**CEEL - ISSN 2596-2221**
**Universidade Federal de Uberlândia**
**05 a 09 de Dezembro de 2022**

# ALGORITMO DE BINARIZAÇÃO PARA IMAGENS QR CODE CONTENDO ILUMINAÇÃO NÃO UNIFORME

Alailton José Alves Júnior*[1], Luciano Xavier Medeiros[1], Alexandre Coutinho Mateus[1] e André Luiz Aguiar da Costa[1]

[1]FEELT - Universidade Federal de Uberlândia

***Resumo* - O QR Code é um exemplo de código de barras 2D amplamente utilizado devido a sua praticidade. A baixa iluminação do ambiente e/ou o sombreamento da imagem criam diferentes tonalidades nos pixels que formam os quadrados neste tipo de código, dificultando sua decodificação. Pensando nisso, este trabalho propõe um novo algoritmo para binarização de QR Code, baseando-se na transição abrupta entre os quadrados de cores diferentes que formam esse código. E assim, apresentando bons resultados nos quais todas as imagens utilizadas acima de 240 pixels puderam ser lidas corretamente com um aparelho celular após a sua binarização pelo algoritmo proposto.**

***Palavras-Chave*- Binarização de Imagem, Detecção de borda, Iluminação não linear,Imagem QR Code**

## BINARIZATION ALGORITHM FOR QR CODE IMAGE WITH NON-UNIFORM ILLUMINATION

***Abstract* - The QR Code is an example of 2D barcode widely used due to its practicality. The low ambient light and/or the shaded image create different tonality in the pixels that forms the squares in this type of code, making it difficult to decode. With that in mind, this work proposes a new algorithm for QR Code binarization, based on the transition between the different colors, in the way that possibility the recognition of the lines that delimit the set of squares of opposite colors. And so, presenting good results in which all images used above 240 pixels could be read correctly with a mobile device after their binarization by the proposed algorithm.**

***Keywords* - Edge Detection, Image Binarization, QR Code Image, Uneven Illumination**

## I. INTRODUÇÃO

O *Quick Response Code* (QR Code) foi desenvolvido pela empresa Denso-Wave Incorporated com o intuito de facilitar a catalogação de componentes fornecidos pela companhia [1]. Porém, o código demonstrou-se ser rápido e prático na sua forma de leitura através de dispositivos móveis e por esse motivo, foi mundialmente difundido na área de *marketing* para compartilhar informações em forma de *links* ou em textos.

Com a ampla gama de aplicações dos códigos QR, notou-se problemas recorrentes nas suas decodificações, pois durante a leitura do código, é bastante comum a não-uniformidade na iluminação causada por sombras ou excesso de luz na imagem, conforme mostrado pelos autores Zhou et al. [2]. Desse modo, fez-se necessário o estudo de técnicas de segmentação que consistem na conversão de uma imagem colorida em uma imagem somente com pixels pretos e brancos, voltadas para as imagens QR Code.

Entretanto, a leitura de códigos QR sob iluminação não-uniforme dificulta a decodificação deles, já que algumas regiões da imagem ficam mais escuras que outras, fazendo com que os quadrados de mesma cor possuam diferentes tonalidades e dessa maneira, impossibilita a escolha de somente um limiar que consiga binarizar a imagem com excelência. Este tipo de problema é minimizado, ou até mesmo contornado, através do uso de métodos de limiarizações locais.

Otsu [3] desenvolveu um algoritmo eficiente em imagens com iluminação uniforme, o qual possui como base a divisão dos pixels da imagem em dois grupos, utilizando-se de um limiar $k$ contido no intervalo de $[0, L-1]$, em que $L-1$ representa o maior nível de cinza de uma imagem digital. Desse modo, para cada valor de intensidade $k$, calcula-se a variância entre os pixels dos grupos, no qual a medida de $k^*$ que possuir a maior variância, será o limiar escolhido.

Xiong e outros [4] criaram um método de binarização que contorna o problema da não-uniformidade da iluminação, no qual a imagem de QR Code é dividida recursivamente em imagens menores até que cada imagem gerada tenha histograma com apenas 2 máximos e essas imagens resultantes possibilitam a aplicação do método de Otsu.

Zhang et al. [5] apresentaram um trabalho em que processaram sub-blocos da imagem para determinar uma estimativa dos níveis de cinza de cada bloco e o algoritmo cria uma matriz de fundos dos níveis de cinza, em que cada valor de intensidade é determinado dependendo do valor da variância na vizinhança. Caso a variância seja maior que $T$ (parâmetro escolhido pelo usuário), a intensidade do pixel será o resultado da interpolação bicúbica naquela posição; caso contrário, o valor de intensidade será a média dos 4 vértices que delimitam a

*alailtonjunior@ufu.br

janela. Por fim, subtrai-se a imagem original com matriz criada anteriormente e por conseguinte, a iluminação não-uniforme é corrigida, viabilizando a aplicação do método de Otsu.

Bernsen [6] propôs um algoritmo que binarizasse uma imagem de iluminação não-uniforme, o qual consiste em definir uma janela de tamanho fixo $N \times N$ e o valor do limiar para o pixel central dessa janela é determinado através da média aritmética entre a menor e a maior intensidade de nível de cinza da vizinhança limitada por essa janela. Para que a imagem seja completamente binarizada, a janela citada anteriormente deve percorrer toda a imagem e $N$ é menor que as dimensões da imagem.

Zhang e Yang [7] apresentaram um método aprimorado de Sauvola, em que, primeiramente, calcula-se a integral da imagem; em seguida, estima-se a média e o desvio padrão a partir dessa integral; e por último, o limiar é determinado com o auxílio do desvio local médio em torno de cada pixel. Com este procedimento, os autores conseguiram reduzir o tempo de processamento do método de Sauvola [8].

Este artigo propõe uma nova técnica de binarização de QR Codes com iluminação não-uniforme, onde primeiramente, calcula-se a matriz gradiente da imagem e os limiares de Otsu em sub-regiões da figura e em seguida, considera-se que as bordas dos quadrados que formam os QR Codes estão posicionadas justamente aonde acontecem os picos do gradiente. Desse modo, identifica-se as linhas das bordas utilizando os limiares para selecionar os valores altos no gradiente, e por fim, binariza-se a imagem QR alternando a cor do módulo entre as transições das bordas.

O algoritmo desenvolvido neste trabalho, que será apresentado na Seção III, demonstrou bons resultados onde foi possível binarizar as imagens QR Code contendo iluminação não uniforme e lê-las sem nenhum erro nas resoluções acima de 240 pixels.

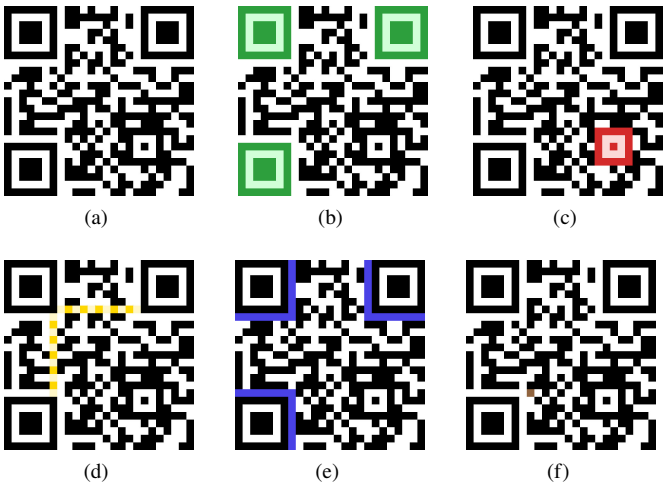
## II. FUNDAMENTOS TEÓRICOS

### A. *QR Code*

O QR Code é um código bidimensional de rápida leitura e existem quatros formatos de dados que este código suporta: *numérico*, *alfanumérico*, *binário* e *kanji*. A *versão* é definida como o tamanho de um QR Code e é diretamente proporcional a quantidade de dados a serem armazenados. A menor unidade de informação desse tipo de código chama-se *módulo* que são os quadrados nas cores branca ou preta que formam a imagem e o número total de módulos é igual a $17+4\times$*Versão*, sendo que o maior tamanho é a Versão 40, contendo 177 módulos e podendo armazenar até 7049 caracteres numéricos ou 4296 alfanuméricos, conforme descrito em [9].

A Figura 1 mostra os elementos que compõem um QR Code, onde na Figura 1(b) estão destacados os *finder patterns* em verde que são os maiores quadrados e permitem identificar se a imagem está rotacionada ou não, devido as suas disposições. O *alignment pattern* é utilizado para corrigir distorções geométricas na imagem e são representados na Figura 1(c) por módulos de cor vermelha. Os *timing patterns* são duas linhas que possuem módulos brancos e pretos dispostos de forma alternada e são exibidos em amarelo na Figura 1(d) e têm como finalidade, determinar deformações dos módulos centrais (região de dados) quando a imagem está distorcida.

Figura 1: Elementos que compõem uma imagem QR Code, onde (a) imagem original, (b) *finder patterns*, (c) *alignment pattern*, (d) *timing patterns*, (e) *separators* e (f) *dark module*.




Para evitar que os *finder patterns* se misturem com os outros módulos do código, são utilizados os *separators* que são regiões brancas ao redor dos *finder patterns* e são realçados na Figura 1(e) pela cor azul. Por último, o *dark module*, sobressaído em marrom na Figura 1(f), é um módulo preto e tem como função delimitar a área encarregada das informações do formato da imagem QR.

### B. *Algoritmo de Otsu*

O Algoritmo de Otsu realiza uma limiarização global baseada na bimodalidade do histograma da imagem. A ideia principal do algoritmo é determinar todos os valores possíveis para o limiar e a partir desses valores, encontrar aquele que após dividir os pixeis em duas categorias, branco ou preto, resultará na maior variância estatística entre os dois grupos, o detalhamento matemático do método pode ser encontrado em [3].

### C. *Gradiente Da Imagem*

O gradiente da imagem é uma ferramenta utilizada em processamento de imagem para determinar a taxa de variação de uma função $f$ e é definido pela Equação (1)

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ \\ g_y \end{bmatrix} \tag{1}$$

onde $g_x$ e $g_y$ são as componentes do gradiente nas direções de $x$ e $y$, respectivamente. Neste artigo, essas componentes são calculadas por intermédio dos *Operadores de Sobel*, por apresentarem uma maior supressão de ruído. As duas máscaras de convolução utilizadas nos cálculo de $g_x$ e $g_y$ são mostradas na Figura 2.

Figura 2: *Operadores de Sobel* utilizados para calcular as componentes do gradiente na direção de: (a) $x$ e (b) $y$.

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| −1 | −2 | −1 |

(a)

| | | |
|---|---|---|
| 1 | 0 | −1 |
| 2 | 0 | −2 |
| 1 | 0 | −1 |

(b)

### D. Processamento Morfológico de Imagens

Os processamentos morfológicos baseam-se na teoria do conjuntos, buscando similaridades de intensidade e de formatos geométricos entre os pixels de uma imagem, fazendo com esses métodos tornam-se bastantes úteis na manipulação de imagens segmentadas.

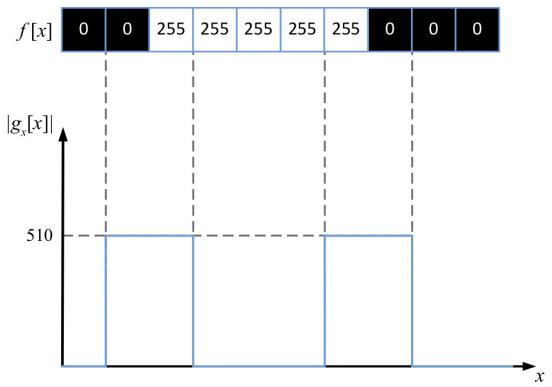
As três operações morfológicas que foram aplicadas neste artigo, as quais estão mais detalhadas em Gonzalez e Woods [10], são: a *erosão*, a *dilatação* e o *afinamento*.

## III. BINARIZAÇÃO DE IMAGEM QR CODE COM ILUMINAÇÃO NÃO UNIFORME

O método para binarizar imagens de QR Codes proposto neste trabalho independe se a iluminação é uniforme ou não, pois baseia-se nas transições abruptas de nível de cinza entre módulos brancos e pretos, as quais são identificadas através do limiar de Otsu que indica quando há um valor muito grande no gradiente da imagem.

A Figura 3 descreve o princípio de como identificar os pixels de borda, na qual a função $f[x]$ exemplifica uma linha de uma imagem e $g_x[x]$ representa o gradiente na direção de $x$ dessa função.

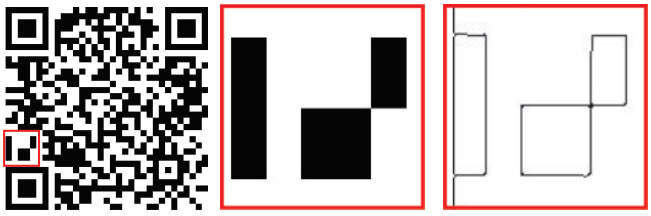
Figura 3: Representação da funcionalidade da ferramenta gradiente.




Como exemplificado pela Figura 3, as bordas podem ser localizadas quando houverem transições de cores entre 2 pixels adjacentes, ou seja, no momento em que a cor entre pixels vizinhos mudar de branca para preta ou de preta para branca. Para a Figura 3, essas transições são identificadas quando o valor de $|g_x[x]|$ passa de zero para 510. Se não houver mudança de cor entre 2 pixels contíguos, o valor do gradiente é igual a zero.

Na Figura 4 é possível verificar o resultado do módulo do gradiente de uma região de um QR Code.

Figura 4: (a) Exemplo de um QR Code com uma região delimitada por um quadrado vermelho, (b) imagem ampliada dessa região e (c) resultado da operação do gradiente para a região destacada.



(a) (b) (c)

Na Figura 4(c) são destacadas, por pontos pretos, as posições onde $|g_x[x]|$ possuem máximos locais, ou seja, esses pontos são as localizações dos pixels que formam as bordas dos módulos do QR Code. A união dos pontos destacados pela Figura 5(c) formam 3 polígonos fechados e desse modo, a cor de dentro deles é sempre diferente da cor do lado de fora.

Em seguida, utilizando a imagem resultante do gradiente, percorremos a matriz gerada partindo do pixel central do primeiro módulo da imagem como referência até o pixel central de um outro módulo qualquer, durante o percurso, foi contabilizado a quantidade de bordas que foram atravessadas, e assim, se esse número for par, o módulo será da cor preta e caso contrário, será da cor branca.

As etapas do algoritmo proposto neste trabalho são descritas abaixo:

No Algoritmo 1 as Linhas 7 e 8, dividem a imagem original em sub-regiões de tamanho $N \times N$ e em cada uma das novas regiões aplica-se o método de Otsu para determinar o limiar local, dessa forma, a matriz $limiar$ de tamanho $N \times N$ contém os valores calculados por Otsu de cada região em uma escala normalizada de 0 a 1.

Nas linhas subsequentes, entre 11 e 18, ocorre o ajuste dos valores obtidos anteriormente, essa modificação é necessária, pois em imagens com intensidade de pixels elevadas, o valor do limiar também será elevado, porém os valores do gradiente que são afetados apenas pelas variações entre as intensidades dos pixels continuam os mesmos. Logo, quando a grandeza do limiar é maior que 0.5, ou seja, a metade da intensidade máxima, utiliza-se o valor de $1 - limiar$, desse modo o problema é amenizado.

Na etapa descrita na Linha 21, os valores de intensidade do gradiente são selecionados com base na matriz dos limiares de Otsu. De modo que, se a intensidade do pixel na matriz gradiente for maior que a intensidade do limiar local, este pixel será escolhido como um pixel de borda. Um exemplo de uma matriz gerada por essa operação é mostrada na Figura 5(a)

Depois de determinar as linhas que dividem os módulos de cores diferentes, é necessário descobrir a versão do QR Code, ou seja, a quantidade de módulos presentes na figura. Para isso, na Linha 22 do algoritmo, foi utilizados os *finders patterns*, já que independente da resolução da imagem, esses quadrados sempre estarão localizados nas extremidades e terão

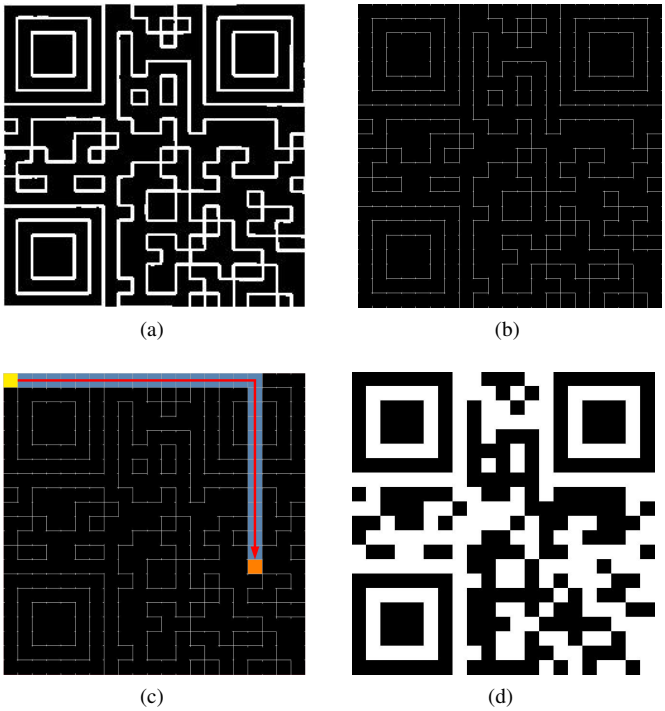
dimensões iguais a $7 \times 7$ módulos. O cálculo é repetido para os outros 2 *patterns* e um valor médio mais preciso do número de módulo ($Q_m$) é estimado.

**Algoritmo 1:** Binarização de Imagens QR Code.

```
1  var
2      N, i, j, M, f, g, x_i, x_f, y_i, Q_m, Q_f: inteiros
3      bord, g_x, g_y: matriz[1..m,1..n] de reais
4      limiar: matriz[1..N,1..N] de reais
5      Imagem: matriz[1..Q_m,1..Q_m] de inteiros
6  início
7      Divide-se a imagem original em N × N regiões;
8      Cria-se a matriz de limiares (limiar);
9      m ← número de linhas da imagem;
10     n ← número de colunas da imagem;
11     para i de 0 até m faça
12         para j de 0 até n faça
13             se limiar[i, j] > 0.5 então
14                 limiar[i, j] ← 1 − limiar[i, j];
15             fim
16             limiar[i, j] ← limiar[i, j] ∗ 255;
17         fim
18     fim
19
20     Calcula-se os gradientes nas direções g_x e g_y;
21     Gera-se a matriz das bordas (bord);
22     Calcula-se a quantidade de módulos Q_m;
23     Afina-se as bordas;
24     Retira-se o serrilhamento das linhas;
25     M ← número de linhas da matriz bord;
26     b ← M/Q_m
27     x_i ← b/2
28     y_i ← x_i
29
30     para f de 0 até Q_m − 1 faça
31         para g de 0 até Q_m − 1 faça
32             Q_f ← 0
33             y_f ← y_i + b ∗ g
34             para i de y_i até y_f faça
35                 se bord[x_i, i] = 255 então
36                     Q_f ← Q_f + 1
37                 fim
38             fim
39             x_f ← x_i + b ∗ f
40             para j ← x_i até x_f faça
41                 se bord[j, i] = 255 então
42                     Q_f ← Q_f + 1
43                 fim
44             fim
45             se Q_f mod 2 = 0 então
46                 Imagem[f, g] ← 0
47             senão
48                 Imagem[f, g] ← 255
49             fim
50         fim
51     fim
52 fim
```

Após descobrir a versão do QR, nas Linhas 23 e 24 foram utilizadas as transformações morfológicas com o intuito de afinar as bordas de modo que a espessura das mesmas seja de apenas um pixel e, através da quantidade de módulos $Q_m$ e o tamanho total da figura em pixels, é possível estimar as coordenadas corretas de cada uma das bordas. Assim, percorre-se a matriz da Figura 5(a) nas posições corretas de cara linha e verifica se há um pixel branco na região, se sim, o mesmo é realocado na posição correto. o resultado da operação é mostrado na Figura 5(b) e é utilizado para corrigir o serrilhamento.

Figura 5: Figuras adquiridas durante as etapas do algoritmo.



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

Na etapa final da binarização, indicada nas Linhas 25 a 52, através da linhas da matriz *bord* da Figura 5(b), percorre-se os quadrados da imagem até o modulo desejado, sendo eles pintados da cor branca ou preta com base na quantidade de fronteiras que foram atravessadas, como mostrado na Figura 5(c). A função *mod* utilizada representa o operador módulo que retorna o resto da divisão inteira de $Q_f$ por 2.

Em suma, o algoritmo se torna eficaz devido a facilidade de se determinar as linhas que dividem os conjuntos módulos de cores diferentes, pois na região de transição entre esses conjuntos, os valores da matriz gradiente possuem picos de intensidade e assim, a utilização da limiarização computada na imagem de níveis de cinza, é mais eficiente para detectar as linhas de fronteira ao em vez de classificar os pixels entre branco ou preto.

## IV. RESULTADOS EXPERIMENTAIS

O método proposto neste trabalho foi escrito em Octave 5.2.0 e as especificações do computador utilizado são: Intel Core i7-8750H, 16 GB de memória RAM do tipo DDR4 de 2.666 MHz e Windows 10 Home de 64 bits.

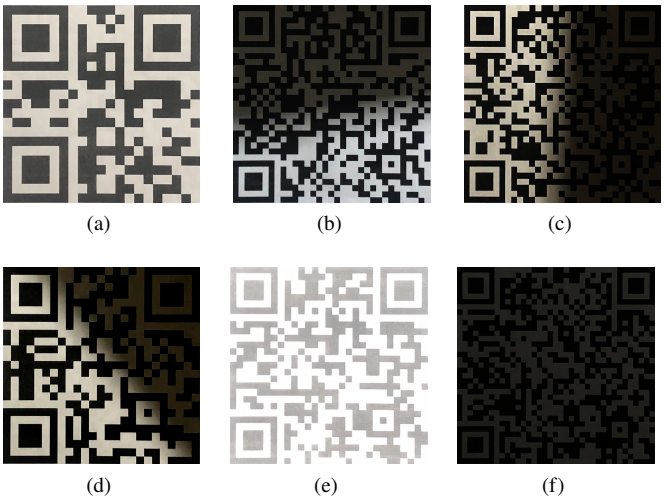
As imagens testes foram coletadas com auxílio de uma câ-

mera de 12 MP de um iPhone 8. A principal justificativa da utilização desse tipo de dispositivo na aquisição das imagens, em vez de uma câmera profissional ou qualquer outro tipo de equipamento, é que a maioria das pessoas realizarão as leituras de QR Codes com aparelhos celulares. Logo, o intuito deste trabalho é chegar o mais próximo de situações cotidianas.

Para os testes foram experimentadas 6 imagens QR Codes de versões diferentes e cada QR Code foi exposto a um dos seguintes padrões de iluminação: *sombreamento vertical*, *sombreamento horizontal*, *sombreamento diagonal*, *iluminação saturada* e *pouca iluminação*, onde cada padrão de iluminação é exemplificado na Figura 6.

Através do programa GIMP 2.10.18, as imagens foram recortadas e redimensionadas para as seguintes resoluções: $128\times128$, $184\times184$, $240\times240$, $480\times480$ e $720\times720$ pixels, a fim de descobrir se o algoritmo proposto consegue trabalhar com imagens de resoluções menores, totalizando assim, 180 imagens utilizadas nos testes.

Figura 6: Padrões de iluminação utilizados nos testes, onde: (a) iluminação uniforme, (b) sombreamento horizontal, (c) sombreamento vertical, (d) sombreamento diagonal, (e) iluminação saturada e (f) pouco iluminação.



(a) (b) (c)

(d) (e) (f)

As Tabelas 1 a 5 mostram a porcentagem de módulos que foram binarizados incorretamente pela técnica proposta neste artigo, para diferentes resoluções e versões. Na coleta dos dados, foi utilizado o número $N$ de 20 divisões da imagem original.

Tabela 1: Percentual de módulos errados no padrão Sombreamento Horizontal

| ***Versão*** | **$128\times128$** | **$184\times184$** | **$240\times240$** | **$480\times480$** | **$720\times720$** |
|---|---|---|---|---|---|
| 1 | 16,33% | 0,00% | 0,00% | 0,00% | 0,00% |
| 2 | 22,56% | 0,16% | 0,32% | 0,00% | 0,00% |
| 3 | 36,39% | 1,19% | 0,00% | 0,00% | 0,00% |
| 4 | 41,32% | 33,47% | 0,18% | 0,00% | 0,00% |
| 5 | 43,24% | 41,32% | 0,80% | 0,00% | 0,00% |

Tabela 2: Percentual de módulos errados no padrão Sombreamento Vertical

| ***Versão*** | **$128\times128$** | **$184\times184$** | **$240\times240$** | **$480\times480$** | **$720\times720$** |
|---|---|---|---|---|---|
| 1 | 16,33% | 0,00% | 0,00% | 0,00% | 0,00% |
| 2 | 22,56% | 0,16% | 0,32% | 0,00% | 0,00% |
| 3 | 36,39% | 1,19% | 0,00% | 0,00% | 0,00% |
| 4 | 41,32% | 33,47% | 0,18% | 0,00% | 0,00% |
| 5 | 43,24% | 41,34% | 0,80% | 0,00% | 0,00% |

Tabela 3: Percentual de módulos errados no padrão Sombreamento Diagonal

| ***Versão*** | **$128\times128$** | **$184\times184$** | **$240\times240$** | **$480\times480$** | **$720\times720$** |
|---|---|---|---|---|---|
| 1 | 15,33% | 0,00% | 0,00% | 0,00% | 0,00% |
| 2 | 21,92% | 0,00% | 0,00% | 0,00% | 0,00% |
| 3 | 38,29% | 1,31% | 0,00% | 0,00% | 0,00% |
| 4 | 39,49% | 26,63% | 0,00% | 0,00% | 0,00% |
| 5 | 43,10% | 45,00% | 0,29% | 0,00% | 0,00% |

Tabela 4: Percentual de módulos errados no padrão Iluminação Saturada

| ***Versão*** | **$128\times128$** | **$184\times184$** | **$240\times240$** | **$480\times480$** | **$720\times720$** |
|---|---|---|---|---|---|
| 1 | 14,06% | 0,00% | 0,91% | 0,00% | 0,00% |
| 2 | 19,20% | 0,80% | 2,08% | 0,00% | 0,00% |
| 3 | 32,34% | 3,69% | 0,59% | 0,00% | 0,00% |
| 4 | 37,51% | 18,92% | 0,18% | 0,00% | 0,05% |
| 5 | 41,12% | 39,81% | 4,97% | 0,00% | 0,00% |

Tabela 5: Percentual de módulos errados no padrão Pouca Iluminação

| ***Versão*** | **$128\times128$** | **$184\times184$** | **$240\times240$** | **$480\times480$** | **$720\times720$** |
|---|---|---|---|---|---|
| 1 | 14,06% | 0,00% | 0,00% | 0,00% | 0,00% |
| 2 | 13,44% | 0,00% | 0,00% | 0,00% | 0,00% |
| 3 | 30,92% | 0,00% | 0,00% | 0,00% | 0,00% |
| 4 | 40,59% | 22,64% | 0,00% | 0,00% | 0,00% |
| 5 | 41,56% | 40,54% | 0,00% | 0,00% | 0,00% |

Observa-se nas tabelas anteriores que para as resoluções maiores ou iguais a $240\times240$, a porcentagem de erro não teve valores significantes, e por isso foi possível a sua leitura por um dispositivo celular, pois a quantidade de módulos binarizados incorretamente não atingiu o limite dos códigos corretores de erros (15% para o nível M).

Percebe-se também, que as porcentagens de erro da resolução $128\times128$ aumentou drasticamente com o decorrer das versões, ou seja, em versões maiores, a quantidade de pixels em cada módulo é menor e consequentemente o erro do algoritmo proposto aumentou. Essa característica também pode ser observada na resolução $184\times184$, em que, para a *Versão* 1, o erro foi nulo, mas para a *Versão* 5, essa porcentagem atingiu a média de $41,60\%$ de módulos incorretos.

Pela análise dos resultados, conclui-se que cada versão do QR Code possui uma resolução correspondente, na qual o método possui erro nulo. Desse modo, baseando-se nos resultados obtidos, notamos que nas imagem binarizadas corretamente, tiveram pelo menos $7\times7$ pixels em cada módulo. Essa afirmação pode ser comprovada observando-se as figuras de

resolução $480 \times 480$, que estão isentas de erros em todas as versões estudadas.

Em uma das etapas do algoritmo, a imagem QR Code é dividida em sub-regiões de tamanho $N \times N$ para calcular o limiar local. Desse modo, a fim de obter-se um valor de $N$ que diminua a ocorrência de módulos errados e também o esforço computacional, foram feitos testes considerando as 5 versões utilizadas na resolução $480 \times 480$, pois ela obteve erro nulo, e variamos a quantia $N$ entre 1 a 10 para todos os padrões de iluminação, os resultados alcançados são mostrados na Tabela 6.

Tabela 6: Porcentagem de módulos errados por sub divisão da imagem

| *Número de divisões* (N) | *Somb. Horizontal* | *Somb. Vertical* | *Somb. Diagonal* | *Iluminação Saturada* | *Pouca Iluminação* | Tempo gasto (s) |
|---|---|---|---|---|---|---|
| 1 | 30,49% | 27,59% | 32,30% | 0,07% | 0,00% | 12,25 |
| 2 | 9,17% | 4,07% | 11,22% | 0,07% | 0,00% | 12,80 |
| 3 | 9,33% | 7,70% | 10,82% | 0,00% | 0,00% | 13,30 |
| 4 | 6,95% | 5,87% | 7,28% | 0,00% | 0,00% | 14,41 |
| 5 | 6,27% | 5,54% | 5,90% | 0,00% | 0,00% | 14,97 |
| 6 | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% | 16,03 |
| 7 | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% | 17,46 |
| 8 | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% | 17,97 |
| 9 | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% | 18,34 |
| 10 | 0,00% | 0,00% | 0,00% | 0,00% | 0,00% | 19,67 |

Percebe-se que para os padrões de *Iluminação Saturada* e *Pouca Iluminação*, os erros foram nulos para todos os valores de $N$. Isso ocorre porque nesses padrões, os módulos de mesma cor possuem valores de intensidade muito semelhantes e mesmo que a intensidade média seja muito alta ou baixa, o limiar local será sempre o mesmo em todas as divisões.

Nos outros padrões remanescentes, os módulos de uma mesma cor na imagem não possuem valores de intensidade próximos e por isso, os limiares de cada região possuem valores diferentes. Desse modo, faz-se necessário identificar se o limiar local diminuiu ou aumentou para que não haja erros quando for compara-los com a matriz gradiente, pois se a intensidade média dos pixels em uma determinada região for menor que em outra, os valores de intensidade no gradiente também serão menores.

Por fim, com base nos dados da Tabela 6, concluí-se que para as imagens testadas, o valor de $N$ que possui erro nulo em todos os padrões de iluminação tem que ser necessariamente maior ou igual a 6.

## V. CONCLUSÃO

Percebe-se pelos testes realizados que o novo método para binarizar imagens de QR Codes com iluminação não-uniforme proposto por este artigo, obtém bons resultados, pois para algumas resoluções, a taxa de erro se aproxima de zero ou até mesmo é nula para as maiores imagens testadas.

Também fica evidenciado na seção de resultados que para o método proposto funcionar corretamente, a resolução mínima deve ser de pelo menos de $480 \times 480$ pixeis, o que é uma resolução baixa para os padrões atuais de câmeras de celulares e consequentemente, resultará em um menor tempo de processamento do que para imagens de dimensões superiores, além de requisitar uma câmera de resolução inferior ao *High Definition* (HD), que são $1280 \times 720$ pixeis.

Importante salientar que para o funcionamento correto do algoritmo proposto neste artigo, é necessário que o QR Code não possua deformações geométricas geradas por perspectiva, já que uma das etapas desse algoritmo consiste em percorrer a imagem nas direções horizontal e vertical e casos as bordas dos módulos tenham alguma angulação, isso impossibilitará essa ação.

## REFERÊNCIAS


[1] Denso Wave (2020). *What is a QR Code?*. Acedido em 20 de Abril de 2022, em: `https://www.denso-wave.com/en/adcd/fundamental/2dcode/qrc/`.

[2] J. Zhou and Y. Liu and P. Li, "Research on Binarization of QR Code Image", *International Conference on Multimedia Technology*, pp. 1-4, 2010.

[3] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp 62-66, 1979.

[4] Z. Xiong and H. Cuiqun and L. Guodong and L. Zhijun, "A binarization method of quick response code image", *2nd International Conference on Signal Processing Systems*, vol. 3, pp. 317-320, 2010.

[5] Y. Zhang and T. Gao and D. Li and H. Lin, "An improved binarization algorithm of QR code image", *2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pp. 2376-2379, 2012.

[6] J. Bernsen, "Dynamic Thresholding of Gray Level Image", *Proceedings of International Conference on Pattern Recognition*, pp. 1251-255, 1986.

[7] W. Zhang, T. Yang, "An Improved Algorithm for QR Code Image Binarization", *International Conference on Virtual Reality and Visualization*, pp. 154-159, 2014.

[8] J. Sauvola, M. Pietikäinen, "Adaptive document image binarization", *Pattern Recognition*, vol. 33, pp. 225-236, 2000.

[9] Thonky (2015). *QR Code Tutorial*. Acedido em 20 de Abril de 2022, em: `https://www.thonky.com/qr-code-tutorial/`.

[10] R. C. Gonzalez, R. E. Woods, *Digital Image Processing*, Prentice-Hall, 3ª Edição, USA, 2006.