RICARDO NOÉ BRETIN DE MELLO

ESTUDO COMPARATIVO DA TRANSFORMADA KARHUNEN-LOÈVE NA COMPRESSÃO DE IMAGENS

Porto Alegre

2003

RICARDO NOÉ BRETIN DE MELLO

ESTUDO COMPARATIVO DA TRANSFORMADA KARHUNEN-LOÈVE NA COMPRESSÃO DE IMAGENS

ORIENTADOR: Prof. Dr. Altamiro Amadeu Susin

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), da Universidade Federal do Rio Grande do Sul (UFRGS), como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: Automação e Instrumentação Eletro-Eletrônica

Porto Alegre

2003

RICARDO NOÉ BRETIN DE MELLO

ESTUDO COMPARATIVO DA TRANSFORMADA KARHUNEN-LOÈVE NA COMPRESSÃO DE IMAGENS

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Altamiro Amadeu Susin, UFRGS

Doutor INPG - Universidade de Grenoble – França

Banca Examinadora:

Prof. Dr. Flávio Bortolozzi,
PUCPR – Curitiba PR - Brasil

Prof. Dr. Renato Machado de Brito,
UFRGS – Porto Alegre RS - Brasil

Prof. Dr. Luigi Carro,
UFRGS – Porto Alegre RS - Brasil

Coordenador do PPGEE: _____

Prof. Dr. Carlos Eduardo Pereira

UFRGS – Porto Alegre RS - Brasil

Porto Alegre, 11 de junho de 2003.

DEDICATÓRIA

Dedico este trabalho aos meus pais, a minha esposa e aos meus filhos, em especial pela dedicação e apoio sem limites em todos os momentos difíceis.

AGRADECIMENTOS

Ao Programa de Pós-Graduação em Engenharia Elétrica, PPGEE, pela oportunidade de realização do trabalho em minha área de pesquisa.

Ao Professor Dr. Renato Machado de Brito, pelo apoio e confiança em mim depositados.

Ao Professor Dr. Altamiro Amadeu Susin, pela orientação, confiança, incentivo, paciência, apoio e dedicação permanentes.

A todos os professores do PPGEE pelo conhecimento e orientações ministrados.

Aos colegas pelo companheirismo e apoio em todos os momentos.

À minha chefia na Telefônica Celular RS pelo apoio concedido.

Aos meus pais pelo apoio e incentivo permanentes.

À minha família pelo apoio, incentivo, paciência e compreensão sem limites.

RESUMO

A representação de funções através da utilização de bases (KERNEL) de representação tem sido fundamental no processamento digital de sinais. A Transformada KARHUNEN-LOÈVE (KLT), também conhecida como Transformada HOTELLING, permite a representação de funções utilizando funções-base formadas pelos autovetores da matriz de correlação do sinal considerado. Nesse aspecto essa transformada fornece uma base ótima, isto é, aquela que proporciona o menor valor de Erro Quadrático Médio entre o sinal reconstruído e o original, para um determinado número de coeficientes. A dificuldade na utilização da KLT está no tempo adicional para calcular os autovetores (base) da matriz de correlação, o que muitas vezes inviabiliza a sua utilização nas aplicações em tempo real. Em muitas aplicações a KLT é utilizada em conjunto com outras transformadas melhorando os resultados destas aplicações.

Sendo considerada a transformada ótima no sentido do Erro Quadrático Médio, este trabalho apresenta um estudo da Transformada KARHUNEN-LOÈVE nas aplicações de compressão de imagens bidimensionais estáticas e em tons de cinza, realizando também a comparação desta técnica com outras técnicas (DCT e WAVELET) buscando avaliar os pontos fortes e fracos da utilização da KLT para este tipo de aplicação. Duas técnicas importantes para solucionar o problema de cálculo dos autovalores e autovetores da matriz de correlação (Método de JACOBI e Método QL) são também apresentadas neste trabalho. Os resultados são comparados utilizando a Razão Sinal/Ruído de Pico (PSNR), a Razão de Compressão (CR) e os tempos de processamento (em segundos) para geração dos arquivos compactados.

Palavras-chaves: Engenharia Elétrica, Processamento de Sinais, Transformada Karhunen-Loève, Compressão de Imagens.

ABSTRACT

The function representation using basis (KERNEL) of representation has been crucial in digital signal processing. The KARHUNEN-LOÈVE (KLT) Transform, or HOTELLING Transform, as it has been also known, allows the representation of functions using basis functions formed by the eigenvectors of the correlation matrix of the signal being used. In this way, this transform gives an optimal basis as it minimizes the value of the Mean Square Error between the reconstructed and the original signals for a given number of coefficients. The difficulty in using the KLT is the additional time to compute the eigenvectors (base) of the correlation matrix, that prevent its use in real time applications. In many applications the KLT is used together with other transforms leading to a better results for these applications.

As considered the best transform in the sense of Mean Square Error, this work studies the KARHUNEN-LOÈVE Transform in the compression of 2-D grayscale still images, also comparing with other techniques (DCT and WAVELET) leading to a better evaluation of the advantages and disadvantages of using the KLT in this type of application. Two important techniques for solving the problem of finding the eigenvalues and eigenvectors of the correlation matrix (JACOBI Method and QL Method) are also presented in this work. The results are compared using the Peak Signal-to-Noise Ratio (PSNR), the Compression Ratio (CR) and the processing time (in seconds) to generate the reduced files.

Keywords: Electrical Engineering, Signal Processing, Karhunen-Loève Transform, Image Compression.

SUMÁRIO

1 INTRODUÇÃO 17

2 REVISÃO DE LITERATURA 21

3 FUNDAMENTOS TEÓRICOS 24

3.1 O SISTEMA DE COMPRESSÃO DE IMAGENS 24

3.2 TRANSFORMADA KARHUNEN-LOÈVE 28

3.2.1 ABORDAGEM MATEMÁTICA DA KLT 28

3.2.1.1 Interpretação Matemática e Gráfica da KLT 29

3.2.1.2 Estratégia de Cálculo dos Autovetores e Autovalores 39

3.2.1.3 O Método de JACOBI 40

3.2.1.4 O Método HOUSEHOLDER 43

3.2.1.5 O Método QL 45

3.2.2 APLICAÇÕES GERAIS DA KLT 46

3.2.3 APLICAÇÕES DA KLT NA COMPRESSÃO DE IMAGENS 48

3.3 O ALGORITMO DE HUFFMAN 60

3.4 A TRANSFORMADA DISCRETA DO COSENO (DCT) 63

3.5 A TRANSFORMADA WAVELET DISCRETA (DWT) 65

4 MATERIAIS E MÉTODOS 70

4.1 CONDIÇÕES EXPERIMENTAIS 71

4.2 DETALHES DE IMPLEMENTAÇÃO 72

4.2.1 PARÂMETROS E CONFIGURAÇÕES DO PROGRAMA 72

4.2.1.1 Tamanho das Variáveis 72

4.2.1.2 Alinhamento das Variáveis de Estruturas e Uniões 72

4.2.1.3 Valores dos PIXELS 73

4.2.1.4 Medida do Tempo de Processamento 73

4.2.1.5 Quantização dos Coeficientes da KLT 74

4.2.1.6 Teste do Programa 74

4.2.2 DETALHES DOS ARQUIVOS DE IMAGEM 75

4.2.2.1 Características da Imagem BITMAP 75

4.2.2.2 Alinhamento dos Arquivos BITMAP 76

4.2.2.3 Visualização das Imagens Processadas com vários PSNR 76

5 RESULTADOS 81

5.1 RESULTADOS OBTIDOS COM A IMAGEM LENNA 83
5.2 RESULTADOS OBTIDOS COM A IMAGEM BABUÍNO 99
5.3 RESULTADOS OBTIDOS COM A IMAGEM CASA 115
5.4 RESULTADOS OBTIDOS COM A IMAGEM MENINA (GIRL) 131
5.5 RESULTADOS OBTIDOS COM A IMAGEM TIGRE 147
5.6 RESULTADOS OBTIDOS COM A IMAGEM VENTILADOR 163
5.7 RESULTADO CONSOLIDADO KLT QL 179
6 DISCUSSÃO DOS RESULTADOS 191
7 CONCLUSÕES E RECOMENDAÇÕES 199
8 REFERÊNCIAS 202
9 APÊNDICE A: PROGRAMA EXPERIMENTAL 208
10 APÊNDICE B: TESTES DE QUANTIZAÇÃO 210
11 APÊNDICE C: TESTES DO PROGRAMA 215
12 APÊNDICE D: ALINHAMENTO DOS ARQUIVOS BITMAP ... 219
13 APÊNDICE E: EFEITO DA BLOCAGEM NA DCT 224

LISTA DE ILUSTRAÇÕES

Figura 1 – Modelo Geral do Sistema de Compressão de Imagens 25
Figura 2 – Modelo Simplificado de Compressão de Imagens 26
Figura 3 – Decomposição de um Vetor em outra Base Ortonormal 30
Figura 4 – Representação de 2 vetores-bloco sobre uma base Ortonormal original 36
Figura 5 – Posição da nova base dos autovetores (“p” e “q”) 37
Figura 6 – Decomposição dos Vetores-bloco na nova base dos autovetores 37
Figura 7 – Redução Dimensional pela eliminação do autovetor “v” 38
Figura 8 – Reconstituição dos Vetores-base após a redução dimensional 39
Figura 9 – Seqüência do Algoritmo Piramidal de Cálculo da DWT 69
Figura 10 – (a) Imagem LENNA Original (512x512) e (b) LENNA PSNR = 31,75687 77
Figura 11 – Autovalores da Imagem LENNA 2x2 77
Figura 12 – (a) LENNA (PSNR=26,977861) e (b) LENNA (PSNR=23,710419) 78
Figura 13 – Autovalores da Imagem LENNA 4x4 79
Figura 14 – Autovalores da Imagem LENNA 8x8 79
Figura 15 – (a) LENNA (PSNR=40,448388) e (b) LENNA (PSNR=58,65534) 80
Figura 16 – Figura LENNA (512 x 512) 83
Figura 17 – Histograma da Imagem LENNA (512x512) 83
Figura 18 – PSNR da Imagem LENNA (Bloco 2x2) 84
Figura 19 – CR da Imagem LENNA (Bloco 2x2) 85
Figura 20 – Tempo (seg) de Gravação da Imagem LENNA (Bloco 2x2) 86
Figura 21 – PSNR da Imagem LENNA (Bloco 4x4) 87
Figura 22 – CR da Imagem LENNA (BLOCO 4x4) 88
Figura 23 – Tempo (seg) de Gravação da Imagem LENNA (BLOCO 4x4) 89
Figura 24 – PSNR da Imagem LENNA (BLOCO 8x8) 90
Figura 25 – CR da Imagem LENNA (BLOCO 8x8) 91
Figura 26 – Tempo de Gravação (seg) da Imagem LENNA (BLOCO 8x8) 92
Figura 27 – PSNR da Imagem LENNA (BLOCO 16x16) 93
Figura 28 – CR da Imagem LENNA (BLOCO 16x16) 94
Figura 29 – Tempo de Gravação (seg) da Imagem LENNA (BLOCO 16x16) 95
Figura 30 – PSNR da Imagem LENNA – Compressão QL 96
Figura 31 – CR da Imagem LENNA – Compressão QL 97
Figura 32 – Tempo de Gravação da Imagem LENNA – Compressão QL 98
Figura 33 – Figura BABUÍNO (512 x 512) 99
Figura 34 – Histograma da Imagem BABUÍNO (512x512) 99
Figura 35 – PSNR da Imagem BABUÍNO (Bloco 2x2) 100
Figura 36 – CR da Imagem BABUÍNO (Bloco 2x2) 101
Figura 37 – Tempo (seg) de Gravação da Imagem BABUÍNO (Bloco 2x2) 102
Figura 38 – PSNR da Imagem BABUÍNO (Bloco 4x4) 103
Figura 39 – CR da Imagem BABUÍNO (BLOCO 4x4) 104
Figura 40 – Tempo (seg) de Gravação da Imagem BABUÍNO (BLOCO 4x4) 105

Figura 41 – PSNR da Imagem BABUÍNO (BLOCO 8x8) 106
Figura 42 – CR da Imagem BABUÍNO (BLOCO 8x8) 107
Figura 43 – Tempo de Gravação (seg) da Imagem BABUÍNO (BLOCO 8x8) 108
Figura 44 – PSNR da Imagem BABUÍNO (BLOCO 16x16) 109
Figura 45 – CR da Imagem BABUÍNO (BLOCO 16x16) 110
Figura 46 – Tempo de Gravação (seg) da Imagem BABUÍNO (BLOCO 16x16) 111
Figura 47 – PSNR da Imagem BABUÍNO – Compressão QL 112
Figura 48 – CR da Imagem BABUÍNO – Compressão QL 113
Figura 49 – Tempo de Gravação da Imagem BABUÍNO – Compressão QL 114
Figura 50 – Figura CASA (256 x 256) 115
Figura 51 – Histograma da Imagem CASA (256x256) 115
Figura 52 – PSNR da Imagem CASA (Bloco 2x2) 116
Figura 53 – CR da Imagem CASA (Bloco 2x2) 117
Figura 54 – Tempo (seg) de Gravação da Imagem CASA (Bloco 2x2) 118
Figura 55 – PSNR da Imagem CASA (Bloco 4x4) 119
Figura 56 – CR da Imagem CASA (BLOCO 4x4) 120
Figura 57 – Tempo (seg) de Gravação da Imagem CASA (BLOCO 4x4) 121
Figura 58 – PSNR da Imagem CASA (BLOCO 8x8) 122
Figura 59 – CR da Imagem CASA (BLOCO 8x8) 123
Figura 60 – Tempo de Gravação (seg) da Imagem CASA (BLOCO 8x8) 124
Figura 61 – PSNR da Imagem CASA (BLOCO 16x16) 125
Figura 62 – CR da Imagem CASA (BLOCO 16x16) 126
Figura 63 – Tempo de Gravação (seg) da Imagem CASA (BLOCO 16x16) 127
Figura 64 – PSNR da Imagem CASA – Compressão QL 128
Figura 65 – CR da Imagem CASA – Compressão QL 129
Figura 66 – Tempo de Gravação da Imagem CASA – Compressão QL 130
Figura 67 – Figura MENINA (256 x 256) 131
Figura 68 – Histograma da Imagem MENINA (256x256) 131
Figura 69 – PSNR da Imagem MENINA (Bloco 2x2) 132
Figura 70 – CR da Imagem MENINA (Bloco 2x2) 133
Figura 71 – Tempo (seg) de Gravação da Imagem MENINA (Bloco 2x2) 134
Figura 72 – PSNR da Imagem MENINA (Bloco 4x4) 135
Figura 73 – CR da Imagem MENINA (BLOCO 4x4) 136
Figura 74 – Tempo (seg) de Gravação da Imagem MENINA (BLOCO 4x4) 137
Figura 75 – PSNR da Imagem MENINA (BLOCO 8x8) 138
Figura 76 – CR da Imagem MENINA (BLOCO 8x8) 139
Figura 77 – Tempo de Gravação (seg) da Imagem MENINA (BLOCO 8x8) 140
Figura 78 – PSNR da Imagem MENINA (BLOCO 16x16) 141
Figura 79 – CR da Imagem MENINA (BLOCO 16x16) 142
Figura 80 – Tempo de Gravação (seg) da Imagem MENINA (BLOCO 16x16) 143
Figura 81 – PSNR da Imagem MENINA – Compressão QL 144
Figura 82 – CR da Imagem MENINA – Compressão QL 145
Figura 83 – Tempo de Gravação da Imagem MENINA – Compressão QL 146
Figura 84 – Figura TIGRE (256 x 256) 147
Figura 85 – Histograma da Imagem TIGRE (256x256) 147
Figura 86 – PSNR da Imagem TIGRE (Bloco 2x2) 148
Figura 87 – CR da Imagem TIGRE (Bloco 2x2) 149
Figura 88 – Tempo (seg) de Gravação da Imagem TIGRE (Bloco 2x2) 150
Figura 89 – PSNR da Imagem TIGRE (Bloco 4x4) 151

Figura 90 – CR da Imagem TIGRE (BLOCO 4x4)..... 152
Figura 91 – Tempo (seg) de Gravação da Imagem TIGRE (BLOCO 4x4).....153
Figura 92 – PSNR da Imagem TIGRE (BLOCO 8x8) 154
Figura 93 – CR da Imagem TIGRE (BLOCO 8x8)..... 155
Figura 94 – Tempo de Gravação (seg) da Imagem TIGRE (BLOCO 8x8).....156
Figura 95 – PSNR da Imagem TIGRE (BLOCO 16x16) 157
Figura 96 – CR da Imagem TIGRE (BLOCO 16x16)..... 158
Figura 97 – Tempo de Gravação (seg) da Imagem TIGRE (BLOCO 16x16).....159
Figura 98 – PSNR da Imagem TIGRE – Compressão QL 160
Figura 99 – CR da Imagem TIGRE – Compressão QL..... 161
Figura 100 – Tempo de Gravação da Imagem TIGRE – Compressão QL.....162
Figura 101 – Figura VENTILADOR (256 x 256)..... 163
Figura 102 – Histograma da Imagem VENTILADOR (256x256).....163
Figura 103 – PSNR da Imagem VENTILADOR (Bloco 2x2) 164
Figura 104 – CR da Imagem VENTILADOR (Bloco 2x2)..... 165
Figura 105 – Tempo (seg) de Gravação da Imagem VENTILADOR (Bloco 2x2) 166
Figura 106 – PSNR da Imagem VENTILADOR (Bloco 4x4) 167
Figura 107 – CR da Imagem VENTILADOR (BLOCO 4x4) 168
Figura 108 – Tempo (seg) de Gravação da Imagem VENTILADOR (BLOCO 4x4).....169
Figura 109 – PSNR da Imagem VENTILADOR (BLOCO 8x8).....170
Figura 110 – CR da Imagem VENTILADOR (BLOCO 8x8) 171
Figura 111 – Tempo de Gravação (seg) da Imagem VENTILADOR (BLOCO 8x8).....172
Figura 112 – PSNR da Imagem VENTILADOR (BLOCO 16x16).....173
Figura 113 – CR da Imagem VENTILADOR (BLOCO 16x16) 174
Figura 114 – Tempo de Gravação (seg) da Imagem VENTILADOR (BLOCO 16x16).....175
Figura 115 – PSNR da Imagem VENTILADOR – Compressão QL.....176
Figura 116 – CR da Imagem VENTILADOR – Compressão QL.....177
Figura 117 – Tempo de Gravação da Imagem VENTILADOR – Compressão QL 178
Figura 118 – Gráficos PSNR para Compressão 4:4 (sem compressão) 179
Figura 119 – Gráficos CR para Compressão 4:4 (sem compressão).....180
Figura 120 – Gráfico de Tempo (seg.) para Compressão 4:4 (sem compressão) 181
Figura 121 – Gráficos PSNR para Compressão 4:3 182
Figura 122 – Gráficos CR para Compressão 4:3..... .183
Figura 123 – Gráfico de Tempo (seg.) para Compressão 4:3 184
Figura 124 – Gráficos PSNR para Compressão 4:2 185
Figura 125 – Gráficos CR para Compressão 4:2..... .186
Figura 126 – Gráfico de Tempo (seg.) para Compressão 4:2 187
Figura 127 – Gráficos PSNR para Compressão 4:1 188
Figura 128 – Gráficos CR para Compressão 4:1..... .189
Figura 129 – Gráfico de Tempo (seg.) para Compressão 4:1 190
Figura 130 – Imagem de Teste (“padrao4x4256.bmp”) 210
Figura 131 – Imagem “tigra.bmp” (Tamanho 640 x 480 PIXELS).....211
Figura 132 – Erro na Imagem devido a Quantização dos Componentes KL.....211
Figura 133 – Imagem 640x480 PIXELS utilizada no teste do Programa.....216
Figura 134 – Imagem Original (1194 x 1794) 219
Figura 135 – Redução da Imagem Original para uma Imagem de 512 x 512 PIXELS 220
Figura 136 – Problemas de Alinhamento (a)Imagem Original e (b)Imagem com Problemas222
Figura 137 – Ampliação do local com problema de alinhamento.....222
Figura 138 – Histograma LENNA 2x2 (Compressão de Bloco 2:1) para DCT INT.....225

Figura 139 – Histograma LENNA 2x2 (Compressão de Bloco 2:1) para DCT BLOCO 225
Figura 140 – Histograma para DCT INT sem o coeficiente de valor *0 (zero)*.....226
Figura 141 – Histograma para DCT BLOCO sem o coeficiente de valor *0* (*zero*).....227

LISTA DE TABELAS

Tabela 1 – Transmissão de Dados e Imagem sem Compressão 17
Tabela 2 – PSNR da Imagem LENNA (BLOCO 2x2) 84
Tabela 3 – CR da Imagem LENNA (BLOCO 2x2) 85
Tabela 4 – Tempo (seg) de Gravação da Imagem LENNA (BLOCO 2x2) 86
Tabela 5 – PSNR da Imagem LENNA (BLOCO 4x4) 87
Tabela 6 – CR da Imagem LENNA (BLOCO 4x4) 88
Tabela 7 – Tempo (seg) de Gravação da Imagem LENNA (BLOCO 4x4) 89
Tabela 8 – PSNR da Imagem LENNA (BLOCO 8x8) 90
Tabela 9 – CR da Imagem LENNA (BLOCO 8x8) 91
Tabela 10 – Tempo de Gravação (seg) da Imagem LENNA (BLOCO 8x8) 92
Tabela 11 – PSNR da Imagem LENNA (BLOCO 16x16) 93
Tabela 12 – CR da Imagem LENNA (BLOCO 16x16) 94
Tabela 13 – Tempo de Gravação (seg) da Imagem LENNA (BLOCO 16x16) 95
Tabela 14 – PSNR da Imagem LENNA – Compressão QL 96
Tabela 15 – CR da Imagem LENNA – Compressão QL 97
Tabela 16 – Tempo de Gravação da Imagem LENNA – Compressão QL 98
Tabela 17 – PSNR da Imagem BABUÍNO (BLOCO 2x2) 100
Tabela 18 – CR da Imagem BABUÍNO (BLOCO 2x2) 101
Tabela 19 – Tempo (seg) de Gravação da Imagem BABUÍNO (BLOCO 2x2) 102
Tabela 20 – PSNR da Imagem BABUÍNO (BLOCO 4x4) 103
Tabela 21 – CR da Imagem BABUÍNO (BLOCO 4x4) 104
Tabela 22 – Tempo (seg) de Gravação da Imagem BABUÍNO (BLOCO 4x4) 105
Tabela 23 – PSNR da Imagem BABUÍNO (BLOCO 8x8) 106
Tabela 24 – CR da Imagem BABUÍNO (BLOCO 8x8) 107
Tabela 25 – Tempo de Gravação (seg) da Imagem BABUÍNO (BLOCO 8x8) 108
Tabela 26 – PSNR da Imagem BABUÍNO (BLOCO 16x16) 109
Tabela 27 – CR da Imagem BABUÍNO (BLOCO 16x16) 110
Tabela 28 – Tempo de Gravação (seg) da Imagem BABUÍNO (BLOCO 16x16) 111
Tabela 29 – PSNR da Imagem BABUÍNO – Compressão QL 112
Tabela 30 – CR da Imagem BABUÍNO – Compressão QL 113
Tabela 31 – Tempo de Gravação da Imagem BABUÍNO – Compressão QL 114
Tabela 32 – PSNR da Imagem CASA (BLOCO 2x2) 116
Tabela 33 – CR da Imagem CASA (BLOCO 2x2) 117
Tabela 34 – Tempo (seg) de Gravação da Imagem CASA (BLOCO 2x2) 118
Tabela 35 – PSNR da Imagem CASA (BLOCO 4x4) 119
Tabela 36 – CR da Imagem CASA (BLOCO 4x4) 120
Tabela 37 – Tempo (seg) de Gravação da Imagem CASA (BLOCO 4x4) 121
Tabela 38 – PSNR da Imagem CASA (BLOCO 8x8) 122
Tabela 39 – CR da Imagem CASA (BLOCO 8x8) 123
Tabela 40 – Tempo de Gravação (seg) da Imagem CASA (BLOCO 8x8) 124

Tabela 41 – PSNR da Imagem CASA (BLOCO 16x16) 125
Tabela 42 – CR da Imagem CASA (BLOCO 16x16) 126
Tabela 43 – Tempo de Gravação (seg) da Imagem CASA (BLOCO 16x16) 127
Tabela 44 – PSNR da Imagem CASA – Compressão QL 128
Tabela 45 – CR da Imagem CASA – Compressão QL 129
Tabela 46 – Tempo de Gravação da Imagem CASA – Compressão QL 130
Tabela 47 – PSNR da Imagem MENINA (BLOCO 2x2) 132
Tabela 48 – CR da Imagem MENINA (BLOCO 2x2) 133
Tabela 49 – Tempo (seg) de Gravação da Imagem MENINA (BLOCO 2x2) 134
Tabela 50 – PSNR da Imagem MENINA (BLOCO 4x4) 135
Tabela 51 – CR da Imagem MENINA (BLOCO 4x4) 136
Tabela 52 – Tempo (seg) de Gravação da Imagem MENINA (BLOCO 4x4) 137
Tabela 53 – PSNR da Imagem MENINA (BLOCO 8x8) 138
Tabela 54 – CR da Imagem MENINA (BLOCO 8x8) 139
Tabela 55 – Tempo de Gravação (seg) da Imagem MENINA (BLOCO 8x8) 140
Tabela 56 – PSNR da Imagem MENINA (BLOCO 16x16) 141
Tabela 57 – CR da Imagem MENINA (BLOCO 16x16) 142
Tabela 58 – Tempo de Gravação (seg) da Imagem MENINA (BLOCO 16x16) 143
Tabela 59 – PSNR da Imagem MENINA - Compressão QL 144
Tabela 60 – CR da Imagem MENINA – Compressão QL 145
Tabela 61 – Tempo de Gravação da Imagem MENINA – Compressão QL 146
Tabela 62 – PSNR da Imagem TIGRE (BLOCO 2x2) 148
Tabela 63 – CR da Imagem TIGRE (BLOCO 2x2) 149
Tabela 64 – Tempo (seg) de Gravação da Imagem TIGRE (BLOCO 2x2) 150
Tabela 65 – PSNR da Imagem TIGRE (BLOCO 4x4) 151
Tabela 66 – CR da Imagem TIGRE (BLOCO 4x4) 152
Tabela 67 – Tempo (seg) de Gravação da Imagem TIGRE (BLOCO 4x4) 153
Tabela 68 – PSNR da Imagem TIGRE (BLOCO 8x8) 154
Tabela 69 – CR da Imagem TIGRE (BLOCO 8x8) 155
Tabela 70 – Tempo de Gravação (seg) da Imagem TIGRE (BLOCO 8x8) 156
Tabela 71 – PSNR da Imagem TIGRE (BLOCO 16x16) 157
Tabela 72 – CR da Imagem TIGRE (BLOCO 16x16) 158
Tabela 73 – Tempo de Gravação (seg) da Imagem TIGRE (BLOCO 16x16) 159
Tabela 74 – PSNR da Imagem TIGRE – Compressão QL 160
Tabela 75 – CR da Imagem TIGRE – Compressão QL 161
Tabela 76 – Tempo de Gravação da Imagem TIGRE – Compressão QL 162
Tabela 77 – PSNR da Imagem VENTILADOR (BLOCO 2x2) 164
Tabela 78 – CR da Imagem VENTILADOR (BLOCO 2x2) 165
Tabela 79 – Tempo (seg) de Gravação da Imagem VENTILADOR (BLOCO 2x2) 166
Tabela 80 – PSNR da Imagem VENTILADOR (BLOCO 4x4) 167
Tabela 81 – CR da Imagem VENTILADOR (BLOCO 4x4) 168
Tabela 82 – Tempo (seg) de Gravação da Imagem VENTILADOR (BLOCO 4x4) 169
Tabela 83 – PSNR da Imagem VENTILADOR (BLOCO 8x8) 170
Tabela 84 – CR da Imagem VENTILADOR (BLOCO 8x8) 171
Tabela 85 – Tempo de Gravação (seg) da Imagem VENTILADOR (BLOCO 8x8) 172
Tabela 86 – PSNR da Imagem VENTILADOR (BLOCO 16x16) 173
Tabela 87 – CR da Imagem VENTILADOR (BLOCO 16x16) 174
Tabela 88 – Tempo de Gravação (seg) da Imagem VENTILADOR (BLOCO 16x16) 175
Tabela 89 – PSNR da Imagem VENTILADOR - Compressão QL 176

Tabela 90 – CR da Imagem VENTILADOR - Compressão QL.....177
Tabela 91 – Tempo de Gravação da Imagem VENTILADOR – Compressão QL.....178
Tabela 92 – Tabela da PSNR para Compressão de Bloco 4:4.....179
Tabela 93 – Tabela da CR para Compressão de Bloco 4:4.....180
Tabela 94 – Tabela de Tempo (seg.) para Compressão de Bloco 4:4.....181
Tabela 95 – Tabela da PSNR para Compressão de Bloco 4:3.....182
Tabela 96 – Tabela da CR para Compressão de Bloco 4:3.....183
Tabela 97 – Tabela de Tempo (seg.) para Compressão de Bloco 4:3.....184
Tabela 98 – Tabela da PSNR para Compressão de Bloco 4:2.....185
Tabela 99 – Tabela da CR para Compressão de Bloco 4:2.....186
Tabela 100 – Tabela de Tempo (seg.) para Compressão de Bloco 4:2.....187
Tabela 101 – Tabela da PSNR para Compressão de Bloco 4:1.....188
Tabela 102 – Tabela da CR para Compressão de Bloco 4:1.....189
Tabela 103 – Tabela de Tempo (seg.) para Compressão de Bloco 4:1.....190
Tabela 104 – Comparação de PSNR após Quantização.....210
Tabela 105 – Comparação de PSNR após Quantização (Corrigida).....212

LISTA DE ABREVIATURAS

ANSI: AMERICAN NATIONAL STANDARD INSTITUTE

CCITT: INTERNATIONAL TELEGRAPH AND TELEPHONE CONSULTATIVE COMMITTEE (atualmente denominado ITU)

CPGEE: Curso de Pós-Graduação em Engenharia Elétrica

DCT: DISCRETE COSINE TRANSFORM

DWT: DISCRETE WAVELET TRANSFORM

FFT: FAST FOURIER TRANSFORM

GPRS: GENERAL PACKET RADIO SERVICE

HDTV: HIGH DEFINITION TELEVISION

ISO: INTERNATIONAL STANDARD ORGANIZATION

ITU: INTERNATIONAL TELECOMMUNICATION UNION

JPEG: JOINT PHOTOGRAPHIC EXPERTS GROUP (Padrão ISO/1992)

JPEG2000: JOINT PHOTOGRAPHIC EXPERTS GROUP 2000

KL/KLT: KARHUNEN-LOÈVE/KARHUNEN-LOÈVE TRANSFORM

MB: MEGA BYTES (10^6 BYTES)

MPEG: MOVIE PICTURE EXPERTS GROUP

ms: milisegundos

MSE: MEAN SQUARE ERROR

PEG: PHOTOGRAPHIC EXPERTS GROUP

PSNR: PEAK SIGNAL-TO-NOISE RATIO

UFRGS: Universidade Federal do Rio Grande do Sul

UMTS: UNIVERSAL MOBILE TELECOMMUNICATIONS SYSTEM

1 INTRODUÇÃO

A utilização de técnicas de representação de sinais utilizando funções-base (KERNEL) tem possibilitado o processamento e a transmissão de sinais de forma mais eficiente e conveniente do que aquela que seria obtida pela simples transmissão direta do próprio sinal.

Neste contexto, quando comparada a qualquer outra transformada, a Transformada KARHUNEN-LOÈVE se destaca pois ela utiliza uma base de representação que permite alcançar o menor valor de erro de reconstrução do sinal (Erro Quadrático Médio) para uma determinada quantidade de coeficientes transmitidos. O processamento envolvido, porém, na obtenção desta base ótima, tem limitado o uso desta Transformada a aplicações específicas, como o reconhecimento de imagens e outras aplicações que não necessitem de execução *em tempo real*.

A motivação principal deste trabalho surge da necessidade cada vez maior de utilização de técnicas de compressão de sinais (áudio e vídeo) capazes de transmitir de forma eficiente e rápida as informações necessárias à comunicação do homem moderno.

A observação da Tabela 1 evidencia a importância e a motivação para o estudo da compressão de sinais em geral e da compressão de imagens em especial. Nela se pode verificar claramente a necessidade de suficiente espaço de armazenamento, largura de banda de transmissão e tempo de transmissão de dados, de imagens estáticas e em movimento.

Tabela 1 – Transmissão de Dados e Imagem sem Compressão

DADOS MULTIMÍDIA	TAMANHO/ DURAÇÃO	BITS/PIXEL OU BITS/ AMOSTRA	TAMANHO SEM COMPRESSÃO (B BYTES)	TOTAL DE BITS À TRANSMITIR (b bits)	TEMPO DE TRANSMISSÃO (MODEM DE 28,8 Kbps
UMA PÁGINA DE TEXTO	11” x 8,5”	8 (bits/car)	4 – 8 (KB/Página)	32 – 64 (Kb/Página)	1,1 – 2,2 (seg)
VOZ POR TELEFONE	10 (seg)	8 (bits/amost)	80 (KB / 10seg)	640 (Kb/10 seg)	22,2 (seg)
IMAGEM PRETO E BRANCO	512 x 512 (pixels)	8 (bits/pixel)	262 (KB/imagem)	2,1 (Mb/imagem)	1 min 13 seg
IMAGEM COLORIDA	512 x 512 (pixels)	24 (bits/pixels)	786 (KB/imagem)	6,29 (Mb/imagem)	3 min 39 seg
IMAGEM EM MOVIMENTO 30 quadros/seg	640 x 480 (1 min)	24 (bits/pixel)	1,66 (GB/minuto)	13,28 (Gb/minuto)	5 dias 8 horas

Embora se tenha incluído vários tipos de fontes de sinal na Tabela 1, para efeito de comparação, toda a análise deste trabalho foi baseada em imagens estáticas em preto e branco (item marcado em azul na tabela). A compressão atingida por cada fonte de sinal apresentada na Tabela 1 depende da técnica utilizada na compressão, a qual varia de acordo com a própria fonte de sinal sobre a qual ela é aplicada. Neste estudo foram utilizadas 3 técnicas na compressão de imagens estáticas em preto e branco, a saber: a Transformada KARHUNEN-LOÈVE, a Transformada Discreta do Coseno e a Transformada WAVELET Discreta.

O objetivo deste trabalho é o estudo da Transformada KARHUNEN-LOÈVE e a sua aplicação na compressão de imagens bidimensionais estáticas e em 256 tons de cinza. Sendo considerada ótima no sentido do Erro Quadrático Médio, entre todas as demais transformadas existentes, a KLT é aqui detalhada na sua formulação matemática, no seu significado físico, na descrição de suas aplicações atuais, na utilização dessa transformada na compressão de imagens e na sua comparação com outras duas técnicas de compressão, mais especificamente com a Transformada Discreta do Coseno (DCT - DISCRETE COSINE TRANSFORM) e com a Transformada WAVELET Discreta (DWT – DISCRETE WAVELET TRANSFORM). Apresentam-se ainda 2 técnicas importantes utilizadas para o cálculo dos autovalores e autovetores da matriz de correlação dos pontos componentes da imagem (Método de JACOBI e Método QL), fundamentais para o cálculo da KLT, e a elaboração de um programa experimental desenvolvido especialmente para efetuar a comparação de resultados com a DCT e a DWT. Os experimentos realizados foram descritos e os seus resultados apresentados neste trabalho.

Como protocolo de comparação entre a KLT, a DCT e a DWT utilizou-se, para avaliação dos resultados, os seguintes parâmetros:

- A Razão Sinal-Ruído de Pico (*PSNR – PEAK SIGNAL-TO-NOISE RATIO*);
- a Razão de Compressão (*CR – COMPRESSION RATIO*);
- o Tempo de processamento do arquivo compactado;
- os tamanhos de bloco 2x2, 4x4, 8x8 e 16x16;
- a seleção dos coeficientes de representação pelo maior autovalor e maior valor absoluto (para a KLT), maior valor absoluto e ZIGZAG (para a DCT) e maior valor absoluto (para a DWT).

Existem muitos artigos dedicados a compressão de imagens, que envolvem também comparações entre técnicas de compressão. Na Seção 2 apresentam-se os artigos mais importantes nessa área, especialmente aqueles que mais se relacionam com este trabalho, e na Seção 3.2.3 são comentados os principais resultados de cada um. De um modo geral,

entretanto, foram constatadas algumas carências de detalhamento nas técnicas apresentadas nos artigos pesquisados, e que apenas aumentaram a motivação para a realização deste trabalho. De forma geral, as carências mais comuns encontradas foram:

- a inexistência de referência quanto aos algoritmos utilizados no cálculo dos autovalores e autovetores da matriz de correlação, quando o cálculo envolve a utilização de processos analíticos;
- a utilização, nas comparações de resultados, de apenas um dos indicadores (MSE - MEAN SQUARE ERROR / PSNR – PEAK SIGNAL-NOISE RATIO ou CR – COMPRESSION RATIO), limitando a análise;
- a falta de indicação dos tempos de processamento envolvidos nos experimentos apresentados;
- a predominância da formulação matemática em detrimento da experimentação e comparação com outras técnicas;
- a ausência de comparação com a DCT ou DWT ou ambas, hoje utilizadas como padrão internacional;
- o baixo (ou mesmo ausência) de detalhamento dos critérios de comparação com a KLT, gerando comparações em bases diferentes (com programas diferentes ou com quantizações diferentes ou com razões de compressão não equivalentes);
- a utilização de imagens não padrão na realização de comparações;
- a utilização de uma quantidade de imagens insuficientes para inferir conclusões gerais;
- a utilização de várias imagens para os testes sem apresentação das razões de escolha das mesmas, nem as características estatísticas de cada uma que as qualificasse para a realização dos testes;
- a falta de detalhamento do ambiente da experimentação;
- as experimentações baseadas exclusivamente em quantização por THRESHOLD para comparação das técnicas e não na equivalência do número de coeficientes da transformada.

Na Seção 2 apresenta-se a Revisão de Literatura, onde se destacam os principais artigos relacionados com o objetivo deste trabalho. Na Seção 3 são apresentados os fundamentos teóricos que embasam o trabalho, iniciando por uma breve introdução, definições e a estrutura básica do Modelo de Compressão aplicado aos Sistemas de Compressão de Imagens em geral. Também são definidas nesta Seção as equações dos

parâmetros utilizados neste trabalho para efetuar as comparações das técnicas (CR e PSNR). Em seguida, descreve-se o desenvolvimento matemático, as aplicações e os artigos (e seus algoritmos) relacionados com a compressão de imagens utilizando a KLT. Na Seção 3.3 inclui-se a descrição do Algoritmo de HUFFMAN utilizado para a compressão da imagem na fase experimental. Na Seção 3.4 descreve-se a Transformada Discreta do Coseno, utilizada na comparação com a KLT na fase experimental. Na Seção 3.5 descreve-se a Transformada WAVELET Discreta também utilizada na comparação com a KLT na fase experimental. A Seção 4 fornece uma descrição dos experimentos realizados neste trabalho. Nas Seções 5 e 6 os resultados obtidos nos experimentos são apresentados e discutidos. Na Seção 7 são feitas as conclusões do trabalho bem com as sugestões para trabalhos futuros. Ao final, é acrescentada a bibliografia com as principais fontes de consulta utilizadas. Na Seção 9 está incluída uma referência ao programa implementado na etapa experimental deste trabalho, bem como a forma de obtenção do mesmo. Na Seção 10 é apresentado o tipo de quantização utilizada neste programa e os testes realizados. Na Seção 11 são descritos os testes do programa implementado na fase experimental, na Seção 12 são descritos os problemas ocasionados pelo alinhamento de BYTES e na Seção 13 é abordado o efeito do tamanho do bloco de imagem nos resultados obtidos na DCT. Em (MELLO,2003) foram condensadas todas as definições teóricas básicas que são mencionadas e utilizadas neste trabalho, além de apresentadas as técnicas e algoritmos pesquisados em artigos com aplicações e melhoramentos de compressão de imagens utilizando a KLT.

As técnicas de compressão de imagens utilizando a tecnologia baseada em Redes Neurais e na Teoria dos Fractais não são objeto de análise neste trabalho de dissertação, embora alguns resultados encontrados nos artigos pesquisados tenham sido mencionados para o caso das Redes Neurais.

2 REVISÃO DE LITERATURA

O trabalho apresentado em (HOTELLING,1933) foi o primeiro a propor e publicar a operação que transforma variáveis discretas em coeficientes não correlacionados. A técnica foi referida como o Método dos Componentes Principais (METHOD OF PRINCIPAL COMPONENTS). Seu artigo, publicado em duas partes, nos meses de setembro e outubro de 1933, aplicava esta técnica para a análise de características individuais de crianças em idade escolar, relativamente a aspectos de habilidade motora e intelectual. Esse estudo forneceu uma visão significativa do método e viria a se constituir na base teórica aplicada em trabalhos posteriores utilizando esta técnica.

A transformação análoga a Análise dos Componentes Principais, para transformar dados contínuos num conjunto de coeficientes descorrelacionados foi descoberta por KARHUNEN (1947 apud GONZALEZ,1993,p.157), e por LOÈVE (1948 apud GONZALEZ, 1993,p.157), e foi chamada de expansão de KARHUNEN-LOÈVE.

O resultado de que a expansão de KARHUNEN-LOÈVE minimiza o Erro Quadrático Médio truncado foi primeiramente publicado por KOSCHMAN (1954 apud GONZALEZ, 1993,p.157), e posteriormente por (BROWN,1960), que ofereceu a prova matemática do resultado da integral do Erro Quadrático Médio entre uma função randômica *x(t)* e a representação por funções ortonormais de *x(t)* através de uma série truncada. Esta prova é bastante semelhante ao desenvolvimento matemático apresentado por (LINDQUIST,1989).

A Análise dos Componentes Principais introduzida em (HOTELLING,1933) foi utilizada posteriormente por (KRAMER, 1956) para descrever um esquema de codificação para a transmissão de *n* sinais contínuos correlacionados utilizando *m* canais. Cada um dos *m* sinais resultantes (a serem transmitidos nos *m* canais) era o resultado de uma combinação linear dos n sinais originais. Os coeficientes dessa transformação linear constituíam uma matriz *m x n*, e eram as constantes do esquema de codificação. Foi demonstrado nesse trabalho que os coeficientes que minimizam o Erro Quadrático Médio entre os sinais reconstituídos na recepção e os originais transmitidos eram os componentes dos autovetores da matriz de correlação dos sinais originais. O trabalho foi testado com sucesso num VOCODER no qual a voz humana era transmitida em diversas bandas de freqüência por meio de um conjunto de sinais proporcionais à energia da voz. No trabalho apresentado em (HUANG,1963) também foi utilizada essa técnica para descorrelacionar um conjunto de N

variáveis aleatórias gaussianas, propondo em seguida, sobre os coeficientes assim criados, um critério ótimo de quantização.

Na seção 3.2.3 descrevem-se algumas técnicas apresentadas nos artigos pesquisados relativas a utilização da Transformada KL para a compressão de imagens, bem como os algoritmos utilizados. Foram incluídos neste trabalho a descrição dos artigos referentes aos seguintes tópicos: a condição de minimização do Erro Quadrático Médio da KLT na presença do Ruído de Quantização (DIAMANTARAS,1999), a Codificação de Imagem com uma transformada adaptativa otimizada (DONY,1995), a técnica de compressão universal de imagens utilizando a Transformada KL (WUTC) (EFFROS,1995) bem como uma forma de cálculo que aumenta a performance da técnica WUTC, chamada de SWUTC (FENG,1999), a estimação dos coeficientes da KLT diretamente pela estatística dos dados quantizados (GOYAL,1996), a utilização da Transformada KL Aproximada (AKLT) (LEU,1994), o método de compressão de tamanho fixo de alta performance para imagens usando a Transformada KL (NASIOPOULOS,1995), o estudo comparativo entre as técnicas JPEG, KLT e Redes Neurais (OLIVEIRA,2000), a comparação da eficiência das transformadas analíticas KLT, DCT e DWT na compressão de imagens (RAGAB,1998), o cálculo da base ótima através da utilização dos vetores base da DCT em coordenadas polares (UENOHARA,1998), a compressão de imagem utilizando uma técnica híbrida com a Transformada KL e a decomposição de valor único (SVD-SINGLE VALUE DECOMPOSITION) (WALDEMAR,1997) e a técnica da utilização da Transformada KL ponderada para redução dos defeitos da blocagem (YAMASHITA,1996).

Diversos outros artigos foram pesquisados e são mencionados a seguir para fornecer uma dimensão da multiplicidade de aplicações desta transformada na codificação de imagens, embora não analisados no âmbito deste trabalho: a utilização da Transformada KL para o aumento da performance da compressão FRACTAL (BREAZU,1999), o aumento da eficiência da Transformada KL na compressão de imagens através da técnica de mesclagem dos passos da transformada e da quantização (BREAZU,2000), a compressão de imagens multi-espectrais através da combinação da Transformada KL e WAVELET (EPSTEIN,1992), os procedimentos para o processamento de imagens estáticas e em movimento utilizando a Transformada KL (KIRBY,1992), a identificação de objetos em imagens comprimidas (LEW,1994), a análise da extração seqüencial das bases da Transformada KL (LEVY,1998), a técnica de compressão de banda baseada em transformada adaptativa parcial para imagens multi-espectrais (SAGHRI,1993), a utilização de um algoritmo robusto para imagens multi-espectrais (SAGHRI,1995), a relação entre a Transformada KL e a Transformada Discreta do

Coseno (UENOHARA,1998) e o aumento da performance da codificação MPEG-4 VTC (VISUAL TEXTURE CODING) (WANG,2001). Outros artigos são mencionados na Seção 3.2.2 e referem-se a utilização da KLT em outras aplicações atualmente.

Como se percebe, as aplicações são bastante variadas e as teorias envolvidas são igualmente amplas. Em especial para a compressão de imagens, foram mencionados neste trabalho os artigos atuais que mais se relacionam com a compressão de imagens, publicados no período de 1990 até janeiro de 2002, e da sua leitura se pode constatar a extensa utilização da KLT e a intensa busca dos pesquisadores por métodos eficientes para sua implementação.

3 FUNDAMENTOS TEÓRICOS

3.1 O SISTEMA DE COMPRESSÃO DE IMAGENS

Antes de iniciar a análise da Transformada KARHUNEN-LOÈVE, é necessário mostrar o Modelo do Sistema de Compressão de Imagens, sobre o qual estão baseados quase a totalidade dos artigos pesquisados sobre compressão de imagem.

Para se compreender adequadamente as funções de cada módulo do modelo é necessário identificar claramente de que forma a compressão de imagens é realizada, isto é, de que maneira se pode atuar para realizar a compressão de uma imagem.

Um componente fundamental da compressão de imagens é a redução da redundância da informação. A *redução da redundância* objetiva remover a informação duplicada na fonte de sinal (imagem/vídeo).

Pode-se identificar (GONZALEZ,1993) as seguintes formas de redundâncias, que podem ser trabalhadas pelo Sistema de Compressão de Imagens:

- Redundância INTERPIXEL;
- Redundância DE CÓDIGO;
- Redundância PSICOVISUAL;
- Redundância TEMPORAL.

As técnicas de redução da *Redundância INTERPIXEL*, ou mapeadores, atuam no sentido de eliminar esse tipo de redundância, quer atuando sobre a redução da correlação existente entre os pontos da imagem, quer atuando no espectro de freqüência do sinal ou utilizando bases que se expandem e contraem buscando se adequarem a qualquer tipo de sinal. A efetividade dessas técnicas é alcançada através de um mapeamento, geralmente reversível, transformando o sinal de entrada num formato cuja representação não necessite de todos os coeficientes (mapeamento com compressão e com perda) ou que apenas permita uma redução adicional de tamanho da imagem, pelos módulos codificadores de símbolos (mapeamento sem compressão e sem perda). As técnicas baseadas na aplicação de *transformadas* (KLT, DCT, WLT, entre outras) que reduzem (ou mesmo eliminam) a redundância INTERPIXEL são exemplos de técnicas que atuam sobre a *Redundância INTERPIXEL*. As transformadas, como foi dito, podem ou não ser utilizadas como elementos de compressão da imagem propriamente dita.

As técnicas de redução da *Redundância DE CÓDIGO*, ou codificadores de símbolos, objetivam a codificação do sinal com a menor quantidade de unidades de representação (bits,

por exemplo) possível, sem degradar o sinal. Como exemplo, é possível citar os algoritmos de HUFFMANN, ZIV-LEMPEL CODING, BIT PLANE CODING, DOUBLE DELTA CODIGN, DELTA MODULATION, DIFFERENTIAL PULSE CODE MODULATION, entre outros. Esse processo é reversível.

As técnicas de redução da *Redundância PSICOVISUAL* (também chamada de *irrelevância)*, ou quantizadores, objetivam omitir parte do sinal que não será percebido pelo receptor, o Sistema Visual Humano (HVS – HUMAN VISUAL SYSTEM).

As técnicas de redução da *Redundância TEMPORAL* objetivam reduzir a correlação temporal existente numa seqüência de imagens estáticas ou nas imagens em movimento e não serão abordadas no escopo deste trabalho.

A Figura 1 representa o Modelo Geral do Sistema de Compressão de Imagens (GONZALEZ,1993), incluindo o ruído do canal de transmissão e os blocos responsáveis pela codificação e decodificação do canal, que atuam no sentido de anular o efeito do ruído do canal sobre a imagem *f(x,y)* a ser transmitida.

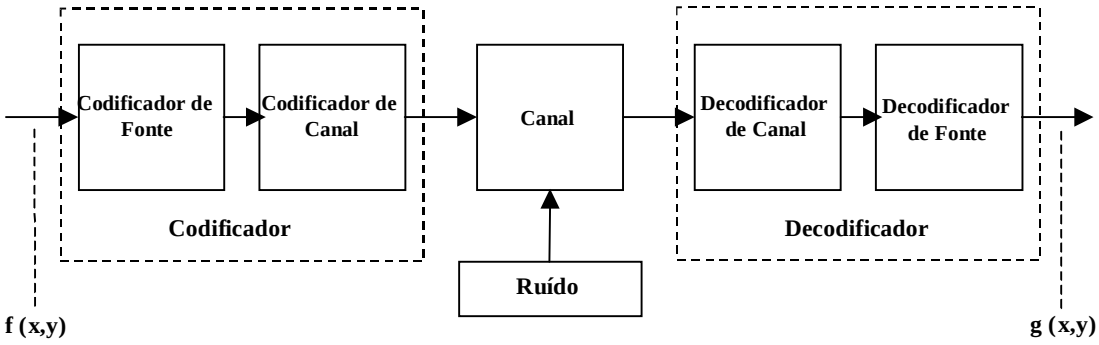
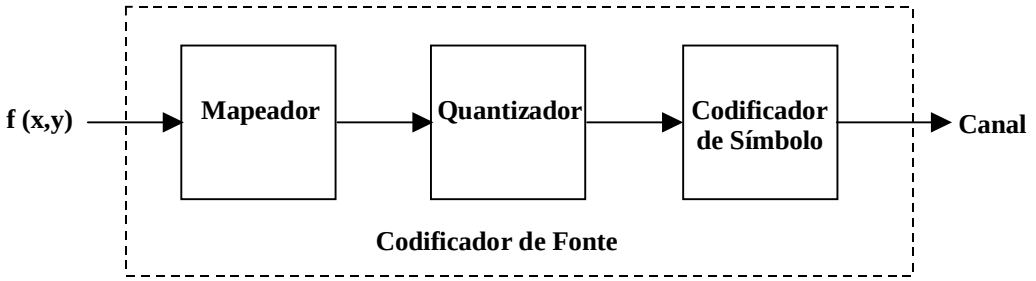


Figura 1 – Modelo Geral do Sistema de Compressão de Imagens

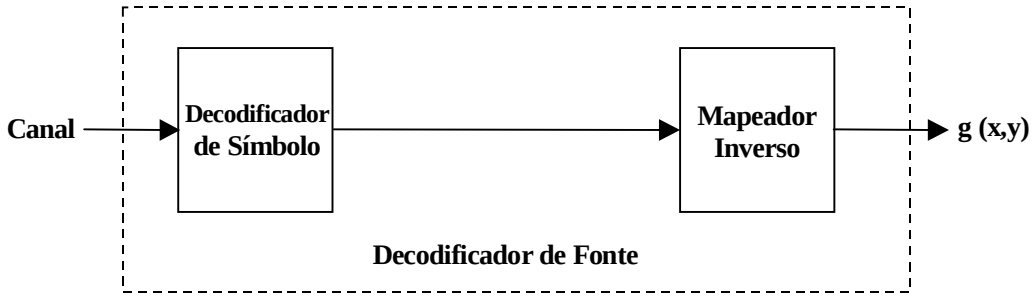
Especialmente para este trabalho utilizou-se o modelo simplificado, sem a presença do ruído de canal e, portanto, sem a necessidade de implementar o Codificador e o Decodificador de Canal, conforme indicado na Figura 2. Nela pode-se observar que apenas o Codificador e o Decodificador de Fonte foram representados.

É importante mencionar que, nestes modelos, *f(x,y)* representa os valores dos PIXELS da imagem a ser transmitida e *g(x,y)* representa a mesma imagem recuperada pela canal de transmissão.

O *Codificador de Fonte* é composto pelo *Mapeador*, pelo *Quantizador* e pelo *Codificador de Símbolos*. O *Decodificador de Fonte* é composto pelo *Decodificador de Símbolo* e pelo *Mapeador Inverso*.



(a) Codificador de Fonte



(b) Decodificador de Fonte

Figura 2 – Modelo Simplificado de Compressão de Imagens

Na Figura 2 o *Mapeador* é responsável pela eliminação da *Redundância INTERPIXEL*, o *Quantizador* é responsável pela eliminação da *Redundância PSICOVISUAL* e o *Codificador de Símbolos* é responsável pela eliminação da *Redundância de Código*, conforme a definição de *redundância* apresentada anteriormente nesta seção.

Nota-se que o *Decodificador de Fonte* não possui um bloco que realize o processo inverso da quantização presente no *Codificador de Fonte*. Isso ocorre porque o processo da quantização não é reversível pois implica em retirada definitiva de informação do sinal *f(x,y)*.

Cabe ressaltar que na compressão sem perdas a imagem reconstruída a partir da compressão é idêntica a imagem original. A capacidade de compressão, porém, neste caso, é modesta comparada com a compressão com perdas. Nesta, a imagem reconstruída contém alguma degradação em relação a imagem original, ocasionada pelo completo descarte da informação redundante.

O *Quantizador* simplesmente reduz o número de bits necessários para armazenar os coeficientes da imagem transformada reduzindo a precisão destes valores. Como é um processo do tipo "*vários para um*", implicará em perda e também numa fonte importante de compressão do sistema. A quantização pode ser feita em cada coeficiente (Quantização Escalar – SQ) ou num grupo de coeficientes (Quantificação Vetorial – VQ), podendo ser realizada de forma uniforme ou não uniforme (dependendo do nível de definição requerido nas partes que compõem a imagem).

O *Codificador de Símbolos* codifica, sem perdas, os valores quantizados permitindo uma compressão adicional ao sistema. O *Codificador de Entropia* é um tipo de *Codificador de Símbolos* que utiliza um modelo estatístico para determinar precisamente as probabilidades de cada valor quantizado produzindo um código apropriado baseado nestas probabilidades de tal forma que o fluxo de dados na saída seja menor do que o fluxo de entrada. Os Codificadores de Entropia mais comuns utilizados são o Codificador de HUFFMAN (utilizado neste trabalho) e o Codificador Aritmético, embora o simples Codificador RUN-LENGHT (RLE - RUN-LENGHT ENCODER) seja bastante utilizado para aplicações que requeiram execução rápida.

Neste trabalho, foram implementados 3 Mapeadores lineares: a KLT em sua forma tradicional, a DCT (base do padrão JPEG) e a DWT (base do padrão MPEG4). Foram utilizados 2 parâmetros, presentes também em vários artigos pesquisados (GONZALEZ,1993; CASTRO,1996; DIAMANTARAS,1999; RAGAB,1998), para comparação dos resultados experimentais da compressão da imagem de teste: a Razão Sinal-Ruído de Pico (PSNR) e a Razão de Compressão (CR). A Razão Sinal-Ruído de Pico (PSNR) é definida, para imagens quadradas N x N, como:

$$\mathrm{PSNR[dB]} = 10\log_{10}\left(\frac{X_{max}^2}{\frac{1}{NxN}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}[f(x,y)-g(x,y)]^2}\right) = 10\log_{10}\left(\frac{X_{max}^2}{MSE}\right) \qquad \textbf{(Eq. 1)}$$

onde X_{max} = 255 para imagens na escala de cinza (8 bits por pixel), e o denominador é o Erro Quadrático Médio (MSE).

A Razão de Compressão (CR) é definida como:

$$CR = \frac{\text{Tamanho da Imagem Original (BYTES)}}{\text{Tamanho da Imagem Compactada (BYTES)}} \qquad \textbf{(Eq. 2)}$$

3.2 TRANSFORMADA KARHUNEN-LOÈVE

Uma transformada é uma operação matemática que permite mudar a representação de pontos (vetores) ou de funções (sinais) dentro de um subespaço vetorial, utilizando um conjunto de vetores (ou funções) chamadas de “base” (KERNEL). De forma geral, a base de representação pode ser ortogonal ou não. A transformação de uma imagem ou sub-imagem (vetor-bloco) utilizando a base ortonormal dos autovetores da sua matriz de correlação decomporá esta imagem (ou subimagem) em componentes não correlacionados projetados sobre essa base. No caso de utilização de uma base ortogonal, o conjunto de vetores (ou funções) que formam esta base são linearmente independentes (CHEN,1999). A ortogonalidade permite eliminar os efeitos recíprocos de dois ou mais vetores (ou funções) componentes da base, simplificando assim a abordagem matemática envolvida. Outra vantagem da utilização da base ortogonal é que se pode truncar alguns componentes (ou coeficientes) da imagem transformada sem afetar os demais componentes (RAGAB,1998).

A KARHUNEN-LOÈVE é uma Transformada baseada em propriedades estatísticas de representação de vetores, o que a torna uma importante ferramenta no processamento de sinais, com aplicações tanto na área de compressão quanto na segmentação de imagens, entre outras aplicações que estão mencionadas neste trabalho.

Um subespaço N-dimensional proporcionará a melhor aproximação para um conjunto de pontos (sinais ou vetores) se e somente se for um subespaço de KARHUNEN-LOÈVE constituído pelos primeiros N autovalores de KARHUNEN-LOÈVE (KL). Esta afirmação constitui um Teorema já provado matematicamente (OGAWA,1992). Desta forma, a utilização dos autovetores calculados pela Transformada KL como base de representação do sinal (ou vetor) de entrada, constitui-se numa condição necessária e suficiente para a constituição de uma base que fornecerá a melhor aproximação deste sinal (ou vetor), isto é, o menor Erro Quadrático Médio entre o sinal original e o sinal reconstruído a partir do sinal codificado. Apresenta-se a seguir a abordagem matemática utilizada na análise da Transformada KL.

3.2.1 ABORDAGEM MATEMÁTICA DA KLT

Na abordagem matemática da KLT são utilizados alguns conceitos fundamentais e que possuem abrangência multi-disciplinar. Esses conceitos e definições já foram também utilizados em (CASTRO,1996) para um espaço real ou complexo e são complementados por

conceitos referentes a Vetores e Matrizes (BRONSON,1991; GANTMACHER,1974; GANTMACHER,1977; GOLUB,1996; HORN,1985), Processos Estocásticos (SHANMUGAN,1988), Processamento Digital de Imagens (GONZALEZ,1993), Sistemas Lineares (CHEN,1999) e Programação Linear (GOLDBARG,2000). As definições básicas utilizadas neste trabalho estão descritas no relatório apresentado em (MELLO,2003).

3.2.1.1 Interpretação Matemática e Gráfica da KLT

Num espaço bidimensional quaisquer dois vetores linearmente independentes podem representar qualquer ponto (ou vetor) deste plano.

Dado, por exemplo, o vetor $\mathbf{x} = (x_1, y_1)$ (notação vetorial) cujos componentes x_1 e y_1 são definidos como as projeções do vetor **x** sobre uma base ortonormal $B = \{\mathbf{i_1}, \mathbf{i_2}\}$com $\mathbf{i_1}, \mathbf{i_2} \in \mathbf{R^2}$, sendo que $\mathbf{i_1} = (1,0)$ e $\mathbf{i_2} = (0,1)$, pode-se escrever, na notação pelas bases canônicas $\mathbf{i_1}$ e $\mathbf{i_2}$:

$$\mathbf{x} = x_1\mathbf{i_1} + y_1\mathbf{i_2} = x_1(1,0) + y_1(0,1) = (x_1,0) + (0,y_1) = (x_1,y_1) \qquad \textbf{(Eq. 3)}$$

Esse mesmo vetor **x** pode ser representado por sua notação matricial (vetor-coluna), conforme definido em (ANTON,2001), como $\mathbf{x} = [\ x_1 \quad y_1\ ]^T$ com seus componentes x_1 e y_1 definidos novamente como as projeções do vetor **x** sobre uma base ortonormal $B = \{\mathbf{i_1}, \mathbf{i_2}\}$com $\mathbf{i_1}, \mathbf{i_2} \in \mathbf{R^2}$, sendo $\mathbf{i_1} = [\ 1 \quad 0\ ]^T$ e $\mathbf{i_2} = [\ 0 \quad 1\ ]^T$. Nesta notação matricial do vetor pode-se escrever:

$$\mathbf{x} = \mathbf{I}\,\mathbf{x} = \begin{bmatrix}\mathbf{i_1} & \mathbf{i_2}\end{bmatrix}\begin{bmatrix}x_1\\ y_1\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ y_1\end{bmatrix} = \begin{bmatrix}x_1\\ y_1\end{bmatrix} \qquad \textbf{(Eq. 4)}$$

onde **I** representa a matriz da base, sendo cada coluna desta matriz um vetor desta base. Neste trabalho utilizou-se a notação matricial para os vetores nos desenvolvimentos apresentados.

Para uma visualização clara da decomposição de um vetor em componentes sobre uma outra base ortonormal, realizou-se a seguir um exemplo considerando um único vetor de duas dimensões $\mathbf{x} = [x_{i1} \quad x_{i2}]^T = [3 \quad 2]^T$, com seus componentes representados numa base ortonormal $B_1 = \{\mathbf{i_1}, \mathbf{i_2}\}$ onde $\mathbf{i_1}, \mathbf{i_2} \in \mathbf{R^2}$ e são definidos por: $\mathbf{i_1} = [1 \quad 0]^T$ e $\mathbf{i_2} = [0 \quad 1]^T$.

O vetor **x**, neste exemplo, será decomposto em uma outra base ortonormal $B_2 = \{\mathbf{q_1}, \mathbf{q_2}\}$ onde $\mathbf{q_1}, \mathbf{q_2} \in \mathbf{R^2}$ e são definidos por: $\mathbf{q_1} = [2^{-1/2} \quad 2^{-1/2}]^T$ e $\mathbf{q_2} = [-(2^{-1/2}) \quad 2^{-1/2}]^T$.

Na Figura 3 representa-se graficamente o vetor **x** e sua decomposição em relação as bases B_1 e B_2. A nova representação do vetor **x** em relação a nova base B_2 é chamada de mudança de base do vetor **x** e como os componentes sobre a nova base são diferentes daqueles definidos originalmente para o vetor **x**, este passa a denominar-se $\mathbf{x_q}$.

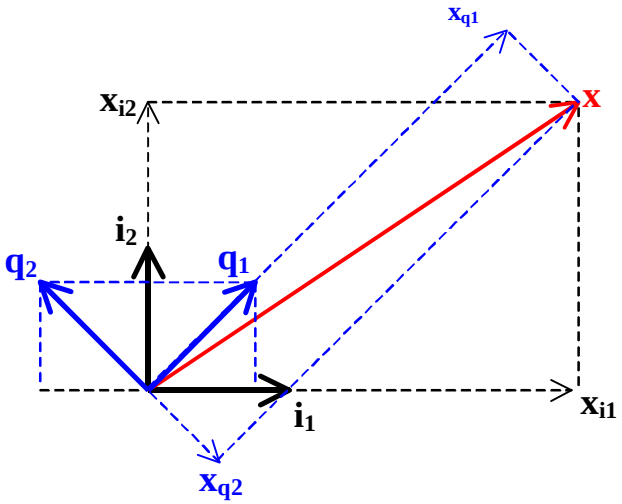


Figura 3 – Decomposição de um Vetor em outra Base Ortonormal

Neste exemplo:

$$\mathbf{x_q} = \begin{bmatrix} x_{q1} \\ x_{q2} \end{bmatrix} = \mathbf{Q}^{-1}\,\mathbf{x} = \begin{bmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix} = \begin{bmatrix} 2^{-1/2} & 2^{-1/2} \\ -(2^{-1/2}) & 2^{-1/2} \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 3{,}5355 \\ -0{,}7071 \end{bmatrix}$$

onde $\mathbf{x_q}$ é o mesmo vetor $\mathbf{x_i}$ representado pela nova base $\mathbf{q_1}$ e $\mathbf{q_2}$. Neste exemplo, como $\mathbf{Q}$ é ortogonal então $\mathbf{Q}^{-1} = \mathbf{Q}^T$.

Para voltar a representar o vetor **x** pela base formada pelos vetores **i1** e **i2** realiza-se o cálculo a seguir:

$$\mathbf{x_i} = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix} = \mathbf{Q}\,\mathbf{x_q} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} \begin{bmatrix} x_{q1} \\ x_{q2} \end{bmatrix} = \begin{bmatrix} 2^{-1/2} & -(2^{-1/2}) \\ 2^{-1/2} & 2^{-1/2} \end{bmatrix} \begin{bmatrix} 3{,}5355 \\ -0{,}7071 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Pode-se utilizar uma Transformada para representar então um vetor x em relação a uma nova base formada por outro conjunto **Q** de vetores linearmente independentes $\mathbf{q_1}$ e $\mathbf{q_2}$, ortogonais ou não, do mesmo espaço vetorial $\mathbf{R^2}$. O vetor **x** será então transformado em um vetor **x'** = $[x_2 \quad y_2]^T$ cujos componentes x_2 e y_2 são a representação do vetor **x** nesta nova base Q. Pode-se escrever então:

$$\mathbf{x} = \mathbf{Q}\,\mathbf{x}' \qquad \textbf{(Eq. 5)}$$

onde **Q** é a matriz formada pelo conjunto de vetores que forma esta nova base de representação, isto é, $\mathbf{Q = [q_1 \;\; q_2]}$. Assim os novos componentes de **x'** serão dados por:

$$\mathbf{x}' = \mathbf{Q}^{-1}\,\mathbf{x} \qquad \textbf{(Eq. 6)}$$

onde $\mathbf{Q}^{-1}$ é a transformada que aplicada sobre os componentes de **x** gera os novos componentes (x_2 e y_2) de **x**, que é chamado **x'** na nova base **Q**. Os novos componentes (x_2 e y_2) correspondem as projeções de **x** sobre a nova base **Q**. Se a base **Q** for ortonormal então (GONZALEZ,1993) $\mathbf{Q}^{-1} = \mathbf{Q}^{T}$ onde $\mathbf{Q}^{-1}$ é a matriz inversa de **Q** e $\mathbf{Q}^{T}$ é a sua matriz transposta. Neste caso, então, $\mathbf{Q}^{-1}$ é a matriz composta pelos vetores da base onde cada linha representa um vetor da base.

É fundamental que a base de representação seja completa para o espaço vetorial de tamanho **n** ($\mathbf{R}^{n}$) utilizado. A matriz da base **Q**, de tamanho n x n, será completa se: *RANK (Q) = n* (CHEN,1999), pois neste caso todas as **n** colunas (n vetores) da matriz **Q** (base) são linearmente independentes. Esta condição também é satisfeita quando a matriz **Q** for *não singular*, isto é, quando o seu determinante for diferente de zero. A base necessita ser completa para permitir a representação de todos os vetores pertencentes ao espaço vetorial utilizado.

Para simplificação dos cálculos é importante também que a base de representação utilizada **Q** seja ortonormal para o espaço vetorial considerado. Isto simplifica o cálculo matemático e faz também com que os vetores que formam a base não tenham componentes entre si.

Suponha-se, agora, que $\mathbf{Q}^{-1}$ (com $\mathbf{Q}^{-1} = \mathbf{Q}^{T}$) seja uma matriz cujas linhas sejam formadas pelos autovetores da matriz de covariância de **x** ($\mathbf{C_X}$), sendo **x** uma variável aleatória, constituída pela variação dos componentes dos vetores-bloco ao longo de toda a imagem. Neste caso, como **Q** representa a nova base de representação do vetor **x,** transformando-o em **x'**, a matriz de covariância $\mathbf{C_X}$ representada em função da base **Q**, pode ser transformada (é chamada de Transformação de Similaridade em (CHEN,1999)) e representada em função da nova base **Q** por $\mathbf{C_{X'}}$ **,** de tal forma que se pode escrever:

$$\mathbf{C_X Q} = \mathbf{Q}\,\mathbf{C_{X'}} \qquad \textbf{(Eq. 7)}$$

ou seja:

$$\mathbf{C_{X'}} = \mathbf{Q}^{-1}\,\mathbf{C_X Q} \qquad \textbf{(Eq. 8)}$$

Para chegar-se a representação acima basta observar que:

$$\mathbf{C_X} = \mathbf{x}\,\mathbf{x}^{T} \qquad \textbf{(Eq. 9)}$$

$$\mathbf{C}_{\mathbf{X}'} = \mathbf{x}'\,\mathbf{x}'^{T} \qquad \textbf{(Eq. 10)}$$

Mas $\mathbf{x}' = \mathbf{Q}^{-1}\mathbf{x}$ logo:

$$\mathbf{C}_{\mathbf{x}'} = (\mathbf{Q}^{-1}\,\mathbf{x})(\mathbf{Q}^{-1}\,\mathbf{x})^{T} = (\mathbf{Q}^{-1}\,\mathbf{x})(\mathbf{Q}^{T}\,\mathbf{x})^{T} = \mathbf{Q}^{-1}\,\mathbf{x}\mathbf{x}^{T}\mathbf{Q} = \mathbf{Q}^{-1}\,\mathbf{C}_{x}\mathbf{Q}$$

Logo: $\mathbf{C}_{\mathbf{x}}\mathbf{Q} = \mathbf{Q}\mathbf{C}_{\mathbf{x}'}$ conforme (Eq. 7).

A matriz $\mathbf{C}_{\mathbf{x}'}$ é a matriz diagonal dos autovalores da matriz $\mathbf{C}_{\mathbf{x}}$ e a matriz **Q** é a matriz dos autovetores de $\mathbf{C}_{\mathbf{x}}$ (cada coluna desta matriz é um autovetor de $\mathbf{C}_{\mathbf{x}}$). Por definição (GONZALEZ,1993), os autovetores ($\mathbf{q_i}$) e os autovalores (λ_i) de uma matriz **C** , de tamanho n x n, satisfazem a relação:

$$\mathbf{C}\mathbf{q}_{\mathbf{i}} = \lambda_i \mathbf{q}_{\mathbf{i}} \qquad \textbf{(Eq. 11)}$$

$$\mathbf{C}_{\mathbf{x}} \cdot \mathbf{Q} = \mathbf{C}_{\mathbf{x}}[\mathbf{q}_1\ \mathbf{q}_2\ ....\mathbf{q}_n] = [\mathbf{q}_1\ \mathbf{q}_2\ ....\mathbf{q}_n] \begin{bmatrix} \lambda_1 & 0 & \cdots & \cdots & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ \vdots & 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & \lambda n \end{bmatrix} = \mathbf{Q} \cdot \mathbf{C}_{\mathbf{x}'} \qquad \textbf{(Eq. 12)}$$

Comparando a (Eq. 7) com a (Eq. 12) chega-se a seguinte conclusão:

$$\mathbf{C}_{\mathbf{x}'} = \begin{bmatrix} \lambda_1 & 0 & \cdots & \cdots & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ \vdots & 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & \lambda n \end{bmatrix} \qquad \textbf{(Eq. 13)}$$

A (Eq. 13) mostra então, claramente, que a Matriz de Covariância dos novos componentes $\mathbf{x}'$ é uma matriz diagonal com os valores da diagonal principal igual aos autovalores da Matriz de Covariância dos vetores **x**. Elimina-se, assim, qualquer correlação

cruzada entre as dimensões dos vetores **x**, eliminando-se, consequentemente, a *Redundância INTERPIXEL* com relação aos novos componentes.

O trabalho em (CHEN,1999) mostra que qualquer matriz real e simétrica **M** pode ser diagonalizada utilizando uma Transformação de Similaridade da forma $\mathbf{M = QDQ^T}$ ou $\mathbf{D = Q^TMQ}$ (mostrada também na (Eq. 8) mesmo que ela possua autovalores λ repetidos. Os autovalores λ serão também todos reais e positivos neste caso (RAGAB,1998).

A Transformada KARHUNEN-LOÈVE, também conhecida como Transformada HOTELLING, considera um conjunto de vetores aleatórios **x** da forma:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \qquad \textbf{(Eq. 14)}$$

e é definida pela seguinte equação matricial conforme (GONZALEZ,1993):

$$\mathbf{y = A(x - m_x)} \qquad \textbf{(Eq. 15)}$$

onde:

- **A** é a matriz onde cada linha é um autovetor da matriz de correlação $\mathbf{C_X}$ de **x** e corresponde a matriz $\mathbf{Q^{-1}}$ na (Eq. 6);
- $\mathbf{m_X}$ é o vetor que representa a média de cada componente do conjunto de vetores **x** do tipo indicado na (Eq. 14);
- $(\mathbf{x - m_X})$ representa a variação de cada componente em relação à média e corresponde ao vetor **x** na (Eq. 6);
- **y** é a representação do vetor $(\mathbf{x - m_X})$ na nova base $\mathbf{A^{-1}}$ **(**dos autovetores da matriz de covariância $\mathbf{C_x}$**)** corresponde a **x'** na (Eq. 6).

Assim, como $\mathbf{A^{-1}}$ é a nova base de representação de $(\mathbf{x\text{-}m_X})$, onde cada coluna é formada pelos autovetores da matriz de covariância $\mathbf{C_x}$, então é possível escrever da (Eq. 8) (GONZALEZ,1993):

$$\mathbf{C_y = A\, C_{x\text{-}m_x} A^{-1}} \qquad \textbf{(Eq. 16)}$$

A matriz $\mathbf{C_y}$ assim calculada resulta em uma matriz diagonal, cujos elementos da diagonal principal são os autovalores da matriz $\mathbf{C_x}$, e os demais elementos são zeros,

evidenciando que os vetores **y** não possuem correlação entre si. As matrizes $\mathbf{C_y}$ e $\mathbf{C_x}$ possuem os mesmos autovalores e autovetores(GONZALEZ,1993).

$$\mathbf{C}_Y = \begin{bmatrix} \lambda_1 & 0 & \cdots & \cdots & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ \vdots & 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & \lambda n \end{bmatrix} \qquad \textbf{(Eq. 17)}$$

Os autovalores da matriz $\mathbf{C_y}$ acima representam a variância dos componentes dos vetores **y** em relação à média de cada componente**.** Se estes elementos tem a mesma variação em relação a média, os autovalores da matriz $\mathbf{C_y}$ serão todos iguais. Os autovalores serão tanto maiores quanto maior for a dispersão dos elementos respectivos das amostras em relação a média dos mesmos. Portanto, as componentes das amostras que variam pouco em relação a sua média, possuem autovalor baixo e consequentemente, no caso de uma imagem, é possível dizer que esta componente possui baixo contraste.

A aplicação da **(Eq. 15)** resulta em um conjunto de vetores **y,** cuja média dos elementos componentes é zero ($\mathbf{m_y} = 0$) e não há correlação cruzada entre eles. Isto fica evidenciado observando o resultado da matriz de covariância $\mathbf{C_y}$,.

A reconstrução dos vetores **x** a partir dos vetores **y** criados conforme acima descrito é outro aspecto de importante análise. Calculando as linhas da matriz **A** como vetores ortonormais (GONZALEZ,1993), $\mathbf{A}^{-1} = \mathbf{A}^T$, e o vetor **x** pode ser recuperado do seu correspondente **y** utilizando a equação a seguir, obtida da **(Eq. 15)**:

$$\mathbf{x} = \mathbf{A}^T\mathbf{y} + \mathbf{m_x} \qquad \textbf{(Eq. 18)}$$

Suponha-se, contudo, que ao invés de utilizar todos os autovetores de $\mathbf{C_x}$ seja criada a matriz $\mathbf{A}_K$ dos K autovetores correspondentes aos K maiores autovalores de $\mathbf{C_x}$ (ordenando os autovalores de forma que $\lambda_1 > \lambda_2 > \lambda_3 > \ldots > \lambda_n$), resultando numa matriz transformada de ordem *K x n.* Os vetores **y** teriam então a dimensão K, e a reconstrução dada pela (Eq. 18) não reconstruiria mais a matriz **x** de forma exata. O vetor reconstruído utilizando-se a matriz $\mathbf{A}_K$ seria então dado por:

$$\mathbf{x'} = \mathbf{A}_K^T\mathbf{y} + \mathbf{m_x} \qquad \textbf{(Eq. 19)}$$

Pode ser mostrado (GONZALEZ,1993) que o Erro Quadrático Médio e_{ms} entre **x** e **x'** é dado pela expressão:

$$e_{ms} = \sum_{j=1}^{n} \lambda_j - \sum_{j=1}^{K} \lambda_j = \sum_{j=K+1}^{n} \lambda_j \qquad \textbf{(Eq. 20)}$$

A primeira parte da (Eq. 20) indica que o erro é zero se $K = n$, isto é, se todos os autovetores são utilizados na transformação. Porque os λ_i decrescem monotonicamente, a (Eq. 20) também mostra que o erro pode ser minimizado pela escolha dos K autovetores associados com os K maiores autovalores λ. Portanto a Transformada Hotelling é ótima no sentido de que minimiza o Erro Quadrático Médio entre os vetores **x** e sua aproximação **x'**.

Nos resultados acima, as matrizes que representam a média e a covariância de **x** são dadas por (SHANMUGAN,1988):

$$\mathbf{m}_x = E\{\mathbf{x}\} \qquad \textbf{(Eq. 21)}$$

$$\mathbf{C}_x = \mathbf{E}\{(\mathbf{x} - \mathbf{m}_x)(\mathbf{x} - \mathbf{m}_x)^T\} \qquad \textbf{(Eq. 22)}$$

Na (Eq. 21) e na (Eq. 22), E{arg} representa o valor esperado do argumento e é definido matematicamente pela seguinte expressão para uma variável aleatória contínua **x** (SHANMUGAN,1988) :

$$E\{x\} = \int_{-\infty}^{\infty} x \cdot f(x)dx \qquad \textbf{(Eq. 23)}$$

onde *f(x)* é a função densidade de probabilidade da variável *x*.

Para uma variável discreta aleatória a equação o valor esperado do argumento é:

$$E\{X\} = \sum_{i=1}^{n} x_i P(X = x_i) \qquad \textbf{(Eq. 24)}$$

onde ***P(X)*** é a função Distribuição de Probabilidade da variável discreta **x**.

Como as funções Densidade de Probabilidade e Distribuição de Probabilidade são geralmente desconhecidas, utilizam-se algumas aproximações para o cálculo deste valor esperado. No caso de variáveis discretas, por exemplo, o cálculo da média e da covariância pode ser aproximado pelas seguintes equações (considera-se que todas as ocorrências da variável aleatória têm a mesma probabilidade de ocorrência):

$$\mathbf{m}_x = \frac{1}{M} \sum_{k=1}^{M} \mathbf{x}_k \qquad \textbf{(Eq. 25)}$$

$$\mathbf{C_x} = \frac{1}{M}\sum_{k=1}^{M}(\mathbf{x_k}\cdot\mathbf{x_k}^T) - \mathbf{m_x}\cdot\mathbf{m_x}^T \qquad \textbf{(Eq. 26)}$$

Nas bibliografias referenciadas em (HAYKIN,1999) e (LINDQUIST,1989) encontram-se duas abordagens matemáticas complementares para a análise tanto da KLT Discreta quanto da KLT Contínua.

Graficamente, pode-se entender o cálculo da KLT como um processo de mudança de base de representação dos vetores-bloco da imagem, de uma base ortonormal original para uma nova base ortonormal formada pelos autovetores da matriz de covariância desses vetores-bloco. A base original contém os valores de PIXEL da imagem como projeções ou componentes dos vetores-bloco sobre essa base. Na base dos autovetores, cada autovetor está posicionado na direção da maior variância dos vetores-bloco, entre todas as direções possíveis dentro do espaço vetorial considerado. A Figura 4 apresenta um exemplo de dois vetores-bloco ($\mathbf{V_1}$ e $\mathbf{V_2}$) formados por 2 PIXELS (espaço vetorial de 2 dimensões), isto é, $\mathbf{V_1}$ é composto por 2 valores de PIXEL (V_{1x} e V_{1y}) e $\mathbf{V_2}$ é composto também por 2 valores de PIXEL (V_{2x} e V_{2y}). A base ortonormal original é composta pelos vetores **i** (na direção "x") e **j** (na direção "y").

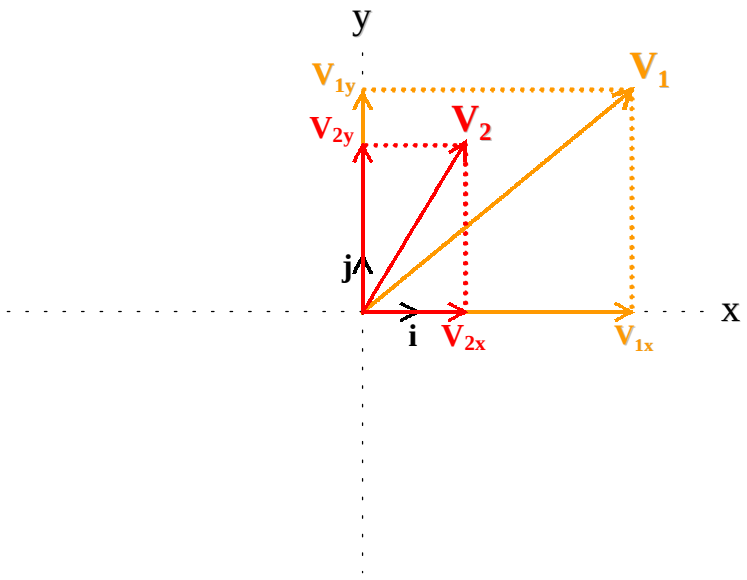


Figura 4 – Representação de 2 vetores-bloco sobre uma base Ortonormal original

A direção de maior variância entre os vetores-bloco $\mathbf{V_1}$ e $\mathbf{V_2}$ localiza-se entre as direções desses 2 vetores, que na Figura 5 está representado pela direção " *x'* ", onde estará

situado o primeiro autovetor **q** da nova base dos autovetores. O segundo autovetor da nova base será ortogonal ao primeiro, e está representado Figura 5 pelo vetor **v** na direção " y' ".

Na Figura 6 observa-se que os vetores-bloco $\mathbf{V_1}$ e $\mathbf{V_2}$ são então decompostos na nova base dos autovetores **q** e **v**, fazendo com que os novos componentes sobre a base **q** possuam valores muito superiores aos componentes sobre o autovetor **v**, situação que ocorre quanto mais ajustado o vetore-bloco se apresenta em relação ao autovetor **q**. Na Figura 6 isto é verificado pelo fato dos componentes $V_{1x'}$ e $V_{2x'}$ possuirem um tamanho bem superior ao dos componentes $V_{1y'}$ e $V_{2y'}$. Estes, por sua vez, são bem inferiores aos seus correspondentes componentes originais V_{1y} e V_{2y} obtidos a partir da base original **i** e **j**.

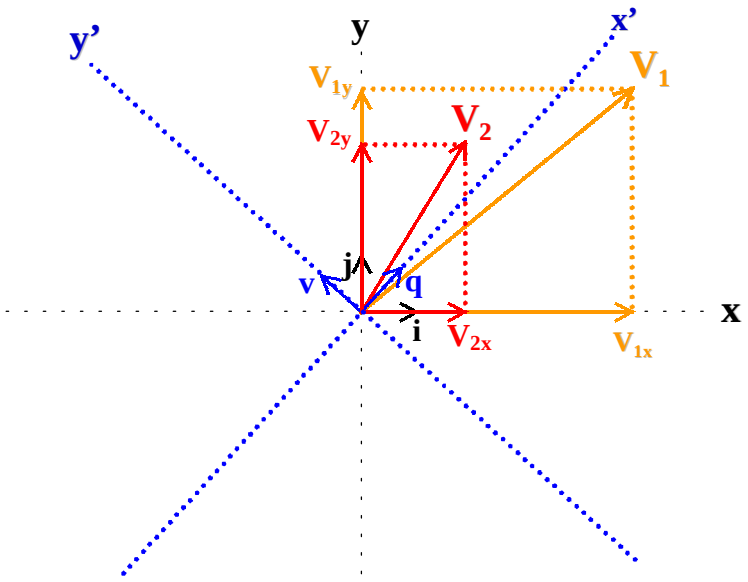


Figura 5 – Posição da nova base dos autovetores ("p" e "q")

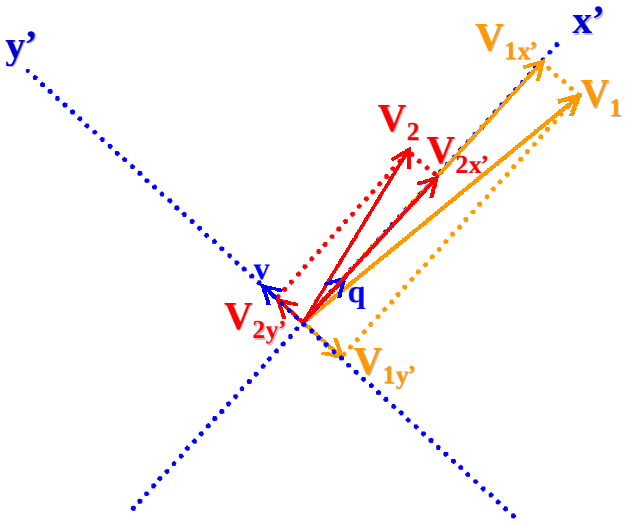


Figura 6 – Decomposição dos Vetores-bloco na nova base dos autovetores

O fato dos componentes obtidos pela decomposição dos vetores-bloco sobre o autovetor **v** possuirem um valor inferior aos obtidos sobre o autovetor **q**, permite implementar a redução dimensional, processo utilizado em conjunto com a redução da redundância de código permitindo um amento da compressão da imagem. Esta redução dimensional está mostrada na Figura 7, onde o autovetor **v** foi completamente eliminado, sobrando apenas o vetor **q**. Neste caso, a única referência para reconstrução dos vetores-bloco originais $\mathbf{V_1}$ e $\mathbf{V_2}$ corresponde aos componentes $V_{1x'}$ e $V_{2x'}$, que passam a se constituir na aproximação daqueles vetores e base para a reconstrução dos memos.

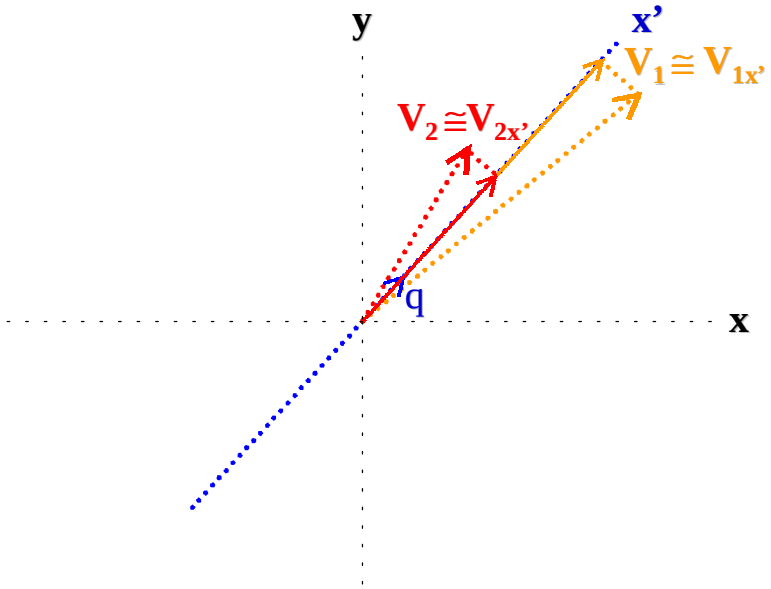


Figura 7 – Redução Dimensional pela eliminação do autovetor "v"

A Figura 8 mostra a reconstrução dos componentes originais a partir das aproximações dos vetores $\mathbf{V_1}$ e $\mathbf{V_2}$ na direção do único autovetor **q**. Nota-se claramente que a eliminação do autovetor **v**, provoca pequenas alterações em todos os componentes originais, que serão tanto maiores quanto maiores forem as diferenças entre os vetores-bloco e o autovetor **q**, ou seja, quanto menos estacionários forem os componentes dos vetores-bloco, considerados neste caso como uma variável aleatória.

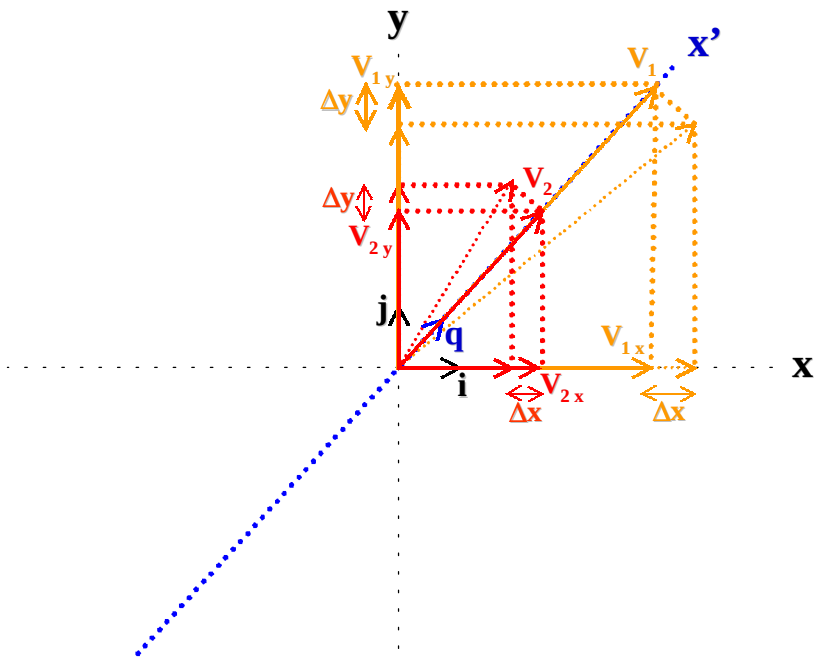


Figura 8 – Reconstituição dos Vetores-base após a redução dimensional

3.2.1.2 Estratégia de Cálculo dos Autovetores e Autovalores

A estratégia de praticamente todas as rotinas modernas de cálculo de autovetores e autovalores utilizam seqüências de transformações de similaridade para transformar uma matriz **R** numa matriz diagonal (PRESS,1994). No trabalho de (HAYKIN,1999) é definida uma *transformação de similaridade* sobre a matriz de correlação **R** como:

$$\mathbf{Q}^{-1}\mathbf{R}\mathbf{Q} = \boldsymbol{\lambda}, \text{ onde } \boldsymbol{\lambda} \text{ é a matriz diagonal dos autovalores de } \mathbf{R} \qquad \textbf{(Eq. 27)}$$

A matriz resultante da transformação de similaridade possui os mesmos autovalores da matriz original, ou seja:

$$\det\left|\mathbf{Q}^{-1}\mathbf{R}\mathbf{Q} - \lambda\mathbf{I}\right| = \det\left|\mathbf{Q}^{-1}(\mathbf{R} - \lambda\mathbf{I})\mathbf{Q}\right| = \det\left|\mathbf{Q}\right|\det\left|\mathbf{R} - \lambda\mathbf{I}\right|\det\left|\mathbf{Q}^{-1}\right| = \det\left|\mathbf{R} - \lambda\mathbf{I}\right|$$

A estratégia de cálculo dos autovetores e autovalores consiste em aplicar sucessivas transformações de similaridade sobre a matriz de correlação **R** transformando-a numa matriz diagonal. Quando isso acontecer os elementos da diagonal serão os autovalores da matriz resultante da transformação de similaridade que são também iguais aos autovalores da matriz **R**, e os autovetores serão dados pelo produto das matrizes ortogonais que foram utilizadas para a transformação. Assim, dadas as matrizes ortogonais $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$,...., estas são aplicadas

na forma de uma transformação de similaridade de forma a diagonalizar a matriz de correlação **R**, em sucessivas operação da forma:

$$\begin{aligned}\mathbf{R} \;&\rightarrow\; \mathbf{P}_1^{-1}\cdot\mathbf{R}\cdot\mathbf{P}_1 \;\rightarrow\; \mathbf{P}_2^{-1}\cdot\mathbf{P}_1^{-1}\cdot\mathbf{R}\cdot\mathbf{P}_1\cdot\mathbf{P}_2 \\ &\rightarrow\; \mathbf{P}_3^{-1}\cdot\mathbf{P}_2^{-1}\cdot\mathbf{P}_1^{-1}\cdot\mathbf{R}\cdot\mathbf{P}_1\cdot\mathbf{P}_2\cdot\mathbf{P}_3 \;\rightarrow\; \text{etc.}\end{aligned} \qquad \textbf{(Eq. 28)}$$

No final dessa operação, os autovalores estarão presentes na matriz diagonal resultante das sucessivas transformações de similaridade aplicadas sobre **R**, enquanto a matriz **Q** dos autovetores será obtida de:

$$\mathbf{Q} = \mathbf{P}_1\cdot\mathbf{P}_2\cdot\mathbf{P}_3\cdot\ldots$$

È importante ressaltar que para matrizes reais e simétricas, os autovetores são reais e ortonormais e portanto a matriz de transformação $\mathbf{P}_j$ é ortogonal. A transformação de similaridade é também uma transformação ortogonal.

Duas técnicas utilizando a transformação de similaridade foram implementadas no trabalho (PRESS,1994): a *Transformação de Jacobi de uma Matriz Simétrica* e a combinação da técnica de *Redução de uma Matriz para a Forma Tridiagonal (HOUSEHOLDER REDUCTIONS)* com o *Algoritmo QL* de cálculo dos autovalores e autovetores da matriz tridiagonal.

3.2.1.3 O Método de JACOBI

A Transformação de Jacobi (PRESS,1994) consiste na aplicação de transformações de similaridade **P** do tipo mostrado na (Eq. 28) sobre uma matriz. Cada aplicação, que é chamada de *rotação de Jacobi,* representa apenas uma rotação de plano e contribui para o zeramento de um elemento da matriz situado fora da sua diagonal principal. Aplicando essas rotações a cada elemento acima da diagonal principal, para o caso de uma matriz simétrica, anula-se também o correspondente elemento abaixo da diagonal principal. Porém os zeros calculados por uma rotação anterior são modificados pelas rotações dos pontos subsequentes localizados na mesma linha ou coluna, e são necessárias, por isso, várias seqüências de rotações sobre a matriz, para tornar os elementos fora da diagonal principal iguais a zero até alcançar, como já explicado antes, uma matriz diagonal, que é a Matriz dos Autovalores. A multiplicação das transformações de similaridade $\mathbf{P}_{pq}$ aplicadas sucessivamente à matriz original fornece a sua Matriz dos Autovetores. O *Método de Jacobi* sempre funciona para matrizes reais e simétricas.

Uma *rotação de Jacobi* básica é dada por:

$$P_{pq} = \begin{bmatrix} 1 & & & & & & \\ & \cdots & & & & & \\ & & c & \cdots & s & & \\ & & \vdots & 1 & \vdots & & \\ & & -s & \cdots & c & & \\ & & & & & \cdots & \\ & & & & & & 1 \end{bmatrix}$$

Nesta matriz, os elementos não incluídos são todos zero. Os números c e s são, respectivamente, o coseno e o seno de um ângulo de rotação ϕ tal que: $c^2 + s^2 = 1$.

Uma rotação de Jacobi $\mathbf{P}_{pq}$ aplicada sobre uma matriz **R** (N x N) produz uma matriz $\mathbf{R}'$ definida por:

$$\mathbf{R}' = \mathbf{P}_{pq}^{T} \cdot \mathbf{R} \cdot \mathbf{P}_{pq} \qquad \textbf{(Eq. 29)}$$

A operação $\mathbf{P}_{pq}^{T} \cdot \mathbf{R}$ modifica somente as linhas *p* e *q* da matriz **R** enquanto a operação seguinte, $\mathbf{R} \cdot \mathbf{P}_{pq}$ modifica apenas as colunas *p* e *q*. Os índices *p* e *q*, neste caso, representam as linhas e colunas afetadas pela operação.

A equação (Eq. 30) mostra a matriz $\mathbf{R}'$, resultante da operação descrita anteriormente pela (Eq. 29), onde então somente os elementos situados nas linhas e colunas *p* e *q* são afetados.

$$R' = \begin{bmatrix} & \cdots & a'_{1p} & \cdots & a'_{1q} & \cdots & \\ \vdots & & \vdots & & & & \vdots \\ a'_{p1} & \cdots & a'_{pp} & \cdots & a'_{pq} & \cdots & a'_{pN} \\ \vdots & & \vdots & & \vdots & & \vdots \\ a'_{q1} & & a'_{qp} & \cdots & a'_{qq} & \cdots & a'_{qN} \\ \vdots & & \vdots & & \vdots & & \vdots \\ & \cdots & a'_{Np} & \cdots & a'_{Nq} & \cdots & \end{bmatrix} = \mathbf{P}_{pq}^{T} \cdot \mathbf{R} \cdot \mathbf{P}_{pq} \qquad \textbf{(Eq. 30)}$$

O objetivo da aplicação da *rotação de Jacobi* é tornar o elemento localizado na linha *p* e coluna *q* (a'_{pq}) da matriz $\mathbf{R}'$ igual a zero. Pela simetria existente na Matriz $\mathbf{R}'$ também o elemento a'_{qp} será automaticamente anulado.

Após a realização do produto da (Eq. 29) e após uma manipulação algébrica (PRESS,1994) chega-se ao conjunto de equações para a aplicação do algoritmo numérico:

$$a'_{pq} = 0$$
$$a'_{pp} = a_{pp} - ta_{pq}$$
$$a'_{qq} = a_{qq} + ta_{pq}$$
$$a'_{rp} = a_{rp} - s(a_{rq} + \tau a_{rp})$$
$$a'_{rq} = a_{rq} + s(a_{rp} - \tau a_{rq})$$

$$\text{onde:} \quad \theta \equiv \cot 2\phi \equiv \frac{c^2 - s^2}{2sc} = \frac{a_{qq} - a_{pp}}{2a_{pq}}$$

$$t = \frac{\text{sgn}(\theta)}{|\theta| + \sqrt{\theta^2 + 1}}, \text{ onde "sgn" é o sinal de "}\theta\text{"}$$

$$c = \frac{1}{\sqrt{t^2 + 1}}$$

$$s = tc$$

$$\tau \equiv \frac{s}{1 + c} = \tan(\phi/2)$$

A convergência do *Método de Jacobi* é testada pela soma dos quadrados dos elementos fora da diagonal principal, ou seja:

$$S = \sum_{r \# s} |a_{rs}|^2$$

Uma vez que a transformação é ortogonal a soma dos quadrados da diagonal principal aumenta por um valor igual ao dobro do quadrado do termo anulado a_{pq}, o que equivale a dizer que:

$$S' = S - 2|a_{pq}|^2$$

Embora o algoritmo de Jacobi original, elaborado em 1846, eliminasse primeiramente o maior elemento da matriz triangular superior, ideal porque eliminava os maiores valores do cálculo (já que o objetivo é aproximar de zero os elementos fora da diagonal principal da matriz original), computacionalmente só essa busca do maior valor consome muito tempo de processamento. Assim, o *Método Cíclico de Jacobi* é utilizado geralmente, onde cada elemento é eliminado na ordem estrita do seu posicionamento na matriz, sucessivamente linha a linha, de cima a baixo da matriz triangular superior: $\mathbf{P}_{12}$,$\mathbf{P}_{13}$,...,$\mathbf{P}_{1N}$; então $\mathbf{P}_{23}$,$\mathbf{P}_{24}$,...,$\mathbf{P}_{2N}$; e assim sucessivamente. Cada matriz $\mathbf{P}_{pq}$ chama-se de uma *Rotação de Jacobi*. O conjunto acima, aplicado sobre toda a matriz apresenta *N(N-1)/2 rotações de Jacobi*. Esse conjunto,

aplicado uma vez sobre a matriz original, é chamado de *uma varredura*. Tipicamente são necessárias 6 a 10 varreduras para atingir a convergência, mas para matrizes de tamanho acima de N=10, o algoritmo apresenta uma ineficiência computacional que faz cair em aproximadamente 5 vezes sua performance na realização do cálculo quando comparado ao Método QL, também apresentado neste trabalho.

Cada *rotação de Jacobi* consome 4N operações, cada uma consistindo de uma multiplicação e uma adição. Considerando que são necessárias geralmente de 6 a 10 *varreduras* para atingir a convergência, tem-se aproximadamente $3N^2$ a $5N^2$ *rotações de Jacobi* , o que resulta num esforço computacional da ordem de $12N^3$ a $20N^3$ operações.

Foi implementado neste trabalho o Método de Jacobi para o cálculo da inversa da matriz de correlação **R** dos PIXEL dos blocos da imagem, a partir da rotina *"jacobi.c"* apresentada em (PRESS,1994) e que foi incluída no programa de implementação (Seção 9).

Os autovalores que surgem a partir da aplicação do *Método de Jacobi* não estão ordenados. Para ordená-los foi incluída a *Rotina de Inserção Direta "eigsrt.c"* após a rotina "*jacobi.c*", introduzindo neste processo um custo adicional de computação da ordem de N^2 operações.

3.2.1.4 O Método HOUSEHOLDER

Outra estratégia para o cálculo dos autovalores e autovetores de uma matriz consiste em, num primeiro momento, reduzir a matriz a uma forma mais simples que, para o caso de matrizes simétricas (como a matriz de correlação, por exemplo), é reduzi-la a uma forma tridiagonal com um número finito de passos (ao contrário do *Método de Jacobi* que requer um número variável de iterações para convergir diretamente para uma matriz diagonal) e, num segundo momento, aplicar o processo iterativo sobre a matriz tridiagonal para atingir a diagonalização completa da matriz.

O algoritmo HOUSEHOLDER (PRESS,1994) reduz uma matriz simétrica **R** de NxN (no caso deste trabalho **R** é a matriz de correlação) numa matriz tridiagonal através de (N - 2) transformações ortogonais. Cada transformação, a partir de um elemento da matriz, anula seletivamente todos os elementos da coluna e da linha correspondente com exceção do próprio elemento.

O componente básico deste método é a matriz HOUSEHOLDER **P**. Esta matriz é definida como:

$$\mathbf{P} = \mathbf{I} - 2\mathbf{w} \cdot \mathbf{w}^T$$

onde **w** é um vetor real tal que $|\mathbf{w}|^2 = 1$.

Dada essa definição tem-se que $\mathbf{P}^2 = \mathbf{I}$ logo $\mathbf{P} = \mathbf{P}^{-1}$. Como $\mathbf{P}^T=\mathbf{P}$ então $\mathbf{P}^T=\mathbf{P}^{-1}$. A matriz HOUSEHOLDER pode ser redefinida como:

$$\mathbf{P} = \mathbf{I} - \frac{\mathbf{u} \cdot \mathbf{u}^T}{H}$$

onde:

$$H = \frac{1}{2}|\mathbf{u}|^2$$

e:

$$\mathbf{u} = \mathbf{x} \mp |\mathbf{x}|\mathbf{e}_1$$

sendo que $\mathbf{e}_1$= [1,0,0,...,0]^T e o sinal é definido como igual ao sinal original do único elemento da coluna que não é anulado no cálculo (visando minimizar erros de arredondamento). Pode-se escrever então:

$$\mathbf{P} \cdot \mathbf{x} = \mathbf{x} - \mathbf{u} = \pm|\mathbf{x}|\mathbf{e}_1 \qquad \textbf{(Eq. 31)}$$

A (Eq. 31) mostra que a matriz HOUSEHOLDER aplicada sobre uma coluna da matriz **R**, zera todos os componentes da coluna com exceção do primeiro elemento que assume o valor ±|**x**|.

A seqüência de transformações de similaridade aplicadas sobre uma matriz **R,** para transformá-la numa matriz tridiagonal, inicia com a escolha do vetor **x** a partir dos (N-1) elementos da primeira coluna da matriz **R**. Após a aplicação da matriz **P** os (N-2) elementos de **x** serão anulados. Pode-se escrever:

$$P_1 \cdot R = \left[\begin{array}{c|ccc} 1 & 0 & 0 & \cdots & 0 \\ \hline 0 & & & & \\ 0 & & {}^{(N-1)}\mathbf{P}_1 & & \\ \vdots & & & & \\ 0 & & & & \end{array}\right] \cdot \left[\begin{array}{c|ccc} a_{11} & a_{12} & a_{13} & \cdots & a_{1N} \\ \hline a_{21} & & & & \\ a_{31} & & \text{Não relevante} & & \\ \vdots & & & & \\ a_{N1} & & & & \end{array}\right]$$

$$P_1 \cdot R = \left[\begin{array}{c|cccc} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ \hline k & & & & \\ 0 & & \text{Não relevante} & & \\ \vdots & & & & \\ 0 & & & & \end{array}\right]$$

Neste caso, $^{(N-1)}\mathbf{P}$ é a matriz HOUSEHOLDER com dimensões (N-1) x (N-1). O valor k é igual a mais ou menos o módulo do vetor $[a_{21}, ...,a_{N1}]^{T}$. A transformação de similaridade completa neste caso é:

$$\mathbf{A}' = \mathbf{P}^{T} \cdot \mathbf{A} \cdot \mathbf{P} = \mathbf{P} \cdot \mathbf{A} \cdot \mathbf{P} = \left[\begin{array}{c|ccc} a_{11} & k & 0 & \dots & 0 \\ \hline k & & & & \\ 0 & & \text{Não relevante} & & \\ \vdots & & & & \\ 0 & & & & \end{array}\right]$$

Este procedimento é, então, novamente realizado, com a criação da nova matriz HOUSEHOLDER $^{(N-2)}\mathbf{P}$ sobre a matriz transformada $\mathbf{A}'$, e assim sucessivamente até que a matriz inicial **R** se transforme numa matriz tridiagonal. A quantidade total de operações necessárias para a redução da matriz **R** é de $4N^{3}/3$.

Este trabalho também implementa o cálculo dos autovalores e autovetores da matriz de correlação R utilizando o Método de HOUSEHOLDER em conjunto com o Método QL que será analisado na próxima seção. A rotina implementada no programa para implementar esse método foi a “tred2.c” (PRESS,1994).

3.2.1.5 O Método QL

A idéia básica do Método QL (PRESS,1994) considera inicialmente que qualquer matriz **A** pode ser decomposta na forma:

$$\mathbf{A}_s = \mathbf{Q}_s \cdot \mathbf{L}_s \qquad \textbf{(Eq. 32)}$$

onde $\mathbf{Q_s}$ é uma matriz ortogonal e $\mathbf{L_s}$ é uma matriz triangular à esquerda.

Pode-se montar a (Eq. 32) em ordem inversa, tal que:

$$\mathbf{A}_{s+1} = \mathbf{L}_s \cdot \mathbf{Q}_s \qquad \textbf{(Eq. 33)}$$

Isolando a matriz **L** na (Eq. 32) e substituindo na (Eq. 33) tem-se:

$$\mathbf{A_{s+1}} = \mathbf{L}_s \cdot \mathbf{Q}_s = \mathbf{Q}_s^T \cdot \mathbf{A}_s \cdot \mathbf{Q}_s \qquad \textbf{(Eq. 34)}$$

A (Eq. 34) mostra que a matriz $\mathbf{A}_{s+1}$ é uma transformação ortogonal da matriz $\mathbf{A}_s$. O seguinte teorema (PRESS,1994) é a base do algoritmo QL para uma matriz geral **A**: (i) $\mathbf{A}_s \rightarrow$ [forma triangular à esquerda] quando $s \rightarrow \infty$, se **A** tem autovalores de diferentes valores absolutos $|\lambda_i|$, e os autovalores aparecem na diagonal principal em ordem crescente de valor; (ii) $\mathbf{A}_s \rightarrow$ [forma triangular à esquerda] quando $s \rightarrow \infty$, se **A** tem autovalores de diferentes valores absolutos $|\lambda_i|$, e os autovalores aparecem na diagonal principal em ordem crescente de valor; exceto para o bloco matriz diagonal de ordem *p*, cujos autovalores tendem a λ_i, quando **A** apresenta um autovalor $|\lambda_i|$ de multiplicidade *p*.

Tendo em vista este teorema e para uma matriz tridiagonal **A**, o algoritmo QL tem uma melhor performance quando se utiliza o método da *Rotação de Jacobi* para transformar a matriz tridiagonal **A** numa matriz triangular à esquerda. Neste caso utiliza-se a seqüência $\mathbf{P}_{12}$, $\mathbf{P}_{23}$,.....,$\mathbf{P}_{n-1,n}$ sobre a matriz tridiagonal **A**, o que a torna, neste caso, devido à sua simetria, diretamente uma matriz diagonal. Assim, tendo em vista a definição da (Eq. 32) obtém-se:

$$\mathbf{Q_s^T} = \mathrm{P}_1^{(s)} \cdot \mathrm{P}_2^{(s)} \cdots \mathrm{P}_{n-1}^{(s)} \qquad \textbf{(Eq. 35)}$$

onde $\mathbf{P}_i$ anula $a_{i,i+1}$.

O número de operações (complexidade) do método QL quando aplicado a uma matriz tridiagonal, considerando o cálculo de todos os autovalores e autovetores é de $3N^3$.

Na implementação deste trabalho, utilizou-se a rotina "*tqli.c*" logo após a rotina "*tred2.c*" para realizar o cálculo dos autovalores e autovetores. Esta rotina, juntamente com os demais módulos implementados encontra-se na Seção 9 deste trabalho.

3.2.2 APLICAÇÕES GERAIS DA KLT

Uma das aplicações na qual a KLT já é utilizada como padrão é na identificação de objetos numa imagem, devido a ótima capacidade desta transformada em orientar os autovetores da matriz de correlação dos blocos de imagem (vetores-bloco) na direção de maior variância destes. Em uma imagem (GONZALEZ,1993), por exemplo, um objeto qualquer pode ser representado por um conjunto de pontos num plano bidimensional, onde cada ponto representa um vetor (uma amostra) **x** de componentes (x_1, x_2). Ao ser aplicada a Transformada KARHUNEN-LOÈVE ao conjunto de vetores (amostras) **x**, obtém-se um

conjunto de vetores (amostras) **y**, onde cada amostra será composta de componentes (y_1, y_2), de tal forma que a média dos componentes y_1 dos diversos vetores (amostras) **y** será igual a zero, o mesmo ocorrendo com a média dos componentes y_2 das diversas amostras. Assim, tem-se a figura em questão posicionada em torno do seu centro geométrico e estes novos pontos da figura (vetores **y)** ficariam simetricamente distribuídos em relação a sua nova base de representação (aos seus autovetores). Esta técnica é utilizada para a identificação de objetos em uma imagem, pois um padrão da figura é armazenado já considerando esse aspecto de simetria, e os objetos a serem identificados, independentemente de sua posição física (translação ou rotação) na imagem, após aplicada a Transformada KL ficariam na mesma posição do padrão, facilitando o algoritmo de comparação e identificação do objeto. Neste tipo de aplicação a KLT atua como uma transformação de rotação (GONZALEZ,1993) que alinha os dados com os autovetores. Este alinhamento é precisamente o mecanismo que descorrelaciona os dados. Além disso, os autovalores λ_i que aparecem na diagonal principal da matriz de correlação $\mathbf{C_y}$ representam a variância do componente y_i em relação ao autovetor $\mathbf{q_i}$.

No trabalho apresentado em (UENOHARA,1998) foi apresentada uma técnica de identificação de imagens na qual foi provado que a aproximação ótima de um conjunto de imagens giradas uniformemente a partir de uma imagem padrão original é dada pelos vetores bases para a DCT da imagem original em coordenadas polares. Embora estes vetores bases possam também ser obtidos pela KLT das imagens giradas uniformemente, a DCT torna possível o cômputo dos vetores bases de forma muito mais eficiente. Esta técnica pode proporcionar um aumento na performance dos algoritmos de aproximação da KLT no reconhecimento de imagens.

No trabalho apresentado em (MALINA,2001) é também apresentada uma abordagem importante nas aplicações de identificação de imagem, através da generalização do Critério de FISHER multiclasse e sua relação com a metodologia proposta na Transformada KARHUNEN-LOÈVE.

Outros artigos sobre aplicações desta técnica na segmentação de imagens incluem a utilização da Transformada KL na identificação de tumores de pele (UMBAUGH,1993), reconhecimento de alvos móveis em vídeo comprimido (SEALES,1997), procedimentos de classificação após a segmentação de caracteres de placas de automóveis (HEGT,1998), e muitos outras aplicações.

Como exemplo de aplicação em compressão de imagens multi-espectrais, pode-se ter um conjunto de 3 imagens da mesma cena, onde cada imagem seria, por exemplo, criada com

câmeras de sensores diferentes (infra-vermelho, ultra-violeta e raio-X, por exemplo) e teria dimensões de, por exemplo, 120 pontos x 300 pontos (total de 36.000 pontos ou amostras). O conjunto sob análise, neste caso, é composto por 36.000 vetores **x.** Cada vetor x, neste caso, é composto por 3 componentes, onde cada componente representa o valor da intensidade deste ponto em cada imagem. O vetor **x**, portanto, seria representado por 3 componentes (x_1, x_2, x_3). Ao se aplicar a KLT neste caso se encontraria o conjunto de vetores **y**, de componentes (y_1, y_2, y_3) em que, novamente, a média dos componentes y_1, por exemplo, dos 36.000 pontos (amostras) da imagem é zero. Se, neste caso, for desprezada a imagem formada pelos pontos y_i correspondentes aos autovalores que possuem valores baixos (baixo contraste), se poderá diminuir sensivelmente o número de imagens necessárias para representar as imagens originais, sem perda significativa do conteúdo da imagem. Esta técnica poderia ser utilizada para a compressão de imagens, pois menos pontos seriam necessários para representá-la e/ou transmiti-la, recuperando a mesma em etapa posterior com o mínimo de perda visual. Esta técnica foi utilizada em (TINTRUP,1998) na descorrelação das imagens multiespectrais gerados por satélite LANDSAT pelo sistema Thematic Mapper (TM) e posterior processamento comparando as técnicas de compressão por Quantificação Vetorial, DCT-JPEG e WAVELET. O mesmo tipo de aplicação foi também utilizada em (SHEN,1993) no seu trabalho sobre o Efeito da Compressão Espectral na Exploração por Máquina.

As aplicações da Transformada KL na compressão de imagens serão descritas na próxima seção deste trabalho. Os artigos apresentados na revisão de literatura contêm muitas informações sobre técnicas de compressão de imagens utilizando a Transformada KARHUNEN-LOÈVE e algumas serão objeto de análise mais detalhada.

3.2.3 APLICAÇÕES DA KLT NA COMPRESSÃO DE IMAGENS

Existem muitos artigos publicados sobre compressão de imagens, devido a grande aplicabilidade das teorias e estudos sobre a KLT. Nesta seção foram selecionados e comentados os resultados dos artigos que mais se relacionam com o objetivo deste trabalho.

O trabalho de (RAGAB,1998) comparou 5 transformadas para tentar mostrar a de melhor performance para a compressão de imagens. Em seu trabalho, utilizou os parâmetros *Razão de Compressão (CR)* e *Razão Sinal/Ruído de Pico (PSNR),* da mesma forma que foram definidos neste trabalho. Seu objetivo foi comparar a eficiência das transformadas analíticas na compressão de imagens, realizando essa comparação para a *Transformada KARHUNEN-*

LOÈVE (KLT), Transformada Discreta do Coseno (DCT), Transformada Discreta HARTLY (DHT), Transformada Discreta GABOR (DGT) e Transformada WAVELET Discreta (DWT).

O processo de compressão apresentado em (RAGAB,1998) utilizou o mesmo modelo do Sistema de Compressão utilizado neste trabalho para a KLT. Para a DCT, (RAGAB,1998) acrescentou uma quantização adicional para o coeficiente DC e para os coeficientes AC nos quais o elemento DC participa no cálculo. Este procedimento difere do adotado neste trabalho, onde não se utilizou essa quantização adicional, trabalhando com a transformada da forma como é apresentada em (PRESS,1994). Na computação realizada em (RAGAB,1998), foi utilizado o algoritmo da FFT para N pontos para calcular os coeficientes da DCT de 2 dimensões, aplicando o algoritmo da DCT de uma dimensão nas linhas e, sucessivamente, nas colunas da imagem, da mesma forma como realizado neste trabalho.

A Transformada WAVELET de 2 dimensões (2-D) utilizada por (RAGAB,1998), foi obtida mediante a aplicação da Transformada DWT de uma dimensão (1-D) às linhas e, sucessivamente, às colunas da imagem, exatamente como também realizado neste trabalho. O trabalho de (RAGAB,1998) ampliou a análise dos resultados calculando a DWT para 4 bases diferentes: Haar, Daub4, Daub6 e Daub8. Neste trabalho utilizou-se apenas a base Daub4, mas analisando os resultados obtidos em (RAGAB,1998) percebe-se que esta base é a que proporciona o melhor resultado na *PSNR* e um resultado apenas 0,4 % inferior na *CR* com relação as outras três bases DWT, e foi, também por essa razão, a base utilizada para a DWT neste trabalho. Para a DWT, o trabalho em (RAGAB,1998) realiza o cálculo sem blocagem.

A imagem padrão (LENNA) com 512 x 512 PIXELS e 256 níveis de cinza foi utilizada no trabalho em (RAGAB,1998) para testar a eficiência das transformadas analíticas. Quatro tamanhos de bloco diferentes foram selecionados em (RAGAB,1998), a saber: 4x4, 8x8, 16x16 e 32x32 (por limitações no HARDWARE utilizado). Os resultados apresentados (CR e PSNR) foram obtidos com a escolha de coeficientes por THRESHOLD com limites de $T_h = 0,1$ (para *KLT* e *DWT*) e $T_h = 0,001$ (para *DCT*) para os quatro tamanho de blocos.

O trabalho de (RAGAB,1998) forneceu uma excelente sistemática de procedimentos que foram aproveitados neste trabalho. Alguns aspectos, porém, dificultaram uma análise comparativa mais efetiva dos seus resultados, quais sejam:

- A utilização de um *THRESHOLD* (T_h), calculado como uma percentagem dos valores máximo e mínimo para cada bloco de imagem, pode resultar numa quantidade diferente de coeficientes para cada transformada prejudicando a análise para um mesmo número de coeficientes que se pretende conservar;

- a utilização de um THRESHOLD diferente para a DCT prejudica essa comparação, e não foi justificada a relação entre essa mudança e seu efeito no número de coeficientes mantidos e na comparação dos resultados;
- não foi testada a DWT blocada nem a DCT em tela inteira para comparação com as demais;
- os tempos de processamento não foram incluídos no quadro comparativo;
- não foram mencionados os algoritmos utilizados para o cálculo da KLT;

As conclusões importantes do trabalho de (RAGAB,1998) foram:

- Para efeito de compressão de imagem não existe uma transfomada única que seja a melhor em todos os testes realizados para os parâmetros *CR* e *PSNR*;
- em se tratando apenas de *PSNR*, a DHT $T_h=10^{-3}$ (*PSNR=43,7 dB)* e a DGT $T_h=10^{-2}$ (*PSNR=39 dB)* com tamanho de bloco 4x4 são as que forneceram os melhores resultados;
- em se tratando apenas de *CR*, a DCT $T_h=10^{-3}$ (*CR=7,2:1*) com tamanho de bloco de 32x32 PIXEL foi a que apresentou o melhor resultado;
- foi ressaltado a simplicidade da DWT com um menor tempo de processamento (embora não o tenha divulgado).

O trabalho de (OLIVEIRA,2000) também apresentou resultados de um estudo comparativo entre a técnica de compressão utilizando o padrão JPEG, a Análise dos Componentes Principais (PCA ou KLT), e uma Rede Neural PCA, que utiliza a técnica de Redes Neurais para a obtenção dos coeficientes da KLT.

A comparação com a DCT foi realizada através da utilização da rotina padrão JPEG, assim como definida pela ISO e CCITT. A KLT foi implementada com a mesma técnica utilizada neste trabalho.

A Rede Neural utilizada foi uma rede de realimentação composta de uma camada simples de neurônios lineares sob a regra Hebbiana de aprendizagem modificada, modelo referenciado em seu trabalho.

Nos experimentos realizados por (OLIVEIRA,2000) para a comparação entre as três técnicas mencionadas foi utilizado um conjunto de 10 imagens médicas em *escala de cinza* com uma definição de 640 X 480 pixels do fígado humano, obtidas por meio de um microscópio a laser. Foram utilizados blocos de imagem de 32 X 32 PIXELS por apresentarem, em testes experimentais realizados, a mais alta taxa de compressão com uma boa performance em comparação a utilização de blocos de 8 X 8, 16 X 16 e 32 X 32 PIXELS. Foram utilizados 3 componentes principais no experimento de (OLIVEIRA,2000). Para

avaliar quantos componentes principais (quantos autovalores) deveriam ser utilizados foram feitos testes experimentais, com blocos de 32 X 32 pixels, utilizando-se o método PCA clássico e variando o número de componentes principais, sucessivamente, de 1 a 32 e calculando-se o Erro Quadrático Médio (MSE) entre o bloco original e o reconstruído a partir do bloco comprimido. Desta análise foi verificado que transformando cada bloco utilizando 1, 2 e 3 componentes principais, correspondentes a razões de compressão (CR) de 96,875 %, 93,75 % e 90,625%, seria suficiente para reconstruir uma representação num espaço vetorial de dimensão menor mantendo as suas informações relevantes.

O trabalho de (OLIVEIRA,2000) incluiu a aplicação da teoria de Redes Neurais na compressão de imagens e constitui-se numa ótima referência de comparação das técnicas. Alguns aspectos do trabalho de (OLIVEIRA,2000), porém, dificultaram uma análise comparativa mais direta para o que este trabalho se propunha, quais sejam:

- A não inclusão da DWT nas comparações realizadas;
- as razões de compressão da DCT não foram as mesmas da KLT, o que significa que não foi mantido o mesmo número de coeficientes para as transformadas, o que se procurou evitar neste trabalho, tendo como premissa a manutenção de um mesmo número de coeficientes a serem transmitidos por todas as transformadas para então realizar as comparações;
- os tempos de processamento não foram incluídos no quadro comparativo;
- não foram utilizadas imagens padrão, nem comentadas as características estatísticas das imagens utilizadas nos testes;
- não foram mencionados os algoritmos utilizados para o cálculo da KLT;

As conclusões importantes do trabalho de (OLIVEIRA,2000) foram:

- A utilização da Rede Neural proporciona uma implementação simples do algoritmo quando comparada as demais técnicas;
- a utilização de uma técnica adaptativa pode fornecer vantagens quando um número maior de imagens com a mesma característica estatística tiverem que ser transmitidas;
- o tempo total envolvido na codificação pela Rede Neural (incluindo o tempo necessário de aprendizagem para o algoritmo de Rede Neural) tende a decrescer de forma exponencial na medida em que várias imagens com as mesmas características estatísticas tiverem que ser transmitias;

- a performance da implementação da Rede Neural foi superior a DCT, em termos de *PSNR* e *CR*, em todos os testes realizados, e bastante próxima (porém inferior) à PCA clássica.

O trabalho apresentado em (CASTRO,1996) aborda a utilização de Redes Neurais Artificiais (RNA) à compressão de imagens digitais, pela extração dos componentes principais da matriz de correlação (autovalores e autovetores da matriz de correlação dos vetores-bloco), utilizando para isso o Algoritmo Hebiano Generalizado Complexo. A imagem digital é transformada no domínio complexo através da aplicação direta da Transformada Rápida de FOURIER Bidimensional (FFT Bidimensional).

O trabalho em (CASTRO,1996) incluiu a aplicação da teoria de Redes Neurais na compressão de imagens e constitui-se numa importante referência para obtenção dos coeficientes KLT através desta técnica.

. Especificamente em comparação com este trabalho, o trabalho apresentado em (CASTRO,1996) diverge nos objetivos a que se propõe, e entre as diferenças menciona-se as que seguem:

- A não inclusão de comparações com a DWT e com a DCT;
- foram utilizados apenas subimagens de tamanho 8x8;
- a comparação pelo número de componentes, ao invés do THRESHOLD, foi utilizada da mesma forma que neste trabalho;
- os tempos de processamento não foram incluídos no quadro comparativo;
- a seleção de 5 imagens nos testes, com vários tipos de histogramas, representa um ampliação importante na análise dos resultados;
- não foi mencionada a utilização de algoritmos no cálculo da KLT (somente a implementação do cálculo pela RNA);

As conclusões importantes do trabalho em (CASTRO,1996) foram:

- A utilização da Rede Neural como método de extração dos componentes principais proporciona uma implementação mais simples quando comparada ao cálculo da KLT tradicional, além de não necessitar o cálculo de todos os autovalores e autovetores (somente os principais);
- foi constatada experimentalmente que as vantagens da análise dos componentes principais no domínio complexo estão estreitamente relacionadas com a maior razão de decréscimo dos autovalores da imagem no

domínio complexo, com relação à razão de decréscimo dos autovalores da imagem no domínio real;

- quanto maior for a descontinuidade entre os valores de freqüência absoluta de PIXELS no histograma, maior será a razão de decréscimo dos autovalores do domínio complexo em relação à razão de decréscimo do domínio real;
- os ganhos apresentados pela aplicação da técnica no domínio freqüência, tanto na *CR* quanto na *PSNR*, foram significativos chegando a apresentar um ganho de 20,51 % na *CR* e 19,21 % na *PSNR* para uma imagem de teste (mapa) de tamanho 128x128, com blocos de 8x8 PIXELS, 256 níveis de cinza e compressão de bloco de 2:1 (32 PIXELS do total de 64 PIXELS do bloco).

O trabalho proposto por (LEU,1994) trata de um aperfeiçoamento no Codificador de Imagem JPEG com a utilização de um algoritmo rápido denominado Transformada Aproximada de KARHUNEN-LOÈVE (AKLT - APPROXIMATE KARHUNEN-LOÈVE TRANSFORM), na sua versão adaptativa.

Segundo (LEU,1994), para algumas classes de dados de entrada que satisfazem o modelo de MARKOV de primeira ordem, um algoritmo rápido para a Transformada KL existe, aplicando a Análise de Perturbação sobre a Matriz de Covariância da seqüência de entrada na vizinhança de r = 1, onde " r " é o coeficiente de correlação dos dados de entrada. Esta Análise de Perturbação resulta no algoritmo rápido Aproximado da Transformada KL (AKLT). As comparações realizadas por (LEU,1994) revelam que para uma seqüência de MARKOV de primeira ordem, a AKLT apresenta uma performance melhor do que a DCT no aspecto de compactação de energia e na capacidade de descorrelacionar os elementos que formam o sinal de entrada.

A principal conclusão do trabalho de (LEU,1994) é que para ambas as imagens testadas ("*LENNA*" e "*PEPPER*") quando a razão de compressão da imagem compactada situa-se acima de 0,4 bits/PIXEL, o incremento na Razão de Compressão (CR), resultante da aplicação da AKLT, é de cerca de 5%, quando comparada com a CR resultante da aplicação do algoritmo JPEG nas mesmas condições. A Razão Sinal-Ruído de Pico (PSNR), por sua vez, apresenta um aumento de 0,3 dB numa razão de pixel de 1 bit/pixel. O trabalho de (LEU,1994) mostra o interesse da comunidade científica na pesquisa de algoritmos rápidos para aplicação da KLT, devido a sua alta capacidade de eliminação da correlação dos dados de entrada.

No trabalho apresentado em (DONY,1995) foi observado que a assunção de estacionariedade presente quando se fala na condição ótima da Transformada KL está longe

de existir para imagens. Conforme mencionam os autores, as imagens são compostas de regiões cuja estatística local (espacial) pode variar largamente ao longo da imagem. Foi proposto, então, um novo método de transformada onde a adaptação é otimizada. A imagem é segmentada em classes que formam os módulos do sistema. Cada módulo consiste numa transformação linear cujas bases são calculadas durante o período de treinamento inicial utilizando um algoritmo adaptativo. A classe apropriada para um dado vetor de entrada é determinada pelo classificador de subespaço. A performance do sistema adaptativo resultante mostra-se superior à da abordagem não adaptativa da transformação linear. O método apresentado pode também ser utilizado como segmentador. A segmentação proporcionada pelo algoritmo é independente das variações na iluminação da imagem e a representação em classes é análoga à direcionalidade das colunas sensitivas no córtex visual, segundo (DONY,1995). Algumas conclusões e observações importantes no trabalho desenvolvido em (DONY,1995) merecem destaque:

- Os testes foram realizados utilizando-se uma determinada quantidade de coeficientes da transformada, da mesma forma que os testes realizados neste trabalho;
- o novo método OIAL apresentado mostrou-se bastante superior a KLT tradicional chegando a introduzir uma redução de 22,68 % no valor do MSE nos testes realizados com a imagem padrão "*LENNA*", mesmo utilizando, neste caso, as mesmas classes geradas a partir de outras imagens de teste (imagens médicas);
- a divisão em classes visa adequar os autovetores às características dos vetores-bloco, o que constatamos também neste trabalho (quando da implementação do cálculo da KLT para cada bloco realizada no programa);
- a adaptação realizada é ótima desde que a representação dos dados é realizada pelo mínimo valor do MSE;
- a adaptação das classes (e a conseqüente criação das bases) é orientada ao vetor-bloco o que maximiza os resultados;
- a fase de treinamento do algoritmo para criação das classes é auto-organizada;
- o conjunto de bases criadas servem para realizar a mudança de base e também para a inclusão do vetor na respectiva classe;
- sendo independente da norma do vetor-bloco, o classificador é imune ao nível de iluminação da imagem;

- para a mesma velocidade de código, o uso do algoritmo OIAL decresce o MSE em 40 - 50% sobre a KLT tradicional; isto significa que se um valor fixo de distorção MSE de 60 for requerido, por exemplo, a Razão de Compressão (*CR*) poderia ser melhorada para 10 : 1 na KLT e para 19 : 1 no Algoritmo OIAL;
- embora não exista uma configuração única que permita a compressão de qualquer tipo de imagem com uma performance sempre ótima, o sistema OIAL (otimizado para um determinado tipo de imagem), mostrou-se também com uma boa performance na compressão de outros tipos de imagens.

Alguns aspectos não apresentados em (DONY,1995) foram os seguintes:

- Comparação dos tempos da KLT e do método OIAL;
- comparação dos resultados com a DCT e DWT;
- detalhamento dos algoritmos utilizados no cálculo da KLT;

O trabalho publicado em (DIAMANTARAS,1999) apresentou uma visão alternativa do modelo de compressão tradicional ao provar que a condição ótima da utilização da KLT tradicional quanto a minimizar o Erro Quadrático Médio não se mantém na presença do Ruído de Quantização. Foi provado que na presença do Ruído de Quantização pode ser definido um conjunto de vetores não ortogonais que formam a base de transformação linear, tal que a utilização desta base sobre um vetor de entrada torna o Erro Quadrático Médio (MSE) menor do que o que seria encontrado pela aplicação direta da KLT tradicional sobre o mesmo vetor de entrada.

Algumas conclusões e observações importantes no trabalho desenvolvido em (DIAMANTARAS,1999) merecem destaque:

- Na comparação da técnica apresentada com a KLT tradicional para a imagem LENNA (512x512) houve um ganho máximo de 0,39 dB (0,9 %) para um coeficiente de quantização $q = 6$ e $q = 8$;
- na comparação com a KLT foi constatado que se os coeficientes da transformada tiverem valores baixos, um aumento dos valores do *coeficiente de quantização q* (aumento da quantização) praticamente não provocam ganhos de PSNR (neste caso a correlação cruzada entre a PSNR e o q é muito reduzida); ao contrário, quanto maiores os valores dos coeficientes da correlação, maiores serão os ganhos obtidos pelo método com o aumento de q;
- na comparação com a DCT foram utilizadas 4 imagens de tamanho 256x256 (LENNA, BIRD, BRIDGE e CAMERAMAN) e duas imagens de tamanho

176x144 (MISS AMERICA e CLAIRE); o tamanho de bloco utilizado foi de 8x8 PIXELS com um conjunto de 64 parâmetros de cálculo para cada imagem; foi constatado, pelos resultados obtidos, um ganho máximo de 0,62 dB (1,74%) para um coeficiente de quantização *q=4*;

- na comparação com a DCT foi utilizada ainda uma seqüência de 20 FRAMES (SALESMAN), com o mesmo tamanho de bloco de 8x8 PIXELS e um mesmo conjunto de 64 parâmetros porém para todas as 20 imagens (FRAMES); foi constatado, neste caso, um ganho máximo de 0,49 dB (1,23%) na PSNR com resultados semelhantes aos obtidos com o conjunto de imagens de teste anterior, porém, com um OVERHEAD bem menor, pelo fato de utilizar um mesmo conjunto de parâmetros para todas as 20 imagens (FRAMES).

Alguns aspectos não abordados no trabalho de (DIAMANTARAS,1999) e que seriam importantes de serem avaliados, foram os seguintes:

- A verificação dos resultados para diferentes tamanhos de bloco;
- a discussão das características estatísticas das imagens de teste;
- a utilização de mais imagens nos testes com a KLT;
- a utilização de imagens com tamanhos diferentes;
- a não inclusão da *CR* na comparação das técnicas apresentadas;
- a comparação dos tempos de processamento das técnicas apresentadas;
- a comparação dos resultados com a DWT;
- o detalhamento dos algoritmos utilizados no cálculo da KLT;

No trabalho publicado em (EFFROS,1995) foi desenvolvido um sistema de compressão de imagem universal denominado Codificação pela Transformada Ponderada Universal (*WUTC - WEIGHTED UNIVERSAL TRANSFORM CODING*), onde através de um conjunto de pares “Transformada / Alocação de Bits” (CODEBOOK) é feita a codificação de blocos de imagem, associando cada bloco a um destes pares. A Transformada do par é criada a partir da estatística dos dados a serem codificados pela KLT dos blocos de entrada (vetores-bloco). Para um certo grupo de imagens, cujas características estatísticas são semelhantes àquelas para as quais foi criado o CODEBOOK, este pode ser utilizado com bons resultados, pois a etapa de quantização já está sintonizada com a respectiva Transformada do par. A cada bloco de imagem será associado um par “Transformada / Alocação de Bits” do CODEBOOK que é determinado a partir de limitações na razão de bits utilizada na etapa de quantização de forma a minimizar a distorção do sinal recuperado no decodificador (mínimo de LAGRANGE). O algoritmo de projeto das Transformadas no par, utiliza uma técnica iterativa

de descida (ITERATIVE DESCENT TECHNIQUE) para sua obtenção (mínimo de LAGRANGE). Os resultados apresentados pelos autores demonstram melhorias da ordem de 2 dB sobre o WUBA (codificador baseado na Transformada DCT que substitui a Matriz de Quantização fixa por uma coleção de matrizes), 3 dB sobre o codificador JPEG com alocação de um BIT e 10 dB sobre o Quantizador Vetorial de Entropia Limitada (ECVQ).

Mais tarde, o trabalho em (FENG,1999), com a participação de EFFROS (EFFROS,1995), reconheceu a complexidade computacional e o código extenso necessários para implementar o *WUTC* e sugeriram um algoritmo mais rápido para implementar esta técnica chamado de *KLT Separadas para Codificação por Transformada Universal Ponderada* (*SWUTC*). Esta técnica é baseada na codificação em pares separados da Transformada KL, um que descorrelaciona os blocos de dados coluna por coluna e o outro que descorrelaciona os blocos linha a linha. O par combinado de transformadas aproxima a performance da *KLT* tradicional, porém com 1/8 das multiplicações *FLOATING POINT* e 1/32 de espaço de armazenamento. A Transformada é armazenada em pares de matrizes 8 X 8. A comparação entre as técnica *SWUTC* (FENG,1999) e WUTC (EFFROS,1995), utilizando uma imagem de 2048 X 2048 PIXELS (obtida pela cópia de uma página da revista *IEEE SPECTRUM MAGAZINE* utilizando um SCANNER e contendo a mesma proporção de imagem e texto, e ainda um máximo de 64 pares *transformada/alocação de BIT*), mostrou que o algoritmo *SWUTC*, que emprega uma transformada menos ótima, alcança performance em torno de 1 dB a menos do que o algoritmo WUTC, concluindo-se daí que a troca da KLT ótima pela menos ótima custa 1 dB na performance Razão Sinal/Ruído, com a vantagem da simplificação do algoritmo.

Tanto no trabalho em (EFFROS,1995) como em (FENG,1999) alguns aspectos não foram objeto de análise. Pode-se citar os seguintes:

- A verificação dos resultados para diferentes tamanhos de bloco;
- a discussão das características estatísticas das imagens de teste;
- a especificação da quantidade de imagens de teste utilizadas;
- a utilização de imagens com tamanhos diferentes nas comparações;
- a comparação dos tempos de processamento das técnicas apresentadas;
- a comparação dos resultados com a DCT e DWT;
- o detalhamento dos algoritmos utilizados no cálculo da KLT;

O trabalho em (YAMASHITA,1996) propõe um algoritmo denominado de *Transformada KARHUNEN-LOÈVE Ponderada* (*WKLT*) para reduzir o efeito das bordas em virtude da blocagem da imagem durante a aplicação da Transformada. Este método utiliza a

sobreposição de blocos combinada com a ponderação reduzida dos blocos sobrepostos. Essa ponderação reduzida faz com que a variância dos blocos sobrepostos diminua, melhorando a qualidade da imagem nestas áreas mas à custa de um aumento no MSE. Um filtro passa-baixas é colocado no decodificador para os PIXELS da região de fronteira entre dois blocos diminuindo o erro nestas regiões. Os resultados obtidos sobre a imagem GIRL (no SIDBA), contendo 256 X 256 PIXELS e 8 bits por PIXEL, mostraram que apesar da melhora do efeito negativo da blocagem o MSE aumentou em 33,68 % da DCT 8x8 sem sobreposição para a DCT 10x10 com sobreposição e em 11,8 % da DCT 8X8 sem sobreposição para a KLT 10x10 com sobreposição. São válidas para o trabalho em (YAMASHITA,1996) as mesmas carências observadas nos trabalhos em (EFFROS,1995) e (FENG,1999) descritas no parágrafo anterior. O aumento do MSE por ocasião da aplicação da técnica em (YAMASHITA,1996) poderia também prejudicar a qualidade da imagem quando um elevado nível de compressão é requerido.

O trabalho em (GOYAL,1996) propõe uma codificação por Transformada Universal, baseada na estimativa da *KLT* a partir dos dados quantizados. A utilização dos dados quantizados na estimativa permite ao codificador e decodificador manter o mesmo estado sem a necessidade de envio de qualquer informação colateral (*SIDE INFORMATION*). Para o receptor ser capaz de corretamente decodificar os dados sem estar explicitamente informado do estado do codificador, é necessário que toda a adaptação do codificador dependa somente da informação já disponível no decodificador, isto é, do fluxo de dados codificados, o que é conhecido como adaptação reversa. Com esta estratégia, evita-se o envio da informação colateral porém com uma perda da eficiência na estimativa da estatística da fonte bem como também de performance causada pela restrição causada pelo fato da adaptação ser estritamente causal e devido a ruído na estimativa das estatísticas do sinal.

O trabalho em (GOYAL,1996) constitui-se numa importante abordagem para desenvolvimento futuro. O fato de não ser necessário enviar a base de representação tem uma importância fundamental nos ganhos de compressão que podem ser obtidos pelo algoritmo. Alguns aspectos não foram objeto de análise em (GOYAL,1996). Pode-se citar os seguintes:

- A verificação dos resultados para diferentes tamanhos de bloco;
- a aplicação do método em várias imagens de teste;
- a utilização de imagens com tamanhos diferentes nas comparações;
- a comparação dos resultados com a DCT e DWT;
- o detalhamento dos algoritmos utilizados no cálculo dos coeficientes da KLT tradicional nos testes;

No trabalho apresentado em (WALDEMAR,1997) foi proposto um sistema de chaveamento onde os blocos de imagem são processados em parte utilizando a KLT tradicional e em parte utilizando um sistema híbrido chamado SVD-KLT (SINGLE VALUE DECOMPOSITION–KARHUNEN-LOÈVE TRANSFORM). É realizada uma aproximação dos vetores da SVD utilizando quantização vetorial (VQ). A ortogonalidade das matrizes de transformação é garantida durante a quantização pela utilização do procedimento de ortogonalização de GRAM-SCHMIDT. Foi constatado que a comparação simples do sistema híbrido SVD-KLT com o sistema KLT tradicional não oferece vantagem na PSNR em nenhuma razão de compressão. O chaveamento entre a KLT e a SVD-KLT é superior à KLT tradicional. Este chaveamento entre um e outro sistema é realizado utilizando a técnica de Langrange que minimiza a distorção total da imagem reconstruída controlado pela razão de bits necessária para a quantização da imagem e a correspondente distorção de ambos os sistemas. Nos experimentos, para o sistema com chaveamento foi constatada uma melhora na PSNR de 0,1 a 0,4 dB acima da calculada para a KLT tradicional (0,87 % superior).

Não foram apresentados no trabalho em (WALDEMAR,1997) os seguintes aspectos:

- A verificação dos resultados para diferentes tamanhos de bloco;
- a aplicação do método em várias imagens de teste;
- a utilização de imagens com tamanhos diferentes nas comparações;
- a comparação dos resultados com a DCT e DWT;
- o detalhamento dos algoritmos utilizados no cálculo dos coeficientes da KLT tradicional e da SVD nos testes;

No trabalho apresentado em (NASIOPOULOS,1995) foi apresentado um método de compressão dos coeficientes da KLT para uma imagem colorida numa blocagem de 8x8 PIXELS utilizando uma forma modificada da técnica de codificação AMBTC (ABSOLUTE MOMENT BLOCK CODING). O Método tradicional AMBTC é do tipo FLC (FIXED LENGHT CODING), onde blocos diferentes são codificados com a mesma quantidade de bits. O AMBTC tradicional é aplicado após a etapa de quantização sobre os coeficientes da transformada onde cada bloco de 4x4 PIXELS é codificado em 2 níveis que preservam a média e o primeiro momento central dos valores do bloco. O AMBTC pode ser considerado como um quantizador vetorial adaptativo de 2 níveis, onde cada bloco é codificado por 2 vetores cujos valores mudam ou se adaptam às duas primeiras ordens da estatística do bloco (média e primeiro momento central absoluto). O método proposto em (NASIOPOULOS, 1995) é um método AMBTC modificado, onde os PIXELS são codificados em 4 níveis, e os coeficientes da KLT são divididos em 3 grupos de acordo com a ordem de valor dos

coeficientes no bloco. A aplicação da codificação sobre os coeficientes KLT é feita de forma a sempre preservar os maiores coeficientes, preservando assim a maior variância na posterior recuperação da imagem. Os testes foram realizados sobre a imagem LENNA (com tamanho não definido no artigo) e com blocos de tamanho 8x8, obtendo a AMBTC *modificada* proposta um ganho 68% superior à AMBTC tradicional.

Não foram apresentados no trabalho em (NASIOPOULOS,1995) os seguintes aspectos:

- A verificação dos resultados para diferentes tamanhos de bloco;
- a aplicação do método em várias imagens de teste;
- a utilização de imagens com tamanhos diferentes nas comparações;
- a comparação dos tempos de processamento da técnica apresentada;
- a comparação dos resultados com a DCT e DWT;
- o detalhamento dos algoritmos utilizados no cálculo dos coeficientes da KLT tradicional nos testes;

3.3 O ALGORITMO DE HUFFMAN

A aplicação de uma Transformada sobre os elementos da imagem (PIXELS), por si só, não consegue obter uma redução do tamanho do arquivo a ser transmitido. Ao contrário, aumenta o número de BYTES a transmitir para uma determinada imagem porque se antes da aplicação da transformada, cada PIXEL era codificado com 1 BYTE, após a aplicação da transformada tem-se cada PIXEL representado por um coeficiente da transformada que é definido, no caso deste trabalho, por exemplo, como do tipo *FLOAT*, para aumentar a precisão do cálculo e, portanto, com um tamanho de 4 BYTES (4 vezes maior do que a codificação original do PIXEL). No caso da Transformada KARHUNEN-LOÈVE, a sua aplicação sobre a imagem atua na eliminação da redundância interpixel, através da eliminação da correlação dos elementos que formam esta imagem. Assim, estruturas correlacionadas da imagem, que teriam valores de seus PIXELS diferentes, passam a ter os mesmos valores de coeficiente. Mesmo que se realize um truncamento dos coeficientes da transformada para valores inteiros (BYTES), ainda assim seria necessário enviar 1 BYTE para cada PIXEL, sem contar que os coeficientes, em geral, necessitam de, no mínimo 2 BYTES, para sua representação. Para realizar a compressão da imagem, aproveitando realmente a vantagem da descorrelação proporcionada pela aplicação da Transformada KL, é necessário utilizar algum algoritmo que elimine a *redundância de código*, isto é, a utilização de seqüências de bits menores (menores

do que os 8 bits por PIXEL) para representar valores iguais de PIXEL que se repetem com grande freqüência dentro da imagem transformada, em detrimento de outros valores que se repetem menos, recebendo, por isso, uma codificação com tamanho maior de BITS (até porque não existem infinitas seqüências pequenas de bits para representar todos os valores dos PIXELS).

Existem várias técnicas de compressão disponíveis, classificadas conforme permitam ou não a recuperação total dos valores codificados (ERROR FREE), de tamanho fixo ou variável, preditivos e não-preditivos, com ou sem a utilização de transformadas, entre outras classificações. Alguns exemplos de algoritmos de compressão (GONZALEZ,1993) são, entre outros: *RUN-LENGTH ENCODING*, ARITHMETIC CODING, ZIV-LEMPEL CODING, BIT PLANE CODING, CONSTANT AREA CODING, DIRECT CONTOUR TRACING, PREDICTIVE DIFFERENTIAL QUANTIZING, DOUBLE DELTA CODIGN, DELTA MODULATION e DIFFERENTIAL PULSE CODE MODULATION.

Neste trabalho, para realizar a compressão dos coeficientes da imagem processada pela KLT (e também pela DCT e DWT) utilizou-se o *Algoritmo de HUFFMAN*. Não houve nenhum motivo especial para a utilização deste algoritmo em detrimento dos demais, pois este trabalho não teve seu foco no estudo do codificador de símbolos. Um aspecto que motivou a escolha pelo algoritmo de HUFFMAN é porque ele é considerado o algoritmo ótimo no conceito da entropia de uma fonte de informação aproximando-se, assim, da menor compressão de código possível de ser obtida para esta fonte (SHANNON,1948). Esse algoritmo foi introduzido por David Huffman em 1952 (MURRAY,1994), e definido como um dos padrões de codificação de sinal pelo extinto CCITT (hoje ITU) nos protocolos CCITT T.4 e CCITT T.6 (mais conhecidos, respectivamente, como CCITT GROUP 3 AND GROUP 4 COMPRESSION), que definem os algoritmos de compressão específicos para compressão de dados binários.

O Algoritmo de HUFFMAN, mesmo obtendo a menor compressão da imagem no sentido de sua entropia, necessita da criação de uma tabela adicional de códigos que é tanto maior quanto mais valores diferentes existirem para serem codificados pelo algoritmo. O tamanho desta tabela não se mostrou pequeno como se pode constatar nos testes experimentais efetuados.

O algoritmo de HUFFMAN é definido como um código de comprimento variável com entrada de comprimento fixo. Neste algoritmo, a entrada é dividida em unidades de tamanho fixo (inteiros de 4 BYTES neste trabalho, correspondentes ao arredondamento dos

coeficientes da transformada) enquanto a saída é composta por seqüências de tamanho variável (códigos de HUFFMAN).

O algoritmo de HUFFMAN é considerado ótimo no conceito de *Entropia (H)* de uma fonte de informação, esta definida matematicamente como (SHANNON,1948):

$$H(X) = -K\sum_{i=1}^{n} p_i \log p_i \,(\text{Unidades de Código / mensagem}) \qquad \textbf{(Eq. 36)}$$

onde:

X : Variável Aleatória Discreta, com espaço de amostras definido pelo conjunto $\Omega_X=\{m_i\}=\{m_0,m_{1,\ldots},m_{M-1}\}$ de *M* eventos m_i (no caso de uma imagem, neste trabalho, m_i corresponde ao valor de um PIXEL, de forma que M=256 níveis de escala de cinza);

p_i : probabilidade de ocorrência do evento m_i;

K : constante para ajustar as unidades de medida dos eventos.

Utilizando a base 2 para o logaritmo na (Eq. 36) a *Unidade de Código* será o *BIT* (BINARY DIGITS).

A codificação de HUFFMAN é considerada ótima no sentido de que o número médio de bits requerido por palavra código na saída do compressor utilizados para representar os M níveis de quantização é reduzido ao mínimo possível sem causar uma degradação na mensagem.

A construção do código de HUFFMAN, para uma imagem pode ser realizada através das seguintes etapas, de forma resumida:

- Calculam-se as probabilidades de ocorrência de cada valor de PIXEL;
- agrupam-se duas a duas, sucessivamente, em cada etapa do processo, as menores probabilidades, somando seus valores que passam a compor um único valor para a próxima etapa de agrupamento, até que restem apenas 2 valores de probabilidade;
- determina-se como *bit 0* a probabilidade de menor valor e como *bit 1* a outra;
- a partir daí percorre-se o caminho inverso do agrupamento anterior das probabilidades, acrescentado-se aos bits já definidos sempre um *bit 0* (para a probabilidade menor e um *bit 1* para a probabilidade maior).

Neste trabalho, utilizou-se o Algoritmo de HUFFMAN apresentado na referência (PRESS,1994) através das rotinas "*hufmak.c*", "*hufenc.c*" e "*hufdec.c*", juntamente com os demais módulos do programa implementado.

3.4 A TRANSFORMADA DISCRETA DO COSENO (DCT)

A fim de realizar a comparação dos resultados obtidos na aplicação da KLT na compressão de imagens, utilizou-se a DCT, base do padrão JPEG de compressão. O padrão JPEG foi criado a partir dos estudos desenvolvidos pela união, em 1987, de 2 grupos de trabalho (MURRAY,1994): o PEG (criado em 1982 pela ISO para pesquisar métodos de transmissão de vídeo, imagens paradas e texto sobre linhas ISDN) e um sub-grupo do CCITT (criado em 1986 para pesquisar a transmissão por FACSIMILE de dados coloridos e em tons de cinza).

As etapas utilizadas na compressão por JPEG são as seguintes (GONZALEZ,1993):

- Divide-se a imagem em blocos de tamanho 8 x 8 PIXELS (subimagem), que são processados da esquerda para a direita e de cima para baixo da imagem;
- subtrai-se do valor de cada PIXEL a quantia 2^{n-1} onde 2^n é a quantidade total de níveis na escala de cinza utilizadas na imagem (256 níveis neste trabalho);
- aplica-se a 2D-DCT sobre cada bloco da imagem;
- quantiza-se os coeficientes da DCT gerados na etapa anterior;
- reordena-se os coeficientes pela técnica ZIGZAG, formando-se uma seqüência 1-D (uma dimensão) em que os primeiros coeficientes (componentes de freqüência serão os maiores e, geralmente, os mais significativos para a visualização da imagem.

Como as subrotinas *prontas* de SOFTWARE para JPEG utilizam o conceito de compressão por "*escolha de qualidade" (QUALITY SETTING)*, que adota um critério de eliminação da redundância psicovisual através da seleção de tabelas de quantização especialmente projetadas pelos pesquisadores para fornecer o melhor efeito visual para cada nível de compressão, as comparações com a rotina KLT implementada neste trabalho não seriam realizadas em igualdade de condições, a menos que se pudesse estabelecer uma quantização equivalente para os coeficientes da KLT. Por essa razão, foi necessário implementar a subrotina DCT diretamente na *linguagem C* de forma que, uma vez especificada a quantidade de coeficientes a transmitir (que define *Razão de Compressão* da

imagem), a quantização pela DCT é realizada diretamente pelo mesmo arredondamento estatístico realizado para os coeficientes da KLT).

A forma padrão utilizada no processamento de imagens para a DCT e implementada neste trabalho é definida, em sua forma bidimensional, para imagens de dimensões N x N, por:

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos[\frac{u\pi(x+\frac{1}{2})}{N}]\cos[\frac{v\pi(y+\frac{1}{2})}{N}] \qquad \textbf{(Eq. 37)}$$

e sua inversa:

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u,v)\cos[\frac{u\pi(x+\frac{1}{2})}{N}]\cos[\frac{v\pi(y+\frac{1}{2})}{N}] \qquad \textbf{(Eq. 38)}$$

onde:

f(x,y).....: é o valor do PIXEL na posição *x* e *y* da imagem já amostrada;

C(u,v).....: é o valor do coeficiente na posição *u* e *v* da imagem;

N .....: é o tamanho horizontal (ou vertical) da imagem quadrada (N x N);

$$\alpha(u),\alpha(v): \begin{cases} \sqrt{\frac{1}{N}} & \text{para } u,v = 0 \\ \sqrt{\frac{2}{N}} & \text{para } u,v = 1, 2, ..., N-1 \end{cases}$$

A implementação foi realizada com o auxílio da rotina unidimensional "*cosft2.c*", que se utiliza da rotina de cálculo da FFT para números reais "*realft.c*" para aumentar a velocidade do processamento, utilizando para isso a seguinte função auxiliar (para o caso unidimensional) (PRESS,1994):

$$y_i = \frac{1}{2}(f_j + f_{N-j-1}) - \sin[\frac{\pi(j+\frac{1}{2})}{N}](f_j - f_{N-j-1}) \qquad j = 0,\cdots,N-1 \qquad \textbf{(Eq. 39)}$$

onde y_i é o novo valor do conjunto de PIXELS da imagem sobre os quais se aplica então a rotina FFT cuja saída permite o cálculo direto da DCT unidimensional.

A adaptação para 2 dimensões, neste caso, aproveitando o cálculo da DCT pela FFT de forma unidimensional, utilizou a técnica de computar a DCT para cada linha e depois para as colunas da imagem. Para isso, adaptou-se a mesma estrutura de rotina utilizada em

(PRESS,1994) para a Transformada WAVELET 2-D (2-D WLT) substituindo-se apenas a parte referente ao cálculo para a DCT.

É oportuno salientar aqui que a utilização de rotinas mais sofisticadas que realizem o cálculo diretamente em 2 dimensões acarretam numa diminuição considerável de tempo de processamento, mas como apenas a FFT em duas dimensões (FFT 2D) foi apresentada na literatura consultada (PRESS,1994) (não estendendo o conceito da função auxiliar para a DCT 2 D, o que aproveitaria a 2D-FFT apresentada), sugere-se para um futuro desenvolvimento a pesquisa ou implementação desses algoritmos.

3.5 A TRANSFORMADA WAVELET DISCRETA (DWT)

A fim de realizar a comparação dos resultados obtidos na aplicação da KLT na compressão de imagens, utilizou-se também a DWT, base do padrão MPEG-4, em desenvolvimento pela ISO/IEC (ISO/IEC 14496-2, cujas versões 1 e 2 já foram aprovadas, respectivamente, em outubro/1998 e dezembro/1999), que através de seu subcomitê MPEG, criado em janeiro/1988, já estabeleceu também 2 padrões anteriores de compressão de imagens em movimento, a saber: MPEG-1 (ISO/IEC 11173, produzido em novembro de 1992) e o MPEG-2 (ISO/IEC 13818, produzido em 1994). As utilizações destes padrões envolvem a compressão de qualquer informação áudio-visual, como cinema digital, televisão de alta definição (HDTV), telefonia 2,5G (GPRS), 3G (THIRD GENERATION ou UMTS) e outras aplicações que exijam larguras de banda elevadas.

A DWT é uma transformada que utiliza bases ortonormais (chamadas "*funções mãe*" e "WAVELETS") para, através de operações lineares, transformar o espaço representação de uma função ou sinal (PRESS,1994).

A Transformada WAVELET foi proposta inicialmente por (GROSSMANN,1984) e, nessa representação, a base de decomposição do sinal não é mais composta pelas funções Seno e Coseno, como no caso da Transformada de FOURIER (FT ou FFT) ou da Transformada GABOR (1946 apud BENTLEY,1994,p.176) (STFT - SHORT TIME FOURIER TRANSFORM), mas sim composta por um conjunto de funções ortonormais transladadas e escaladas no tempo, que foram adaptadas ao processamento de sinais graças aos trabalhos de (DAUBECHIES,1988) e (MALLAT,1989), como mencionado por (BENTLEY,1994) em seu artigo.

Enquanto na Transformada GABOR (STFT) calcula-se os coeficientes pelo deslocamento de uma janela fixa sobre o sinal, na Transformada WAVELET as janelas (chamadas WAVELETS) são versões escaladas e transladadas da mesma função mãe.

Devido a mudança de escala, as janelas WAVELETS são estreitas em alta freqüência (alta resolução temporal e baixa resolução em freqüência) e largas em baixas freqüências (baixa resolução temporal e alta resolução em freqüência). Essas janelas de tamanho variável são adequadas na análise de sinais que possuem componentes de alta freqüência de curta duração e componentes de baixa freqüência de longa duração, que é justamente o caso dos sinais encontrados na prática (BENTLEY,1994). No caso da Transformada GABOR, portanto, a resolução é sempre igual tanto na escala de tempo quanto na escala de freqüência. Já no caso da Transformada WAVELET as escalas mais altas (correspondentes às baixas freqüências e a janelas temporais mais largas) possuem baixa resolução temporal e alta resolução em freqüência, enquanto as escalas mais baixas (correspondentes às altas freqüências e a janelas temporais mais estreitas) possuem alta resolução temporal e baixa resolução espectral.

Matematicamente é possível escrever a expressão utilizada para a geração da Transformada WAVELET Contínua como (BENTLEY,1994):

$$\mathrm{CWT}[b,a] = \frac{1}{\sqrt{a}} \int h^{*}\left(\frac{t-b}{a}\right) s(t)\,dt \qquad \textbf{(Eq. 40)}$$

Nesta equação, "*h*((t-b)/a)*" é chamada de função WAVELET, "a" é responsável pelo escalamento no tempo da função WAVELET e "b" pelo escorregamento da janela sobre o sinal "s(t)".

Para a Transformada WAVELET Discreta (DWT), neste trabalho, utilizou-se a base descoberta por (DAUBECHIES,1988) (Daub4), sendo esta a base a mais simples da família (formada apenas com 4 coeficientes), a de mais fácil implementação, a de melhor velocidade de processamento e com resultados superiores ou equivalentes na PSNR e CR em relação a DAUB6 e DAUB8 (RAGAB,1998) (outros tipos de bases WAVELET devem ser utilizadas em trabalhos futuros para ampliação da análise comparativa). A escala e o deslocamento são realizados pelo posicionamento dos coeficientes da matriz que forma a base, que é aplicada sobre o vetor coluna do sinal de entrada. Esta matriz da base (matriz **B**) é definida por (nesta matriz as partes em branco possuem valor zero):

$$\mathbf{B} = \begin{bmatrix}
c_0 & c_1 & c_2 & c_3 & & & & & & & & \\
c_3 & -c_2 & c_1 & -c_0 & & & & & & & & \\
 & & c_0 & c_1 & c_2 & c_3 & & & & & & \\
 & & c_3 & -c_2 & c_1 & -c_0 & & & & & & \\
\vdots & \vdots & & & & & \ddots & & & & & \\
 & & & & & & & c_0 & c_1 & c_2 & c_3 & \\
 & & & & & & & c_3 & -c_2 & c_1 & -c_0 & \\
c_2 & c_3 & & & & & & & & c_0 & c_1 & \\
c_1 & -c_0 & & & & & & & & c_3 & -c_2 &
\end{bmatrix}$$

É importante pensar em c_0, c_1, c_2 e c_3 como coeficientes de um filtro passa-baixas (H) e em c_3, $-c_2$, c_1 e $-c_0$ como coeficientes de um filtro passa-altas (G), também chamados de filtros de imagem em quadratura (*QUADRATURE MIRROR FILTERS*). O filtro G, na realidade, é projetado para que produza uma resposta zero (ou mais próximo possível de zero) para vetores de dados suaves (de pouca variabilidade). Isto é realizado fazendo com que os coeficientes c_3, $-c_2$, c_1 e $-c_0$ tenham um certo número de momentos iguais a zero. Quando isso é feito para *p* momentos (iniciando com o momento zero) diz-se que o conjunto de WAVELETS satisfaz uma "*condição de aproximação de ordem p*". Esse procedimento faz com que o sinal de entrada seja dividido num *sinal suave* (H) e num sinal detalhado (G), cada um com tamanho de *N/2*.

Como a base **B** é, por hipótese, ortogonal então $\mathbf{B^{-1}} = \mathbf{B^T}$ ou seja:

$$\mathbf{B^{-1}} = \mathbf{B^T} = \begin{bmatrix}
c_0 & c_3 & & & \cdots & & & & & c_2 & c_1 \\
c_1 & -c_2 & & & \cdots & & & & & c_3 & -c_0 \\
c_2 & c_1 & c_0 & c_3 & & & & & & & \\
c_3 & -c_0 & c_1 & -c_2 & & & & & & & \\
 & & & & \ddots & & & & & & \\
 & & & & & c_2 & c_1 & c_0 & c_3 & & \\
 & & & & & c_3 & -c_0 & c_1 & -c_2 & & \\
 & & & & & & & c_2 & c_1 & c_0 & c_3 \\
 & & & & & & & c_3 & -c_0 & c_1 & -c_2
\end{bmatrix}$$

A combinação da matriz base **B** e sua inversa, considerando que a base é ortogonal, isto é, $\mathbf{B}\,\mathbf{B}^{-1} = \mathbf{B}\,\mathbf{B}^{T} = \mathbf{I}$, produz as seguintes equações:

$$\begin{aligned} c_0^2 + c_1^2 + c_2^2 + c_3^2 &= 1 \\ c_2 c_0 + c_3 c_1 &= 0 \end{aligned} \qquad \textbf{(Eq. 41)}$$

São necessárias mais 2 equações para completar o cálculo. Essas equações são obtidas da *condição de aproximação de ordem p = 2*, e são dadas por:

$$c_3 - c_2 + c_1 - c_0 = 0 \qquad \textbf{(Eq. 42)}$$
$$0c_3 - 1c_2 + 2c_1 - 3c_0 = 0$$

A solução desse sistema de equações foi realizada por (DAUBECHIES,1988), que obteve os seguintes valores:

$$c_0 = (1+\sqrt{3})/4\sqrt{2} \qquad c_1 = (3+\sqrt{3})/4\sqrt{2}$$
$$c_2 = (3-\sqrt{3})/4\sqrt{2} \qquad c_3 = (1-\sqrt{3})/4\sqrt{2}$$

A DWT consiste em aplicar a matriz base **B** sobre um vetor de dados de N elementos, criando então os componentes suaves (*SMOOTH COMPONENTS*) e os componentes de detalhes, com N/2 componentes para cada tipo de dados. Aplicando novamente a matriz base **B** (agora com dimensão reduzida de N/2) sobre os N/2 componentes *suaves* obtém-se N/4 componentes *mais suaves* (*SMOOTH-...-SMOOTH*) até que, geralmente, apenas 2 componentes suaves restem como resultado do cálculo. A seqüência mostrada na Figura 9 ilustra esse algoritmo (que é conhecido como *Algoritmo Piramidal*) para um conjunto de 16 elementos. Este algoritmo é projetado para trabalhar com tamanhos de vetores de dados que são potências de 2 e, portanto, os testes devem ser efetuados utilizando imagens com dimensões que são potências de 2 (32x32, 64x64, ...). Se o vetor de dados tiver um comprimento numa potência de 2 mais elevada do que os 16 elementos da Figura 9 mais estágios aplicando a matriz base **B**, juntamente com as respectivas permutações, são necessários até que o vetor final contenha 2 componentes *suaves "S" (SMOOTH)* e uma hierarquia de componentes *detalhe "D"*, que uma vez calculados num estágio, se propagam para os demais estágios sem nenhuma modificação. Para inverter a DWT percorre-se o caminho inverso ao mostrado na Figura 9 utilizando-se, entretanto, neste processo, a matriz inversa $\mathbf{B^{-1}}$.

Neste método de cálculo da DWT, cabe salientar que os últimos coeficientes WAVELET de cada estágio são sensibilizados pelos valores de ambos os lados do vetor de dados, o que ocasiona o efeito denominado de "*WRAP-AROUND"* que poderia ser eliminado transformando a matriz da base numa matriz puramente tridiagonal.

Neste trabalho, para implementar a DWT, foram utilizadas três rotinas existentes na referência (PRESS,1994). A rotina "*pwtset.c*" que prepara a rotina principal para o cálculo da Daub4 (4 coeficientes DAUBECHIES); a rotina "*pwt.c*" que realiza o cálculo da DWT propriamente dita e a rotina "*wtn.c*" que utiliza as rotinas anteriores para calcular a DWT em

2 dimensões (*2D-DWT*), cuja estrutura foi adaptada posteriormente para o cálculo da DCT, conforme já mencionado anteriormente.

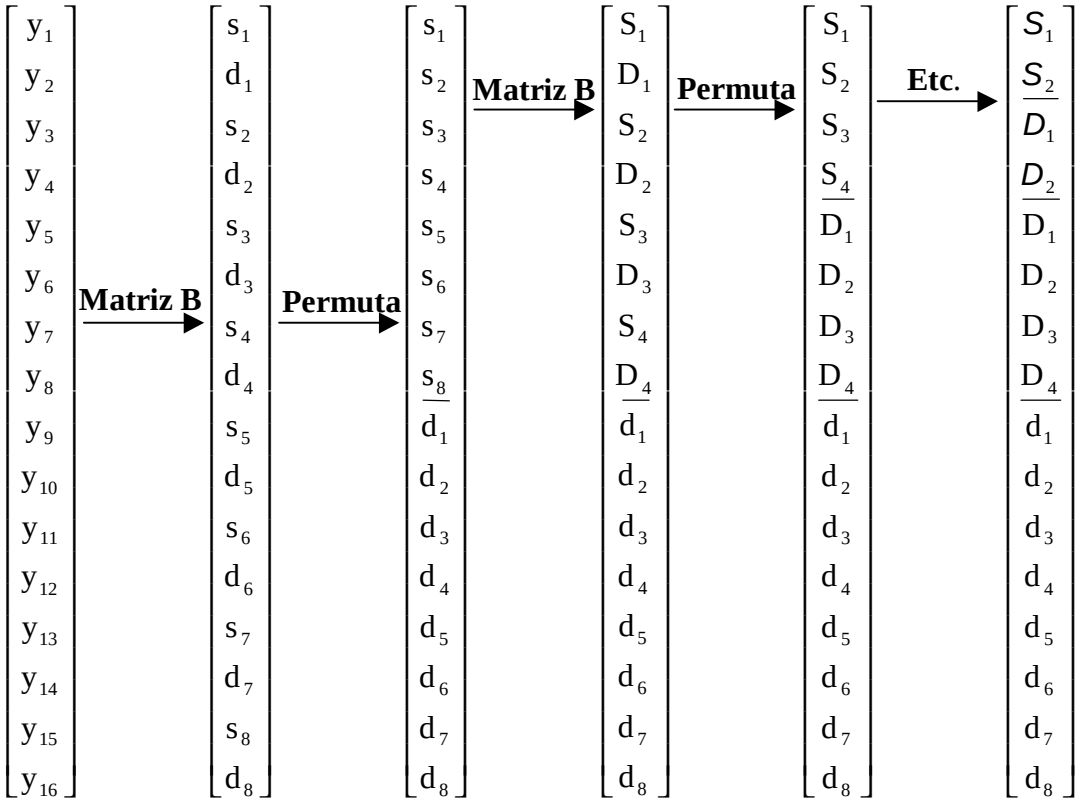


Figura 9 – Seqüência do Algoritmo Piramidal de Cálculo da DWT

4 MATERIAIS E MÉTODOS

Para que se pudesse avaliar de forma mais concreta os detalhes relativos à implementação da Transformada KARHUNEN-LOÈVE, realizou-se neste trabalho os seguintes procedimentos:

- Implementação de um programa desenvolvido totalmente na *linguagem C* padrão ANSI (dez/1989);
- implementação de um módulo específico de leitura e escrita de arquivos GRAYSCALE do tipo *"*.BMP" (BITMAP)*, de qualquer dimensão e utilizando o próprio *PALETTE* do arquivo processado*;*
- implementação de um módulo de cálculo do Algoritmo de HUFFMAN diretamente sobre o arquivo de imagem transformado;
- implementação de um módulo de cálculo da KLT utilizando o Algoritmo de Jacobi com a escolha dos maiores coeficientes do bloco por ordem das posições dos PIXELS no vetor-bloco e por pesquisa direta dos maiores coeficientes do bloco;
- implementação de um módulo de cálculo da KLT utilizando o Algoritmo QL (em conjunto com o Algoritmo HOUSEHOLDER de tridiagonalização de uma matriz qualquer), com escolha dos maiores coeficientes do bloco por ordem da posição dos PIXELS no vetor-bloco;
- implementação de um módulo de cálculo da KLT utilizando o Algoritmo de QL (em conjunto com o Algoritmo HOUSEHOLDER de tridiagonalização de uma matriz qualquer), com escolha dos maiores coeficientes do bloco por ordem da posição dos PIXELS no vetor-bloco e enviando o conjunto de autovetores correspondente aos maiores autovalores para cada vetor-bloco;
- implementação de um módulo de cálculo da DCT para a tela inteira e para o caso de blocagem com a seleção dos maiores coeficientes do bloco para a compressão por pesquisa direta dos maiores coeficientes do bloco ou pela técnica do ZIGZAG;
- implementação de um módulo de cálculo da DWT para a tela inteira e para o caso de blocagem com a seleção dos maiores coeficientes do bloco para a compressão por pesquisa direta dos maiores valores.

- implementação de um módulo de comparação da imagem original com a reconstruída após a compressão através do cálculo da *Razão Sinal Ruído de Pico* (PSNR) das duas imagens;
- implementação de módulo para o cômputo do tempo de processamento das rotinas;
- implementação de módulo para o cômputo da *Razão de Compressão (RC)* do arquivo compactado;
- construção das tabelas de comparação entre as transformadas implementadas.

Nas tabelas comparativas das implementações realizadas utilizou-se a imagem padrão LENNA 512x512, que também é utilizada nos artigos pesquisados. Essa imagem faz parte do SIDBA (*STANDARD IMAGE DATA BASE*), fazendo parte integrante do Banco de Imagens do LAPSI (*Laboratório de Processamento Digital de Sinais da UFRGS*). Outras imagens para testes são também encontradas, em vários tamanhos, na referência (SIDBA,2003).

Nas seções seguintes descrevem-se as condições ambientais utilizadas na fase de experimentação.

4.1 CONDIÇÕES EXPERIMENTAIS

Nos experimentos realizados utilizaram-se os seguintes equipamentos e programas aplicativos:

- Microcomputador VECTRON INTEL PENTIUM 866 MHz, contendo:
  - 128 Mbytes de memória RAM;
  - placa Aceleradora de Vídeo 3dfx VOODOO3 com 16 MB de memória;
  - CD-ROM DRIVE / F5BA 40 GBYTES
  - Controlador IDE Primário (FIFO duplo);
  - Controlador IDE Secundário (FIFO duplo);
  - Via BUS MASTER PCI IDE CONTROLLER;
  - Barramento PCI 32 BITS;
  - Processador de Dados Numéricos;
  - VIA Tech V82C598 CPU TO AGP CONTROLLER;
  - VIA Tech VT82C696 PCI TO ISA BRIDGE;
  - VIA Tech VT82C696 POWER MANAGEMENT CONTROLLER OFF;
  - VIA Tech VT82C69x CPU TO BRIDGE;
  - Monitor STUDIOWORKS 560A
- SCANNER TCÊ S440;
- sistema operacional WINDOWS 98;
- rotinas de processamento de imagem constantes em (GOYAL,1996);
- SOFTWARE de edição de imagens PAINT SHOP PRO 4 Versão 4.12 SHAREWARE distribuído pela JASC INCORPORATED;
- SOFTWARE MICROSOFT VISUAL C/C++ Versão 5.0;

- SOFTWARE MICROSOFT EXCEL/97

4.2 DETALHES DE IMPLEMENTAÇÃO

Nas próximas subseções são abordados detalhes relevantes da implementação realizada neste trabalho e que merecem destaque.

4.2.1 PARÂMETROS E CONFIGURAÇÕES DO PROGRAMA

4.2.1.1 Tamanho das Variáveis

O tamanho em BYTES das variáveis C no ambiente computacional utilizado foi verificado e corresponde a:

- *unsigned char: 1*
- *char: 1*
- *unsigned int: 4*
- *unsigned short int: 2*
- *short int: 2*
- *int: 4*
- *long int: 4*
- *unsigned long int: 4*
- *float: 4*
- *long float: 8*
- *double: 8*
- *long double: 8*

4.2.1.2 Alinhamento das Variáveis de Estruturas e Uniões

Um detalhe importante que foi implementado neste trabalho nas rotinas utilizando a Linguagem C (Padrão ANSI C – Dez/1989) refere-se ao alinhamento de BYTES entre as variáveis nas “estruturas” (tipo STRUCTURE na linguagem Padrão C/C++) e “uniões” (tipo UNION na linguagem Padrão C/C++) utilizadas dentro do programa. Normalmente, se nenhuma alteração for feita no programa, o SOFTWARE MICROSOFT C/C++ realiza a leitura de variáveis em blocos múltiplos de 4 BYTES. Assim, mesmo que uma variável seja definida com um tamanho de 2 BYTES dentro de uma estrutura, quando é lido um arquivo de imagem BITMAP (ou parte dele) para dentro dessa estrutura, o próprio MICROSOFT C se

encarrega de acrescentar tantos BYTES quantos necessários entre duas variáveis contíguas da estrutura para que o número de BYTES entre elas seja um múltiplo de 4.

Para ajustar essa configuração foi necessário acrescentar uma linha de programa contendo a seguinte diretiva:

" #pragma pack(1)";

Essa diretiva orienta o compilador para que o alinhamento seja realizado BYTE a BYTE dentro das estruturas e uniões utilizadas no programa.

4.2.1.3 Valores dos PIXELS

Os valores dos PIXELS para o cálculo dos componentes na base dos autovetores da matriz de covariância foram redefinidos para o tipo FLOAT (ponto flutuante/4 BYTES), ao invés de inteiro (INTEGER), a fim de proporcionar a precisão adequada para o cálculo dos coeficientes nas rotinas de processamento de imagem utilizadas no programa.

4.2.1.4 Medida do Tempo de Processamento

A contagem do tempo de processamento foi realizada dentro do programa com a utilização da função CLOCK da linguagem C, que retorna o número de tiques do relógio que ocorrem 18,2 vezes por segundo (JAMSA,1999). Isto faz com que se tenha os tempos em intervalos de 1/18,2 segundos (= 54,94 ms), gerando medidas que podem variar em 54,94 ms para mais ou para menos em relação ao valor apresentado. A referência (JAMSA,1999) fornece elementos para utilizar as rotinas do DOS para controle e redirecionamento das interrupções do microcomputador, e que podem ser estudadas e testadas, mas essa adaptação não foi realizada no âmbito desta implementação.

Além disso, as medidas tempo estão sujeitas também a interferência constante do próprio Sistema Operacional WINDOWS/98, o que provoca também pequenas alterações adicionais nos valores de medida de tempo das rotinas, ou seja, a não repetibilidade das medidas, uma das condições básicas para uma comparação adequada de medidas de tempo de forma mais rigorosa. Em 1000 medidas dos tempos de gravação tomadas para a imagem "Lenna", para tamanhos de bloco de 2x2 e 4x4, ambas com razão de compressão de bloco de 4:3, constataram-se tempos de, respectivamente, média 578,8 *ms* com desvio padrão de 31,8 *ms* e média 783,1 *ms* com desvio padrão de 40,407 *ms.* Foram incluídos neste trabalho os gráficos de tempo de geração dos arquivos compactados em cada teste, pois servem como uma estimativa dos tempos de processamento, auxiliando na análise dos resultados.

4.2.1.5 Quantização dos Coeficientes da KLT

Após o cálculo dos coeficientes da KLT segue-se a etapa de *quantização* dos mesmos. O processo de *quantização* envolve perdas e é uma etapa necessária uma vez que os coeficientes são definidos como do tipo FLOAT (para possibilitar o máximo de precisão no cálculo dos autovetores e dos próprios coeficientes), ou seja, com um total de 4 BYTES de tamanho para cada coeficiente, e a compressão pelo código de HUFFMAN não pode ser realizada com toda essa precisão, pois as tabelas de gerações do código ficariam muito extensas.

A etapa de quantização é uma etapa importante e atua no processo de redução da redundância psicovisual na imagem, sendo uma fonte adicional de compressão significativa. A aplicação dos *mapeadores* (*transformadas*) faz com que os coeficientes assumam valores *reais* em intervalos maiores do que o intervalo de *0* a *255* assumido pelos valores dos PIXELS de imagem (que podem ser representados por valores inteiros de 1 BYTE). Esse aumento no intervalo de representação dos PIXELS, em conjunto com a necessidade de precisão nos cálculos, implica na definição dos coeficientes como do tipo FLOAT (4 BYTES) ou DOUBLE (8 BYTES). De outro modo, a ação dos mapeadores de modo a eliminar a correlação entre os PIXELS atua no sentido de reduzir a redundância INTERPIXEL e diminuir a quantidade de valores de representação dos PIXELS. Porém, devido a grande quantidade e complexidade dos cálculos envolvidos com a decorrente perda de precisão inerente ao processo de cálculo, a igualdade entre dois coeficientes do tipo *real*, mesmo com valores muito próximos, não se verifica se mantido o coeficiente com todas as suas casas decimais. Assim, após calcular os coeficientes é necessária a etapa de *quantização* para que o arquivo codificado não fique com um tamanho superior ao arquivo original.

Utilizou-se neste trabalho a quantização escalar uniforme dos coeficientes através do simples arredondamento dos mesmos para zero casas decimais. Com esta precisão perde-se parte da informação importante para a reconstrução da imagem. A imagem reconstruída passa, então, a ser diferente da original. O tamanho dessa diferença pode ser avaliado através da utilização do parâmetro *PSNR*. Os testes de quantização realizados neste trabalho bem como outras observações importantes estão descritas na Seção 10 como apêndice.

4.2.1.6 Teste do Programa

Para garantir que o programa realiza de forma correta cada etapa do cálculo, criou-se uma "*imagem*" de teste com as dimensões de 4 PIXELS de LARGURA por 4 PIXELS de

ALTURA, com cada PIXEL contendo um tom de cinza diferente (incluindo o tom preto - 00H - e o tom branco - FFH). Os valores dos PIXEL da imagem original e da imagem reconstruída a partir do cálculo estão mostradas a seguir:

Imagem Original

$$\begin{bmatrix} 228 & 111 & 211 & 100 \\ 240 & 80 & 191 & 103 \\ 175 & 0 & 255 & 15 \\ 118 & 220 & 237 & 116 \end{bmatrix}$$

$\Leftrightarrow$

Imagem Reconstruída

$$\begin{bmatrix} 227{,}9999 & 111{,}0000233 & 211 & 100{,}00002 \\ 240 & 79{,}9999656 & 191 & 102{,}99999 \\ 175{,}0001 & -1{,}40354\text{E}-05 & 255 & 14{,}999969 \\ 118 & 220{,}0000512 & 237 & 115{,}99999 \end{bmatrix}$$

Todos os cálculos intermediários para alcançar o resultado da imagem reconstruída foram realizados de forma manual para a imagem de teste utilizada ("*padrão4x4256.bmp*"). Os detalhes destes testes estão incluídos na Seção 11 como apêndice deste trabalho.

4.2.2 DETALHES DOS ARQUIVOS DE IMAGEM

4.2.2.1 Características da Imagem BITMAP

No trabalho foram utilizados arquivos BITMAP (".BMP") como entrada para as rotinas escritas em linguagem C. Os arquivos BITMAP de imagens em preto e branco são compostos das seguintes seções (MURRAY,1994):

- HEADER, contendo:
  - código do arquivo (= "BM" para arquivos BITMAP/2BYTES);
  - tamanho do arquivo (4 BYTES);
  - área reservada (4 BYTES)
  - OFFSET com o endereço onde inicia os dados da imagem (4 BYTES);
- INFO HEADER, contendo:
  - o tamanho do INFO HEADER (= 40 Bytes em geral / 4BYTES);
  - a largura da imagem (depende da largura da imagem lida / 4 BYTES);
  - a altura da imagem (depende da altura da imagem lida / 4 BYTES);
  - o número de BIT PLANES (= 1 no caso deste trabalho / 1 BYTE);
  - o número de BITS por PIXEL (= 8 bpp (BITS por PIXEL) para as imagens utilizadas neste trabalho / 2 BYTES);
  - o Método de Compressão (= 0 neste trabalho, significando imagens fonte sem compressão / 4 BYTES);
  - o tamanho do BITMAP (depende do tamanho da imagem / 4 BYTES);

- a resolução horizontal (em PIXELS por metro, sendo mantidas as resoluções dos arquivos lidos / 4 BYTES);
- a resolução vertical (em PIXELS por metro, sendo mantidas as resoluções dos arquivos lidos / 4 BYTES);
- o número de cores utilizado (= 256 tons de cinza para imagens com 8 bpp / 4 BYTES);
- o número de cores significativas (= 256 para as imagens utilizadas / 4 BYTES).

- PALETTE, que corresponde a um conjunto de 4 BYTES por tom de cinza (são 256 tons de cinza, logo a tabela PALETTE contém 1024 BYTES);
- BITMAP, que corresponde ao conteúdo da imagem propriamente dita e que na verdade representa um índice na tabela do PALETTE o qual corresponde a um tom de cinza específico armazenado nesta tabela.

4.2.2.2 Alinhamento dos Arquivos BITMAP

Ao ser criada através de um SOFTWARE de edição de imagens ou de um SCANNER, cada linha da imagem é gravada em um arquivo no formato BITMAP (BMP) com algum alinhamento em múltiplos pares de BYTES (geralmente de 4 BYTES – LONG WORD BOUNDARIES (MURRAY,1994)). Assim, uma imagem com largura de 1194 BYTES, por exemplo, é gravada na realidade com 1196 BYTES (2 BYTES a mais do que a largura real da imagem contida no campo LARGURA do INFO HEADER, de forma que 1196 BYTES seja um múltiplo de 4 BYTES). Estes 2 BYTES a mais são acrescentados no arquivo gravado, pelo SOFTWARE de edição de imagens que se está trabalhando Qualquer programa de leitura de imagens deve levar em consideração esse alinhamento.

Neste trabalho foi testado o alinhamento no tratamento do arquivo BITMAP a fim de eliminar possíveis erros nas comparações das imagens devido a esse problema. Os testes e outras observações importantes realizadas sobre o alinhamento dos arquivos BITMAP encontram-se descritos na Seção 12 como apêndice deste trabalho.

4.2.2.3 Visualização das Imagens Processadas com vários PSNR

Nos vários testes realizados na Seção 5 um dos parâmetros apresentados é a *Razão Sinal/Ruído de Pico (PSNR)*, que é uma medida da qualidade da imagem reconstruída, conforme já definido neste trabalho. Apresenta-se a seguir, como referência, alguns resultados

obtidos a partir do processamento da *KLT* pelo programa implementado neste trabalho para vários *PSNR*.

A Figura 10(a) mostra a imagem de teste LENNA (512x512 PIXELS) original, sem nenhuma compressão. A Figura 10(b) mostra a imagem LENNA (512x512) reconstruída a partir do processamento da imagem original, utilizando KLT QL, adotando uma blocagem de 2x2 PIXELS e uma compressão de bloco de 4:1.

(a) Imagem Original LENNA (512x512) (b) Imagem LENNA com PSNR= 31,75687

Figura 10 – (a) Imagem LENNA Original (512x512) e (b) LENNA PSNR = 31,75687

A Figura 11 mostra o gráfico obtido a partir do programa e correspondente aos valores dos 4 autovalores da matriz de correlação dos vetores-bloco 2x2 da Figura 10(b).

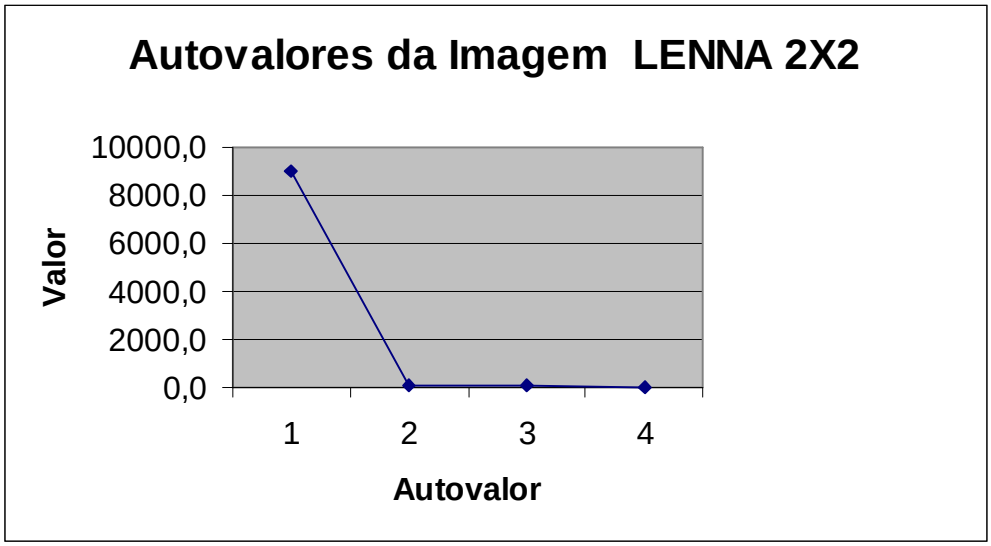


Figura 11 – Autovalores da Imagem LENNA 2x2

A partir das tabelas que possibilitaram a construção da Figura 11, verifica-se que ao ser transmitido somente o primeiro coeficiente da transformada KLT dos blocos da imagem LENNA 2x2, já se conserva uma variância correspondente a 98,1 % da variância do bloco, conforme definido na (Eq. 20). Com 2 coeficientes esse valor aumenta para 99,3 % da variância total do bloco de imagem e com 3 coeficientes esse valor atinge 99,9 %.

A Figura 12(a) mostra a imagem LENNA (512x512) reconstruída a partir do processamento da imagem original, utilizando KLT QL, adotando uma blocagem de 4x4 PIXELS e uma compressão de bloco de 16:1. A Figura 12(b) mostra a imagem LENNA (512x512) reconstruída a partir do processamento da imagem original, utilizando KLT QL, adotando uma blocagem de 8x8 PIXELS e uma compressão de bloco de 64:1.

(a) Imagem LENNA com PSNR=26,977861 (b) Imagem LENNA com PSNR= 23,710419

Figura 12 – (a) LENNA (PSNR=26,977861) e (b) LENNA (PSNR=23,710419)

A Figura 13 mostra o gráfico obtido a partir do programa e correspondente aos 16 autovalores da matriz de correlação dos vetores-bloco 4x4 representado na Figura 12(a).

Observando-se a Figura 13, verifica-se que ao ser transmitido somente o primeiro coeficiente da transformada KLT dos blocos da imagem LENNA 4x4, já se conserva uma variância correspondente a 94,3 % da variância do bloco. Com 2 coeficientes esse valor aumenta para 97,2 % da variância total do bloco de imagem e com 3 coeficientes esse valor atinge 98,3 %.

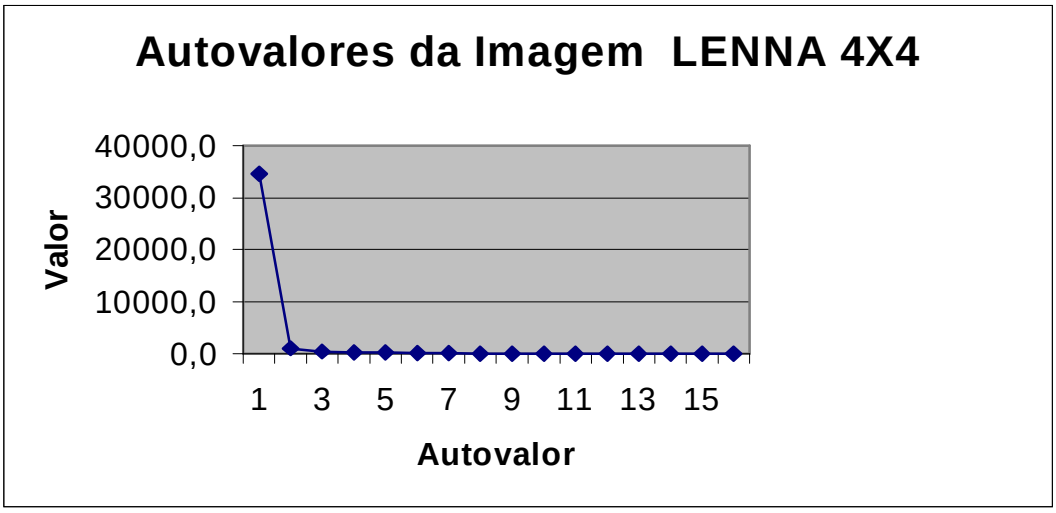


Figura 13 – Autovalores da Imagem LENNA 4x4

A Figura 14 mostra o gráfico obtido a partir do programa e correspondente aos 64 autovalores da matriz de correlação dos vetores-bloco 8x8 representado na Figura 12(b).

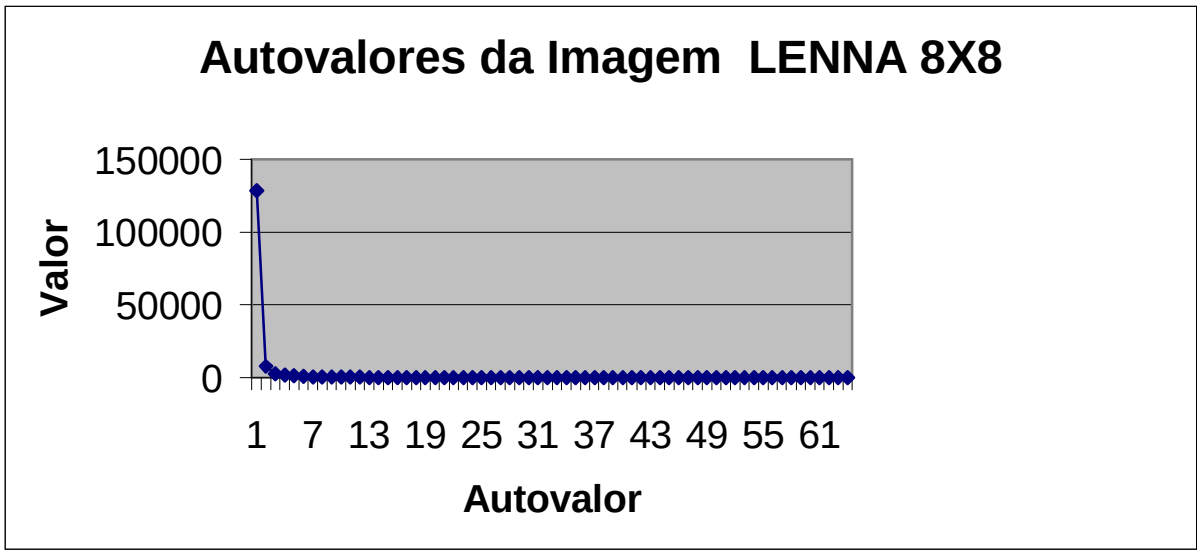


Figura 14 – Autovalores da Imagem LENNA 8x8

Observando-se a Figura 14, verifica-se que ao ser transmitido somente o primeiro coeficiente da transformada KLT dos blocos da imagem LENNA 8x8, já se conserva uma variância correspondente a 87,9 % da variância do bloco. Com 2 coeficientes esse valor aumenta para 93,1 % da variância total do bloco de imagem e com 3 coeficientes esse valor atinge 96,2 %.

A Figura 15(a) mostra a imagem LENNA (512x512) reconstruída utilizando a KLT QL, adotando uma blocagem de 4x4 PIXELS e uma compressão de bloco de 16:8 (ou 2:1). A Figura 15 (b) mostra a imagem LENNA (512x512) reconstruída utilizando KLT QL, adotando uma blocagem de 2x2 PIXELS e uma compressão de bloco de 4:4.

(a) Imagem LENNA com PSNR=40,448388 (b) Imagem LENNA com PSNR= 58,65534

Figura 15 – (a) LENNA (PSNR=40,448388) e (b) LENNA (PSNR=58,65534)

Os gráficos dos autovalores para as blocagens 2x2 e 4x4 são os mesmos já apresentados anteriormente nesta seção.

5 RESULTADOS

As tabelas de resultados foram criadas para comparar diversos tipos de cálculo de transformadas aplicados sobre 6 imagens, a saber: LENNA, BABUINO (BABOON), CASA (HOUSE), MENINA (GIRL), TIGRE E VENTILADOR . Foram utilizados blocos de imagem de dimensão 2x2 PIXELS, 4x4 PIXELS, 8x8 PIXELS e 16x16 PIXELS. Após a aplicação das transformadas sobre os blocos de imagem, os dados de saída foram compactados pelo algoritmo de HUFFMAN. Os tipos de cálculo utilizados nas tabelas e gráficos foram:

a) Compressão pelo **Algoritmo de HUFFMAN** diretamente sobre a imagem original;
b) **KLT JACOBI**: realiza o cálculo dos autovetores e autovalores pelo Algoritmo de JACOBI com a escolha dos maiores coeficientes KL de acordo com a posição do PIXEL no vetor-bloco (considerando que o maior coeficiente está localizado na parte superior à esquerda de cada bloco, o segundo a sua direita e assim sucessivamente de cima a baixo no bloco); após o cálculo os coeficientes são codificados pelo Algoritmo de Compressão de HUFFMAN e o arquivo compactado é gerado;
c) **KLT QL**: realiza o cálculo dos autovetores e autovalores pelo Algoritmo QL (juntamente com o Método HOUSEHOLDER) com a escolha dos maiores coeficientes KL de acordo com a posição do PIXEL no vetor-bloco; após o cálculo os coeficientes são codificados pelo Algoritmo de Compressão de HUFFMAN e o arquivo compactado é gerado;
d) **KLT JACOBI ORD**: realiza o cálculo dos autovetores e autovalores pelo Algoritmo de JACOBI com a escolha dos maiores coeficientes KL do vetor-bloco selecionados por um módulo especial no programa que pesquisa diretamente os maiores coeficientes; após o cálculo os coeficientes são codificados pelo Algoritmo de Compressão de HUFFMAN e o arquivo compactado é gerado;
e) **DCT BLOCO ZIG**: realiza o cálculo pela DCT para cada vetor-bloco de imagem com a escolha dos maiores coeficientes da DCT selecionados pelo Método ZIGZAG; após o cálculo os coeficientes são codificados pelo Algoritmo de Compressão de HUFFMAN e o arquivo compactado é gerado;
f) **DCT BLOCO ORD**: realiza o cálculo pela DCT para cada vetor-bloco de imagem com a escolha dos maiores coeficientes da DCT do vetor-bloco selecionados por um módulo especial no programa que pesquisa diretamente os maiores coeficientes; após o cálculo os coeficientes são codificados pelo Algoritmo de Compressão de HUFFMAN e o arquivo compactado é gerado;

g) **DCT INT ZIG**: realiza o cálculo pela DCT para a imagem inteira (não há blocagem) com a escolha dos maiores coeficientes da DCT selecionados pelo Método ZIGZAG; após o cálculo os coeficientes são codificados pelo Algoritmo de Compressão de HUFFMAN e o arquivo compactado é gerado;

h) **DCT INT ORD**: realiza o cálculo pela DCT para a imagem inteira (não há blocagem) com a escolha dos maiores coeficientes da DCT selecionados por um módulo especial no programa que pesquisa diretamente os maiores coeficientes; após o cálculo os coeficientes são codificados pelo Algoritmo de Compressão de HUFFMAN e o arquivo compactado é gerado;

i) **DWT BLOCO ORD**: realiza o cálculo pela DWT para a imagem inteira (não há blocagem) com a escolha dos maiores coeficientes da DWT selecionados por um módulo especial no programa que pesquisa diretamente os maiores coeficientes; após o cálculo os coeficientes são codificados pelo Algoritmo de Compressão de HUFFMAN e o arquivo compactado é gerado;

j) **DWT INT ORD**: realiza o cálculo pela DWT para cada vetor-bloco de imagem com a escolha dos maiores coeficientes da DWT selecionados por um módulo especial no programa que pesquisa diretamente os maiores coeficientes; após o cálculo os coeficientes são codificados pelo Algoritmo de Compressão de HUFFMAN e o arquivo compactado é gerado;

O "*módulo especial* de *pesquisa direta do maior valor*", mencionado acima, leva em consideração, na compressão, o número de coeficientes eliminados no vetor-bloco, estendendo para a imagem inteira, no caso de não haver blocagem.

As tabelas e gráficos relativos aos blocos de imagem 2x2 e 4x4 não possuem o processamento mencionado no item (i) tendo em vista que não há cálculo da WAVELET para uma dimensão de bloco inferior ou igual a 4 PIXELS, porque a base WAVELET utilizada (DAUBECHIES4) possui apenas 4 elementos (a decomposição já estaria então concluída). A seguir apresentam-se as tabelas obtidas nos experimentos realizados e na Seção 7 foram apresentados os comentários sobre os resultados obtidos.

5.1 RESULTADOS OBTIDOS COM A IMAGEM LENNA

As tabelas e gráficos apresentados abaixo foram obtidos utilizando-se a imagem padrão LENNA (Figura 16), de dimensões 512 x 512 (262.144 PIXELS) com 256 níveis de escala de cinza. Essa é uma imagem bastante utilizada em vários artigos para apresentação e comparação de resultados e também por isso foi utilizada neste trabalho.

Figura 16 – Figura LENNA (512 x 512)

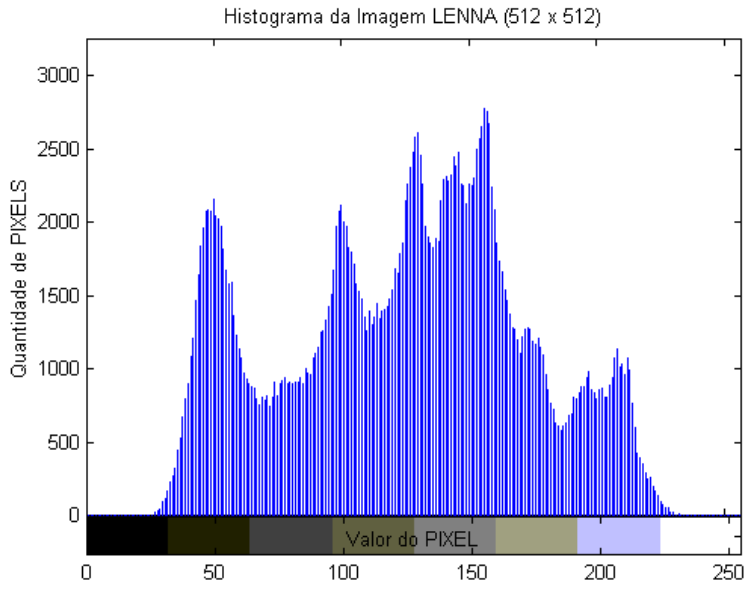


Figura 17 – Histograma da Imagem LENNA (512x512)

Como se pode observar no Histograma da Figura 17, não há uma concentração excessiva dos valores da escala de cinza nesta imagem. O valor máximo da escala de cinza aparece em apenas 1 % dos pontos da imagem (2789 PIXELS) com vários outros valores apresentando também picos locais de densidade de PIXELS. Essa uniformidade de distribuição dos PIXELS foi outro fator importante na escolha dessa imagem.

A Tabela 2 em conjunto com a Figura 18 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 2 – PSNR da Imagem LENNA (BLOCO 2x2)

Imagem:	**lenag**	Bloco:	**2**	x	**2**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
KLT JACOBI + HUFFMAN		31,756868	36,347008	43,385735	58,654776
KLT QL + HUFFMAN		31,756868	36,347065	43,385741	58,655337
KLT JACOBI + ORDEM + HUFFMAN		31,423039	35,725798	42,793060	58,654776
DCT BLOCO+ZIGZAG+HUFFMAN		31,756454	36,222388	43,473127	65,144493
DCT BLOCO+ORDEM+HUFFMAN		31,756454	38,574303	45,950772	65,144493
DCT INTEIRA+ZIGZAG+HUFFMAN		36,465535	42,049816	48,157309	58,937419
DCT INTEIRA+ORDEM+HUFFMAN		40,446750	47,524180	53,761617	58,937419
DWT INTEIRA+ORDEM+HUFFMAN		44,331596	52,346142	59,073790	59,073790

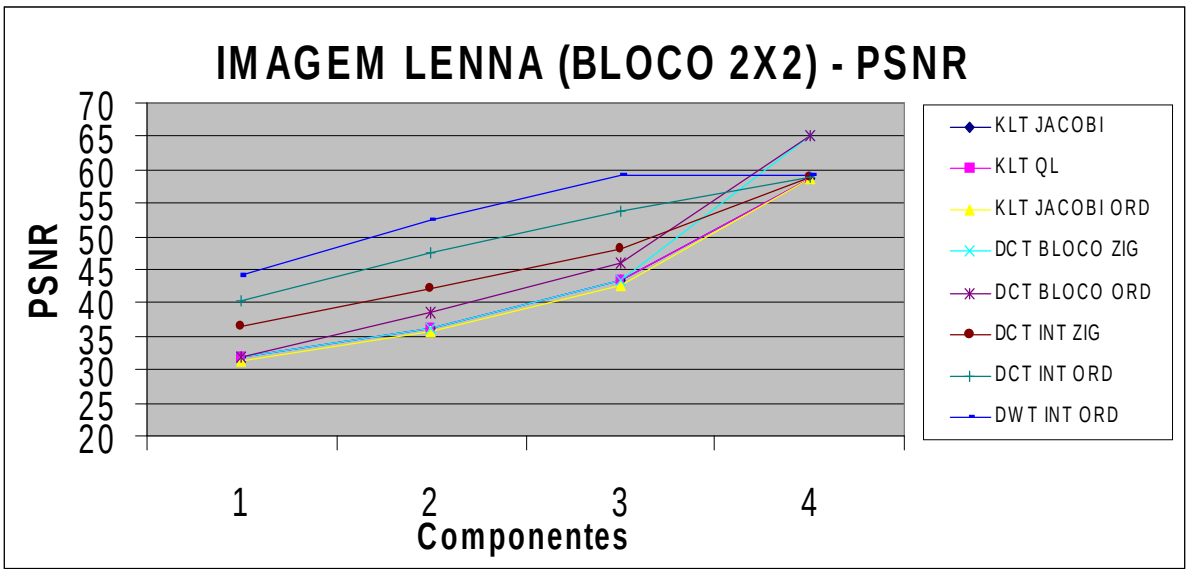


Figura 18 – PSNR da Imagem LENNA (Bloco 2x2)

A Tabela 3 em conjunto com a Figura 19 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 3 – CR da Imagem LENNA (BLOCO 2x2)

Imagem:	**lenag**	Bloco:	**2**	x	**2**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,037250	1,037250	1,037250	1,037250
KLT JACOBI + HUFFMAN		2,237864	1,628547	1,357990	1,276488
KLT QL + HUFFMAN		2,237864	1,628547	1,357990	1,276488
KLT JACOBI + ORDEM + HUFFMAN		2,232928	1,562853	1,334249	1,276488
DCT BLOCO+ZIGZAG+HUFFMAN		1,873173	1,290728	1,092915	1,041375
DCT BLOCO+ORDEM+HUFFMAN		1,873173	1,255597	1,083396	1,041375
DCT INTEIRA+ZIGZAG+HUFFMAN		2,131214	1,619987	1,383035	1,310149
DCT INTEIRA+ORDEM+HUFFMAN		2,180759	1,625479	1,352186	1,310149
DWT INTEIRA+ORDEM+HUFFMAN		2,193827	1,672514	1,446704	1,446704

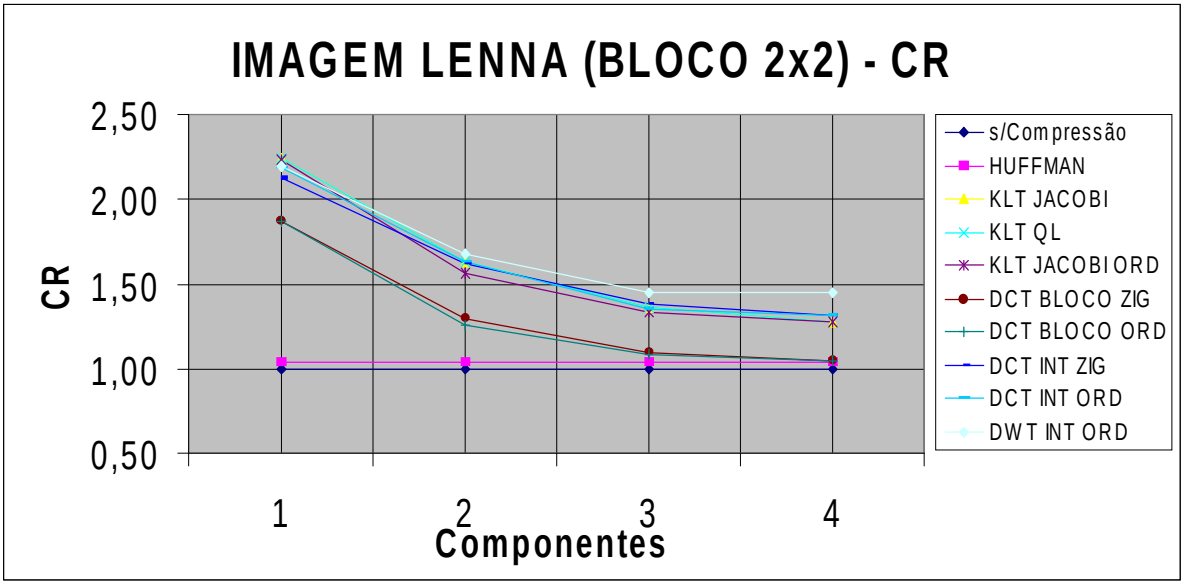


Figura 19 – CR da Imagem LENNA (Bloco 2x2)

A Tabela 4 em conjunto com a Figura 20 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 4 – Tempo (seg) de Gravação da Imagem LENNA (BLOCO 2x2)

Imagem:	**lenag**	Bloco:	**2**	x	**2**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,38	0,44	0,39	0,60
KLT JACOBI + HUFFMAN		0,55	0,55	0,66	0,66
KLT QL + HUFFMAN		0,50	0,55	0,55	0,61
KLT JACOBI + ORDEM + HUFFMAN		1,15	1,20	1,26	1,26
DCT BLOCO+ZIGZAG+HUFFMAN		0,99	1,04	1,05	1,05
DCT BLOCO+ORDEM+HUFFMAN		2,59	2,64	2,64	2,69
DCT INTEIRA+ZIGZAG+HUFFMAN		0,77	0,76	0,76	0,82
DCT INTEIRA+ORDEM+HUFFMAN		0,88	0,93	0,88	0,88
DWT INTEIRA+ORDEM+HUFFMAN		0,77	0,82	0,77	0,77

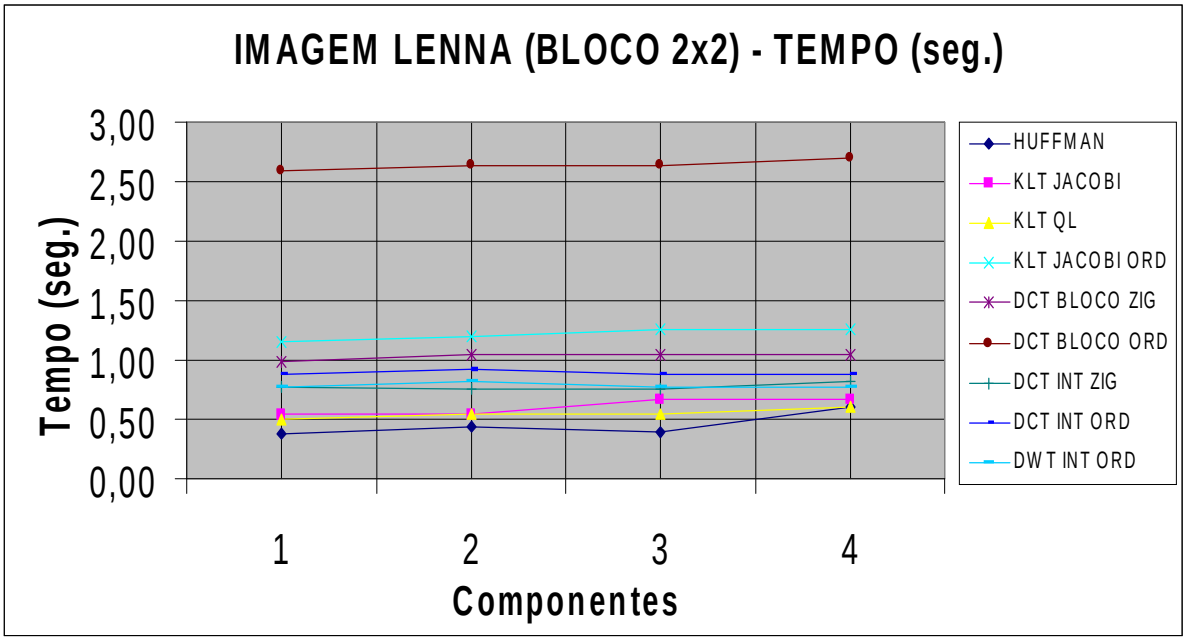


Figura 20 – Tempo (seg) de Gravação da Imagem LENNA (Bloco 2x2)

A Tabela 5 em conjunto com a Figura 21 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 5 – PSNR da Imagem LENNA (BLOCO 4x4)

Imagem:	**lenag**	Bloco:	**4**	x	**4**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
KLT JACOBI + HUFFMAN		26,977861	34,001177	40,448269	59,217773
KLT QL + HUFFMAN		26,977861	34,001153	40,448388	59,217347
KLT JACOBI + ORDEM + HUFFMAN		26,643227	33,600013	40,092550	59,217773
DCT BLOCO+ZIGZAG+HUFFMAN		26,968270	32,607514	39,907130	61,790802
DCT BLOCO+ORDEM+HUFFMAN		26,968270	36,710627	46,189032	61,790802
DCT INTEIRA+ZIGZAG+HUFFMAN		29,813543	36,465535	42,049816	58,937419
DCT INTEIRA+ORDEM+HUFFMAN		32,570055	40,446750	47,524180	58,937419
DWT INTEIRA+ORDEM+HUFFMAN		35,114401	44,331596	52,346142	59,073790

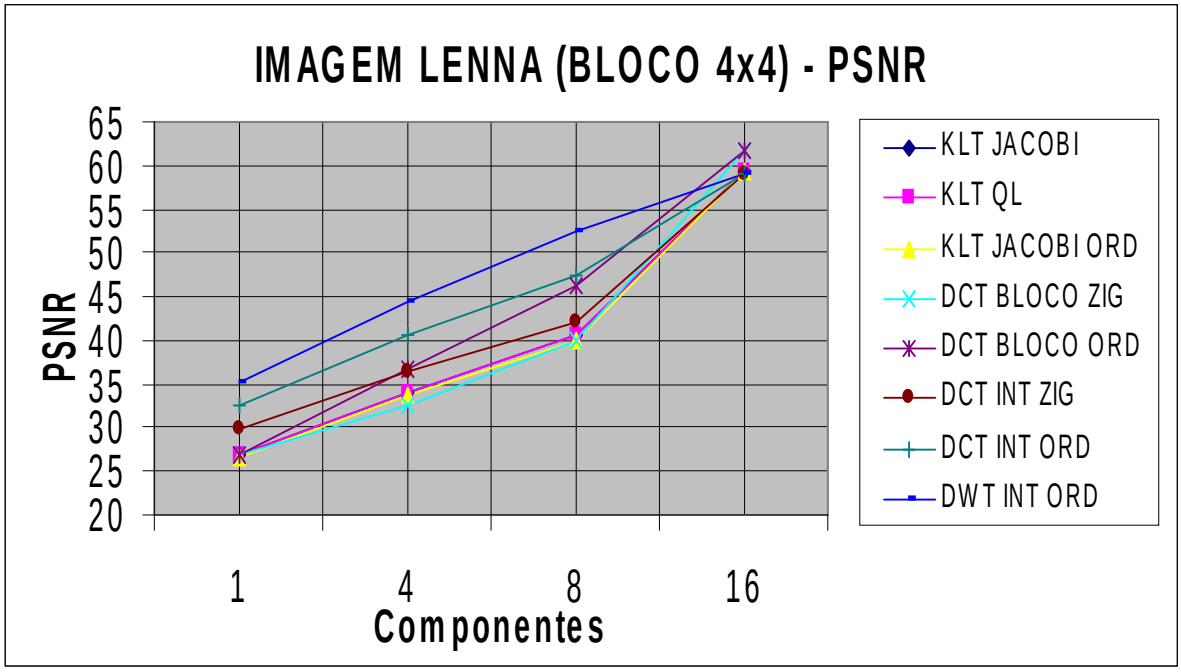


Figura 21 – PSNR da Imagem LENNA (Bloco 4x4)

A Tabela 6 em conjunto com a Figura 22 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 6 – CR da Imagem LENNA (BLOCO 4x4)

Imagem:	**lenag**	Bloco:	**4**	x	**4**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,037250	1,037250	1,037250	1,037250
KLT JACOBI + HUFFMAN		3,249093	2,218905	1,718776	1,454595
KLT QL + HUFFMAN		3,249093	2,218867	1,719315	1,454788
KLT JACOBI + ORDEM + HUFFMAN		3,251541	2,148962	1,640646	1,454595
DCT BLOCO+ZIGZAG+HUFFMAN		2,402976	1,562445	1,289015	1,130387
DCT BLOCO+ORDEM+HUFFMAN		2,402976	1,519231	1,240028	1,130387
DCT INTEIRA+ZIGZAG+HUFFMAN		3,063000	2,131214	1,619987	1,310149
DCT INTEIRA+ORDEM+HUFFMAN		3,158715	2,180759	1,625479	1,310149
DWT INTEIRA+ORDEM+HUFFMAN		3,119372	2,193827	1,672514	1,446704

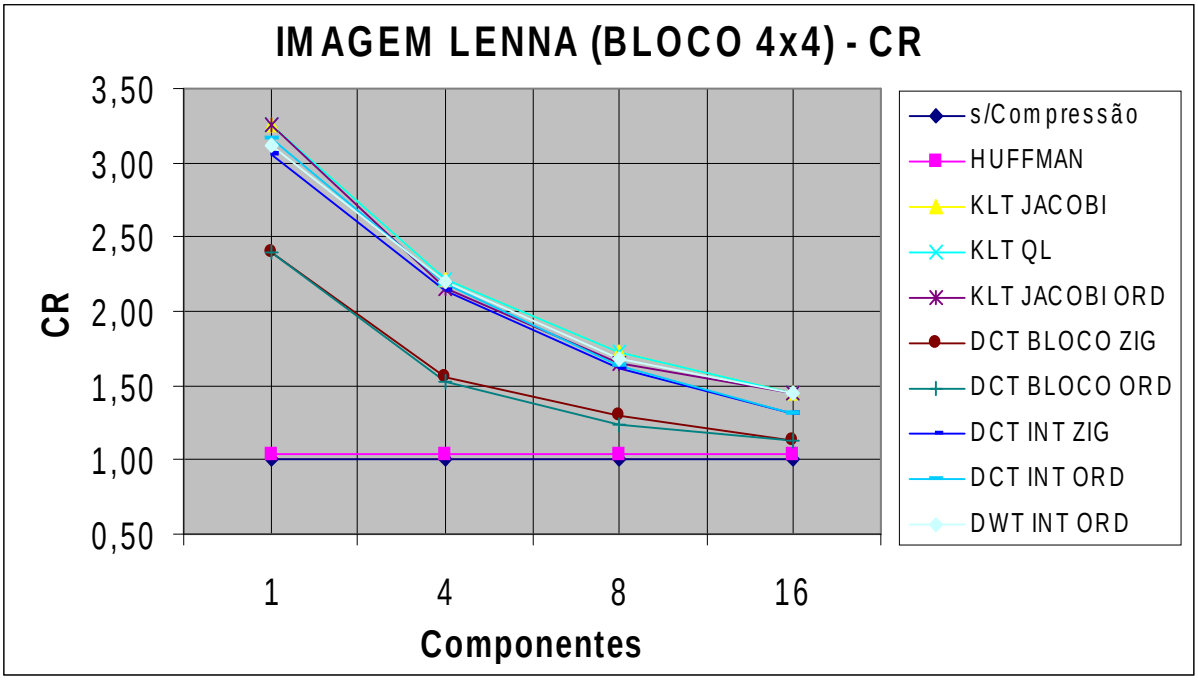


Figura 22 – CR da Imagem LENNA (BLOCO 4x4)

A Tabela 7 em conjunto com a Figura 23 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 7 – Tempo (seg) de Gravação da Imagem LENNA (BLOCO 4x4)

Imagem:	**lenag**	Bloco:	**4**	x	**4**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,38	0,39	0,44	0,39
KLT JACOBI + HUFFMAN		0,71	0,77	0,77	0,83
KLT QL + HUFFMAN		0,72	0,71	0,77	0,83
KLT JACOBI + ORDEM + HUFFMAN		0,99	1,05	1,10	1,10
DCT BLOCO+ZIGZAG+HUFFMAN		0,77	0,94	0,93	0,99
DCT BLOCO+ORDEM+HUFFMAN		1,65	1,75	1,86	1,93
DCT INTEIRA+ZIGZAG+HUFFMAN		0,71	0,82	0,88	0,88
DCT INTEIRA+ORDEM+HUFFMAN		0,77	0,93	0,99	0,94
DWT INTEIRA+ORDEM+HUFFMAN		0,71	0,72	0,77	0,82

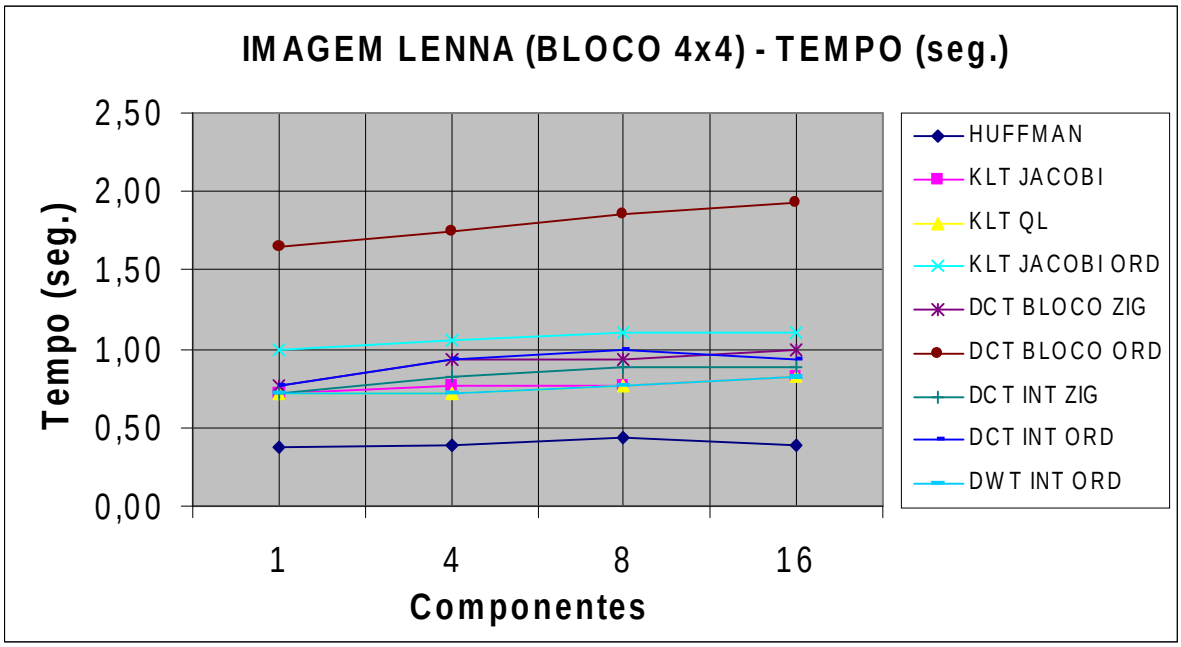


Figura 23 – Tempo (seg) de Gravação da Imagem LENNA (BLOCO 4x4)

A Tabela 8 em conjunto com a Figura 24 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 8 – PSNR da Imagem LENNA (BLOCO 8x8)

Imagem:	**lenag**	Bloco:	**8**	x	**8**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
KLT JACOBI + HUFFMAN		23,710419	35,949811	42,785036	59,447149
KLT QL + HUFFMAN		23,710422	35,949734	42,785820	59,433487
KLT JACOBI + ORDEM + HUFFMAN		23,388763	35,603584	42,579042	59,447149
DCT BLOCO+ZIGZAG+HUFFMAN		23,683710	35,033327	41,725115	62,424375
DCT BLOCO+ORDEM+HUFFMAN		23,683710	41,100670	59,372643	62,424375
DCT INTEIRA+ZIGZAG+HUFFMAN		25,914179	36,465535	42,049816	58,937419
DCT INTEIRA+ORDEM+HUFFMAN		27,814875	40,446750	47,524180	58,937419
DWT INTEIRA+ORDEM+HUFFMAN		29,042612	44,331596	52,346142	59,073790
DWT BLOCO+ORDEM+HUFFMAN		7,676320	36,114649	44,123113	59,104363

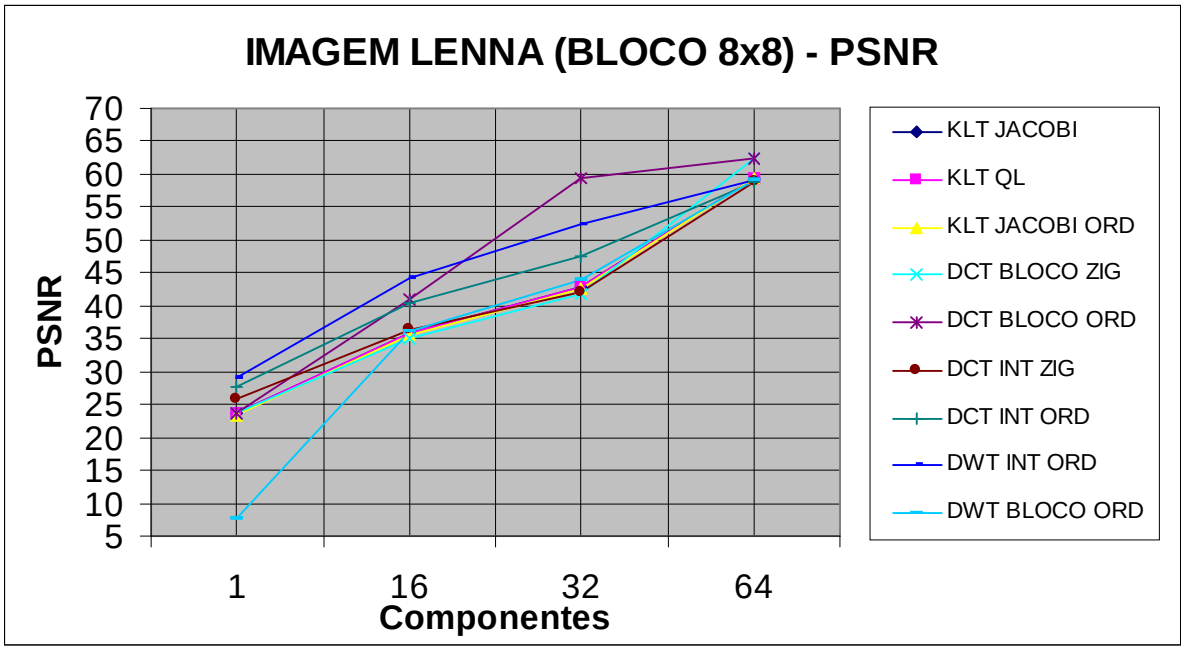


Figura 24 – PSNR da Imagem LENNA (BLOCO 8x8)

A Tabela 9 em conjunto com a Figura 25 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 9 – CR da Imagem LENNA (BLOCO 8x8)

Imagem:	**lenag**	Bloco:	**8**	x	**8**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,037250	1,037250	1,037250	1,037250
KLT JACOBI + HUFFMAN		3,467965	2,040797	1,624045	1,388382
KLT QL + HUFFMAN		3,467965	2,036235	1,621055	1,386284
KLT JACOBI + ORDEM + HUFFMAN		3,456761	1,965766	1,556007	1,388382
DCT BLOCO+ZIGZAG+HUFFMAN		3,846194	2,021286	1,818849	1,735904
DCT BLOCO+ORDEM+HUFFMAN		3,846194	1,878346	1,738805	1,735904
DCT INTEIRA+ZIGZAG+HUFFMAN		3,560615	2,131214	1,619987	1,310149
DCT INTEIRA+ORDEM+HUFFMAN		3,806371	2,180759	1,625479	1,310149
DWT INTEIRA+ORDEM+HUFFMAN		3,768767	2,193827	1,672514	1,446704
DWT BLOCO+ORDEM+HUFFMAN		4,068283	1,903351	1,465911	1,280412

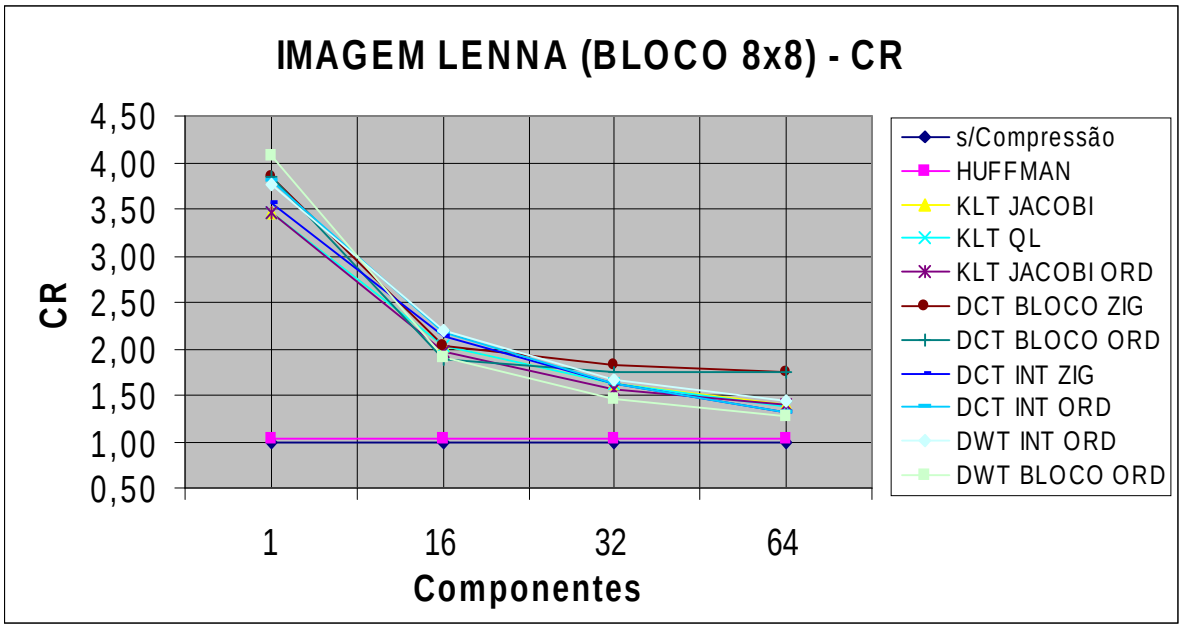


Figura 25 – CR da Imagem LENNA (BLOCO 8x8)

A Tabela 10 em conjunto com a Figura 26 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 10 – Tempo de Gravação (seg) da Imagem LENNA (BLOCO 8x8)

Imagem:	**lenag**	Bloco:	**8**	x	**8**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,44	0,44	0,44	0,44
KLT JACOBI + HUFFMAN		1,82	1,92	1,92	1,93
KLT QL + HUFFMAN		1,70	1,82	1,86	1,82
KLT JACOBI + ORDEM + HUFFMAN		1,97	2,03	2,08	2,08
DCT BLOCO+ZIGZAG+HUFFMAN		0,65	0,82	0,77	0,77
DCT BLOCO+ORDEM+HUFFMAN		1,05	1,26	1,21	1,21
DCT INTEIRA+ZIGZAG+HUFFMAN		0,66	0,77	0,77	0,88
DCT INTEIRA+ORDEM+HUFFMAN		0,77	0,88	0,88	0,93
DWT INTEIRA+ORDEM+HUFFMAN		0,71	0,77	0,77	0,77
DWT BLOCO+ORDEM+HUFFMAN		0,88	0,99	1,04	1,04

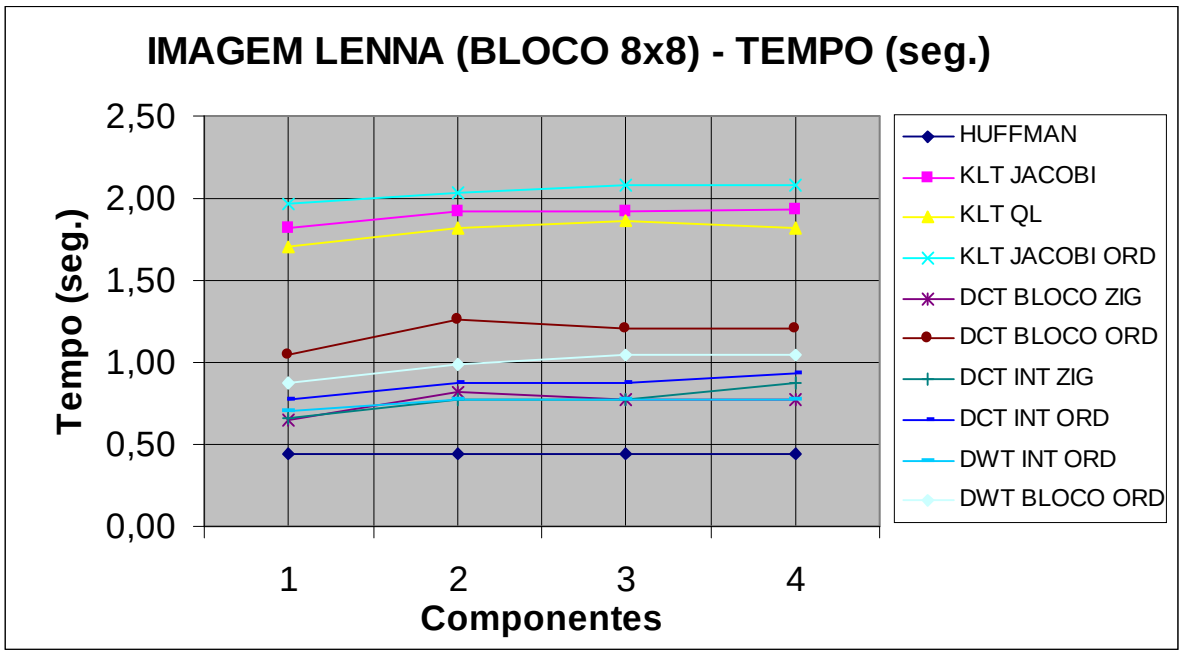


Figura 26 – Tempo de Gravação (seg) da Imagem LENNA (BLOCO 8x8)

A Tabela 11 em conjunto com a Figura 27 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 11 – PSNR da Imagem LENNA (BLOCO 16x16)

Imagem:	**lenag**	Bloco:	**16**	x	**16**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
KLT JACOBI + HUFFMAN		21,036369	29,904226	45,606290	59,324618
KLT QL + HUFFMAN		21,036369	29,904237	45,606938	59,325490
KLT JACOBI + ORDEM + HUFFMAN		20,719569	29,601264	45,402072	59,324618
DCT BLOCO+ZIGZAG+HUFFMAN		20,979116	29,024011	41,889330	60,091639
DCT BLOCO+ORDEM+HUFFMAN		20,979116	32,216879	50,508523	60,091639
DCT INTEIRA+ZIGZAG+HUFFMAN		22,656704	29,813543	42,049816	58,937419
DCT INTEIRA+ORDEM+HUFFMAN		24,401438	32,570055	47,524180	58,937419
DWT INTEIRA+ORDEM+HUFFMAN		24,707564	35,114401	52,346142	59,073790
DWT BLOCO+ORDEM+HUFFMAN		7,693077	29,494727	45,958099	59,083888

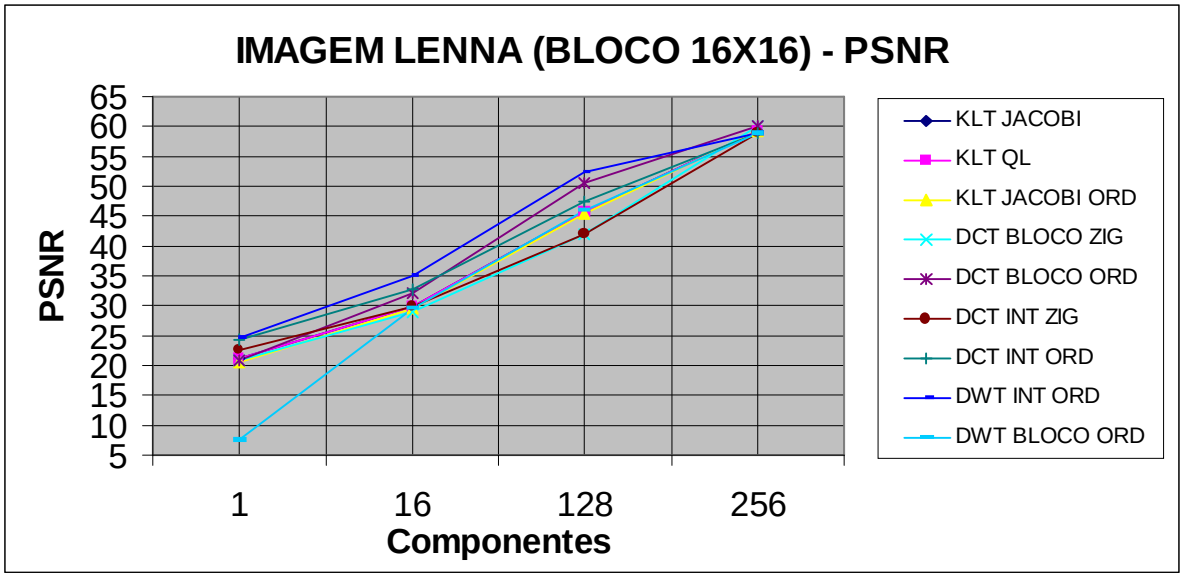


Figura 27 – PSNR da Imagem LENNA (BLOCO 16x16)

A Tabela 12 em conjunto com a Figura 28 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 12 – CR da Imagem LENNA (BLOCO 16x16)

Imagem:	**lenag**	Bloco:	**16**	x	**16**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,037250	1,037250	1,037250	1,037250
KLT JACOBI + HUFFMAN		3,927397	2,301374	0,892771	0,585193
KLT QL + HUFFMAN		3,927397	2,292675	0,891544	0,584670
KLT JACOBI + ORDEM + HUFFMAN		3,918919	2,289086	0,869753	0,585193
DCT BLOCO+ZIGZAG+HUFFMAN		4,074392	2,327277	1,531375	1,323355
DCT BLOCO+ORDEM+HUFFMAN		4,074392	2,304316	1,420334	1,323355
DCT INTEIRA+ZIGZAG+HUFFMAN		4,304459	3,063000	1,619987	1,310149
DCT INTEIRA+ORDEM+HUFFMAN		4,564834	3,158715	1,625479	1,310149
DWT INTEIRA+ORDEM+HUFFMAN		4,547798	3,119372	1,672514	1,446704
DWT BLOCO+ORDEM+HUFFMAN		4,342737	2,182586	1,321734	1,183174

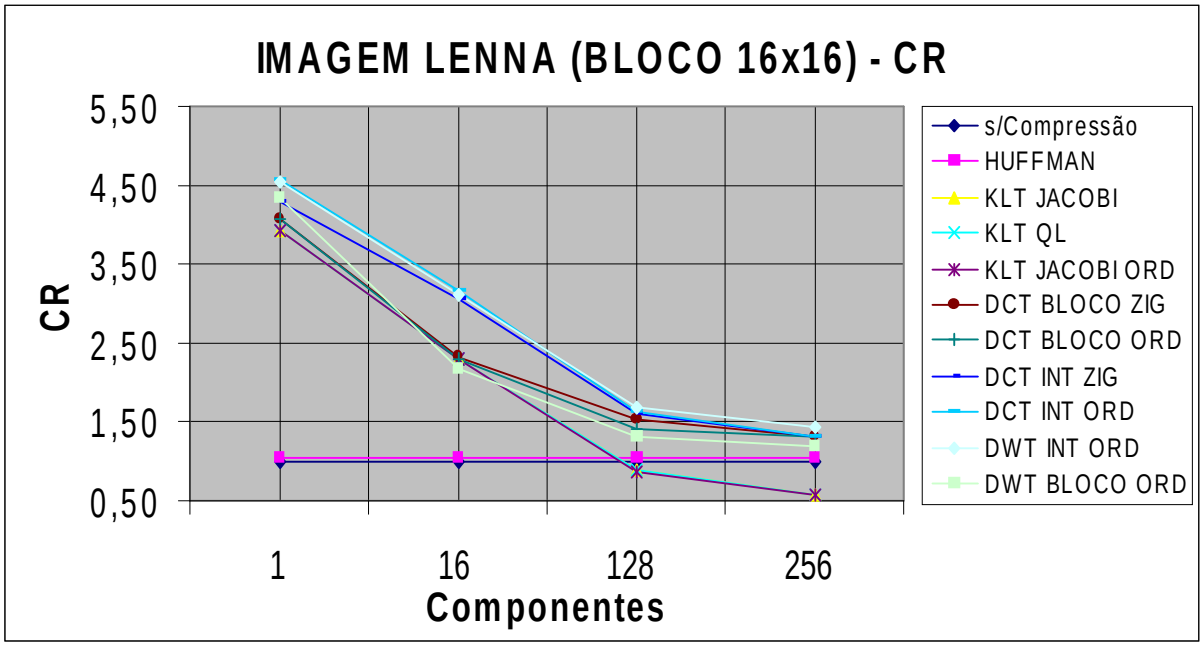


Figura 28 – CR da Imagem LENNA (BLOCO 16x16)

A Tabela 13 em conjunto com a Figura 29 apresentam os resultados obtidos para a imagem LENNA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 13 – Tempo de Gravação (seg) da Imagem LENNA (BLOCO 16x16)

Imagem:	**lenag**	Bloco:	**16**	x	**16**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,60	0,44	0,44	0,38
KLT JACOBI + HUFFMAN		28,45	27,96	26,86	28,07
KLT QL + HUFFMAN		11,75	11,98	12,25	11,53
KLT JACOBI + ORDEM + HUFFMAN		28,12	29,71	29,60	26,80
DCT BLOCO+ZIGZAG+HUFFMAN		0,66	0,71	0,77	0,77
DCT BLOCO+ORDEM+HUFFMAN		0,88	0,93	1,04	1,04
DCT INTEIRA+ZIGZAG+HUFFMAN		0,71	0,77	0,82	0,88
DCT INTEIRA+ORDEM+HUFFMAN		0,71	0,77	0,93	0,93
DWT INTEIRA+ORDEM+HUFFMAN		0,60	0,72	0,76	0,77
DWT BLOCO+ORDEM+HUFFMAN		0,71	0,82	0,93	0,99

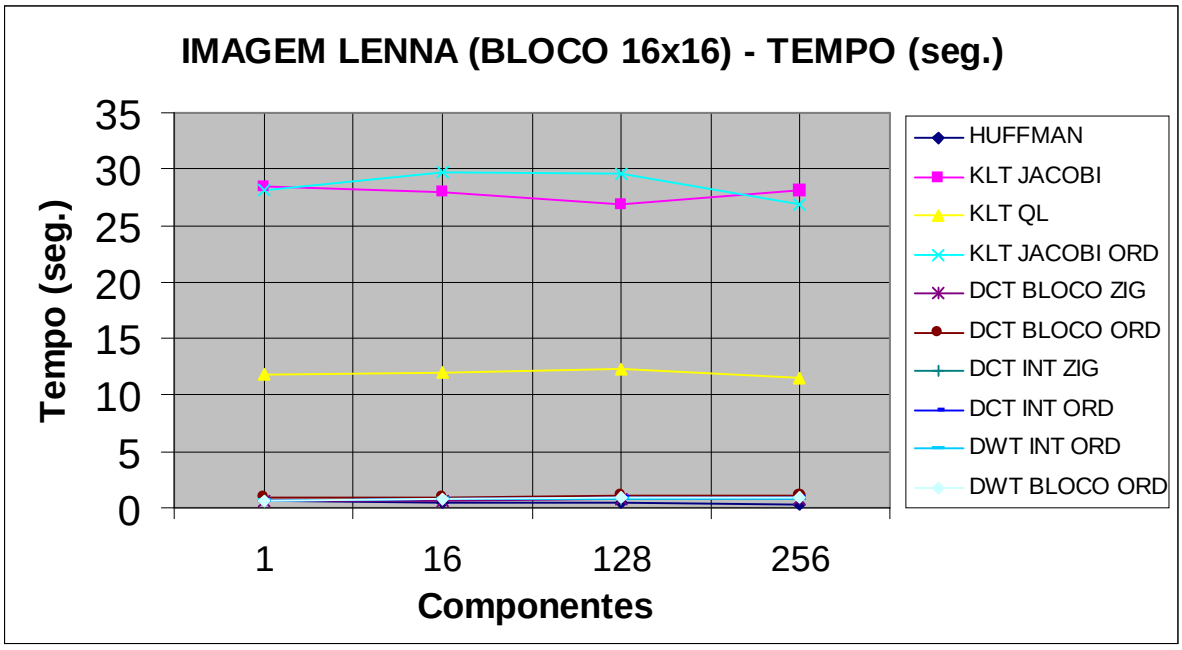


Figura 29 – Tempo de Gravação (seg) da Imagem LENNA (BLOCO 16x16)

A Tabela 14 em conjunto com a Figura 30 apresentam os resultados da *PSNR (Razão Sinal/Ruído de Pico)* obtida para a imagem LENNA (512x512 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e compressão QL.

Tabela 14 – PSNR da Imagem LENNA – Compressão QL

Imagem:	**lenag**			
RAZÃO SINAL/RUÍDO DE PICO (PSNR)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	31,756868	34,001153	35,949734	37,898771
4:2	36,347065	40,448388	42,785820	45,606938
4:3	43,385741	48,691906	52,436503	55,482310
4:4	58,655337	59,217347	59,433487	59,325490

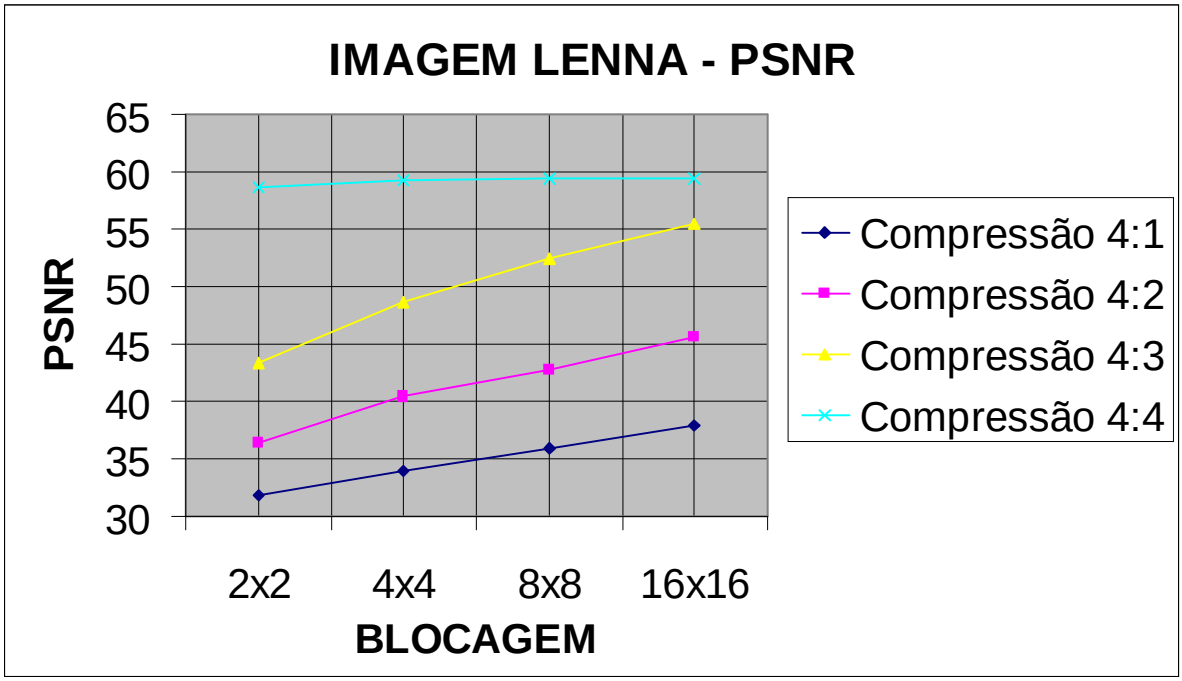


Figura 30 – PSNR da Imagem LENNA – Compressão QL

A Tabela 15 em conjunto com a Figura 31 apresentam os resultados da *CR (Razão de Compressão)* obtida para a imagem LENNA (512x512 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e compressão QL.

Tabela 15 – CR da Imagem LENNA – Compressão QL

Imagem:	lenag			
RAZÃO DE COMPRESSÃO (CR)				
Compressão	Blocagem da Imagem			
	2x2	4x4	8x8	16x16
4:1	2,237864	2,218867	2,036235	1,336383
4:2	1,628547	1,719315	1,621055	0,891544
4:3	1,357990	1,512379	1,451074	0,688241
4:4	1,276488	1,454788	1,386284	0,584670

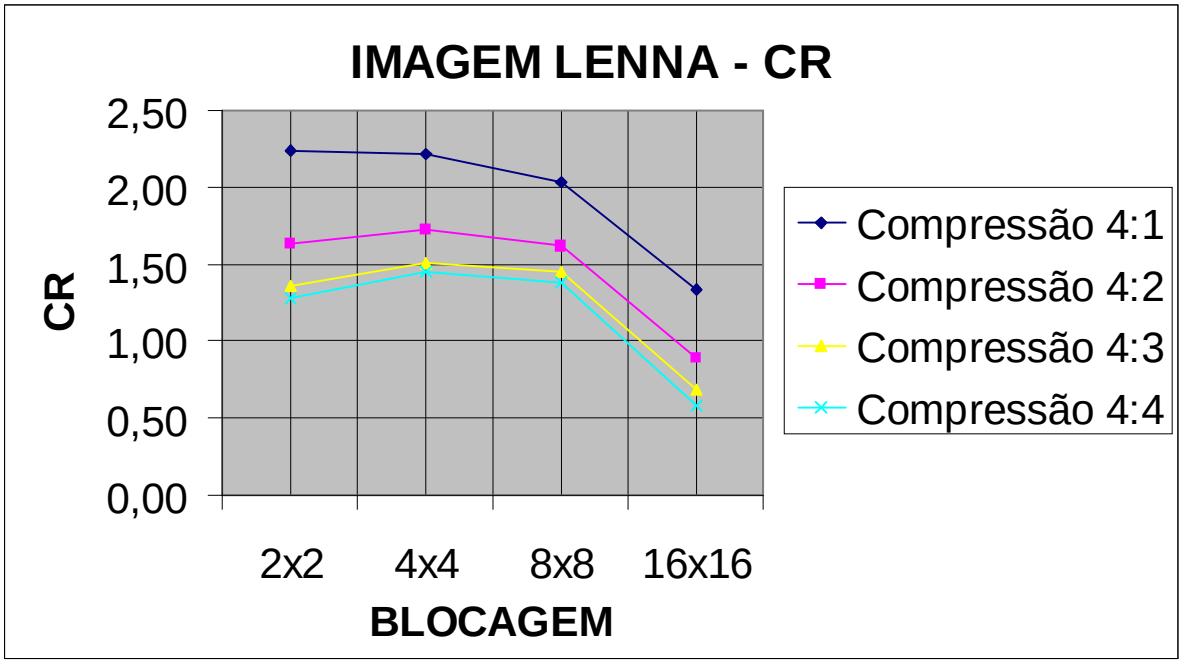


Figura 31 – CR da Imagem LENNA – Compressão QL

A Tabela 16 em conjunto com a Figura 32 apresentam os resultados do *Tempo (em segundos)* obtida para a imagem LENNA (512x512 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e compressão QL.

Tabela 16 – Tempo de Gravação da Imagem LENNA – Compressão QL

Imagem:	lenag			
TEMPO DE GRAVAÇÃO (SEGUNDOS)				
Compressão	Blocagem da Imagem			
	2x2	4x4	8x8	16x16
4:1	0,50	0,71	1,820000	11,920000
4:2	0,55	0,77	1,860000	12,250000
4:3	0,55	0,77	1,870000	13,080000
4:4	0,61	0,83	1,820000	11,530000

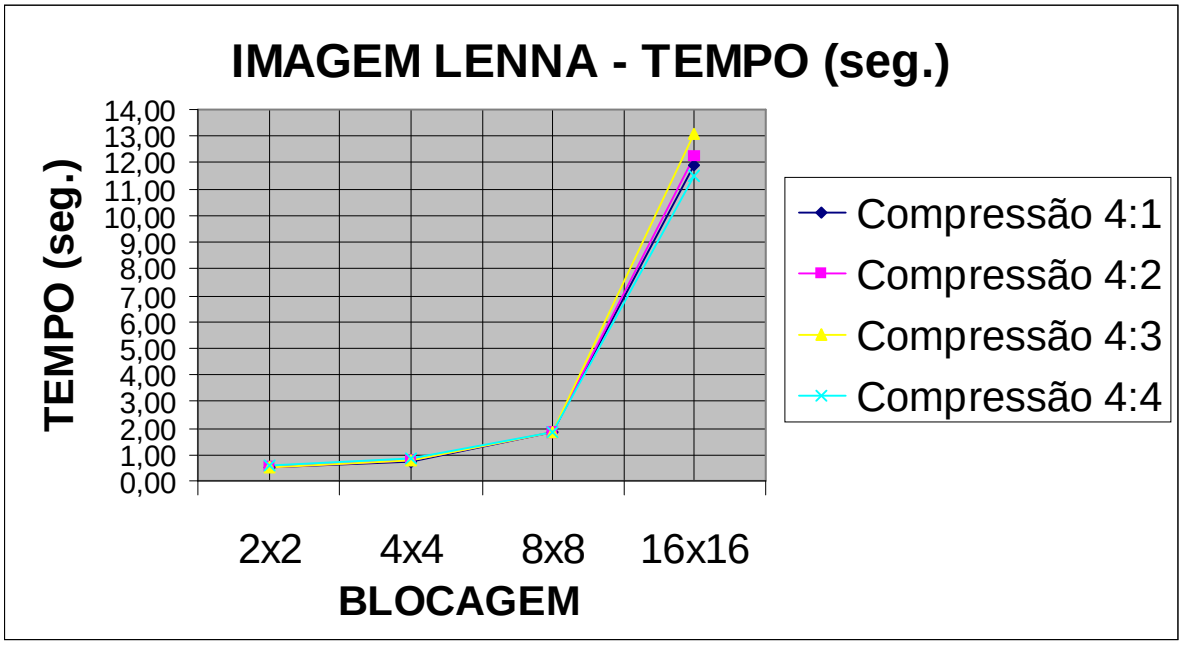


Figura 32 – Tempo de Gravação da Imagem LENNA – Compressão QL

5.2 RESULTADOS OBTIDOS COM A IMAGEM BABUÍNO

As tabelas e gráficos apresentados abaixo foram obtidos utilizando-se a imagem padrão BABUÍNO (BABOON) (Figura 33), de dimensões 512 x 512 (262.144 PIXELS) com 256 níveis de escala de cinza.

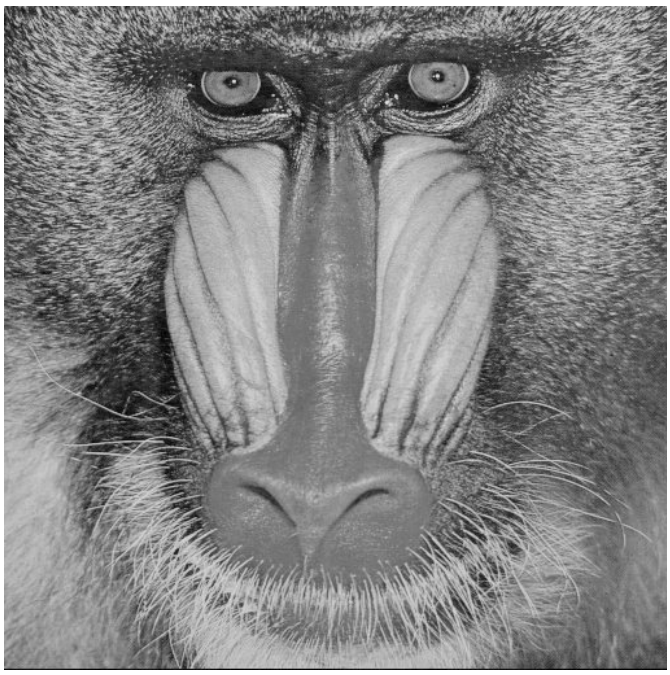


Figura 33 – Figura BABUÍNO (512 x 512)

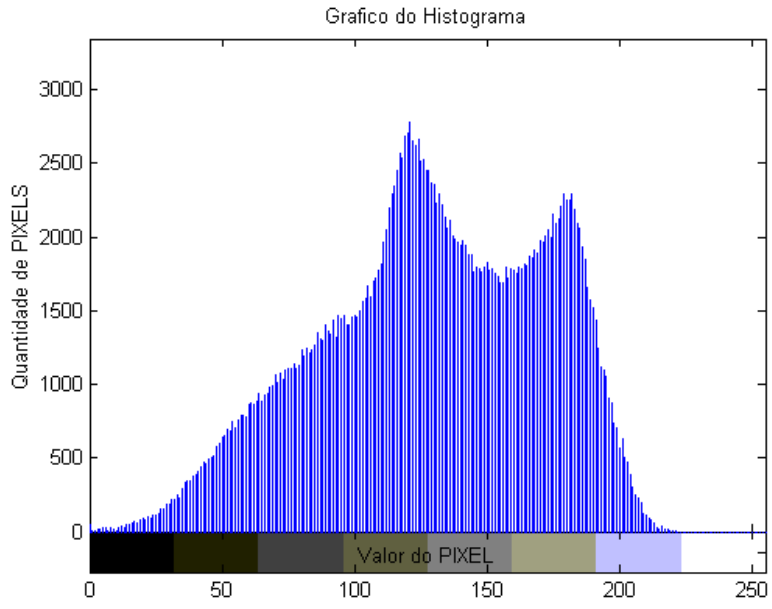


Figura 34 – Histograma da Imagem BABUÍNO (512x512)

Como se pode observar no Histograma da Figura 34, existe uma concentração de PIXELS entre dois valores principais da escala de cinza desta imagem. O valor máximo da escala de cinza aparece em apenas 1 % dos pontos da imagem (2796 PIXELS) com vários outros valores apresentando também picos locais de densidade de PIXELS.

A Tabela 17 em conjunto com a Figura 35 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 17 – PSNR da Imagem BABUÍNO (BLOCO 2x2)

Imagem:	**babuinog**	Bloco:	**2**	x	**2**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
KLT JACOBI + HUFFMAN		23,192588	27,098019	31,387091	58,954609
KLT QL + HUFFMAN		23,192588	27,098019	31,387090	58,955410
KLT JACOBI + ORDEM + HUFFMAN		22,707942	26,526111	30,800362	58,954609
DCT BLOCO+ZIGZAG+HUFFMAN		23,190625	24,470783	31,388897	64,867017
DCT BLOCO+ORDEM+HUFFMAN		23,190625	29,770087	36,022230	64,867017
DCT INTEIRA+ZIGZAG+HUFFMAN		24,617184	28,334416	33,358186	58,944407
DCT INTEIRA+ORDEM+HUFFMAN		28,458544	34,987259	44,233276	58,944407
DWT INTEIRA+ORDEM+HUFFMAN		30,732287	38,656004	48,911563	58,956011

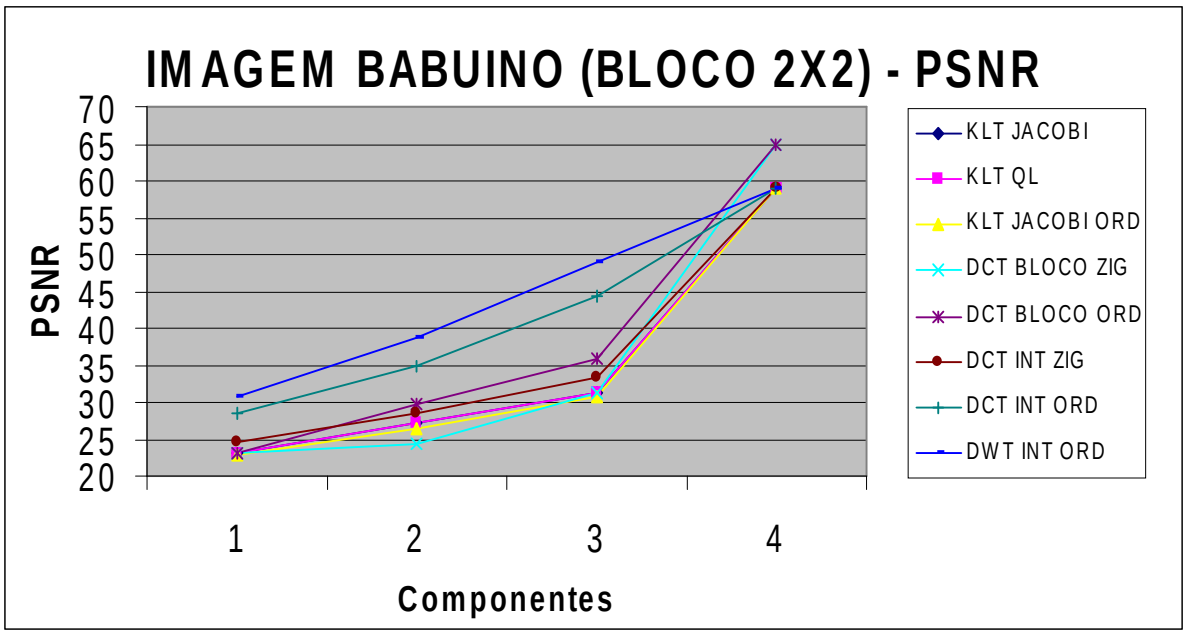


Figura 35 – PSNR da Imagem BABUÍNO (Bloco 2x2)

A Tabela 18 em conjunto com a Figura 36 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 18 – CR da Imagem BABUÍNO (BLOCO 2x2)

Imagem:	**babuinog**	Bloco:	**2**	x	**2**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,046292	1,046292	1,046292	1,046292
KLT JACOBI + HUFFMAN		2,278722	1,521523	1,202262	1,074586
KLT QL + HUFFMAN		2,278722	1,529294	1,207053	1,078443
KLT JACOBI + ORDEM + HUFFMAN		2,255467	1,482072	1,174988	1,074586
DCT BLOCO+ZIGZAG+HUFFMAN		1,908609	1,234712	0,962340	0,873380
DCT BLOCO+ORDEM+HUFFMAN		1,908609	1,170812	0,949698	0,873380
DCT INTEIRA+ZIGZAG+HUFFMAN		2,097637	1,465609	1,172183	1,055628
DCT INTEIRA+ORDEM+HUFFMAN		2,202584	1,515653	1,188828	1,055628
DWT INTEIRA+ORDEM+HUFFMAN		2,169919	1,500388	1,191088	1,089757

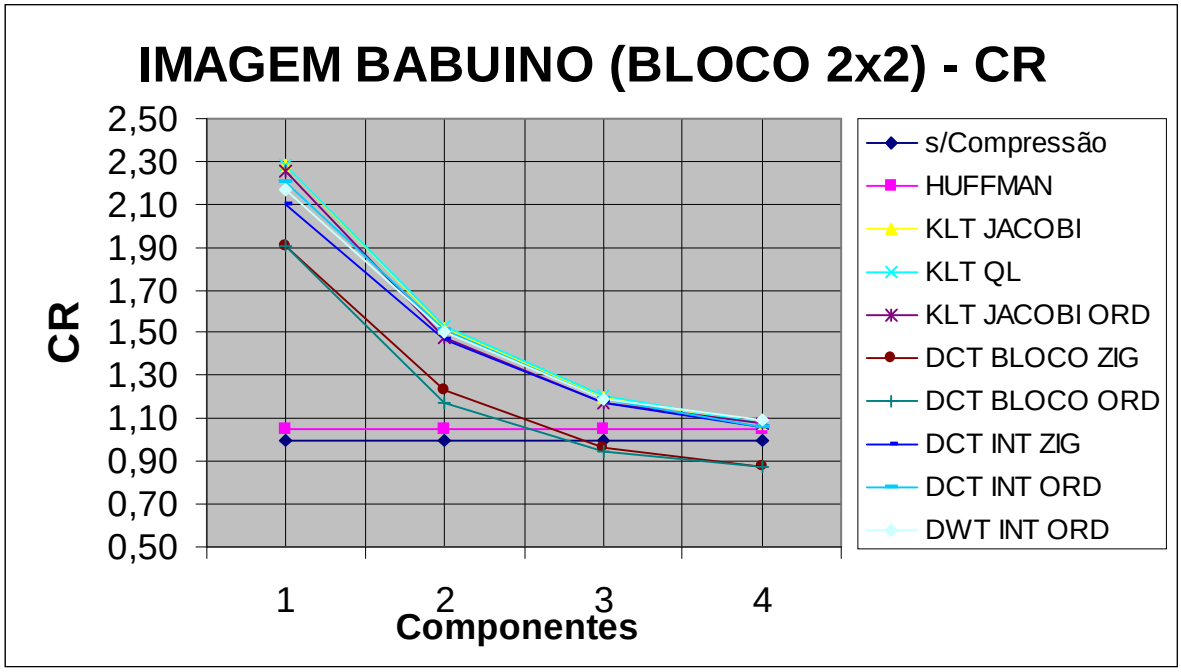


Figura 36 – CR da Imagem BABUÍNO (Bloco 2x2)

A Tabela 19 em conjunto com a Figura 37 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 19 – Tempo (seg) de Gravação da Imagem BABUÍNO (BLOCO 2x2)

Imagem:	**babuinog**	Bloco:	**2**	x	**2**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,44	0,38	0,39	0,38
KLT JACOBI + HUFFMAN		0,55	0,61	0,66	0,66
KLT QL + HUFFMAN		0,55	0,55	0,61	0,60
KLT JACOBI + ORDEM + HUFFMAN		1,15	1,20	1,26	1,26
DCT BLOCO+ZIGZAG+HUFFMAN		0,93	0,93	0,99	1,04
DCT BLOCO+ORDEM+HUFFMAN		2,53	2,64	2,69	2,69
DCT INTEIRA+ZIGZAG+HUFFMAN		0,77	0,93	0,88	0,88
DCT INTEIRA+ORDEM+HUFFMAN		0,88	0,88	0,88	0,93
DWT INTEIRA+ORDEM+HUFFMAN		0,71	0,71	0,82	0,83

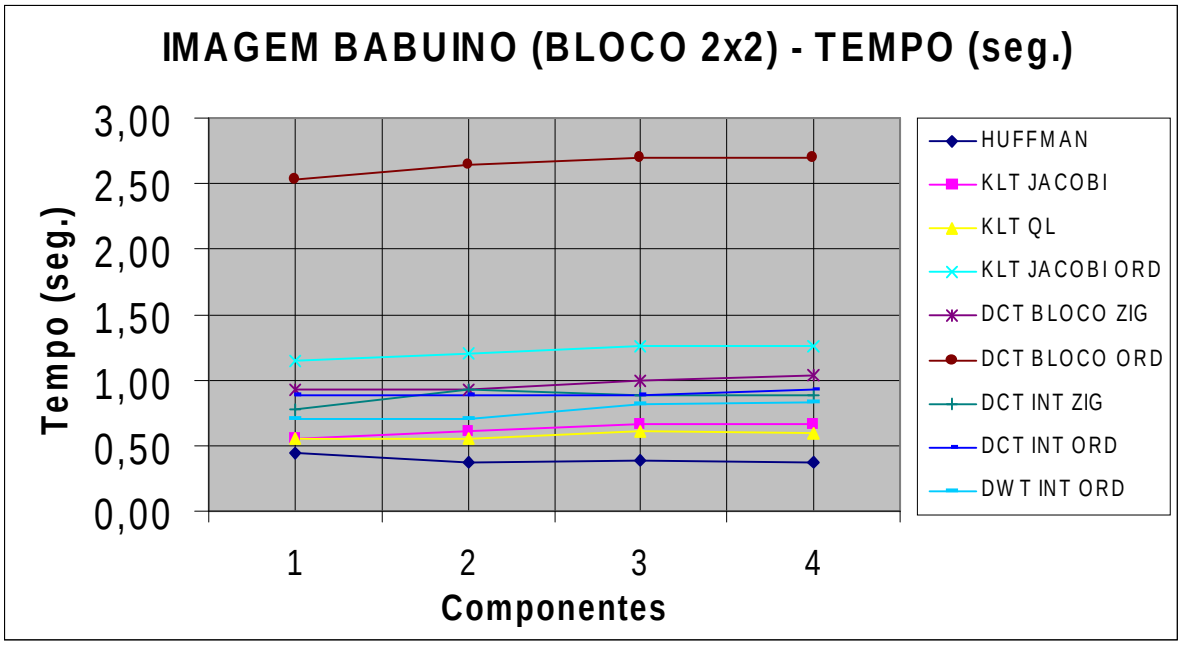


Figura 37 – Tempo (seg) de Gravação da Imagem BABUÍNO (Bloco 2x2)

A Tabela 20 em conjunto com a Figura 38 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 20 – PSNR da Imagem BABUÍNO (BLOCO 4x4)

Imagem:	**babuinog**	Bloco:	**4**	x	**4**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
KLT JACOBI + HUFFMAN		20,750160	24,066834	28,019729	58,957814
KLT QL + HUFFMAN		20,750160	24,066816	28,019715	58,956011
KLT JACOBI + ORDEM + HUFFMAN		20,465975	23,628221	27,628742	58,957814
DCT BLOCO+ZIGZAG+HUFFMAN		20,742997	23,982521	26,393342	61,404459
DCT BLOCO+ORDEM+HUFFMAN		20,742997	26,941741	33,464266	61,404459
DCT INTEIRA+ZIGZAG+HUFFMAN		21,335439	24,617184	28,334416	58,944407
DCT INTEIRA+ORDEM+HUFFMAN		23,125507	28,458544	34,987259	58,944407
DWT INTEIRA+ORDEM+HUFFMAN		24,152180	30,732287	38,656004	58,956011

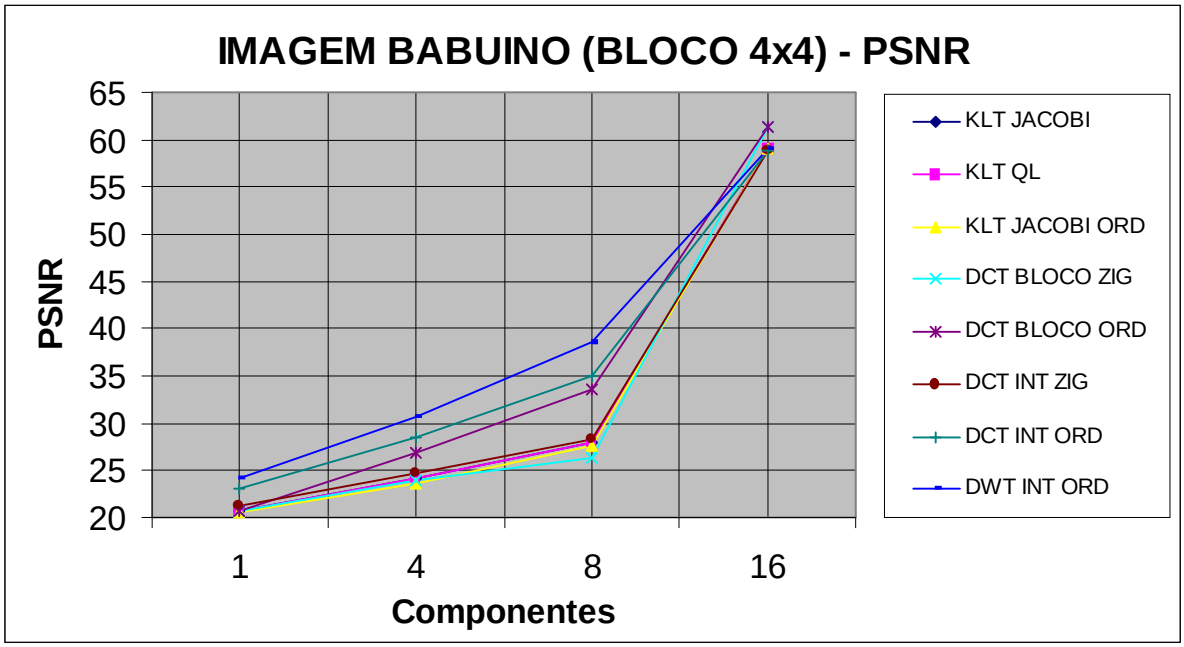


Figura 38 – PSNR da Imagem BABUÍNO (Bloco 4x4)

A Tabela 21 em conjunto com a Figura 39 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 21 – CR da Imagem BABUÍNO (BLOCO 4x4)

Imagem:	**babuinog**	Bloco:	**4**	x	**4**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,046292	1,046292	1,046292	1,046292
KLT JACOBI + HUFFMAN		3,353446	2,115694	1,480972	1,079934
KLT QL + HUFFMAN		3,353446	2,115626	1,481038	1,079960
KLT JACOBI + ORDEM + HUFFMAN		3,372004	2,071978	1,432524	1,079934
DCT BLOCO+ZIGZAG+HUFFMAN		2,541391	1,498329	1,144882	0,879770
DCT BLOCO+ORDEM+HUFFMAN		2,541391	1,477713	1,108340	0,879770
DCT INTEIRA+ZIGZAG+HUFFMAN		3,280679	2,097637	1,465609	1,055628
DCT INTEIRA+ORDEM+HUFFMAN		3,490591	2,202584	1,515653	1,055628
DWT INTEIRA+ORDEM+HUFFMAN		3,443962	2,169919	1,500388	1,089757

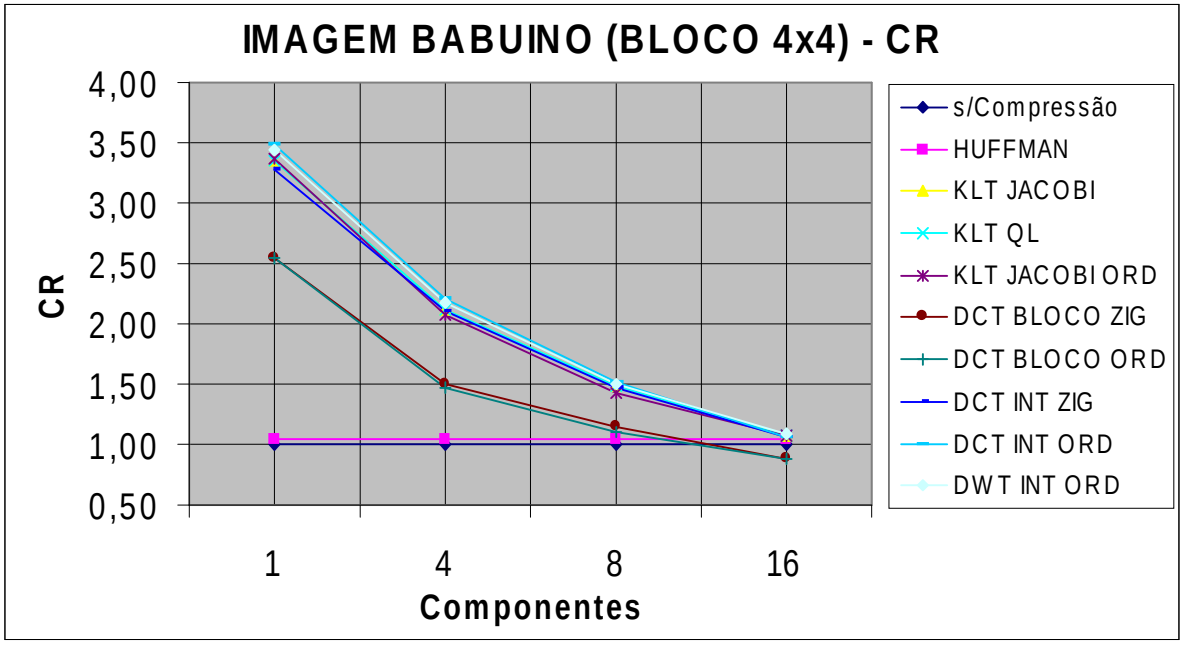


Figura 39 – CR da Imagem BABUÍNO (BLOCO 4x4)

A Tabela 22 em conjunto com a Figura 40 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 22 – Tempo (seg) de Gravação da Imagem BABUÍNO (BLOCO 4x4)

Imagem:	**babuinog**	Bloco:	**4**	x	**4**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,44	0,44	0,38	0,38
KLT JACOBI + HUFFMAN		0,77	0,83	0,82	0,88
KLT QL + HUFFMAN		0,71	0,71	0,77	0,82
KLT JACOBI + ORDEM + HUFFMAN		0,99	1,05	1,09	1,16
DCT BLOCO+ZIGZAG+HUFFMAN		0,83	0,83	0,87	0,99
DCT BLOCO+ORDEM+HUFFMAN		1,70	1,75	1,87	1,87
DCT INTEIRA+ZIGZAG+HUFFMAN		0,71	0,71	0,83	0,88
DCT INTEIRA+ORDEM+HUFFMAN		0,77	0,83	0,88	0,93
DWT INTEIRA+ORDEM+HUFFMAN		0,72	0,71	0,83	0,99

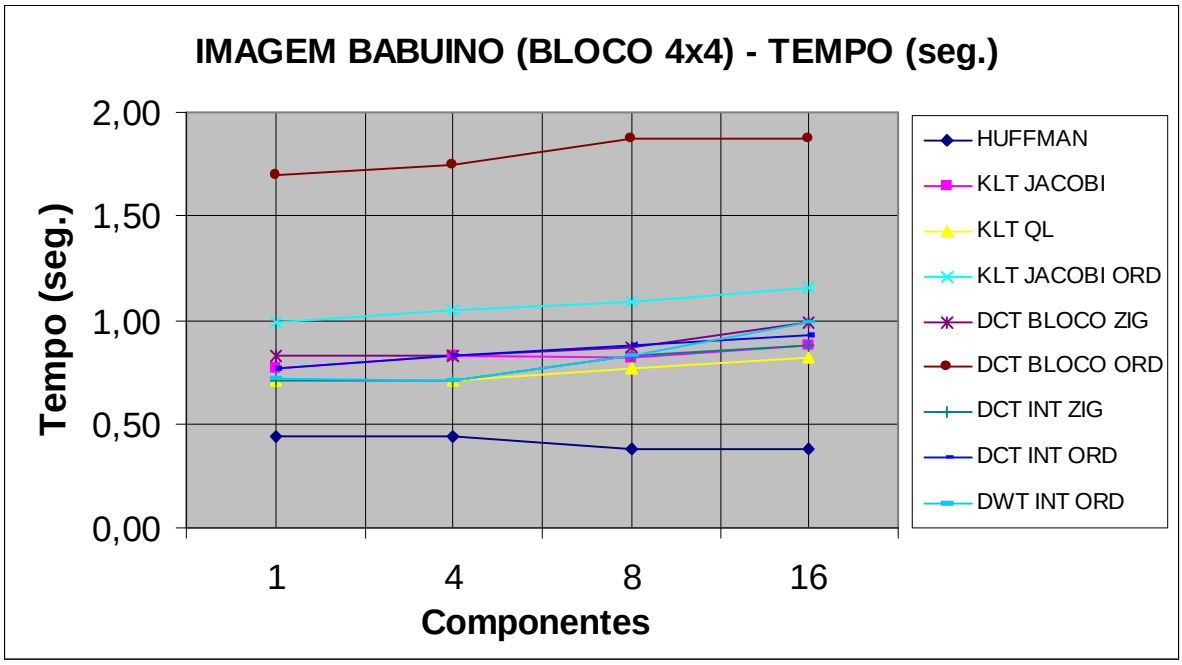


Figura 40 – Tempo (seg) de Gravação da Imagem BABUÍNO (BLOCO 4x4)

A Tabela 23 em conjunto com a Figura 41 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 23 – PSNR da Imagem BABUÍNO (BLOCO 8x8)

Imagem:	**babuinog**	Bloco:	**8**	x	**8**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
KLT JACOBI + HUFFMAN		19,627680	24,599292	28,774105	58,950605
KLT QL + HUFFMAN		19,627683	24,599288	28,774119	58,950205
KLT JACOBI + ORDEM + HUFFMAN		19,416901	24,223202	28,427349	58,950605
DCT BLOCO+ZIGZAG+HUFFMAN		19,619280	23,951447	27,349698	62,327509
DCT BLOCO+ORDEM+HUFFMAN		19,619280	28,528145	37,211790	62,327509
DCT INTEIRA+ZIGZAG+HUFFMAN		20,036330	24,617184	28,334416	58,944407
DCT INTEIRA+ORDEM+HUFFMAN		20,889046	28,458544	34,987259	58,944407
DWT INTEIRA+ORDEM+HUFFMAN		21,365554	30,732287	38,656004	58,956011
DWT BLOCO+ORDEM+HUFFMAN		7,504028	27,057146	33,525920	58,980924

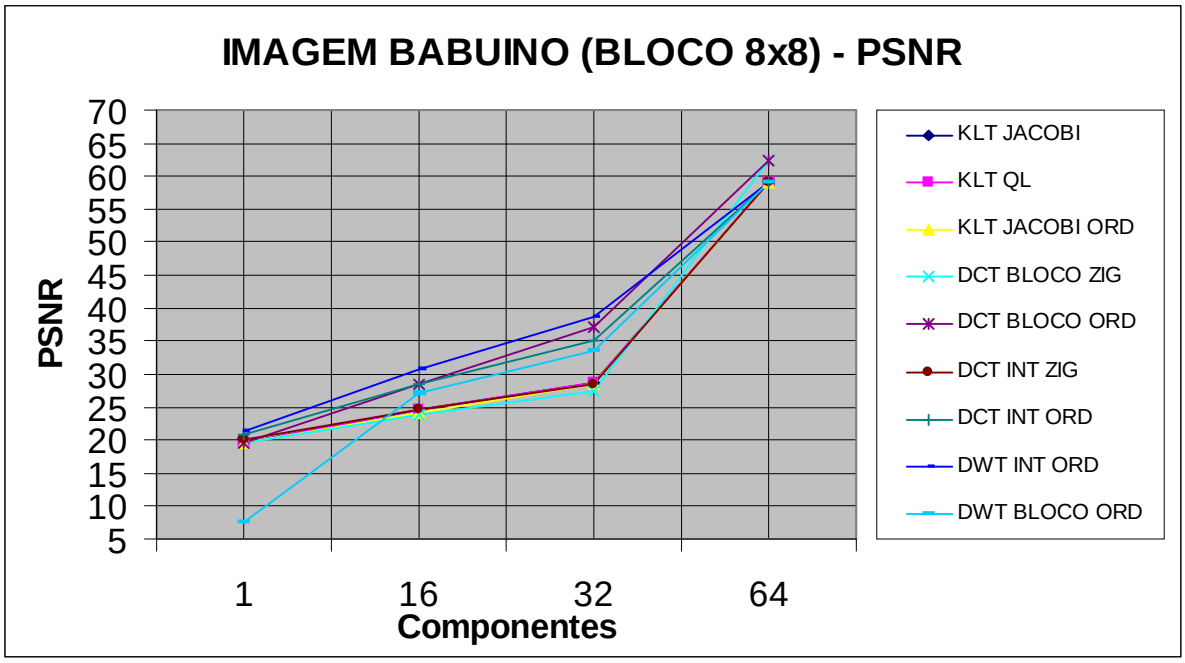


Figura 41 – PSNR da Imagem BABUÍNO (BLOCO 8x8)

A Tabela 24 em conjunto com a Figura 42 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 24 – CR da Imagem BABUÍNO (BLOCO 8x8)

Imagem:	**babuinog**	Bloco:	**8**	x	**8**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,046292	1,046292	1,046292	1,046292
KLT JACOBI + HUFFMAN		3,518964	1,927223	1,362080	0,997987
KLT QL + HUFFMAN		3,518964	1,927759	1,362383	0,998112
KLT JACOBI + ORDEM + HUFFMAN		3,547275	1,889537	1,322990	0,997987
DCT BLOCO+ZIGZAG+HUFFMAN		4,044716	1,851772	1,470259	1,243913
DCT BLOCO+ORDEM+HUFFMAN		4,044716	1,755785	1,347190	1,243913
DCT INTEIRA+ZIGZAG+HUFFMAN		3,923943	2,097637	1,465609	1,055628
DCT INTEIRA+ORDEM+HUFFMAN		4,303122	2,202584	1,515653	1,055628
DWT INTEIRA+ORDEM+HUFFMAN		4,268996	2,169919	1,500388	1,089757
DWT BLOCO+ORDEM+HUFFMAN		4,421001	1,832729	1,308644	0,997340

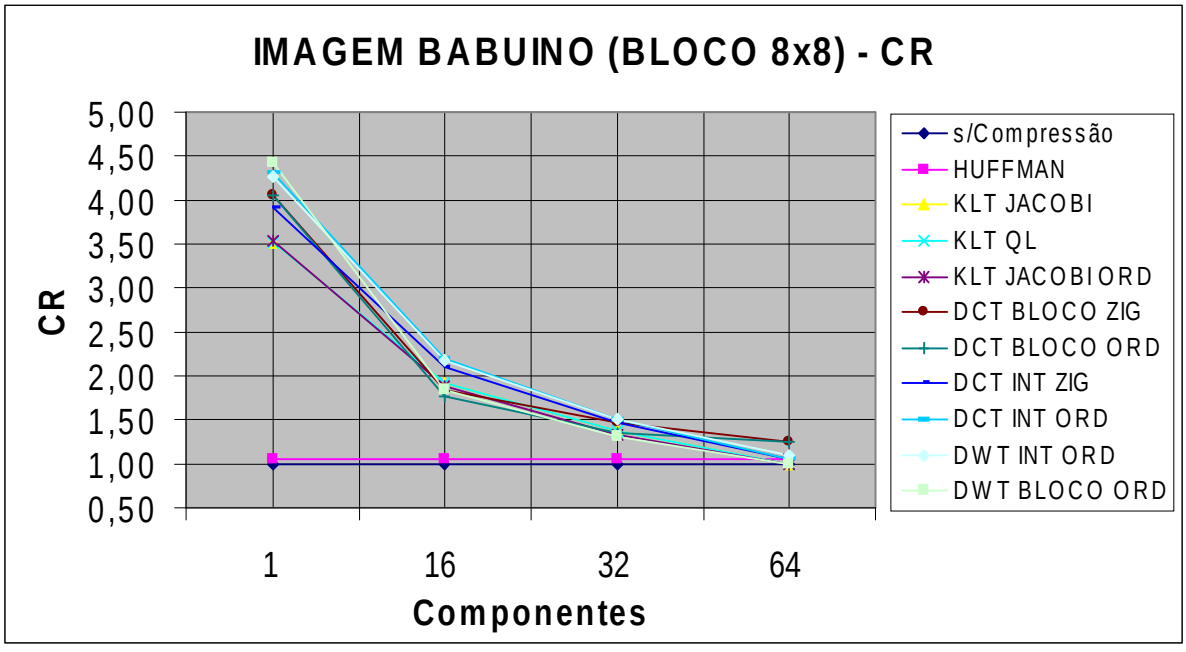


Figura 42 – CR da Imagem BABUÍNO (BLOCO 8x8)

A Tabela 25 em conjunto com a Figura 43 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 25 – Tempo de Gravação (seg) da Imagem BABUÍNO (BLOCO 8x8)

Imagem:	**babuinog**	Bloco:	**8**	x	**8**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,38	0,38	0,38	0,44
KLT JACOBI + HUFFMAN		1,75	1,86	1,87	1,93
KLT QL + HUFFMAN		1,70	1,76	1,75	1,98
KLT JACOBI + ORDEM + HUFFMAN		1,98	1,98	2,08	2,14
DCT BLOCO+ZIGZAG+HUFFMAN		0,71	0,72	0,77	0,77
DCT BLOCO+ORDEM+HUFFMAN		1,16	1,21	1,26	1,32
DCT INTEIRA+ZIGZAG+HUFFMAN		0,72	0,77	0,77	0,83
DCT INTEIRA+ORDEM+HUFFMAN		0,77	0,88	0,88	0,93
DWT INTEIRA+ORDEM+HUFFMAN		0,66	0,71	0,82	0,88
DWT BLOCO+ORDEM+HUFFMAN		0,83	0,94	1,04	1,15

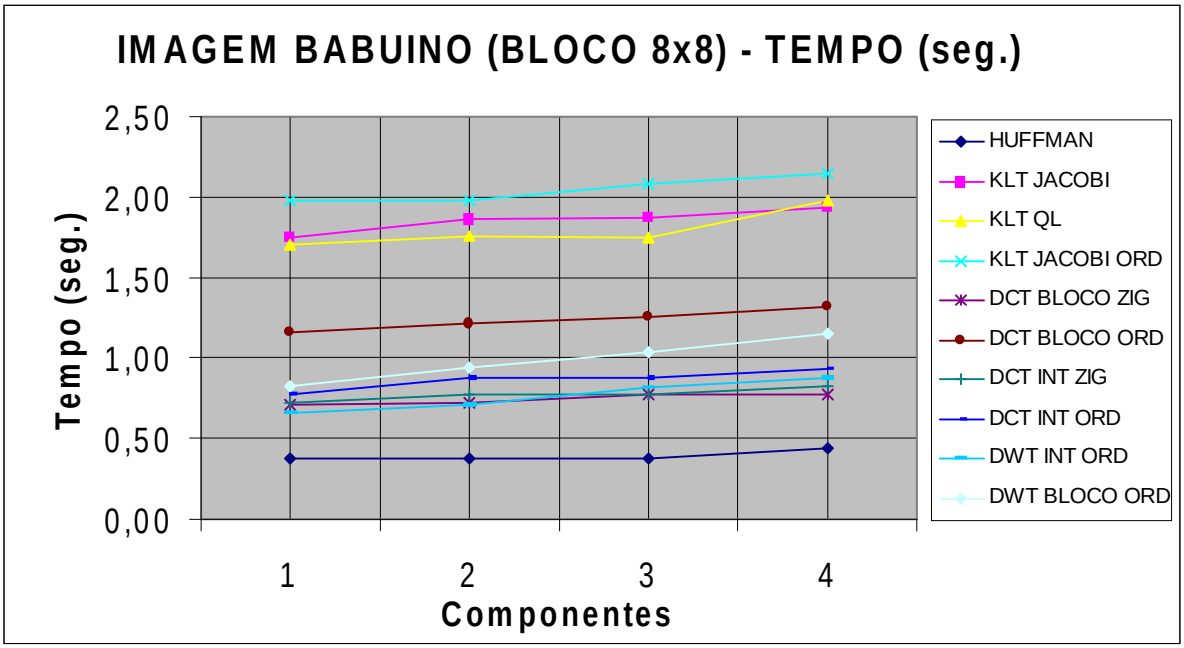


Figura 43 – Tempo de Gravação (seg) da Imagem BABUÍNO (BLOCO 8x8)

A Tabela 26 em conjunto com a Figura 44 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 26 – PSNR da Imagem BABUÍNO (BLOCO 16x16)

Imagem:	**babuinog**	Bloco:	**16**	x	**16**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
KLT JACOBI + HUFFMAN		18,867683	21,524128	30,541696	58,905817
KLT QL + HUFFMAN		18,867681	21,524134	30,541566	58,932236
KLT JACOBI + ORDEM + HUFFMAN		18,754718	21,245606	30,208258	58,905817
DCT BLOCO+ZIGZAG+HUFFMAN		18,850425	21,102983	27,821615	59,728633
DCT BLOCO+ORDEM+HUFFMAN		18,850425	23,192466	35,763642	59,728633
DCT INTEIRA+ZIGZAG+HUFFMAN		19,218503	21,335439	28,334416	58,944407
DCT INTEIRA+ORDEM+HUFFMAN		19,707328	23,125507	34,987259	58,944407
DWT INTEIRA+ORDEM+HUFFMAN		19,985753	24,152180	38,656004	58,956011
DWT BLOCO+ORDEM+HUFFMAN		7,415268	22,613284	34,434426	58,969856

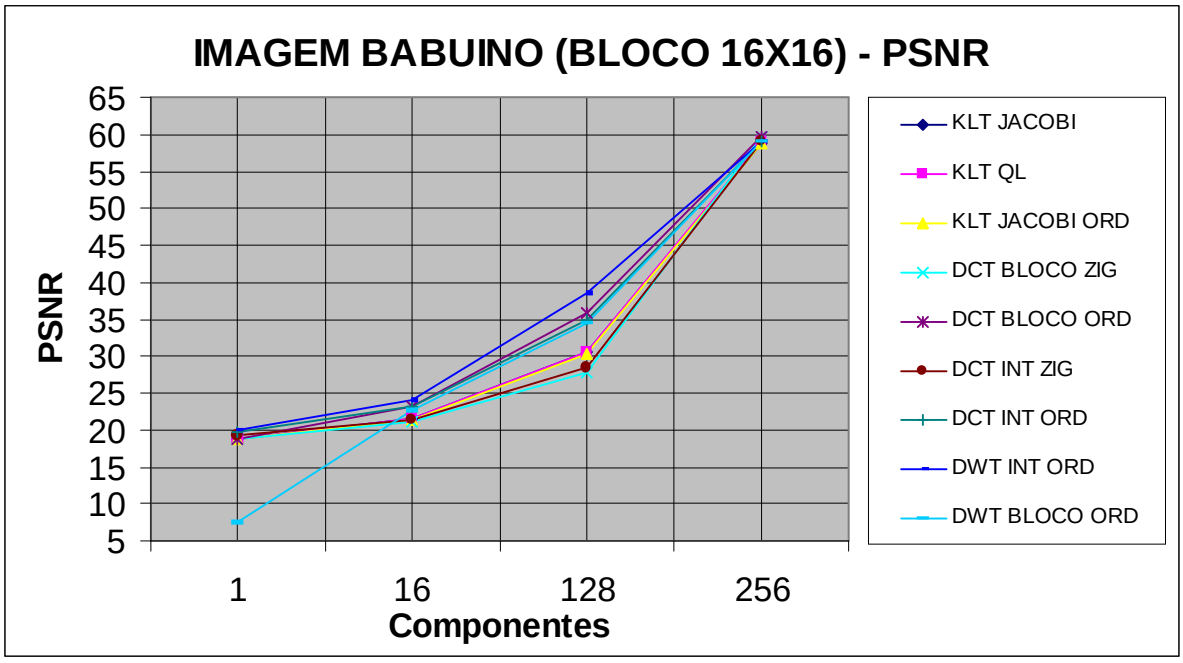


Figura 44 – PSNR da Imagem BABUÍNO (BLOCO 16x16)

A Tabela 27 em conjunto com a Figura 45 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 27 – CR da Imagem BABUÍNO (BLOCO 16x16)

Imagem:	**babuinog**	Bloco:	**16**	x	**16**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,046292	1,046292	1,046292	1,046292
KLT JACOBI + HUFFMAN		4,005631	2,424691	0,822767	0,512090
KLT QL + HUFFMAN		4,005631	2,427799	0,822846	0,512123
KLT JACOBI + ORDEM + HUFFMAN		4,054872	2,423173	0,815732	0,512090
DCT BLOCO+ZIGZAG+HUFFMAN		4,180118	2,421857	1,301443	0,990297
DCT BLOCO+ORDEM+HUFFMAN		4,180118	2,401945	1,242545	0,990297
DCT INTEIRA+ZIGZAG+HUFFMAN		4,550471	3,280679	1,465609	1,055628
DCT INTEIRA+ORDEM+HUFFMAN		5,018819	3,490591	1,515653	1,055628
DWT INTEIRA+ORDEM+HUFFMAN		4,916453	3,443962	1,500388	1,089757
DWT BLOCO+ORDEM+HUFFMAN		4,593191	2,365509	1,230722	0,956677

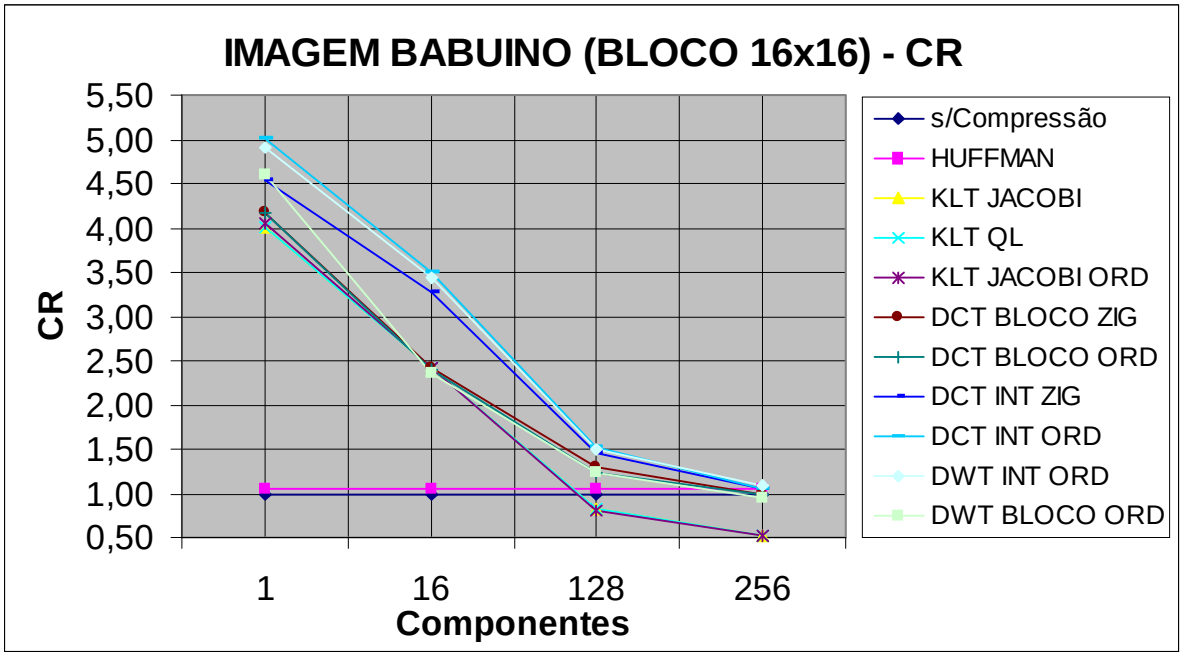


Figura 45 – CR da Imagem BABUÍNO (BLOCO 16x16)

A Tabela 28 em conjunto com a Figura 46 apresentam os resultados obtidos para a imagem BABUÍNO (512x512 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 28 – Tempo de Gravação (seg) da Imagem BABUÍNO (BLOCO 16x16)

Imagem:	**babuinog**	Bloco:	**16**	x	**16**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,39	0,38	0,44	0,39
KLT JACOBI + HUFFMAN		25,92	29,00	27,35	27,79
KLT QL + HUFFMAN		11,81	13,29	12,03	13,18
KLT JACOBI + ORDEM + HUFFMAN		26,31	28,29	26,37	29,76
DCT BLOCO+ZIGZAG+HUFFMAN		0,66	0,72	0,82	0,83
DCT BLOCO+ORDEM+HUFFMAN		0,94	0,99	1,09	1,26
DCT INTEIRA+ZIGZAG+HUFFMAN		0,65	0,72	0,82	0,94
DCT INTEIRA+ORDEM+HUFFMAN		0,72	0,83	0,87	0,99
DWT INTEIRA+ORDEM+HUFFMAN		0,61	0,65	0,77	0,83
DWT BLOCO+ORDEM+HUFFMAN		0,72	0,83	0,93	0,93

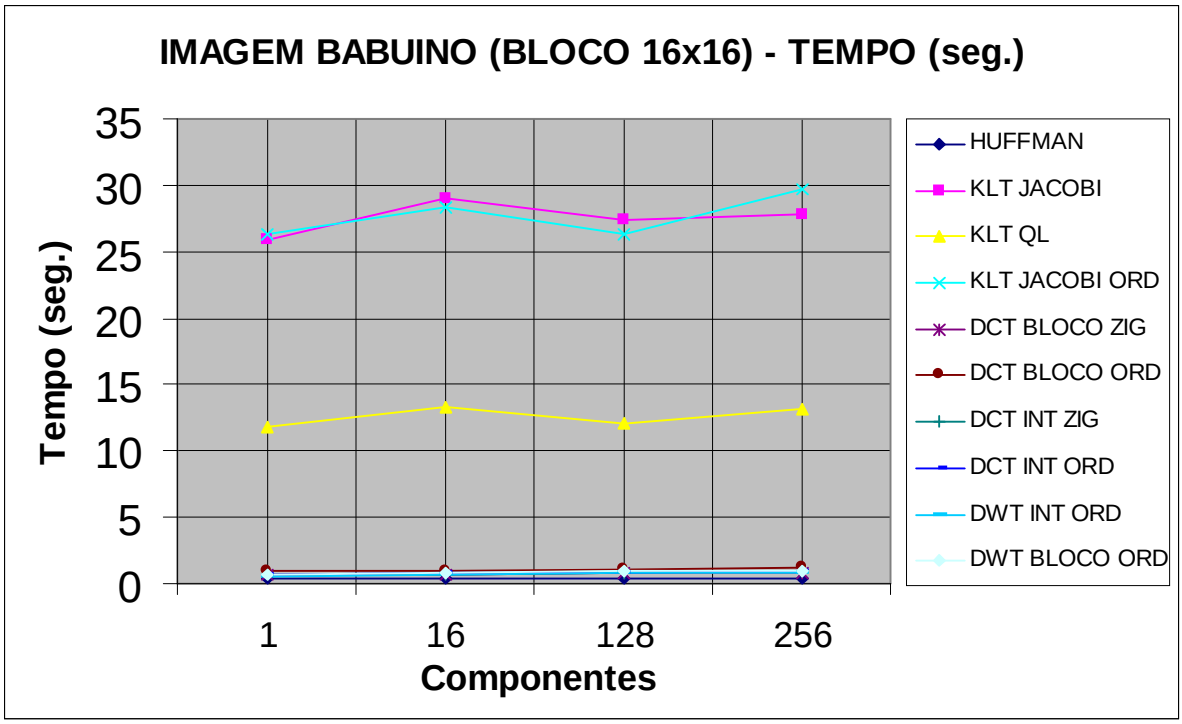


Figura 46 – Tempo de Gravação (seg) da Imagem BABUÍNO (BLOCO 16x16)

A Tabela 29 em conjunto com a Figura 47 apresentam os resultados da *PSNR (Razão Sinal/Ruído de Pico)* obtida para a imagem BABUÍNO (512x512 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e compressão QL.

Tabela 29 – PSNR da Imagem BABUÍNO – Compressão QL

Imagem:	**babuinog**			
RAZÃO SINAL/RUÍDO DE PICO (PSNR)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	23,192588	24,066816	24,599288	25,549885
4:2	27,098019	28,019715	28,774119	30,541566
4:3	31,387090	32,979501	33,910795	36,762812
4:4	58,955410	58,956011	58,950205	58,932236

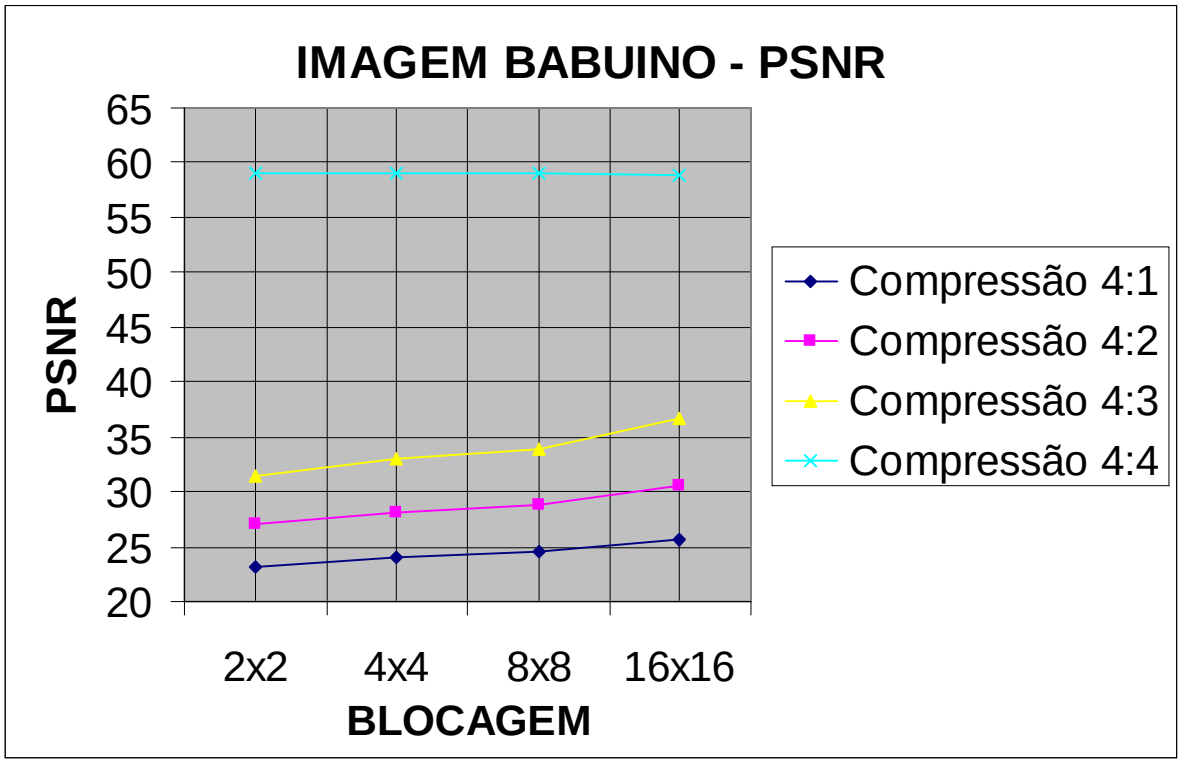


Figura 47 – PSNR da Imagem BABUÍNO – Compressão QL

A Tabela 30 em conjunto com a Figura 48 apresentam os resultados da *CR (Razão de Compressão)* obtida para a imagem BABUÍNO (512x512 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e compressão QL.

Tabela 30 – CR da Imagem BABUÍNO – Compressão QL

Imagem:	babuinog			
RAZÃO DE COMPRESSÃO (CR)				
Compressão	Blocagem da Imagem			
	2x2	4x4	8x8	16x16
4:1	2,278722	2,115626	1,927759	1,304461
4:2	1,529294	1,481038	1,362383	0,822846
4:3	1,207053	1,195209	1,105329	0,615992
4:4	1,078443	1,079960	0,998112	0,512123

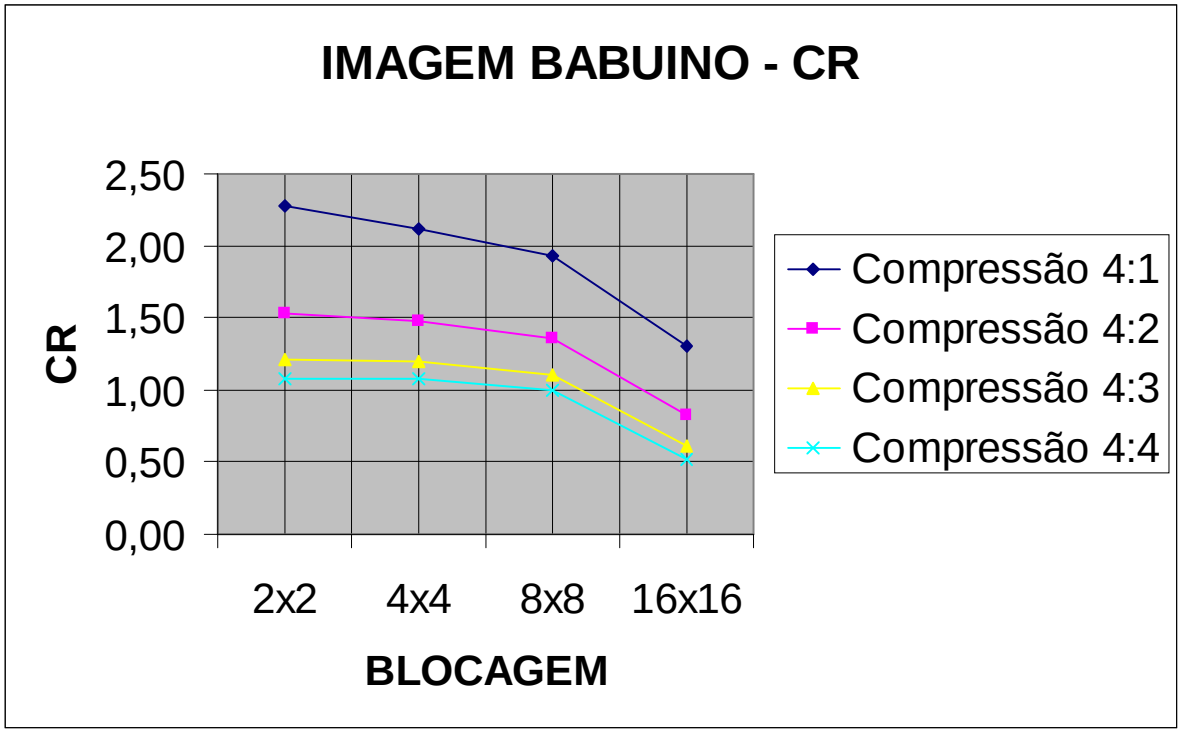


Figura 48 – CR da Imagem BABUÍNO – Compressão QL

A Tabela 31 em conjunto com a Figura 49 apresentam os resultados do *Tempo (em segundos)* obtida para a imagem BABUÍNO (512x512 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e compressão QL.

Tabela 31 – Tempo de Gravação da Imagem BABUÍNO – Compressão QL

Imagem:	**babuinog**			
TEMPO DE GRAVAÇÃO (SEGUNDOS)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	0,55	0,71	1,760000	14,770000
4:2	0,55	0,77	1,750000	12,030000
4:3	0,61	0,82	1,810000	12,690000
4:4	0,60	0,82	1,980000	13,180000

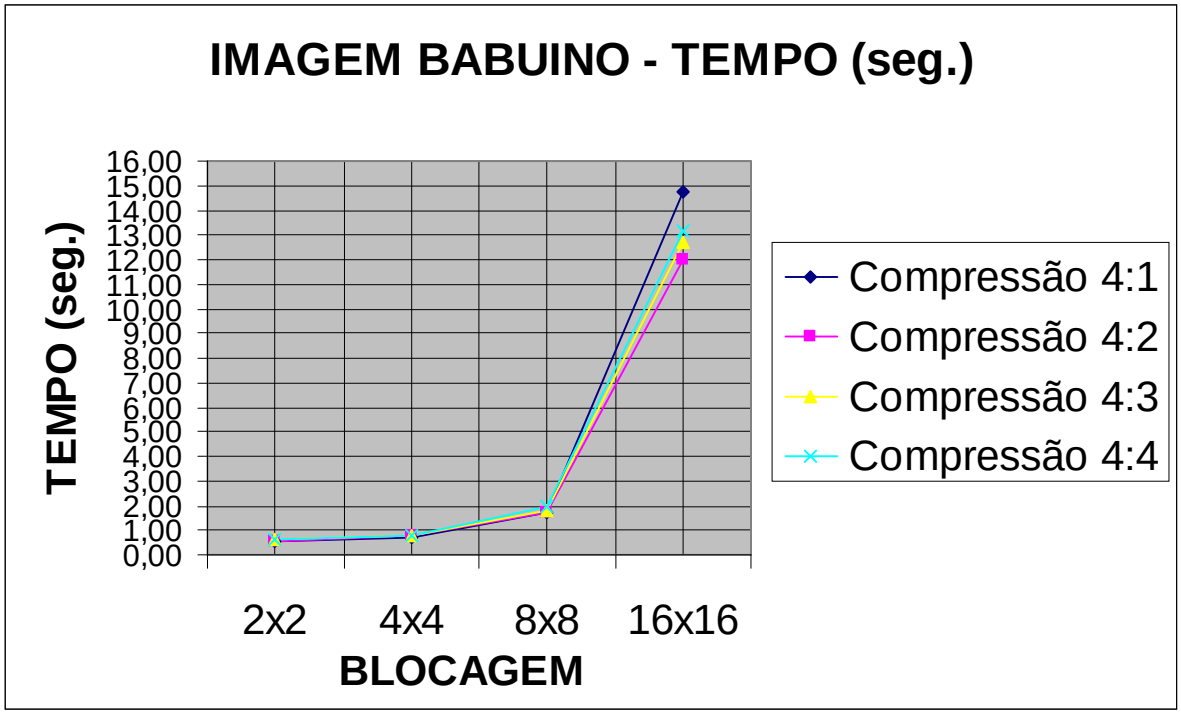


Figura 49 – Tempo de Gravação da Imagem BABUÍNO – Compressão QL

5.3 RESULTADOS OBTIDOS COM A IMAGEM CASA

As tabelas e gráficos apresentados abaixo foram obtidos utilizando-se a imagem padrão CASA (HOUSE) (Figura 50), de dimensões 256 x 256 (65.536 PIXELS) com 256 níveis de escala de cinza.

Figura 50 – Figura CASA (256 x 256)

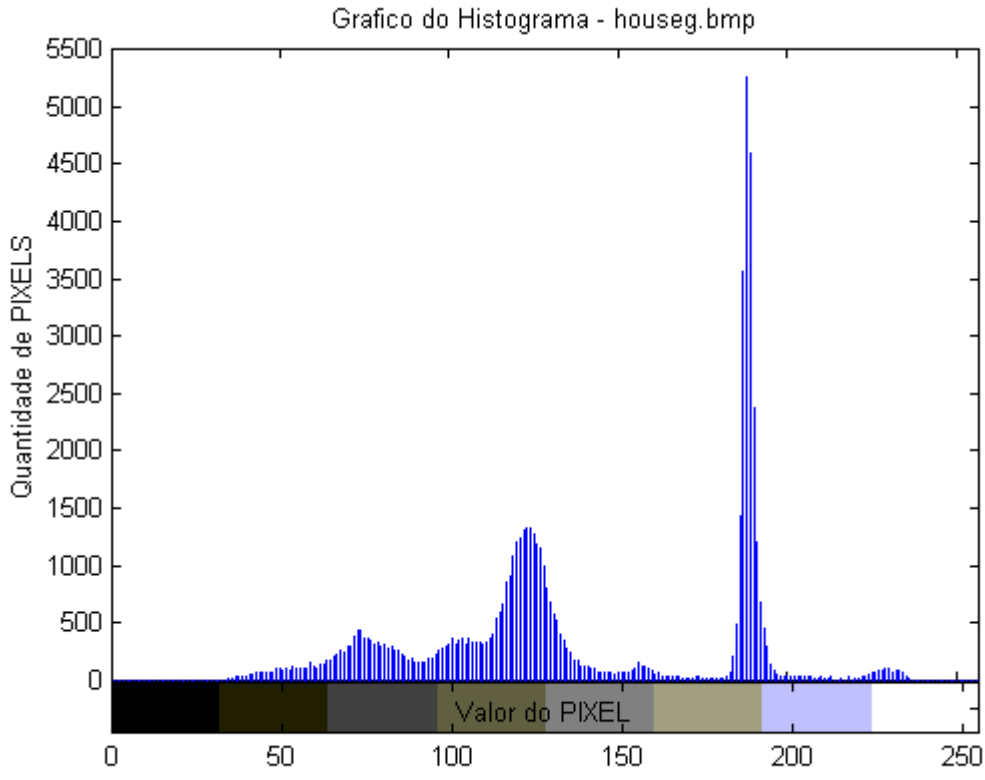


Figura 51 – Histograma da Imagem CASA (256x256)

Como se pode observar no Histograma da Figura 51, existe uma concentração de PIXELS entre dois valores principais da escala de cinza desta imagem. O valor máximo da escala de cinza aparece em 8 % dos pontos da imagem (5275 PIXELS), havendo boa concentração em torno desse valor de PIXEL. O valor médio do PIXEL é igual a 137.

A Tabela 32 em conjunto com a Figura 52 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 32 – PSNR da Imagem CASA (BLOCO 2x2)

Imagem:	**houseg**	Bloco:	**2**	x	**2**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
KLT JACOBI + HUFFMAN		28,930795	34,619904	44,996868	57,912780
KLT QL + HUFFMAN		28,930795	34,619904	44,996675	57,913411
KLT JACOBI + ORDEM + HUFFMAN		28,625322	34,230739	44,427084	57,912780
DCT BLOCO+ZIGZAG+HUFFMAN		28,928828	30,148321	45,108460	66,163016
DCT BLOCO+ORDEM+HUFFMAN		28,928828	39,631612	46,865359	66,163016
DCT INTEIRA+ZIGZAG+HUFFMAN		35,243135	42,179271	48,851421	58,901462
DCT INTEIRA+ORDEM+HUFFMAN		40,925501	48,310201	54,010316	58,901462
DWT INTEIRA+ORDEM+HUFFMAN		44,542863	53,118312	59,228425	59,228425

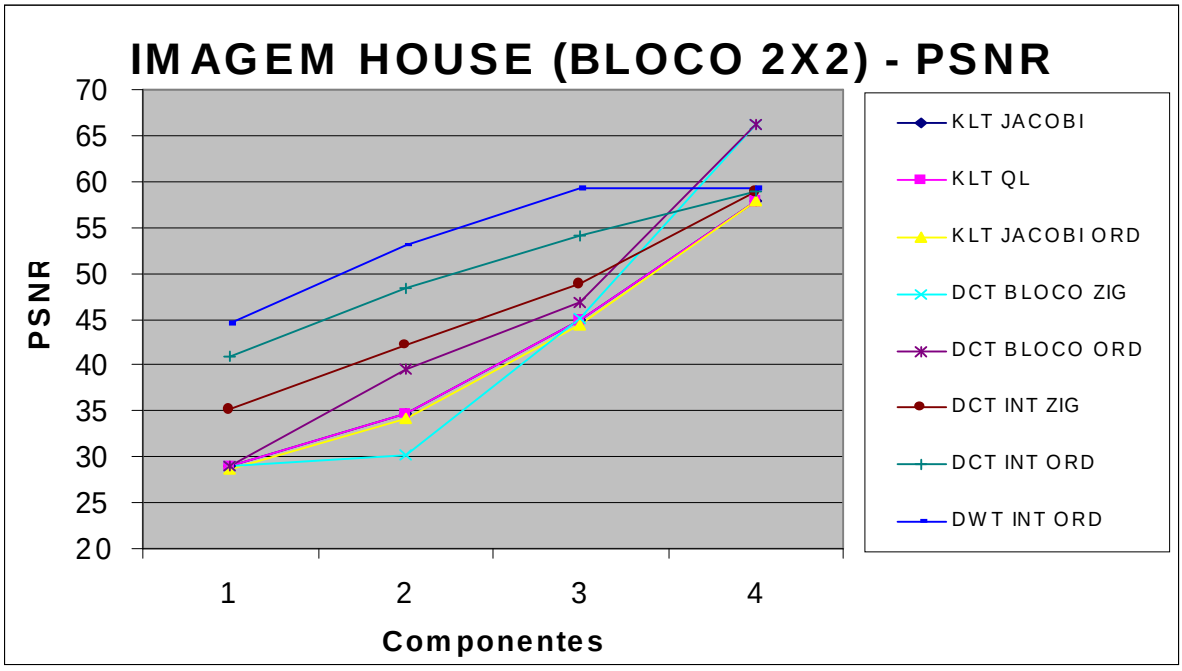


Figura 52 – PSNR da Imagem CASA (Bloco 2x2)

A Tabela 33 em conjunto com a Figura 53 apresentam os resultados obtidos para a imagem CASA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 33 – CR da Imagem CASA (BLOCO 2x2)

Imagem:	**houseg**	Bloco:	**2**	x	**2**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,060378	1,060378	1,060378	1,060378
KLT JACOBI + HUFFMAN		1,678400	1,309135	1,130565	1,075217
KLT QL + HUFFMAN		1,678400	1,311506	1,132102	1,074454
KLT JACOBI + ORDEM + HUFFMAN		1,680686	1,266522	1,115513	1,075217
DCT BLOCO+ZIGZAG+HUFFMAN		1,195878	0,898005	0,771326	0,742383
DCT BLOCO+ORDEM+HUFFMAN		1,195878	0,845420	0,766489	0,742383
DCT INTEIRA+ZIGZAG+HUFFMAN		1,432590	1,182441	1,055622	1,015752
DCT INTEIRA+ORDEM+HUFFMAN		1,454391	1,183407	1,035199	1,015752
DWT INTEIRA+ORDEM+HUFFMAN		1,538927	1,248669	1,139110	1,139110

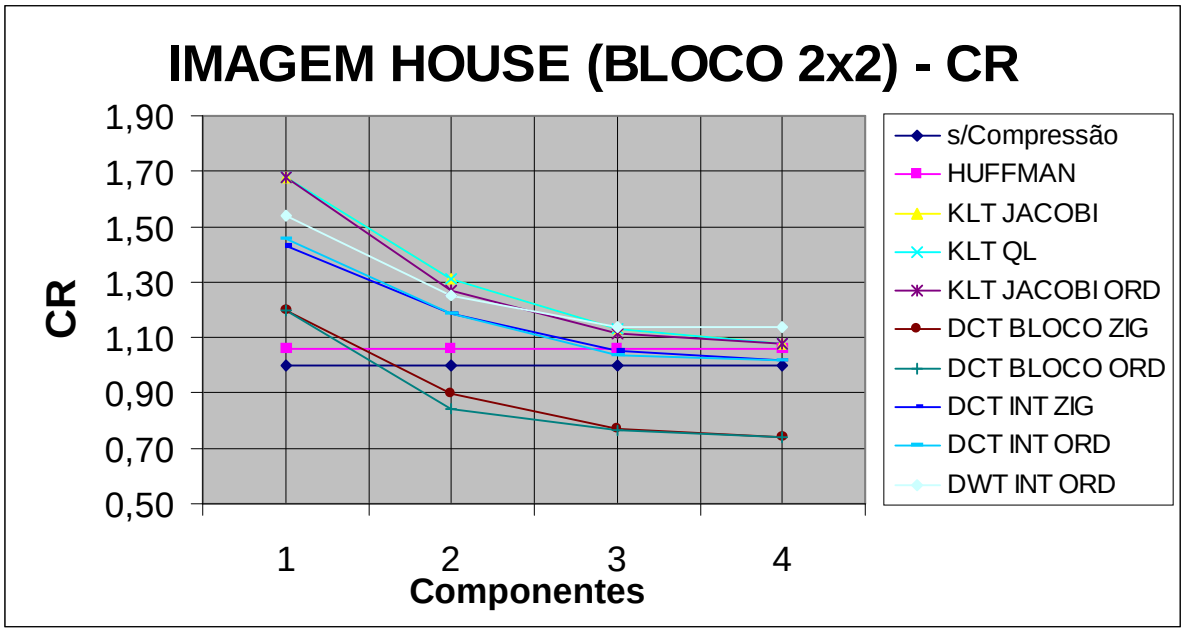


Figura 53 – CR da Imagem CASA (Bloco 2x2)

A Tabela 34 em conjunto com a Figura 54 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 34 – Tempo (seg) de Gravação da Imagem CASA (BLOCO 2x2)

Imagem:	**houseg**	Bloco:	**2**	x	**2**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,16	0,17	0,17	0,27
KLT JACOBI + HUFFMAN		0,11	0,11	0,16	0,22
KLT QL + HUFFMAN		0,11	0,17	0,11	0,22
KLT JACOBI + ORDEM + HUFFMAN		0,28	0,33	0,28	0,33
DCT BLOCO+ZIGZAG+HUFFMAN		0,22	0,33	0,33	0,27
DCT BLOCO+ORDEM+HUFFMAN		0,66	0,72	0,77	0,77
DCT INTEIRA+ZIGZAG+HUFFMAN		0,16	0,22	0,28	0,27
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,22	0,27	0,28
DWT INTEIRA+ORDEM+HUFFMAN		0,17	0,22	0,22	0,16

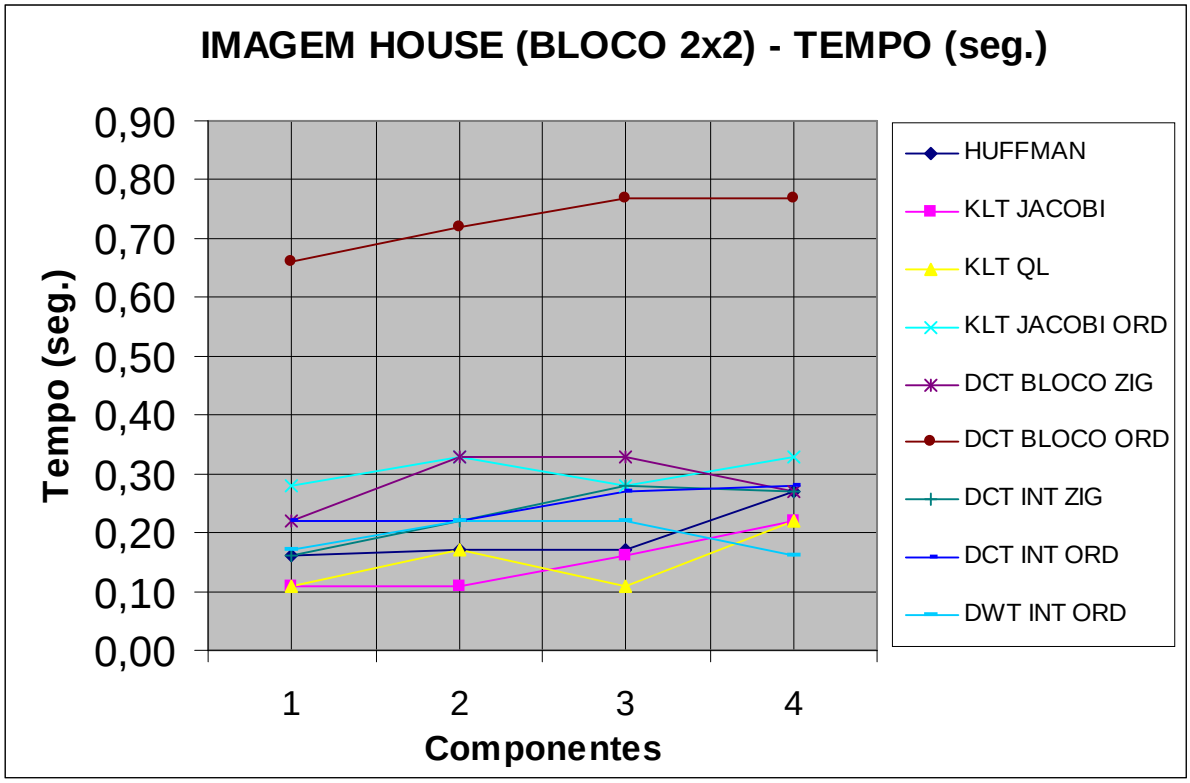


Figura 54 – Tempo (seg) de Gravação da Imagem CASA (Bloco 2x2)

A Tabela 35 em conjunto com a Figura 55 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 35 – PSNR da Imagem CASA (BLOCO 4x4)

Imagem:	**houseg**	Bloco:	**4**	x	**4**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
KLT JACOBI + HUFFMAN		24,905039	33,132473	40,575779	59,277334
KLT QL + HUFFMAN		24,905039	33,132473	40,575721	59,274746
KLT JACOBI + ORDEM + HUFFMAN		24,390336	32,697438	40,225537	59,277334
DCT BLOCO+ZIGZAG+HUFFMAN		24,876577	32,776236	36,395577	61,830231
DCT BLOCO+ORDEM+HUFFMAN		24,876577	36,961489	47,136805	61,830231
DCT INTEIRA+ZIGZAG+HUFFMAN		28,524942	35,243135	42,179271	58,901462
DCT INTEIRA+ORDEM+HUFFMAN		32,150516	40,925501	48,310201	58,901462
DWT INTEIRA+ORDEM+HUFFMAN		34,896450	44,542863	53,118312	59,228425

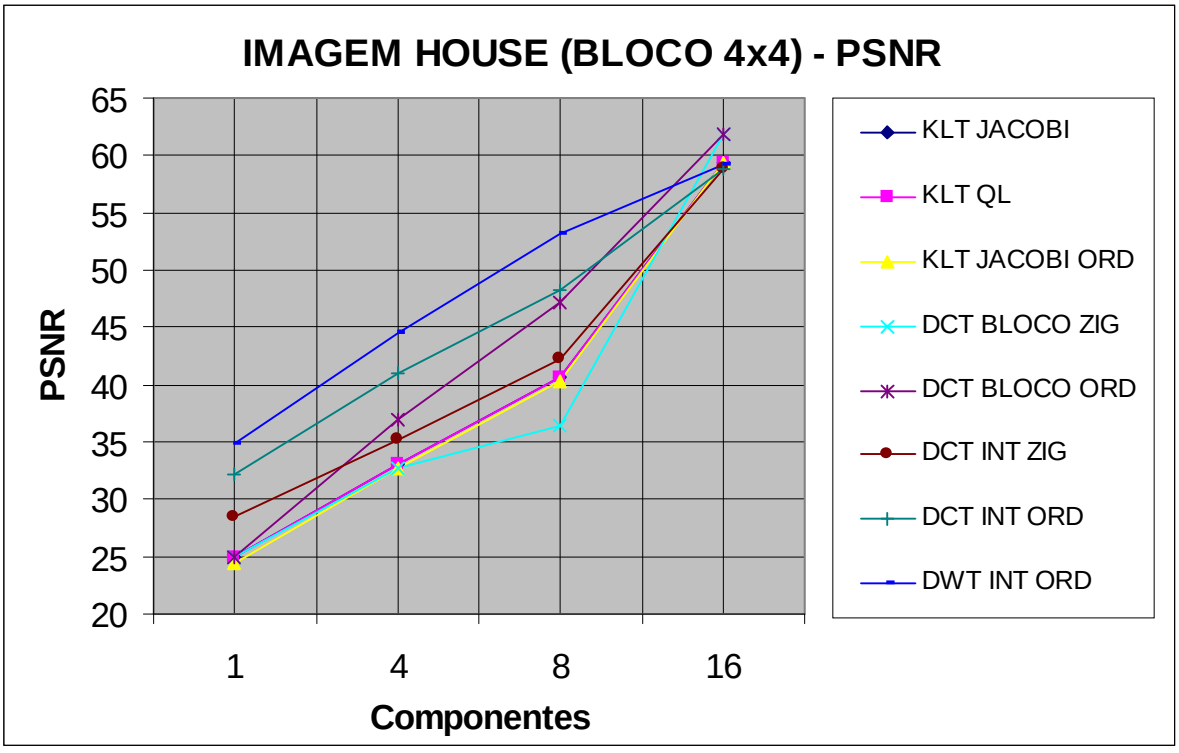


Figura 55 – PSNR da Imagem CASA (Bloco 4x4)

A Tabela 36 em conjunto com a Figura 56 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 36 – CR da Imagem CASA (BLOCO 4x4)

Imagem:	**houseg**	Bloco:	**4**	x	**4**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,060378	1,060378	1,060378	1,060378
KLT JACOBI + HUFFMAN		1,811640	1,379515	1,149865	1,016185
KLT QL + HUFFMAN		1,811640	1,391299	1,156935	1,022424
KLT JACOBI + ORDEM + HUFFMAN		1,871548	1,353613	1,118886	1,016185
DCT BLOCO+ZIGZAG+HUFFMAN		1,327924	0,875452	0,796761	0,724207
DCT BLOCO+ORDEM+HUFFMAN		1,327924	0,858505	0,769286	0,724207
DCT INTEIRA+ZIGZAG+HUFFMAN		1,814947	1,432590	1,182441	1,015752
DCT INTEIRA+ORDEM+HUFFMAN		1,908711	1,454391	1,183407	1,015752
DWT INTEIRA+ORDEM+HUFFMAN		1,990319	1,538927	1,248669	1,139110

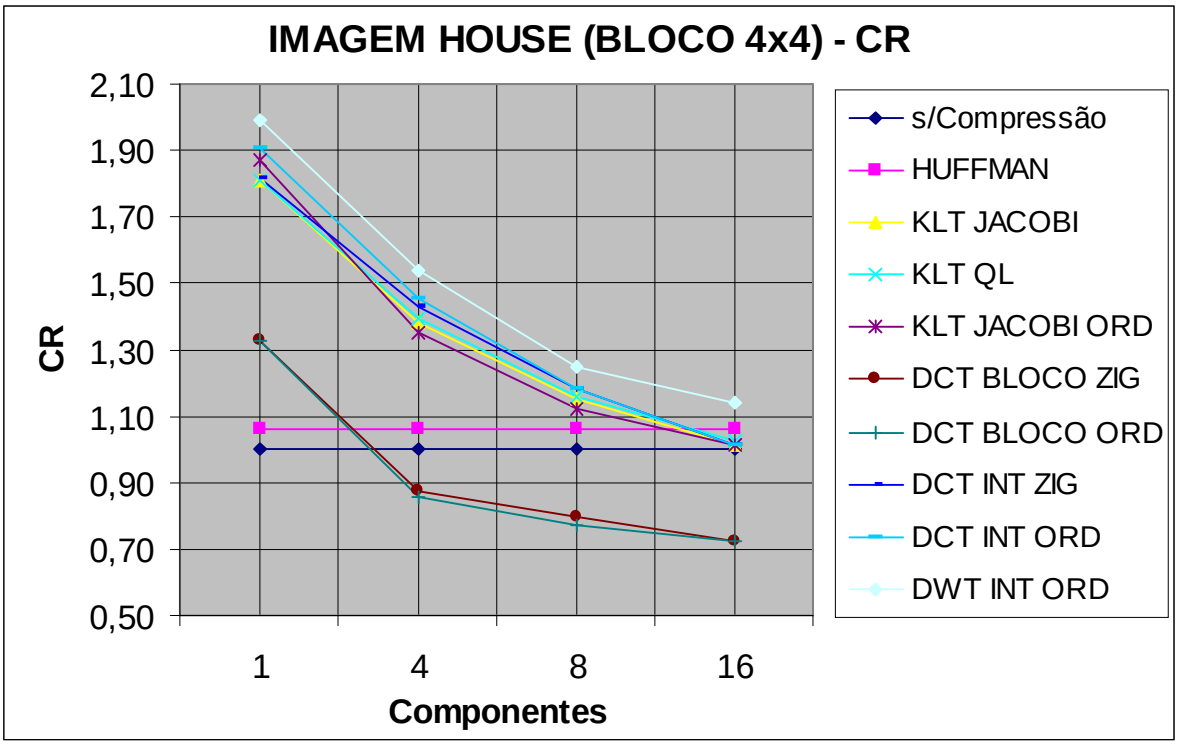


Figura 56 – CR da Imagem CASA (BLOCO 4x4)

A Tabela 37 em conjunto com a Figura 57 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 37 – Tempo (seg) de Gravação da Imagem CASA (BLOCO 4x4)

Imagem:	**houseg**	Bloco:	**4**	x	**4**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,11	0,11	0,11	0,16
KLT JACOBI + HUFFMAN		0,16	0,16	0,22	0,28
KLT QL + HUFFMAN		0,17	0,17	0,22	0,28
KLT JACOBI + ORDEM + HUFFMAN		0,27	0,27	0,28	0,33
DCT BLOCO+ZIGZAG+HUFFMAN		0,22	0,28	0,27	0,27
DCT BLOCO+ORDEM+HUFFMAN		0,44	0,55	0,50	0,50
DCT INTEIRA+ZIGZAG+HUFFMAN		0,22	0,17	0,16	0,22
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,17	0,22	0,27
DWT INTEIRA+ORDEM+HUFFMAN		0,17	0,16	0,22	0,22

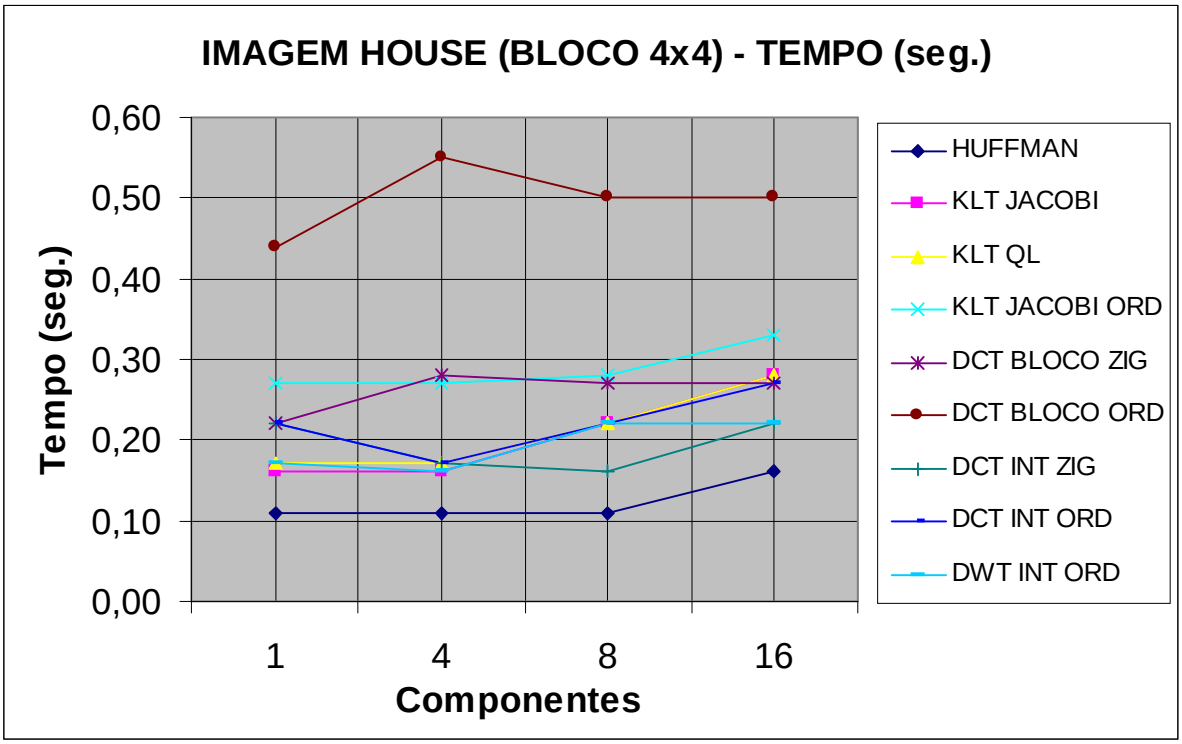


Figura 57 – Tempo (seg) de Gravação da Imagem CASA (BLOCO 4x4)

A Tabela 38 em conjunto com a Figura 58 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 38 – PSNR da Imagem CASA (BLOCO 8x8)

Imagem:	**houseg**	Bloco:	**8**	x	**8**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
KLT JACOBI + HUFFMAN		22,453152	35,432934	44,228234	59,584801
KLT QL + HUFFMAN		22,453152	35,432638	44,228315	59,589434
KLT JACOBI + ORDEM + HUFFMAN		22,171729	35,084833	44,027233	59,584801
DCT BLOCO+ZIGZAG+HUFFMAN		22,370396	33,386391	40,106403	62,879280
DCT BLOCO+ORDEM+HUFFMAN		22,370396	41,997257	62,278152	62,879280
DCT INTEIRA+ZIGZAG+HUFFMAN		23,958729	35,243135	42,179271	58,901462
DCT INTEIRA+ORDEM+HUFFMAN		26,150872	40,925501	48,310201	58,901462
DWT INTEIRA+ORDEM+HUFFMAN		28,529986	44,542863	53,118312	59,228425
DWT BLOCO+ORDEM+HUFFMAN		6,888116	36,602870	45,288762	59,310248

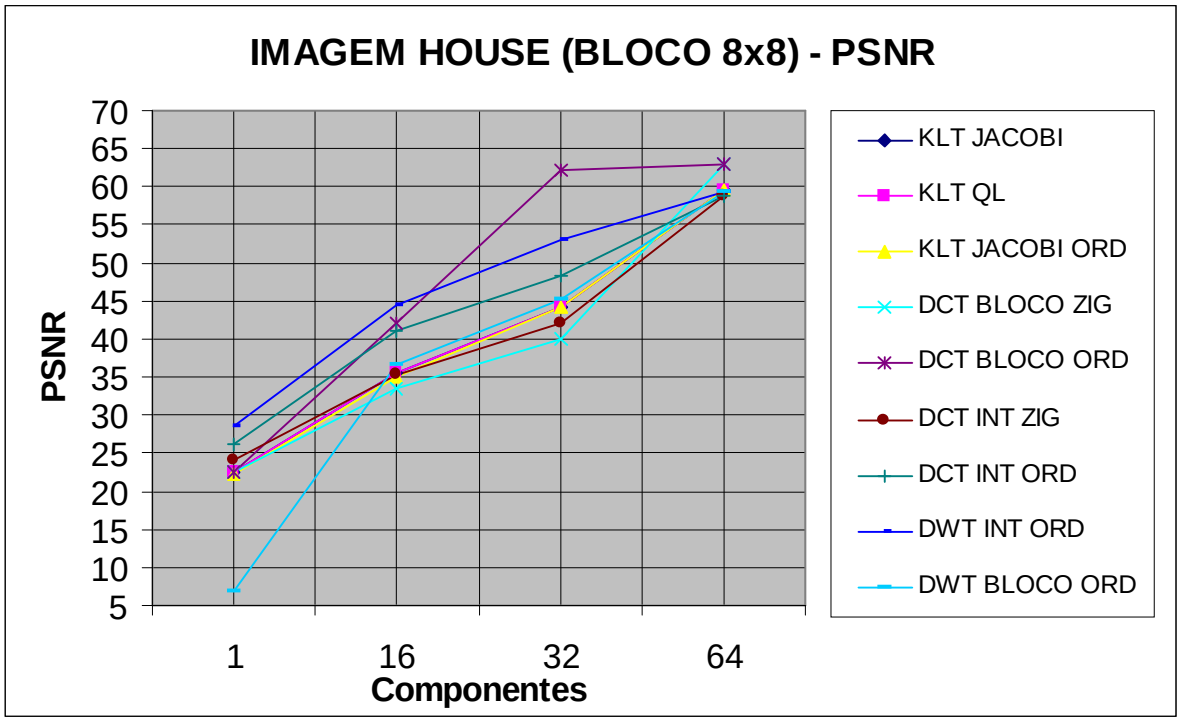


Figura 58 – PSNR da Imagem CASA (BLOCO 8x8)

A Tabela 39 em conjunto com a Figura 59 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 39 – CR da Imagem CASA (BLOCO 8x8)

Imagem:	**houseg**	Bloco:	**8**	x	**8**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,060378	1,060378	1,060378	1,060378
KLT JACOBI + HUFFMAN		2,485968	1,280275	1,038264	0,859791
KLT QL + HUFFMAN		2,485968	1,289819	1,045007	0,864096
KLT JACOBI + ORDEM + HUFFMAN		2,526991	1,251132	1,015132	0,859791
DCT BLOCO+ZIGZAG+HUFFMAN		2,784866	1,340592	1,233182	1,190386
DCT BLOCO+ORDEM+HUFFMAN		2,784866	1,258364	1,191024	1,190386
DCT INTEIRA+ZIGZAG+HUFFMAN		2,178494	1,432590	1,182441	1,015752
DCT INTEIRA+ORDEM+HUFFMAN		2,470204	1,454391	1,183407	1,015752
DWT INTEIRA+ORDEM+HUFFMAN		2,487639	1,538927	1,248669	1,139110
DWT BLOCO+ORDEM+HUFFMAN		2,831265	1,114953	0,964959	0,879335

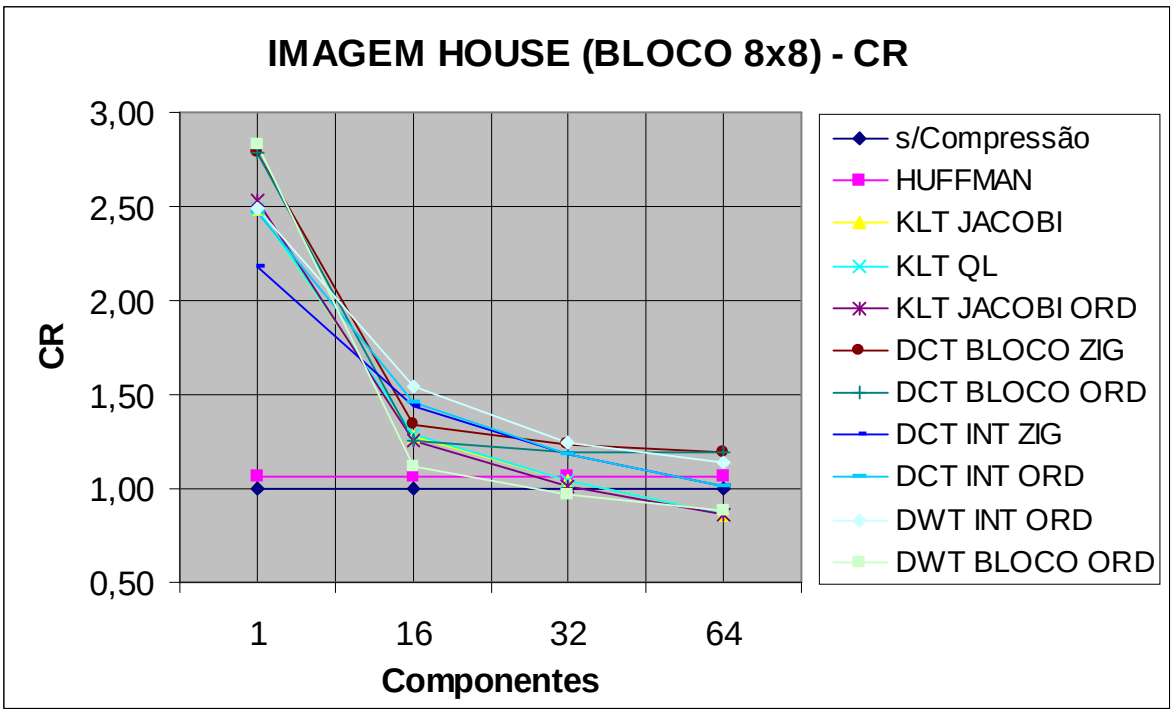


Figura 59 – CR da Imagem CASA (BLOCO 8x8)

A Tabela 40 em conjunto com a Figura 60 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 40 – Tempo de Gravação (seg) da Imagem CASA (BLOCO 8x8)

Imagem:	**houseg**	Bloco:	**8**	x	**8**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,17	0,16	0,11	0,17
KLT JACOBI + HUFFMAN		0,54	0,55	0,66	0,66
KLT QL + HUFFMAN		0,44	0,43	0,50	0,55
KLT JACOBI + ORDEM + HUFFMAN		0,55	0,61	0,66	0,66
DCT BLOCO+ZIGZAG+HUFFMAN		0,17	0,17	0,16	0,16
DCT BLOCO+ORDEM+HUFFMAN		0,27	0,28	0,33	0,33
DCT INTEIRA+ZIGZAG+HUFFMAN		0,16	0,22	0,22	0,27
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,22	0,22	0,27
DWT INTEIRA+ORDEM+HUFFMAN		0,17	0,22	0,16	0,16
DWT BLOCO+ORDEM+HUFFMAN		0,22	0,22	0,33	0,33

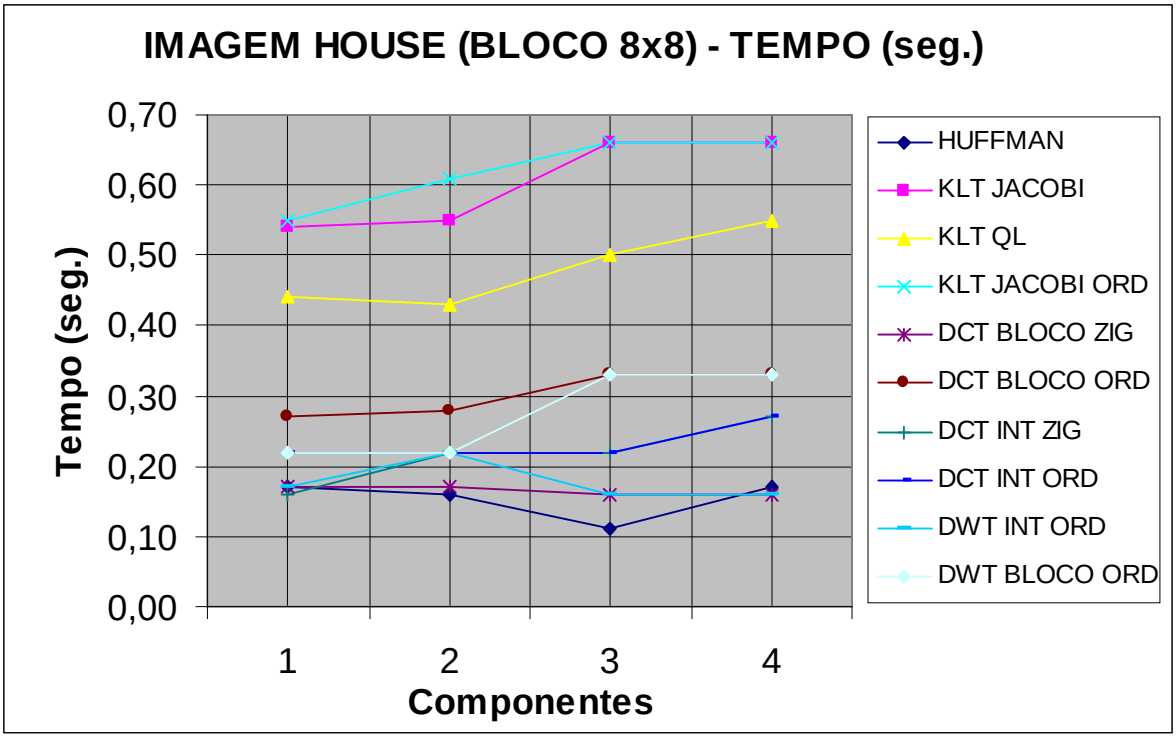


Figura 60 – Tempo de Gravação (seg) da Imagem CASA (BLOCO 8x8)

A Tabela 41 em conjunto com a Figura 61 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 41 – PSNR da Imagem CASA (BLOCO 16x16)

Imagem:	**houseg**	Bloco:	**16**	x	**16**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
KLT JACOBI + HUFFMAN		20,256547	29,894095	53,304236	61,618042
KLT QL + HUFFMAN		20,256547	29,894101	53,304454	61,618042
KLT JACOBI + ORDEM + HUFFMAN		19,938730	29,559107	53,231138	61,618042
DCT BLOCO+ZIGZAG+HUFFMAN		20,145685	27,615942	41,358750	60,286963
DCT BLOCO+ORDEM+HUFFMAN		20,145685	32,172649	52,050870	60,286963
DCT INTEIRA+ZIGZAG+HUFFMAN		20,989402	28,524942	42,179271	58,901462
DCT INTEIRA+ORDEM+HUFFMAN		22,272611	32,150516	48,310201	58,901462
DWT INTEIRA+ORDEM+HUFFMAN		23,849024	34,896450	53,118312	59,228425
DWT BLOCO+ORDEM+HUFFMAN		6,853272	29,445435	47,635685	59,262689

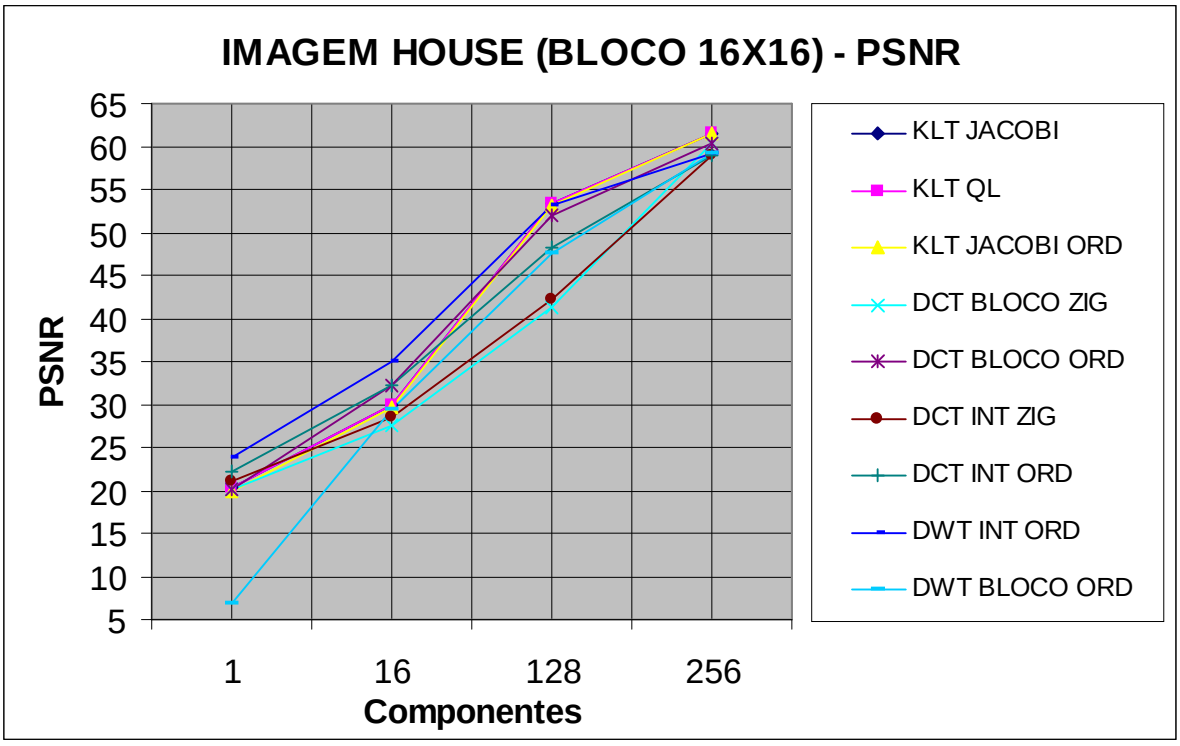


Figura 61 – PSNR da Imagem CASA (BLOCO 16x16)

A Tabela 42 em conjunto com a Figura 62 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 42 – CR da Imagem CASA (BLOCO 16x16)

Imagem:	**houseg**	Bloco:	**16**	x	**16**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,060378	1,060378	1,060378	1,060378
KLT JACOBI + HUFFMAN		3,473821	1,230380	0,359266	0,208858
KLT QL + HUFFMAN		3,473821	1,226258	0,359066	0,208783
KLT JACOBI + ORDEM + HUFFMAN		3,447751	1,222522	0,356782	0,208858
DCT BLOCO+ZIGZAG+HUFFMAN		3,841859	1,600836	1,155670	1,032135
DCT BLOCO+ORDEM+HUFFMAN		3,841859	1,526584	1,097575	1,032135
DCT INTEIRA+ZIGZAG+HUFFMAN		3,724782	1,814947	1,182441	1,015752
DCT INTEIRA+ORDEM+HUFFMAN		3,882161	1,908711	1,183407	1,015752
DWT INTEIRA+ORDEM+HUFFMAN		3,865939	1,990319	1,248669	1,139110
DWT BLOCO+ORDEM+HUFFMAN		4,018217	1,348216	0,979042	0,899460

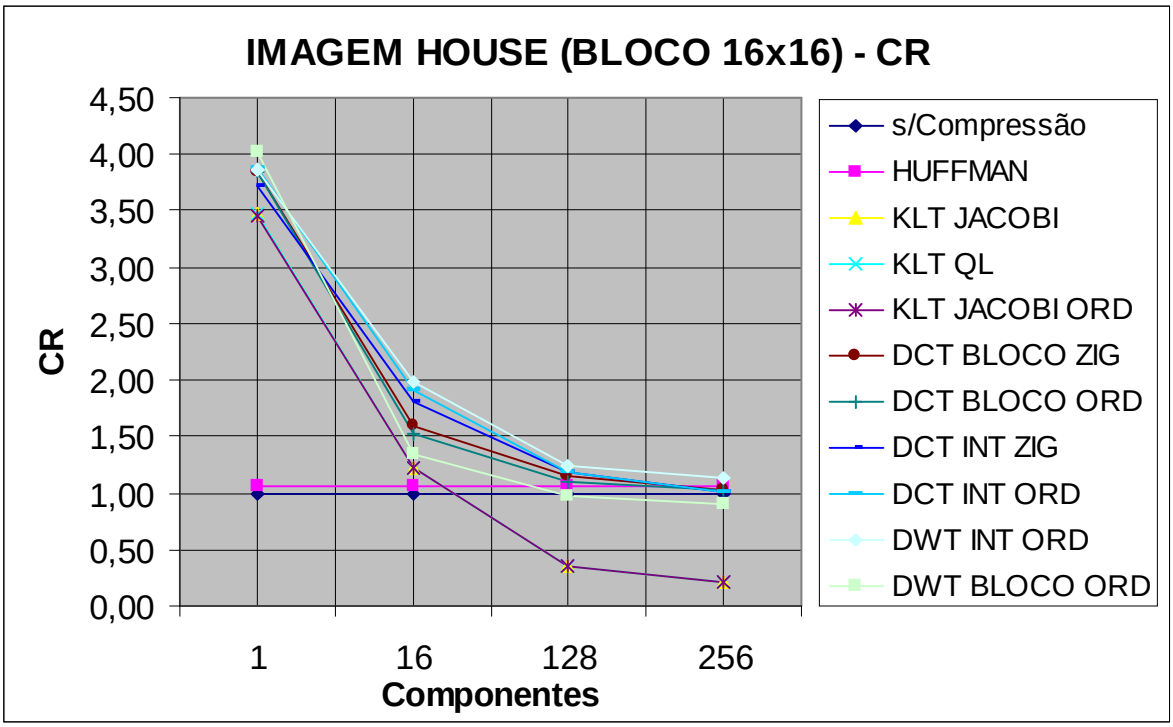


Figura 62 – CR da Imagem CASA (BLOCO 16x16)

A Tabela 43 em conjunto com a Figura 63 apresentam os resultados obtidos para a imagem CASA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 43 – Tempo de Gravação (seg) da Imagem CASA (BLOCO 16x16)

Imagem:	**houseg**	Bloco:	**16**	x	**16**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,16	0,17	0,16	0,17
KLT JACOBI + HUFFMAN		28,45	28,12	26,81	28,78
KLT QL + HUFFMAN		7,69	7,30	7,20	7,52
KLT JACOBI + ORDEM + HUFFMAN		30,37	31,41	28,56	30,15
DCT BLOCO+ZIGZAG+HUFFMAN		0,16	0,22	0,17	0,28
DCT BLOCO+ORDEM+HUFFMAN		0,22	0,22	0,33	0,33
DCT INTEIRA+ZIGZAG+HUFFMAN		0,17	0,22	0,17	0,28
DCT INTEIRA+ORDEM+HUFFMAN		0,17	0,27	0,22	0,33
DWT INTEIRA+ORDEM+HUFFMAN		0,16	0,22	0,16	0,16
DWT BLOCO+ORDEM+HUFFMAN		0,22	0,22	0,33	0,28

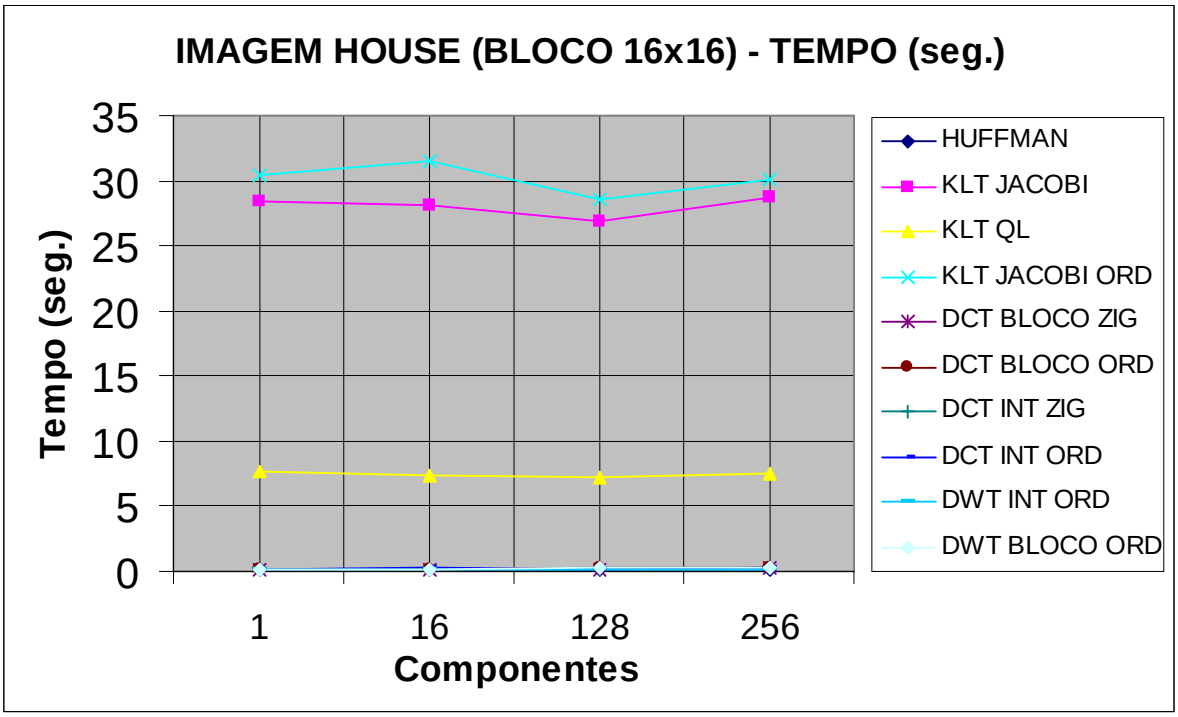


Figura 63 – Tempo de Gravação (seg) da Imagem CASA (BLOCO 16x16)

A Tabela 44 em conjunto com a Figura 64 apresentam os resultados da *PSNR (Razão Sinal/Ruído de Pico)* obtida para a imagem CASA (256X256 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 44 – PSNR da Imagem CASA – Compressão QL

Imagem:	**houseg**			
RAZÃO SINAL/RUÍDO DE PICO (PSNR)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	28,930795	33,132473	35,432638	40,593152
4:2	34,619904	40,575721	44,228315	53,304454
4:3	44,996675	50,237706	54,433953	61,568038
4:4	57,913411	59,274746	59,589434	61,618042

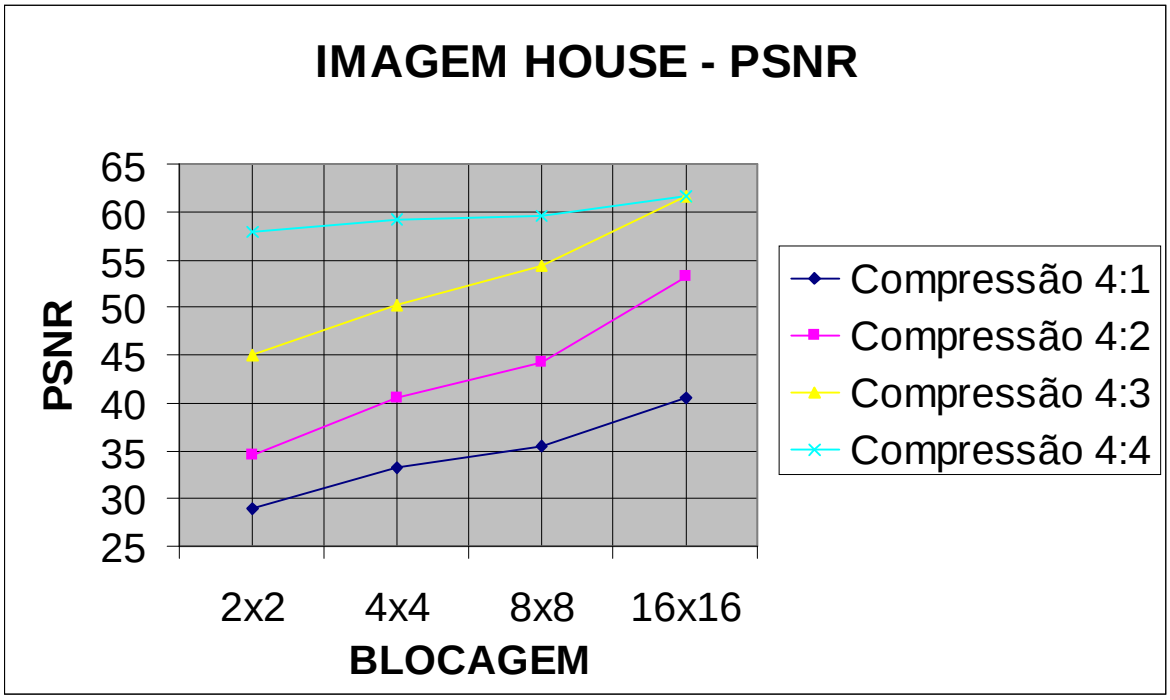


Figura 64 – PSNR da Imagem CASA – Compressão QL

A Tabela 45 em conjunto com a Figura 65 apresentam os resultados da *CR (Razão de Compressão)* obtida para a imagem CASA (256X256 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 45 – CR da Imagem CASA – Compressão QL

Imagem:	houseg			
RAZÃO DE COMPRESSÃO (CR)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	1,678400	1,391299	1,289819	0,592794
4:2	1,311506	1,156935	1,045007	0,359066
4:3	1,132102	1,048858	0,927216	0,262805
4:4	1,074454	1,022424	0,864096	0,208783

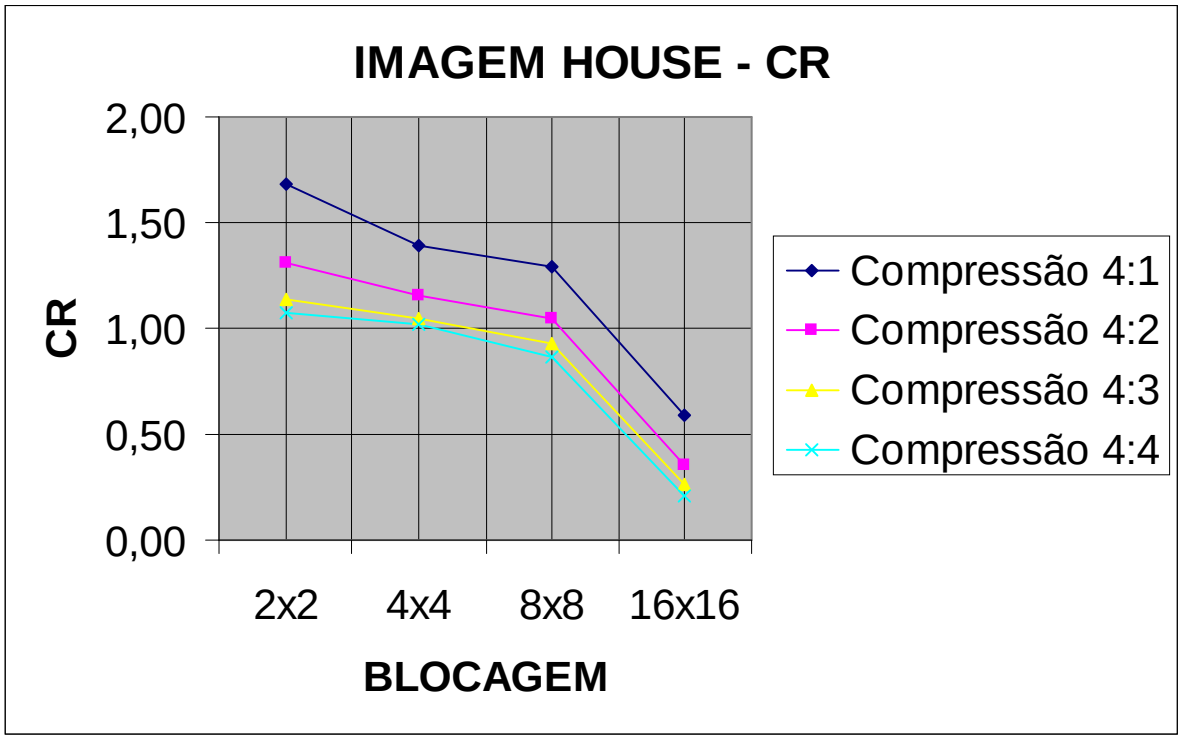


Figura 65 – CR da Imagem CASA – Compressão QL

A Tabela 46 em conjunto com a Figura 66 apresentam os resultados do *Tempo (em segundos)* obtida para a imagem CASA (256X256 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 46 – Tempo de Gravação da Imagem CASA – Compressão QL

Imagem:	**houseg**			
TEMPO DE GRAVAÇÃO (SEGUNDOS)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	0,11	0,17	0,430000	6,970000
4:2	0,17	0,22	0,500000	7,200000
4:3	0,11	0,22	0,550000	6,920000
4:4	0,22	0,28	0,550000	7,520000

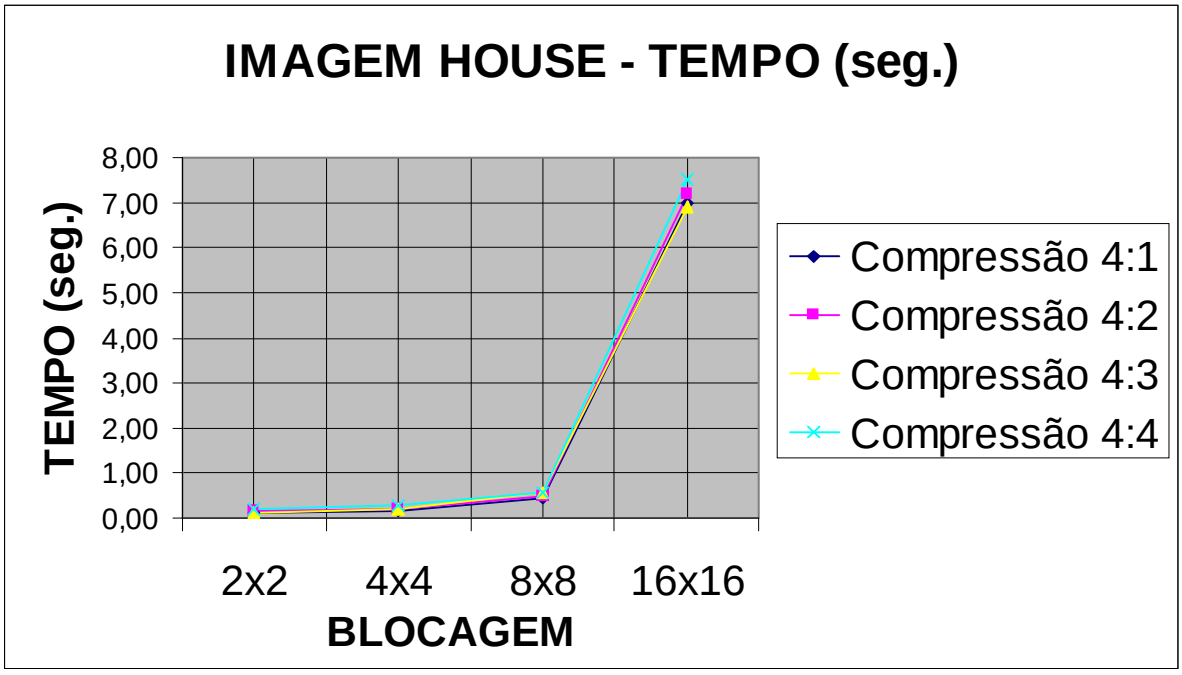


Figura 66 – Tempo de Gravação da Imagem CASA – Compressão QL

5.4 RESULTADOS OBTIDOS COM A IMAGEM MENINA (GIRL)

As tabelas e gráficos apresentados abaixo foram obtidos utilizando-se a imagem padrão MENINA (GIRL) (Figura 67), de dimensões 256 x 256 (65.536 PIXELS) com 256 níveis de escala de cinza.

Figura 67 – Figura MENINA (256 x 256)

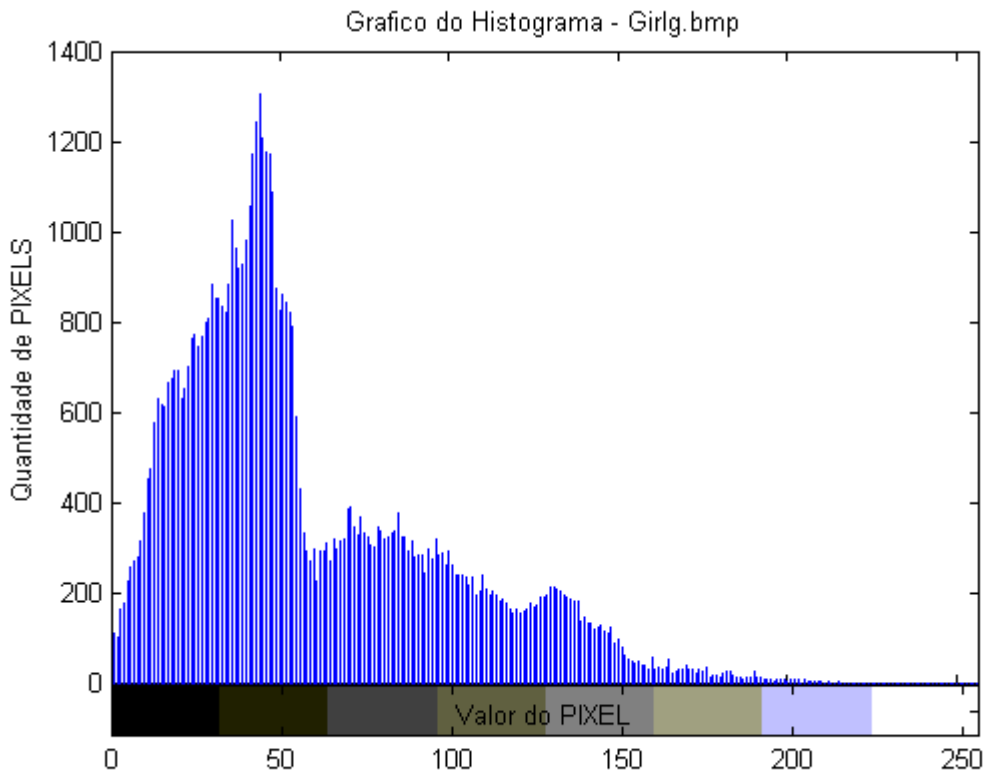


Figura 68 – Histograma da Imagem MENINA (256x256)

Como se pode observar no Histograma da Figura 68, existe uma concentração de PIXELS em valores mais escuros da escala de cinza desta imagem. A freqüência máxima de valores da escala de cinza aparece em 1 % dos pontos da imagem (1310 PIXELS), havendo boa concentração da imagem em torno desse valor de PIXEL. O valor médio do PIXEL para a esta imagem é igual a 58.

A Tabela 47 em conjunto com a Figura 69 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 47 – PSNR da Imagem MENINA (BLOCO 2x2)

Imagem:	**girlg**	Bloco:	**2**	x	**2**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
KLT JACOBI + HUFFMAN		31,834056	34,987986	43,150484	59,293760
KLT QL + HUFFMAN		31,834056	34,987986	43,150484	59,294626
KLT JACOBI + ORDEM + HUFFMAN		31,536565	34,446413	42,525292	59,293760
DCT BLOCO+ZIGZAG+HUFFMAN		31,832100	34,140964	43,209127	65,421338
DCT BLOCO+ORDEM+HUFFMAN		31,832100	38,560613	45,771665	65,421338
DCT INTEIRA+ZIGZAG+HUFFMAN		35,775011	40,572557	46,495888	58,978910
DCT INTEIRA+ORDEM+HUFFMAN		39,057878	45,974777	53,365736	58,978910
DWT INTEIRA+ORDEM+HUFFMAN		44,419268	52,966978	59,128059	59,128059

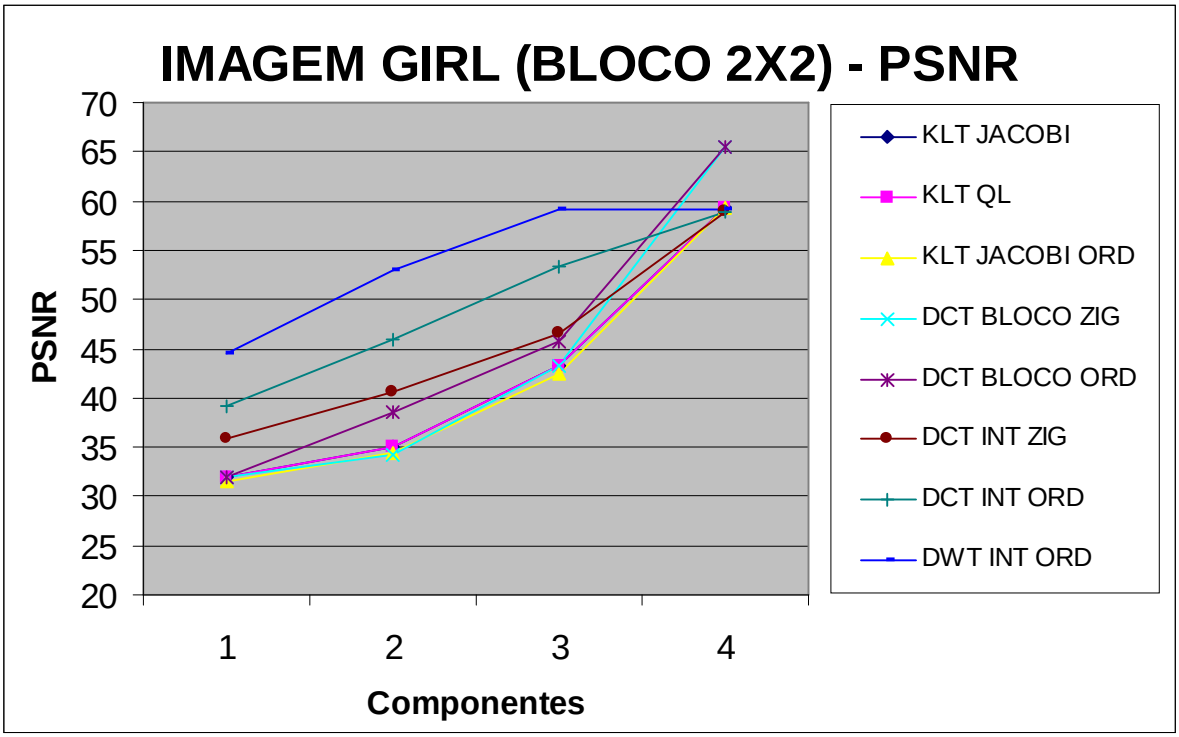


Figura 69 – PSNR da Imagem MENINA (Bloco 2x2)

A Tabela 48 em conjunto com a Figura 70 apresentam os resultados obtidos para a imagem MENINA (512x512 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 48 – CR da Imagem MENINA (BLOCO 2x2)

Imagem:	**girlg**	Bloco:	**2**	x	**2**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		0,989983	0,989983	0,989983	0,989983
KLT JACOBI + HUFFMAN		1,635623	1,303703	1,114394	1,059366
KLT QL + HUFFMAN		1,635623	1,303703	1,113872	1,059265
KLT JACOBI + ORDEM + HUFFMAN		1,634740	1,246729	1,100021	1,059366
DCT BLOCO+ZIGZAG+HUFFMAN		1,192261	0,931052	0,818877	0,789874
DCT BLOCO+ORDEM+HUFFMAN		1,192261	0,903596	0,814123	0,789874
DCT INTEIRA+ZIGZAG+HUFFMAN		1,559718	1,261892	1,104655	1,051988
DCT INTEIRA+ORDEM+HUFFMAN		1,602685	1,271478	1,096130	1,051988
DWT INTEIRA+ORDEM+HUFFMAN		1,585406	1,277868	1,161717	1,161717

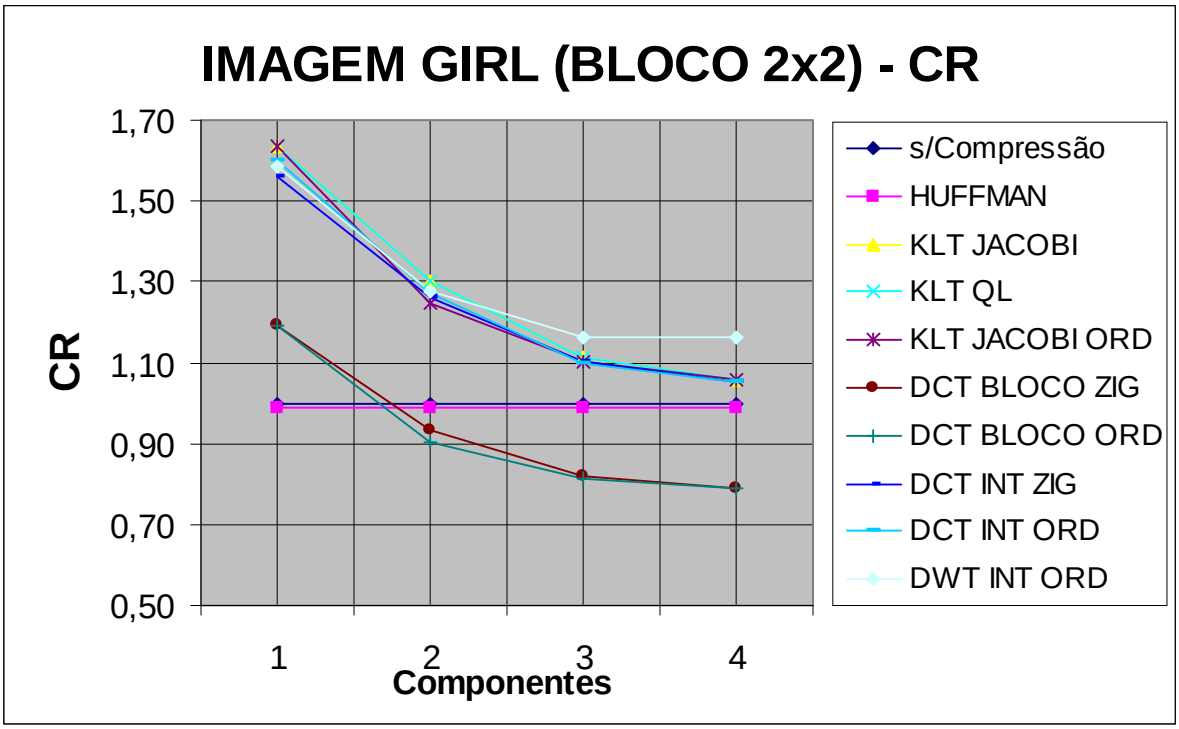


Figura 70 – CR da Imagem MENINA (Bloco 2x2)

A Tabela 49 em conjunto com a Figura 71 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 49 – Tempo (seg) de Gravação da Imagem MENINA (BLOCO 2x2)

Imagem:	**girlg**	Bloco:	**2**	x	**2**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,11	0,11	0,17	0,22
KLT JACOBI + HUFFMAN		0,17	0,11	0,16	0,22
KLT QL + HUFFMAN		0,17	0,17	0,16	0,22
KLT JACOBI + ORDEM + HUFFMAN		0,28	0,28	0,33	0,38
DCT BLOCO+ZIGZAG+HUFFMAN		0,22	0,33	0,32	0,27
DCT BLOCO+ORDEM+HUFFMAN		0,50	0,55	0,55	0,55
DCT INTEIRA+ZIGZAG+HUFFMAN		0,16	0,22	0,16	0,28
DCT INTEIRA+ORDEM+HUFFMAN		0,17	0,22	0,22	0,27
DWT INTEIRA+ORDEM+HUFFMAN		0,17	0,16	0,17	0,16

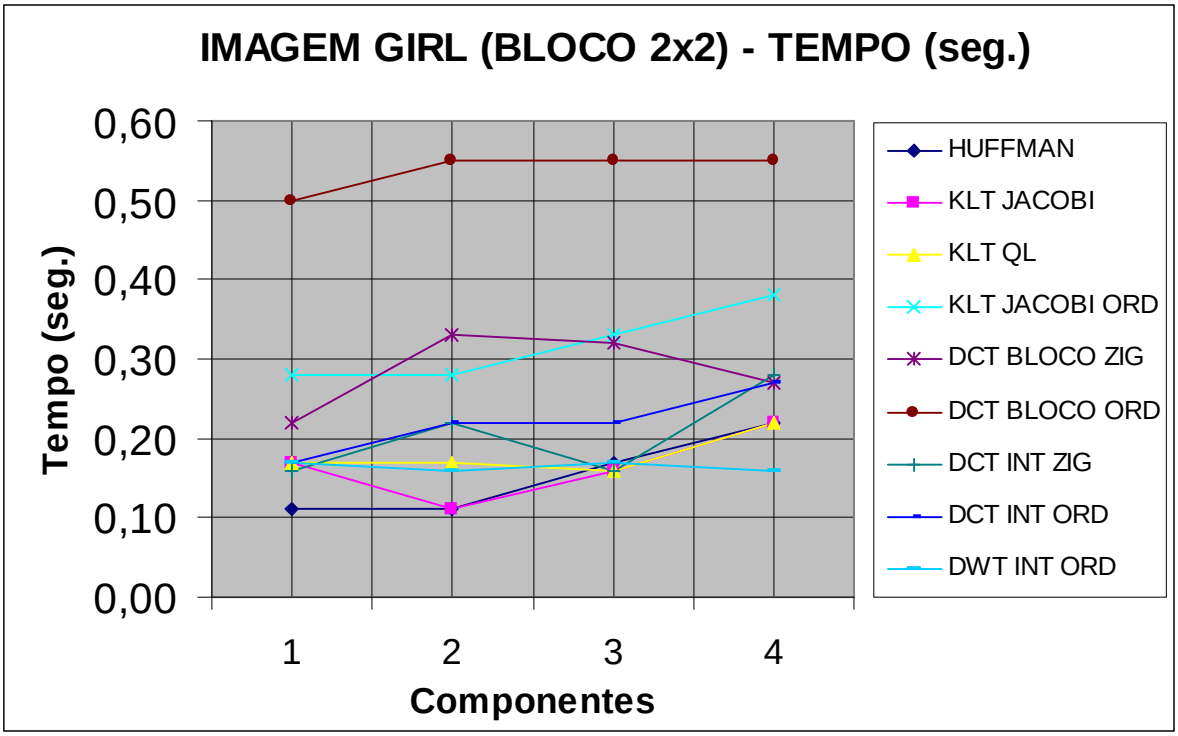


Figura 71 – Tempo (seg) de Gravação da Imagem MENINA (Bloco 2x2)

A Tabela 50 em conjunto com a Figura 72 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 50 – PSNR da Imagem MENINA (BLOCO 4x4)

Imagem:	**girlg**	Bloco:	**4**	x	**4**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
KLT JACOBI + HUFFMAN		27,255469	33,651165	39,300217	59,285107
KLT QL + HUFFMAN		27,255469	33,651158	39,300339	59,285971
KLT JACOBI + ORDEM + HUFFMAN		26,843710	33,192681	38,899713	59,285107
DCT BLOCO+ZIGZAG+HUFFMAN		27,244021	33,400217	37,981135	61,922855
DCT BLOCO+ORDEM+HUFFMAN		27,244021	36,500429	44,540166	61,922855
DCT INTEIRA+ZIGZAG+HUFFMAN		30,067564	35,775011	40,572557	58,978910
DCT INTEIRA+ORDEM+HUFFMAN		32,084362	39,057878	45,974777	58,978910
DWT INTEIRA+ORDEM+HUFFMAN		34,347547	44,419268	52,966978	59,128059

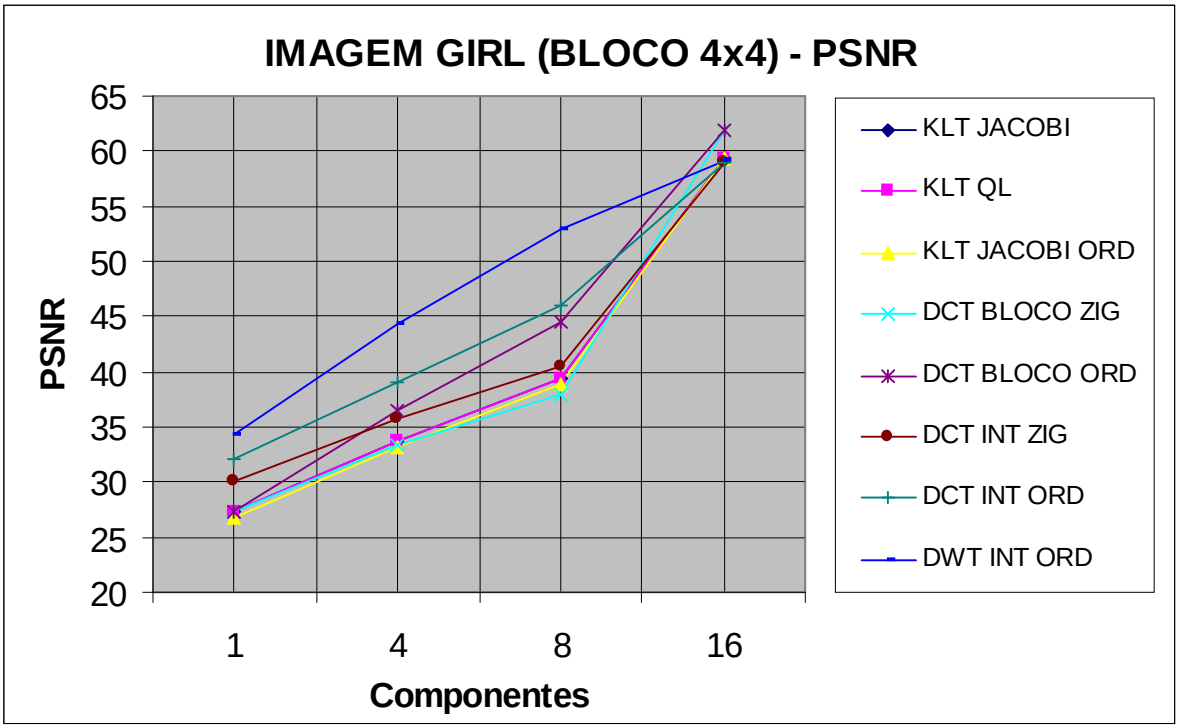


Figura 72 – PSNR da Imagem MENINA (Bloco 4x4)

A Tabela 51 em conjunto com a Figura 73 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 51 – CR da Imagem MENINA (BLOCO 4x4)

Imagem:	**girlg**	Bloco:	**4**	x	**4**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		0,989983	0,989983	0,989983	0,989983
KLT JACOBI + HUFFMAN		1,797415	1,425478	1,202050	1,059164
KLT QL + HUFFMAN		1,797415	1,428014	1,203722	1,060631
KLT JACOBI + ORDEM + HUFFMAN		1,811936	1,390776	1,165579	1,059164
DCT BLOCO+ZIGZAG+HUFFMAN		1,231904	0,933610	0,836197	0,766736
DCT BLOCO+ORDEM+HUFFMAN		1,231904	0,919486	0,814810	0,766736
DCT INTEIRA+ZIGZAG+HUFFMAN		2,003429	1,559718	1,261892	1,051988
DCT INTEIRA+ORDEM+HUFFMAN		2,127359	1,602685	1,271478	1,051988
DWT INTEIRA+ORDEM+HUFFMAN		2,095966	1,585406	1,277868	1,161717

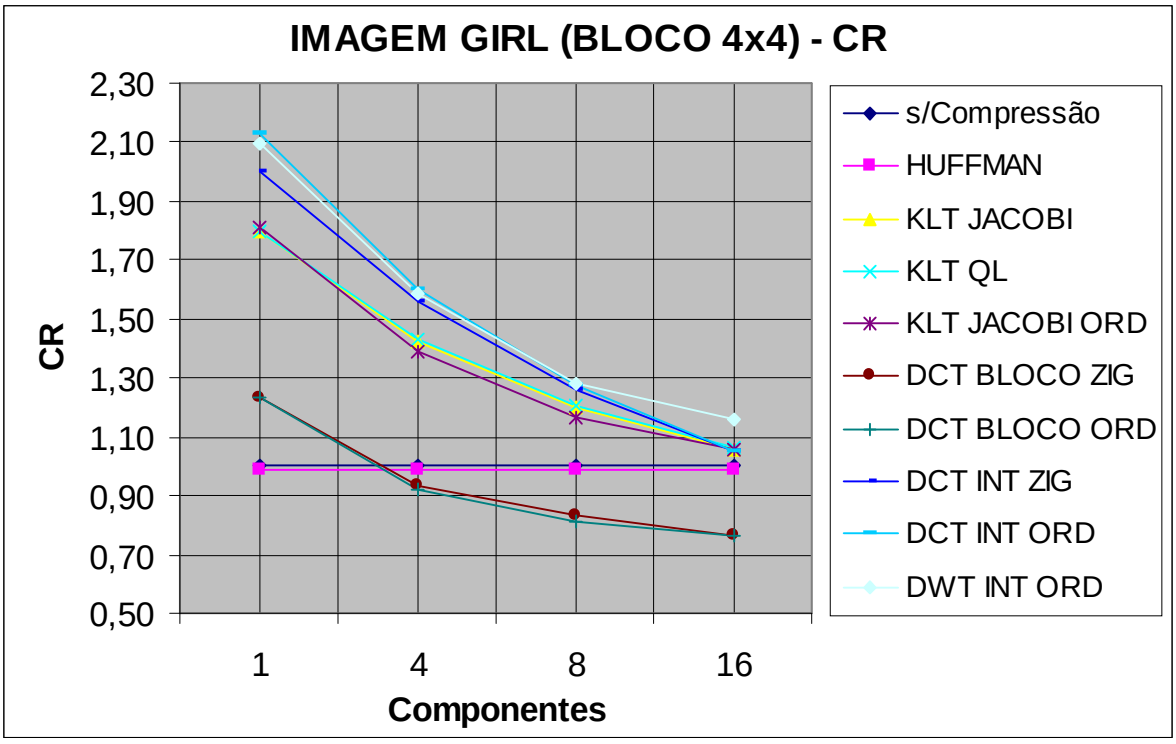


Figura 73 – CR da Imagem MENINA (BLOCO 4x4)

A Tabela 52 em conjunto com a Figura 74 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 52 – Tempo (seg) de Gravação da Imagem MENINA (BLOCO 4x4)

Imagem:	**girlg**	Bloco:	**4**	x	**4**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,16	0,11	0,11	0,16
KLT JACOBI + HUFFMAN		0,22	0,22	0,16	0,27
KLT QL + HUFFMAN		0,16	0,22	0,22	0,27
KLT JACOBI + ORDEM + HUFFMAN		0,27	0,33	0,28	0,33
DCT BLOCO+ZIGZAG+HUFFMAN		0,22	0,22	0,28	0,28
DCT BLOCO+ORDEM+HUFFMAN		0,33	0,38	0,38	0,38
DCT INTEIRA+ZIGZAG+HUFFMAN		0,16	0,22	0,17	0,22
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,22	0,22	0,27
DWT INTEIRA+ORDEM+HUFFMAN		0,16	0,22	0,22	0,16

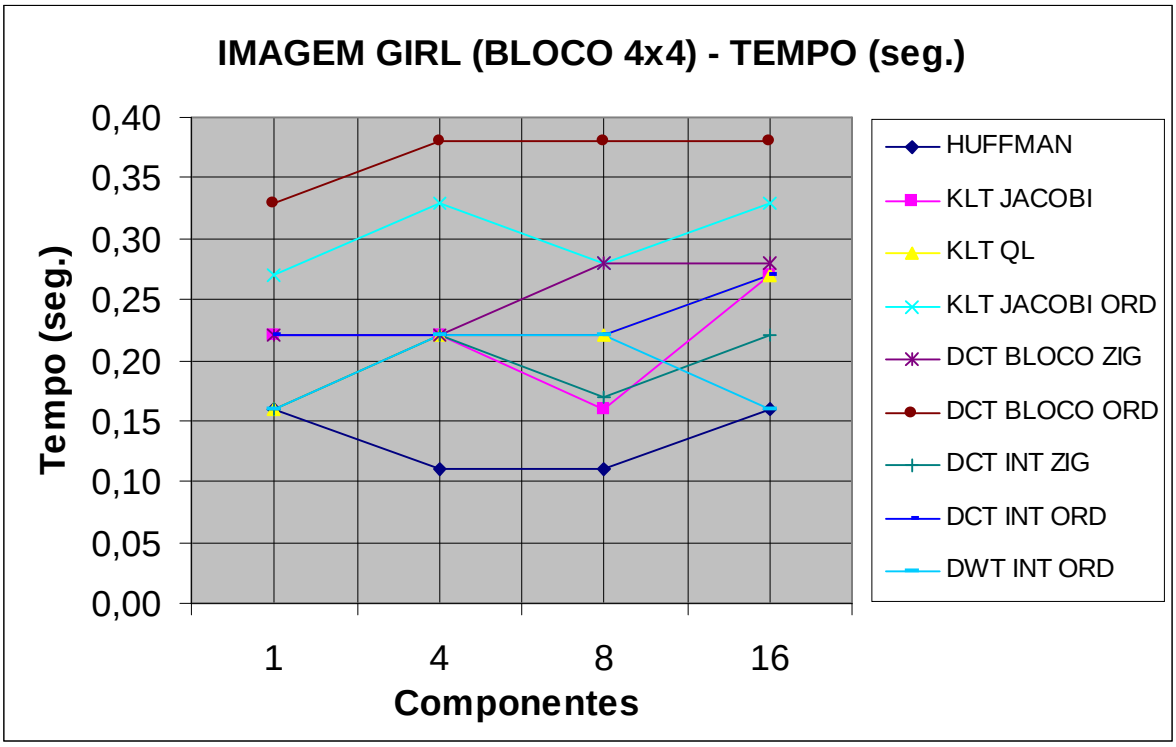


Figura 74 – Tempo (seg) de Gravação da Imagem MENINA (BLOCO 4x4)

A Tabela 53 em conjunto com a Figura 75 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 53 – PSNR da Imagem MENINA (BLOCO 8x8)

Imagem:	**girlg**	Bloco:	**8**	x	**8**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
KLT JACOBI + HUFFMAN		24,281818	35,410100	41,504646	59,348674
KLT QL + HUFFMAN		24,281812	35,410160	41,503623	59,346920
KLT JACOBI + ORDEM + HUFFMAN		24,028887	35,069572	41,208852	59,348674
DCT BLOCO+ZIGZAG+HUFFMAN		24,249102	34,532268	39,790517	62,434182
DCT BLOCO+ORDEM+HUFFMAN		24,249102	39,467963	54,369371	62,434182
DCT INTEIRA+ZIGZAG+HUFFMAN		26,343213	35,775011	40,572557	58,978910
DCT INTEIRA+ORDEM+HUFFMAN		27,730260	39,057878	45,974777	58,978910
DWT INTEIRA+ORDEM+HUFFMAN		28,590872	44,419268	52,966978	59,128059
DWT BLOCO+ORDEM+HUFFMAN		13,330047	36,017477	43,734947	59,185972

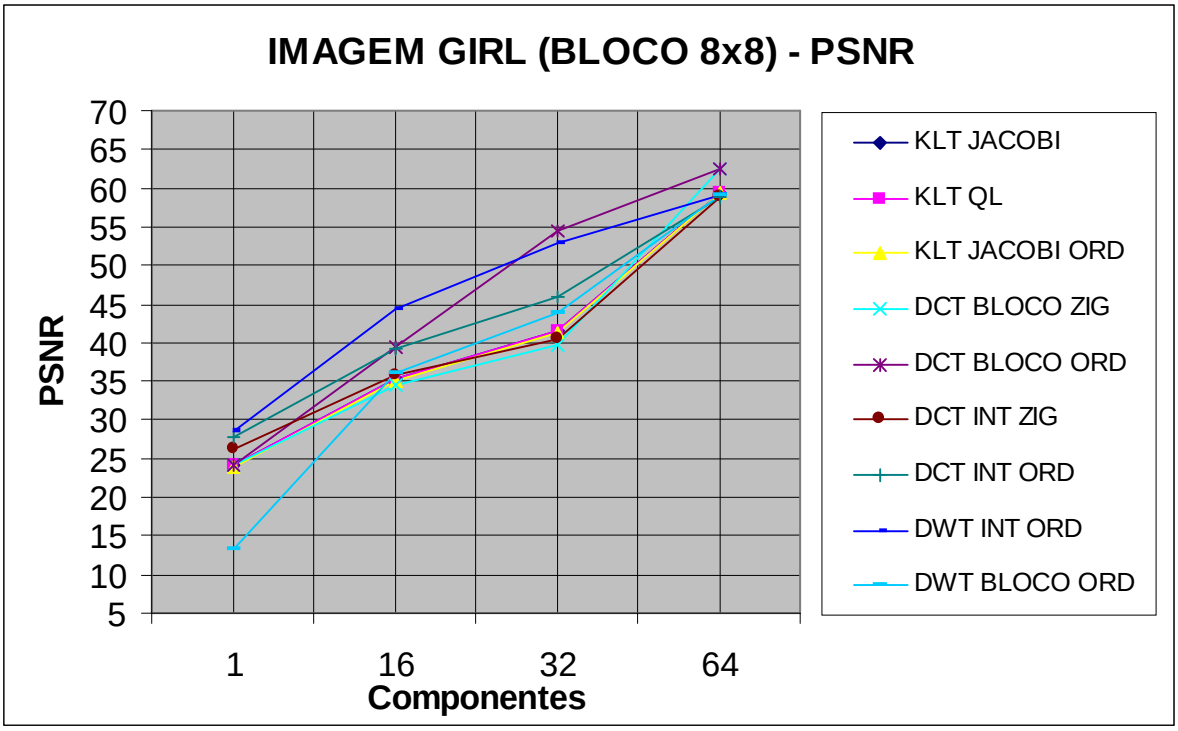


Figura 75 – PSNR da Imagem MENINA (BLOCO 8x8)

A Tabela 54 em conjunto com a Figura 76 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 54 – CR da Imagem MENINA (BLOCO 8x8)

Imagem:	**girlg**	Bloco:	**8**	x	**8**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		0,989983	0,989983	0,989983	0,989983
KLT JACOBI + HUFFMAN		2,305541	1,307387	1,062119	0,858915
KLT QL + HUFFMAN		2,305541	1,313963	1,066438	0,861860
KLT JACOBI + ORDEM + HUFFMAN		2,282474	1,275861	1,031224	0,858915
DCT BLOCO+ZIGZAG+HUFFMAN		2,450576	1,318332	1,230539	1,187288
DCT BLOCO+ORDEM+HUFFMAN		2,450576	1,258911	1,191813	1,187288
DCT INTEIRA+ZIGZAG+HUFFMAN		2,270648	1,559718	1,261892	1,051988
DCT INTEIRA+ORDEM+HUFFMAN		2,610062	1,602685	1,271478	1,051988
DWT INTEIRA+ORDEM+HUFFMAN		2,586650	1,585406	1,277868	1,161717
DWT BLOCO+ORDEM+HUFFMAN		2,538547	1,248739	1,052404	0,955327

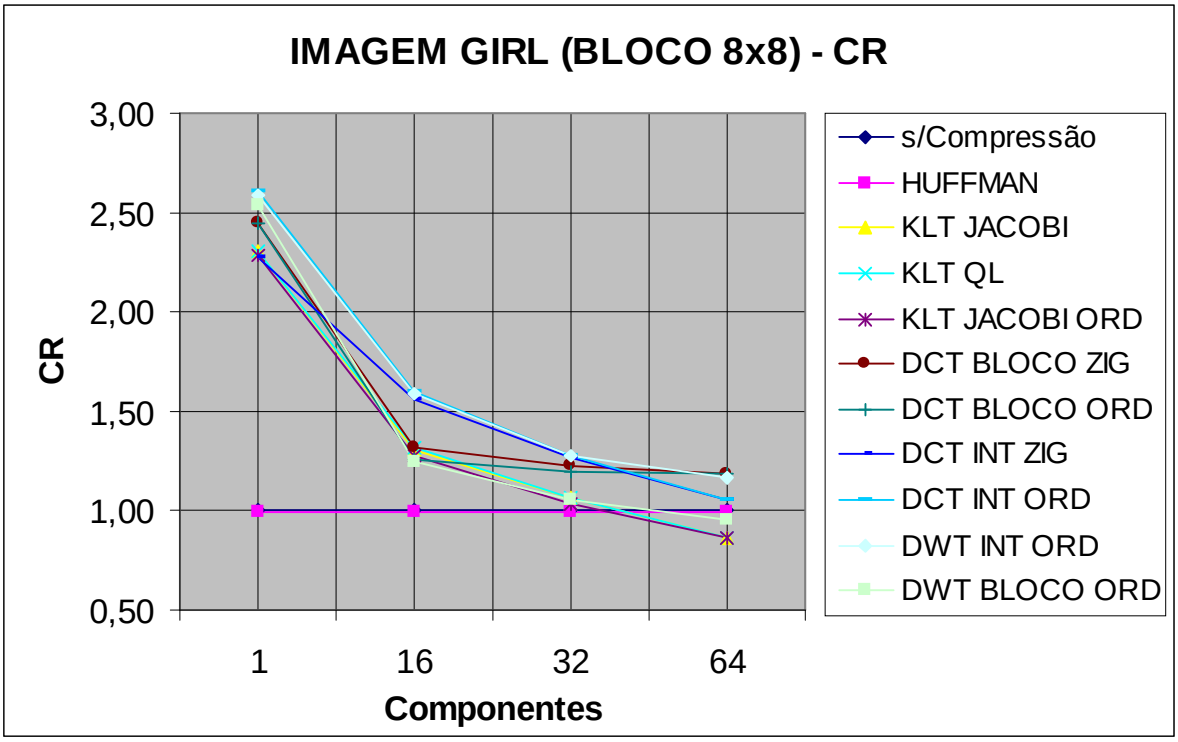


Figura 76 – CR da Imagem MENINA (BLOCO 8x8)

A Tabela 55 em conjunto com a Figura 77 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 55 – Tempo de Gravação (seg) da Imagem MENINA (BLOCO 8x8)

Imagem:	**girlg**	Bloco:	**8**	x	**8**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,16	0,16	0,16	0,17
KLT JACOBI + HUFFMAN		0,55	0,55	0,66	0,66
KLT QL + HUFFMAN		0,44	0,44	0,50	0,55
KLT JACOBI + ORDEM + HUFFMAN		0,60	0,60	0,66	0,66
DCT BLOCO+ZIGZAG+HUFFMAN		0,16	0,22	0,22	0,22
DCT BLOCO+ORDEM+HUFFMAN		0,22	0,28	0,28	0,22
DCT INTEIRA+ZIGZAG+HUFFMAN		0,17	0,21	0,22	0,27
DCT INTEIRA+ORDEM+HUFFMAN		0,16	0,22	0,22	0,22
DWT INTEIRA+ORDEM+HUFFMAN		0,17	0,17	0,16	0,17
DWT BLOCO+ORDEM+HUFFMAN		0,22	0,22	0,33	0,28

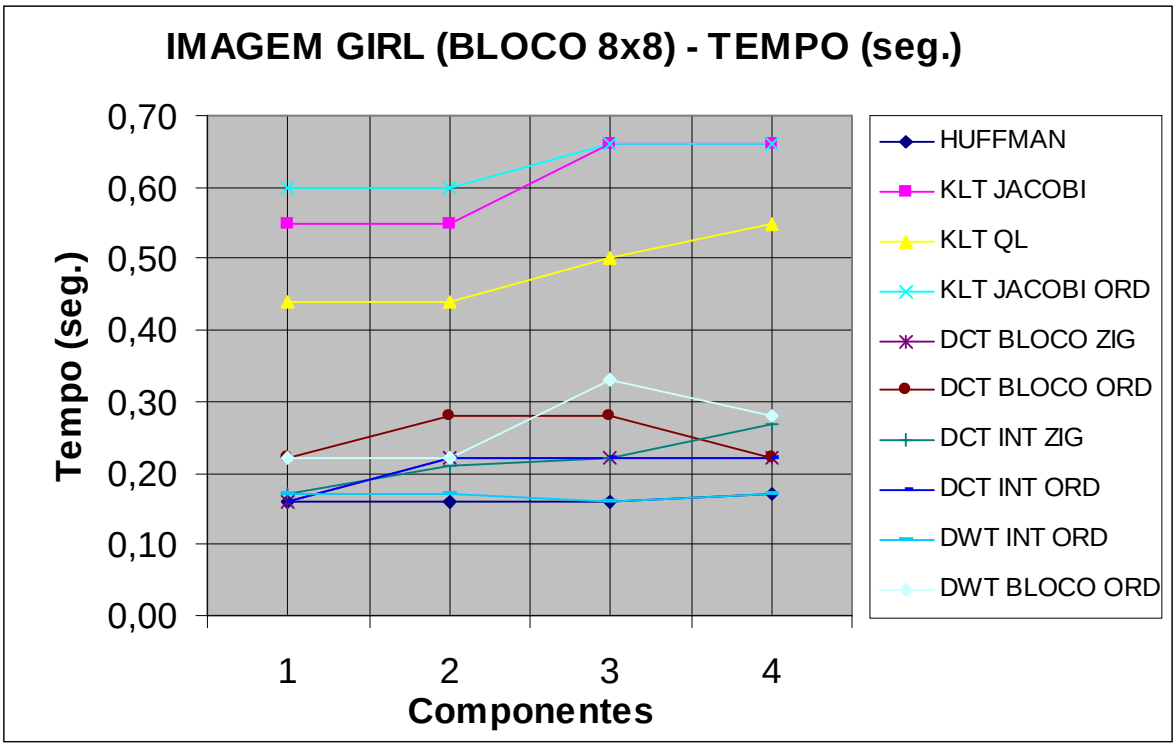


Figura 77 – Tempo de Gravação (seg) da Imagem MENINA (BLOCO 8x8)

A Tabela 56 em conjunto com a Figura 78 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 56 – PSNR da Imagem MENINA (BLOCO 16x16)

Imagem:	**girlg**	Bloco:	**16**	x	**16**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
KLT JACOBI + HUFFMAN		21,869578	30,976849	53,367064	61,429881
KLT QL + HUFFMAN		21,869572	30,976770	53,361756	61,372190
KLT JACOBI + ORDEM + HUFFMAN		21,574058	30,638012	53,331803	61,429881
DCT BLOCO+ZIGZAG+HUFFMAN		21,806234	29,205309	40,447091	60,103626
DCT BLOCO+ORDEM+HUFFMAN		21,806234	31,636515	48,342226	60,103626
DCT INTEIRA+ZIGZAG+HUFFMAN		23,258753	30,067564	40,572557	58,978910
DCT INTEIRA+ORDEM+HUFFMAN		24,483589	32,084362	45,974777	58,978910
DWT INTEIRA+ORDEM+HUFFMAN		24,469880	34,347547	52,966978	59,128059
DWT BLOCO+ORDEM+HUFFMAN		13,358677	29,623215	45,636297	59,113909

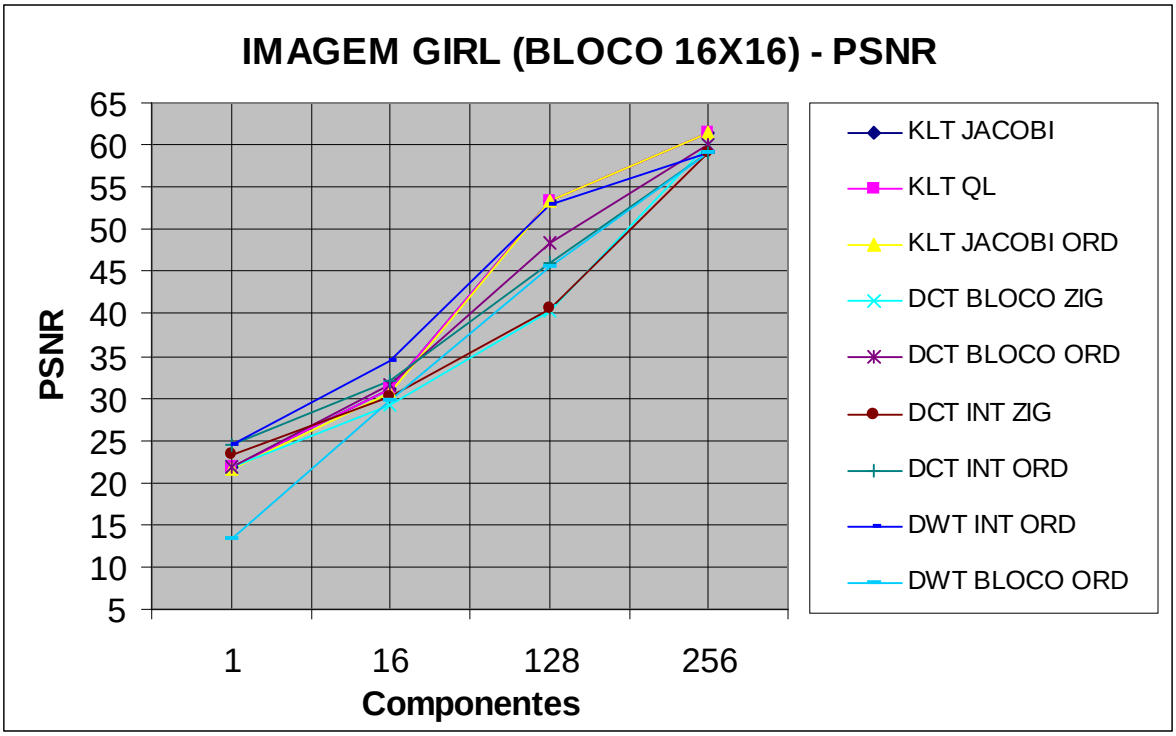


Figura 78 – PSNR da Imagem MENINA (BLOCO 16x16)

A Tabela 57 em conjunto com a Figura 79 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 57 – CR da Imagem MENINA (BLOCO 16x16)

Imagem:	**girlg**	Bloco:	**16**	x	**16**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		0,989983	0,989983	0,989983	0,989983
KLT JACOBI + HUFFMAN		3,328869	1,258340	0,362531	0,209847
KLT QL + HUFFMAN		3,328869	1,253179	0,362176	0,209731
KLT JACOBI + ORDEM + HUFFMAN		3,310835	1,250826	0,359698	0,209847
DCT BLOCO+ZIGZAG+HUFFMAN		3,679925	1,664851	1,211560	1,076886
DCT BLOCO+ORDEM+HUFFMAN		3,679925	1,640901	1,145850	1,076886
DCT INTEIRA+ZIGZAG+HUFFMAN		3,636334	2,003429	1,261892	1,051988
DCT INTEIRA+ORDEM+HUFFMAN		3,817640	2,127359	1,271478	1,051988
DWT INTEIRA+ORDEM+HUFFMAN		3,949134	2,095966	1,277868	1,161717
DWT BLOCO+ORDEM+HUFFMAN		3,833679	1,440085	1,012063	0,932134

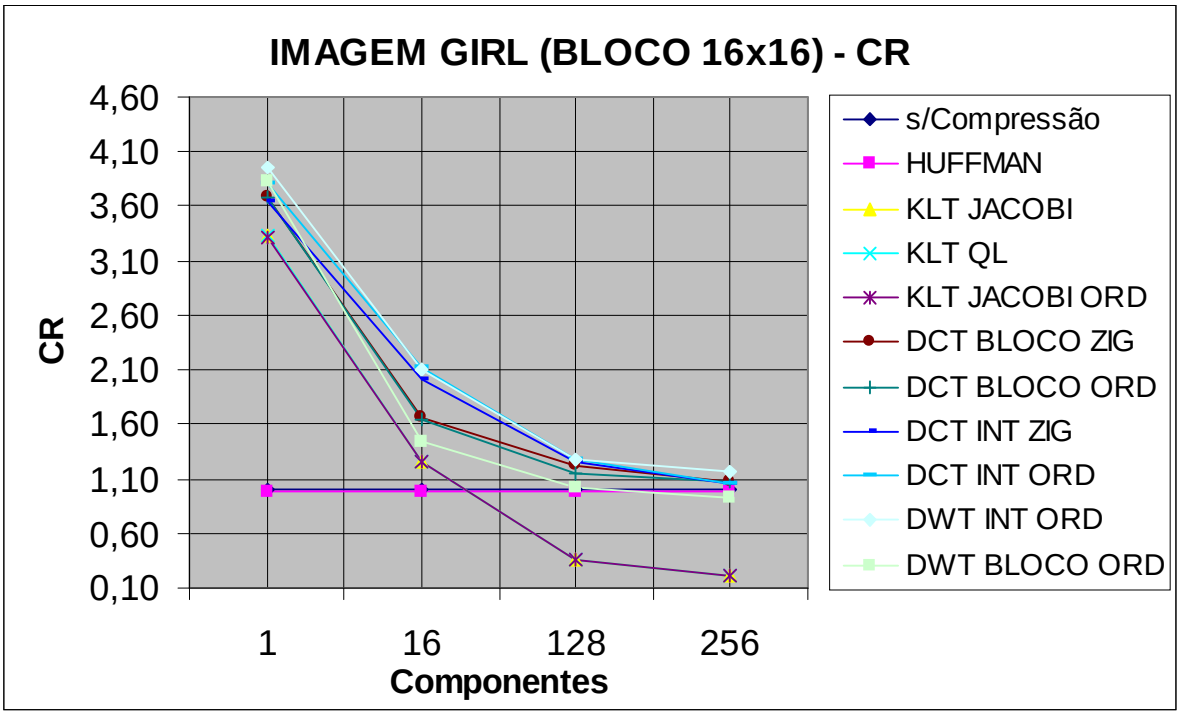


Figura 79 – CR da Imagem MENINA (BLOCO 16x16)

A Tabela 58 em conjunto com a Figura 80 apresentam os resultados obtidos para a imagem MENINA (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 58 – Tempo de Gravação (seg) da Imagem MENINA (BLOCO 16x16)

Imagem:	**girlg**	Bloco:	**16**	x	**16**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,17	0,16	0,11	0,11
KLT JACOBI + HUFFMAN		28,12	27,68	25,60	26,14
KLT QL + HUFFMAN		6,20	6,37	6,43	6,87
KLT JACOBI + ORDEM + HUFFMAN		26,58	27,85	27,46	28,34
DCT BLOCO+ZIGZAG+HUFFMAN		0,16	0,22	0,22	0,27
DCT BLOCO+ORDEM+HUFFMAN		0,22	0,22	0,22	0,27
DCT INTEIRA+ZIGZAG+HUFFMAN		0,16	0,16	0,22	0,27
DCT INTEIRA+ORDEM+HUFFMAN		0,16	0,22	0,22	0,28
DWT INTEIRA+ORDEM+HUFFMAN		0,17	0,16	0,17	0,22
DWT BLOCO+ORDEM+HUFFMAN		0,17	0,16	0,22	0,22

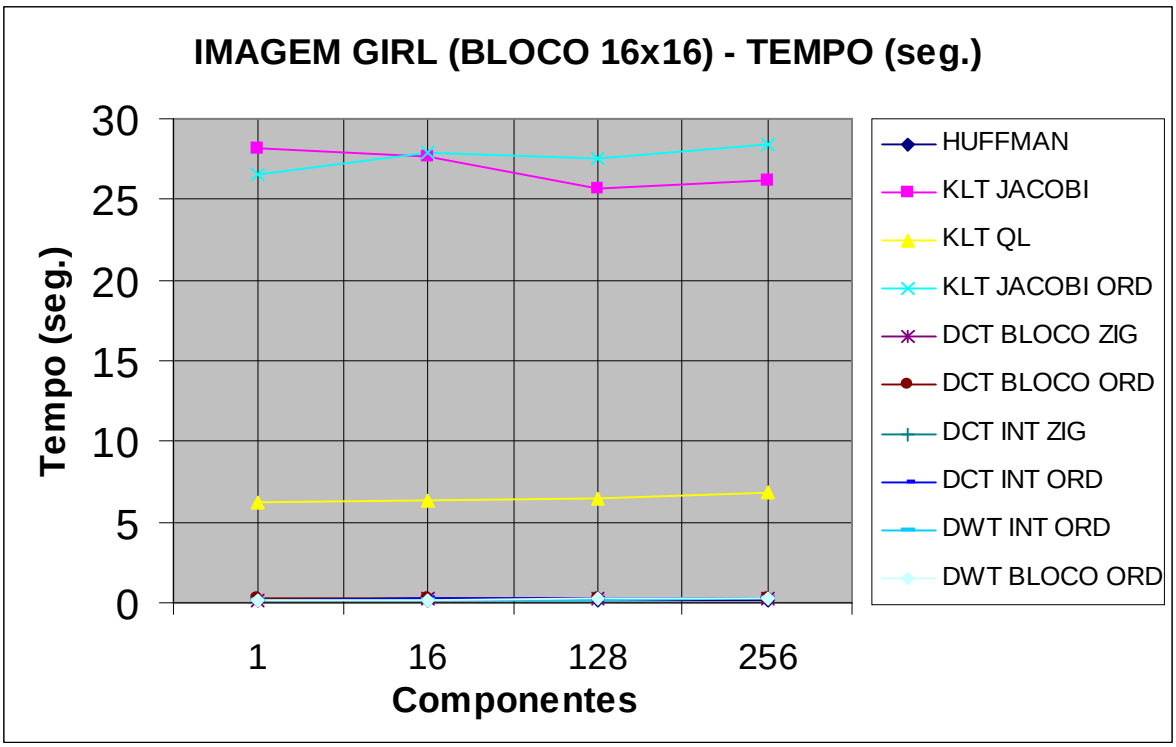


Figura 80 – Tempo de Gravação (seg) da Imagem MENINA (BLOCO 16x16)

A Tabela 59 em conjunto com a Figura 81 apresentam os resultados da *PSNR (Razão Sinal/Ruído de Pico)* obtida para a imagem MENINA (GIRL) (256X256 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 59 – PSNR da Imagem MENINA - Compressão QL

Imagem:	girlg			
RAZÃO SINAL/RUÍDO DE PICO (PSNR)				
Compressão	Blocagem da Imagem			
	2x2	4x4	8x8	16x16
4:1	31,834056	33,651158	35,410160	41,413439
4:2	34,987986	39,300339	41,503623	53,361756
4:3	43,150484	45,847384	49,760005	61,300107
4:4	59,294626	59,285971	59,346920	61,372190

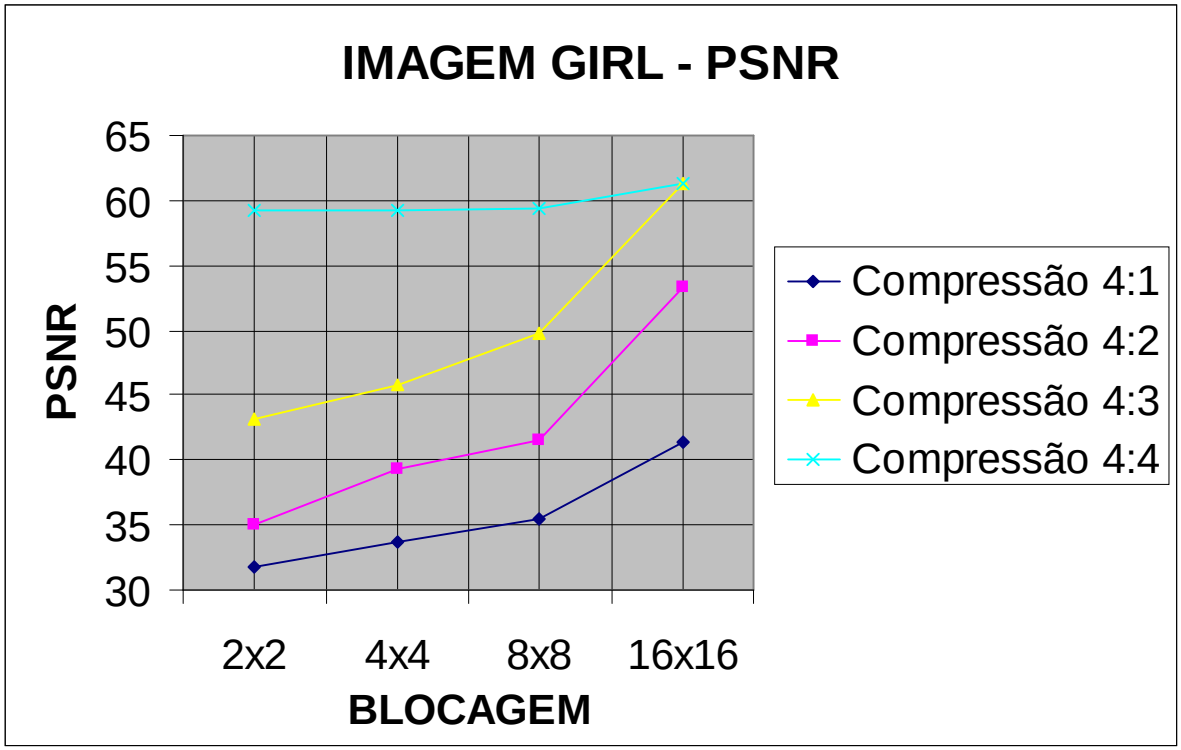


Figura 81 – PSNR da Imagem MENINA – Compressão QL

A Tabela 60 em conjunto com a Figura 82 apresentam os resultados da *CR (Razão de Compressão)* obtida para a imagem MENINA (GIRL) (256X256 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 60 – CR da Imagem MENINA – Compressão QL

Imagem:	girlg			
RAZÃO DE COMPRESSÃO (CR)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	1,635623	1,428014	1,313963	0,600348
4:2	1,303703	1,203722	1,066438	0,362176
4:3	1,113872	1,102078	0,933204	0,264333
4:4	1,059265	1,060631	0,861860	0,209731

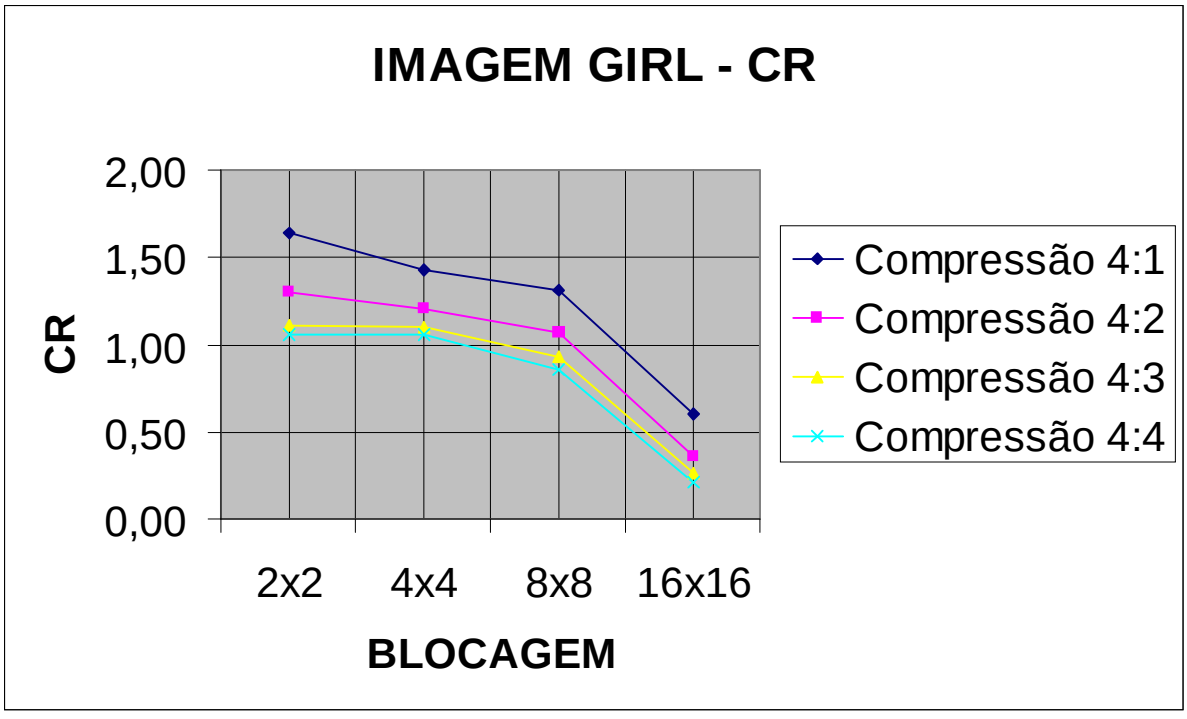


Figura 82 – CR da Imagem MENINA – Compressão QL

A Tabela 61 em conjunto com a Figura 83 apresentam os resultados do *Tempo (em segundos)* obtida para a imagem MENINA (GIRL) (256X256 PIXELS) para 4 tamanhos de bloco de imagem, 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 61 – Tempo de Gravação da Imagem MENINA – Compressão QL

Imagem:	girlg			
TEMPO DE GRAVAÇÃO (SEGUNDOS)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	0,17	0,22	0,440000	6,480000
4:2	0,17	0,22	0,500000	6,430000
4:3	0,16	0,28	0,600000	6,700000
4:4	0,22	0,27	0,550000	6,870000

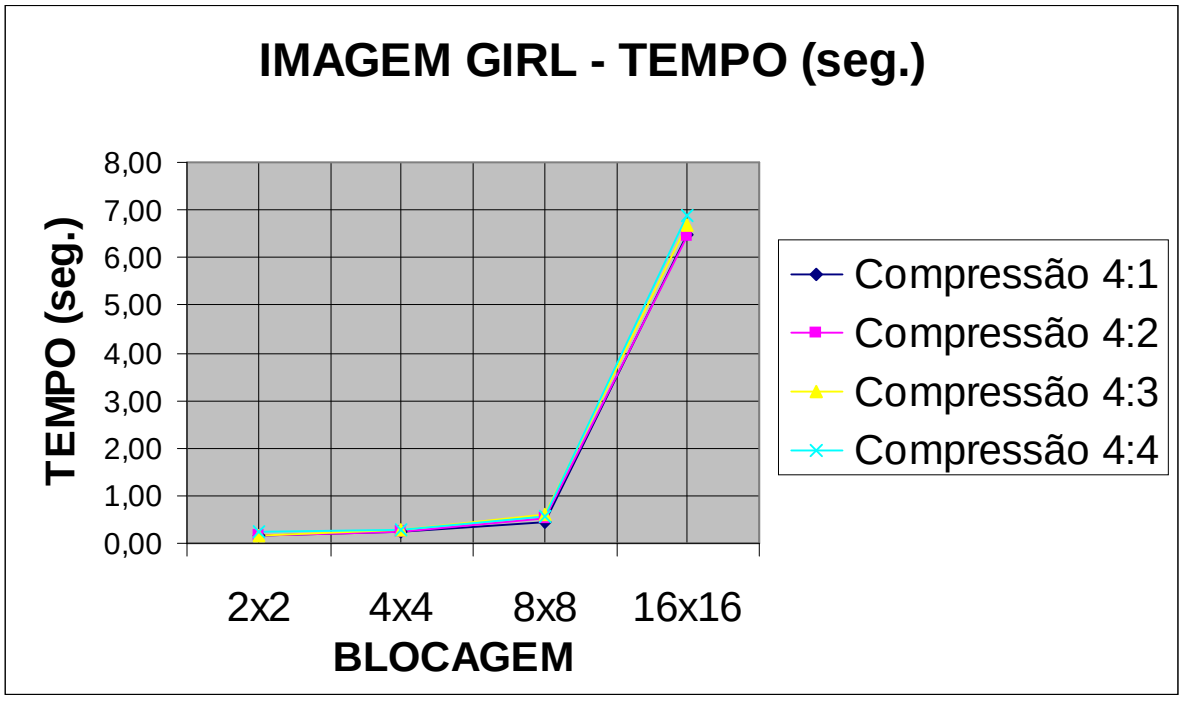


Figura 83 – Tempo de Gravação da Imagem MENINA – Compressão QL

5.5 RESULTADOS OBTIDOS COM A IMAGEM TIGRE

As tabelas e gráficos apresentados abaixo foram obtidos utilizando-se a imagem padrão TIGRE (Figura 84), de dimensões 256 x 256 (65.536 PIXELS) com 256 níveis de escala de cinza.

Figura 84 – Figura TIGRE (256 x 256)

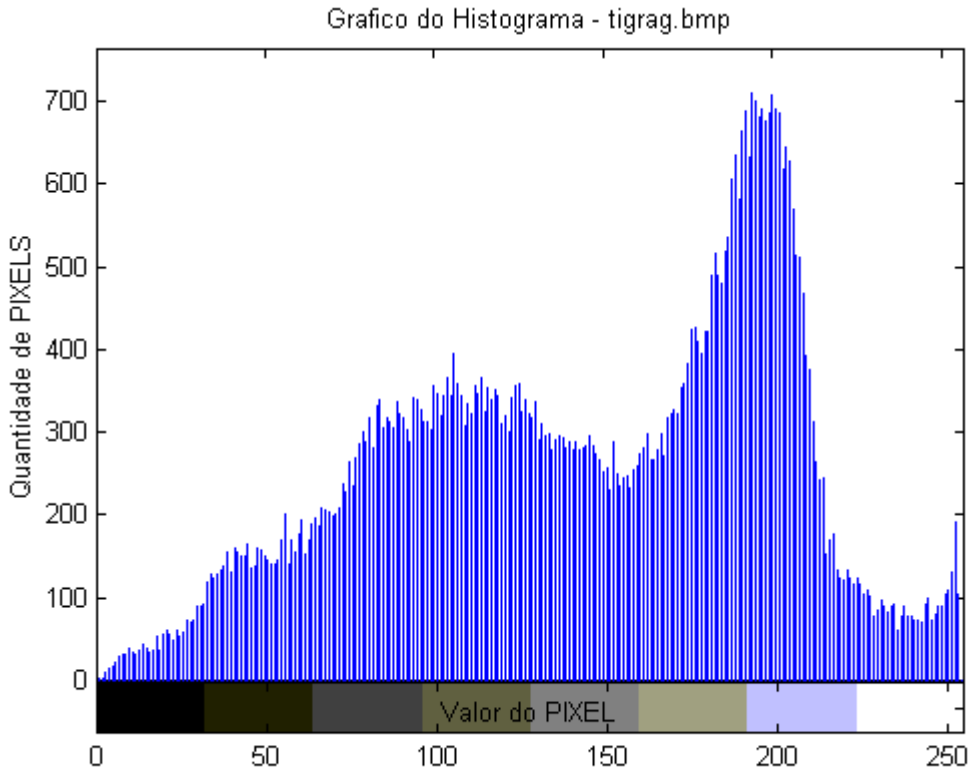


Figura 85 – Histograma da Imagem TIGRE (256x256)

Como se pode observar no Histograma da Figura 85, existe uma concentração de PIXELS em valores mais claros da escala de cinza desta imagem. A freqüência máxima de valores da escala de cinza aparece em 1 % dos pontos da imagem (712 PIXELS), havendo uma dispersão de PIXELS em todo o intervalo de valores da escala (bom contraste). O valor médio do PIXEL para a esta imagem é igual a 143.

A Tabela 62 em conjunto com a Figura 86 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 62 – PSNR da Imagem TIGRE (BLOCO 2x2)

Imagem:	**tigrag**	Bloco:	**2**	x	**2**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
KLT JACOBI + HUFFMAN		24,087323	28,499992	35,033915	58,705727
KLT QL + HUFFMAN		24,087323	28,499992	35,033915	58,705727
KLT JACOBI + ORDEM + HUFFMAN		23,661725	27,965539	34,367941	58,705727
DCT BLOCO+ZIGZAG+HUFFMAN		24,086821	28,476684	35,039723	65,129525
DCT BLOCO+ORDEM+HUFFMAN		24,086821	31,270362	37,993997	65,129525
DCT INTEIRA+ZIGZAG+HUFFMAN		27,412991	33,930324	41,306628	59,009630
DCT INTEIRA+ORDEM+HUFFMAN		31,932583	40,144017	49,564564	59,009630
DWT INTEIRA+ORDEM+HUFFMAN		33,949342	42,847869	51,985183	58,922881

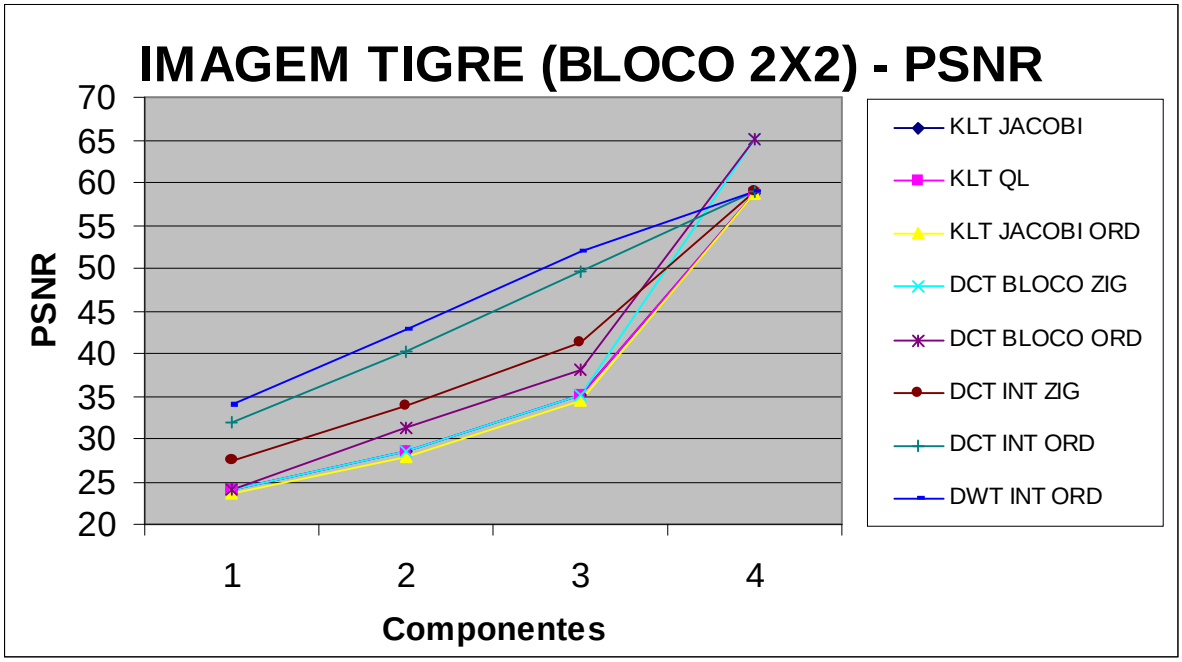


Figura 86 – PSNR da Imagem TIGRE (Bloco 2x2)

A Tabela 63 em conjunto com a Figura 87 apresentam os resultados obtidos para a imagem TIGRE (512x512 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 63 – CR da Imagem TIGRE (BLOCO 2x2)

Imagem:	**tigrag**	Bloco:	**2**	x	**2**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		0,904664	0,904664	0,904664	0,904664
KLT JACOBI + HUFFMAN		1,486920	1,130699	0,953372	0,883416
KLT QL + HUFFMAN		1,486920	1,130699	0,953358	0,883276
KLT JACOBI + ORDEM + HUFFMAN		1,490947	1,103977	0,936353	0,883416
DCT BLOCO+ZIGZAG+HUFFMAN		1,040697	0,743211	0,652260	0,618555
DCT BLOCO+ORDEM+HUFFMAN		1,040697	0,731950	0,646870	0,618555
DCT INTEIRA+ZIGZAG+HUFFMAN		1,261796	1,008753	0,874842	0,825391
DCT INTEIRA+ORDEM+HUFFMAN		1,313782	1,025430	0,880648	0,825391
DWT INTEIRA+ORDEM+HUFFMAN		1,287750	1,019342	0,882936	0,842629

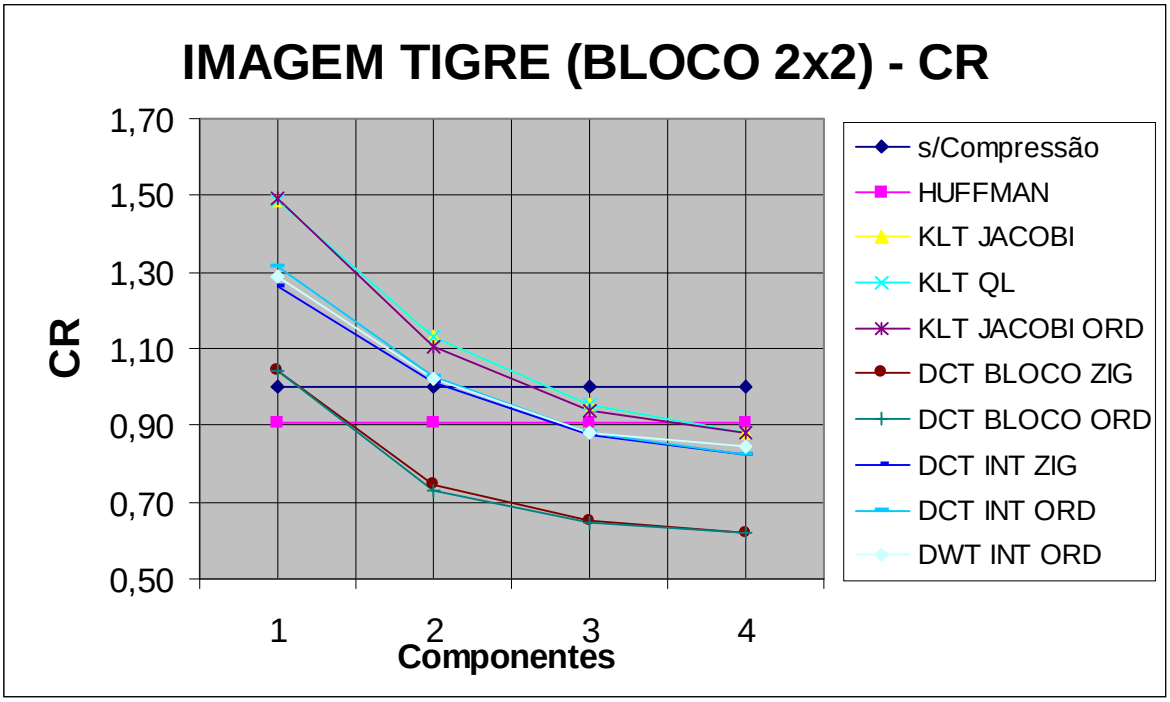


Figura 87 – CR da Imagem TIGRE (Bloco 2x2)

A Tabela 64 em conjunto com a Figura 88 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 64 – Tempo (seg) de Gravação da Imagem TIGRE (BLOCO 2x2)

Imagem:	**tigrag**	Bloco:	**2**	x	**2**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,11	0,17	0,16	0,22
KLT JACOBI + HUFFMAN		0,17	0,16	0,22	0,28
KLT QL + HUFFMAN		0,17	0,16	0,22	0,17
KLT JACOBI + ORDEM + HUFFMAN		0,33	0,38	0,39	0,39
DCT BLOCO+ZIGZAG+HUFFMAN		0,28	0,33	0,33	0,33
DCT BLOCO+ORDEM+HUFFMAN		0,72	0,77	0,76	0,77
DCT INTEIRA+ZIGZAG+HUFFMAN		0,22	0,27	0,28	0,27
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,28	0,33	0,28
DWT INTEIRA+ORDEM+HUFFMAN		0,17	0,21	0,27	0,22

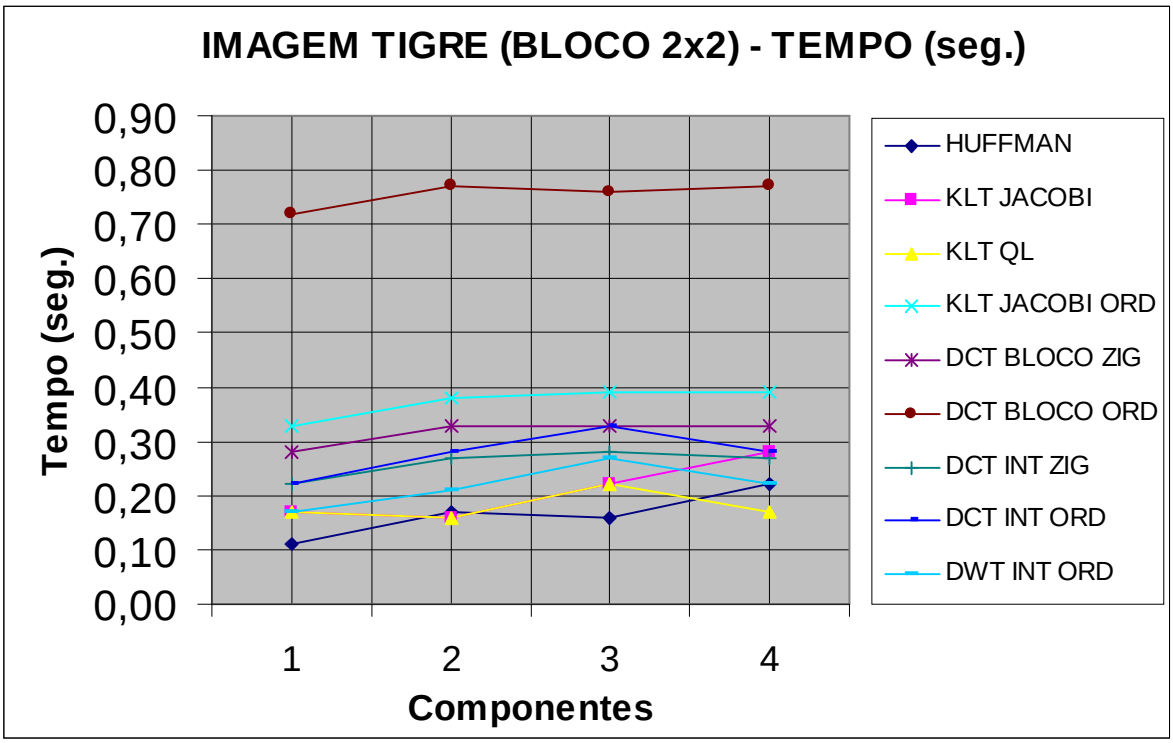


Figura 88 – Tempo (seg) de Gravação da Imagem TIGRE (Bloco 2x2)

A Tabela 65 em conjunto com a Figura 89 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 65 – PSNR da Imagem TIGRE (BLOCO 4x4)

Imagem:	**tigrag**	Bloco:	**4**	x	**4**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
KLT JACOBI + HUFFMAN		19,834923	25,729871	31,516787	58,925268
KLT QL + HUFFMAN		19,834923	25,729851	31,516859	58,926859
KLT JACOBI + ORDEM + HUFFMAN		19,473513	25,270848	31,125516	58,925268
DCT BLOCO+ZIGZAG+HUFFMAN		19,810595	24,350031	31,078551	61,295984
DCT BLOCO+ORDEM+HUFFMAN		19,810595	28,417022	36,305773	61,295984
DCT INTEIRA+ZIGZAG+HUFFMAN		21,430046	27,412991	33,930324	59,009630
DCT INTEIRA+ORDEM+HUFFMAN		23,822356	31,932583	40,144017	59,009630
DWT INTEIRA+ORDEM+HUFFMAN		25,270564	33,949342	42,847869	58,922881

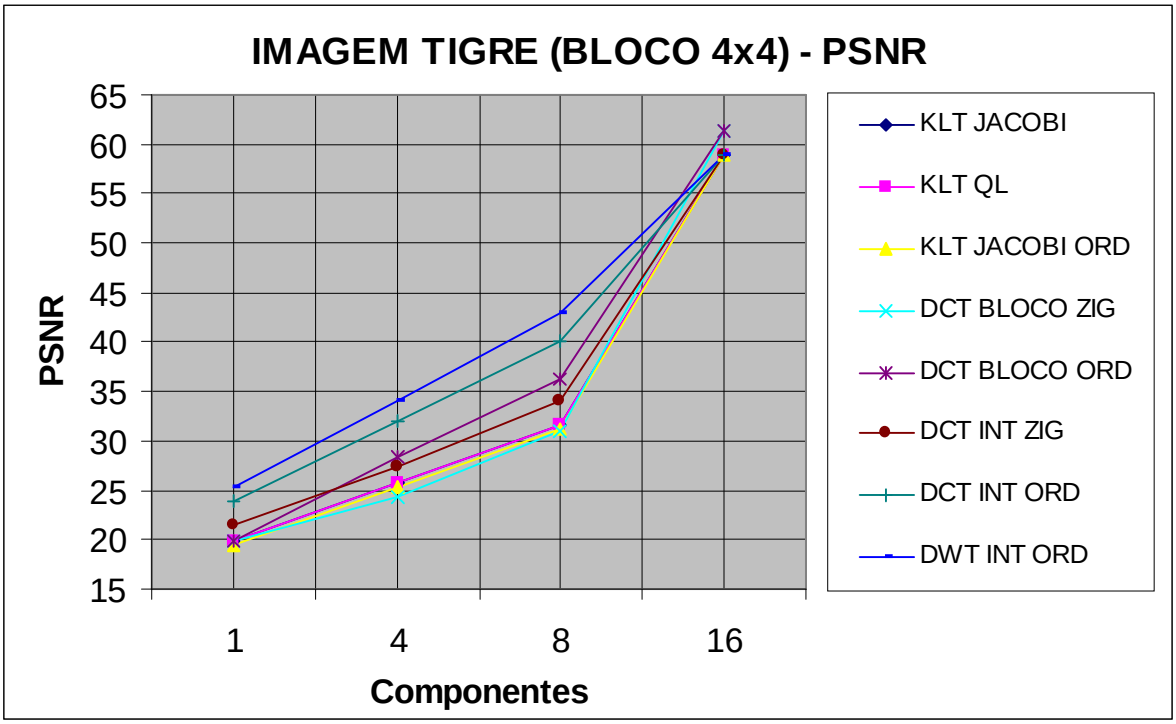


Figura 89 – PSNR da Imagem TIGRE (Bloco 4x4)

A Tabela 66 em conjunto com a Figura 90 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 66 – CR da Imagem TIGRE (BLOCO 4x4)

Imagem:	**tigrag**	Bloco:	**4**	x	**4**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		0,904664	0,904664	0,904664	0,904664
KLT JACOBI + HUFFMAN		1,531849	1,197598	0,972268	0,803304
KLT QL + HUFFMAN		1,531849	1,199064	0,973178	0,803963
KLT JACOBI + ORDEM + HUFFMAN		1,560412	1,186717	0,953126	0,803304
DCT BLOCO+ZIGZAG+HUFFMAN		1,014869	0,671343	0,588530	0,522164
DCT BLOCO+ORDEM+HUFFMAN		1,014869	0,662035	0,581264	0,522164
DCT INTEIRA+ZIGZAG+HUFFMAN		1,626199	1,261796	1,008753	0,825391
DCT INTEIRA+ORDEM+HUFFMAN		1,826442	1,313782	1,025430	0,825391
DWT INTEIRA+ORDEM+HUFFMAN		1,761436	1,287750	1,019342	0,842629

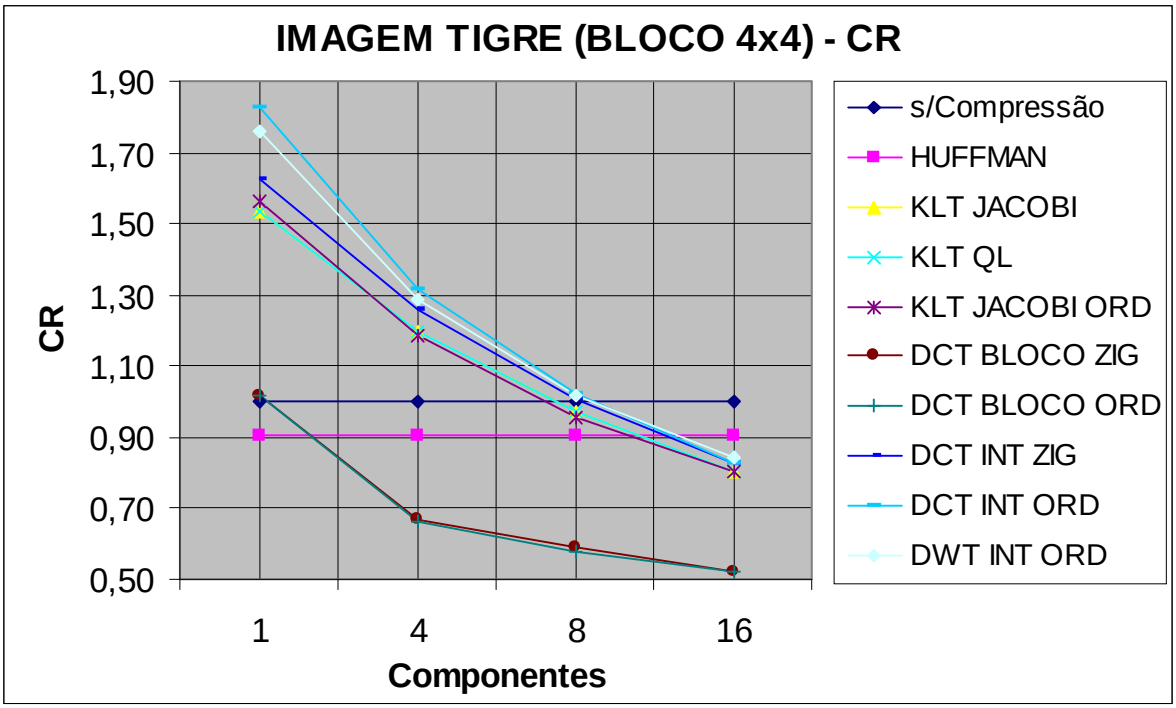


Figura 90 – CR da Imagem TIGRE (BLOCO 4x4)

A Tabela 67 em conjunto com a Figura 91 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 67 – Tempo (seg) de Gravação da Imagem TIGRE (BLOCO 4x4)

Imagem:	**tigrag**	Bloco:	**4**	x	**4**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,11	0,16	0,16	0,16
KLT JACOBI + HUFFMAN		0,22	0,17	0,28	0,28
KLT QL + HUFFMAN		0,22	0,17	0,27	0,28
KLT JACOBI + ORDEM + HUFFMAN		0,27	0,27	0,33	0,38
DCT BLOCO+ZIGZAG+HUFFMAN		0,27	0,28	0,33	0,33
DCT BLOCO+ORDEM+HUFFMAN		0,55	0,55	0,55	0,55
DCT INTEIRA+ZIGZAG+HUFFMAN		0,16	0,22	0,27	0,27
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,22	0,28	0,28
DWT INTEIRA+ORDEM+HUFFMAN		0,17	0,16	0,22	0,28

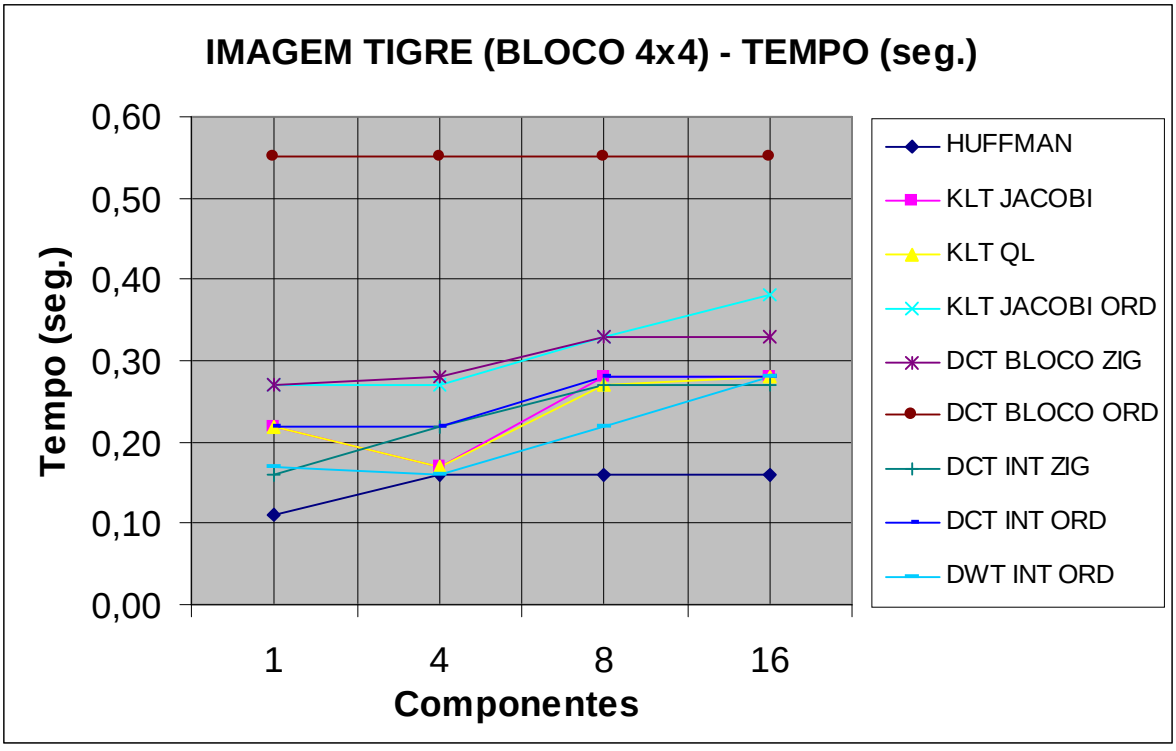


Figura 91 – Tempo (seg) de Gravação da Imagem TIGRE (BLOCO 4x4)

A Tabela 68 em conjunto com a Figura 92 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 68 – PSNR da Imagem TIGRE (BLOCO 8x8)

Imagem:	**tigrag**	Bloco:	**8**	x	**8**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
KLT JACOBI + HUFFMAN		17,744698	27,176473	34,197384	58,997478
KLT QL + HUFFMAN		17,744698	27,176688	34,197880	58,992626
KLT JACOBI + ORDEM + HUFFMAN		17,430270	26,806035	33,844574	58,997478
DCT BLOCO+ZIGZAG+HUFFMAN		17,723006	26,337804	32,646680	60,065179
DCT BLOCO+ORDEM+HUFFMAN		17,723006	30,776918	39,060283	60,065179
DCT INTEIRA+ZIGZAG+HUFFMAN		18,531327	27,412991	33,930324	59,009630
DCT INTEIRA+ORDEM+HUFFMAN		19,997379	31,932583	40,144017	59,009630
DWT INTEIRA+ORDEM+HUFFMAN		20,804364	33,949342	42,847869	58,922881
DWT BLOCO+ORDEM+HUFFMAN		6,471480	28,298770	35,944168	58,924472

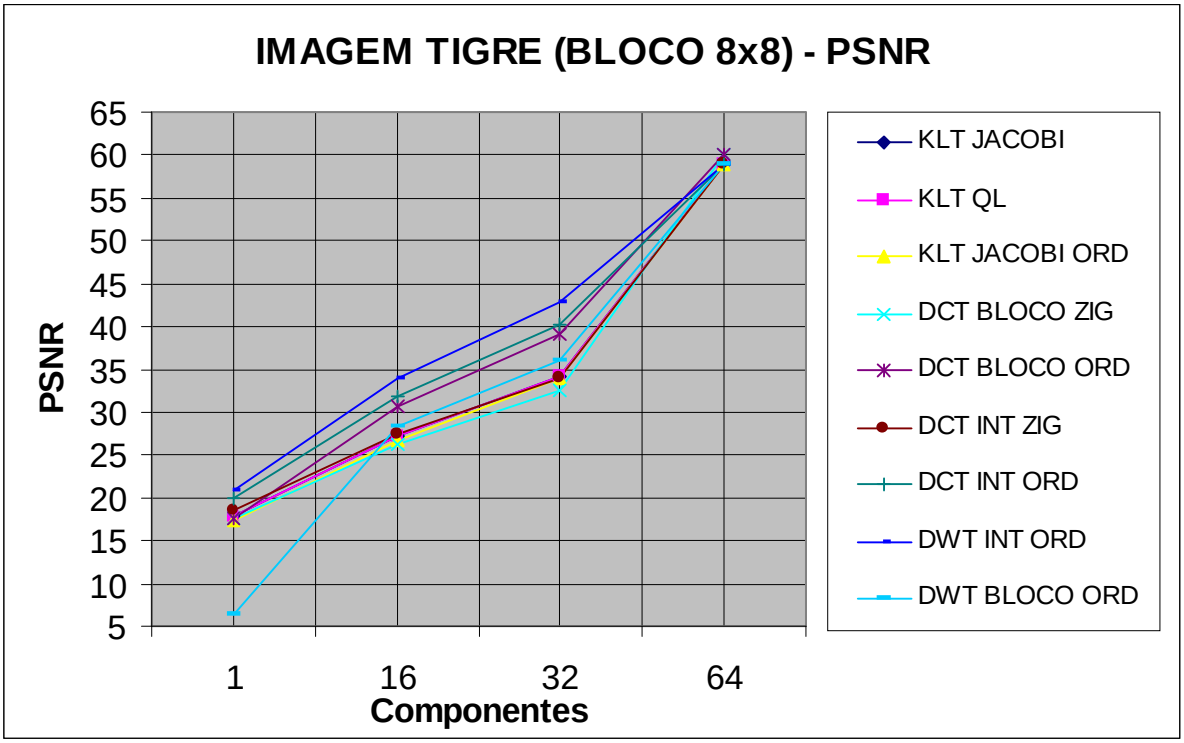


Figura 92 – PSNR da Imagem TIGRE (BLOCO 8x8)

A Tabela 69 em conjunto com a Figura 93 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 69 – CR da Imagem TIGRE (BLOCO 8x8)

Imagem:	**tigrag**	Bloco:	**8**	x	**8**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		0,904664	0,904664	0,904664	0,904664
KLT JACOBI + HUFFMAN		1,827745	1,023634	0,817139	0,648570
KLT QL + HUFFMAN		1,827745	1,027043	0,819290	0,649924
KLT JACOBI + ORDEM + HUFFMAN		1,903638	1,014622	0,804682	0,648570
DCT BLOCO+ZIGZAG+HUFFMAN		1,643289	0,800120	0,696064	0,611991
DCT BLOCO+ORDEM+HUFFMAN		1,643289	0,793846	0,685675	0,611991
DCT INTEIRA+ZIGZAG+HUFFMAN		2,053959	1,261796	1,008753	0,825391
DCT INTEIRA+ORDEM+HUFFMAN		2,408751	1,313782	1,025430	0,825391
DWT INTEIRA+ORDEM+HUFFMAN		2,334548	1,287750	1,019342	0,842629
DWT BLOCO+ORDEM+HUFFMAN		2,339468	0,946114	0,789789	0,681668

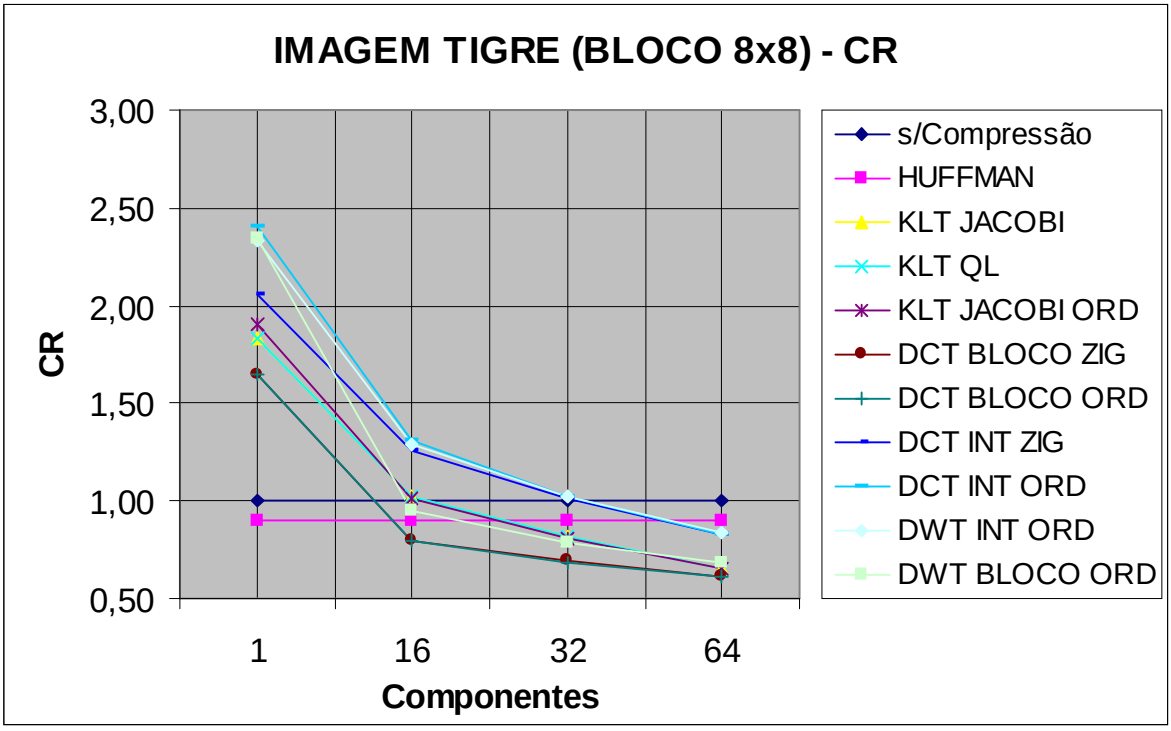


Figura 93 – CR da Imagem TIGRE (BLOCO 8x8)

A Tabela 70 em conjunto com a Figura 94 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 70 – Tempo de Gravação (seg) da Imagem TIGRE (BLOCO 8x8)

Imagem:	**tigrag**	Bloco:	**8**	x	**8**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,17	0,16	0,16	0,16
KLT JACOBI + HUFFMAN		0,60	0,61	0,60	0,66
KLT QL + HUFFMAN		0,49	0,50	0,60	0,60
KLT JACOBI + ORDEM + HUFFMAN		0,60	0,71	0,66	0,72
DCT BLOCO+ZIGZAG+HUFFMAN		0,22	0,27	0,28	0,28
DCT BLOCO+ORDEM+HUFFMAN		0,33	0,38	0,39	0,38
DCT INTEIRA+ZIGZAG+HUFFMAN		0,16	0,17	0,28	0,27
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,22	0,27	0,27
DWT INTEIRA+ORDEM+HUFFMAN		0,16	0,17	0,22	0,22
DWT BLOCO+ORDEM+HUFFMAN		0,22	0,27	0,33	0,33

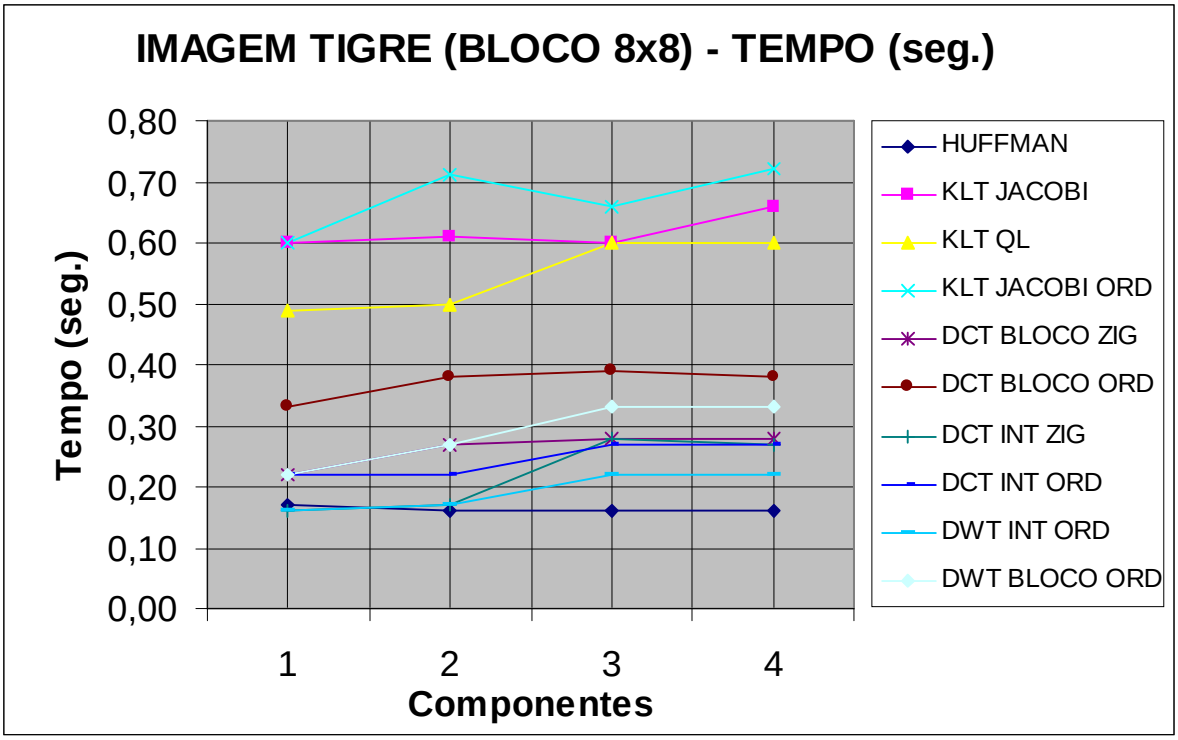


Figura 94 – Tempo de Gravação (seg) da Imagem TIGRE (BLOCO 8x8)

A Tabela 71 em conjunto com a Figura 95 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 71 – PSNR da Imagem TIGRE (BLOCO 16x16)

Imagem:	**tigrag**	Bloco:	**16**	x	**16**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
KLT JACOBI + HUFFMAN		16,374891	22,098073	42,594619	59,851217
KLT QL + HUFFMAN		16,374892	22,097984	42,592082	59,807126
KLT JACOBI + ORDEM + HUFFMAN		16,165481	21,766571	42,352242	59,851217
DCT BLOCO+ZIGZAG+HUFFMAN		16,330537	21,016101	33,352034	59,448047
DCT BLOCO+ORDEM+HUFFMAN		16,330537	23,580649	40,083508	59,448047
DCT INTEIRA+ZIGZAG+HUFFMAN		17,063288	21,430046	33,930324	59,009630
DCT INTEIRA+ORDEM+HUFFMAN		17,820811	23,822356	40,144017	59,009630
DWT INTEIRA+ORDEM+HUFFMAN		18,035561	25,270564	42,847869	58,922881
DWT BLOCO+ORDEM+HUFFMAN		6,408214	22,342964	37,564991	58,977299

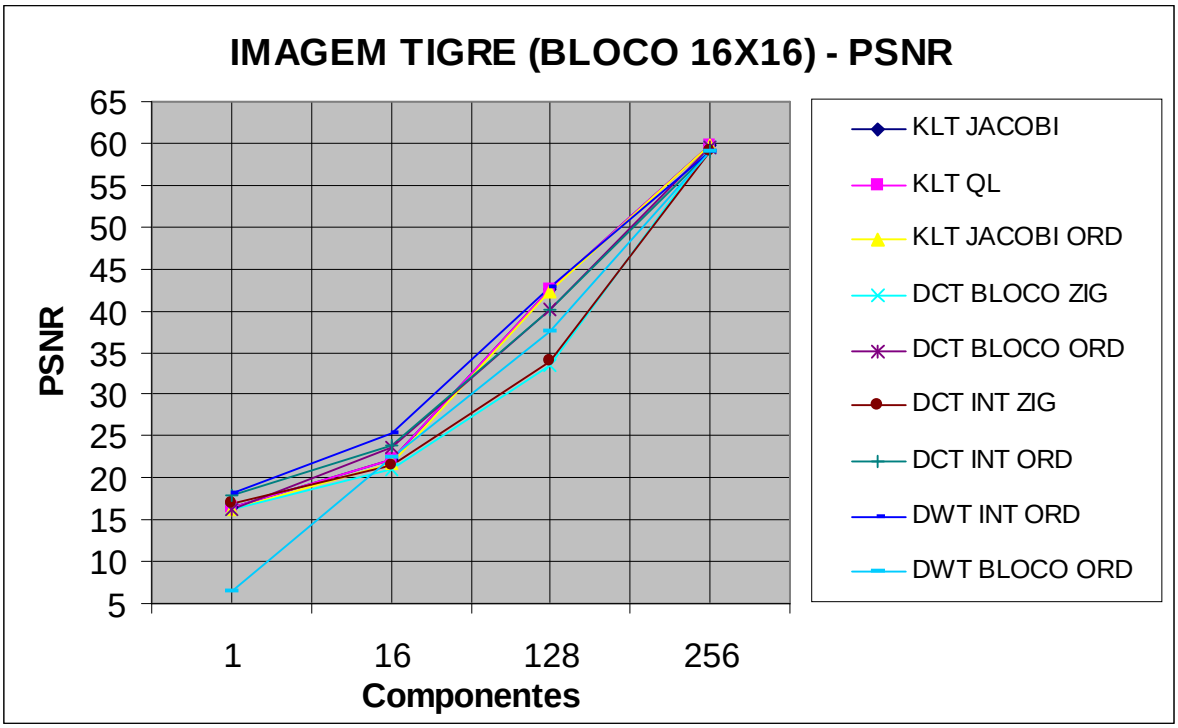


Figura 95 – PSNR da Imagem TIGRE (BLOCO 16x16)

A Tabela 72 em conjunto com a Figura 96 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 72 – CR da Imagem TIGRE (BLOCO 16x16)

Imagem:	**tigrag**	Bloco:	**16**	x	**16**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		0,904664	0,904664	0,904664	0,904664
KLT JACOBI + HUFFMAN		3,246613	1,070294	0,332253	0,195962
KLT QL + HUFFMAN		3,246613	1,075252	0,332545	0,196063
KLT JACOBI + ORDEM + HUFFMAN		3,293158	1,067019	0,331487	0,195962
DCT BLOCO+ZIGZAG+HUFFMAN		3,600562	1,314949	0,882585	0,751207
DCT BLOCO+ORDEM+HUFFMAN		3,600562	1,281236	0,864791	0,751207
DCT INTEIRA+ZIGZAG+HUFFMAN		3,672824	1,626199	1,008753	0,825391
DCT INTEIRA+ORDEM+HUFFMAN		3,809780	1,826442	1,025430	0,825391
DWT INTEIRA+ORDEM+HUFFMAN		3,762865	1,761436	1,019342	0,842629
DWT BLOCO+ORDEM+HUFFMAN		3,762865	1,081958	0,762933	0,665049

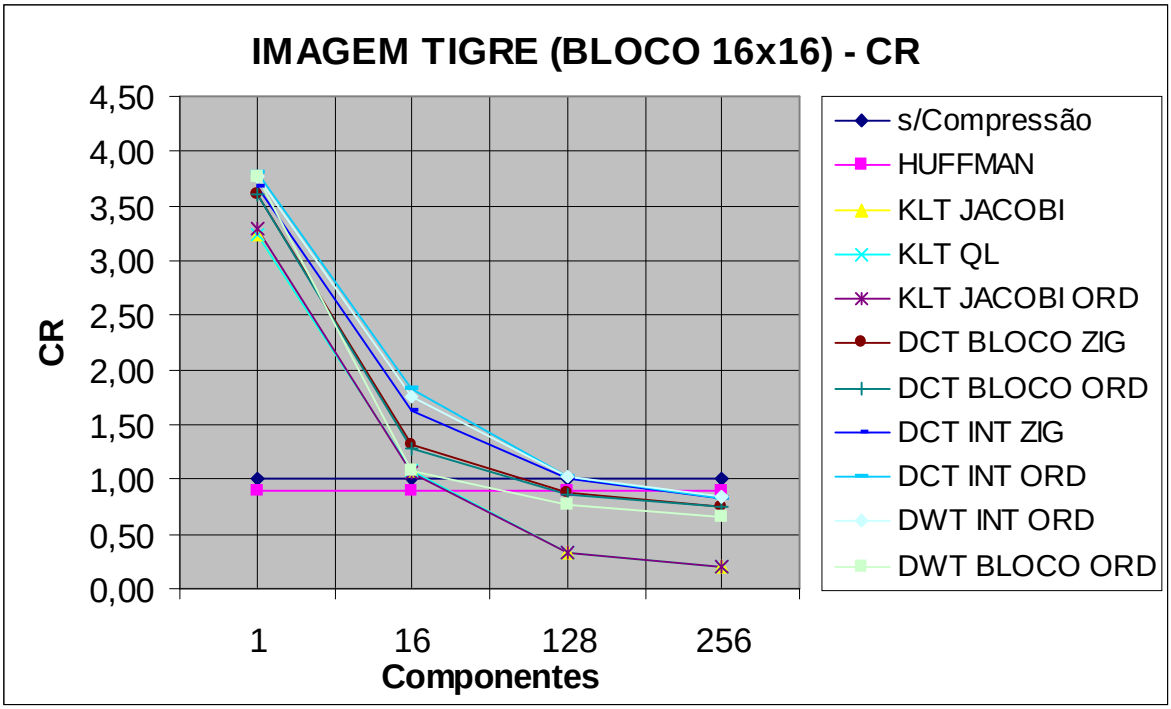


Figura 96 – CR da Imagem TIGRE (BLOCO 16x16)

A Tabela 73 em conjunto com a Figura 97 apresentam os resultados obtidos para a imagem TIGRE (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 73 – Tempo de Gravação (seg) da Imagem TIGRE (BLOCO 16x16)

Imagem:	**tigrag**	Bloco:	**16**	x	**16**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,16	0,11	0,17	0,11
KLT JACOBI + HUFFMAN		26,80	26,37	31,03	30,76
KLT QL + HUFFMAN		6,70	6,48	7,47	8,07
KLT JACOBI + ORDEM + HUFFMAN		26,97	27,30	28,46	29,82
DCT BLOCO+ZIGZAG+HUFFMAN		0,11	0,22	0,22	0,28
DCT BLOCO+ORDEM+HUFFMAN		0,22	0,27	0,33	0,33
DCT INTEIRA+ZIGZAG+HUFFMAN		0,16	0,22	0,27	0,28
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,28	0,33	0,33
DWT INTEIRA+ORDEM+HUFFMAN		0,17	0,17	0,27	0,27
DWT BLOCO+ORDEM+HUFFMAN		0,17	0,22	0,33	0,38

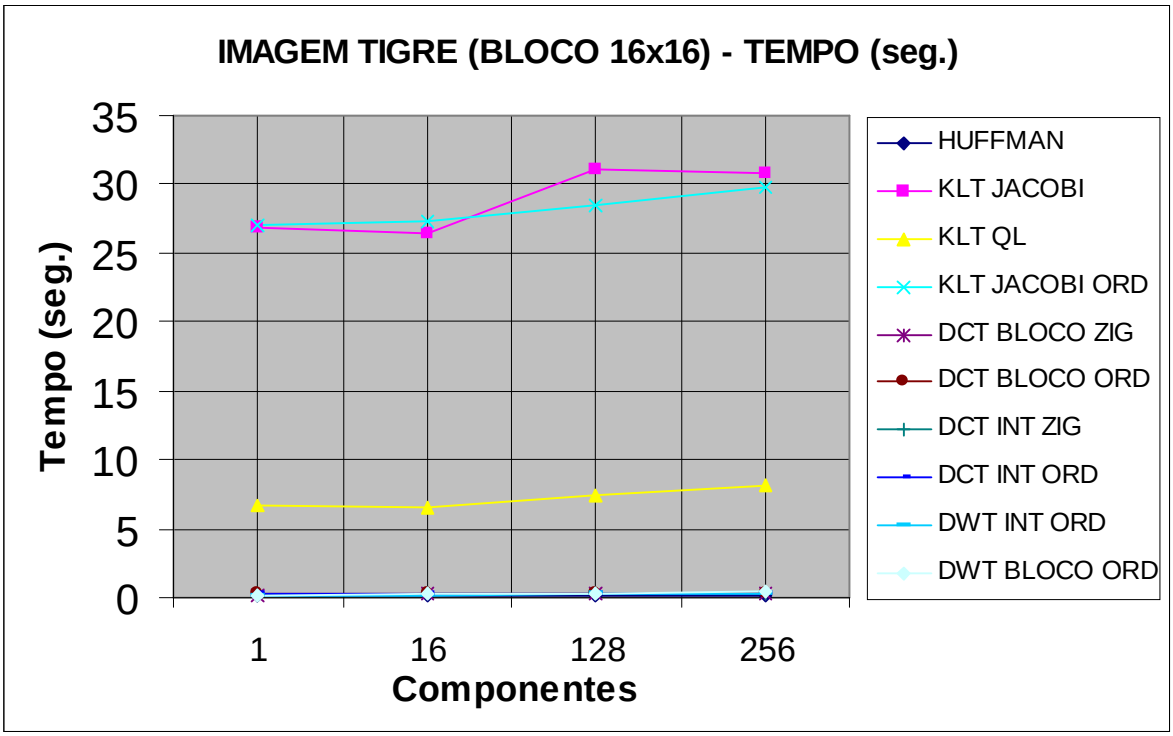


Figura 97 – Tempo de Gravação (seg) da Imagem TIGRE (BLOCO 16x16)

A Tabela 74 em conjunto com a Figura 98 apresentam os resultados da *PSNR (Razão Sinal/Ruído de Pico)* obtida para a imagem TIGRE (256X256 PIXELS) para 4 tamanhos de bloco de imagem, para 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 74 – PSNR da Imagem TIGRE – Compressão QL

Imagem:	tigrag			
RAZÃO SINAL/RUÍDO DE PICO (PSNR)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	24,087323	25,729851	27,176688	31,335009
4:2	28,499992	31,516859	34,197880	42,592082
4:3	35,033915	38,805765	41,948317	54,675638
4:4	58,705727	58,926859	58,992626	59,807126

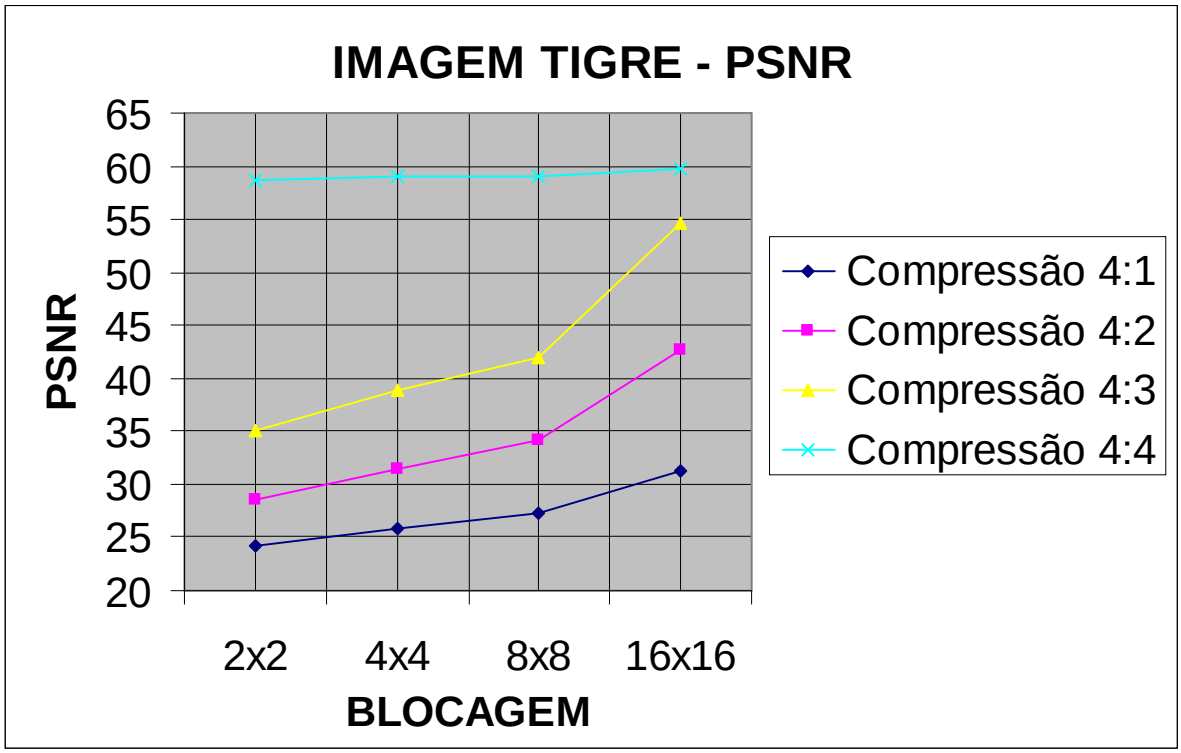


Figura 98 – PSNR da Imagem TIGRE – Compressão QL

A Tabela 75 em conjunto com a Figura 99 apresentam os resultados da *CR (Razão de Compressão)* obtida para a imagem TIGRE (256X256 PIXELS) para 4 tamanhos de bloco de imagem, para 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 75 – CR da Imagem TIGRE – Compressão QL

Imagem:	tigrag			
RAZÃO DE COMPRESSÃO (CR)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	1,486920	1,199064	1,027043	0,541771
4:2	1,130699	0,973178	0,819290	0,332545
4:3	0,953358	0,851863	0,705134	0,243880
4:4	0,883276	0,803963	0,649924	0,196063

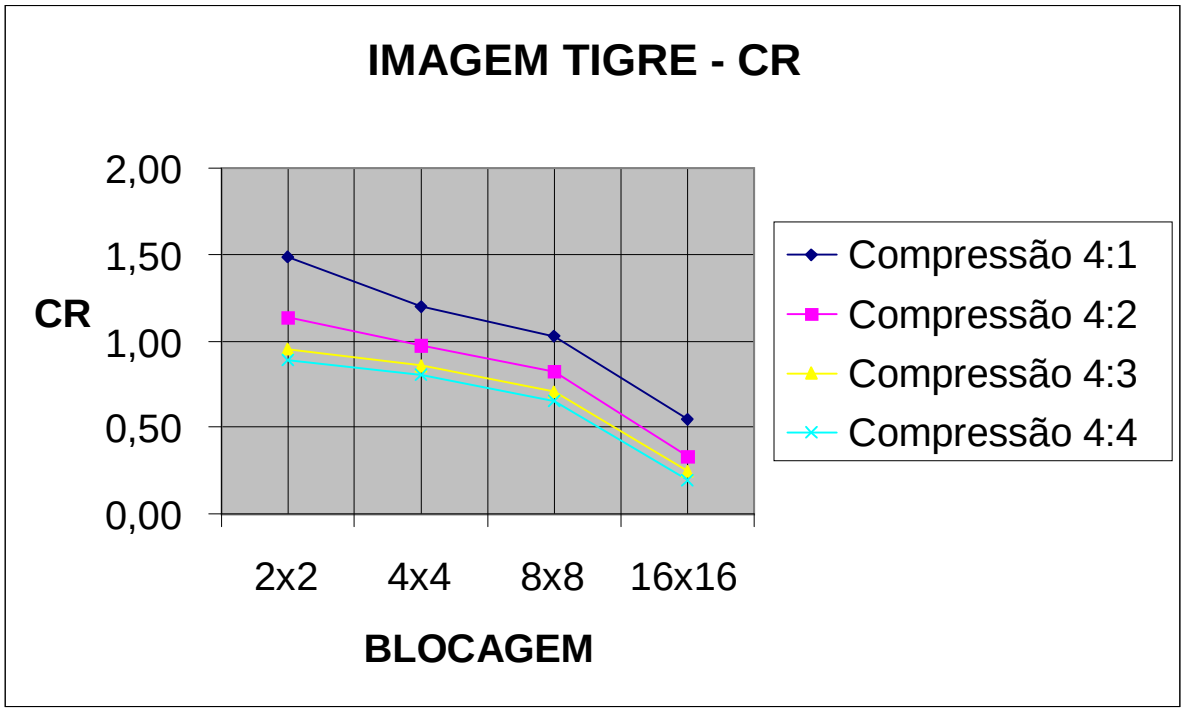


Figura 99 – CR da Imagem TIGRE – Compressão QL

A Tabela 76 em conjunto com a Figura 100 apresentam os resultados do *Tempo (em segundos)* obtida para a imagem TIGRE (256X256 PIXELS) para 4 tamanhos de bloco de imagem, para 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 76 – Tempo de Gravação da Imagem TIGRE – Compressão QL

Imagem:	tigrag			
TEMPO DE GRAVAÇÃO (SEGUNDOS)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	0,17	0,17	0,500000	7,360000
4:2	0,16	0,27	0,600000	7,470000
4:3	0,22	0,33	0,610000	7,300000
4:4	0,17	0,28	0,600000	8,070000

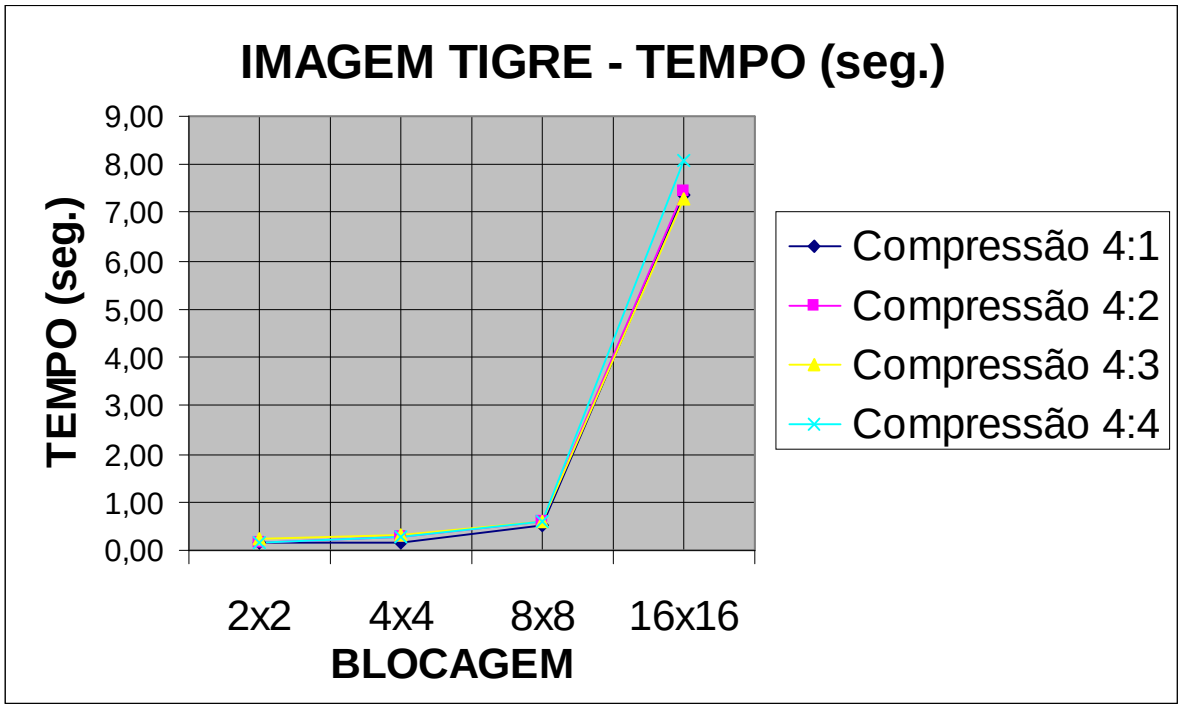


Figura 100 – Tempo de Gravação da Imagem TIGRE – Compressão QL

5.6 RESULTADOS OBTIDOS COM A IMAGEM VENTILADOR

As tabelas e gráficos apresentados abaixo foram obtidos utilizando-se a imagem padrão VENTILADOR (Figura 101), de dimensões 256 x 256 (65.536 PIXELS) com 256 níveis de escala de cinza.

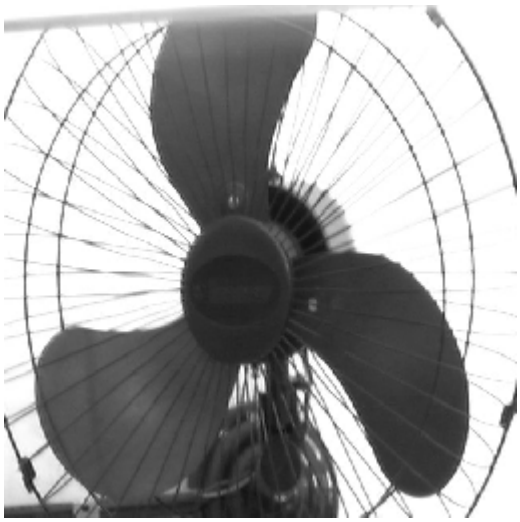


Figura 101 – Figura VENTILADOR (256 x 256)

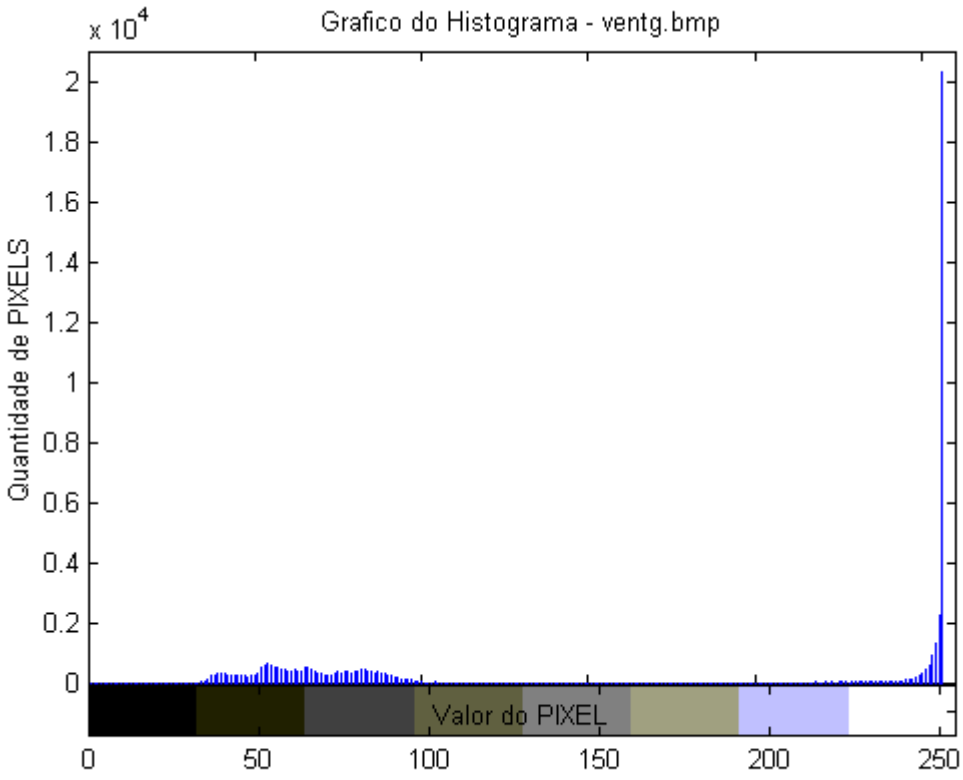


Figura 102 – Histograma da Imagem VENTILADOR (256x256)

Como se pode observar no Histograma da Figura 102, existe uma concentração de PIXELS em valores mais claros da escala de cinza desta imagem. A freqüência máxima de valores da escala de cinza aparece em 31 % dos pontos da imagem (20.380 PIXELS). O valor médio do PIXEL para a esta imagem é igual a 163.

A Tabela 77 em conjunto com a Figura 103 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 77 – PSNR da Imagem VENTILADOR (BLOCO 2x2)

Imagem:	**ventg**	Bloco:	**2**	x	**2**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
KLT JACOBI + HUFFMAN		25,213930	29,435288	34,127504	59,934738
KLT QL + HUFFMAN		25,213930	29,435316	34,127517	59,932730
KLT JACOBI + ORDEM + HUFFMAN		24,747794	28,913713	33,197215	59,934738
DCT BLOCO+ZIGZAG+HUFFMAN		25,211868	29,297266	34,123012	66,396657
DCT BLOCO+ORDEM+HUFFMAN		25,211868	31,935886	37,631678	66,396657
DCT INTEIRA+ZIGZAG+HUFFMAN		26,424731	30,492206	35,052423	59,611744
DCT INTEIRA+ORDEM+HUFFMAN		30,908711	37,809020	46,920223	59,611744
DWT INTEIRA+ORDEM+HUFFMAN		39,539680	49,915305	57,536313	60,002527

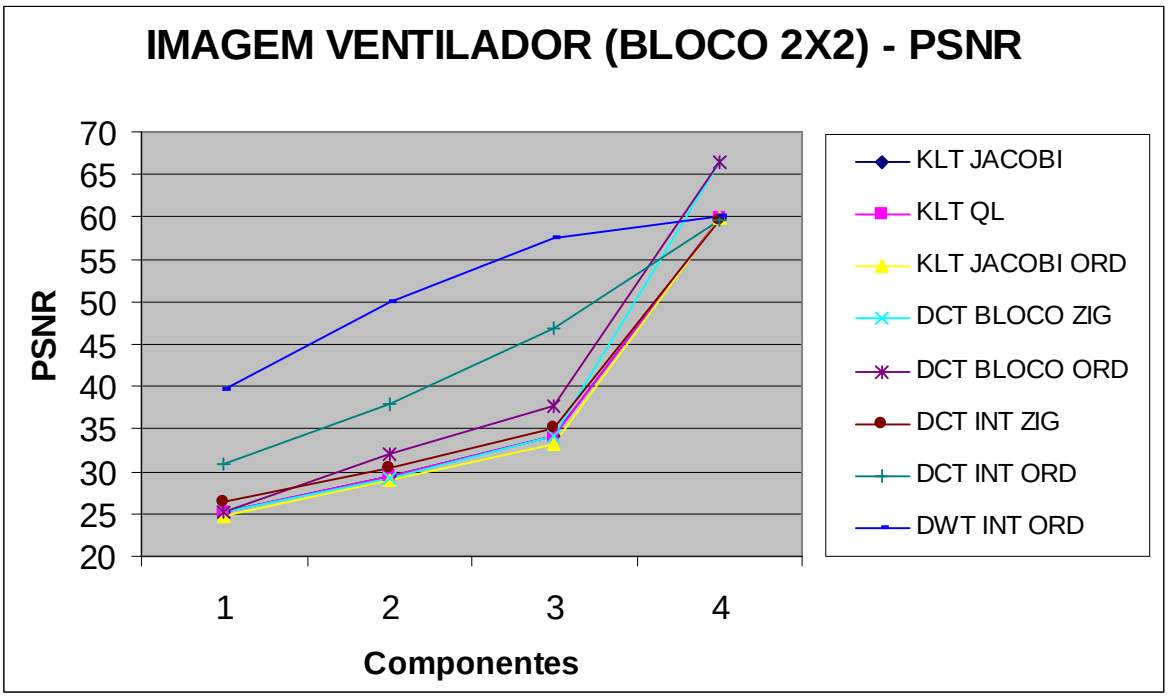


Figura 103 – PSNR da Imagem VENTILADOR (Bloco 2x2)

A Tabela 78 em conjunto com a Figura 104 apresentam os resultados obtidos para a imagem VENTILADOR (512x512 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 78 – CR da Imagem VENTILADOR (BLOCO 2x2)

Imagem:	**ventg**	Bloco:	**2**	x	**2**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,156232	1,156232	1,156232	1,156232
KLT JACOBI + HUFFMAN		1,633176	1,310345	1,100367	1,023194
KLT QL + HUFFMAN		1,633176	1,310345	1,100294	1,023115
KLT JACOBI + ORDEM + HUFFMAN		1,675276	1,222096	1,077095	1,023194
DCT BLOCO+ZIGZAG+HUFFMAN		1,132448	0,829481	0,737860	0,698223
DCT BLOCO+ORDEM+HUFFMAN		1,132448	0,809523	0,731162	0,698223
DCT INTEIRA+ZIGZAG+HUFFMAN		1,330311	1,056912	0,905943	0,840417
DCT INTEIRA+ORDEM+HUFFMAN		1,389181	1,081782	0,911646	0,840417
DWT INTEIRA+ORDEM+HUFFMAN		1,312203	1,056443	0,934841	0,925220

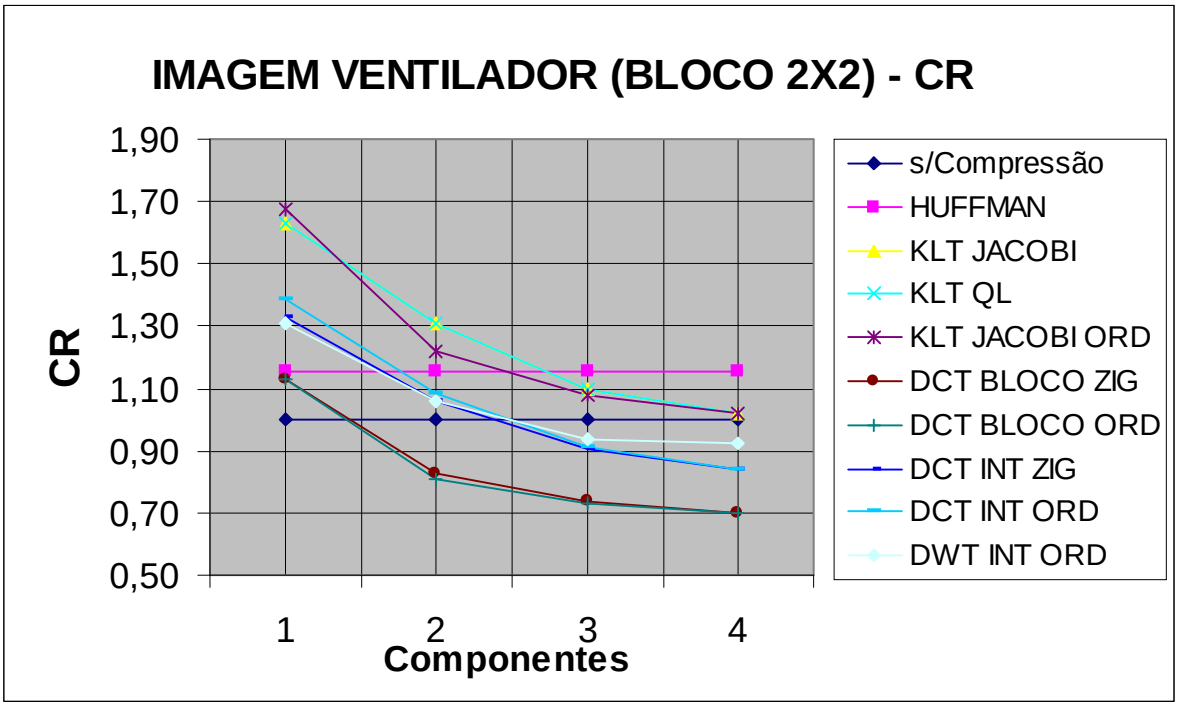


Figura 104 – CR da Imagem VENTILADOR (Bloco 2x2)

A Tabela 79 em conjunto com a Figura 105 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 2x2 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 79 – Tempo (seg) de Gravação da Imagem VENTILADOR (BLOCO 2x2)

Imagem:	**ventg**	Bloco:	**2**	x	**2**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**2**	**3**	**4**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,06	0,11	0,05	0,22
KLT JACOBI + HUFFMAN		0,11	0,11	0,16	0,22
KLT QL + HUFFMAN		0,17	0,11	0,16	0,21
KLT JACOBI + ORDEM + HUFFMAN		0,39	0,39	0,39	0,44
DCT BLOCO+ZIGZAG+HUFFMAN		0,22	0,28	0,33	0,27
DCT BLOCO+ORDEM+HUFFMAN		0,72	0,77	0,82	0,83
DCT INTEIRA+ZIGZAG+HUFFMAN		0,22	0,27	0,33	0,27
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,28	0,33	0,28
DWT INTEIRA+ORDEM+HUFFMAN		0,16	0,28	0,22	0,22

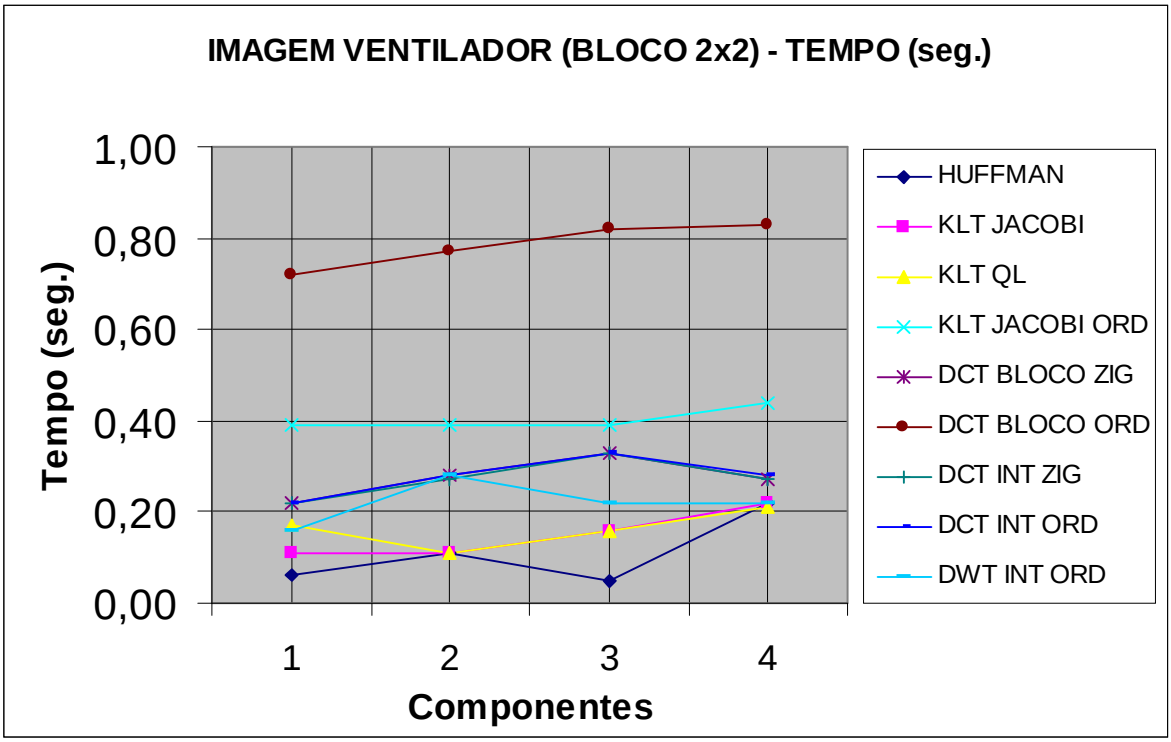


Figura 105 – Tempo (seg) de Gravação da Imagem VENTILADOR (Bloco 2x2)

A Tabela 80 em conjunto com a Figura 106 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 80 – PSNR da Imagem VENTILADOR (BLOCO 4x4)

Imagem:	**ventg**	Bloco:	**4**	x	**4**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
KLT JACOBI + HUFFMAN		21,168433	26,166410	31,047008	59,540569
KLT QL + HUFFMAN		21,168433	26,166436	31,047162	59,535987
KLT JACOBI + ORDEM + HUFFMAN		20,837104	25,703544	30,617298	59,540569
DCT BLOCO+ZIGZAG+HUFFMAN		21,160798	24,925778	30,115347	62,412814
DCT BLOCO+ORDEM+HUFFMAN		21,160798	28,782358	35,934910	62,412814
DCT INTEIRA+ZIGZAG+HUFFMAN		22,565133	26,424731	30,492206	59,611744
DCT INTEIRA+ORDEM+HUFFMAN		24,758385	30,908711	37,809020	59,611744
DWT INTEIRA+ORDEM+HUFFMAN		28,148748	39,539680	49,915305	60,002527

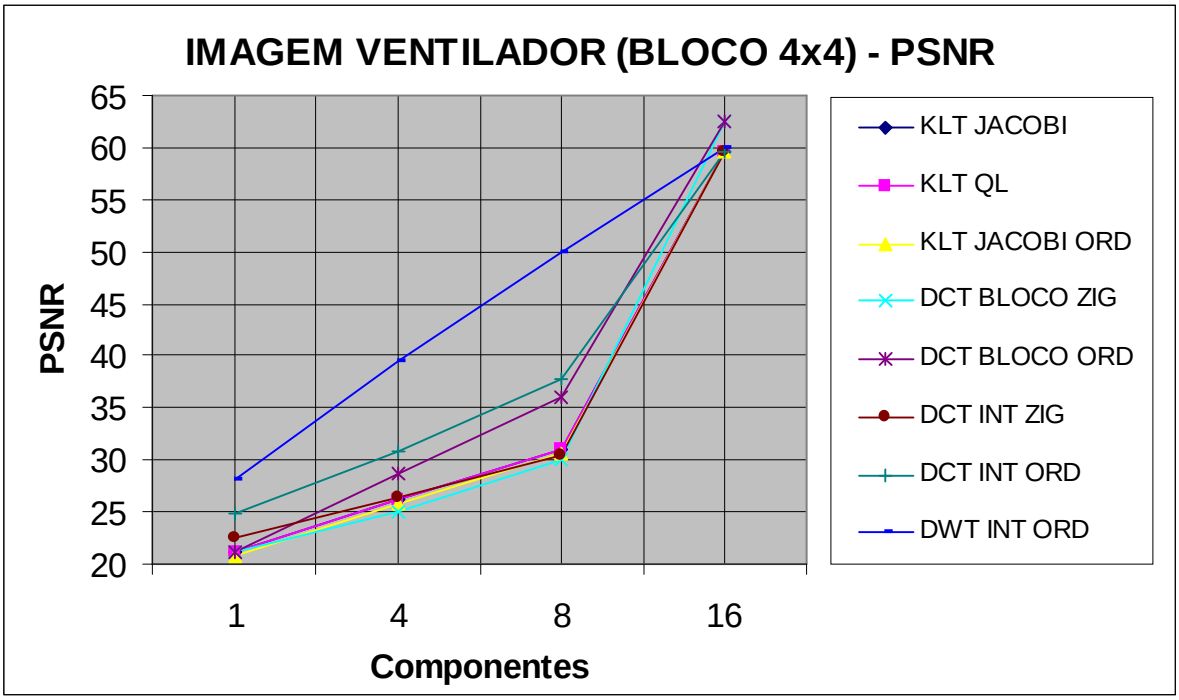


Figura 106 – PSNR da Imagem VENTILADOR (Bloco 4x4)

A Tabela 81 em conjunto com a Figura 107 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 81 – CR da Imagem VENTILADOR (BLOCO 4x4)

Imagem:	**ventg**	Bloco:	**4**	x	**4**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,156232	1,156232	1,156232	1,156232
KLT JACOBI + HUFFMAN		1,633577	1,194013	1,014993	0,843600
KLT QL + HUFFMAN		1,633577	1,194463	1,015070	0,843269
KLT JACOBI + ORDEM + HUFFMAN		1,762835	1,167213	0,967341	0,843600
DCT BLOCO+ZIGZAG+HUFFMAN		1,180409	0,788210	0,699903	0,614407
DCT BLOCO+ORDEM+HUFFMAN		1,180409	0,767973	0,682234	0,614407
DCT INTEIRA+ZIGZAG+HUFFMAN		1,710068	1,330311	1,056912	0,840417
DCT INTEIRA+ORDEM+HUFFMAN		1,891745	1,389181	1,081782	0,840417
DWT INTEIRA+ORDEM+HUFFMAN		1,779173	1,312203	1,056443	0,925220

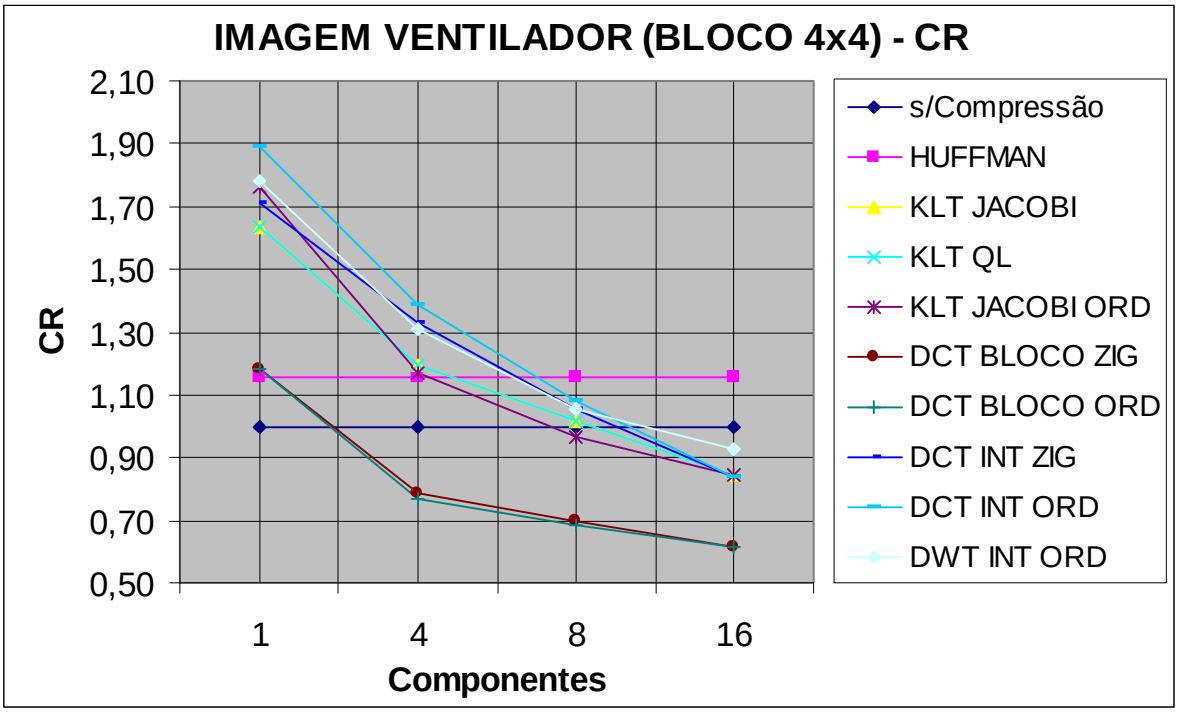


Figura 107 – CR da Imagem VENTILADOR (BLOCO 4x4)

A Tabela 82 em conjunto com a Figura 108 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 4x4 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 82 – Tempo (seg) de Gravação da Imagem VENTILADOR (BLOCO 4x4)

Imagem:	**ventg**	Bloco:	**4**	x	**4**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**4**	**8**	**16**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,11	0,06	0,11	0,05
KLT JACOBI + HUFFMAN		0,22	0,17	0,22	0,27
KLT QL + HUFFMAN		0,21	0,16	0,27	0,22
KLT JACOBI + ORDEM + HUFFMAN		0,28	0,33	0,33	0,39
DCT BLOCO+ZIGZAG+HUFFMAN		0,22	0,27	0,27	0,27
DCT BLOCO+ORDEM+HUFFMAN		0,50	0,55	0,49	0,55
DCT INTEIRA+ZIGZAG+HUFFMAN		0,22	0,21	0,28	0,28
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,22	0,27	0,27
DWT INTEIRA+ORDEM+HUFFMAN		0,16	0,17	0,22	0,27

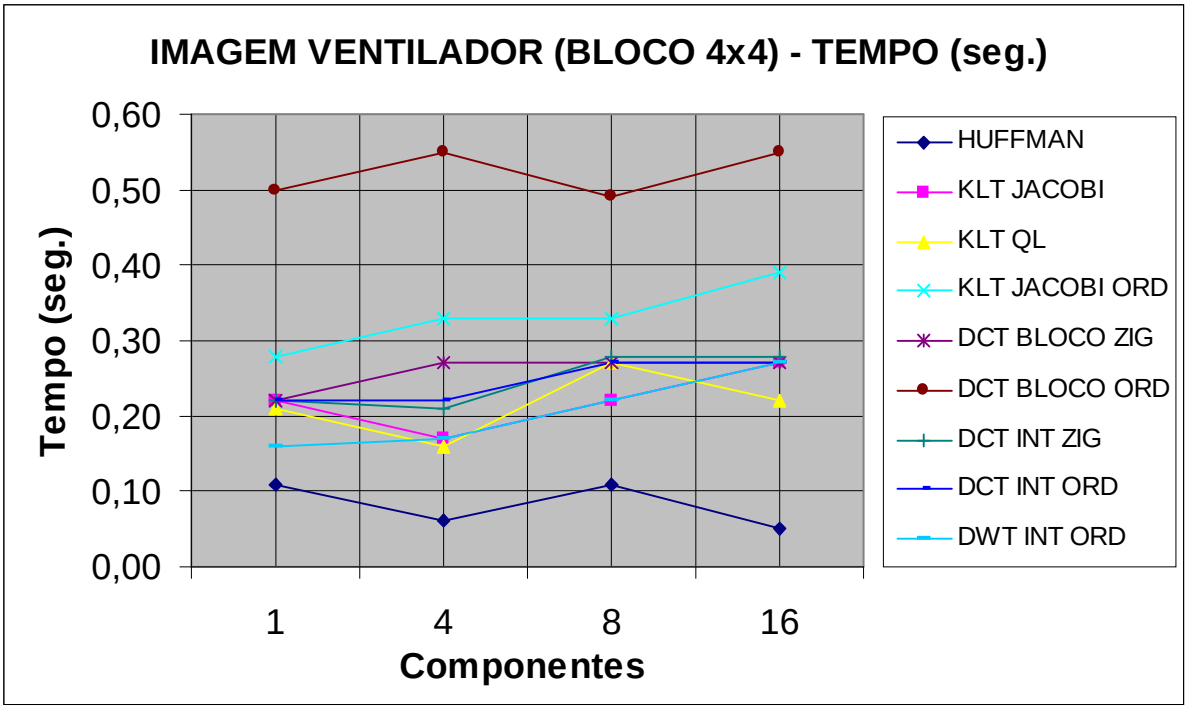


Figura 108 – Tempo (seg) de Gravação da Imagem VENTILADOR (BLOCO 4x4)

A Tabela 83 em conjunto com a Figura 109 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 83 – PSNR da Imagem VENTILADOR (BLOCO 8x8)

Imagem:	**ventg**	Bloco:	**8**	x	**8**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
KLT JACOBI + HUFFMAN		18,947269	27,429979	32,618761	59,698342
KLT QL + HUFFMAN		18,947280	27,430039	32,618846	59,705002
KLT JACOBI + ORDEM + HUFFMAN		18,780065	27,070465	32,291501	59,698342
DCT BLOCO+ZIGZAG+HUFFMAN		18,928095	26,062512	30,477169	60,904842
DCT BLOCO+ORDEM+HUFFMAN		18,928095	31,407609	38,748948	60,904842
DCT INTEIRA+ZIGZAG+HUFFMAN		20,313658	26,424731	30,492206	59,611744
DCT INTEIRA+ORDEM+HUFFMAN		21,547121	30,908711	37,809020	59,611744
DWT INTEIRA+ORDEM+HUFFMAN		22,608663	39,539680	49,915305	60,002527
DWT BLOCO+ORDEM+HUFFMAN		4,743527	30,299082	38,778125	59,955871

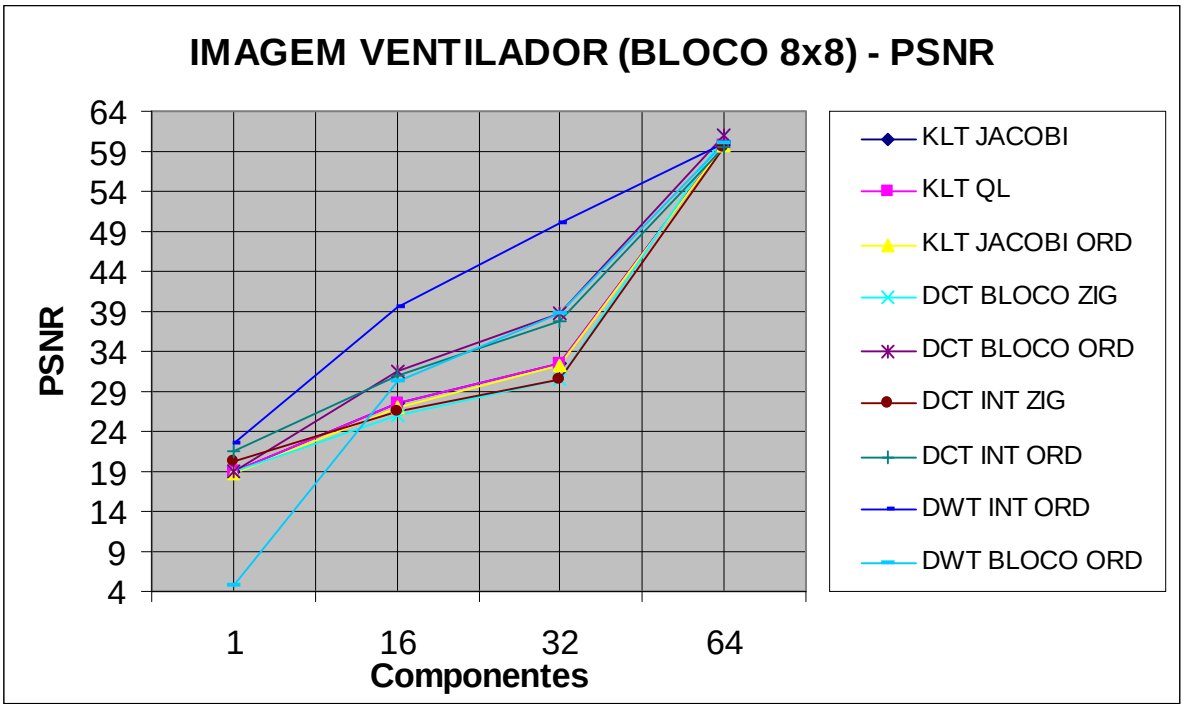


Figura 109 – PSNR da Imagem VENTILADOR (BLOCO 8x8)

A Tabela 84 em conjunto com a Figura 110 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 84 – CR da Imagem VENTILADOR (BLOCO 8x8)

Imagem:	**ventg**	Bloco:	**8**	x	**8**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,156232	1,156232	1,156232	1,156232
KLT JACOBI + HUFFMAN		2,061651	0,995398	0,813725	0,644404
KLT QL + HUFFMAN		2,061651	0,998067	0,815199	0,645341
KLT JACOBI + ORDEM + HUFFMAN		2,087428	0,974231	0,786869	0,644404
DCT BLOCO+ZIGZAG+HUFFMAN		1,786521	0,900603	0,790408	0,680165
DCT BLOCO+ORDEM+HUFFMAN		1,786521	0,879834	0,764089	0,680165
DCT INTEIRA+ZIGZAG+HUFFMAN		2,065422	1,330311	1,056912	0,840417
DCT INTEIRA+ORDEM+HUFFMAN		2,393174	1,389181	1,081782	0,840417
DWT INTEIRA+ORDEM+HUFFMAN		2,320398	1,312203	1,056443	0,925220
DWT BLOCO+ORDEM+HUFFMAN		2,621257	1,000751	0,850167	0,752837

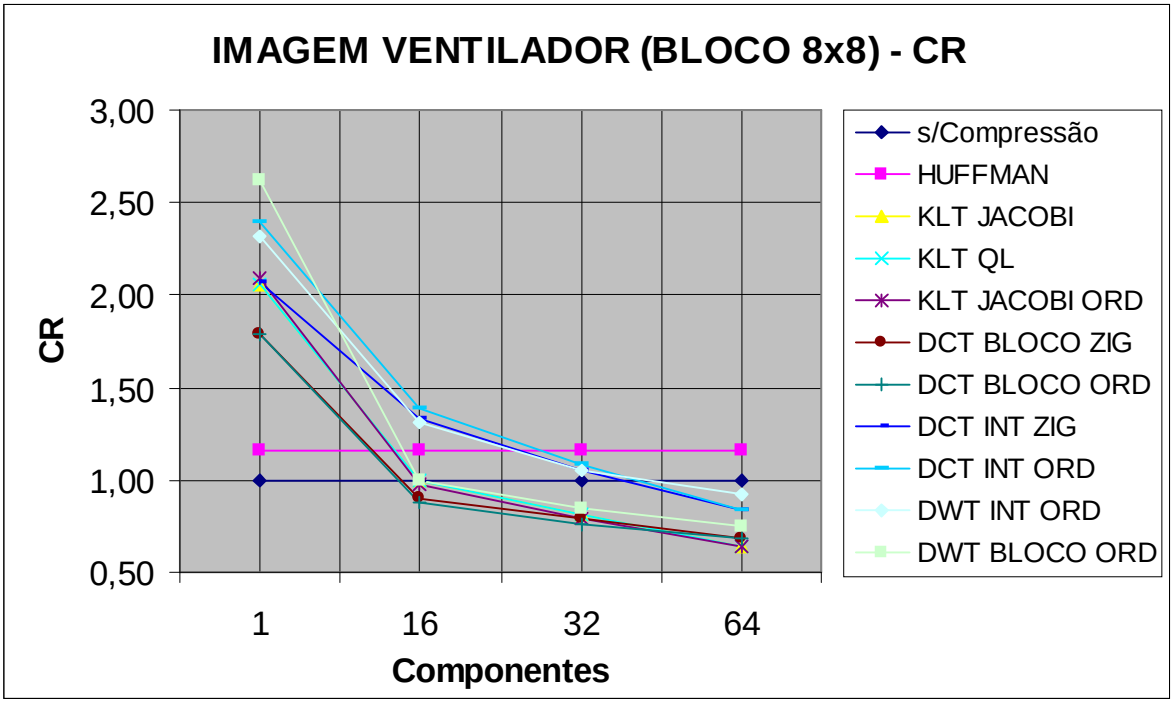


Figura 110 – CR da Imagem VENTILADOR (BLOCO 8x8)

A Tabela 85 em conjunto com a Figura 111 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 8x8 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 85 – Tempo de Gravação (seg) da Imagem VENTILADOR (BLOCO 8x8)

Imagem:	**ventg**	Bloco:	**8**	x	**8**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**32**	**64**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,11	0,05	0,11	0,05
KLT JACOBI + HUFFMAN		0,49	0,60	0,60	0,66
KLT QL + HUFFMAN		0,43	0,50	0,55	0,61
KLT JACOBI + ORDEM + HUFFMAN		0,55	0,66	0,66	0,65
DCT BLOCO+ZIGZAG+HUFFMAN		0,16	0,27	0,28	0,28
DCT BLOCO+ORDEM+HUFFMAN		0,33	0,39	0,44	0,38
DCT INTEIRA+ZIGZAG+HUFFMAN		0,22	0,22	0,27	0,28
DCT INTEIRA+ORDEM+HUFFMAN		0,17	0,22	0,28	0,27
DWT INTEIRA+ORDEM+HUFFMAN		0,16	0,17	0,22	0,22
DWT BLOCO+ORDEM+HUFFMAN		0,22	0,27	0,33	0,33

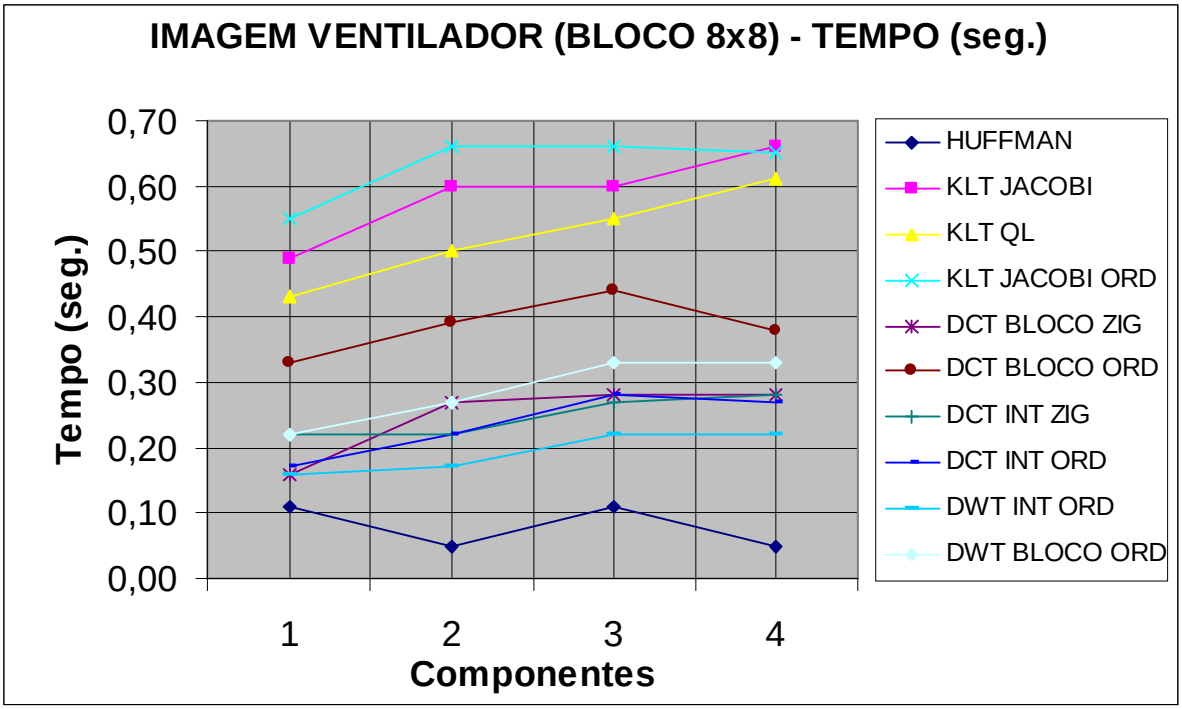


Figura 111 – Tempo de Gravação (seg) da Imagem VENTILADOR (BLOCO 8x8)

A Tabela 86 em conjunto com a Figura 112 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 86 – PSNR da Imagem VENTILADOR (BLOCO 16x16)

Imagem:	**ventg**	Bloco:	**16**	x	**16**
RAZÃO SINAL/RUÍDO DE PICO (PSNR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
KLT JACOBI + HUFFMAN		16,536933	23,501039	42,011760	60,641848
KLT QL + HUFFMAN		16,536932	23,501017	42,010691	60,651310
KLT JACOBI + ORDEM + HUFFMAN		16,194771	23,152097	41,825496	60,641848
DCT BLOCO+ZIGZAG+HUFFMAN		16,496248	22,200020	30,510420	60,240398
DCT BLOCO+ORDEM+HUFFMAN		16,496248	25,114989	39,824121	60,240398
DCT INTEIRA+ZIGZAG+HUFFMAN		18,211236	22,565133	30,492206	59,611744
DCT INTEIRA+ORDEM+HUFFMAN		19,273796	24,758385	37,809020	59,611744
DWT INTEIRA+ORDEM+HUFFMAN		19,359161	28,148748	49,915305	60,002527
DWT BLOCO+ORDEM+HUFFMAN		4,743383	24,114616	42,418982	59,953854

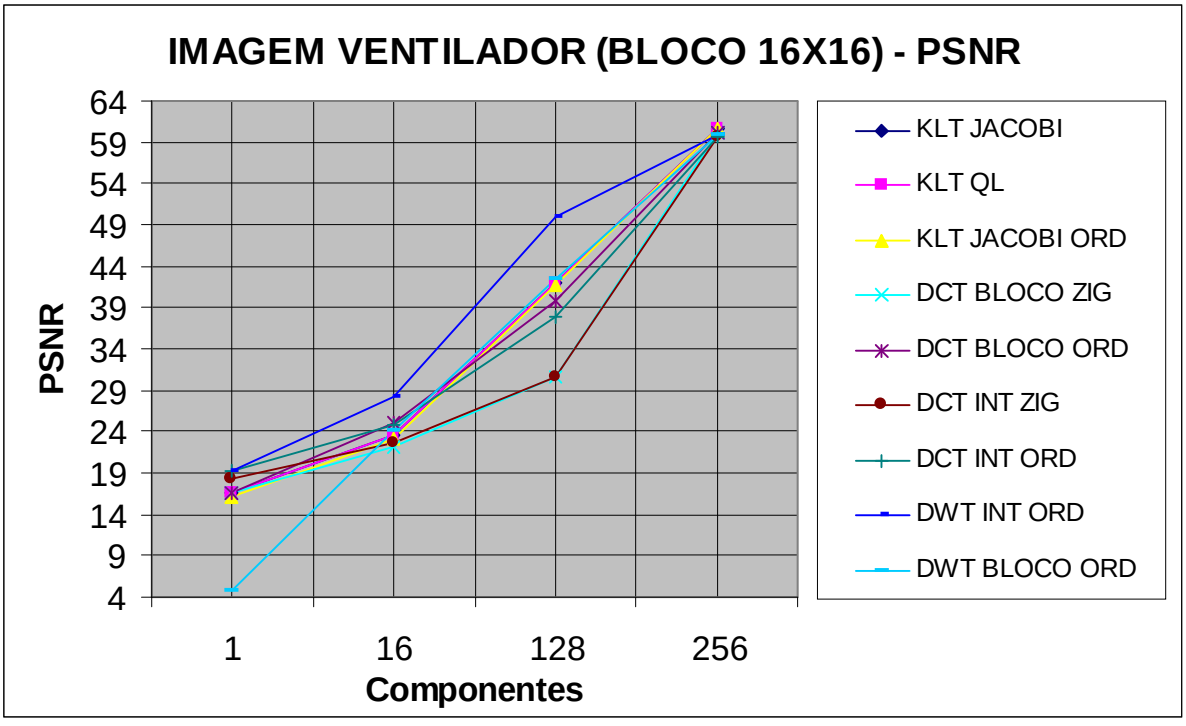


Figura 112 – PSNR da Imagem VENTILADOR (BLOCO 16x16)

A Tabela 87 em conjunto com a Figura 113 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para a *Razão de Compressão (CR)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 87 – CR da Imagem VENTILADOR (BLOCO 16x16)

Imagem:	**ventg**	Bloco:	**16**	x	**16**
RAZÃO DE COMPRESSÃO (CR)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		1,000000	1,000000	1,000000	1,000000
Algoritmo de HUFFMAN		1,156232	1,156232	1,156232	1,156232
KLT JACOBI + HUFFMAN		3,310835	1,097810	0,336463	0,197625
KLT QL + HUFFMAN		3,310835	1,100549	0,336715	0,197713
KLT JACOBI + ORDEM + HUFFMAN		3,310835	1,086937	0,334697	0,197625
DCT BLOCO+ZIGZAG+HUFFMAN		3,607777	1,432960	0,962658	0,797973
DCT BLOCO+ORDEM+HUFFMAN		3,607777	1,370320	0,923988	0,797973
DCT INTEIRA+ZIGZAG+HUFFMAN		3,621901	1,710068	1,056912	0,840417
DCT INTEIRA+ORDEM+HUFFMAN		3,732295	1,891745	1,081782	0,840417
DWT INTEIRA+ORDEM+HUFFMAN		3,717092	1,779173	1,056443	0,925220
DWT BLOCO+ORDEM+HUFFMAN		3,966299	1,145732	0,817982	0,736579

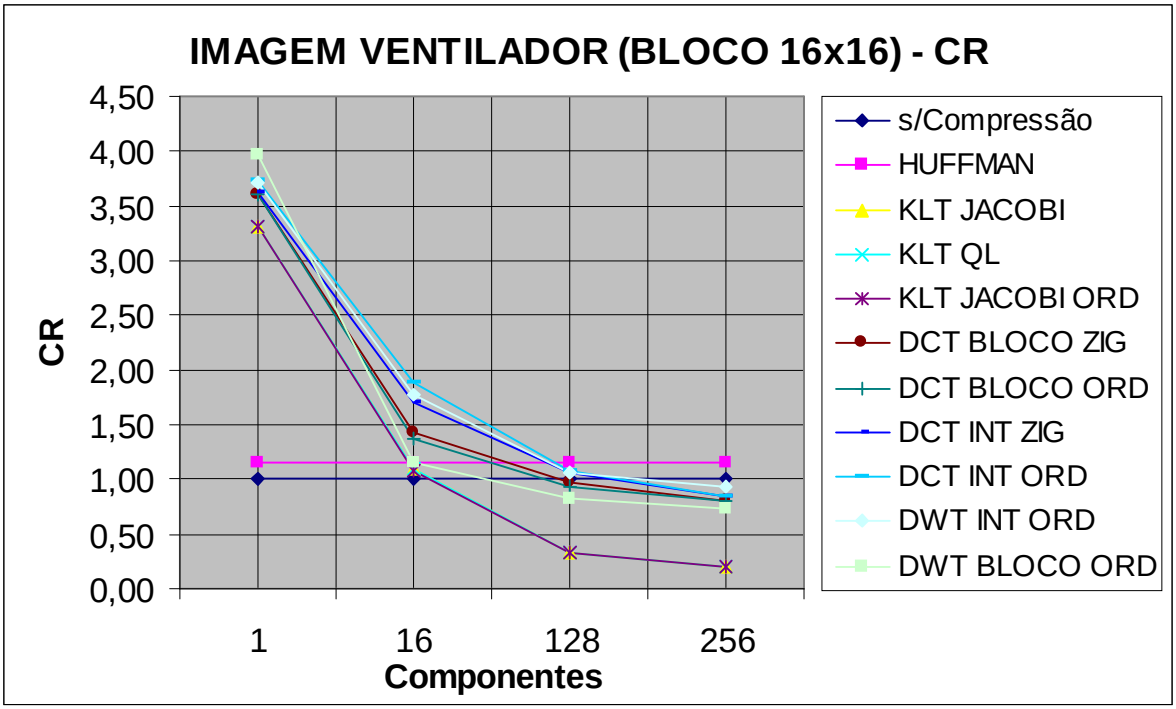


Figura 113 – CR da Imagem VENTILADOR (BLOCO 16x16)

A Tabela 88 em conjunto com a Figura 114 apresentam os resultados obtidos para a imagem VENTILADOR (256x256 PIXELS) utilizando um bloco de imagem de tamanho 16x16 PIXELS, para o *Tempo de Gravação (Tempo em segundos de processamento do arquivo compactado)* e para os vários tipos de cálculo propostos nesta seção.

Tabela 88 – Tempo de Gravação (seg) da Imagem VENTILADOR (BLOCO 16x16)

Imagem:	**ventg**	Bloco:	**16**	x	**16**
Tempo de Gravação (em segundos)					
Transformada		Número de Componentes Mantidos por Bloco			
		1	**16**	**128**	**256**
Imagem sem compressão		-	-	-	-
Algoritmo de HUFFMAN		0,06	0,11	0,05	0,06
KLT JACOBI + HUFFMAN		28,28	27,96	27,19	26,04
KLT QL + HUFFMAN		7,69	7,85	7,91	7,80
KLT JACOBI + ORDEM + HUFFMAN		25,26	26,36	26,74	28,29
DCT BLOCO+ZIGZAG+HUFFMAN		0,16	0,22	0,27	0,27
DCT BLOCO+ORDEM+HUFFMAN		0,28	0,28	0,33	0,33
DCT INTEIRA+ZIGZAG+HUFFMAN		0,16	0,22	0,27	0,27
DCT INTEIRA+ORDEM+HUFFMAN		0,22	0,27	0,22	0,27
DWT INTEIRA+ORDEM+HUFFMAN		0,17	0,22	0,22	0,28
DWT BLOCO+ORDEM+HUFFMAN		0,17	0,22	0,27	0,33

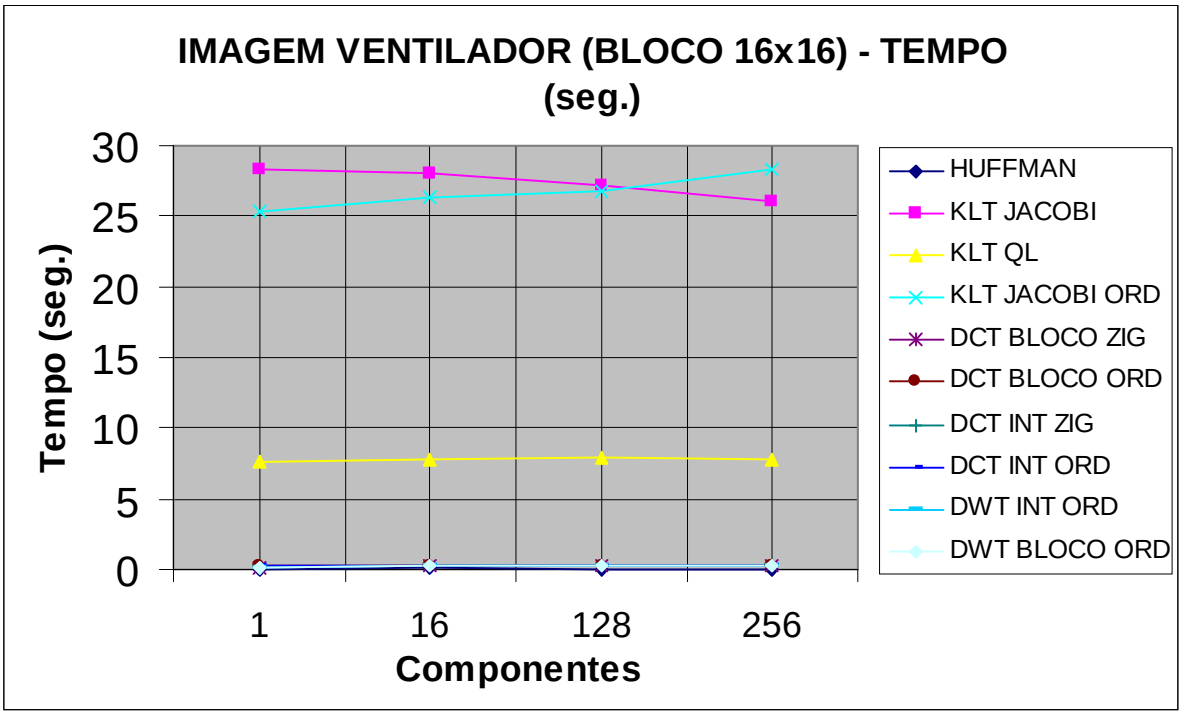


Figura 114 – Tempo de Gravação (seg) da Imagem VENTILADOR (BLOCO 16x16)

A Tabela 89 em conjunto com a Figura 115 apresentam os resultados da *PSNR (Razão Sinal/Ruído de Pico)* obtida para a imagem VENTILADOR (256X256 PIXELS) para 4 tamanhos de bloco de imagem, para 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 89 – PSNR da Imagem VENTILADOR - Compressão QL

Imagem:	ventg			
RAZÃO SINAL/RUÍDO DE PICO (PSNR)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	25,213930	26,166436	27,430039	31,503858
4:2	29,435316	31,047162	32,618846	42,010691
4:3	34,127517	36,815280	38,707453	54,896812
4:4	59,932730	59,535987	59,705002	60,651310

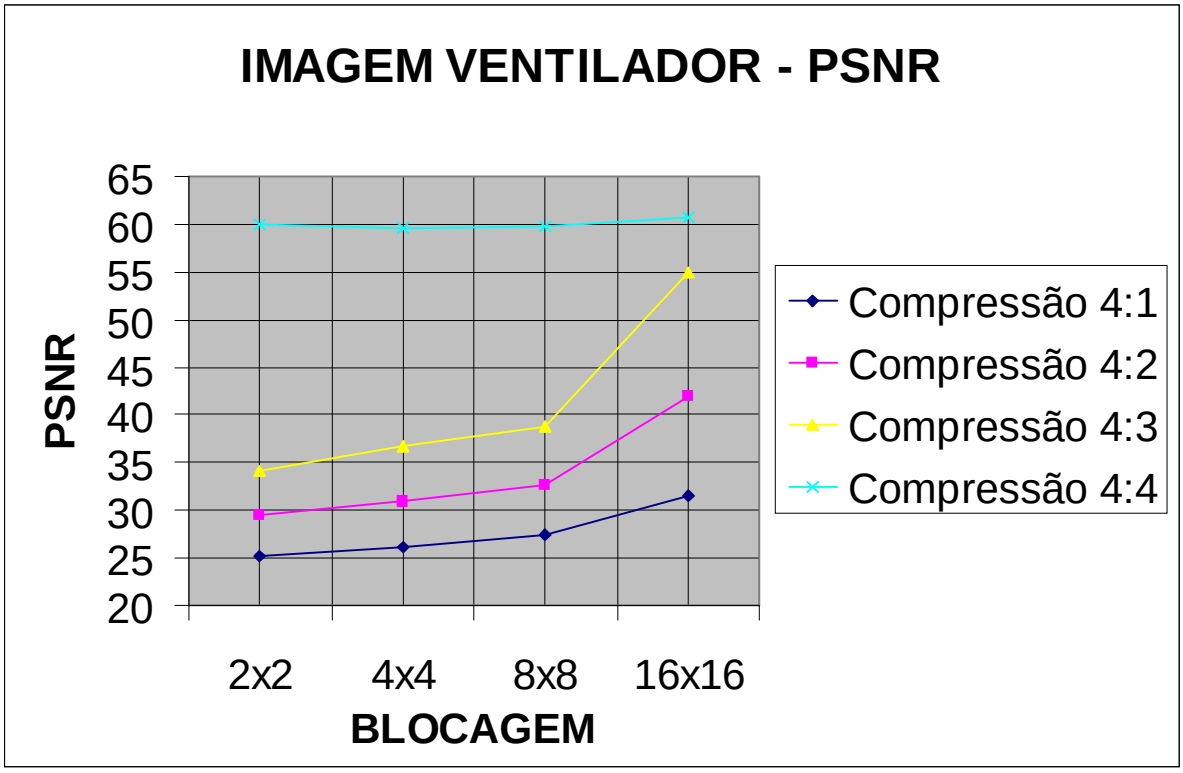


Figura 115 – PSNR da Imagem VENTILADOR – Compressão QL

A Tabela 90 em conjunto com a Figura 116 apresentam os resultados da *CR (Razão de Compressão)* obtida para a imagem VENTILADOR (256X256 PIXELS) para 4 tamanhos de bloco de imagem, para 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 90 – CR da Imagem VENTILADOR - Compressão QL

Imagem:	ventg			
RAZÃO DE COMPRESSÃO (CR)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	1,633176	1,194463	0,998067	0,551130
4:2	1,310345	1,015070	0,815199	0,336715
4:3	1,100294	0,897606	0,705216	0,246305
4:4	1,023115	0,843269	0,645341	0,197713

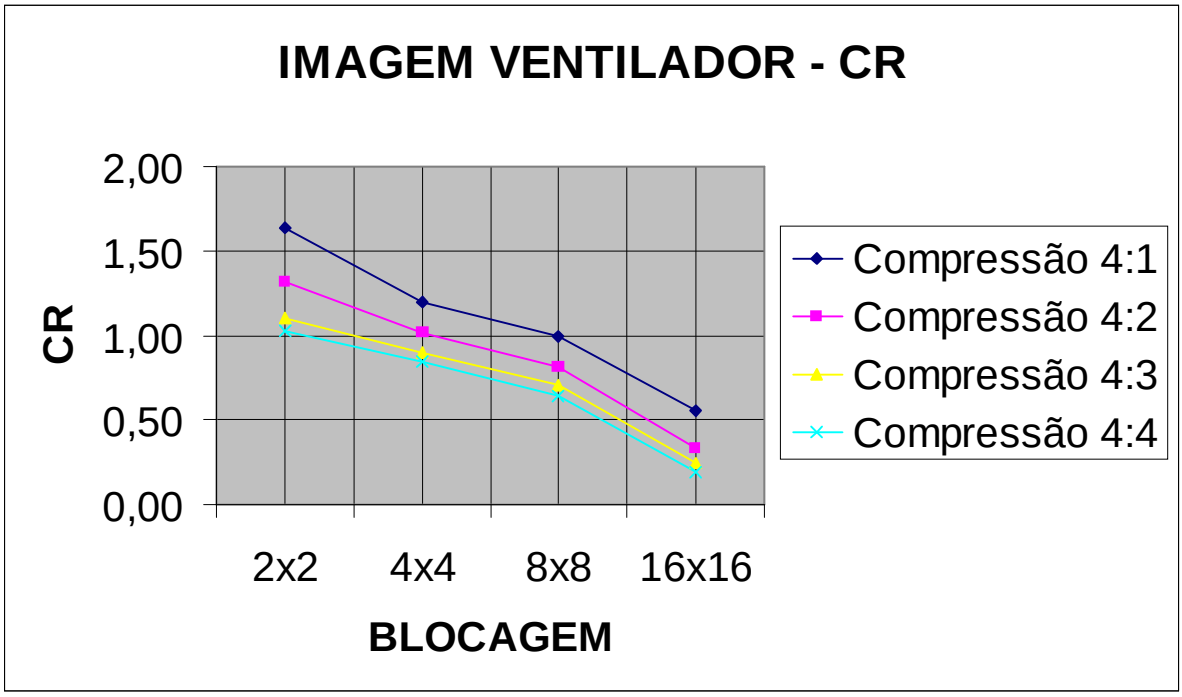


Figura 116 – CR da Imagem VENTILADOR – Compressão QL

A Tabela 91 em conjunto com a Figura 117 apresentam os resultados do *Tempo (em segundos)* obtida para a imagem VENTILADOR (256X256 PIXELS) para 4 tamanhos de bloco de imagem, para 4 razões de compressão de bloco e Método de Compressão QL.

Tabela 91 – Tempo de Gravação da Imagem VENTILADOR – Compressão QL

Imagem:	ventg			
TEMPO DE GRAVAÇÃO (SEGUNDOS)				
Compressão	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
4:1	0,17	0,16	0,500000	7,960000
4:2	0,11	0,27	0,550000	7,910000
4:3	0,16	0,27	0,550000	8,340000
4:4	0,21	0,22	0,610000	7,800000

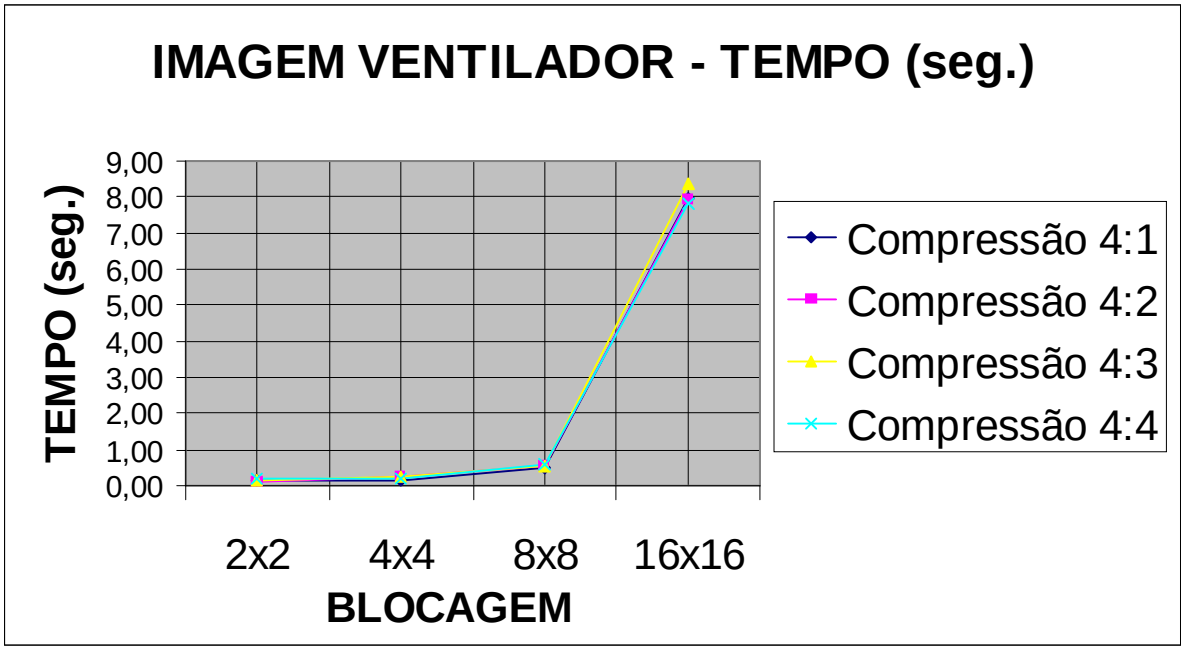


Figura 117 – Tempo de Gravação da Imagem VENTILADOR – Compressão QL

5.7 RESULTADO CONSOLIDADO KLT QL

Apresentam-se, neste item, uma comparação dos resultados da *PSNR*, *CR* e *TEMPO* entre todas as imagens testadas para uma compressão realizada com a KLT QL.

A Tabela 92 em conjunto com a Figura 118 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:4 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)*.

Tabela 92 – Tabela da PSNR para Compressão de Bloco 4:4

RAZÃO SINAL/RUÍDO DE PICO (PSNR) - Compressão de Bloco 4:4				
Imagem	Blocagem da Imagem			
	2x2	4x4	8x8	16x16
LENNA	58,655337	59,217347	59,433487	59,325490
BABUINO	58,955410	58,956011	58,950205	58,932236
GIRL	59,293760	59,285971	59,346920	61,372190
HOUSE	57,913411	59,274746	59,589434	61,618042
VENTILADOR	59,932730	59,535987	59,705002	60,651310
TIGRE	58,705727	58,926859	58,992626	59,807126

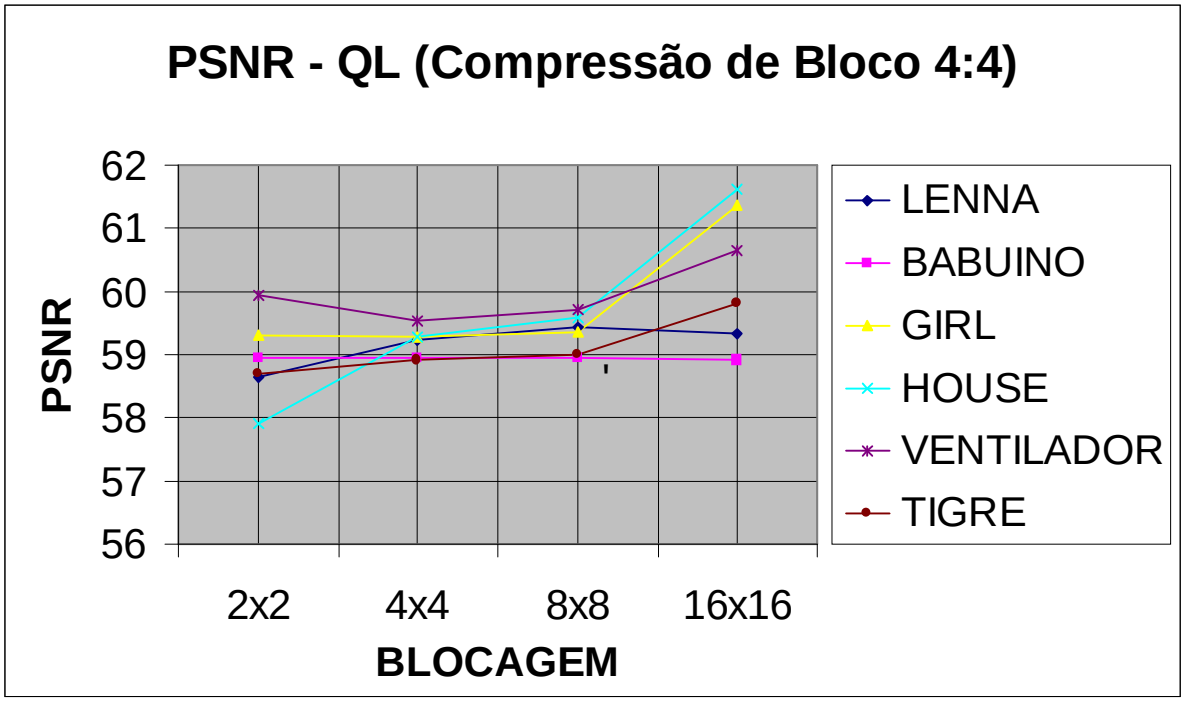


Figura 118 – Gráficos PSNR para Compressão 4:4 (sem compressão)

A Tabela 93 em conjunto com a Figura 119 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:4 e para as 4 blocagens testadas(2x2, 4x4, 8x8 e 16x16) para o parâmetro *Razão de Compressão (CR)*.

Tabela 93 – Tabela da CR para Compressão de Bloco 4:4

RAZÃO DE COMPRESSÃO (CR) - Compressão de Bloco 4:4				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	1,276488	1,454788	1,386284	0,584670
BABUINO	1,078443	1,079960	0,998112	0,512123
GIRL	1,059265	1,060631	0,861860	0,209731
HOUSE	1,074454	1,022424	0,864096	0,208783
VENTILADOR	1,023115	0,843269	0,645341	0,197713
TIGRE	0,883276	0,803963	0,649924	0,196063

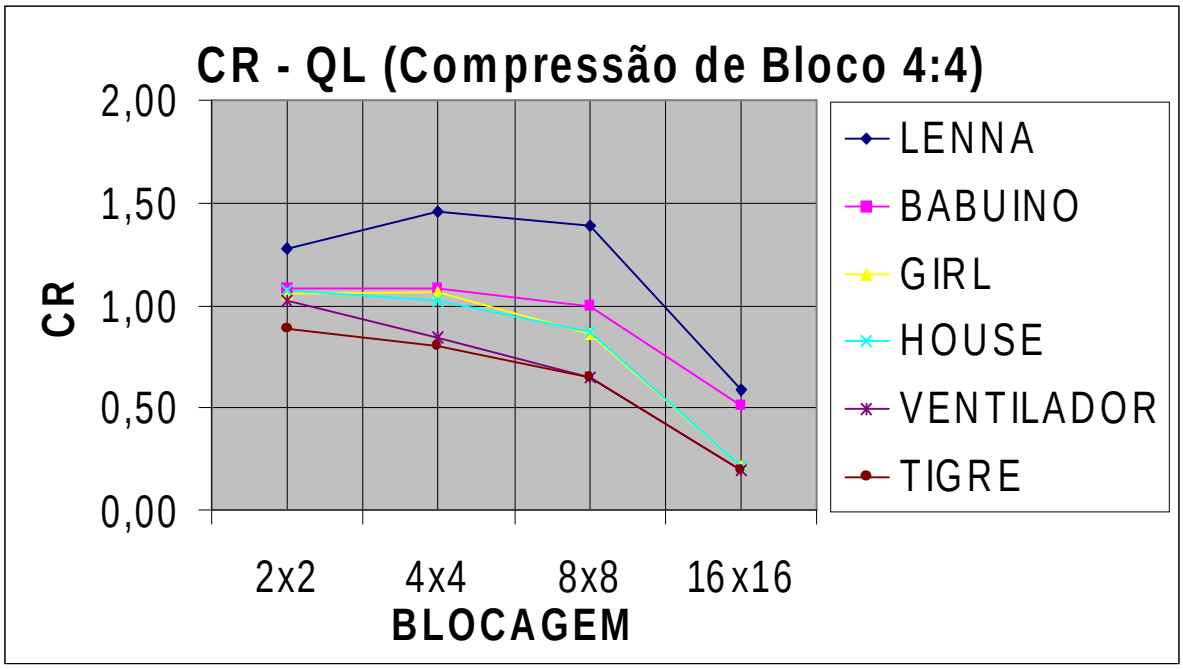


Figura 119 – Gráficos CR para Compressão 4:4 (sem compressão)

A Tabela 94 em conjunto com a Figura 120 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:4 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Tempo de Gravação (em segundos) (TEMPO)*.

Tabela 94 – Tabela de Tempo (seg.) para Compressão de Bloco 4:4

TEMPO DE GRAVAÇÃO (seg.) - Compressão de Bloco 4:4				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	0,610000	0,770000	1,820000	11,530000
BABUINO	0,600000	0,820000	1,980000	13,180000
GIRL	0,220000	0,270000	0,550000	6,870000
HOUSE	0,220000	0,280000	0,550000	7,520000
VENTILADOR	0,210000	0,220000	0,610000	7,800000
TIGRE	0,170000	0,280000	0,600000	8,070000

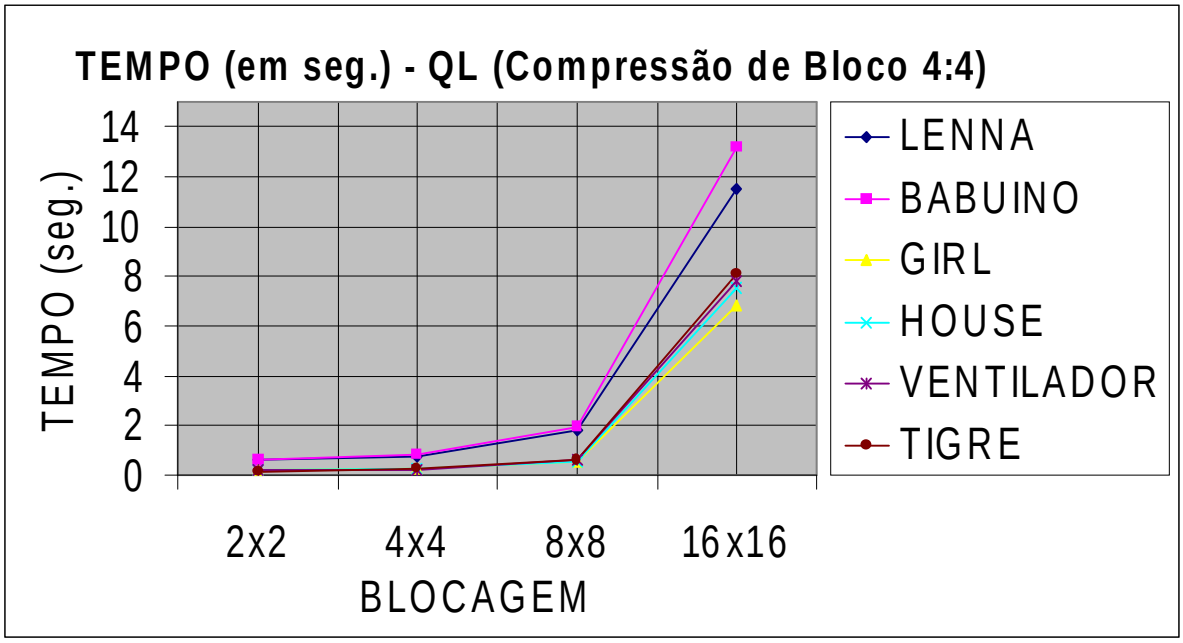


Figura 120 – Gráfico de Tempo (seg.) para Compressão 4:4 (sem compressão)

A Tabela 95 em conjunto com a Figura 121 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:3 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)*.

Tabela 95 – Tabela da PSNR para Compressão de Bloco 4:3

RAZÃO SINAL/RUÍDO DE PICO (PSNR) - Compressão de Bloco 4:3				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	43,385741	48,691906	52,436503	55,482310
BABUINO	31,387090	32,979501	33,910795	36,762812
GIRL	43,150484	45,847384	49,760005	61,300107
HOUSE	44,996675	50,237706	54,433953	61,568038
VENTILADOR	34,127517	36,815280	38,707453	54,896812
TIGRE	35,033915	38,805765	41,948317	54,675638

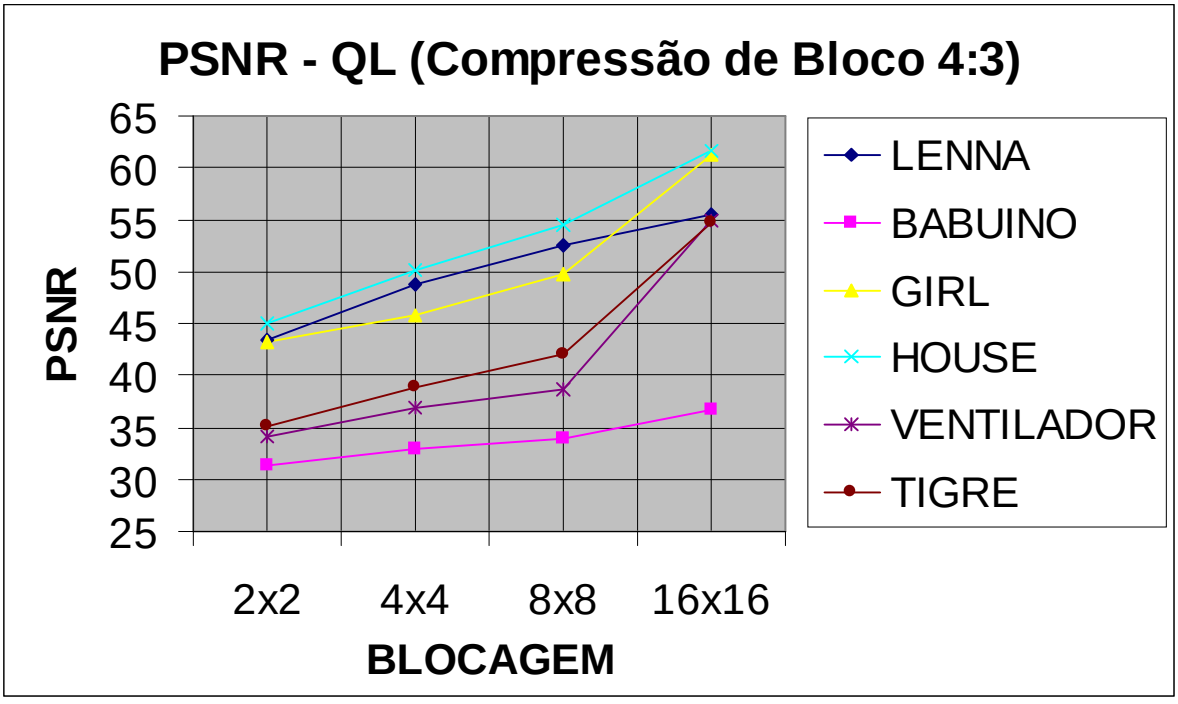


Figura 121 – Gráficos PSNR para Compressão 4:3

A Tabela 96 em conjunto com a Figura 122 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:3 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Razão de Compressão (CR)*.

Tabela 96 – Tabela da CR para Compressão de Bloco 4:3

RAZÃO DE COMPRESSÃO (CR) - Compressão de Bloco 4:3				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	1,357990	1,512379	1,451074	0,688241
BABUINO	1,207053	1,195209	1,105329	0,615992
GIRL	1,113872	1,102078	0,933204	0,264333
HOUSE	1,132102	1,048858	0,927216	0,262805
VENTILADOR	1,100294	0,897606	0,705216	0,246305
TIGRE	0,953358	0,851863	0,705134	0,243880

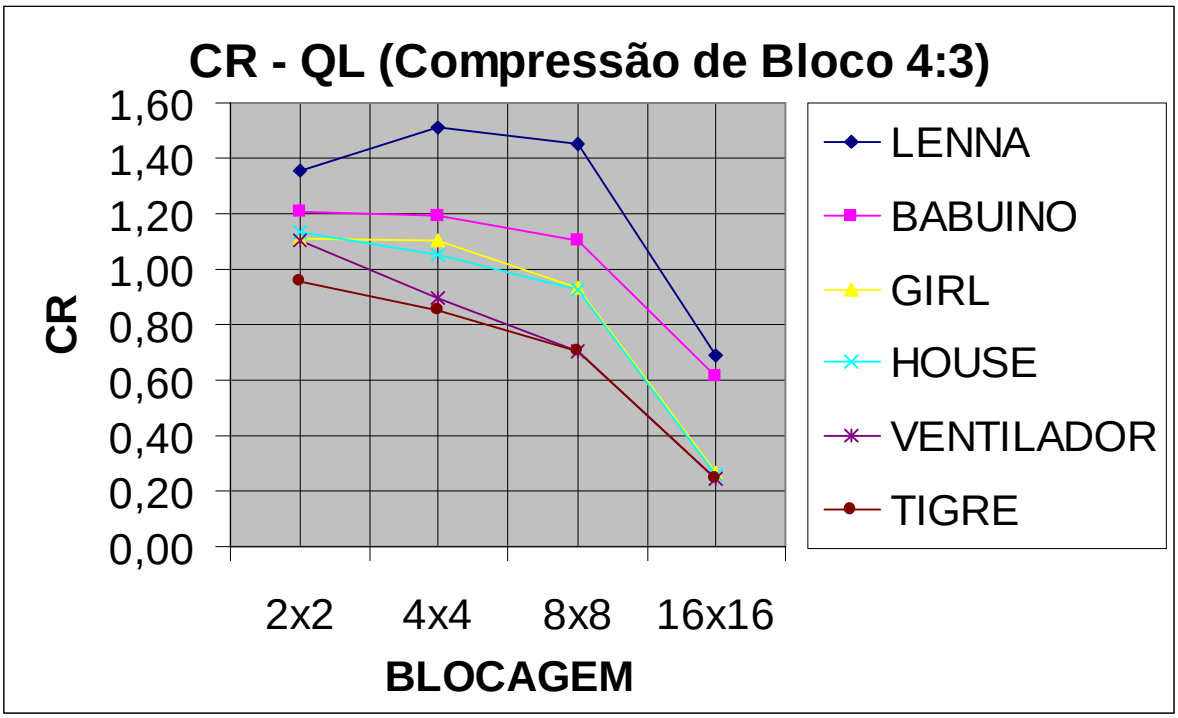


Figura 122 – Gráficos CR para Compressão 4:3

A Tabela 97 em conjunto com a Figura 123 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:3 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Tempo de Gravação (em segundos) (TEMPO)*.

Tabela 97 – Tabela de Tempo (seg.) para Compressão de Bloco 4:3

TEMPO DE GRAVAÇÃO (seg.) - Compressão de Bloco 4:3				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	0,550000	0,770000	1,870000	13,080000
BABUINO	0,610000	0,820000	1,810000	12,690000
GIRL	0,160000	0,280000	0,600000	6,700000
HOUSE	0,110000	0,220000	0,550000	6,920000
VENTILADOR	0,160000	0,270000	0,550000	8,340000
TIGRE	0,220000	0,330000	0,610000	7,300000

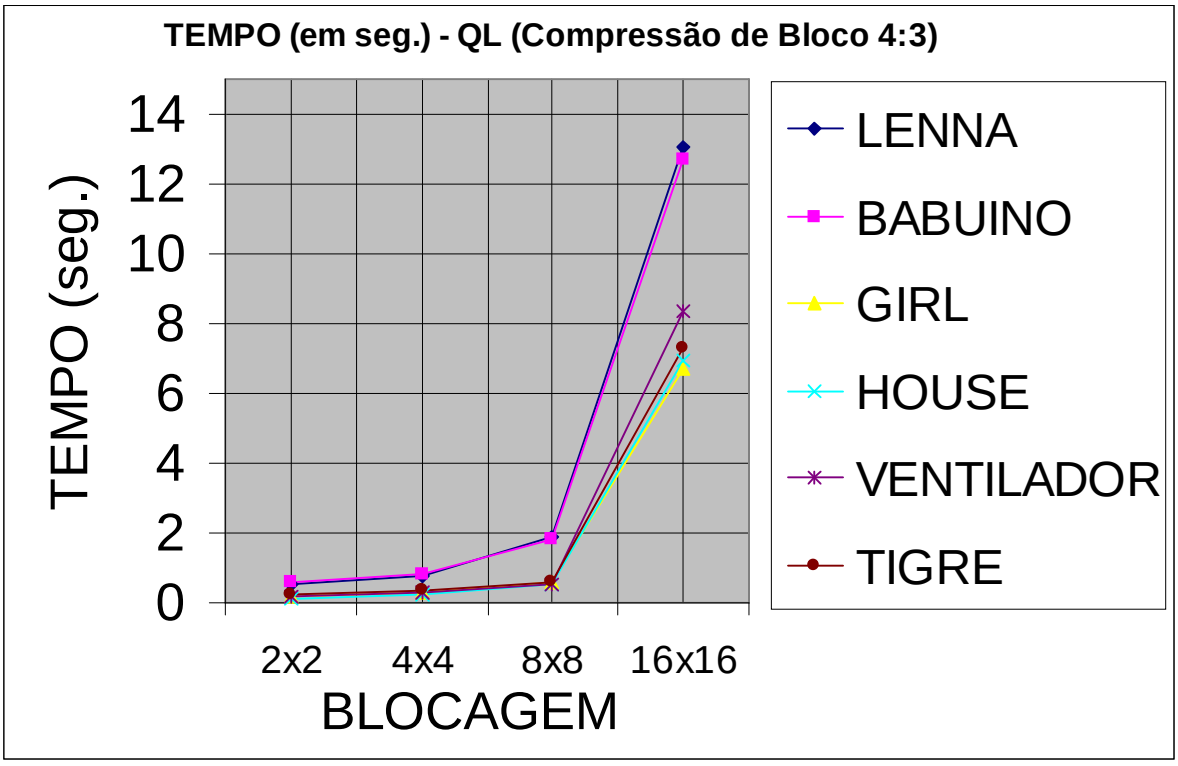


Figura 123 – Gráfico de Tempo (seg.) para Compressão 4:3

A Tabela 98 em conjunto com a Figura 124 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:2 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)*.

Tabela 98 – Tabela da PSNR para Compressão de Bloco 4:2

RAZÃO SINAL/RUÍDO DE PICO (PSNR) - Compressão de Bloco 4:2				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	36,347065	40,448388	42,785820	45,606938
BABUINO	27,098019	28,019715	28,774119	30,541566
GIRL	34,987986	39,300339	41,503623	53,361756
HOUSE	34,619904	40,575721	44,228315	53,304454
VENTILADOR	29,435316	31,047162	32,618846	42,010691
TIGRE	28,499992	31,516859	34,197880	42,592082

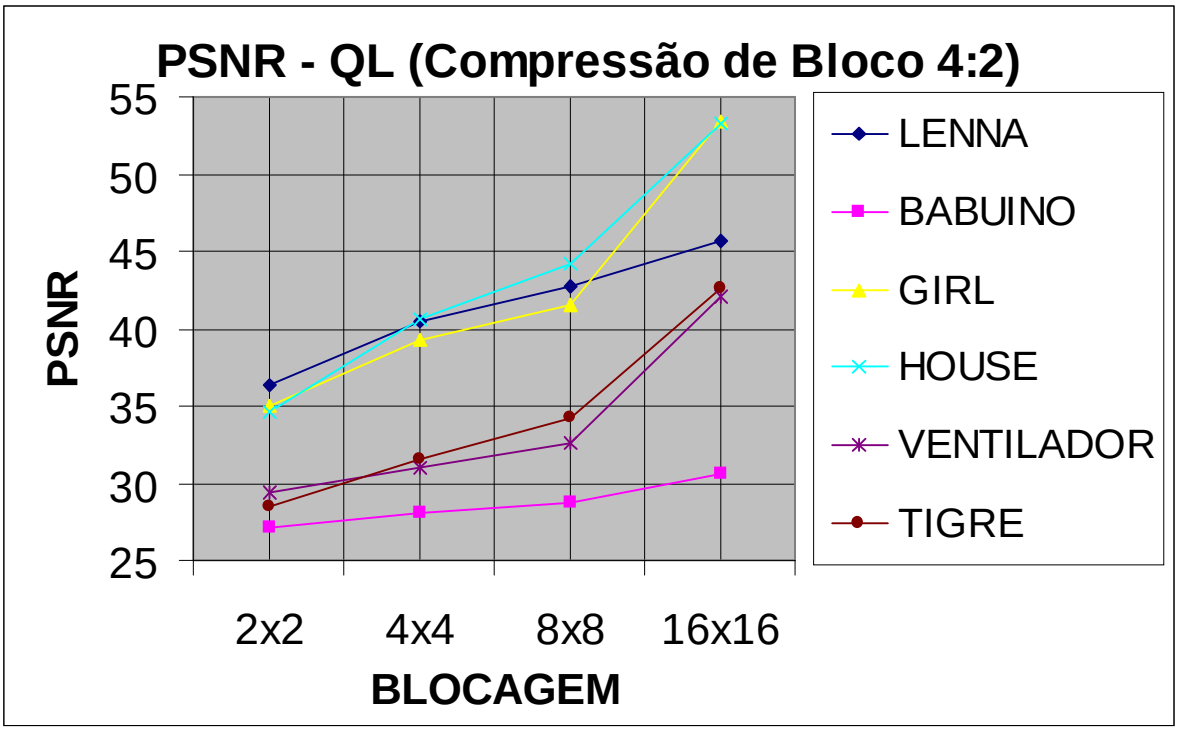


Figura 124 – Gráficos PSNR para Compressão 4:2

A Tabela 99 em conjunto com a Figura 125 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:2 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Razão de Compressão (CR)*.

Tabela 99 – Tabela da CR para Compressão de Bloco 4:2

RAZÃO DE COMPRESSÃO (CR) - Compressão de Bloco 4:2				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	1,628547	1,719315	1,621055	0,891544
BABUINO	1,529294	1,481038	1,362383	0,822846
GIRL	1,303703	1,203722	1,066438	0,362176
HOUSE	1,311506	1,156935	1,045007	0,359066
VENTILADOR	1,310345	1,015070	0,815199	0,336715
TIGRE	1,130699	0,973178	0,819290	0,332545

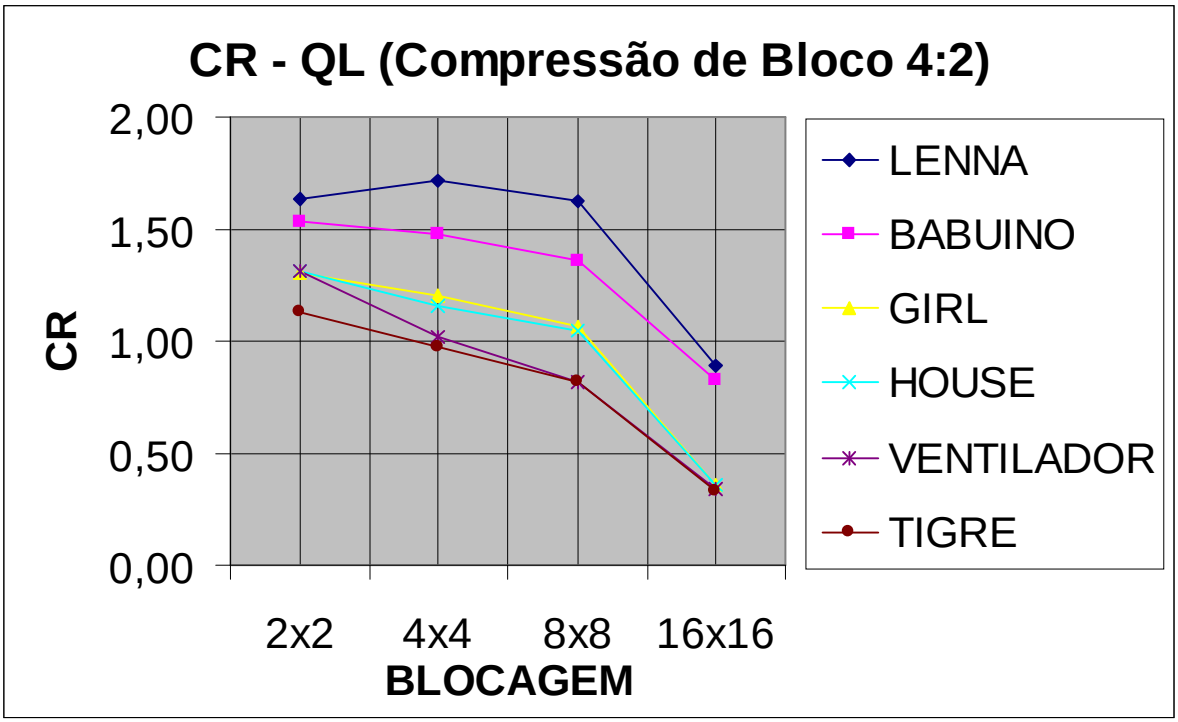


Figura 125 – Gráficos CR para Compressão 4:2

A Tabela 100 em conjunto com a Figura 126 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:2 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Tempo de Gravação (em segundos) (TEMPO)*.

Tabela 100 – Tabela de Tempo (seg.) para Compressão de Bloco 4:2

TEMPO DE GRAVAÇÃO (seg.) - Compressão de Bloco 4:2				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	0,550000	0,770000	1,860000	12,250000
BABUINO	0,550000	0,770000	1,750000	12,030000
GIRL	0,170000	0,220000	0,500000	6,430000
HOUSE	0,170000	0,220000	0,500000	7,200000
VENTILADOR	0,110000	0,270000	0,550000	7,910000
TIGRE	0,160000	0,270000	0,600000	7,470000

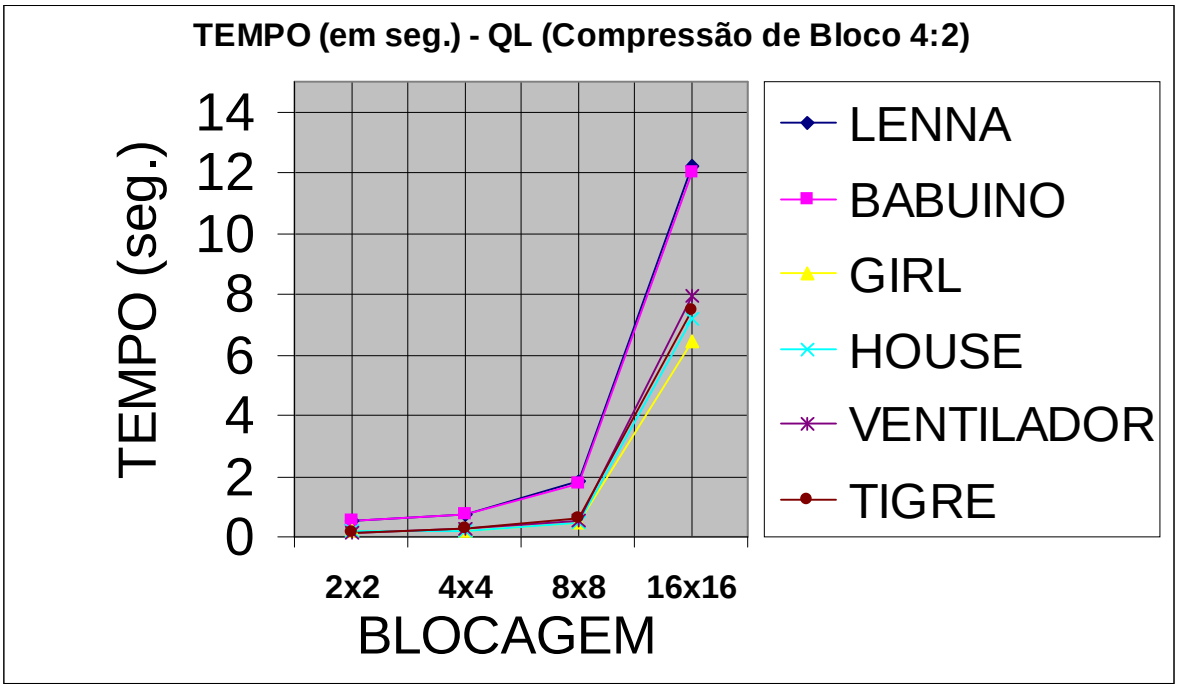


Figura 126 – Gráfico de Tempo (seg.) para Compressão 4:2

A Tabela 101 em conjunto com a Figura 127 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:1 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Razão Sinal/Ruído de Pico (PSNR)*.

Tabela 101 – Tabela da PSNR para Compressão de Bloco 4:1

RAZÃO SINAL/RUÍDO DE PICO (PSNR) - Compressão de Bloco 4:1				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	31,756868	34,001153	35,949734	37,898771
BABUINO	23,192588	24,066816	24,599288	25,549885
GIRL	31,834056	33,651158	35,410160	41,413439
HOUSE	28,930795	33,132473	35,432638	40,593152
VENTILADOR	25,213930	26,166436	27,430039	31,503858
TIGRE	24,087323	25,729851	27,176688	31,335009

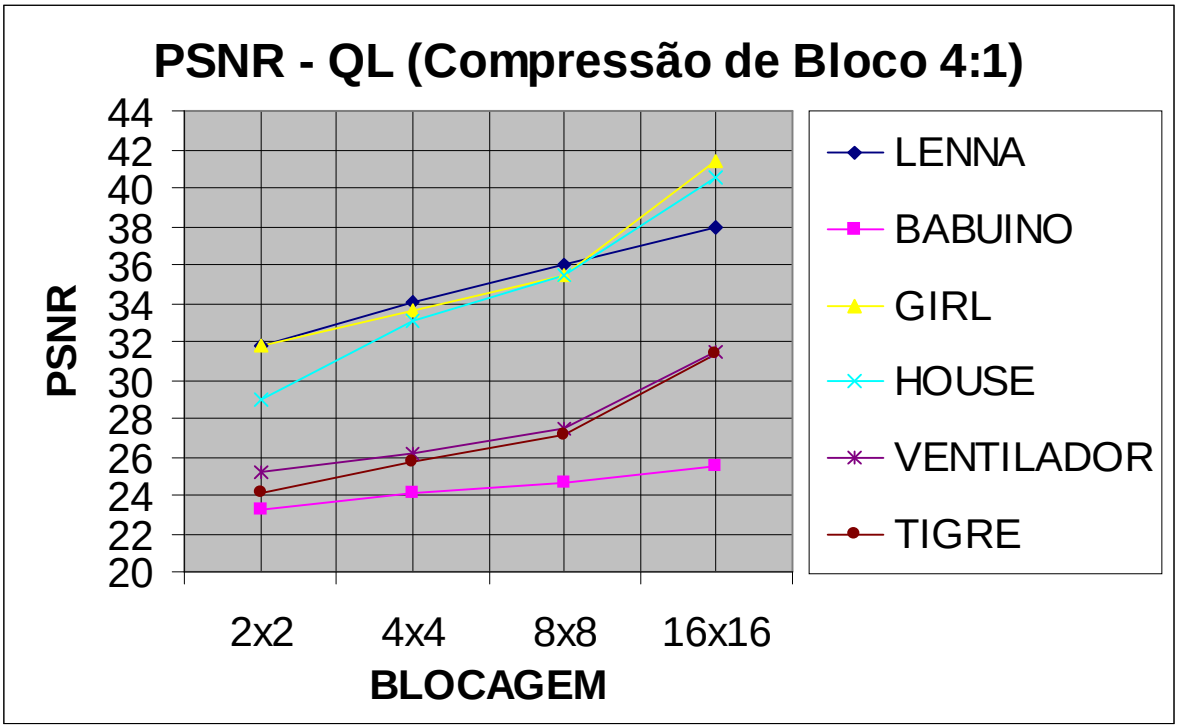


Figura 127 – Gráficos PSNR para Compressão 4:1

A Tabela 102 em conjunto com a Figura 128 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:1 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Razão de Compressão (CR)*.

Tabela 102 – Tabela da CR para Compressão de Bloco 4:1

RAZÃO DE COMPRESSÃO (CR) - Compressão de Bloco 4:1				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	2,237864	2,218867	2,036235	1,336383
BABUINO	2,278722	2,115626	1,927759	1,304461
GIRL	1,635623	1,428014	1,313963	0,600348
HOUSE	1,678400	1,391299	1,289819	0,592794
VENTILADOR	1,633176	1,194463	0,998067	0,551130
TIGRE	1,486920	1,199064	1,027043	0,541771

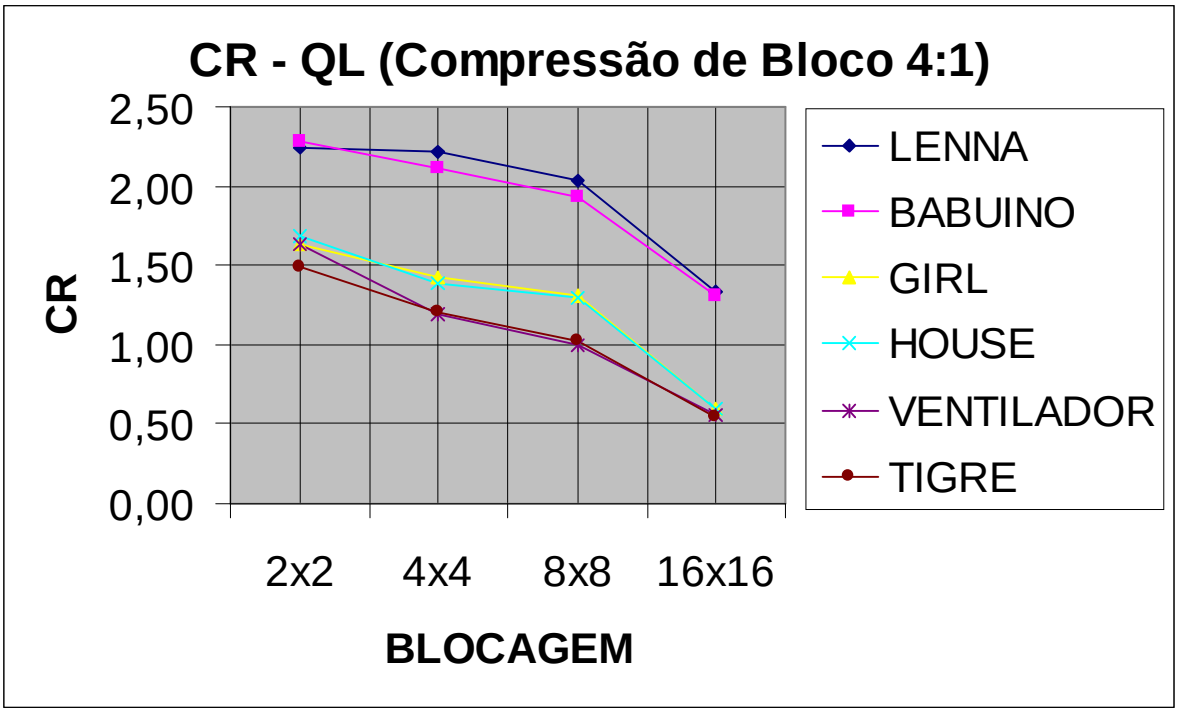


Figura 128 – Gráficos CR para Compressão 4:1

A Tabela 103 em conjunto com a Figura 129 apresentam os resultados obtidos para as 6 imagens, para uma Compressão de Bloco 4:1 e para as 4 blocagens testadas (2x2, 4x4, 8x8 e 16x16) para o parâmetro *Tempo de Gravação (em segundos) (TEMPO)*.

Tabela 103 – Tabela de Tempo (seg.) para Compressão de Bloco 4:1

TEMPO DE GRAVAÇÃO (seg.) - Compressão de Bloco 4:1				
Imagem	**Blocagem da Imagem**			
	2x2	4x4	8x8	16x16
LENNA	0,500000	0,710000	1,820000	11,920000
BABUINO	0,550000	0,710000	1,760000	14,770000
GIRL	0,170000	0,220000	0,440000	6,480000
HOUSE	0,110000	0,170000	0,430000	6,970000
VENTILADOR	0,170000	0,160000	0,500000	7,960000
TIGRE	0,170000	0,170000	0,500000	7,360000

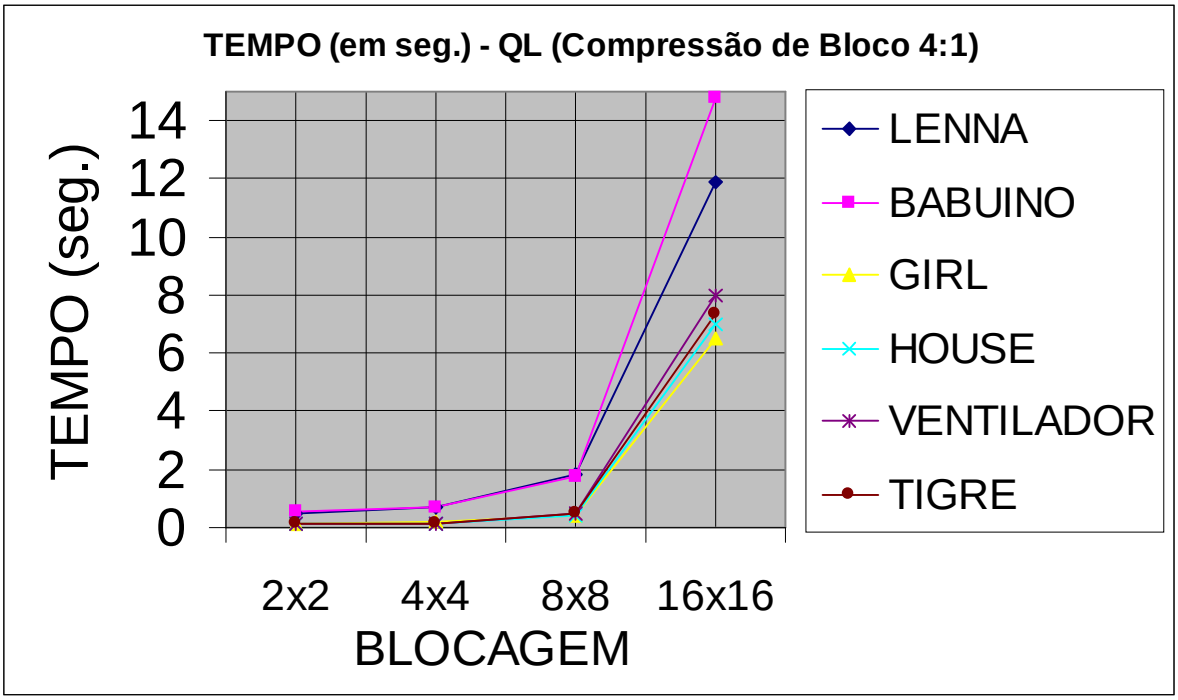


Figura 129 – Gráfico de Tempo (seg.) para Compressão 4:1

6 DISCUSSÃO DOS RESULTADOS

Pela análise das tabelas e gráficos apresentados na Seção 5 é possível apresentar os seguintes comentários:

a) As técnicas testadas se mostraram em alguns casos com performance inferior a aplicação direta do Algoritmo de HUFFMAN para o parâmetro *Razão de Compressão* (*CR*). Se, por um lado, a decomposição da imagem em coeficientes não correlacionados tende a eliminar a redundância interpixel, por outro lado, o efeito da própria aplicação da transformada em si (ampliando o intervalo de números, os coeficientes, que representam os PIXELS), a utilização da blocagem (provocando uma redistribuição da freqüência de ocorrência dos coeficientes no histograma da imagem) e o aumento do tamanho de bloco (provocando um aumento na quantidade de BYTES extras necessários para representar a base de representação no caso da KLT) trazem como conseqüência uma redução drástica de performance dos algoritmos com relação ao parâmetro *CR*, inclusive inferior a própria aplicação direta do algoritmo de HUFFMAN sobre a imagem. A aplicação da transformada amplia o intervalo de representação dos PIXELS da imagem, antes limitados ao intervalo de valores de 0 a 255, chegando a intervalos de coeficientes que podem variar de até –100.000 a +100.000 podendo ocasionar, isoladamente ou em conjunto com o efeito da blocagem, o aumento de entradas ou redistribuição de ocorrências dessas entradas na tabela de HUFFMAN, com o conseqüente aumento do código e do arquivo codificado (reduzindo a *CR*). Quando divide-se a imagem em blocos o efeito da blocagem é o de aumentar a freqüência de ocorrência de determinados coeficientes que, se fosse aplicado o mesmo algoritmo sobre a imagem inteira e selecionados pela quantização, não teriam um número de ocorrências elevado. Isso faz com que mais códigos de HUFFMAN com tamanho de código elevado (maior quantidade de bits de representação) sejam utilizados para representar os coeficientes da imagem aumentando o tamanho do arquivo codificado (e diminuindo a *CR*). O aumento no tamanho do bloco, no caso da KLT, tem um efeito drástico no aumento do tamanho do arquivo (redução da *CR*), tendo em vista que o aumento da quantidade dos vetores que formam a base, que é representada em FLOAT (4 BYTES) aumenta em muito a quantidade de BYTES a serem transmitidos no arquivo compactado (no caso da KLT). Neste caso, por exemplo, uma blocagem de tamanho 8x8 necessita de 16.384 BYTES (64 x 64 coeficientes x 4 BYTES por coeficiente) que são transmitidos no arquivo compactado e contribuem para a diminuição do parâmetro *CR*. Na imagem LENNA, por exemplo, temos uma performance inferior ao

método de HUFFMAN direto apenas para a KLT na blocagem 16x16 com compressões de bloco de 4:3 e 4:4, devido principalmente ao efeito dos BYTES extras para a representação da base de representação. Já na imagem BABUÍNO, o efeito do espalhamento da freqüência de ocorrência dos coeficientes no histograma da imagem é o principal fator que faz com que na DCT o parâmetro *CR* fique abaixo da compressão por HUFFMAN direto, na blocagem 2x2 e 4x4. Na medida em que aumentamos a blocagem esse efeito diminui nitidamente, realçando a maior concentração dos coeficientes em torno de valores específicos na medida em que o tamanho de bloco aumenta (aproximando-se, gradativamente, da imagem inteira). Na imagem BABUÍNO nota-se este resultado inferior da KLT em relação ao algoritmo de HUFFMAN direto, na blocagem 16x16 (compressões de bloco 4:3 e 4:4) e na blocagem 8x8 (compressão de bloco 4:4). O mesmo efeito para a DCT e KLT ocorre para as imagens MENINA (GIRL), CASA (HOUSE), TIGRE e VENTILADOR. Para a imagem CASA, porém, percebe-se um desempenho da DCT INT ORD e DCT INT ZIG ligeiramente inferior ao algoritmo de HUFFMAN direto, para tamanhos de bloco 2x2 (compressão de bloco 4:3 e sem compressão de bloco), 4x4 (sem compressão de bloco), 8x8 (sem compressão de bloco) e 16x16 (sem compressão de bloco), que neste caso, como não há realmente blocagem da imagem, é causado pelo efeito da própria transformada em si na redistribuição das freqüências do histograma (o intervalo de representação dos PIXELS também aumenta mas não é o fator mais importante neste caso).

b) Os resultados relativos a CR (*Razão de Compressão*) foram praticamente idênticos para os 3 tipos de cálculos propostos para a Transformada KARHUNEN-LOÈVE (KLT), isto é: (1) KLT JACOBI, (2) KLT QL e (3) KLT JACOBI ORD. Assim, no que se refere a *CR*, qualquer uma das 3 técnicas utilizadas nos testes para a KLT apresentam resultados praticamente equivalentes.

c) Para a blocagem 2x2 a *KLT* é a que proporciona o melhor resultado em termos de *CR* para uma compressão de bloco de 4:1 (apenas 1 componente é transmitido dos 4 existentes no bloco), ocorrendo esse resultado para todas as 6 imagens testadas. Para uma compressão de bloco de 4:2 (ou 2:1) a KLT tem uma performance inferior para a *DWT INT ORD* (que é 0,56% superior na *CR*) apenas para a imagem LENNA, sendo superior para todas as outras 5 imagens nessa compressão de bloco. Para uma compressão de bloco de 4:3, a KLT perde apenas para a *DCT INT ZIG* (que é 1,8 % superior na *CR*) e para a *DWT INT ORD* (que é 6,5 % superior na *CR*) para a imagem LENNA, sendo superior a todas as demais técnicas para as imagens VENTILADOR, TIGRE e BABUÍNO e inferior

a DWT INT ORD para as imagens HOUSE (7,4 % superior) e GIRL (12,5 % superior). Sem compressão a KLT perde para a *DCT INT ORD* e *DCT INT ZIG* (que são, ambos, 2,6 % superiores na *CR*) e para a *DWT INT ORD* (que é 13,33 % superior na *CR*) para a imagem LENNA, possuindo performance superior a todas as demais técnicas para as imagens VENTILADOR e TIGRE e inferior apenas à *DWT IND ORD* para as imagens HOUSE (6,02% superior), GIRL (9,67% superior) e BABUÍNO (1,41% superior). Como se percebe para a blocagem 2x2 a *KLT* mostra-se superior na maior parte das imagens testados com os vários tamanhos de bloco propostos, quando o requisito de compressão é o recomendado. Porém, utilizando a blocagem 2x2 limita-se a *CR* a uma compressão de bloco de 4:1, o que acaba, na prática, após a aplicação do algoritmo de HUFFMAN (gerando uma *CR* líqüida de apenas 2,237864:1 para a imagem LENNA, por exemplo).

d) Para a blocagem 4x4 a *KLT* é a que proporciona o melhor resultado em termos de *CR* entre todos os níveis de compressão de bloco testados para a imagem LENNA, tendo uma performance inferior para as demais imagens com relação a *DCT INT ZIG, DCT INT ORD* e *DWT INT ORD* (transformadas aplicadas sobre a imagem não blocada). Cabe ressaltar, porém, que utilizando a blocagem 4x4 limita a *CR* a uma compressão de bloco de 16:1 (o que acaba, na prática, após a aplicação do algoritmo de HUFFMAN, gerando uma CR líqüida de 3,249093:1 para a imagem LENNA, por exemplo).

e) Para a blocagem 8x8, os efeitos da transmissão dos componentes dos autovetores já se mostram presentes e o desempenho da *KLT* sofre uma degradação em relação as demais transformadas. Para a imagem LENNA, por exemplo, com uma compressão de bloco de 64:1 a *KLT* perde para todas as demais transformadas sendo o melhor desempenho apresentado pela *DWT BLOCO ORD* (que é 17,31 % superior na *CR*). Para uma compressão de bloco de 64:16 (ou 4:1), a *KLT* perde apenas para a *DWT INT ORD* (que é 7,5 % superior na *CR*), para a *DCT INT ORD* (que é 6,9 % superior na *CR*) e para a *DCT INT ZIG* (4,43 % superior na *CR*). Para a compressão de bloco de 64:32 (ou 2:1) a KLT perde apenas para a *DCT BLOCO ORD* (que é 14,4 % superior na *CR*), *DCT BLOCO ZIG* (que é 12,0 % superior na *CR*), *DWT INT ORD* (que é 3,1 % superior na *CR*) e para a *DCT INT ORD* (que é 0,15 % superior na *CR*). Para o caso de não haver compressão (compressão de bloco 64x64), a KLT perde apenas para a *DCT BLOCO ORD* e *DCT BLOCO ZIG* (que foram, ambas, 25,0 % superior na *CR*) e para a *DWT INT ORD* (que é 4,2 % superior na *CR*). Na blocagem 8x8 limita-se a *CR* a uma compressão de bloco de 64:1, que acaba na prática, com a aplicação do *algoritmo de HUFFMAN*, gerando uma *CR* líqüida de 3,467965:1 para a *KLT* sobre a imagem LENNA, embora se tenha

alcançado, nesta compressão de bloco, uma *CR* de 4,068283 para a *DWT BLOCO ORD* e de 3,846194:1 para a *DCT BLOCO ZIG*.

f) Para a blocagem 16x16, a KLT sofre uma degradação significativa para compressões de bloco de 256:128 (ou 2:1) e 256:256 (quando não há compressão de bloco). Esse comportamento é devido ao excesso de *BYTES extras* necessários para transmitir os autovetores. Observa-se que para uma compressão de bloco de 2:1 são necessários 131.072 *BYTES* extras (= 256 x 128 x 4 *BYTES/componente do autovetor*) e no caso de não haver compressão da imagem serão necessários 262.144 *BYTES* extras (o que significa, em *BYTES*, exatamente o equivalente ao tamanho da imagem original!). Cabe ressaltar que neste programa nenhum tratamento especial foi realizado para transmitir os autovetores, que foram inclusive transmitidos sem compressão pelo Algoritmo de HUFFMAN (assim como todos as demais variáveis que contribuem para os *BYTES extras* a serem transmitidos com a imagem*)*. Para a imagem LENNA e uma compressão de bloco de 256:1, por exemplo, a KLT perde para todas as demais técnicas testadas sendo que as melhores transformadas foram a *DWT INT ORD* (que é 15,9 % superior à KLT na *CR*) e a *DCT INT ORD* (que é 16,2 % superior à *KLT* na *CR)*. Resultados superiores ocorrem também para todas as demais transformadas testadas para um tamanho de bloco de 16x16.

g) Observando os resultados obtidos nas 6 imagens com relação ao parâmetro *CR* verifica-se que este parâmetro tende a diminuir com o aumento da blocagem (mantendo-se constante a compressão de bloco) e tende aumentar com o aumento da compressão de bloco (mantendo-se constante a blocagem). Com o aumento da blocagem a quantidade de BYTES extras vai aumentando diminuindo a *CR.* O efeito do aumento do tamanho de bloco torna praticamente inviável o processamento de algumas blocagens utilizando a KLT, pois geram arquivos comprimidos de tamanho superior à imagem original. Constata-se claramente este aspecto utilizando o tamanho de bloco 16x16, onde são gerados arquivos de tamanho menor do que o original somente as imagens BABUÍNO e LENNA (compressão de bloco 4:1). Utilizando o tamanho de bloco 8x8, observa-se um arquivo compactado de tamanho menor do que o original para todas as imagens com exceção da imagem VENTILADOR (compressão de bloco 4:1), todas menos VENTILADOR e TIGRE (compressão de bloco 4:2), apenas LENNA e BABUÍNO (compressão de bloco 4:3) e apenas LENNA (compressão de bloco 4:4). Utilizando o tamanho de bloco 4x4, observa-se um arquivo compactado de tamanho menor do que o original para todas as imagens (para a compressão de bloco 4:1), todas menos TIGRE (para a compressão de bloco 4:2), todas as imagens menos VENTILADOR e TIGRE

(para a compressão de bloco 4:3) e todas as imagens menos VENTILADOR e TIGRE (para a compressão de bloco 4:4). Utilizando o tamanho de bloco 2x2, observa-se um arquivo compactado de tamanho menor do que o original para todas as imagens (para as compressões de bloco de 4:1 e 4:2) e para todas as imagens com exceção da imagem TIGRE (para compressões de bloco 4:3 e 4:4).

h) Os resultados relativos ao *PSNR* (*Razão Sinal/Ruído de Pico*) foram praticamente idênticos para a *KLT JACOBI* e a *KLT QL* havendo uma pequena diferença para o tipo *KLT JACOBI ORD*. Observa-se que quando não há compressão de bloco os 3 tipos de cálculo fornecem praticamente o mesmo resultado. A escolha do maior coeficiente pela seleção direta de valor, embora cause uma diminuição da *CR*, como já explicado antes, ocasiona uma aumento na *PSNR* chegando esta razão a ser 12,89% superior para a *KLT BLOCO ORD* em relação as outras duas técnicas KLT para a imagem LENNA, por exemplo. Esse resultado é coerente porque estão sendo selecionandos sempre os maiores coeficientes e, portanto, devem gerar uma imagem reconstruída de melhor qualidade, em detrimento de um tempo maior de processamento.

i) Para a blocagem 2x2 a *KLT* proporciona o pior resultado entre praticamente todas as demais transformadas testadas em termos de *PSNR* para quase todas as compressões de bloco. Somente quando não há compressão (compressão de bloco 4:4) é que a *KLT* se aproxima das demais transformadas perdendo para a imagem LENNA, por exemplo, apenas para a *DCT BLOCO ZIG* (que é 11,06% superior na *PSNR*) e para a *DCT BLOCO ORD* (que é 9,36% superior na *PSNR*). A diferença na *PSNR* pode ser até 45,11% superior à *KLT*, como no caso *DWT INT ORD* com a compressão de bloco 4:2. Esse comportamento inferior da *KLT* em detrimento das demais transformadas justamente na *PSNR* em que a *KLT* deveria possuir teoricamente a melhor performance em relação a qualquer outra transformada para o mesmo número de coeficientes é totalmente explicável. Ocorre que a técnica apresentada para o cálculo da *KLT* aproxima o cálculo da matriz de correlação dos blocos de imagem como a média das matrizes de correlação dos blocos da imagem. Essa média, geralmente não é a base adequada para todos os blocos da imagem, e até mesmo poderá ser totalmente inadequada para muitos dos blocos se a correlação entre os PIXELS de um vetor-bloco variar de forma acentuada em relação à média, como é o caso dos blocos que formam as bordas das figuras na imagem (efeito da blocagem). A prova disso é que, na medida em que aumentamos o tamanho do bloco (4x4, 8x8, etc) essa diferença vai diminuindo sensivelmente (os gráficos da *PSNR* das diversas transformadas vão se aproximando) e passa a ser, para a imagem LENNA, por exemplo,

de 6,12 % a favor da *DWT INT ORD*, para a mesma compressão de bloco de 256:128 (ou 4:2) anterior (quando a diferença foi de 45,11 %). Para os demais tamanhos de bloco de 4x4, 8x8 e 16x16 têm-se, então, que a *KLT* é inferior com relação à PSNR mas num percentual cada vez menor de diferença, conforme explicado acima, o mesmo ocorrendo com as demais imagens.

j) Tendo em vista os resultados inferiores da KLT relativamente à PSNR, a fim de que fosse verificado de forma conclusiva que esse comportamento realmente é causado pelo descasamento dos autovetores calculados para a matriz de correlação média dos vetores-bloco, implementou-se no programa o cálculo da KLT BLOCO que calcula um conjunto de autovetores independentes para cada bloco da imagem e transmite todos esses autovetores dentro do arquivo compactado, reconstruindo posteriormente a imagem. Evidentemente, não se levou em consideração o tamanho final do arquivo compactado (que para um processamento da imagem LENNA, por exemplo, com uma blocagem de 2x2 e compressão de bloco de 4:1, acabou ficando com um tamanho total de 1.166.477 BYTES, sendo 1.048.576 BYTES destes somente para transmitir as dimensões dos autovetores). Porém, ao reconstruir a imagem verificou-se que esta era absolutamente idêntica à original. Além disso, constatou-se também que, neste caso, exatamente como a teoria prevê, só existe 1 único coeficiente correspondente a uma única dimensão do bloco (o valor deste coeficiente é igual a distância euclidiana das componentes do vetor-bloco de imagem) e, portanto, seja qual for o tamanho do bloco, apenas 1 coeficiente de cada bloco e apenas 1 autovetor correspondente necessitam ser compactados e transmitidos. Essa é a razão principal de vários pesquisadores estarem trabalhando em técnicas eficazes de adequar a matriz de correlação às características individuais dos blocos de imagem (Ex.: (HAYKIN,1999)).

k) Os resultados relativos ao *Tempo (Tempo de Gravação em segundos)* incluem o tempo de gravação do arquivo compactado inclusive. O tempo de reconstrução do arquivo chega a ser 50% inferior ao de gravação (devido aos autovetores já serem transmitidos no arquivo compactado) e não é um fator limitante no processamento. Os resultados relativos ao *Tempo de Gravação* são bem diferentes para os 3 tipos de cálculos propostos para a KLT JACOBI, a KLT JACOBI ORD e a KLT QL. A utilização da KLT QL é superior a dos outros dois tipos de cálculo, para todas as compressões de bloco utilizadas chegando, por exemplo, a ser 147,69 % superior a KLT JACOBI para tamanho de bloco de 16x16 e compressão de bloco de 256x1 e 144,75% superior a KLT JACOBI ORD para tamanho de bloco de 16x16 e compressão de bloco de 256x128, obtidos para a imagem LENNA.

l) Para a blocagem 2x2, a KLT QL demonstrou possuir o melhor *Tempo de Processamento* entre praticamente todos os algoritmos testados neste trabalho para qualquer compressão de bloco utilizado, superando inclusive para a imagem LENNA, por exemplo, a *DCT INT ZIG* (45,78 % superior para compressão de bloco de 4:1) e a *DWT INT ORD* (38,55 % superior para compressão de bloco de 4:1), que são aplicadas na imagem inteira (sem blocagem). Isso ocorre porque tanto o algoritmo de pesquisa do maior valor na *DWT* quanto o algoritmo de seleção em *ZIGZAG* no caso da *DCT* ocasionam também um tempo adicional de processamento que a *KLT* também possui quando realiza o cálculo dos autovetores (que já vêm ordenados pelos autovalores da matriz de correlação do vetor-bloco). Em poucas imagens e para algumas compressões de bloco a DWT INT ORD possuiu um tempo de processamento equivalente a KLT QL na blocagem 2x2 e para a imagem GIRL sem compressão de bloco, a DWT INT ORD foi 75% superior.

m) Para a blocagem 4x4, a *KLT QL* demonstrou possuir o melhor *Tempo de Processamento* juntamente com a *KLT JACOBI* e a *DWT INT ORD*, com pequenas vantagens de um ou outro dependendo da compressão de bloco utilizada.

n) Para a blocagem 8x8 a *KLT QL* mostra-se bem inferior as demais transformadas, mas ainda superior a *KLT JACOBI e KLT JACOBI ORD* (chegando a *KLT JACOBI ORD* a possuir um tempo de processamento 15,53 % inferior à *KLT QL* para a imagem LENNA). As melhores performances neste caso são da *DCT BLOCO ZIG* (com um tempo de processamento da *KLT QL* que chega a ser 81,82 % a mais do que o da *DCT para a imagem LENNA*) e da *DWT INT ORD* (com um tempo de processamento da *KLT QL* que chega a ser 91,30 % a mais do que o da *DWT para a imagem LENNA*).

o) Para a blocagem 16x16 a *KLT QL* mostra-se ainda pior do que para a blocagem 8x8, com relação as demais transformadas, mas muito superior a *KLT JACOBI* (que chega a possuir um tempo de processamento 147,69 % a mais para uma compressão de bloco de 256:1 para a imagem LENNA) e a *KLT JACOBI ORD* (que chega a possuir um tempo de processamento 144,75 % a mais para uma compressão de bloco de 256:128 para a imagem LENNA). As melhores performances neste caso, para a imagem LENNA, por exemplo, são da *DCT BLOCO ZIG* (o tempo de processamento da *KLT QL* chega a ser 725,55 % a mais do que o da *DCT*) e da *DWT INT ORD* (com um tempo de processamento da *KLT QL* que chega a ser 898,08 % a mais do que o da *DWT*).

A degradação significativa no tempo de processamento para blocagens de 8x8 e 16x16 PIXELS é ocasionada pela tempo necessário para o cálculo dos autovetores da matriz de correlação dos vetores-bloco. Basta observar que para blocagens de 8x8 o algoritmo de

JACOBI ou QL terá que trabalhar sobre uma matriz de correlação de dimensões 64x64 PIXELS e aniquilar 4032 (= 64 x 64 – 64) desta matriz enquanto para uma blocagem de 16x16 PIXELS o algoritmo terá que aniquilar 65.280 posições da matriz de correlação.

7 CONCLUSÕES E RECOMENDAÇÕES

Pelos resultados obtidos nos experimentos, concluímos que a Transformada KL é uma ferramenta poderosa em aplicações envolvendo a compressão de imagens, e não apenas nas aplicações de identificação de imagens onde a sua utilização já se tornou um padrão. O fato de utilizar uma base de representação que minimiza o Erro Quadrático Médio (MSE) entre as imagens original e reconstruída a torna um poderoso instrumento na área da compressão de imagens.

Utilizando-se a base ajustada a cada bloco de imagem, se pôde obter com a KLT a melhor qualidade (PSNR) e a melhor Razão de Compressão (CR) da imagem do que a obtida com a DCT e DWT.

Utilizando-se uma base única para todos os blocos de imagem, a KLT permite obter a melhor qualidade (PSNR) e a melhor Razão de Compressão (CR) da imagem do que a obtida com a DCT e DWT, quanto maior for o tamanho do bloco de imagem utilizado no processamento.

O tempo de processamento e o espaço necessário para armazenamento da base KLT são os dois principais fatores a serem reduzidos no processo de cálculo a fim de se poder utilizar a KLT nas aplicações em tempo real e sem aumentar o tamanho do arquivo final compactado (que acaba reduzindo a CR total com a inclusão da base KLT).

A seguir apresenta-se um conjunto de observações e sugestões para aprimoramento e para futura pesquisa sobre a KLT:

a) O trabalho poderá ser ampliado com a implementação dos vários algoritmos sugeridos nos artigos (particularmente os apresentados em (HAYKIN,1999), (GOYAL,1996) e (CASTRO,1996)), buscando melhorias nos três parâmetros (PSNR,CR e Tempo de Processamento) e nas mesmas condições da avaliação realizada neste trabalho.
b) O trabalho poderá ser ampliado com a implementação de outras técnicas de cálculo analítico dos autovetores e autovalores, como o Algoritmo SVD (*SINGLE VALUE DECOMPOSITION*), Algoritmo QR, etc, buscando reduzir o tempo de processamento da KLT.
c) Poderá ser estudado uma implementação mais eficiente do Algoritmo de *HUFFMAN*, buscando reduzir o tamanho das tabelas de código e aumentar a Razão de Compressão que pode ser obtida no arquivo final compactado.

d) A aplicação de outras imagens padrão para teste, com características estatísticas diferentes, deve ser objeto de implementação futura. Deve ser aplicado o procedimento estatístico mais rigoroso para os testes realizados com o conjunto de imagens a fim de poder generalizar as conclusões observadas neste trabalho.

e) A ampliação dos testes realizados neste trabalho para incluir blocos 32x32 e 64x64 pode ser realizada futuramente para comprovação das tendências observadas nos resultados atuais.

f) Um estudo futuro poderá aprofundar a análise sobre a existência de autovetores *degenerados* (que são combinação linear de outros) e que retardam a convergência do *Algoritmo de JACOBI*.

g) Os testes realizados neste trabalho poderão ser repetidos com a utilização dos coeficientes como inteiros de 2 BYTES (eles foram implementados como inteiros de 4 BYTES), e verificar os resultados.

h) A utilização da *KLT* em conjunto com outras transformadas em geral aumenta a performance do conjunto, na medida em que esta transformada proporciona a descorrelação do sinal de entrada considerado. Esse aspecto não foi explorado neste trabalho mas foi observado em alguns artigos mencionados na bibliografia e seu estudo poderia ser aprofundado no futuro.

i) Pela observação dos artigos pesquisados, as técnicas adaptativas demonstram apresentar vantagens, na medida em que permitem a adaptação dos autovetores às características estatísticas dos vetores-bloco da imagem. Esse é um dos pontos importantes a ser aprofundado num trabalho futuro.

j) Conforme descrito nos artigos pesquisados, os algoritmos recursivos de estimação da Transformada KL, utilizando Redes Neurais, destacam-se sobre os procedimentos algébricos de cálculo direto, pela simplicidade e rapidez na obtenção dos resultados (convergência). Esse estudo deve merecer uma atenção especial, particularmente com relação ao tempo de aprendizado necessário para o algoritmo atingir a convergência necessária.

Com relação as principais contribuições deste trabalho, destacam-se:

- A utilização de três critérios de comparação (CR, PSNR e tempo real), o que amplia o universo de abordagem do problema;
- a comparação com a DCT e a DWT, inexistente em alguns outros artigos;
- a descrição e implementação de 2 técnicas determinísticas de cálculo dos autovetores, não mencionadas na maioria dos artigos pesquisados;

- a realização de comparações baseadas no mesmo número de coeficientes da transformada, também ausente em alguns outros trabalhos;
- a disponibilização de um programa experimental, construído especialmente para esse trabalho e colocado a disposição dos pesquisadores como base para a implementação de todas os trabalhos futuros recomendados.

8 REFERÊNCIAS

ANTON H.; RORRES C. **Álgebra Linear com Aplicações**. 8. ed. Porto Alegre: Bookman, 2001. 572 p. ISBN: 85-7307-847-2.

BENTLEY P.M.; McDONNELL J.T.E. Wavelet Transforms: an Introduction. **Electronics & Communications Engineering Journal**, [S.L.], v. 6, n. 4, p. 175-186, 1994.

BREAZU, M.; TODEREAN, G.; VOLOVICI, D.; IRIDON, M. Speeding Up Fractal Compression by Working in Karhunen-Loeve Transform Space. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, 1999, Washington, DC. **Proceedings...** IEEE, 1999. v. 4, p. 2694-2697.

BREAZU, M.; BEGGS, B.J.; TODEREAN, G.; MIHU, I.P. Merging the Transform Step and the Quantization Step for Karhunen-Loève Transform based Image Compression. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, 2000, Como, Itália. **Proceedings...** IEEE, 2000. v. 5, p. 483-488.

BRONSON, R. **Matrix Methods**. 2. ed. New Jersey: Academic Press, 1991. 503 p. ISBN: 0-12-135251-X.

BROWN, J. L. J. Mean-Square Truncation Error in Series Expansions of Random Functions. **J. Soc. Indust. Appl. Math.**, v. 8, n. 1, p. 18-32, 1960.

CASTRO, M. C. F. **Algoritmo Hebbiano Generalizado para Extração dos Componentes Principais de um Conjunto de Dados no Domínio Complexo**. Porto Alegre, 1996. 121 p. Dissertação de Mestrado – Programa de Pós-Graduação em Engenharia, Pontifícia Universidade Católica do Rio Grande do Sul.

CHEN, C. T. **Linear System Theory and Design.** 3. ed. New York: Oxforrd University Press, 1999. 334 p. ISBN: 0-19-511777-8.

DAUBECHIES, I. Orthonormal bases of compactly supported wavelets. **Communications on Pure and Applied Mathematics**, v. 41, p. 909-996, 1988.

DIAMANTARAS, K.I.; STRINTZIS, M. G. Optimal Transform Coding in the Presense of Quantization Noise. **IEEE Transactions on Image Processing**, v. 8, n.11, p. 1508-1515, nov. 1999.

DONY, R.D.; HAYKIN, S. Optimally Adaptive Transform Coding. **IEEE Transactions on Image Processing**, v. 4, n. 10, p. 1358-1370, oct. 1995.

EFFROS, M.; CHOU, P.A. Weighted Universal Transform Coding: Universal Image Compression with the Karhunen-Loève Transform. In: INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 1995, Washington, DC. **Proceedings...** IEEE, 1995. v. 2, p. 61-64.

EPSTEIN, B.R.; HINGORANI, R.; SHAPIRO, J.M.; Czigler, M. Multispectral Image Compression by Wavelet/Karhunen-Loève Transformation. In: GEOSCIENCE AND

REMOTE SENSING SYMPOSIUM, 1992. **Procedings...** IEEE. p.672–674. Disponível em: <http://ieeexplore.ieee.org/xplore/Dynwel.jsp>. Acessado em 14/12/2001.

FENG, H.; EFFROS, M. Separable Karhunen-Loève Transforms for the Weighted Universal Transform Coding Algorithm. In: INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 1999, Phoenix, AZ. **Procedings...** IEEE, 1999. v. 5, p. 2435-2438.

GANTMACHER, F. R. **The Theory of Matrices**. 4. ed. New York: Chelsea, 1977. v. 1, 374 p. ISBN: 0-8284-0131-4.

GANTMACHER, F. R. **The Theory of Matrices**. 3. ed. New York: Chelsea, 1974. v. 2, 276 p. ISBN: 0-8284-0133-0.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização Combinatória e Programação Linear, Modelos e Algoritmos**. 3. ed. Rio de Janeiro: Campus, 2000. 649 p. ISBN: 85-352-0541-1.

GOLUB, G. H.; LOAN, C. F. V. **Matrix Computations**. 3. ed. Baltimore and London: The Johns Hopkins University Press., 1996. 694 p. ISBN: 0-8018-5414-8.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing.** 3. ed. Massachusetts: Addison-Wesley, 1993. 716 p. ISBN: 0-201-50803-6.

GOYAL, K. V.; ZHUANG, J.; VETTERLI, M.; CHAN, C. Transform Coding using Adaptive Bases and Quantization. In: INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 1996, Lausanne, Suiça. **Proceedings...** IEEE, 1996. v. 1, p. 365-368.

GROSSMANN, A.; MORLET, J. Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape. **SIAM J. Math. Anal.**, v. 15, n. 4, p. 733-736, 1984.

HAYKIN, S. **Neural Networks**. 2. ed. New Jersey: Prentice Hall, 1999. 842 p. ISBN: 0-13-273350-1.

HEGT, H.A.; HAYE, R.J.; KHAN, N.A. A High Performance License Plate Recognision System. In: INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 1998, San Diego, CA. **International Conference on Systems, Man and Cybernetics**. IEEE, 1998. v. 5, p. 4357-4362.

HORN, R. A.; JOHNSON, C. R. **Matrix Analysis**. New York: Cambridge University Press, 1985. 561 p. ISBN: 0-521-38632-2.

HOTELLING, H. Análysis of a Complex of Statistical Variables into Principal Components. **J. Educ. Psychol.**, Columbia University, v. 24, p. 417-441 e 498-520, 1933.

HUANG, Y.; SCHULTHEISS, P. M. Block Quantization of Correlated Gaussian Random Variables. **IEEE Transactions on Communications Systems**, v. CS-11, p. 289-296, 1963.

JAMSA, K.; KLANDER, L. **Programando em C/C++ A Bíblia.** São Paulo: Makron Books, 1999. 1012 p. ISBN: 85.346.1025-8.

KIRBY, M. Low-dimensional Processing of Still and Moving Images. In: CONFERENCE RECORD OF THE TWENTY-SIXTH ASILOMAR CONFERENCE ON SIGNALS,

SYSTEMS AND COMPUTERS, 1992, Pacific Grove, CA. **Proceedings...** IEEE, 1992. v. 2, p. 1026-1030.

KRAMER, H. P.; MATHEWS, M. V. A Linear Coding for Transmiting a Set of Correlated Signals. **IRE Trans. Info. Theory**, v. IT-2, p. 41-46, 1956.

LEU, S.L.; REED, I.S. An improved JPEG image coder using the adaptive fast approximate Karhunen-Loeve transform (AKLT). In: INTERNATIONAL SYMPOSIUM ON SPEECH, IMAGE PROCESSING AND NEURAL NETWORKS, 1994, Hong Kong. **Proceedings...** IEEE, 1994. v. 1, p. 160-163.

LEW, M.S.; HUANG, T.S. Image Compression and Matching, In: INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 1994, Austin, TX. **Proceedings...** IEEE, 1994. v. 2, p. 720–724.

LEVY, A.; LINDENBAUM, M**.** Sequential Karhunen-Loève Basis Extraction and its Application to Images. In: INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 1998, Chicago, IL. **Proceedings...** IEEE, 1998. v. 2 , p. 456-460.

LINDQUIST, C. S. **Adaptive & Digital Signal Processing with Digital Filtering Applications.** 10. ed. Miami: Steward, 1989. v. 2, 845 p. ISBN: 0-917144-03-1.

MALINA, W. Two-Parameter Fisher Criterion. **IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics**, v. 31, n. 4, p. 629–636, Aug. 2001.

MALLAT, S. A Theory of multiresolution signal decomposition: the wavelet representation. **IEEE Transaction on Pattern Analysis and Machine Intelligence**, v. 11, n. 7, p. 674-693, July 1989.

MELLO, R.N.B. **Relatório Complementar: Estudo Comparativo da Transformada KARHUNEN-LOÈVE na Compressão de Imagens**. Disponível em: <http://www.lapsi.eletro.ufrgs.br>. Acessado em 11/06/2003.

MURRAY, JAMES D; VANRYPER, WILLIAM. **Encyclopedia of Graphics File Formats**. Sebastopol, CA: O'Reilly Associates, 1994. 894 p. ISBN: 1-56592-058-9.

NASIOPOULOS, P.; YEDLIN, M.; WARD, R.K. A High Performance Fixed-length Compression Method using the Karhunen-Loève Transform. **IEEE Transactions on Consumer Electronics**, v. 41, n. 4, p. 1189–1196, nov. 1995.

OGAWA, H. Karhunen-Loève Subspace. In: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 11., 1992, Hague, Países Baixos. **Proceedings...** IEEE, 1992. v. 2, p. 75–78.

OLIVEIRA, P. R. et al. Techniques for Image Compression: a Comparative Analysis. In: SYMPOSIUM ON NEURAL NETWORKS, 6., 2000, Rio de Janeiro, RJ. **Procedings...** IEEE, 2000. p. 249–254.

PRESS, W. H.; TEUKOLSKY, S. A.; VETTERLING, W. T.; FLANNERY,B.P. **Numerical Recipes in C - The Art of Scientific Computing**. 2. ed. New York: Cambridge University, 1994. 994 p. ISBN: 0-521-43108-5.

RAGAB, A. S.; MOHAMED, A. S. A.; HAMID, M. S. Efficiency of Analytical Transforms for Image Compression. In: NATIONAL RADIO SCIENCE CONFERENCE, 15., 1998, Helvan, Cairo, Egito. **Proceedings...** IEEE, 1998. p. B16/1 – B16/10.

SAGHRI, J.A.; TESCHEZ, A.G.; REAGAN, J.T. Three-dimensional Transform Coding of Multispectral Data. In: ASILOMAR CONFERENCE ON SIGNALS, SYSTEMS AND COMPUTERS, 27., 1993, Pacific Grove, CA. **Conference Record...** IEEE, 1993. v. 2, p. 1342-1346.

SAGHRI, J.A.; TESCHER, A.G.; REAGAN, J.T. Practical Transform Coding of Multispectral Imagery. **IEEE Signal Processing Magazine**, v.12, n. 1 , p. 32–43, Jan 1995.

SEALES, W. B.; LUMPP, J.E.Jr.; YUAN, C.J. Model-Based Tracking in Compressed Video. In: AEROSPACE CONFERENCE, 1997, Aspen, CO. **Proceedings...** IEEE, 1997. v. 1, p. 157-168.

SHANMUGAN, K. S.; BREIPOHL, A. M. **Randon Signals: Detection, Estimation and Data Analysis**. New York: John Wiley, 1988. 664 p.

SHANNON, C. E. A Matematical Theory of Communication. **The Bell System Technical Journal**, v. 27, p. 379-423 e 623-656, July/Oct. 1948.

SHEN, S.S.; LINDGREN, J.E; PAYTON, P.M. Effects of Multispectral Compression on Machine Exploitation. In: ASILOMAR CONFERENCE ON SIGNALS, SYSTEMS AND COMPUTERS, 27., 1993, Pacific Grove, CA. **Conference Record...** IEEE, 1993. v. 2, p. 1352-1356.

SIDBA Standard Image Database.Laboratório de Análise de Sinais e Percepção de Máquina, Departamento de Engenharia Elétrica da Universidade Estadual de Ohio. Disponível em <http://sampl.eng.ohio-state.edu/~sampl/ database.htm> Acesso em 08 fev. 2003.

TINTRUP, F.; NATALE, F.D.;GIUSTO, D. Compression Algorithms for Classification of Remotely Sensed Images. In: INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 1998, Seattle, WA. **Proceedings...** IEEE, 1998. v. 5, p. 2565-2568.

UENOHARA, M.; KANADE, T. Optimal Approximation of Uniformly Rotated Images: Relationship between Karhunen-Loève Expansion and Discrete Cosine Transform. **IEEE Transactions on Image Processing,** v. 7, n. 1, p. 116-119, Jan. 1998.

UMBAUGH, S. E.; MOSS, R. H.; STOECKER, W.V.; HANCE, G. A. Automatic Color Segmentation Algorithms with Application to Skin Tumor Feature Identification. In: ENGINEERING IN MEDICINE AND BIOLOGY. **Proceedings...** IEEE, 1993. v. 12, n. 3, p. 75-82. Disponível em: <http://ieeexplore.ieee.org/xplore/Dynwel.jsp>. Acessado em 16/12/2001.

WALDEMAR, P.; RAMSTAD, T.A. Hybrid KLT-SVD Image Compression. In: INTERNATIONAL CONFERENCE ON ACOUSTICS SPEECH AND SIGNAL PROCESSING, 1997, Munich, Alemanha. **Proceedings...** IEEE, 1997. v. 4, p.2713–2716.

WANG. C. N.; CHIANG, T.; LIU, C. M.; LEE, H. J. Improved MPEG-4 Visual Texture Coding using Double Transform Coding. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 2001, Sydney, NSW, Austrália. **Symposium...** IEEE, 2001. v. 5, p. 227-230.

YAMASHITA, Y. Image Compression by Weighted Karhunen-Loève Transform. In: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 13., 1996, Viena, Áustria. **Proceedings...** IEEE, 1996. v. 2 , p. 636-640.

APÊNDICE A:

Descrição do Programa

9 APÊNDICE A: PROGRAMA EXPERIMENTAL

Para a realização da etapa experimental deste trabalho, foi desenvolvido e implementado um programa de computador em linguagem C (Padrão ANSI C – Dez/1989), que foi especialmente construído para realizar a comparação entre a Transformada KARHUNEN-LOÈVE (KLT), a Transformada Discreta do Coseno (DCT) e a Transformada WAVELET Discreta (DWT). Este programa (código fonte, descrição e fluxogramas), bem como outras informações complementares deste trabalho foram incluídas num relatório publicado em (MELLO,2003).

APÊNDICE B:

Testes de Quantização

10APÊNDICE B: TESTES DE QUANTIZAÇÃO

A quantização dos coeficientes é uma etapa importante do processo de compressão de imagens. Neste trabalho utilizou-se uma quantização dos coeficientes baseada no simples arredondamento dos mesmos para *0* (*zero*) casas decimais, e realizaram-se alguns testes que são descritos a seguir.

A quantização dos coeficientes foi testada no programa utilizando-se o arquivo de imagem de teste (“padrao4x4256.bmp”) mostrada na Figura 130, com tamanho de 4 PIXELS por 4 PIXELS (4x4), com todo cálculo dos coeficientes KLT realizado manualmente, e também com um arquivo GRAYSCALE (“tigra.bmp”) mostrado na Figura 131(a), de tamanho de 640 PIXELS por 480 PIXELS, calculando o PSNR entre as imagens originais e aquelas reconstituídas a partir dos coeficientes KLT quantizados e não quantizados.

228	111	211	100
240	80	191	103
175	0	255	15
118	220	237	116

Figura 130 – Imagem de Teste (“padrao4x4256.bmp”)

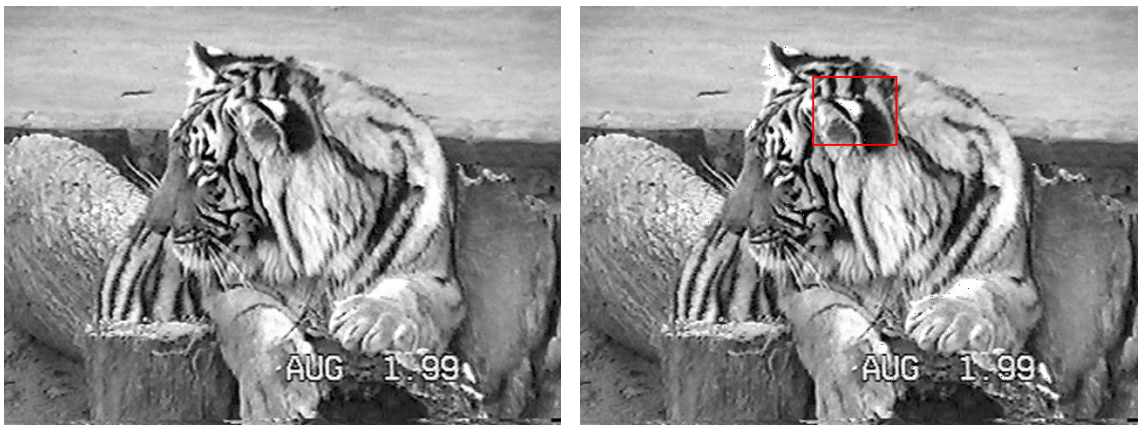
Os resultados inicialmente obtidos estão representados na Tabela 104.

Tabela 104 – Comparação de PSNR após Quantização

ARQUIVOS	PSNR (em dB)	
	Sem Quantização	Com Quantização
“padrao4x4256.bmp”	Arquivos idênticos	60,172003 1 PIXEL Alterado (6,25 % da imagem)
“tigra.bmp”	Arquivos idênticos	38,113150 28.385 PIXELS Alterados (9,23 % da imagem)

O valor de PSNR = 60,172003 para o arquivo “padrao4x4256” deveu-se a uma diferença de 1 único PIXEL que teve seu valor alterado de 220 na imagem original para 221 na imagem reconstruída após a quantização.

Em termos de qualidade de imagem, do ponto de vista visual, o valor de PSNR de 38,113150 dB, obtido na reconstrução da imagem “tigra.bmp”, não causa uma deformação visualmente perceptível na mesma como se pode observar na Figura 131.



(a) Imagem Original **(b) Imagem Reconstituída**

Figura 131 – Imagem “tigra.bmp” (Tamanho 640 x 480 PIXELS)

Ampliando, porém, a área marcada na Figura 131(b), observou-se uma série de pequenos pontos brancos onde deveriam ser pretos e vice-versa, como se percebe mais claramente na Figura 132.

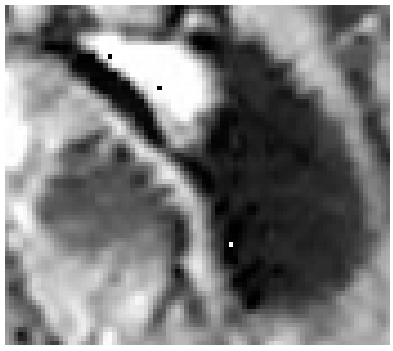


Figura 132 – Erro na Imagem devido a Quantização dos Componentes KL

Após análise do problema constatou-se que essa aparente inversão correspondia a pontos que possuíam valores de PIXEL de 0 (PRETO) ou de 255 (BRANCO) na imagem original e que na reconstrução da imagem, devido a quantização dos componentes KL, transformaram-se em valores menores ou iguais -0,5 (que se tornaram brancos pelo arredondamento estatístico utilizado) ou maiores ou iguais a 255,5 (que se tornaram pretos

pelo arredondamento estatístico utilizado) gerando a aparente inversão no valor do código ASCII correspondente aos mesmos.

O problema foi corrigido acrescentando, na rotina de reconstrução da imagem original da KLT, um segmento de programa para tratamento específico para esta situação (para valores menores do que –0,5 é colocado o valor zero e para valores maiores do que +255,5 é colocado o valor 255). A Tabela 105 mostra os novos valores obtidos após essa correção.

Tabela 105 – Comparação de PSNR após Quantização (Corrigida)

ARQUIVOS	PSNR (em dB)	
	Sem Quantização	Com Quantização
“padrao4x4256.bmp”	Arquivos idênticos	60,172003 1 PIXEL Alterado (6,25 % da imagem)
“tigra.bmp”	Arquivos idênticos	58,481324 28.338 PIXELS Alterados (9,23 % da imagem)

Apesar da imagem de teste (“padrao4x4256”) possuir os valores de PIXEL 0 (PRETO) e 255 (BRANCO) a quantização dos componentes KL associado ao reduzido número de pontos da imagem de teste não foram suficientes para ocasionar uma diferença de valor fora do intervalo acima mencionado e, por isso, o valor do PSNR não se alterou.

Já na imagem “tigra.bmp” a correção implementada no programa reduziu em 47 o número de PIXELS alterados pela quantização, número correspondente aos PIXELS responsáveis pelas imperfeições mostradas na Figura 132 e que foram eliminadas pela correção.

O número de PIXELS alterados (bem como o PSNR) variará dependendo do tipo e do tamanho da imagem processada. Essa alteração, porém, em geral, não difere de 1 unidade do valor do PIXEL. Para o caso da imagem “tigra.bmp”, por exemplo, que possui 307.200 pontos (= 640 x 480 PIXELS), nenhum ponto da imagem reconstruído excedeu a uma unidade de PIXEL.

Cabe observar também, neste ponto da análise, que os coeficientes foram arredondados para 0 (zero) casas decimais. Se arredondados para 1 casa decimal, a imagem reconstruída já fica idêntica à original porém a tabela de compressão de HUFFMAN cresce de forma tão significativa que o arquivo final compactado ultrapassa o tamanho do arquivo original sem compactação. Portanto, em todos os testes realizados neste trabalho a

quantização foi realizada através do simples arredondamento dos coeficientes para 0 (zero) casas decimais, o que equivale a utilizar a parte inteira do coeficiente, desprezando a parte fracionária.

APÊNDICE C:

Testes do Programa

11 APÊNDICE C: TESTES DO PROGRAMA

Nos testes realizados no programa implementado foi utilizada a imagem de teste "*padrao4x4256.bmp*" e comparados passo a passo, manualmente, os seguintes resultados (valores) intermediários e o resultado final (utilizando blocos de imagem de tamanho 2 x 2 PIXELS), para cada uma das matrizes abaixo:

- A Matriz Média, obtida pela comparação com o cálculo manual da média de cada uma das dimensões da matriz original;
- a Matriz Imagem Média, obtida pelo cálculo manual dos desvios da Matriz Imagem Original com relação a Matriz Média;
- a Matriz de Covariância de cada vetor (bloco) da imagem, obtida pelo cálculo manual;
- a Matriz de Covariância Média, formada pela média aritmética das Matrizes de Covariância de cada vetor (bloco de imagem), obtida pelo cálculo manual;
- a Matriz dos Autovetores, testada de forma indireta através da verificação da igualdade: $\mathbf{C}\ \mathbf{v} = \mathbf{v}\ \lambda$, onde **C** é a Matriz de Covariância Média da Imagem, **v** é a Matriz dos Autovetores da Matriz de Covariância e λ é a Matriz dos Autovalores da Matriz de Covariância (matriz diagonal formada pelos autovalores na diagonal principal);
- a Matriz dos novos componentes dos PIXELS de imagem, testada de forma indireta através da verificação da igualdade: $\mathbf{x} = \mathbf{v}\ \mathbf{x'}$, onde **x** é a matriz imagem média, **v** é a matriz dos autovetores da Matriz de Covariância Média da imagem e **x'** é a matriz dos componentes da imagem na base formada por esses autovetores;
- a Matriz Imagem Média Recriada, composta a partir dos novos componentes calculados e da base dos autovetores;
- a Matriz Imagem Recriada, composta a partir da Matriz Imagem Média Recriada e da Matriz Média, que foi então comparada com a Matriz Imagem Original, através do cálculo do PSNR das duas imagens.

Após a verificação acima, confirmou-se que cada pixel da Matriz Imagem Recriada era idêntico ao da Matriz Imagem Original. Realizou-se a seguir testes com uma imagem do de 640 x 480 mostrada na Figura 133, para blocos de 2x2, 4x4, 16x16 e 32x32, calculando, para cada bloco utilizado no teste, o PSNR entre os arquivos da imagem original e da imagem recriada. Neste teste utilizou-se toda a precisão dos coeficientes sem arredondamentos.

Figura 133 – Imagem 640x480 PIXELS utilizada no teste do Programa

Todos os testes demonstraram a igualdade entre a imagem original e a recriada a partir da base dos autovetores da matriz de covariância da Matriz Imagem Média.

Outra verificação realizada foi a comprovação de que o valor máximo dos componentes KL da imagem transformada (PixelKLMax) se relaciona com o valor máximo dos componentes da imagem original (PixelOrigMax) através da seguinte expressão:

$$\text{PixelKLMax} \leq \text{PixelOrigMax}\sqrt{\text{dim}} \qquad \textbf{(Eq. 43)}$$

onde “dim” é a dimensão do vetores correspondentes aos blocos em que a imagem é decomposta.

Comprovou-se, por exemplo, para a imagem de teste (“tigra.bmp”) e para um tamanho de bloco de 2x2 (dim=4), que o valor máximo dos componentes da imagem média é PixelOrigMax =110,889 e, portanto, que o valor máximo que os componentes KL poderiam assumir corresponde, assim, a PixelKLMax ≤ 221,778. Constatou-se esse cálculo pela

varredura da matriz de componentes KL e que o maior componente KL correspondeu ao valor 221, comprovando a correção do cálculo.

Essa comprovação está totalmente aderente a teoria descrita para a Transformada KL porque se os autovetores da matriz de correlação dos vetores-bloco se orientam nas direções de maior variância, é razoável admitir que o máximo valor possível que um componente KL poderia assumir corresponda a distância euclidiana de um hipotético vetor-bloco que tivesse todos os seus componentes iguais a PixelOrigMax, o que nos leva à (Eq. 43).

APÊNDICE D:

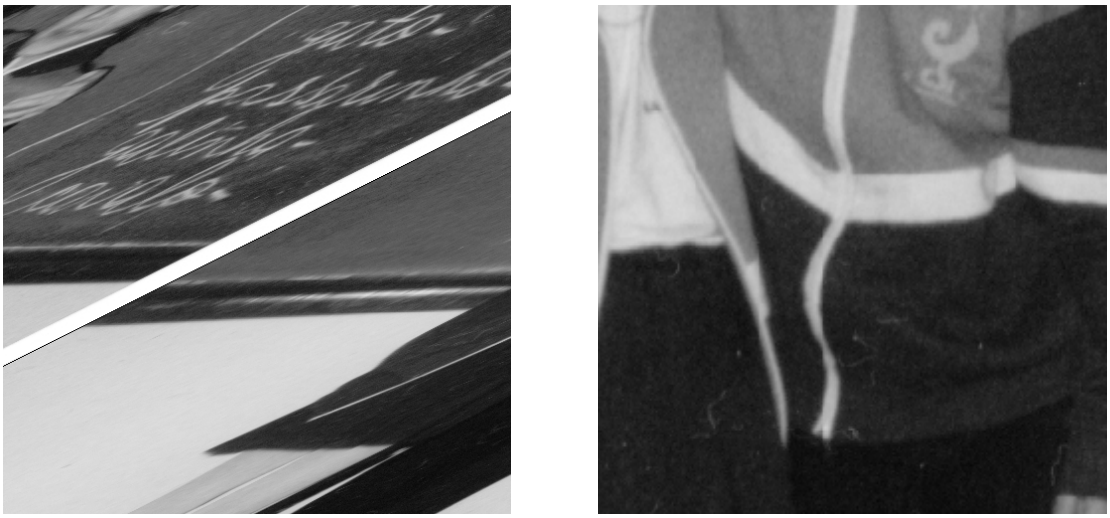
Alinhamento dos Arquivos BITMAP

12 APÊNDICE D: ALINHAMENTO DOS ARQUIVOS BITMAP

A Figura 134 apresenta uma imagem original com dimensões de 1194 PIXELS de LARGURA por 1794 PIXELS de ALTURA, obtida através da reprodução (SCANNER) de uma fotografia comum.

Figura 134 – Imagem Original (1194 x 1794)

A Figura 135(a) mostra a imagem obtida pela redução da parte central desta imagem para outra de 512 x 512 PIXELS, sem levar em consideração o alinhamento da imagem original gravada, isto é, lendo a imagem como se os 2 BYTES a mais acrescentados pelo SOFTWARE não existissem (notar a deformação total da janela assim criada). A Figura 135 (b) mostra a imagem obtida levando em consideração o alinhamento da imagem original gravada (notar a correta criação da janela).



(a) Redução sem Alinhamento **(b) Redução com Alinhamento**

Figura 135 – Redução da Imagem Original para uma Imagem de 512 x 512 PIXELS

A falta de cuidado no tratamento do alinhamento da imagem pode também levar a conclusões equivocadas na comparação de 2 imagens. Considerando, por exemplo, as imagens da Figura 136, observa-se que elas parecem idênticas. Ao calcular-se o PSNR destas 2 imagens, utilizando uma rotina desenvolvida neste trabalho e que lê os arquivos BYTE a BYTE sem levar em consideração o alinhamento de BYTES, o resultado também indica que as 2 imagens são realmente idênticas! Ao se ampliar, entretanto, as primeiras linhas da imagem recriada na Figura 136(b), pode-se notar na Figura 137 uma diferença significativa entre as duas imagens (com mais de 2 linhas de diferença!). A razão dessa aparente contradição ocorre quando o programa inicialmente desconsidera o alinhamento de 4 BYTES na leitura do arquivo original e posteriormente realiza a gravação do arquivo de saída também sem levar em consideração esse alinhamento. Neste caso, os 2 BYTES adicionais existentes no arquivo gravado são lidos como se fossem parte da imagem e uma parte inicial da imagem é perdida (porque no formato de arquivo BITMAP a imagem é gravada linha a linha iniciando pela última linha, sempre no sentido da esquerda para a direita, e terminando na primeira linha). A partir daí, se o arquivo for novamente gravado, sem considerar o alinhamento, será feita a gravação com esses 2 BYTES adicionais lidos anteriormente, mas faltando a parte inicial da imagem. A partir daí, quando se utiliza qualquer SOFTWARE de edição de imagens para ler o arquivo original e o copiado na gravação sem o cuidado do alinhamento (arquivo copiado), serão observadas as imagens reproduzidas na Figura 137(a) e (b),

respectivamente. Para o caso do arquivo copiado, o SOFTWARE retira sempre os 2 BYTES adicionais que foram gravados como se imagem fossem e por isso a imagem, na sua maior parte, parece igual a original, com exceção do inicio da imagem que não é apresentada porque o arquivo copiado não possui realmente essa parte inicial da imagem.

Quando, entretanto, se compara as 2 imagens utilizando uma rotina que compara BYTE a BYTE o arquivo original e o arquivo copiado, sem nenhuma preocupação com o alinhamento de BYTES dentro dos arquivos, os BYTES acrescidos dos 2 arquivos (geralmente BYTES 00H) são lidos normalmente na seqüência de BYTES resultando na indicação de "arquivos idênticos" (mesmo sem ter comparado realmente todo o conteúdo real da imagem original).

Como ilustração, mostra-se na (Eq. 44) o relacionamento do total de BYTES da imagem (à direita da igualdade) correspondente ao seu tamanho total (ALTURA x LARGURA), com a linha n correspondente ao ponto em que a imagem parou de ser copiada.

$$\left[1194 - 1194 \text{ x } \text{frac}\left(\frac{(n-1)x2}{1194}\right)\right] + (n-1) \text{ x } 1194 + (n-1) \text{ x } 2 \equiv 1754 \text{ x } 1194 \quad \text{(Eq. 44)}$$

Para n = 1752 a igualdade acima se verifica corretamente (2.094.276=2.094.276). Isso significa que nem todas as 1754 linhas da imagem serão gravadas no arquivo, se não for levado em consideração o alinhamento. O primeiro termo da adição na (Eq. 44) representa a quantidade de BYTES de imagem restantes após descontar os 2 BYTES adicionais por linha que estão presentes nas (n –1) linhas da imagem, e corresponde a 80 BYTES (para n = 1752). O segundo termo da adição corresponde ao total de BYTES da imagem gravados até a penúltima linha (n=1752) e corresponde a 2.090.694 BYTES (1751 linhas completas de imagem de 1194 BYTES cada uma). O terceiro termo da adição representa o total de BYTES do arquivo devido aos 2 BYTES adicionados a cada linha por conseqüência do alinhamento de BYTES e corresponde a 3502 BYTES. Por conseqüência deste último termo é que foram subtraídos 3502 BYTES da imagem original. Como a imagem original possui, na realidade, 1754 linhas, 3502 BYTES equivalem a mais de 2 linhas inteiras de imagem (1194 BYTES x 2 linhas + 1114 BYTES =3502 BYTES) que não são gravadas no arquivo. A Figura 137 mostra a ampliação da área marcada na Figura 137(b), e pode-se verificar claramente a posição onde termina a gravação da imagem (largura = 79+1 = coluna 80 e altura = 2+1 = linha 3). As imagens da Figura 137 foram geradas a partir do SOFTWARE PAINTSHOP PRO que leva em consideração o alinhamento nos arquivos de imagem.

(a) Imagem Original **(b) Imagem Copiada**

Figura 136 – Problemas de Alinhamento (a)Imagem Original e (b)Imagem com Problemas

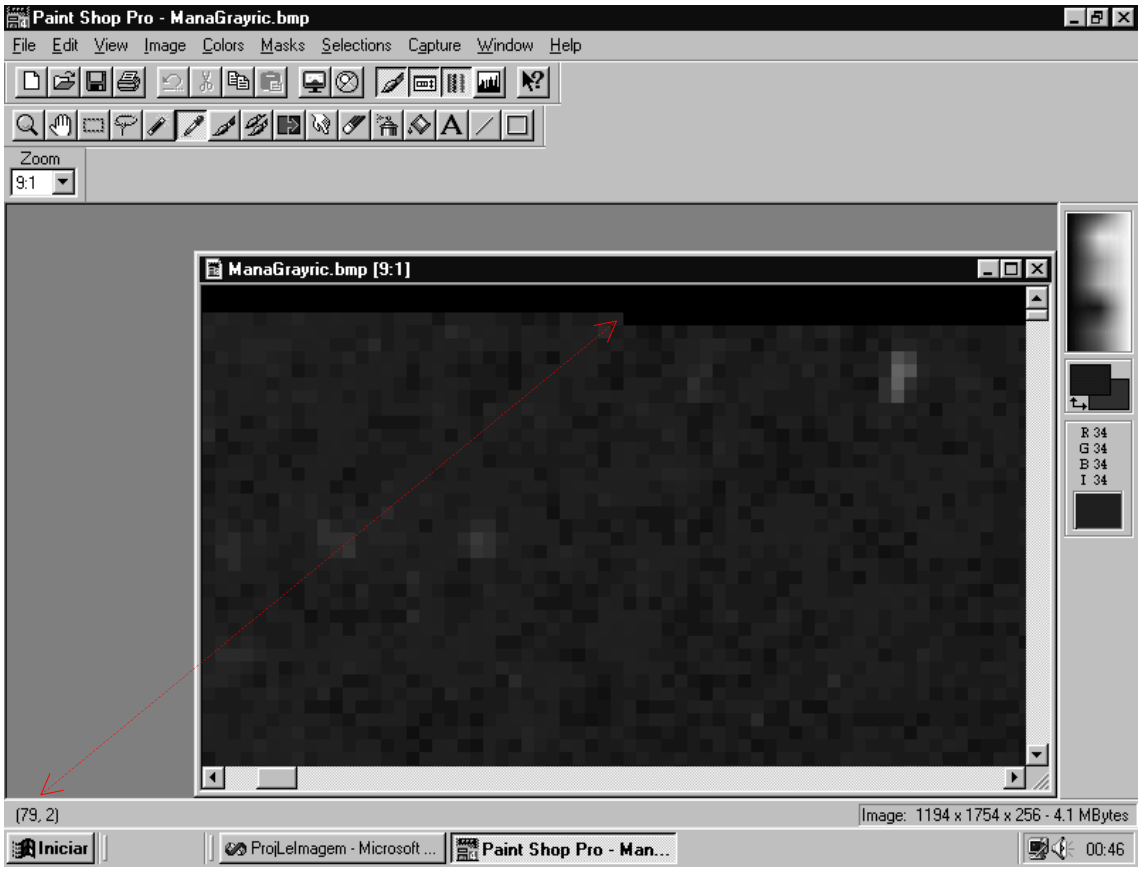


Figura 137 – Ampliação do local com problema de alinhamento

APÊNDICE E:

Efeito da Blocagem na DCT BLOCO

13 APÊNDICE E: EFEITO DA BLOCAGEM NA DCT

Os resultados inferiores da *CR* (COMPRESSION RATIO) obtidos pela DCT quando aplicada a imagem é dividida em blocos (imagem blocada), e mostrados na Figura 19 para um tamanho de bloco 2x2 e na Figura 22 para um bloco de 4x4, foram verificados em detalhe buscando confirmar as causas dessa degradação.

Os procedimentos realizados foram os seguintes:

a) Utilização da imagem LENNA 512x512 com blocagem 2x2 (256 x 256 blocos) e seleção de 4 blocos de amostragem (blocos (1,1), (2,2), (255,255) e (256,256));
b) nova análise do cálculo da Transformada DCT dos blocos no programa comparando os resultados com o realizado manualmente para os 4 blocos de amostra;
c) comparação da Tabela de HUFFMAN para as imagens INTEIRA e BLOCADA;
d) conclusões a partir dos resultados obtidos nos testes.

A "nova análise" mencionada na alínea (b) acima foi realizada sobre as subrotinas *ProcJPGBloco (CopiaImagem_b, FazCosfn, QuantCompWvt1, QuantCompWvt2), CopiaImagem_a,TabFreqMat* e *TabFreqHufWvt,* relacionadas ao cálculo da DCT BLOCO. A partir dos valores dos PIXELS na imagem foram comparados os valores de entrada e saída de cada subrotina com os valores calculados manualmente para cada um dos 4 blocos de amostra selecionados. Nenhum problema ou cálculo incorreto foi constatado nesta parte da análise.

Uma outra hipótese para uma redução na *CR* seria o aumento excessivo do número de entradas na Tabela de HUFFMAN. Ao rodar o programa, entretanto, constata-se que o número de entradas para a DCT INT é de 983 e para a DCT BLOCO é de 1050 (blocos 2x2 com compressão de bloco de 2:1), com uma diferença de apenas 67 entradas. Como cada entrada na Tabela de HUFFMAN representa aproximadamente 2 x 4 x 4 (tamanho do ARRAY x número de ARRAYS x tamanho do LONG INT), as 67 entradas adicionais no máximo poderiam representar um aumento de 2.144 BYTES, o que não ocasionaria a degradação observada no gráfico.

Ocorre que não somente as entradas adicionais mas todas as entradas na Tabela de HUFFMAN possuem uma freqüência de ocorrência, que depende do cálculo dos coeficientes DCT para a imagem inteira ou blocada. Como o cálculo do bloco redistribui as freqüências de

ocorrência dos coeficientes entre mais valores de entrada da tabela, então mais coeficientes são codificados com um código de HUFFMAN de tamanho maior e o efeito final é representado pela degradação observada. Para mostrar esse efeito construiu-se os histogramas da DCT INTEIRA e da DCT BLOCO e comparou-se os resultados.

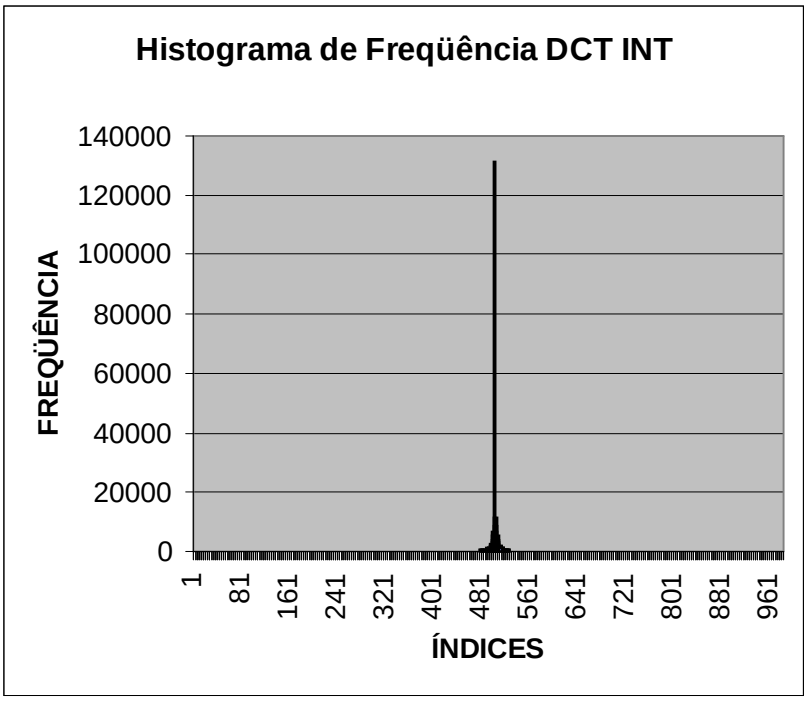


Figura 138 – Histograma LENNA 2x2 (Compressão de Bloco 2:1) para DCT INT

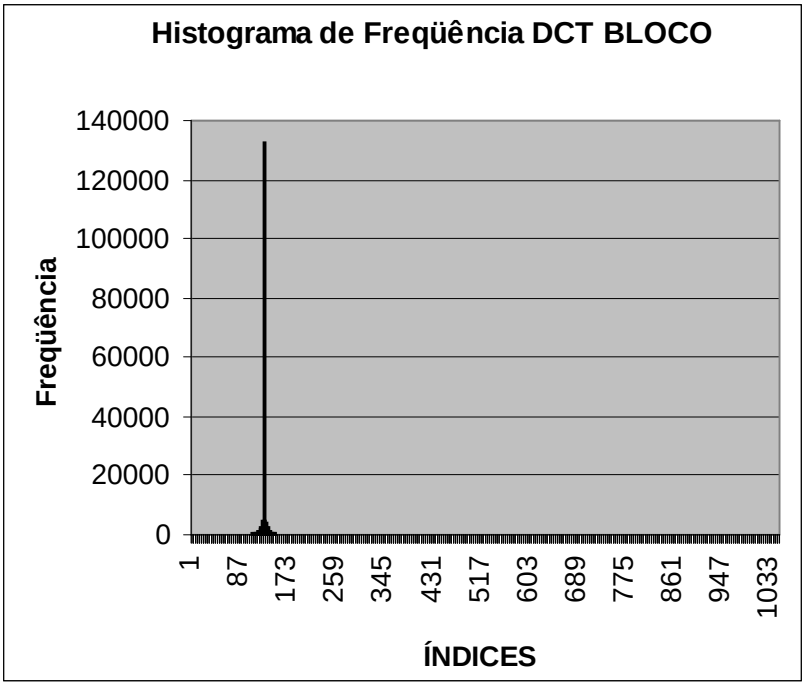


Figura 139 – Histograma LENNA 2x2 (Compressão de Bloco 2:1) para DCT BLOCO

Observa-se, pela comparação do histograma da DCT INT (Figura 138) com o da DCT BLOCO (Figura 139), que ambos possuem o mesmo formato básico, com uma freqüência máxima, para os dois histogramas, em torno de 135.000 pontos, nas posições correspondentes ao valor de coeficiente igual a *0* (zero). O número de entradas na Tabela de HUFFMAN para ambos os gráficos (chamados de índices nas figuras) também são muito próximos em torno de 1.000 entradas. Aparentemente, não haveria motivo para grandes diferenças na *CR*.

Se, entretanto, retira-se do gráfico o coeficiente de valor *0* e plota-se novamente os dois histogramas, observa-se claramente o espalhamento causado pelo cálculo da DCT BLOCO.

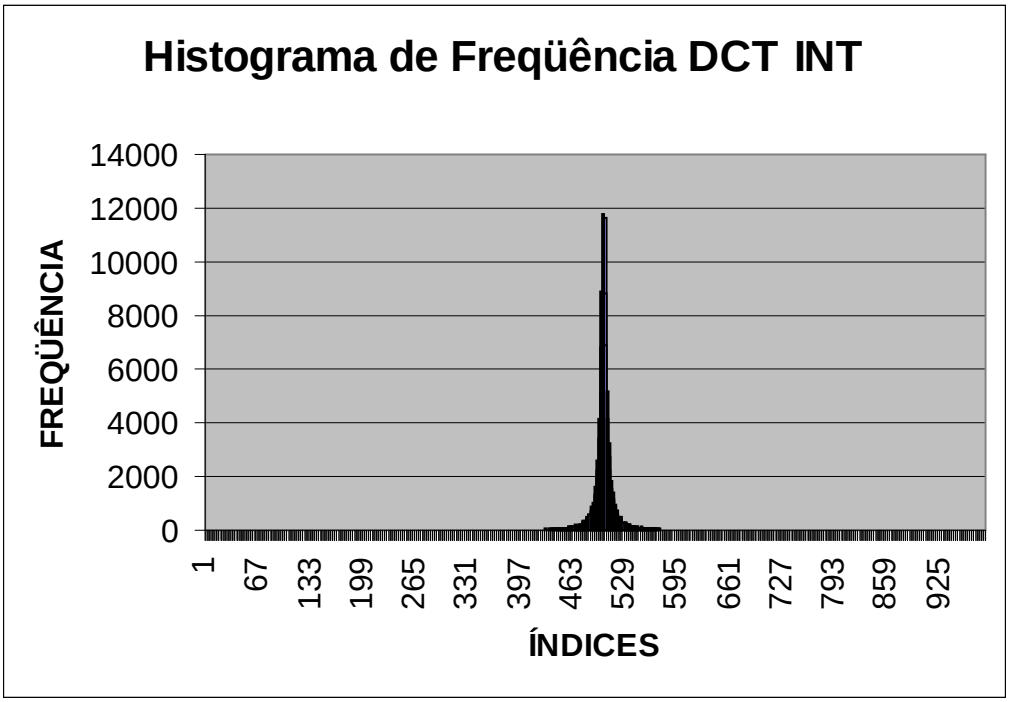


Figura 140 – Histograma para DCT INT sem o coeficiente de valor *0 (zero)*

Na Figura 140 verifica-se uma concentraçao dos coeficientes em torno do coeficiente de valor 0 (*zero*), isto é, a maioria dos coeficientes concentram-se em torno do valor zero, e estes são codificados com um número menor de bits de representação, enquanto a minoria dos coeficientes de valores altos são codificados com uma quantidade de bits maior. O efeito neste caso é o de uma otimização na *CR* em relação à DCT BLOCO.

A Figura 141 mostra o espalhamento da freqüência resultante do cálculo dos coeficientes dentro de cada bloco de imagem e que ocasiona a degradação na *CR* em relação ao cálculo pela DCT INT, porque mais coeficientes de valores elevados (distantes dos valores do pico de freqüência) serão codificados com uma quantidade maior de bits.

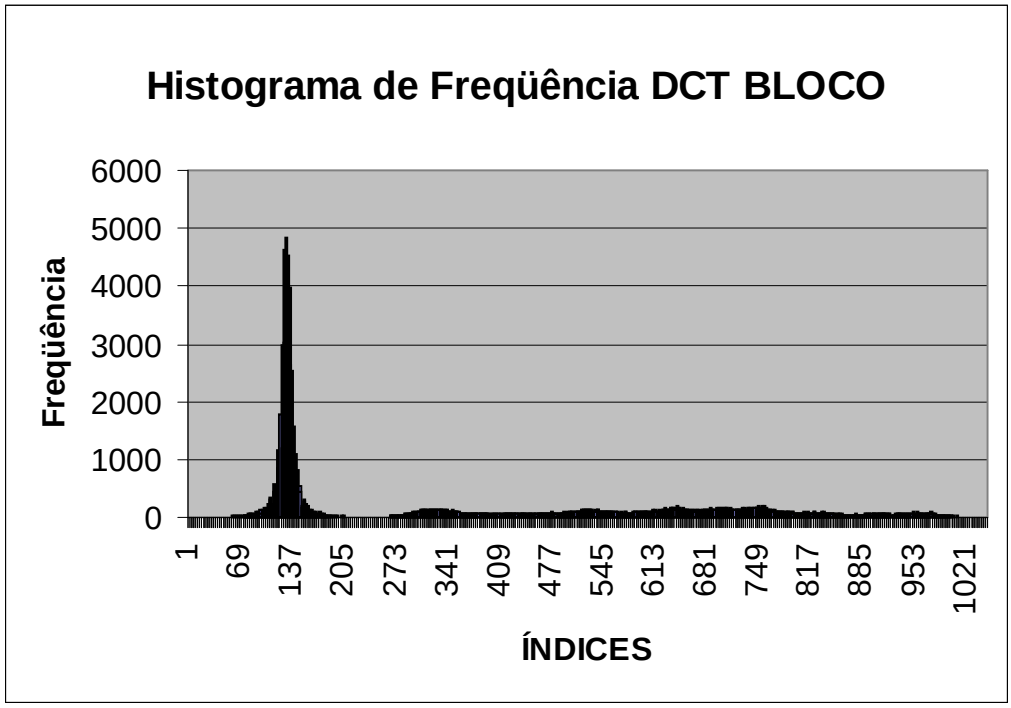


Figura 141 – Histograma para DCT BLOCO sem o coeficiente de valor *0* (*zero*)

Para se ter uma idéia de valores, até o índice 600 do gráfico da DCT INT o valor acumulado de BYTES era de 123.130,40 enquanto para esse mesmo índice no gráfico da DCT BLOCO esse valor era de 116.985,25 BYTES, ou seja, uma valor até inferior à DCT INT. Já no índice 800 a situação se inverte totalmente e para a DCT INT o valor acumulado de BYTES foi de 124.852,40 enquanto para a DCT BLOCO foi de 153.227,38. Vê-se, claramente, que a degradação é ocasionada então pelo próprio cálculo dos coeficientes da DCT BLOCO, que provoca o espalhamento do histograma de freqüência dos mesmos, fazendo com que mais coeficientes sejam codificados com uma maior quantidade de bits, degradando com isso a *CR*.