UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ADRIANO CARAFINI

PROJETO DE DIPLOMAÇÃO

QUANTIZAÇÃO VETORIAL DE IMAGENS COLORIDAS ATRAVÉS DO ALGORITMO LBG

Porto Alegre
2015

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

QUANTIZAÇÃO VETORIAL DE IMAGENS COLORIDAS ATRAVÉS DO ALGORITMO LBG

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para Graduação em Engenharia Elétrica.

ORIENTADOR: Prof. Dr. ADALBERTO SCHUCK JÚNIOR.

Porto Alegre
2015

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ADRIANO CARAFINI

QUANTIZAÇÃO VETORIAL DE IMAGENS COLORIDAS ATRAVÉS DO ALGORITMO LBG

Este projeto foi julgado adequado para fazer jus aos créditos da Disciplina de "Projeto de Diplomação", do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. ADALBERTO SCHUCK JÚNIOR, UFRGS

Doutor em Engenharia de Minas (Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil)

_____

Prof. Dr. Ály Ferreira Flores Filho

Aprovado em: __/__/_____

Banca Examinadora:

Prof. Dr. Altamiro Amadeu Susin, UFRGS

Doutor pela Institut National Polytechnique de Grenoble – Grenoble, França

Assinatura: _____

Dra. Leticia Vieira Guimarães, UERGS

Doutora pela Muroran Institute of Technology – Muroran, Japão

Assinatura: _____

Dedico esse trabalho aos meus pais, por todo apoio e incentivo à minha formação.

RESUMO

Neste trabalho, a técnica de quantização vetorial para compressão de imagens foi implementada em linguagem C/C++, utilizando-se o algoritmo LBG (Linde-Buzo-Gray) para a obtenção de *codebook*. O algoritmo implementado foi aplicado a uma mesma imagem e em diferentes configurações, com o intuito de avaliar a influência do espaço de cor, método de inicialização de *codebook* e taxa de compressão de cada componente de cor na qualidade e nível de compressão da imagem final. A diferença mínima e máxima entre os valores de PSNR (*P*eak *Signal-to-Noise Ratio*) obtidos nos métodos de inicialização por amostragem e *splitting* foram, respectivamente, de 0,01 dB e 0,84 dB. Os valores de PSNR obtidos a uma taxa de compressão de 7,95, que foi a que apresentou melhor compromisso entre distorção agregada e nível de compressão, para os espaços de cor RGB, HSV e CIE Lab foram de 28,9 dB, 32,4 dB e 32,1 dB, respectivamente. Os três canais do sistema de cor RGB apresentaram pouca diferença quanto a sensibilidade às variações das taxas de compressão. Contudo os canais V e L, correspondentes aos espaços de cor HSV e CIE Lab, respectivamente, apresentaram maior sensibilidade à quantização vetorial. Uma comparação de performance do algoritmo de quantização vetorial desenvolvido foi realizada utilizando-se o algoritmo JPG como referência. Para um mesmo nível de distorção, obteve-se, para o algoritmo JPG, razões de compressão até 4 vezes maiores que aquelas obtidas através do algoritmo LBG. E, para um mesmo nível de compressão, obteve-se valores de PSNR até 4,6 dB acima daqueles obtidos empregando-se o método implementado. Contudo, apesar da superioridade do algoritmo JPG frente ao método de quantização vetorial utilizado, o desenvolvimento desse trabalho permitiu verificar o funcionamento, vantagens e limitações do algoritmo LBG para quatização vetorial, que, invariavelmente, será a técnica utilizada em qualquer aplicação do padrão CADRG (*Compressed ARC Digitized Raster Graphics*).

Palavras-chaves: Quantização Vetorial. LBG. Espaço de cor. CADRG.

ABSTRACT

In this work, the vector quantization technique for image compression was implemented in C/C++ programming language, using the LBG algorithm for the codebook design. The implemented algorithm was applied to the same image, in different configurations, with the objective of evaluating the effect of color space, codebook initialization method and compression ratio of each color channel on the overall quality and compression ratio of the final image. The minimum and maximum difference between the PSNR (Peak Signal-to-Noise Ratio) values obtained through the codebook initialization methods, by sampling and splitting, were, respectively, 0.01 dB and 0.84 dB. The PSNR values obtained at an overall compression ratio of 7.95, which was the one presenting the best tradeoff between compression ratio and final distortion, for the color spaces RGB, HSV and CIE Lab were 28.9 dB, 32.4 dB and 32.1 dB, respectively. The three-color channels of the RGB color system presented the same sensibility to the vector quantization. However, the channels V and L, corresponding to the HSV and CIE Lab color spaces, presented greater sensibility to the vector quantization. A performance comparison of the implemented vector quantization algorithm was executed, using the JPG algorithm as reference. For an approximately equal distortion level, the JPG algorithm reached compression ratios up to 4 times greater than the ones obtained through the implemented LBG algorithm. In addition, for an approximately equal compression ratio, the JPG algorithm obtained PSNR values up to 4 dB greater than the ones obtained through the implemented LBG algorithm. Moreover, though the obvious superiority of the JPG algorithm over the implemented one, the development of this work enabled the verification of the behavior, advantages and limitations of the LGB algorithm for vector quantization, which, invariably, will be the used technique on every CADRG (Compressed ARC Digitized Raster Graphics) standard application.

Keywords: Vector Quantization. LBG. Color space. CADRG.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

LBG - Linde-Buzo-Gray

CIE - International Commission on Illumination

ITU - International Telegraph Union

CADRG - Compressed ARC Digitized Raster Graphics

ARC - Equal Arc-second Raster Chart/Map

GLA – Generalized Lloyd's Algorithm

RGB – Red, Green and Blue color space

HSV – Hue, Saturation and Value color space

Lab – L, *luminosidade*, e a e b, dimensões de cores

MSE – Mean Squared Error

NMSE – Normalized Mean Squared Error

PSNR – Peak Signal to Noise Ratio

LISTA DE SÍMBOLOS

C – Conjunto correspondente ao *codebook*

I – Conjunto de vetores de treino

L – Tamanho do conjunto de vetores de treino

Q – Operador quantizador vetorial

y – *Codeword*

x – Vetores de treino

ε – Tolerância para convergência do algoritmo LBG

G – Fator de qualidade da compressão JPG

SUMÁRIO

1 INTRODUÇÃO

A informação, em suas várias formas, é um bem de imprescindível valor para a sociedade moderna. Imagens digitais são um bom exemplo de categoria de informação cada vez mais importante em aplicações pessoais, comerciais, industriais e acadêmicas. Seja qual for a aplicação, porém, o manuseio de imagens digitais pode implicar em grande complexidade computacional, espaço para armazenamento e banda de transmissão. Velocidades de acesso em meios de armazenamento ou transmissão, usualmente, variam inversamente com a capacidade de armazenamento exigida. Portanto, surge a necessidade de desenvolver mecanismos eficientes para armazenamento, acesso e transmissão de imagens digitais.

Compressão de imagem é um ramo da ciência no qual métodos que exploram a significativa quantidade de redundância presente em imagens digitais para reduzir o número de bits necessários para representar a mesma imagem. Tal processo reduz a memória necessária para armazenamento e a banda necessária para transmissão da imagem digital e, consequentemente, facilita o acesso e transmissão da imagem. Os tipos de redundância são tipicamente classificados em espacial – relacionado à correlação entre *pixels* vizinhos e temporal – relacionados à correlação entre sucessivos quadros de imagens (RABBANI, 1995).

Técnicas de compressão de imagem são classificadas, essencialmente, como *lossless* (sem perdas) ou *lossy* (com perdas). Em compressão de imagens sem perdas, a imagem reconstruída após o processo de compressão é idêntica à original. Contudo, as taxas de compressão obtidas são usualmente modestas (RABBANI, 1995). Exemplos de técnicas de compressão sem perdas são: *Run length coding* e codificação de Huffman. A primeira, *Run length coding*, por exemplo, consiste no processo de representação da imagem digital no formato [**valor**, **comprimento**], onde **valor** corresponde ao valor numérico do *pixel* que se repete **comprimento** vezes.

Em compressão de imagens com perdas, a imagem reconstruída possui uma distorção associada quando comparada à imagem original (RABBANI, 1995). Contudo, técnicas de compressão com perda permitem alcançar taxas de compressão mais altas, e em alguns casos as taxas podem ser independentes das características estatísticas da imagem, como no método da quantização vetorial. Na quantização vetorial, a imagem original é incialmente segmentada em blocos. Então, um conjunto de blocos de número muito inferior, em relação ao número de blocos que compõem a imagem original, é utilizado para representar a imagem original da maneira mais fiel possível. Outros exemplos de técnicas de compressão de imagem com perdas

são: Codificação por sub banda (SBC, *Subband Coding* – em inglês) e Transformada Discreta de Cosseno (DCT, *Discrete Cosine Transform* – em inglês) (RABBANI, 1995).

O objetivo deste trabalho é implementar e avaliar a técnica de quantização vetorial, utilizando o algoritmo LBG, na compressão de imagens coloridas em três diferentes espaços de cor. A classe de imagens escolhida para aplicação da técnica foi o de mapas coloridos, uma vez que, além da cor, possuem uma riqueza de detalhes (símbolos, rios, linhas de nível, etc.) que evidenciam as vantagens e desvantagens do método aplicado.

A escolha do método de quantização vetorial para compressão de imagens foi motivada pelo fato de ser essa a técnica exigida pelo padrão CADRG (*Compressed ARC Digitized Raster Image* – em inglês), padrão utilizado em aplicações de mapa na empresa AEL Sistemas. O CADRG é de um padrão militar aberto de compressão de imagens georeferenciadas para uso em aplicações militares e civis, onde muitas vezes grandes quantidades de mapas devem ser carregadas por sistemas com capacidades limitadas (PEIXOTO, 2015). Por fim, o baixo nível de complexidade de uma verificação manual do funcionamento do algoritmo implementado foi o ponto crucial para escolha do algoritmo de LBG.

2 FUNDAMENTAÇÃO TEÓRICA

Nessa seção, são apresentados os conceitos, equações e algoritmos fundamentais para a implementação e compreensão do trabalho.

2.1 QUANTIZAÇÃO VETORIAL

Quantização vetorial é um método usualmente utilizado na compressão de dados. Porém, também encontra aplicações no campo de reconhecimento de padrões em sinais, classificação e extração de dados. Em quantização vetorial, o objetivo é representar certa distribuição de dados utilizando um número de protótipos significativamente menor que o número de dados, desse modo, quantizando a distribuição de dados original.

De maneira formal, o processo de quantização vetorial é definido através de um operador, o quatizador vetorial. Um quantizador vetorial, Q, de dimensão *k* e de tamanho N é definido como o mapeamento de um conjunto $\mathbf{I}$ de L vetores no espaço $\boldsymbol{\mathcal{R}}^k$, em um conjunto $\mathbf{C}$ que possui N vetores de saída, onde L ≫ N, contidos no mesmo espaço $\boldsymbol{\mathcal{R}}^k$ (GERSHO, 1992). Então,

$$Q: \mathbf{I} \rightarrow \mathbf{C}$$

onde, $\mathbf{I} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{L-1}\}$ com $\mathbf{x}_l \in \boldsymbol{\mathcal{R}}^k$. E, $\mathbf{C} = \{\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_{N-1}\}$ com $\mathbf{y}_i \in \boldsymbol{\mathcal{R}}^k$, $\forall\, i \in \mathbf{J} = \{0, 1, ..., N-1\}$. O conjunto **C** é denominado *codebook*, ou alfabeto de reprodução, e cada vetor que o compõe, $\mathbf{y}_i$, é denominado *codeword*.

O quantizador vetorial ainda pode ser visto como uma operação composta. Primeiramente, é realizado o mapeamento do conjunto **I**, de vetores de entrada, no conjunto **J**, que é o conjunto de índices relacionados às posições de cada *codeword* no *codebook*. Tal procedimento é denominado compressão e, usualmente, o operador responsável por efetuá-la é denominado *encoder*. Em seguida, o conjunto **I**, agora representado pelos elementos do conjunto **J**, é mapeado no conjunto **C**, ou *codebook*. Esse último processo é denominado descompressão e, usualmente, o operador responsável por efetuá-la é denominado *decoder*. Ao fim do processo de compressão e descompressão, tem-se que o conjunto de vetores de entrada foi vetorialmente quantizado, uma vez que N é muito menor que L (GERSHO, 1992).

No contexto de compressão de imagens, um vetor é um bloco de *k pixels* adjacentes, ou pontos adjacentes, caso a compressão seja feita independente em cada canal (ex.: os canais R, G e B são quantizados separadamente). E o conjunto **I** de vetores de entrada do quantizador vetorial é o resultado da decomposição em blocos da imagem de entrada. Para cada vetor de

entrada, o *encoder* varre o *codebook* de modo a determinar o *codeword* que melhor o representa. O critério para determinação do *codeword* ótimo para cada vetor de entrada é uma métrica de distorção. A medida usualmente utilizada na literatura, e utilizada nesse trabalho, é o Erro Quadrático (SE – *Squared Error*, em inglês), dada pela Expressão (2.1.1) (GERSHO, 1992):

$$SE(\mathbf{x},\mathbf{y}) = \sum_{j=0}^{k-1} \left\|\mathbf{x}_j - \mathbf{y}_j\right\|^2 \tag{2.1.1}$$

onde *k* é a dimensão do quantizador vetorial e a operação $\|\cdot\|$ corresponde à norma Euclidiana.

Outras métricas também são empregadas, tais como a norma de Minskowski (MN – *Minkowski Norm*, em inglês), dada pela Expressão (2.1.2), e o Erro Quadrático Ponderado (WSE – *Weighted Squared Error*, em inglês), dado pela Expressão (2.1.3) (LINDE, 1980).

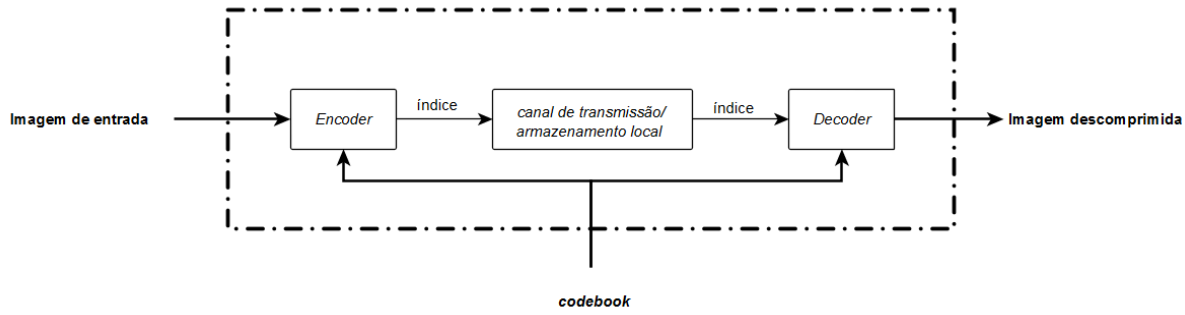
$$MN(\mathbf{x},\mathbf{y}) = \max_{0\le j<k} \left\|\mathbf{x}_j - \mathbf{y}_j\right\| \tag{2.1.2}$$

$$WSE(\mathbf{x},\mathbf{y}) = \sum_{j=0}^{k-1} w_j \cdot \left\|\mathbf{x}_j - \mathbf{y}_j\right\|^2 \tag{2.1.3}$$

onde $w_j \geq 0, j = 0, 1, \ldots, k-1$.

Ainda durante a compressão da imagem, uma vez identificado o *codeword* ótimo para um dado vetor de entrada, o índice correspondente ao *codeword* é enviado ao *decoder*, ou armazenado localmente, ao invés do próprio *codeword*. Por fim, seja o caso de transmitir a imagem comprimida ou salvá-la em um arquivo local, o *codebook* utilizado pelo *encoder* deve estar disponível para o processo de descompressão, que é o mapeamento inverso dos índices para os *codewords* correspondentes. A Figura 1 ilustra o processo de quantização vetorial.

Figura 1 – Ilustração do quantizador vetorial



O processo mais complexo computacionalmente e, além disso, o mais importante da quantização vetorial é o método pelo qual se obtém o *codebook*, pois é esse que determinará a qualidade da imagem comprimida. Nesse processo, uma imagem de treino, que pode ser a imagem a ser comprimida, ou um conjunto de vetores representativos da imagem, é analisada de modo a determinar o *codebook* que irá minimizar uma certa medida de distorção (SOUTHARD, 1992).

Um dos métodos para obtenção do *codebook*, é o algoritmo de Linde-Buzo-Gray (LBG), também conhecido como algoritmo generalizado de Lloyd (GLA – *Generalized-Lloyd's Algorithm*, em inglês) (SOUTHARD, 1992), uma vez que é a generalização vetorial do algoritmo escalar proposto por Lloyd (LLOYD, 1957). O algoritmo de LBG iterativamente ajusta um *codebook* inicial de modo a reduzir a medida de distorção até que um mínimo local, através de algum critério de convergência, seja atingido (LINDE, 1980).

Uma vez que a obtenção do *codebook* usualmente se dá ainda no lado do *encoder*, a maior parte da atividade computacional requerida para os processos de compressão e descompressão recai sobre o processo de compressão. A descompressão resume-se ao simples processo de *look-up*, ou seja, buscar numa tabela de referência (*codebook*) o índice recebido do *encoder*. Outros algoritmos tendem a exigir o mesmo nível de atividade computacional em ambos os lados, compressão e descompressão (AKRAMULLAH, 2014). Tal característica faz com que, na quantização vetorial, os requisitos em relação a poder de processamento do dispositivo no lado da descompressão não sejam tão altos.

A razão de compressão (CR – *Compression Ratio*, em inglês) obtida empregando-se o processo de quantização vetorial é independente das propriedades estatísticas do sinal de entrada (SOUTHARD, 1992). Uma vez fixada a dimensão do quantizador vetorial, *k*, o tamanho do *codebook*, e tendo-se conhecimento das dimensões da imagem, determina-se também a razão de compressão. Tal propriedade simplifica o gerenciamento de espaço para armazenamento, ou

banda requerida para transmissão de dados. Para a aplicação em compressão de imagens, por exemplo, a razão de compressão (RC) é calculada como sendo a razão entre o número de bytes utilizados para representar cada vetor que compõe a imagem de entrada sobre o número de bytes necessários para representar os índices dos *codewords*, que compõe o *codebook*, dada pela Expressão (2.1.4):

$$RC = \frac{k \cdot c}{\left\lceil \frac{\log_2 N}{8} \right\rceil} \tag{2.1.4}$$

onde *c* é o número de componentes de cor, *k* representa a dimensão dos vetores, $\lceil \cdot \rceil$ corresponde à operação de arredondamento para cima e N é o tamanho do *codebook*.

Para casos em que um novo *codebook* é gerado para cada imagem de entrada, deve-se levar em consideração o espaço ocupado pelo *codebook* no cálculo da razão de compressão. A Expressão (2.1.5) dá a razão de compressão considerando a geração de um *codebook* para cada nova imagem de entrada.

$$RC = \frac{k \cdot c \cdot \mathrm{L}}{L \cdot \left\lceil \frac{\log_2 N}{8} \right\rceil + k \cdot c \cdot N} \tag{2.1.5}$$

onde L é o número de vetores que compõem a imagem de entrada. Observe que a Expressão (2.1.5) apresenta dependência da razão de compressão em função das dimensões da imagem. Tal dependência, porém, pode ser desconsiderada quando $\mathrm{L} \gg \mathrm{N}$, caso em que a Expressão (2.1.5) se reduz à Expressão (2.1.4).

Por fim, as Expressões (2.1.4) e (2.1.5) levam em consideração que as componentes de cor são comprimidas em conjunto, ou seja, cada elemento do vetor é um *pixel* em um determinado espaço de cor. Porém, caso seja feita a compressão de cada canal, ou componente de cor, separadamente, deve-se modificar as expressões de modo a levar em conta as diferentes dimensões de vetor e tamanho de *codebook*, utilizados para a compressão de cada canal. As Expressões (2.1.6) e (2.1.7) são as versões adaptadas das Expressões (2.1.4) e (2.1.5), respectivamente.

$$RC = \frac{\sum_{i=1}^{3} k_i}{\sum_{i=1}^{3} \left\lceil \frac{\log_2 N_i}{8} \right\rceil} \tag{2.1.6}$$

$$RC = \frac{\sum_{i=1}^{3} k_i \cdot \mathrm{L_i}}{\sum_{i=1}^{3} L_i \cdot \left\lceil \frac{\log_2 N_i}{8} \right\rceil + k_i \cdot N_i} \tag{2.1.7}$$

onde *i* refere-se ao i-ésimo canal da imagem de entrada. Observe que, nas Expressões (2.1.6) e (2.1.7), o tamanho, em *bytes*, ocupado originalmente pela imagem (numerador) e após a quantização vetorial (denominador) são dados pela soma das contribuições de cada canal, ou componente de cor.

2.2 ALGORITMO DE LBG

O algoritmo de Linde-Buzo-Gray, é um método para obtenção de *codebook* utilizado no processo de quantização vetorial. O método iterativamente aperfeiçoa um *codebook* inicial, baseando-se no conjunto de vetores de treino e na medida de distorção adotada. A convergência do método é atingida quando a variação da distorção total entre uma iteração e outra é inferior a uma tolerância previamente estabelecida. Uma vez que o algoritmo converge, tem-se que uma solução ótima local para o problema de determinação do alfabeto de reprodução foi encontrada. Além disso, o *codebook* é uma condição inicial para obtenção da solução, e irá determinar o quão distante da solução ótima global se encontrará o alfabeto determinado pelo algoritmo.

Duas maneiras possíveis de inicialização de *codebook*, e que são exploradas neste trabalho, são: inicialização por amostragem e inicialização por *splitting*. Na inicialização por amostragem, cada elemento do *codebook* é selecionado amostrando-se, em intervalos regulares, Δs, o vetor de treino (EQUITZ, 1989). O intervalo de amostragem Δs é calculado através da Expressão (2.2.1):

$$\Delta s = \left\lfloor \frac{L}{N} \right\rfloor \tag{2.2.1}$$

onde L e N são, respectivamente, o tamanho do vetor de treino e do *codebook*.

No método de inicialização por *splitting*, o *codebook* inicial tem tamanho unitário e é dado pelo centroide do conjunto formado por todos os vetores de treino. Desse centroide são derivados dois vetores, obtendo-se então um *codebook* de tamanho igual a dois. O algoritmo de LBG é então aplicado, seguido de uma nova divisão que resultará em um alfabeto de reprodução

de quatro elementos. Esse processo, descrito nos passos abaixo, é repetido até obter-se o número de *codewords* desejado (LINDE, 1980).

INICIALIZAÇÃO POR SPLITTING:

Passo 1 Faça N = 1. Calcule o centroide do conjunto formado por todos os vetores de treino, e inicialize o *codebook* com esse valor, conforme a Expressão (2.2.2):

$$C^{(0)} = \left\{ \left\lfloor \frac{1}{L} \cdot \sum \mathbf{x} \right\rfloor \right\} \tag{2.2.2}$$

onde C $^{(0)}$ é o *codebook* inicial, L é o tamanho do conjunto de vetores de treino e o operador $\lfloor \cdot \rfloor$ corresponde à operação de arredondamento para baixo.

Passo 2 Divida cada *codeword* $\mathbf{y_i}$ do alfabeto de reprodução da iteração passada em dois novos *codewords,* conforme a Expressão (2.2.3), e, então, faça N = 2N.

$$\begin{gathered} y_{2i}^{(m)} = y_i^{(m-1)} \\ y_{2i+1}^{(m)} = y_i^{(m-1)} + \mathbf{d} \\ C^{(m)} = \left\{ \mathbf{y}_0^{(m-1)}, \mathbf{y}_0^{(m-1)} + \mathbf{d}, \cdots, \mathbf{y}_i^{(m-1)}, \mathbf{y}_i^{(m-1)} + \mathbf{d} \right\}, \ \forall\, i = 0, 1, \ldots, N-1 \end{gathered} \tag{2.2.3}$$

onde **d** é uma perturbação da mesma dimensão que os *codewords,* e *m* é a iteração atual.

Passo 3 Se o tamanho do alfabeto de reprodução, N, é o desejado, tem-se que a inicialização finalizou. Caso contrário o método prossegue para o Passo 4.

Passo 4 Atualize o novo *codebook* aplicando o algoritmo de LBG e retorne ao Passo 2.

A perturbação **d**, utilizada na inicialização por *splitting* e sugerida por Linde, em (LINDE, 1980), é fixa. Porém, é possível determinar **d** a cada divisão seguindo algum critério de minimização de distorção. Nesse trabalho, assim como sugerido por Ivanov em (IVANOV, 2001), **d** é determinado a cada divisão de modo a minimizar o máximo erro associado a cada *codeword*. A Expressão (2.2.4) é utilizada para o cálculo de **d.**

$$\mathbf{d} = \left\lfloor \frac{\mathbf{x}_{max} - \mathrm{y}_i}{2} \right\rfloor \tag{2.2.4}$$

onde $\mathbf{x}_{max}$ é o vetor de treino associado ao i-ésimo *codeword*, que possui a maior distância quadrática (ou erro quadrático). Dessa forma, a cada divisão, um *codeword* se desprende no sentido do vetor de treino mais distante do *codeword* original.

Após selecionados os N *codewords* através do método de inicialização por amostragem, ou por *splitting*, o algoritmo LBG irá iterativamente refiná-lo.

ALGORITMO LBG:

Passo 1 Para cada vetor que compõe o conjunto de vetores de treino, determina-se o *codeword* que o representa com a menor distorção. A métrica utilizada, nesse caso, é o erro quadrático, dada pela Expressão (2.1.1).

Passo 2 Calcula-se a distorção total somando os erros quadráticos entre cada vetor de treino e seu respectivo *codeword* ótimo (determinado no Passo 1). Caso a diferença relativa, Δd, entre a distorção total da iteração corrente e a da iteração passada, dada pela Expressão (2.2.5), seja menor que uma tolerância, ε, ou a distorção total da iteração corrente seja nula, tem-se que o algoritmo convergiu. Caso contrário o algoritmo prossegue para o Passo 3.

$$\Delta d = \frac{D^{m-1} - D^{m}}{D^{m-1}} \tag{2.2.5}$$

onde D e *m* são respectivamente, a distorção total e a iteração corrente.

Passo 3 A cada *codeword*, $\mathbf{y}_i$, estarão associados R_i vetores de treino. Para cada *codeword* com R_i não nulo, seu respectivo valor é atualizado com o centroide do conjunto formado pelos vetores a ele associados, dado pela Expressão (2.2.6). Nas ocasiões em que R_i é nulo, o *codeword* não é atualizado (LINDE, 1980).

$$\bar{R}_i = \frac{1}{R_i} \cdot \sum_{\mathbf{i=1}}^{L} \mathbf{x_i} \tag{2.2.6}$$

onde R_i é o número de vetores de treino, **x**, associados ao i-ésimo *codeword*, $\bar{R}_i$ é o centroide do conjunto formado pelos R_i vetores e L é o tamanho do conjunto de vetores de treino.

Por fim, a convergência do algoritmo LBG depende do *codebook* inicial e da tolerância ε provida. Em implementação, usualmente, determina-se um número máximo de iterações, K_{max}, para o algoritmo LBG (CHEN, 2014). Dessa forma, tem-se que o tempo consumido pelo algoritmo na compressão de imagens coloridas é da ordem de *O*, dado pela Expressão (2.2.7):

$$O = \sum_{i=1}^{3} (\alpha_i \cdot L_i \cdot k_i \cdot N_i) \qquad (2.2.7)$$

onde α_i é o número de iterações necessárias para a convergência do algoritmo no i-ésimo componente de cor, e está no intervalo [1, K_{max}]. As variáveis N_i, k_i, e L_i são o tamanho do *codebook*, dimensão do quantizador vetorial, e tamanho do vetor de treino do i-ésimo canal, respectivamente.

2.3 ESPAÇOS DE COR

O sistema utilizado para a representação de cor, ou espaço de cor, é um fator com significativa implicação nos resultados obtidos nos processos de compressão de imagem, seja em termos de fidelidade de cor, contraste ou inteligibilidade da imagem final. Considerando, por exemplo, que o sistema de visão humano é menos sensível à perda de informação de cor, é de interesse utilizar espaços de cor que possibilitem a aplicação de taxas de compressão, nos canais correspondentes às componentes de cor, superiores aos aplicados no canal de intensidade. Dessa forma, razões de compressão superiores podem ser alcançadas sem que a qualidade visual da imagem comprimida seja significativamente afetada (AKRAMULLAH, 2014).

Os sistemas de cor são usualmente classificados em: orientado a dispositivo ou orientados ao usuário. O sistema de representação RGB, por exemplo, é classificado como orientado a dispositivo. Os valores de cada cor primária no sistema RGB podem ser relacionados, por exemplo, às tensões aplicadas aos canhões de elétrons de um CRT (*Cathode Ray Tube* – em inglês) (SOUTHARD, 1992). Contudo, sistemas orientados a dispositivo não são intuitivos e, assim como é o caso do sistema de cor RGB, não possuem canais exclusivos de croma e, conseguintemente, impossibilitam que a informação de cor seja quantizada independente da luminância.

Dois exemplos de espaços de cores ditos perceptuais, ou orientados ao usuário, e que possuem canais exclusivos para a informação de cor, são o HSV e CIE Lab (SOUTHARD, 1992). Na Seção 2.3.1, são apresentadas as propriedades do espaço de cor HSV, bem como o procedimento para realizar a transformação entre os espaços de cor RGB e HSV. As características do espaço de cor CIE Lab, bem como o procedimento utilizado para transformação entre os espaços de cor Lab e RGB, são apresentados na Seção 2.3.3. E, uma vez

que o sistema de cor CIE Lab foi desenvolvido a partir do espaço de cor CIE XYZ, uma apresentação sobre as propriedades do sistema CIE XYZ e procedimento para realizar a transformação entre os espaços de cor RGB e CIE XYZ é feita na Seção 2.3.2.

2.3.1 ESPAÇO DE COR HSV

As componentes de cor no espaço HSV são representadas por matiz (*Hue* – em inglês), saturação e valor (ou brilho). O espaço de cor HSV é tradicionalmente representado por uma pirâmide invertida de base hexagonal, assim como ilustrado na Figura 2, porém o espaço HSV, na realidade, tem o formato de um cilindro, conforme ilustrado pela Figura 3. Em coordenadas cilíndricas, a altura é dada pelo valor (V), o raio é dado pela saturação (S) e o azimute é definido pelo matiz (H). Nesse sistema de cores, o preto corresponde ao ponto **S** da Figura 3, onde saturação e valor são nulos, e o branco corresponde ao ponto **W** da mesma figura, onde o brilho é máximo e a saturação é nula (BURGER, 2009).

Para a conversão do sistema de cores RGB para o espaço de cores HSV, inicialmente calcula-se a saturação das componentes de cores, S_{HSV}, dada pela Expressão (2.3.1.1).

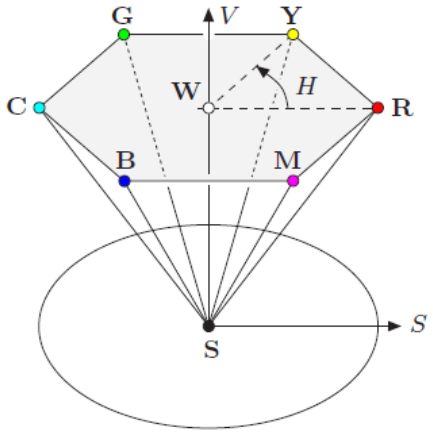
$$S_{HSV} = \begin{cases} \dfrac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B)} & ,\max(R,G,B) > 0 \\ 0 & ,caso\ contrário \end{cases} \qquad (2.3.1.1)$$

Em seguida, determina-se o valor, V_{HSV}, através da Expressão (2.3.1.2).

$$V_{HSV} = \frac{\max(R,G,B)}{2^n - 1} \qquad (2.3.1.2)$$

onde *n* é o número de bits utilizados na representação de cada componente de cor.

Figura 2 – Espaço de cores HSV representado por pirâmide de base hexagonal



Fonte: BURGER, 2009

Por fim, para cálculo do matiz, primeiramente considera-se o caso em que todas as componentes de cor são iguais. Neste caso, a saturação S_{HSV} é nula e, portanto, o ângulo H_{HSV}, ou matiz, é indefinido. Tais casos correspondem aos *pixels* acromáticos, ou de tom cinza. Em seguida, para cálculo do matiz quando a saturação, S_{HSV}, é diferente de zero, define-se os valores normalizados das componentes de cor, R', G' e B', de acordo com as Expressões (2.3.1.3), (2.3.1.4) e (2.3.1.5), respectivamente (BURGER, 2009).

$$R' = \frac{\max(R,G,B) - R}{\max(R,G,B) - \min(R,G,B)} \qquad (2.3.1.3)$$

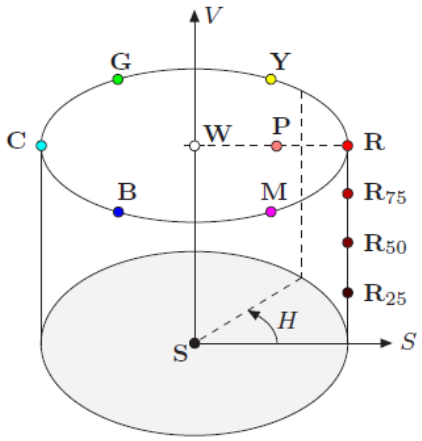
$$G' = \frac{\max(R,G,B) - G}{\max(R,G,B) - \min(R,G,B)} \qquad (2.3.1.4)$$

$$B' = \frac{\max(R,G,B) - B}{\max(R,G,B) - \min(R,G,B)} \qquad (2.3.1.5)$$

Utilizando os valores normalizados das componentes de cor, calcula-se o matiz através da Expressão (2.3.1.6). O valor de H_{HSV}, bem como os valores de S_{HSV} e V_{HSV}, estão contidos no intervalo [0, 1] (BURGER, 2009).

$$H_{HSV} = \frac{1}{6} \cdot \begin{cases} B' - G' & , \quad se\ \max(R,G,B) = R\ e\ G \geq B \\ B' - G' + 6\,, & se\ \max(R,G,B) = R\ e\ B > G \\ R' - B' + 2\,, & se\ \max(R,G,B) = G \\ G' - R' + 4\,, & se\ \max(R,G,B) = B \end{cases} \tag{2.3.1.6}$$

Figura 3 – Espaço de cores HSV em forma cilíndrica



Fonte: BURGER, 2009

Para que o resultado da compressão da imagem, realizada no espaço de cores HSV, possa ser exibido em um computador pessoal, dispondo dos recursos básicos de um sistema operacional tradicional, é necessário converter as componentes de cor para o espaço RGB. Para tanto, é necessário classificar a tupla (H_{HSV}, S_{HVS}, V_{HVS}) de acordo com o valor H_p, dado pela Expressão (2.3.1.7) (BURGER, 2009).

$$H_p = (6 \cdot H_{HSV}) \bmod 6 \tag{2.3.1.7}$$

onde *mod* corresponde à operação modulo, que retorna o resto da divisão entre dois números.

Então, define-se as variáveis auxiliares *x*, *y* e *z*, na Expressão (2.3.1.8), para convenientemente determinar a Expressão (2.3.1.9), utilizada para calcular os valores normalizados, no intervalo [0, 1], R_n, G_n e B_n das componentes de cor (BURGER, 2009).

$$\begin{aligned} x_{HSV} &= (1 - S_{HSV}) \cdot V_{HVS} \\ y_{HSV} &= \left(1 - S_{HSV} \cdot \left(H_p - \lfloor H_p \rfloor\right)\right) \cdot V_{HVS} \end{aligned} \tag{2.3.1.8}$$

$$z_{HSV} = \left(1 - S_{HSV} \cdot \left(\lfloor H_p \rfloor - H_p + 1\right)\right) \cdot \mathrm{V}_{HVS}$$

onde $\lfloor . \rfloor$ corresponde à operação de arredondamento para baixo.

$$\langle R_n | G_n | B_n \rangle = \begin{cases} \langle V_{HSV} | z_{HSV} | x_{HSV} \rangle & , se\ \lfloor H_p \rfloor = 0 \\ \langle y_{HSV} | V_{HSV} | x_{HSV} \rangle & , se\ \lfloor H_p \rfloor = 1 \\ \langle x_{HSV} | V_{HSV} | z_{HSV} \rangle & , se\ \lfloor H_p \rfloor = 2 \\ \langle x_{HSV} | y_{HSV} | V_{HSV} \rangle & , se\ \lfloor H_p \rfloor = 3 \\ \langle z_{HSV} | x_{HSV} | V_{HSV} \rangle & , se\ \lfloor H_p \rfloor = 4 \\ \langle V_{HSV} | x_{HSV} | y_{HSV} \rangle & , se\ \lfloor H_p \rfloor = 5 \end{cases} \quad (2.3.1.9)$$

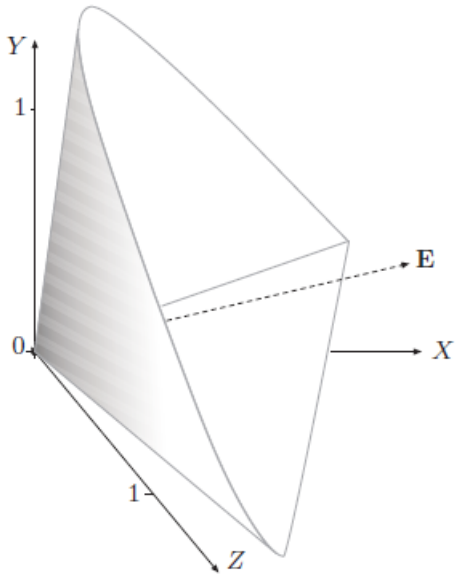
Finalmente, usando a Expressão (2.3.1.10), obtém-se os valores RGB dentro do intervalo de valores determinado por *n* (BURGER, 2009).

$$\begin{aligned} R &= \min([2^n \cdot R_n], 2^n - 1) \\ G &= \min([2^n \cdot G_n], 2^n - 1) \\ B &= \min([2^n \cdot B_n], 2^n - 1) \end{aligned} \quad (2.3.1.10)$$

2.3.2 ESPAÇO DE COR CIE XYZ

O sistema de cores CIE XYZ, desenvolvido pela *Commission Internationale d'Èclairage* entre as décadas de 20 e 30, foi criado a partir de uma série de medidas que exploravam as características do sistema de visão humana. As cores primárias desse sistema formam uma tupla de três componentes abstratas X, Y e Z, escolhidas de tal modo que todas as cores visíveis pudessem ser representadas pela combinação de valores positivos das componentes da tupla. A Figura 4 ilustra o formato do espaço de cores CIE XYZ (BURGER, 2009).

Figura 4 – Espaço de cores CIE XYZ



Fonte: BURGER, 2009

No sistema CIE XYZ, a luminância é determinada pela componente Y e, portanto, a origem, ou vértice, do espaço em forma de cone corresponde à luminância nula, ou preto. O CIE ainda define valores para crominância *x*, *y* e *z*, dados na Expressão (2.3.2.1), que são utilizados para representar matiz e saturação (KERR, 2010).

$$x = \frac{X}{X + Y + Z}$$

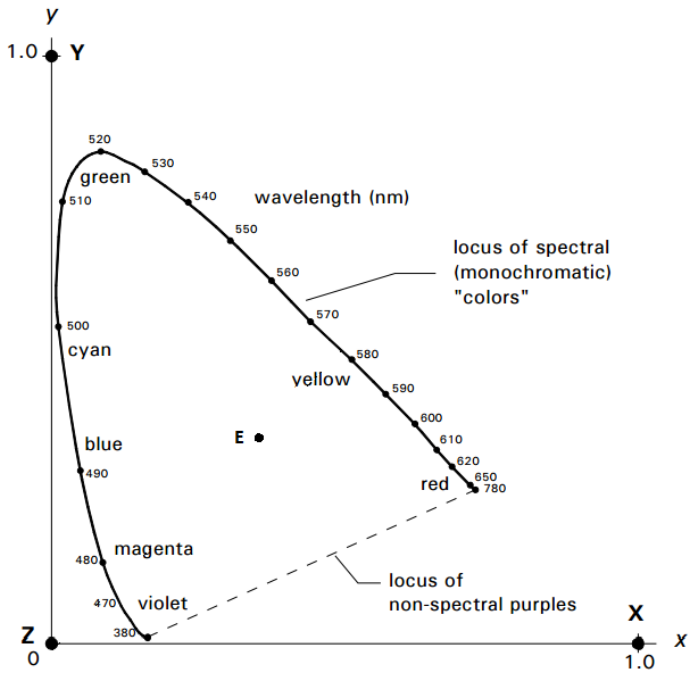
$$y = \frac{Y}{X + Y + Z} \qquad (2.3.2.1)$$

$$z = \frac{Z}{X + Y + Z}$$

Dado um ponto qualquer **P** = (X_o, Y_o, Z_o), no espaço de cores CIE XYZ, o ponto que corresponde aos valores de crominância, **C** = (x_o, y_o, z_o), é determinado a partir das relações apresentadas na Expressão (2.3.2.1). E, uma vez que a combinação linear $x + y + z = 1$ é verdadeira, tem-se que uma das componentes de crominância pode ser descartada. O CIE descarta a componente *z*, o que corresponde à projeção do ponto **C** sobre o plano XY, formando assim o diagrama de cromaticidade do CIE, ilustrado na Figura 5 (BURGER, 2009).

No diagrama, qualquer ponto (*x*, *y*) dentro do espaço delimitado pela linha em negrito representa um valor de matiz e saturação de uma cor visível. Os pontos que residem exatamente sobre a linha que delimita o espaço correspondem às cores monocromáticas com máxima saturação, com exceção daqueles que residem sobre a linha pontilhada. O ponto **E** corresponde ao ponto de referência, ou ponto neutro, no qual a saturação é nula. Logo, de um ponto qualquer sobre a linha limítrofe do espaço convexo definido pela linha em negrito, a saturação decai continuamente à medida que se aproxima do ponto **E** (BURGER, 2009).

Figura 5 – Diagrama de cromaticidade CIE xyY



Fonte: KERR, 2010

Dado um ponto (x_a, y_a) no diagrama de cromaticidade do CIE, obtém-se o ponto **C** = (x_a, y_a, z_a), definido no espaço XYZ, fazendo-se $z_a = 1 - x_a - y_a$. Contudo, uma vez que qualquer ponto **P** = (αX_a, αY_a, αZ_a) resulta no mesmo ponto de cromaticidade **C** para qualquer valor de α maior que zero, é necessário prover informação adicional para se obter os valores (X_a, Y_a, Z_a) originais a partir de **C**. Usualmente, define-se uma componente de cor no sistema CIE XYZ através dos valores de cromaticidade, *xy*, e o valor original de luminância, Y. Desse modo, a tupla (X_a, Y_a, Z_a) original pode ser determinada através das relações dadas pela Expressão (2.3.2.2) (BURGER, 2009).

$$X_a = x_a \cdot \frac{Y_a}{y_a}$$

$$Y_a = Y_a \qquad (2.3.2.2)$$

$$Z_a = \frac{1 - x_a - y_a}{y_a} \cdot Y_a$$

A transformação do espaço de cores RGB para o espaço CIE XYZ depende de um ponto de referência $C_{ref} = (X_{ref}, Y_{ref}, Z_{ref})$, a partir do qual as coordenadas de cromaticidade das cores primárias, vermelho (x_r y_r), verde (x_g, y_g) e azul (x_b, y_b), do sistema RGB sejam conhecidas (LINDBLOOM, 2015). O CIE define iluminantes padrões, tais como os iluminantes padrões de luz do dia série D, determinados com base na combinação de diferentes medições que variam com localização geográfica e condições atmosféricas e climáticas, que são utilizadas como pontos de referência durante a conversão entre espaços de cores que envolvam o espaço CIE XYZ (KERR, 2010). Na Tabela 1, estão dispostos os valores da tupla (X, Y, Z) bem como da coordenada de cromaticidade do ponto de referência para os iluminantes **D_{50}** e **D_{65}**.

Tabela 1 – Parâmetros do sistema CIE XYZ para os iluminantes padrões D_{50} e D_{65}

	X	**Y**	**Z**	*x*	y
D_{50}	0.964296	1.000000	0.825105	0.3457	0.3585
D_{65}	0.950456	1.000000	1.088754	0.3127	0.3290

Fonte: BURGER, 2009

A conversão do espaço de cores RGB para o espaço de cores CIE XYZ é feita através da matriz de transformação M, e a conversão inversa, isto é, do espaço de cores CIE XYZ para o espaço RGB é feita através da matriz inversa M^{-1}. Ambas conversões estão definidas na Expressão (2.3.2.3), e a matriz de transformação, M, é dada pela Expressão (2.3.2.4) (LINDBLOOM, 2015).

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = M \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M^{-1} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad (2.3.2.3)$$

$$M = \begin{bmatrix} S_r \cdot \frac{x_r}{y_r} & S_g \cdot \frac{x_g}{y_g} & S_b \cdot \frac{x_b}{y_b} \\ S_r & S_g & S_b \\ S_r \cdot \frac{1 - x_r - y_r}{y_r} & S_g \cdot \frac{1 - x_g - y_g}{y_g} & S_b \cdot \frac{1 - x_b - y_b}{y_b} \end{bmatrix} \quad (2.3.2.4)$$

onde S_r, S_g e S_b são obtidas através do sistema dado pela Expressão (2.3.2.5).

$$\begin{bmatrix} S_r \\ S_g \\ S_b \end{bmatrix} = \begin{bmatrix} \frac{x_r}{y_r} & \frac{x_g}{y_g} & \frac{x_b}{y_b} \\ 1 & 1 & 1 \\ \frac{1 - x_r - y_r}{y_r} & \frac{1 - x_g - y_g}{y_g} & \frac{1 - x_b - y_b}{y_b} \end{bmatrix}^{-1} \cdot \begin{bmatrix} X_{ref} \\ Y_{ref} \\ Z_{ref} \end{bmatrix} \quad (2.3.2.5)$$

Nesse trabalho, utilizou-se o iluminante padrão D_{65}, e os valores de crominância de cada cor primária do espaço de cor RGB dispostos na Tabela 2, para obtenção da matriz de transformação, M. A configuração escolhida é uma das recomendações ITU-R BT.709, que definem valores padrões para os parâmetros de imagem da TV de alta resolução, e trata-se da mesma configuração utilizada pelo algoritmo utilizado como referência na transformação entre os sistemas RGB e CIE Lab, assim como mencionado na Seção 3.1.

Tabela 2 – Valores de crominância das componentes do espaço de cor RGB

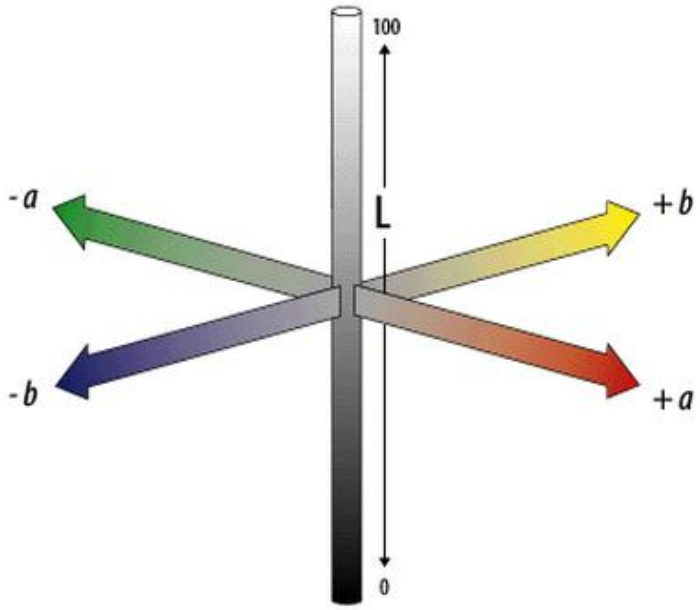
Cor primária	**Crominância**	
R	0.64	0.33
G	0.30	0.60
B	0.15	0.06

Fonte: http://www.itu.int

2.3.3 ESPAÇO DE COR CIE Lab

O sistema de cores Lab foi desenvolvido com o objetivo de tornar uniforme, em relação à percepção humana, a representação da cor obtida através do espaço de cor CIE XYZ. As dimensões no espaço de cor Lab são: luminância, L, e duas componentes de cor, *a* e *b*. Na coordenada de crominância *a*, os valores positivos e negativos correspondem, respectivamente, ao vermelho e verde. Já na coordenada de crominância *b*, os valores positivos e negativos correspondem, respectivamente ao amarelo e azul. A Figura 6 ilustra as coordenadas/dimensões do sistema de cores CIE Lab.

Figura 6 – Ilustração das coordenadas do espaço de cor CIE Lab



Fonte: http://dba.med.sc.edu/price/irf/Adobe_tg/models/cielab.html

A transformação entre os espaços de cor RGB e CIE Lab é realizada em dois passos. Inicialmente é efetuada a transformação para o espaço CIE XYZ e, então, a conversão para o sistema CIE Lab utilizando as equações dispostas na Expressão (2.3.3.1) (BURGER, 2009):

$$\begin{aligned} L &= 116 \cdot Y' - 16 \\ a &= 500 \cdot (X' - Y') \\ b &= 200 \cdot (Y' - Z') \end{aligned} \qquad (2.3.3.1)$$

onde Y', X' e Z' são dados pelas equações presentes na Expressão (2.3.3.2):

$$\begin{aligned} Y' &= f\left(\frac{Y}{Y_{ref}}\right) \\ X' &= f\left(\frac{X}{X_{ref}}\right) \\ Z' &= f\left(\frac{Z}{Z_{ref}}\right) \end{aligned} \qquad (2.3.3.2)$$

onde a função *f* é definida na Expressão (2.3.3.3):

$$f(k) = \begin{cases} k^{\frac{1}{3}} & para\ k > 0.008856 \\ 7.787 \cdot k + \dfrac{16}{116} & para\ k \leq 0.008856 \end{cases} \qquad (2.3.3.3)$$

A conversão inversa, isto é, do espaço de cor CIE Lab para o espaço de cor RGB também é realizada em dois passos. Primeiramente, as tuplas (L, a, b) são transformadas para o espaço CIE XYZ e, então, são finalmente convertidas para o espaço RGB. A conversão para o sistema CIE XYZ, do sistema CIE Lab, é efetuada utilizando as equações dispostas na Expressão (2.3.3.4) (BURGER, 2009):

$$\begin{aligned} X &= X_{ref} \cdot g\left(\frac{a}{500} + Y'\right) \\ Y &= Y_{ref} \cdot g(Y') \\ Z &= Z_{ref} \cdot g\left(Y' - \frac{b}{200}\right) \end{aligned} \qquad (2.3.3.4)$$

onde Y’ e a função *g* são definidos, respectivamente, pelas Expressões (2.3.3.5) e (2.3.3.6).

$$Y' = \frac{L + 16}{116} \qquad (2.3.3.5)$$

$$g(k) = \begin{cases} k^3 & para\ k > 0.008856 \\ \dfrac{k - \frac{16}{116}}{7.787} & para\ k \leq 0.008856 \end{cases} \qquad (2.3.3.6)$$

Os valores de referências X_{ref}, Y_{ref} e Z_{ref} correspondem, nesse trabalho, ao iluminante padrão D_{65}, apresentado na Seção 2.3.2 e dispostos na Tabela 1. Na mesma seção, são apresentados os procedimentos para as transformações entre os espaços de cores RGB e CIE XYZ utilizados nesta seção para conversão entre os sistemas de cor RGB e CIE Lab.

2.4 CRITÉRIOS PARA AVALIAÇÃO DE QUALIDADE DA QUANTIZAÇÃO

Uma vez que o método de compressão por quantização vetorial é classificado como com perdas, é necessário um estudo sobre os critérios e métricas utilizadas para avaliar o compromisso entre a distorção e a razão de compressão obtidas.

Três medidas usualmente utilizadas são, o MSE (*Mean Squared Error*), o PSNR (*Peak Signal-to-Noise Ration*) (VELDHUIZEN, 1998) e o NMSE (*Normalized Mean Squared Error*) (MANDAL, 2003). O erro quadrático médio (MSE) entre a imagem original, $\mathbf{I}_{\mathrm{M\,x\,N}}$, e sua versão comprimida, $\hat{\mathbf{I}}_{\mathrm{M\,x\,N}}$, é dado pela Expressão (2.4.1).

$$MSE\left(\mathbf{I},\hat{\mathbf{I}}\right) = \frac{1}{M}\cdot\frac{1}{N}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left\|\mathbf{I}_{i,j} - \hat{\mathbf{I}}_{i,j}\right\|^{2} \tag{2.4.1}$$

onde M e N são as dimensões espaciais das imagens de entrada e saída, e $\|\cdot\|$ é a norma Euclidiana de um vetor.

Observa-se, porém, que o valor resultante do MSE está fortemente relacionado ao escalonamento da intensidade da imagem. Um erro significativo para uma imagem cuja intensidade dos *pixels* é representada por 8 *bits*, pode ser desprezível para uma outra imagem com escalonamento em 12 *bits* (VELDHUIZEN, 1998). O PSNR, dado pela Expressão (2.4.2), mitiga essa dependência ao normalizar o valor do MSE obtido pela energia do valor máximo que possa ser atribuído a um *pixel* (por exemplo, 255 para uma representação em 8 *bits*).

$$PSNR_{dB}\left(\mathbf{I},\hat{\mathbf{I}}\right) = -10\cdot\log_{10}\left[\frac{MSE\left(\mathbf{I},\hat{\mathbf{I}}\right)}{P^{2}}\right] \tag{2.4.2}$$

onde MSE é o erro quadrático médio dado pela Expressão (2.4.1), e P é o máximo valor de um *pixel*.

O erro quadrático médio normalizado (NMSE), dado pela Expressão (2.4.3) (MANDAL, 2003), leva em consideração a energia dos *pixels* das imagens original e comprimida ao efetuar a normalização do MSE. E, portanto, contorna os pontos negativos do MSE e PSNR e torna-se a escolha ideal para avaliação quantitativa dos resultados de compressão de imagem. Esse método, porém, implica em maior complexidade computacional quando comparado ao PSNR ou MSE.

$$NMSE(\mathbf{I}, \hat{\mathbf{I}}) = \frac{1}{M} \cdot \frac{1}{N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \frac{\left\| \mathbf{I}_{i,j} - \hat{\mathbf{I}}_{i,j} \right\|^2}{\left\| \mathbf{I}_{i,j} \right\|^2 \cdot \left\| \hat{\mathbf{I}}_{i,j} \right\|^2} \tag{2.4.3}$$

3 PROCEDIMENTO EXPERIMENTAL

Nessa seção, são descritas as ferramentas utilizadas para confecção do trabalho, os procedimentos de verificação de funcionalidade e os detalhes de cada experimento realizado, sendo eles: quantização vetorial no espaço de cores RGB, HSV e CIE Lab e comparação de performance entre algoritmo JPG e LBG.

3.1 FERRAMENTAS UTILIZADAS

O trabalho, no que concerne a implementação dos algoritmos, foi desenvolvido utilizando C/C++ como linguagem de programação. A imagem comprimida foi armazenada em arquivo binário utilizando a função fwrite, e a descompressão do arquivo foi feita utilizando a função fread. Logo, para facilitar o processo de compressão e descompressão, o tamanho do *codebook* foi fixado em N = 256, já que tal valor corresponde ao número de níveis representáveis por um byte, que por sua vez é a menor unidade de informação digital que pode ser gravada em arquivo utilizando-se a função fwrite. Por fim, as medidas de tempo foram efetuadas através da função clock, da biblioteca time.h.

O OpenCV é uma biblioteca livre que fornece recursos para aplicações em visão computacional e aprendizagem de máquina. Os recursos da biblioteca do OpenCV foram utilizados para decodificar os arquivos de imagem, dentro das extensões suportadas pela biblioteca, e para escrever a imagem comprimida pelo algoritmo desenvolvido em um formato reconhecido pelo sistema operacional (*.bmp*, por exemplo). Além disso, os resultados de conversão entre espaços de cores obtidos utilizando-se as funções do OpenCV foram utilizados como referência na verificação das funções implementadas.

A imagem de teste, utilizada nos procedimentos experimentais das Seções 3.2 e 3.5, é um recorte de um mapa do tipo TPC (*Tactical Pilotage Chart* – em inglês), obtido do acervo digital de mapas de domínio público da biblioteca da Universidade de Texas em Austin. A Figura 7 ilustra a imagem de teste e a Tabela 3 apresenta informações relevantes da imagem, tais como dimensões, extensão, profundidade de *pixel* e espaço em disco.

Tabela 3 – Propriedades da imagem de teste

Dimensões [largura x altura]	**Extensão**	**Profundidade de *pixel***	**Tamanho**
2312 x 1576	.bmp	24	10675 KBytes

Figura 7 – Recorte de mapa TPC da região nordeste do estado do RS

Fonte: http://www.lib.utexas.edu/maps/tpc/

Por fim, as versões de compilador C/C++ e de OpenCV, bem como as configurações do *hardware* utilizado são:

- Compilador Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 16.00;
- OpenCV 3.0.0;
- Processador i7 @ 2.9 GHz;
- Memória RAM de 12 GB.

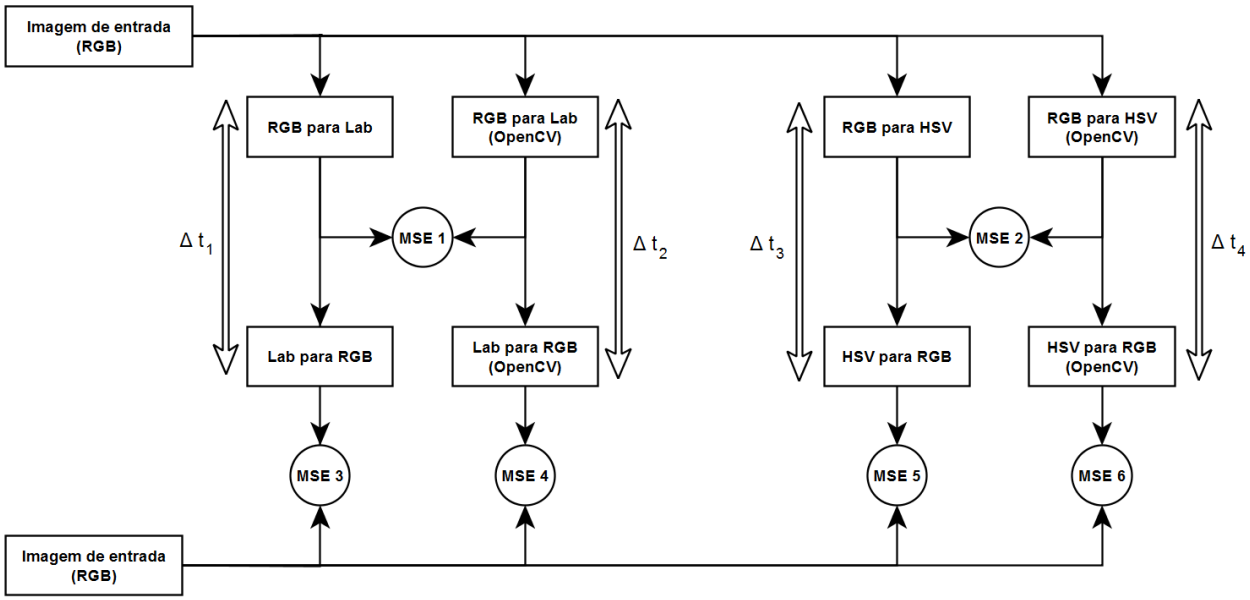
3.2 ALGORITMOS DE CONVERSÃO ENTRE ESPAÇOS DE COR

Utilizando as fórmulas e procedimentos descritos nas Seções 2.3.1, 2.3.2 e 2.3.3, as funções para conversão do espaço de cores RGB para os espaços HSV e CIE Lab foram desenvolvidas. Além disso, as funções para conversão no sentido inverso, isto é, dos sistemas de cor CIE Lab e HSV para o RGB, também foram implementadas.

Para a verificação dos resultados obtidos empregando-se as funções implementadas, foi utilizada a função de conversão entre espaços de cores da biblioteca do OpenCV. O sistema

ilustrado em diagrama de blocos pela Figura 8 representa o método de verificação adotado. A imagem utilizada como entrada do sistema é a imagem de teste escolhida, e que é ilustrada pela Figura 7 na Seção 3.1.

Figura 8 – Representação em diagrama de blocos do arranjo utilizado na verificação de funções de conversão entre espações de cores



Os erros quadráticos médios MSE 1 e MSE 2 determinam o quão distante os resultados obtidos através das funções implementadas estão daqueles obtidos utilizando a biblioteca do OpenCV. E os demais erros quadráticos médios apontam o grau de distorção introduzido no processo de transformação entre espaço de cores. Por fim, o tempo consumido para efetuar a transformação RGB – HSV/CIE Lab – RGB foi medido para fins de comparação de velocidade entre as funções implementadas e as do OpenCV. Foram efetuadas quatro medidas de tempo, e a média das medidas foi tomada como estimativa do tempo consumido.

3.3 INICIALIZAÇÃO DE *CODEBOOK*

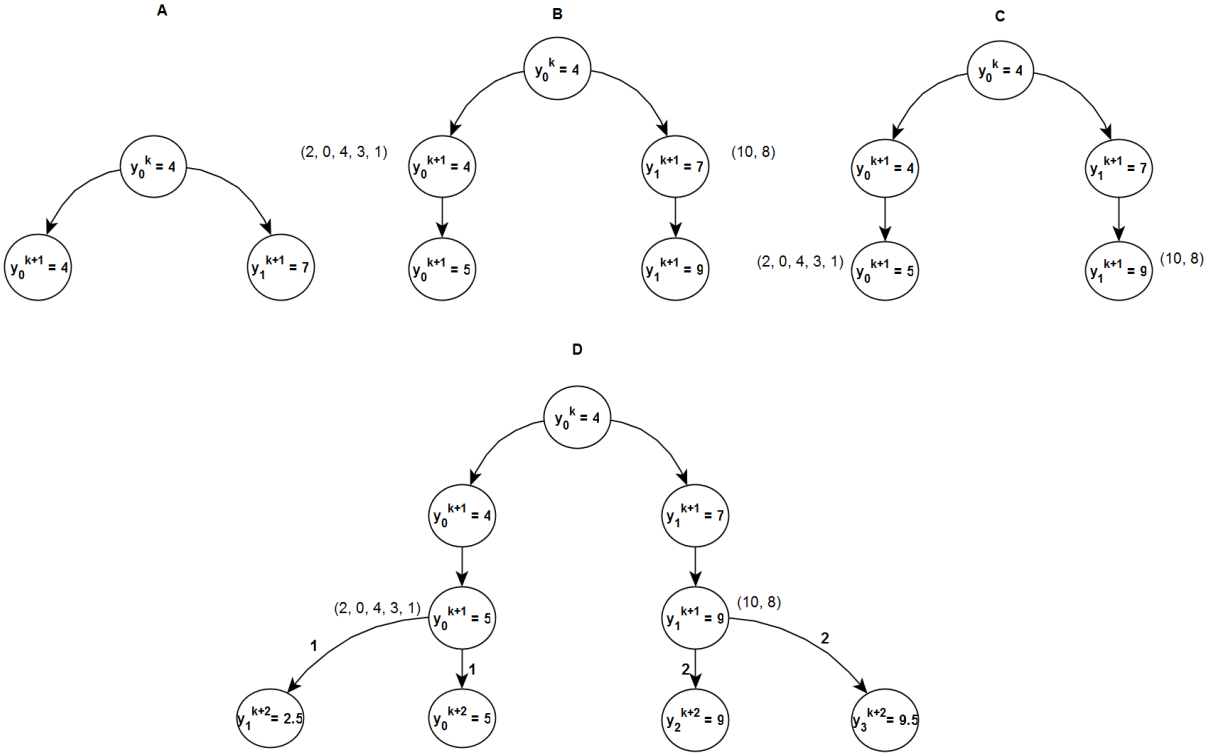
A inicialização de *codebook* por amostragem foi implementada com um simples laço for cujo avanço Δs é dado pela Expressão (2.2.1). O índice do laço varre o vetor de treino, enquanto um outro índice de avanço unitário preenche as posições do *codebook*.

O método de inicialização por *splitting* foi implementado utilizando uma estrutura de árvore binária, com o objetivo de reduzir o tempo de busca no passo do algoritmo de inicialização em que se aplica o LBG (Seção 2.2). Além disso, para reduzir ainda mais o tempo

consumido pela inicialização por *splitting*, e dessa forma não impactar em grande proporção o tempo total do processo de geração de *codebook*, fixou-se o número máximo de iterações, K_{max}, do algoritmo de LBG em 1.

Na estrutura de árvore binária utilizada, os nós do tipo folha correspondem aos *codewords*. As folhas são divididas seguindo o critério de redução do máximo erro quadrático, dado pela Expressão (2.2.4). E a ordem em que as folhas são selecionadas para divisão é determinada pelo número de vetores associados a cada uma, isto é, as folhas com maior número de vetores associados sofrem a divisão primeiro. Por fim, a Figura 9 ilustra um exemplo de divisão de uma folha, ou *codeword*.

Figura 9 – Exemplo de divisão dos *codewords* utilizando a estrutura de árvore binária



No exemplo ilustrado pela Figura 9, utilizou-se o conjunto de vetores de treino unidimensionais dispostos na Tabela 4. Em **A**, o centroide inicial do conjunto se divide em dois novos *codewords*, assim como descrito no Passo 2 do Algoritmo de inicialização por *splitting*, na Seção 2.2. Em **B**, os vetores de treino são designados a cada folha seguindo o critério de distorção adotado na Seção 2.2 (erro quadrático) e, então, o centroide de cada folha é recalculado. Os *codewords* atualizados herdam o conjunto de vetores de treino, em **C**, e uma

nova divisão é realizada em **D**. Observe que os índices presentes ao lado das transições, que indicam a direção de divisão, representam a ordem na qual as divisões ocorrem.

Tabela 4 – Vetores de treino utilizados no exemplo da Figura 9

Vetores de treino						
10	2	0	4	8	3	1

3.4 ALGORITMO LBG

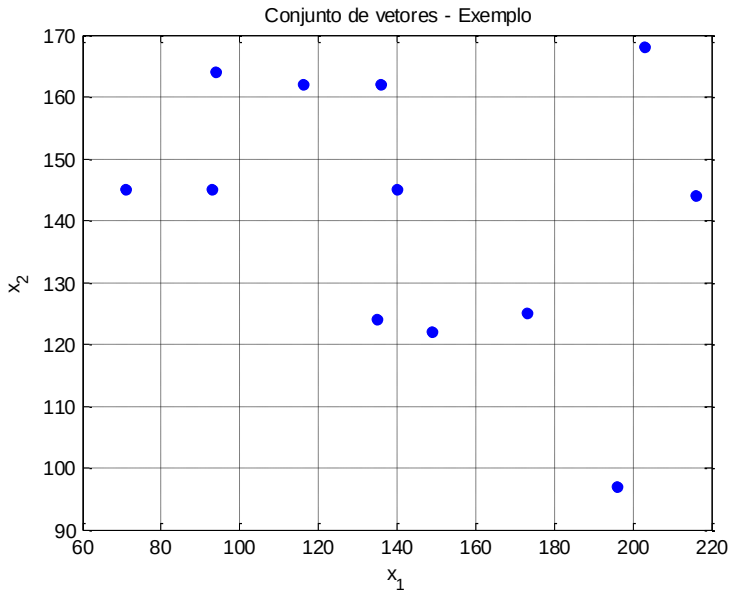
Similar à implementação do algoritmo de inicialização de *codebook* por amostragem, o algoritmo LBG foi desenvolvido utilizando-se apenas dois laços aninhados. O laço externo percorre todos os vetores de treino, enquanto, para cada um deles, o laço interno varre o alfabeto de reprodução em busca do *codeword* que melhor o representa. Durante esse processo, que corresponde ao Passo 1 do algoritmo LBG descrito na Seção 2.2, o erro quadrático é acumulado para posteriormente ser utilizado na verificação de convergência do algoritmo. Por fim, caso a convergência não se confirme, todo o *codebook* é atualizado.

Embora pouco complexa, a implementação do algoritmo é o núcleo de todo o trabalho e, por tanto, deve ser verificada minuciosamente de modo a garantir resultados corretos. Para tanto, um exemplo de proporções reduzidas foi criado com o objetivo de confrontar os resultados obtidos através de cálculo manual com aqueles obtidos por meio do algoritmo implementado. A Figura 10 ilustra o conjunto de vetores, que também está disposto na Tabela 5, utilizado no exemplo.

Tabela 5 – Conjunto de vetores de treino utilizado no exemplo a ser desenvolvido manualmente e através do algoritmo LBG implementado

	Conjunto de vetores - Exemplo											
	x [1]	**x** [2]	**x** [3]	**x** [4]	**x** [5]	**x** [6]	**x** [7]	**x** [8]	**x** [9]	**x** [10]	**x** [11]	**x** [12]
x_1	140	173	71	149	135	94	116	136	216	196	93	203
x_2	145	125	145	122	124	164	162	162	144	97	145	168

Figura 10 – Conjunto de vetores de treino utilizado no exemplo a ser desenvolvido manualmente e através do algoritmo LBG implementado



O conjunto de vetores bidimensionais foi criado utilizando-se a função randn do *software* MATLAB® R2012b. O intervalo de valores foi escolhido com base nos valores típicos de uma imagem com profundidade de *pixel* igual a 24 bits, isto é, com 1 byte para cada cor.

Para o exemplo, o tamanho do *codebook* é 4, a tolerância utilizada no critério de convergência é de 0,1% e o número máximo de iterações, K_{max}, não foi fixado por se tratar de um exemplo de proporções reduzidas. Além disso, o algoritmo será verificado com dois alfabetos iniciais: um obtido pelo método de amostragem e outro pelo método de *splitting*. Por fim, tem-se que os parâmetros tamanho do conjunto de vetores de treino, dimensão dos vetores, tamanho de *codebook* e tolerância, foram escolhidos com base no exemplo feito por LINDE em (LINDE, 1980).

Os passos a serem executados para verificação dos resultados estão dispostos logo abaixo.

Inicialização por amostragem:

1. Manualmente, calcula-se o passo Δs através da Expressão (2.2.1).
2. Manualmente, seleciona-se os vetores de treino do conjunto de vetores de entrada, com avanço Δs.
3. Manualmente, aplica-se o algoritmo de LBG definido na Seção 2.2.
4. Compara-se os valores obtidos manualmente com aqueles obtidos através do algoritmo implementado.

Inicialização por splitting:

1. Manualmente, aplica-se o algoritmo de inicialização por *splitting* definido na Seção 2.2.
2. Manualmente, aplica-se o algoritmo de LBG definido na Seção 2.2.
3. Compara-se os valores obtidos manualmente com aqueles obtidos através do algoritmo implementado.

3.5 QUANTIZAÇÃO VETORIAL NOS DIFERENTES ESPAÇOS DE CORES

Uma vez verificada a funcionalidade dos algoritmos desenvolvidos, resta avaliar a eficácia da compressão da imagem de teste escolhida nos diferentes espaços de cores. Os parâmetros escolhidos para a avaliação da eficácia foram: razão de compressão, velocidade de compressão e distorção da imagem final. Dentre as métricas de cálculo da distorção apresentadas na Seção 2.4, foi utilizado apenas o PSNR. Por fim, foram efetuadas quatro medidas de tempo, e a média das medidas foi tomada como estimativa do tempo consumido.

As dimensões dos vetores foram escolhidas de modo a evidenciar a sensibilidade de cada canal à quantização vetorial, nos diferentes espaços de cores explorados. Os valores das dimensões dos vetores, e das dimensões, largura, **w**, e altura, **h**, dos blocos que formam os vetores, estão dispostas nas Tabela 6 e 7, respectivamente.

Tabela 6 – Dimensões de cada canal utilizadas nos experimentos de compressão, nos diferentes espaços de cores

	Canal		
	R/L/H	**G/a/S**	**B/b/V**
Dimensão 1	2	4	4
Dimensão 2	4	2	4
Dimensão 3	4	4	2
Dimensão 4	4	16	16
Dimensão 5	16	4	16
Dimensão 6	16	16	4
Dimensão 7	16	16	16

Tabela 7 – Dimensões dos blocos que formam os vetores

Dimensão	**w**	**h**
2	2	1
4	2	2
16	4	4

Com o objetivo de avaliar o efeito da utilização de apenas uma fração do conjunto de vetores que compõem a imagem de entrada, cada experimento de compressão foi repetido utilizando-se tamanhos diferentes de subconjunto de vetores de treino. Desse modo, será possível avaliar o compromisso entre a redução do número de vetores de treino, e, por conseguinte, do tempo consumido pelo algoritmo LBG, e o aumento da distorção da imagem comprimida. Por fim, a seleção do subconjunto, nesse trabalho, foi realizada pelo processo de amostragem, assim como é feita a seleção do *codebook* inicial pelo método de amostragem. As razões, $\mathbf{L}_{\%}$, entre os tamanhos do subconjunto escolhido e do conjunto total de vetores de treino estão dispostas na Tabela 8.

Tabela 8 – Tamanhos de subconjuntos de vetores de treino, relativos ao tamanho do conjunto total de vetores que compõem a imagem, utilizados nos experimentos de compressão de imagem

$\mathbf{L}_{\%}$	1	5	10	50	100

Por fim, cada experimento foi executado utilizando-se os dois modos de inicialização do alfabeto de reprodução: por amostragem e por *splitting*. Logo, tem-se que os fatores controláveis são: tamanho do conjunto de vetores de treino, dimensões do quantizador vetorial, método de inicialização do *codebook* e espaço de cor.

As Figuras 11 e 12 ilustram, em diagrama de blocos, os sistemas desenvolvidos para realizar as compressões nos espaços de cor RGB, HSV e CIE Lab. Nas figuras, a determinação da dimensão dos vetores é feita no bloco intitulado “Decomposição em vetores” e o parâmetro $\mathbf{L}_{\%}$ é utilizado no bloco “Seleção dos vetores de treino”. Além disso, assim como é evidenciado pelas Figuras 11 e 12, os blocos “Encoding” e “Decoding” constituem, junto ao armazenamento em arquivo local, o quantizador vetorial, apresentado na Seção 2.1.

O critério utilizado para a análise do efeito da redução do tamanho do conjunto de vetores de treino foi o cálculo da derivada do PSNR em relação ao tempo consumido, Δt, dado pela Expressão (3.5.1). O mesmo cálculo foi realizado para todas as combinações dos fatores controláveis, com exceção de $L_{\%}$, e a média dos resultados foi tomada para análise.

$$\frac{dPSNR}{d\Delta t} = \frac{PSNR_{L_{\%}[i]} - PSNR_{L_{\%}[i-1]}}{\Delta t_{L_{\%}[i]} - \Delta t_{L_{\%}[i-1]}} \quad (3.5.1)$$

onde *i* refere-se ao i-ésimo tamanho do conjunto de vetores de treino utilizado.

A Tabela 9 apresenta outros parâmetros utilizados nos experimentos de compressão, apresentados nesta seção, tais como tolerância de convergência e número máximo de iterações do algoritmo LBG.

Tabela 9 – Parâmetros do algoritmo LBG utilizados nos experimentos de compressão da imagem de teste

K_{max}	Tolerância (ε)
100	0.1%

Figura 11 – Diagrama de blocos do sistema desenvolvido para compressão da imagem de teste no espaço de cor RGB

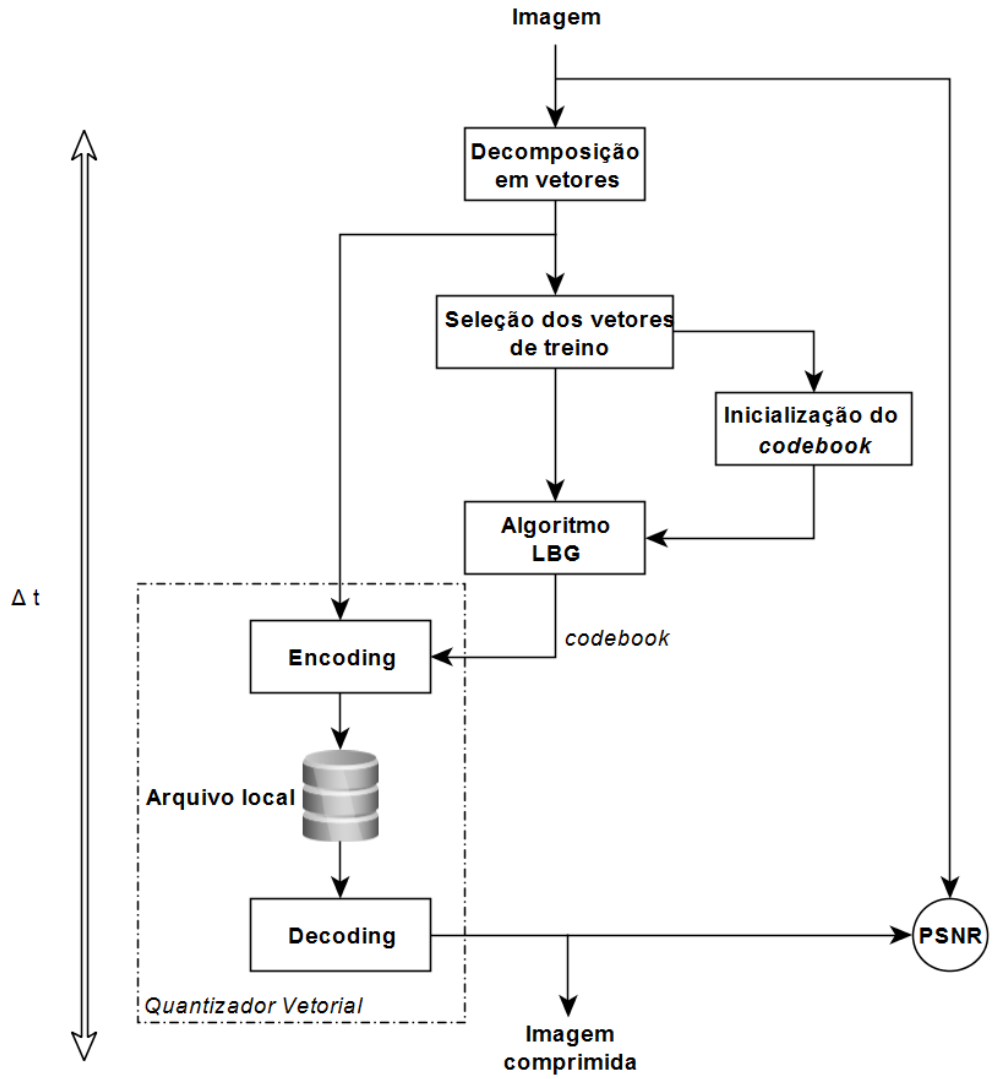
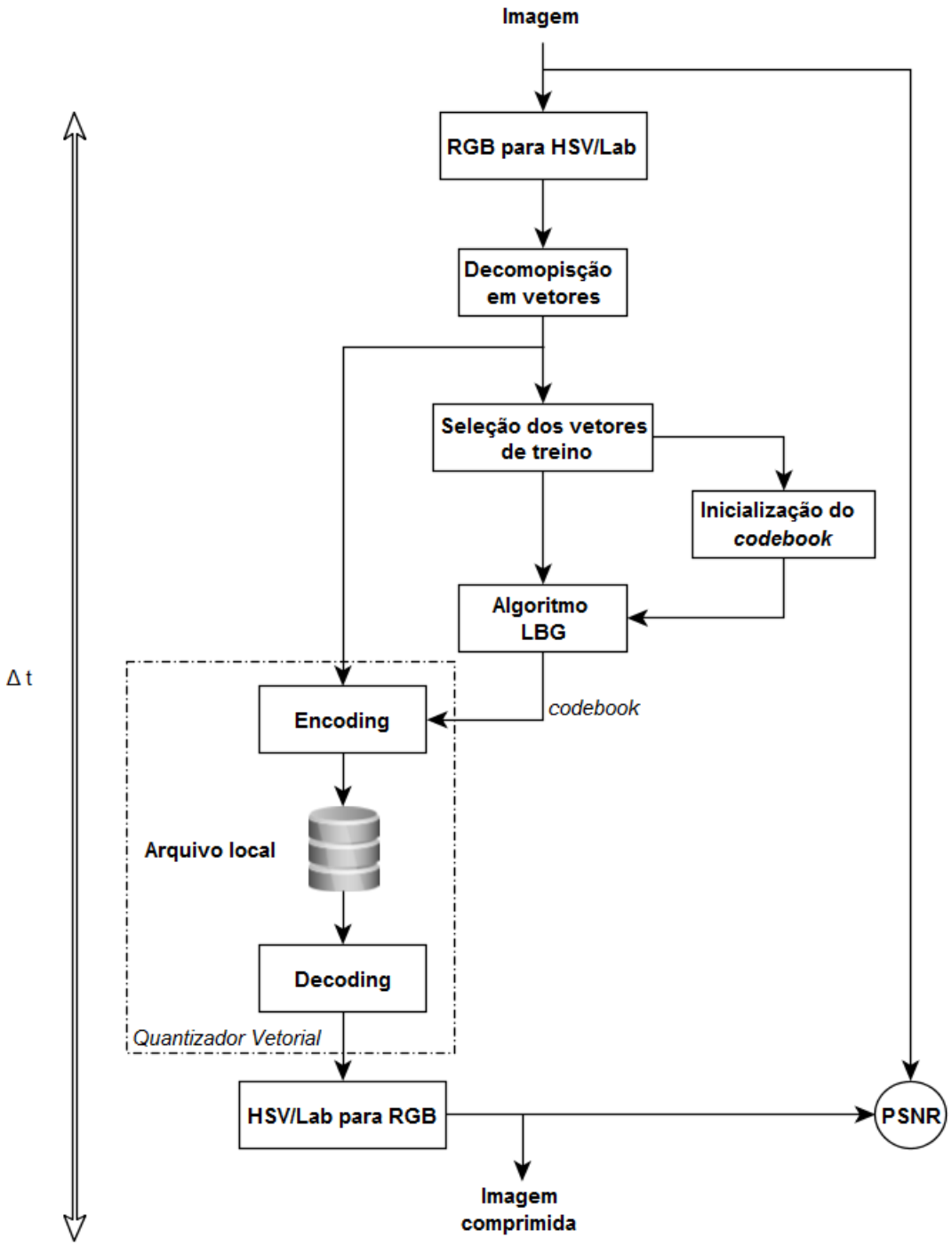


Figura 12 – Diagrama de blocos dos sistemas desenvolvidos para compressão da imagem de teste nos espaços de cor HSV e CIE Lab



3.6 COMPARAÇÃO DE PERFORMANCE

Com o objetivo de obter uma referência de performance do algoritmo de compressão implementado, uma comparação de resultados é efetuada com base nos resultados obtidos aplicando-se o tradicional método de compressão JPG. O algoritmo JPG utilizado foi o disponibilizado pela biblioteca do OpenCV.

As imagens utilizadas para a comparação são ilustradas pela Figuras 13 e 14, onde a Figura 13 é um recorte do mapa utilizado no experimento descrito na Seção 3.5. Nas tabelas 11 e 12 estão dispostas as propriedades das imagens utilizadas.

Inicialmente, as imagens foram comprimidas pelo algoritmo de LBG utilizando as dimensões de vetores 1, 4 e 7, dispostas na Tabela 6, e os demais parâmetros dispostos na Tabela 10. Então, o algoritmo JPG foi aplicado utilizando fatores de qualidade, *G*, que resultam em valores de PSNR próximos daqueles obtidos pelo algoritmo LBG. Por fim, o algoritmo JPG foi novamente aplicado nas imagens, porém agora utilizando fatores de qualidade que resultam em níveis de compressão próximos daqueles obtidos pelo algoritmo LBG.

Tabela 10 – Parâmetros do algoritmo LBG: espaço de cor utilizado, método de inicialização de codebook, número máximo de iterações (K_{max}), tolerância de convergência (ε) e tamanho relativo do conjunto de treino ($L_{\%}$)

Espaço de Cor	**Inicialização de *codebook***	**K_{max}**	**ε**	**$L_{\%}$**
Lab	*Splitting*	50	1%	10%

Tabela 11 – Propriedades da figura “Litoral Norte do RS”

Dimensões [largura x altura]	**Extensão**	**Profundidade de *pixel***	**Tamanho**
1024 x 696	.bmp	24 bits	2142 KBytes

Tabela 12 – Propriedades da figura “Sul de Madagascar”

Dimensões [largura x altura]	**Extensão**	**Profundidade de *pixel***	**Tamanho**
1024 x 680	.bmp	24 bits	2089 KBytes

Figura 13 – Litoral Norte do RS

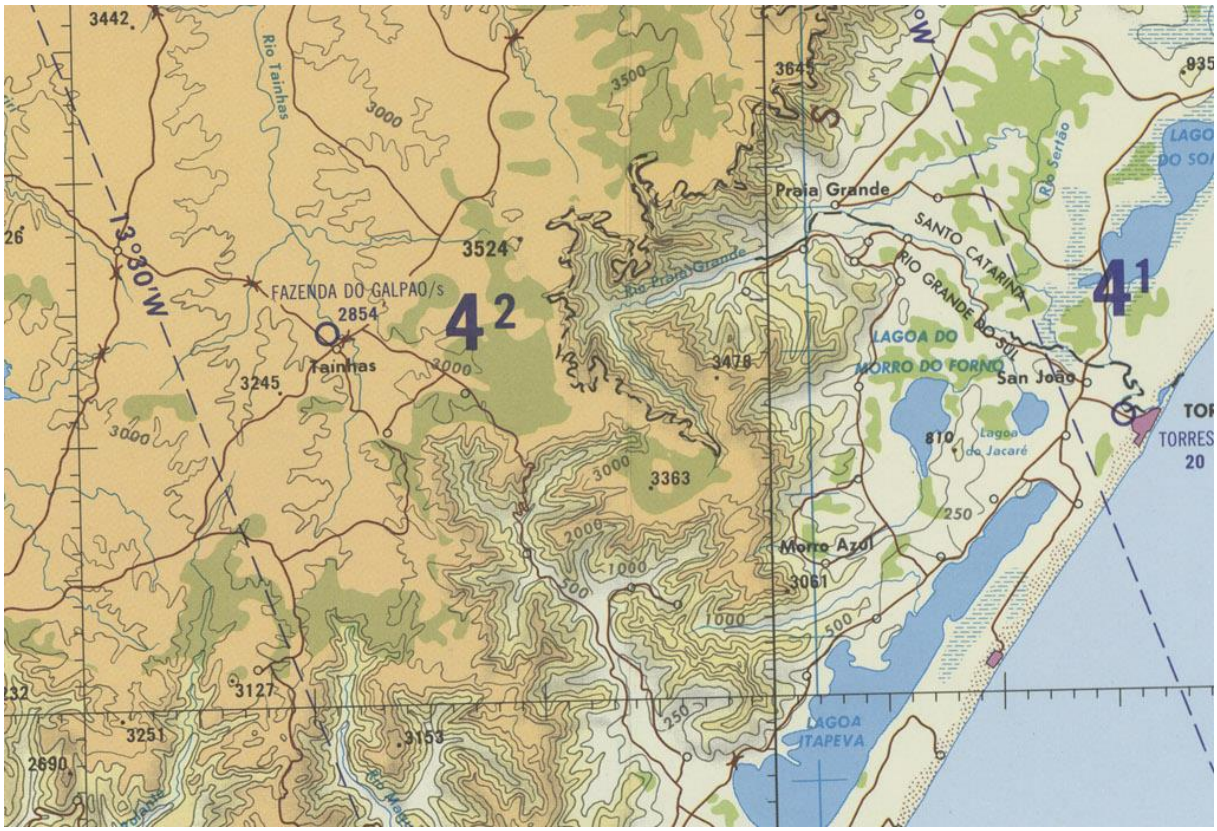


Figura 14 – Sul de Madagascar

Os resultados a serem comparados são a razão de compressão e PSNR, dados valores fixos, e aproximados entre os dois algoritmos, de PSNR e razão de compressão, respectivamente. Além disso, será analisada a velocidade de compressão dos dois algoritmos.

4 RESULTADOS E DISCUSSÕES

Nessa seção, são apresentados os resultados e breves comentários dos experimentos descritos nas seções anteriores.

4.1 VERIFICAÇÃO DAS TRANSFORMAÇÕES ENTRE ESPAÇOS DE COR

Seguindo o procedimento descrito na Seção 3.2, foram obtidos os erros médios quadráticos e tempos consumidos. Os valores dos erros médios quadráticos estão dispostos na Tabela 13.

Tabela 13 – Erros médios quadráticos

MSE 1	**MSE 2**	**MSE 3**	**MSE 4**	**MSE 5**	**MSE 6**
0,55	0,13	1,61	0,45	0,15	0,15

A partir dos dados apresentados na Tabela 13, é possível observar que o maior erro quadrático médio entre os resultados de conversão entre os espaços de cor, obtidos através dos algoritmos desenvolvidos e das funções do OpenCV, foi de 0,55 [.], que corresponde a conversão para o espaço de cor CIE Lab. Além disso, a maior distorção introduzida por uma conversão completa foi de 1,61 [.], que corresponde a conversão RGB-CIE Lab-RGB, através da função implementada.

A partir dos valores dos tempos médios dispostos na Tabela 14, tem-se que a conversão entre os espaços de cor RGB-Lab é 16,31 vezes mais rápida utilizando-se a função do OpenCV, e entre os espaços de cor RGB-HSV, 6,45 vezes mais rápida.

Tabela 14 – Tempos médios consumidos

Δt_1 [s]	**Δt_2 [s]**	**Δt_3 [s]**	**Δt_4 [s]**
1,044	0,064	0,284	0,044

4.2 VERIFICAÇÃO DOS MÉTODOS DE INICIALIZAÇÃO E DO ALGORITMO LBG

A partir do exemplo proposto na Seção 3.4, e dos algoritmos e expressões descritas na Seção 2.2, a seguir são apresentados os passos e resultados dos métodos de inicialização de *codebook* e do algoritmo de LBG.

Inicialização por amostragem:

Segundo a Expressão (2.2.1), tem-se que o passo para escolha dos *codewords* iniciais é:

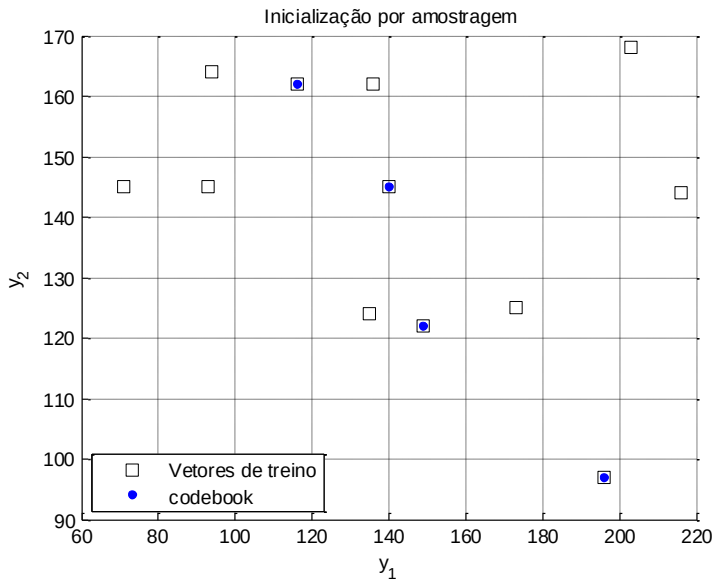
$$\Delta s = \left\lfloor \frac{12}{4} \right\rfloor = 3$$

Logo, o *codebook* inicial é aquele disposto na Tabela 15, e ilustrado na Figura 15 junto aos vetores de treino.

Tabela 15 – *Codebook* inicial obtido pelo método de amostragem

	Codebook inicial - Amostragem			
y_1	140	149	116	196
y_2	145	122	162	97

Figura 15 – Disposição do *codebook* inicial, obtido pelo método de amostragem, junto aos vetores de treino



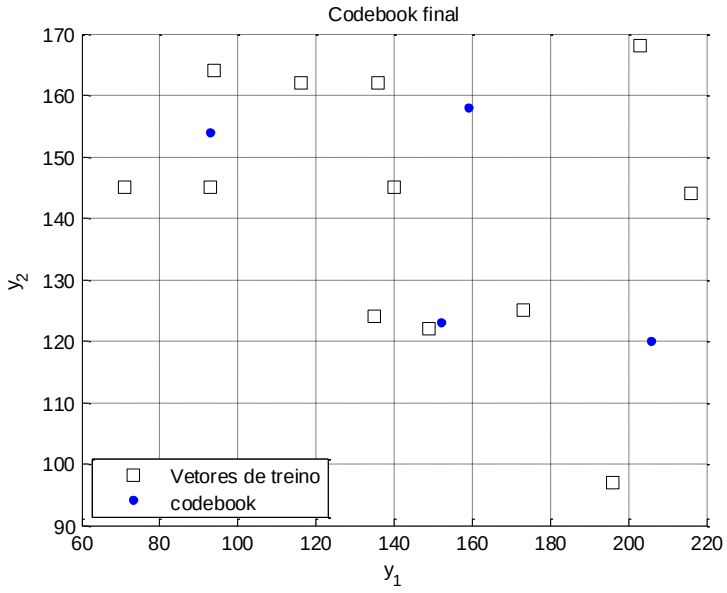
O resultado da inicialização do *codeboook* é então utilizado no algoritmo de LBG. A Tabela 16 apresenta os resultados de cada iteração do algoritmo de LBG desenvolvido

manualmente. O *codebook* final, isto é, da última iteração, está ilustrado na Figura 16 junto aos vetores de treino.

Tabela 16 – Parâmetro de convergência (ε), calculados a partir do erro quadrático total (TSE – Total Squared Error, em inglês), codebook e erro médio quadrático (MSE) de cada iteração do algoritmo LBG, aplicado sobre o codebook inicial obtido pelo método de amostragem

Iteração	**1**		**2**		**3**	
Codebook	y_1	y_2	y_1	y_2	y_1	y_2
	140	145	159	158	159	158
	149	122	152	123	152	123
	116	162	93	154	93	154
	196	97	206	120	206	120
TSE	11817		6501		6501	
ε	-		0.450		0	
MSE	492.4		270.9		270.9	

Figura 16 – Disposição do *codebook* final, com inicialização por amostragem, junto aos vetores de treino



Por fim, na Tabela 17 estão dispostos os *codebooks* inicial e final obtidos utilizando os algoritmos de inicialização e LBG implementados.

Tabela 17 – Codebooks inical e final obtidos, respectivamente, através dos algoritmos de inicialização por amostragem e LBG implementados

Codebook			
Inicial		**Final**	
y_1	y_2	y_1	y_2
140	145	159	158
149	122	152	123
116	162	93	154
196	97	206	120

Inicialização por splitting:

Seguindo o algoritmo de inicialização por *splitting* apresentado na Seção 2.2, tem-se que o centroide de todo o conjunto de vetores de treino, através da Expressão (2.2.2), é:

$$C^{(0)} = \left\{ \left\lfloor \frac{1}{12} \cdot \sum_{i=1}^{12} \mathbf{x_i} \right\rfloor \right\} = \left\{ \begin{bmatrix} 143 \\ 141 \end{bmatrix} \right\}$$

Através da Expressão (2.2.4), tem-se que a perturbação **d** é:

$$\mathbf{d} = \left\lfloor \frac{\mathbf{x}_{max} - y_i}{2} \right\rfloor = \left\lfloor \frac{\begin{bmatrix} 216 \\ 144 \end{bmatrix} - \begin{bmatrix} 143 \\ 141 \end{bmatrix}}{2} \right\rfloor = \begin{bmatrix} 36 \\ 1 \end{bmatrix}$$

Em seguida, através da Expressão (2.2.3), tem-se que os dois novos *codewords* são:

$$C^{(1)} = \left\{ \begin{bmatrix} 143 \\ 141 \end{bmatrix}, \begin{bmatrix} 179 \\ 142 \end{bmatrix} \right\}$$

Seguindo o Passo 4 do algoritmo de inicialização por *splitting*, o *codebook* resultante após a aplicação de uma iteração do algoritmo LBG é:

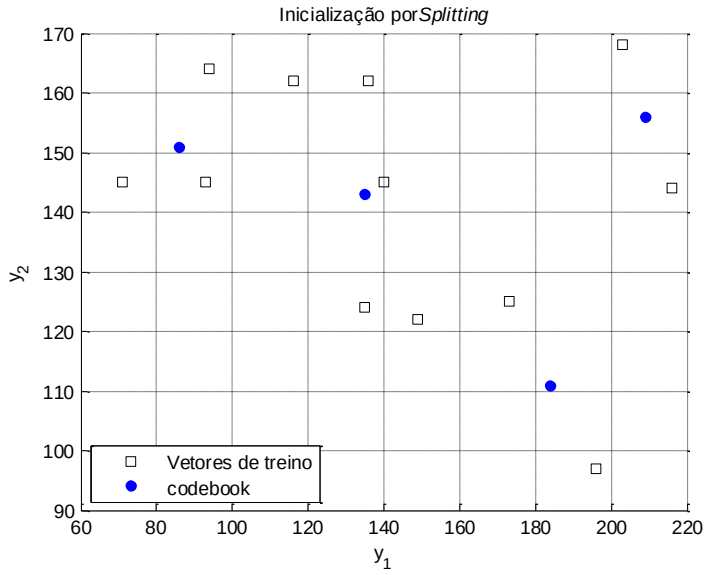
$$C^{(1)} = \left\{ \begin{bmatrix} 116 \\ 146 \end{bmatrix}, \begin{bmatrix} 197 \\ 133 \end{bmatrix} \right\}$$

Novas perturbações são calculadas, e o processo de divisão, seguido de uma iteração do algoritmo de LBG, é repetido uma vez mais. Nesse ponto, tem-se que o número de nós do tipo folha é igual ao tamanho desejado do *codebook* e, por tanto, o algoritmo termina. O alfabeto de reprodução gerado pelo método *splitting* está disposto na Tabela 18, e ilustrado na Figura 17 junto aos vetores de treino.

Tabela 18 – *Codebook* inicial obtido pelo método de *splitting*

	Codebook inicial - *Splitting*			
y_1	135	86	209	184
y_2	143	151	156	111

Figura 17 – Disposição do *codebook* inicial, obtido pelo método de *splitting*, junto aos vetores de treino

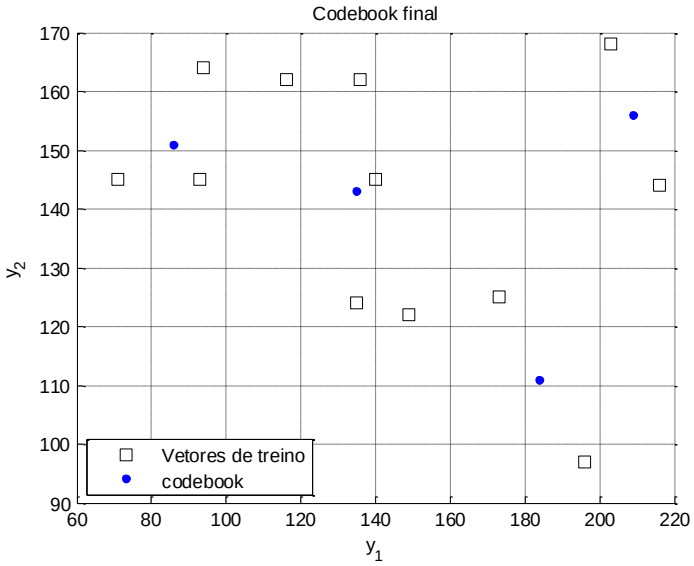


O resultado da inicialização do *codeboook* é então utilizado no algoritmo de LBG. A Tabela 19 apresenta os resultados de cada iteração do algoritmo de LBG desenvolvido manualmente. O *codebook* final, isto é, da última iteração, está ilustrado na Figura 18 junto aos vetores de treino.

Tabela 19 – Parâmetro de convergência (ε), calculados a partir do erro quadrático total (TSE – Total Squared Error, em inglês), codebook e erro médio quadrático de cada iteração do algoritmo LBG, aplicado sobre o codebook inicial obtido pelo método de splitting

Iteração	**1**		**2**	
Codebook	y_1	y_2	y_1	y_2
	135	143	135	143
	86	151	86	151
	209	156	209	156
	184	111	184	111
TSE	3720		3720	
ε	-		0.0	
MSE	155,0		155,0	

Figura 18 – Disposição do *codebook* final, com inicialização por *splitting*, junto aos vetores de treino



Por fim, na Tabela 20 estão dispostos os *codebooks* inicial e final obtidos utilizando os algoritmos de inicialização e LBG implementados.

Tabela 20 – Codebooks inical e final obtidos, respectivamente, através dos algoritmos de inicialização por splitting e LBG implementados

Codebook			
Inicial		**Final**	
y_1	y_2	y_1	y_2
135	143	135	143
86	151	86	151
209	156	209	156
184	111	184	111

Por fim, nesse exemplo, o erro médio quadrático final obtido através do método de inicialização por *splitting* é 1,75 vezes menor que aquele obtido através do método de inicialização por amostragem.

4.3 RESULTADOS DAS QUANTIZAÇÕES VETORIAIS

As razões de compressão, calculadas através da Expressão (2.1.7) para cada uma das dimensões na Tabela 6, estão dispostas na Tabela 21 junto aos tamanhos dos arquivos locais resultantes do algoritmo de compressão.

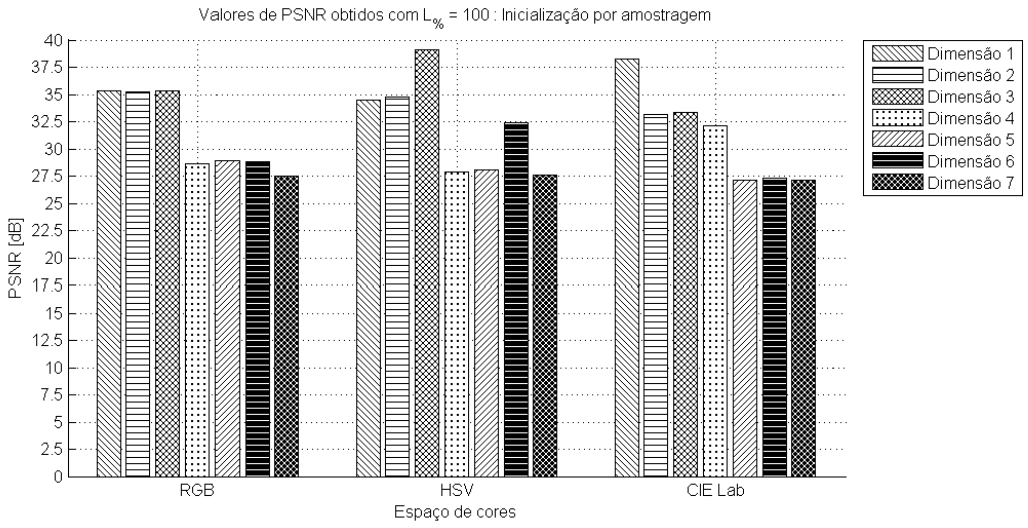
Tabela 21 – Razão de compressão (RC) e tamanho do arquivo local, em Kbytes, para cada uma das dimensões exploradas

	RC	**Tamanho (Kb)**
Dimensão 1	3,00	3561
Dimensão 2	3,00	3561
Dimensão 3	3,00	3561
Dimensão 4	7,95	1344
Dimensão 5	7,95	1344
Dimensão 6	7,95	1344
Dimensão 7	15,72	680

Os 210 valores de PSNR e médias de tempos consumidos, que cobrem todas as combinações dos fatores controláveis do experimento descrito na Seção 3.5, estão dispostos nas Tabelas A.1, A.2 e A.3, no ANEXO A.

Na Figura 19, se encontra ilustrado o gráfico de barras que apresenta os valores de PSNR para as diferentes dimensões de vetores e diferentes espaços de cor. Para os resultados da Figura 19, o método de inicialização é o de amostragem e, além disso, todo o conjunto de vetores que formam a imagem é utilizado no processo de treinamento do *codebook* (i.e., $\mathbf{L_{\%}} = 100$).

Figura 19 – Valores de PSNR obtidos com $L_{\%} = 100$ e inicialização de *codebook* por amostragem, para as diferentes dimensões e espaços de cor



Os mesmos resultados, apresentados na Figura 19 para o método de inicialização por amostragem, são apresentados na Figura 20 para o método de inicialização por *splitting*. Por fim, a diferença entre os resultados dos dois métodos é ilustrada pela Figura 21.

Figura 20 – Valores de PSNR obtidos com $L_{\%}$ = 100 e inicialização de *codebook* por *splitting*, para as diferentes dimensões e espaços de cor

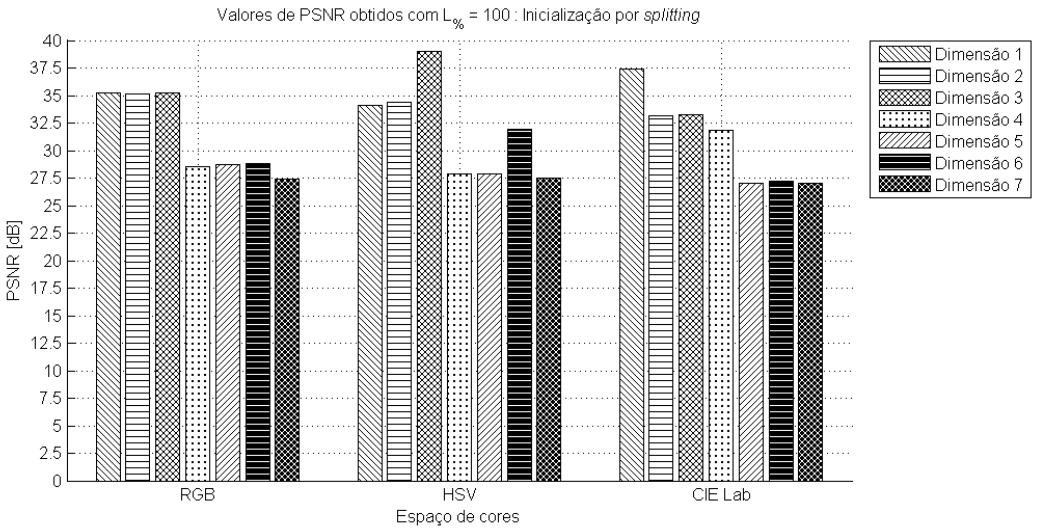
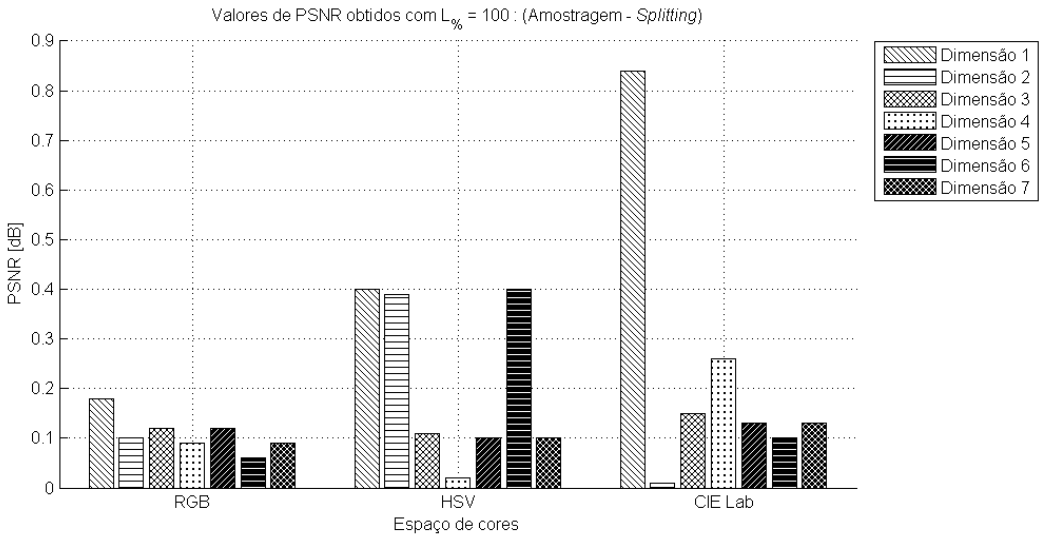


Figura 21 – Diferença entre os valores de PSNR obtidos com os métodos de inicialização de *codebook* por amostragem e por *splitting*, ambos utilizando $L_{\%}$ = 100, para as diferentes dimensões e espaços de cor



A partir dos resultados apresentados pela Figura 21, observa-se que a diferença máxima e mínima de PSNR entre os dois métodos foi de 0,84 dB e 0,01 dB, respectivamente. Além disso, com base nos resultados ilustrados pelas Figuras 19 e 20, observa-se que, para os espaços de cor HSV e Lab, os canais V e L são mais sensíveis à quantização vetorial, quando comparados aos demais canais dos seus respectivos espaços de cor.

As Figuras 22, 23 e 24 ilustram as imagens comprimidas utilizando-se as configurações que obtiveram os melhores resultados, 28,9 dB, 32,4 dB e 32,1 dB, a uma razão de compressão de 7,95, para os espaços de cor RGB, HSV e CIE Lab, respectivamente. Por fim, a Figura 25

ilustra o melhor resultado, 27,6 dB, obtido para a dimensão com maior razão de compressão, Dimensão 7.

Figura 22 – Ilustração do resultado de compressão obtido no espaço de cor RGB utilizando o método de inicialização por amostragem e a Dimensão 5, com $L_{\%} = 100$

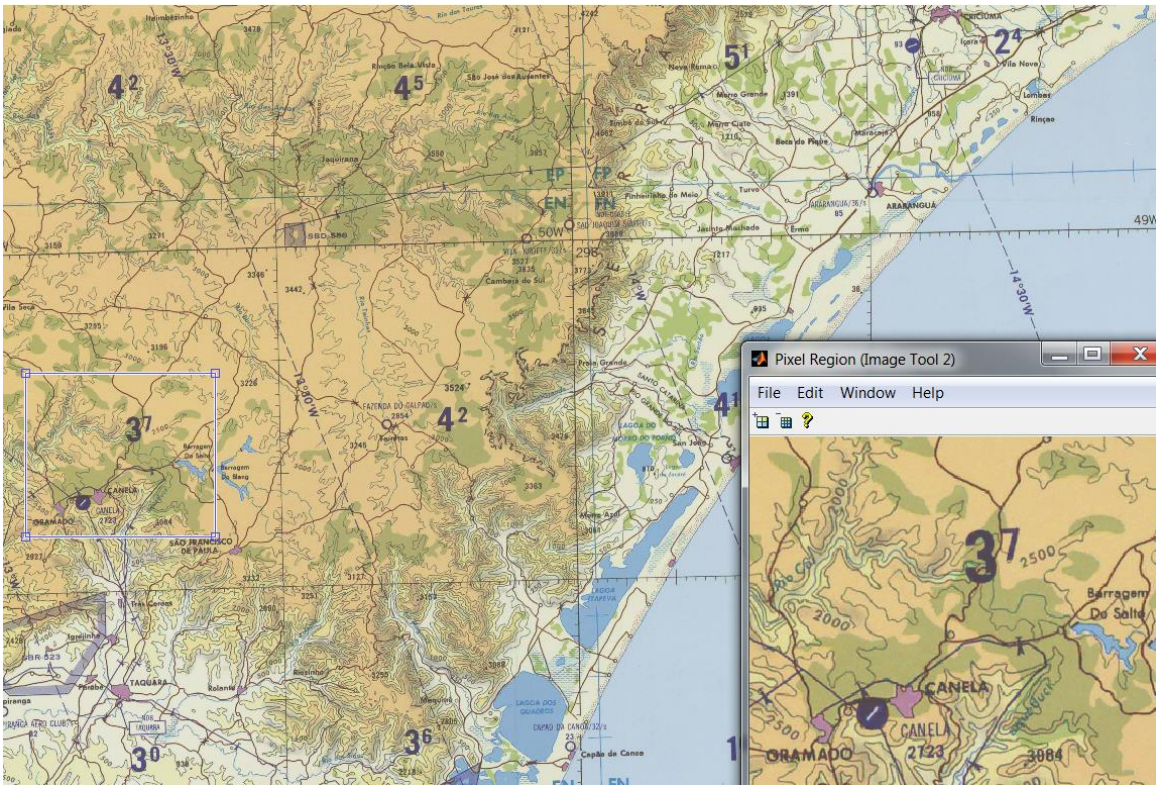


Figura 23 – Ilustração do resultado de compressão obtido no espaço de cor HSV utilizando o método de inicialização por amostragem e a Dimensão 6, com $L_{\%} = 100$

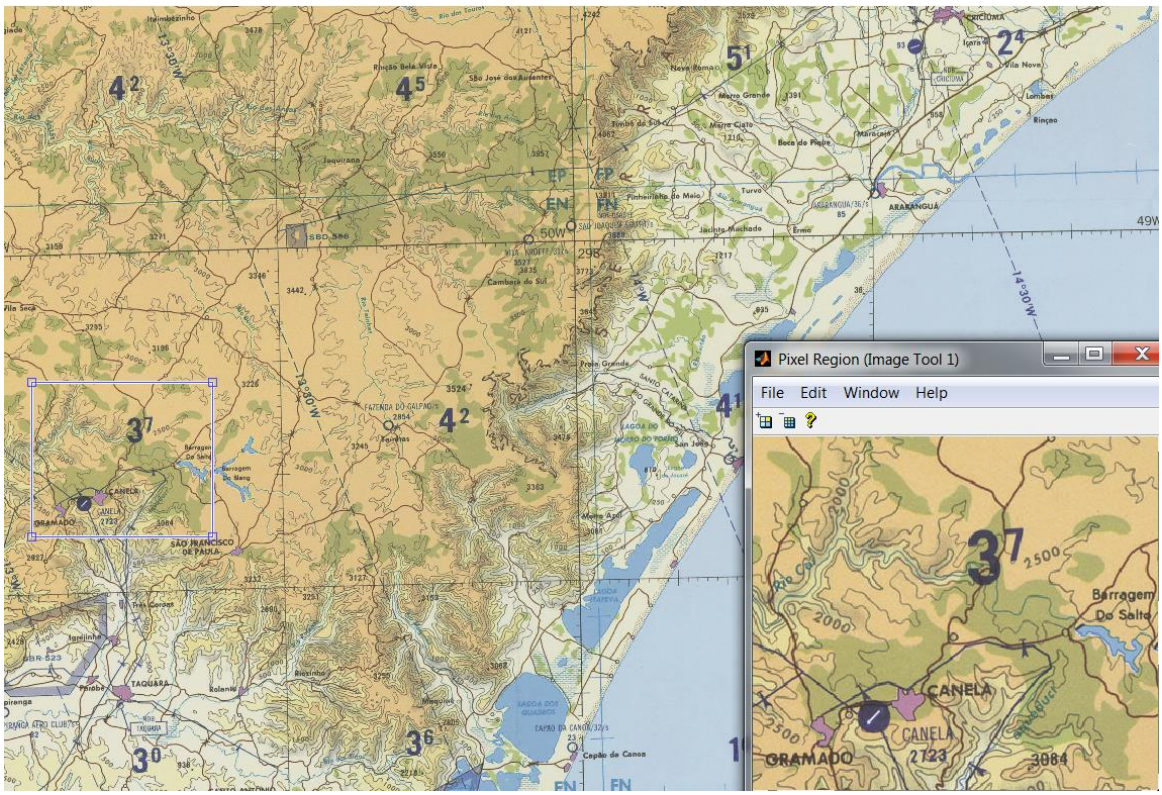


Figura 24 – Ilustração do resultado de compressão obtido no espaço de cor CIE Lab utilizando o método de inicialização por amostragem e a Dimensão 4, com $L_{\%} = 100$

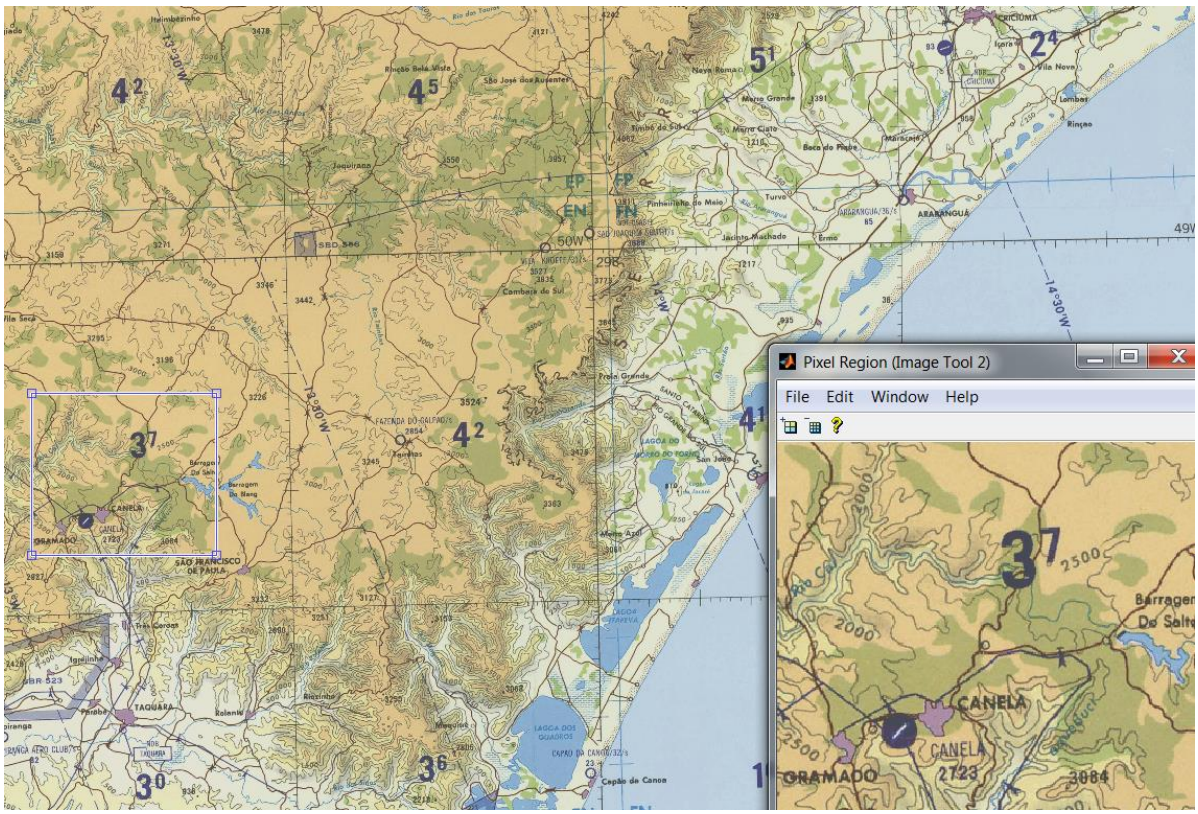


Figura 25 - Ilustração do resultado de compressão obtido no espaço de cor HSV utilizando o método de inicialização por amostragem e a Dimensão 7, com $L_{\%} = 100$

Nas Figuras 26 e 27, estão dispostos os tempos médios consumidos, com $L_{\%} = 100$, para os métodos de inicialização por amostragem e *splitting*, respectivamente. A partir dos resultados, tem-se que o tempo de convergência do algoritmo LBG utilizando-se inicialização por *splitting* é até 2,87 vezes menor quando comparado ao método de inicialização por amostragem,

Figura 26 – Valores de tempo médio consumido com $L_{\%} = 100$ e inicialização de codebook por amostragem, para as diferentes dimensões e espaços de cor

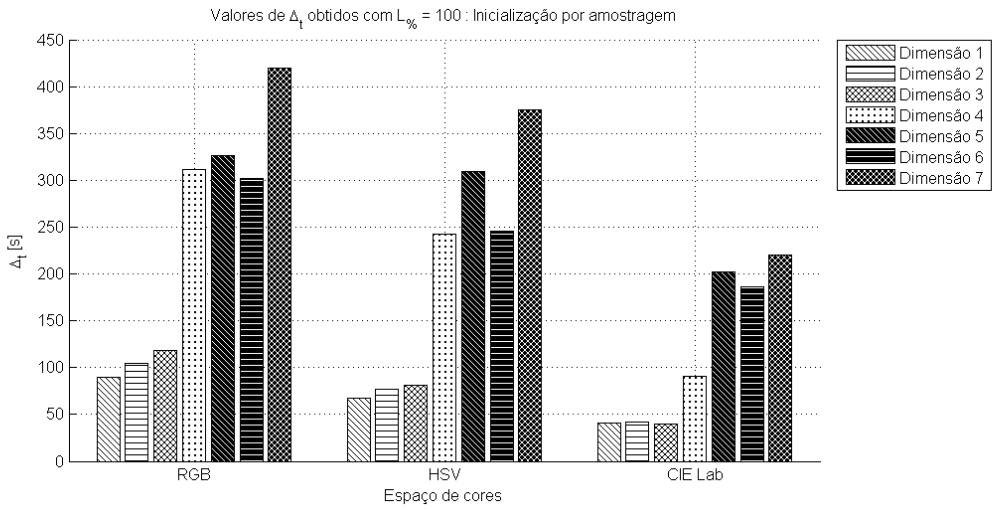
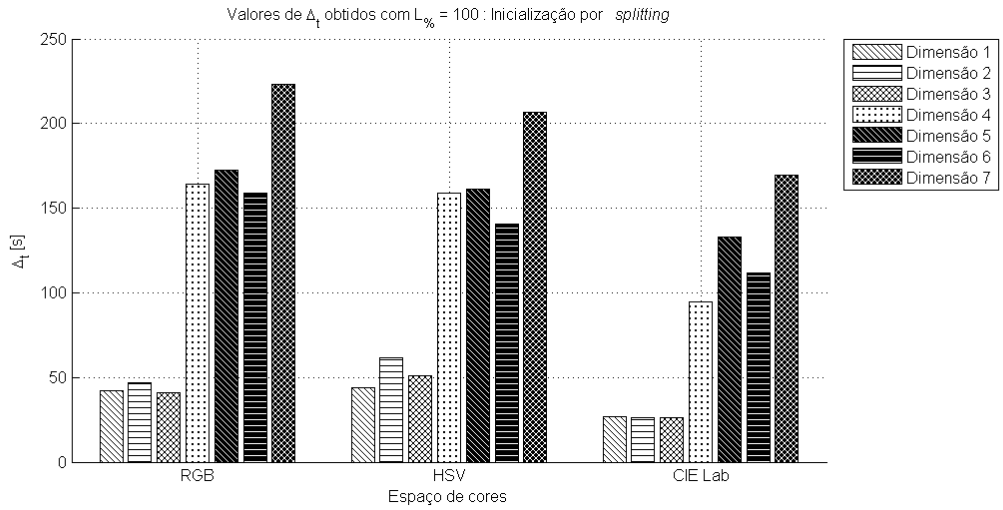
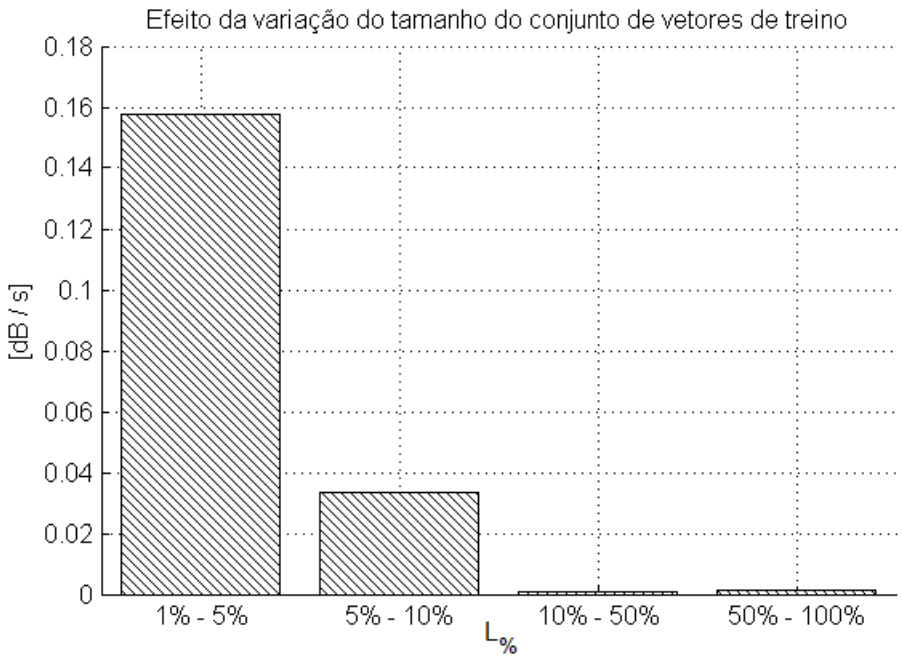


Figura 27 – Valores de tempo médio consumido com $L_{\%}$ = 100 e inicialização de codebook por splitting, para as diferentes dimensões e espaços de cor



O resultado da análise sobre o efeito da redução do conjunto de vetores de treino está disposto na Figura 28.

Figura 28 – Efeito da variação do tamanho do conjunto de vetores de treino utilizado no refinamento do codebook inicial



Para fins de comparação de resultados, a Figura 29 ilustra o mesmo resultado apresentado pela Figura 23, porém utilizando-se apenas 10% do conjunto total de vetores de treino, isto é, $L_{\%}$ = 10. Além disso, as Figuras 30 e 31 apresentam as médias dos tempos consumidos, ao se utilizar $L_{\%}$ = 10, assim como é apresentado pelas Figuras 26 e 27 para $L_{\%}$ =

100. Por fim, com base nesses resultados tem-se que o tempo de convergência do algoritmo LBG é até 11,8 vezes menor ao se utilizar 10% do conjunto total de vetores de treino, quando comparado com os resultados das Figuras 26 e 27.

Figura 29 – Ilustração do resultado de compressão obtido no espaço de cor HSV utilizando o método de inicialização por amostragem e a Dimensão 6, com $L_{\%} = 10$

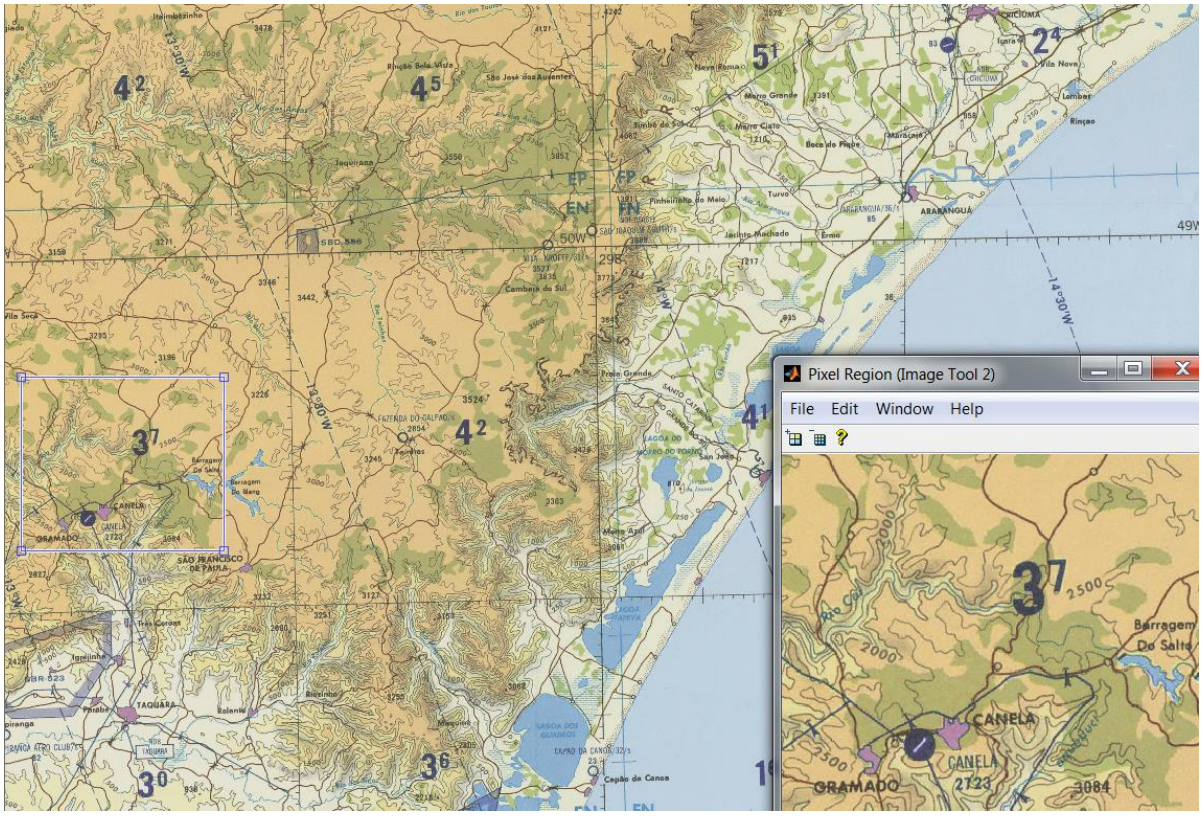


Figura 30 – Valores de tempo médio consumido com $L_{\%} = 10$ e inicialização de *codebook* por amostragem, para as diferentes dimensões e espaços de cor

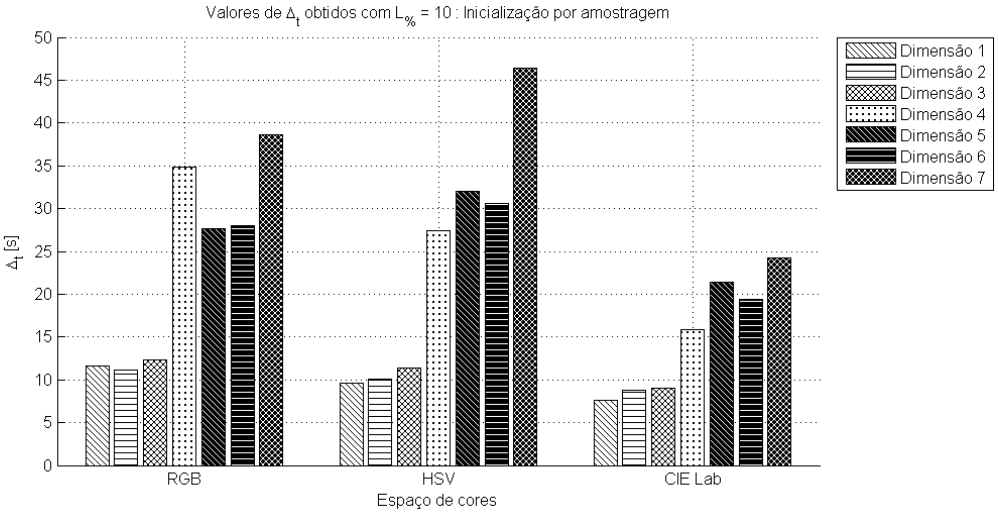
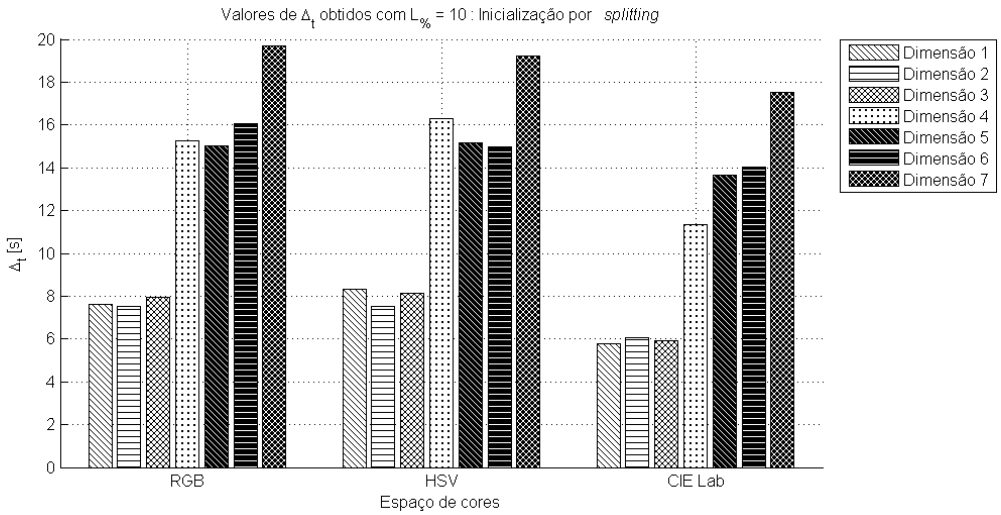


Figura 31 – Valores de tempo médio consumido com $L_{\%}$ = 10 e inicialização de *codebook* por *splitting*, para as diferentes dimensões e espaços de cor



4.4 RESULTADOS DA COMPARAÇÃO DE PERFORMANCE

Os resultados do experimento, no qual, utilizando o JPG, buscou-se atingir valores de PSNR próximos daqueles obtidos utilizando o algoritmo LBG, estão dispostos nas Tabelas 22 e 23 para as figuras “Litoral Norte do RS” e “Sul de Madagascar”, respectivamente.

Tabela 22 – Resultados de performance para a figura “Litoral Norte do RS”, fixando-se o PSNR e comparando os níveis de compressão

	Dimensão	**1**	**4**	**7**
LBG	**PSNR [dB]**	37,66	31,07	25,61
	Tempo [s]	23,06	12,08	10,22
	Tamanho [Kbytes]	715,28	270,49	145,94
	RC	2,99	7,92	14,68
JPG	**PSNR [dB]**	37,24	31,08	25,66
	Tempo [s]	0,23	0,23	0,21
	Tamanho [Kbytes]	412,94	186,06	64,13
	RC	5,19	11,51	33,40
	***G* [%]**	96	78	20

Tabela 23 – Resultados de performance para a figura “Sul de Madagascar”, fixando-se o PSNR e comparando os níveis de compressão

	Dimensão	**1**	**4**	**7**
LBG	**PSNR [dB]**	36,87	31,47	27,36
	Tempo [s]	23,02	14,1	11,87
	Tamanho [Kbytes]	698,89	270,35	142,87
	RC	3,07	7,92	14,99
JPG	**PSNR [dB]**	36,75	36,75	27,51
	Tempo [s]	0.325	0,273	0,252
	Tamanho [Kbytes]	373,13	120,71	35,75
	RC	5,74	17,75	59,92
	***G* [%]**	96	76	18

Com base nos resultados das Tabelas 22 e 23, tem-se que, para um mesmo valor de PSNR, o algoritmo JPG obteve razões de compressão até 2,2 vezes maior que o obtido através do algoritmo LBG, para a figura “Litoral Norte do RS”, e até 4 vezes maior, para a figura “Sul de Madagascar”. As Figuras 32, 33, 34 e 35 ilustram os resultados obtidos para as duas figuras, “Litoral Norte do RS” e “Sul de Madagascar”, para uma das dimensões (Dimensão 4) utilizada do algoritmo LBG.

Figura 32 – Ilustração do resultado da compressão através do algoritmo LBG da figura “Litoral Norte do RS”, fixando-se o PSNR e comparando os níveis de compressão

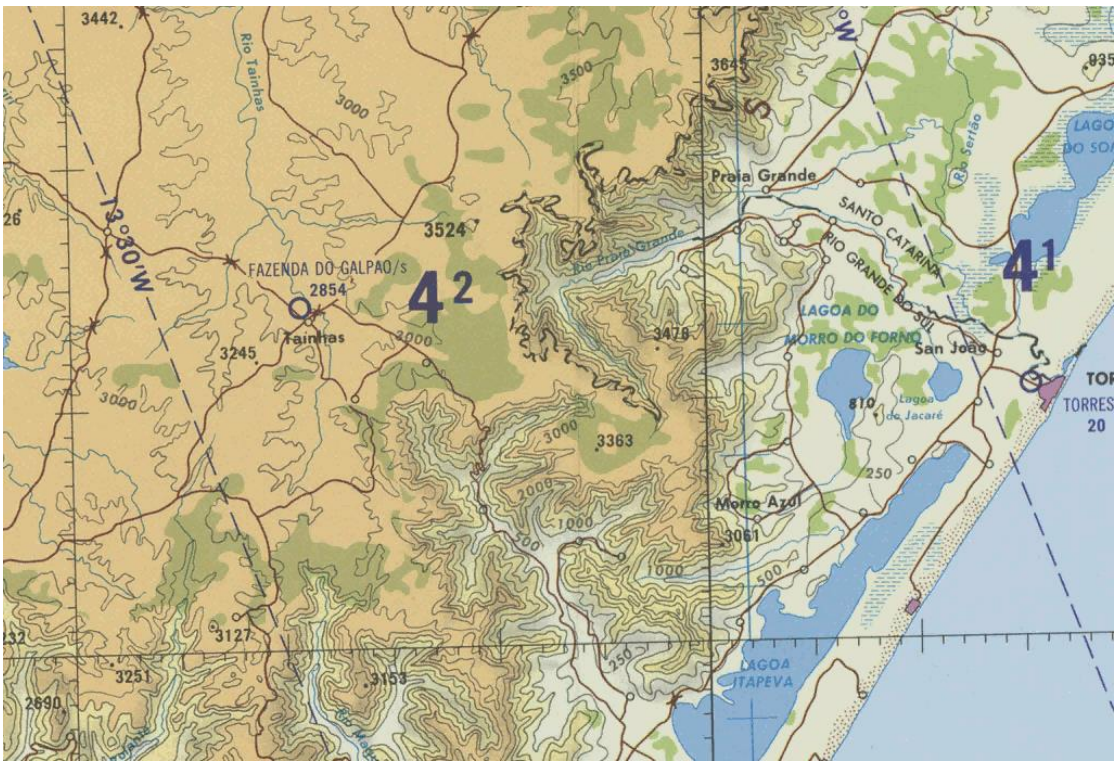


Figura 33 – Ilustração do resultado da compressão através do algoritmo JPG da figura “Litoral Norte do RS”, fixando-se o PSNR e comparando os níveis de compressão

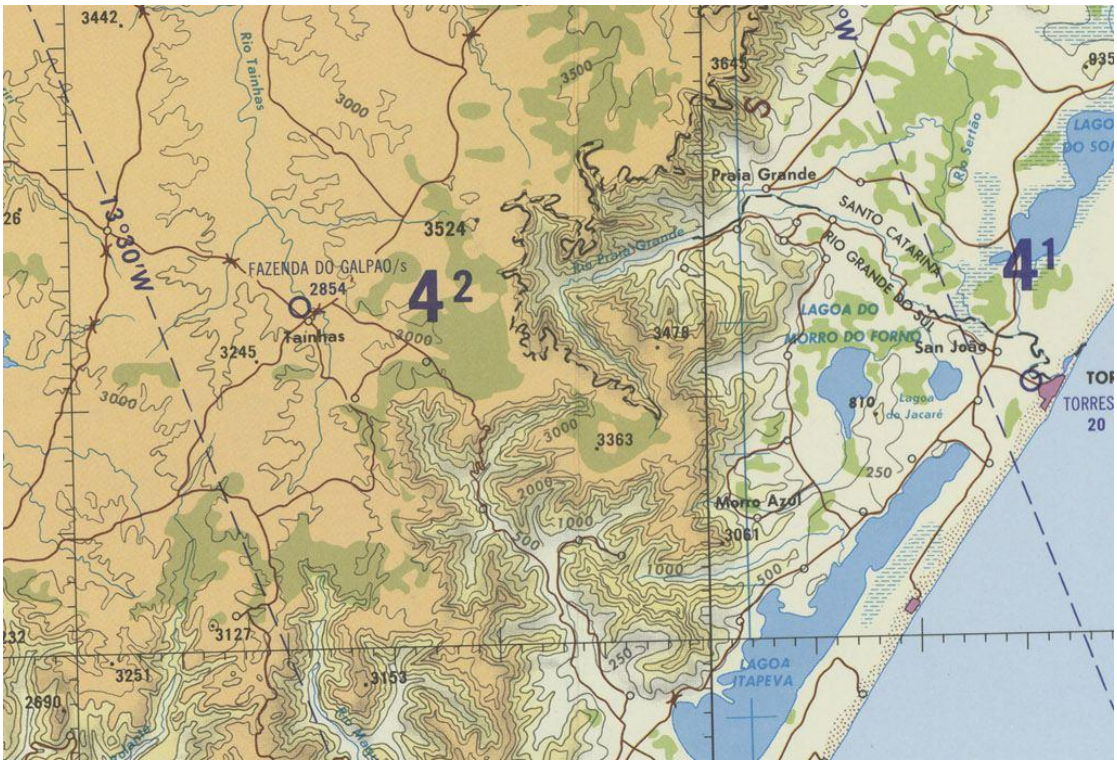


Figura 34 – Ilustração do resultado da compressão através do algoritmo LBG da figura “Sul de Madagascar”, fixando-se o PSNR e comparando os níveis de compressão

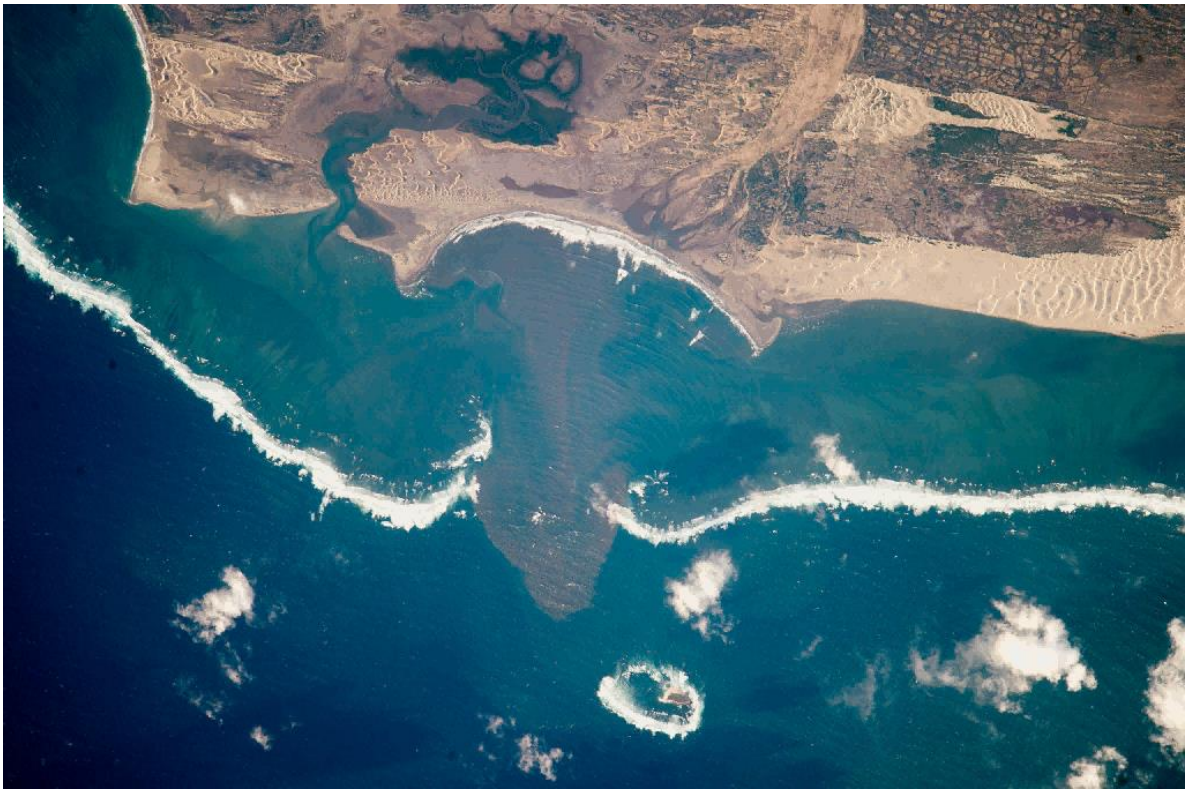


Figura 35 – Ilustração do resultado da compressão através do algoritmo JPG da figura “Sul de Madagascar”, fixando-se o PSNR e comparando os níveis de compressão

Os resultados do experimento, no qual, utilizando o JPG, buscou-se atingir valores de razão de compressão próximos daqueles obtidos utilizando o algoritmo LBG, estão dispostos nas Tabelas 24 e 25 para as figuras “Litoral Norte do RS” e “Sul de Madagascar”, respectivamente.

Tabela 24 – Resultados de performance para a figura “Litoral Norte do RS”, fixando-se os níveis de compressão e comparando os valores de PSNR

	Dimensão	**1**	**4**	**7**
LBG	**PSNR [dB]**	37,66	31,07	25,61
	Tempo [s]	23,06	12,08	10,22
	Tamanho [Kbytes]	715,28	270,49	145,94
	RC	2,99	7,92	14,68
JPG	**PSNR [dB]**	39,60	33,96	29,38
	Tempo [s]	0,53	0,29	0,31
	Tamanho [Kbytes]	644,96	273,47	64,13
	RC	3,32	7,83	14,66
	***G* [%]**	100	90	66

Tabela 25 – Resultados de performance para a figura "*Sul de Madagascar*", fixando-se os níveis de compressão e comparando os valores de PSNR

	Dimensão	**1**	**4**	**7**
LBG	**PSNR [dB]**	36,87	31,47	27,36
	Tempo [s]	23,02	14,1	11,87
	Tamanho [Kbytes]	698,89	270,35	142,87
	RC	3,07	7,92	14,99
JPG	**PSNR [dB]**	39,60	35,25	32,04
	Tempo [s]	0,24	0,25	0,21
	Tamanho [Kbytes]	666,53	267,42	143,32
	RC	3,13	7,81	14,57
	***G* [%]**	100	93	81

Com base nos resultados das Tabelas 24 e 25, tem-se que, para uma dada razão de compressão, o algoritmo JPG obteve valores de PSNR até 3,7 dB acima daqueles obtidos através do algoritmo LBG, para a figura "Litoral Norte do RS", e até 4,6 dB acima, para a figura "Sul de Madagascar". As Figuras 36 e 37, 38 e 39 ilustram os resultados obtidos para as duas figuras, "Litoral Norte do RS" e "Sul de Madagascar", para uma das dimensões (Dimensão 4) utilizada do algoritmo LBG. Além disso, pelos tempos consumidos, e dispostos nas Tabelas 22, 23, 24 e 25, tem-se que a velocidade do algoritmo JPG é, entre os resultados dos experimentos executados nessa Seção, até 100 vezes maior que a do algoritmo desenvolvido.

Figura 36 – Ilustração do resultado da compressão através do algoritmo LBG da figura “Litoral Norte do RS”, fixando-se a razão de compressão e comparando os valores de PSNR

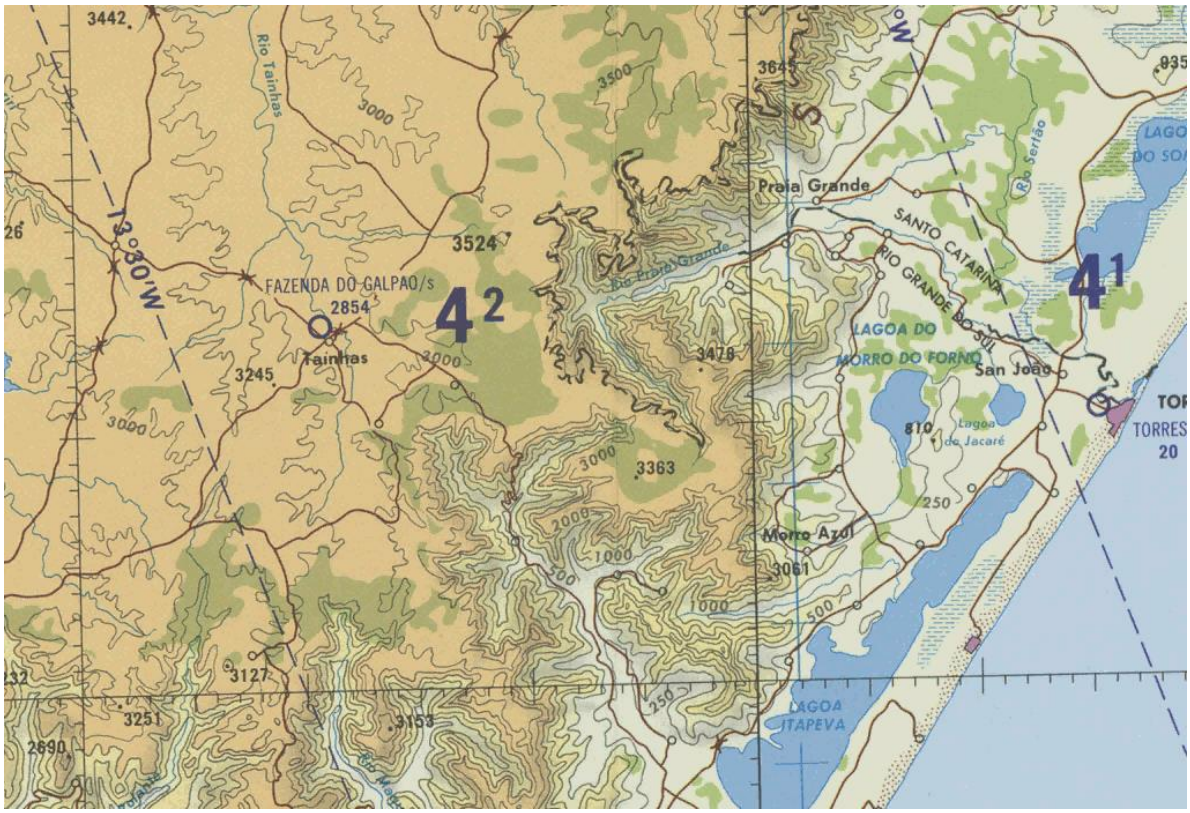


Figura 37 - Ilustração do resultado da compressão através do algoritmo JPG da figura “Litoral Norte do RS”, fixando-se a razão de compressão e comparando os valores de PSNR

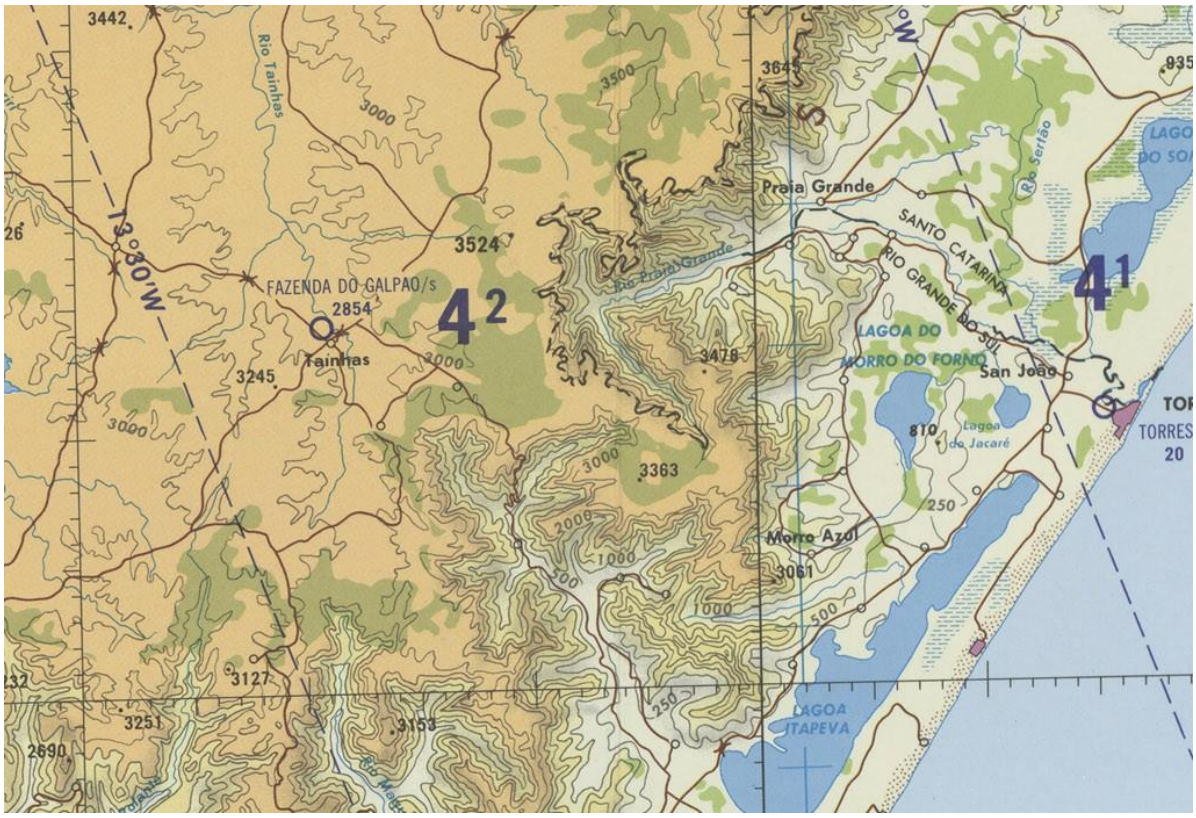


Figura 38 – Ilustração do resultado da compressão através do algoritmo LBG da figura “Sul de Madagascar”, fixando-se a razão de compressão e comparando os valores de PSNR

Figura 39 – Ilustração do resultado da compressão através do algoritmo JPG da figura “Sul de Madagascar”, fixando-se a razão de compressão e comparando os valores de PSNR

5 CONCLUSÃO

A partir dos resultados obtidos para os experimentos de verificação de funcionalidade, é possível concluir que as funções implementadas para conversão entre os espaços de cor produzem resultados satisfatórios, considerando os baixos valores de erro médio quadrático obtidos em relação aos resultados das funções do OpenCV. Além disso, os códigos implementados em C/C++ para o método LBG e de inicialização de *codebook* por amostragem e *splitting* produzem resultados verossímeis, assim como demostrado pelo *benchmarking* manual.

Os resultados da compressão da imagem de teste, mostram que, de fato, o *codebook* inicial influi no resultado final da compressão, e que o método de inicialização por *splitting* propicia maior velocidade de convergência no algoritmo LBG. Os melhores resultados correspondem aos casos em que menores taxas de compressão são aplicadas aos canais de luminância, nos espaços de cor HSV e CIE Lab. Além disso, constata-se que a partir de certo ponto não há vantagem em aumentar o conjunto de vetores de treino, pois o ganho de PSNR torna-se insignificante frente ao aumento do tempo consumido para a convergência do algoritmo LBG. Tal resultado irá variar entre imagens de teste, uma vez que o acréscimo de distorção, associado à redução do conjunto de vetores de treino, está atrelado à redundância presente na imagem, isto é, quanto maior a redundância espacial de informação, menor será o acréscimo de distorção.

Os resultados dos experimentos de comparação de performance entre o algoritmo implementado e o algoritmo JPG demostram que o último é superior em todos os aspectos, qualidade e velocidade. Apesar disso, o desenvolvimento desse trabalho permitiu a verificação do funcionamento, vantagens e limitações do algoritmo LBG para quantização vetorial, que, invariavelmente, será a técnica utilizada em qualquer aplicação do padrão CADRG.

Por fim, as sugestões de modificações e melhorias para trabalhos futuros partindo do mesmo algoritmo são: implementação de um algoritmo de busca de *codeword* ótimo mais rápido e transformação dos *codewords* de cada canal de volta para o espaço de cor RGB, de modo a eliminar o tempo de conversão no lado da descompressão. Para reduzir o tempo do algoritmo de busca do *codeword* ótimo, é possível utilizar a estrutura *k-d tree* proposta por Bentley em (BENTLEY, 1975), ou mesmo classificar os vetores de treino em função da distância a origem do espaço vetorial e efetuar a busca sobre aqueles *codewords* que residem sobre a mesma classificação.

REFERÊNCIAS

[1] RABBANI, Majid; JONES, Paul W. **Digital Image Compression Techniques**. Vol. TT7. SPIE, 1995.

[2] PEIXOTO, Virgilio Marcos Tort. **Pesquisa, Escolha e Implementação de um Algoritmo de Compressão de Imagens para CADRG em C++**. 2015. 77f. Tese (Bacharelado em Ciência da Computação) – Pontifícia Universidade Católica do Rio Grande do Sul Faculdade de Informática, Porto Alegre, 2015.

[3] GERSHO, Allen; WANG, Shihua; ZEGER, Kenneth. Vector Quantization Techniques in Speech Coding. In: FURUI, S. ; SONDHI, M. **Advances in Speech Signal Processing**. [S.I.]. Marcel Dekker, 1992. cap. 2, p. 49-84.

[4] LINDE, Yoseph; BUZO, Andrés; GRAY, Robert M. An Algorithm for Vector Quantizer Design. **IEEE TRANSACTIONS ON COMMUNICATIONS**, [S.I.], vol. 28, n. 1, p. 84-95, Jan. 1980.

[5] SOUTHARD, David A. Compression of Digitized Map Images. **Computers & Geosciences**, [S.I.], vol. 18, n. 9, p. 1213-1253, Abril 1992.

[6] LLOYD, Stuart P. Least Squares Quantization in PCM. **IEEE TRANSACTIONS OF INFORMATION THEORY**, [S.I.], vol. 28, n. 2, p. 129-137, Mar. 1982.

[7] AKRAMULLAH, Shahriar. Digital Video Compression Techniques. In: AKRAMULLAH, Shahriar. **Digital Video Concepts, Methods, and Metrics – Quality, Compression, Performance, and Power Trade-off Analysis**. [S.I.]. Appress Media, 2014. cap. 2, p.11-54.

[8] EQUITZ, William H. A New Vector Quantization Clustering Algorithm. **IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING**, [S.I.], vol. 37, n. 10, p. 1568-1575, Mar. 1989.

[9] IVANOV, Ivan-Assen. GAMASUTRA. Disponível em: < http://www.gamasutra.com/>. Acesso em: 20 novembro 2015.

[10] CHEN, Chaur-Chin. Professor Chaur-Chin Chen. Disponível em: < http://www.cs.nthu.edu.tw/~cchen/> . Acesso em: 20 novembro 2015.

[11] BURGER, Wilhelm; BURGE, Mark J. Color Images. In: BURGER, Wilhelm; BURGE, Mark J. **Principles of Digital Image Processing – Fundamental Techniques**.[S.I.]. Springer, 2009. cap. 8, p.185-228.

[12] KERR, Douglas A. **The CIE XYZ and xyZ Color Spaces**. [S.I.: s.n.], 2010. 10 p.

[13] LINDBLOOM, Bruce J. **BruceLindbloom.com**. Disponível em: < http://www.brucelindbloom.com/>. Acesso em: 30 agosto 2015.

[14] VELDHUIZEN, Todd Lawrence. **Grid Filters for Local Nonlinear Image Restoration**. 1998. 129 f. Tese (Master of Applied Science in Systems Design Engineering) – University of Waterloo, Waterloo, Ontario, Canadá, 1998.

[15] MANDAL, Mrinal Kr. Digital Image Compression Techniques. In: MANDAL, Mrinal Kr. **Multimedia Signals and Systems**. 1. ed. [S.I.]. Kluwer Academic, 2003. cap. 8, p. 169-202.

[16] BENTLEY, J. L. Multidimensional binary search trees used for associative searching. **Communication ACM**, [S.I.], vol. 18, n. 9, p. 509-517, Setembro 1975.

ANEXO A – RESULTADOS DA COMPRESSÃO DA IMAGEM DE TESTE

Tabela A.1 – Valores de PSNR e tempo médio consumidos para todos as combinações de fatores controláveis no espaço de cor RGB

		Amostragem		**Splitting**	
		PSNR [dB]	**Δt [s]**	**PSNR [dB]**	**Δt [s]**
1	**Dim 1**	34.24	3.52	34.8	3.21
	Dim 2	34.21	3.7	34.69	3.23
	Dim 3	34.27	3.56	34.71	3.24
	Dim 4	27.16	4.39	27.63	4.37
	Dim 5	27.48	4.48	27.77	4.34
	Dim 6	27.47	4.51	27.81	4.51
	Dim 7	26.04	5.19	26.39	5.28
5	**Dim 1**	35.05	7.08	35.06	5.31
	Dim 2	35.03	8.05	35.02	5.61
	Dim 3	35.02	8.15	35.04	5.023
	Dim 4	27.99	15.36	28.15	9.18
	Dim 5	28.26	14.63	28.31	8.4
	Dim 6	28.19	12.96	28.29	9.05
	Dim 7	26.82	18.11	26.93	10.96
10	**Dim 1**	35.31	11.66	35.11	7.62
	Dim 2	35.19	11.19	35.09	7.53
	Dim 3	35.17	12.29	35.13	7.93
	Dim 4	28.29	34.85	28.28	15.28
	Dim 5	28.48	27.67	28.44	15
	Dim 6	28.47	28.05	28.46	16.06
	Dim 7	27.09	38.67	27.08	19.7
50	**Dim 1**	35.35	45.03	35.18	27.36
	Dim 2	35.29	49.87	35.12	25
	Dim 3	35.23	51.44	35.16	27.28
	Dim 4	28.55	141.17	28.49	79.22
	Dim 5	28.79	149.21	28.72	92.79
	Dim 6	28.78	148.44	28.75	90.89
	Dim 7	27.41	197.11	27.36	120.23
100	**Dim 1**	35.37	89.53	35.19	42.38
	Dim 2	35.28	104.77	35.18	46.6
	Dim 3	35.33	118.08	35.21	41.08
	Dim 4	28.63	311.89	28.54	164.41
	Dim 5	28.88	325.94	28.76	172.36
	Dim 6	28.85	302.32	28.79	158.64
	Dim 7	27.49	419.27	27.4	223.38

Tabela A.2 – Valores de PSNR e tempo médio consumidos para todos as combinações de fatores controláveis no espaço de cor HSV

		Amostragem		**Splitting**	
		PSNR [dB]	**Δt [s]**	**PSNR [dB]**	**Δt [s]**
1	**Dim 1**	33.11	3.57	33.78	3.36
	Dim 2	33.27	3.53	33.98	3.38
	Dim 3	38.3	4.05	38.39	3.31
	Dim 4	26.35	4.7	26.8	4.5
	Dim 5	26.42	4.71	26.93	4.55
	Dim 6	31	4.79	31.57	4.73
	Dim 7	26.08	5.53	26.57	5.51
5	**Dim 1**	34.21	6.72	34.11	5.76
	Dim 2	34.39	7.14	34.27	4.91
	Dim 3	38.94	7.87	38.68	5.41
	Dim 4	27.12	13.59	27.37	9.69
	Dim 5	27.2	13.15	27.47	10
	Dim 6	31.92	14.07	31.75	10.17
	Dim 7	26.84	16.91	27.06	12.59
10	**Dim 1**	34.37	9.62	34.1	8.34
	Dim 2	34.57	10.12	34.25	7.52
	Dim 3	39.12	11.34	38.84	8.14
	Dim 4	27.47	27.47	27.59	16.32
	Dim 5	27.56	31.98	27.63	15.18
	Dim 6	32.13	30.65	31.69	14.98
	Dim 7	27.19	46.41	27.23	19.24
50	**Dim 1**	34.23	27.12	34.17	24.3
	Dim 2	34.45	37.53	34.33	30.96
	Dim 3	39.21	52.64	38.99	27.89
	Dim 4	27.82	122.54	27.79	79.54
	Dim 5	27.94	143.23	27.88	77.93
	Dim 6	32.14	103.27	32.02	79.77
	Dim 7	27.55	180.09	27.49	106.19
100	**Dim 1**	34.5	67.86	34.1	43.97
	Dim 2	34.73	76.89	34.34	61.9
	Dim 3	39.14	81.33	39.03	51.25
	Dim 4	27.89	242.43	27.87	158.69
	Dim 5	28.02	309.11	27.92	161.16
	Dim 6	32.36	245.23	31.96	140.7
	Dim 7	27.63	375.09	27.53	206.73

Tabela A.3 – Valores de PSNR e tempo médio consumidos para todos as combinações de fatores controláveis no espaço de cor CIE Lab

		Amostragem		**Splitting**	
		PSNR [dB]	**Δt [s]**	**PSNR [dB]**	**Δt [s]**
1	**Dim 1**	37.68	4.62	37.66	3.98
	Dim 2	32.29	4.55	32.79	3.97
	Dim 3	32.48	4.13	32.93	3.92
	Dim 4	31.26	5.75	31.51	5.07
	Dim 5	25.74	5.42	26.09	5.09
	Dim 6	25.87	5.29	26.24	4.97
	Dim 7	25.74	6.12	26.06	6.74
5	**Dim 1**	37.94	5.58	37.83	4.65
	Dim 2	32.91	5.84	32.93	4.84
	Dim 3	33.1	5.88	33.29	4.82
	Dim 4	31.84	9.29	31.76	8.44
	Dim 5	26.56	12.69	26.61	8.76
	Dim 6	26.71	11.39	26.76	8.601
	Dim 7	26.55	14.02	26.56	11.49
10	**Dim 1**	38.19	7.631	37.74	5.78
	Dim 2	33.25	8.75	33.06	6.05
	Dim 3	33.46	8.99	33.32	5.9
	Dim 4	32.12	15.83	31.86	11.35
	Dim 5	26.79	21.39	26.76	13.64
	Dim 6	26.95	19.44	26.95	14.05
	Dim 7	26.78	24.19	26.74	17.51
50	**Dim 1**	38.09	20.68	37.58	13.96
	Dim 2	33.17	24.3	33.03	16.45
	Dim 3	33.35	22.48	33.24	13.86
	Dim 4	32.09	49.41	31.83	35.67
	Dim 5	27.1	108.11	26.97	53.33
	Dim 6	27.26	95.11	27.14	53.11
	Dim 7	27.08	110.86	26.94	75.28
100	**Dim 1**	38.28	41.08	37.44	26.92
	Dim 2	33.19	42.15	33.18	26.17
	Dim 3	33.37	39.82	33.22	26.25
	Dim 4	32.11	90.38	31.85	94.49
	Dim 5	27.13	202.42	27	132.87
	Dim 6	27.29	185.94	27.19	111.59
	Dim 7	27.12	220.14	26.99	169.25