2 obtidas não contém cores do gamute da imagem, ou o número desejado, K, de células de quantização seja obtido.

Após a subdivisão do espaço de cor no número de células desejado, determinamos o nível de quantização de cada célula. Esse nível de quantização é dado pela média das cores contidas em cada uma dessas células. Para se obter o valor de quantização de um pixel da imagem, devemos localizar a célula que contém a cor desse pixel e fazer a quantização para o nível correspondente a essa célula.

O Algoritmo do corte mediano aplicado na distribuição de cores no plano RG mostrada na Figura 4.1 para uma quantização para 4 níveis pode ser visto na Figura 4.5. O nível de quantização de cada célula é representado pelo círculo maior.

A Figura 4.6 (esquerda) mostra a imagem das araras quantizada para 8 bits com o algoritmo do corte mediano. Na Figura 4.6 (direita) mostramos a mesma imagem quantizada para 4 bits com o mesmo algoritmo.

4.3 Algoritmo da Divisão pela Média

Baseado no algoritmo do corte mediano, o *algoritmo da divisão pela média* proposto por Wu e Witten (Wu & Witten, 1985) em 1985 faz as subdivisões espaciais baseados na média, ou invés da mediana. Suas características principais estão resumidas abaixo:

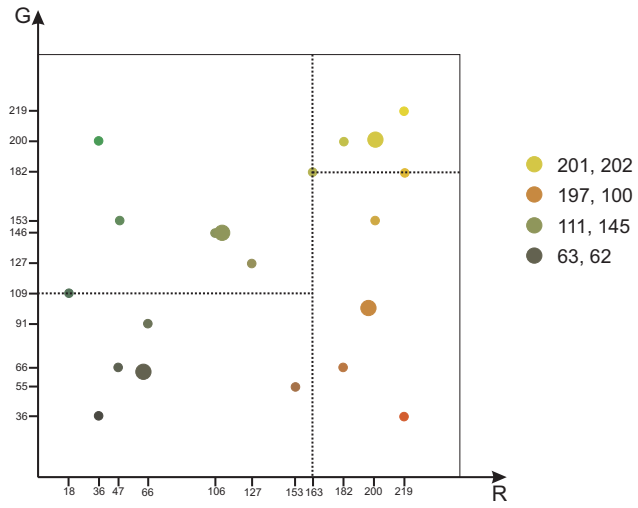


Figura 4.5: Algoritmo do corte mediano para 4 níveis no plano RG.

Figura 4.6: Algoritmo do Corte Mediano: (esquerda) quantização para 8 bits, (direita) quantização para 4 bits.

- Particionamos os paralelepípedos de cor por planos perpendiculares à componente de cor de maior comprimento, como no algoritmo do corte mediano.
- O ponto de divisão é escolhido como sendo a média (ou a média ponderada pela freqüência de ocorrência de cada cor) da componente de cor de maior comprimento.
- Um tamanho mínimo é pré-estabelecido para o paralelepípedo de cor. Todo paralelepípedo menor que este tamanho não será mais subdividido.

Assim, o algoritmo básico é o mesmo proposto por Heckbert (Heckbert, 1980), mudando somente no critério de subdivisão do espaço de cor: neste algoritmo subdividimos o espaço de cor pela média da componente de cor de maior comprimento.

Na Figura 4.7 temos o resultado da quantização para 4 níveis da distribuição de cor no plano RG mostrado na Figura 4.1. Para esta quantização usamos a média da componente de

maior comprimento como critério de subdivisão do espaço de cor. A mesma quantização só que usando a média ponderada pela freqüência de ocorrência de cada cor na imagem como critério de subdivisão do espaço de cor pode ser vista na Figura 4.8. Representamos o nível de quantização de cada célula pelo círculo maior.

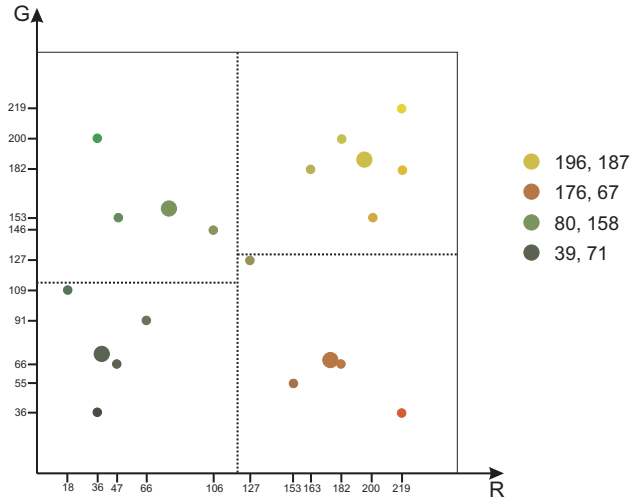


Figura 4.7: Algoritmo da divisão pela média para 4 níveis no plano RG.

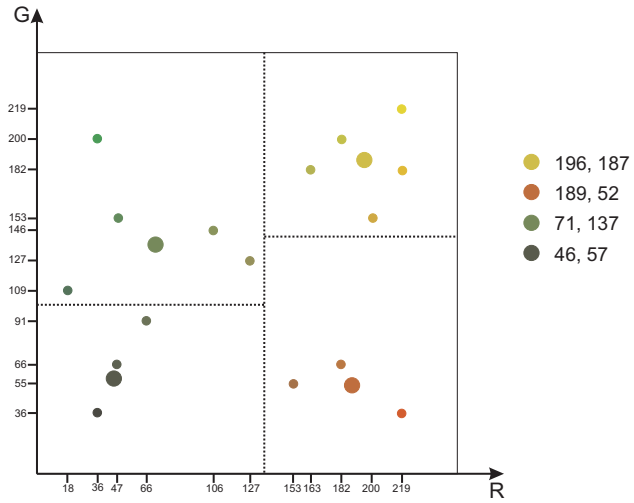


Figura 4.8: Algoritmo da divisão pela média ponderada pela freqüência de ocorrência de cada cor para 4 níveis no plano RG.

O que é melhor, dividir pela mediana ou pela média? A divisão pela mediana nos garante que toda célula de quantização terá o mesmo número de elementos, isso é chamado de equalização de histograma pois todas as cores presentes na imagem quantizada terão o mesmo

número de ocorrências. A divisão pela média não garante uma equalização do histograma. A Figura 4.9 (A) mostra o histograma de uma imagem em tons de cinza. Esta imagem será quantizada para 4 cores e o resultado desta quantização pode ser visto nas Figuras 4.9 (B), onde usamos um algoritmo de equalização de histograma, ou seja, divisão pela mediana; Figuras 4.9 (C), onde utilizamos um algoritmo de divisão pela média; e Figuras 4.9 (D), onde foi usado um algoritmo de divisão pela média ponderada pela freqüência de ocorrência de cada cor na imagem.

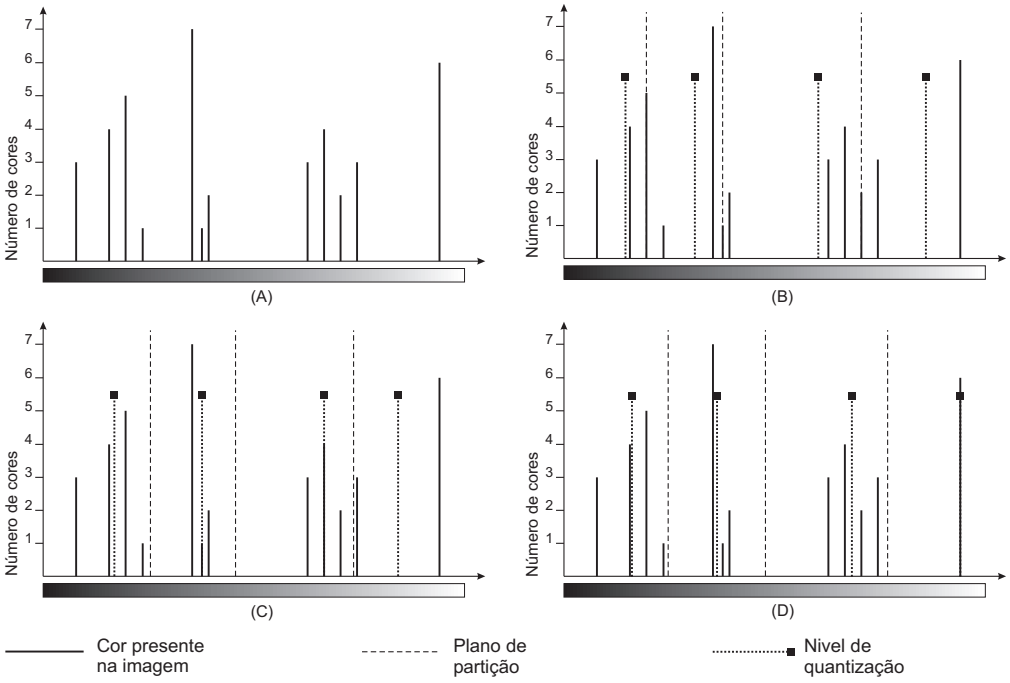


Figura 4.9: Comparação dos algoritmos de divisão pela mediana e pela média.

Algoritmo	Erro de quantização
Mediana	1860.0
Média	867.0
Média ponderada	301.5

Tabela 4.1: Erros de quantização gerados pela subdivisão do espaço pela mediana e pela média.

Como podemos observar na Figura 4.9, o erro de quantização gerado pelo algoritmo de divisão pela mediana é maior que o erro gerado pela divisão pela média. Na Tabela 4.1 temos os erros de quantização para a quantização da Figura 4.9. Note que o algoritmo que apresentou menor erro de quantização foi o que se utiliza da divisão pela média ponderada pela freqüência de ocorrência de cada na imagem. Assim, apesar de equalizar o histograma de

uma imagem, o algoritmo de divisão pela mediana não garante um menor erro de quantização que a divisão pela média. Mas também não podemos garantir que se usarmos divisão pela média sempre obteremos melhores resultados que a divisão pela mediana, isso vai depender, em muito, da distribuição de cores na imagem.

4.4 Algoritmo da Divisão pela Variância

Existem dois objetivos fundamentais no desenvolvimento de um algoritmo de subdivisão espacial: a cada passo do processo de particionamento devemos decidir que paralelepípedo de cor deve ser particionado e ao mesmo tempo escolher um plano de partição apropriado para subdividir o paralelepípedo. No *algoritmo da divisão pela variância*, introduzido na literatura em 1990 por S. J. Wan, S. K. M. Wong e P. Prusinkiewicz (Wan *et al.* , 1990), ambas as decisões são tomadas no sentido de minimizar o erro de quantização.

O sistema visual humano não é capaz de determinar o valor absoluto de uma cor. Ele é mais sensível a variações de cores. Assim, minizando o erro de quantização garantimos que o processo de quantização representa corretamente as cores da imagem original, porém não garantimos que a variação das cores na imagem quantizada seja a mesma, ou próxima, da variação das cores na imagem original. Um algoritmo de quantização que produz valores pequenos para a variância introduz quase a mesma distorção de cor para cada pixel. Logo, a minimização da variância nos ajuda a preservar as variações das cores na imagem quantizada.

Assim, com o intuito de reduzir o erro de quantização total e também a variância a cada passo da subdivisão, o algoritmo da divisão pela variância particiona o paralelepípedo de cor de maior variância. Isto é, o paralelepípedo com o maior erro de quantização é escolhido para ser subdividido. O idéia básica por trás desse algoritmo é de adaptativamente atribuir mais células à regiões com erros de quantização maiores de tal modo que o erro de quantização total seja reduzido.

4.4.1 Como Escolher o Paralelepípedo que Será Particionado

Como no algoritmo do corte mediano, devemos inicialmente encontrar o mínimo e o máximo valor de cada componente de cor. Chamaremos r_{min}, g_{min} e b_{min} aos valores dos mínimos das componentes R, G e B respectivamente. Os valores dos máximos das componentes R, G e B são dados por r_{max}, g_{max} e b_{max} respectivamente. Estes valores de mínimo e máximo de R, G e B nos fornecem um paralelepípedo

$$V = \{[r_{min}, r_{max}] \times [g_{min}, g_{max}] \times [b_{min}, b_{max}]\},$$

de volume mínimo que contém todas as cores no gamute da imagem a ser quantizada. A freqüência de ocorrência de cada cor c_i é dada por $F_i = F(c_i)$. Dado um paralelepípedo $V_l \subseteq V$, seu centróide (média) μ_l e sua variância σ_l^2 são definidos por

$$\mu_l = \sum_{c_i \in V_l} \frac{F_i}{W_l} c_i, \tag{4.1}$$

$$\sigma_l^2 = \sum_{c_i \in V_l} \|c_i \Leftrightarrow \mu_l\|^2 \frac{F_i}{W_l}, \tag{4.2}$$

onde $W_l = \sum_{c_i \in V_l} F_i \leq 1$ que chamaremos de peso do paralelepípedo V_l.

O erro de quantização atribuído ao paralelepípedo V_l é determinado pela sua *variância ponderada* $\hat{\sigma}_l^2$, que nada mais é do que o erro introduzido ao substituirmos cada cor contida em V_l pelo seu nível de quantização, que é dado por μ_l. Assim, temos que a variância ponderada é dada por

$$\hat{\sigma}_l^2 = W_l \sigma_l^2 = \sum_{c_i \in V_l} \|c_i \Leftrightarrow \mu_l\|^2 F_i. \tag{4.3}$$

Dessa forma, a cada passo de subdivisão, o paralelepípedo de cor que deve ser escolhido para ser subdivido é aquele com a maior variância ponderada. Isso nos garante uma redução no erro de quantização.

4.4.2 Como Escolher o Plano de Partição

Suponha que o paralelepípedo V_l é dividido em dois paralelepípedos menores V_{l1} e V_{l2} pelo plano de partição , perpendicular a um dos eixos coordenados, ou seja, aos eixos R, G ou B. O erro de quantização é dado pela soma das variâncias ponderada pela freqüência de ocorrência de cada cor contida em cada paralelepípedo

$$E(,\ ) = W_{l1}\sigma_{l1}^2 + W_{l2}\sigma_{l2}^2 = \sum_{c_i \in V_{l1}} \|c_i \Leftrightarrow \mu_{l1}\|^2 p(c_i) + \sum_{c_i \in V_{l2}} \|c_i \Leftrightarrow \mu_{l2}\|^2 p(c_i), \tag{4.4}$$

onde W_{lj}, μ_{lj} e σ_{lj}^2 são o peso, a média, e a variância do j-ésimo paralelepípedo de cor $(j = 1, 2)$. O plano de partição ótimo , $_{opt}$ é definido como sendo o plano que minimiza o erro de quantização

$$,\ _{opt} = arg \quad \underset{\{\Gamma\}}{Min} \quad E(,\ ), \tag{4.5}$$

onde $\{,\ \}$ é o conjunto de todos os possíveis planos de partição perpendiculares ao eixo coordenado.

Na prática, encontrar o plano de partição ótimo definido pela equação 4.5 é uma tarefa que consume muito tempo para ser computada. Um modo de simplificar este cálculo é considerarmos as distribuições projetadas nos eixos coordenados.

Seja μ a média e σ^2 a variância de uma distribuição unidimensional, ou seja, a variância e a média de cada componente de cor RGB. Seja t um ponto de corte que divide a distribuição em dois intevalos. O ponto de corte ótimo t_{opt} é definido como sendo o que maximiza a redução das variâncias

$$t_{opt} = arg \quad \underset{t}{Max} \quad [\sigma^2 \Leftrightarrow (w_1\sigma_1^2(t) + w_2\sigma_2^2(t))], \tag{4.6}$$

onde w_j e $\sigma_j^2(t)$ são o peso e a variância do j-ésimo intervalo ($j = 1, 2$). Pode ser verificado facilmente que a redução esperada da variância pode ser calculada pela fórmula simplificada

$$\sigma^2 \Leftrightarrow (w_1\sigma_1^2(t) + w_2\sigma_2^2(t)) = \frac{w_1}{w_2}[\mu \Leftrightarrow \mu_1(t)]^2, \tag{4.7}$$

onde $\mu_1(t)$ é a média do primeiro intervalo. Além disso, temos que o ponto de corte ótimo é dado por

$$t_{opt} = arg \quad \underset{(\mu+lower)/2 \quad \leq \quad t \quad \leq \quad (\mu+upper)/2}{Max} \quad \left[\frac{w_1}{w_2}[\mu \Leftrightarrow \mu_1(t)]^2\right], \tag{4.8}$$

onde os símbolos *lower* e *upper* especificam os limites do intervalo. Baseado na equação 4.8, o ponto de corte ótimo e a redução esperada da variância podem ser calculados ao mesmo tempo. Assim, somente as componentes de cores dentro da faixa de variação $(\mu_{r,g,b} + lower_{r,g,b})/2 \leq r, g, b \leq (\mu_{r,g,b} + upper_{r,g,b})/2$ são candidatas a ponto de corte ótimo.

Logo, o plano de partição pode ser escolhido da seguinte maneira. Dado um paralelepípedo de cor, calcule as distribuições unidimensionais relativas às componetes R, G e B projetando todas as cores dentro do paralelepípedo em cada eixo coordenado. Para cada distribuição projetada, calcule o ponto de corte ótimo e a redução esperada na variância usando a equação 4.8. Escolhemos o plano de partição perpendicular ao eixo ao longo do qual a redução esperada na variância é a maior sobre todas as distribuições projetadas. O plano de partição passa pelo ponto de corte ótimo deste eixo.

4.4.3 Como Formar as Células de Quantização

O processo de partição acima é repetido até que o número de paralelepípedos de cor seja igual ao número de cores que desejamos na imagem quantizada. Os níveis de quantização são escolhidos como sendo os centróides dos paralelepípedos de cor resultantes. Todas as cores contidas em uma determinada célula de quantização são mapeadas no seu nível de quantização.

4.4.4 O Algoritmo

O algoritmo da divisão pela variância pode ser resumido da seguinte maneira:

1. Escolha o paralelepípedo com a maior variância ponderada (veja equação 4.3) para ser particionado.

2. Projete todas as cores contidas no paralelepípedo em cada um dos eixos coordenados R, G e B. Para cada uma das distribuições projetadas, calcule o ponto de corte ótimo e a redução esperada da variância pela equação 4.8.

3. Particione esse paralelepípedo pelo plano perpendicular ao eixo ao longo do qual a redução esperada da variância é maior. Esse plano de partição intercepta esse eixo no ponto de corte ótimo.

4. Calcule a variância ponderada para cada um dos dois paralelepípedos menores.

5. Repita os passos 1 a 4 até que o número de paralelepípedos alcance o número de células de quantização desejadas.

6. Calcule os centróides dos paralelepípedos resultantes, que formam os níveis de quantização desejados.

7. Mapeie cada cor contida em uma determinada célula em seu respectivo nível de quantização.

4.4.5 Exemplo de Execução 2-D do Algoritmo

Vamos mostrar uma execução bidimensional deste algoritmo usando a distribuição de cor mostrada na Figura 4.1. Inicialmente só temos um quadrilátero (pois estamos no caso bidimensional), onde a componente r varia na faixa $18 \leq r \leq 219$ e a componente g varia na faixa $36 \leq r \leq 219$. Devemos encontrar o ponto de corte ótimo em cada uma das componentes r e g. Mas não precisamos fazer esta busca em todas as componentes projetadas das cores, somente precisamos verificar as cores contidas nos intervalos $(\mu_r + lower_r)/2 \leq r \leq (\mu_r + upper_r)/2$ e $(\mu_g + lower_g)/2 \leq g \leq (\mu_g + upper_g)/2$. Para este quadrilátero inicial, temos que verificar as cores nos intervalos: $77 \leq r \leq 177.5$ e $80.5 \leq g \leq 172$. Assim, no eixo R devemos calcular a redução esperada na variância para as cores com componentes r iguais à 106, 127, 153 e 163. E no eixo G, verificaremos para as cores com componentes g iguais à 91, 109, 127, 146 e 153. Ao fazermos estes cálculos, encontramos que a maior redução na variância ocorre para $r = 106$. Na Figura 4.10 vemos a primeira subdivisão do plano RG no ponto de corte ótimo $r = 106$.

Calculamos a variância ponderada em cada um dos novos quadriláteros, escolhendo o de maior variância poderada para ser subdividido. O quadrilátero escolhido foi o da direita, que apresenta a maior variância poderada. O ponto de corte ótimo encontrado para este quadrilátero foi $g = 127$. Na Figura 4.11 vemos o resultado da segunda iteração deste algoritmo.

Devemos calcular a variância ponderada para estes dois novos quadriláteros e escolher o quadrilátero de maior variância ponderada para ser subdividido. Escolhemos então o quadrilátero esquerdo resultante da primeira subdivisão, pois este tem variância ponderada maior

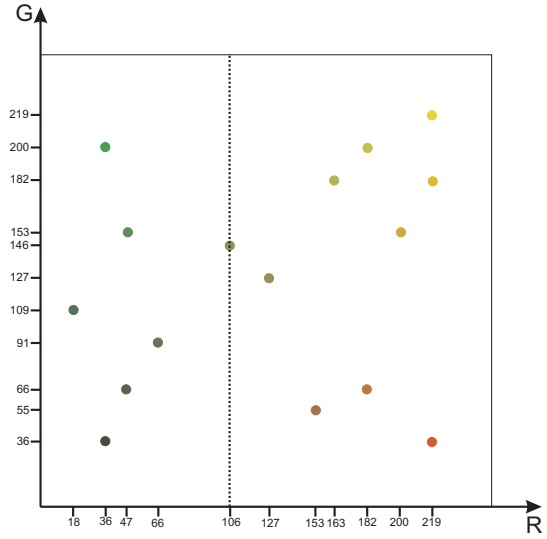


Figura 4.10: Algoritmo da divisão pela variância: primeira subdivisão espacial.

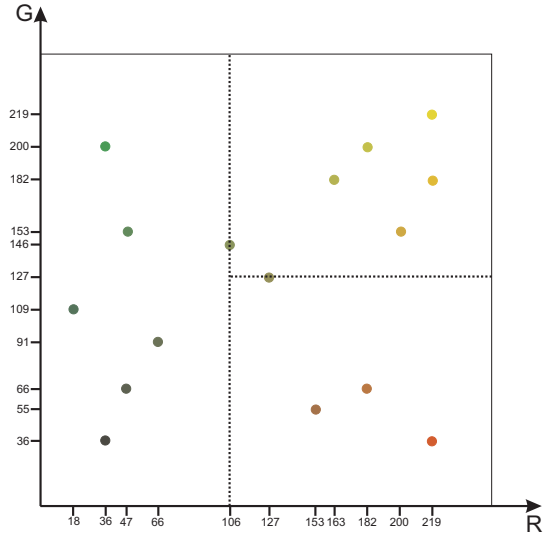


Figura 4.11: Algoritmo da divisão pela variância: segunda subdivisão espacial.

que estes dois novos quadriláteros recém-gerados. O ponte de corte ótimo para esta subdivisão está em $g = 91$. Na Figura 4.12 vemos o resultado da quantização para 4 níveis desta distribuição de cores. Os círculos maiores representam os níveis de quantização de cada célula.

Aplicando este algoritmo para quantizar a imagem da arara temos na Figura 4.13 (esquerda) uma quantização para 8 bits. Na Figura 4.13 (direita) mostramos a mesma imagem quantizada para 4 bits utilizando o algoritmo da divisão pela variância.

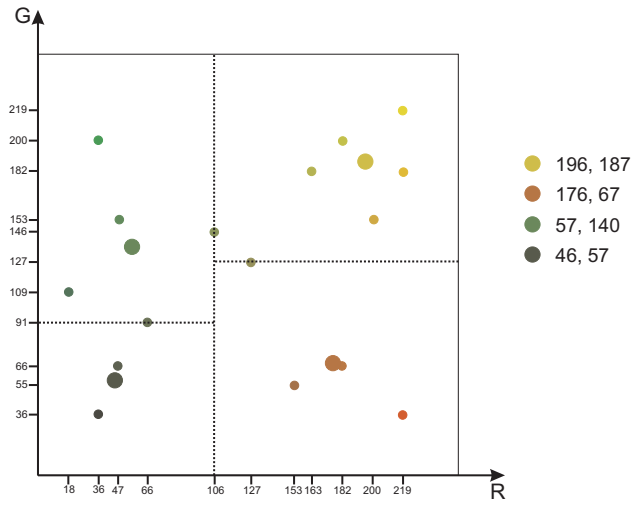


Figura 4.12: Algoritmo da divisão pela variância: quantização para 4 níveis.

Figura 4.13: Algoritmo da Divisão pela Variância: (esquerda) quantização para 8 bits, (direita) quantização para 4 bits.

4.5 Algoritmo da Divisão Binária

Este algoritmo foi introduzido na literatura por Michael T. Orchard e Charles A. Bouman (Bouman & Orchard, 1991) em 1991. Usando uma abordagem hierárquica, este algoritmo tenta produzir um mapa de cores que minimiza um critério objetivo de erro de quantização.

O objetivo deste algoritmo é particionar C, o conjunto de todos os pixels da imagem, em K conjuntos disjuntos ou aglomerados onde K é o número de cores que se deseja na imagem quantizada. O algoritmo obriga que a partição de C tenha a estrutura de uma árvore binária. Cada nó da árvore representa um subconjunto de C, e os filhos de qualquer nó particiona os membros dos nós pais em dois conjuntos. O conjunto dos pixels da imagem correspondentes ao nó n é chamado de C_n. Para ilustrar as operações do algoritmo, vamos numerar os nós de modo que a raiz seja 1 e os filhos do nó n são $2n$ e $2n + 1$.

O método de geração da árvore binária é especificado pelo número de folhas, K, o método

de divisão de um nó em seus dois filhos, e a ordem na qual os nós são divididos. Os métodos usados para a divisão dos nós e determinação de quais nós serão divididos tentam minimizar o erro de quantização, que é dado por

$$E(c_i) = \sum_{todas \quad as \quad folhas \quad n} \sum_{i \in C_n} \| c_i \Leftrightarrow q_n \|^2, \tag{4.9}$$

onde q_n é o nível de quantização usado para representar as cores no conjunto C_n. Propriedades estastísticas de segunda ordem são usadas para determinar a ordem e a maneira na qual os nós serão divididos. As três estatísticas do aglomerado necessárias são

$$\begin{aligned} R_n &= \sum_{i \in C_n} c_i c_i^t \\ m_n &= \sum_{i \in C_n} c_i \\ N_n &= |C_n| \end{aligned} \tag{4.10}$$

onde R_n é uma matriz 3×3 e m_n é um vetor de 3 componentes. Como o valor médio do aglomerado é o ponto no qual o desvio padrão quadrado é mínimo, o valor de quantização do aglomerado q_n é assumido como sendo igual a sua média

$$q_n = \frac{m_n}{N_n} \tag{4.11}$$

Também definimos a covariância do aglomerado como sendo

$$\tilde{R}_n = R_n \Leftrightarrow \frac{1}{N_m} m_n m_n^t. \tag{4.12}$$

A divisão de um nó em dois nós é equivalente à escolher de dois novos níveis de quantização, q_{2n} e q_{2n+1}, e associar cada membro do aglomerado com o nível de quantização mais próximo. Isso, por sua vez, é equivalente a selecionar um plano que melhor divide o aglomerado de cor. Neste algoritmo, determinamos a direção na qual a variação do aglomerado é maior, e então dividimos o aglomerado com um plano que é perpendicular a essa direção e passa pela média do aglomerado, como pode ser visto na Figura 4.14 para o caso da distribuição de cor bidimensional mostrada na Figura 4.1. Para um aglomerado muito grande com distribuição Gaussiana pode ser mostrado que essa estratégia é ótima.

Mais especificamente, determinamos o vetor unitário que minimiza a expressão

$$\sum_{i \in C_n} ((c_i \Leftrightarrow q_n)^t e)^2 = e^t \tilde{R}_n e. \tag{4.13}$$

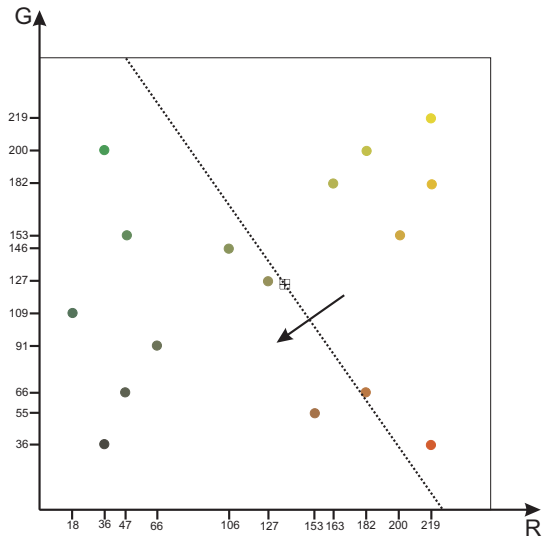


Figura 4.14: Divisão de um espaço de cor bidimensional pelo algoritmo de divisão binária: a seta indica a direção de variação máxima de cor e a linha pontilhada indica o plano de divisão.

Como $\tilde{R}$ é simétrico, a solução é o autovetor e_n correspondente ao autovalor maior ou principal λ_n de $\tilde{R}$. A variação quadrada total na direção e_n é conseqüentemente

$$\sum_{i \in C_n} ((c_i \Leftrightarrow q_n)^t e)^2 = \lambda_n. \tag{4.14}$$

Com o autovetor principal determinado, os pontos em C_n podem ser ordenados em dois conjuntos C_{2n} e C_{2n+1} da seguinte maneira:

$$\begin{aligned} C_{2n} &= \{c \in C_n : e_n^t \leq e_n^t q_n\} \\ C_{2n+1} &= \{c \in C_n : e_n^t > e_n^t q_n\}. \end{aligned} \tag{4.15}$$

Finalmente, as novas estatísticas para cada nó podem ser calculadas primeiramente calculando R_{2n}, m_{2n} e N_{2n}, e depois aplicando as relações

$$\begin{aligned} R_{2n+1} &= R_n \Leftrightarrow R_{2n} \\ m_{2n+1} &= m_n \Leftrightarrow m_{2n} \\ N_{2n+1} &= N_n \Leftrightarrow N_{2n}. \end{aligned} \tag{4.16}$$

A ordem na qual os nós são divididos é escolhida com o objetivo de produzir a maior redução no erro de quantização em cada iteração do algoritmo. Como esta estratégia não olha para resultados futuros para outras divisões, ela não é necessariamente ótima; entretanto,

podemos esperar que ela se comporte bem na maioria das situações práticas. Quando um nó é dividido sua variância é reduzida na direção do autovetor principal. Conseqüentemente, é razoável assumirmos que a redução no erro de quantização deve ser proporcional à variação quadrada total na direção do autovetor principal e_n de $\tilde{R}$. Por essa razão, o autovalor principal λ_n é usado como uma medida da redução esperada no erro de quantização se o nó n é dividido. Dada essa aproximação, a melhor alocação de um nível de quantização é dividir a folha pelo maior autovetor principal.

A algoritmo de quantização binária básico pode ser descrito como se segue:

1. Faça $C_1 = C$.
2. Calcule R_1, m_1, e N_1.
3. Repita a seguinte sequência $M \Leftrightarrow 1$ vezes:

   Encontre a folha n tal que λ_n seja o maior.

   Use (4.15) para formar os novos nós $2n$ e $2n + 1$.

   Calcule R, m, e N para os novos nós usando (4.16).

Nas Figuras 4.15 e 4.16 temos os planos de subdivisão do espaço bidimensional de cor gerado por este algoritmo para a distribuição de cores da Figura 4.1. As setas representam as direções de maior variação de cor e as linha pontilhadas os planos de divisão. A média de cada aglomerado está sendo mostrada na Figura 4.17.

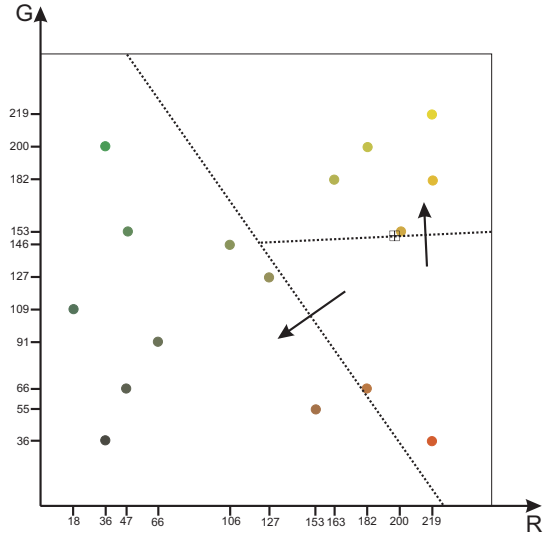


Figura 4.15: Algoritmo da divisão binária: segundo plano de divisão.

O resultado da quantização para 4 níveis pode ser visto na Figura 4.17.

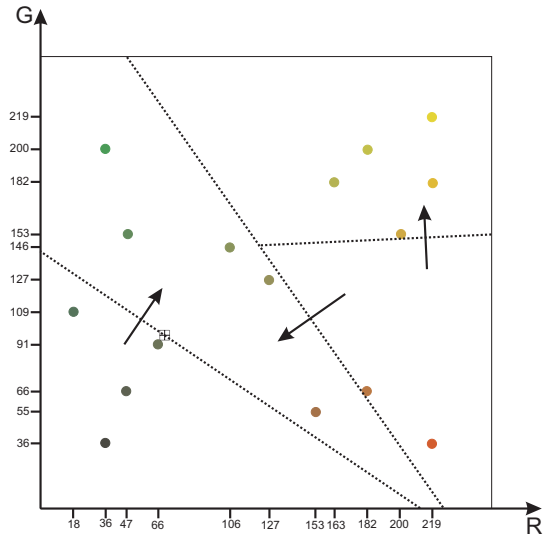


Figura 4.16: Algoritmo da divisão binária: terceiro plano de divisão.

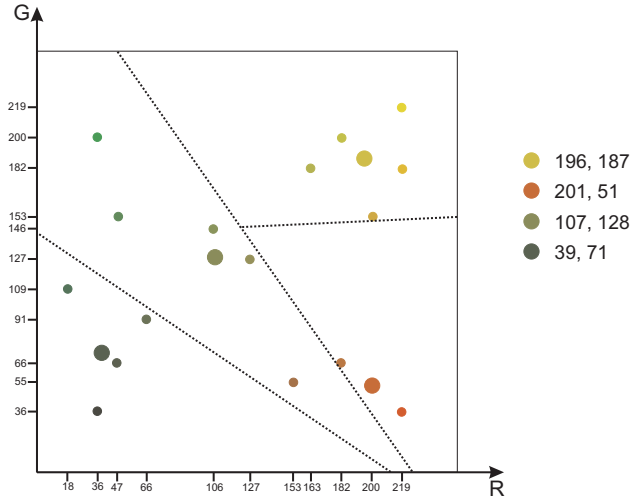


Figura 4.17: Algoritmo da divisão binária: quantização para 4 níveis

Capítulo 5

Quantização por Métodos de Otimização

Já vimos anteriormente que a quantização introduz um erro que pode ser estimado pela fórmula

$$E(c, q(c)) = \int_C p(c) d(c, q(c)) dc. \tag{5.1}$$

Se estamos realizando uma quantização de N níveis, isto é, particionando o espaço de cor em N células $K_1, \ldots, K_N$, com valores de quantização $q_1, \ldots, q_N$, a equação anterior fica da forma

$$E(c, q(c)) = \sum_{1 \leq j \leq N} \int_{K_j} p(c) d(c, q_j) dc. \tag{5.2}$$

Ou, como temos um número finito de cores em cada célula,

$$E(c, q(c)) = \sum_{1 \leq j \leq N} \sum_{c \in K_j} p(c) d(c, q_j). \tag{5.3}$$

Idealmente, entretanto, devemos tentar minimizar a quantidade (5.3) sobre todas as possíveis partições de N elementos do espaço de cor. O fato é que existe um número enorme de tais partições, o que significa que este problema é muito dificil de ser computado, logo a otimização completa geralmente não é possível de ser obtida. Os métodos de otimização existentes, entretanto, tendem a resolver somente uma forma restrita do problema, ou usam alguma heurística, ou retornam uma resposta aproximada. Vamos agora analisar alguns destes métodos.

5.1 Quantização Ótima Uni-dimensional

A noção de otimalidade depende das estatísticas da distribuição de cor na imagem e na medida de distorção que estamos usando. Suponha, no caso uni-dimensional, que a função de distribuição de probabilidade de cor é $p(c)$ e que estamos usando o erro médio quadrático como nossa medida de distorção. Assim temos que minimizar

$$D = \int_{-\infty}^{+\infty} p(c)(q(c) \Leftrightarrow c)^2 dc. \tag{5.4}$$

As células de quantização são intervalos $[c_{i-1}, c_i]$, cada um correspondendo a um valor de quantização q_i. Como $q(c) = q_i$ no intervalo $[c_{i-1}, c_i]$, a quantidade (5.4) que queremos minimizar pode ser reescrita como

$$D(c_0, \ldots, c_N, q_1, \ldots, q_{N-1}) = \sum_{i=1}^{N} \int_{c_{i-1}}^{c_i} p(c)(q_i \Leftrightarrow c)^2 dc. \tag{5.5}$$

As condições necessárias para o que erro de quantização global seja mínimo são obtidas calculando-se os pontos críticos da função D. Para fazermos isso, derivamos (5.5) com respeito a c_k e q_k e igualamos isso a zero. Obtemos

$$\frac{\partial D}{\partial c_k} = (c_k \Leftrightarrow q_{k-i})^2 p(c_k) \Leftrightarrow (c_k \Leftrightarrow q_k)^2 p(c_k) = 0, \tag{5.6}$$

$$\frac{\partial D}{\partial q_k} = 2 \int_{c_{i-1}}^{c_i} (c \Leftrightarrow q_k) p(c) dc = 0. \tag{5.7}$$

Resolvendo as duas equações, temos

$$q_k = \frac{\int_{c_{i-1}}^{c_i} c p(c) dc}{\int_{c_{i-1}}^{c_i} p(c) dc} \quad e$$

$$c_k = \frac{q_k + q_{k+1}}{2}, \tag{5.8}$$

onde $k = 1, \ldots, N$.

A primeira equação nos diz que o valor de quantização é o centróide da distribuição de probabilidade $p(c)$ no intervalo $[c_{k-1}, c_k]$. A segunda equação nos diz que os pontos de fronteira dos intervalos de quantização são dados pela média entre valores de quantização consecutivos. É fácil verificarmos que, quando a distribuição de probabilidade é uniforme, isto é, $p(c) = 1/(c_N \Leftrightarrow c_0)$, a solução ótima resulta na quantização uniforme uni-dimensional descrita anteriormente.

Note que uma solução de (5.8) é a priori somente um ponto crítico de D, e não necessariamente um mínimo. Podemos mostrar que, para algumas distribuições de probabilidade, a

solução realmente minimiza D. Esse é o caso, por exemplo, quando temos uma distribuição de probabilidade uniforme ou normal (gaussiana). Como mencionado anteriormente, a solução no caso de uma distribuição uniforme corresponde a uma quantização uniforme. Para uma distribuição normal a solução é mais complicada e deve ser calculada numericamente.

5.2 Quantização Ótima por Relaxação

Os cálculos da seção anterior estendem-se para dimensões maiores que 1, mas neste caso as equações em (5.8) são mais complicadas, porque envolvem integração sobre o contorno das células de quantização. Um método de quantização mais apropriado é um procedimento de relaxação, que nos fornece uma solução através de aproximações sucessivas.

Considere um vetor de cor c, que deve ser mapeado em um vetor c', e suponha que temos N níveis de quantização $c'_1, \ldots, c'_N$. Além disso, usando a métrica Euclideana quadrática $d(c_1, c_2) = \langle c_2 \Leftrightarrow c_1, c_2 \Leftrightarrow c_1 \rangle$ ($\langle a, b \rangle$ indica o produto escalar de a e b), a equação (5.1) implica que o erro médio quadrático é

$$D = \int_{-\infty}^{+\infty} d(c, c')p(c)dc = \sum_{i=1}^{N} \int_{c \in C_i} d(c, c')p(c)dc,$$

onde p é a distribuição de probabilidade e C_i é a célula de quantização correspondente ao nivel c'_i. Duas condições devem ser satisfeitas para obtermos uma quantização ótima:

1. O mapa de quantização q deve mapear cada cor de entrada para o nível de quantização mais próximo de si:

$$q(c) = c'_i \Leftrightarrow d(c, c'_i) \leq d(c, c'_j), \quad para \quad todo \quad 1 \leq j \leq N \quad com \quad j \neq i.$$

2. Em cada célula C_i devemos calcular o nivel de quantização c'_i de modo a minimizar o erro médio D em C_i.

A condição 1 nos permite calcular C_i pela métrica d e os níveis de quantização. Condição 2 nos permite calcular c'_i da célula C_i. Como na quantização ótima uni-dimensional, uma vez sabendo a métrica d, as células de quantização e os níveis de quantização são interdependentes; se soubermos um podemos determinar o outro:

- Comece com os níveis de quantização $c_i^{(1)}$, escolhidos de algum modo.
- Com $c_i^{(1)}$ e a métrica d, calcule as células de quantização $C_i^{(1)}$ usando a condição 1.
- Das células de quantização $C_i^{(1)}$ e da métrica d, encontre novos níveis de quantização usando a condição 2.

- Repita os dois passos anteriores até o erro médio quadrático não mudar mais (dentro de uma determinada precisão).

Esse processo envolve muitas dificuldades:

- Devemos calcular os níveis de quantização $c_i^{(1)}$ para todas as possibilidades de cores no espaço, o que é computacionalmente caro.
- A distribuição de probabilidade de cores geralmente não é conhecida exatamente.
- As condições 1 e 2 são necessárias, mas não suficientes, para uma quantização ótima, logo o processo pode não convergir, ou pode convergir para uma solução não-ótima.

A literatura inclui várias maneiras de driblarmos esses problemas. Condições adicionais podem ser adicionadas para garantir que o processo de relaxação convirja e para melhorar sua eficiência computacional.

5.3 Quantização Ótima por Relaxação Estocástica

Como a quantização ótima depende de propriedades estatísticas da imagem, é natural tentar obtê-las usando um procedimento de *relaxação estocástica*. Tais métodos funcionam através de aproximações sucessivas como no método anterior, mas baseados em algoritmos estocásticos, ao invés de determinísticos para passar de uma aproximação para a próxima. A mecânica estatística mostra que as configurações de menor energia de certos sistemas físicos são os mais prováveis. Os funcionais de energia descrevendo estes sistemas estão associados às distribuições de Gibbs (Fiume & Ouellette, 1989a). O mínimo do sistema pode ser obtido, por exemplo, através de um algoritmo de relaxação estocástica. Tal algoritmo usa a distribuição de probabilidade a priori e realiza perturbações aleatórias na configuração inicial do sistema até que o estado de energia mínimo seja atingido. De uma maneira inteiramente análoga, essa teoria pode ser aplicada para quantização de cor.

Capítulo 6

Quantização por Aglomerados

Neste capítulo introduziremos o problema de aglomerados, abordando sua utilização para quantização de imagens digitais. Para tal, primeiro daremos uma breve introdução sobre análise de aglomerados, dando ênfase às principais heurísticas para solução de tais problemas. Dois bons trabalhos sobre o assunto podem ser encontrados em (Brucker, 1977) e (Procopiuc, 1996), que foram usados como base para este pequeno resumo.

Alguns algoritmos de quantização de imagens que se utilizam de técnicas de formação de aglomerados são apresentados. Daremos ênfase aos principais algoritmos, ou por sua importância histórica ou por sua utilidade.

6.1 Análise de Aglomerados

Problemas de aglomerados estão presentes em muitas áreas e têm uma grande faixa de aplicações, tais como compressão de dados, recuperação de informação, reconhecimento de padrão, localização de facilidades, e agrupamento de dados. Devido à grande variedade de domínios nos quais estes problemas aparecem, muitas variantes do problema de aglomerados vem sendo estudadas. Entretanto, todos elas exigem um particionamento de um dado conjunto de objetos em aglomerados que otimizem uma determinada função objetivo.

Seja S um conjunto de m objetos, geralmente representados como pontos em um espaço métrico n-dimensional. Um partição de S em k conjuntos $C_1, C_2, \ldots, C_k$ é chamado de k-aglomerado, e todo C_i é chamado de um aglomerado.

Definimos o *tamanho do aglomerado* de um k-aglomerado como sendo o menor valor d para o qual todos os pontos em todos os aglomerados ou estão

(a) dentro de uma distância d de cada um, ou

(b) dentro de uma distância $d/2$ de algum ponto chamado de centro do aglomerado.

A distância entre dois pontos é medida na métrica do espaço a que estes pontos pertencem (na realidade, podemos definir uma função de distância arbitrária $f : S \times S \to \mathbb{R}^+$ que não necessariamente satisfaça a desigualdade triangular). A escolha da função distância é intrinsecamente relacionada com a aplicação em particular.

Dependendo de como definimos o tamanho do aglomerado como em (a) ou em (b), o problema é chamado *aglomerado por par* ou *aglomerado central*, respectivamente. Além disso, problemas de aglomerados centrais podem ser divididos em problemas de *aglomerados centrais gerais*, se os centros dos aglomerados podem ser algum ponto no espaço n-dimensional, ou problemas de *aglomerados centrais discretos*, se os centros do aglomerado são necessariamente pontos de S.

Um problema de aglomerado geralmente tem uma das seguintes formas:

(P1) Dados S e k, encontre um k-aglomerado de S de tamanho de aglomerado mínimo.

(P2) Dados S e um número real w, particione S em um número mínimo de aglomerados de tal forma que o tamanho do aglomerado seja pelo menos w (isto é, encontre o menor valor de k de modo que S admita um k-aglomerado de tamanho de aglomerado menor ou igual a w).

Para problemas do tipo (P1), a função objetivo é o tamanho do aglomerado. Se definimos o valor d_i para o i-ésimo aglomerado como sendo a maior distância entre dois pontos em um aglomerado (ou, alternativamente, duas vezes a distância máxima entre um ponto no aglomerado ao seu "centro"), então (P1) é o problema de minimizar sobre todos os possíveis subconjuntos de S a função objetivo

$$c(S) = max \quad d_i, \quad 1 \leq \ i \leq k. \tag{6.1}$$

Estes são, particularmente, os tipos de problemas que estamos interessados na quantização de imagens, ou seja, estamos interessados em achar uma partição do espaço de cor com o menor erro de quantização possível dado um número de cores que desejamos obter na imagem quantizada.

6.1.1 Complexidade do Problema de Aglomerados

Em uma dimensão, aglomerados centrais e por pares podem ser resolvidos em tempo polinomial por programação dinâmica (Brucker, 1977). Em duas dimensões, aglomerados centrais é NP-completo (R. J. Fowler & Tanimoto, 1981) e (Megiddo & Supowit, 1984), mesmo quando procuramos por uma solução aproximada (Feder & Greene, 1988) (Gonzales, 1985) e (M. T. Ko & Chang, 1990). Para métricas satisfazendo a desigualdade triangular, aglomerados por pares se reduzem a aglomerados centrais, sendo então também NP-completos.

Assim, devemos encontrar soluções aproximadas para o problema de aglomerados de tal forma que estas sejam realizáveis em computadores.

6.1.2 Heurísticas

Ao contrário dos algoritmos que tentam encontrar uma solução exata para o problema de aglomerados, as heurísticas não se baseiam em uma função distância particular, e não fazem uso da "geometria"do problema para encontrar uma solução. Estas características as

fizeram muito populares em muitas aplicações onde algum tipo de aglomerado é necessário, mas o critério de aglomeramento não tem uma interpretação intuitiva no espaço geométrico. Sua principal desvantagem, no entanto, é que elas não garantem "bons"aglomerados, ou podem produzir aglomerados aonde não existem aglomerados "naturais". Muitas aplicações que usam tais heurísticas têm que adaptar e melhorar seu uso de acordo com conhecimentos específicos do problema que se propõem a resolver. E apesar de tais melhoramentos fazerem com que se comportem razoavelmente bem para a maioria das entradas, sempre existem casos nos quais a performance do método é muito fraca. As várias heurísticas em uso atualmente podem ser classificadas em duas categorias: *aglomerados hierárquicos* e *aglomerados particionais*. Nas duas próximas seções apresentaremos um resumo dessas categorias, que é baseado em (Anderberg, 1973) e (Jain & Dubes, 1988).

Heurísticas Hierárquicas de Aglomerados

Se C e C' são duas partições (aglomerados) do conjunto de entrada, dizemos que C está aninhado em C' se todo componente de C é um subconjunto próprio de C'. O objetivo de um método de aglomerado hierárquico é produzir uma série de partições aninhadas do conjunto de entrada, de tal forma que o nível mais baixo da partição consiste de n conjuntos, cada um contendo pontos do conjunto de entrada, e no nível mais alto a partição consiste do conjunto todo. O método pode ser desenhado por uma estrutura de árvore chamada de *dendograma*, no qual os nós correspondem a aglomerados gerados durante a execução do algoritmo, e existe uma aresta entre o aglomerado C e C' se C está incluso em C'. Uma camada do dendograma respresenta o aglomeramento do conjunto de entrada em alguma etapa do algoritmo (daí o nome aglomerado hierárquico).

O método é chamado de *agrupamento* se o algoritmo começa com n aglomerados individuais e os combina para obter todo o conjunto, ou *divisivo* se começa com o conjunto todo e o divide para obter aglomerados individuais. Claramente um método pode ser visto como o contrário do outro, assim, só consideraremos os algoritmos de agrupamento neste estudo.

Algoritmo de Agrupamento

1. Comece com n aglomerados, cada um contendo um ponto do conjunto de entrada. Numere os aglomerados de 1 até n.

2. Encontre o par de aglomerados r, s mais similares (isto é, $d(r, s)$ é o elemento mínimo de d). Combine os aglomerados r e s, exclua as linhas e as colunas de d correspondentes a r e s e insira uma nova linha e coluna para o novo aglomerado. Para economizar espaço, a nova coluna e linha podem ser escritas sobre a coluna e a linha de r, por exemplo, e então o aglomerado combinado será denominado r.

3. Repita o passo 2 um total de $n \Leftrightarrow 1$ vezes, até só restar um aglomerado.

As diferenças entre os vários métodos de agrupamento estão no modo de como o passo 2 é implementado, mais precisamente, do modo de como as distâncias de dissimilaridade do novo aglomerado são calculadas. Vamos mostrar aqui as alternativas mais populares. No

restante desta seção, chamaremos de t o aglomerado obtido no passo 2 da combinação dos aglomerados r e s, e por k qualquer outro aglomerado do aglomeramento corrente.

Aglomerado Simplesmente-ligado: $d(t,k) = min(d(r,k), d(s,k))$. Uma maneira de interpretar o aglomeramento a cada etapa deste método é a seguinte: considere um grafo cujos nós correspondem aos pontos de entrada, e cujas arestas são ponderadas pelos valores das entradas correspondentes em d. Se olharmos para o subgrafo obtido ao excluírmos todas as arestas com pesos maiores ou iguais à $d(r,s)$, cada componente conectada máxima desse subgrafo corresponde a um aglomerado logo antes dos aglomerados r e s serem escolhidos no passo 2 do algoritmo. Além disso, os aglomerados r e s correspondem às duas componentes conectadas do subgrafo que podem ser conectadas entre si por uma simples ligação de peso mínimo igual à $d(r,s)$ (daí o nome do método).

Aglomerado Completamente-ligado: $d(t,k) = max(d(r,k), d(s,k))$. Se considerarmos o subgrafo descrito anteriormente, os cliques máximos desse subgrafo correspondem a aglomerados que existem logo antes de r e s serem escolhidos durante o passo 2. Além disso, os cliques correspondentes aos aglomerados r e s podem ser transformados em um simples clique pela adição de uma aresta de tamanho mínimo igual à $d(r,s)$ (o nome do método é esse porque um clique é um (sub)grafo completo).

Aglomerado de Ligação Média: $d(t,k) = n_r/(n_r+n_s)d(r,k) + n_s/(n_r+n_s)d(s,k)$ (caso sem pesos), ou $d(t,k) = 1/2(d(r,k) + d(s,k))$ (caso com pesos). Quando calculamos a distância entre dois pares, a média das dissimilaridades entre elementos dos dois aglomerados são usados. No caso sem peso, levamos em consideração o número de elementos em cada aglomerado, enquanto que no caso com peso ignoramos esta informação, assim elementos em aglomerados menores tem peso maior que elementos em aglomerados maiores.

Aglomerado pelo Centróide: $d(t,k) = n_r/(n_r+n_s)d(r,k)+n_s/(n_r+n_s)d(s,k) \Leftrightarrow n_r n_s/(n_r+n_s)d(r,s)$ (caso sem pesos), ou $d(t,k) = 1/2(d(r,k) + d(s,k)) \Leftrightarrow 1/4 d(r,s)$ (caso com pesos, também conhecido como *aglomerado pela mediana*). Similar ao aglomerado de ligação média, exceto que neste caso um centróide é calculado para cada aglomerado, e a distância entre aglomerados é dada pela distância de seus respectivos centróides.

Aglomerado de Ward: $d(t,k) = (n_r+n_k)/(n_r+n_s+n_k)d(r,k) + (n_s+n_k)/(n_r+n_s+n_k)d(s,k) \Leftrightarrow n_k/(n_r+n_s+n_k)d(r,s)$. Esse método é estabelecido para combinar à cada etapa o par de aglomerados no qual a mudança na variância no aglomerado é minimizada. A variância é definida como a soma do quadrado dos erros do aglomerado, onde para cada aglomerado o erro quadrático é calculado adicionando a distância do erro quadrático entre um ponto no aglomerado e o seu centróide.

Na prática, o dendograma de um algoritmo simplesmente-ligado sempre tende a se parecer uma cadeia, um resultado que é indesejado pois não nos fornece uma boa idéia do que está se passando com os pontos de entrada. Por outro lado, aglomerados gerados por métodos

completamente-ligados podem não ser bem separados. Os outros três métodos descritos acima tentam eliminar estes problemas, evitando os extremos no modo como as dissimilaridades são definidas. Estudos comparativos indicam que o método de Ward tem uma performance melhor que os outros métodos hierárquicos.

Heurísticas Particionais de Aglomerados

Enquanto que o objetivo das heurísticas descritas na seção anterior era criar uma hierarquia de aglomerados que otimize o aninhamento de um aglomerado no outro (no sentido de que o dendograma resultante deve corresponder ao agrupamento dos pontos de entrada no espaço), as heurísticas nessa seção geram um k-aglomerado do conjunto de entrada refinando-o sucessivamente, até que um determinado critério de parada seja alcançado. Basicamente estamos procurando por uma solução ótima, ou perto da ótima, através da busca de possíveis aglomerados em alguma ordem heurística. O desejo é que para a maioria dos problemas, essa ordem seja boa o suficiente de tal modo que o aglomeramento encontre a condição de parada em poucas iterações. Entretanto, pode ser provado que um escolha ruim dos aglomerados iniciais pode nos levar a um aglomeramento final que aproxima a solução ótima por um fator arbitrariamente grande.

A seguir, apresentaremos três tipos de aglomerados por particionamento: *aglomerados por pontos sementes*, *aglomerados por grafos* e *aglomerados por vizinhança mais próxima*.

Um algoritmo de aglomeramento baseado em pontos sementes procede da seguinte maneira:

1. Escolha k pontos iniciais, chamados sementes. Dependendo do método empregado para sua escolha, as sementes podem ou não ser pontos do conjunto de entrada. Cada semente corresponderá a um aglomerado, que inicialmente está vazio.

2. Atribua cada ponto do conjunto de entrada ao aglomerado cuja semente está mais próxima do ponto.

3. Para cada aglomerado, calcule uma nova semente, de acordo com algum método que leva em consideração os pontos do conjunto de entrada dentro do aglomerado.

4. Repita os passos 2 a 3 até que uma condição de parada seja alcançada.

Várias heurísticas podem ser usadas para a escolha das k sementes iniciais, tais como: aleatoriamente escolha k pontos do conjunto de entrada (algoritmo de MacQueen); pegue qualquer partição dos pontos do conjunto de entrada em k aglomerados e escolha a semente como sendo o centróide desses aglomerados (algoritmo de Forgy); use algum algoritmo hierárquico de aglomeramento para obter um k-aglomerado inicial (tome o mais alto nível no dendograma que tenha pelo menos k aglomerados, e combine os aglomerados mais similares, se necessário, para obter exatamente k aglomerados).

Uma função distância deve ser especificada para a execução do passo 2 do algoritmo. O método pelo qual uma nova semente do aglomerado é calculado no passo 3 está também de

perto relacionado com a escolha da distância. A distância mais popular em várias aplicações parece ser o erro quadrático, neste caso a nova semente de um aglomerado é o seu centróide.

O algoritmo pára no passo 4 quando um número máximo de iterações foi executado, ou quando a "diferença"entre dois aglomeramentos gerados por iterações consecutivas é menor que um certo limiar (a definição dessa diferença depende da função distância).

Outra categoria de algoritmos de particionamento usa grafos para dividir o conjunto de pontos de entrada em aglomerados. Primeiro, um grafo completo é definido de tal forma que seus nós correspondam aos pontos de entrada, e suas arestas são ponderadas pela distância entre os pontos correspondentes. O algoritmo então se processa da seguinte maneira:

1. Construa algum subgrafo do grafo completo.

2. Remova as arestas inconsistentes do subgrafo, e chame cada componente conectada resultante um aglomerado.

Várias estruturas de subgrafos têm sido consideradas para a implementação do passo 1: árvore de caminho mínimo (minimum spanning tree), grafo de vizinhança relativa (pontos p_i e p_j estão conectados por uma aresta se e somente se $d(p_i, p_j) \leq max(d(p_i, p_k), d(p_j, p_k))$ para algum ponto de entrada $p_k \neq p_i, p_j$), o grafo de Gabriel (pontos p_i e p_j estão conectados por uma aresta se e somente se $d^2(p_i, p_j) < d^2(p_i, p_k) + d^2(p_j, p_k)$ para todo ponto de entrada $p_k \neq p_i, p_j$), ou a triangulação de Delaunay dos pontos de entrada.

Uma aresta inconsistente é aquela cujo peso é significantemente maior que a média dos pesos das arestas próximas.

O último algoritmo que apresentamos usa a regra da vizinhança mais próxima para aglomerar os pontos. O usuário fornece um parâmetro de limiar t, que irá influenciar no número de aglomerados gerados:

1. Seja $S = \{p_1, \ldots, p_n\}$. Atribua p_1 ao aglomerado C_1.

2. Para cada p_k $(2 \leq k \leq n)$, encontre seu vizinho mais próximo p_j dentre todos os pontos $\{p_1, \ldots, p_{k-1}\}$. Se $d(p_j, p_k) \leq t$, atribua p_k ao aglomerado p_j. Senão, crie um novo aglomerado e atribua p_k a esse aglomerado.

6.1.3 Quantização e Aglomerados

Assim, se considerarmos que o espaço de cor no computador é discreto, temos que cada cor pode ser representada como um elemento de um conjunto de entrada. Consideramos, então, o problema de quantização de imagem como um problema de formação de aglomerados de cores similares que, ao serem substituídas pelo seu valor representativo dá origem a uma quantização do espaço de cor discreto. O conceito de similaridade está ligado a quão próximas estas cores estão no espaço de cor. Logo, podemos naturalmente escrever o problema de quantização como sendo um problema de otimização em análise de aglomerado: a solução ótima deve minimizar o erro de quantização em todas as possíveis partições de N-elementos do espaço de cor discreto.

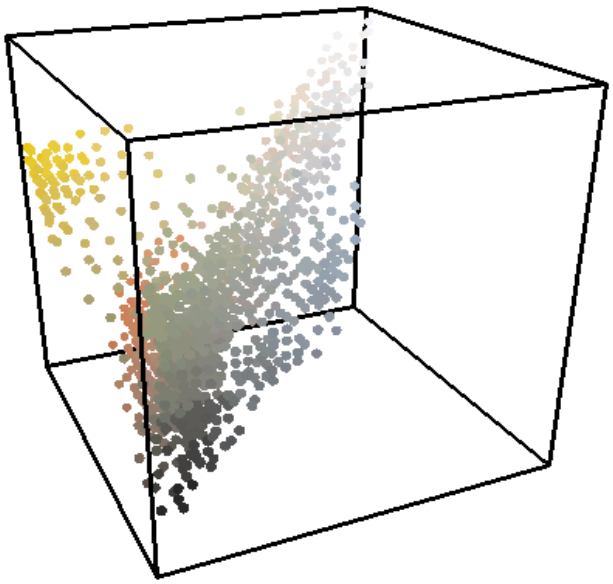


Figura 6.1: Cubo RGB discreto com um conjunto de cores.

Na Figura 6.1 vemos o cubo RGB com as cores presentes na imagem da arara. Vemos claramente que as cores presentes nesta imagem formam aglomerados de cores que podem ser substituídos pelo seu nível de quantização, dando origem à imagem quantizada.

6.2 Quantização com Aglomerados por Curva de Peano

Introduzida na literatura por A. F. Lehar e R. J. Stevens em 1984, este algoritmo utiliza-se de algumas propriedades das curvas de Peano para dar origem a um algoritmo de quantização baseado em clustering. As seguintes propriedades das curvas de Peano (Sagan, 1994) devem ser levadas em consideração:

- Uma curva de Peano é um curva sem interrupções que passa somente uma vez por cada elemento (cor) do espaço de cor.
- Pontos próximos na curva de Peano estão próximos no espaço de cor.
- Pontos próximos no espaço de cor tendem a estar próximos na curva de Peano.
- A curva de Peano atua como uma transformação de um espaço n-dimensional em um espaço unidimensional, preservando algumas propriedades espaciais nessa varredura.

Assim, uma curva de Peano passa por todo espaço de cor tri-dimensional e é então "esticada", ou seja, transformamos um espaço 3-D em um espaço 1-D. Na Figura 6.2 temos o exemplo de uma curva de Peano que preenche todo o cubo RGB. O histograma de cor unidimensional gerado ao "esticarmos"esta curva pode ser visto na Figura 6.3, onde temos

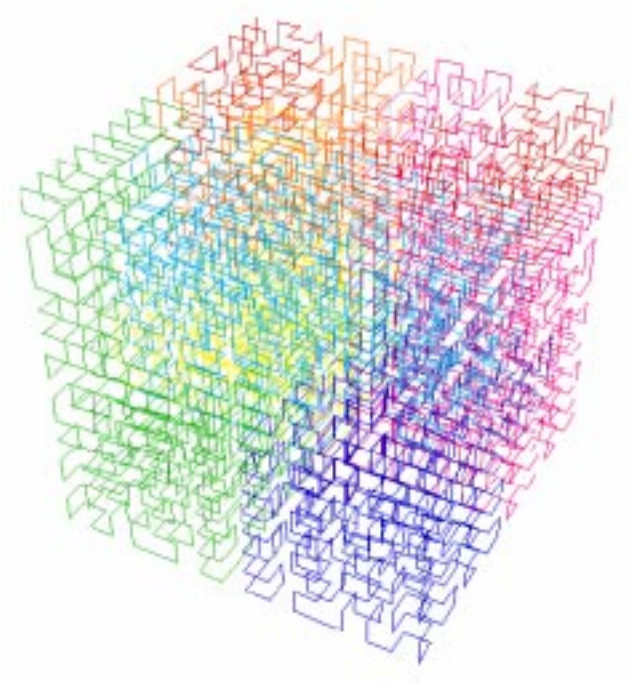


Figura 6.2: Curva de Peano preenchendo todo o cubo RGB.

todas as cores do cubo RGB. A distância entre cores sai direto do histograma, onde podemos considerar que cada cor esta localizada em um número inteiro. Assim, para o cubo RGB, teríamos que as cores mais distantes distam 2^{24} unidades.

Se o dado (cor) de um pixel pertence a um aglomerado no espaço, ele também pertencerá ao mesmo aglomerado no histograma unidimensional. Logo, as cores do mapa de cor podem ser escolhidas com base nessa distribuição.

O mapa de cor é então gerado pela técnica, já descrita, conhecida como equalização de histograma, ou seja, tentamos atribuir a cada célula de quantização o mesmo número de pixels. O processo de quantização por equalização de histograma consiste em fazermos subdivisões sucessivas do intervalo de intensidades da imagem utilizando a mediana desse conjunto em cada processo de subdivisão.

Temos o seguinte algoritmo:

1. Gere uma curva de Peano que preencha todo o espaço de cor.

2. "Estique"esta curva para que ela fique unidimensional.

3. Varra a imagem original mapeando suas cores nesta curva de Peano "esticada". Isso nos fornecerá o histograma unidimensional do espaço RGB onde cores próximas no cubo RGB estarão próximas no histograma devido às propriedades das curvas de Peano.

4. Utilizando uma técnica de equalização de histograma gere o número de níveis de quantização desejados.

5. Depois de encontrado os níveis de quantização, encontre suas respectivas células de quantização e mapeie as cores da imagem original em seus níveis de quantização.

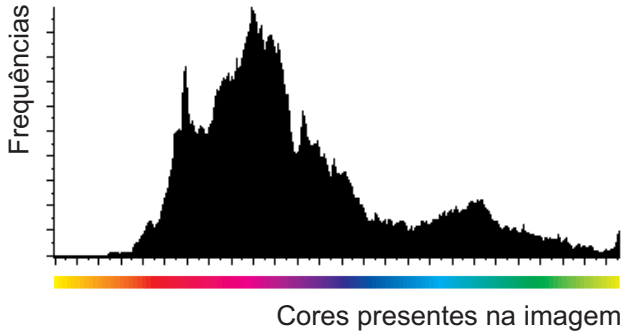


Figura 6.3: Histograma de cor unidimensional com todas as cores do cubo RGB.

6.3 Quantização por Octree

Este algoritmo foi introduzido na literatura por Michael Gervautz e Werner Purgathofer em 1990.

A imagem é lida sequencialmente. As primeiras K cores diferentes são inseridas na octree. Se uma outra cor é adicionada, que significa que a parte processada da imagem já tem $K + 1$ cores diferentes, os vizinhos mais próximos são agrupados e suas cores são substituídas pela média das cores presentes nos respectivos sub-cubos de cor. Esse passo é repetido para toda cor adicional, de modo que a qualquer momento não exista mais do que K cores representativas. Essa propriedade, é claro, continua verdadeira depois da imagem ser completamente processada.

Para exemplificar este método de quantização considere a imagem da Figura 6.4 como sendo a imagem que queremos quantizar. Esta imagem apresenta 10 cores distintas e queremos quantizá-la para 6 cores. Logo, para este exemplo $K = 6$.

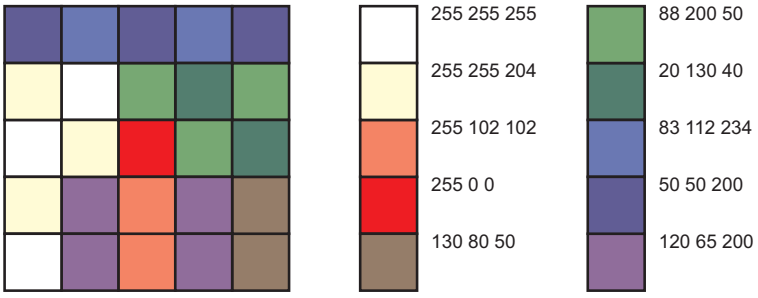


Figura 6.4: Imagem que será usada para exemplificar a Quantização por Octree.

6.3.1 A Octree

Para esse método de quantização devemos usar uma estrutura de dados que possibilite uma detecção rápida de cores que sejam próximas no espaço de cor. Uma árvore octree se adequa bem para este problema. O cubo RGB pode ser facilmente administrado por uma octree (Figura 6.5).

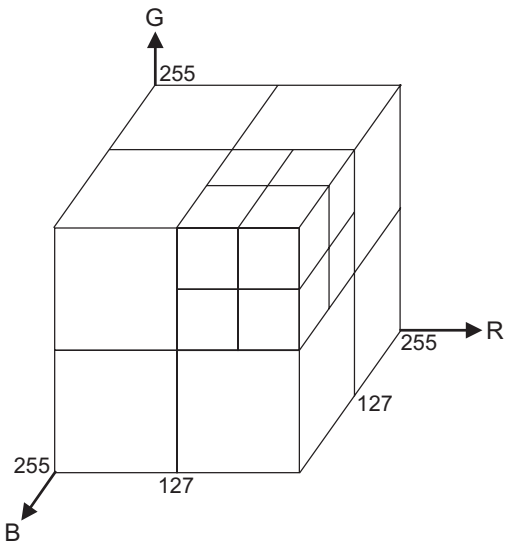


Figura 6.5: Uma octree para aglomerados de cores em um cubo RGB.

Com o uso da octree temos em cada nível da árvore os limites do cubo em que as cores estão contidas. Assim, se considerarmos todas as cores no gamute da imagem, temos o cubo com as componentes R, G e B variando de [0 - 255]. No sub-cubo com as componentes R variando no intervalo [0 - 127), G variando no intervalo [127 - 255] e B variando no intervalo [0 -127) temos todas cores do gamute da imagem que fazem parte deste sub-cubo, como por exemplo a cor R = 100, G = 200 e B = 50.

À medida que caminhamos a níveis mais profundos na árvore octree temos sub-cubos menores. Assim, quanto maior a profundidade de um nó, menor é o sub-cubo de cor representado por ele. Consequentimente, a profundidade de um nó é uma medida para a distância máxima de suas cores.

6.3.2 O Algoritmo

A quantização por octree é feita em três fases:

- Determinação das cores representativas.
- Preenchimento da tabela de cores.
- Mapeamento das cores originais nas cores representativas.

Vamos descrever com mais detalhes cada um dos passos acima usando a octree de cor.

Determinação das Cores Representativas

A octree é construída somente nas partes que são necessárias para a imagem sendo quantizada, ou seja, não teremos sub-cubos se não tivermos pelo menos uma cor contida no mesmo. No início a octree está vazia. Toda cor que ocorre na imagem é então inserida.

Existem três casos possíveis ao inserirmos uma cor na octree:

Caso 1 O elo da árvore octree selecionado é um ponteiro vazio, logo o sub-cubo correspondente ainda está vazio. Neste caso podemos simplesmente criar um novo nó folha e armazenar a nova cor neste nó. Na Figura 6.6 (A) temos uma octree no momento em que vamos inserir a cor de componentes RGB (255, 255, 204). Esta cor faz parte de um sub-cubo que ainda não está presente na octree. Assim, criamos um novo nó, que representa o sub-cubo onde esta cor está contida, e inserindo-a e incrementando o contador de cores contidas no nó, que chamamos de NP (Figura 6.6 (B)).

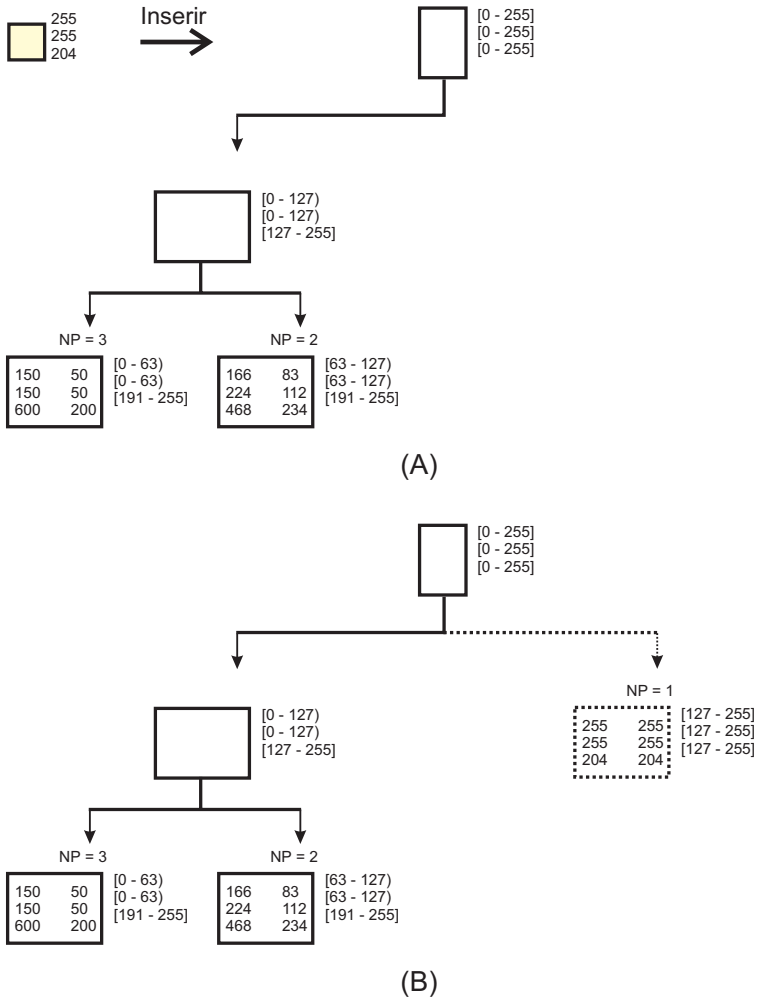


Figura 6.6: Inserção de uma nova cor em uma árvore octree: caso 1.

Caso 2 O elo da árvore octree aponta para um nó folha e a cor que queremos inserir é igual a cor armazenada no nó. Nenhum novo nó é criado, incrementamos o contador de cores contidos no nó (NP) e somamos as componentes R, G e B da nova cor aos respectivos acumuladores de R, G e B do nó em questão. Na Figura 6.7 (A) temos a octree antes de inserirmos a cor de componentes RGB (50, 50, 200). Como esta cor já está presente na árvore, simplesmente incrementamos NP e atualizamos os acumuladores R, G e B somando as respectivas componentes RGB da cor sendo inserida, como pode ser visto na Figura 6.7 (B).

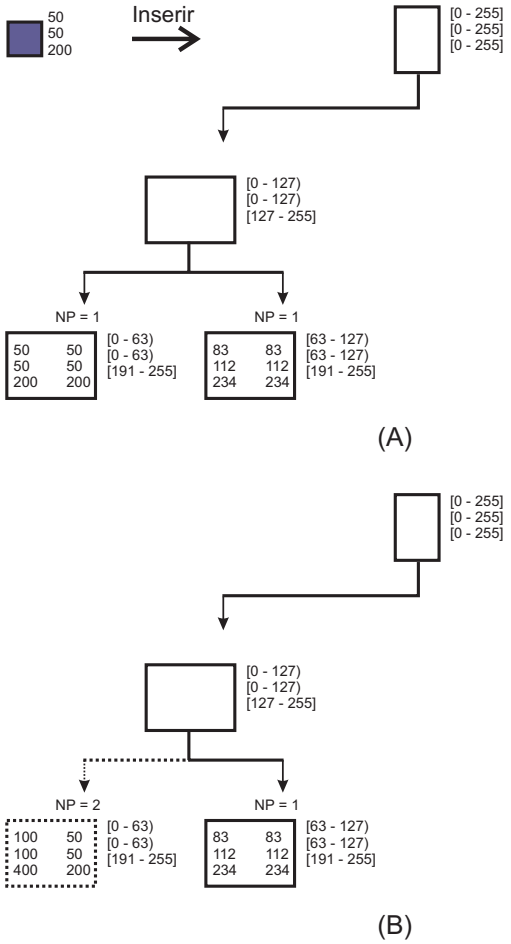


Figura 6.7: Inserção de uma nova cor em uma árvore octree: caso 2.

Caso 3 O elo da árvore octree aponta para um nó folha mas a nova cor não é igual a cor armazenada no nó. Neste caso, a cor contida no nó e a nova cor estão contidas no mesmo sub-cubo, logo a árvore octree deve ser subdividida neste nó. Um novo nó intermediário é criado e o antigo nó folha e o novo nó contendo a nova cor são inseridos como filhos do novo nó intermediário. A octree antes de inserirmos a cor de componentes RGB (83, 112, 234) pode ser vista na Figura 6.8 (A). Como a nova cor a ser inserida está contida

em um sub-cubo já existente e ela é diferente da cor presente no mesmo, devemos subdividir este sub-cubo de modo que possamos inserir esta nova cor no sub-cubo onde está contida, como vemos na Figura 6.8 (B).

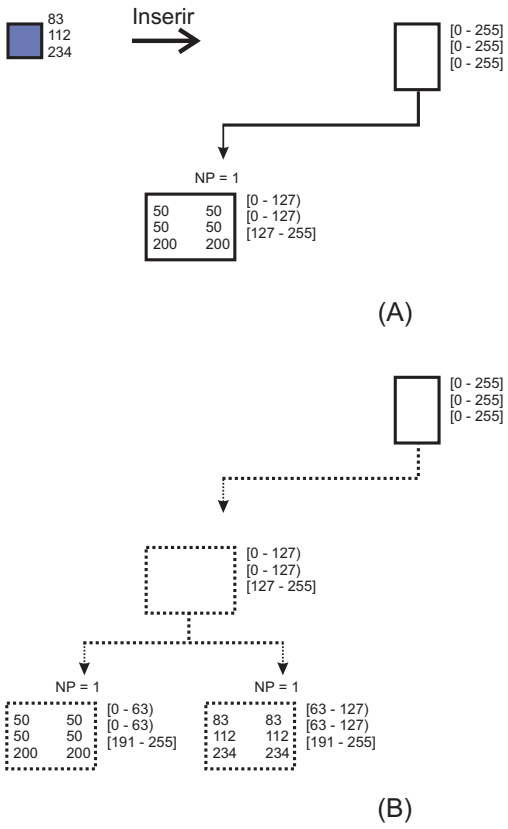


Figura 6.8: Inserção de uma nova cor em uma árvore octree: caso 3.

Desse modo uma árvore octree incompleta é criada na qual muitos ramos são perdidos. Na verdade, essa octree não tem que ser preenchida com todas as cores porque cada vez que o número de cores atinge $K + 1$, cores similares podem ser combinadas, logo nunca existirá mais que K cores restantes. Chamaremos essa ação de redução de cores.

Toda vez que o número de nós folhas (isto é, o número de cores representativas encontradas até o momento) excede K, a octree é reduzida. A redução começa na parte inferior da octree através da substituição de algumas folhas pelo seu predecessor.

Na redução da octree, os seguintes critérios são relevantes:

- De todos os nós redutíveis, aqueles que tiverem mais profundos na octree devem ser escolhidos primeiro, porque eles representam cores que estão mais próximas.

- Se existir mais de um nó nas maiores profundidades, um critério adicional pode ser usado para um solução ótima. Por exemplo, reduza o nó que representa a menor quantidade de pixels até o momento. Desse modo a soma do erro se manterá pequena.

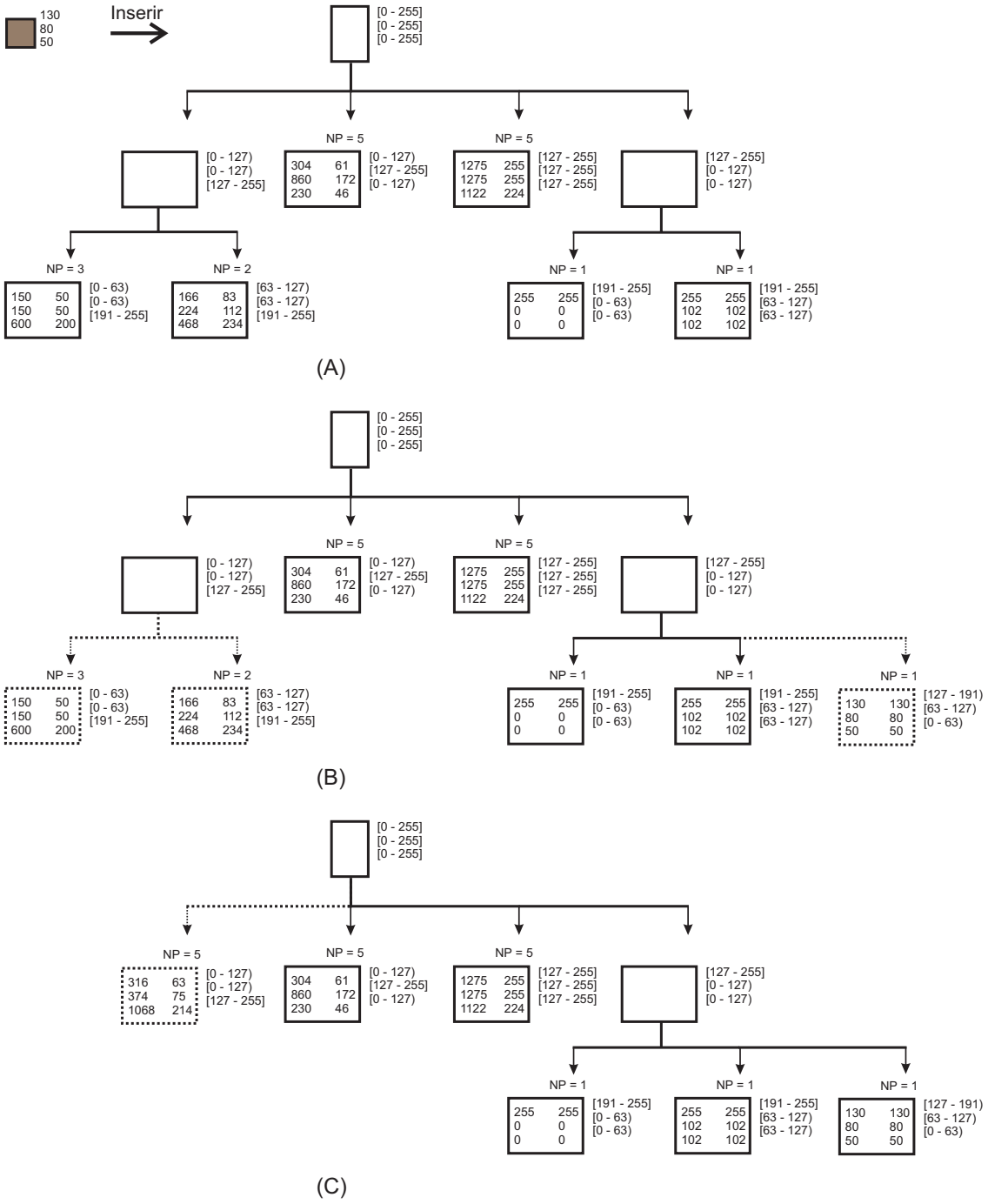


Figura 6.9: Quando uma árvore octree conter $K+1$ cores, devemos fazer uma ação de redução desse número de cores para K.

Na Figura 6.9 (A) temos o estado da octree antes de inserirmos a cor com componentes RGB $(130, 80, 50)$. Ao inserirmos esta cor, o número de cores representativas excede o número

de cores que desejamos para a imagem quantizada, no nosso exemplo $K = 6$ (Figura 6.9 (B)). Devemos então fazer uma redução na octree, como pode ser visto na Figura 6.9 (C).

Para construir a octree, toda imagem deve ser lida uma vez e todas as cores da imagem têm que ser inseridas na octree.

Preenchendo a Tabela de Cor

No final, os K nós folhas da octree contém as cores para a tabela de cores (o valor médio de todas as cores representativas = (acumuladores RGB) / NP). Elas podem ser escritas na tabela de cores através de um exame recursivo da octree. Durante esta passagem recursiva pela árvore em cada nó folha da octree seu próprio índice de cor é também armazenado.

Na Figura 6.10 temos a árvore octree gerada após lermos todas as cores na imagem.

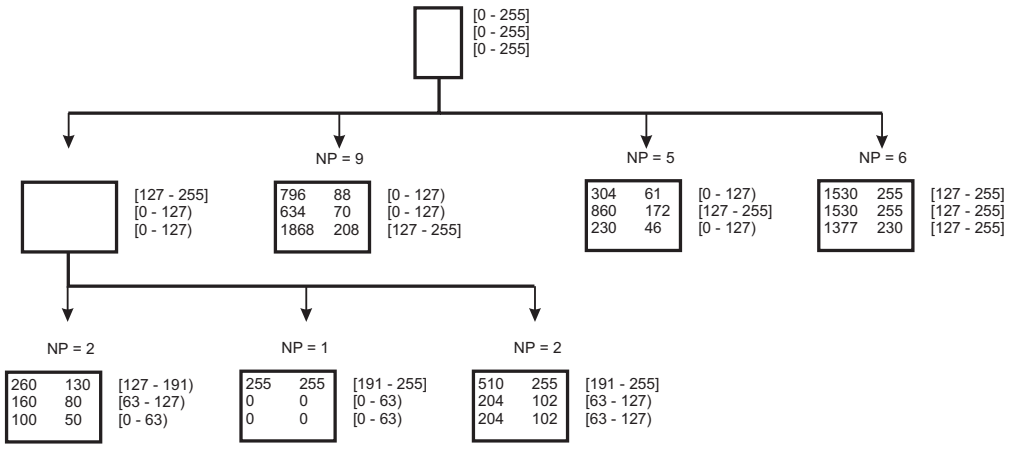


Figura 6.10: Árvore gerada após o termino do algoritmo de Quantização por Octree.

Mapeando nas Cores Representativas

O mapeamento das cores originais em suas cores representativas agora pode ser facilmente gerenciado com a octree também. Tentando encontrar qualquer cor original na octree chegaremos em um nó folha em algum nível da árvore. Este nó contém uma cor muito similar à cor que se procura, sendo portanto sua cor representativa. Como o índice da tabela de cor também está armazenado no nó, não será necessário fazer mais buscas.

Se a imagem original tiver menos que K cores não será necessário fazer nenhuma redução, e a entrada na tabela de cores encontrada conterá exatamente a cor original. Como a octree contém K folhas, todas as cores originais são mapeadas em entradas válidas da tabela de cor. Para isso a imagem deve ser lida uma segunda vez.

O resultado da quantização da imagem que usamos para exemplificar este método pode ser visto na Figura 6.11.

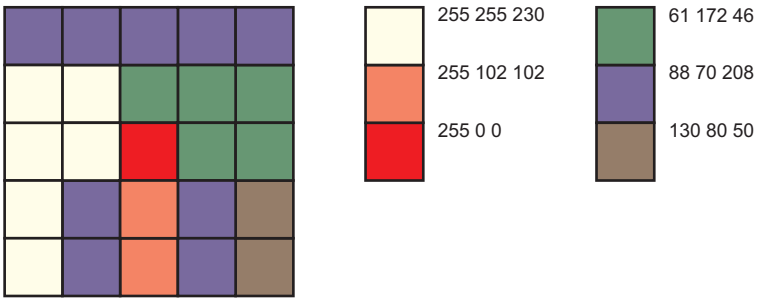


Figura 6.11: Imagem usada para exemplificar o método de Quantização por Octree quantizada para 6 cores.

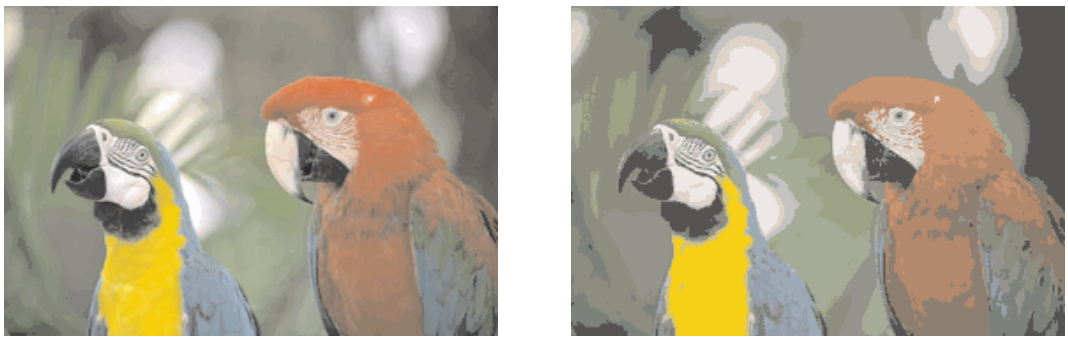


Figura 6.12: Quantização por Octree: (esquerda) quantização para 8 bits, (direita) quantização para 4 bits.

A Figura 6.12 (esquerda) mostra a imagem das araras quantizada para 8 bits utilizando o método de quantização por octree. Na Figura 6.12 (direita) mostramos a mesma imagem quantizada para 4 bits com o mesmo método.

6.4 Quantização por K-means Local

Introduzido na literatura por Oleg Verevka e John W. Buchanan em 1995 este algoritmo tem como objetivo avaliar uma combinação da distorção média das cores

$$E(C) = \frac{1}{N}\sum_{i=1}^{N} \|c_i \Leftrightarrow q(c_i)\| \tag{6.2}$$

e o desvio padrão da distorção por pixel

$$\sigma = \sqrt{\frac{\sum_{(x,y)\in I}(\|c_{(x,y)} \Leftrightarrow q(c_{(x,y)})\| \Leftrightarrow E(C,I))^2}{M}}, \tag{6.3}$$

O objetivo deste algoritmo é minimizar ambas as medidas simultaneamente. Valores pequenos de $E(C)$ garantem que o processo de quantização representa corretamente as cores da imagem original. Infelizmente o sistema visual humano não é capaz de determinar o valor absoluto da cor. Ele é mais sensível a variações de cores. Um algoritmo de quantização que produz valores pequenos de σ introduz quase a mesma distorção de cor para cada pixel. Logo, a minimização do desvio padrão da distorção σ nos ajuda a preservar as variações das cores na imagem quantizada.

Deve ser notado que estas medidas têm uma limitação significante. Apesar de $E(C)$ e σ serem medidas dependentes da imagem elas tratam cada pixel independentemente. A correlação espacial entre as cores não é levada em consideração, o que pode levar a quantizações muito ruins.

6.4.1 Algoritmo K-means e suas Variações

O algoritmo K-means (também conhecido como LBG - de Linde, Buzo e Gray, seus autores (Y. Linde & Gray, 1980)) é um método de quantização vetorial para refinamentos iterativos em dicionário de dados. O algoritmo é muito geral e também vem sendo aplicado em uma grande variedade de problemas em reconhecimento de padrões e aglomerados. Como o problema de construção de um dicionário de dados ótimo é equivalente ao problema de encontrarmos um conjunto de cores para formar uma paleta de cores, $\{q_n : 1 \leq n \leq M\}$, que melhor aproxima o conjunto de cores dos pixels da imagem original $\{c_n : 1 \leq n \leq N\}$, o algoritmo K-means é diretamente aplicável ao problema de quantização de cor. No caso de construção de paletas com erro médio quadrático mínimo, o algoritmo é definido como se segue:

1. Escolha uma paleta de cor inicial
$$\{q_n : 1 \leq n \leq M\}$$
2. Forme M aglomerados
$$C_m = \{c_n : \|c_n \Leftrightarrow q_m\| \leq \|c_n \Leftrightarrow q_k\|, \quad 1 \leq \ k \leq \ M\}$$
3. Recalcule a paleta de cor usando
$$q_m = \frac{1}{|C_m|} \sum_{s \in C_n} c_s$$
4. Repita os passos 2 e 3 até não ocorrer mais mudanças ou a diminuição do erro médio quadrático ser menor que um determinado limiar.

Cada passo do algoritmo K-means faz com que o erro médio quadrático ou diminua ou permaneça o mesmo. Conseqüentemente, o algoritmo é garantido que converge para um mínimo do erro médio quadrático. Entretanto, esse mínimo geralmente não é o mínimo global do erro médio quadrático. Logo, a paleta inicial tem um papel muito importante na determinação do mínimo local que o algoritmo converge. Concluímos então que a melhor utilização do algoritmo LBG é como um pós-processamento, melhorando a palete sub-ótima inicial.

6.4.2 Mapas auto-organizáveis de Kohonen

Um mapa auto-organizável (SOM - self-organizing map) é um esquema de pós-aglomerado. Foi introduzido por Kohonen (Jari A. Kangas & Laaksonen, 1990) como uma solução para um problema de quantização vetorial geral. O SOM é uma rede neuronal que impõe uma estrutura topológica uni ou bidimensional sobre um conjunto de aglomerados em um espaço de dimensão maior. O processo de adaptação tenta aproximar a função de densidade da entrada.

O uso de um mapa auto-organizável unidimensional para quantização de imagens foi proposto por Dekker (Dekker, 1993). A palete inicial é ajustada para valores igualmente espaçados de cinza. Os valores de entrada são obtidos de amostragens múltiplas da imagem com tamanhos de passos grandes. A cor mais próxima $\overline{c_k}^{(t)}$ da palete é ajustada para melhor concordar com a entrada $c^{(t)}$.

A rede é considerada *elástica*. Então, quando $\overline{c_k}^{(t)}$ é atualizada outro $\overline{c_k}^{(t)}$: $|k \Leftrightarrow j| \leq r$ é também movido. O parâmetro r é o raio de elasticidade que diminui com o tempo. O processo de adaptação do SOM é definido como se segue:

$$\overline{c_j}^{(t+1)} = \overline{c_j}^{(t)} + \alpha_t \rho(t,j) ||c^{(t)} \Leftrightarrow \overline{c_j}^{(t)}||, \tag{6.4}$$

onde o parâmetro de adaptação $0 < \alpha_t < 1$ decresce exponencialmente. O coeficiente de elasticidade $\rho(t,j)$ garante que somente entradas na r-vizinhança (vizinhança que distam no máximo r de $\overline{c_j}^{(t)}$) são atualizadas.

Como a atualização das vizinhanças sempre se sobrepõe os valores de $\overline{c_k}$ tendem a se tornar suaves. Para garantir uma representação satisfatória das regiões de cor pela palete $\overline{C}$ Desieno (Hecht-Nielsen, 1990) propôs o uso de um valor especial de tendência (bias) $b_j^{(t)}$. A cor de entrada $c^{(t)}$ atualiza a entrada na palete $\overline{c_k}^{(t)}$ encontrada pela seguinte regra:

$$||c^{(t)} \Leftrightarrow \overline{c_k}^{(t)}|| \Leftrightarrow b_k^{(t)} = min \quad ||c^{(t)} \Leftrightarrow \overline{c_j}^{(t)}|| \Leftrightarrow b_j^{(t)}, \quad j = 1, 2, \ldots, K. \tag{6.5}$$

O fator de tendência aumenta para vetores escolhidos com menos freqüência. Assim uma cor que foi escolhida muitas vezes antes não será mais escolhida adiante.

6.4.3 O Algoritmo de K-means Local

Este método pode ser considerado um caso especial de um mapa auto-organizável. Ao contrário da rede de Kohonen, o passo de adaptação do processo de LKM (Local K-means - nome em inglês do algoritmo de K-means local) atualiza somente as cores mais próximas:

$$\overline{c_j}^{(t)} = \begin{cases} \overline{c_j}^{(t-1)} + \alpha_t ||c^{(t)} \Leftrightarrow \overline{c_j}^{(t)}|| & j = k \\ \overline{c_j}^{(t-1)} & j \neq k \end{cases} \tag{6.6}$$

No caso da rede de Kohonen o fator de tendência assegura que um aglomerado de cores ligeiramente distintos seja representado por uma entrada separada na palete de cores (Dekker,

1993). O mesmo resultado pode ser obtido através de uma escolha adequada da palete inicial. Construiremos a palete inicial através de inserção incremental de uma cor da imagem original. Uma nova cor é adicionada à palete se sua distância para as cores já inseridas excede um limiar especificado.

Os conjuntos de dados de entrada são construídos através de amostragens na imagem original com passos que diminuem de tamanho: 1009, 757, 499, 421, 307, 239, 197,... Escolhemos estes tamanhos de passos por serem números primos, logo os conjuntos de entrada não se interceptam muito. O processo de iteração pára quando as mudanças na palete $\overline{C}^{(t)}$ em uma varredura completa da imagem se torna pequeno. Segundo testes realizados pelos autores, a união dos conjuntos de entrada não incluem mais do que 10% do que todos os pontos da imagem.

Apesar do algoritmo examinar somente uma parte da imagem de entrada, ele é capaz de gerar boas paletes aproximativas. Esse resultado poder ser explicado pelo fato de que as cores de uma imagem geralmente estão aglomeradas no espaço de cor. Portanto, é suficiente usarmos poucas cores do aglomerado para aproximarmos todos seus membros. Como cores similares estão geralmente próximas umas das outras na superfície da imagem, esperamos que os conjuntos de entrada contenham cores representativas para a maioria dos aglomerados.

Depois de gerada a palete de cores, temos que mapear as cores da imagem em suas cores representativas. Para isso devemos varrer mais uma vez a imagem e para cada cor de entrada, escolher a cor mais próxima na palete de cores.

Figura 6.13: Quantização por K-means local: (esquerda) quantização para 8 bits, (direita) quantização para 4 bits.

A Figura 6.13 (esquerda) mostra a imagem das araras quantizada para 8 bits utilizando o método de quantização por K-means local. Na Figura 6.12 (direita) mostramos a mesma imagem quantizada para 4 bits com o mesmo método.

6.5 Quantização usando Heurísticas Hierárquicas de Aglomerados por Agrupamento

Nesta estratégia de aglomeramento começamos com N aglomerados, cada um contendo uma cor da imagem. Repetidamente combinamos aglomerados de uma maneira hierárquica até que o número desejado de K aglomerados seja obtido. A forma genérica a seguir descreve o algoritmo básico:

1. $n = N$
2. Enquanto $(n > K)$ faça

   Encontre dois aglomerados c_i e c_j que desejamos que sejam agrupados de acordo com alguma regra.

   Combine c_i e c_j.

   $n = n \Leftrightarrow 1$.

Alguns métodos de quantização se baseiam nesta técnica de aglomeramento para quantizar imagens. Vamos falar sobre dois algoritmos nesta seção: um proposto por Sudhir S. Dixit (Dixit, 1991) em 1991 e um outro mais recente, proposto em 1994 por Zhigang Xiang e Gregory Joy (Xiang & Joy, 1994a).

Dixit propõe um algoritmo em que pares de cores que estão próximos são combinadas para formar um aglomerado maior, sendo representado por uma nova cor chamada de centróide ponderado que substitui as cores que compõem o par.

Todas as cores presentes no histograma de cor da imagem são rearranjadas em ordem ascendente de freqüência de ocorrência da cor na imagem. Começando pela primeira entrada (r_1, g_1, b_1), nesta tabela de cores, a cor mais próxima, (r_j, g_j, b_j), é encontrada no conjunto. Esse par de cores é então excluído do processo de busca pelo vizinho mais próximo quando procuramos pelo próximo par de cores que deve ser excluído. A tabela é percorrida de cima para baixo de modo a encontrar pares de aglomerados, esse processo é repetido até que não tenha mais pares a serem combinados ou o número de cores desejado na imagem quantizada já tenha sido obtido. O par mais próximo é determinado calculando-se a distância Euclidiana ponderada, d_{ij}, entre as cores c_i e c_j na tabela de cores, como mostrado abaixo

$$d_{ij}^2 = \frac{F_i F_j}{F_i + F_j}[(r_i \Leftrightarrow r_j)^2 + (g_i \Leftrightarrow g_j)^2 + (b_i \Leftrightarrow b_j)^2]. \tag{6.7}$$

Esta ponderação é feita no sentido de influenciar a formação de pares nos quais as cores presentes no mesmo tenham freqüências de ocorrência baixa na imagem, depois pares de altas e baixas freqüências, e por último pares com cores que tenham freqüências de ocorrência elevada na imagem, nesta ordem. Os pares de cores são em seguida combinados de uma maneira ponderada de tal modo que o centróide esteja mais perto da cor de maior ocorrência, como se segue

$$
\begin{aligned}
r'_{ij} &= \frac{r_i F_i + r_j F_j}{F_i + F_j} \\
g'_{ij} &= \frac{g_i F_i + g_j F_j}{F_i + F_j} \\
b'_{ij} &= \frac{b_i F_i + b_j F_j}{F_i + F_j}
\end{aligned} \tag{6.8}
$$

A freqüência da nova cor $(r'_{ij}, g'_{ij}, b'_{ij})$ é dada pela soma das freqüências das cores que compõem o par, ou seja $F'_{ij} = F_i + F_j$.

Cada iteração completa reduz o número de cores na tabela de cores pela metade. Esse processo é repetido, recursivamente, até que o número de cores desejado seja obtido. Note que o número de cores na tabela de cores original não pode ser igual a um múltiplo inteiro ímpar de K (o número de cores que se deseja na imagem quantizada). Conseqüentemente, a última iteração pode terminar assim que K cores sejam obtidas na tabela de cores.

Na Figura 6.14 temos um exemplo unidimensional do algoritmo proposto por Dixit.

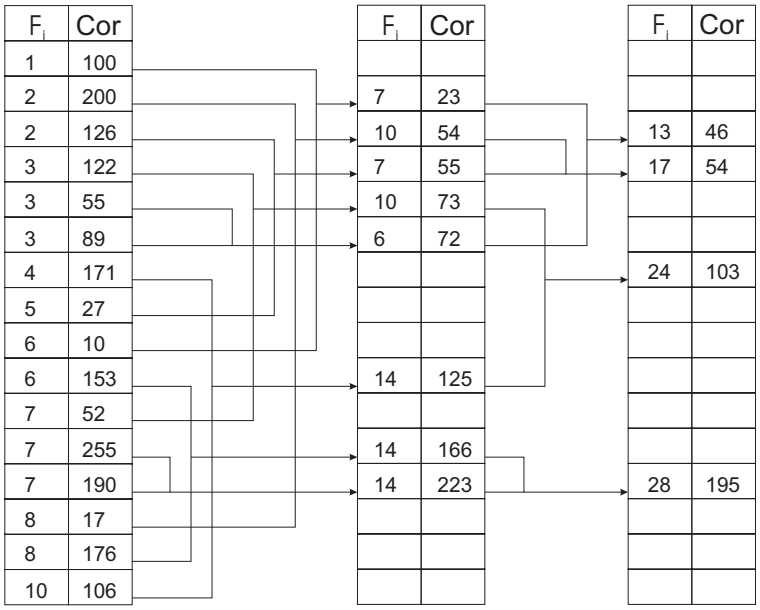


Figura 6.14: Exemplo de um quantização usando heurística hierárquica de aglomerados por agrupamento proposta por Dixit para um espaço unidimensional.

O algoritmo proposto por Zhigang Xiang e Gregory Joy começa definindo cada aglomerado pelo seu menor paralelepípedo envolvente (como no algoritmo do corte mediano). Medimos o erro (máximo) de quantização associado a um paralelepípedo de cor (geralmente o resultado da combinação de dois paralelepípedos menores) em termos do tamanho do paralelepípedo em cada uma das três direções primárias (componentes R, G e B). Isso significa que a cada

iteração, o tamanho limite de um paralelepípedo de cor resultante de uma operação de agrupamento especifica o critério de seleção para o agrupamento.

Como usaremos o espaço RGB, regulamos os paralelepípedos de cor com um tamanho limite proporcional nos seus três lados. Recomendamos uma relação de 2:1:4 para vermelho (r - red): verde (g - green): azul (b - blue). Esta escolha vem da fórmula do componente de luminância do modelo de cor NTSC YIQ: $0.30 \times R + 0.59 \times G + 0.11 \times B$. Aproximadamente 2 unidades de deslocamento no vermelho ou 4 unidades de deslocamento no azul causam a mesma mudança na luminância que uma unidade de deslocamento no verde. Assim de fato restringimos deslocamentos na luminância. Preservar o contraste de luminância é muito importante em qualquer processo de quantização visto que o sistema visual humano é mais sensível a variações na luminância do que na crominância.

A natureza discreta do sistema de exibição também é aproveitada na construção da representação de dados do algoritmo. É usada uma tabela hash 2-D, juntamente com estruturas ligadas a ela incluindo listas e árvores, fornecendo uma solução de implementação integrada para os três mais importantes passos do algoritmo: inserção das cores da imagem a ser quantizada na tabela hash 2-D, agrupamento dos paralelepípedos de cores e mapeamento das cores originais nos seus respectivos níveis de quantização para a criação da imagem quantizada.

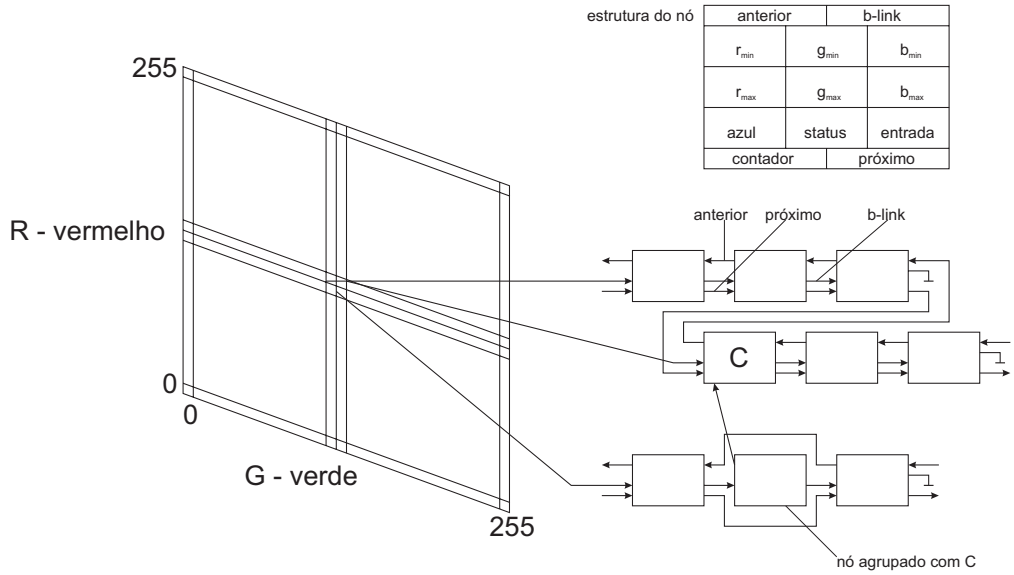


Figura 6.15: Representação de dados, incluindo o vetor de ponteiros 2-D indexado pelas componentes R e G das cores contidas na imagem.

A Figura 6.15 apresenta a representação de dados. Indexamos o vetor bidimensional de ponteiro pelas componentes vermelho e verde das cores de 24-bits da imagem de entrada. Quando uma imagem de entrada é percorrida, cada nova cor origina um novo nó para ser inserido na tabela. A componente azul da cor é armazenada no nó. O nó é inserido na lista encadeada correspondente (se existir alguma) através do ponteiro b-link de forma que a lista esteja sempre em ordem ascendente de acordo com a componente azul da cor. O contador

de pixel é atualizado.

Depois de inserir todas as cores da imagem de entrada, fazemos uma lista duplamente encadeada que consiste de todos os nós ligando as listas individuais uma atrás da outra mantendo os ponteiros b-link intactos. Esses nós duplamente encadeados são os aglomerados iniciais para o processo de aglomeração do algoritmo. Dado um nó arbitrário C, restringimos nossas buscas a vizinhos que sejam possíveis candidatos para agrupamentos dentro de uma pequena região definida pelo tamanho limite do aglomerado corrente. Quando um vizinho é agrupado com C, ele é removido da lista duplamente encadeada, recebe um status especial, se tornando um filho de C usando um dos dois ponteiros recém-liberados. Os seis campos r_{min}, r_{max}, g_{min}, g_{max}, b_{min}, e b_{max} do nó C são modificados para representar o paralelepípedo de cor resultante. Este processo de aglomerados termina quando a lista duplamente encadeada é reduzida para o número de nós desejados. Cada nó na lista é agora a raiz de uma árvore de cores agrupadas no mesmo aglomerado.

Depois que cada aglomerado recebe uma entrada na tabela de cores (do inglês "look-up table") e todas as cores na árvore correspondente ao aglomerado (encontrada dentro da caixa de cor especificada na raiz) recebem o endereço de entrada, percorremos a imagem original uma segunda vez para criarmos a imagem quantizada. Toda cor de 24-bits corresponde a um nó na tabela hash, e o nó contém o valor de quantização para a cor de saída.

Os autores sugerem duas maneiras de selecionar a cor representativa do aglomerado (isto é, seu nível de quantização), ou usando o centro geométrico ou o centróide de cada caixa de cor resultante do processo de quantização.

Capítulo 7

Quantização por Aglomerados Duplos

Baseado em um algoritmo de aglomerados por pares e usando uma heurística hierárquica de agrupamento, como exposto no capítulo anterior, estamos propondo um novo algoritmo de quantização de imagens.

Usamos uma estratégia de otimização local do erro de quantização, gerando níveis de quantização que estão perto do ótimo. De fácil implementação, este algoritmo gera resultados melhores que os demais algoritmos para quantização de imagens, apesar de ser um pouco mais lento que os demais.

Neste capítulo vamos expor o algoritmo proposto bem como a teoria por trás de sua concepção.

7.1 Quantização de um Aglomerado de Cores

Considere o problema de encontrarmos o nível de quantização ótimo associado a um aglomerado de cores. O teorema abaixo fornece o nível de quantização ótimo de um aglomerado de cores.

Teorema Seja $K = \{c_1, c_2, \ldots, c_M\}$ um aglomerado com M cores de um conjunto de cor $C \subset \mathbb{R}^n$ do gamute de uma imagem. O nível de quantização ótimo c do aglomerado K é dado por

$$c = \frac{1}{\sum_i F_i} \sum_{j=1}^{M} F_j c_j, \tag{7.1}$$

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

$$E(K) = \frac{1}{(\sum_k F_k)^2} \sum_{j=1}^{M} F_j || \sum_{i=1}^{M} F_i (c_i \Leftrightarrow c_j) ||^2. \tag{7.2}$$

A primeira parte deste Teorema nos diz que o nível de quantização ótimo de um aglomerado é dado pelo seu centróide. Vamos agora fazer a demonstração do Teorema.

Demonstração Tomemos como a função distância no espaço de cor o quadrado da métrica euclidiana, $d(c, c_j) = ||c \Leftrightarrow c_j||^2$. Usando a equação (2.4) e aproximando a distribuição de probabilidade de ocorrência de uma cor na imagem pelo seu histograma de frequência, o erro de quantização no aglomerado K é dado por

$$E(c) = \sum_{j=1}^{M} F_j \, ||c \Leftrightarrow c_j||^2. \tag{7.3}$$

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

O gradiente de E é dado por

$$\text{grad}(E) = \sum_{j=1}^{M} 2F_j(c \Leftrightarrow c_j). \tag{7.4}$$

Da equação (7.4) obtemos seu ponto crítico c

$$c = \frac{1}{\sum_{i=1}^{M} F_i} \sum_j F_j c_j. \tag{7.5}$$

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

Substituindo o ponto de mínimo c de (7.5) na equação (7.3), obtemos o erro de quantização dado na equação (7.2). 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

$$c = \frac{F_i}{F_i + F_j} c_i + \frac{F_j}{F_i + F_j} c_j. \tag{7.6}$$

O erro de quantização associado é dado por

$$E(c_i, c_j) = \frac{F_i F_j^2 + F_j F_i^2}{(F_i + F_j)^2} \, ||c_i \Leftrightarrow c_j||^2 \tag{7.7}$$

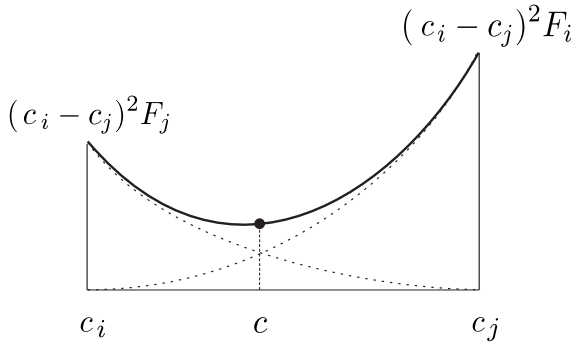


Figura 7.1: Gráfico do erro de quantização.

Uma idéia desse resultado pode ser observada na interpretação geométrica desse Corolário mostrada na Figura 7.1. Da equação (7.3) temos que o erro de quantização é dado por uma função polinomial de segundo grau

$$E_q(c) = F_i(c \Leftrightarrow c_i)^2 + F_j(c \Leftrightarrow c_j)^2. \tag{7.8}$$

Essa função é um arco de parábola obtido da soma dos dois arcos de parábola $F_i(c \Leftrightarrow c_i)^2$ e $F_j(c \Leftrightarrow c_j)^2$, que são mostrados como linhas pontilhadas na Figura 7.1.

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

$$c = \frac{c_i + c_j}{2}. \tag{7.9}$$

7.2 Quantização por Aglomerados Duplos

Nesta seção vamos usar os resultados do Corolário da seção anterior para formular um novo algoritmo para quantização de imagens coloridas. O método consiste de um processo de relaxação que calcula uma sequência de níveis de quantização através de operações locais de agrupamento de aglomerados duplos.

Temos como entrada do algoritmo o conjunto finito C de M cores contidas no gamute da imagem a ser quantizada, $C = \{c_1, c_2, \ldots, c_M\}$. Cada cor c_i tem uma frequência de ocorrência na imagem $F_i = F(c_i)$. Associamos um erro de quantização acumulado $E_A(c_i)$ à cada cor c_i. Inicialmente este erro é 0.

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

1. Calcule o histograma de frequência da imagem.

2. Usando a equação (7.7), calcule o erro de quantização $E(c_i, c_j)$ resultante da combinação de cada par de cor $\{c_i, c_j\}$ do conjunto de cores de entrada.

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 (7.6) 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 resulta em um conjunto de cores quantizadas C' com $M \Leftrightarrow 1$ cores. A frequência de ocorrência $F(c_{K_0})$ da cor c_{K_0} é dada pela soma das frequências 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

$$E_A(c_{K_0}) = E(c_i, c_j) + E_A(c_i) + E_A(c_j). \tag{7.10}$$

6. Calcule o erro de quantização para todos os aglomerados de cores $\{c_k, c_l\}$, 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

$$q(c) = c'_i \Longleftrightarrow d(c, c'_i) \leq d(c, c'_j), \tag{7.11}$$

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

Figura 7.2: Quantização por Aglomerados Duplos: (esquerda) quantização para 8 bits, (direita) quantização para 4 bits.

A Figura 7.2 (esquerda) mostra a imagem das araras quantizada para 8 bits utilizando o algoritmo de quantização por aglomerados duplos. Na Figura 7.2 (direita) mostramos a mesma imagem quantizada para 4 bits com o mesmo algoritmo.

7.2.1 Melhoria no Algoritmo

O processo de agrupar as cores de uma imagem por pares na quantização faz com que percamos a correlação espacial das cores no gamute na imagem. Podemos minimizar esta perda acrescentando um passo adicional no final do algoritmo de quantização por aglomerados duplos: após o cálculo das células de quantização, recalculamos o nível de quantização em cada célula utilizando a equação (7.1).

7.3 Exemplo de uma Execução 2-D do Algoritmo

Na Figura 7.3 (esquerda) temos a arara projetada no plano RG, ou seja, toda componente azul da imagem é 0. Usaremos para exemplo de uma execução do algoritmo de quantização por aglomerados duplos essa figura da arara projetada no plano RG porém já pré-quantizada para 16 cores, como mostrada na Figura 7.3 (direita). Faremos uma quantização dessa imagem para 4 cores.

Figura 7.3: Arara projetado no plano RG (esquerda). Arara projetada no plano RG e pré-quantizada para 16 cores (direita).

A Figura 7.4 mostra a distribuição das cores presentes na imagem da Figura 7.3 (direita) no plano RG. O diâmetro de cada círculo colorido é proporcional ao número de ocorrências de sua cor na imagem. Seu histograma de frequência pode ser visto na Figura 7.5. No eixo horizontal temos as cores presentes na imagem e a altura de cada barra colorida é proporcional à frequência de ocorrência de sua cor na imagem.

As cores presentes no histograma da imagem serão o conjunto de cores de entrada do algoritmo de quantização. A partir desse conjunto de cores de entrada calculamos o erro de quantização $E(c_i, c_j)$ gerado ao combinarmos cada uma dessas cores, duas a duas, e substituí-las pelo nível de quantização ótimo do aglomerado originado dessa combinação, como dado na equação (7.7).

Depois de calculado todos os erros de quantização de cada aglomerado duplo $\{c_i, c_j\}$, devemos encontrar o aglomerado duplo $K_0 = \{c_i, c_j\}$ que minimiza o erro de quantização $E(c_i, c_j)$ já calculado anteriormente. Na Figura 7.6 vemos o par de cores que ao serem combinadas resultam em uma redução no erro de quantização.

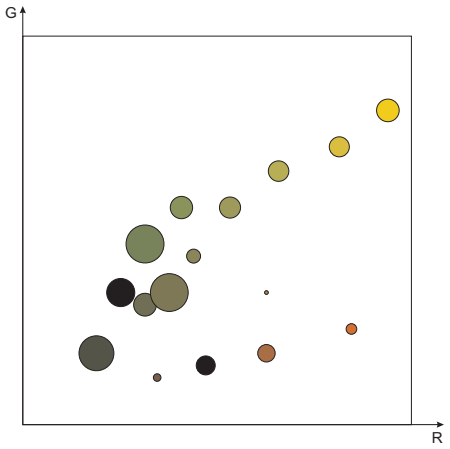


Figura 7.4: Distribuição das cores presentes na imagem que será quantizada no plano RG.

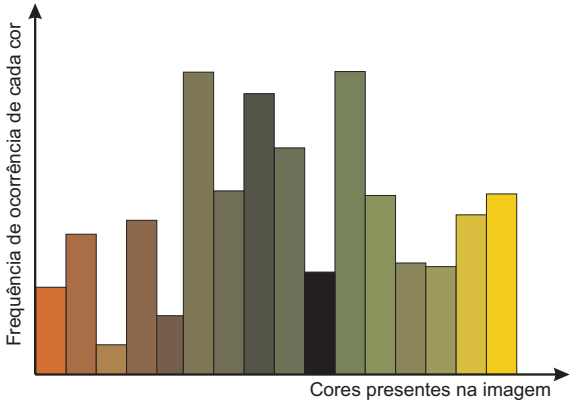


Figura 7.5: Histograma de frequência da imagem que será quantizada.

Após encontrado o par de cores que minimiza o erro de quantização, devemos encontrar o nível de quantização associado a este aglomerado duplo, que é dado pela equação (7.6). Na Figura 7.7 vemos uma ilustração desse nível de quantização, que é dado pelo ponto de mínimo da função do erro de quantização. A cor desse nível de quantização tem uma frequência de ocorrência que é a soma da ocorrência das duas cores que a originou.

Assim, substituimos este par de cores pelo seu respectivo nível de quantização, atualizamos sua nova freqüência de ocorrência na imagem e calculamos o erro de quantização acumulado dessa nova cor (equação (7.10)). A essa altura, já temos menos uma cor no conjunto de cores quantizadas da imagem. Recalculamos os erros de quantização para os novos pares de aglomerados. Este novo conjunto de cores é usado como entrada para uma nova iteração do algoritmo. Na Figura 7.8 (A) até (L) temos uma execução completa do algoritmo de quantização por aglomerados duplos para quantizarmos a imagem da Figura 7.3 (direita) de 16 cores para 4 cores (as setas indicam as cores que são escolhidas para serem combinadas).

Ao final de execução do algoritmo temos os níveis de quantização gerados pelo algoritmo,

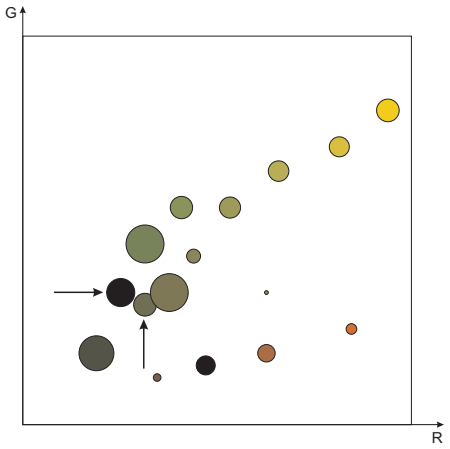


Figura 7.6: Par de cores escolhidas para serem combinadas em um aglomerado duplo.

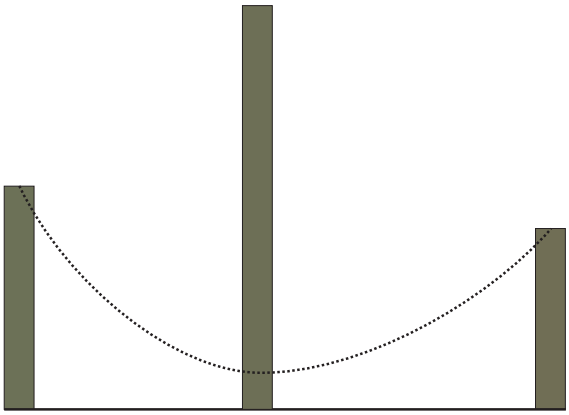


Figura 7.7: Nível de quantização ótimo do aglomerado duplo escolhido e suas respectiva frequência de ocorrência.

como mostrados na Figura 7.9.

Devemos agora mapear as cores da imagem original em seus respectivos níveis de quantização. Isso é feito procurando para cada cor da imagem o nível de quantização que melhor representa esta cor. Assim, devemos encontrar o nível de quantização mais próximo na cor que estamos mapeando. O resultado desse mapeamento são as células de quantização gerados pelo algoritmo de quantização por aglomerados duplos que, no caso, nos fornece o diagrama de Voronoi dos níveis de quantização, como pode ser visto na Figura 7.10. Nesta figura temos os níveis de quantização representados por círculos maiores e as cores da imagem original que serão quantizadas para o seu respectivo nível de quantização, ou seja, todas as cores contidas em um determinada célula são mapeadas no nível de quantização desta célula.

O resultado da quantização da arara no plano RG com 16 para 4 cores pode ser vista na Figura 7.11.

7.4 Outras Estratégias de Quantização

A estratégia do algoritmo de quantização por aglomerados duplos consiste em fazermos um agrupamento ótimo de aglomerados de duas cores que constituem um simplex unidimensional (c_i, c_j). A otimalidade deste procedimento de agrupamento é garantida pelo Corolário apresentado.

Podemos ter diferentes abordagens para este método de quantização, usando a mesma estratégia, consistindo em agrupar aglomerados de cores constituídos de simpliciais de dimensões maiores (triângulos ou tetraedros). Uma triangulação do espaço de cor deve ser feita no sentido de obtermos os aglomerados para serem agrupados.

Certamente, os resultados serão melhores se usarmos simpliciais de dimensões maiores. Isso ocorre porque eles induzem uma correlação espacial melhor no cálculo do erro de quantização. Infelizmente, a complexidade computacional aumenta com a dimensão do simplicial, e heurísticas melhores devem ser encontradas para tornarmos esta estratégia viável.

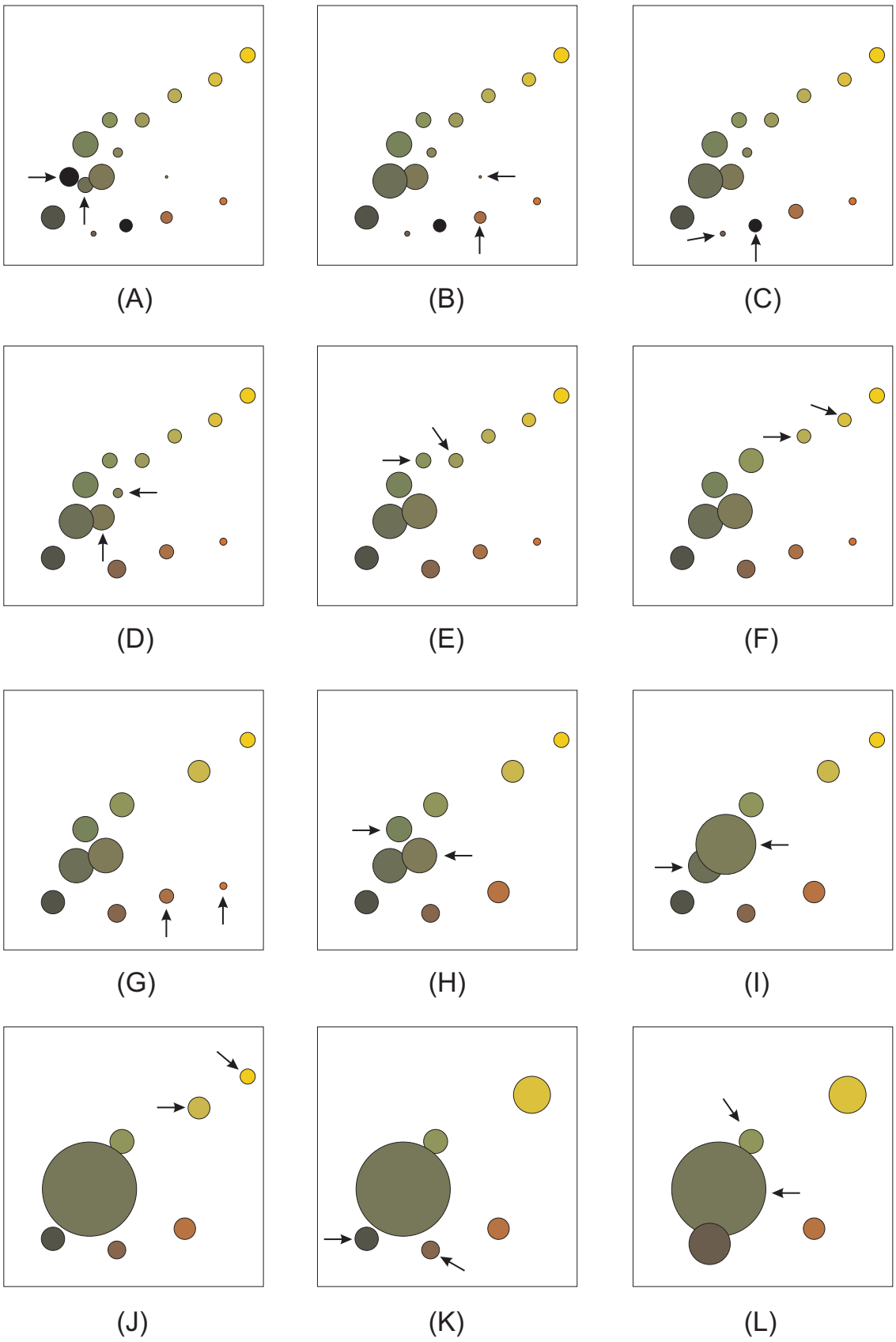


Figura 7.8: Todos os passos da execução do algoritmo para quantizar a imagem para 4 cores.

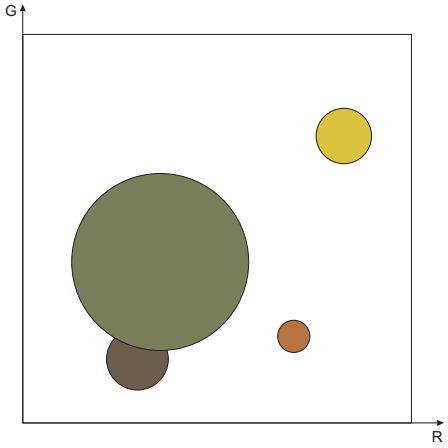


Figura 7.9: Níveis de quantização gerados pelo algoritmo de quantização por aglomerados duplos.

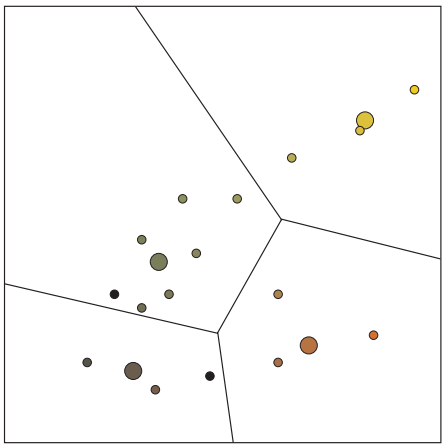


Figura 7.10: Células de quantização geradas pelo algoritmo de quantização por aglomerados duplos.

Figura 7.11: Arara no plano RG quantizada para 4 cores.

Capítulo 8

Conclusões

Neste capítulo faremos um estudo comparativo entre os diversos métodos estudados e implementados.

Começaremos com uma comparação visual dos resultados, onde veremos imagens quantizadas com os algoritmos apresentados e seus respectivos erros de quantização. Uma comparação numérica dos erros de quantização também será alvo de estudo. Nesta seção, mostraremos uma outra imagem de teste, uma cena gerada por computador que também será usada para compararmos os algoritmos. O último alvo de comparação será o tempo de execução dos algoritmos.

Apresetaremos algumas propostas de trabalhos futuros bem como outras possibilidades de utilização para o algoritmo de quantização por aglomerados duplos.

8.1 Comparação entre Algoritmos para Quantização de Imagens

Nesta seção faremos uma comparação entre alguns dos algoritmos para quantização de imagens apresentados anteriormente. Os seguintes algoritmos foram utilizados para esta comparação: populosidade, quantização escalar uniforme, corte mediano, divisão pela variância, quantização por octree, K-means local e quantização por aglomerados duplos.

Faremos três tipos de comparação: uma comparação visual dos resultados obtidos ao quantizarmos a imagem da arara; uma comparação numérica, onde apresentaremos o erro de quantização gerado por cada algoritmo para as imagens quantizadas; e uma comparação do tempo de execução de cada algoritmo.

8.1.1 Comparação Visual

No lado esquerdo das Figuras 8.1 e 8.2 vemos os resultados de quantizações da imagem da arara para para 256 e 16 cores respectivamente, utilizando os diversos algoritmos que vamos comparar. No lado direito temos as imagens dos erros de quantização normalizados de cada imagem. Como esta imagem é gerada? Primeiro calculamos os erros de quantização de cada

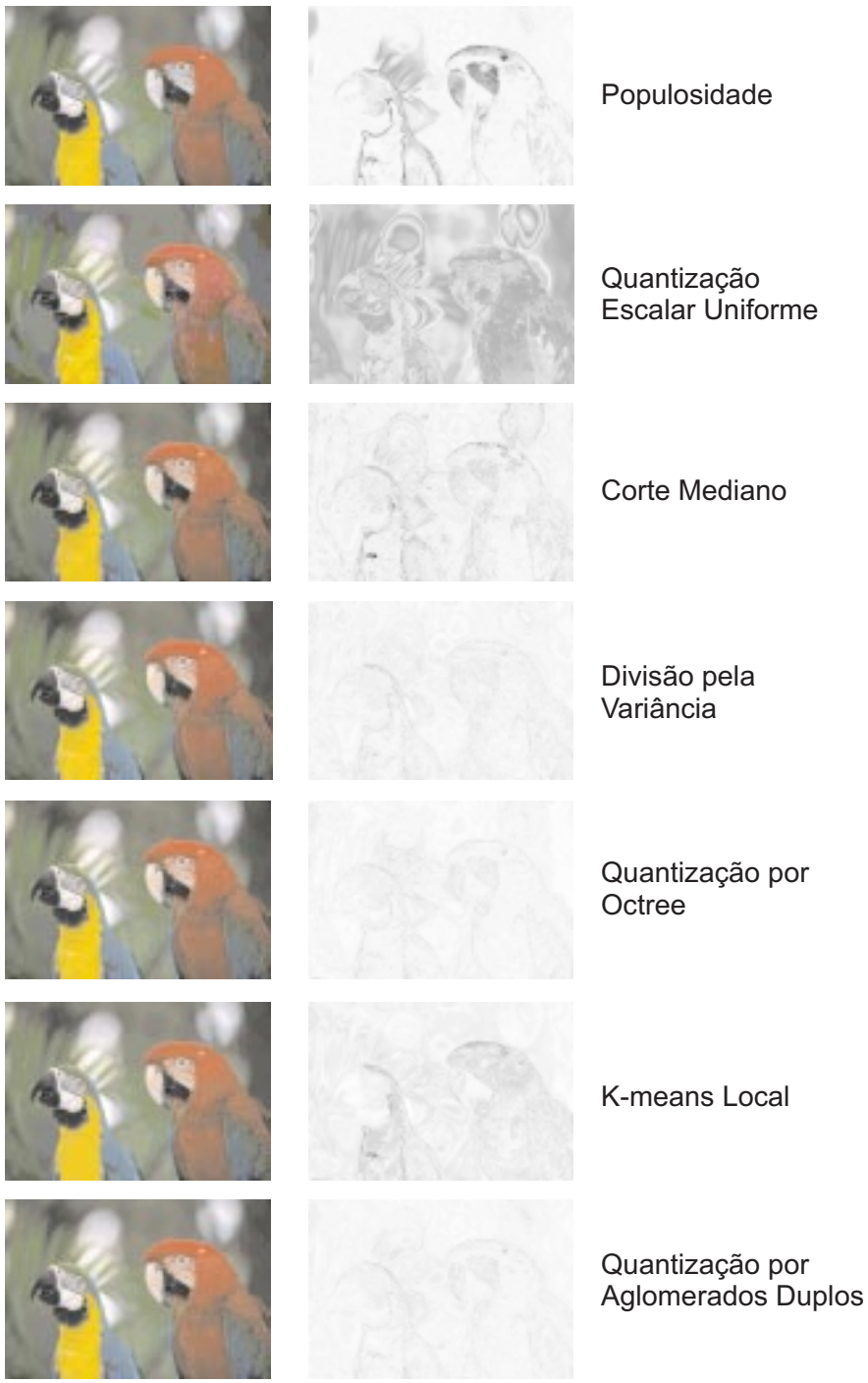


Figura 8.1: Quantização das araras para 256 cores: (esquerda) imagem quantizada, (direita) erro de quantização.

imagem, o maior erro é normalizado para 1, e todos os demais valores são multiplicados pela mesma razão, de modo a termos imagens normalizadas onde podemos fazer comparações entre

Figura 8.2: Quantização das araras para 16 cores: (esquerda) imagem quantizada, (direita) erro de quantização.

os erros de quantização gerados por cada algoritmo. Mas, para fins de melhor visualização, transformamos cada imagem de erro de quantização em seu negativo, assim, valores mais

escuros na imagem significam erros maiores.

Como era de se esperar, os priores resultados são obtidos com os algoritmos da populosidade e da quantização escalar uniforme. O algoritmo da populosidade é o que apresenta maiores erros de quantização, que pode ser verificado por áreas bastante escuras nas imagens dos erros de quantização. O algoritmo da quantização escalar uniforme apresenta muito erro de quantização porém distribuído por toda a imagem, ao contrário da populosidade, onde o erro é grande também mas concentrado em áreas específicas.

Quando quantizamos para 256 cores, quase todos os algoritmos testados apresentam bons resultados visuais, menos os algoritmos da populosidade e da quantização escalar uniforme. Os algoritmos do corte mediano e de K-means local apresentam algumas áreas onde o erro de quantização está mais concentrado, apesar deste erro ser pequeno. Já os demais algoritmos, ou seja, divisão pela variância, quantização por octree e quantização por aglomerados duplos tendem a distribuir equivalentemente o erro de quantização por toda a imagem. Assim, nestes algoritmos além do erro de quantização ser menor que nos demais, ele se apresenta mais bem distribuído por toda imagem.

Na quantização para 16 cores temos praticamente a mesma situação da quantização para 256 cores. Os piores resultados são obtidos pelos algoritmos da populosidade e da quantização escalar uniforme, sendo que agora o resultado ainda é pior, onde verificamos a presença de cores que fogem totalmente da imagem original. Os demais algoritmos apresentam resultados melhores que os da populosidade e da quantização escalar uniforme, ou seja, as cores escolhidas por estes algoritmos se aproximam mais das cores presentes na imagem original. Como na quantização para 256, os algoritmos da divisão pela variância, quantização por octree e quantização por aglomerados duplos apresentam resultados melhores que os algoritmos do corte mediano e de K-means local. Isso pode ser comprovado tanto nas imagens quantizadas, onde as cores escolhidas estão mais próximas perceptualmente das cores originais como nas imagens do erro de quantização, onde este se apresenta em tons mais claros (erro de quantização menores) e mais bem distribuído por toda imagem.

8.1.2 Comparação Numérica

Nas Tabelas 8.1 e 8.2 vemos os erros de quantização por pixel gerado por cada um dos algoritmos que estão sendo comparados para a imagem das araras. Como podemos observar, o algoritmo de quantização por aglomerados duplos apresenta resultados melhores que os demais algoritmos. Na quantização para 256 cores, o algoritmo de quantização por aglomerados duplos apresenta um erro de quantização por pixel por volta de 6% menor que o melhor algoritmo, já na quantização para 16 cores, esse valor aumenta, passando a ser por volta de 8% menor que o melhor resultado obtido pelo melhor dos demais algoritmos. Esse resultados mostram uma tendência nas diversas imagens testadas: para imagens reais digitalizadas, onde temos a presença de ruídos, o algoritmo de quantização por aglomerados duplos é sempre melhor que os demais algoritmos comparados, sendo que, à medida que o número de cores na imagem quantizada diminui, maior se torna a superioridade deste algoritmo com relação aos demais algoritmos comparados.

Para imagens geradas por computador, ou seja, cenas renderizadas onde não temos ruído,

Algoritmo	Erro de quantização por pixel
Populosidade	9.15
Quantização escalar uniforme	29.58
Corte mediano	11.20
Divisão pela variância	7.13
Quantização por octree	6.65
K-means local	9.75
Quantização por aglomerados duplos	6.25

Tabela 8.1: Erro por pixel gerado pela quantização da imagem das araras para 256 cores.

Algoritmo	Erro de quantização por pixel
Populosidade	54.23
Quantização escalar uniforme	104.69
Corte mediano	26.27
Divisão pela variância	20.18
Quantização por octree	21.09
K-means local	24.98
Quantização por aglomerados duplos	18.52

Tabela 8.2: Erro por pixel gerado pela quantização da imagem das araras para 16 cores.

Figura 8.3: Imagem renderizada quantizada com 24 bits: peixe.

o algoritmo de quantização por aglomerados duplos também apresenta ótimos resultados porém, nem sempre ele é o de melhor desempenho. Este fato é explicado pelo fato do algoritmo de quantização por aglomerados duplos fazer uma pré-quantização da imagem original para 15 bits antes da geração do histograma de freqüências. Com esta pré-quantização perdemos informações importantes de cores, principalmente nas áreas onde ocorrem variações gradativas de tons, como é o caso dos degradés, muito comuns em cenas geradas por computador. Usaremos a imagem do peixe mostrada na Figura 8.3 como teste para imagens geradas

por computador. Como podemos notar, esta imagem apresenta uma grande área em degradé, sendo assim, esta imagem é ideal para nossos testes. Não faremos os testes para todos os algoritmos, usaremos somente os de melhor desempenho: divisão pela variância, quantização por octree, K-means local e quantização por aglomerados duplos. Os testes serão feitos com duas imagens, uma delas será o próprio peixe com 24-bits, a segunda imagem será o mesmo peixe porém pré-quantizado para 15-bits. Os resultados do erro de quantização por pixel gerado por esses algoritmos podem ser vistos nas Tabelas 8.3, 8.4, 8.5 e 8.6. As imagens do peixe quantizado para 256 e 16 cores, bem como seus respectivos erros de quantização podem ser vistos nas Figuras 8.4 e 8.5.

Algoritmo	Erro de quantização por pixel
Divisão pela variância	5.44
Quantização por octree	5.21
K-means local	4.82
Quantização por aglomerados duplos	5.37

Tabela 8.3: Erro por pixel gerado pela quantização da imagem do peixe com 24-bits para 256 cores.

Algoritmo	Erro de quantização por pixel
Divisão pela variância	6.70
Quantização por octree	10.01
K-means local	7.42
Quantização por aglomerados duplos	5.37

Tabela 8.4: Erro por pixel gerado pela quantização da imagem do peixe com 15-bits para 256 cores.

Algoritmo	Erro de quantização por pixel
Divisão pela variância	16.26
Quantização por octree	17.69
K-means local	16.13
Quantização por aglomerados duplos	14.03

Tabela 8.5: Erro por pixel gerado pela quantização da imagem do peixe com 24-bits para 16 cores.

Como podemos observar na Tabela 8.3, quando quantizamos a imagem do peixe com 24-bits para 256 cores o algortimo da quantização por aglomerados duplos já não apresenta o melhor resultado, sendo superado pelos algoritmos de k-means loval e de quantização por octree. Porém, usando a imagem já pré-quantizada para 15-bits, novamente o algoritmo

Algoritmo	Erro de quantização por pixel
Divisão pela variância	15.65
Quantização por octree	18.41
K-means local	18.37
Quantização por aglomerados duplos	14.03

Tabela 8.6: Erro por pixel gerado pela quantização da imagem do peixe com 15-bits para 16 cores.

da quantização por aglomerados duplos se apresenta como o de melhor desempenho, como podemos ver na Tabela 8.4. Ou seja, em igualdades de condições o algoritmo de quantização por aglomerados duplos é melhor que os demais.

Já quando quantizamos o peixe para 16 cores, não importa se usamos a imagem com 24-bits ou a pré-quantizada para 15-bits o algoritmo de quantização por aglomerados duplos é sempre melhor que os demais.

Assim, podemos chegar as seguintes conclusões sobre o algoritmo de quantização por aglomerados duplos:

- Para imagens reais digitalizadas ele apresenta melhores resultados que os demais algoritmos comparados.
- Para imagens geradas por computador ele nem sempre apresenta melhores resultados que os demais algoritmos comparados, porém, em igualdades de condições, ou seja, se utilizarmos os outros métodos com imagens pré-quantizadas para 15-bits, este algoritmo apresenta resultados melhores que os demais.
- À medida que o número de cores na imagem quantizada diminui, a superioridade do algoritmo de quantização por aglomerados duplos aumenta. Isso pode ser verificado para qualquer tipo de imagem, não importanto se é real ou gerada por computador.

8.1.3 Comparação dos Tempos de Execução

Entre todos os algoritmos testados, o mais rápido é o do corte mediano e o mais lento é o de quantização por aglomerados duplos.

Para as várias imagens testadas o algoritmo de quantização por aglomerados duplos se mostrou em média 6 vezes mais lento que o corte mediano. Apesar deste tempo não ser muito grande, ele aumenta na medida que o número de cores presentes na imagem aumenta. Para uma imagem com 40.000 cores e uma resolução espacial de 640×480 pixels, o tempo de execução do algoritmo de quantização por aglomerados duplos é, em média, 1 minuto.

8.2 Outras Utilizações para o Algoritmo de Quantização por Aglomerados Duplos

Como a base deste algoritmo é aglomerados, em qualquer problema que se deseje a formação de aglomerados podemos usar as idéias deste algoritmo, bastando para isso adaptarmos o cálculo do erro inserido ao combinarmos os aglomerados e o cálculo do nível de quantização de cada aglomerado para a situação em questão.

Assim, na área de animação de imagens podemos usar essas idéias para simplificar informações de movimentos capturados visto que estas informações ocupam um volume de dados muito grande, podendo ser simplificadas. Podemos utilizar estas idéias em sistemas geográficos de informação para simplificação de dados de terrenos. Em modelagem também podemos utilizar as mesmas idéias para simplificarmos modelos, ideal para quando temos cenas em várias resoluções diferentes.

Como podemos ver, existem inúmeras possibilidades de adaptação do algoritmo de quantização por aglomerados duplos dentro da área de computação gráfica.

8.3 Trabalhos Futuros

O principal objetivo agora é melhorar o tempo de execução do algoritmo de quantização por aglomerados duplos visando assim competir comercialmente com os demais algoritmos que são mais rápidos apesar de não apresentarem tão bons resultados como a quantização por aglomerados duplos. Para isso estão sendo feitos estudos para utilizarmos estruturas de dados geométricas com aplicações em problemas de apredizado geométrico, chamadas Balltree (Omohundro, 1989). Também estamos avaliando a possibilidade de utilizarmos árvores de caminho mínimo (Minimum Spanning Tree) (Thomas H. Cormen & Rivest, 1994) para acelerar o passo de busca pelo melhor par de cor para ser combinado, que é a etapa mais custosa do algoritmo. Outra estratégia que podemos tentar utilizar é a de dividir para conquistar, ou seja, usando estruturas tipo Octree, podemos dividir o cubo RGB em cubos menores em cada um desses cubos aplicar a quantização por aglomerados duplos, combinando depois os resultados obtidos para alcançarmos a quantização final.

Alcançado o objetivo de acelerar o algoritmo, o próximo passo será o de acabar com a limitação de termos de pré-quantizar a imagem para 15-bits antes de executarmos o algoritmo. Para isso temos de pensar em uma estrutura inteligente para armazenarmos o histograma de cor da imagem de modo eficiente e que não ocupe muito espaço em memória.

Finalmente, tentar aplicar estas idéias em outras áreas da computação gráfica também um objetivo para trabalhos futuros.

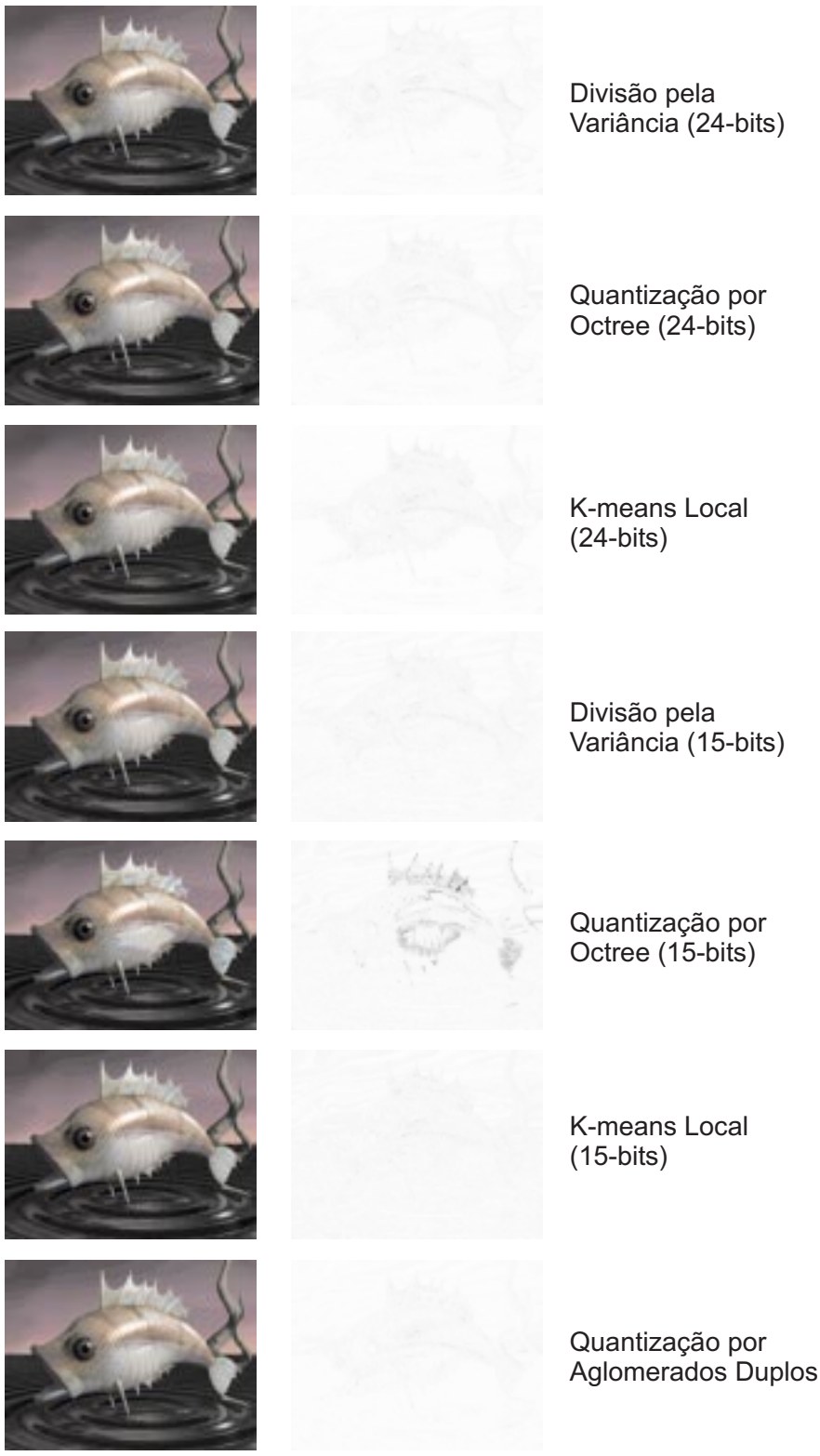


Figura 8.4: Quantização do peixe para 256 cores: (esquerda) imagem quantizada, (direita) erro de quantização.

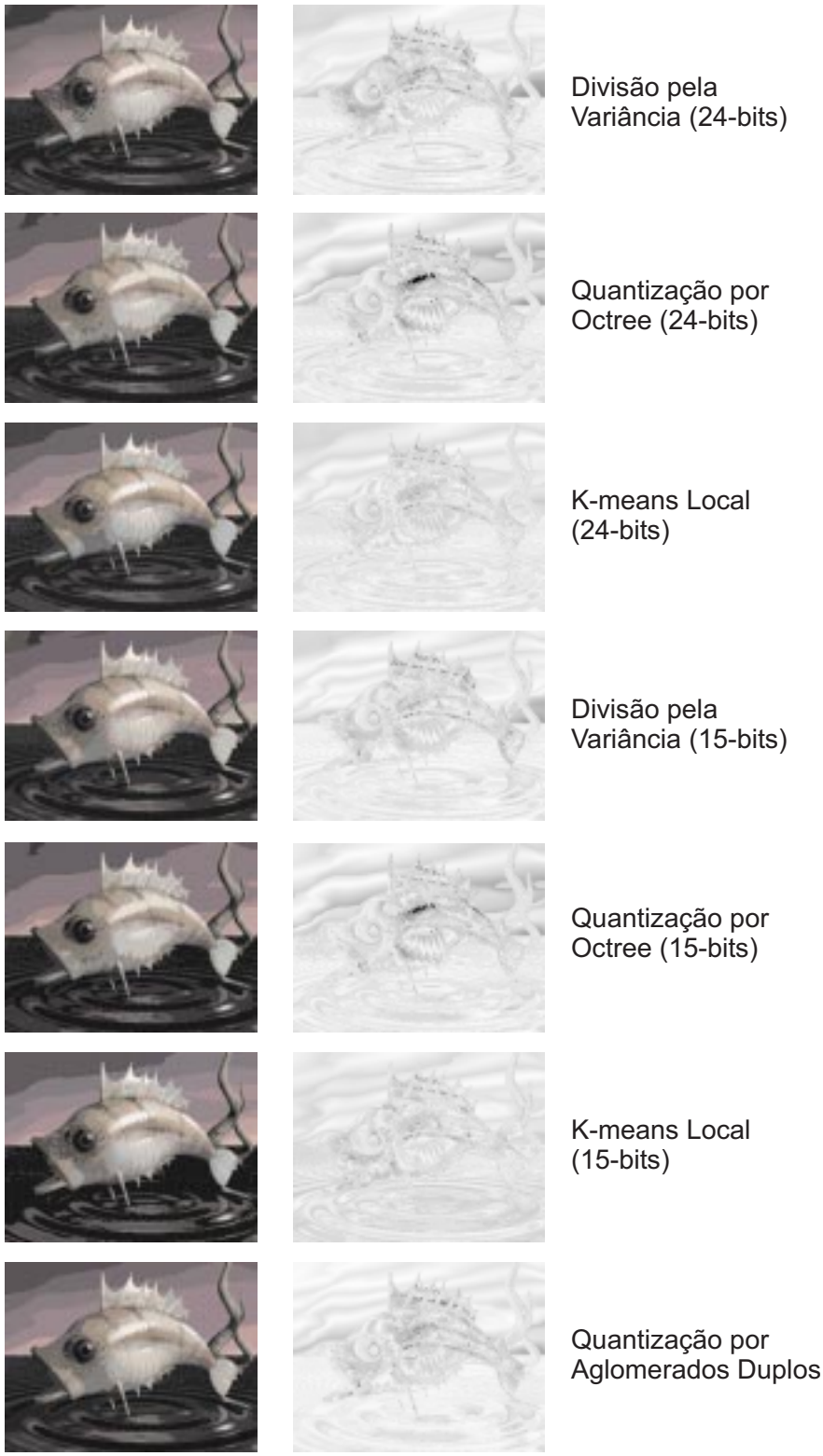


Figura 8.5: Quantização do peixe para 16 cores: (esquerda) imagem quantizada, (direita) erro de quantização.

Referências Bibliográficas

Akarun, L.; Ozdemir, D. & Yalcin, O. 1996. Modified Quantisation Algorithm for Dithering of Colour Images. *Electronics Letters*, **32**(13), 1185–1194.

Anderberg, M. R. 1973. *Cluster Analysis for Applications*. New York: Academic Press.

Ashdown, I. 1994. *Radiosity: A Programmer's Perspective*. New York, NY: John Wiley and Sons.

Ashdown, I. 1995. Octree Color Quantization. *C/C++ Users Journal*, **13**(3), 31–44.

Atkins, C. B.; Flohr, T. J.; Hilgenberg, D. P.; Bouman, C. A. & Allebach, J. P. 1994. Model-Based Color Image Sequence Quantization. *Human Vision, Visual Processing, and Digital Display IV (1993)*, **SPIE 2179**, 310–317.

Balasubramaian, R. & Allebach, J. 1991a. A New Approach to Palette Selection for Color Images. *Human Vision, Visual Processing, and Digital Display III (1991)*, **SPIE 1453**, 58–69.

Balasubramaian, R. & Allebach, J. 1991b. A New Approach to Palette Selection for Color Images. *Journal of Imaging Technology*, **17**(6), 284–290.

Balasubramaian, R.; Bouman, C. A. & Allebach, J. P. 1993. Sequential Scalar Quantization of Color Images. *Imaging Science and Technology Annual Conference 1993 (IS&T '93)*, May, 97–100.

Balasubramaian, R.; Allebach, J. & Bouman, C. A. 1994a. Color-Image Quantization with Use of a Fast Binary Splitting Technique. *Journal of the Optical Society of America*, **11**(11), 2777–2786.

Balasubramaian, R.; Bouman, C. A. & Allebach, J. 1994b. New Results in Color Image Quantization. *Image and Video Processing II (1994)*, **SPIE 2182**, 34–42.

Balasubramaian, R.; Bouman, C. A. & Allebach, J. 1994c. Sequential Scalar Quantization of Color Images. *Journal of Electronic Imaging*, **3**(1), 45–59.

Betz, M. 1993. VGA Palette Mapping Using BSP Trees. *Dr. Dobb's Journal*, **18**(7), 28–36, 94.

Bouman, C. 1989 (October). *Hierarchical Modeling and Processing of Images*. Ph.D. thesis.

Bouman, C. & Orchard, M. 1989. Color Image Display with a Limited Palette Size. *Visual Communications and Image Processing IV (1989)*, **SPIE 1199**, 522–533.

Bouman, C. & Orchard, M. 1991. Color Quantization of Images. *IEEE Transactions on Signal Processing*, **39**(12), 2677–2690.

Braudaway, G. 1985. A Procedure for Optimum Choice of a Small Number of Colors from a Large Color Palette for Color Imaging. *Internal report RC 11367 (#51227)*, September 16,.

Brucker, P. 1977. *On the Complexity of Clustering Problems*. New York: Springer Verlag.

Buckley, R. R. 1993. The Quantization of the CIE Uniform Color Spaces Using Cbic Lattices. *Color 93*, C21–1 – C21–2.

Budge, Scott Erza. 1985. *Vector Quantization of Color Digital Images Using Product Codes*. M.Sc. thesis.

Burger, P. & Gillies, D. 1989. *Interactive Computer Graphics: Functional, Procedural, and Device- Level Methods*. Wokingham, UK: Addison-Wesley.

Campbell, G.; DeFanti, T. A.; Frederiksen, J.; Joyce, S. A.; Leske, L. A.; Lindberg, J. A. & Sandin, D. J. 1986. Two Bit/Pixel Full Color Encoding. *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, **20**(4), 215–223.

Chaddha, Navid; Tan, Wee-Chiew & Meng, Teresa H. Y. 1994. Color Quantization of Images Based on Human Visual Perception. *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-94)*, **5 (Image and Multidimensional Signal Processing)**(April), V89–V92.

Chan, S. S. & Nerheim-Wolfe, R. 1994. An Empirical Assessment of Selected Color-Quantizing Algorithms. *Human Vision, Visual Processing, and Digital Display IV (1993)*, **SPIE 2179**, 298–309.

Chang, L.-W. & Chang, H.-H. 1993. Fast Algorithm for Color Vector Quantization. *Applications of Digital Image Processing XVI (1993)*, **SPIE 2028**(July), 106–117.

Chang, L.-W. & Liu, T.-S. 1993. 24-bit Color Image Quantization for 8-bit Color Display Based on Y-Cr-Cb. *Applications of Digital Image Processing XVI (1993)*, **SPIE 2028**(July), 94–105.

Chau, Wing-Ki Wilkin. 1992. *Additive Color Quantization Algorithm*. M.Sc. thesis.

Chen, X.; Kothari, R. & Klinkhachorn, P. 1993. Reduced Color Image Based on Adaptive Color Selection Using Neural Networks. *World Congress on Neural Networks*, **Vol. I**(July 11-15,), 555–558.

Christiansen, Kevin & Barrett, William. 1993. An Improved Method for Color Quantization. *Page 461 of:* Storer, James A. & Cohn, Martin (eds), *DCC '93 Data Compression Conference*. Los Alamitos, CA: IEEE Computer Society Press.

Clark, Dean. 1995. The Popularity Algorithm. *Dr. Dobb's Journal*, **20**(7), 121–127.

Clark, Dean. 1996. Color Quantization Using Octrees. *Dr. Dobb's Journal*, **21**(1), 54–57, 102–104.

Crissey, James Melvin. 1991. *An Iteratively Interpolative Vector Quantization Algorithm for Color Image Data Compression*. M.Sc. thesis.

Dekker, A. 1993 (October). *Optimal Colour Quantization Using Kohonen Neural Networks*. Technical Report TR10/93. Dept. of Information Systems and Computer Science, National University of Singapore, Kent Ridge, Singapore.

Dekker, A. 1994. Kohonen Neural Networks for Optimal Colour Quantization. *Network: Computation in Neural Systems*, **5**, 351–367.

Dixit, Sudhir S. 1991. Quantization of Color Images for Display/Printed on Limited Color Output Devices. *Computers and Graphics*, **15**(4), 561–568.

Dyck, Robert E. Van. 1992a. *Subband-vector Quantization Coding of Color Images with Perceptually Optimal Bit Allocation*. Ph.D. thesis.

Dyck, Robert E. Van. 1992b. *Subband-Vector Quantization Coding of Color Images with Perceptually Optimal Bit Allocation*. Ph.D. thesis, North Carolina State University.

Feder, T. & Greene, D. H. 1988. Optimal Algorithms for Approximate Clustering. *Procedings of the 20th Annual ACM Symposium on Theory Computational*, 434–444.

Feng, Y. S. & Nasrabadi, N. M. 1991. Dynamic Address-Vector Quantization of RGB Colour Images. *IEEE Proceedings*, **138**(4), 225–234.

Firlani, J. L.; McMillan, L. & Westover, L. 1994. Adaptive Color Selection Algorithm for Motion Sequences. *Proc. ACM Multimedia '94*, 341–347.

Fiume, E. & Ouellette, M. 1989a. On distributed, probabilistic algorithms for computer graphics. *Proceedings of Graphical Interface '89*, **8**, 211–218.

Fiume, E. & Ouellette, M. 1989b. On Distributed, Probablistic Algorithms for Computer Graphics. *Graphics Interface '89*, 211–218.

Fletcher, P. 1989. Adaptive Selection of Optimised Colour Subsets for Displaying 24-bit Colour Images on 8-bit Graphics Workstations. *Image Processing and the Impact of New Technologies Proceedings*, December, 193–196.

Fletcher, P. 1991. A SIMD Parallel Colour Quantization Algorithm. *Computers and Graphics*, **15**(3), 365–373.

Flohr, T. J.; Kolpatzik, B. W.; Balasubramanian, R.; Carrara, D. A.; Bouman, C. A. & Allebach, J. P. 1993. Model Based Color Image Quantization. *Human Vision, Visual Processing, and Digital Display IV (1993)*, **SPIE 1913**, 270–281.

Frederick, Stephen M. 1992. *Quantization of Color Images Using the Modified Median Cut Algorithm.* M.Sc. thesis.

Galli, I.; Mecocci, A. & Cappellini, V. 1994. Improved Colour Image Vector Quantisation by Means of Self-Organizing Neural Networks. *Electronics Letters*, **30**(4), 333–345Liu.

Gallo, Andrew L. 1993. *Color Image Cartooning for Augmentative and Alternative Communication Systems Through Small Palette Quantization.* M.Sc. thesis.

Gentile, R. 1989 (August). *Color Image Processing for High Quality Reproduction Based on Uniform Color Spaces.* Ph.D. thesis.

Gentile, R.; Allebach, J. & Walowit, E. 1990a. Quantization and Multilevel Halftoning of Color Images for Near- Original Image Quality. *Journal of the Optical Society of America*, **7**(6), 1019–1026.

Gentile, R.; Allebach, J. & Walowit, E. 1990b. Quantization of Color Images Based on Uniform Color Spaces. *Journal of Imaging Technology*, **16**(1), 11–21.

Gentile, R.; Walowit, E. & Allebach, J. P. 1994. Quantization and Multilevel Halftoning of Color Images for Near Original Image Quality. *Human Vision, Visual Processing, and Digital Display II (1990)*, **SPIE 1249**, 249–260.

Gervautz, M. & Purgathofer, W. 1988. A Simple Method for Color Quantization: Octree Quantization. *Pages 219–231 of:* Magnenat-Thalmann, N. & Thalmann, D. (eds), *New Trends in Computer Graphics.* New York, NY: Springer-Verlag.

Gervautz, M. & Purgathofer, W. 1990. A Simple Method for Color Quantization: Octree Quantization. *Pages 287–293 of:* Glassner, A. S. (ed), *Graphics Gems.* Cambridge, MA: Academic Press Professional.

Goldberg, N. 1991. Colour Image Quantization for High Resolution Graphics Display. *Image and Vision Computing*, **9**(1), 303–312.

Gomes, Jonas & Velho, Luiz. 1995a. Abstraction paradigms for computer graphics. *The Visual Computer*, **11**(5), 227–239.

Gomes, Jonas & Velho, Luiz. 1995b. *Computação Gráfica: Imagem.* IMPA-SBM.

Gomes, Jonas & Velho, Luiz. 1997. *Image Processing for Computer Graphics.* New York: Springer Verlag.

Gomes, Jonas; Costa, Bruno; Darsa, Lucia & Velho, Luiz. 1996. Graphical Objects. *The Visual Computer*, **12**, 269–282.

Gong, Y.; Zen, H.; Ohsawa, Y. & Sakauchi, M. 1992. A Color Video Image Quantization Method with Stable and Efficient Color Selection Capability. *Pattern Recognition '92 (International Conference on Pattern Recognition) Volume 3: Image, Speech and Signal Analysis*, **3**(August), 33–36.

Gonzales, T. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Thoeretical Computer Science*, **38**, 293–306.

Gotsman, C. 1994. Dynamic Color Quantization of Video Sequences. *Proceedings of Computer Graphics International '94.*

Gruber, Diane. 1996. Color Reduction for Windows Games. *Visual Developer*, **7**(1), 88–91.

Hecht-Nielsen, Robert. 1990. *Neurocomputing.* New York: Adison-Wesley.

Heckbert, P. S. 1980 (May). *Color Image Quantization for Frame Buffer Display.* B.S. thesis. Architecture Machine Group, MIT.

Heckbert, P. S. 1982. Color Image Quantization for Frame Buffer Display. *ACM Computer Graphics (ACM SIGGRAPH '82 Proceedings)*, **16**(3), 297–307.

Hernandez, Julio A. 1996. *Color Image Compression Using Vector Quantization and Neural Networks.* M.Phil. thesis, University of Puerto Rico, Mayaguez Campus.

Houle, G. 1985. Quantization des Images Couleurs pour Appareils Graphiques. *INRS-Telecommunications Technical Report 85-37.*

Houle, G. & Dubois, E. 1991. Quantization of Color Images for Display on Graphics Terminals. *Proc. IEEE Global Telecommunications Conference (GLOBE_COM '86)*, December, 284–297.

Ignasiak, K.; Lukaszewicz, W. & Skarbek, W. 1994. Colour Quantization Methods. *Pages 451–456 of: Machine Graphics and Vision - Special Issue: Proceedings of the Third Conference on Computer Graphics and Image Processing (GKPO '94)*, vol. 3. Institute of Computer Science, ul. Ordona 21, 01-237 Warszawa, Poland.

Iverson, V. S. & Riskin, E. A. 1993. A Fast Method for Combining Palettes of Color Quantized Images. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing '93*, **5**, 317–320.

Jacobson, N. & Bender, W. 1989. Strategies for Selecting a Fixed Palette of Colors. *Human Vision, Visual Processing, and Digital Display (1989)*, **SPIE 1077**, 333–341.

Jain, A. K. & Dubes, R. C. 1988. *Algorithms for Clustering Data.* Englewood Cliffs - NJ: Prentice-Hall.

Jain, A. K. & Pratt, W. K. 1972. Color Image Quantization. *National Telecommunications Conference 1972 Record*, December.

Jari A. Kangas, Teuvo A. Kohonen & Laaksonen, Jorma T. 1990. Variants of self-organizing maps. *IEEE Transaction on Neural Networks*, **1**(1), 93–99.

Jones, Stephen Charles. 1992. *Image-sequence Dependent Color Quantization Used for the Animation of Computational Fluid Simulation Data.* M.Sc. thesis.

Joy, G. & Xiang, Z. 1993. Center-Cut for Color-Image Quantization. *The Visual Computer*, **10**(1), 62–66.

Joy, G. & Xiang, Z. 1996. Reducing False Contours in Quantized Color Images. *Computers and Graphics*, **20**(2), 231–242.

Jun, J.-A.; Kim, K.-B. & Cha, E.-Y. 1994. Vector Quantization Using an Improved Competitive Learning Neural Network for Color Images. *Simulators XI (Eleventh Annual Simulators Conference)*, **26**(3), 466–474.

Kopec, Thomas E. 1985. *Adaptive Quantization of Color Images.* M.S.E.C.E. thesis.

Kruger, A. 1994. Median-Cut Color Quantization. *Dr. Dobb's Journal*, **19**(9), 46–54, 91–92.

Kurz, B. J. 1983. Optimal Color Quantization for Color Displays. *IEEE Computer Vision and Pattern Recognition Proceedings*, January, 217–224.

Lee, W. F. & Chan, C. K. 1992. Dynamic Finite-State Vector Quantisation of Colour Images. *Electronics Letters*, **28**(5), 460.

Lehar, A. F. & Stevens, R. J. 1984. High-Speed Manipulation of the Color Chromaticity of Digital Images. *IEEE Computer Graphics and Applications*, **4**(2), 34–39.

Lindbloom, Bruce J. 1989. Accurate Color Reproduction for Computer Graphics Applications. *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, **23**(3), 117–126.

Lindley, C. 1992. *Practical Ray Tracing in C.* New York, NY: John Wiley and Sons.

Liu, Tsann-Shyong & Chang, Long-Wen. 1994. Greedy Tree Growing for Color Image Quantization. *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-94)*, **5 (Image and Multidimensional Signal Processing)**(April), V97–V100.

Liu, Tsann-Shyong & Chang, Long-Wen. 1995. Fast Color Image Quantization with Error Diffusion and Morphological Operations. *Signal Processing*, **43**(3), 293–302.

M. T. Ko, R. C. T. Lee & Chang, J. S. 1990. An Optimal Approximation Algorithm for the Rectilinear m-center Problem. *Algorithmica*, **5**, 341–352.

Marcinek, W. 1995. On Color Quantization: Relations and Parastatistics. *International Journal of Modern Physics*, **10**(10), 1465–1481.

McColl, R. W. 1987. *Quantisation of Digitised Colour Images.* Department of Computer Science, University of Warwick.

Megiddo, N. & Supowit, K. J. 1984. On the Complexity of Some Commom Geometric Lacation Problems. *SIAM Journal on Computer*, **13**, 182–196.

Mutz, A. H. 1994a. Effect of Gamt Range Upon Color Quantization During JPEG Coding. *Digest of Technical Papers*, **25**, 891–910. ISSN 0097-966X.

Mutz, A. H. 1994b. Effect of Gamut Range Upon Color Quantization During JPEG Coding. *Digest of Technical Papers (The IDEA Symposium: Information Display, Evolution and Advances)*, **25**, 891.

Niwa, Y.; Tajima, J. & Numata, K. 1995. Software Implementation of a Real Time Color Quantization for Full-Color Video Quality. *Digest of Technical Papers*, 284–294. ISSN 0747-947500.

Omohundro, Stephen M. 1989 (November). *Five Balltree Construction Algorithms.* Tech. rept. TR-89-063. International Computer Science Institute, Berkeley, CA.

Oomen, J. A.; Schuurbiers, J. E. H.; Lehmann, K. G. & Slager, C. J. 1996. Color Quantization in Angioscopic Images. *Developments in Cardiovascular Medicine*, **186**, 367–382.

Orchard, M. T. & Bouman, C. A. 1991. Color Quantization of Images. *IEEE Transactions on Signal Processing*, **39**(12), 2677–2690.

Paeth, A. W. 1990. Mapping RGB Triples Onto Four Bits. *Pages 233–245, 718 of:* Glassner, A. S. (ed), *Graphics Gems.* Cambridge, MA: Academic Press Professional.

Paeth, A. W. 1991. Mapping RGB Triples Onto 16 Distinct Values. *Pages 143–146 of:* Arvo, J. (ed), *Graphics Gems II.* Cambridge, MA: Academic Press Professional.

Patrick, E. A.; Anderson, D. R. & Bechtel, F. K. 1968. Mapping Multidimensional Space to One Dimension for Computer Output Display. *IEEE Transactions on Computers*, October, 949–953.

Pei, S.-C. & Cheng, C.-M. 1995. Dependent Scalar Quantization of Color Images. *IEEE Transactions on Circuits and Systems for Video Technology*, **5**(2), 124–132.

Peterson, H. A.; Peng, H.; Morgan, J. H. & Pennebaker, W. B. 1990. *Quantization of Color Image Components in the DCT Domain.* Society of Photo-optical Instrumentation Engineers.

Po, L.-M. & Tan, W.-T. 1994. Block Address Predicitive Colour Quantisation Image Compression. *Electronics Letters*, **30**(2), 120—135.

Poe, Daryl Thomas. 1986. *One- and Two-pass Color Quantization Methods in Computer Graphics.* M.Sc. thesis.

Pomerantz, D. 1990. A Few Good Colors. *Computer Language*, **7**(8), 32–41.

Potlapalli, Harshavardhan. 1989. *Digital Map Classification and Compression Using Vector Quantization.* M.Phil. thesis, Department of Electrical Engineering, Tulane University.

Pratt, W. K. 1971. Spatial Transform Coding of Color Images. *IEEE Transactions on Communications*, **COM-19**(December), 980–992.

Procopiuc, Cecilia M. 1996 (March). *Clustering Problems and their Applications (a survey).* Technical Report -. Department of Computer Science, Duke University, http://www.cs.duke.edu:80/ magda/HomePage.html.

Prosise, Jeff. 1995. Color Optimization and Dithering. *PC Magazine*, **14**(6), 253–257.

Prosise, Jeff. 1996. Wicked Code. *Microsoft Systems Journal*, **11**(8), 97–106.

R. J. Fowler, M. S. Paterson & Tanimoto, S. L. 1981. Optimal Packing and Covering in the Plane are NP-Complete. *Information Processing Letters*, **12**, 133–137.

Roytman, E. & Gotsman, G. 1995. Dynamic Color Quantization of Video Sequences. *IEEE Transactions on Visualization and Computer Graphics*, **1**(3), 274–286.

Sagan, Hans. 1994. *Space-filling curves.* New York: Springer Verlag.

Sakauchi, M.; Suzuki, T.; Toriumi, Y. & Ohsawa, Y. 1989. A Flexible and High-Speed Color Image Quantization Using a 3-D Pattern Data Structure. *Applications of Digital Image Processing XII (1989)*, **SPIE 1153**.

Scharcanski, J.; Chen, H. C. & Silva, A. P. Alves Da. 1993. Colour Quantisation for Colour Texture Analysis. *IEEE Proceedings*, **140**(2), 109–114.

Schmitz, B. E. & Stevenson, R. L. 1994. Color Palette Restoration. *Human Vision, Visual Processing, and Digital Display IV (1993)*, **SPIE 2179**, 327–338.

Schore, M. 1991. Octree Method of Color Matching. *The C Users Journal*, **9**(8), 43–52.

Seecheran, Cris Anand. 1974. *Colour Image Coding Using Linear Transformation and Block Quantization.* M.App.Sc. thesis.

Shufelt, Jefferey A. 1995. *Color Image Quantization Enhancement Techniques.* Pittsburgh, PA: School of Computer Science, Carnegie Mellon University.

Silverstein, Louis D.; Krantz, John H.; Gomer, Frank E.; Yeh, Yei-Yu & Monty, Robert W. 1990. Effect of Spatial Sampling and Luminance Quantization on the Image Quality of Color Matrix Displays. *Journal of the Optical Society of America*, **7**(10), 1955–1969.

Smith, N. S.; Whitfield, T. W. A. & Wiltshire, T. J. 1990. A Color Notation Conversion Program. *Color Research and Application*, **15**(6), 338–343.

Sol, Javier Antonio. 1995. *Implementation and Evaluation of a Fast Clustering Algorithm for Color Quantization.* M.Phil. thesis, Florida Institute of Technology.

Soubigou, Andre. 1976. *Three-dimensional Transform Encoding and Block Quantization of Still Colour and Monochrome Moving Pictures.* Ph.D. thesis.

Spaulding, K. E.; Ray, L. A. & Sullivan, J. R. 1993. Secondary Quantization of Color Images for Minimum Visual Distortion. *Human Vision, Visual Processing, and Digital Display IV (1993)*, **SPIE 1913**, 261–269.

Stenger, L. 1977. Quantization of TV Chrominance Signals Considering the Visibility of Small Color Differences. *IEEE Transactions on Communications*, **COM-25**(11), 1393.

Stevens, R. J.; Lehar, A. F. & Preston, F. H. 1983. Manipulation and Presentation of Multidimensional Image Data Using the Peano Scan. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-5**(5), 520–526.

Stone, M. C.; Cowan, W. B. & Beatty, J. C. 1988. Color Gamut Mapping and the Printing of Digital Images. *ACM Transactions on Graphics*, **7**(3), 249–292.

Thomas, S. W. 1991. Efficient Inverse Color Map Computation. *Pages 116–125, 528–535 of:* Arvo, J. (ed), *Graphics Gems II.* Cambridge, MA: Academic Press Professional.

Thomas, Troy Maurice. 1988. *Vector Quantization of Color Images Using Distributed Multiprocessors.* M.Sc. thesis.

Thomas H. Cormen, Charles E. Leiserson & Rivest, Ronald L. 1994. *Introduction to Algorithms.* New York: Mc Graw Hill.

Tremeau, A. 1993 (October). *Contribution des Modeles de la Perception Visuelle a l'Analyse d'Image Couleur.* M.Sc. thesis.

Tremeau, A. & Laget, B. 1995. Color Quantization and Image Analysis. *Traitement du Signal*, **12**(1), 1–22.

Tremeau, A.; Calonnier, M. & Laget, B. 1993. Evaluation of Color Quantization Errors. *Proc. Thirteenth International Display Research Conference (EuroDisplay '93)*, **VIQ-P5**(September), 423–426.

Tremeau, A.; Calonnier, M. & Laget, B. 1994. Color Quantization Error in Terms of Perceived Image Quality. *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-94)*, **5 (Image and Multidimensional Signal Processing)**(April), V93–V96.

Velho, Luiz; Gomes, Jonas & Sobreiro, Marcus R. 1997. Color image quantization by pairwise clustering. *Pages 203–210 of: X brazilian symposium of computer graphics and image processing.* Campos do Jordao, SP: IEEE Computer Society, Los Alamitos, CA.

Verevka, O. 1995. Color Image Quantization in Window Systems with Local K-Means Algorithm. *Proc. Western Computer Graphics Symposium '95*, March 20-22,.

Verevka, O. & Buchanan, J. 1995. Local K-Means Algorithm for Color Image Quantization. *Graphics/Vision Interface '95*, May.

Voloboj, A. G. 1993. The Method of Dynamic Palette Construction in Realistic Visualization Systems. *Computer Graphics Forum*, **12**(5), 277–296.

Wan, S.; Wong, S. & Prusinkiewicz, P. 1988. An Algorithm for Multidimensional Data Clustering. *ACM Transactions on Mathematical Software*, **14**(2), 153–162.

Wan, S. J.; Prusinkiewicz, P. & Wong, S. K. M. 1990. Variance-Based Color Image Quantization for Frame Buffer Display. *Color Research and Application*, **15**(1), 52–58.

Watkins, C. D.; Coy, S. B. & Finlay, M. 1993. *Photorealism and Ray Tracing in C.* Redwood City, CA: M and T Books.

Wu, Xiaolin. 1991a. Efficient Statistical Computations for Optimal Color Quantization. *Pages 126–133 of:* Arvo, J. (ed), *Graphics Gems II.* Cambridge, MA: Academic Press Professional.

Wu, Xiaolin. 1991b. Optimal Quantization by Matrix-Searching. *Journal of Algorithms*, **12**(4), 663–673.

Wu, Xiaolin. 1992a. Color Quantization by Dynamic Programming and Principal Analysis. *ACM Transactions on Graphics*, **11**(4), 348–372.

Wu, Xiaolin. 1992b. Statistical Color Quantization for Minimum Distortion. *Pages 189–202 of:* Falcidieno, B.; Herman, I. & Pienovi, C. (eds), *Computer Graphics and Mathematics.* Berlin, Germany: Springer-Verlag. ISBN 038755582X.

Wu, Xiaolin & Witten, I. 1985. *A Fast k-Means Type Clustering Algorithm.* Research Report 85/197/10. Department of Computer Science, University of Calgary, Alberta.

Xiang, Z. & Joy, G. 1994a. Color Image Quantization by Agglomerative Clustering. *IEEE Computer Graphics and Applications*, **14**(3), 44–48.

Xiang, Z. & Joy, G. 1994b. Feedback-based Quantization of Color Images. *SPIE Image and Video Processing II (1994)*, **SPIE 2182**(February), 34–42.

Y. Linde, A. Buzo & Gray, R. 1980. An Algorithm for Vector Quantizer Design. *IEEE Transaction on Communication*, **COM-28**(1), 84–95.

Yang, Ching-Yung & Lin, Ja-Chen. 1996. RWM-Cut for Color Image Quantization. *Computers and Graphics*, **20**(4), 577–590.