PROJETO DE UM CODIFICADOR EFICIENTE ATRAVÉS DA TÉCNICA DE QUANTIZAÇÃO VETORIAL RECURSIVA DE ESTADOS FINITOS

Rodrigo Marendaz Silva Pimenta

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Rio de Janeiro

Março de 2016

PROJETO DE UM CODIFICADOR EFICIENTE ATRAVÉS DA TÉCNICA DE QUANTIZAÇÃO VETORIAL RECURSIVA DE ESTADOS FINITOS

Rodrigo Marendaz Silva Pimenta

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

_____

Prof. José Gabriel Rodríguez Carneiro Gomes, Ph.D.

_____

Profª. Mariane Rembold Petraglia, Ph.D.

_____

Prof. David Fernandes Cruz Moura, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2016

Pimenta, Rodrigo Marendaz Silva

Projeto de um codificador eficiente através da técnica de quantização vetorial recursiva de estados finitos/ Rodrigo Marendaz Silva Pimenta. – Rio de Janeiro: UFRJ/COPPE, 2016.

XIII, 64p.: il.; 29,7 cm.

Orientador: José Gabriel Rodríguez Carneiro Gomes.

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia Elétrica, 2016.

Referências Bibliográficas: p. 57-58

1. Compressão de Dados. 2. Quantização Vetorial. 3. Entropia. 4. Multiplicadores de Lagrange. I. Gomes, José Gabriel Rodríguez Carneiro. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título

À minha família

AGRADECIMENTOS

Agradeço a Deus, por trilhar meu caminho, por entregar meu destino nas mãos de pessoas tão especiais: meus pais, e pela benção da resiliência.

Agradeço aos meus pais, Adilson e Nezilda, por terem me iniciado no caminho da educação, e por terem me ensinado desde cedo a acreditar que tudo é possível, e que as ferramentas básicas necessárias são esforço e dedicação.

Agradeço à minha esposa Simone, por estar sempre ao meu lado, me fazendo acreditar que o impossível é apenas mais um obstáculo a ser ultrapassado. Ao seu companheirismo, amor, paciência, compreensão, ciúmes e flutuações hormonais. Seu apoio fez do meu sonho, o nosso sonho.

Agradeço a meus filhos Gabriel e Giovana, pelas noites em claro, pela energia que não cessa, e pelos carinhos que não tem preço. Vocês me fizeram ter a certeza de que todo esforço valeria a pena.

Agradeço à minha irmã Vanessa, por sempre me apoiar, e ao amigo-irmão Newton, que enxergou em uma pedra bruta a possibilidade de lapidação, e que teve a ousadia assertiva de me incentivar na volta ao caminho acadêmico.

Agradeço aos professores do Programa de Engenharia Elétrica, todos importantes para minha formação, à professora Mariane e ao professor David pela participação na banca examinadora, ao professor Petraglia e à professora Mariane pelo acolhimento, e em especial, ao meu orientador José Gabriel, pelos conselhos, paciência e dedicação. Ao Mestre, com carinho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

PROJETO DE UM CODIFICADOR EFICIENTE ATRAVÉS DA TÉCNICA DE QUANTIZAÇÃO VETORIAL RECURSIVA DE ESTADOS FINITOS

Rodrigo Marendaz Silva Pimenta

Março/2016

Orientador: José Gabriel Rodríguez Carneiro Gomes

Programa: Engenharia Elétrica

Esse trabalho desenvolve um estudo da técnica conhecida como Quantização Vetorial de Estados Finitos (*finite state vector quantization* – FSVQ), técnica que advém da Quantização Vetorial (*vector quantization* – VQ), mas com a característica particular de ser um codificador com memória e com estados finitos. A fundamentação teórica e a metodologia são apresentadas para, na sequência, permitir a aplicação através da criação de codificadores de diferentes taxas de bits por vetor, para cada VQ e FSVQ projetado. Codificadores diferentes são comparados em termos de taxa de bits, distorção, e requisitos de armazenamento de memória. Uma ênfase é dada para a avaliação global do compromisso entre o desempenho taxa-distorção e os requisitos de armazenamento de memória. Para ilustrar os algoritmos utilizados nessa investigação, o código-fonte completo é dado no apêndice.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

PROJECT OF AN EFFICIENT ENCODER BASED ON FINITE-STATE RECURSIVE VECTOR QUANTIZATION

Rodrigo Marendaz Silva Pimenta

March/2016

Advisor: José Gabriel Rodríguez Carneiro Gomes

Department: Electrical Engineering

This work presents a study of the technique known as Finite State Vector Quantization (FSVQ), which derives from Vector Quantization (VQ), but with the particular feature of being an encoder with memory and finite states. The theoretical foundation and methodology are presented, in order to explain the subsequent design of VQ and FSVQ encoders at different vector bit rates. Different encoders are compared in terms of bit rate, distortion, and memory storage requirements. An emphasis is placed on the overall assessment of the trade-off between rate-distortion performance and memory storage requirements. To further illustrate the algorithms used in this investigation, the complete source code is given in an appendix.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

Um dos maiores dilemas do avanço tecnológico na área de Engenharia de Telecomunicações está na demanda cada vez maior por capacidade de rede de transmissão de dados frente à capacidade de expansão dessa mesma rede. Aumentar a taxa de transmissão de dados infinitamente não é uma escolha.

Essa rede de transmissão pode ser pensada como uma rede de transporte de uma operadora comercial de telefonia, que possui uma escala de rede de transmissão de grandes dimensões, assim como uma rede de transmissão para aplicações mais específicas, de menor escala, e com capacidades muito mais limitadas.

Redes de transmissão de dados por satélite apresentam limitações físicas. Os *transponders* possuem recursos limitados e precisam ser utilizados de maneira eficiente. Canais de comunicação militares são redes de transmissão com características de troca de informação intermitente para que não sejam interceptadas pelas forças inimigas. Exigem alta eficiência na transmissão de informações. O processamento de imagens de câmeras digitais envolve o processo de captura, codificação e transmissão das imagens em equipamentos portáteis, de mínimas dimensões. Realizando uma codificação eficiente é possível minimizar a quantidade de informações a serem transmitidas nesses circuitos internos.

A eficiência a ser atingida envolve técnicas de compressão, em sua grande maioria com perdas, que permitem minimizar a quantidade de informação a ser transmitida, ou até mesmo transmiti-la com utilização de menos recursos do canal de transmissão, mas com o compromisso de permitir a recomposição da informação, mesmo que com perda, mas que seja o mais próximo possível da informação original, permitindo a necessária fidelização dos dados para sua aplicação.

Duas técnicas de compressão são amplamente difundidas: quantizadores escalares e quantizadores vetoriais. Os quantizadores escalares são comumente utilizados como sistemas de compressão de dados por conta de sua simplicidade e bom desempenho quando utilizam uma banda de transmissão suficientemente grande. Existe a possibilidade de melhora do desempenho de compressão utilizando técnicas de

quantização vetorial, permitindo transmitir os dados utilizando menos recursos do canal de transmissão. A contrapartida é o aumento na complexidade do sistema.

A utilização de Quantização Vetorial para compressão de imagens é detalhadamente analisada por Gersho [1] [2] e é uma prática bem difundida no meio acadêmico. Como exemplo, podemos citar Cosman et al [3], que revisou a técnica de Quantização Vetorial para mapeamento de vetores de intensidade de pixels em um número limitado de possíveis vetores binários indexados (O autor denomina essa técnica como algoritmo de compressão de imagens popular), e estuda a utilização de Quantização Vetorial para simplificar tarefas de processamento de imagens e reduzir a complexidade computacional dessas tarefas; Gomes [4] e Haas et al [5] que estudaram o método de processamento de imagem no plano focal, adotando esquemas de compressão que realizam a codificação antes da conversão analógico-digital, concentrando a maior parte do processamento em circuitos CMOS (*complementary metal-oxide silicon*). Para permitir a implementação do processamento no sensor, foram utilizadas variações de baixa complexidade da técnica de Quantização Vetorial; Swaszek [6] que estudou o algoritmo denominado Quantizador Vetorial de uma iteração (*one-pass vector quantization*) baseado na análise da sequência estatística das características locais dos dados de treino, inspirado no conceito de projeto de uma rede neural para classificação e reconhecimento de padrão [7]; Nasrabadi e King [8] que estudaram diversas técnicas de Quantização Vetorial, como exemplo a espacial, a preditiva, a de transformada, a híbrida e a binária, usadas para codificar imagens digitais, combinado com técnicas convencionais de codificação de imagens; e Gersho e Cuperman [9] que revisaram e apresentaram novos resultados obtidos na codificação de formas de onda da fala utilizando Quantização Vetorial.

Esse projeto se propõe ao estudo da técnica de compressão conhecida como Quantização Vetorial de Estados Finitos (*finite state vector quantization* – FSVQ), técnica que advém da Quantização Vetorial (*vector quantization* – VQ), mas com a característica particular de ser um codificador com memória e com estados finitos. A escolha de um codificador com memória para esse projeto se deu pela intenção de utilizar a compressão de imagens através da codificação de blocos de pixels, aproveitando a existência de memória entre os pixels de imagens, característica explorada na técnica do FSVQ.

O Capítulo 2 traz a base teórica e conceitos de Quantização Vetorial (VQ), como exemplo a entropia, estabelecendo os fundamentos para o entendimento da pesquisa proposta. Permite o entendimento das diferenças e semelhanças entre a Quantização Vetorial com memória e sem memória, além de introduzir o conceito de Quantização Vetorial de Estados Finitos (FSVQ). O Capítulo 3 discorre sobre a metodologia da pesquisa utilizada, apresentando e comentando os algoritmos que foram aplicados na obtenção dos resultados. O Capítulo 4 apresenta os resultados da pesquisa, desde a validação dos métodos de Quantização Vetorial com vetores de duas e cinco dimensões, passando pela criação de dicionários ótimos através de uma imagem utilizada para o aprendizado, em diferentes taxas de bit por vetor, até os testes dos dicionários em imagens naturais de pessoas, animais e paisagem. O Capítulo 5 traz a conclusão da pesquisa proposta.

2. CONCEITOS FUNDAMENTAIS

Esse capítulo traz a base teórica e conceitos de Quantização Vetorial (VQ), estabelecendo os fundamentos para o entendimento da pesquisa proposta. Permite o entendimento das diferenças e semelhanças entre a Quantização Vetorial com memória e sem memória, além de introduzir o conceito de Quantização Vetorial de Estados Finitos (FSVQ).

2.1. Considerações Iniciais

A possibilidade de compressão de uma imagem está diretamente relacionada à distribuição estatística presente no arranjo dos pixels. Existe grande probabilidade que pixels vizinhos pertençam a uma mesma forma da imagem. Essa característica permite a dedução de um pixel através do conhecimento prévio de outro pixel vizinho.

Dependendo da área de aplicação, a compressão de uma imagem pode ser definida de duas maneiras: sem perda ou com perda. Quando a compressão deve garantir a recuperação exata dos dados da informação original, é conhecida como sem perda (*lossless*). Quando se permite algum tipo de perda de informação, com a reconstrução aproximada dos dados da informação original, é conhecida como com perda (*lossy*) [10].

Para decidir se uma taxa média de bits é adequada para codificar uma imagem, medimos a entropia da imagem. A entropia permite identificar o número médio de bits necessários para codificação sem repetição de símbolos de um alfabeto. Métodos de compressão com perdas produzem um número médio de bits inferior à entropia da imagem [11].

2.2. Entropia

Suponha uma série de experimentos independentes, onde cada experimento resulte em L valores ($\boldsymbol{v}_1$, $\boldsymbol{v}_2$, ..., $\boldsymbol{v}_L$) de uma variável aleatória V qualquer. Shannon [12] associa a cada experimento uma quantidade i cujo nome é informação-própria e está diretamente ligada ao logaritmo do inverso da probabilidade de ocorrência do vetor v_k. . Essa quantidade é definida pela Equação 1.

$$i(v_k) = \log \frac{1}{P(v_k)}. \quad (1)$$

Quando o logaritmo está na base dois, a unidade designada para a quantidade informação-própria é bits. A quantidade de informação-própria média associada ao experimento é chamada de entropia *H* e é definida por [13] [14] conforme Equações 2 e 3.

$$H = E[i(v)] = \sum_{k=1}^{L} i(v_k)P(v_k) = \sum_{k=1}^{L} P(v_k) \log \frac{1}{P(v_k)} \quad (2)$$

$$H = -\sum_{k=1}^{L} P(v_k) \log P(v_k). \quad (3)$$

Utilizaremos dois exemplos de cálculo de entropia para facilitar a compreensão. O primeiro caso será o envio de cinco pacotes de informações $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5\}$ de uma fonte por um mesmo canal. Cada pacote de informação pode ser enviado, com igual probabilidade de ocorrência.

$$P(v_k) = \frac{1}{5}, \quad k = 1,2,3,4,5. \quad (4)$$

Então a entropia dessa fonte será 2.3219 bits.

No segundo caso, teremos o envio dos mesmos cinco pacotes de informação dessa fonte, pelo mesmo canal, mas agora com probabilidades de ocorrência diferentes.

$$P(v_1) = \frac{1}{2}, \ P(v_2) = \frac{1}{4}, \ P(v_3) = \frac{1}{8}, \ P(v_4) = \frac{1}{16}, \ P(v_5) = \frac{1}{16}. \quad (5)$$

Para o segundo caso, a entropia da mesma fonte será 1.8750 bits. Essa diferença ocorre porque a entropia leva em consideração as diferentes probabilidades de ocorrência com objetivo de codificar a informação esperada com maior eficiência dos símbolos. Para duas fontes com o mesmo número de símbolos, exigirá mais recursos do sistema aquela que tiver maior entropia. Quando os símbolos forem independentes e equiprováveis, chegaremos à entropia máxima. Quando existir correlação, e os símbolos futuros dependerem de símbolos passados, a entropia será menor [15].

Para atingir a entropia calculada no segundo caso, faz-se necessário utilizar uma representação eficiente, com menos bits representando os pacotes de informação que ocorrem com mais frequência. Um algoritmo largamente difundido para gerar essa

representação eficiente é o código de Huffman [16]. O funcionamento desse código está apoiado em três importantes premissas:

1. Os códigos correspondentes às ocorrências de maior probabilidade não podem ser maiores que os códigos associados às ocorrências de menor probabilidade;
2. As duas ocorrências de menor probabilidade devem possuir códigos associados de mesmo tamanho;
3. As duas ocorrências de menor probabilidade devem possuir códigos associados idênticos, com exceção para o último bit.

Aplicando as premissas do código de Huffman, podemos definir os seguintes códigos associados aos pacotes de informação do segundo caso exemplificado conforme Figura 1.

pacote de informação	código associado
v_1	0
v_2	10
v_3	110
v_4	1110
v_5	1111

Figura 1 – Exemplo de códigos associados de Huffman

Shannon [12] também comprovou que ao dividirmos as mensagens em blocos de informação, calculando a entropia de cada bloco da mesma maneira que se calculava para cada componente escalar individual, existe a tendência de a entropia se aproximar da entropia máxima.

Para aplicações com imagens que exigem alta fidelidade, são permitidas técnicas de compressão com métodos de pouca, ou nenhuma perda, como CALIC [17] e JPEG-LS [18]. Porém, essas técnicas alcançam taxas de compressão muito baixas, ou até nenhuma taxa de compressão. Os recentes avanços das técnicas de compressão com perdas incluem métodos que não permitem a reconstrução exata dos dados originais de entrada, mas permitem altas taxas de compressão. Dentre esses métodos, podemos citar as técnicas de Transformada Direta de Cosseno (*discrete cosine transform* - DCT) [14], Redes Neurais [19] e Quantização Vetorial [1].

A técnica de DCT é excelente em termos de compactação de energia. Os coeficientes mais significativos se acumulam no início do vetor de dados. Entretanto, a necessidade de representação com precisão finita sempre causará perda de informação antes mesmo da aplicação de qualquer forma de quantização. Com o objetivo de minimizar a distorção, é indicado para compressão de imagens com poucas componentes de alta frequência (bordas).

Métodos de Redes Neurais tem a característica de tender o resultado para mínimos locais e pode apresentar problemas de aprendizagem como a convergência prematura e o *overtraining*. Além da necessidade de definição e ajuste de parâmetros críticos.

2.3. Quantização Vetorial

Quantização é o processo de representar um pacote de informação de uma fonte, cujo alfabeto original possui tamanho infinito, por um alfabeto de tamanho finito. Esse processo de mapeamento é irreversível e por isso classificado como uma técnica de compressão com perdas. Esse processo é o mesmo que ocorre em um conversor analógico-digital.

A Quantização consiste em dois processos: um processo de codificação que permite mapear o pacote de informação da fonte utilizando uma das palavras do alfabeto finito, e um processo de decodificação que permite reconstruir o pacote de informação através da palavra mapeada no processo anterior. O processo de codificação pode ser definido como o ato de particionar o alfabeto e o processo de decodificação como o ato de obter palavras que representem cada partição.

O caso mais simples de Quantização ocorre quando as partições do alfabeto possuem o mesmo tamanho e é conhecido como Quantização Uniforme. Quando as partições do alfabeto acompanham a não uniformidade de probabilidade de ocorrência dos pacotes de informação, as partições possuem tamanhos diferentes e a Quantização é conhecida como Quantização Não-Uniforme. Exemplos de Quantização Uniforme e Não-Uniforme podem ser vistas na Figura 2.

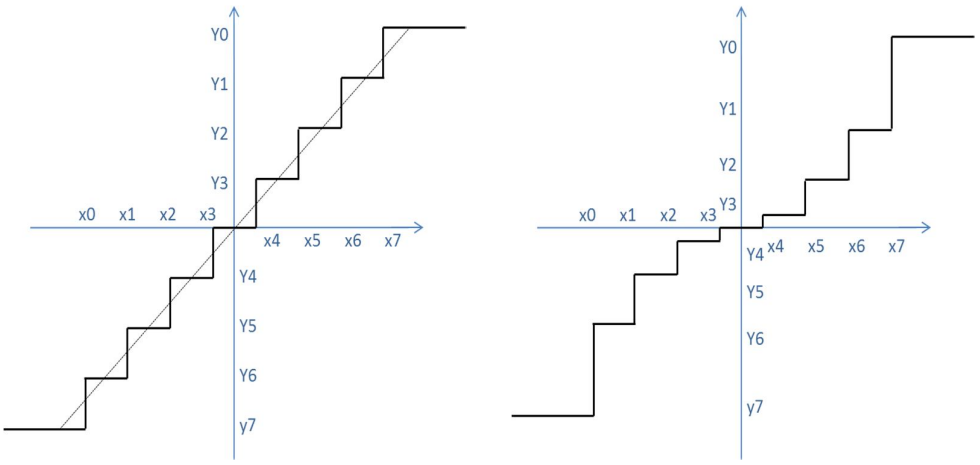


Figura 2 – Exemplo de Quantização Uniforme e Quantização Não Uniforme

Quando o processo de Quantização ocorre na base de amostra-a-amostra, o processo é chamado de Quantização Escalar. Quando ocorre na base de conjuntos de amostras, o processo é chamado de Quantização Vetorial. A Quantização que ocorre na base de conjunto de amostras (vetores), principalmente quando a sequência de amostras é correlacionada, resulta em menor distorção quando comparada à utilização de Quantização Escalar na mesma taxa. A contrapartida dessa redução da distorção é o aumento da complexidade do codificador e a criação de um codificador personalizado pelas características dos pacotes de informação da fonte [14]. Exemplo dos processos de Quantização Escalar e Quantização Vetorial podem ser vistos na Figura 3.

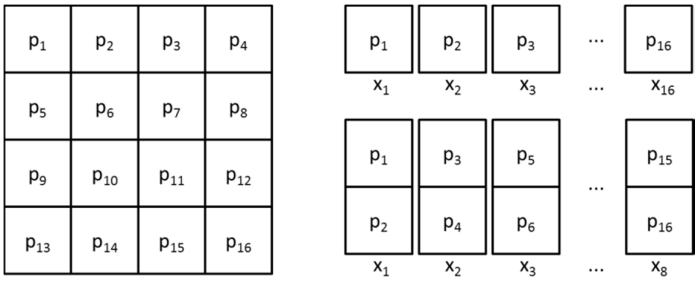


Figura 3 – Exemplo de figura composta por 16 pixels (p_1 até p_{16}) com processo de Quantização Escalar formando 16 amostras (x_1 a x_{16}) e com processo de Quantização Vetorial formando 8 vetores (x_1 a x_8) de 2-D

Um Quantizador Vetorial (VQ) [4] de dimensão *M* e tamanho *Y* será definido pelo mapeamento de vetores de entrada de dimensão *M* em vetores que pertençam a um conjunto finito de tamanho *Y*, conforme Equação 6.

$$Q : \mathbb{R}^M \rightarrow Y. \quad (6)$$

Ao conjunto $Y = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_K\}$ damos o nome de dicionário. E cada vetor $\mathbf{y}_k$ é conhecido como uma palavra do dicionário. O VQ permite que a informação seja codificada por uma palavra $\mathbf{y}_k$ do dicionário, e que apenas o índice *k* seja transmitido e que seja minimamente necessário para que o decodificador recupere a informação.

Os dados de entrada (fonte) são organizados em uma matriz de entrada **X**, composta por *N* vetores de tamanho *M*, conforme Equações 7 e 8.

$$X \in \mathbb{R}^{M \times N} \quad (7)$$

$$\mathbf{x}(n) \in \mathbb{R}^M, \quad n = 1, \ldots, N. \quad (8)$$

No codificador, esses dados são quantizados e codificados por $\boldsymbol{y}_k$ palavras definidas pelo dicionário **Y** de tamanho *K*, sendo α o mapeamento do codificador e k(n) o índice *k* do dicionário Y que codifica a entrada **x**(n) a ser transmitida e pode ser verificado nas Equações 9 e 10.

$$\mathbf{x}(n) \longrightarrow \boxed{\alpha} \longrightarrow k(n). \quad (9)$$

$$\mathbf{Y} \in \mathbb{R}^{M \times K}. \quad (10)$$

É possível realizar apenas a transmissão do índice *k* referente à palavra $\boldsymbol{y}_k$ do dicionário **Y** que representa o vetor de entrada. O decodificador é capaz de recuperar a informação x(n) realizando a predição $\hat{\mathbf{x}}(n)$ correspondente ao índice *k* e sua respectiva palavra do dicionário **Y**, conforme exemplo da Tabela 1, utilizando o mapeamento do decodificador β, conforme Equação 11.

Tabela 1 – Exemplo de tabela de relação do índice *k* com a palavra y_k do dicionário Y

k(n)	1	2	3	...	K
$\hat{\mathbf{x}}(n)$	$\mathbf{y}_1$	$\mathbf{y}_2$	$\mathbf{y}_3$	...	$\mathbf{y}_K$

$$k(n) \longrightarrow \boxed{\beta} \longrightarrow \hat{\mathbf{x}}(n). \quad (11)$$

Para medir a eficiência do VQ, é utilizada uma medida de distorção *D* baseada na distância euclidiana (*mean-squared error* - MSE) entre o vetor de entrada e a palavra y_k do dicionário **Y** que a representa, conforme desenvolvido nas Equações de 12 a 15.

$$\hat{x}(n) = \beta(\alpha(x(n))), \quad (12)$$

$$d(x,y) = \|x - y\|^2 = \sum_{m=1}^{M}(x_m - y_m)^2, \quad (13)$$

$$D = \frac{1}{N}\sum_{n=1}^{N} d(x(n),\hat{x}(n)) = \frac{1}{N}\sum_{n=1}^{N}\|x(n) - \hat{x}(n)\|^2, \quad (14)$$

$$D = E[d(x,\hat{x})]. \quad (15)$$

Para o projeto de um VQ ótimo com relação a *D*, são utilizadas duas condições de optimalidade conhecidas como Condição da Partição e Condição do Centroide, descritas a seguir.

2.3.1. Condição de Partição (Codificador):

Fixando **Y**, devemos calcular a divisão de **X** em *k* subconjuntos, considerando que $\mathbf{y}_1$, $\mathbf{y}_2$, ...,$\mathbf{y}_k$ são conhecidos [4] [20]:

$$D = \int d(x,\hat{x})\, f_x(x)dx, \quad (16)$$

$$D \geq \int [\min_{k\in\{1,2,\ldots,k\}} d(x,y_k)]\, f_x(x)dx. \quad (17)$$

A partição ótima é então atingida através das sub-regiões R_i nas quais a distância *d* entre o vetor de entrada ***x*** e sua respectiva representação decodificada $\hat{x}$ for a menor possível, conforme Equação 19.

$$R_i = \{x \mid d(x,y_i) < d(x,y_j)\ \forall j \neq i\}. \quad (18)$$

$$d(x,\hat{x}) = \min_{k\in\{1,2,\ldots,k\}} d(x,y_k). \quad (19)$$

Obedecendo aos critérios pré-estabelecidos será possível garantir a menor distorção no conjunto de vetores **x** de entrada e $\hat{\mathbf{x}}$ de saída, garantindo que haverá uma distorção total D mínima para o conjunto de dados.

$$D_{min} = \frac{1}{N}\sum_{n=1}^{N}\left\|x(n) - y_{k(n)}\right\|^2. \quad (20)$$

2.3.2. Condição de Centroide (Decodificador):

Fixando a divisão de X em sub-regiões R_i, para cada sub-região é calculado um novo centroide, e consequentemente é estabelecido um novo dicionário Y com novas palavras $\mathbf{y}_k$ [4] [20], estabelecidas pela Equação 24.

$$D_{min} = \frac{1}{N} \sum_{n=1}^{N} \left(\mathbf{x}(n) - \mathbf{y}_{k(n)}\right)^T \left(\mathbf{x}(n) - \mathbf{y}_{k(n)}\right), \quad (21)$$

$$D_{min} = \frac{1}{N} \sum_{n=1}^{N} \left(\mathbf{x}(n)^T\mathbf{x}(n) - \mathbf{x}(n)^T\mathbf{y}_{k(n)} - \mathbf{y}_{k(n)}^T\mathbf{x}(n) + \mathbf{y}_{k(n)}^T\mathbf{y}_{k(n)}\right), \quad (22)$$

$$\frac{\partial D_{min}}{\partial y_k} = \frac{2}{N} \left(N_k\mathbf{y}_k - \sum_{\mathbf{x}(n)\in R_k} \mathbf{x}(n)\right) = 0, \quad (23)$$

$$\boldsymbol{y}_k = \frac{1}{N_k} \sum_{\mathbf{x}(n)\in R_k} \mathbf{x}(n). \quad (24)$$

2.3.3. O Algoritmo de Lloyd

Existe uma diversidade de algoritmos de otimização empregados em projetos de quantização vetorial. Em sua grande maioria, são apresentados como algum tipo de variação do Algoritmo de Lloyd Generalizado (*Generalized Lloyd Algorithm* – GLA) [20] [21].

Basicamente, o GLA segue um algoritmo pré-definido, aplicado reiteradamente por *i* vezes, em um conjunto de dados de entrada para treino X = {**x**(1), **x**(2), ..., **x**(N)}, partindo de um dicionário inicial $\mathbf{Y}_0$ pré-definido.

O primeiro passo é a atualização do codificador através da aplicação da condição de partição. Na sequência é feita a atualização do decodificador através da aplicação da condição de centroide. E por último é verificada a condição de convergência, quando é medida a distorção D da iteração e comparada com a mesma medida da iteração anterior, garantindo que o GLA apresente uma redução de distorção a cada iteração. Esses três passos descritos são repetidos por um número arbitrário de iterações.

Para vetores de dados com duas dimensões, que ocorrem quando os pixels são agrupados de dois em dois, formando vetores de tamanho dois, é possível visualizar as atualizações dos codificadores e dicionários através das partições de Voronoi [1]. As condições de fronteira e de partição são traduzidas em polígonos que delimitam as

fronteiras de subespaço ótimo a cada iteração. O centroide de cada subespaço representa as palavras do dicionário.

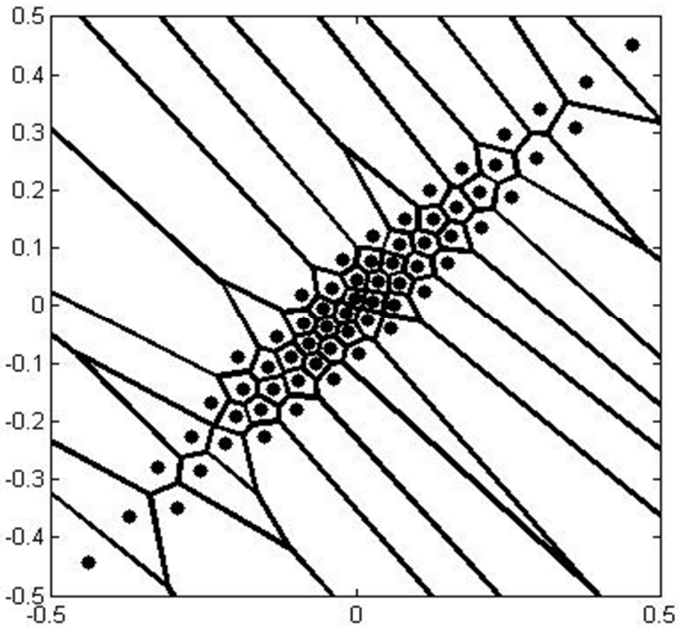


Figura 4 – Exemplo de partições de Voronoi aplicadas a um dicionário de 6 bits, com polígonos delimitando as fronteiras de 64 subespaços ótimos de cada iteração e pontos indicando seus respectivos centróides

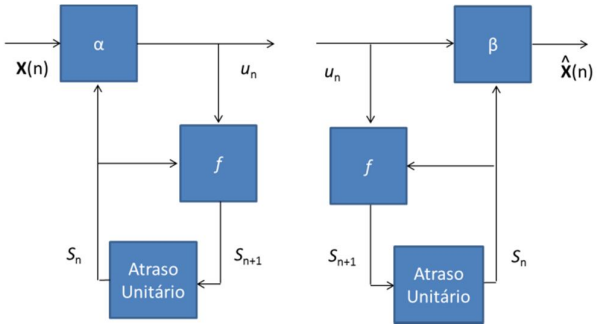
2.4. Quantizador Vetorial de Estados Finitos - FSVQ

Um Quantizador Vetorial de Estados Finitos é um Quantizador Vetorial com memória. Um sistema de codificação com memória tem como característica que o índice gerado pelo codificador não depende apenas do vetor de entrada atual, mas também dos vetores de entrada anteriores ao atual. O mesmo ocorre para o sistema de decodificação. Esse também dependerá de informações passadas [1] [4] [20].

O FSVQ pode ser considerado como uma coleção de Quantizadores Vetoriais adequada ao tratamento de fontes vetoriais aleatórias com memória. Uma versão mais simplista do FSVQ é o Quantizador Vetorial Recursivo (RVQ) que generaliza para a forma vetorial o Quantizador Escalar Recursivo.

O funcionamento do RVQ está descrito na Figura 5. Dada uma sequência de entrada de vetores aleatórios $\mathbf{x}(n)$, n = 0, 1, ..., o codificador produz uma sequência de

símbolos u_n, n = 0, 1, ..., para o canal e uma sequência de estados S_n, n = 0, 1, ..., que descreverão de maneira eficiente o caminho do codificador passando por diferentes estados em resposta à sequência de dados entrantes no sistema. O vetor de entrada **x**(n) é um vetor real de dimensão k, ou seja, $\mathbf{x}(n) \in \mathbb{R}^k$, R é a taxa de codificação em bits por vetor, e a relação R/k é a resolução, ou taxa de bits por componente do vetor de entrada.



(a) (b)

Figura 5 – RVQ. (a) Codificador; (b) Decodificador

Para garantir que o decodificador seja capaz de entender a mensagem que será transmitida sem a necessidade de informações adicionais, é exigido que o primeiro estado S_0 seja conhecido. Dessa maneira, o próximo estado S_{n+1} será determinado pelo estado atual S_n junto com a sequência de símbolos u_n, através de algum mapeamento de função de próximo estado *f*, também conhecida como função transição de estado, exemplificada na Tabela 2.

$$S_{n+1} = f(u_n, S_n), n = 0, 1, \dots . \quad (25)$$

Tabela 2 – Exemplo numérico de mapeamento de uma função de próximo estado *f*, onde é possível estabelecer o próximo estado S_{n+1} com base nas informações do estado atual S_n e da sequência de símbolos u_n

$S_n \setminus u_n$	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
S_1	1	8	3	4	5	6	2	7
S_2	2	5	7	4	6	1	8	3
S_3	4	5	6	8	1	7	2	3
S_4	4	5	8	6	2	1	7	3
S_5	5	4	2	6	7	8	1	3
S_6	1	8	3	4	6	5	2	7
S_7	7	2	5	6	4	1	8	3
S_8	8	4	1	3	5	6	2	7

Dado o estado S_n, o codificador fará o mapeamento conforme Equação 26.

$$u_n = \alpha(\boldsymbol{X}_n, S_n). \quad (26)$$

Já o codificador, sabendo a sequência de símbolos e o estado atual, será capaz de gerar e representar o vetor de entrada pela sequência de símbolos u_n através da Equação 27.

$$\widehat{\boldsymbol{X}}_n = \beta(u_n, S_n). \quad (27)$$

Supondo que o codificador e decodificador se encontrem em um estado *S*, o decodificador terá a sua disposição uma reprodução de todos os vetores possíveis de serem reproduzidos. A essa coleção damos o nome de dicionário do estado S, conforme Equação 28.

$$Y_S = \{\beta(u, S); u \in N\}. \quad (28)$$

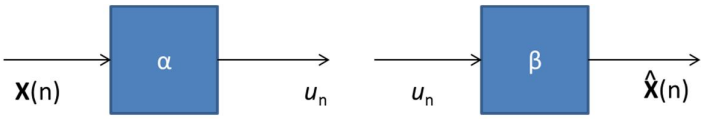
Logo, um RVQ sempre escolhe o índice de canal u_n que irá produzir a menor distorção possível no decodificador, dado que tanto o codificador quanto o decodificador estão em um estado comum. O RVQ depende de f, β, S, S_0 e da distância *d*.

$$\alpha(\boldsymbol{x}, S) = \min^{-1}{}_u \, d\big(\boldsymbol{x}, \beta(u, S)\big). \quad (29)$$

O mapeamento α do codificador da Equação 29 é determinado pela propriedade de mínima distorção. Consequentemente, em algum instante de tempo n, codificador e decodificador apresentam desempenho de um VQ com dicionário Y_{Sn}.

$$d\big(\boldsymbol{x}, \beta(\alpha(\boldsymbol{x}, S), S)\big) = \min_{\widehat{\boldsymbol{x}} \in Y_S} d(\boldsymbol{x}, \widehat{\boldsymbol{x}}). \quad (30)$$

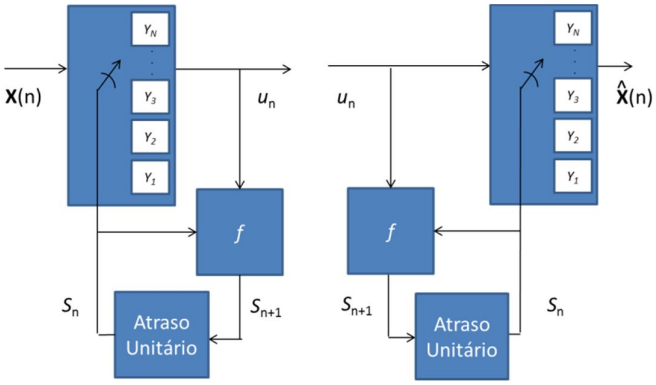
A única diferença do RVQ para o VQ fica por conta de que, a cada instante sucessivo, no RVQ, codificador e decodificador migram para um novo dicionário. A Figura 6 mostra o diagrama de blocos para o VQ. Pode-se perceber na figura que o mapeamento α do codificador e β do decodificador permanecem constante, trabalhando em um único dicionário $\boldsymbol{Y}$.



(a) (b)

Figura 6 – VQ. (a) Codificador; (b) Decodificador

O FSVQ é um RVQ que possui número finito de estados S = {1, 2, ... , N}. O desafio está na escolha dos melhores dicionários que representarão cada estado de maneira a possibilitar o decodificador chegar ao melhor resultado com a mínima distorção.



(a) (b)

Figura 7 – FSVQ. (a) Codificador; (b) Decodificador

O FSVQ pode ser entendido como uma coleção de N distintos VQs sem memória, juntos com uma regra que estabelece qual dicionário Y do VQ será utilizado para codificar o vetor de entrada corrente em um índice de canal. Esse índice u_n, acompanhado do estado atual S_n, permite determinar o próximo estado S_{n+1}e o próximo dicionário Y_N do VQ. O funcionamento do FSVQ pode ser visto na Figura 7.

3. METODOLOGIA DA PESQUISA

No início do trabalho foram utilizados como entrada do sistema 81.920 valores reais pseudoaleatórios e correlacionados. Esses pontos foram obtidos através de um processo auto-regressivo com coeficiente 0.95 para a memória referente à última amostra e com coeficiente 0.1 para o ruído aditivo gaussiano. A quantidade de pontos foi definida baseada na quantidade mínima necessária para estudo dos quantizadores pesquisados. O processo pseudoaleatório e correlacionado permitiu o controle dos avanços na pesquisa por trabalhar com dados de entrada previamente conhecidos.

Em um primeiro momento, esses pontos foram rearranjados em 40.960 vetores de duas dimensões. E em um segundo momento, os mesmos 81.920 pontos originais formaram 16.384 vetores de cinco dimensões. A formação de vetores 2-D foi importante no início da pesquisa. Permitiu um acompanhamento através de gráficos de 2-D durante as fases de desenvolvimento e análise do algoritmo utilizado. O código-fonte principal com o qual foram realizados os projetos de FSVQ e as comparações entre FSVQ e VQ é dado no Apêndice A.

A Figura 8 mostra um gráfico 2-D dos pontos pseudoaleatórios gerados (em preto) e com o diagrama de Voronoi durante quatro iterações do aprendizado do FSVQ de taxa final igual a 3 bpv. A primeira iteração é representada pelas linhas e pontos em azul. A segunda, terceira e quarta iteração correspondem às linhas e pontos em ciano, verde e amarelo, respectivamente. É possível identificar a migração dos centroides e a redefinição das 8 regiões a cada iteração.

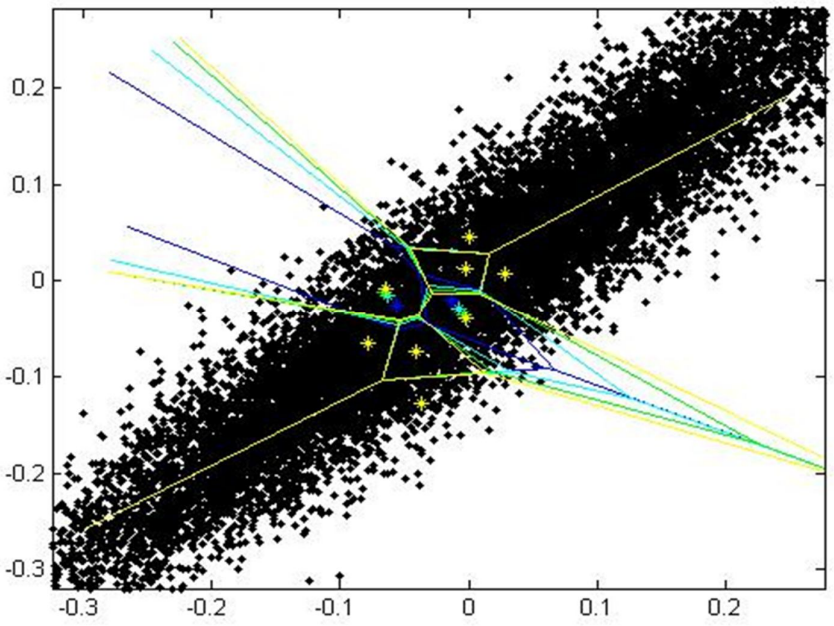


Figura 8 – Exemplo do diagrama de Voronoi para vetores de 2-D mostrando a modificação das regiões e suas respectivas centroides durante 4 iterações. A iteração 1 é representada pelas linhas e pontos em azul, a iteração 2 pelas linhas e pontos em ciano, a iteração 3 pelas linhas e pontos em verde e a iteração 4 pelas linhas e pontos em amarelo.

A dimensão e quantidade dos vetores de entrada devem ser escolhidas de forma a facilitar a obtenção de bons resultados de projeto. Aumentando a quantidade de vetores nos dados de treino, e a dimensão do vetor, espera-se a geração de dicionários e palavras ótimos. Porém, para um mesmo tamanho de imagem, não é possível o aumento concomitante da quantidade e do tamanho dos vetores. O aumento do tamanho do vetor gera a redução da quantidade de vetores, e vice-versa. Foi escolhido o vetor 5-D como o vetor mais apropriado para essa pesquisa. A análise detalhada dessa equação não é objeto de estudo nesse trabalho.

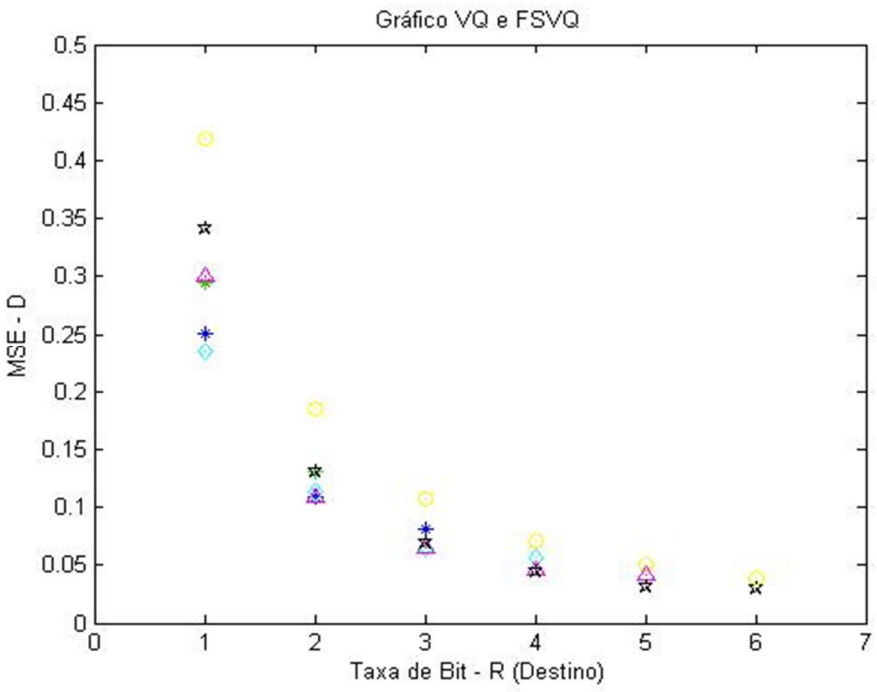


Figura 9 – Exemplo de gráfico para avaliação de desempenho de quantização vetorial em termos de distorção e taxa de bits, com o círculo amarelo indicando o MSE do VQ para a taxa de R bpv, o xis vermelho indicando o VQ original em 1 bpv com FSVQ final em 1 bpv, a cruz verde indicando o VQ original em 2 bpv com o FSVQ final em taxa de 1 ou 2 bpv, o asterisco azul indicando o VQ original em 3 bpv com o FSVQ final em taxa de 1, 2 ou 3 bpv, o diamante ciano indicando o VQ original em 4 bpv com o FSVQ final em taxa de 1, 2, 3 ou 4 bpv, o triângulo lilás indicando o VQ original em 5 bpv com o FSVQ final em taxa de 1, 2, 3, 4 ou 5 bpv, e a estrela preta indicando o VQ original em 6 bpv com o FSVQ final em taxa de 1, 2, 3, 4, 5 ou 6 bpv

Após análise e validação do método, utilizamos para treino e criação dos dicionários e palavras uma imagem de 512 × 512 pontos, em tons de cinza, o que nos permitiu gerar 52.428 vetores de cinco dimensões. Foi utilizada a imagem Lena por ser amplamente utilizada em pesquisas de Processamento Digital de Imagens.

A pesquisa se desenvolveu em VQs de taxa R bpv (bits por vetor), com R variando de 1 até 6 bpv. Não foi utilizada taxa além de 6 bpv pois foi percebido durante os testes que os codificadores e decodificadores acima dessa taxa estavam superdimensionados para tratar imagens de 512 × 512 pontos.

Para análise dos resultados foram gerados gráficos como os apresentados nas Figuras 9 a 12. O círculo amarelo indica o MSE do VQ para a taxa de R bpv. O xis vermelho indica o MSE do FSVQ que é baseado em um VQ original (sem memória) de 1 bit por vetor e que, a partir desse projeto original, transmite vetores à taxa original de

1 bit por vetor. Em outras palavras, a máquina de estados desse FSVQ tem dois estados e, em cada estado, o dicionário tem dois centroides. Na cruz verde, o VQ original tem 2 bpv e o FSVQ final pode ter taxa de 1 bpv ou 2 bpv. No asterisco azul, o VQ original tem 3 bpv e o FSVQ final pode ter taxa de 1, 2 ou 3 bpv. Os três símbolos restantes mostram: 4 bpv para 1 até 4 bpv (diamante ciano), 5 bpv para 1 até 5 bpv (triângulo lilás) e 6 bpv para 1 até 6 bpv (estrela preta).

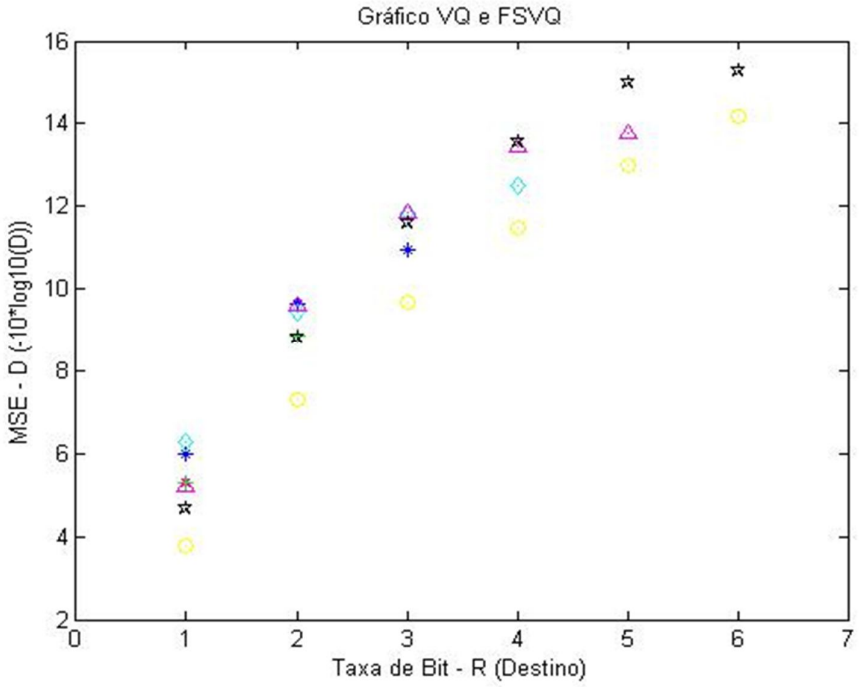


Figura 10 – Exemplo de gráfico para avaliação do desempenho de quantização vetorial em termos de distorção (*peak signal-to-noise ratio*, em dB) e taxa de bits, com o círculo amarelo indicando o MSE do VQ para a taxa de R bpv, o xis vermelho indicando o VQ original em 1 bpv com FSVQ final em 1 bpv, a cruz verde indicando o VQ original em 2 bpv com o FSVQ final em taxa de 1 ou 2 bpv, o asterisco azul indicando o VQ original em 3 bpv com o FSVQ final em taxa de 1, 2 ou 3 bpv, o diamante ciano indicando o VQ original em 4 bpv com o FSVQ final em taxa de 1, 2, 3 ou 4 bpv, o triângulo lilás indicando o VQ original em 5 bpv com o FSVQ final em taxa de 1, 2, 3, 4 ou 5 bpv, e a estrela preta indicando o VQ original em 6 bpv com o FSVQ final em taxa de 1, 2, 3, 4, 5 ou 6 bpv

Os gráficos mostrados nas Figuras 9, 10, 11 e 12 foram utilizados para medir os desempenhos taxa-distorção, entropia-distorção, e armazenamento-custo dos quantizadores vetoriais estudados. Os gráficos das Figuras 9 e 10 apresentam um exemplo da relação entre taxa e distorção MSE para cada R considerado. A diferença entre os gráficos está na escala do eixo MSE. No gráfico da Figura 4 a escala da distorção aparece em dB, que leva a uma forma alternativa, porém usual, para a apresentação da relação entre distorção e taxa de bits.

É visível a formação de uma casca inferior no gráfico da Figura 9, onde temos a menor distorção projetada para cada R estudado. Essa solução geométrica é conhecida como casca convexa inferior (Lower Convex Hull – LCH) [22] [23]. O mesmo pode ser visualizado no gráfico da Figura 10, mas com a formação de uma provável casca superior.

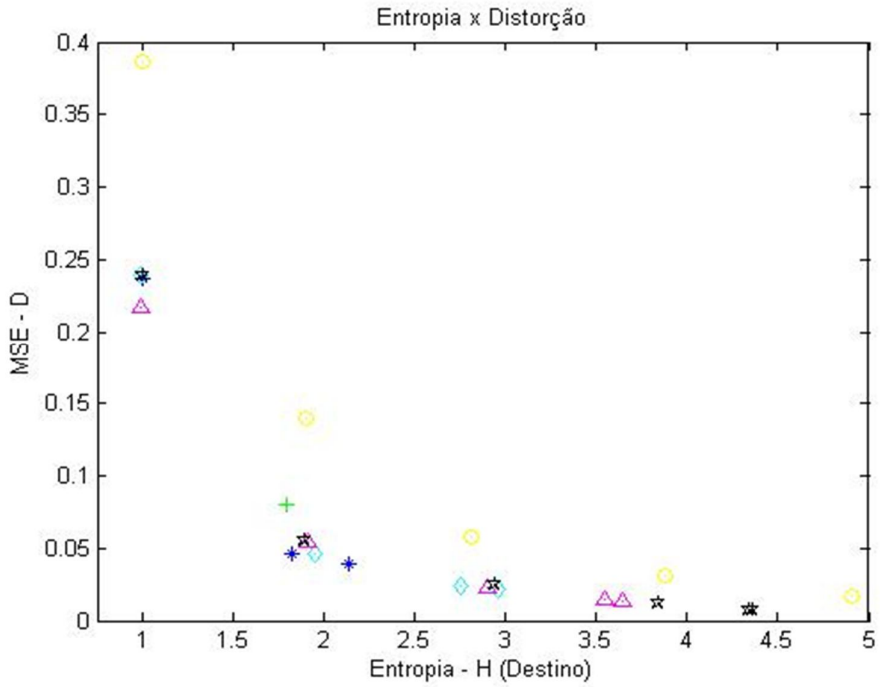


Figura 11 – Exemplo de gráfico para avaliação da relação entropia-distorção, com o círculo amarelo indicando o MSE do VQ para a taxa de R bpv, o xis vermelho indicando o VQ original em 1 bpv com FSVQ final em 1 bpv, a cruz verde indicando o VQ original em 2 bpv com o FSVQ final em taxa de 1 ou 2 bpv, o asterisco azul indicando o VQ original em 3 bpv com o FSVQ final em taxa de 1, 2 ou 3 bpv, o diamante ciano indicando o VQ original em 4 bpv com o FSVQ final em taxa de 1, 2, 3 ou 4 bpv, o triângulo lilás indicando o VQ original em 5 bpv com o FSVQ final em taxa de 1, 2, 3, 4 ou 5 bpv, e a estrela preta indicando o VQ original em 6 bpv com o FSVQ final em taxa de 1, 2, 3, 4, 5 ou 6 bpv

O gráfico da Figura 11 apresenta um exemplo de conjunto de pontos entropia-distorção de medidas de MSE para cada H calculado. Os valores de entropia mostrados na Figura 11 são valores médios: em cada estado, calcula-se a entropia com base nas probabilidades de utilização de cada centroide. Em seguida, a entropia global é calculada através de uma média ponderada das entropias de cada estado. É visível a formação de uma casca inferior que apresenta as prováveis melhores escolhas da relação analisada.

O gráfico da Figura 12 apresenta um exemplo de conjunto de pontos com coordenadas armazenamento versus desempenho (ou seja, medidas do espaço de

armazenamento necessário e do desempenho taxa-distorção) para cada VQ e FSVQ projetado. Essa relação de pontos permite a análise e escolha de quanto se estará disposto a comprometer em J (combinação linear lagrangeana entre D e H) e A (número de operações computacionais) de maneira a obter o melhor codificador para cada aplicação. Os números ao lado de cada ponto identificam as taxas de bits por vetor obtidas com o VQ-base sem memória, e as taxas de bpv finais de cada FSVQ.

O custo de armazenamento foi calculado como a quantidade de posições de memória mínima para manter as informações necessárias para codificação ou decodificação dos vetores de dados. Cada estado S custa θ multiplicações, onde θ é definido pelo produto do tamanho do vetor pelo tamanho dos dicionários. O armazenamento final será dado pelo produto das θ multiplicações de cada S pela quantidade de estados.

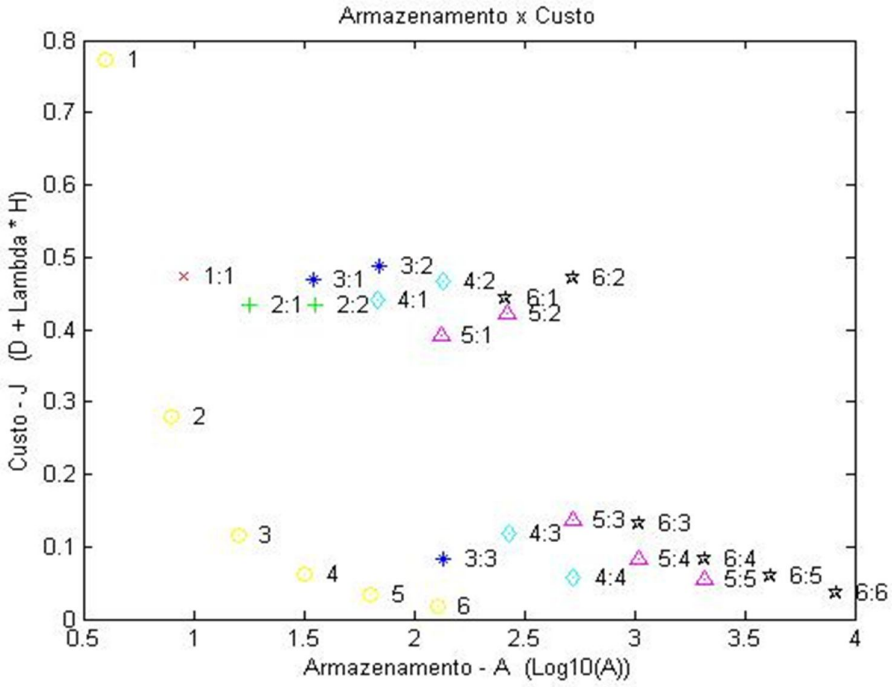


Figura 12 – Exemplo de gráfico para avaliação da relação entre o custo de implementação do FSVQ (armazenamento) e o "custo" J que representa o desempenho do codificador em termos de taxa e distorção, com o círculo amarelo indicando o MSE do VQ para a taxa de R bpv, o xis vermelho indicando o VQ original em 1 bpv com FSVQ final em 1 bpv, a cruz verde indicando o VQ original em 2 bpv com o FSVQ final em taxa de 1 ou 2 bpv, o asterisco azul indicando o VQ original em 3 bpv com o FSVQ final em taxa de 1, 2 ou 3 bpv, o diamante ciano indicando o VQ original em 4 bpv com o FSVQ final em taxa de 1, 2, 3 ou 4 bpv, o triângulo lilás indicando o VQ original em 5 bpv com o FSVQ final em taxa de 1, 2, 3, 4 ou 5 bpv, e a estrela preta indicando o VQ original em 6 bpv com o FSVQ final em taxa de 1, 2, 3, 4, 5 ou 6 bpv

O cálculo de J foi feito através da função lagrangeana da Equação 31.

$$J = D + \lambda \times H. \quad (31)$$

O multiplicador de Lagrange λ é calculado através da solução geométrica LCH aplicada a dados equivalentes aos apresentados na Figura 5. Cada um dos VQs e dos FSVQs estudados apresenta um par de informações D e H que especificam seu desempenho em distorção e entropia, respectivamente. Para um conjunto de quantizadores definidos, e através da solução LCH, é possível selecionar os valores de λ através do cálculo de inclinação de cada reta da casca formada conforme,

$$\lambda_i = -\frac{D_i - D_{i-1}}{H_i - H_{i-1}}, i = 2, \dots, I', \quad (32)$$

$$\lambda_1 = \lambda_2. \quad (33)$$

Um exemplo numérico de mapeamento de λ calculado para cada FSVQ com base na inclinação da casca convexa inferior pode ser visto na Tabela 3.

Tabela 3 – Exemplo numérico de mapeamento de λ calculado para cada FSVQ com base na inclinação da casca convexa

VQ			**FSVQ**	**1 bpv**	**2 bpv**	**3 bpv**	**4 bpv**	**5 bpv**	**6 bpv**
1 bpv	0,2304		**1 bpv**	0,1898					
2 bpv	0,1464		**2 bpv**	0,1900	0,0950				
3 bpv	0,0456		**3 bpv**	0,2728	0,1514	0,0150			
4 bpv	0,0223		**4 bpv**	0,2766	0,1613	0,0278	0,0048		
5 bpv	0,0118		**5 bpv**	0,2439	0,1501	0,0356	0,0095	0,0020	
6 bpv	0,0039		**6 bpv**	0,2521	0,1526	0,0355	0,0130	0,0087	0,0046

Para os VQs com cinco dimensões, no caso do projeto feito com a imagem Lena, com objetivo de validar o resultado encontrado para os dicionários gerados, utilizamos cinco imagens naturais distintas, sendo três de pessoas e duas de paisagens naturais. Foram geradas as medidas de distorção para cada imagem codificada e os resultados obtidos foram comparados com os valores de taxa e distorção esperados.

A pesquisa, cujos resultados são mostrados no Capítulo 4, se desenvolveu em torno do estudo de seis cenários de compressão de dados realizados através de um VQ, com as taxas de bit por vetor variando no intervalo de 1 bpv até 6 bpv.

Sequencialmente, foram estudados 21 cenários de compressão de dados realizados através de um FSVQ, sendo esse um VQ com memória, com as taxas de bit

por vetor variando no intervalo de 1 bpv até 6 bpv, e com a possibilidade de redução do tamanho dos dicionários, e consequente redução do custo de Armazenamento.

Para os seis cenários de compressão de dados realizados através do VQ, foi definido o dicionário localmente ótimo inicial, base para o projeto subsequente dos FSVQs, através de aprendizado sem memória, conforme algoritmo abaixo descrito para o caso dos experimentos com imagens naturais:

1. Carregar a imagem Lena de 512 × 512 pixels;
2. Rearranjar a matriz para uma nova matriz de tamanho 5 × 52.428 pixels;
3. Gerar um dicionário inicial através da média da matriz de entrada, adicionada de um ruído aleatório;
4. Aplicar a Condição de Partição e a Condição de Centroide reiteradamente para atingir um dicionário ótimo;
5. Definir o dicionário localmente ótimo do VQ.

No passo três do algoritmo descrito acima é necessária a adição de ruído para que seja possível formar centroides distintas no dicionário inicial. Como o primeiro dicionário será gerado através da média da matriz de entrada, ou seja, por um único vetor, repetido por k vezes para um dicionário de tamanho k, a cada repetição, do vetor 2 até o vetor k, é gerado e adicionado um ruído ao vetor.

Para os cenários de compressão de dados do FSVQ foi trabalhada a variação no intervalo inicial de 1 bpv até 6 bpv, e com a possibilidade de redução do tamanho dos dicionários desde 1 bpv até a taxa bpv inicial escolhida. A taxa bpv final estará associada ao tamanho do dicionário de cada estado do FSVQ.

Essa redução de estados é possível porque em cada estado são conhecidos e ordenados, do mais provável para o menos provável, os próximos estados mais prováveis dado que sabemos o estado atual. Na prática, a probabilidade está concentrada em alguns poucos estados mais prováveis, permitindo a não-utilização dos estados menos prováveis.

Essa abordagem permite também a “falsa” redução de uma taxa bpv inicial para uma mesma taxa bpv final. A criação de dicionários distintos para cada estado e a utilização da memória da tabela de estados (representando a função de próximo estado

apresentada na seção 2.3) permite obter um melhor desempenho comparado ao VQ de mesma taxa bpv.

Para facilitar o entendimento, podemos utilizar como exemplo a taxa inicial de 3 bpv e final de 2 bpv, para um vetor de 5-D. Serão criados oito estados iniciais, e no FSVQ cada estado possuirá 4 dicionários de 5-D. Existirá também uma tabela de próximo estado que permitirá saber, dado o estado atual, qual será o estado futuro dentre os 4 estados possíveis em um subconjunto do total de 8 estados, e consequentemente informando qual dos 4 dicionários de 5-D será utilizado para estimar e representar o vetor de 5-D de origem.

Segue abaixo o algoritmo aplicado para o projeto através do FSVQ:

1. Utilizar como dicionário inicial o dicionário localmente ótimo definido através do VQ com a mesma taxa bpv que está sendo trabalhada;
2. Utilizar como matriz de entrada a matriz de tamanho 5 × 52.428 pixels do algoritmo do projeto VQ, carregado da imagem Lena de 512 × 512 pixels;
3. Definir a função de próximo estado;
4. Criar os dicionários iniciais de cada estado com base na função de próximo estado e no dicionário ótimo do VQ;
5. Aplicar a Condição de Partição para definir o estado atribuído ao primeiro vetor 5-D de entrada;
6. Reiteradamente, para todos os vetores de entrada (iniciando com o segundo vetor 5-D de entrada):
    a. Definir o dicionário do estado atual;
    b. Aplicar a Condição de Partição para definir o estado atribuído ao próximo vetor 5-D de entrada;
7. Aplicar a Condição de Centroide para definir os dicionários ótimos de cada estado;
8. Loop com retorno ao passo seis até que a condição de convergência definida para o projeto seja atingida;
9. Definir os dicionários localmente ótimos do FSVQ.

Durante a fase de projeto, para cada VQ e para cada FSVQ, além da definição dos dicionários ótimos baseados nas características da imagem Lena, está sendo medida a distorção D, a entropia H, o custo J e o armazenamento A.

De posse dos dicionários localmente ótimos do VQ e do FSVQ, e da tabela de estados do FSVQ, com objetivo de estudos em casos diversos, foram utilizadas as imagens de teste mostradas no Capítulo 4 (Gabriel - Figura 21 e Giovana, Menina, Arara e Paisagem – Figura 28) e, utilizando o mesmo codificador e decodificador definidos no treino realizado com a imagem Lena, as imagens foram geradas e reconstruídas, permitindo gerar as medidas de distorção D de cada imagem testada.

4. RESULTADOS

O Capítulo 4 apresenta os resultados da pesquisa divididos em cinco partes. A seção 4.1 trata dos resultados experimentais com dados pseudo-aleatórios, validando os métodos de Quantização Vetorial com vetores de 2-D e 5-D. A seção 4.2 mostra o resultado de projeto de quantizadores com base na imagem Lena. Critérios de parada precisaram ser definidos durante a fase de pesquisa. Essas escolhas foram detalhadas e explicadas na seção 4.3. Já nas seções 4.4 e 4.5, foram apresentados resultados experimentais com imagens naturais.

4.1. Estudo de Dados Pseudoaleatórios

A primeira iteração no gráfico da Figura 13 (diamante vermelho) mostra o Erro Médio Quadrático (MSE) utilizado como medida de distorção D para a compressão de dados realizada através de um VQ sem memória, projetado com base em 40.960 vetores de duas dimensões. As iterações subsequentes (diamante preto), de 2 a 11, identificam o MSE para o treino do FSVQ. Todas essas informações descritas estão exemplificadas nesse gráfico para um VQ de 2 bpv e um FSVQ de 2 bpv projetado com base em um VQ de 6 bpv sem memória.

O gráfico da Figura 14 tem a mesma característica descrita no parágrafo anterior, apresentando o MSE do VQ de 2 bpv (diamante vermelho) na iteração 1, e o MSE da curva de aprendizado do FSVQ de 2 bpv projetado com base em um VQ de 6 bpv sem memória nas iterações subsequentes (diamante preto). A diferença entre os gráficos da Figura 13 e da Figura 14 é que este utiliza 16.384 vetores de cinco dimensões como entrada do sistema e aquele utiliza 40.960 vetores de duas dimensões como entrada do sistema.

Outros 40 gráficos foram gerados, sendo 20 gráficos para os vetores de duas dimensões e 20 gráficos para os vetores de cinco dimensões, todos eles com a mesma característica de aprendizado.

Pode ser visto, para todos os gráficos gerados, que a primeira medida MSE do FSVQ (iteração 2) é, em maioria quase absoluta, maior que a medida MSE do VQ. Mesmo sendo um sistema sem memória, o VQ passou por um aprendizado ao aplicar a Condição de Partição e a Condição de Centroide reiteradamente, atingindo um

dicionário ótimo, e se encontra no seu mínimo MSE local. A primeira iteração do FSVQ ocorre utilizando uma metodologia com memória, mas baseada em dicionários ótimos estabelecidos pela metodologia VQ, sem memória. Após poucas iterações de aprendizado do FSVQ, já é possível identificar medida MSE menor que a melhor medida MSE do VQ.

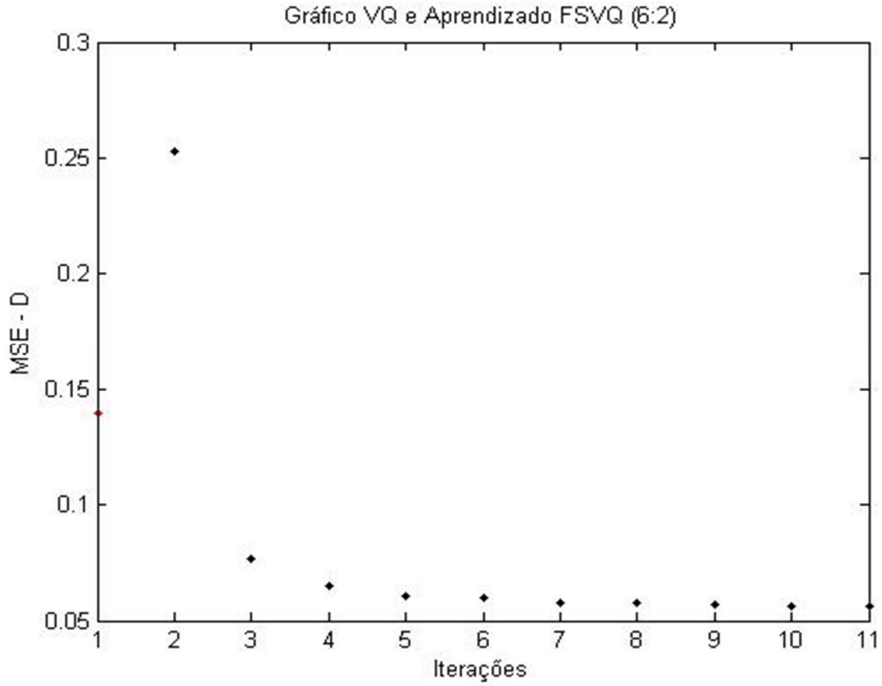


Figura 13 – VQ de 2 bpv (iteração 1) e Aprendizado de um FSVQ de 2 bpv projetado com base em um VQ de 6 bpv sem memória (iterações 2 até 11) para vetores de entrada 2-D

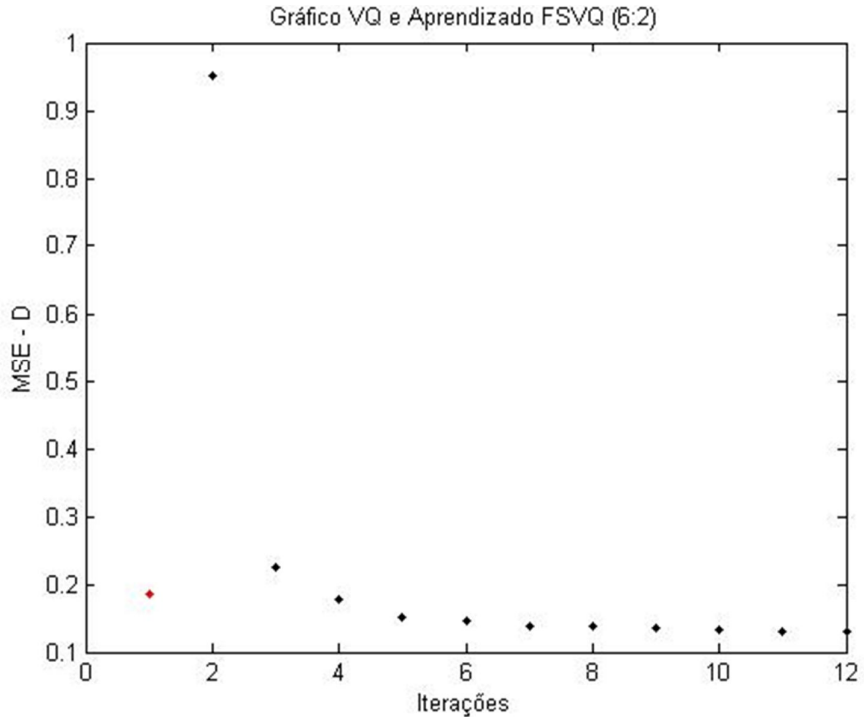


Figura 14 – VQ de 2 bpv (iteração 1) e aprendizado de um FSVQ de 2 bpv projetado com base em um VQ de 6 bpv sem memória (iterações 2 a 12) para vetores de entrada 5-D

Percebe-se também que ao se aplicar o aprendizado no FSVQ existe a tendência natural de se chegar a um mínimo MSE local, que é, para os casos estudados, menor que o mínimo local atingido pelo VQ.

Em primeira análise, fica evidente que o mínimo MSE atingido com o aprendizado do VQ é maior no gráfico da Figura 13. Porém, para esse tipo de comparação, faz-se mister a normalização dos dados pelo tamanho do vetor. Ao trabalhar com um vetor de 2-D à taxa de 2 bpv, como na Figura 13, divide-se o MSE resultante por 2 e se obtém a taxa em bits por pixel. O mesmo pode ser feito para a Figura 14, onde o vetor é de 5-D à mesma taxa de 2 bpv. Mas agora divide-se o MSE resultante por 5. Após normalização, será possível constatar que o MSE em bits por pixel no gráfico da Figura 14 será menor que o MSE na mesma medida da Figura 13. Esses cálculos não serão apresentados por não fazerem parte do objetivo desse trabalho.

Em seguida, estão apresentados os valores encontrados para D, D (dB), H, A, λ e J durante estudo dos dados pseudo-aleatórios em 2-D (Tabelas 4, 6, 8, 10, 12 e 14) e em 5-D (Tabelas 5, 7, 9, 11, 13 e 15), sempre relacionados à taxa de bits por vetor.

Tabela 4 – Distorção após treino com VQ e FSVQ, projetado com base em 40.960 vetores de 2-D

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	0,3826		1 bpv	0,2370					
2 bpv	0,1396		2 bpv	0,2369	0,0804				
3 bpv	0,0574		3 bpv	0,2370	0,0460	0,0395			
4 bpv	0,0307		4 bpv	0,2395	0,0466	0,0243	0,0218		
5 bpv	0,0167		5 bpv	0,2171	0,0548	0,0230	0,0144	0,0132	
6 bpv	0,0088		6 bpv	0,2390	0,0560	0,0250	0,0123	0,0079	0,0078

Tabela 5 – Distorção após treino com VQ e FSVQ, projetado com base em 16.384 vetores de 5-D

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	0,4195		1 bpv	0,2945					
2 bpv	0,1854		2 bpv	0,2948	0,1305				
3 bpv	0,1078		3 bpv	0,2513	0,1085	0,0808			
4 bpv	0,0709		4 bpv	0,2353	0,1139	0,0656	0,0560		
5 bpv	0,0503		5 bpv	0,3005	0,1093	0,0656	0,0455	0,0420	
6 bpv	0,0381		6 bpv	0,3419	0,1313	0,0693	0,0441	0,0315	0,0296

Tabela 6 – Distorção (dB) após treino com VQ e FSVQ, projetado com base em 40.960 vetores de 2-D

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	4,1726		1 bpv	6,2525					
2 bpv	8,5511		2 bpv	6,2543	10,9474				
3 bpv	12,4109		3 bpv	6,2525	13,3724	14,0340			
4 bpv	15,1286		4 bpv	6,2069	13,3161	16,1439	16,6154		
5 bpv	17,7728		5 bpv	6,6334	12,6122	16,3827	18,4164	18,7943	
6 bpv	20,5552		6 bpv	6,2160	12,5181	16,0206	19,1009	21,0237	21,0791

Tabela 7 – Distorção (dB) após treino com VQ e FSVQ, projetado com base em 16.384 vetores de 5-D

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	3,7727		1 bpv	5,3091					
2 bpv	7,3189		2 bpv	5,3047	8,8439				
3 bpv	9,6738		3 bpv	5,9981	9,6457	10,9259			
4 bpv	11,4935		4 bpv	6,2838	9,4348	11,8310	12,5181		
5 bpv	12,9843		5 bpv	5,2216	9,6138	11,8310	13,4199	13,7675	
6 bpv	14,1908		6 bpv	4,6610	8,8174	11,5927	13,5556	15,0169	15,2871

Tabela 8 – Entropia após treino com VQ e FSVQ, projetado com base em 40.960 vetores de 2-D

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	0,9997		1 bpv	0,9988					
2 bpv	1,9011		2 bpv	0,9985	1,7886				
3 bpv	2,8119		3 bpv	0,9988	1,8180	2,1355			
4 bpv	3,8803		4 bpv	0,9946	1,9489	2,7565	2,9586		
5 bpv	4,9040		5 bpv	0,9928	1,9146	2,8991	3,5421	3,6412	
6 bpv	5,8860		6 bpv	0,9968	1,8936	2,9380	3,8352	4,3407	4,3629

Tabela 9 – Entropia após treino com VQ e FSVQ, projetado com base em 16.384 vetores de 5-D

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	0,9997		**1 bpv**	0,9988					
2 bpv	1,8968		**2 bpv**	0,9984	1,7479				
3 bpv	2,8848		**3 bpv**	0,9918	1,9204	2,4478			
4 bpv	3,9144		**4 bpv**	0,9915	1,9405	2,8494	3,1904		
5 bpv	4,9240		**5 bpv**	0,9915	1,8938	2,9331	3,7485	3,9112	
6 bpv	5,9562		**6 bpv**	0,9775	1,8811	2,8999	3,8452	4,5469	4,6543

Tabela 10 – Medidas do espaço de armazenamento necessário para VQ e FSVQ, projetado com base em 40.960 vetores de 2-D

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	4		**1 bpv**	9					
2 bpv	8		**2 bpv**	18	36				
3 bpv	16		**3 bpv**	35	70	137			
4 bpv	32		**4 bpv**	68	136	268	528		
5 bpv	64		**5 bpv**	133	266	527	1044	2073	
6 bpv	128		**6 bpv**	262	524	1042	2072	4126	8228

Tabela 11 – Medidas do espaço de armazenamento necessário para VQ e FSVQ, projetado com base em 16.384 vetores de 5-D

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	10		**1 bpv**	21					
2 bpv	20		**2 bpv**	42	84				
3 bpv	40		**3 bpv**	83	166	329			
4 bpv	80		**4 bpv**	164	328	652	1296		
5 bpv	160		**5 bpv**	325	650	1295	2580	5145	
6 bpv	320		**6 bpv**	646	1292	2578	5144	10270	20516

Tabela 12 – Valores do multiplicador de Lagrange λ após treino com VQ e FSVQ, projetado com base em 40.960 vetores de 2-D

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	0,3863		**1 bpv**	0,2373					
2 bpv	0,0734		**2 bpv**	0,1981	0,1981				
3 bpv	0,0204		**3 bpv**	0,2331	0,2434	0,0204			
4 bpv	0,0079		**4 bpv**	0,2021	0,2159	0,0338	0,0122		
5 bpv	0,0034		**5 bpv**	0,1761	0,1922	0,0390	0,0193	0,0117	
6 bpv	0,0015		**6 bpv**	0,2041	0,2189	0,0368	0,0185	0,0119	0,0063

Tabela 13 – Valores do multiplicador de Lagrange λ após treino com VQ e FSVQ, projetado com base em 16.384 vetores de 5-D

VQ			**FSVQ**	**1 bpv**	**2 bpv**	**3 bpv**	**4 bpv**	**5 bpv**	**6 bpv**
1 bpv	0,4197		**1 bpv**	0,2948					
2 bpv	0,0977		**2 bpv**	0,2191	0,2191				
3 bpv	0,0374		**3 bpv**	0,1538	0,1801	0,0527			
4 bpv	0,0181		**4 bpv**	0,1280	0,1545	0,0672	0,0282		
5 bpv	0,0102		**5 bpv**	0,2119	0,2329	0,0544	0,0353	0,0213	
6 bpv	0,0064		**6 bpv**	0,2330	0,2635	0,0742	0,0357	0,0268	0,0178

Tabela 14 – Custo J após treino com VQ e FSVQ, projetado com base em 40.960 vetores de 2-D

VQ			**FSVQ**	**1 bpv**	**2 bpv**	**3 bpv**	**4 bpv**	**5 bpv**	**6 bpv**
1 bpv	0,7724		**1 bpv**	0,4740					
2 bpv	0,2792		**2 bpv**	0,4346	0,4346				
3 bpv	0,1149		**3 bpv**	0,4698	0,4884	0,0831			
4 bpv	0,0615		**4 bpv**	0,4405	0,4674	0,1174	0,0580		
5 bpv	0,0334		**5 bpv**	0,3920	0,4229	0,1360	0,0828	0,0560	
6 bpv	0,0175		**6 bpv**	0,4425	0,4706	0,1330	0,0833	0,0596	0,0352

Tabela 15 – Custo J após treino com VQ e FSVQ, projetado com base em 16.384 vetores de 5-D

VQ			**FSVQ**	**1 bpv**	**2 bpv**	**3 bpv**	**4 bpv**	**5 bpv**	**6 bpv**
1 bpv	0,8391		**1 bpv**	0,5890					
2 bpv	0,3707		**2 bpv**	0,5135	0,5135				
3 bpv	0,2156		**3 bpv**	0,4038	0,4544	0,2097			
4 bpv	0,1417		**4 bpv**	0,3622	0,4137	0,2571	0,1458		
5 bpv	0,1005		**5 bpv**	0,5106	0,5504	0,2252	0,1779	0,1251	
6 bpv	0,0762		**6 bpv**	0,5697	0,6269	0,2845	0,1813	0,1533	0,1123

4.2. Treino do Codificador: Imagem Lena

Para análise e validação visual do estudo, foi utilizada para treino e escolha dos dicionários localmente ótimos a imagem Lena da Figura 15, em tons de cinza, de tamanho 512 × 512 pontos, permitindo trabalhar com até 52.428 vetores de 5-D.

Figura 15 – Imagem Lena, 512 × 512

O estudo da imagem Lena foi acompanhado do cálculo de medidas de desempenho como a taxa-distorção, a entropia-distorção, e o armazenamento-custo. Essas medidas foram representadas nas Figuras 16 a 19.

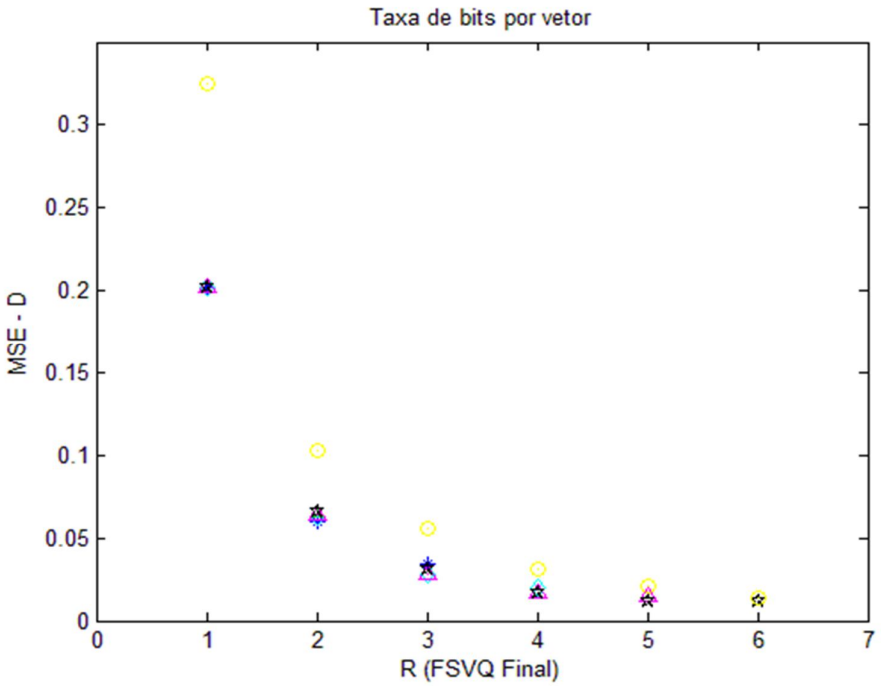


Figura 16 – Gráfico taxa-distorção após treino

Para facilitar a compreensão dos dados, foi gerada a matriz da Tabela 16 com todas as informações de MSE do VQ e do FSVQ, nas taxas de bits por vetor estudadas. Essas informações foram representadas na Figura 16. As linhas da matriz indicam a taxa R do VQ, que é igual à taxa inicial do FSVQ. As colunas da matriz identificam a taxa R final do FSVQ. A Tabela 17 possui as mesmas informações da Tabela 16, porém apresentadas em forma alternativa e usual para estudo da relação entre distorção e taxa de bits (dB). Os dados da Tabela 16 foram representados na Figura 17.

Tabela 16 – Resultado (MSE na célula e taxa final de bits por vetor na coluna)

VQ			**FSVQ**	**1 bpv**	**2 bpv**	**3 bpv**	**4 bpv**	**5 bpv**	**6 bpv**
1 bpv	0,3251		**1 bpv**	0,2022					
2 bpv	0,1030		**2 bpv**	0,2022	0,0632				
3 bpv	0,0560		**3 bpv**	0,2022	0,0596	0,0343			
4 bpv	0,0311		**4 bpv**	0,2022	0,0630	0,0281	0,0198		
5 bpv	0,0213		**5 bpv**	0,2021	0,0636	0,0283	0,0164	0,0144	
6 bpv	0,0142		**6 bpv**	0,2021	0,0661	0,0312	0,0166	0,0121	0,0115

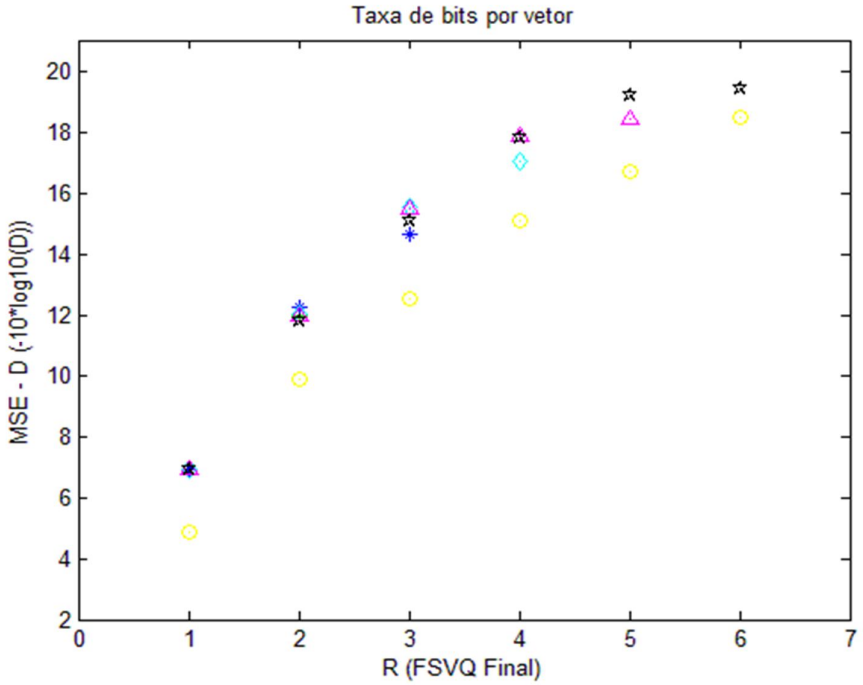


Figura 17 – Gráfico taxa-distorção após treino com taxa em dB

Através dessa análise, pode ser identificado que os menores MSE (casca convexa inferior da Figura) ocorreram para a relação da taxa R_i base do VQ original com a taxa R_f do FSVQ final (R_i:R_f) apresentadas a seguir: 5:1, 2:2, 4:3, 5:4, 6:5 e 6:6.

Com base apenas nessa análise, essas relações R_i:R_f seriam as mais indicadas para projeto.

Tabela 17 – Resultado (MSE em dB na célula e taxa final de bits por vetor na coluna)

VQ			**FSVQ**	**1 bpv**	**2 bpv**	**3 bpv**	**4 bpv**	**5 bpv**	**6 bpv**
1 bpv	4,8798		**1 bpv**	6,9422					
2 bpv	9,8716		**2 bpv**	6,9422	11,9928				
3 bpv	12,5181		**3 bpv**	6,9422	12,2475	14,6471			
4 bpv	15,0724		**4 bpv**	6,9422	12,0066	15,5129	17,0333		
5 bpv	16,7162		**5 bpv**	6,9443	11,9654	15,4821	17,8516	18,4164	
6 bpv	18,4771		**6 bpv**	6,9443	11,7980	15,0585	17,7989	19,1721	19,3930

Após identificar a menor distorção de cada relação de taxa de bits por vetor, foi possível calcular o valor da entropia. A entropia foi calculada com o objetivo de considerar a possibilidade de codificação dos índices transmitidos através de códigos de comprimento variável, aproveitando as diferentes probabilidades de ocorrência destes índices em cada estado. Esses resultados foram apresentados no gráfico da Figura 18.

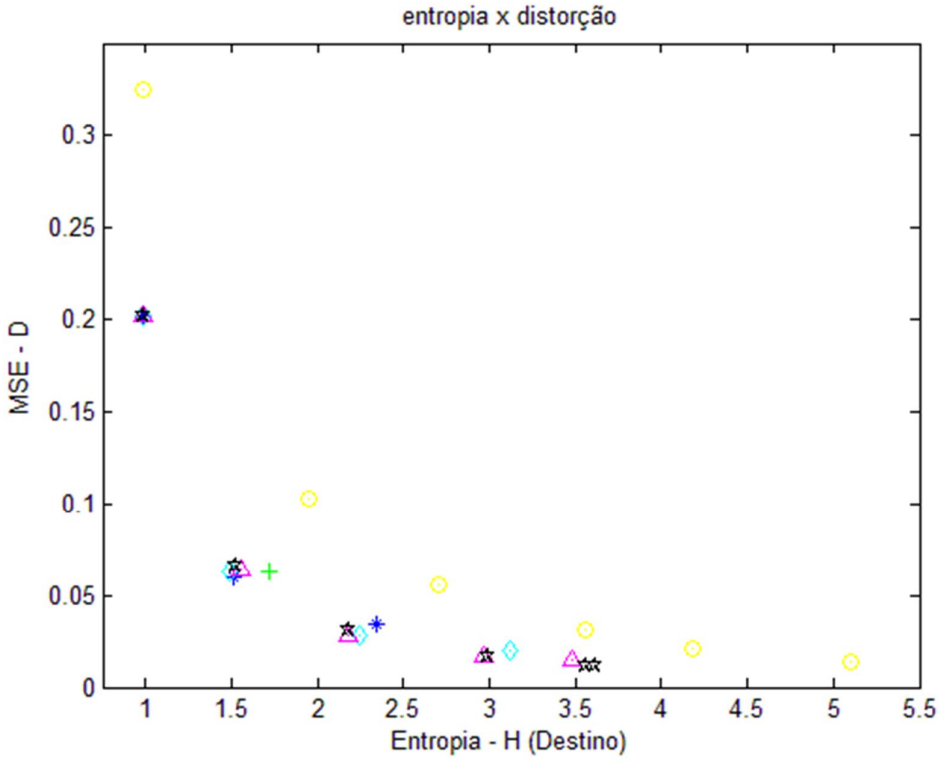


Figura 18 – Gráfico entropia-distorção após treino

Para facilitar a compreensão dos dados, foi gerada a matriz da Tabela 18 com todas as informações de H associadas ao MSE (visto nas tabelas anteriores) do VQ e do FSVQ, em todas as taxas R estudadas. As linhas da matriz indicam a taxa R do VQ e a taxa R_i inicial do FSVQ. As colunas da matriz identificam a taxa R_f final do FSVQ.

Através dessa análise, pode ser identificado que as taxas R_f finais que obtiveram os menores H (casca inferior da Figura 18) foram 4:1, 4:2, 5:3, 5:4, 5:5 e 6:6. Com base apenas nessa nova análise, as relações R_i:R_f mais indicadas para projeto foram redefinidas.

Tabela 18 – Valores de entropia associados aos diferentes FSVQs projetados

VQ			**FSVQ**	**1 bpv**	**2 bpv**	**3 bpv**	**4 bpv**	**5 bpv**	**6 bpv**
1 bpv	0,9805		**1 bpv**	0,9828					
2 bpv	1,9439		**2 bpv**	0,9833	1,7147				
3 bpv	2,6992		**3 bpv**	0,9833	1,5059	2,3468			
4 bpv	3,5610		**4 bpv**	0,9827	1,4861	2,2444	3,1158		
5 bpv	4,1860		**5 bpv**	0,9831	1,5514	2,1776	2,9713	3,4800	
6 bpv	5,0983		**6 bpv**	0,9831	1,5224	2,1805	2,9906	3,5562	3,6154

De posse das informações de distorção e de entropia, foi possível calcular a função-custo J localmente minimizada para cada taxa R do VQ e para cada uma das 21 relações entre taxa inicial e taxa final do FSVQ estudadas. Essas informações foram cruzadas com o custo de armazenamento A. Para cálculo de A, foram multiplicados os números referentes à quantidade de estados (dois elevado à taxa R inicial do FSVQ), ao tamanho do dicionário (dois elevado à taxa R final do FSVQ) e à dimensão dos vetores de entrada. Essas informações foram representadas na Figura 19.

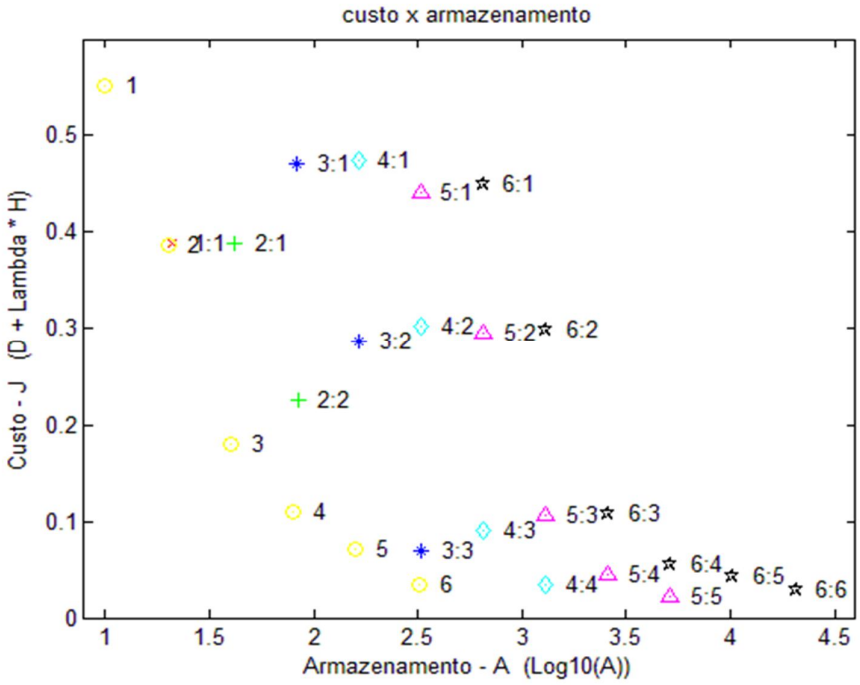


Figura 19 – Gráfico da relação entre desempenho taxa-distorção (função-custo J) e custo de armazenamento (armazenamento)

Para facilitar a compreensão dos dados, foram geradas a matrizes das Tabelas 19 e 20 com todas as informações de A e J do VQ e do FSVQ, em todas as Taxas R estudadas.

Tabela 19 – Armazenamento (Posições de Memória)

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	10		**1 bpv**	21					
2 bpv	20		**2 bpv**	42	84				
3 bpv	40		**3 bpv**	83	166	329			
4 bpv	80		**4 bpv**	164	328	652	1.296		
5 bpv	160		**5 bpv**	325	650	1.295	2.580	5.145	
6 bpv	320		**6 bpv**	646	1.292	2.578	5.144	10.270	20.516

Tabela 20 – Função-custo J

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	0,5510		**1 bpv**	0,3887					
2 bpv	0,3876		**2 bpv**	0,3889	0,2261				
3 bpv	0,1790		**3 bpv**	0,4704	0,2876	0,0696			
4 bpv	0,1104		**4 bpv**	0,4740	0,3026	0,0904	0,0347		
5 bpv	0,0705		**5 bpv**	0,4419	0,2964	0,1059	0,0447	0,0213	
6 bpv	0,0342		**6 bpv**	0,4500	0,2985	0,1087	0,0556	0,0428	0,0281

Obviamente, o armazenamento do VQ é menor que o armazenamento no FSVQ. E também, quanto maior a taxa bpv final do FSVQ, maior o armazenamento. O armazenamento, quando analisado junto com o custo, mostra que a redução de bpv no FSVQ pode ser interessante, dependendo de quanto armazenamento se dispõe e quanto de custo se admite perder. Com base nessa nova análise, as relações R_i:R_f mais indicadas para projeto foram redefinidas para 1:1, 2:2, 3:3, 4:4, 5:5 e 6:6.

4.3. Critérios de Parada para o Projeto do FSVQ

Diferente do que ocorre com o aprendizado do VQ, que sempre converge para um dicionário de menor distorção, para o FSVQ, essa característica não é garantida durante o aprendizado. Foi verificado durante o estudo que no cálculo dos novos dicionários ocorreram iterações onde houve aumento da medida de distorção.

Um critério de parada que pode ser utilizado é a definição de uma quantidade de iterações. Outro critério possível é a definição de um critério de parada com base na

redução mínima aceitável na medida de distorção. Os dois critérios de parada possuem pontos positivos e negativos. Ao utilizarmos o critério de parada baseado na quantidade de iterações é permitido o aprendizado até o limite de iterações estabelecidas, mesmo que a parada ocorra durante o processo de piora do dicionário. Já o segundo critério definido, pode levar à obtenção da redução mínima aceitável na medida de distorção, mas que poderia ainda ser melhorada se houvessem mais iterações no processo.

Foram feitas experiências e registrados os resultados para 500 iterações. Esses resultados foram apresentados na Tabela 21 em intervalo de 50 em 50 iterações. Para facilitar o entendimento foi gerada a Tabela 22 que apresenta a diferença entre a iteração indicada na coluna e a iteração ocorrida 50 vezes antes. Por exemplo, na coluna da iteração 250 da Tabela 22 é apresentada a diferença entre a distorção da iteração 250 e a distorção da iteração 200.

Tabela 21 - Distorção registrada para 500 iterações apresentadas em intervalo de 50 em 50 iterações

	1	50	100	150	200	250	300	350	400	450	500
01:01	0,3251	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022
02:01	0,4832	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022
02:02	0,1030	0,0632	0,0632	0,0632	0,0632	0,0632	0,0632	0,0632	0,0632	0,0632	0,0632
03:01	0,7159	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022
03:02	0,0806	0,0596	0,0596	0,0596	0,0596	0,0596	0,0596	0,0596	0,0596	0,0596	0,0596
03:03	0,0560	0,0342	0,0343	0,0343	0,0343	0,0343	0,0343	0,0343	0,0343	0,0343	0,0343
04:01	0,7779	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022	0,2022
04:02	0,1078	0,0627	0,0630	0,0630	0,0630	0,0630	0,0630	0,0630	0,0630	0,0630	0,0630
04:03	0,0389	0,0281	0,0281	0,0282	0,0282	0,0281	0,0281	0,0281	0,0281	0,0281	0,0281
04:04	0,0311	0,0195	0,0197	0,0199	0,0199	0,0199	0,0198	0,0198	0,0198	0,0198	0,0198
05:01	0,8120	0,2021	0,2021	0,2021	0,2021	0,2021	0,2021	0,2021	0,2021	0,2021	0,2021
05:02	0,1448	0,0652	0,0636	0,0636	0,0636	0,0636	0,0636	0,0636	0,0636	0,0636	0,0636
05:03	0,0463	0,0285	0,0286	0,0285	0,0285	0,0283	0,0283	0,0283	0,0283	0,0283	0,0283
05:04	0,0238	0,0163	0,0163	0,0164	0,0164	0,0164	0,0164	0,0164	0,0164	0,0164	0,0164
05:05	0,0213	0,0136	0,0137	0,0139	0,0139	0,0141	0,0141	0,0142	0,0142	0,0142	0,0144
06:01	0,8917	0,2021	0,2021	0,2021	0,2021	0,2021	0,2021	0,2021	0,2021	0,2021	0,2021
06:02	0,2341	0,0659	0,0660	0,0661	0,0661	0,0661	0,0661	0,0661	0,0661	0,0661	0,0661
06:03	0,0703	0,0307	0,0305	0,0307	0,0307	0,0311	0,0311	0,0312	0,0314	0,0313	0,0312
06:04	0,0232	0,0154	0,0158	0,0156	0,0156	0,0158	0,0160	0,0161	0,0163	0,0166	0,0166
06:05	0,0146	0,0105	0,0107	0,0109	0,0109	0,0110	0,0112	0,0115	0,0117	0,0119	0,0121
06:06	0,0142	0,0100	0,0102	0,0105	0,0105	0,0106	0,0108	0,0110	0,0111	0,0114	0,0115

Tabela 22 – Resultado da diferença entre a distorção da iteração indicada na própria coluna e a distorção da iteração indicada na coluna imediatamente à esquerda. Para a iteração 50, essa diferença é relacionada à iteração 1

	50	100	150	200	250	300	350	400	450	500
01:01	-0,1229	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
02:01	-0,2811	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
02:02	-0,0398	**0,0000**	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
03:01	-0,5138	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
03:02	-0,0210	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
03:03	-0,0218	**0,0001**	**0,0000**	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
04:01	-0,5757	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
04:02	-0,0451	**0,0002**	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
04:03	-0,0108	**0,0000**	**0,0001**	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
04:04	-0,0117	**0,0003**	**0,0002**	0,0000	0,0000	-0,0001	0,0000	0,0000	0,0000	0,0000
05:01	-0,6098	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
05:02	-0,0795	-0,0017	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
05:03	-0,0178	**0,0001**	-0,0001	0,0000	-0,0001	0,0000	0,0000	0,0000	0,0000	0,0000
05:04	-0,0075	0,0000	**0,0001**	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
05:05	-0,0077	**0,0001**	**0,0002**	0,0000	**0,0001**	**0,0000**	**0,0000**	**0,0001**	**0,0000**	**0,0001**
06:01	-0,6896	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
06:02	-0,1682	**0,0001**	**0,0001**	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
06:03	-0,0396	-0,0003	**0,0003**	0,0000	**0,0004**	-0,0001	**0,0001**	**0,0002**	-0,0001	-0,0001
06:04	-0,0078	**0,0004**	-0,0002	0,0000	**0,0002**	**0,0002**	**0,0001**	**0,0002**	**0,0003**	**0,0000**
06:05	-0,0040	**0,0002**	**0,0001**	0,0000	**0,0002**	**0,0002**	**0,0003**	**0,0002**	**0,0001**	**0,0002**
06:06	-0,0042	**0,0003**	**0,0003**	0,0000	**0,0001**	**0,0001**	**0,0002**	**0,0001**	**0,0002**	**0,0001**

Pode ser verificado na Tabela 22 que partindo da iteração 100 em diante existe diferença apenas na quarta casa decimal. Para muitos casos, nem diferença existe. Foi gerado então o gráfico da Figura 20 onde se verifica que em torno da iteração de número 100 as melhorias de desempenho entram em um regime próximo ao permanente. No gráfico da Figura 20 são apresentadas apenas três relações Ri:Rf do FSVQ como exemplo didático, mas o mesmo comportamento foi verificado para as demais 18 relações.

Foi definida como critério de parada a quantidade de 200 iterações. Esse número de iterações foi considerado suficiente para garantir que não houvesse impacto nos resultados da pesquisa.

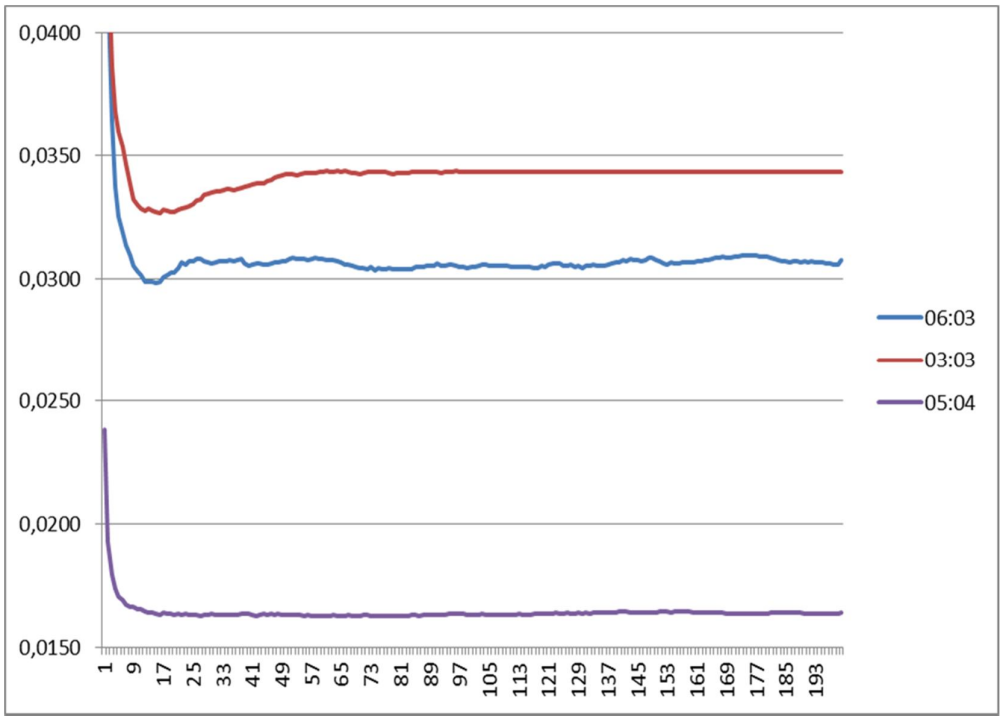


Figura 20 – Gráfico representando o número de iterações no eixo horizontal e a distorção de cada iteração no eixo vertical para a amostra de relações R_i:R_f do FSVQ 6:3, 3:3 e 5:4

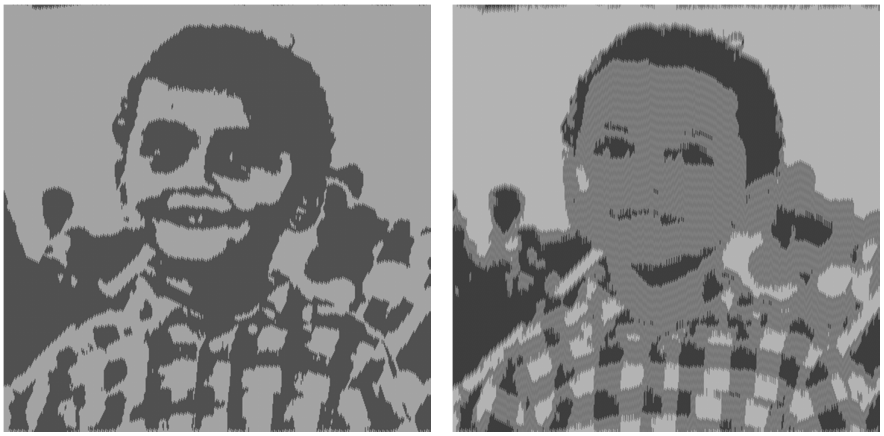
4.4. Teste Imagem: Gabriel

Figura 21 – Imagem Gabriel, 512 × 512

A imagem Gabriel foi escolhida para teste por possuir características de imagem natural de pessoas, com muitas bordas (alta frequência) e diversidade de tons de cinza, sendo um teste desafiador para o projeto do codificador/decodificador com base na imagem Lena.

A Figura 22 apresenta as imagens quantizadas (a) geradas após o VQ de R = 1 bpv e (b) geradas após o FSVQ com R = 1 bpv, projetado com base em VQ de 1 bpv

sem memória. Percebe-se visualmente o que será constatado numericamente na Tabela 23: o FSVQ, mesmo que sem redução absoluta na taxa de compressão, apresenta uma melhor representação da imagem original da Figura 21. Pode ser verificado no FSVQ da Figura 22 que existem quatro tons de cinza na imagem reconstruída, indicando a presença de dois dicionários com quatro palavras distintas, sendo duas palavras pertencentes a cada dicionário. Diferente do VQ da mesma Figura 22 onde existe apenas um dicionário com duas palavras distintas.



(a) VQ com R = 1 bpv; (b) FSVQ com R = 1 bpv, projetado com base em VQ de 1 bpv sem memória

Figura 22 – Imagens quantizadas com base em VQ de 1 bpv sem memória

As Figuras 23 a 27 apresentam o mesmo comportamento observado para a Figura 22. Conforme a taxa inicial R_i (taxa de bits por vetor do VQ sem memória que serve de base para o projeto do FSVQ) cresce, maior é a percepção visual de qualidade do FSVQ relacionada ao VQ.

O FSVQ com R = 1 bpv, projetado com base em VQ de 5 bpv ou 6 bpv, apresentam uma peculiaridade. Utilizando como exemplo a base em VQ de 5 bpv, existirão 32 estados possíveis, mas cada estado possuirá apenas dois dicionários representando os dois estados possíveis no subconjunto total de 32 estados. A tabela de estados apresentará 32 estados de origem, mas apenas dois estados possíveis de destino, o que aumenta a probabilidade de criar um subconjunto de estados preferenciais. Imaginemos que o estado oito possua na tabela de estados os destinos estado sete e estado oito. E que o estado sete possua na tabela de estados os destinos estado oito e estado sete. A quantização da imagem de teste, em algum momento do processo, ao

entrar no estado sete ou no estado oito, já que outros estados podem ter o estado sete ou estado oito como destinos na tabela de estados, utilizará recursivamente o dicionário pertencente aos dois estados, em *loop*, não estimando e representando adequadamente os vetores da imagem de origem.

A comparação visual da imagem codificada utilizando dicionários estabelecidos pelo VQ, com as imagens codificadas pelos dicionários estabelecidos pelo FSVQ, com taxa bpv final igual à do VQ sem memória que foi base para o respectivo projeto, mostra um resultado visual melhor para o FSVQ.

(a) VQ com R = 2 bpv;

(b) FSVQ com R = 1 bpv, projetado com base em VQ de 2 bpv sem memória;

(c) FSVQ com R = 2 bpv, projetado com base em VQ de 2 bpv sem memória

Figura 23 – Imagens quantizadas com base em VQ de 2 bpv sem memória

(a) VQ com R = 3 bpv;

(b) FSVQ com R = 1 bpv, projetado com base em VQ de 3 bpv sem memória;

(c) FSVQ com R = 2 bpv, projetado com base em VQ de 3 bpv sem memória;

(d) FSVQ com R = 3 bpv, projetado com base em VQ de 3 bpv sem memória

Figura 24 – Imagens quantizadas com base em VQ de 3 bpv sem memória

(a) VQ com R = 4 bpv;

(b) FSVQ com R = 1 bpv, projetado com base em VQ de 4 bpv sem memória;

(c) FSVQ com R = 2 bpv, projetado com base em VQ de 4 bpv sem memória;

(d) FSVQ com R = 3 bpv, projetado com base em VQ de 4 bpv sem memória;

(e) FSVQ com R = 4 bpv, projetado com base em VQ de 4 bpv sem memória.

Figura 25 – Imagens quantizadas com base em VQ de 4 bpv sem memória

(a) VQ com R = 5 bpv;

(b) FSVQ com R = 1 bpv, projetado com base em VQ de 5 bpv sem memória;

(c) FSVQ com R = 2 bpv, projetado com base em VQ de 5 bpv sem memória;

(d) FSVQ com R = 3 bpv, projetado com base em VQ de 5 bpv sem memória;

(e) FSVQ com R = 4 bpv, projetado com base em VQ de 5 bpv sem memória

(f) FSVQ com R = 5 bpv, projetado com base em VQ de 5 bpv sem memória.

Figura 26 – Imagens quantizadas com base em VQ de 5 bpv sem memória

(a) VQ com R = 6 bpv;

(b) FSVQ com R = 1 bpv, projetado com base em VQ de 6 bpv sem memória;

(c) FSVQ com R = 2 bpv, projetado com base em VQ de 6 bpv sem memória;

(d) FSVQ com R = 3 bpv, projetado com base em VQ de 6 bpv sem memória;

(e) FSVQ com R = 4 bpv, projetado com base em VQ de 6 bpv sem memória;

(f) FSVQ com R = 5 bpv, projetado com base em VQ de 6 bpv sem memória;

(g) FSVQ com R = 6 bpv, projetado com base em VQ de 6 bpv sem memória.

Figura 27 – Imagens quantizadas com base em VQ de 6 bpv sem memória

Para permitir uma análise numérica da imagem testada, foi criada a Tabela 23 com as medidas de distorção entre os vetores originais da imagem Gabriel e os vetores do dicionário que melhor representam os respectivos vetores. Será apresentada na Tabela 28, da seção 4.5, uma comparação entre as medidas de distorção das diversas imagens testadas nesse projeto.

Tabela 23 – Distorção medida no teste da Imagem Gabriel

VQ			**FSVQ**	**1 bpv**	**2 bpv**	**3 bpv**	**4 bpv**	**5 bpv**	**6 bpv**
1 bpv	0,0927		**1 bpv**	0,0530					
2 bpv	0,0310		**2 bpv**	0,0534	0,0173				
3 bpv	0,0174		**3 bpv**	0,0530	0,0157	0,0100			
4 bpv	0,0113		**4 bpv**	0,0535	0,0173	0,0081	0,0057		
5 bpv	0,0079		**5 bpv**	0,1692	0,0180	0,0087	0,0051	0,0048	
6 bpv	0,0052		**6 bpv**	0,1391	0,0196	0,0089	0,0053	0,0046	0,0048

Existe uma observação relacionada ao comportamento do FSVQ com R = 1 bpv, projetado com base em VQ de 5 bpv ou 6 bpv. O tamanho do dicionário de cada estado ficou bastante reduzido (tamanho dois), levando a uma concentração de resultados em "falsos" vetores ótimos, conforme explicado nessa mesma seção 4.4.

4.5. Teste Imagens: Diversas

Após detalhamento do método de teste utilizando a imagem Gabriel na seção 4.4, com o objetivo de validar o resultado para um universo maior de imagens, foi repetido o mesmo experimento para as imagens Giovana, Menina, Arara e Paisagem, mostradas na Figura 28, e também para a imagem Lena (Figura 15), que foi utilizada

previamente para treino e da qual se espera um resultado de teste naturalmente maior, e sem valor prático.

(a) (b) (c) (d)

Figura 28 – Imagens testadas: (a) Giovana; (b) Menina; (c) Arara; (d) Paisagem

A Figura 29 apresenta os resultados dos testes da imagem Giovana. As linhas dessa matriz de imagens representam a taxa bpv de cada VQ testado, e também a taxa bpv do VQ-base de cada FSVQ testado. A coluna (a) indica o resultado do teste de cada VQ, conforme bpv definido pela linha em que se encontra a imagem. As colunas de (b) a (g) indicam os resultados dos testes de cada FSVQ, conforme bpv do VQ-base, e bpv final, definidos pela linha e coluna em que a imagem se encontra.

Tabela 24 – Distorção medida no teste da Imagem Giovana

VQ			**FSVQ**	**1 bpv**	**2 bpv**	**3 bpv**	**4 bpv**	**5 bpv**	**6 bpv**
1 bpv	0,0508		**1 bpv**	0,0300					
2 bpv	0,0233		**2 bpv**	0,0300	0,0135				
3 bpv	0,0130		**3 bpv**	0,0300	0,0133	0,0083			
4 bpv	0,0071		**4 bpv**	0,0300	0,0139	0,0071	0,0051		
5 bpv	0,0054		**5 bpv**	0,0301	0,015	0,0074	0,0046	0,0043	
6 bpv	0,0035		**6 bpv**	0,0301	0,0162	0,0083	0,0047	0,0039	0,0041

Percebe-se para o teste da Imagem Giovana o mesmo comportamento visual encontrado no teste da imagem Gabriel. Porém não foi observado o mesmo comportamento do FSVQ com R = 1 bpv, projetado com base em VQ de 5 bpv ou 6 bpv da imagem Gabriel. Mesmo com o tamanho do dicionário de cada estado bastante reduzido (tamanho dois) e com uma concentração de resultados em “falsos” vetores ótimos, conforme explicado na seção 4.4, a imagem Giovana se mostrou mais adaptada aos codificadores e decodificadores projetados.

A análise numérica da imagem testada ocorreu através da Tabela 24 que possui as medidas de distorção entre o vetor original da imagem e o vetor do dicionário que melhor representa o vetor na reconstrução da imagem Giovana quantizada. Os resultados obtidos aqui são semelhantes aos resultados obtidos na Tabela 23.

(a) (b) (c) (d) (e) (f) (g)

Figura 29 – Resultado visual do teste da imagem Giovana.

(a) (b) (c) (d) (e) (f) (g)

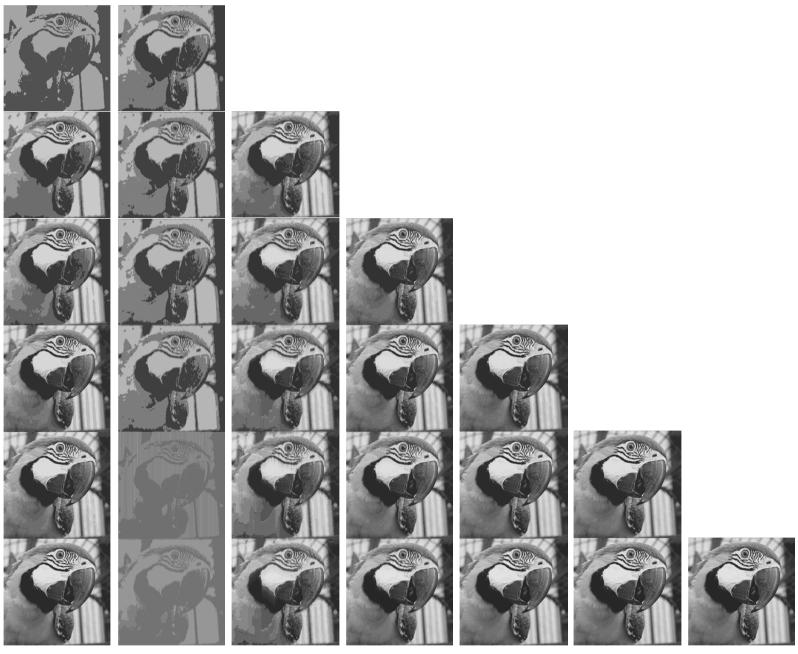
Figura 30 – Resultado visual do teste da imagem Menina

As Figuras 30, 31 e 32 apresentam os resultados dos testes com as imagens Menina, Arara e Paisagem. As imagens estão dispostas da mesma maneira que foram apresentadas anteriormente as imagens Giovana (Figura 29).

Tabela 25 – Distorção medida no teste da imagem Menina

VQ			**FSVQ**	**1 bpv**	**2 bpv**	**3 bpv**	**4 bpv**	**5 bpv**	**6 bpv**
1 bpv	0,0653		**1 bpv**	0,0407					
2 bpv	0,0250		**2 bpv**	0,0408	0,0157				
3 bpv	0,0169		**3 bpv**	0,0410	0,0153	0,0099			
4 bpv	0,0093		**4 bpv**	0,0408	0,0152	0,0087	0,0064		
5 bpv	0,0066		**5 bpv**	0,1797	0,0154	0,0089	0,0057	0,0053	
6 bpv	0,0047		**6 bpv**	0,1516	0,0163	0,0093	0,0058	0,0048	0,0049

As análises numéricas das imagens Menina, Arara e Paisagem ocorreram através das Tabelas 25, 26 e 27. Os resultados apresentados nessas Tabelas são semelhantes aos resultados obtidos nas Tabelas 23 e 24. O mesmo comportamento relacionado ao FSVQ com R = 1 bpv, projetado com base em VQ de 5 bpv ou 6 bpv pode ser observado. E os resultados apresentados na Tabela 27 apresentam piora na distorção para todos os VQs e FSVQs testados.



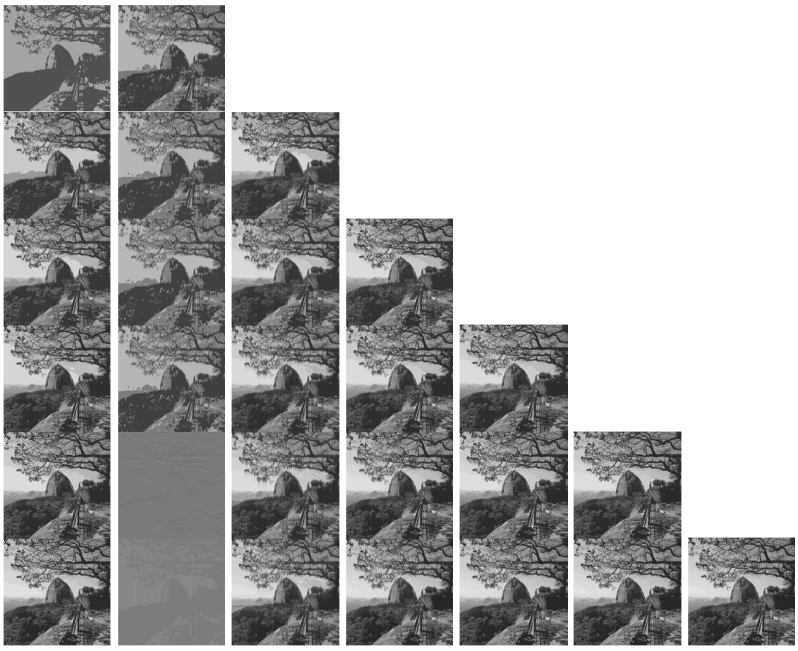
(a) (b) (c) (d) (e) (f) (g)

Figura 31 – Resultado visual do teste da imagem Arara

Tabela 26 - Distorção medida no teste da imagem Arara

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	0,0992		1 bpv	0,0635					
2 bpv	0,0359		2 bpv	0,0635	0,0275				
3 bpv	0,0251		3 bpv	0,0634	0,0275	0,0169			
4 bpv	0,0157		4 bpv	0,0636	0,0277	0,0156	0,0111		
5 bpv	0,0117		5 bpv	0,2865	0,0283	0,0166	0,0113	0,0100	
6 bpv	0,0088		6 bpv	0,2248	0,0290	0,0173	0,0117	0,0102	0,0096

O aumento da distorção constatado para os testes com a imagem Paisagem está relacionado diretamente à diferença de características entre a imagem Lena, utilizada para treino, e a imagem Paisagem. Provavelmente, o treino em uma imagem com características de paisagem apresente melhores resultados no teste da imagem Paisagem. Mas essa análise não faz parte do escopo desse trabalho.



(a) **(b)** **(c)** **(d)** **(e)** **(f)** **(g)**

Figura 32 – Resultado visual do teste da imagem Paisagem

Tabela 27 - Distorção medida no teste da imagem Paisagem

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	0,1500		1 bpv	0,1258					
2 bpv	0,0922		2 bpv	0,1258	0,0858				
3 bpv	0,0718		3 bpv	0,1258	0,0838	0,0670			
4 bpv	0,0562		4 bpv	0,1258	0,0826	0,0666	0,055		
5 bpv	0,0485		5 bpv	0,3229	0,0837	0,0666	0,0543	0,0499	
6 bpv	0,0413		6 bpv	0,3408	0,0848	0,0670	0,0551	0,0489	0,0479

A Tabela 28 apresenta uma síntese comparativa entre os resultados encontrados para cada imagem testada. Os resultados estão expressos em dB para facilitar a análise. Como a imagem Lena foi imagem utilizada para treino, tomaremos o teste com a imagem Lena como uma referência de desempenho irrealisticamente bom, que não ocorreria na prática. Também utilizaremos para exemplificar os resultados o desempenho do FSVQ 3:3 por ser uma das melhores escolhas na análise armazenamento-custo.

Tabela 28 – Distorção (dB) medida nos testes de imagem

VQ			FSVQ	1 bpv	2 bpv	3 bpv	4 bpv	5 bpv	6 bpv
1 bpv	12,0761		**1 bpv**	14,1341					**Lena**
	10,3292			12,7572					**Gabriel**
	12,9414			15,2288					**Giovana**
	11,8509			13,9041					**Menina**
	10,0349			11,9723					**Arara**
	8,2391			9,0032					**Paisagem**
2 bpv	17,0553		**2 bpv**	14,1341	19,1721				
	15,0864			12,7246	17,6195				
	16,3264			15,2288	18,6967				
	16,0206			13,8934	18,0410				
	14,4491			11,9723	15,6067				
	10,3527			9,0032	10,6651				
3 bpv	19,7062		**3 bpv**	14,1341	19,4310	21,8046			
	17,5945			12,7572	18,0410	20,0000			
	18,8606			15,2288	18,7615	20,8092			
	17,7211			13,8722	18,1531	20,0436			
	16,0033			11,9791	15,6067	17,7211			
	11,4388			9,0032	10,7676	11,7393			
4 bpv	22,2915		**4 bpv**	14,1341	19,2082	22,6761	24,2022		
	19,4692			12,7165	17,6195	20,9151	22,4413		
	21,4874			15,2288	18,5699	21,4874	22,9243		
	20,3152			13,8934	18,1816	20,6048	21,9382		
	18,0410			11,9654	15,5752	18,0688	19,5468		
	12,5026			9,0032	10,8302	11,7653	12,5964		
5 bpv	23,8722		**5 bpv**	14,1341	19,1721	22,6761	25,0864	25,0864	
	21,0237			7,7160	17,4473	20,6048	22,9243	23,1876	
	22,6761			15,2143	18,2391	21,3077	23,3724	23,6653	
	21,8046			7,4545	18,1248	20,5061	22,4413	22,7572	
	19,3181			5,4288	15,4821	17,7989	19,4692	20,0000	
	13,1426			4,9093	10,7727	11,7653	12,6520	13,0190	
6 bpv	25,6864		**6 bpv**	14,1341	18,9963	22,3657	25,2288	26,1979	25,6864
	22,8400			8,5667	17,0774	20,5061	22,7572	23,3724	23,1876
	24,5593			15,2143	17,9048	20,8092	23,2790	24,0894	23,8722
	23,2790			8,1930	17,8781	20,3152	22,3657	23,1876	23,0980
	20,5552			6,4820	15,3760	17,6195	19,3181	19,9140	20,1773
	13,8405			4,6750	10,7160	11,7393	12,5885	13,1069	13,1966

A teste da imagem Lena apresentou um nível de 21.8 dB. As imagens de pessoas testadas (Gabriel, Giovana e Menina), apresentaram níveis 20.0 dB, 20.8 dB e 20.0 dB, respectivamente, resultando em uma média de 20.3 dB por imagem.

A imagem Arara é uma imagem natural de um animal e tem características visuais semelhantes às das imagens de pessoas, mas ainda assim as suas estatísticas são consideravelmente diferentes das estatísticas da fonte aleatória pela qual pode ser modelada a geração do retrato de uma pessoa. Por isso, a qualidade da reconstrução (17.7 dB) é mais baixa no teste. Já a imagem Paisagem, apresenta distribuição totalmente diversa da encontrada em imagens de pessoas. Por esse motivo, os resultados foram bastante ruins no teste: 11.7 dB.

Em resumo, pode-se concluir que os testes nos sistemas FSVQ projetados apresentam consistentemente um desempenho melhor do que aquele observado no VQ-base de mesma quantidade de bits por vetor. Para a obtenção deste desempenho melhorado, perde-se em termos de custo de armazenamento (posições de memória).

5. CONCLUSÃO

Foi constatado nos resultados dos testes que é possível a otimização de um codificador com eficiência medida em termos de desempenho taxa-distorção e custo de armazenamento através da técnica de VQ com memória, que no caso desse trabalho é conhecida como FSVQ.

Para se medir a qualidade e capacidade do codificador, foram utilizadas medidas específicas para possibilitar a escolha de bons codificadores: taxa-distorção, entropia-distorção e armazenamento-custo.

As medidas de taxa-distorção demonstram que a utilização do FSVQ é muito mais interessante que a utilização do VQ. Essa constatação pode ser verificada na seção 4.5, Tabela 28. Tomando como exemplo o FSVQ com R = 3 bpv, projetado com base em VQ de 3 bpv, a reconstrução da imagem Gabriel atingiu 17.6 dB e 20.0 dB, utilizando as técnicas de VQ e FSVQ, respectivamente.

A análise da medida armazenamento-custo (onde custo é desempenho taxa-distorção) demonstra que a utilização do VQ pode até ser menos complexa que a utilização do FSVQ, mas com menor eficiência das medidas taxa-distorção. Novamente podemos recorrer à seção 4.5, Tabela 28. Utilizando o mesmo exemplo do parágrafo anterior: FSVQ com R = 3 bpv, projetado com base em VQ de 3 bpv. E utilizaremos as Tabelas 19 e 20 da Seção 4.2, onde podemos constatar armazenamento de 40 e 329 posições de memória, e custo J de 0.1790 e 0.0696 de distorção MSE para VQ e FSVQ, respectivamente.

Analisando apenas a técnica FSVQ na medida armazenamento-custo, verifica-se no projeto uma casca convexa inferior no conjunto de pontos mostrados na Figura 13 que inclui FSVQs com taxas de bit do VQ-base e taxas de bit finais dadas por 1:1, 2:2, 3:3, 4:4, 5:5 e 6:6 bpv, sendo essas as melhores taxas de bits por vetor obtidas.

É possível perceber, através da Figura 19, que a medida de armazenamento será sempre menor para o VQ-base quando comparado com os FSVQs de mesma taxa de bit por vetor final. Porém, é constatado também que a medida de custo-J será sempre menor para os FSVQs quando confrontado com o VQ-base na mesma base de comparação.

Outro ponto a salientar é que os dicionários localmente ótimos permitiram a reconstrução de imagens de pessoas com perda média de 1.5 dB, no caso do FSVQ 3:3, mas houve problemas com a reconstrução de imagens de paisagens. Para o teste da

imagem Arara a perda foi de 4.1 dB e para o teste da imagem Paisagem, a perda foi de 10.1 dB.

Pontos que podem ser trabalhados para dar continuidade a essa pesquisa são (a) a otimização dos dicionários para o FSVQ com R = 1 bpv, projetado com base em VQ de 5 bpv ou 6 bpv, conforme problema exposto na seção 4.4, (b) a criação de dicionários localmente ótimos para cada tipo de imagem natural: pessoa, paisagem e animais, e (c) estudar o comportamento da função custo J através da adição de outras técnicas de compressão como Redes Neurais e Transformada Direta de Cosseno.

APÊNDICE A

A.1 – Código-Fonte Principal

```
%% Geração dos vetores 2-D e 5-D Pseudo-Aleatórios e Correlacionados
N = 81920; X = zeros(1,N); randn('state',0);
X(1) = randn(1,1);
for k = 2:N,
   X(k) = 0.95*X(k-1) + 0.05*randn(1,1);
end;
Y = reshape(X,5,N/5); % or Y = reshape(X,2,N/2);
X = Y; clear Y;

%% Geração dos vetores 5-D da imagem Lena para projeto
X = imread('lena512.bmp');
X = double(X); VOLTA = max(max(X)); TAMANHO = size(X,2); X = X/VOLTA;
Y = reshape(X,[],1)';
Y = Y(1:262140);  % Usado no Vetor 5D
X = reshape(Y,5,size(Y,2)/5); TAMANHO1 = size(X,1); TAMANHO2 =
size(X,2);

%% VQ – GLA Básico
NB = 3
randn('state',0); LM = 0;
J0 = mean(sum((X - repmat(mean(X,2),1,size(X,2))).^2)); BKJ = [];
for u=1:length(LM);
    lambda = LM(u);
    M = size(X,1); N = size(X,2); K = 2^NB; randn('state',0);
    Y = repmat(mean(X,2),1,K) + 0.05*randn(M,K);
    l = log2(K)*ones(1,size(Y,2));
    F = 200; BK = zeros(F,4);
    for i=1:F-1,
        [Y,p] = XYltoYp_MF(X,Y,l,lambda);
        Y = Y(:,find(p~=0)); p = p(find(p~=0)); p = p/sum(p);
        l = -log2(p);
    end;
    [Y,p,D,k] = XYltoYpDk(X,Y,l,lambda);
    p = p(find(p~=0)); p = p/sum(p);
    H = -sum(p.*log2(p));
    BKJ = [BKJ ; [lambda D H D+lambda*H size(Y,2)]];
end;
```

```
D0 = D/J0;

%% FSVQ com NB2 bpv projetado com base em VQ de NB bpv
NB2 = 3
P = zeros(size(Y,2),size(Y,2));
for i=1:size(Y,2),
    I = k(find(k(1:(length(k)-1)) == i) + 1);
    for j=1:size(Y,2),
        P(i,j) = sum(I == j);
    end;
    P(i,:) = P(i,:)/sum(P(i,:));
end;
P = [P' (1:size(Y,2))'];
clear p; clear f;
for i=1:size(Y,2),
    P = flipud(sortrows(P,i));
    if size(Y,2) < 2.^NB2
        f(i,1:size(Y,2)) = P(1:size(Y,2),size(Y,2)+1)';
        p(i,1:size(Y,2)) = P(1:size(Y,2),i)';
    else
        f(i,1:2^NB2) = P(1:2^NB2,size(Y,2)+1)';
        p(i,1:2^NB2) = P(1:2^NB2,i)';
    end
end;
u = zeros(1,size(X,2)+1);
[a,b,c,u(1)] = XYltoYpDk(X(:,1),Y,l,lambda);
for i=1:size(Y,2),
    Ys(i).Y = Y(:,f(i,:));
    Yf(i).Y = zeros(size(Ys(i).Y));
end;
c = zeros(size(f)); m = 1; clear D; clear s;
MELHORA  = 0.006;
while MELHORA > 0.0005
    D(m) = 0;
    for n=1:(size(X,2)-1),
        s = u(n);
        [a,b,d,j] =
XYltoYpDk(X(:,n+1),Ys(s).Y,zeros(1,size(Ys(s).Y,2)),lambda);
        u(n+1) = f(s,j);
        D(m) = D(m)+d;
        Yf(s).Y(:,j) = Yf(s).Y(:,j) + X(:,n+1); c(s,j) = c(s,j) + 1;
```

```
    end;
    D(m) = D(m)/(N*J0); D1(NB).D(NB2,m) = D(m);
    if m == 1,
        MELHORA = 0.0006;
    else
        MELHORA = D1(NB).D(NB2,m-1) - D1(NB).D(NB2,m);
    end;
    if MELHORA > 0.0005;
        for i=1:size(Y,2),
            for j=1:size(Ys(i).Y,2),
                Ys(i).Y(:,j) = Yf(i).Y(:,j)/c(i,j);
            end
            Yf(i).Y = zeros(size(Yf(i).Y));
            c(i,:) = zeros(size(c(i,:)));
        end
    else
        D1(NB).D(NB2,m) = D1(NB).D(NB2,m-1);
        Yf(i).Y = zeros(size(Yf(i).Y));
        c(i,:) = zeros(size(c(i,:)));
    end
        m = m+1;
end
while m < 101
    D1(NB).D(NB2,m) = D1(NB).D(NB2,m-1);
    m = m+1;
end

%% Cálculo de Armazenamento
Estados = 2.^NB; Palavras = 2.^NB2; DimensaoVetor = size(X,1);
Theta = Palavras * DimensaoVetor;
A(NB,NB2) = (Estados * Theta) + (NB*NB2);
if NB2 == NB,
    Palavras = 1;
    Theta = Palavras * DimensaoVetor;
    A0(NB) = Estados * Theta;
end

%% Cálculo de Entropia
j = u(1);
clear Euclid; clear EuclidOrdem; clear InfoOutput;
for g = 1:size(X,2),
```

```
    Antes = j;
    for i = 1:size(Ys(j).Y,2),
        Euclid(:,i) = (X(:,g) - Ys(j).Y(:,i)).^2;
    end
    EuclidOrdem = sum(Euclid,1);
    Linha = size(EuclidOrdem,1);
    for i = 1:size(EuclidOrdem,2),
        EuclidOrdem(Linha+1,i) = i;
    end
    EuclidOrdem = EuclidOrdem';
    EuclidOrdem = flipud(sortrows(EuclidOrdem,1));
    j = f(Antes,EuclidOrdem(size(EuclidOrdem,1),2));
    InfoOutput(1,g) = Antes;
    InfoOutput(2,g) = j;
    if g == 1,
        InfoOutput(3,g) = u(1);
    else
        InfoOutput(3,g) = InfoOutput(4,g-1);
    end
    InfoOutput(4,g) = EuclidOrdem(size(EuclidOrdem,1),2);
end
[HVQ,HFSVQ,r] = Entropia(NB,NB2,InfoOutput,BKJ);

%% [HVQ,HFSVQ,r] = Entropia(NB,NB2,InfoOutput,BKJ);
function [H_VQ,H_FSVQ,r] = Entropia(NB,NB2,InfoOutput,BKJ);
for i = 1:2.^NB,
    for j = 1:2.^NB2
        Dados(i,j) = sum((InfoOutput(1,:)==i)&(InfoOutput(4,:)==j));
    end
    Dados(i,j+1) = sum(Dados(i,:));
    if Dados(i,j+1) == 0,
        Dados(i,j+1) = 0.000000001;
    end
end
for i = 1:2.^NB,
    for j = 1:2.^NB2
        DadosProb(i,j) = Dados(i,j)/Dados(i,2.^NB2+1);
        if DadosProb(i,j) == 0,
            DadosProb(i,j) = 0.000000001;
        end
    end
```

```
end
H_VQ = BKJ(1,3);
r = -sum(DadosProb.*log2(DadosProb),2);
H_FSVQ = (sum(Dados(:,2.^NB2+1).*r))/(sum(Dados(:,2.^NB2+1)));
clear Dados; clear DadosProb;

%% [Y,p] = XYltoYp_MF(X,Y,l,lambda);
% Condição de Partição e Condição de Centróide - 1
function [Yout,p] = XYltoYp_MF(X,Yin,l,lambda);
```

Função desenvolvida por Gomes [4].

```
%% [Y,p,D,k] = XYltoYpDk(X,Y,l,lambda);
% Condição de Partição e Condição de Centróide - 2
function [Yout,p,D,k] = XYltoYpDk(X,Yin,l,lambda);
```

Função desenvolvida por Gomes [4].

```
%% [XcVQ,DVQ1] = TesteCompressaoVQ(NB,Y);
% Teste de imagem com VQ
function [Xc,DVQ] = TesteCompressaoVQ(NB,Y);
% Xt = imread('paisagem512.bmp');
% Xt = imread('arara512.bmp');
% Xt = imread('menina512.bmp');
% Xt = imread('Giovana512.bmp');
% Xt = imread('lena512.bmp');
Xt = imread('Gabriel512.bmp');
Xt = double(Xt); VOLTAt = max(max(Xt)); TAMANHOt = size(Xt,2);
Xt = Xt/VOLTAt; Yt = reshape(Xt,[],1)'; Yt = Yt(1:262140);
Xt = reshape(Yt,5,size(Yt,2)/5);
TAMANHO1t = size(Xt,1); TAMANHO2t = size(Xt,2);
DVQ = 0; k1 = zeros(1,size(Xt,2));
for n=1:size(Xt,2),
    j = sum((repmat(Xt(:,n),1,size(Y,2)) - Y).^2,1);
    k1(n) = min(find(j==min(j)));
    DVQ = DVQ + sum((Xt(:,n)-Y(:,k1(n))).^2);
    Xc(:,n) = Y(:,k1(n));
end
DVQ = DVQ/size(Xt,2);
Xc = reshape(Xc,[],1)';
Xc(:,(size(Xc,2)+1):(size(Xc,2)+4)) = Xc(:,size(Xc,2));
Xc = Xc * VOLTAt;
Xc = reshape(Xc,TAMANHOt,TAMANHOt);
```

```
imwrite(uint8(Xc),'C:\TESE\TESTE_VQ.bmp','bmp');


%% [XcFSVQ2,D_FSVQ] = TesteCompressaoFSVQ(NB,NB2,f,Y,Ys);
% Teste de imagem com FSVQ
function [Xc,DFSVQ] = TesteCompressaoFSVQ(NB,NB2,f,Y,Ys);
% Xt = imread('paisagem512.bmp');
% Xt = imread('parrots512.bmp');
% Xt = imread('menina512.bmp');
% Xt = imread('Giovana512.bmp');
% Xt = imread('lena512.bmp');
Xt = imread('Gabriel512.bmp');
Xt = double(Xt); VOLTAt = max(max(Xt)); TAMANHOt = size(Xt,2);
Xt = Xt/VOLTAt; Yt = reshape(Xt,[],1)'; Yt = Yt(1:262140);
Xt = reshape(Yt,5,size(Yt,2)/5);
TAMANHO1t = size(Xt,1); TAMANHO2t = size(Xt,2);
DFSVQ = 0; k1 = zeros(1,size(Xt,2));
j = sum((repmat(Xt(:,1),1,size(Y,2)) - Y).^2,1);
k1(1) = min(find(j==min(j)));
DFSVQ = DFSVQ + sum((Xt(:,1)-Y(:,k1(1))).^2);
Xc(:,1) = Y(:,k1(1));
for i = 1:size(Ys,2),
    for j = 1:size(Ys(i).Y,2)
        if isnan(Ys(i).Y(:,j)),
            Ys(i).Y(:,j) = 0.5 + 0.05*randn(1,1);
        end
    end
end
for n=1:size(Xt,2)-1,
    j = sum((repmat(Xt(:,n+1),1,size((Ys(k1(n)).Y),2)) -
(Ys(k1(n)).Y)).^2,1);
    k1(n+1) = min(find(j==min(j)));
    DFSVQ = DFSVQ + sum((Xt(:,n+1)-Ys(k1(n)).Y(:,k1(n+1))).^2);
    Xc(:,n+1) = Ys(k1(n)).Y(:,k1(n+1));
    k1(n+1) = f(k1(n),k1(n+1));
end
DFSVQ = DFSVQ/size(Xt,2);
Xc = reshape(Xc,[],1)';
Xc(:,(size(Xc,2)+1):(size(Xc,2)+4)) = Xc(:,size(Xc,2));
Xc = Xc * VOLTAt;
Xc = reshape(Xc,TAMANHOt,TAMANHOt);
```

```
imwrite(uint8(Xc),'C:\TESE\TESTE_FSVQ.bmp','bmp');


%% Gráficos
Cores = {'xr--' '+g--' '*b--' 'dc--' '^m--' 'pk--' 'oy--'};
% Gráfico taxa-distorção
figure;
for i = 1:NB
    for j = 1:i
        plot (j,D1(i).D(j,size(D1(NB).D,2)),cell2mat(Cores(i)));
    end
    hold on;
    plot (i,Vetor_D0(i),cell2mat(Cores(NB+1)));
    xlabel('R (FSVQ Final)'); ylabel('MSE - D'); title('Taxa de bits
por vetor');
end
% Gráfico taxa-distorção (dB)
figure;
for i = 1:NB
    for j = 1:i
        plot (j,-
10*log10(D1(i).D(j,size(D1(NB).D,2))),cell2mat(Cores(i)));
    end
    hold on;
    plot (i,-10*log10(Vetor_D0(i)),cell2mat(Cores(NB+1)));
    xlabel('R (FSVQ Final)'); ylabel('MSE - D (-10*log10(D))');
title('Taxa de bits por vetor');
end
% Gráfico entropia-distorção
figure;
for i = 1:NB
    for j = 1:i
        plot (H_FSVQ(i,j),Dc(i,j),cell2mat(Cores(i)));
    end
    hold on;
    plot (H_VQ(1,i),D0(i,1),cell2mat(Cores(NB+1)));
    xlabel('Entropia - H (Destino)'); ylabel('MSE - D');
title('Entropia x Distorção');
end
% Gráfico armazenamento-custo (J)
figure
```

```
for i = 1:NB
    for j = 1:i
        plot (log10(A(i,j)),Custo_J(i,j),cell2mat(Cores(i)));
        hold on
        text (log10(A(i,j))+0.1,Custo_J(i,j),cellstr([num2str(i) ':'
num2str(j)]));
    end
    hold on;
    plot (log10(A0(1,i)),Custo_J0(i,1),cell2mat(Cores(NB+1)));
    text (log10(A0(1,i))+0.1,Custo_J0(i,1),cellstr([num2str(i)]));
    xlabel('Armazenamento - A  (Log10(A))'); ylabel('Custo - J   (D +
Lambda * H)'); title('Armazenamento x Custo');
end
```

A.2 – Modos de Utilização do Código-Fonte Principal

O código foi dividido em blocos. Cada bloco se inicia com os caracteres “%%” seguidos do texto com uma espécie de título. Os blocos podem ser utilizados de maneira independente, mas necessitam de encadeamento lógico para que as variáveis necessárias em um bloco sejam devidamente geradas em bloco anterior.

No bloco de projeto do VQ, NB significa a taxa de bits por vetor do VQ-base. O mesmo ocorre no bloco de projeto do FSVQ, onde NB2 significa a redução da taxa de bits por vetor referente ao VQ-base correspondente.

No bloco de testes, existem algumas imagens comentadas com “%”, e que podem ser alternadamente escolhidas para teste de cada imagem. Até mesmo modificado para inserção de novas imagens de teste.

No bloco dos gráficos, algumas pequenas modificações podem e devem ser feitas quando forem utilizadas para projeto imagens de dimensões diferentes, tipos diferentes, que podem modificar a escala e comportamento da distribuição de pontos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] A. Gersho and R.M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Press, 1992.

[2] R.M. Gray. Vector Quantization. IEEE Acoustics, Speech, and Signal Processing Magazine, 1:4–29, April 1984.

[3] P. C. Cosman, E. A. Riskin, K. L. Oehler, and R. M. Gray, “Using Vector Quantization for Image Processing,” Proceedings of the IEEE, vol 81, no 9, pp. 1326–1341, September 1993.

[4] J. G. R. C. Gomes, CPE786 - Processamento de Imagens no Plano Focal, Notas de Aula.

[5] H. L. Haas, J. G. R. C. Gomes e A. Petraglia. Analog hardware implementation of a vector quantizer for focal-plane image compression. Proceedings of the 21st Symposium on Integrated Circuits and Systems Design, Gramado. Artigo aceito para publicação em maio de 2008.

[6] P. Swaszek. Vector Quantization. In I.F. Blake and H.V. Poor, editors, Communications and Networks: A Survey of Recent Advances, pages 362–389. Springer-Verlag, 1986.

[7] Tufts, D. W. and Li, Q., “Principal feature classification,” in Neural Networks for Signal Processing V, Proceedings of the 1995 IEEE Workshop, (Cambridge MA), August 1995.

[8] N.M. Nasrabadi and R.A. King. Image Coding Using Vector Quantization: A Review. IEEE Transactions on Communications, August 1988.

[9] A. Gersho and V. Cuperman. A Pattern Matching Technique for Speech Coding. IEEE Communications Magazine, pages 15–21, December 1983.

[10] J. Gomes, L. Velho, "Computação Gráfica: Imagem", Rio de Janeiro, IMPA/SBM, 1994.

[11] P. A. Chou, T. Lookabaugh, and R. M. Gray, “Entropy constrained vector quantization,” IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP–37, pp. 31–42, January 1989.

[12] C. E. Shannon, "A Mathematical Theory of Communication", The Bell System Technical Journal, v. 27, p. 379-423, 623-656, July, October, 1948.

[13] R. C. Gonzalez, R. E. Woods, "Digital Image Processing", Addison-Wesley Publishing Company, 1992.

[14] Sayood, K. (2000). Introduction to Data Compression, Second Edition. San Mateo, CA: Morgan Kauffman/Academic Press

[15] Memon, N. D., and Sayood, K. (1995). Lossless Image Compression: A Comparitive Study. In Proceedings SPIE Conference on Electronic Imaging. SPIE.

[16] Huffman, D. A. (1951). A method for the construction of minimum redundancy codes. Proc. IRE. 40:1098–1101.

[17] Wu, X., and Memon, N. D. (May 1996). CALIC—A context based adaptive lossless image coding scheme. IEEE Trans. Communications.

[18] Weinberger, M., Seroussi, G., and Sapiro, G. (November 1998). The LOCO-I Lossless Compression Algorithm: Principles and Standardization into JPEG-LS. Technical Report HPL-98-193, Hewlett-Packard Laboratory.

[19] S. Haykin, Neural Networks and Learning Machines, third ed., Pearson, Upper Saddle River, NJ, 2009

[20] Y. Linde, A. Buzo e R. M. Gray. An algorithm for Vector Quantizer design. IEEE Transactions on Communications, COM-28(1):84-95, janeiro de 1980.

[21] Lloyd, S. P., “Least Squares Quantization in PCM’s,” Bell Telephone Laboratories Paper, Murray Hill, NJ, 1957.

[22] H. Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. Operations Research, 11:399– 417, 1963.

[23] G. Cheung and A. Zakhor. Bit allocation for joint source/channel coding of scalable video. IEEE Trans. Image Processing, March 2000.