

DESENVOLVIMENTO DE CÓDIGO DE BARRAS BIDIMENSIONAL CUSTOMIZÁVEL

Cleyson Gustavo Reinhold, Aurélio Faustino Hoppe – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

cgreinhold@furb.br, aureliof@furb.br

Resumo: *O presente artigo discorre sobre conceitos para codificação de uma informação em texto para uma forma visual, tendo sido desenvolvido uma aplicação web que permite os usuários criarem e customizarem um novo tipo de código de barras bidimensional a partir de textos com codificação UTF-8 como uma alternativa ao QR Code. Além da aplicação web, desenvolveu-se um aplicativo para dispositivos Android capaz de decodificar essas informações fazendo uso de técnicas de processamento de imagens para decodificação usando operadores morfológicos como de erosão e dilatação, além da detecção de contornos para localização dos símbolos que representam o conteúdo a ser interpretado. A solução proposta conseguiu, de forma geral, elaborar uma alternativa viável aos códigos de barras atuais, porém apresenta limitações técnicas, principalmente em sua decodificação, como a falta de um mecanismo de correção que em alguns cenários gera erros na leitura.*

Palavras-chave: QR Code. Código de barras. Codificação. Decodificação.

1 INTRODUÇÃO

Com uma incrível capacidade de invenção e com a evolução tecnológica ao qual se chegou, Harari (2015) afirma que o ser humano foi capaz de criar indústrias e marcas conhecidas globalmente. Diante disso, segundo Holt (2006), estabeleceu-se uma busca constante pela otimização de espaços em conteúdos de mídia e não somente isto, mas também pela consolidação de marcas em âmbito mundial através de uma divulgação cada vez mais chamativa e atraente ao público alvo.

Com a evolução tecnológica, o próprio mercado econômico necessitava de uma forma de automatizar a catalogação de produtos, integração de produtos físicos com o mundo digital e o contato das empresas com sua audiência (DENSO WAVE INCORPORATED, 2003). A partir deste contexto, diversas tecnologias foram desenvolvidas para este fim, tais como o código de barras, tecnologia utilizada principalmente para catalogação de produtos e o QR Code, um código de barras bidimensional mais robusto que o código de barras padrão, que passou a ser utilizado de forma mais ampliada e difundida em diferentes meios tecnológicos (DENSO WAVE INCORPORATED, 2003).

O código de barras é um mecanismo utilizado a décadas em lojas e indústrias para a separação de produtos. Esse código é uma imagem que representa valores numéricos através de barras verticais que poderiam ser decodificadas através de um leitor digital. Esta foi uma forma popular de armazenar dados em um papel que pudessem ser lidos por uma máquina. Contudo, o código de barras possui limites quanto a quantidade de informação e tipos de dados que consegue armazenar e por conta disso, indústrias passaram a estudar novas formas de representação de códigos em imagens (MOTAHARI; ADJOUADI, 2015). Já o QR Code é uma dessas tecnologia que permite codificar uma mensagem em imagens. Criado pela empresa Denso Wave no ano de 1994, ele é uma evolução dos códigos de barras tradicionais, com a possibilidade de armazenar uma capacidade maior de informação em uma matriz bidimensional (ARYACHANDRAN; JYOTHI, 2014). Ele foi criado com intuito de facilitar a catalogação de produtos, mas rapidamente conquistou popularidade após a indústria de marketing começar a utilizá-lo em campanhas publicitárias, sendo uma forma mais eficaz de apresentar websites ao seu público (DENSO WAVE INCORPORATED, 2003).

A partir disso, Lin et al. (2015) afirmam que atualmente vários profissionais e pesquisadores estão tentando realizar modificações na estrutura de representação do QR Code para melhorá-lo no quesito estético. Contudo, esbarram nas limitações impostas pela forma que ele é gerado, impossibilitando-o de ser utilizado como mecanismo de divulgação de uma marca. Os autores ainda citam que os principais problemas estão relacionados a perda dos elementos de identificação e a qualidade da imagem final.

Diante do exposto, este artigo apresenta o desenvolvimento de uma nova forma de codificação de mensagens através de um código de barras bidimensional customizável, assim como uma forma de decodificação dessas mensagens através de técnicas de processamento de imagens. Os objetivos específicos da ferramenta desenvolvida são: (i) disponibilizar um mecanismo capaz de gerar uma representação bidimensional interpretável a partir de uma sequência de caracteres contendo formas geométricas; (ii) disponibilizar um mecanismo ao qual o usuário possa customizar o código

de barras utilizando diferentes cores, formato, símbolos e tamanho; (iii) desenvolver um leitor capaz de decodificar o código gerado, detectando os padrões de reconhecimento e símbolos através de técnicas de processamento de imagens.

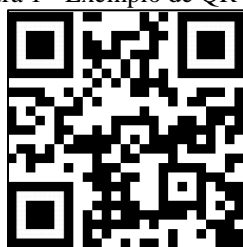
2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo expõe conceitos do QR Code, no qual é descrito sua codificação, decodificação e funcionamento. Além disso, também são expostas algumas técnicas de processamento de imagens utilizadas na detecção e interpretação dos símbolos. Por fim, são apresentados os trabalhos correlatos.

2.1 QR CODE

De acordo com Aryachandran e Jyothi (2014), o QR Code é um código de barras bidimensional que permite armazenar mais informações do que os códigos de barras tradicionais. Ele foi desenvolvido pela empresa japonesa Denso Wave em 1994, que disponibilizou o algoritmo publicamente, tornando-o mundialmente conhecido. O QR Code se apresenta como um quadrado de pixels brancos e pretos que representam uma informação binária. Alguns dos pixels são utilizados como padrões de reconhecimento, e por isso se repetem em todos os códigos gerados. A Figura 1 apresenta um exemplo de QR Code.

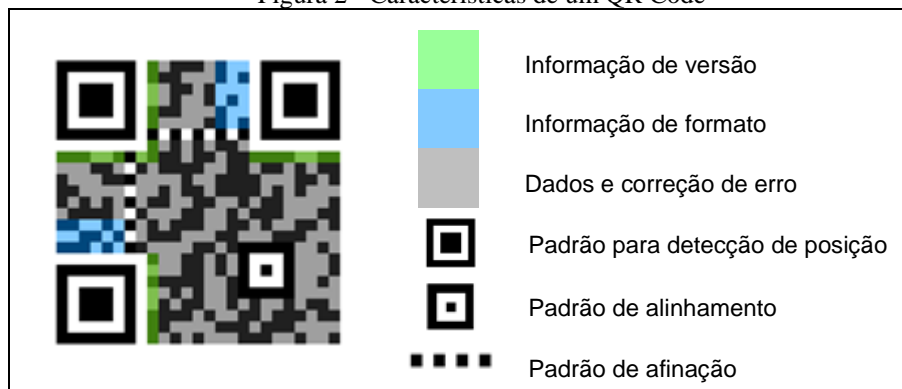
Figura 1 - Exemplo de QR Code



Fonte: elaborado pelo autor.

Os padrões de reconhecimento de um QR Code são, segundo Qianyu (2014), utilizados como forma de definir a orientação e as bordas da imagem. Em um QR Code são adicionados padrões para detecção de posição, representados por quadrados pretos com um traço retangular branco interno, em três vértices do quadrado. Dessa forma, o único vértice do quadrado que não possui esse padrão fica sempre no canto inferior direito. Com três vértices definidos é possível encontrar a altura e largura do QR Code, assim como a orientação. Com base nisso é possível detectar qual a rotação da imagem. Segundo Qianyu (2014) também são adicionados padrões de alinhamento ao longo da imagem. Esses padrões são pequenos quadrados que servem para que o reconhecedor possa alinhar a imagem caso ela tenha sido obtida de um ângulo em que a imagem final não represente um quadrado (QIANYU, 2014). Na Figura 2 são exemplificadas as principais características e os padrões do QR Code.

Figura 2 - Características de um QR Code



Fonte: adaptado de Aryachandran e Jyothi (2014).

Qianyu (2014) ainda afirma que existem diversas versões do QR Code com tamanhos diferentes, que permitem armazenar mais dados. Além do padrão, existem o Micro QR Code, com um tamanho menor, IQR Code, que permite a geração de códigos de barras retangulares, SQRC, que possui mecanismos para que sejam decodificados por leitores específicos e o Logo Q, que permite adicionar uma logo ao QR Code alterando as cores de seus pixels.

Aryachandran e Jyothi (2014) descrevem que essa forma de codificação pode ser facilmente decodificada pois possui informações de forma vertical e horizontal, podendo ser lido por scanners independentemente de sua rotação. Outras vantagens do QR Code descritas pelo autor são a possibilidade de ser representado em tamanhos pequenos, a capacidade de ser lido rapidamente e a possibilidade de se codificar caracteres japoneses e chineses, este último sendo um dos grandes fatores para o crescimento da sua popularidade.

Lin et al. (2015) apresentam outro mecanismo utilizado pelo QR Code, a redundância de informação. Esse mecanismo é utilizado para corrigir erros de leitura dos códigos de barras, ou seja, para que seja possível decodificar imagens mesmo em situações em que há danos na imagem. Além de garantir que a decodificação de QR Codes seja mais eficiente. Esse mecanismo também ajudará nos casos em que a imagem sofre alterações de iluminação e sombras ou de cortes e erros de impressão que danificam o código.

É muito comum que QR Codes sejam posicionados em lugares com texturas irregulares e com variação de iluminação constante, tornando a decodificação desafiadora (TRIBAK; ZAZ, 2017). Por conta disso, Tribak e Zaz (2017) ressaltam que os mecanismos de correção de erros são de extrema importância já que os algoritmos são executados em grande maioria em dispositivos móveis ou sistemas embarcados, que são caracterizados por ter uma capacidade computacional limitada. Outro fator importante é o fato de a informação ter que ser decodificada em tempo real, não sendo possível utilizando algoritmos com altos custos computacionais (TRIBAK; ZAZ, 2017).

2.2 PROCESSAMENTO DE IMAGENS PARA RECONHECIMENTO DE SÍMBOLOS

Segundo Gharde, Nemade e Adhiya (2013), símbolos são representações de informações, que por sua vez, quando agrupadas formam uma mensagem. Eles também afirmam que técnicas para reconhecimento digital de símbolos são utilizadas com diferentes fins, como por exemplo digitalização de circuitos elétricos, diagramas ou fórmulas matemáticas. Os autores ainda listam alguns algoritmos que são comumente utilizados para isso, como o modelo de Markov, redes neurais, máquina de vetores de suporte e classificadores de árvores de decisão.

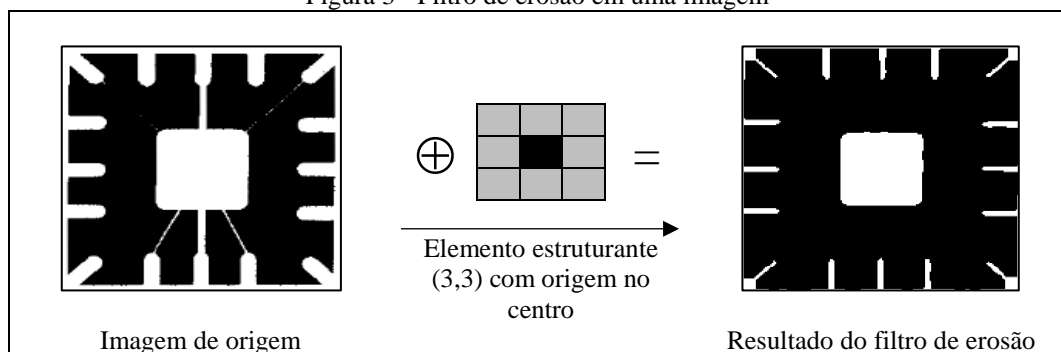
Llados e Sánchez (2003) apontam que duas questões são importantes em relação ao reconhecimento de um símbolo: primeiro, o modelo de definição de cada símbolo, já que é a partir desta definição que o algoritmo deverá se pautar para encontrá-los e segundo, uma abordagem para o reconhecimento, pois os símbolos podem ter uma grande variação de formatos, sendo necessário a aplicação de diferentes técnicas para realizar o reconhecimento. Para iniciar a análise de uma imagem, Szeliski (2010) afirma que inicialmente é preciso entender a formação geométrica da imagem. Com isso, ele se refere a quais primitivas geométricas são utilizadas na imagem, desde pixels, até linhas e formas bidimensionais. A partir dessa análise, pode-se descrever o que está sendo exibido (SZELISKI, 2010).

Gharde, Nemade e Adhiya (2013) dividem o processamento para reconhecimento de símbolos em quatro estágios. A primeira etapa é o pré-processamento, no qual são executados diversos filtros morfológicos e de remoção de ruídos para preparar a imagem para o processamento. Após isso, os autores sugerem uma etapa de segmentação, na qual a imagem é decomposta em diferentes partes, separando assim seus símbolos. Em seguida a etapa de extração de características e utilizada para descrever cada um dos símbolos encontrados na decomposição. Por fim, os autores detalham a etapa de classificação. Nesta etapa são utilizadas as informações extraídas de cada símbolo para definir o que eles representam na imagem.

Gonzalez e Woods (2008) apontam que os operadores morfológicos podem ser utilizados como ferramentas para conseguir descrever componentes da imagem ou separar regiões internas da mesma (GONZALEZ; WOODS, 2008). Dentre as operações morfológicas existentes, duas são as mais comumente utilizadas: erosão e dilatação. Estas operações acabam sendo base também para outros filtros morfológicos que acabam muitas das vezes fazendo uma combinação destes para criar novos filtros tais como abertura e fechamento (RUSS, 2002).

A erosão é um filtro morfológico que possui a capacidade de afinar componentes de uma imagem, além de remover ruídos e sujeiras pequenas na imagem. Este algoritmo é utilizado para imagens binarizadas e funciona aplicando um elemento estruturante quadrado que irá agir como uma máscara para eliminar elementos que não se aplicam a ela (GONZALEZ, WOODS; 2008). Na Figura 3 é possível visualizar quais os efeitos do filtro de erosão em uma imagem utilizando um elemento estruturante 3x3 convoluído a partir de uma imagem original.

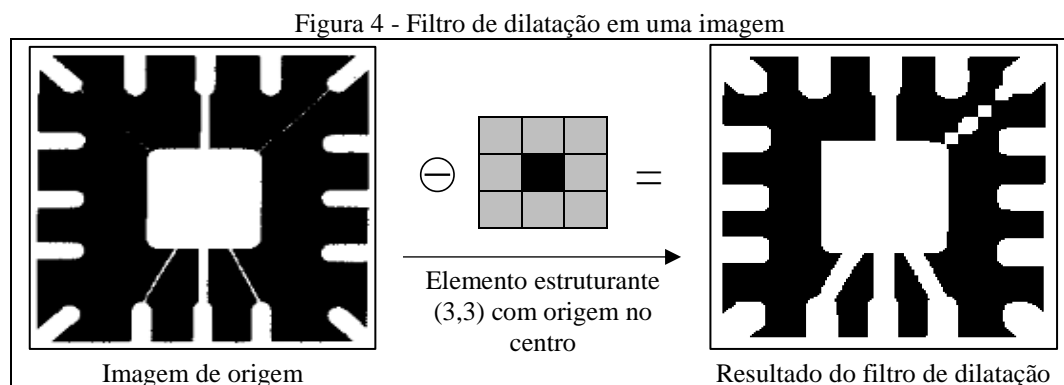
Figura 3 - Filtro de erosão em uma imagem



Fonte: adaptado de Gonzalez e Woods (2008).

Russ (2002) afirma que o principal propósito do filtro de erosão é remover pixels da imagem que não deveriam estar lá, geralmente por conta de uma sujeira na imagem ou uma iluminação indesejada. Russ (2002) também afirma que este filtro pode acabar removendo regiões inteiras de uma imagem quando utilizado um elemento estruturante muito grande, por isso é necessário validar se utilizar somente este filtro atinge o objetivo ou se utilizar operações complementares não trarão um resultado melhor.

A dilatação, por sua vez, é uma operação contrária a erosão. Ao aplicar esse filtro morfológico em uma imagem binarizada, seus componentes internos tendem a crescer ou engrossar (GONZALEZ; WOODS, 2008). Gonzalez e Woods (2008) também afirmam que, da mesma forma que a erosão utiliza um elemento estruturante quadrado para realizar sua operação, a dilatação o utiliza para definir de que forma o engrossamento de cada região será feita. A Figura 4 demonstra os efeitos do filtro de dilatação em uma imagem utilizando um elemento estruturante 3x3 convoluído a partir de uma imagem original.

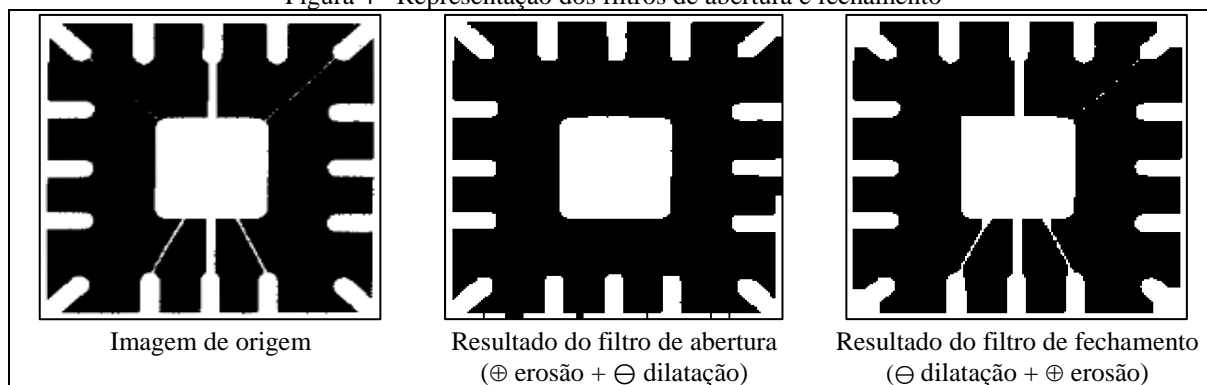


Fonte: adaptado de Gonzalez e Woods (2008).

Gonzalez e Woods (2008) afirmam que uma das vantagens dessa técnica dos filtros de dilatação é que eles podem conseguir unir vãos existentes entre diferentes componentes da imagem, juntando assim regiões que estavam quebradas. Da mesma forma que a erosão, a dilatação também é geralmente utilizada em conjunto com outros filtros morfológicos para conseguir atingir um resultado desejável no processamento de uma imagem (RUSS, 2002).

Os operadores morfológicos de abertura e fechamento são operadores que usam tanto o operador de erosão quanto o de dilatação. O principal objetivo destes filtros é remover sujeiras da imagem sem afetar as características dos principais componentes existentes na imagem (RUSS, 2002). Na Figura 5 são apresentados exemplos com ambos os filtros demonstrando como cada um se comporta.

Figura 4 - Representação dos filtros de abertura e fechamento



Fonte: adaptado de Gonzalez e Woods (2008).

Segundo Gonzalez e Woods (2008), o filtro de abertura se caracteriza por remover as sujeiras do fundo da imagem, além de separar componentes que possuem uma conexão muito fraca. Esse filtro se dá pela execução dos filtros de erosão, seguido do filtro de dilatação utilizando um elemento estruturante. Desta forma, ao afinar os componentes da imagem e remover as sujeiras com o filtro de erosão, a dilatação irá engrossá-los novamente retornando-os a seu tamanho original (GONZALEZ; WOODS, 2008). Já o filtro de fechamento por sua vez, tem como principais características juntar componentes que estão bastante próximos, além de remover furos dentro das regiões da imagem (GONZALEZ; WOODS, 2008). Russ (2002) afirma que, ao contrário do filtro de abertura, o filtro de fechamento é a junção do filtro de dilatação seguido do filtro de erosão. A alteração da ordem dos filtros serve justamente para primeiramente realizar a junção dos componentes e furos, engrossando as regiões da imagem e posteriormente retorná-las a seu tamanho inicial afinando com o filtro de dilatação.

2.3 TRABALHOS CORRELATOS

A seguir são apresentados quatro trabalhos que possuem características semelhantes aos principais objetivos do trabalho desenvolvido. O Quadro 1 apresenta a ferramenta Visual QR Codes, desenvolvida pela empresa chinesa Visualead (VISUALEAD, 2014) que realiza a customização de QR Codes alterando as cores e aplicando uma imagem ao fundo. No Quadro 2, é descrito o Halftone QR Codes (CHU et al., 2013) que apresentam uma técnica de customização de QR Codes utilizando o algoritmo de Halftone. Já no Quadro 3 é descrita a técnica Efficient QR Code Beautification (LIN et al., 2015), em que os autores propõem inserir imagens dentro de um QR Code sem perder sua qualidade. Por fim, no Quadro 4 é apresentado o método ART-UP (XU et al., 2015) que permite a junção de imagens ao QR Code sem perda de qualidade.

Quadro 1 – Visual QR Codes

Referência	Visualead (2014)
Objetivos	De acordo com os autores, este serviço permite aos usuários customizar um QR Code com cores e imagens, tentando manter a estrutura original do QR Code, sem remover os pixels de controle para leitura.
Principais funcionalidades	A ferramenta busca, segundo os autores, permitir que o usuário aplique uma imagem em um QR Code e, ao mesmo tempo, permite alterar as cores e arredondar as bordas dos padrões de orientação do QR Code.
Ferramentas de desenvolvimento	O serviço utiliza a mutação dos pixels do QR Code para conseguir exibir uma imagem e alterar as cores. Essa tarefa é feita no momento da geração da imagem final do QR Code. Este serviço pode ser acessado através de uma interface WEB que possibilita a geração de códigos com qualquer imagem.
Resultados e conclusões	O código final gerado mantém os padrões de orientação do QR Code quase que intactos. Por isso, a sua leitura não é danificada. Os autores ressaltam que a imagem inserida, por sua vez, acaba sofrendo bastante distorção para que os pixels do QR Code continuem visíveis, por isso imagens com muitos detalhes acabam perdendo sua definição. A ferramenta ainda é proprietária, não existindo outra forma de gerar imagens a não ser acessando sua ferramenta WEB.

Fonte: elaborado pelo autor.

Quadro 2 – Halftone QR Codes

Referência	Chu et al. (2013)
Objetivos	Os autores objetivavam exibir uma figura dentro da estrutura de um QR Code utilizando a técnica Halftone.
Principais funcionalidades	Utilizando a técnica Halftone, os autores buscam inserir imagens dentro de um QR Code alterando a quantidade de pixels internos, porém mantendo a mesma densidade de modo que a leitura ainda seja possível.
Ferramentas de desenvolvimento	Os autores desenvolveram o próprio código utilizando a técnica Halftone para geração de imagens. Com base na estrutura padrão de um QR Code, são alterados os pixels do código para uma quantidade maior de pixel, porém com uma densidade igual, permitindo assim desenhar a imagem desejada de uma forma que o QR Code mantenha a sua estrutura padrão.
Resultados e conclusões	Este método consegue inserir imagens não muito complexas dentro do QR Code, porém de acordo com os autores, possui como limitação a falta de coloração devido a utilização do algoritmo Halftone. Da mesma forma, os padrões de orientação do código necessários para leitura continuam ressaltados na imagem final. Além disso, se o tamanho da imagem de entrada for muito grande, o resultado não terá boa qualidade.

Fonte: elaborado pelo autor.

Quadro 3 – Efficient QR Code Beautification

Referência	Lin et al. (2015)
Objetivos	Nesta publicação os autores buscam explorar os dispositivos de correção de erros do QR Code para que seja possível inserir uma imagem em pontos que não danificam a estrutura principal de leitura do QR Code.
Principais funcionalidades	Os autores permitem, através deste projeto, adicionar uma imagem completa com cores e sem deformação na parte interna de um QR Code, deixando com que a leitura seja feita somente com a parte externa.
Ferramentas de desenvolvimento	A estrutura padrão de um QR Code possui diversos mecanismos de redundância para correção de erros na hora da leitura. Sabendo disso, Lin et al. (2015) exploraram esses mecanismos, removendo pixels que não seriam necessários em um QR Code na sua leitura ideal, para inserir uma imagem por completo, sem que ela sofra distorções.
Resultados e conclusões	A partir dos resultados obtidos por Lin et al. (2015), percebe-se que o método se mostra bastante eficaz, pois consegue manter boa parte da imagem original. Contudo, o algoritmo perde performance na decodificação do código pois muitos pontos de controle são removidos. Outra limitação é que somente o centro da imagem possui coloração, restringindo a porção de customização.

Fonte: elaborado pelo autor.

Quadro 4 – ART-UP

Referência	Xu et al. (2015)
Objetivos	Os autores buscavam utilizar técnicas de mutação de pixels e exploração dos mecanismos de redundância do QR Code para tornar possível a inserção dele em uma imagem, mantendo boa resolução e sem perder a capacidade de leitura.
Principais funcionalidades	Com as técnicas utilizadas, os autores conseguem inserir imagens coloridas dentro de um QR Code com uma quantidade bem baixa de distorção da imagem original. Contudo, ainda respeitando os padrões de orientação e pixels de leitura do código.
Ferramentas de desenvolvimento	Explorando tanto os dispositivos de correção de erro do QR Code, quanto ajustando os pixels da imagem que se deseja inserir, os autores conseguiram fazer a inserção de uma imagem com o máximo de aproveitamento dentro do código, permitindo que detalhes continuem visíveis sem que o QR Code perca sua legibilidade.
Resultados e conclusões	Segundo Xu et al. (2015), os padrões de orientação do QR Code permanecem sem alteração. Ou seja, não geram dificuldades para os leitores encontrá-los na imagem. As imagens geradas possuem um pequeno nível de distorção, mas mantendo grande parte de seus detalhes. Mesmo assim, o autor ressalta que o resultado é bastante satisfatório, gerando imagens com boa resolução.

Fonte: elaborado pelo autor.

3 DESCRIÇÃO DA APLICAÇÃO

Este capítulo tem como objetivo descrever os aspectos técnicos e as especificações utilizadas para o desenvolvimento de um código de barras bidimensional customizável. Também será descrita a abordagem utilizada para o desenvolvimento da solução.

3.1 ESPECIFICAÇÃO

Para que o protótipo pudesse ter um ciclo de vida similar ao de um código de barras bidimensional, dividiu-se o processo de elaboração em duas partes. A primeira etapa é o *encoder*, ou seja, a ferramenta utilizada para geração das imagens codificadas. Já a segunda, trata-se do *decoder*, que é a ferramenta que permite decodificar essas imagens através de uma interface gráfica. Com base nisso, foram definidos alguns Requisitos Funcionais (RF) a serem atendidos pelas ferramentas, relacionando-os com cada funcionalidade mencionada nos casos de usos conforme descrito no Quadro 5. Além disso, também são descritos os Requisitos Não Funcionais (RNF) no Quadro 6.

Quadro 5 – Requisitos funcionais

Requisito funcional	Caso de uso
RF01: permitir a codificação de mensagens em imagens	UC01
RF02: permitir a customização de códigos gerados	UC02
RF03: permitir adicionar figura ao código	UC03
RF04: permitir a decodificação de mensagens codificadas	UC04

Fonte: elaborado pelo autor.

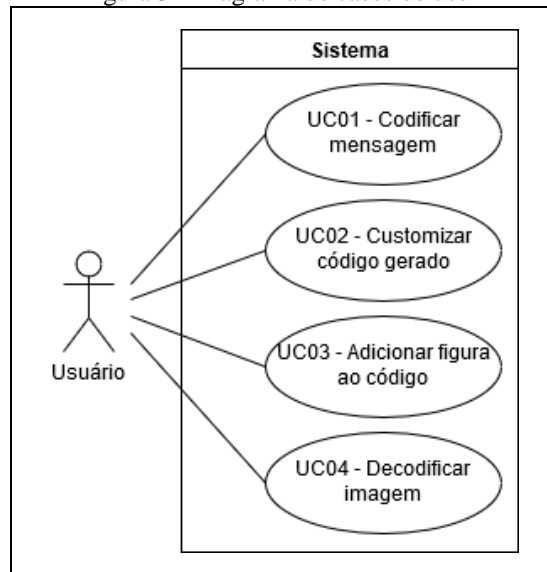
Quadro 6 – Requisitos não funcionais

Requisito não funcional
RNF01: permitir a geração de imagens codificadas através de plataforma WEB
RNF02: desenvolver o <i>encoder</i> utilizando a linguagem Javascript
RNF03: permitir a decodificação de imagens utilizando a câmera através de uma aplicação Android
RNF04: utilizar as bibliotecas OpenCV para o processamento de imagens.

Fonte: elaborado pelo autor.

A Figura 5 apresenta o diagrama de caso de uso, que especifica a relação do usuário do protótipo com as funcionalidades desenvolvidas. Neste caso, o usuário sendo o único ator na utilização da aplicação ele terá a possibilidade de acionar todos os casos de usos descritos.

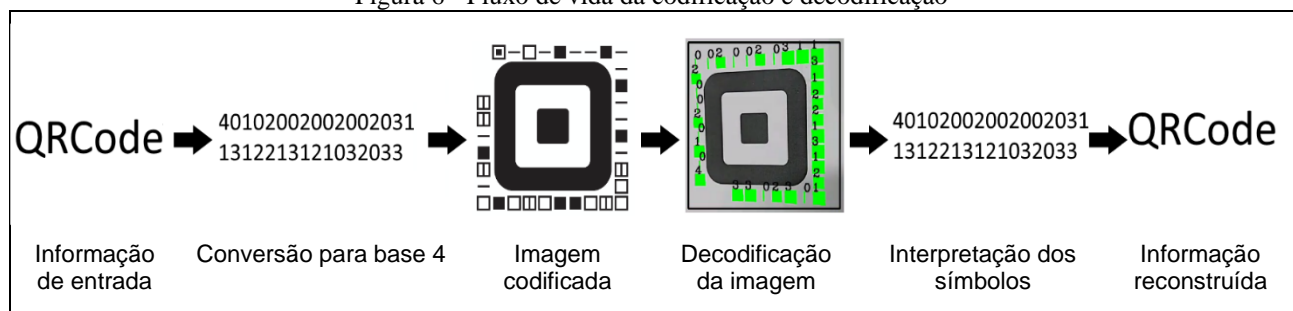
Figura 5 - Diagrama de casos de uso



Fonte: elaborado pelo autor.

O fluxo de vida da solução proposta inicia-se pela conversão da informação de entrada para um texto em base 4, transformando-o em uma imagem. Esta mesma imagem é decodificada pelo *decoder*, que por sua vez interpreta os valores dos símbolos encontrados, reconstruindo a informação original de entrada. A Figura 6 representa como este fluxo ocorre e, nas próximas seções, ele é detalhado.

Figura 6 - Fluxo de vida da codificação e decodificação

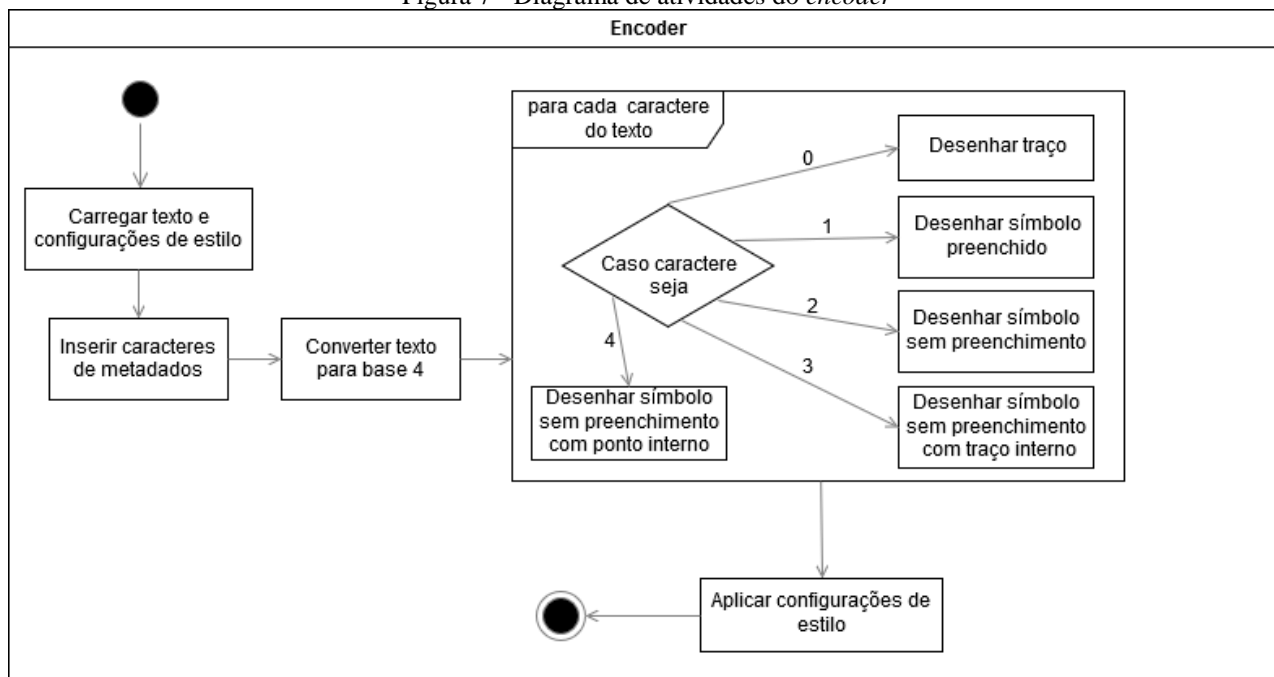


Fonte: elaborado pelo autor.

3.2 ENCODER

Para criar um código de barras bidimensional customizável primeiramente foi estabelecido uma forma de transformar uma informação em imagem. Portanto, é importante definir um padrão de imagem que seja diferente para toda informação, sem causar ambiguidade, e que mesmo assim siga uma sequência que seja decodificada. A partir disso, a solução proposta foi o desenvolvimento de um padrão de codificação que permite adicionar imagens ao centro da informação decodificável e com diferentes formatos e símbolos para sua exibição. Na Figura 7 é apresentado um diagrama de atividades exemplificando os passos utilizado no desenvolvimento do *encoder*.

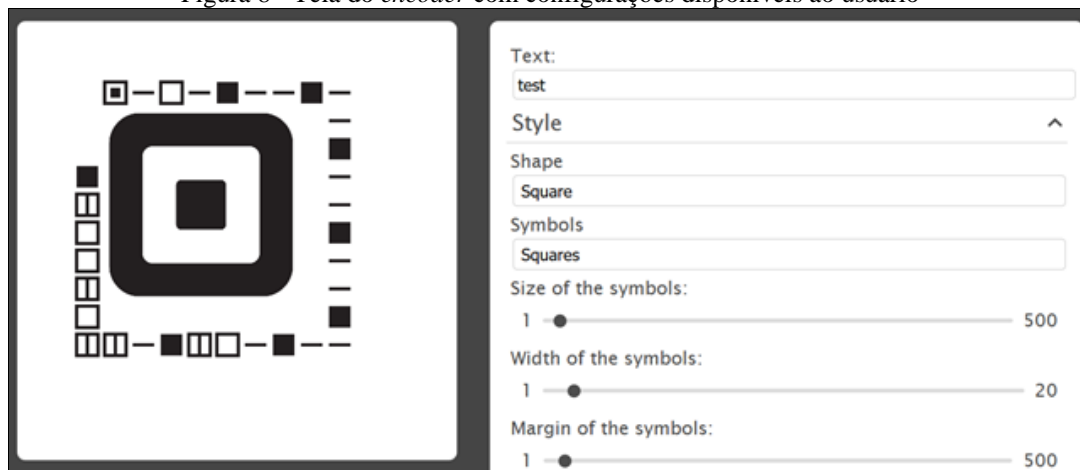
Figura 7 - Diagrama de atividades do *encoder*



Fonte: elaborado pelo autor.

Primeiramente, o usuário informa o texto de entrada que será codificado e quais serão as configurações de estilos aplicados na imagem gerada. É com base neste texto que será gerado um código válido com estilos e posição de cada símbolo, além do formato, cor e rotação da imagem final gerada. Como o algoritmo efetua a codificação com base no texto informado, é possível que em uma sequência de caracteres muito grande a imagem fique com um tamanho que não seja possível decodificá-la, sendo esse um dos principais limitadores da funcionalidade. A Figura 8 apresenta a tela ao qual o usuário pode definir o texto de entrada e as configurações de estilo. Nela, a visualização do código gerado pode ser visualizada no canto esquerdo.

Figura 8 - Tela do *encoder* com configurações disponíveis ao usuário



Fonte: elaborado pelo autor.

A partir das informações de entrada, o algoritmo converte a mensagem do usuário para um novo formato que será utilizado para codificá-lo. Este formato será uma conversão de cada caractere da mensagem para uma codificação em base 4, ou seja, um texto que possui até 4 caracteres diferentes sendo eles 0, 1, 2 ou 3. Porém, antes de efetuar a conversão, o tipo de texto informado é analisado a fim de criar os caracteres de metadados. Os caracteres de metadados são os três primeiros caracteres da codificação. O primeiro serve simplesmente para definir o início da mensagem, sendo um símbolo único. O segundo, define se a mensagem representa um endereço de rede, e o terceiro define quais os tipos de caracteres serão utilizados na mensagem. No Quadro 7 é possível verificar quais são os valores válidos para cada caractere dos metadados.

Quadro 7 – Valores válidos para os caracteres de metadados

Função do caractere	Valores válidos	Descrição do valor
Definir início do código	4	Valor único que define o ponto de início do código
Definir se o código é um endereço de rede	0	Define que o código não representa um endereço de rede
	1	Define que o código representa um endereço http
	2	Define que o código representa um endereço https
Definir tipos de caracteres do código	0	Define que o código possui apenas números
	1	Define que o código possui apenas números ou letras
	2	Define que o código possui qualquer caractere UTF-8

Fonte: elaborado pelo autor.

Após a definição dos metadados, os caracteres da mensagem serão codificados de acordo com seu tipo. Para os caracteres que constituem apenas de números ou letras, os mesmos são codificados para valores pré-definidos em uma tabela conforme mostra o Quadro 8. Isto é feito para reduzir a quantidade de símbolos utilizados para representar cada caractere, já que na codificação UTF-8, utilizada pela aplicação, cada caractere é representado por 8 bits. Os demais tipos de textos são transformados para símbolos utilizando o valor decimal da representação UTF-8 convertido para base 4, ou seja, para o valor decimal de cada caractere do texto, como por exemplo a letra “c” que é representada pelo valor decimal “99”. Esse valor decimal é convertido para um numeral com até quatro dígitos distintos, neste exemplo sendo convertido para “1203”. Por fim, alguns caracteres são adicionados no texto para casos em que a mensagem do usuário não seja grande o suficiente para gerar um código relevante, o que resultaria numa imagem com poucos símbolos sem o formato desejado. Desta forma, quando se trata de mensagens que constituem apenas números são adicionados zeros a esquerda para que a mensagem possua pelo menos 15 caracteres. Em textos com letras e números, são adicionados caracteres ponto final (.) a esquerda a fim de ter pelo menos 10 caracteres na mensagem, e nos demais textos também são adicionados caracteres ponto final (.) a esquerda mantendo a mensagem com no mínimo 5 caracteres.

Quadro 8 – Tabela de conversão de caracteres para letras e números

Texto								Números	
caractere	código	caractere	código	caractere	código	caractere	código	caractere	código
0	000	f	100	v	200	L	300	0	01
1	001	g	101	w	201	M	301	1	02
2	002	h	102	x	202	N	302	2	03
3	003	i	103	y	203	O	303	3	10
4	010	j	110	z	210	P	310	4	11
5	011	k	111	A	211	Q	311	5	12
6	012	l	112	B	212	R	312	6	13
7	013	m	113	C	213	S	313	7	20
.	020	n	120	D	220	T	320	8	21
8	021	o	121	E	221	U	321	9	22
9	022	p	122	F	222	V	322		
a	023	q	123	G	223	W	323		
b	030	r	130	H	230	X	330		
c	031	s	131	I	231	Y	331		
d	032	t	132	J	232	Z	332		
e	033	u	133	K	233	/	333		

Fonte: elaborado pelo autor.

Com o texto convertido, o próximo passo é a geração da imagem. Para cada caractere, será utilizado um símbolo que irá representá-lo nos quais serão utilizadas as definições dadas pelo usuário para desenhá-los. A posição de cada símbolo é dada com base no formato de imagem escolhido pelo usuário, sendo possível optar por quadrado, triângulo ou círculo. O caractere inicial, definido com o valor 4, será utilizado somente uma vez na imagem e será representado na imagem por um símbolo sem preenchimento, porém com um ponto interno preenchido. Para o resto dos caracteres, a representação é dada de acordo com as descrições do Quadro 9.

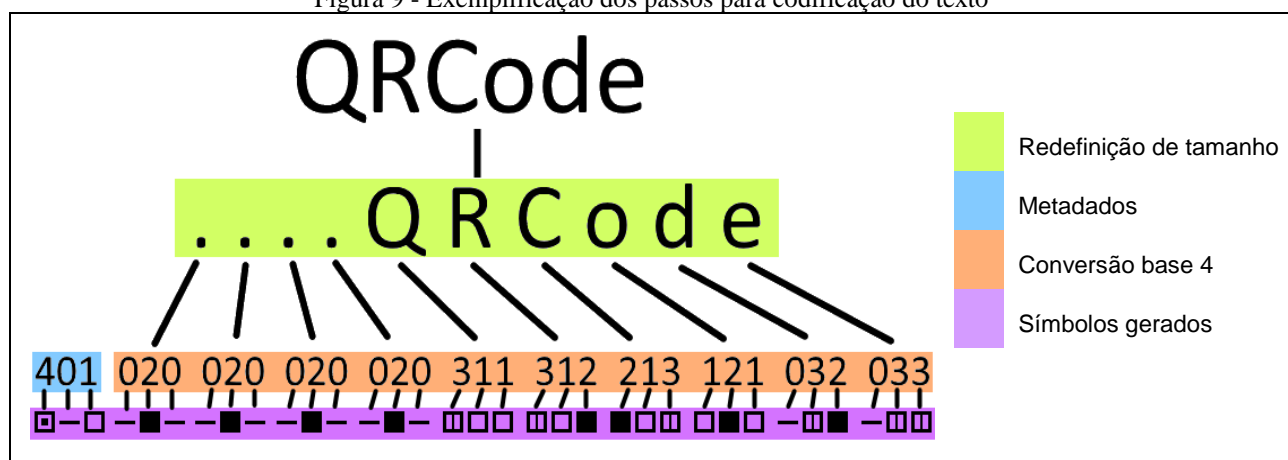
Quadro 9 – Representações para cada valor numérico

Caractere	Símbolo	Representação
0	—	Representado com um traço
1	□ ○ △	Representado com o símbolo escolhido sem preenchimento
2	■ ● ▲	Representado com o símbolo escolhido com preenchimento
3	▢ ⊖ ▲	Representado com o símbolo escolhido com um traço ao centro
4	▣ ⊙ ▲	Representado com o símbolo escolhido com um ponto interno

Fonte: elaborado pelo autor.

Na Figura 9 é possível verificar os passos da conversão de um texto para os símbolos utilizados na codificação/decodificação da mensagem.

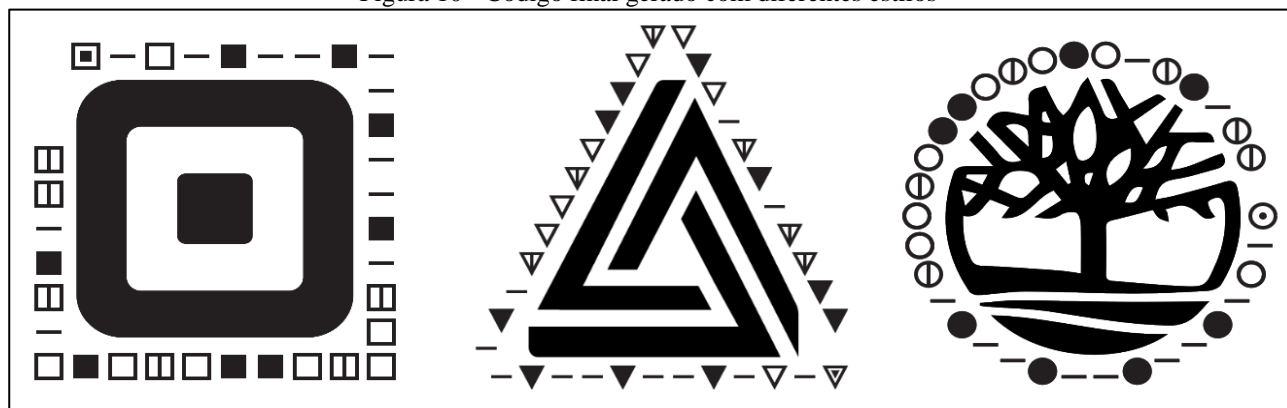
Figura 9 - Exemplificação dos passos para codificação do texto



Fonte: elaborado pelo autor.

Após ter todos os caracteres projetados na imagem, aplica-se as configurações finais. A imagem informada pelo usuário é inserida no centro do código gerado, as cores do fundo e dos símbolos são alteradas para as desejadas e a rotação, tanto da imagem quanto da forma final do código é realizada. É importante ressaltar que a imagem escolhida pelo usuário é a componente mais destacada da imagem, tornando a informação descodificável em segundo plano, ou seja, perde a sua notoriedade. Isso é uma característica que permite com que os códigos gerados tenham seu foco na ideia central, porém sem perder a capacidade da decodificação. Na Figura 10 são apresentadas algumas das formas visuais que podem ser geradas a partir do mesmo texto de entrada.

Figura 10 - Código final gerado com diferentes estilos



Fonte: elaborado pelo autor.

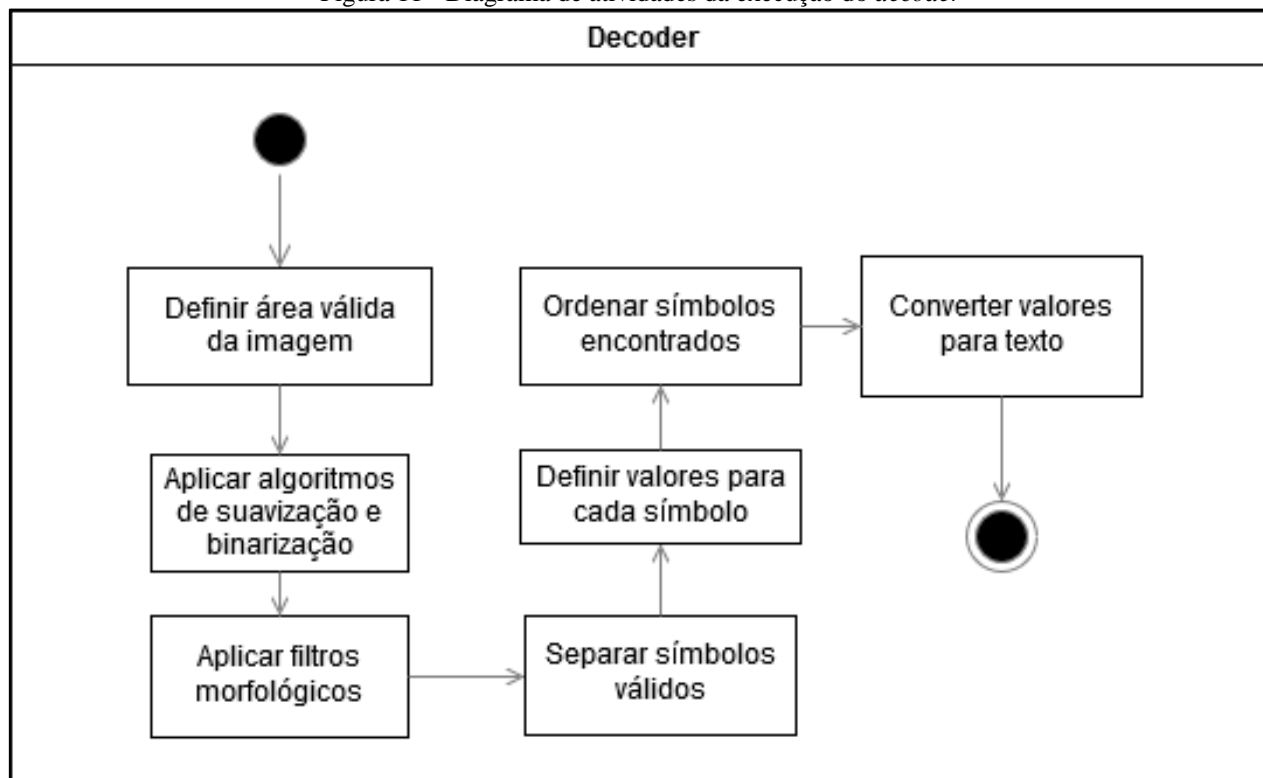
Com a imagem final gerada, o usuário pode verificar se o resultado é o desejado, e, caso deseje, pode voltar a alterar as configurações de estilo e gerar um novo código. Por mais que o algoritmo consiga gerar as imagens

automaticamente com base nas configurações informadas, o usuário possui uma função crucial que é verificar se o código possui o visual esperado e se os símbolos não se sobrepõem, o que causaria erros da hora de decodificá-lo.

3.3 DECODER

O *decoder* é a ferramenta que conseguirá, a partir de uma imagem com o código gerado, decodificá-la, retornando a mensagem escondida. A decodificação funciona a partir de uma imagem de entrada que será processada, e, caso seja encontrado alguma ocorrência do código criado, retorna o texto encontrado. Danificações na imagem de entrada podem acabar afetando o processamento, causando resultados inesperados e por isso, é importante que ela seja gerada corretamente. Na Figura 11 é apresentado um diagrama de atividades com os passos realizados na implementação do *decoder*.

Figura 11 - Diagrama de atividades da execução do *decoder*

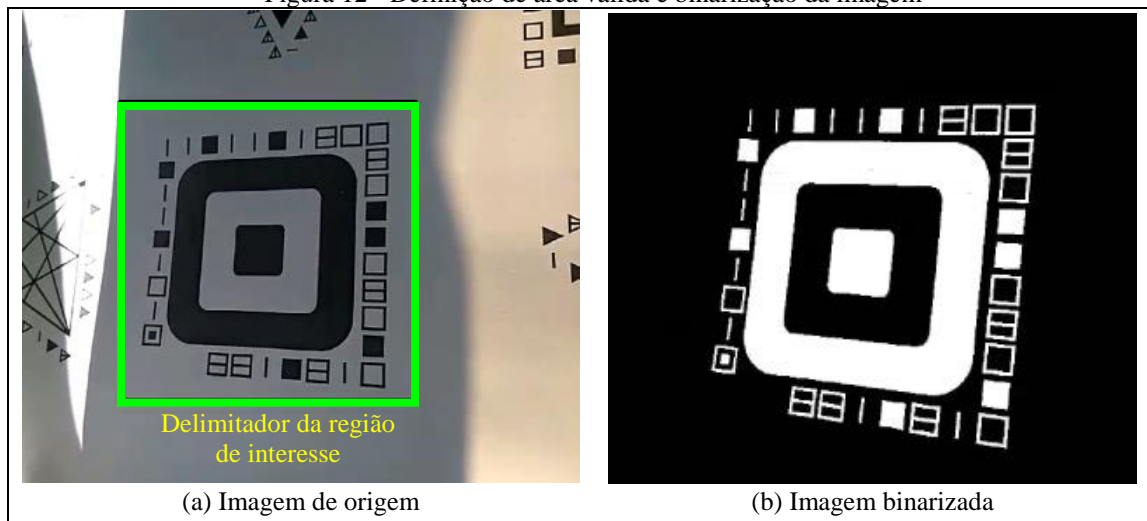


Fonte: elaborado pelo autor.

Inicialmente o algoritmo estabelece uma área válida para detecção da imagem. Como a imagem pode possuir muitos ruídos externos ao código válido, foi definida uma região de interesse ao qual o símbolo deve estar posicionado. Desta forma, o algoritmo consegue ignorar todas as informações externas e focar apenas no que será decodificado.

Ao definir a região de interesse que será decodificada, são aplicados alguns algoritmos para suavização e binarização a fim de remover ruídos e melhorar a detecção dos símbolos. Com isso, primeiramente é aplicado o filtro Gaussiano para suavizar a imagem e remover pequenos pontos e imperfeições que poderiam afetar o algoritmo. Na sequência é aplicado o método Otsu de Thresholding, que a partir da média de valores do histograma da imagem consegue detectar qual será o limiar utilizado para formação da imagem binarizada. Na Figura 12 é possível verificar a delimitação da área de interesse e o processo de binarização que irá separar apenas as informações relevantes.

Figura 12 - Definição de área válida e binarização da imagem

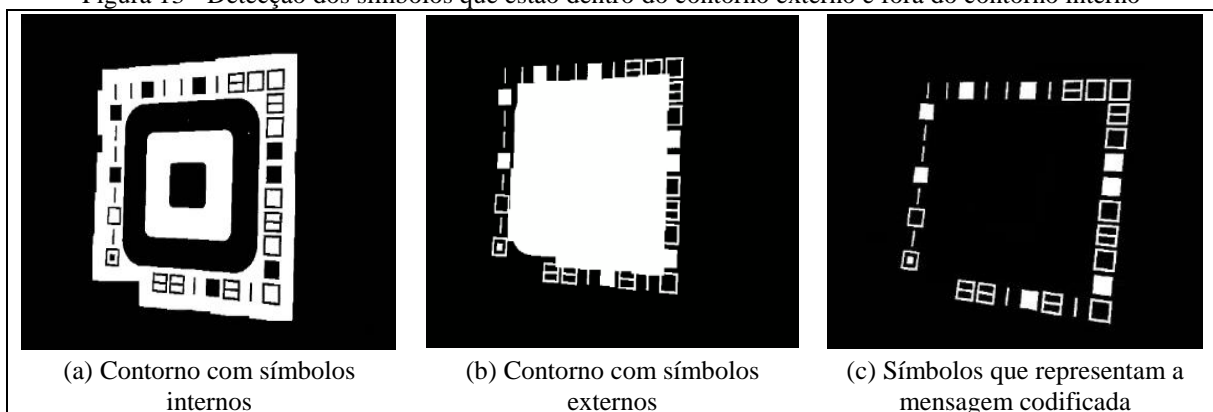


Fonte: elaborado pelo autor.

Com a imagem binarizada, o próximo passo é a execução de filtros morfológicos. Estes são utilizados para ressaltar os símbolos válidos da imagem e detectar qual o formato do código final. O primeiro filtro aplicado é o de dilatação, que irá remover pequenos pontos da imagem e manter somente símbolos de interesse. Posteriormente, utiliza-se o filtro de fechamento, onde os símbolos do código irão se juntar, formando assim o contorno do formato geral da imagem. É desse local que o algoritmo buscará os símbolos para decodificação.

Tendo o contorno final da imagem, aplica-se um filtro morfológico de erosão para gerar uma versão reduzida. Isso é feito pois os símbolos válidos da codificação ficam na parte externa da imagem, logo, é necessário que toda informação externa a ele seja ignorada. Tendo o contorno externo, encontrado com o filtro de fechamento, e o contorno interno, encontrado com o filtro de erosão, são separados todos os símbolos que estão contidos no contorno externo, porém, que não estão no contorno interno, sendo esses os símbolos que representam a mensagem codificada. Estas etapas são exemplificadas na Figura 13.

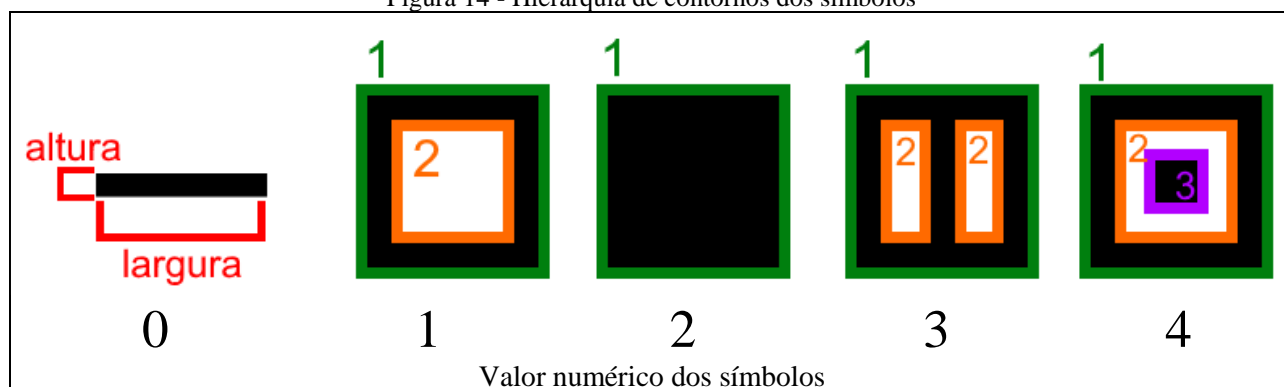
Figura 13 - Detecção dos símbolos que estão dentro do contorno externo e fora do contorno interno



Fonte: elaborado pelo autor.

Após a obtenção dos símbolos que representam a mensagem codificada, o algoritmo passa a realizar validações para definir qual é o valor numérico de cada um, neste caso, comparando suas características, como largura, altura e preenchimentos, com os símbolos gerados pelo *encoder*. Para o símbolo 0, definido por um traço, o algoritmo verifica se a sua largura possui mais que o dobro da sua altura, sendo o único dos símbolos nesta condição. Para os outros símbolos utilizados, são verificados a hierarquia de contornos que cada um possui e, com base nessa informação, os valores são determinados. A Figura 14 exemplifica como esses contornos são definidos.

Figura 14 - Hierarquia de contornos dos símbolos



Fonte: elaborado pelo autor.

A partir da Figura 15 percebe-se que todos símbolos possuem diferentes tipos de contornos internos. Sendo assim, a verificação da hierarquia possibilita definir para cada símbolo o valor numérico original da codificação. Os símbolos se dispõem entre os que possuem apenas contorno externo, os que possuem 1 ou 2 contornos internos, e os que possui um contorno interno no qual também possui um contorno filho. De acordo com essas características, os valores numéricos para cada símbolo são determinados conforme descritos no Quadro 10.

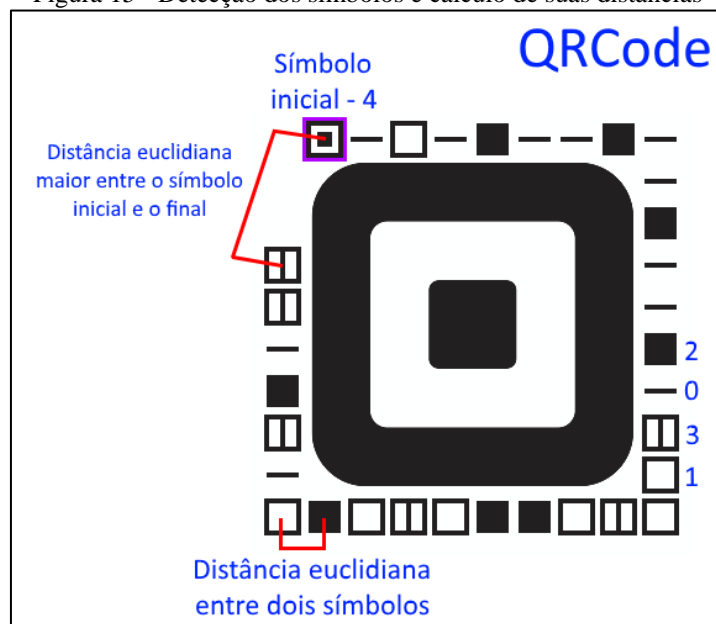
Quadro 10 – Definições de valores com base em características dos símbolos

Valor	Símbolo	Características
0	—	Símbolos em que a largura é duas vezes maior que a altura
1	□ ○ △	Símbolos que não possuem preenchimento nenhum
2	■ ● ▲	Símbolos que possuem preenchimento completo
3	▢ ⊖ ▲	Símbolos que não possuem preenchimento, porém possuem uma linha interna
4	◻ ⊙ ▲	Símbolos que não possuem preenchimento, porém possuem um ponto interno

Fonte: elaborado pelo autor.

Com os valores definidos para todos os símbolos, o algoritmo passa a realizar a concatenação desses símbolos em ordem para decodificar a mensagem. O símbolo inicial é representado pelo valor quatro, que só existe uma vez na codificação. Ao identificá-lo, o algoritmo consegue estabelecer a coordenada inicial do código e, com base nisso, identifica o resto dos valores. Como no *encoder* foi utilizada a estratégia de manter o último símbolo da mensagem um pouco distante do símbolo inicial, é possível definir que o símbolo com menor distância euclidiana em relação ao símbolo atual é o próximo na sequência. Desta forma, o algoritmo segue para cada símbolo, encontrando o próximo com a menor distância euclidiana que ainda não tenha sido percorrido. Na Figura 15 é demonstrado como é feita a definição dos valores para cada símbolo e a forma em que a distância euclidiana é calculada. Nela, é possível ver também, além da distância entre dois símbolos, no qual está definido o símbolo inicial, como ele tem uma distância maior em relação ao último, evitando assim que a decodificação ocorra de forma reversa.

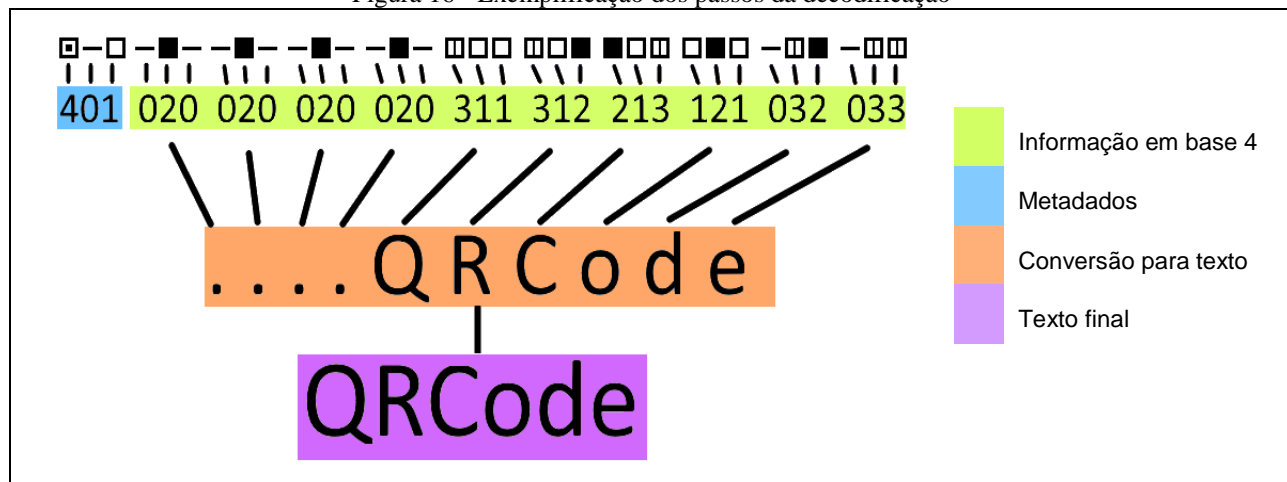
Figura 15 - Detecção dos símbolos e cálculo de suas distâncias



Fonte: elaborado pelo autor.

Tendo a sequência de valores, o próximo passo é realizar a validação dos caracteres a partir dos metadados. Neste caso, de forma reversa ao que é feito no *encoder*, o algoritmo define se o texto codificado é um endereço de rede e qual o tipo de caracteres utilizados, conforme Quadro 7. Caso a mensagem seja constituída apenas por números ou letras, os valores são convertidos utilizando as tabelas do Quadro 8. Caso contrário, os valores são convertidos para decimais, buscando sua representação de acordo com a codificação UTF-8. A Figura 16 exemplifica os passos executados para conversão dos símbolos encontrados para a informação inicial.

Figura 16 - Exemplificação dos passos da decodificação



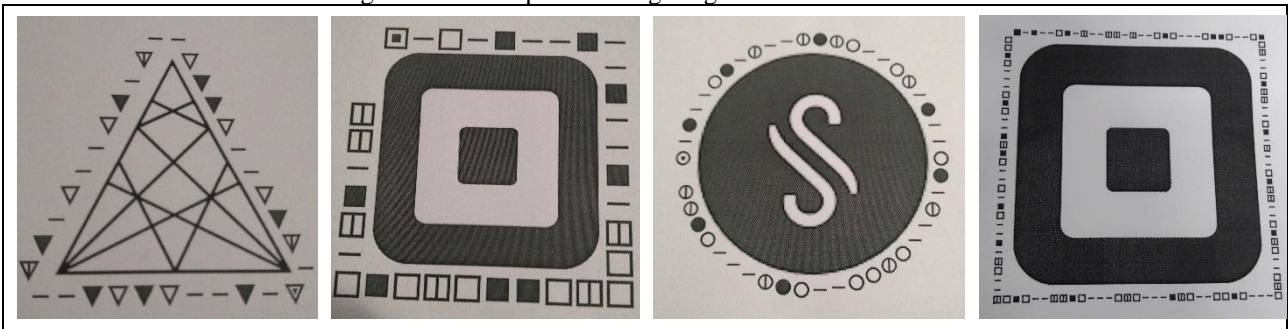
Fonte: elaborado pelo autor.

Ao final, têm-se a mensagem completamente decodificada. Ela é apresentada ao usuário no canto superior da tela da aplicação. Caso os metadados identifiquem que a mensagem representa um endereço de rede, o mesmo é aberto utilizando a aplicação padrão do dispositivo para conexão em websites.

4 RESULTADOS E DISCUSSÕES

Para a realização dos testes, tanto do *encoder* quanto do *decoder*, foi criada uma base de imagens com diversas situações. A base possui cerca de 30 imagens, contendo códigos em formatos quadrados, circulares e triangulares, além de utilizar mensagens de diferentes comprimentos, contendo caracteres numéricos, alfanuméricos e outros. Também foram criadas imagens codificando endereços de rede para validar o redirecionamento automático do *decoder*. A Figura 17 apresenta alguns exemplos constantes na base de imagens. No Apêndice A têm-se a base completa, com todas as imagens utilizadas nos testes.

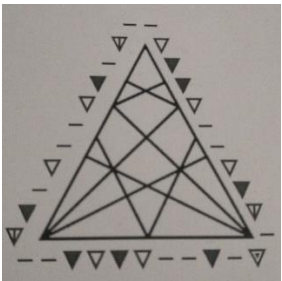
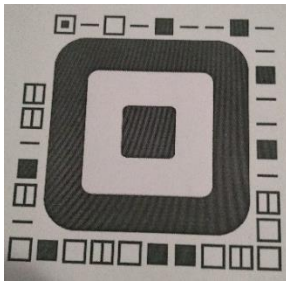
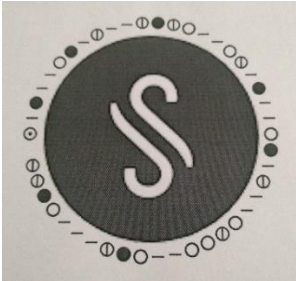

Figura 17 - Exemplos de imagens geradas com o *encoder*



Fonte: elaborado pelo autor.

A solução desenvolvida conseguiu, de forma geral, codificar os mais variados tipos de informação em imagens. No entanto, o *encoder* pode encontrar limitações em relação a quantidade de informação a serem codificadas. Como a mensagem do usuário é convertida para base quatro, é possível que a quantidade de símbolos utilizados para representar cada caractere no código final aumente substancialmente em mensagens muito extensas, dificultando a decodificação. Este problema é mais evidente ao utilizar caracteres que não sejam letras ou números, que, por não possuir uma forma eficaz de codificar o texto, acabam gerando um código com muitos símbolos, sem espaço para representação de todos na imagem. Portanto, para os testes do *decoder*, não foram utilizadas imagens com textos com mais de 25 caracteres, pois a decodificação ficaria comprometida por conta da quantidade de símbolos. Na Figura 18 é possível ver um exemplo de imagens que foram selecionadas e os casos em que elas foram desconsideradas.

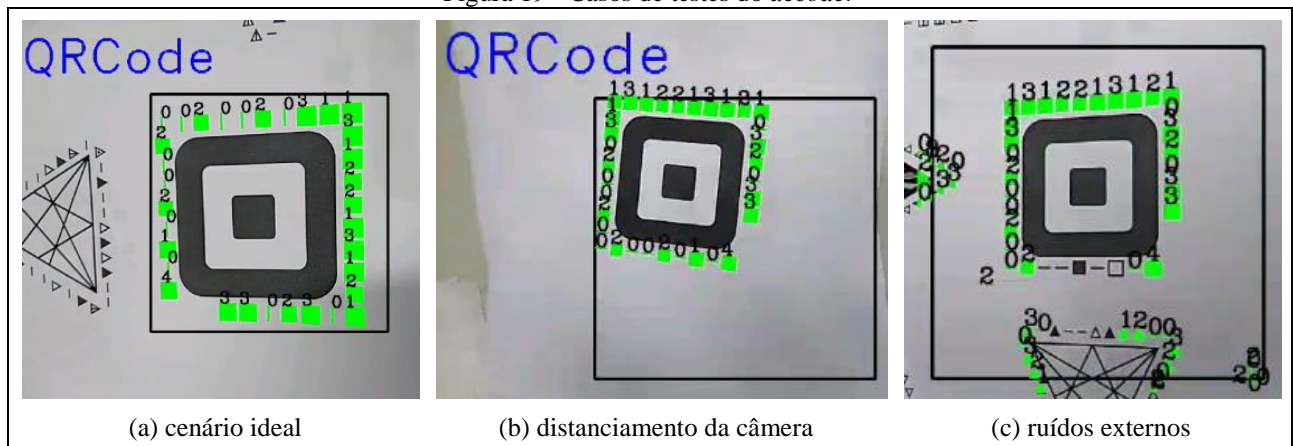
Figura 18 - Exemplo de imagens consideradas e desconsideradas

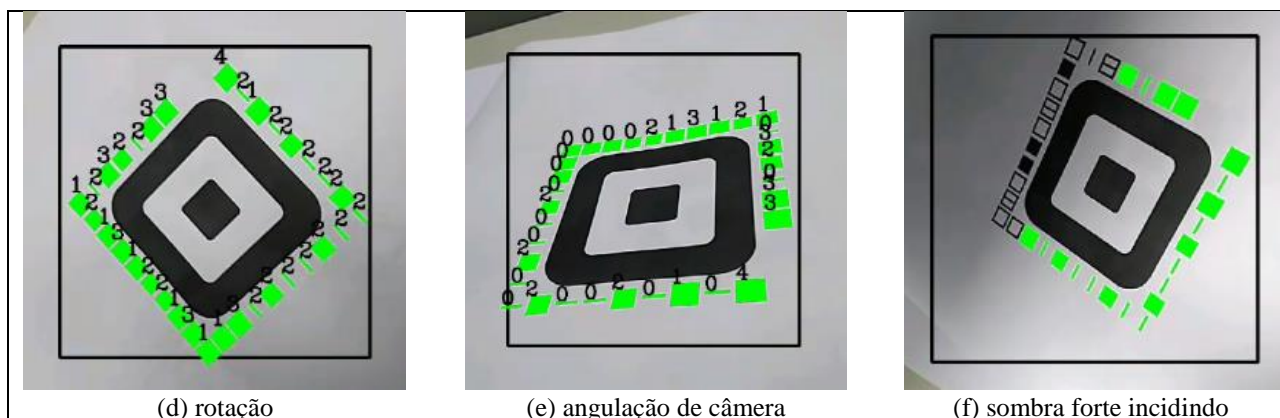
			
Texto codificado => fácil	QRCode	círculo	códigocommuitotextoescrito
Imagens com texto válido			Imagem com texto muito extenso

Fonte: elaborado pelo autor.

Na validação do *decoder* foram realizados vários testes em diferentes cenários. Foram realizados testes em situações ideais, na qual o código fica exatamente no centro da câmera sem ruídos ou sombras incidindo na imagem, testes com sombra incidindo sobre o código, testes com a câmera rotacionada e inclinada e testes apresentando símbolos externos dentro da área utilizada para decodificação. Na Figura 19 é possível verificar como cada caso de teste foi aplicado em uma imagem.

Figura 19 - Casos de testes do *decoder*





Fonte: elaborado pelo autor.

De acordo com os testes realizados, o *decoder* consegue realizar o seu papel em diversos casos quando a área de validação é respeitada, possuindo algumas deficiências em imagens com muitos ruídos externos ou sombras. Diante de cenários favoráveis, todas as imagens da base foram decodificadas. Apesar disso, mesmo sendo realizados filtros para remoção de ruídos, os testes não apresentam sucesso em ocasiões em que a imagem apresenta sombras, rasuras ou cortes incidindo na parte em que o código é disposto, já que o algoritmo não dispõe de mecanismos de redundância de informação.

A distância da câmera em relação ao código não gera muitos problemas para decodificação quando não existem muitos ruídos na imagem, porém em casos que existem pixels ou regiões dentro da área válida utilizada pelo aplicativo e que são externos aos símbolos decodificáveis, o algoritmo acaba considerando estes também como símbolos. Por conta disso, é sugerido que o usuário respeite sempre o quadrado que define a área válida do aplicativo.

Ao tentar decodificar um código rotacionado, o *decoder* acaba tendo problemas quando a rotação está com aproximadamente 45°. Isso acontece, pois, para decodificar as linhas do código ele procura símbolos cujo a *bounding box* seja um retângulo com largura maior que a altura. Nestas rotações, essa validação não é verdadeira, causando erro na hora de decodificar o código completo. Além disso, pode-se notar que diferentes angulações da câmera também causaram erros na decodificação. Estes erros ocorrem quando a câmera está disposta em um ângulo que faça com que o primeiro símbolo do código fique mais próximo do último, causando com que o algoritmo decodifique a mensagem na direção errada, ou em casos que alguns símbolos acabam sendo confundidos por outros por conta do ângulo de visão.

5 CONCLUSÕES

O desenvolvimento do presente artigo possibilitou compreender de forma mais aprofundada as características de um código de barras bidimensional, além de permitir a análise de diferentes técnicas para a codificação da mensagem. Ao desenvolver uma nova forma de codificação e representação visual, foi possível perceber a real importância de alguns mecanismos utilizados pelas alternativas já existentes, como redundância para a correção de erros e padrões de fácil decodificação.

Apesar das limitações encontradas, a criação do codificador conseguiu alcançar o seu objetivo sem grandes problemas, permitindo ao usuário codificar qualquer mensagem. É relevante ressaltar que a escolha dos símbolos utilizados na hora da codificação é pertinente, já que símbolos com muitos ângulos ou vértices, como por exemplo uma cruz, podem ser mais difíceis de serem decodificados e podem acabar gerando uma demora maior na decodificação. Também é necessário levar em consideração a forma em que uma informação é codificada, já que a codificação de uma mensagem com muitos caracteres pode acabar gerando códigos com uma quantidade muito grande de símbolos para serem decodificados.

Customizar um código de barras bidimensional pode ser uma tarefa trabalhosa, porém com o desenvolvimento do artigo foi possível notar que ao projetar uma ferramenta tendo a customização como foco principal, a ação mostra-se mais simples de ser alcançada. Por mais que a customização acabe gerando algumas limitações na quantidade de caracteres da mensagem codificável, ela consegue atingir propósitos visuais mais facilmente quando utilizado mensagens com caracteres.

A decodificação das imagens, por sua vez, se mostrou um pouco mais trabalhosa por ter que lidar com mais adversidades externas na sua execução. A solução desenvolvida conseguiu realizar o proposto quando apresentado em condições favoráveis, porém alguns fatores dificultadores foram encontrados que reduzem a sua taxa de acerto. Condições como sombreado e rotação são alguns dos fatores encontrados que podem prejudicar a leitura do código pela ferramenta, sendo estes fatores em que se pode buscar serem corrigidos em eventual extensão deste trabalho. É visto que ainda se tem

outros fatores a evoluir, como a implantação de um mecanismo de correção de erros, que poderia causar uma taxa de reconhecimento ainda maior.

Por fim, a aplicação desenvolvida se mostrou satisfatória dentro do escopo desejado, conseguindo atingir seu objetivo de gerar uma codificação visual customizável e decodificável. Dentro disso, a ferramenta desenvolvida permite, em conteúdos de mídia, inserir códigos reconhecíveis digitalmente trazendo uma forma simples, porém ainda visualmente agradável, de interligar o mundo real com o digital. O artigo pode ainda contribuir para o desenvolvimento de novas formas de codificações visuais que buscam novos tipos de códigos de barras bidimensionais ou até mesmo novas formas de customização de informações em forma de imagem.

REFERÊNCIAS

- ARYACHANDRAN, S.; JYOTHI, R L. Secure Color QR Codes. **IOSR Journal Of Computer Engineering**, Kollam, v. 16, p. 77-85. 2014.
- CHU, Hung-kuo et al. Halftone QR Codes. **Acm Transactions On Graphics**. Hong Kong, v. 32, n. 6, p. 1-8. jul. 2013.
- DENSO WAVE INCORPORATED. **History of QR Code**, 07 abr. 2003. Disponível em: <<https://www.qrcode.com/en/history/>>. Acesso em: 2 jun. 2019.
- GHARDE, Sanjay S.; NEMADE, Vidya A.; ADHIYA, K. P. Design And Implementation of Special Symbol Recognition System using Support Vector Machine. **International Journal Of Innovative Technology And Exploring Engineering**. Jalgaon, v. 2, n. 6, p. 145-148. maio 2013.
- GONZALEZ, Rafael C.; WOODS, Richard E. **Digital image processing**. 3rd ed. Upper Saddle River: Pearson Prentice Hall, 2008.
- HARARI, Yuval Noah. **Sapiens: Uma Breve História da Humanidade**. São Paulo: L&pm Editores, 2015.
- HOLT, Douglas B. **Como As Marcas Se Tornam Ícones**. São Paulo: Cultrix, 2006.
- LIN, Shih-syun et al. Efficient QR Code Beautification With High Quality Visual Content. **IEEE Transactions On Multimedia**, Taiwan, v. 17, n. 9, p.1515-1524, set. 2015.
- LLADOS, J.; SANCHEZ, G. Symbol recognition using graphs. **Proceedings 2003 International Conference On Image Processing**. Barcelona, v. 3, p. 49. set. 2003.
- MOTAHARI, Amin; ADJOUADI, Malek. Barcode Modulation Method for Data Transmission in Mobile Devices. **IEEE Transactions On Multimedia**, Florida, v. 17, n. 1, p.118-127, jan. 2015.
- RUSS, John C. **The Image Processing Handbook**. 4. ed. Raleigh: Crc Press, 2002.
- QIANYU, Ji. **Exploring concept of QR Code and the benefits of using QR Code for companies**. 2014. 54 f. Dissertação (Bacharelado em Administração), Lapland University of Applied Sciences, Rovaniemi, 2014.
- SZELISKI, Richard. **Algorithms and Applications**. New York: Springer, 2010.
- TRIBAK, Hicham; ZAZ, Youssef. QR Code Recognition based on Principal Components Analysis Method. **International Journal Of Advanced Computer Science And Applications**. Tetouan, p. 241-248. abr. 2017.
- VISUALEAD. **Visualead QR codes**, 13 mai. 2014. Disponível em: <<http://blog.visualead.com/visualead-qr-codes/>>. Acesso em: 2 jun. 2019.
- XU, Mingliang et al. ART-UP: A Novel Method for Generating Scanning-robust Aesthetic QR codes. **Journal Of Latex Class Files**. Zhengzhou, p. 1-14. ago. 2015.

APÊNDICE A – BASE DE IMAGENS CODIFICADAS

O Quadro 11 apresenta a base de imagens que foram utilizadas para validação do *Encoder* e *Decoder*.

Quadro 11 - Base de imagens codificadas

2RakHcH	2019	22061996	123456789101112	apple
https://apple.com	ciencia	circulo	círculo	computacao
facebook	https://facebook.com	facil	fácil	frameworks
furb	https://furb.br	FW	https://google.com	instagram
https://instagram.com	mitsubishi	novoteste	QRCode	quadrado
squares	starbucks	https://starbucks.com	testando	test

Fonte: elaborado pelo autor.