UNIVERSIDADE DE BRASÍLIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

GERAÇÃO E LEITURA DE QR CODES COLORIDOS

MAX EDUARDO VIZCARRA MELGAR

ORIENTADOR: PhD. ANDERSON CLAYTON ALVES NASCIMENTO

DISSERTAÇÃO DE MESTRADO EM
ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGEE.DM - 519/2013

BRASÍLIA/DF: MARÇO - 2013.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

GERAÇÃO E LEITURA DE QR CODES COLORIDOS

MAX EDUARDO VIZCARRA MELGAR

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

Anderson C A Nascimento

ANDERSON CLAYTON ALVES NASCIMENTO, Dr., ENE/UNB
(ORIENTADOR)

LRAXdeMenezes

LEONARDO RODRIGUES ARAÚJO XAVIER DE MENEZES, Dr., ENE/UNB
(EXAMINADOR INTERNO)

CAMILO CHANG DÓREA, Dr., CIC/UNB
(EXAMINADOR EXTERNO)

Brasília, 21 de março de 2013.

FICHA CATALOGRÁFICA

VIZCARRA, MAX EDUARDO VIZCARRA

Geração e Leitura de QR Codes com Cinco ou mais Cores [Distrito Federal] 2013.

xvii, 79p., 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2013).

Dissertação de Mestrado - Universidade de Brasília.

Faculdade de Tecnologia.

Departamento de Engenharia Elétrica.

1. Colored QR Codes
2. Reed-Solomon
3. Processamento de Imagens
4. RGB

I. ENE/FT/UnB

II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

VIZCARRA, Melgar Max E. (2013). Geração e Leitura de QR Codes Coloridos. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGEE.DM - 519/2013, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 79p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Max Eduardo Vizcarra Melgar.

TÍTULO DA DISSERTAÇÃO DE MESTRADO: Geração e Leitura de QR Codes Coloridos.

GRAU / ANO: Mestre / 2013

Max Eduardo Vizcarra Melgar
QS 5, Rua 100 Bloco B Apt 202, Aguas Claras
71.963-000 Brasília - DF - Brasil.

DEDICATÓRIA

Dedico este trabajo final de maestría a mi madre MSc. Aida Victoria Melgar Santander, mi padre Dr. Max Wilfredo Vizcarra Maldonado, mi hemana y futura médica Julissa Aida Vizcarra Melgar y también con mucho cariño a mi tía Dr. Luz Amanda Melgar Santander. Solo me queda agradecerles por el constante apoyo brindado en esta nueva etapa de mi vida :)

AGRADECIMENTOS

Agradeço aos meus professores Dr. Alexandre Zaghetto e PhD. Anderson Nascimento pelo apoio no crescimento acadêmico que ganhei nesses dois anos de aprendizagem e novas experiências. Aos professores PhD. Anderson Nascimento, Dr. Alexandre Zaghetto, Dr. Camilo Chang Dórea e PhD. Leonardo Rodrigues Araujo Xavier de Menezes que compõem a banca dessa dissertação.

Ao professor Dr. Bruno Macchiavello pela ajuda nos trabalhos publicados, à professora PhD. Mylène Farias pela oportunidade de orientação para futuros trabalhos, ao Departamento de Engenharia Elétrica da Universidade de Brasília e finalmente à CAPES pelo apoio financeiro recebido nesse período.

RESUMO

GERAÇÃO E LEITURA DE QR CODES COLORIDOS

Autor: Max Eduardo Vizcarra Melgar
Orientador: PhD. Anderson Clayton Alves Nascimento
Programa de Pós-graduação em Engenharia Elétrica
Brasília, março de 2013

Colored Quick Response Code ou CQR Code é um código matricial ou código de barra de duas dimensões que utiliza cores com o objetivo de conseguir maior quantidade de armazenamento de dados, os CQR Codes foram desenvolvidos pelo autor dessa dissertação nos anos 2011 e 2012. O CQR Code possui a mesma estrutura física do tradicional *Quick Response Code* ou QR Code. O primeiro modelo do CQR Code armazena o dobro de quantidade de dados e o segundo modelo armazena o triplo de quantidade de dados em comparação à mesma área de impressão do QR Code tradicional. Nessa dissertação será mostrada a geração (codificação) e a leitura (decodificação) de dois novos modelos de CQR Codes no ambiente de simulação iterativo MATLAB. A geração e leitura dos CQR Codes teve como base teórica o padrão internacional *ISO/IEC 18004:2005*. O primeiro modelo de CQR Code conta com cinco cores (branco, preto, vermelho, verde e azul) e armazena exatamente 1024 bits de informação e 3092 bits de redundância. O segundo modelo de CQR Code conta com nove cores (branco, preto, vermelho, verde, azul, ciano, amarelo, magenta e cinza) e armazena exatamente 2048 bits de informação e 4576 bits de redundância. A quantidade de 1024 e 2048 bits de informação que os dois modelos de CQR Codes armazenam foi projetada para futuramente poder armazenar e transmitir protocolos de criptografia desses comprimentos.

ABSTRACT

GENERATION AND RECOGNITION OF COLORED QR CODES

Author: Max Eduardo Vizcarra Melgar
Supervisor: Anderson Clayton Alves Nascimento
Programa de Pós-graduação em Engenharia Elétrica
Brasilia, March of 2013

Colored Quick Response Code or CQR Code is a matrix code or a two-dimensional bar code which uses colors in order to achieve higher data storage capacity, CQR Codes were developed by the author of this document in 2011 and 2012. CQR Code has the same physical structure of the traditional Quick Response Code or QR Code. The first CQR Code model stores twice the data amount as compared with the traditional QR Code. The second CQR Code model stores triple the data amount as compared with the traditional QR Code. In this document we show the generation (coding) and recognition (decoding) of two proposed CQR Code models created in the MATLAB simulation environment. The generation and recognition of the CQR Codes has been developed based on the document ISO/IEC 18004:2005. The first model of CQR Code uses five colors (white, black, red, green and blue) and stores exactly 1024 information bits and 3392 redundancy bits. The second model of CQR Code uses nine colors (white, black, red, green, blue, cyan, yellow, magenta and gray) and stores exactly 2048 information bits and 4576 redundancy bits. The amount of 1024 and 2048 information bits that the two CQR Code models can store was designed for future storage and transmission of cryptography protocols of these sizes.

SUMÁRIO

1 INTRODUÇÃO **1**

1.1 Contexto 1

1.2 Apresentação do Problema e Justificativa 1

1.3 Definições Iniciais 2

1.3.1 CQR Code 2

1.3.2 Reed-Solomon 2

1.4 Métodos Propostos 3

1.4.1 CQR Code - Modelo 1 3

1.4.2 CQR Code - Modelo 2 3

1.5 Organização da Dissertação 4

2 PROCESSAMENTO DIGITAL DE IMAGENS **6**

2.1 Definições e Aplicações 6

2.2 Aquisição e Representação de Imagens 7

2.3 Espaço de Cores Primárias e Secundárias RGB 9

2.4 Compressão de Imagens em Tons Contínuos 11

2.4.1 Padrão JPEG 13

2.4.2 Padrão JPEG2000 14

3 CÓDIGOS CORRETORES DE ERRO: REED-SOLOMON **16**

3.1 Propriedades dos Códigos Reed-Solomon 16

3.2 Descrição dos Códigos de Correção de Erro Reed-Solomon 17

3.3 Códigos de Correção Reed-Solomon - Exemplificação 18

4 QR CODES **20**

4.1 Geração e Descrição de QR Codes 20

4.1.1 Área Padrão 21

4.1.2 Área de Codificação 21

4.2 Correção de Erro na Área de Codificação 22

4.3 Posicionamento de Bits no QR Code 23

4.4 Mascaramento de Bits 24

4.5 Decodificação do QR Code 26

5 SOLUÇÃO PROPOSTA: CQR CODES 28

5.1 Primeiro Modelo de CQR Code 28

5.1.1 Estrutura do Primeiro Modelo de CQR Code 28

5.1.2 Capacidade de Armazenamento do Primeiro Modelo de CQR Code 29

5.1.3 Geração do Primeiro Modelo de CQR Code 29

5.1.4 Leitura do Primeiro Modelo de CQR Code 31

5.2 Segundo Modelo de CQR Code 38

5.2.1 Estrutura do Segundo Modelo de CQR Code 38

5.2.2 Capacidade de Armazenamento do Segundo Modelo CQR Code 38

5.2.3 Geração do Segundo Modelo de CQR Code 39

5.2.4 Leitura do segundo modelo de CQR Code 40

5.2.5 Identificação do Segundo Modelo de CQR Code 41

6 DECODIFICAÇÃO DO PRIMEIRO MODELO DE CQR CODE NAS COMPRESSÕES JPEG E JPEG2000 46

6.1 Compressão JPEG 46

6.2 Compressão JPEG2000 47

7 ANÁLISE DOS RESULTADOS 56

7.1 Apresentação dos Resultados - Primeiro Modelo de CQR Code 58

7.2 Apresentação dos Resultados - Segundo Modelo de CQR Code 61

7.3 Análise de Área de Codificação para CQR Codes de 5 Cores 65

7.4 Análise de Área de Codificação para CQR Codes de 9 Cores 70

8 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS 72

8.1 Considerações Finais 72

8.2 Trabalhos futuros 73

REFERÊNCIAS BIBLIOGRÁFICAS 75

APÊNDICES 77

A ESTRUTURAS DOS PRINCIPAIS QR CODES 78

LISTA DE TABELAS

2.1 Componentes RGB utilizados nos modelos de CQR Codes. 11

4.1 Níveis de correção de erro dos QR Codes. 23
4.2 Penalidades de QR Codes mascarados. 27

5.1 Distribuição dos módulos no primeiro modelo de CQR Code. 29
5.2 Equivalência de cores de módulos e bits por módulo no primeiro modelo de CQR Code. 29
5.3 Equivalência de cores de módulos e bits por módulo no segundo modelo de CQR Code. 38

6.1 %Erro de Decodificação x Bits por *Pixel* no tamanho aumentado 10 vezes - JPEG. 47
6.2 %Erro de Decodificação x Bits por *Pixel* no Tamanho aumentado 20 vezes - JPEG. 47
6.3 %Erro de Decodificação x Bits por *Pixel* no Tamanho aumentado 30 vezes - JPEG. 48
6.4 %Erro de Decodificação x Bits por *Pixel* no Tamanho aumentado 10 vezes - JPEG2000. 49
6.5 %Erro de Decodificação x Bits por *Pixel* no Tamanho aumentado 20 vezes - JPEG2000. 51
6.6 %Erro de Decodificação x Bits por *Pixel* no aumentado 30 vezes - JPEG2000. 53

7.1 Decodificação do primeiro modelo de CQR Code. 59
7.2 Decodificação do segundo modelo de CQR Code. 65
7.3 Simulações de diversos tamanhos de CQR Codes com cinco cores. . . . 70
7.4 Simulações de diversos tamanhos de CQR Codes com nove cores. 71

LISTA DE FIGURAS

1.1 Estocabilidade de Códigos de Barras e QR Codes. 1
1.2 Primeiro modelo de CQR Code. 4
1.3 Segundo modelo de CQR Code. 4

2.1 *Pixels* de imagem digital [9]. 8
2.2 Representação matricial de imagem digital [9]. 8
2.3 Sistema de coordenadas do modelo RGB. 10
2.4 Representação do modelo 24-bits RGB em cubo [9]. 10
2.5 Imagem "Lena", representação das componentes do modelo RGB [13]. . 11
2.6 Diagrama de blocos do JPEG: codificação e decodificação [13]. 13
2.7 Varredura *zig-zag*: padrão para reordenação dos coeficientes da DCT [13]. 14
2.8 Diagrama simplificado do JPEG2000. 15

3.1 Categorias dos Códigos de Blocos Lineares 17
3.2 Utilização do Reed-Solomon em um sistema de comunicação. 17
3.3 Distribuição dos símbolos do código Reed-Solomon sistemático. 18

4.1 Estrutura do QR Code comum [2]. 20
4.2 Estrutura do Módulo de localização do QR Code [2]. 21
4.3 Informação de formato no QR Code [2]. 22
4.4 Equivalência entre versões e módulos do QR Code. 22
4.5 Sentidos de posicionamento do QR Code [2]. 23
4.6 Exemplo de posicionamento na presença de um módulo de alinhamento [2]. 24
4.7 Posicionamento de bits no QR Code de versão 2 e formato M [22]. . . . 24
4.8 Oito máscaras de QR Code [22]. 25
4.9 Processo de escolha dos QR Code mascarados [22]. 26
4.10 Processo de decodificação da imagem do QR Code [22]. 27

5.1 Estrutura do primeiro e segundo modelo de CQR Codes [7]. 28
5.2 Posicionamento dos módulos na Área de Codificação dos CQR Codes [7]. 31

5.3 Exemplo do primeiro modelo de CQR Code utilizado no decorrer da dissertação. 31
5.4 Primeiro modelo de CQR Code - Menor 1, 3cm x 1, 3cm. 32
5.5 Primeiro modelo de CQR Code - Médio 2, 6cm x 2, 6cm. 32
5.6 Imagem adquirida do primeiro modelo de CQR Code - Menor 1, 3cm x 1, 3cm. 32
5.7 Histograma típico utilizado para calcular o limiar th. 33
5.8 Imagem original com as cores primárias ressaltadas. 34
5.9 Imagem recortada. 35
5.10 Correção de rotação. 35
5.11 Imagem recortada, rodada e com as cores primárias ressaltadas exceto a cor preta. 36
5.12 Sentido do mapeamento determinístico dos módulos do CQR Code. . . 37
5.13 Fatores de correção do terceiro quadrante. 37
5.14 Exemplo do segundo modelo de CQR Code utilizado no decorrer da dissertação. 40
5.15 Segundo modelo de CQR Code - Menor 1, 3cm x 1, 3cm. 40
5.16 Segundo modelo de CQR Code - Médio 2, 6cm x 2, 6cm. 40
5.17 Imagem adquirida do segundo modelo de CQR Code - Menor 1, 3cm x 1, 3cm. 41
5.18 Imagem do segundo modelo de CQR Code com cores preto, branco e vermelho ressaltadas. 42
5.19 Imagem do Segundo Modelo de CQR Code recortada. 43
5.20 Imagem do Segundo Modelo de CQR Code rodada. 43
5.21 Processo de decodificação dos CQR Codes. 45

6.1 CQR Code aumentado 10 vezes e comprimido com o maior parâmetro de compressão JPEG 0. 48
6.2 CQR Code aumentado 20 vezes e comprimido com o maior parâmetro de compressão JPEG 0. 48
6.3 CQR Code aumentado 30 vezes e comprimido com o maior parâmetro de compressão JPEG 0. 49
6.4 CQR Code aumentado x10 e comprimido com o maior parâmetro de compressão JPEG2000 175. 50
6.5 CQR Code aumentado 10 vezes e comprimido com o maior parâmetro de compressão JPEG2000 200. 50
6.6 CQR Code aumentado 10 vezes e comprimido com o maior parâmetro de compressão JPEG2000 250. 50

6.7 CQR Code aumentado 10 vezes e comprimido com o maior parâmetro de compressão JPEG2000 400. 51
6.8 CQR Code aumentado 20 vezes e comprimido com o maior parâmetro de compressão JPEG2000 600. 51
6.9 CQR Code aumentado 20 vezes e comprimido com o maior parâmetro de compressão JPEG2000 700. 52
6.10 CQR Code aumentado 20 vezes e comprimido com o maior parâmetro de compressão JPEG2000 800. 52
6.11 CQR Code aumentado 30 vezes e comprimido com o maior parâmetro de compressão JPEG2000 2250. 53
6.12 CQR Code aumentado 30 vezes e comprimido com o maior parâmetro de compressão JPEG2000 2500. 54
6.13 Percentagem de símbolos errados na decodificação x BPP para imagem de CQR Code aumentado 10 vezes nos padrões JPEG e JPEG2000. . . 54
6.14 Percentagem de símbolos errados na decodificação x BPP para imagem de CQR Code aumentado 20 vezes nos padrões JPEG e JPEG2000. . . 55
6.15 Percentagem de símbolos errados na decodificação x BPP para imagem de CQR Code aumentado 30 vezes nos padrões JPEG e JPEG2000. . . 55

7.1 Constelação RGB do QR Code. 57
7.2 Constelação RGB do primeiro modelo de CQR Code. 57
7.3 Constelação RGB do segundo modelo de CQR Code. 58
7.4 Processo de decodificação - Primeira imagem de 1a na Tabela 7.1. . . . 60
7.5 Processo de decodificação - Imagem recortada de 1a na Tabela 7.1. . . 60
7.6 Processo de decodificação - Imagem rodada de 1a na Tabela 7.1. . . . 60
7.7 Processo de decodificação - Imagem limiarizada nas cores primárias de 1a na Tabela 7.1. 61
7.8 Processo de decodificação - Primeira imagem de 2a na Tabela 7.1. . . . 61
7.9 Processo de decodificação - Primeira imagem de 3a na Tabela 7.1. . . . 62
7.10 Processo de decodificação - Imagem recortada de 3a na Tabela 7.1. . . 62
7.11 Processo de decodificação - Imagem rodada de 3a na Tabela 7.1. . . . 62
7.12 Processo de decodificação - Imagem limiarizada nas cores primárias de 3a na Tabela 7.1. 63
7.13 Processo de decodificação - Primeira imagem de 4a na Tabela 7.1. . . . 63
7.14 Processo de decodificação - Imagem recortada de 4a na Tabela 7.1. . . 63
7.15 Processo de decodificação - Imagem rodada de 4a na Tabela 7.1. . . . 64
7.16 Processo de decodificação - Imagem limiarizada nas cores primárias de 4a na Tabela 7.1. 64

7.17 Processo de decodificação - Primeira imagem de 1 na Tabela 7.2. . . . 65
7.18 Processo de decodificação - Imagem recortada de 1 na Tabela 7.2. . . . 66
7.19 Processo de decodificação - Imagem rodada de 1 na Tabela 7.2. 66
7.20 Processo de decodificação - Imagem limiarizada nas cores primárias de 1 na Tabela 7.2. 66
7.21 Processo de decodificação - Primeira imagem de 2 na Tabela 7.2. . . . 67
7.22 Processo de decodificação - Imagem recortada de 2 na Tabela 7.2. . . . 67
7.23 Processo de decodificação - Imagem rodada de 2 na Tabela 7.2. 67
7.24 Processo de decodificação - Imagem limiarizada nas cores primárias de 2 na Tabela 7.2. 68
7.25 Processo de decodificação - Primeira imagem de 3 na Tabela 7.2. . . . 68
7.26 Processo de decodificação - Imagem recortada de 3 na Tabela 7.2. . . . 68
7.27 Processo de decodificação - Imagem rodada de 3 na Tabela 7.2. 69
7.28 Processo de decodificação - Imagem limiarizada nas cores primárias de 3 na Tabela 7.2. 69

A.1 Versões 1 e 2 do QR Code [2]. 78
A.2 Versão 6 do QR Code [2]. 78
A.3 Versão 7 do QR Code [2]. 78
A.4 Versão 14 do QR Code [2]. 79
A.5 Versão 21 do QR Code [2]. 79
A.6 Versão 40 do QR Code [2]. 79

1 INTRODUÇÃO

1.1 Contexto

Os *Quick Response Code* conhecidos popularmente como QR Codes são códigos matriciais que transmitem bits por meio de imagens digitais ou analógicas, a principal característica dos QR Codes é a capacidade de armazenar informações digitais nas dimensões vertical e horizontal à diferença do código de barra [1] que armazena informações digitais somente na dimensão vertical, conforme ilustrado na Figura 1.1. Os QR Codes foram criados no ano 1994 e são marca registrada da empresa japonesa *Denso Wave Incorporated* [2].

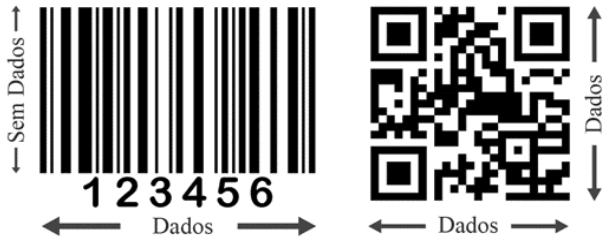

Figura 1.1: Estocabilidade de Códigos de Barras e QR Codes.

Atualmente os QR Codes são os códigos matriciais mais utilizados no mundo do marketing e vendas, os QR Codes aparecem nos lugares mais cotidianos como lojas, revistas, ônibus, cartões de visitas e até em lápides de cemitério [3]. A utilidade mais comum no momento da leitura do QR Code é ser encaminhado para um site na internet, também é possível encontrar algumas outras utilidades no dia a dia como baixar algum programa, visualizar um vídeo ou imagem e até curtir um perfil no Facebook entre outros [4, 5]. É claro que podem ser criados vários tipos de utilidades na área de informática que tenham um pouco mais de complexidade computacional como armazenar e rodar programas com diferentes aplicativos vinculados à leitura dos QR Codes com a utilização de criptografia.

1.2 Apresentação do Problema e Justificativa

Apesar da vasta quantidade de aplicações baseadas em QR Codes, não existe na literatura estudo acerca dos limites fundamentais da sua capacidade de armazenamento de informações. Não se sabe, por exemplo, quanta informação pode ser armazenada na

estrutura de um QR Code de determinado tamanho associado a uma câmera de determinada resolução. Atualmente existe a necessidade de armazenar e transmitir maior quantidade de informações digitais em áreas de menor tamanho, há uma tendência de ampliar a quantidade de informações digitais em meios impressos. Com os códigos de barra implementados mundialmente como a primeira versão de armazenamento e transmissão de informações digitais, surgem como evolução os códigos matriciais como o QR Code. Nessa dissertação é apresentado o CQR Code em dois modelos com o objetivo de armazenar e transmitir maior quantidade de informações digitais na menor quantidade de área de impressão possível. Uma aplicação com a solução proposta nessa dissertação é armazenar e transmitir dados de criptografia como assinaturas digitais cujo comprimento pode ser de 1024 e 2048 bits, quantidade de informação que não cabe em um QR Code pequeno, somente um QR Code de tamanho grande e com alto nível de capacidade de correção de erros (85 x 85 módulos ou 5,6 cm x 5,6 cm) poderia conter 1024 bits de informação [6], o que torna inviável a utilização de QR Codes tradicionais em embalagens de tamanho pequeno como medicamentos, cigarros e até armamento para esses fins.

1.3 Definições Iniciais

Para que o restante do texto seja melhor compreendido, será feita uma definição conceitual do CQR Code e do algoritmo de correção de erro Reed-Solomon.

1.3.1 CQR Code

Os denominados *Colored Quick Response Code*, apresentados em [7], são códigos matriciais com capacidade superior de armazenamento e transmissão de informações digitais do que o QR Code tradicional devido à utilização de cores na região de codificação. O CQR Code utiliza a estrutura do QR Code [2] sem a necessidade das linhas de espaçamento e dos módulos de alinhamento que serão mostrados na seção 5.

1.3.2 Reed-Solomon

São códigos cíclicos de correção de erros não binários inventados por Irving S. Reed e Gustave Solomon. Os Códigos RS constituem uma sub-classe de uma ampla classe de códigos cíclicos denominada Códigos BCH (*Bose-Chaudhuri-Hocquenghem*). Nos códigos BCH foi descrito uma forma sistemática de construção de códigos capazes de detectar e corrigir vários erros aleatórios de símbolos [8]. O código Reed-Solomon recebe como dados de entrada as informações em símbolos (conjunto de bits de um

determinado tamanho, no caso desse trabalho são conjuntos de 16 bits expressados em números de 0 até 65535), para posteriormente gerar de forma sistemática símbolos de redundância, tendo como saída o conjunto de símbolos de informação seguido dos símbolos de redundância. Existe a necessidade de converter os bits de informação para símbolos (entrada do processo de codificação Reed-Solomon) e posteriormente os símbolos de informação e redundância para bits (saída do processo de decodificação Reed-Solomon).

1.4 Métodos Propostos

Essa dissertação inicia os estudos definidos no item 1.2 e aborda os limites da capacidade de transmissão no canal em função da complexidade computacional do algoritmo de leitura, do modelo de celular (câmera) utilizada e da quantidade de cores no CQR Code. Nessa dissertação foram refeitos do zero, com base em [2], a geração e leitura de QR Codes com a utilização de cores de modo a permitir que eles possam armazenar assinaturas digitais de até 2048 bits. O uso de criptografia possibilitará uma nova gama de aplicações para códigos bidimensionais, como por exemplo, o rastreamento seguro de objetos com pouca área disponível de impressão. Com base no item 1.2 dessa dissertação, foram criados dois modelos de CQR Codes utilizando cores.

1.4.1 CQR Code - Modelo 1

O primeiro modelo de CQR Code conta com a estrutura da Versão 8 do QR Code e utiliza 5 cores para armazenar informações digitais. As cores são preto, branco, vermelho, verde e azul. O modelo 1 foi implementado para armazenar e transmitir 1024 bits de informação e 3392 bits de redundância. O menor tamanho possível é de $1,3$cm x $1,3$cm (um *pixel* por módulo). O modelo 1 conta com a taxa de recuperação de bits de até $38,40$ % e está mostrado na Figura 1.2.

1.4.2 CQR Code - Modelo 2

O segundo modelo de CQR Code é uma evolução do primeiro modelo e também conta com a estrutura da Versão 8 do QR Code. O segundo modelo utiliza 9 cores para armazenar informações digitais As cores são preto, branco, vermelho, verde, azul, ciano, amarelo, magenta e cinza. O modelo 2 foi implementado para armazenar e transmitir 2048 bits de informação e 4576 bits de redundância. O modelo 2 conta com a taxa de recuperação de bits de até $34,54$ % e está mostrado na Figura 1.3.

Figura 1.2: Primeiro modelo de CQR Code.

Figura 1.3: Segundo modelo de CQR Code.

1.5 Organização da Dissertação

A dissertação está dividida em oito capítulos. O primeiro é constituído pela presente introdução. Os Capítulos 2, 3 e 4 destinam-se à revisão da literatura técnica, necessária ao pleno entendimento dos métodos propostos nos Capítulos 5 e 6.

No Capítulo 2 serão tratados conceitos básicos em processamento digital de imagens, tais como aquisição e representação de imagens, espaço de cores primárias e secundárias RGB e alguns padrões de compressão utilizados nos experimentos.

No Capítulo 3 é explicado o conceito de Códigos Corretores de Erros com ênfase no código Reed-Solomon.

No Capítulo 4 é explicado o funcionamento detalhado dos QR Codes com base no padrão internacional *ISO/IEC 18004:2005 - Information technology - Automatic identification and data capture techniques - QR Code 2005 bar code symbology specification* [2]. É indispensável conhecer como o QR Code é gerado e lido dado que o CQR Code utiliza uma estrutura muito parecida.

Os Capítulos 5 e 6 apresentam as contribuições da presente dissertação, eles exploram com detalhes a geração e leitura do primeiro e segundo modelo de CQR Code respectivamente e também de um análise da decodificação dos CQR Codes em cima de imagens do primeiro modelo do CQR Code com compressão JPEG e JPEG2000.

O Capítulo 7 mostra os resultados finais dessa dissertação.

Por fim, o Capítulo 8 é destinado à apresentação das conclusões, bem como das propostas para trabalhos futuros.

2 PROCESSAMENTO DIGITAL DE IMAGENS

Nesse capítulo serão apresentadas as definições e os modelos matemáticos das ferramentas utilizadas no processamento de imagens e na codificação e decodificação dos modelos propostos de CQR Code. Os conceitos apresentados podem ser aprofundados em [9].

2.1 Definições e Aplicações

A visão é o mais avançado dos nossos sentidos, portanto, não é surpreendente que as imagens desempenhem o papel mais importante na percepção humana. Uma imagem pode ser definida como uma função bidimensional $f(x, y)$, onde x e y são coordenadas espaciais. A amplitude de f, em qualquer par de coordenadas (x, y), é o de nível de intensidade naquele ponto. Uma imagem é considerada imagem digital quando os valores de amplitude de f são finitos e discretos. A área de processamento digital de imagens é qualquer forma de processamento de dados no qual a entrada e saída são imagens tais como fotografias ou quadros de vídeo digitais. Ao contrário do tratamento de imagens, que preocupa-se somente pela manipulação de figuras para sua representação final, o processamento de imagens é uma ferramenta para novos processamentos de dados tais como aprendizagem de máquina ou reconhecimento de padrões [9].

Uma imagem digital é composta de um número finito de elementos, cada elemento conta com um determinado local e valor. Estes elementos são referidos como elementos de imagem, pels ou *pixels*. *Pixel* é o termo mais utilizado para designar os elementos de uma imagem digital [9].

Hoje em dia, praticamente não existe área de atividade técnica que não seja afetada de alguma forma pelo processamento digital de imagens. Uma das maneiras mais simples de desenvolver uma compreensão básica do ponto de vista de aplicações de processamento de imagens é categorizar imagens de acordo com sua fonte (por exemplo, visual, raio X, faixa infravermelha, faixa ultravioleta e assim por diante). A principal fonte de energia para as imagens é o espectro de energia eletromagnética. Outras fontes importantes de energia incluem acústica e eletrônicos (sob a forma de feixes de elétrons utilizada em microscopia eletrônica). Imagens sintéticas, utilizadas para a modelagem e visualização, são geradas por computador [9].

Uma das primeiras aplicações de imagens digitais foi na indústria jornalística,

quando as imagens foram enviadas pelo primeiro cabo submarino entre Londres e Nova York. A introdução do sistema de transmissão de imagens via cabo Bartlane no início de 1920 reduziu o tempo necessário para o transporte de uma imagem através do Atlântico de mais de uma semana a menos de três horas. Os primeiros sistemas Bartlane foram capazes de codificar imagens em cinco níveis distintos de cinza. Essa capacidade foi aumentada para 15 níveis em 1929 [9].

As técnicas de processamento digital de imagens começaram a ser utilizadas no final da década de 1960 e início de 1970 na área de imagiologia médica, observação remota da Terra e astronomia. A invenção da tomografia computadorizada no início da década de 1970, é um dos eventos mais importantes na aplicação de processamento de imagens no diagnóstico médico [9].

2.2 Aquisição e Representação de Imagens

Imagens digitais são geradas pela combinação de uma fonte de iluminação direta ou de reflexão, por exemplo, a iluminação pode originar de uma fonte de energia eletromagnética, tais como, infravermelho e energia de raios-X ou também poderia provir de fontes menos tradicionais, como o ultra-som ou mesmo um padrão de iluminação gerados por computador. Dependendo da natureza da fonte, a energia de iluminação é refletida ou transmitida através de objetos [9].

As imagens são capturadas mediante sensores digitais que captam a luminosidade das imagens que são projetadas sobre ele continuamente e dá início ao processo de captura de uma instância ou de uma seqüência de instâncias da imagem consecutivamente. Para captação de imagens a cores é comum que câmeras de vídeo utilizem três sensores (sistema 3CCD), cada sensor com um filtro de uma tripla de filtros tricrômicos sobre ele, sendo que câmeras fotográficas geralmente contam com um único sensor de imagem que agrupa seus *photosites* sob um mosaico de filtros de luminosidade e de cor [9].

A saída da maioria dos sensores é uma forma de onda contínua cuja amplitude e comportamento espacial estão relacionadas com o fenômeno físico a ser detectado. Para criar uma imagem digital é preciso converter os dados do sensoriamento contínuo em formato digital, isto envolve dois processos: amostragem e quantização. A idéia básica por trás da amostragem e quantização consiste em processar uma imagem $f(x, y)$ contínua em relação às coordenadas x, y e também em amplitude; para converter a imagem para o formato digital, temos de armazenar a função em ambas coordenadas e em amplitude. Digitalizar os valores das coordenadas é conhecido como amostragem e digitalizar os valores de amplitude é conhecido como quantização [9].

O resultado da amostragem e quantização é uma matriz de números reais. Suponha que uma imagem $f(x, y)$ é amostrada resultando em uma imagem digital com M linhas e N colunas. Os valores das coordenadas (x, y) agora são quantidades discretas. Para maior clareza e conveniência de notação, vamos usar valores inteiros para essas coordenadas discretas. Os valores das coordenadas estão na origem $(x, y) = (0, 0)$ como mostrado na Figura 2.1.

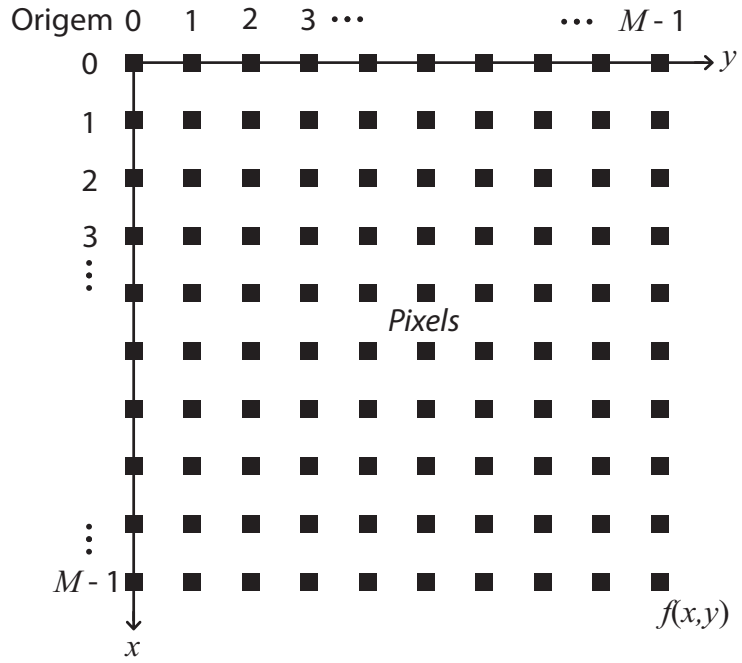

Figura 2.1: *Pixels* de imagem digital [9].

Podemos representar a Figura 2.1. matematicamente como uma matriz com valores inteiros como mostrado na Figura 2.2.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N\text{-}2) & f(0,N\text{-}1) \\ f(1,0) & f(1,1) & \cdots & f(1,N\text{-}2) & f(1,N\text{-}1) \\ \vdots & \vdots & & \vdots & \vdots \\ f(M\text{-}2,0) & f(M\text{-}2,1) & \cdots & f(M\text{-}2,N\text{-}2) & f(M\text{-}2,N\text{-}1) \\ f(M\text{-}1,0) & f(M\text{-}1,1) & \cdots & f(M\text{-}1,N\text{-}2) & f(M\text{-}1,N\text{-}1) \end{bmatrix}$$

Figura 2.2: Representação matricial de imagem digital [9].

Sejam Z e R os números inteiros e os números reais respectivamente, o processo de amostragem pode ser visto como o particionamento do plano xy em uma grade, com as coordenadas do centro de cada grade, sendo um par de elementos do produto cartesiano Z_2 os valores que representam o conjunto de todos os pares ordenados dos

elementos (z_i, z_j), sendo z_i e z_j inteiros de Z. Assim, $f(x, y)$ é uma imagem digital se (x, y) são inteiros que pertencem a Z_2 e f é uma função que atribui um valor (ou seja, um número real a partir do conjunto dos números reais, R) para cada par distinto de coordenadas (x, y) [9]. O processo de digitalização exige determinar os valores de M, N, e de L (níveis de cinza discretos permitidos para cada *pixel*). Não há exigências sobre os valores de M e N, com exceção de que têm de ser inteiros positivos. No entanto, devido ao processamento, armazenamento e considerações sobre hardware de amostragem, o número de níveis de cinza normalmente é uma potência inteira de 2 mostrado na Equação 2.1:

$$L = 2^k. \tag{2.1}$$

A quantidade de bits necessários para o armazenamento da imagem digital está mostrada na Equação 2.2:

$$b = M \times N \times k. \tag{2.2}$$

A quantidade de bits b é diretamente proporcional aos valores M, N e k.

2.3 Espaço de Cores Primárias e Secundárias RGB

No modelo RGB, cada *pixel* de uma imagem é mostrado em função dos componentes espectrais primários das cores vermelho, verde e azul; esse modelo tem como base o sistema de coordenadas cartesiano, as cores primárias e suas coordenadas são mostradas na Figura 2.3. As diferentes cores no modelo RGB são pontos que se encontram dentro ou acima do cubo da Figura 2.3. e são definidos como vetores com início na origem. As imagens representadas no modelo RGB contam com três componentes primários aditivos (vermelho, verde e azul) e com três componentes secundários aditivos (ciano, amarelo e magenta). Os valores de cada componente primário RGB são misturados para representar a cor de cada *pixel* [9].

O número de bits utilizados na representação de cada *pixel* no espaço primário aditivo RGB é chamado de profundidade de *pixel*. Considerando que cada *pixel* usado na geração do CQR Code conta com uma imagem de 8-bits para cada cor primária

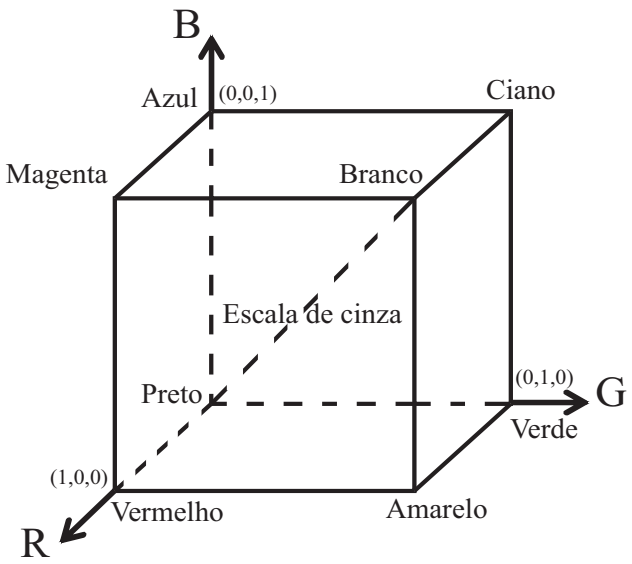

Figura 2.3: Sistema de coordenadas do modelo RGB.

(vermelho, verde e azul) é concluído que cada *pixel* tem uma profundidade de 24-bits, dessa maneira, a quantidade total de possíveis cores que cada *pixel* poderá representar é de $(2^8)^3$ ou de 16777216 cores [9].

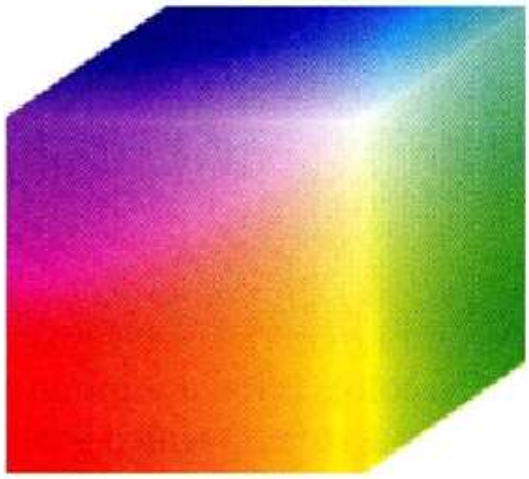

Figura 2.4: Representação do modelo 24-bits RGB em cubo [9].

As cores primárias, secundárias e em escala de cinza de profundidade de 24-bits utilizados nos modelos dos CQR Codes são mostradas na Tabela 2.1:

As imagens coloridas armazenam informações de cor em toda a área da imagem. No espaço de cores RGB, por exemplo, a imagem é definida a partir de três imagens componentes, cada qual responsável por registrar as intensidades de vermelho, verde e azul refletidas por uma cena. A Figura 2.5 mostra um exemplo de imagem colorida acompanhada de suas três componentes RGB.

Tabela 2.1: Componentes RGB utilizados nos modelos de CQR Codes.

Cores/Componentes	R	G	B
Branco	255	255	255
Preto	0	0	0
Vermelho	255	0	0
Verde	0	255	0
Azul	0	0	255
Ciano	0	255	255
Amarelo	255	255	0
Magenta	255	0	255
Cinza	128	128	128

Figura 2.5: Imagem "Lena", representação das componentes do modelo RGB [13].

2.4 Compressão de Imagens em Tons Contínuos

Os padrões de compressão de tons contínuos citados nessa dissertação foram utilizados como parâmetros de medição de robustez do decodificador do primeiro modelo de CQR Code. Os padrões JPEG e JPEG2000 utilizam técnicas de codificação por transformada com perdas de acordo com [9, 10, 11]. O funcionamento da compressão consiste em levar os *pixels* da imagem que estão originalmente no domínio espacial para outro domínio por meio da utilização de transformadas, onde a maior parte da informação visual é concentrada em poucos elementos, também chamados de coeficientes. Uma vez verificada a existência dos coeficientes, são selecionados os coeficientes de menor relevância e quantizados para zero, sendo possível obter a compressão. Os

coeficientes zerados não poderão ser recuperados na decodificação, resultando em uma compressão com perdas.

As perdas resultantes da compressão têm como conseqüência a distorção na imagem resultante, essa distorção pode ser avaliada como subjetiva e objetiva [12].

A medida subjetiva está baseada na opinião de um conjunto de pessoas que avaliam a qualidade da imagem resultante de acordo ao critério de cada pessoa.

Já na medida objetiva calcula-se a diferença entre a imagem original F_o e a imagem reconstruída F_r. Medidas largamente utilizadas na literatura são o erro médio quadrático (*MSE*, do inglês *Mean Squared Error*), a soma das diferenças absolutas (*SAD*, do inglês *Sum of Absolute Differences*), o erro médio absoluto (*MAE*, do inglês *Mean Absolute Error*) e a razão sinal ruido de pico (*PSNR*, do inglês *Peak Signal-to-Noise Ratio*). As seguintes equações mostram o cálculo das medidas objetivas:

$$MRE = \frac{1}{M \times N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(F_o(m,n) - F_r(m,n)\right)^2 \tag{2.3}$$

$$SAD = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |F_o(m,n) - F_r(m,n)| \tag{2.4}$$

$$MAE = \frac{1}{M \times N} \times SAD \tag{2.5}$$

$$PSNR = 10 log_{10} \frac{MAX^2}{MSE} \; dB \tag{2.6}$$

Aqui, *MAX* representa o máximo valor que um *pixel* pode assumir [13].

Nessa dissertação alguns experimentos foram trabalhados com o conceito de Profundidade de Cor ou Bits Por *Pixel* (bpp). A quantidade de bits por *pixel* indica a quantidade de bits utilizados para a descrição da cor de um único *pixel* em uma imagem mapeada, esse conceito é um parâmetro utilizado para descrever de forma objetiva a quantidades de bits por *pixel* utilizados em uma imagem. A quantidade de bits por *pixel* das imagens que são analisadas no Capítulo 5 é determinada pela Equação 2.7:

$$BPP = \frac{TAM}{M \times N} \tag{2.7}$$

Onde TAM é o tamanho do arquivo em bits e M, em conjunto com N, são as dimensões da imagem em *pixels*.

2.4.1 Padrão JPEG

Criado em 1986, o *Joint Photograph Experts Group* (JPEG) é um grupo formado pelos órgãos ISO, IEC e ITU. Em 1992, o grupo publicou o primeiro padrão internacional para compressão de imagens em tons contínuos, popularmente conhecido como JPEG [14, 15]. O funcionamento do padrão JPEG está ilustrado na Figura 2.6.

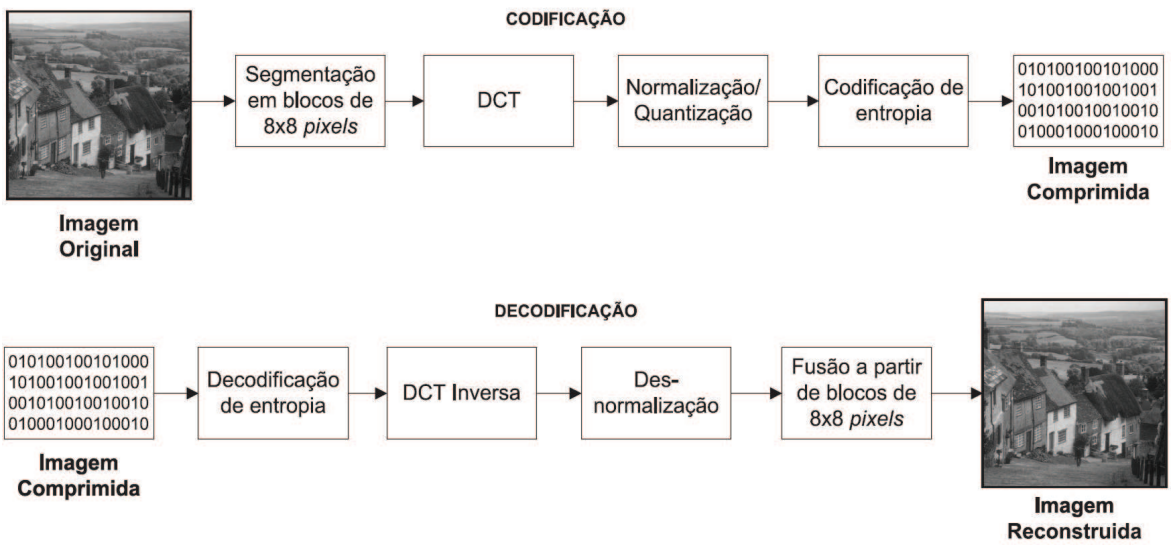

Figura 2.6: Diagrama de blocos do JPEG: codificação e decodificação [13].

O algoritmo de compressão JPEG divide a imagem de entrada em blocos de 8 x 8 *pixels* e processa cada bloco de esquerda para a direita e de cima para baixo. Do valor de intensidade de cada *pixel* são subtraídos 128 níveis de cinza e a Transformada de Cosseno Discreta (DCT do inglês *Discrete Cosine Transform*) [16] bidimensional é computada. Em seguida, são realizadas simultaneamente a normalização e a quantização escalar uniforme [17] dos coeficientes da transformada. É nessa etapa que, dependendo da quantização/normalização aplicada, uma quantidade maior ou menor de coeficientes é zerada, resultando em uma maior ou menor taxa de compressão. É importante ressaltar, porém, que quanto maior é a taxa de compressão, pior é a qualidade da imagem comprimida. Após a quantização/normalização, a matriz 8×8 de coeficientes é reordenada a partir da varredura em sentido *zigzag* [15] conforme ilustrado na Figura 2.7, resultando em uma seqüência unidimensional de 64 coeficientes. A etapa final da compressão consiste na codificação de entropia dos coeficientes, desenvolvida principalmente com o objetivo de se tirar vantagem das seqüência de zeros que aparecem quando os coeficientes são reordenados segundo a varredura proposta.

Na decodificação, o processo inverso é realizado [13].

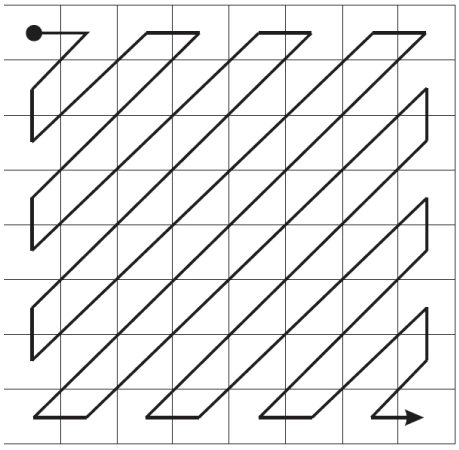

Figura 2.7: Varredura *zig-zag*: padrão para reordenação dos coeficientes da DCT [13].

Atualmente o padrão JPEG é o mais utilizado para compressão de imagens, existe um *software* livre implementado pelo *Independent JPEG Group* (IJP) que é utilizado em muitos programas de computador na atualidade [18].

2.4.2 Padrão JPEG2000

Este padrão é resultado do esforço conjunto do ISO/IEC (do inglês *International Standards Organization/ International Electrotechnical Commission*) e ITU (do inglês *International Telecommunication Union*) ou União Internacional de Telecomunicações. O próprio nome JPEG (*Joint Photographic Experts Team Group*) é uma referencia à união dessas organizações. Em março de 1997, uma chamada de contribuições técnicas foi feita contendo várias características desejáveis, algumas das quais são apresentadas a seguir. As contribuições foram avaliadas em uma reunião em Sidney em novembro do mesmo ano [19]:

- Alto desempenho em baixas taxa. Obviamente desejava-se um desempenho superior aos codificadores de imagem predecessores em todas as faixas de taxa. Contudo, os ganhos em baixas taxas deveriam ser significativamente maiores.
- Compressão de imagens com amostras de 1 a 16 bits.
- Transmissão progressiva, tanto em qualidade como em resolução.
- Compressão com e sem perdas.
- Acesso espacial aleatório sem a necessidade de descompressão de toda imagem.

- Robustez a erros.
- Processamento seqüencial. Não havendo necessidade de armazenamento da imagem inteira em *buffer*.

O resultado foi um padrão que atendia todos os requisitos iniciais e apresentava características inovadoras, algumas das quais são apresentadas a seguir.

- Tamanho do arquivo pode ser definido antes do início da codificação.
- Região de interesse.
- Inclusão de metadados, essa característica foi inserida na parte 2 do padrão.
- Suporte a imagens grandes. O padrão prevê suporte a imagens de até (2^{32} - 1) x (2^{32} - 1) *pixels*.

A Figura 2.8 apresenta um diagrama simplificado do JPEG2000.

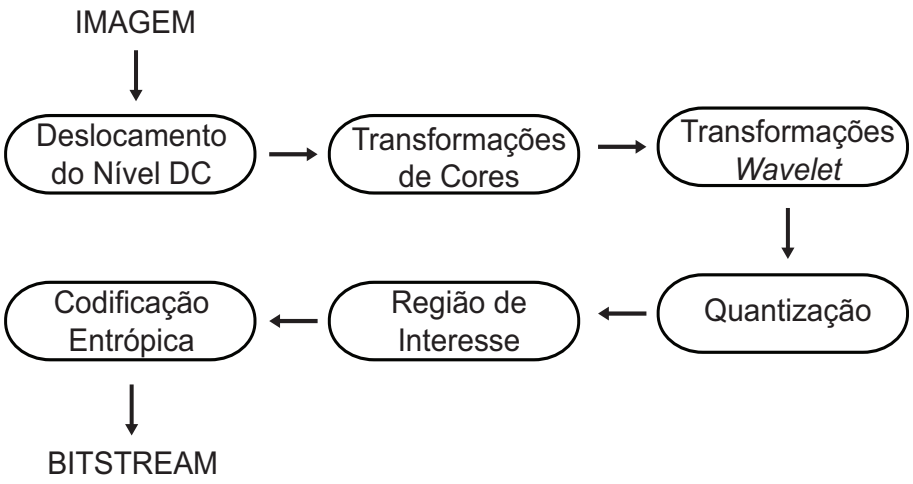

Figura 2.8: Diagrama simplificado do JPEG2000.

3 CÓDIGOS CORRETORES DE ERRO: REED-SOLOMON

Os *Códigos Corretores de Erro* possuem a capacidade de corrigir erros introduzidos em informações digitais como o resultado da sua transmissão por meio de um canal de comunicação. A correção desses erros é feita em função da recepção da informação digital que sai do canal de informação. Existem também os *Códigos Detectores de Erro* que somente detectam erros na saída de um canal de informação em função das informações digitais recebidas. Os códigos corretores de erro e os códigos detectores de erro são *Códigos Controladores de Erro*. A diferenciação dos códigos corretores de erro e os códigos detectores de erro pode especificar a definição de sistemas operacionais comunicativos e sistemas disfuncionais. A implementação dos códigos corretores de erro tive muita importância na revolução das telecomunicações nas últimas décadas sendo utilizados na internet, gravações digitais e comunicação satelital entre outros. Todo tipo de comunicação digital utiliza códigos controladores de erro dado que nenhum canal de comunicação digital garante confiabilidade plena na transmissão [20].

3.1 Propriedades dos Códigos Reed-Solomon

Os códigos corretores de erro utilizados na geração e leitura dos CQR Codes são os Reed-Solomon. O código corretor de erro Reed-Solomon foi criado por Irving Reed e Gus Solomon no ano 1960 e foram conhecidos mediante a publicação de um artigo no *Journal of the Society for Industrial and Applied Mathematics* [21]. Reed-Solomon é utilizado em múltiplos sistemas de comunicação como [2, 20]:

- Equipamentos de armazenamento de informações digitais como CDs, DVDs, Barcodes e QR Codes.
- Comunicação sem fio incluindo comunicação de telefones móveis e *links* micro-onda.
- Comunicação satelital.
- Televisão digital.
- Comunicação entre modens ADSL, xDSL, etc.

3.2 Descrição dos Códigos de Correção de Erro Reed-Solomon

Os códigos corretores de erro Reed-Solomon são códigos lineares cíclicos de blocos não binários, eles pertencem à família dos códigos cíclicos BCH como mostrado na Figura 3.1. Reed-Solomon é utilizado em múltiplos sistemas de comunicação [2, 20]:

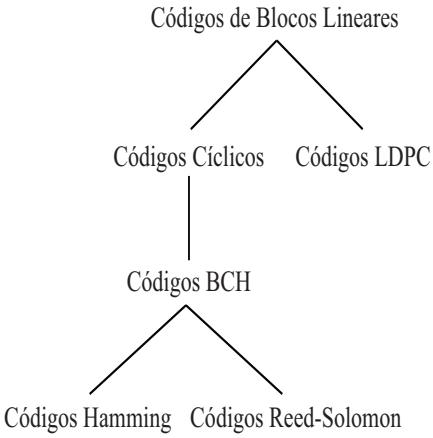

Figura 3.1: Categorias dos Códigos de Blocos Lineares

Um sistema de comunicação típico com a utilização de Reed-Solomon é mostrado na Figura 3.2:

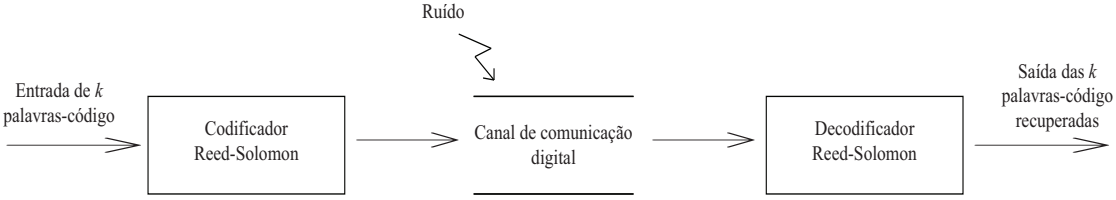

Figura 3.2: Utilização do Reed-Solomon em um sistema de comunicação.

Os códigos Reed-Solomon são especificados na literatura como RS(n,k) com s bits por símbolo, isso significa que o codificador pega k símbolos de informação, cada um composto por s bits, e adiciona símbolos de paridade para produzir no total n símbolos como saída, o codificador gera, em função da entrada de k símbolos, n-k símbolos de paridade. O conjunto de n símbolos é denominado como palavra-código. O Reed-Solomon pode corrigir até t símbolos errados na decodificação. O valor de t é definido na Equação 3.1.

$$t = \frac{n-k}{2}. \tag{3.1}$$

Dado o tamanho de s bits por símbolo, o máximo tamanho de palavra-código

possível n é mostrado na Equação 3.2.

$$n = 2^s - 1. \tag{3.2}$$

Para maior facilidade de extração de dados, os códigos Reed-Solomon podem ser sistemáticos, o que significa que na saída do codificador e na entrada do decodificador, os símbolos de redundância ou padridade estão concatenados logo após os símbolos de informação como mostrado na Figura 3.3:

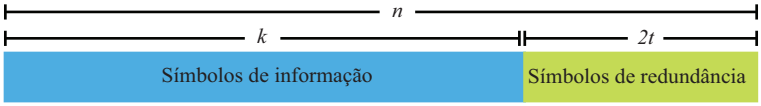

Figura 3.3: Distribuição dos símbolos do código Reed-Solomon sistemático.

No momento do codificador gerar os n - k símbolos de paridade, o algoritmo se baseia em operações aritméticas em um Campo de Galois, para isso o algoritmo utiliza um polinômio gerador que fará a função de interpolador linear em cima dos k símbolos de informação, sendo que todas as palavras-código possíveis são divisíveis pelo polinômio gerador. A equação geral do polinômio gerador é:

$$g(x) = (x - \alpha^i)(x - \alpha^{i+1})...(x - \alpha^{i+2t}). \tag{3.3}$$

Onde alfa é chamado de elemento primitivo do Campo de Galois. A palavra-código é construída em função do polinômio gerador e dos símbolos de informação como mostrado na Fórmula 3.4:

$$c(x) = g(x) \times i(x). \tag{3.4}$$

Onde $g(x)$ é o polinômio gerador, $i(x)$ representa os símbolos de informação e $c(x)$ é uma palavra-código válida.

3.3 Códigos de Correção Reed-Solomon - Exemplificação

Um exemplo popular de codificação Reed-Solomon é RS(255,223) com s=8 (cada símbolo equivale a um byte), onde a palavra-código contém 255 símbolos dos quais

223 símbolos são de informação e 32 são de redundância. Nesse caso a capacidade máxima de símbolos corrigíveis de acordo à Equação 3.1 é de t=16, dessa maneira a percentagem de recuperação de erro é 16/255 ou $6,27\%$. Nesse exemplo, dado o tamanho do símbolo 8 bits, o máximo tamanho da palavra-código é 255 bytes.

É possível executar o Reed-Solomon encurtado, ou seja, com menor quantidade de símbolos de informação mantendo a quantidade de símbolos de paridade. Às vezes essa configuração é necessária porque nem sempre temos a quantidade de símbolos de informação necessários para atingir n. O seguinte exemplo mostra o código Reed-Solomon RS(255,223) encurtado.

Nesse exemplo no lugar de colocar 223 símbolos de informação k, podemos colocar menos, por exemplo 168 e para manter a mesma quantidade de símbolos de paridade n-k ou 32 símbolos, nosso novo n será 200. No codificador, o n prático vai continuar sendo 255, de qualquer forma, serão inseridos 223 símbolos de informação, sendo que os primeiros 168 vão ter informação válida e os 55 restantes serão zerados, assim, em função dos 168 símbolos de informação mais os 55 símbolos de informação zerados, vamos obter os 32 símbolos de redundância. Dessa maneira o nosso novo código, na prática RS(200,168) e na teoria RS(255,223), dessa maneira t:

$$t = \frac{200 - 168}{2} \tag{3.5}$$

ou 16 símbolos. A percentagem de recuperação de símbolos será nosso t dividido por nosso n prático de 200, resultando em 8%.

No processo de decodificação, o Reed-Solomon decodificará os n símbolos de entrada e detectará a quantidade de símbolos errôneos, caso a quantidade de símbolos corrompidos seja maior do que t símbolos o algoritmo vai indicar que os símbolos de informação não puderam ser recuperados, caso seja menor ou igual à t, o algoritmo vai recuperar os k símbolos de informação inseridos na codificação.

4 QR CODES

Nesse Capítulo serão apresentadas as características do QR Code encontrado atualmente em revistas, jornais e propagandas; tais como as versões, código corretor de erro, processo de geração e processo de leitura [2].

4.1 Geração e Descrição de QR Codes

Os QR Codes atualmente encontrados no dia a dia contam com as cores preto e branco. É possível encontrar alguns QR Codes com outras cores (vermelho, verde, azul, rosa, etc), as quais se comportam como a cor preta no processo de decodificação, a finalidade dessas cores é de decoração estética na apresentação do QR Code e não o aumento de armazenamento de informação.

Os QR Codes contam com 40 versões, a versão 1 é a menor versão e conta com 21 x 21 módulos, as versões aumentam até a versão 40, aumentando de 4 em 4 módulos em altura e largura. O Apêndice A mostra as Figuras das versões 1, 2, 6, 7, 14, 21 e 40. O QR Code conta com a seguinte estrutura:

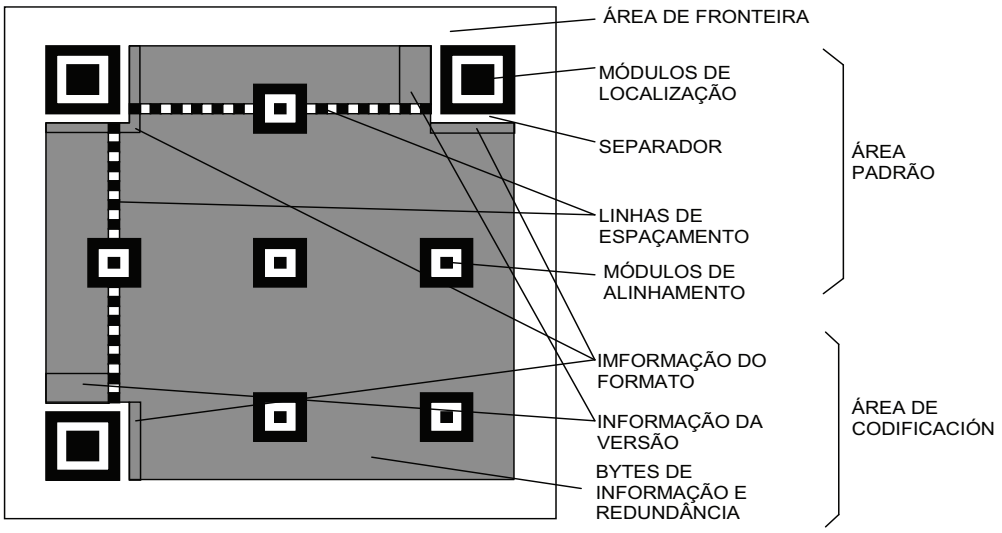

Figura 4.1: Estrutura do QR Code comum [2].

A descrição da Figura 4.1 é mostrada nos seguintes itens:

4.1.1 Área Padrão

- **Módulos**: Representação dos bits 1 e 0 de dimensão 1, 2^2, 3^2, ... , (n^2) *pixels* brancos ou pretos, de acordo ao tamanho desejado.
- **Área de fronteira**: Região livre fora das fronteiras dos módulos do QR Code (4 módulos de largura para cada eixo).
- **Módulos de localização**: Componentes para localização do QR Code.

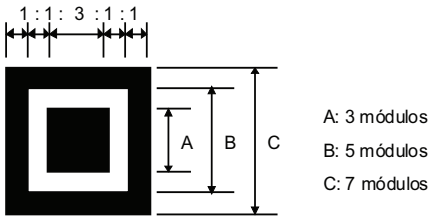

Figura 4.2: Estrutura do Módulo de localização do QR Code [2].

- **Separador**: Linha branca de 1 x 8 módulos que separa o Módulo de localização da Área de codificação.
- **Linhas de espaçamento**: Linha de módulos preto e branco posicionada de forma alternada em direção vertical e horizontal, a sua função é identificar o tamanho (em *pixels*) e a versão do QR Code.
- **Módulos de alinhamento**: Encontra-se nos QR Codes a partir da versão 2, cada Módulo de alinhamento conta com três quadrados concêntricos sobrepostos na ordem de um módulo preto, três módulos brancos e cinco módulos pretos de largura.

4.1.2 Área de Codificação

- **Informação de formato**: Seqüência de cinco bits de informação e dez bits de redundância, aparece duas vezes no QR Code informando o tipo de máscara e formato de correção de erro usado na área de codificação.
- **Informação de versão**: Seqüência de seis bits de informação e doze bits de redundância, os quais contém a informação da versão do QR Code. Os bits de informação de versão estão presentes nas versões de QR Code igual ou superior a sete, conforme mostrado na Figura 4.3.

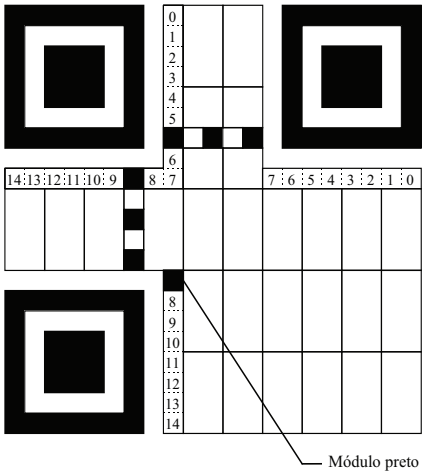

Figura 4.3: Informação de formato no QR Code [2].

- **Bytes de informação e redundância**: Seqüência de bits de informação e de redundância, os bytes de informação e redundância dependem da versão e do formato de correção de erro escolhido.

As quarenta possíveis versões contam com diferentes números de módulos, começando pela versão 1 de 21 x 21 módulos e aumentando de quatro em quatro módulos por eixo em cada versão superior, finalizando na versão quarenta de 177 x 177 módulos como mostrado na Figura 4.4.

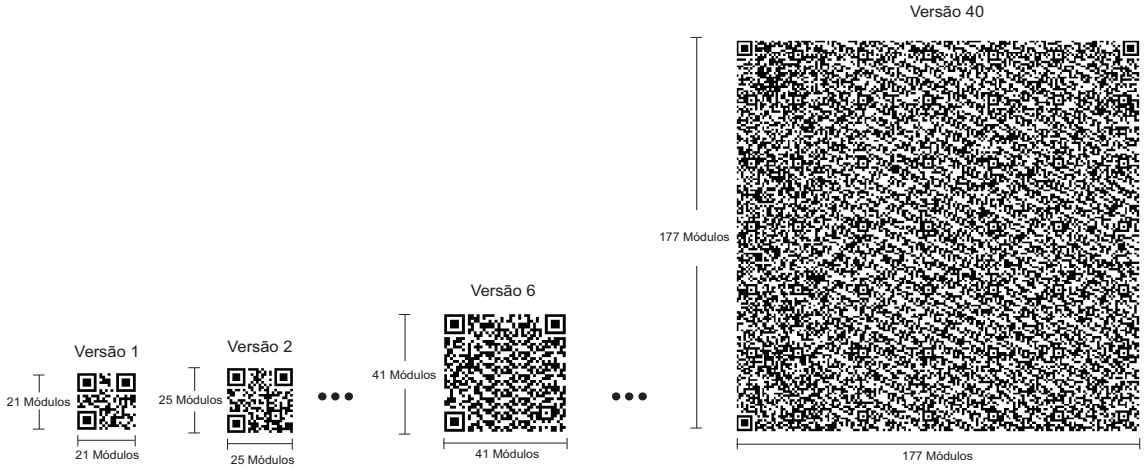

Figura 4.4: Equivalência entre versões e módulos do QR Code.

4.2 Correção de Erro na Área de Codificação

Conforme mencionado no capítulo anterior, os QR Codes utilizam o código corretor de erros Reed-Solomon. Para ser viável a recuperação de bits de informação no QR Code parcialmente corrompido, há quatro níveis de correção de erro, escolhidos na

geração de cada QR Code e detectado na decodificação do QR Code, os níveis são mostrados na Tabela 4.1.

Tabela 4.1: Níveis de correção de erro dos QR Codes.

Capacidade de correção de erro do QR Code	
Nível L	Aproximadamente 7%
Nível M	Aproximadamente 15%
Nível Q	Aproximadamente 25%
Nível H	Aproximadamente 30%

Quanto maior o nível de correção de erro, maior quantidade de bits de redundância e menor quantidade de bits de informação são requeridos na criação do QR Code.

A correção de erro do QR Code é implementada usando o corretor de blocos Reed-Solomon, a capacidade de correção de erro depende da quantidade de informação que será corrigida.

4.3 Posicionamento de Bits no QR Code

Os bits de informação e os bits de redundância são colocados no QR Code começando do extremo inferior direito no sentido superior, uma vez alcançado o extremo superior da área de codificação, retorna seu trajeto no sentido inferior conforme mostrado na Figura 4.5.

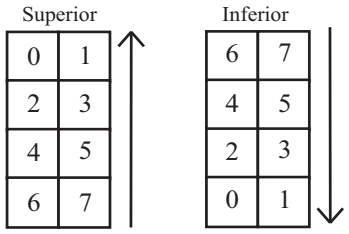

Figura 4.5: Sentidos de posicionamento do QR Code [2].

Na área de codificação os bytes são posicionados em colunas de dois módulos no sentido horizontal por quatro módulos no sentido vertical, sendo o bit más significativo o 7 e o menos significativo o 0 na Figura 4.5. O posicionamento de bytes para os casos onde estejam presentes os módulos de alinhamento no trajeto dos bytes é mostrado na Figura 4.6. Quando um byte encontra um limite horizontal, seja um módulo de localização ou alguma fronteira do QR Code, alguns bits restantes deverão ser acomodados na coluna da esquerda.

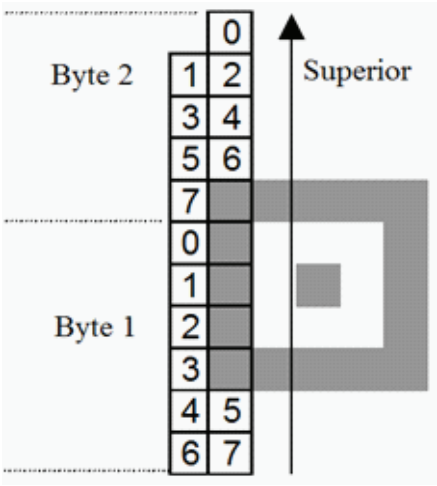

Figura 4.6: Exemplo de posicionamento na presença de um módulo de alinhamento [2].

Uma vez completado o posicionamento dos bytes no QR Code, como mostrado na linha vermelha da Figura 4.7, é verificado se foi preenchida toda a área de codificação, caso seja necessário, o final do QR Code poderá ser completado com bits restantes 0.

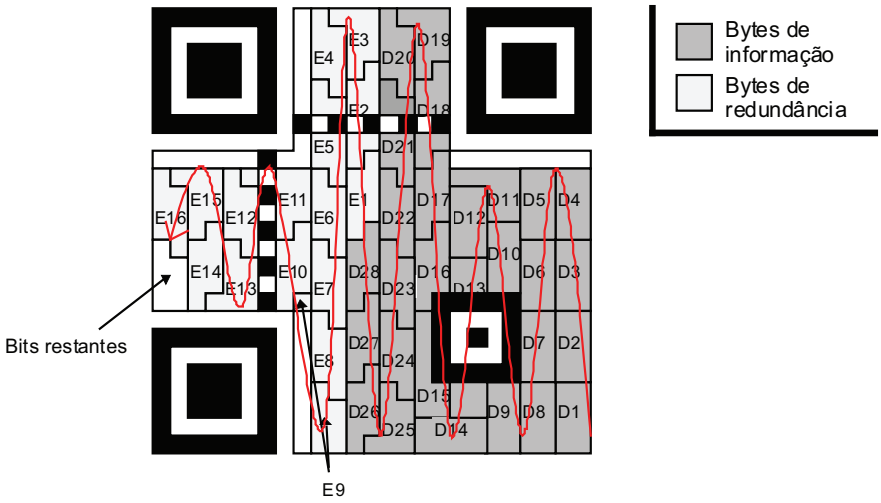

Figura 4.7: Posicionamento de bits no QR Code de versão 2 e formato M [22].

4.4 Mascaramento de Bits

Para que a leitura do QR Code seja efetiva, é preferível contar com um número equilibrado de módulos pretos e brancos, particularmente evitando a presença do padrão dos módulos de localização 1011101 (preto : branco : 3 pretos : branco : preto). Para atingir esse objetivo, os seguintes passos são considerados:

- O mascaramento não é aplicado aos módulos de localização.
- Executar a operação OR EXCLUSIVO (XOR) em todos os módulos da área de codificação (com exceção da área de informação de formato e de informação de

versão) do QR Code criado anteriormente com cada una das oito matrizes padrão mostradas na Figura 4.8.

A condição de geração das matrizes padrão está em função das fórmulas de geração mostradas na Figura 4.8. O processo de posicionamento das máscaras segue a ordem dos indicadores de fila i e dos indicadores de coluna j, sendo $(i,j) = (0,0)$ a posição superior esquerda da máscara.

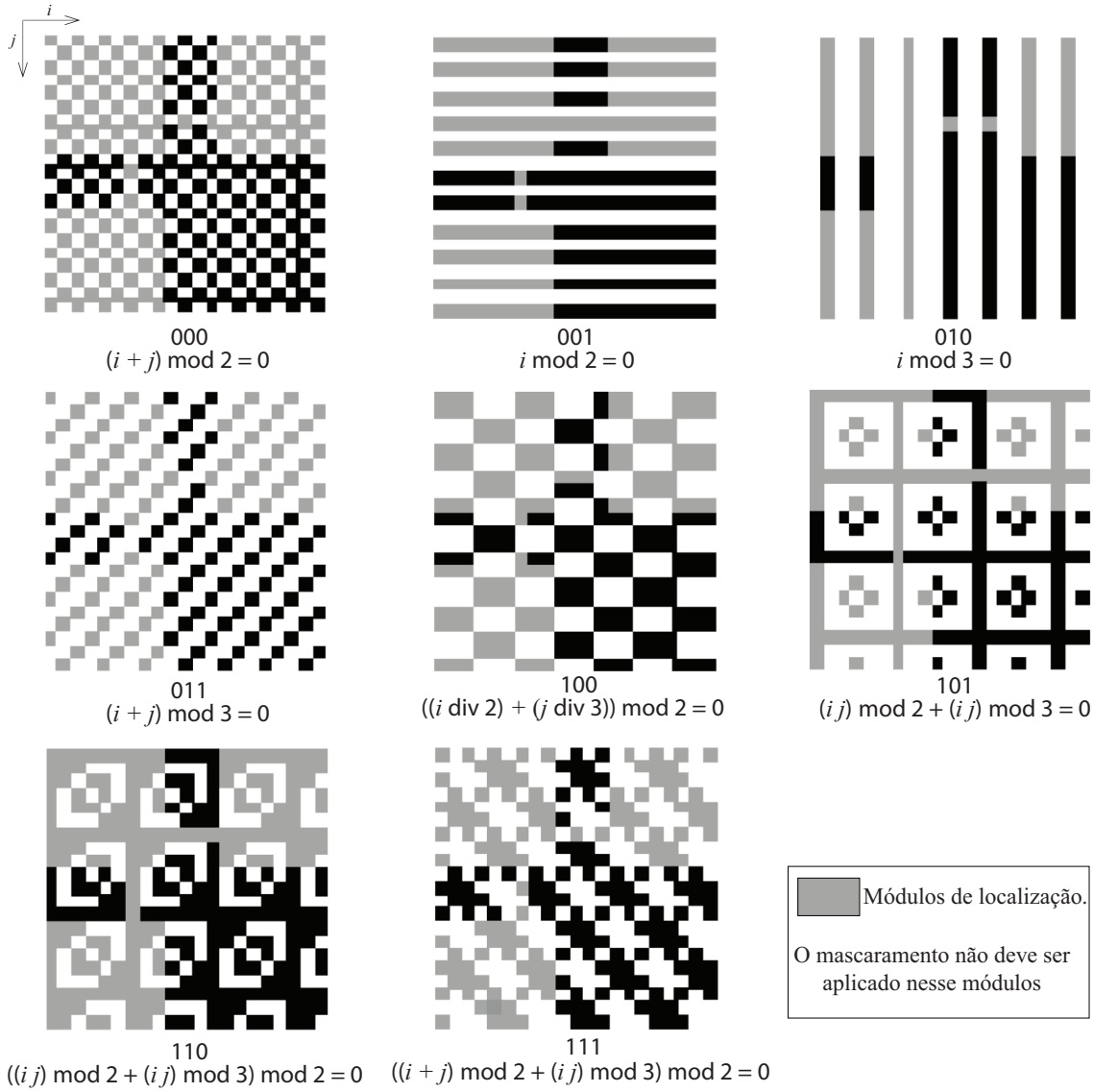

Figura 4.8: Oito máscaras de QR Code [22].

Como resultado da operação XOR das oito máscaras e o QR Code gerado, são gerados oito candidatos para ser o QR Code definitivo. Uma vez gerados os oito QR Codes candidatos, o próximo passo é avaliar cada QR Code candidato mediante julgamento por pontos de penalidade. Em quanto maior seja o número de pontos de penalidade por cada QR Code mascarado, menor será sua aceitação.

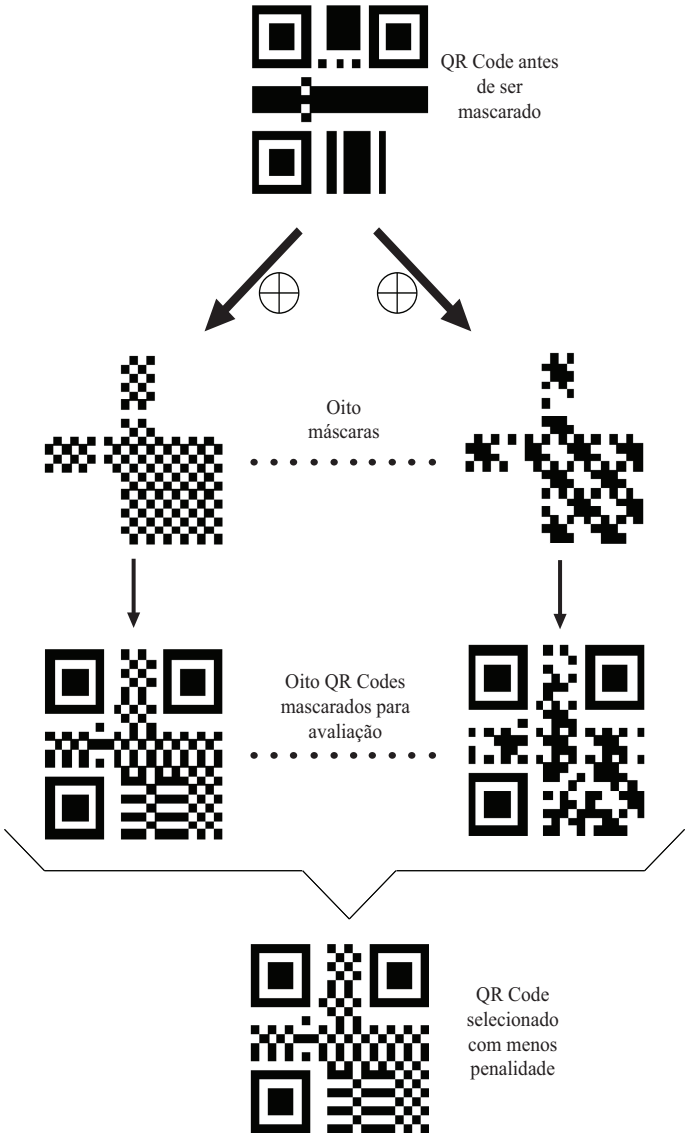

Figura 4.9: Processo de escolha dos QR Code mascarados [22].

As penalidades são mostradas na Tabela 4.2, onde os valores de N são N_1=3, N_2=3, N_3=40 e N_4=10.

A máscara escolhida é colocada no QR Code no campo de informação de formato. De acordo ao resultado obtido após o mascaramento e a avaliação das penalidades é gerado o QR Code final.

4.5 Decodificação do QR Code

O processo de decodificação é exatamente o processo inverso ao de codificação ilustrado na Figura 4.10.

Tabela 4.2: Penalidades de QR Codes mascarados.

Características	**Condição de penalidade**	**Pontos**
Módulos adjacentes em fila ou coluna da mesma cor	Número de módulos = (5+i)	N_1+i
Bloco de módulos da mesma cor	Tamanho do bloco = m x n	N_2 x (m-1) x (n-1)
Módulos de localização (preto : banco : preto : branco : preto) ou módulos de localização em fila/coluna, precedido ou seguido por 4 módulos brancos	Existência do módulo de localização	N_3
Proporção de módulos pretos no QR Code	50±(5 x k)% até 50±(5 x (k+1))%	N_4 x k

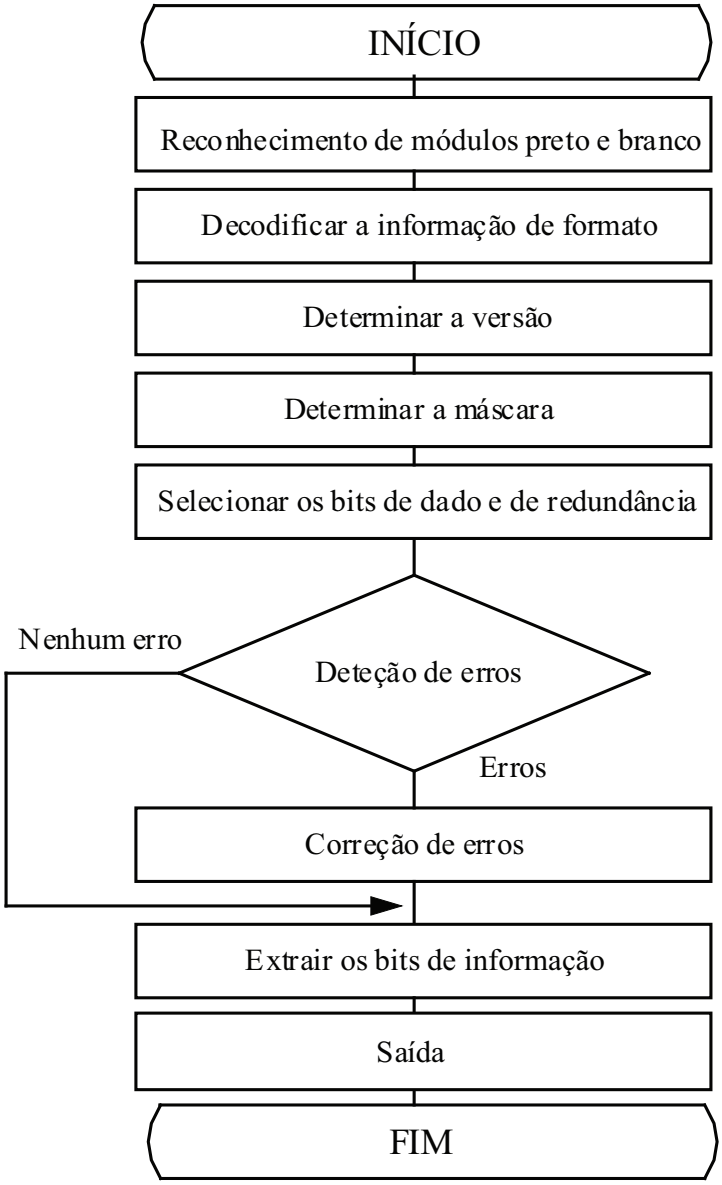

Figura 4.10: Processo de decodificação da imagem do QR Code [22].

5 SOLUÇÃO PROPOSTA: CQR CODES

Nesse Capítulo será explicado detalhadamente o processo de geração e leitura dos CQR Codes propostos. No período de realização desse trabalho foram desenvolvidos dois modelos de CQR Codes com a mesma estrutura física e diferente capacidade de armazenamento e transmissão de bits.

5.1 Primeiro Modelo de CQR Code

5.1.1 Estrutura do Primeiro Modelo de CQR Code

O primeiro modelo de CQR Code proposto conta com cinco cores (branco, preto, vermelho, verde e azul) e com a mesma estrutura do QR Code comum na versão 8 (49 x 49 módulos) com exceção de que o primeiro modelo de CQR Code proposto não tem os campos de informação de formato nem informação de versão já que ele é desenhado para armazenar exatamente 1024 bits de informação e 3392 bits de redundância. O processo de geração e leitura do primeiro modelo de CQR Code proposto foi implementado no ambiente de simulação iterativo Matlab. Os módulos que contém dados de informação e redundância estão distribuídos na área de codificação, os módulos de área de padrão são módulos dedicados exclusivamente para o reconhecimento da existência de um CQR Code na imagem. A estrutura está mostrada na Figura 5.1.

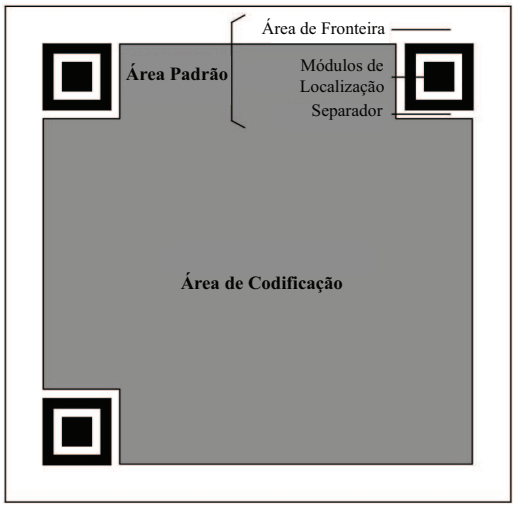

Figura 5.1: Estrutura do primeiro e segundo modelo de CQR Codes [7].

5.1.2 Capacidade de Armazenamento do Primeiro Modelo de CQR Code

O CQR Code descrito nesse item é o primeiro QR Code a cores robusto tanto na codificação como na decodificação, foi adotado o QR Code versão 8 como referência de tamanho em módulos para a geração do nosso CQR Code. O CQR Code, conforme a versão 8 dos QR Codes, tem 49 x 49 módulos, desses 2401 módulos, 147 módulos são destinados para os Módulos de Localização, 45 módulos para os Separadores e 2209 módulos para a Área de Codificação conforme mostrado na Tabela 5.1. O primeiro modelo de CQR Code conta com 4 cores dedicadas para armazenamento de bits (branco, vermelho, verde e azul) e duas cores para os módulos de localização (branco e preto), dessa maneira é possível armazenar dois bits em cada módulo no lugar de um bit como é feito no QR Code tradicional como mostrado na Tabela 5.2.

Tabela 5.1: Distribuição dos módulos no primeiro modelo de CQR Code.

	Total	**Módulos de Localização**	**Separadores**	**Área de Codificação**
Módulos	49 x 49 = 2401	64 x 3 = 192	15 x 3 = 45	49 x 49 + 64 x 3 - 45 x 3 = 2209

Tabela 5.2: Equivalência de cores de módulos e bits por módulo no primeiro modelo de CQR Code.

Cor	**Vermelho**	**Verde**	**Azul**	**Branco**
Bits	00	01	10	11

Conforme mostrado na Tabela 5.1, na Área de Codificação o primeiro modelo de CQR Code utiliza 2208 módulos dos 2209 módulos disponíveis, ou seja, utiliza 4416 bits, dos quais 1024 são de informação e 3392 são de redundância. O módulo restante ou dois bits restantes não são utilizados por ser restantes da operação matemática da geração de bits de redundância.

5.1.3 Geração do Primeiro Modelo de CQR Code

O primeiro modelo de CQR Code foi gerado de forma matricial no ambiente de simulação iterativo MATLAB com a ferramenta *Image Processing Toolbox* [22]. Uma vez obtidos os 1024 bits de informação, devemos gerar os 3392 bits de redundância, para isso, foi utilizada a função *rsenc* da ferramenta MATLAB *Communications Toolbox* [23], cuja finalidade é criar os símbolos de redundância Reed-Solomon a partir dos 1024 bits ou 64 símbolos de informação (cada símbolo conta com s=16 bits). Mediante a geração de Campos de Galois compostos por 64 símbolos de informação como entrada representados por D_x, são gerados 212 símbolos de redundância representados por RS_x (Reed-Solomon) em um único Campo de Galois, como mostrado na Equação 5.1.

$$RS(276, 64) = [D_1 \cdots D_{64}\ RS_1 \cdots RS_{212}]. \tag{5.1}$$

Cada símbolo de informação e de redundância contém 16 bits por símbolo, ou seja, $s = 16$, a quantidade de bits por símbolo não pode ser menor, nesse caso 8, porque somente com s=8 a Equação 3.2 não seria válida. No caso de s=8 teríamos:

$$n = 2^8 - 1 \tag{5.2}$$

$$n = 255 \tag{5.3}$$

No caso de ser n=255 e s=8, teríamos como máximo n x 8 = 2040 bits disponíveis, entre bits de informação e redundância, para armazenar no CQR Code com capacidade de armazenamento de 2209 x 2 = 4418 bits o que não satisfaz a capacidade de armazenamento no CQR Code, é por isso que são utilizados 16 bits por símbolo na geração de símbolos de redundância no algoritmo Reed-Solomon encurtado.

O polinômio gerador utilizado para executar o algoritmo Reed-Solomon conforme a Equação 5.1 foi:

$$p(D) = D^{16} + D^{12} + D^3 + D + 1. \tag{5.4}$$

De acordo com as Equações 3.1 e 5.1 o valor de t é 106, o que significa que mesmo se $38,40\%$ dos símbolos de informação e de redundância são corrompidos, será possível recuperar os 64 símbolos ou 1024 bits de informação no CQR Code, nesse caso esse é o limite teórico de recuperação dos símbolos de informação.

À diferença dos QR Codes tradicionais, o CQR Code utiliza a cor preta somente para os Módulos de Localização, dessa maneira não há necessidade de distinguir os módulos da Área de Codificação com os dos Módulos de Localização e assim não é utilizado o mascaramento que o QR Code utiliza.

Uma vez que são obtidos todos os símbolos de informação e redundância, eles são convertidos para bits e com base neles é possível começar o preenchimento do CQR Code seguindo a ordem de equivalência mostrada na Tabela 5.2. O sentido de preenchimento utilizado no CQR Code é mais simples e direto em comparação ao sentido

de preenchimento do QR Code mostrado na Figura 4.7. O sentido de preenchimento do CQR Code começa no canto inferior direito e termina no canto superior esquerdo, como mostrado na Figura 5.2.

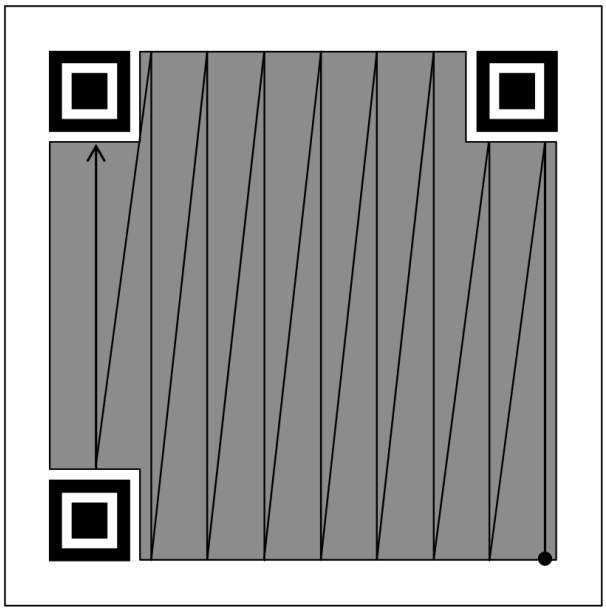

Figura 5.2: Posicionamento dos módulos na Área de Codificação dos CQR Codes [7].

Finalmente após o preenchimento obtemos o primeiro modelo de CQR Code mostrado na Figura 5.3.

5.1.4 Leitura do Primeiro Modelo de CQR Code

Figura 5.3: Exemplo do primeiro modelo de CQR Code utilizado no decorrer da dissertação.

A leitura do primeiro modelo de CQR Code foi realizada com a câmera de 3.2 megapixels do celular *Nokia 5800 XpressMusic*. O primeiro modelo de CQR Code é

lido de forma onidirecional. Foram impressos dois tamanhos do primeiro modelo de CQR Code para decodificação, ambos possuem tamanho mostrado nas Figuras 5.4 e 5.5, respectivamente.

Figura 5.4: Primeiro modelo de CQR Code - Menor 1, 3cm x 1, 3cm.

Figura 5.5: Primeiro modelo de CQR Code - Médio 2, 6cm x 2, 6cm.

O processo de leitura do primeiro modelo de CQR Code baseiam-se em um conjunto de técnicas de processamento de imagens. Os resultados obtidos foram satisfatórios para todos os tamanhos de impressão, foi indispensável a impressão na melhor qualidade possível, nesse caso as impressões foram feitas na impressora *Xerox WorkCentre M24*. Os resultados obtidos foram satisfatórios desde o ponto de vista de decodificação para todos os tamanhos de impressão. A seguir é explicado o processo de decodificação acompanhado de um exemplo de leitura do primeiro modelo de CQR Code.

O primeiro passo é capturar a imagem do primeiro modelo de CQR Code para que sirva como entrada do processamento de imagem como mostrado na Figura 5.6, cujo tamanho de impressão é de 1, 3cm x 1, 3cm.

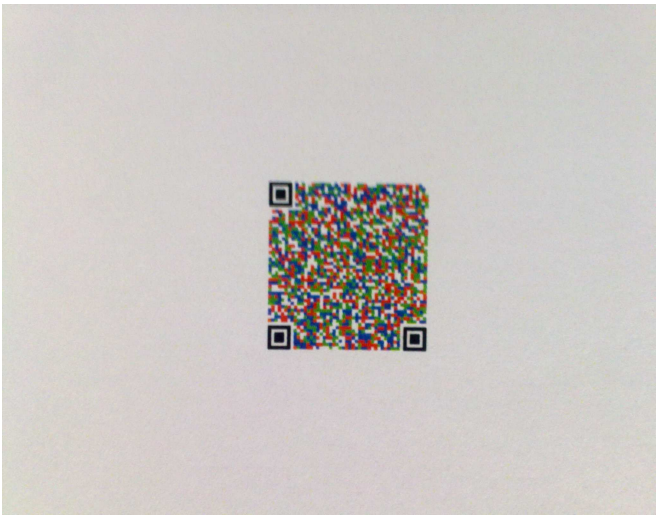

Figura 5.6: Imagem adquirida do primeiro modelo de CQR Code - Menor 1, 3cm x 1, 3cm.

Uma vez obtida a imagem é necessário o reconhecimento das coordenadas do centro dos Módulos de Localização. Em função dos diferentes cenários de luminosidade que podem ser tiradas as fotos, as vezes os *pixels* brancos em algumas imagens podem estar mais perto do valor (255, 255, 255) do que em outras imagens, o equivalente acontece com a cor preta e o valor (0, 0, 0). É por isso que é necessário a utilização de um parâmetro que possa indicar um limiar com o valor médio entre os *pixels* que devem ser brancos e pretos para que dessa maneira possam ser identificados os Módulos de Localização. O método adotado para encontrar esse limiar está baseado no uso do histograma da imagem original em tons de cinza. Foi percebido que as imagens dos CQR Codes seguem um padrão de histograma com dois valores pico correspondente aos valores preto e branco da imagem, os valores dos *pixels* coloridos vão ficar esparzidos no meio da cor branca e preta no histograma da imagem em tons de cinza. A Figura 5.7 mostra um histograma típico de uma imagem adquirida.

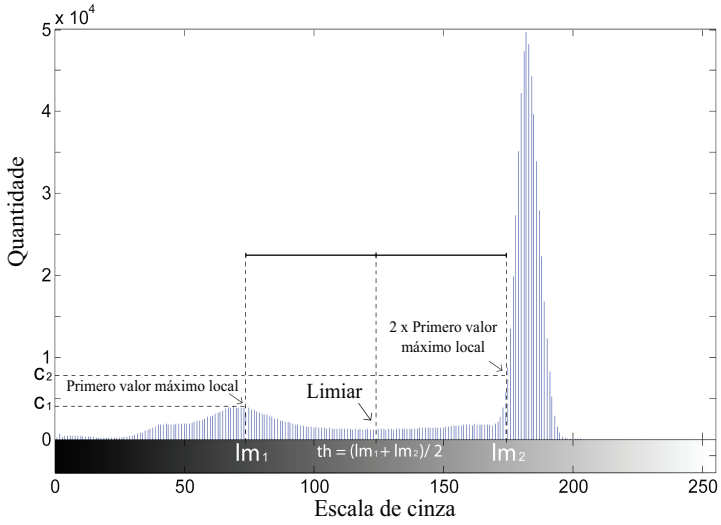

Figura 5.7: Histograma típico utilizado para calcular o limiar th.

No histograma da Figura 5.7 é encontrado o primeiro valor máximo entorno do valor 30 até 80 (valor aproximado dos *pixels* escuros), após encontrado o primeiro valor máximo é percorrido o histograma em sentido crescente no eixo X até encontrar o valor do dobro ou maior do que o dobro do primeiro valor máximo (sem a necessidade de utilizar o último pico do histograma), o ponto meio de lm_1 e lm_2 no eixo X da Figura 5.7 será o limiar mais indicado para a diferenciação dos *pixels* brancos e pretos. Esse limiar é variável de imagem para imagem em função da luminosidade da foto e sempre procura o valor médio adequado entre os *pixels* que representam as cores branco e preto.

Com o valor do limiar disponível, nesse caso com o valor de 139, a imagem original é transformada para as cores primárias mais a cor preta e branca. As regras para

determinar os valores vermelho, verde, azul, branco e preto são:

- **Preto**: Os valores das três coordenadas são próximas e estão abaixo do limiar.
- **Branco**: Os valores das três coordenadas são próximas e estão acima do limiar.
- **Vermelho**: O valor da coordenada R é muito mais alto do que o valor das coordenadas G e B além das coordenadas G e B serem próximas.
- **Verde**: O valor da coordenada G é muito mais alto do que o valor das coordenadas R e B além das coordenadas R e B serem próximas.
- **Azul**: O valor da coordenada B é muito mais alto do que o valor das coordenadas R e G além das coordenadas R e G serem próximas.

Seguindo as regras acima é gerada a imagem da Figura 5.8.

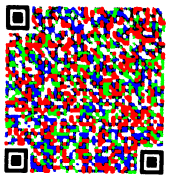

Figura 5.8: Imagem original com as cores primárias ressaltadas.

Aproveitando a natureza onidirecional da proporcionalidade dos Módulos de Localização (1 módulo preto : 1 módulo branco : 3 módulos pretos : 1 módulo branco : 1 módulo preto) é possível detectar o meio de cada Módulo de Localização, vale a pena lembrar que na execução desse algoritmo para encontrar os Módulos de Localização a presença de um *pixel* que não seja nem branco nem preto na contagem da proporcionalidade descrita zera o algoritmo. Em função dos pontos meios dos Módulos de Localização encontrados é possível estimar o valor em *pixels* de cada módulo. Por meio das coordenadas de cada centro de Módulo de Localização fazemos a média das

distâncias dos dois Módulos de Localização externos para o Módulo de Localização central:

$$P_m = \frac{\sqrt{(ML1_y - MLC_y)^2 + (ML1_x - MLC_x)^2} + \sqrt{(ML2_y - MLC_y)^2 + (ML2_x - MLC_x)^2}}{2}. \quad (5.5)$$

Onde P_m é o valor médio de *pixels* por módulo no eixo x e y, $ML1_y$ é a coordenada y de um dos dois Módulos de Localização não central, $ML1_x$ é a coordenada x de um dos dois Módulos de Localização não central, $ML2_y$ é a coordenada y do outro Módulos de Localização não central, $ML2_x$ é a coordenada x do outro Módulos de Localização não central, MLC_y é a coordenada y do Módulo de Localização central e MLC_x é a coordenada x do Módulo de Localização central.

Figura 5.9: Imagem recortada.

Por meio dos valores centrais dos três módulos de localização é possível realizar a rotação precisa para posicionar o CQR Code da maneira adequada para a extração de bits nos respectivos módulos, a Figura 5.10 mostra o processo de rotação.

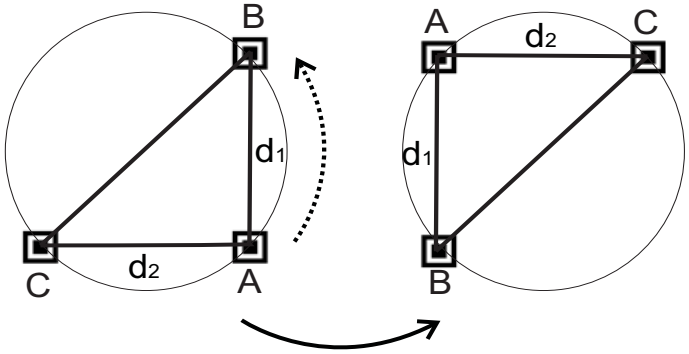

Figura 5.10: Correção de rotação.

Após a rotação a imagem original cortada e rotada corretamente é ressaltada nas cores vermelho, verde, azul e branco desconsiderando a cor preta dado que os Módulos

de Localização já cumpriram o objetivo de rotação e localização. A Figura 5.11 mostra o CQR Code na posição ideal para a decodificação.

Após a correção de rotação do CQR Code é executado o mapeamento de cada módulo por quadrante partindo da esquina do cuadrante em direção ao centro do CQR Code, conforme a quantidade de *pixels* por módulo estimada anteriormente, são mapeados os módulos do CQR Code desde o ponto inicial por quadrante pulando a cada P_m *pixels* no eixo x e y, em cada pulo é mapeado o valor determinístico do módulo, há um ajuste do valor de P_m no eixo x e y em casos que os Módulos de Localização vizinhos, após a rotação e posição ideal tenham diferença significativa nos valores do eixo x e y, essa diferença é originada por causa do ângulo da câmera em relação ao CQR Code impresso. A correção é mostrada nas Figuras 5.12 e 5.13 e na Equação 5.3.

O método de fator de correção aplicado no terceiro quadrante arrasta uma quantidade diretamente proporcional de *pixels* para a direita a medida que o mapeamento é acrescentado no eixo y, o objetivo desse método é corrigir erros de posicionamento que são arrastados a medida que é feito o mapeamento de cada quadrante do CQR Code. A quantidade de *pixels* que serão arrastados para a direita estão mostrados na Equação 5.6 com base na Figura 5.13.

Figura 5.11: Imagem recortada, rodada e com as cores primárias ressaltadas exceto a cor preta.

$$P_x = P_y \left(\frac{P_c x - P_i x}{P_i y - P_c y} \right). \tag{5.6}$$

Onde P_y é a quantidade de *pixels* no eixo y que o decodificador avançou a partir do Módulo de Localização, $P_c x$ é a coordenada x do Módulo de Localização central (nesse caso 47), $P_i x$ é a coordenada x do Módulo de Localização inferior (terceiro quadrante, nesse caso 40), $P_i y$ é a coordenada y do Módulo de Localização inferior e $P_c y$ é a coordenada y do Módulo de Localização central. O fator de correção é aplicado de

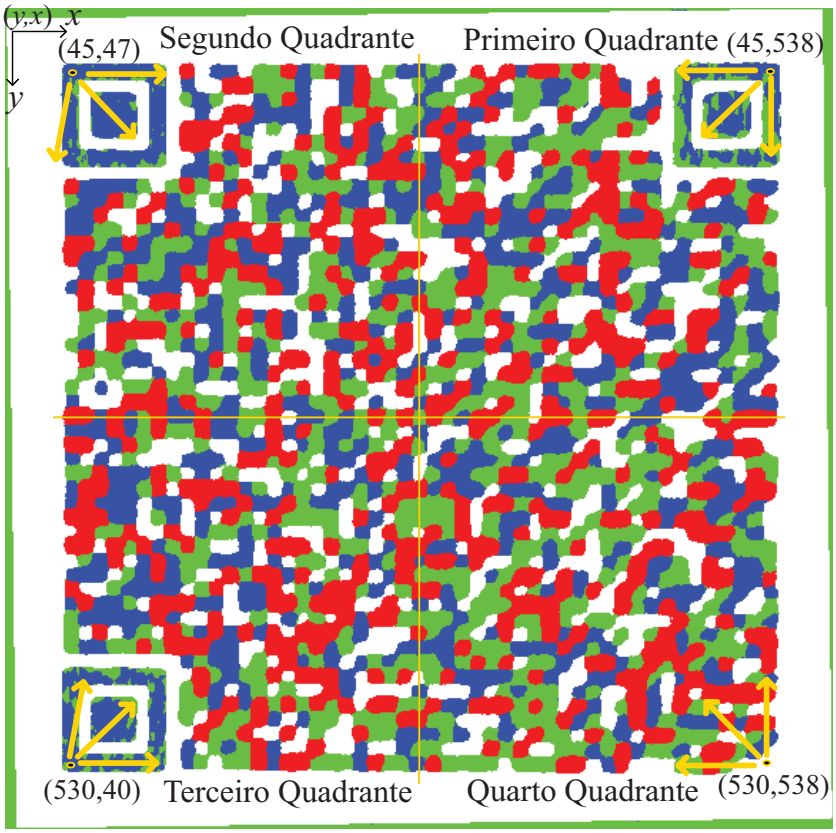

Figura 5.12: Sentido do mapeamento determinístico dos módulos do CQR Code.

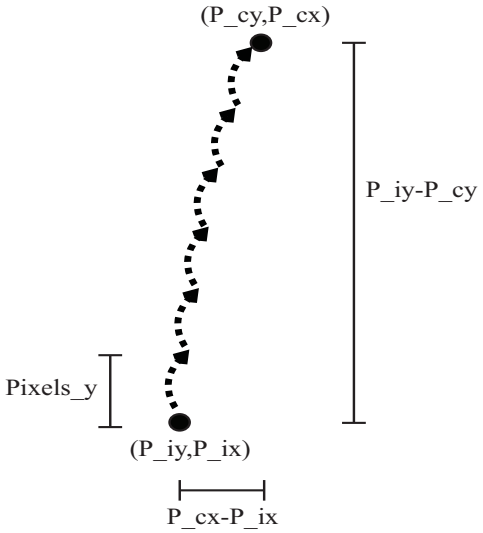

Figura 5.13: Fatores de correção do terceiro quadrante.

forma similar nos quatro quadrantes tanto na direção x e y conforme exemplo anterior e está ilustrado na Figura 5.13.

O processo de decodificação do primeiro modelo de CQR Code mostrado teve 16 símbolos corrompidos, ou seja, a penas $5,8\%$ de erro, dessa maneira, a capacidade de correção de erro de $38,40\%$ permitiu recuperar os 64 símbolos de informação.

5.2 Segundo Modelo de CQR Code

5.2.1 Estrutura do Segundo Modelo de CQR Code

O segundo modelo de CQR Code proposto conta com nove cores (branco, preto, vermelho, verde, azul, ciano, amarelo, magenta e cinza) e com a mesma estrutura do primeiro modelo de CQR Code. O segundo modelo de CQR Code é desenhado para armazenar exatamente 2048 bits de informação e 4576 bits de redundância. O processo de geração e leitura do primeiro modelo de CQR Code proposto também foi implementado no ambiente de simulação iterativo Matlab. O segundo modelo de CQR Code mantém a mesma estrutura do primeiro modelo de CQR Code mostrado na Figura 5.1.

5.2.2 Capacidade de Armazenamento do Segundo Modelo CQR Code

Para o segundo modelo de CQR Code também foi adotado o QR Code versão 8 como referência de tamanho em módulos para a geração do nosso CQR Code. O CQR Code versão 8 tem 49 x 49 módulos, desses 2401 módulos, 147 módulos são destinados para os Módulos de Localização, 45 módulos para os Separadores e 2209 módulos para a Área de Codificação conforme mostrado na Tabela 5.1. A principal diferença entre o segundo e o primeiro modelo de CQR Code é que o segundo modelo conta com 8 cores dedicadas para armazenamento de bits (branco, vermelho, verde, azul, ciano, amarelo, magenta e cinza) e duas cores para módulos de localização (branco e preto), dessa maneira é possível armazenar três bits em cada módulo, no lugar de um bit como é feito no QR Code tradicional ou dois bits no primeiro modelo de CQR Codes, conforme mostrado na Tabela 5.3.

Tabela 5.3: Equivalência de cores de módulos e bits por módulo no segundo modelo de CQR Code.

Cor	**Vermelho**	**Verde**	**Azul**	**Ciano**	**Magenta**	**Amarelo**	**Branco**	**Cinza**
Bits	000	001	010	011	100	101	110	111

Conforme mostrado na Tabela 5.1, na Área de Codificação o segundo modelo de CQR Code utiliza 2208 módulos dos 2209 módulos disponíveis, ou seja, o segundo modelo de CQR Code utiliza 6624 bits, dos quais 2048 são de informação e 4576 são de redundância. O módulo restante ou três bits restantes não são utilizados por ser restantes da operação matemática da geração de bits de redundância.

5.2.3 Geração do Segundo Modelo de CQR Code

O segundo modelo de CQR Code também foi gerado de forma matricial no ambiente de simulação iterativo MATLAB com a ferramenta *Image Processing Toolbox* [22]. Uma vez obtidos os 2048 bits de informação, devemos gerar os 4576 bits de redundância, para isso, foi utilizada a função *rsenc* da ferramenta MATLAB *Communications Toolbox* [23], cuja finalidade é criar os símbolos de redundância Reed-Solomon a partir dos 2048 bits ou 128 símbolos de informação (cada símbolo conta com s=16 bits). Mediante a geração de Campos de Galois compostos por 128 símbolos de informação como entrada representados por D_x, são gerados 286 símbolos de redundância representados por RS_x (Reed-Solomon) em um único Campo de Galois, como mostrado na Equação 5.7.

$$RS(414, 128) = [D_1 \cdots D_{128}\ RS_1 \cdots RS_{286}]. \tag{5.7}$$

No caso do segundo modelo de CQR Code, os símbolos devem contar com 16 bits cada um e ser executado o algoritmo Reed-Solomon encurtado pelo mesmo motivo do primeiro modelo de CQR Code.

O polinômio gerador utilizado para o segundo modelo de CQR Code foi o mesmo do primeiro modelo de CQR Code mostrado na Equação 5.1.

Conforme mostrado na Equação 5.7, o valor de n para o segundo modelo de CQR Code é de 414 e de k é 128, dessa maneira o valor de correção de símbolos t é de 143 símbolos, o que significa que mesmo se $34,54\%$ dos símbolos de informação e de redundância são corrompidos, vai ser possível recuperar os 128 símbolos ou 2048 bits de informação nesse modelo de CQR Code, nesse caso esse é o limite teórico de recuperação da palavra-código.

Pelo mesmo motivo do primeiro modelo de CQR Code, o segundo modelo de CQR Code não utiliza o mascaramento que o QR Code utiliza.

O preenchimento do segundo modelo de CQR Code é realizado da mesma maneira do primeiro modelo de CQR Code seguindo a ordem de equivalência mostrada na Tabela 5.3. O sentido de preenchimento utilizado no segundo modelo de CQR Code é o mesmo do primeiro modelo de CQR Code.

Finalmente após o preenchimento, é obtido o segundo modelo de CQR Code mostrado na Figura 5.14.

Figura 5.14: Exemplo do segundo modelo de CQR Code utilizado no decorrer da dissertação.

5.2.4 Leitura do segundo modelo de CQR Code

A leitura do primeiro modelo de CQR Code foi realizada com a câmera de 8 megapixels do celular *Samsung Galaxy SIII*. O segundo modelo de CQR Code também pode ser lido de forma onidirecional e foi impresso nos seguintes tamanhos:

Figura 5.15: Segundo modelo de CQR Code - Menor 1,3cm x 1,3cm.

Figura 5.16: Segundo modelo de CQR Code - Médio 2,6cm x 2,6cm.

O processo de leitura do segundo modelo de CQR Code é exatamente o mesmo processo de leitura do primeiro modelo de CQR Code. Os resultados obtidos foram satisfatórios para todos os tamanhos de impressão, foi indispensável a impressão na melhor qualidade possível, nesse caso as impressões foram feitas na impressora *Xerox WorkCentre M24*. A seguir serão mostrados os mesmos passos de decodificação em relação ao primeiro modelo de CQR Code, as explicações das técnicas de processamento de imagens são as mesmas, com exceção do último passo de mapeamento de bits onde a constelação do segundo modelo de CQR Code está mostrada na Tabela 5.3.

5.2.5 Identificação do Segundo Modelo de CQR Code

Mais uma vez, o primeiro passo é capturar a imagem do segundo modelo de CQR Code para que sirva como entrada do processamento de imagem mostrado na Figura 5.17, cujo tamanho de impressão é de 1, 3cm x 1, 3cm.

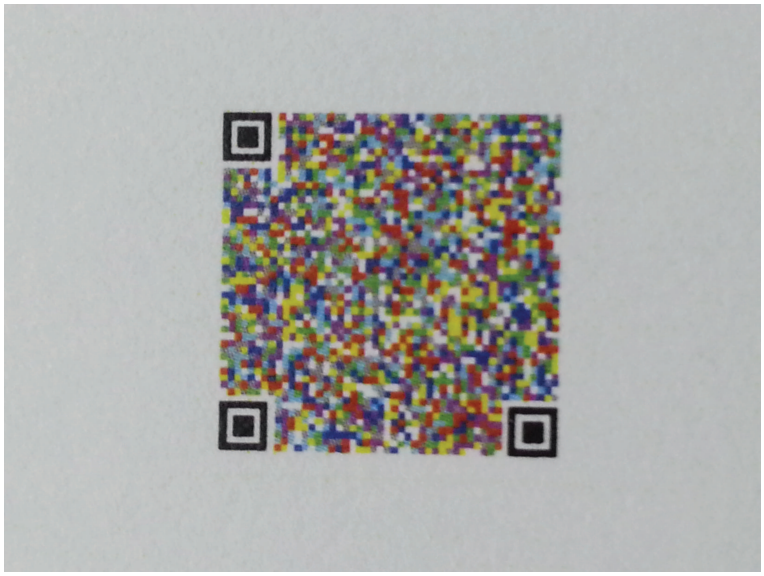

Figura 5.17: Imagem adquirida do segundo modelo de CQR Code - Menor 1, 3cm x 1, 3cm.

O segundo modelo de CQR Code, da mesma forma que no primeiro modelo de CQR Code, utiliza somente a cor preta para o reconhecimento dos Módulos de Localização, mais uma vez, em função dos diferentes cenários de luminosidade que podem ser tiradas as fotos, as vezes os *pixels* brancos em algumas imagens podem estar mais perto do valor (255, 255, 255) do que em outras imagens, o equivalente acontece com a cor preta e o valor (0, 0, 0). É por isso que o segundo modelo de CQR Code também utiliza o método de limiar de decisão por histograma em cima da imagem do segundo modelo de CQR Code em escala de cinza da mesma maneira que foi feito no primeiro modelo de CQR Code, a Figura 5.7 mostra um histograma típico de uma imagem adquirida.

Com o valor do limiar disponível, nesse caso 130, a imagem original é transformada para as cores primárias, secundárias e as cores preto e branco, no caso do segundo CQR Code as cores que não sejam preto nem branco serão consideradas vermelho (poderia ter sido outra cor) somente para facilitar a detecção dos Módulos de Localização. As regras para determinar os valores preto, branco e vermelho são:

- **Preto**: Os valores das três coordenadas são próximas e estão bem abaixo do limiar.

- **Branco**: Os valores das três coordenadas são próximas e estão bem acima do limiar.
- **Vermelho**: Caso o *pixel* não tenha sido considerado preto nem branco.

Dessa maneira é gerada a imagem da Figura 5.18:

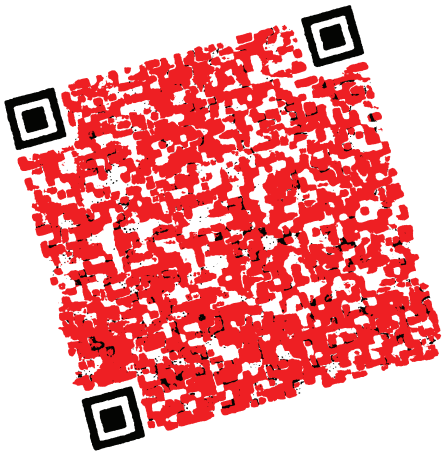

Figura 5.18: Imagem do segundo modelo de CQR Code com cores preto, branco e vermelho ressaltadas.

No segundo modelo de CQR Code é utilizado o mesmo método do primeiro modelo de CQR Code para encontrar os três Módulos de Localização e o valor de *pixels* por módulo para posteriormente também ser recortado conforme a Figura 5.19.

O processo de rotação no segundo modelo de CQR Code é realizado da mesma maneira que no primeiro modelo de rotação, a Figura 5.20 mostra o segundo modelo de CQR Code rodado e na posição ideal para a extração dos bits por módulo.

O próximo passo é mapear cada módulo para a respectiva cor designada na Tabela 5.3, para atingir esse objetivo foram criadas as seguintes regras de decisão:

- **Preto**: Caso a maior componente *RGB* dividido pelo limiar seja menor do que $0,3$.
- **Branco**: Caso a menor componente *RGB* dividido pelo limiar seja maior do que $1,3$.
- **Cinza**: Caso a maior e menor componente *RGB* dividido pelo limiar estejam entre $0,7$ e $1,3$.

Figura 5.19: Imagem do Segundo Modelo de CQR Code recortada.

Figura 5.20: Imagem do Segundo Modelo de CQR Code rodada.

- **Vermelho**: Caso a maior componente seja R, caso a componente R dividido pelo limiar seja maior do que $1,2$ e caso as componentes G e B divididas pelo limiar sejam menor do que $0,9$.

- **Verde**: Caso a maior componente seja G, caso a componente G dividida pelo limiar seja maior do que $1,2$ e caso as componentes R e B divididas pelo limiar sejam menor do que $0,9$.

- **Azul**: Caso a maior componente seja R, caso a componente R dividido pelo limiar seja maior do que $1,2$ e caso as componentes G e B divididas pelo limiar

sejam menor do que $0,9$.

- **Ciano**: Caso a menor componente seja R, caso a componente R dividido pelo limiar seja menor ou igual do que $1,1$, caso as componentes G e B divididas pelo limiar sejam maior do que $1,1$.

- **Amarelo**: Caso a menor componente seja B, caso a componente B dividido pelo limiar seja menor ou igual do que $1,1$, caso as componentes R e G divididas pelo limiar sejam maior do que $1,1$.

- **Magenta**: Caso a menor componente seja G, caso a componente G dividido pelo limiar seja menor ou igual do que $1,1$, caso as componentes R e G divididas pelo limiar sejam maior do que $1,1$.

As regras citadas foram elaboradas de forma empírica em função das imagens processadas.

A decisão do valor de cada módulo e a correção ilustrada na Figura 5.13 também são aplicados no segundo modelo de CQR Code.

O processo de decodificação do segundo modelo de CQR Code mostrado teve 103 símbolos corrompidos, ou seja, $24,88\%$ de erro, dessa maneira, a capacidade de correção de erro de $34,54\%$ permitiu recuperar os 128 símbolos de informação.

O fluxograma mostrado na Figura 5.21 ilustra o processo de decodificação utilizado para ambos modelos de CQR Code.

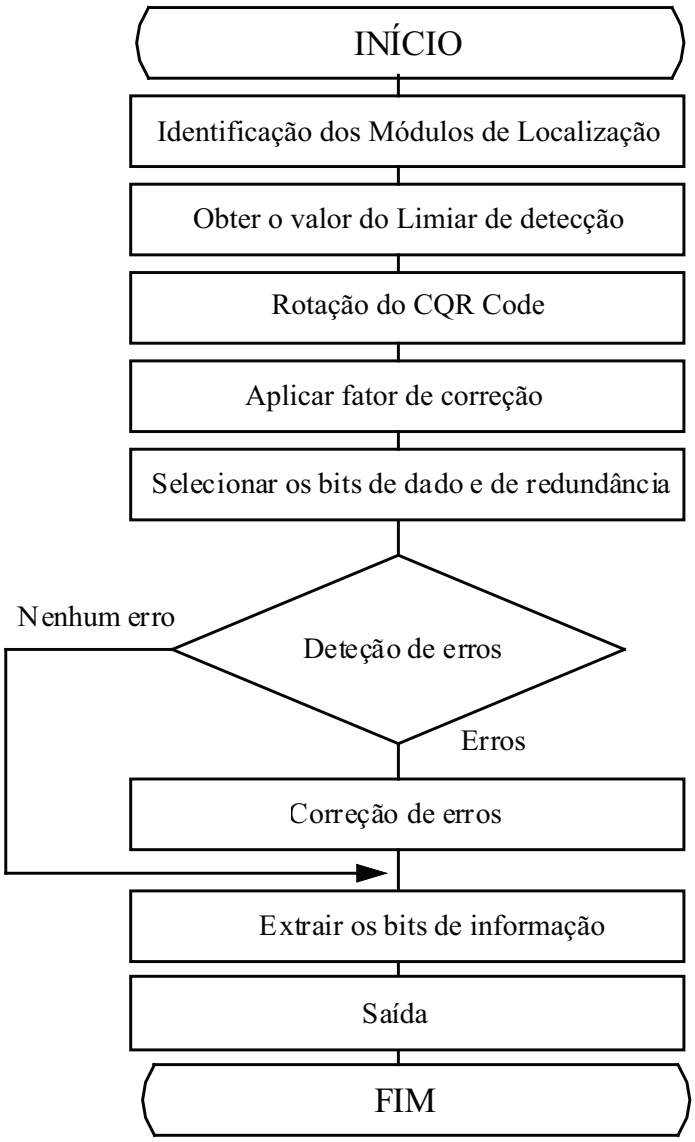

Figura 5.21: Processo de decodificação dos CQR Codes.

6 DECODIFICAÇÃO DO PRIMEIRO MODELO DE CQR CODE NAS COMPRESSÕES JPEG E JPEG2000

Nesse Capítulo será mostrado um procedimento de avaliação do comportamento da decodificação do primeiro modelo de CQR Code em cima de imagens com diversos graus de compressão nos padrões JPEG e JPEG2000. O objetivo desse capítulo é verificar o comportamento da percentagem de erro da decodificação do primeiro modelo de CQR Code em função da quantidade de bits por *pixels* que uma imagem comprimida pode ter nos dois padrões de compressão mencionados.

O procedimento é simples, basta definir uma imagem de CQR Code sem nenhum tipo de compressão no formato BMP como base para diversas compressões com diversas taxas de bits por *pixels*. O CQR Code base está ilustrado na Figura 5.4, essa figura contém a imagem de um primeiro modelo de CQR Code onde cada módulo é um *pixel*, para o processo de compressão a quantidade de módulos por *pixels* será importante dado que se é efetuada a compressão em cima da Figura 5.4 os *pixels* serão totalmente corrompidos e dessa maneira os módulos também. Por esse motivo foi ampliada com a técnica *nearest* [9] o CQR Code da Figura 5.4 dez, vinte e trinta vezes o tamanho original. O objetivo desse capítulo é avaliar a robustez da decodificação em imagens com baixo nível de Bits por *Pixel*.

6.1 Compressão JPEG

A compressão feita em cima dos três tamanhos de imagem foi realizada no ambiente de simulação MATLAB mediante a função *imwrite*, os três tamanhos do primeiro modelo de CQR Code são escritos no padrão JPEG, a função *imwrite*, no caso do JPEG, permite escolher a escrita entre os parâmetros de compressão 100 (menor compressão possível) e 0 (maior compressão possível). As imagens com aumento de 10, 20 e 30 vezes foram decodificadas com os parâmetros de compressão JPEG 100, 80, 60, 40, 30, 20, 10, 6, 4 e 0. Os resultados são mostrados nas Tabelas 6.1, 6.2 e 6.3.

As Figuras 6.1, 6.2 e 6.3 mostram as imagens aumentadas em tamanho por 10, 20 e 30 vezes, comprimidas com maior parâmetro de compressão 0.

Conforme mostrado nas Tabelas 6.1, 6.2 e 6.3 o decodificador do primeiro modelo de CQR Code decodificou todas as imagens para todos os níveis de compressão, atingindo taxas baixas de Bits por *Pixels* como de $0,3839$ para o tamanho aumentado

10 vezes, 0,2508 para o tamanho aumentado 20 vezes e 0,2270 para o tamanho aumentado 30 vezes. É possível verificar que a medida que a proporção de aumento da imagem é maior, a decodificação apresenta menor quantidade de símbolos errôneos, isso é normal devido ao mapeamento determinístico dos módulos que pulam entre os *pixels* centrais de todos os módulos. A medida que o módulo possui um tamanho maior é menos provável que o mapeamento determinístico de cada módulo mapeie o centro do módulo fora deste.

Tabela 6.1: %Erro de Decodificação x Bits por *Pixel* no tamanho aumentado 10 vezes - JPEG.

Compressão JPEG	**Símbolos iguais**	**Símbolos corrompidos**	**Status**	**%Erro**	**Arquivo em KB**	**BPP**
100	276	0	Aprovado	0,00%	243	5,0155
80	276	0	Aprovado	0,00%	95,1	1,9629
60	276	0	Aprovado	0,00%	71,3	1,4716
40	275	1	Aprovado	0,36%	57,9	1,1951
30	269	7	Aprovado	2,54%	50,6	1,0444
20	253	23	Aprovado	8,33%	39,7	0,8194
10	245	31	Aprovado	11,23%	28,9	0,5965
6	224	52	Aprovado	18,84%	23,6	0,4871
4	213	63	Aprovado	22,83%	20,6	0,4252
0	197	79	Aprovado	28,62%	18,6	0,3839

Tabela 6.2: %Erro de Decodificação x Bits por *Pixel* no Tamanho aumentado 20 vezes - JPEG.

Compressão JPEG	**Símbolos iguais**	**Símbolos corrompidos**	**Status**	**%Erro**	**Arquivo em KB**	**BPP**
100	276	0	Aprovado	0,00%	323	1,6667
80	276	0	Aprovado	0,00%	164	0,8462
60	276	0	Aprovado	0,00%	132	0,6811
40	276	0	Aprovado	0,00%	112	0,5779
30	276	0	Aprovado	0,00%	100	0,5160
20	276	0	Aprovado	0,00%	82,5	0,4257
10	268	8	Aprovado	2,90%	65,6	0,3385
6	262	14	Aprovado	5,07%	56,3	0,2905
4	261	15	Aprovado	5,43%	51	0,2632
0	257	19	Aprovado	6,88%	48,6	0,2508

6.2 Compressão JPEG2000

Mais uma vez, a compressão feita em cima dos três tamanhos de imagem foi realizada no ambiente de simulação MATLAB mediante a função *imwrite*, os três tamanhos do primeiro modelo de CQR Code são escritos no padrão JPEG2000, a função *imwrite*,

Tabela 6.3: %Erro de Decodificação x Bits por *Pixel* no Tamanho aumentado 30 vezes - JPEG.

Compressão JPEG	**Símbolos iguais**	**Símbolos corrompidos**	**Status**	**%Erro**	**Arquivo em KB**	**BPP**
100	276	0	Aprovado	0,00%	627	1,4379
80	276	0	Aprovado	0,00%	330	0,7568
60	276	0	Aprovado	0,00%	265	0,6077
40	276	0	Aprovado	0,00%	226	0,5183
30	276	0	Aprovado	0,00%	204	0,4678
20	276	0	Aprovado	0,00%	168	0,3853
10	276	0	Aprovado	0,00%	133	0,3050
6	276	0	Aprovado	0,00%	115	0,2637
4	276	0	Aprovado	0,00%	105	0,2408
0	276	0	Aprovado	0,00%	99	0,2270

Figura 6.1: CQR Code aumentado 10 vezes e comprimido com o maior parâmetro de compressão JPEG 0.

Figura 6.2: CQR Code aumentado 20 vezes e comprimido com o maior parâmetro de compressão JPEG 0.

no caso do JPEG2000, permite escolher a escrita sem limite de parâmetros de compressão. A taxa de compressão é definida por valores reais de compressão começando pelo valor 1 onde a compressão é mínima, seguido pelo valor 2 onde o tamanho da imagem, na teoria, deve ser a metade ou menor da metade do tamanho da imagem de

Figura 6.3: CQR Code aumentado 30 vezes e comprimido com o maior parâmetro de compressão JPEG 0.

entrada e assim sucessivamente. Enquanto maior o parâmetro de compressão menor será o tamanho do arquivo e menor será a qualidade da imagem.

A imagem com aumento de 10 vezes foi testada com os parâmetros de compressão JPEG2000 de 1, 20, 40, 60, 100, 150, 175, 200, 250 e 400. Os resultados são mostrados na Tabela 6.4.

Tabela 6.4: %Erro de Decodificação x Bits por *Pixel* no Tamanho aumentado 10 vezes - JPEG2000.

Compressão JPEG2000	**Símbolos iguais**	**Símbolos corrompidos**	**Status**	**%Erro**	**Arquivo em KB**	**BPP**
1	276	0	Aprovado	0,00%	294	6,0681
20	276	0	Aprovado	0,00%	58	1,1971
40	276	0	Aprovado	0,00%	29	0,5986
60	273	3	Aprovado	1,09%	19	0,3922
100	258	18	Aprovado	6,52%	12	0,2477
150	257	19	Aprovado	6,88%	8	0,1651
175	177	99	Aprovado	35,87%	7	0,1445
200	174	102	Aprovado	36,96%	6	0,1238
250	159	117	Reprovado	42,39%	5	0,1032
400	—	—	Não Decodificado	—	3	0,0619

A tabela 6.4 mostra a melhor performance que a compressão JPEG2000 possui e que o algoritmo de decodificação, para a imagem com aumento de 10 vezes consegue decodificar a imagem comprimida em até 0,1238 Bits por *Pixels*.

É possível perceber a baixa qualidade subjetiva da das Figuras 6.4, 6.5, 6.6 e 6.7 aumentadas 10 vezes. O decodificador do CQR Code decodificou Figura 6.6 porém obteve mais do que $t = 106$ símbolos corrompidos e não pode recuperar os 64 símbolos

de informação, já no caso da decodificação da Figura 6.7 o decodificador não conseguiu definir com precisão o centro dos três Módulos de Localização e dessa maneira o programa apresentou erro na decodificação.

A imagem aumentada 20 vezes foi testada com os parâmetros de compressão JPEG2000 de 1, 40, 100, 200, 400, 600, 700 e 800. Os resultados são mostrados na Tabela 6.5.

Figura 6.4: CQR Code aumentado x10 e comprimido com o maior parâmetro de compressão JPEG2000 175.

Figura 6.5: CQR Code aumentado 10 vezes e comprimido com o maior parâmetro de compressão JPEG2000 200.

Figura 6.6: CQR Code aumentado 10 vezes e comprimido com o maior parâmetro de compressão JPEG2000 250.

Para a imagem aumentada 20 vezes, o algoritmo consegue decodificar a imagem comprimida em até $0,0334$ Bits por *Pixels*.

As Figuras 6.8 e 6.9 foram decodificadas corretamente, já a Figura 6.10 não pode

Figura 6.7: CQR Code aumentado 10 vezes e comprimido com o maior parâmetro de compressão JPEG2000 400.

Tabela 6.5: %Erro de Decodificação x Bits por *Pixel* no Tamanho aumentado 20 vezes - JPEG2000.

Compressão JPEG2000	**Símbolos iguais**	**Símbolos corrompidos**	**Status**	**%Erro**	**Arquivo em KB**	**BPP**
1	276	0	Aprovado	0,00%	565	2,9154
40	276	0	Aprovado	0,00%	115	0,5934
100	276	0	Aprovado	0,00%	46	0,2374
200	269	7	Aprovado	2,54%	22,7	0,1171
400	253	23	Aprovado	8,33%	11,1	0,0573
600	251	25	Aprovado	9,06%	7,45	0,0384
700	179	97	Aprovado	35,14%	6,47	0,0334
800	—	—	Não Decodificado	—	5,64	0,0291

Figura 6.8: CQR Code aumentado 20 vezes e comprimido com o maior parâmetro de compressão JPEG2000 600.

ser decodificada pelo alto nível de embaçamento presente nos Módulos de Localização o que impossibilitou detectar a presença destes.

Na compressão JPEG2000 do CQR Code aumentado 30 vezes, foi obtido um melhor desempenho na taxa de Bits por *Pixels*, a imagem aumentada 30 vezes foi testada com os parâmetros 1, 50, 100, 200, 400, 600, 1000, 1500, 2000, 2250 e 2500, os resultados

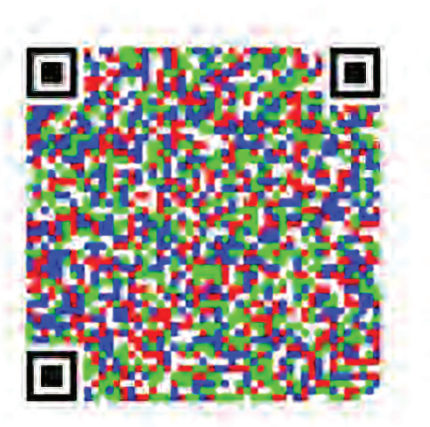

Figura 6.9: CQR Code aumentado 20 vezes e comprimido com o maior parâmetro de compressão JPEG2000 700.

Figura 6.10: CQR Code aumentado 20 vezes e comprimido com o maior parâmetro de compressão JPEG2000 800.

são mostrados na Tabela 6.6.

No caso das compressões mostradas na Tabela 6.6, o parâmetro de compressão JPEG2000 que o decodificador do CQR Code conseguiu processar foi muito mais alto do que os parâmetros de compressão que foram processados para as imagens aumentadas 10 e 20 vezes. A menor taxa de compressão de Bits por *Pixels* que foi corretamente decodificada para a imagem aumentada 30 vezes foi de $0,0102$. As Figuras 6.11 e 6.12 mostram, respectivamente, a imagem mais comprimida que foi corretamente decodificada e uma imagem comprimida cuja decodificação não teve sucesso.

Conforme mostrado nas Tabelas dessa subseção o decodificador do primeiro CQR Code decodificou todas as imagens cujo nível de embaçamento subjetiva não foi tão alto, somente não conseguiu decodificar imagens muito embaçada. Foram atingindas taxas baixas de Bits por *Pixels* como $0,1238$ para o tamanho aumentado 10 vezes, $0,0334$ para o tamanho aumentado 20 vezes e $0,0102$ para o tamanho aumentado 30 vezes. Mais uma vez, é possível verificar que a medida que a proporção de aumento da imagem é maior, a decodificação apresenta menor quantidade de símbolos errôneos,

isso é normal devido ao mapeamento determinístico dos módulos que pulam entre os *pixels* centrais de todos os módulos. A medida que o módulo possui um tamanho maior é menos provável que o mapeamento determinístico de cada módulo mapeie o centro do módulo fora deste.

As Figuras 6.13, 6.14 e 6.15 mostram o resultado comparativo de percentagem de Erro x BPP para os padrões JPEG e JPEG2000 para os tamanhos aumentados 10, 20 e 30 vezes respectivamente.

Tabela 6.6: %Erro de Decodificação x Bits por *Pixel* no aumentado 30 vezes - JPEG2000.

Compressão JPEG2000	**Símbolos iguais**	**Símbolos corrompidos**	**Status**	**%Erro**	**Arquivo em KB**	**BPP**
1	276	0	Aprovado	0,00%	818	1,8759
50	276	0	Aprovado	0,00%	208	0,4770
100	276	0	Aprovado	0,00%	104	0,2385
200	276	0	Aprovado	0,00%	51,9	0,1190
400	276	0	Aprovado	0,00%	25,7	0,0589
600	270	6	Aprovado	2,17%	17	0,0390
1000	229	47	Aprovado	17,03%	10	0,0229
1500	201	75	Aprovado	27,17%	6,8	0,0156
2000	181	95	Aprovado	34,42%	5	0,0115
2250	175	101	Aprovado	36,59%	4,46	0,0102
2500	153	123	Reprovado	44,57%	4	0,0092

Figura 6.11: CQR Code aumentado 30 vezes e comprimido com o maior parâmetro de compressão JPEG2000 2250.

O processo de decodificação de imagens comprimidas nos padrões JPEG e JPEG2000 teve um desempenho muito bom no quesito de somente não conseguir decodificar imagens muito embaçadas, houveram taxas de compressão de Bits por *Pixels* muito baixas que foram decodificadas. No padrão JPEG a melhor taxa de compressão de Bits por *Pixels* foi de 0,2270 e no padrão JPEG2000 foi de 0,0102.

Figura 6.12: CQR Code aumentado 30 vezes e comprimido com o maior parâmetro de compressão JPEG2000 2500.

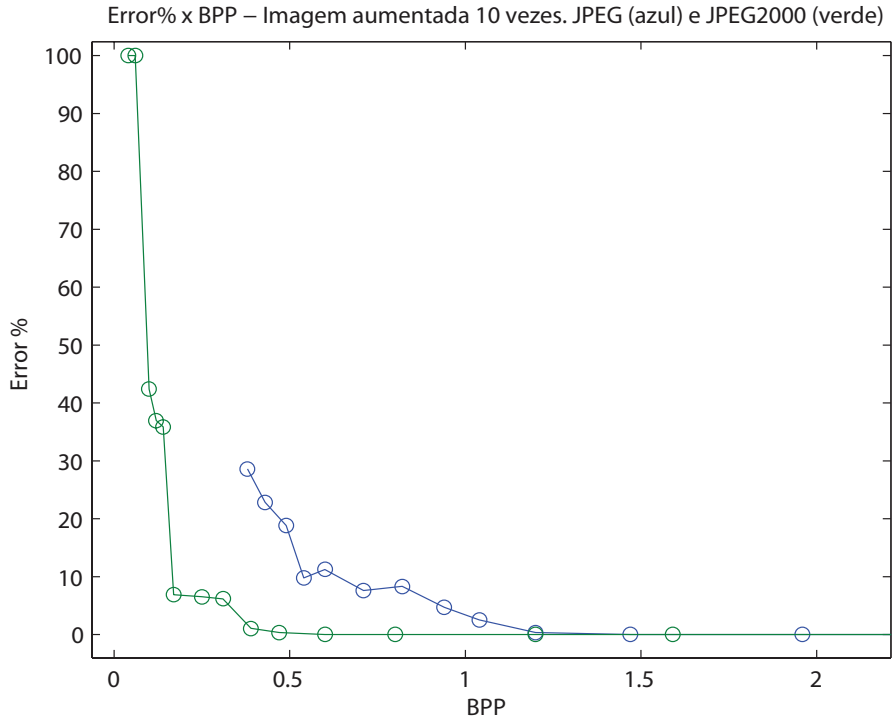

Figura 6.13: Percentagem de símbolos errados na decodificação x BPP para imagem de CQR Code aumentado 10 vezes nos padrões JPEG e JPEG2000.

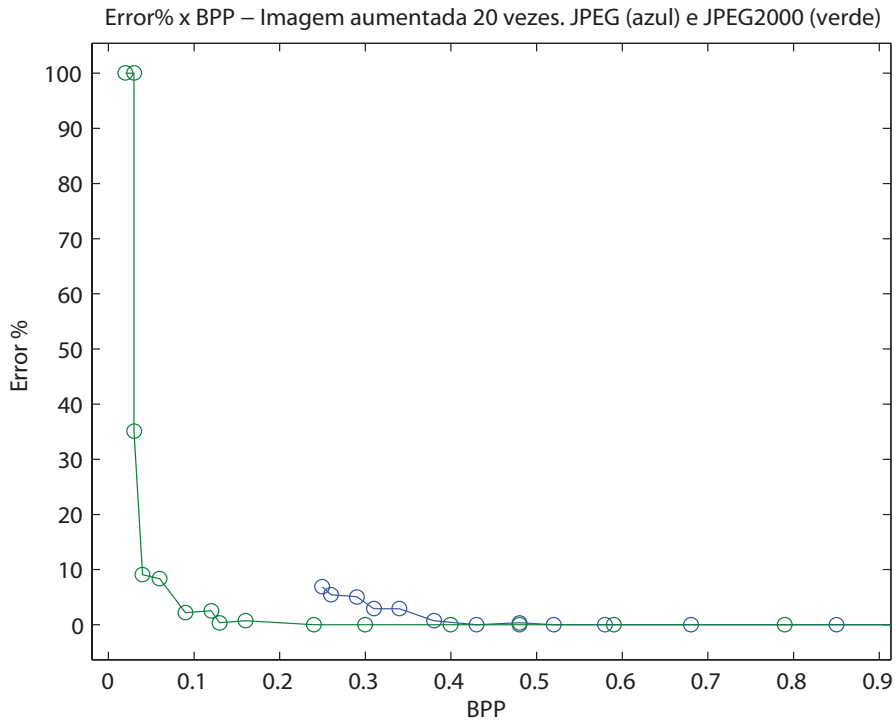

Figura 6.14: Percentagem de símbolos errados na decodificação x BPP para imagem de CQR Code aumentado 20 vezes nos padrões JPEG e JPEG2000.

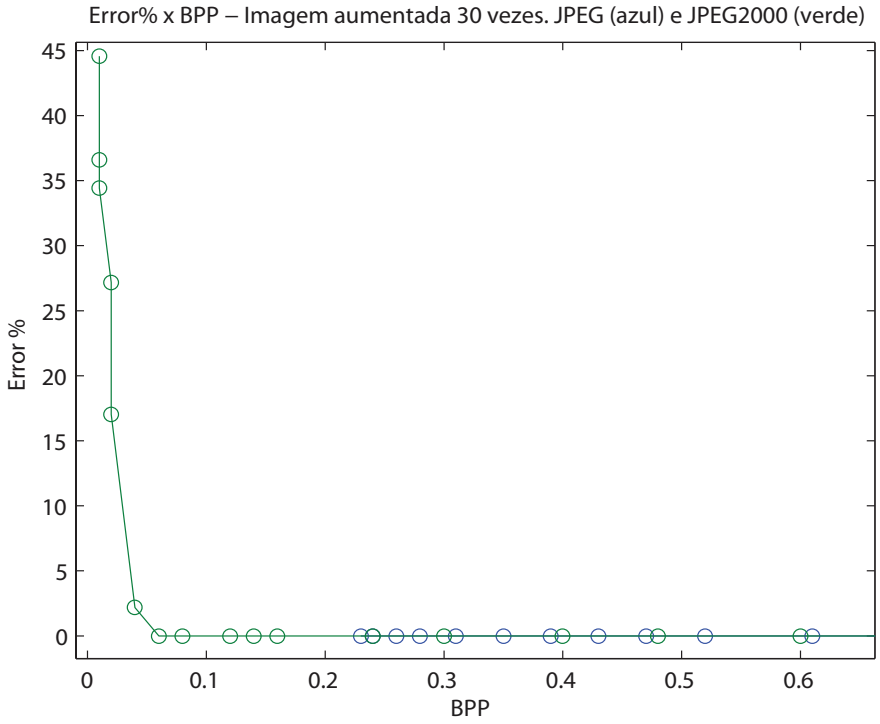

Figura 6.15: Percentagem de símbolos errados na decodificação x BPP para imagem de CQR Code aumentado 30 vezes nos padrões JPEG e JPEG2000.

7 ANÁLISE DOS RESULTADOS

O processo mais importante na decodificação para qualquer QR Code ou CQR Code que existe hoje em dia é a correta definição dos Módulos de Localização, sem a definição do dentro dos Módulos de Localização não seria viável a decodificação de nenhum QR Code ou CQR Code. Pode ser feito o teste em qualquer aplicativo de QR Code disponível que quando se obstrui pelo menos um Módulo de Localização o aplicativo não decodifica o QR Code. Nesse quesito ambos CQR Codes possuem um desempenho satisfatório conseguindo a definição dos Módulos de Localização em todas as imagens que não estejam corrompidas ou muito embaçadas. Para a decodificação do primeiro modelo de CQR Code foi utilizada a câmera de 3.2 *megapixels* do celular *Nokia 5800 Xpress Music* e para a decodificação do segundo modelo de CQR Code foi utilizada a câmera de 8 *megapixels* do celular *Samsung Galaxy S III*.

O QR Code também pode armazenar grande quantidade de bits, por exemplo, para armazenar 1024 bits de informação com capacidade de correção de erros H de aproximadamente 30% é necessário um QR Code do tamanho de 61 x 61 módulos que pode ser impresso em uma área não menor do que 3, 3cm x 3, 3cm (de acordo a testes realizados em diversas impressões, todo QR Code para ser corretamente decodificado deve ter pelo menos 4 *pixels* por módulo). Da mesma maneira o QR Code pode armazenar 2048 bits de informação com capacidade de correção de erros H de aproximadamente 30% é necessário um QR Code do tamanho de 85 x 85 módulos que pode ser impresso em uma área não menor do que 4, 6cm x 4, 6cm.

Com a utilização do primeiro modelo de CQR Code a área necessária para armazenar 1024 bits de informação com capacidade de correção de erros de 38, 4% é 2, 53 vezes menor do que no QR Code e a área necessária para armazenar 2048 bits de informação com capacidade de correção de erros de 34, 54% no segundo modelo de CQR Code é 3, 53 vezes menor do que no CQR Code, obtendo ganhos significativos de bits por área.

A diferença entre os QR Codes e os CQR Codes é que os QR Codes podem ser configurados em diversos tamanhos e capacidade de correção de erros, já os dois modelos de CQR Codes possuem tamanho estático com o único objetivo de atingir o transporte de 1024 bits e 2048 bits de informação em uma área de no mínimo 1, 3cm x 1, 3cm.

O QR Code, o primeiro modelo de CQR Code e o segundo modelo de CQR Code utilizam uma quantidade de cores na Área de Codificação estrategicamente escolhidas em uma constelação RGB conforme mostrado nas Figuras 7.1, 7.2 e 7.3 respectiva-

mente.

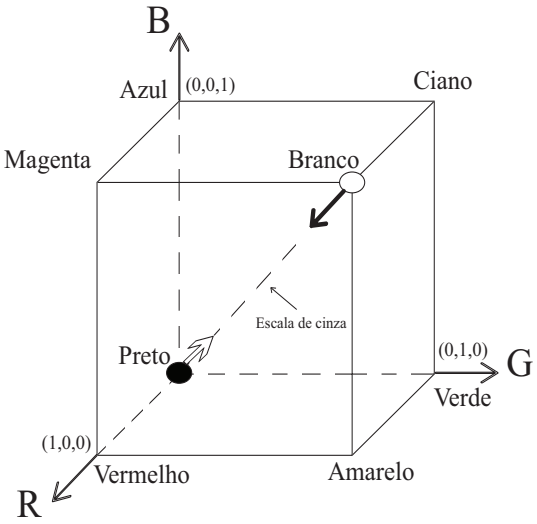

Figura 7.1: Constelação RGB do QR Code.

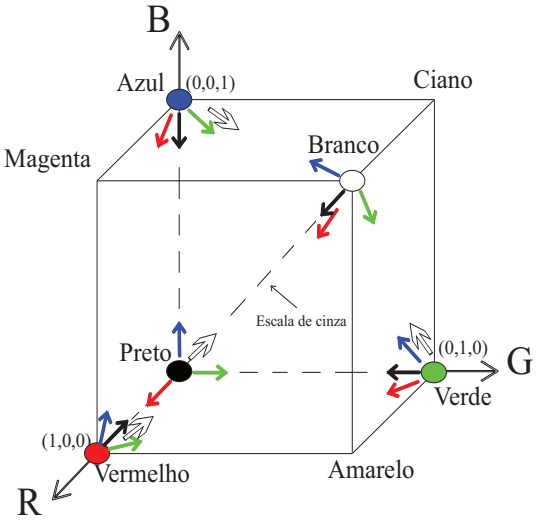

Figura 7.2: Constelação RGB do primeiro modelo de CQR Code.

Na Figura 7.1 é possível observar que as coordenadas das cores preto e branco estão o mais eqüidistante possível e eles também estão nos extremos da linha de escala de cinza, o que é favorável para o processamento binário dessas imagens, sem a necessidade da utilização dos vetores RGB, essa vantagem faz que os QR Codes possam processar as imagens em forma binária e que não precisem o custo computacional de processar as três dimensões do padrão RGB.

O primeiro modelo de CQR Codes também utiliza as cores mais eqüidistantes para as cinco cores escolhidas, facilitando a detecção das cores por meio de limiares de detecção baseado no maior componente RGB conforme as regras mostradas no item 5.1.4.1. Já o segundo modelo de CQR Code possui uma dificuldade grande na hora de detectar a cor cinza, dado que é intermediária entre a cor branca e preta, em algumas

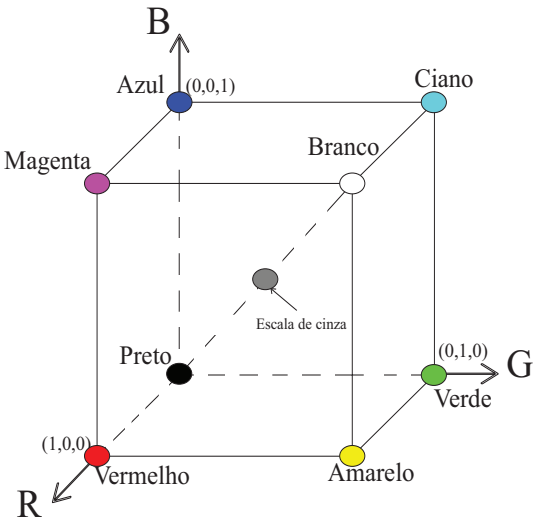

Figura 7.3: Constelação RGB do segundo modelo de CQR Code.

situações fica difícil de definir, principalmente quando o intervalo de intensidade entre a cor preta e branca é baixa, por exemplo, no lugar do preto ser (0,0,0) e o branco ser (255, 255, 255) , eles são (40, 40, 40) e (180, 180, 180) respectivamente em algumas imagens. São nessas situações que alguns *pixels* a cor cinza podem ser decodificados de forma errônea, mesmo com essa dificuldade o CQR Code atinge o objetivo de decodificar imagens pequenas de 1, 3cm x 1, 3cm, para imagens de maior comprimento o decodificador tem um desempenho melhor.

7.1 Apresentação dos Resultados - Primeiro Modelo de CQR Code

A seguir são mostradas as estatísticas de algumas decodificações de quatro diferentes CQR Codes do primeiro modelo, os quatro diferentes CQR Codes foram impressos em papel comum A4 de 75gr. na impressora *Xerox WorkCentre M24* e possuem o tamanho de 1, 3cm x 1, 3cm.

Na Tabela 7.1 os CQR Codes com identificador inicial 1 foram imagens tiradas às 17:00 (horário de verão) com luz artificial e uma janela pequena aberta (sem luz solar direta), os CQR Codes com identificador inicial 2 foram fotos tiradas às 21:00 (horário de verão) com luz artificial, os CQR Codes com identificador inicial 3 foram fotos tiradas às 14:00 (horário de verão) no sol direto com sombra pequena artificial e finalmente os CQR Codes com identificador inicial 4 foram fotos tiradas às 12:00 (horário de verão) com luz solar direta, mas com uma pequena sombra bem em cima do CQR Code.

O algoritmo de decodificação do primeiro CQR Code consegue decodificar qualquer imagem bem focada, ou seja, sem embaçamento nem inclinação pronunciada. A única

Tabela 7.1: Decodificação do primeiro modelo de CQR Code.

Identificador de CQR Code	**Símbolos corrompidos**	**Status**	**%Erro**
1a	16	Aprovado	5,80%
1b	3	Aprovado	1,09%
1c	0	Aprovado	0,00%
1d	2	Aprovado	0,72%
2a	—	Não Decodificou	—
2b	4	Aprovado	1,45%
2c	0	Aprovado	0,00%
2d	2	Aprovado	0,72%
3a	4	Aprovado	1,45%
3b	0	Aprovado	0,00%
3c	2	Aprovado	0,72%
3d	2	Aprovado	0,72%
4a	0	Aprovado	0,00%
4b	6	Aprovado	2,17%
4c	20	Aprovado	7,25%
4d	13	Aprovado	4,71%

imagem que não pode ser decodificada da tabela anterior foi a imagem de um segundo modelo de CQR Code muito embaçada, o que dificultou encontrar um dos Módulos de Localização.

A continuação é mostrado o processo de decodificação do primeiro CQR Code para cada número de CQR Code na Tabela 7.1 que finaliza com a letra a (1a, 2a, 3a e 4a). As imagens seguem o mesmo procedimento mostrado na seção 5.1.4.

O CQR Code com identificação 1a teve o processo de decodificação das Figuras 7.4, 7.5, 7.6 e 7.7, essa decodificação teve 16 símbolos corrompidos, os quais foram corrigidos com o algoritmo de correção de erros Reed-Solomon e dessa forma, a imagem foi corretamente decodificada.

Para o CQR Code com identificação 2a, o algoritmo de decodificação não pode encontrar um dos Módulos de Localização e assim não decodificou o CQR Code, é possível ver na Figura 7.8 que os Módulos de Localização estão borrosos.

O CQR Code com identificação 1a teve o processo de decodificação das Figuras 7.9, 7.10, 7.11 e 7.12, essa decodificação teve 4 símbolos corrompidos os quais foram corrigidos com o algoritmo de correção de erros Reed-Solomon e dessa forma, a imagem foi corretamente decodificada.

O processo de decodificação das Figuras 7.13, 7.14, 7.15 e 7.16 teve 0 símbolos corrompidos, o algoritmo de correção de erros Reed-Solomon foi aplicado e decodificou sem relatar nenhum símbolo errôneo.

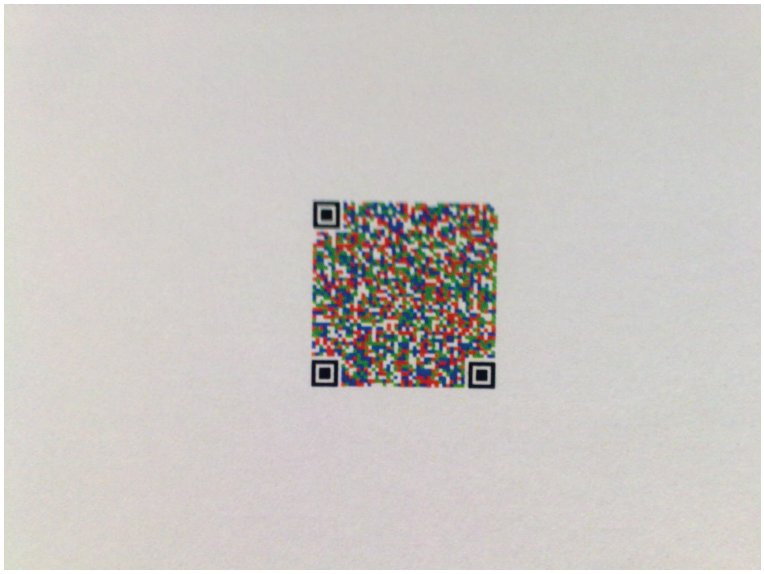

Figura 7.4: Processo de decodificação - Primeira imagem de 1a na Tabela 7.1.

Figura 7.5: Processo de decodificação - Imagem recortada de 1a na Tabela 7.1.

Figura 7.6: Processo de decodificação - Imagem rodada de 1a na Tabela 7.1.

Figura 7.7: Processo de decodificação - Imagem limiarizada nas cores primárias de 1a na Tabela 7.1.

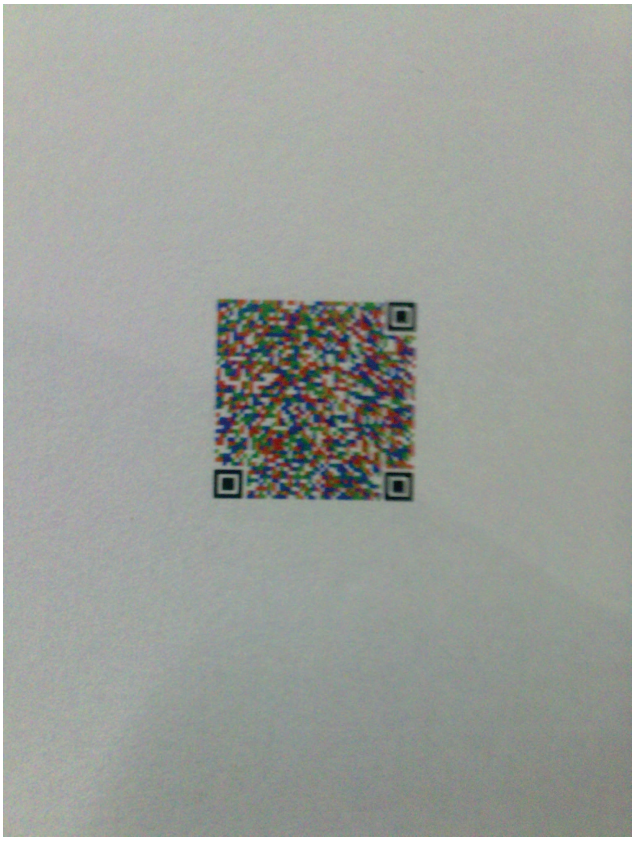

Figura 7.8: Processo de decodificação - Primeira imagem de 2a na Tabela 7.1.

7.2 Apresentação dos Resultados - Segundo Modelo de CQR Code

A seguir são mostradas as estatísticas das decodificações de três diferentes CQR Codes do segundo modelo, os três diferentes CQR Codes foram impressos em papel comum A4 de 75gr. na impressora *Xerox WorkCentre M24*. Dos três CQR Codes,

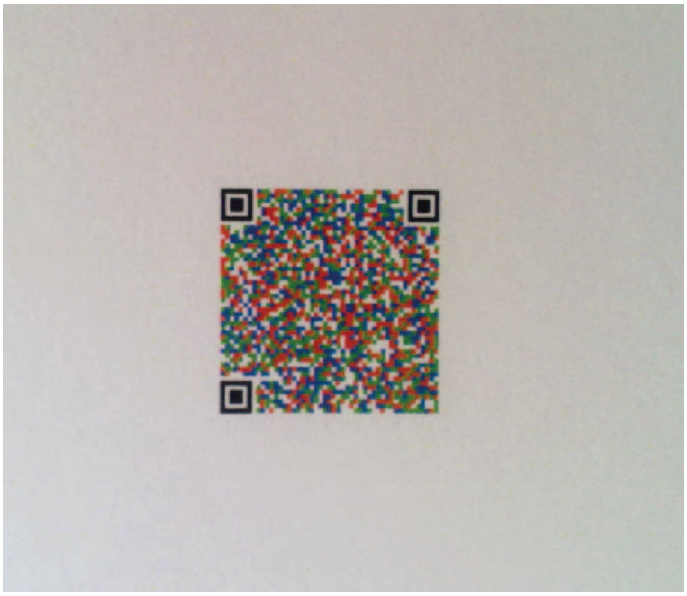

Figura 7.9: Processo de decodificação - Primeira imagem de 3a na Tabela 7.1.

Figura 7.10: Processo de decodificação - Imagem recortada de 3a na Tabela 7.1.

Figura 7.11: Processo de decodificação - Imagem rodada de 3a na Tabela 7.1.

os CQR Codes com identificação 1 e 2 são decodificações de impressões de tamanho $1,3$cm x $1,3$cm e o CQR Code com identificação 3 é decodificação de impressão de

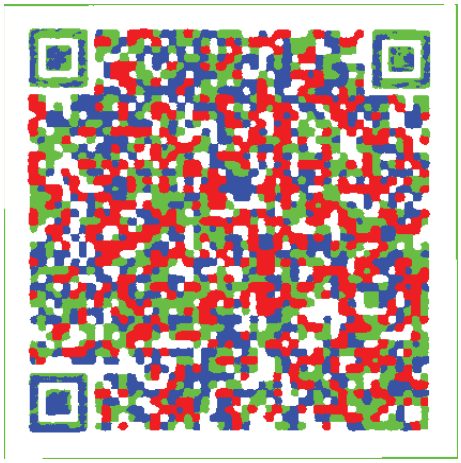

Figura 7.12: Processo de decodificação - Imagem limiarizada nas cores primárias de 3a na Tabela 7.1.

Figura 7.13: Processo de decodificação - Primeira imagem de 4a na Tabela 7.1.

Figura 7.14: Processo de decodificação - Imagem recortada de 4a na Tabela 7.1.

Figura 7.15: Processo de decodificação - Imagem rodada de 4a na Tabela 7.1.

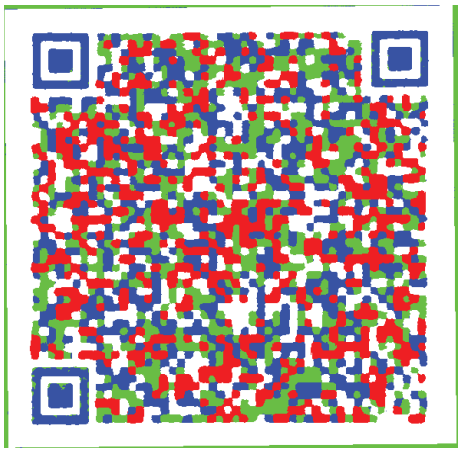

Figura 7.16: Processo de decodificação - Imagem limiarizada nas cores primárias de 4a na Tabela 7.1.

tamanho 2, 6cm x 2, 6cm (quatro *pixels* por módulo). Os resultados são mostrados na Tabela 7.2.

As três imagens processadas foram imagens capturadas em um ambiente fechado com luz artificial.

O algoritmo de decodificação do segundo CQR Code consegue identificar os Módulos de Localização de qualquer imagem bem focada mas ainda não é tão eficiente no quesito de identificar os módulos da Área de Codificação por possuir uma constelação maior de cores e também porque a impressão das nove cores em uma área tão pequena não fica com as cores o suficientemente definidas, uma possível solução é encontrar uma impressora com maior qualidade.

As imagens do processo de decodificação do segundo CQR Code seguem o mesmo procedimento mostrado na seção 5.2.4.

O processo de decodificação das Figuras 7.17, 7.18, 7.19 e 7.20 do CQR Code com identificador 1 teve 103 símbolos corrompidos os quais foram corrigidos com o algoritmo de correção de erros Reed-Solomon e dessa forma, a imagem foi corretamente

decodificada.

O processo de decodificação das Figuras 7.21, 7.22, 7.23 e 7.24 do CQR Code com identificador 2 teve 133 símbolos corrompidos os quais foram corrigidos com o algoritmo de correção de erros Reed-Solomon e dessa forma, a imagem foi corretamente decodificada.

O segundo modelo de CQR Code está funcionando tanto na codificação como na decodificação, os limiares de detecção entre as cores primárias, secundárias, preto, branco e cinza permitem que a decodificação seja viável no primeiro código matricial com nove cores inventado.

O processo de decodificação das Figuras 7.25, 7.26, 7.27 e 7.28 do CQR Code com identificador 3 teve 13 símbolos corrompidos os quais foram corrigidos com o algoritmo de correção de erros Reed-Solomon e dessa forma, a imagem foi corretamente decodificada.

Tabela 7.2: Decodificação do segundo modelo de CQR Code.

Identificador de CQR Code	**Símbolos corrompidos**	**Status**	**%Erro**
1	103	Aprovado	$24,88\%$
2	133	Reprovado	$32,13\%$
3	13	Aprovado	$3,14\%$

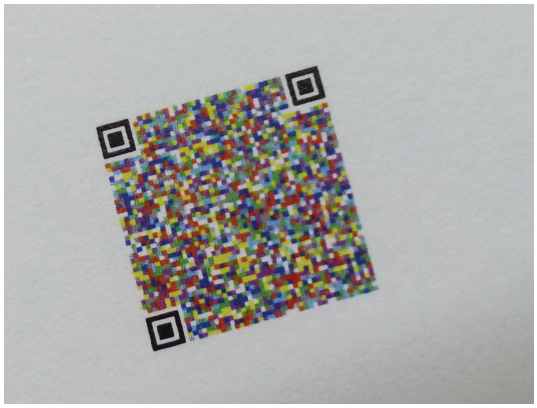

Figura 7.17: Processo de decodificação - Primeira imagem de 1 na Tabela 7.2.

7.3 Análise de Área de Codificação para CQR Codes de 5 Cores

Nessa subseção é feito um breve estudo que pode servir como base para futuras pesquisas de CQR Codes com cinco e nove cores com tamanhos variáveis, ou seja, CQR Codes baseados em diferentes tamanhos de módulos.

Com base na Figura 1.1 do QR Code é possível ver que os QR Codes, da mesma maneira do que os CQR Codes, são códigos matriciais com dimensão de N x N módulos, no caso dos CQR Codes apresentados nessa dissertação temos 49 x 49 módulos. É

Figura 7.18: Processo de decodificação - Imagem recortada de 1 na Tabela 7.2.

Figura 7.19: Processo de decodificação - Imagem rodada de 1 na Tabela 7.2.

Figura 7.20: Processo de decodificação - Imagem limiarizada nas cores primárias de 1 na Tabela 7.2.

possível esboçar um cenário onde as dimensões do CQR Code sejam variáveis em função da quantidade de bits de informação que o usuário final deseje colocar e garantindo uma percentagem de recuperação de dados distorcidos de no mínimo 30%.

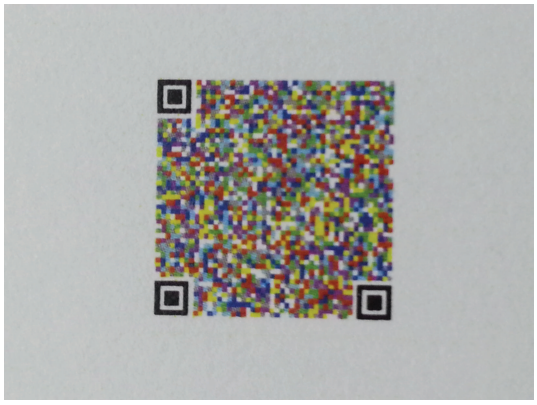

Figura 7.21: Processo de decodificação - Primeira imagem de 2 na Tabela 7.2.

Figura 7.22: Processo de decodificação - Imagem recortada de 2 na Tabela 7.2.

Figura 7.23: Processo de decodificação - Imagem rodada de 2 na Tabela 7.2.

Sendo N a quantidade de módulos por cada lado que o CQR Code possui e garantindo que no máximo 30% dos dados nos CQR Codes poderão ser corrompidos, podemos obter a quantidade de bits disponíveis na Área de Codificação mostrada na Equação 7.1.

Figura 7.24: Processo de decodificação - Imagem limiarizada nas cores primárias de 2 na Tabela 7.2.

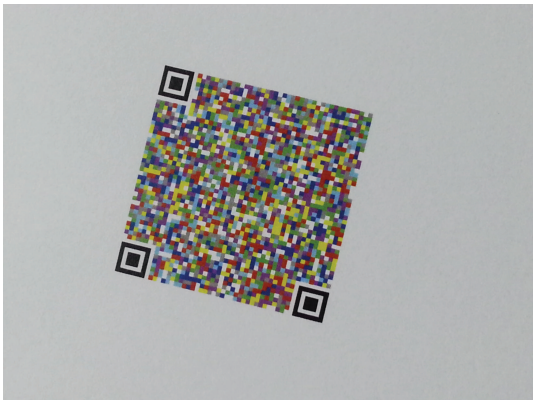

Figura 7.25: Processo de decodificação - Primeira imagem de 3 na Tabela 7.2.

Figura 7.26: Processo de decodificação - Imagem recortada de 3 na Tabela 7.2.

$$B_d = (N \times N - 192)2. \tag{7.1}$$

Em função dos bits disponíveis no CQR Code, podemos estimar a quantidade de

Figura 7.27: Processo de decodificação - Imagem rodada de 3 na Tabela 7.2.

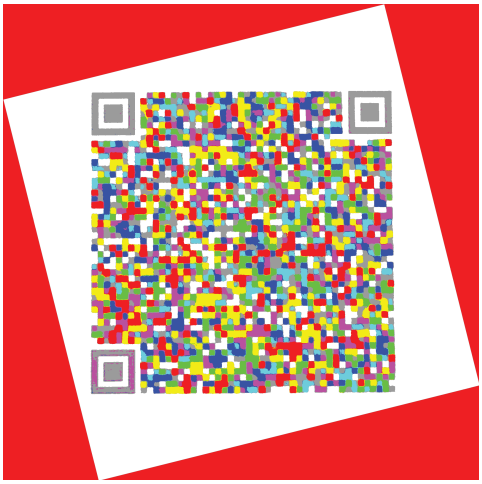

Figura 7.28: Processo de decodificação - Imagem limiarizada nas cores primárias de 3 na Tabela 7.2.

símbolos (cada um com 16 bits) que o CQR Code pode armazenar, conforme mostrado na Equação 7.2.

$$S_d = floor(B_d/16). \tag{7.2}$$

Onde *floor* representa a escolha do menor número inteiro dentro dessa função. A percentagem de correção de erro no algoritmo Reed-Solomon está mostrado na Equação 7.3.

$$C_e = \left(\frac{t}{n}\right) 100. \tag{7.3}$$

Fixando nossa percentagem de correção de erro em no mínimo 30%, temos o valor

de t em função de n como mostrado na Equação 7.4.

$$t = 0,3n. \tag{7.4}$$

Aplicando essa relação na Equação 3.1, temos:

$$k = 0,4n. \tag{7.5}$$

Dessa maneira, fixando o valor do tamanho do CQR Code de N x N, é possível determinar quantos bits de informação, em função dos símbolos de informação k, o CQR Code vai poder armazenar com uma percentagem de correção mínima de 30%.

A Tabela 7.3 mostra alguns valores obtidos para a geração de CQR Codes com diferentes tamanhos.

Tabela 7.3: Simulações de diversos tamanhos de CQR Codes com cinco cores.

CQR Code NxN	n	k	t	**% Correção**	**Bits de Informaçao**
41x41	186	74	56	$30,1075\%$	1184
45x45	229	91	69	$30,1310\%$	1456
49x49	276	110	83	$30,0724\%$	1760
53x53	327	129	99	$30,2752\%$	2064
59x59	382	152	115	$30,1047\%$	2432
61x61	441	175	133	$30,1587\%$	2800

7.4 Análise de Área de Codificação para CQR Codes de 9 Cores

Em função das mesmas equações da seção anterior, somente com a diferência na Equação 7.1 mostrada na Equação 7.6 é possível gerar o mesmo cenário de simulação para o CQR Code com 9 cores conforme apresentado na Tabela 7.4.

$$B_d = (N \times N - 192)3. \tag{7.6}$$

Como mostrado na Tabela 7.4, os CQR Codes com nove cores podem ter uma capacidade de armazenamento muito grande, atinjindo fácilmente mais de 4Kb em uma área aproximada de $1,62$cm x $1,62$cm de impressão. O desenvolvimento e melhoria dessa tecnologia deve ser abordado como trabalho futuro.

Tabela 7.4: Simulações de diversos tamanhos de CQR Codes com nove cores.

CQR Code NxN	n	k	t	**% Correção**	**Bits de Informaçao**
41x41	279	111	84	$30,1075\%$	1776
45x45	343	137	103	$30,0291\%$	2192
49x49	414	164	125	$30,1932\%$	2624
53x53	490	196	147	30%	3136
59x59	573	229	172	$30,0174\%$	3664
61x61	708	282	213	$30,0847\%$	4512

8 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Nesse Capítulo serão mostradas as informações mais importantes da dissertação e serão explicados os trabalhos futuros que poderão ser realizados em próximos projetos de pesquisa.

8.1 Considerações Finais

Atualmente o maior uso de QR Codes está na codificação e decodificação de URLs [24], utilizando como entrada e saída dados alfanuméricos ou ASCII, a proposta dos CQR Codes é ser um aplicativo que utiliza dados de entrada e saída em forma de bits, resultando em una possível ferramenta de armazenamento e transporte de assinaturas digitais para rastreamento de produtos, criptografia simétrica e assimétrica. Os CQR Codes foram implementados com o objetivo de armazenar e transmitir a maior quantidade de bits de informação com um alto nível de correção de erro (mínimo 30%) na menor área de impressão utilizada.

Essa dissertação mostra a implementação da geração e leitura de dois modelos de CQR Codes. As especificações técnicas do primeiro modelo de CQR Code estão mostradas na seção 5.1 e de acordo ao mostrado nessa seção é concluído que o primeiro modelo de CQR Code é o suficientemente robusto como para começar a ser utilizado em aplicações reais.

Uma justificativa da robustez do primeiro modelo dos CQR Codes é a pesquisa mostrada no capítulo 6, onde é mostrado o comportamento da decodificação do primeiro modelo de CQR Code em cima de imagens com alta taxa de compressão nos formatos JPEG e JPEG2000, outro parâmetro que justifica a robustez do primeiro modelo de CQR Code é o ajuste aplicado em situações de imagens um pouco tortas em relação ao ângulo da captura da imagem o ajuste conhecido também como Transformada Afim foi testado apenas para imagens muito pouco tortas (questão de dezenas de *pixels*) e não em imagens muito ou completamente tortas o que pode ser desconsiderado no caso que o usuário final do *software* capture a imagem do CQR Code da melhor maneira possível.

O segundo modelo dos CQR Codes mostrado nessa dissertação representa um avanço significativo no aumento de capacidade de armazenamento e transmissão de informações, a principal dificuldade do segundo modelo dos CQR Codes está no esta-

belecimento de limiares de detecção em função da constelação mostrada na Figura 7.3. Os pontos da constelação estão muito mais próximos no segundo modelo de CQR Code do que no primeiro modelo de CQR Code. Essa dificuldade está presente na quantidade de símbolos errôneos na decodificação do segundo modelo de CQR Code. O segundo modelo de CQR Code ainda está em fase inicial e a pesquisa deve ser continuada para aperfeiçoar ou encontrar o limite de eficiência do *software* decodificador.

8.2 Trabalhos futuros

Os resultados apresentados poderiam ser muito melhores com o uso de melhores câmeras disponíveis no mercado. Como trabalho futuro deve ser verificado o comportamento da decodificação de ambos modelos de CQR Code com a utilização de diversos modelos de câmeras de melhor oi pior resolução das utilizadas atualmente.

Outra característica do projeto é a associação da utilização de CQR Codes com criptografia. Os QR Codes hoje existentes não permitem o seu uso associado a protocolos de criptografia (como assinaturas digitais) em áreas pequenas dadas as limitações na capacidade de armazenamento. Para trabalhos futuros fica como pendência o armazenamento de assinaturas digitais de tamanho considerável. O uso de criptografia possibilitará uma nova gama de aplicações para códigos bidimensionais, por exemplo, o rastreamento seguro de remédios, cigarros e até armas. A seguir são listadas as variáveis de modificações técnicas na geração e leitura dos CQR Codes como trabalho futuro:

- **Múltiples versões dos CQR Codes**: é necessária a implementação dos CQR Codes em diferentes versões ou tamanhos de N x N módulos (da mesma maneira dos QR Codes), o fundamento teórico foi apresentado no Capítulo 7 para ambos modelos dos CQR Codes e precisa ser aperfeiçoado.

- **Utilização de diversas câmeras**: o *software* de decodificação deve ser homologado com a maior quantidade de câmeras disponíveis no mercado, tanto de celulares como de câmeras tradicionais.

- **Código corretor de erro**: deve ser avaliada a utilização de outros códigos corretores de erro mais eficientes e livres como o LDPC entre outros.

- **Criptografia**: deve ser testado também a utilização de diversos protocolos de criptografia como entrada dos bits de informação dos CQR Codes.

- **Exibição do CQR Code**: a leitura dos CQR Code deve ser testada em diferentes plataformas de exibição dos CQR Codes como papel de diversas gramaturas, plástico, pintura em parede e inclusive televisões ou monitores em geral.

O primeiro item da lista anterior pode determinar um novo padrão de QR Code que poderia ser utilizado no dia a dia por qualquer pessoa com um *smartphone*.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] "ISO/IEC 15420:20005 -Information technology – Automatic identification and data capture techniques – Bar code symbology specification," Dezembro 2009.

[2] "ISO/IEC 18004:2005 - Information technology - Automatic identification and data capture techniques - QR Code 2005 bar code symbology specification," Agosto 2005.

[3] AIPSYS SOFTWARE LABORATORY. QR Code Introduction. http://www.aipsys.com/qrcodeintro.htm, último acesso em 15/02/2012.

[4] ROUILLARD, J., "Contextual QR Codes," in *Proc. of the Computing in the Global Information Technology, 2008. ICCGI '08. The Third International Multi-Conference on Communication, Networking and Broadcasting*, Agosto 2008, pp. 50 – 55.

[5] H. KATO; K. T. TAN, "2D Barcodes for mobile phones," in *Proc. of the 2nd Int. Conf. on Mobile Tech., Applications and Systems*, Novembro 2005.

[6] GOQR.ME. QR Code Generator. http://goqr.me/, último acesso em 15/02/2012.

[7] MELGAR M. E. V.; ZAGHETTO A.; MACCHIAVELLO B.; NASCIMENTO A. C. A., "CQR Codes: Colored Quick-Response Codes," in *Proc. of the 2nd IEEE International Conference on Consumer Electronics - Berlin (IEEE 2012 ICCE-Berlin)*, Setembro 2012, pp. 321 – 325.

[8] BARRY, A. CIPRA, "The Ubiquitous Reed-Solomon Codes," in *Society for Industrial and Applied Mathematics*, Janeiro 1993.

[9] RAFAEL C. GONZALES; RICHARD E. WOODS, *Digital Image Processing Second Edition*, 2nd ed. Prentice Hall, 2002.

[10] SAYOOD, K., *Introduction to Data Compression*, 3rd ed. EUA: Morgan Kaufmanl, 2006.

[11] POULARIKAS, A. D., *The Transform and Applications Handbook*, 2nd ed. EUA: IEEE Press, 2000.

[12] WU, H. R.; RAO, K. R., *Digital Video Image Quality and Perceptual Coding*, 3rd ed. EUA: Willey, 2001.

[13] ZAGHETTO, A., *Compressão de Documentos Compostos Utilizando o H.264/AVC-Intra*. Departamento de Engenharia Elétrica, Universidade de Brasília, 2009.

[14] JPEG. Information Technology - Digital Compression and Coding of Continuous-tone Still Images - Requirements and Guidelines. ITU-T Recommendation T.81., Setembro 1992.

[15] PENNEBAKER, W. B.; MITCHELL, J. L., *JPEG Still Image Data Compression Standard*. Chapman and Hall, 1993.

[16] RAO, K. R.; YIP, P., *Discrete Cosine Transform - Algorithms, Advantages, Applications*. EUA: Academic Press, 1990.

[17] YU, J., "Advantages of Uniform Scalar Dead-zone Quantization in Image Coding System," in *Proceedings of IEEE International Conference on Communications, Circuits and Systems*, vol. 2, Junho 2004, pp. 805 – 808.

[18] IJG - Independent JPEG Group. http://www.ijg.org/, último acesso em 15/02/2012.

[19] GALVÃO de O. R., *Avaliação do Desempenho de Transformadas Sobrepostas e Wavelets nos Codificadores Padrão JPEG2000 e H.264/AVC*. Departamento de Engenharia Elétrica, Universidade de Brasília, 2008.

[20] TODD K. MOON, *Error Correction Coding Mathematical Methods and Algorithms*. EUA: John Wiley and Sons, 2005.

[21] REED, I. S.; SOLOMON, G., "Polynomial Codes Over Certain Finite Fields," in *SIAM Journal of Applied Math*, vol. 8, 1960, pp. 300–304.

[22] MATLAB, "Image Processing Toolbox."

[23] MATLAB , "Communication Toolbox."

[24] YU-HSUAN CHANG; CHUNG-HUA CHu; MING-SYAN CHEN, "A General Scheme for Extracting QR Code from a Non-uniform Background in Camera Phones and Applications," in *IEEE International Symposium on Multimedia*, 2007.

APÊNDICES

A ESTRUTURAS DOS PRINCIPAIS QR CODES

Esse Apêndice mostra as figuras das estruturas dos QR Codes com versões 1, 2, 6, 7, 14, 21 e 40.

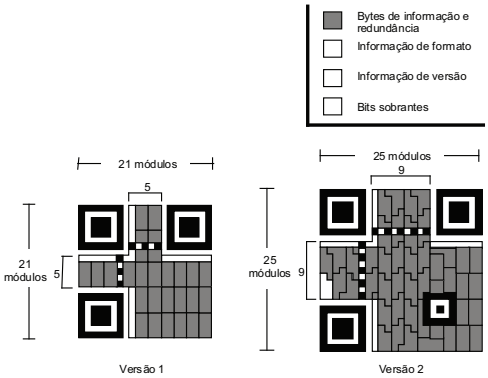

Figura A.1: Versões 1 e 2 do QR Code [2].

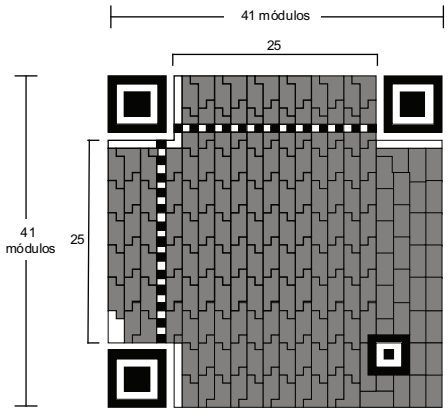

Figura A.2: Versão 6 do QR Code [2].

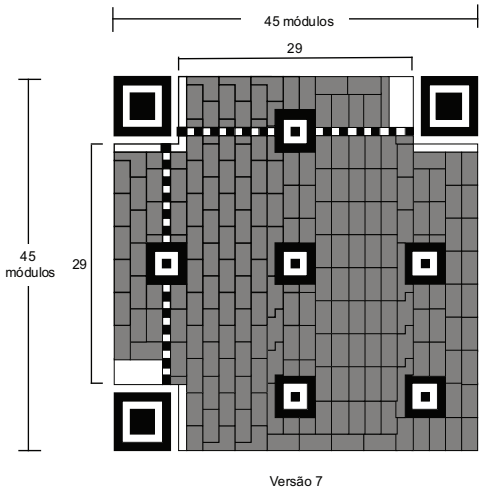

Figura A.3: Versão 7 do QR Code [2].

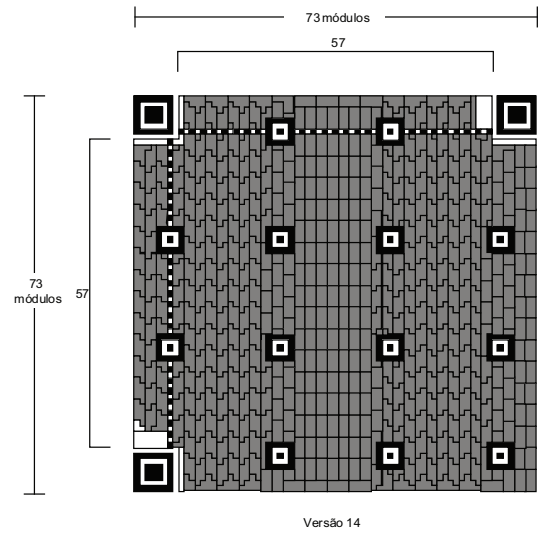

Figura A.4: Versão 14 do QR Code [2].

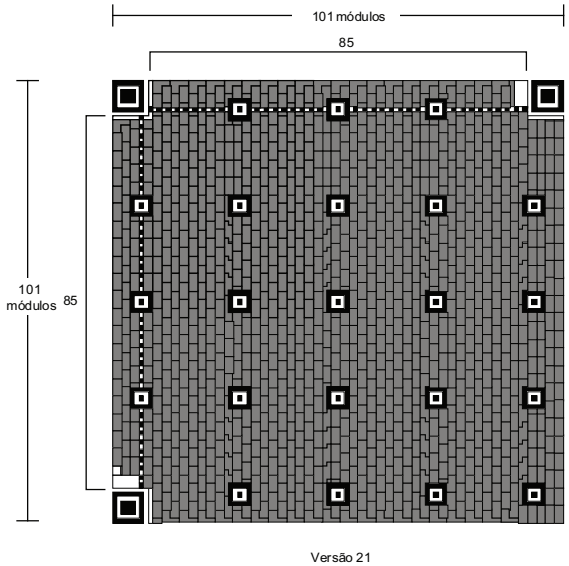

Figura A.5: Versão 21 do QR Code [2].

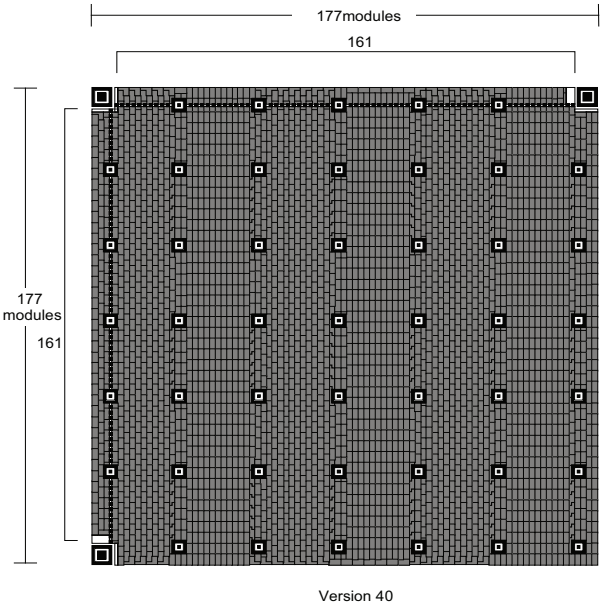

Figura A.6: Versão 40 do QR Code [2].