Representação de uma imagem digital através do processo de Quantização Vetorial

Amarildo Martins de Mattos e Diego Ferreira dos Santos
Universidade Estadual de Campinas – UNICAMP, Campinas-SP

Abstract *— O objetivo deste artigo é apresentar os conceitos da representação de uma imagem através do processo de Quantização Vetorial(QV) e desenvolver uma ferramenta computacionalmente didática que ilustra a quantização de cores utilizando a técnica de QV. Assim, o projeto de QV baseia-se o Método de Lloyds e o algoritmo LBG (de Linde, Buzo e Gray). O software é bastante versátil, permitindo que o usuário defina o tamanho de bits de codificação, isto é, o alfabeto de reprodução (codebook). Os resultados para obtenção do alfabeto de reprodução são apresentados em gráficos, que permitem ao usuário analisar cada etapa do processo da quantização de cores de uma imagem. Este programa foi desenvolvido sobre a plataforma Matlab® versão 5.3.*

INTRODUÇÃO

A qualidade da digitalização de uma imagem está diretamente relacionada com o número de *pixels* e linhas, e com a gama de intensidades de brilho que se pode ter numa imagem. Estes dois aspectos são conhecidos como resolução da imagem, que pode ser definida por dois fatores: a "resolução espacial" e a "resolução de brilho" (ou "resolução de cores" no caso de se tratar de imagens coloridas).

O termo "resolução espacial" (no nosso caso, espacial se refere ao espaço 2D) é usado para descrever quantos *pixels* compõem a imagem digital, desta forma, quanto maior o número de *pixels* maior será a "resolução espacial".

Quando falamos de Processamento de Imagens estamos necessariamente a falar de Imagens Digitais, já que a idéia de processamento está relacionada com as tarefas computacionais, e a imagem computacional é a Imagem Digital. Portanto, processar uma imagem é na verdade fazer algum tipo de alteração na sua estrutura inicial.

Cada *pixel* numa imagem digital representa a intensidade luminosa de um determinado ponto da imagem original. Sendo assim, o conceito de "resolução de brilho" refere-se ao brilho de cada pixel para representar a intensidade luminosa da imagem original. O processo de quantização converte uma intensidade de tons contínuos, num valor de brilho. A precisão deste valor digital está diretamente relacionada com o número de bits que serão utilizados na quantização. Como exemplo, adaptando-se uma imagem digital que possua somente tons de cinza, se forem utilizados 3 bits, o brilho pode ser convertido em 8 tons de cinza, ao passo que se forem utilizados 8 *bits*, este valor passará para 256 tons.

Assim, com o uso cada vez maior de todo tipo de documento eletrônico, por uma quantidade cada vez maior de pessoas, das mais variadas áreas e com diferentes aplicações, descobriu-se rapidamente que os usuários enfrentariam dificuldades para armazenar, manipular e transmitir imagens digitais.

De forma específica, cita-se as imagens digitais obtidas através de alguns satélites (orbitais) em que todas as informações passam por algum tipo de processamento computacional. Chama atenção o tamanho do espaço requerido para o armazenamento das imagens orbitais, da ordem de milhões de bytes. Parte deste problema pode ser resolvida quando se faz uso de técnicas de compressão de dados.

No processo de compressão objetiva-se a redução da quantidade de dados necessários para representar uma imagem digital, e a base para que esta redução ocorra está na remoção de dados redundantes, a qual deve ser feita antes do armazenamento ou transmissão da imagem, realizando o processo de quantização.

Dessa forma, existem dois processos de quantização: escalar e vetorial. A quantização vetorial (QV) [1] é uma generalização da quantização escalar, isto é, a quantização não é mais em uma única dimensão e sim em múltiplas dimensões, deste modo, permite o aparecimento de novas idéias, conceitos, técnicas, e aplicações que freqüentemente não se tem em quantização escalar. A QV é usada com processamento de imagens digitais, onde em muitos casos a imagem de entrada é digital e o de saída é a imagem de entrada comprimida. QV é usualmente, mas não exclusivamente, usada para compressão de dados. Deste modo, torna-se necessário introduzir os conceitos referentes a este assunto para os alunos de maneira clara e objetiva, tornando-se o processo de aprendizado mais eficiente.

Neste artigo, apresenta-se uma abordagem educacional para estudo sobre QV, utilizando o algoritmo LBG (de Linde, Buzo e Gray) [2]. Esta abordagem é realizada através do uso de um programa de simulação desenvolvido na plataforma Matlab® versão 5.3, permitindo que o usuário o tamanho de bits de codificação, isto é, o alfabeto de reprodução (*codebook*). Devido a sua complexidade, os conceitos envolvidos neste processo não são bem compreendidos pelos alunos. Por este motivo, é importante apresentar abordagens que permitam analisar o sistema de uma maneira clara e objetiva, ilustrando todo o processo e facilitando a compreensão do aluno. Serão apresentados resultados de simulação, que serão analisados para demonstrar o potencial didático da ferramenta computacional desenvolvida.

A seguir, apresentamos alguns conceitos iniciais quantização de imagens.

CARACTERÍSTICAS DAS IMAGENS

Uma imagem tem natureza digital e discreta, sendo constituída por um arranjo de elementos sob a forma de uma matriz, de dimensões: x linhas por y colunas. Cada elemento desta matriz representa um *pixel* que possui um atributo z, indicando um nível de cinza, comumente variando de 0 (preto) a 255 (branco) e representa a intensidade da energia eletromagnética (refletida ou emitida) correspondente ao tamanho do *pixel*.

Deve-se, no entanto, ressaltar que a escala de tons, que atualmente ainda é mais comum com a variação de 0 a 255 pode chegar a representar milhares de tons, o que certamente proporcionará um material de melhor qualidade visual, às custas é claro, de um maior espaço de armazenamento.

A COR NO PROCESSAMENTO DE IMAGENS DIGITAIS

A compressão de imagens está fundamentada na eliminação de redundâncias irrelevantes para a representação de dada informação. A percepção das cores pelo mecanismo de visão do ser humano tem papel importante no estabelecimento destas redundâncias.

Devido à estrutura do olho humano, todas as cores são vistas como combinações variáveis das três chamadas *cores primárias*: vermelho (R, do inglês "*red*"), verde (G, do inglês "*green*") e azul (B, do inglês "*blue*"). Variando a quantidade relativa de vermelho, verde e azul é possível produzir uma enorme variedade de cores, incluindo diversos tons para cada uma. Fato este da maior importância, pois enquanto o número de tons de cinza discernidos pelo olho humano não passa de duas dúzias, a cor além de um poderoso descritor de objetos é discernida em milhares de tons e intensidades, sendo, portanto decisiva na análise de imagens por seres humanos [6].

Existem muitas formas de se representar cores numericamente. Um sistema para representar cores é chamado de modelo de cores e a maioria dos modelos de cores em uso objetiva uma maior vantagem sobre um tipo específico de *hardware* ou facilidade de manipulação de cores em imagens coloridas.

Os modelos mais populares são o RGB (*red, green, blue*) para monitores coloridos, e o CMY (*cyan, magenta, yellow*) para impressoras coloridas; são modelos orientados para o *hardware*. Outros modelos existentes como HSI (*hue, saturation, intensity*) ou (matiz, saturação, intensidade) e o HSV (matiz, saturação e valor), por exemplo, são modelos orientados para a manipulação de cores.

REDUNDÂNCIAS

Diferentes quantidades de dados podem ser utilizadas para transmitir uma mesma informação. Esta diferença consiste na redundância de dados. Os tipos de redundância encontrados são:

Redundância de codificação - ocorre quando os níveis de cor de uma imagem são codificados com mais símbolos de codificação do que o necessário.

Redundância interpixel - ou *redundância espacial* são as correlações resultantes das relações geométricas ou estruturais entre os objetos na imagem.

Redundância espectral - em imagens com mais de uma faixa espectral, os valores espectrais para a mesma posição na matriz de *pixels* de cada banda, são geralmente correlacionados.

Redundância psicovisual - o olho humano não responde com a mesma sensibilidade a todas as informações visuais. Certas informações simplesmente têm menos importância relativamente a outras no processamento visual normal. Tais informações são ditas psicovisualmente redundantes e podem ser eliminadas sem prejudicar significativamente a qualidade de percepção da imagem.

A eliminação de dados psicovisualmente redundantes resulta numa perda de informação e é comumente denominada quantização. Como essa operação é irreversível, ou seja, informação visual é perdida, a quantização resulta em compressão de dados com perda[6].

COMPRESSÃO DE IMAGENS

Compressão de dados é a arte ou ciência de representar informações em uma forma compacta. Cria-se essas representações por identificação e uso de estruturas que existem nos dados. Dados podem ser caracteres em um arquivo texto, números que são amostrados de várias formas como palavras ou imagens, ou seqüências de números que são gerados por outros processos[7]. Assim, um dos processos de compressão é a quantização, que é o processo de redução do número de *bits* necessários para armazenar um valor reduzindo sua precisão para um inteiro. É nesta redução da precisão que acontece o processo irreversível de perda da informação.

Dessa forma, existem dois processos de quantização: escalar e vetorial. A quantização vetorial (QV) [1] é uma generalização da quantização escalar, isto é, a quantização não é mais em uma única dimensão e sim em múltiplas dimensões, deste modo, permite o aparecimento de novas idéias, conceitos, técnicas, e aplicações que freqüentemente não se tem em quantização escalar. A QV é usada com processamento de imagens digitais, onde em muitos casos a imagem de entrada é digital e o de saída é a imagem de entrada comprimida. QV é usualmente, mas não exclusivamente, usada para compressão de dados. Deste modo, torna-se necessário introduzir os conceitos referentes a este assunto para os alunos de maneira clara e objetiva, tornando-se o processo de aprendizado mais eficiente.

QUANTIZADOR VETORIAL

A Quantização Vetorial (QV) é uma generalização do problema da Quantização Escalar, direcionada para um conjunto de vetores reais. A QV é usada normalmente em processamento de imagem e permite transformar uma imagem de entrada, normalmente com uma representação digital, numa imagem de saída que é uma versão comprimida do original. É aplicada em situações em que a imagem, depois de descomprimido, não necessita de ser exatamente igual ao original: a QV é uma técnica que implica perdas (‘*lossy*’).

A QV codifica seqüências de entrada - vetores – fazendo o seu mapeamento a seqüências de menores dimensões – *codewords ou palavra código*. Ao conjunto de todas as *codewords*, com um índice respectivo associado, dá-se o nome de *codebook*. Comprimir dados usando QV consiste em identificar a *codeword* para cada vetor da imagem. Para reaver a imagem, com perdas, basta recuperar as *codewords* correspondentes aos índices sendo necessário que o dispositivo de descompressão conheça o *codebook*.

A similaridade à imagem original depende fortemente do *codebook*. Se os vetores estiverem próximos das *codewords* então a imagem será semelhante ao original. A proximidade (ou distância) entre os vetores e as *codewords* é um critério muito importante na QV e pode ser medida de diversas formas. A mais usada é a Distância Euclidiana ao Quadrado (‘*Squared Euclidean Distance’*) dada por:

$$d(x,\hat{x}) = \sum_{i=0}^{k-1} \left| x_i - \hat{x}_i \right|^2$$

A distorção [1] [3] causada pela reprodução de um vetor de entrada *x* por um vetor de reprodução $\hat{x}$ é dada por uma medida de distorção não negativa .

A idéia do QV é gerar um *codebook* que minimize a distância entre os vetores que compõem a imagem e as suas *codewords* de modo a minimizar a diferença entre a imagem original e a comprimida, ou seja, a *distorção*. Para minimização da distorção o algoritmo de Lloyd é à base dos métodos de geração de um *codebook* para compressão de uma imagem.

PROJETO DO QUANTIZADOR VETORIAL

Definição do Método de Lloyds

Suponha que a distribuição dos símbolos da fonte seja conhecida. Esta distribuição pode ser tanto contínua como discreta, e não se faz nenhuma suposição quanto à existência de derivadas.

Dado um quantizador *q* [1] [3] descrito por um alfabeto de reprodução $\hat{A} = \{y_i; i = 1,...,N\}$ e uma partição $S = \{S_i; i = 1,...,N\}$, a distorção esperada

$$D(\{\hat{A}, S\}) \underline{\underline{\Delta}} D(q)$$

pode ser escrita como:

$$D(\{\hat{A}, S\}) = E\{d(x, q(x))\} = \sum_{i=1}^{N} E\{d(X, y_i) \mid X \in S_i\} P(X \in S_i)$$

onde $E\{d(X, y_i) \mid X \in S_i\}$ é a distorção esperada condicional dado que $X \in S_i$ ou, equivalentemente, dado que $q(x) = y_i$.

Suponha que tem-se um alfabeto de reprodução $\hat{A}$, mas a partição não foi especificada. Uma partição ótima para $\hat{A}$ é facilmente construída mapeando cada vetor de entrada *x* em $y_i \in \hat{A}$ minimizando a distorção $d(x, y_i)$, isto é, escolhendo o vetor de reprodução que cause a mínima distorção (ou vetor de reprodução mais próximo) para cada entrada.

No caso em que mais de um vetor de reprodução que minimiza a distorção, define-se uma regra de desempate, por exemplo, o vetor de reprodução com o menor índice.

A partição $P(\hat{A}) = (P_i; i = 1,...,N)$ construída desta maneira é tal que $x \in P_i$ (ou $q(x) = P_i$) somente se $d(x, y_i) \le d(x, y_j), \forall j$, e portanto

$$D(\{\hat{A}, P(\hat{A})\}) = E\left\{\min_{y \in \hat{A}} d(X, y)\right\}$$

o que, por sua vez, implica para qualquer partição *S* que

$$D(\{\hat{A}, S\}) \ge D(\{\hat{A}, P(\hat{A})\})$$

Portanto, para um alfabeto de reprodução $\hat{A}$ fixo, a melhor partição possível é $P(\hat{A})$.

Por outro lado, assuma que exista uma partição $S = \{S_i; i = 1,..., N\}$ descrevendo um quantizador. Assuma também que a medida de distorção e a distribuição dos símbolos da fonte são tais que, para cada conjunto *S* com probabilidade não nula no espaço euclidiano *k*-dimensional, existe um vetor $\hat{x}(S)$ de mínima distorção para o qual

$$E\{d(X, \hat{x}(S)) \mid X \in S\} = \min_{u} E\{d(X, u) \mid X \in S\}$$

De forma análoga ao caso de uma medida de distorção quadrática e uma distribuição de probabilidade uniforme, chamamos o vetor $\hat{x}(S)$ de centróide ou centro de gravidade de *S*.

Se tais pontos existem, então claramente para uma partição fixa $S = \{S_i; i = 1,..., N\}$, nenhum alfabeto de reprodução $\hat{A} = \{y_i; i = 1,..., N\}$ pode gerar uma distorção média menor do que o alfabeto de reprodução $\hat{x}(S) \overset{\Delta}{=} \{\hat{x}(S_i); i = 1,..., N\}$ contendo os centróides dos conjuntos em *S* pois

$$D(\{\hat{A}, S\}) = \sum_{i=1}^{N} E\{d(X, y_i) \mid X \in S_i\} P(X \in S_i)$$

$$\geq \sum_{i=1}^{N} \min_{u} E\{d(X,u) | X \in S_i\} P(X \in S_i)$$

$$= D(\{\hat{x}(S), S\})$$

O algoritmo de Lloyd determina o alfabeto de reprodução a partir da distribuição de símbolos disponíveis. Veremos adiante que a partir do trabalho de Lloyd, formulou-se o algoritmo mais utilizado atualmente em quantização vetorial: o LBG

Definição do algoritmo LBG

Antes de descrever o algoritmo LBG, será necessário apresentar o conceito sobre a técnica de *splitting*.

Definição da técnica de *splitting*:

Dada uma palavra código de dimensão *K*, seja y_0, esta pode ser "dividida" em duas outras novas palavras código, $y_0 + \varepsilon$ e $y_0 - \varepsilon$, onde ε é um vetor de valor positivo muito pequeno de dimensão *K*. Assim, a cada processo de "divisão" o número de palavras códigos dobra gerando novo alfabeto de reprodução.

Descrição do algoritmo LBG

O algoritmo mais utilizado para a criação do alfabeto de reprodução é o LBG [1] [3][5]. O algoritmo LBG é uma extrapolação do trabalho de Llyod [1] [2], para o caso de vetores de dimensão *K* [1]. A seguir, será descrita de maneira sucinta uma implementação do algoritmo LBG. O objetivo desta descrição é apresentar uma visão prática do algoritmo de forma a simplificar a tarefa daqueles que queiram implementá-lo.

1º Passo – Inicialização dos dados:

N- número de palavras código do alfabeto de reprodução.
K- dimensão das palavras código e dos vetores da seqüência de treinamento.
Seq_Trein[*V*][*K*] – Sequência de treinamento com *V* vetores de treinamento dimensão *K*.
Lim_Dist – limiar de distorção.
Lim_Min - limiar de parada do algoritmo, por exemplo, 0.0001.
Dist_Ant - distorção média da iteração anterior. Inicialmente, assume um valor muito grande.
Dist_Atual – distorção média.
Centroide[*N*][*K*]- matriz que armazena o alfabeto de reprodução. Inicialmente, a primeira palavra código é a média estatística da sequência de treinamento.
Num_Centroide – número máximo de palavras código desejada.
ε - um vetor constante com valor fixos e muito pequenos de dimensão K, que representa a perturbação para determinação das novas palavras *código*.

2º Passo - Gerar o alfabeto de reprodução

Enquanto o comprimento do alfabeto de reprodução não for igual *Num_Centroide*, gerar o novo alfabeto a partir da técnica de "*splitting*" para cada palavra *código*.

3º Passo - Codificação da sequência de treinamento:

Para cada ponto da *Seq_Trein* encontrar a palavra *código* que tiver menor distorção, isto é, a palavra *código* mais próxima de cada ponto. E, em seguida, calcula-se a distorção provocada por toda sequência de treinamento (*V*).

4º Passo – Determinar a distorção média atual

No passo anterior obtêm-se a distorção total provocada pela *Seq_Trein*, deste modo, a distorção média é dada por *Dist_Atual* e é igual a distorção total dividido por *V*, onde *Dist_Atual* é a média estatística.

5º Passo – Cálculo do limiar de distorção

O *Lim_Dist* é igual a (*Dist_Ant* - *Dist_Atual)*/ *Dist_Ant*. Se o *Lim_Dist* for maior que *Lim_Min*, significa que o algoritmo ainda está convergindo e deve retornar ao passo 3. Caso contrário, o processo segue para o próximo passo.

5º Passo: Teste de parada do algoritmo:

O teste de parada é definido através do tamanho do alfabeto de reprodução desejado, caso o tamanho não esteja satisfeito retornar ao passo 2.

QUANTIZAÇÃO VETORIAL DE UMA IMAGEM

Um quantizador vetorial [4] pode ser visto como a cascata de um codificador e um decodificador. O codificador identifica em qual região do espaço o vetor a ser quantizado se encontra, associando-lhe a *codeword* correspondente à região identificada. O decodificador gera o vetor de saída a partir do índice desta *codeword*. Essencialmente, o codificador é definido pelo mapeamento do espaço K-dimensional em um número finito de regiões. Pode-se utilizar o conhecimento acerca da geometria destas regiões na elaboração de métodos de busca rápida que permitam diminuir a complexidade da QV.

A complexidade da QV pode ser classificada de acordo com a Figura 3 [3]. A etapa de projeto do quantizador é realizada *off-line*. Quando estamos realizando simulações experimentais que envolvam o projeto de diversos quantizadores, a complexidade desta etapa torna-se um fator significativo. Contudo, uma vez projetado o quantizador, o custo de sua implementação é determinado apenas pela memória e pelo número de cálculos necessários para selecionar a me-

lhor *codeword*. Com isto, a complexidade da etapa de operação do quantizador é geralmente a mais relevante.

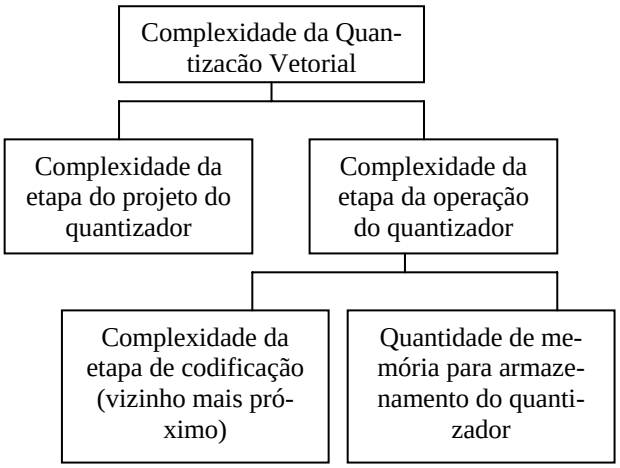

Figura 1. Complexidade QV

Desta forma, a quantização de cores reduz o número de bits de cores que irá representar uma imagem. Este número de cores define o número de níveis de quantização de cores. Sendo N = número de níveis de níveis de cores e k = o número de bits, tem-se a seguinte equação:

$$N = 2^K \text{ níveis de cores}$$

Para quantização de cores na escala RGB, cada cor será representada por N níveis, e o total de cores será:

$$N = 2^K \text{ níveis de cores}$$
$$\text{Total de cores RGB} = 2^N . 2^N . 2^N = 2^{3N} \text{ cores (RGB)}$$

SIMULAÇÃO

Nesta sessão serão apresentados alguns resultados de simulação e suas respectivas análises, com o intuito de demonstrar o potencial didático desta ferramenta para determinação da palavra código (centróides) e também os resultados obtidos para os níveis de cores.

Para inicializar o programa, basta digitar *quantvetorial* na janela de comando do Matlab®. A primeira etapa da simulação consiste no processo de QV como esta descrita abaixo:

Simulação do quantizador vetorial

Para verificar a influência do processo de busca da palavra código foi construído um QV de ordem 4 (*N*=4) e dimensão 2 (*K*=2). Para a construção do QV, obtendo-se assim o alfabeto de reprodução, foram gerados 20.000 vetores distribuídos propositadamente da seguinte maneira:

- 5000 pontos com distribuição gaussiana bidimensional de médias $\mu_x = -1$, $\mu_y = -1$ e variâncias $\sigma_x^2 = -1$ e $\sigma_y^2 = -1$.
- 5000 pontos com distribuição gaussiana bidimensional de médias $\mu_x = -1$, $\mu_y = -1$ e variâncias $\sigma_x^2 = 1$ e $\sigma_y^2 = 1$.
- 5000 pontos com distribuição gaussiana bidimensional de médias $\mu_x = 1$, $\mu_y = 1$ e variâncias $\sigma_x^2 = -1$ e $\sigma_y^2 = -1$.
- 5000 pontos com distribuição gaussiana bidimensional de médias $\mu_x = 1$, $\mu_y = 1$ e variâncias $\sigma_x^2 = 1$ e $\sigma_y^2 = 1$.

Assim, foi construído o QV da Figura 4.

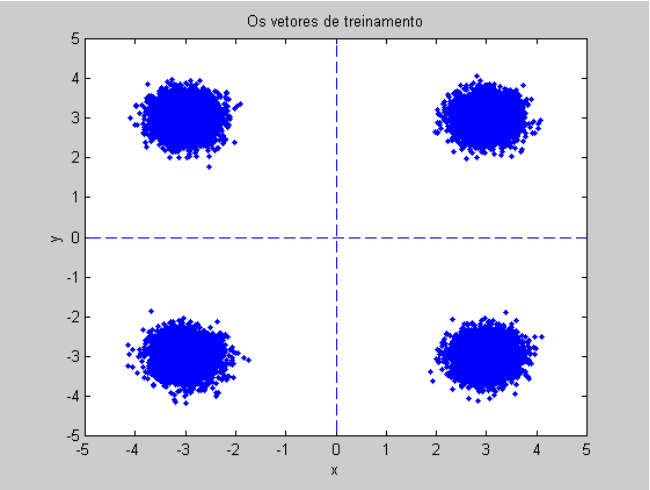

Figura 4: Os vetores.

Deste modo, a obtenção de um alfabeto de reprodução de ordem 4, é necessário gerar um alfabeto de reprodução de ordem 1, 2 e 4 respectivamente como apresentamos na seção sobre o algoritmo LBG. A seguir ilustraremos a obtenção de cada *codebook* .

A figura 5 ilustra a posição do codebook de ordem 1. Como podemos observar na figura abaixo a posição do alfabeto de reprodução (ponto de destaque em vermelho) está localizada na posição central da figura, isto é, na posição x = 0 e y = 0, como era previsto, pois todos os 20.000 foram distribuídos propositadamente conforme descrito anteriormente.

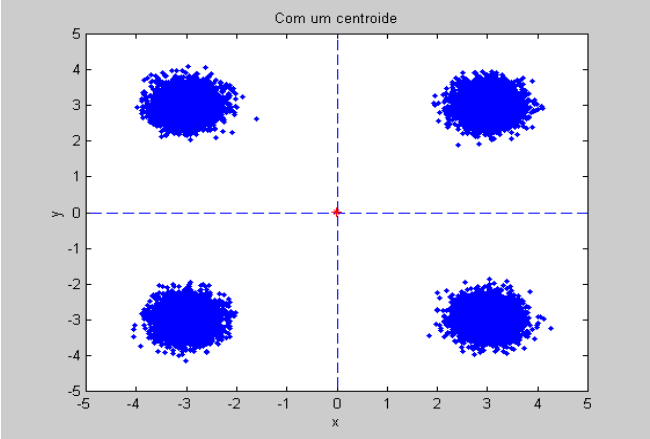

Figura 5: *Codebook* de ordem 1.

A próxima figura ilustra o processo para obtenção do *codebook* de ordem 2, após o processo de *splitting* sobre o codebook de ordem 1.

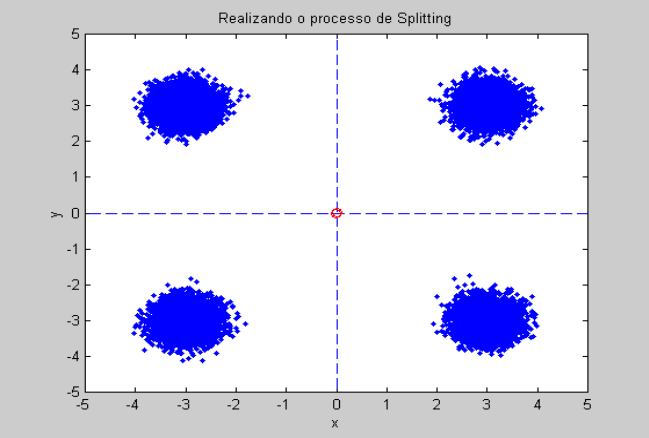

Figura 6: *Splitting sobre o codebook de ordem 1*.

Após o processo de s*plitting*, inicializa-se a etapa de reorganização dos centroídes até a melhor coloção em torno dos pontos, esta reorganização do *codebook* é baseado no algoritmo LBG e o processo de parada depende do limiar de probabilidade definido pelo usuário conforme descrito anteriormente. Assim, as Figuras 7, 8, 9, 10, 11 e 12 ilustra cada etapa do processo de QV.

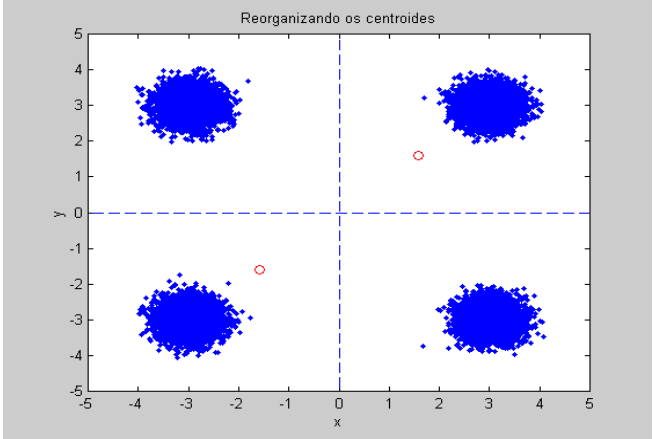

Figura 7: 1º Etapa da r*eorganizando os centroides.*

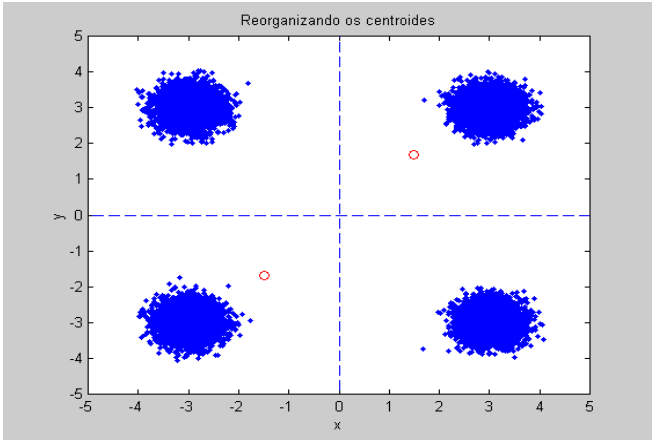

Figura 8: 2º Etapa da r*eorganizando os centroides*

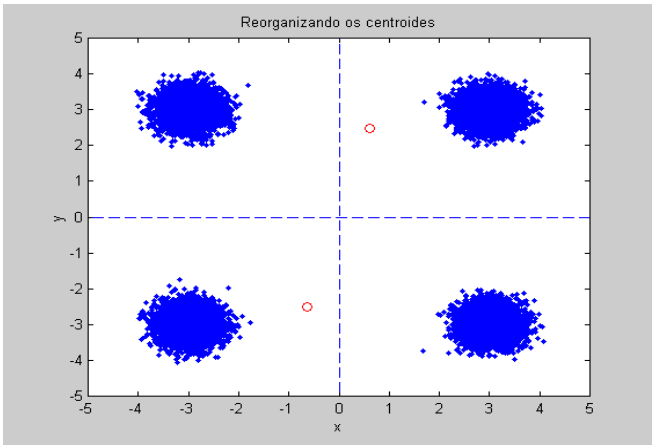

Figura 9: 3º Etapa da r*eorganizando os centroides*

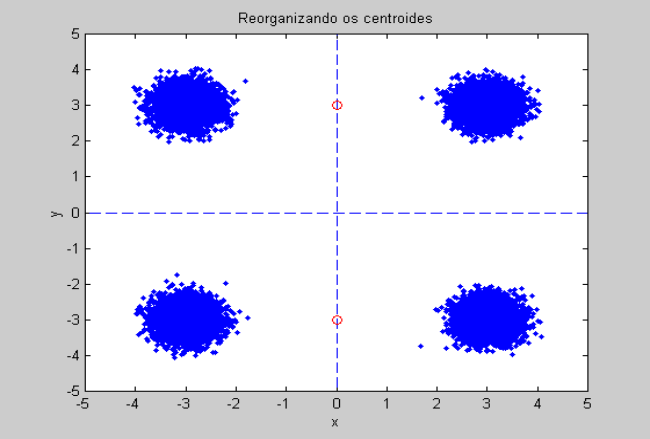

Figura 10: 3º Etapa da r*eorganizando os centroides*

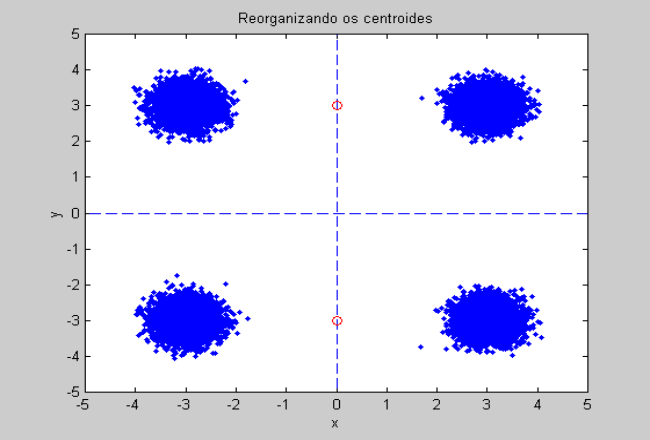

Figura 11: 4º Etapa da r*eorganizando os centroides*

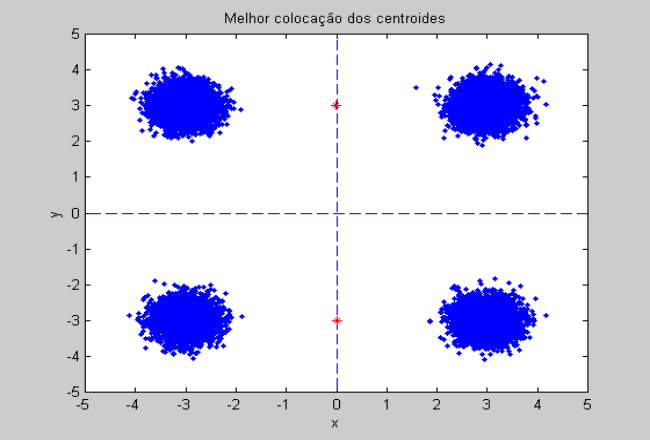

Figura 12: *Centroides ótimos de ordem* .

Realizando s*plitting* sobre o codebook de ordem 2 obtemos o codbook de ordem 4 como mostrado na próxima figura :

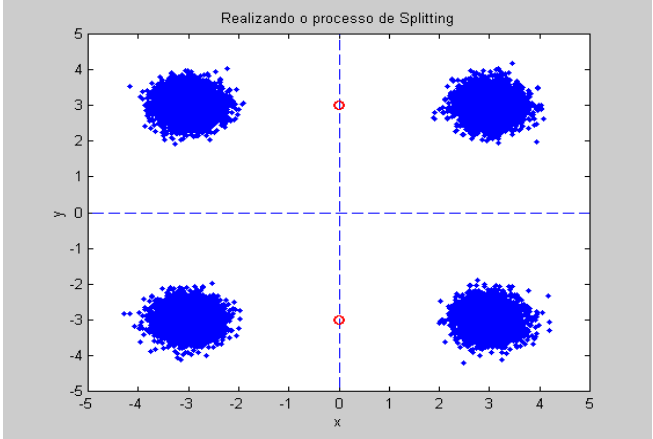

Figura 13: *Splitting sobre o codebook de ordem 2*.

Novamente, após s*plitting*, inicializa-se a etapa de reorganização dos centroídes até a melhor coloção em torno dos pontos, conforme ilusta as Figuras.

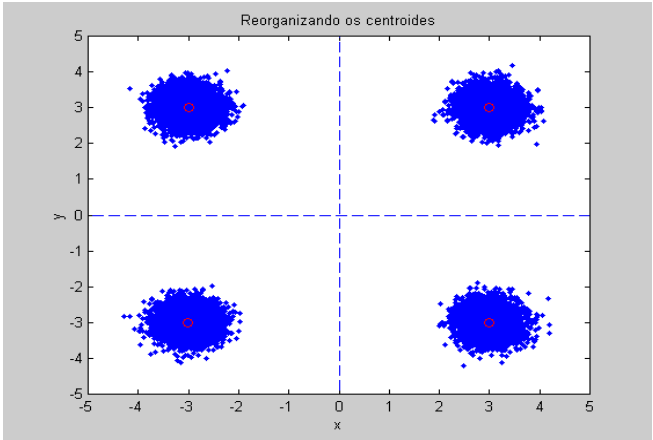

Figura 14: *Reorganizando o codebook.*

Como foi proposto, a figura a seguir apresenta o alfabeto de reprodução de ordem 4.

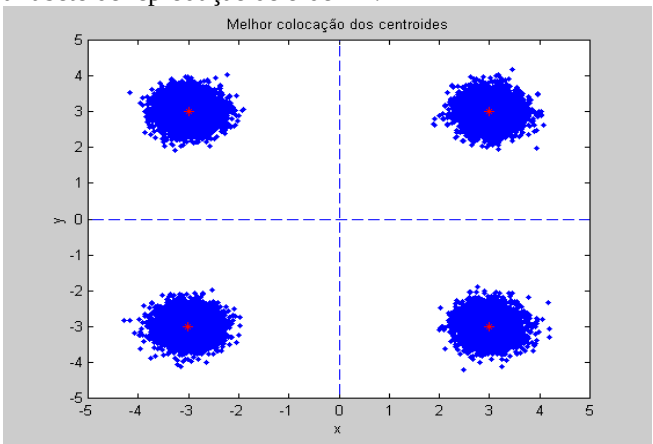

Figura 15: *Alfabeto de reprodução de ordem 4*.

Quantização de Cores de uma imagem

Para verificar a influência do processo de quantização de cores de uma imagem, foi realizado testes para diferentes níveis de quantização: 1 nível, 2 níveis e 3 níveis. A figura 14 representa uma imagem real e nas figuras 15, 16 e 17 representa a quantização de cores para 1 nível, 2 níveis e 3 níveis, respectivamente.

Figura 16: Imagem Original

Assim, a quantização com 1 nível representa um total de 8 níveis de cores como mostrado na figura 17.

Figura 17: Imagem com 1 nível

A quantização com 2 níveis representa um total de 64 níveis de cores como mostrado na figura 18.

Figura 18: Imagem com 2 níveis

A quantização com 3 níveis representa um total de 512 níveis de cores como mostrado na figura 19.

Figura 19: Imagem com 3 níveis

CONCLUSÃO

Neste artigo foi apresentada a teoria de quantização vetorial no processo de quantização de cores, junto com a implementação do algoritmo LBG, apresentando passo a passo o processo de implementação deste algoritmo.
Como trabalho futuro é a adaptação deste algoritmo para imagens reais, apresentando todo o processo de quantização de cores.

REFERÊNCIAS

[1] Gersho Allen e M. Gray Robert, "**Vector Quantization and Signal Compression**", pag 307-585, 1991.

[2] Y. Linde, A. Buzo & R. Gray, "An Algorithm for Vector Quantization Design", *IEEE Trans. on Comm.* 28:84-95, Jan. 1980.

[3] Alberto Ynoguti Prof. Dr. Carlos Apostila de "**Quantização Vetorial**".

[4] Klautau Aldebaro, Dissertação de Mestrado, "**Codificação CELP com Quantização Vetorial do Filtro LPC Utilizando busca rápica**

em Árvore K–d" http://speech.ucsd.edu/aldebaro/papers/ak_msc_thesis.pdf, em 06 de julho de 2002.

[5] Klautau Aldebaro, Vinicius Lamar Marcus, M. Bermudez José Carlos e Seara Rui, " *Codificação de Imagens em Sub-Bandas Usando Técnicas de Busca Rápida*", 06 de julho de 2002, http://mars.elcom.nitech.ac.jp/~lamar/public/cba94.pdf.

[6] GONZALEZ, R. C.; WOODS, R. E**. Processamento de imagens digitais.** Tradução de Roberto M. C. Junior, Luciano da F. Costa. São Paulo: Edgard Blücher, 2000. 509p.

[7] SAYOOD, K**. Introduction to data compression.** San Francisco: Morgan Kaufmann, 1996. 475 p.