

# Worksheet2\_pdfversion

Creation date: 2024-11-08 15:54

Modification date: Friday 8th November 2024 15:54:10

PDF Ref: [Worksheet 2 - Espaços de cor e histogramas.pdf](#)

## Section 1 - Espaços de cor

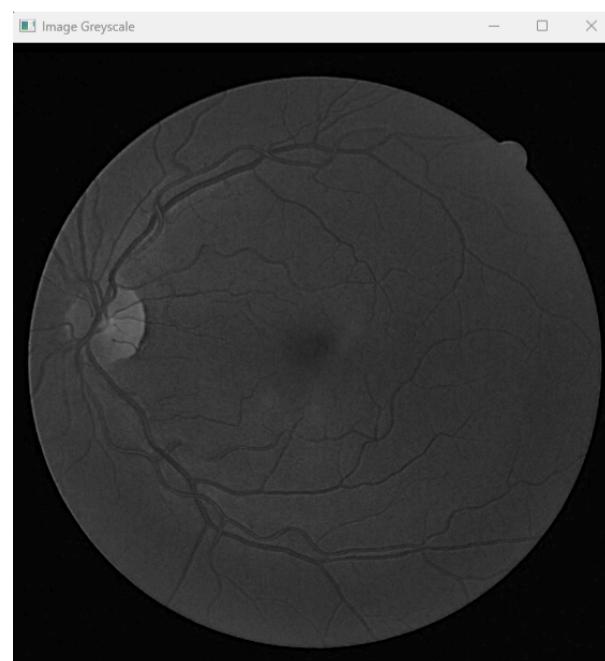
**1. Construa uma função que crie imagens grayscale de cada canal a partir de uma imagem RGB. A função deve entrar como parâmetro o canal a processar. Mostre as imagens resultantes.**

Código:

```
1 void exercise1_ws2(int channel) {  
2  
3     // channels B-G-R  
4     Mat image_extract = imread("J:/MEEC/PIVC/Projects/Project_PIVC_Worksheets/retina.tif");  
5     // greyscale reference image / CV_8UC1 -> 1 channel  
6     Mat grayImage(image_extract.rows, image_extract.cols, CV_8UC1, Scalar(0));  
7     // go through all the pixels in the image  
8     for (int r = 0; r < image_extract.rows; r++) {  
9         for (int c = 0; c < image_extract.cols; c++) {  
10             Vec3b pixel_value = image_extract.at<Vec3b>(r, c); // extract the values from the picture  
11             grayImage.at<uchar>(r, c) = pixel_value[channel]; // transfer the values to the reference  
image  
12         }  
13     }  
14     imshow("Image Greyscale", grayImage);  
15     waitKey(0);  
16     destroyAllWindows();  
17 }
```

Resultados:

- O resultado é uma imagem em greyscale com apenas o canal escolhido pelo utilizador.
- Inicialmente a imagem é percorrida pixel a pixel e são extraídos os valores dos pixels para cada matriz de cor. Os seus valores são guardados numa variável do tipo Vec3b.
- Por fim, apenas os pixels correspondentes ao canal escolhido são "transferidos" para a imagem de referência criada com o tipo CV\_8UC1 (tipo 1 canal por pixel).
- A imagem é greyscale porque nenhum dos outros dois canais foi anulado.
- É possível também verificar uma diferença na intensidade dos pixels entre os dois exemplos seguintes, onde no primeiro exemplo foi retirado o canal azul e no segundo exemplo, o canal vermelho.



Pasted image 20241108180125.png#center



Pasted image 20241108180152.png#center

## 2. Construa funções que elimine as componentes R, G ou B de uma imagem RGB. A função deve entrar como parâmetro o canal a extrair. Mostre as imagens resultantes.

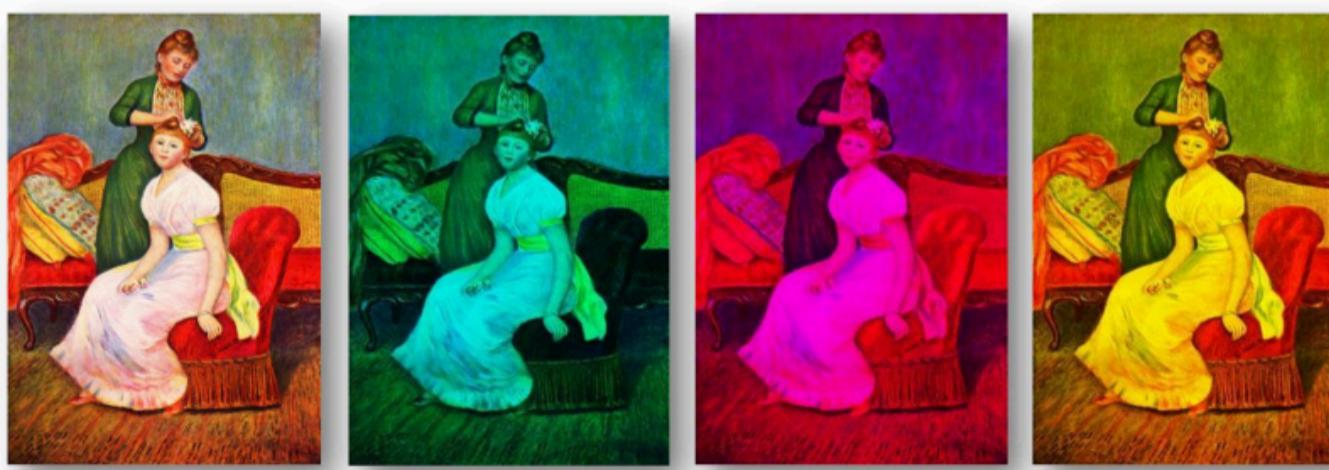
Código:

```

1 void exercise2_ws2(int channel) {
2     // channels B-G-R
3     Mat image_extract = imread("J:/MEEC/PIVC/Projects/Project_PIVC_Worksheets/retina.tif");
4     // reference image / CV_8UC3 -> 3 channel
5     Mat newImage(image_extract.rows, image_extract.cols, CV_8UC3, Scalar(0));
6
7     for (int r = 0; r < image_extract.rows; r++) {
8         for (int c = 0; c < image_extract.cols; c++) {
9             Vec3b pixel_value = image_extract.at<Vec3b>(r, c); // extract the values from the picture
10            pixel_value[channel] = 0; // remove the channel that the user typed // all pixels to 0
11            newImage.at<Vec3b>(r, c) = pixel_value; // save the other channels in the reference image
12        }
13    }
14    imshow("Image 2 channels", newImage);
15    waitKey(0);
16    destroyAllWindows();
17 }
```

Referências:

Decomposição da imagem RGB



Original

Sem Vermelho

Sem Verde

Sem Azul

→ Imagem de três canais com duas componentes representadas:  
 $I = \{I_R, I_G, I_B\}$

Pasted image 20241108163051.png > center

Operação	Fórmula	Efeito
<b>Negativo</b>	$C = 255 - C$	Calcula a cor oposta, por exemplo, o preto torna-se branco, o vermelho em ciano, etc.
<b>Escurecer</b>	$C = C - P$	Subtrai por uma constante, para a tornar mais escura.
<b>Clarear</b>	$C = C + P$	Adiciona uma constante, para a tornar mais clara.
<b>Cinzento</b>	$(R + G + B) / 3$	Calcula a média dos três canais (componentes) de cor, de modo a obter uma tonalidade cinzenta.
<b>Remover Canal</b>	$R = 0, G = 0$ e/ou $B = 0$	Colocando um ou mais canais a 0 (zero), remove-se completamente a contribuição dessa componente de cor.
<b>Trocar Canais</b>	$R = G, G = R, \dots$	Trocar valores entre dois canais. Esta operação altera a cor.

Pasted image 20241108174352.png#center

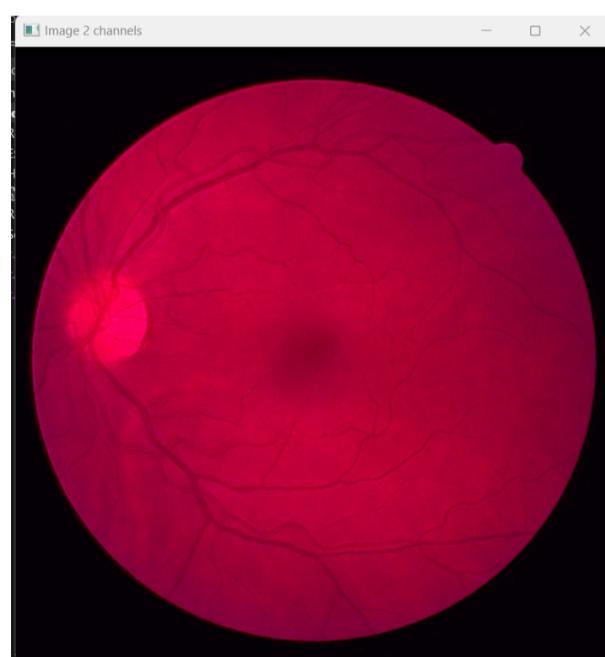
### Resultados:

- Extrair um canal é o mesmo que o anular (todos os pixels = 0).
- A função percorre toda a imagem pixel a pixel, e extrai todos os valores das matrizes RGB para uma variável Vec3b.
- Em seguida anula todos os valores que correspondem ao index da matriz escolhida.
- Por último, copia os valores das outras duas matrizes para a nova imagem criada com o tipo CV\_8UC3 (tipo 3 canais por pixel).
- A imagem seguinte é o resultado da remoção do canal azul, que resulta numa imagem com um tom amarelado, da mistura dos canais vermelho e verde.
- O canal azul normalmente tem menos influência nesses tipos de imagens médicas, capturando menos detalhes devido ao comprimento de onda mais curto e à maior dispersão no tecido.



Pasted image 20241108180641.png#center

- A imagem seguinte representa a remoção do canal verde, e demonstra que o canal vermelho tem uma maior intensidade em relação ao azul.
- O canal vermelho tende a capturar mais detalhes da estrutura geral e do fundo, à medida que a luz vermelha penetra mais profundamente no tecido.



Pasted image 20241108180714.png#center

- A imagem seguinte representa a remoção do canal vermelho.
- Verifica-se que o canal azul tem muito baixa influência.
- O canal verde fornece maior contraste para os vasos sanguíneos e certas estruturas dentro da retina devido à forma como o olho absorve a luz verde.



Pasted image 20241108180739.png#center

### 3. Construa uma função que converta uma imagem no espaço RGB para uma imagem em tons de cinzento

Código:

```
1 void exercise3_ws2(int userChoice) {
2     // channels B-G-R
3     Mat image_extract = imread("J:/MEEC/PIVC/Projects/Project_PIVC_Worksheets/retina.tif");
4     // greyscale reference image / CV_8UC1 -> 1 channel
5     Mat grayImage(image_extract.rows, image_extract.cols, CV_8UC1, Scalar(0));
6     cv::String imageLabel;
7     if (userChoice) { //weighted
8         for (int r = 0; r < image_extract.rows; r++) {
9             for (int c = 0; c < image_extract.cols; c++) {
10                 Vec3b pixel_value = image_extract.at<Vec3b>(r, c);
11                 // calculate the intensity using the weighted formula
12                 uchar intensity = (
13                     (pixel_value[0] * 0.144) +
14                     (pixel_value[1] * 0.587) +
15                     (pixel_value[2] * 0.299));
16                 grayImage.at<uchar>(r, c) = intensity; // save
17             }
18         }
19         imageLabel = "Weighted Formula Greyscale Image";
20     }
21     else { //normal
22         for (int r = 0; r < image_extract.rows; r++) {
23             for (int c = 0; c < image_extract.cols; c++) {
24                 Vec3b pixel_value = image_extract.at<Vec3b>(r, c); // extract the values from the picture
25                 // calculate the intesity between all channels
26                 uchar intensity = (pixel_value[0] + pixel_value[1] + pixel_value[2]) / 3;
27                 grayImage.at<uchar>(r, c) = intensity; // save the other channels
28             }
29         }
30         imageLabel = "Normal Formula Greyscale Image";
31     }
32     imshow(imageLabel, grayImage);
33     waitKey(0);
34     destroyAllWindows();
35 }
```

## Referências:

Operação	Fórmula	Efeito
<b>Negativo</b>	$C = 255 - C$	Calcula a cor oposta, por exemplo, o preto torna-se branco, o vermelho em ciano, etc.
<b>Escurecer</b>	$C = C - P$	Subtrai por uma constante, para a tornar mais escura.
<b>Clarear</b>	$C = C + P$	Adiciona uma constante, para a tornar mais clara.
<b>Cinzento</b>	$(R + G + B) / 3$	Calcula a média dos três canais (componentes) de cor, de modo a obter uma tonalidade cinzenta.
<b>Remover Canal</b>	$R = 0, G = 0$ e/ou $B = 0$	Colocando um ou mais canais a 0 (zero), remove-se completamente a contribuição dessa componente de cor.
<b>Trocar Canais</b>	$R = G, G = R, \dots$	Trocar valores entre dois canais. Esta operação altera a cor.

Pasted image 20241108174352.png#center

$$\text{Intensidade} = R * 0.299 + G * 0.587 + B * 0.144$$

Pasted image 20241108173002.png > center

## Resultados:



Pasted image 20241108181625.png#center



Pasted image 20241108181716.png#center

- A imagem é percorrida pixel a pixel e os valores da intensidade para as 3 matrizes são extraídos da imagem original para um vetor Vec3b.
- De seguida, calculou-se o valor da nova intensidade que corresponde a uma média aritmética dos valores dos 3 canais (ordem BGR).
- No caso da média ponderada, recorreu-se a uma fórmula que corresponde ao peso das cores na percepção do olho humano.
- Por fim, os valores da intensidade são "transferidos" para a nova imagem greyscale com apenas 1 canal (tipo CV\_8UC1).
- Em comparação aos dois outputs, não são detectáveis quaisquer diferenças consideráveis, no entanto, com a média pesada, a imagem parece ligeiramente mais clara na zona do nervo óptico.

## Section 2 - Histogramas e operações ponto-a-ponto

### 4. Construa uma função que calcule o negativo de uma imagem. Faça para uma imagem RGB e para uma imagem em grayscale.

- Use a imagem "breast.png";
- Use a imagem "retina.tif";

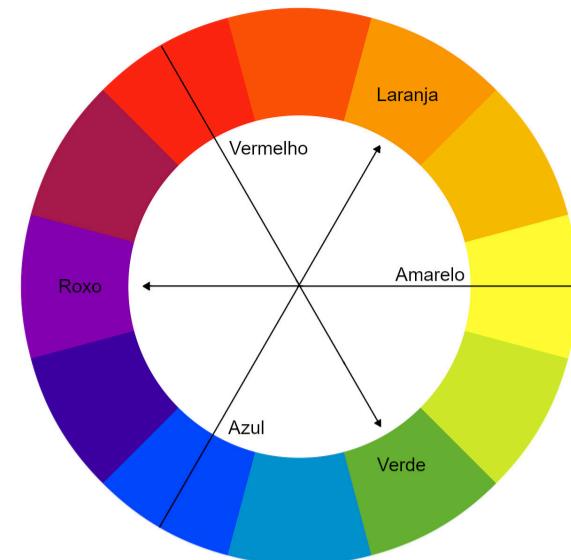
Código:

```
1 void exercise4_ws2(int userChoice) {  
2  
3     if (userChoice) { // breast  
4         Mat image = imread("J:/MEEC/PIVC/Projects/Project_PIVC_Worksheets/breast_image.png",  
5         IMREAD_GRAYSCALE);  
6         // greyscale reference image / CV_8UC1 -> Channel  
7         Mat newImage(image.rows, image.cols, CV_8UC1, Scalar(0));  
8  
9         //since all pixels and channels are modified using the same expression, we can avoid the use of 2  
10        loops and if condition for gray and rgb  
11        for (int i = 0; i < image.rows * image.cols * image.channels(); i++) {  
12            newImage.data[i] = 255 - image.data[i];  
13        }  
14        imshow("Image Negative Greyscale", newImage);  
15    }  
16    else { // retina  
17        Mat image = imread("J:/MEEC/PIVC/Projects/Project_PIVC_Worksheets/retina.tif");  
18        // reference image / CV_8UC3 -> 3 channels  
19        Mat newImage(image.rows, image.cols, CV_8UC3, Scalar(0));  
20  
21        //since all pixels and channels are modified using the same expression, we can avoid the use of 2  
22        loops and if condition for gray and rgb  
23        for (int i = 0; i < image.rows * image.cols * image.channels(); i++) {  
24            newImage.data[i] = 255 - image.data[i];  
25        }  
26        imshow("Image Negative RGB", newImage);  
27    }  
28    waitKey(0);  
29    destroyAllWindows();  
30}
```

Referências:

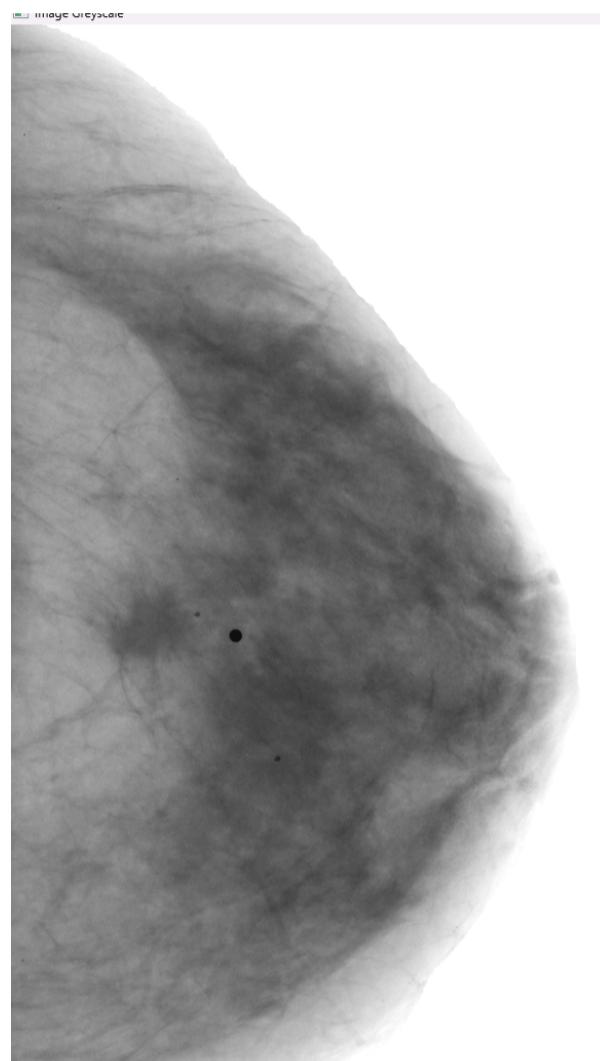
Operação	Fórmula	Efeito
<b>Negativo</b>	$C = 255 - C$	Calcula a cor oposta, por exemplo, o preto torna-se branco, o vermelho em ciano, etc.
<b>Escurecer</b>	$C = C - P$	Subtrai por uma constante, para a tornar mais escura.
<b>Clarear</b>	$C = C + P$	Adiciona uma constante, para a tornar mais clara.
<b>Cinzento</b>	$(R + G + B) / 3$	Calcula a média dos três canais (componentes) de cor, de modo a obter uma tonalidade cinzenta.
<b>Remover Canal</b>	$R = 0, G = 0$ e/ou $B = 0$	Colocando um ou mais canais a 0 (zero), remove-se completamente a contribuição dessa componente de cor.
<b>Trocar Canais</b>	$R = G, G = R, \dots$	Trocar valores entre dois canais. Esta operação altera a cor.

Pasted image 20241108174352.png#center

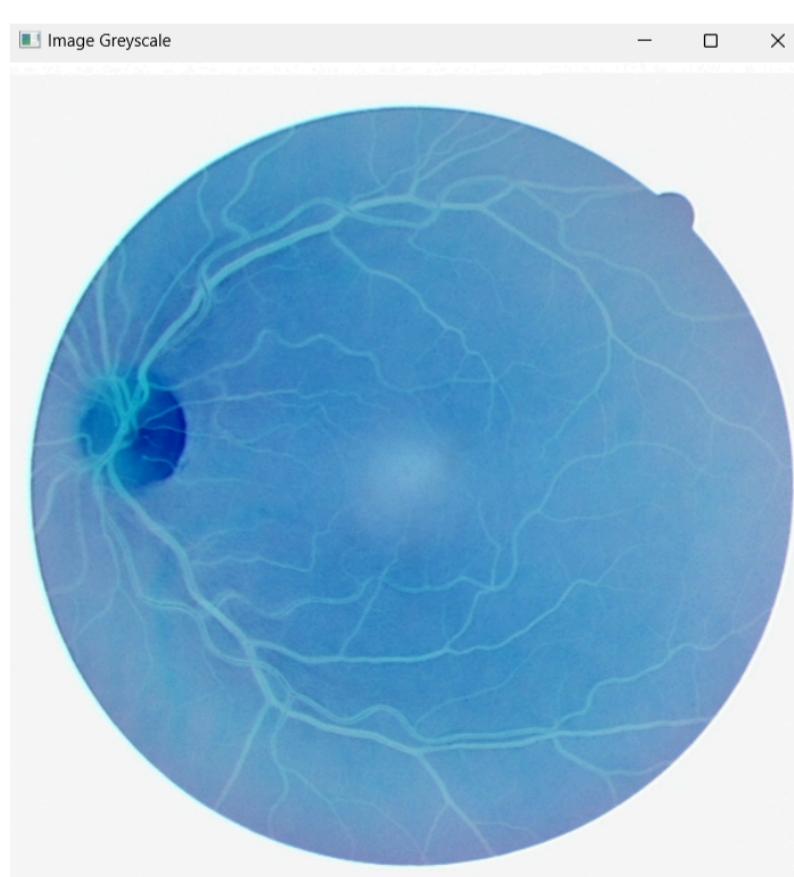


Pasted image 20241112154044.png#center

*Resultados:*



Pasted image 20241108175223.png#center



Pasted image 20241109154427.png#center

- O código funciona da mesma maneira em termos de inversão de cor, as diferenças estão na quantidade de canais, que para RGB são 3 e greyscale é 1. Logo, a imagem de referência criada terá os tipos CV\_8UC1 para 1 canal e CV\_8UC3 para 3 canais.
- No caso greyscale, a chamada da função imread(), tem o parâmetro adicional IMREAD\_GRAYSCALE para realizar a leitura direta, sem necessidade de converter.
- Como todos os pixels sofrem alteração, não existe necessidade dos dois loops para fazer as alterações ponto a ponto. Para calcular o negativo recorreu-se à tabela nos slides, respeitando a fórmula de  $C = 255 - C$ .
- Na imagem Breast verificou-se que a inversão de cor vai colocar o background da imagem a branco, e as zonas dentro do RaioX que anteriormente eram mais claras, são agora as mais escuras.
- Conforme verificado nas referências o negativo de uma cor alaranjada é uma cor azulada, e na conversão realizada na imagem Retina, o output corresponde ao inverso de uma cor preta e laranja (background e foreground), que é branco e azul.

**5. Repita o exercício anterior mas evitando a alteração onde não é pretendido (background da imagem). Nota: para a imagem "breast.png", utilize a máscara fornecida. Para a imagem "retina.tif" deverá utilizar uma condição para verificar os valores que não devem ser alterados.**

*Código:*

```

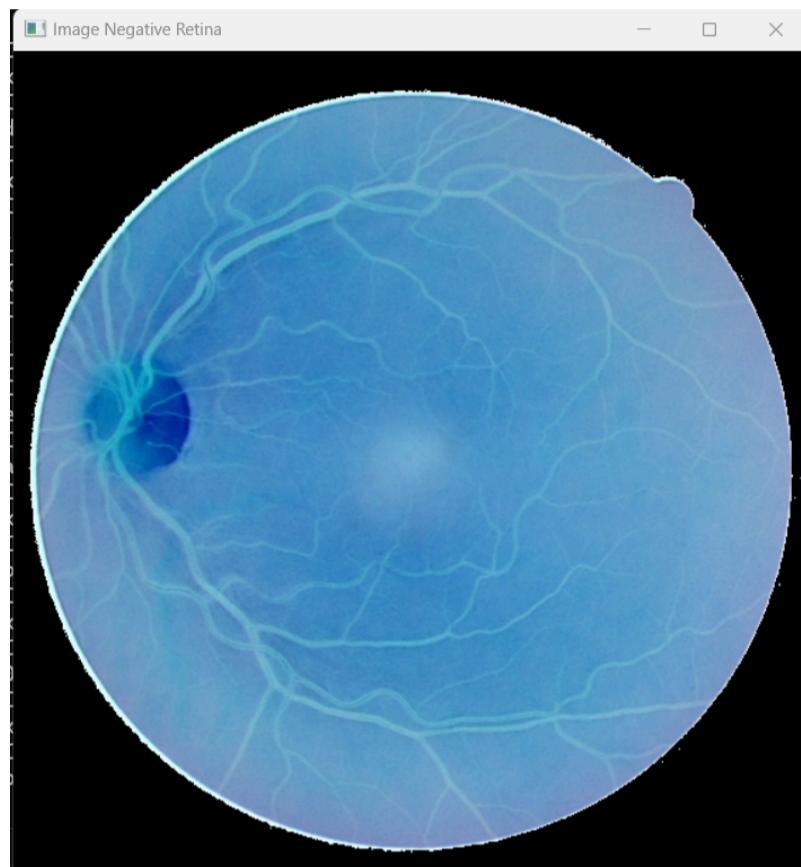
1 void exercise5_ws2(int imageChoice) {
2
3     if (imageChoice) { // breast
4         Mat image = imread("J:/MEEC/PIVC/Projects/Project_PIVC_Worksheets/breast_image.png",
5         IMREAD_GRAYSCALE);
6         Mat mask = imread("J:/MEEC/PIVC/Projects/Project_PIVC_Worksheets/breast_mask.png",
7         IMREAD_GRAYSCALE);
8
9         Mat invertedImage(image.rows, image.cols, CV_8UC1, Scalar(0));
10        //since all pixels and channels are modified using the same expression, we can avoid the use of 2
11        //loops and if condition for gray and rgb
12        for (int i = 0; i < image.rows * image.cols * image.channels(); i++) {
13            if (mask.data[i] != 0) { // apply the negative only on the foreground
14                invertedImage.data[i] = 255 - image.data[i];
15            }
16        }
17        imshow("Image Negative Breast Mask", invertedImage);
18    }
19    else { // retina
20        Mat image = imread("J:/MEEC/PIVC/Projects/Project_PIVC_Worksheets/retina.tif");
21        // rgb reference image / CV_8UC3 -> Channel
22        Mat inverted_img(image.rows, image.cols, CV_8UC3, Scalar(0));
23        // iterate through all the pixels
24        for (int i = 0; i < image.rows; i++) {
25            for (int j = 0; j < image.cols; j++) {
26                Vec3b vec_value = image.at<Vec3b>(i, j);
27                // inversion of all channels in a threshold
28                if ((vec_value[0] > 15) || (vec_value[1] > 15) || (vec_value[2] > 15)) {
29                    vec_value[0] = 255 - vec_value[0];
30                    vec_value[1] = 255 - vec_value[1];
31                    vec_value[2] = 255 - vec_value[2];
32                    inverted_img.at<Vec3b>(i, j) = vec_value; // save on the reference image
33                }
34            }
35        }
36        imshow("Image Negative Retina", inverted_img);
37    }
38    waitKey(0);
39    destroyAllWindows();
40 }
```

## Referências:

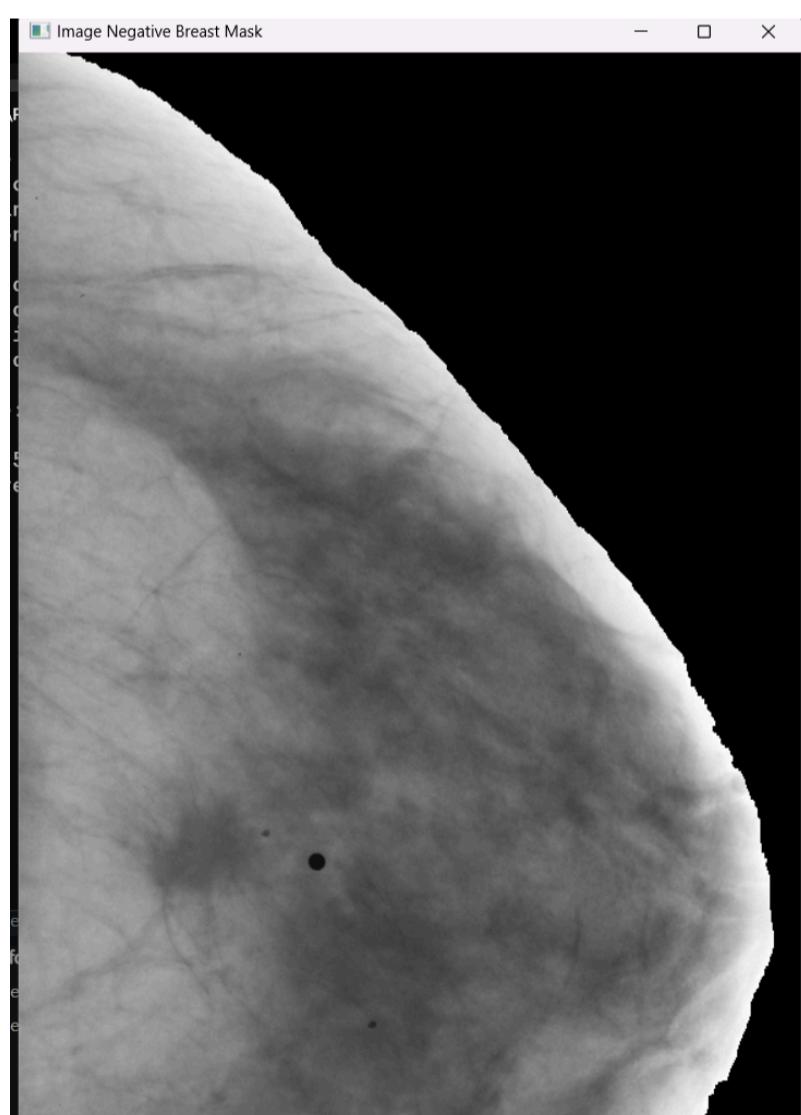
Operação	Fórmula	Efeito
<b>Negativo</b>	$C = 255 - C$	Calcula a cor oposta, por exemplo, o preto torna-se branco, o vermelho em ciano, etc.
<b>Escurecer</b>	$C = C - P$	Subtrai por uma constante, para a tornar mais escura.
<b>Clarear</b>	$C = C + P$	Adiciona uma constante, para a tornar mais clara.
<b>Cinzenço</b>	$(R + G + B) / 3$	Calcula a média dos três canais (componentes) de cor, de modo a obter uma tonalidade cinzenta.
<b>Remover Canal</b>	$R = 0, G = 0$ e/ou $B = 0$	Colocando um ou mais canais a 0 (zero), remove-se completamente a contribuição dessa componente de cor.
<b>Trocar Canais</b>	$R = G, G = R, \dots$	Trocar valores entre dois canais. Esta operação altera a cor.

Pasted image 20241108174352.png#center

## Resultados:



Pasted image 20241111151303.png#center



Pasted image 20241111151338.png#center

- A abordagem para realizar o exercício da imagem Retina passou por realizar a inversão de cada canal conforme um threshold de cor. Esse threshold foi definido para isolar a cor branca do background, e eliminar essa zona.
- Como não será aplicado a toda a imagem o negativo, voltou-se a utilizar os dois loops para percorrer a imagem pixel a pixel, e para todas as intensidades acima do valor 15, aplica-se a fórmula.
- No caso da imagem Breast, utilizou-se uma máscara binária para isolar o background da imagem.
- Através de uma simples condição, toda a zona que estiver a preto na máscara é correspondente ao background e o valor das intensidades dos pixels nessa zona, é ignorado. Apenas a zona de interesse do RaioX é transferida e calculado o negativo para a imagem de referência final.

## 6. Faça uma função para alterar o brilho de uma imagem. Aplique à imagem “breast.png”. Mostre o resultado de duas operações (diminuir e aumentar o brilho).

Código:

```

1 void exercise6_ws2(int userChoice, int bvalue) {
2
3     Mat image = imread("J:/MEEC/PIVC/Projects/Project_PIVC_Worksheets/breast_image.png");
4     //reference image / CV_8UC3 -> 3 channels
5     Mat outputImage(image.rows, image.cols, CV_8UC3, Scalar(0));
6     cv::String imageLabel;
7
8     if (userChoice) { // brigther
9
10         //since all pixels and channels are modified using the same expression, we can avoid the use of 2
11         //loops and if condition for gray and rgb
12         for (int i = 0; i < image.rows * image.cols * image.channels(); i++) {
13             outputImage.data[i] = image.data[i] + bvalue; // addition of a constant to increase the
14             brightness
15         }
16         imageLabel = "Increased Brightness";
17     }
18     else {
19
20         //since all pixels and channels are modified using the same expression, we can avoid the use of 2
21         //loops and if condition for gray and rgb
22         for (int i = 0; i < image.rows * image.cols * image.channels(); i++) {
23             outputImage.data[i] = image.data[i] - bvalue; // addition of a constant to decrease the
24             brightness
25         }
26         imageLabel = "Decreased Brightness";
27     }
28     imshow(imageLabel, outputImage);
29     waitKey(0);
30     destroyAllWindows();
31 }
```

Referências:

Operação	Fórmula	Efeito
<b>Negativo</b>	$C = 255 - C$	Calcula a cor oposta, por exemplo, o preto torna-se branco, o vermelho em ciano, etc.
<b>Escurecer</b>	$C = C - P$	Subtrai por uma constante, para a tornar mais escura.
<b>Clarear</b>	$C = C + P$	Adiciona uma constante, para a tornar mais clara.
<b>Cinzento</b>	$(R + G + B) / 3$	Calcula a média dos três canais (componentes) de cor, de modo a obter uma tonalidade cinzenta.
<b>Remover Canal</b>	$R = 0, G = 0$ e/ou $B = 0$	Colocando um ou mais canais a 0 (zero), remove-se completamente a contribuição dessa componente de cor.
<b>Trocar Canais</b>	$R = G, G = R, \dots$	Trocar valores entre dois canais. Esta operação altera a cor.

Pasted image 20241108174352.png#center



Decreased brightness



Input image



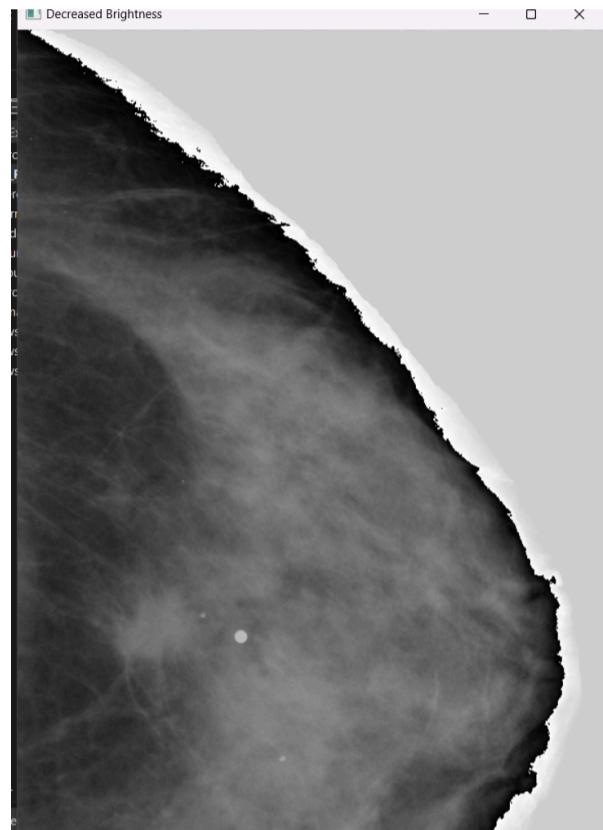
Increased brightness

Pasted image 20241109225514.png > center

*Resultados:*

*Diminuição da luminosidade:*

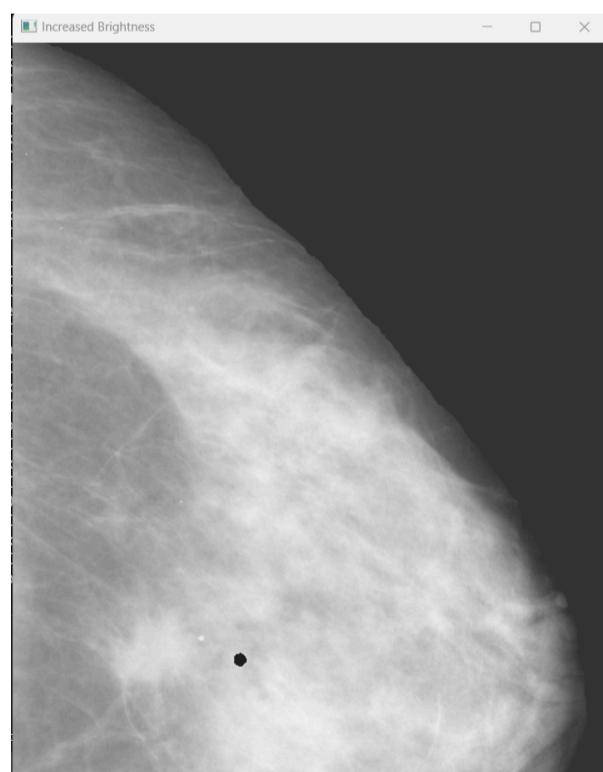
$b=50$



Pasted image 20241109224906.png#center

*Aumento de luminosidade:*

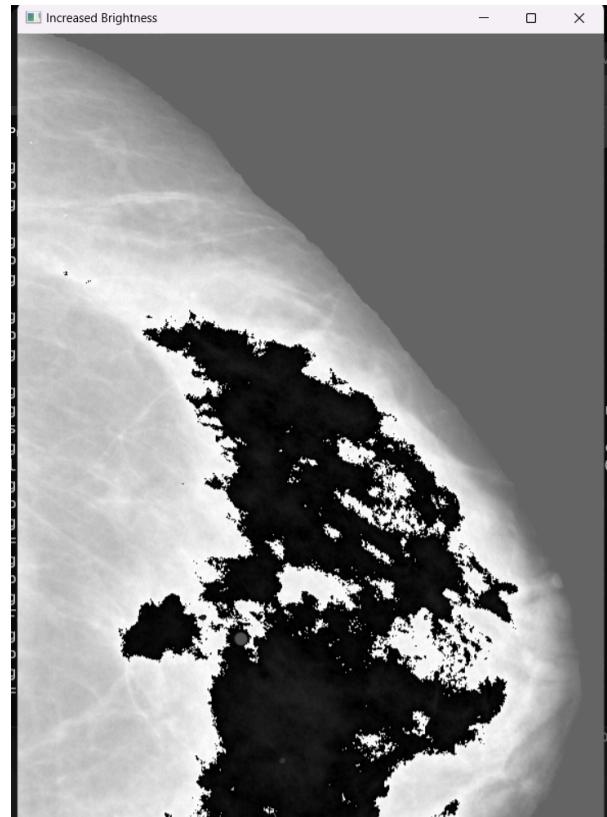
$b=50$



Pasted image 20241109225014.png#center

*Aumento brusco de luminosidade:*

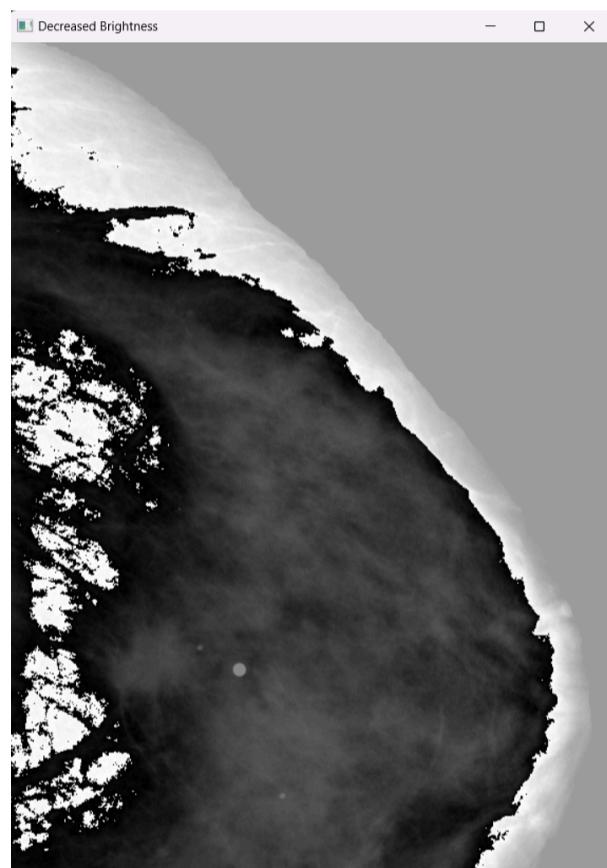
b=100



Pasted image 20241112181753.png#center

*Diminuição brusca de luminosidade:*

b=100



Pasted image 20241112181832.png#center

- O código vai ponto a ponto por todos os pixels de todos os canais (1 canal), para poder adicionar ou subtrair um valor constante ao valor da intensidade dos pixels.
- A abordagem para o aumento ou decréscimo da luminosidade foi baseado no exemplos dos slides, onde é introduzida uma constante na fórmula de "transferência" entre a imagem original e a imagem de referência final.
- Clarear ou escurecer é literalmente manipular o valor da intensidade dos pixels da imagem (neste caso, com a constante b).
- Se o valor for muito alto, vai danificar a imagem com o aparecimento de artefactos.

**7. Faça uma função para alterar o contraste de uma imagem. Aplique à imagem "breast.png". Mostre o resultado de duas operações (diminuir e aumentar o contraste).**

Código:

```
1 void exercise7_ws2(float alphavalue) {  
2  
3     Mat image = imread("J:/MEEC/PIVC/Projects/Project_PIVC_Worksheets/breast_image.png");  
4     // reference image / CV_8UC3 -> 3 channels  
5     Mat outputImage(image.rows, image.cols, CV_8UC3, Scalar(0));  
6     cv::String imageLabel;  
7  
8     //since all pixels and channels are modified using the same expression, we can avoid the use of 2  
9     loops and if condition for gray and rgb  
10    for (int i = 0; i < image.rows * image.cols * image.channels(); i++) {  
11        outputImage.data[i] = image.data[i] * alphavalue; // addition of a constant increase the contrast  
12    }  
13    if (alphavalue > 1) { // more contrast  
14        imageLabel = "Increased Contrast";  
15    }  
16    else{ //less contrast  
17        imageLabel = "Decreased Contrast";  
18    }  
19    imshow(imageLabel, outputImage);  
20    waitKey(0);  
21    destroyAllWindows();  
21 }
```

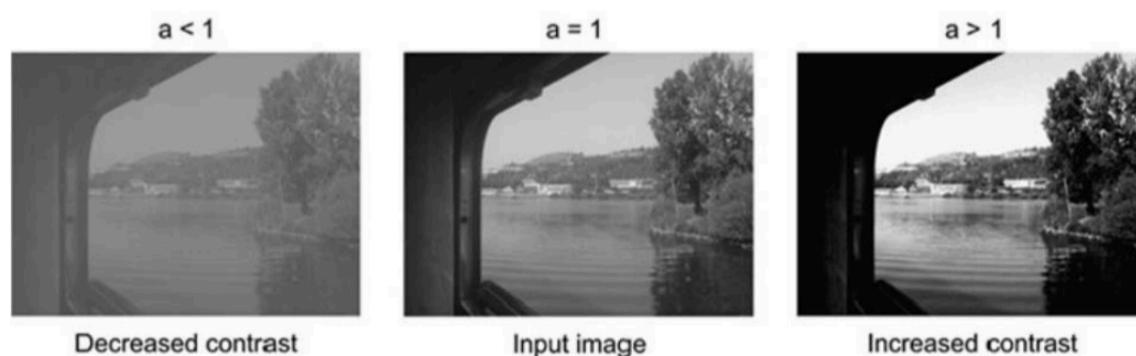
Referências:

#### MANIPULAÇÃO DO CONTRASTE

$$g(i,j) = f(i,j) \times \alpha + b;$$

- se  $\alpha < 1$ , há uma atenuação do contraste;
- se  $\alpha > 1$ , há um aumento do contraste;
- Devemos fazer o clipping dos valores que excedam a gama dinâmica permitida.

Pasted image 20241109225405.png#center

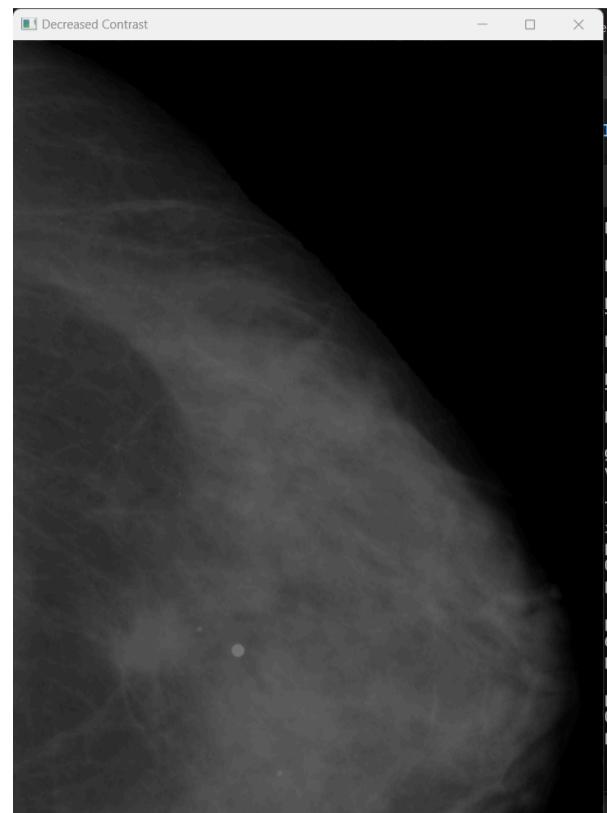


Pasted image 20241109225635.png#center

## Resultados:

### Menor contraste:

- $a = 0.5$



Pasted image 20241109225942.png#center

### Maior contraste:

- $a = 1.5$



Pasted image 20241109230026.png#center

- Novamente o código vai percorrer a imagem ponto a ponto, e manipular a intensidade de cada pixel.
- A abordagem para o aumento ou decréscimo do contraste foi baseado no exemplos dos slides, onde é introduzida uma constante na fórmula de "transferência" entre a imagem original e a imagem de referência final.
- Neste caso o valor da constante é multiplicado pelo valor da intensidade do pixel.
- Se o valor for muito alto, vai danificar a imagem com o aparecimento de artefatos.

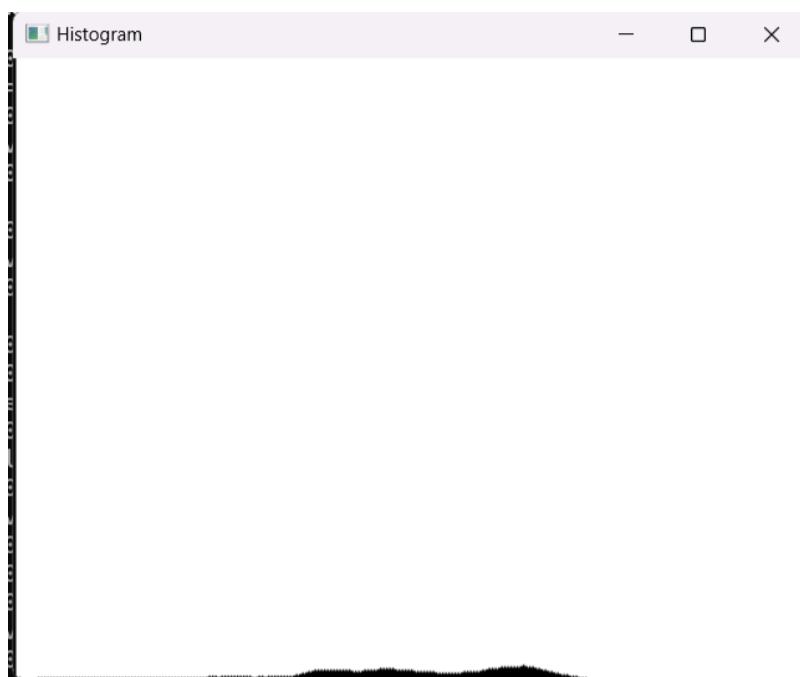
**8. Construa uma função que exiba o histograma de uma imagem em tons de cinzento. Aplique para a imagem “breast.png” e para as imagens transformadas da questão 6 e 7.**

**O que verifica nos histogramas?**

*Código:*

```
1
```

\*Resultados normal:



*Resultados :*

**9. Construa a função que realize a equalização de imagens em tons de cinzento. Aplique à imagem “breast.png”.**

Equalizar um histograma é o mesmo que realizar uma conversão de RGB para HSV.