CEEL - ISSN 2596-2221
Universidade Federal de Uberlândia
07 a 11 de dezembro de 2020

ALGORITMO PARA DETERMINAR OS VÉRTICES DE QR CODES COM PERSPECTIVA

Heitor Eugênio Gonçalves*[1], Luciano Xavier Medeiros[1] e Alexandre Coutinho Mateus[1]

[1]FEELT - Universidade Federal de Uberlândia

***Resumo* - As distorções ocasionadas devido à perspectiva da imagem podem dificultar, ou até mesmo impedir, que QR Codes sejam decodificados. As coordenadas dos vértices de um QR Code são informações importantíssimas quando técnicas de remoção de deformações geométricas são utilizadas. Pensando nisso, esse artigo apresentará um algoritmo capaz de identificar os vértices de QR Codes por intermédio das equações de retas das bordas do mesmo, ainda que as imagens tenham perspectivas.**

***Palavras-Chave*- distorção geométrica, localização de vértices, perspectiva, processamento de imagem, QR Code.**

ALGORITHM FOR DETERMINING THE VERTICES OF QR CODES WITH PERSPECTIVE

***Abstract* - Distortions caused by image perspective can make it difficult, or even prevent, QR Codes from being decoded. The vertices coordinates of a QR Code are very important information when geometric deformation removal techniques are used. With this in mind, this article will present an algorithm capable of identifying the vertices of QR Codes through the equations of lines of the edges of the same, even if the images have perspectives.**

***Keywords* - geometric distortion, image processing, perspective, vertex location, QR Code.**

I. INTRODUÇÃO

O *Quick Response Code*, mais conhecido pela abreviatura QR Code, é uma importante ferramenta de armazenamento de informação, sendo utlizados por exemplo, para realizar pagamentos digitais, divulgação de *links* de internet, identificação de pessoas e produtos, dentre outras aplicações.

O fato de o QR Code poder ser decodificado com rapidez por meio de câmeras de celulares, permite que esse tipo de código bidimensional seja extremente acessível. As fotografias dos códigos QR podem apresentar fatores capazes de dificultar a decodificação, como iluminação irregular, baixa resolução e distorções geométricas.

Com o intuito de permitir a leitura de QR Codes, mesmo em fotos de baixa resolução, em Kato et al. [1] apresenta um método capaz de criar uma imagem de alta resolução a partir da análise de múltiplas fotos de baixa resolução, obtidas a partir de pequenos deslocamentos da câmera. Para isso, esse método usa a maximização da função de verossimilhança, que estima valores mais prováveis para os parâmetros das amostras observadas.

Para corrigir fotos de QR Codes feitas em ambientes com iluminação irregular, destaca-se o trabalho desenvolvido em Berrios, Medeiros e Mateus [2]. São desenvolvidas 3 técnicas de binarização para imagens de QR Codes, onde o método que possui os melhores resultados divide a imagem em n regiões, em seguida realiza a equalização por histograma de cada região separadamente e, por último, binariza a imagem a partir do nível de cinza médio de cada área.

No âmbito de distorções de perspectiva, é possível encontrar trabalhos que utilizam a transformada de Hough para determinar o polígono que envolve o QR Code, como os trabalhos de Wei e de Zhou [3, 4]. A maior vantagem da técnica de Hough é a capacidade de detectar grupos de pixels que pertencem a uma linha reta mesmo que esteja quebrada e/ou com ruídos. No entanto, essa transformada apresenta um elevado tempo de processamento, além de necessitar que a imagem tenha as bordas (contornos) identificadas.

Em Tribak e Zaz [5], aplica-se a técnica dos 7 momentos invariantes de Hu para encontrar os *finder patterns* dos QR Codes, já que esses momentos não são modificados com mudança na escala, rotação e/ou translação da imagem e assim, podem ser usados para determinar distorções apresentadas por um QR Code.

Suran [6] desenvolveu um trabalho que, primeiramente, usa o método de Harris para detectar quinas e, em seguida, aplica o algoritmo envoltório convexo para determinar o contorno externo do QR Code. Assim, com essas posições, aplica-se uma transformação inversa de perspectiva que mapeia o quadrilátero identificado e, com o uso de mudanças de coordenadas, torna o código QR quadrado. Entretanto, as imagens testadas nesse artigo são limitadas a quadriláteros de lados opostos paralelos..

Este artigo propõe um algoritmo capaz de identificar os vértices do QR Code, bem como os seus lados. A obtenção dessas posições é necessária para a implementação de métodos que removem distorções desses códigos bidimensionais.

*heitor.goncalves@ufu.br

Para realizar a localização dos vértices de um QR Code, o algoritmo proposto neste artigo utiliza, no máximo, 4 equações de retas. Este método otimiza o processo de obtenção dessas coordenadas por mapear somente os 4 lados do QR Codes. Isso não ocorre nos trabalhos anteriormente apresentados, que necessitam do mapeamento de toda a região do QR Code ou, pelo menos, das 3 áreas dos *finder patterns* e do *aligment pattern*.

II. FUNDAMENTOS TEÓRICOS

A. Estrutura do QR Code

O menor elemento de informação presente em um QR Code é o *módulo*, que apresenta o formato quadrado. A quantidade desses elementos depende da quantidade de caracteres que serão codificados.

A versão do QR Code depende da quantidade de módulos que o constitui, onde a Versão 1 têm 21×21 módulos, a Versão 2 possui dimensão de 25×25, aumentando sempre de 4×4 e limitando-se a Versão 40 que possui 177×177 módulos, como pode ser consultado em [7, 8].

A Figura 1(a) mostra um QR Code de Versão 1, com 21×21 módulos de dimensão.

Figura 1: (a) Exemplo de um QR Code de Versão 1, (b) *finder patterns* destacados em verde e (c) *separators* coloridos em azul.

(a) (b) (c)

Um conjunto de módulos que é de extrema importância para determinar as deformações geométricas de um QR Code é o *finder pattern*. Cada QR Code possui 3 desses conjuntos, como exemplificado na Figura 1(b) pela cor verde. Independente da versão do QR Code, os *finder patterns* são quadrados de tamanho 7×7 módulos e com um quadrado menor de 3×3 módulos em seu interior.

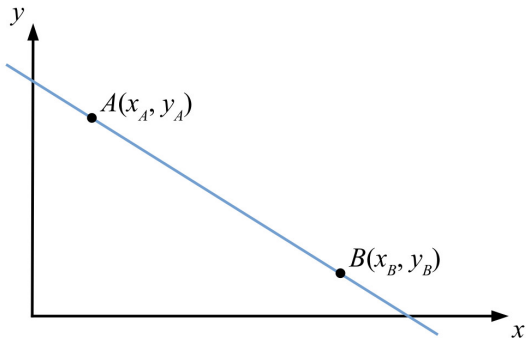
Para desassociar os *finder patterns* dos restantes módulos do código, são utilizados os *separators* que são realçados em azul na Figura 1(c) e esse padrão consiste de linhas brancas em torno dos *finder patterns*.

Tanto os *finder patterns* quanto os *separators* serão utilizados pelo método desenvolvido neste trabalho para determinar retas que delimitam os QR Codes e dessa forma, encontrar os vértices desses códigos, como será apresentado na Seção III.

B. Equação de reta

Considere que os pontos $A(x_A, y_A)$ e $B(x_B, y_B)$ pertencentes ao plano $z = 0$, conforme mostrado pela Figura 2.

Figura 2: Representação de uma reta no $\mathbb{R}^2$.



É possível traçar uma reta que passa pelos pontos A e B da Figura 2 e de forma genérica, a equação dessa reta pode ser obtida a partir de [9, 10]:

$$y = m\,x + n \tag{1}$$

O coeficiente angular da Equação (1) é calculado por

$$m = \frac{y_B - y_A}{x_B - x_A} \tag{2}$$

enquanto o coeficiente linear é igual a

$$n = y_A - m\,x_A \tag{3}$$

Foi utilizado as coordenadas do ponto A na Equação (3) para determinação do coeficiente linear, mas também poderia ter sido usadas as coordenadas do ponto B, já que ambos os pontos pertencem a mesma reta.

III. ALGORITMO PROPOSTO

O método proposto neste artigo é dividido em 4 etapas: *Binarização da imagem*, *Varredura da imagem*, *Análise dos vértices* e *Localizção dos vértices*. As etapas do algoritmo serão explanadas a seguir.

A. Binarização da imagem

Primeiramente, as imagens processadas pelo algoritmo proposto são necessariamente pretas e brancas. Logo, todas as fotografias são binarizadas por intermédio da função *im2bw*( ), contida no pacote *image* 2.12.0 do GNU Octave de versão 5.2.0.

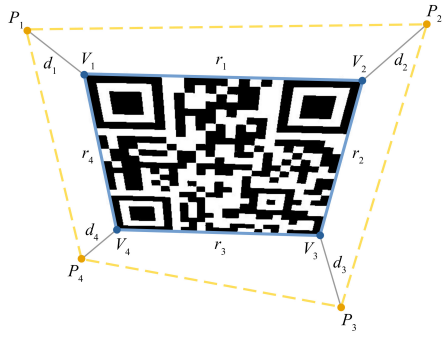
B. Varredura da Imagem

Realiza-se uma varredura da imagem por meio de 4 conjuntos de laços aninhados que a percorrem verticalmente e horizontalmente. Cada um desses conjuntos armazenará o primeiro pixel de cor preta encontrado por eles. Essa etapa é descrita no Algoritmo 1, onde L e C são os números de linhas e coluna da matriz Img da imagem binarizada. Esse algoritmo tem como saída os vetores A_i, para $i = 1,2,3$ e 4, os quais possuem as coordenadas dos possíveis vértices do QR Code.

Algoritmo 1: Localização dos vértices

```
1  var
2      i, j: inteiros;
3      A1, A2, A3, A4: vetor[1..2] de inteiros;
4      Img: matriz[1..L,1..C] de inteiros;
5  início
6      A[1] ← −1;
7      B[1] ← −1;
8      C[1] ← −1;
9      D[1] ← −1;
10     para i de 1 até L faça
11         para j de 1 até C faça
12             se Img[i,j] = 0 e A1[1] = −1 então
13                 A1[1] ← j;
14                 A1[2] ← i;
15             fim
16         fim
17     fim
18     para i de L até 1 passo −1 faça
19         para j de 1 até C faça
20             se Img[i,j] = 0 e A2[1] = −1 então
21                 A2[1] ← j;
22                 A2[2] ← i;
23             fim
24         fim
25     fim
26     para j de 1 até C faça
27         para i de 1 até L faça
28             se Img[i,j] = 0 e A3[1] = −1 então
29                 A3[1] ← j;
30                 A3[2] ← i;
31             fim
32         fim
33     fim
34     para j de C até 1 passo −1 faça
35         para i de 1 até L faça
36             se Img[i,j] = 0 e A4[1] = −1 então
37                 A4[1] ← j;
38                 A4[2] ← i;
39             fim
40         fim
41     fim
42 fim
```

Figura 3: Convenção das posições do vértices reais e estimados.



C. *Análise dos Pontos*

Nesta etapa, as coordenadas dos pontos coletados pelo Algoritmo 1 são analisadas com o objetivo de rotular esses pontos, obedecendo o padrão da Figura 3.

Para que essa análise seja feita, utiliza-se o ponto central $P_{med}(x_{med}, y_{med})$ do QR Code e suas coordenadas são calculadas pela Equação (4).

$$x_{med} = \frac{1}{2}[\max(x_1,x_2,x_3,x_4) - \min(x_1,x_2,x_3,x_4)] + \min(x_1,x_2,x_3,x_4) \quad (4a)$$

$$y_{med} = \frac{1}{2}[\max(y_1,y_2,y_3,y_4) - \min(y_1,y_2,y_3,y_4)] + \min(y_1,y_2,y_3,y_4) \quad (4b)$$

onde max e min são, respectivamente, os operadores máximo e mínimo.

Em seguida, aplica-se o Algoritmo 2, tendo o ponto P_{med} como referência e assim, verifica-se se os pontos estimados como vértices, P_i, estão na mesma sequência que a mostrada na Figura 3. Nesse Algoritmo, os vetores P_i, para $i = 1,2$ e 4, representam as coordenadas dos vértices do QR Code. Porém, nota-se que nem todos esses vetores podem ser encontrados nesse etapa.

Algoritmo 2: Análise dos vértices

```
1  var
2      A1, A2, A3, A4: vetor[1..2] de inteiros;
3      P1, P2, P4: vetor[1..2] de inteiros;
4  início
5      se A3[1] < ymed então
6          P1[1] ← A3[1];
7          P1[2] ← A3[2];
8      senão
9          P4[1] ← A3[1];
10         P4[2] ← A3[2];
11     fim
12     se A4[1] < ymed então
13         P2[1] ← A4[1];
14         P2[2] ← A4[2];
15     fim
16     se A1[2] < xmed então
17         P1[1] ← A1[1];
18         P1[2] ← A1[2];
19     senão
20         P2[1] ← A1[1];
21         P2[2] ← A1[2];
22     fim
23     se A2[2] < xmed então
24         P4[1] ← A2[1];
25         P4[2] ← A2[2];
26     fim
27 fim
```

D. Localização dos vértices

O Algoritmo 1 não é capaz de localizar a posição de P_3, pois este vértice pode ou não ter um pixel de cor preta em sua extremidade, da mesma forma que ocorre na Figura 1. Porém, destaca-se que a varredura pode encontrar 2 ou 3 vértices, isso porque, além do P_3, é possível que outro vértice também não seja determinado.

Para detectar o vértice P_3 e os outros não localizados pelo Algoritmo 1, deve-se determinar as equações das retas r_1, r_2, r_3 e r_4, as quais podem ser escritas, de forma geral, como r_i: $y = m_i\,x\, +\, n_i$, onde m_i e n_i são, respectivamente, os coeficientes angular e linear da reta r_i, para $i = 1, 2, 3$ e 4.

As retas são obtidas a partir das inclinações das bordas externas dos *finder patterns* onde o vértice é conhecido. Para calcular essa inclinação, utiliza-se o ponto Q_1 localizado na fronteira com o *separator* em cada *finder pattern*, como mostra a Figura 2. Posteriormente, os coeficientes lineares dessas retas são calculados conforme a Equação (3). O Algoritmo 3 exemplifica esse processo, além de encontrar a equação da reta r_1 a partir dos pontos Q_1 e P_1, que estão ilustrados na Figura 4.

Algoritmo 3: Cálculo da equação da reta de r_1

```
1  var
2      i, j, k, u1, v1, aux: inteiros;
3      m1, n1: reais;
4      P1: vetor[1..2] de inteiros;
5      Img: matriz[1..L,1..C] de inteiros;
6  início
7      k ← 1;
8      v1 ← −1;
9      para j de P1[1] até C faça
10         para i de P1[2] até L faça
11             se Img[i,j] = 0 e v1 = −1 então
12                 aux[k] ← i;
13                 se k ≥ 2 e |aux[k] − aux[k − 1]| ≥ 2
                    então
14                     v1 ← j − 2;
15                 fim
16                 k ← k + 1;
17             fim
18         fim
19     fim
20     u1 ← −1;
21     para i de P1[2] até L faça
22         se Img[i,v1] = 0 e u1 = −1 então
23             u1 ← i;
24         fim
25     fim
26     m1 ← (u1 − P1[2])/(v1 − P1[1]);
27     n1 ← v1 − m1 ∗ u1;
28 fim
```

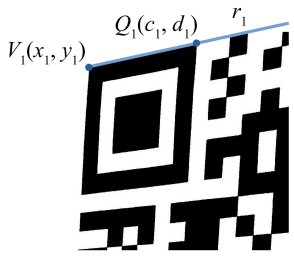
Devido às quinas presentes nas bordas, com o intuito de obter resultados mais satisfatórios, pode-se usar também a quina antecessora à Q_1, o que torna a inclinação da reta r_1 mais próxima de seu valor real. Assim, neste trabalho, optaremos por utilizar a quina precursora à Q_1.

Para determinar as retas r_2, r_3 e r_4, utiliza-se o procedimento semelhante ao feito no Algoritmo 3, calculando os coeficientes angular e linear da reta que passa pela borda do *finder pattern* do lado que se deseja obter a equação. Após concluir esse procedimento, é realizado uma nova varredura com direcionamento definido pelas equações das retas, permitindo encontrar os demais vértices que não foram localizados no escaneamento inicial, com excessão do P_3. Este ponto é determinado pela a intersecção entre as retas r_2 e r_3. Assim, a abscissa de P_3 é dada por

$$x_3 = \frac{n_2 - n_3}{m_3 - m_2} \tag{5}$$

e a ordenada desse ponto pode ser calculada por $y_3 = m_3\,x_3 + n_3$.

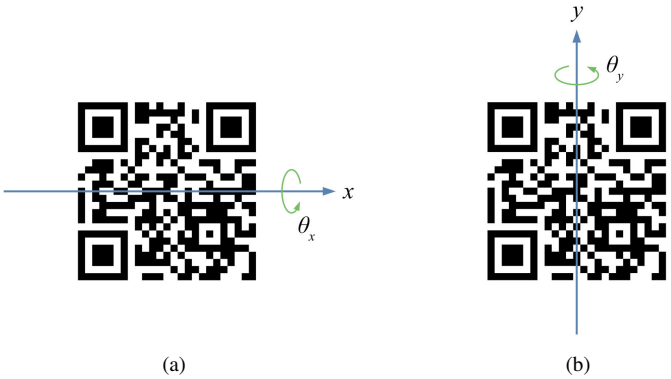
Figura 4: Exemplo da reta r_1 que passa por $V_1(x_1, y_1)$ e $Q_1(u_1, v_1)$.



IV. RESULTADOS

Com o intuito de analisar o funcionamento do algoritmo desenvolvido neste artigo, foram realizados testes com 4 QR Codes de versões diferentes. Suas imagens foram rotacionadas em torno dos eixos x ou y, considerando que esses eixos passam pelo centro das imagens, como exemplificado pela Figura 5.

Figura 5: Convenção do posicionamento dos eixos de rotação em relação a (a) x e a (b) y.



(a) (b)

As perspectivas foram adicionadas às imagens dos QR Codes pelo software GIMP 2.10, o qual permite especificar os ângulos de rotação (θ_x e/ou θ_y) e dessa forma, simular perspectiva de imagens que foram fotografadas por uma câmera que não está posicionada paralelamente ao plano onde se encontra o QR Code. Para cada QR Code, são geradas 8 imagens, onde cada uma possui um ângulo de rotação diferente e resolução de 1200×1200 píxeis.

A *distância euclidiana* é a métrica utilizada para analisar a eficiência do algoritmo proposto neste artigo e a expressão

para o seu cálculo é dada por

$$d_i = \sqrt{(a_i - x_i)^2 + (b_i - y_i)^2} \quad (6)$$

onde d_i é a distância entre os pontos $P_i(a_i, b_i)$ e $V_i(x_i, y_i)$, para i = 1, 2, 3 e 4. Fica-se convencionado, conforme a Figura 3, que V_i representa as coordenadas exatas do i-ésimo vértice do QR Code, enquanto P_i são as coordenadas estimadas, pelo algoritmo proposto, de V_i.

A Tabela 1 apresenta os resultados obtidos em imagens com perspectivas em x.

Tabela 1: Distâncias entre os vértices e os pontos estimados pelo algoritmo proposto para imagens com perspectiva em x.

Nome da Imagem	Número de Módulos	θ_x	d_1	d_2	d_3	d_4
QR Code 1	21 × 21	5°	0,00	0,00	0,00	0,00
		15°	0,00	0,00	2,00	0,00
		25°	0,00	0,00	1,00	0,00
		35°	0,00	0,00	2,00	0,00
		45°	0,00	0,00	1,00	0,00
		55°	0,00	0,00	3,00	0,00
		65°	0,00	0,00	0,00	0,00
		75°	0,00	0,00	3,00	0,00
QR Code 2	37 × 37	5°	0,00	0,00	2,00	0,00
		15°	0,00	0,00	2,00	0,00
		25°	0,00	0,00	3,00	0,00
		35°	0,00	0,00	3,00	0,00
		45°	0,00	0,00	3,00	0,00
		55°	0,00	0,00	1,00	0,00
		65°	0,00	0,00	1,00	0,00
		75°	0,00	0,00	30,07	0,00
QR Code 3	41 × 41	5°	0,00	0,00	0,00	0,00
		15°	0,00	0,00	4,00	0,00
		25°	0,00	0,00	3,00	0,00
		35°	0,00	0,00	1,00	0,00
		45°	0,00	0,00	6,00	0,00
		55°	0,00	0,00	7,00	0,00
		65°	0,00	0,00	10,00	0,00
		75°	0,00	0,00	57,43	0,00
QR Code 4	57 × 57	5°	0,00	0,00	7,00	0,00
		15°	0,00	0,00	4,00	0,00
		25°	0,00	0,00	2,00	0,00
		35°	0,00	0,00	2,00	0,00
		45°	0,00	0,00	5,00	0,00
		55°	0,00	0,00	13,00	0,00
		65°	0,00	0,00	5,00	0,00
		75°	0,00	0,00	21,10	0,00

Nota-se na Tabela 1 que as distâncias euclidianas d_1, d_2 e d_4 possuem valor igual a 0 para todas as imagens, ou seja, os pontos P_1, P_2 e P_4 foram estimados corretamente. Isso ocorreu porque os três pontos encontrados pelo Algoritmo 1, representam as localizações exatas dos vértices V_1, V_2 e V_4.

Também percebe-se na Tabela 1 que 75% dos QR Codes tiverem $d_3 \leq 5$ píxeis e assim, o ponto P_3 se aproximou bem do vértice V_3.

Uma segunda bateria de teste é feita, porém agora as imagens tem perspectiva em relação ao eixo y e as distâncias entre os vértices dos QR Codes e os pontos estimadas são discriminadas na Tabela 2.

Verifica-se na Tabela 2 que o método proposto neste trabalho detectou corretamente os vértices V_1, V_2 e V_4 para todas as imagens testadas com perspectiva somente em relação ao eixo vertical, visto que as distâncias $d_1 = d_2 = d_4 = 0$. No entanto, P_3 não foi localizado na posição exata de V_3, porém os valores de d_3 são abaixo de 6 píxeis e assim, podendo assumir que o algoritmo tem uma boa precisão, levando em consideração que os QR Codes testados são de 1.200×1.200 píxeis.

Tabela 2: Distâncias entre os vértices e os pontos estimados pelo algoritmo proposto para imagens com perspectiva em y.

Nome da Imagem	Número de Módulos	θ_y	d_1	d_2	d_3	d_4
QR Code 1	21 × 21	5°	0,00	0,00	1,00	0,00
		15°	0,00	0,00	1,00	0,00
		25°	0,00	0,00	1,00	0,00
		35°	0,00	0,00	0,00	0,00
		45°	0,00	0,00	0,00	0,00
		55°	0,00	0,00	1,00	0,00
		65°	0,00	0,00	1,00	0,00
		75°	0,00	0,00	3,00	0,00
QR Code 2	37 × 37	5°	0,00	0,00	0,00	0,00
		15°	0,00	0,00	2,00	0,00
		25°	0,00	0,00	2,00	0,00
		35°	0,00	0,00	0,00	0,00
		45°	0,00	0,00	2,00	0,00
		55°	0,00	0,00	0,00	0,00
		65°	0,00	0,00	1,00	0,00
		75°	0,00	0,00	4,00	0,00
QR Code 3	41 × 41	5°	0,00	0,00	3,00	0,00
		15°	0,00	0,00	4,00	0,00
		25°	0,00	0,00	3,00	0,00
		35°	0,00	0,00	1,00	0,00
		45°	0,00	0,00	1,00	0,00
		55°	0,00	0,00	0,00	0,00
		65°	0,00	0,00	0,00	0,00
		75°	0,00	0,00	5,00	0,00
QR Code 4	57 × 57	5°	0,00	0,00	1,00	0,00
		15°	0,00	0,00	6,00	0,00
		25°	0,00	0,00	0,00	0,00
		35°	0,00	0,00	3,00	0,00
		45°	0,00	0,00	0,00	0,00
		55°	0,00	0,00	1,00	0,00
		65°	0,00	0,00	1,00	0,00
		75°	0,00	0,00	103,00	0,00

Nas Tabelas 1 e 2, d_3 apresente um pequeno erro e o primeiro motivo para isso é que as equações das retas r_2 e r_3 são determinadas a partir das coordenadas dos píxeis de uma imagem digital, os quais são valores inteiros, e não por valores reais das coordenadas dos pontos que formam as retas. Segunda razão é que as coordenadas do ponto $P_3(a_3, b_3)$ são determinadas através da Equação (5), resultando em a_3 e b_3 que são valores reais e ao final, essas coordenadas são truncadas para valores inteiros.

Tanto para o eixo horizontal quanto o vertical, os maiores valores das distâncias euclidianas foram para o ângulo de 75°. Este fato é explicado pela grande diminuição do tamanho dos *finder patterns* para essa angulação, o que conduz a erros no cálculo das inclinações das retas utilizadas para determinar as posições dos vértices.

Outrossim, ressalta-se que o algoritmo proposto neste artigo também é capaz de encontrar vértices quando as distorções ocorrem no eixo x e y ao mesmo tempo, com mostra a Tabela 3.

Pela Tabela 3, nota-se um aumento nas distâncias euclidianas d_2 e d_3. Isso ocorreu porque o Algoritmo 1 estimou as posições dos pontos V_2 e V_3 por meio de equações de retas. É notório, pela Tabela 3, que d_3 possui os maiores erros, isso porque, o P_3 é dado pela intersecção de duas retas. Assim, a determinação desse ponto pode acumular os erros da reta r_2 e da reta r_3. Isso é diferente do que ocorre na localização do P_2, que necessita apenas de r_1.

Tabela 3: Distâncias entre os vértices e os pontos estimados pelo algoritmo proposto para imagens com perspectiva em x e y.

Nome da Imagem	Número de Módulos	$\theta_x = \theta_y$	d_1	d_2	d_3	d_4
QR Code 1	21 × 21	5°	0,00	0,00	1,00	0,00
		15°	0,00	0,00	2,24	0,00
		25°	0,00	0,00	2,24	0,00
		35°	0,00	0,00	3,16	0,00
		45°	0,00	0,00	3,61	0,00
QR Code 2	37 × 37	5°	0,00	0,00	0,00	0,00
		15°	0,00	0,00	3,00	0,00
		25°	0,00	0,00	4,12	0,00
		35°	0,00	0,00	2,24	0,00
		45°	0,00	0,00	1,00	0,00
QR Code 3	41 × 41	5°	0,00	0,00	3,00	0,00
		15°	0,00	0,00	2,24	0,00
		25°	0,00	0,00	3,16	0,00
		35°	0,00	0,00	4,47	0,00
		45°	0,00	4,24	7,07	0,00
QR Code 4	57 × 57	5°	0,00	0,00	4,12	0,00
		15°	0,00	0,00	10,30	0,00
		25°	0,00	3,16	4,00	0,00
		35°	0,00	0,00	1,42	0,00
		45°	0,00	3,60	1,00	0,00

V. CONCLUSÃO

A localização dos vértices de um QR Code é uma das etapas mais importantes durante o processo de correção de distorções de perspectiva, pois sem o mapeamento da posição desse código bidimensional, não é possível aplicar métodos para corrigir seu formato. Para realizar esse processo, os algoritmos desenvolvidos a partir da transformada de Hough e do método de Harris podem apresentar um elevado tempo de processamento, além de necessitar que a imagem tenha as bordas (contornos) identificadas para que possam ser empregados. Além disso, a utilização da técnica de Hough pode não permitir encontrar as 4 retas do quadrilátero que envolve o QR Code sem determinar outras retas.

Desse modo, este artigo apresenta um algoritmo capaz de localizar, com uma boa precisão, os 4 vértices do QR Code a partir da utilização de no máximo 4 equações de retas e sem a necessidade de implementar métodos como a transformada de Hough e de Harris.

Ao realizar os testes, notou-se que para imagens com perspectiva somente em relação ao eixo x ou y, os vértices V_1, V_2 e V_4 tiveram suas posições encontradas de forma exata. No caso do V_3, percebe-se que as distâncias entre os pontos encontrados pelo algoritmo e as verdadeiras coordenadas, resultaram em valores menores que 3 píxeis em 75% das imagens. A maior distância ocorreu no QR Code 4 de Versão 10 e angulação $\theta_y = 75°$. No entanto, ressalta-se que não é usual inclinações próximas a 75° dos aparelhos celulares em relação à superfície dos QR Codes, quando esses dispositivos estão sendo utilizados para leituras desses códigos.

Para imagens com perspectiva em relação a x e y simultaneamente, foram testados angulações máximas de $\theta_x = 45°$ e $\theta_y = 45°$. Para essas distorções, observou-se um leve aumento nos valores de d_2, porém 85% dos valores foram 0. No caso do d_3, apesar de também ter um aumento em seus valores, percebe-se que a média dessa variável foi de 3,17 pixels. Assim, é evidenciado que o algoritmo também é útil para deformações em ambos os eixos de coordenadas.

REFERÊNCIAS

[1] Y. Kato, D. Deguchi, T. Takahashi, I. Ide, H. Mu-rase "Low resolution QR Code recognition by applying super-resolution using the property of QR Codes", *International Conference on Document Analysis and Recognition*, IEEE, 2011, pp. 992–996.

[2] L. V. M. Berrios, L. X. Medeiros, A. C. Mateus, "Métodos de binarização de imagens QR Code em distintos padrões de luminosidade", *Conferência de Estudos de Engenharia Elétrica da Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia*, 2019. Disponível em: `Available:https://www.peteletricaufu.com/static/ceel/artigos/artigo_376.pdf`

[3] J.-w. Wei, S.-g. Dai, A. Mu, "Rectification and localization of QR Code image based on methmatical morpholoy and hough transformation", *Computer and Information Technology*, vol. 6, 2010.

[4] J. Zhou, Y. Liu, A. Kumar, "Research on distortion correction of QR Code images", *International Journal of Computer Science and Technology*, 2012.

[5] H. Tribak, Y. Zaz, "QR Code patterns localization based on hu invariant moments", *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 9, p. 162, 2017.

[6] K. Suran, "QR Code image correction based on corner detection and convex hull algorithm", *Journal of multimedia*, vol. 8, no. 6, p. 662, 2013.

[7] "Qr code tutorial", 2015. Disponível em: `https://www.thonky.com/qr-code-tutorial/`

[8] T. J. Soon, "Qr Code", *Synthesis Journal*, vol. 2008, pp. 59–78, 2008.

[9] I. Boulos, P. de Camargo E. Oliveira, "Geometria analítica: um tratamento vetorial", *Prentice Hall Brasil*, 2005.

[10] A. Steinbruch, P. Winterle, "Geometria Analítica", *Pearson Makron Books*, 2004.