

Project 2 Writeup

In the beginning...

Em processamento de imagens, trabalha-se com tons de cinza, conhecidos como *digital number* ou DNs, que são atribuídos aos pixels de uma imagem. O histograma é uma das formas mais comuns de se representar a distribuição DNs de uma imagem. Ele fornece a informação sobre quantos pixels na imagem possuem cada valor possível de DN (no caso das imagens de 8 bits, variam de 0 a 255) ou, de forma equivalente, onde a proporção da imagem que corresponde a cada valor de DN [engcar 2018].

Os histogramas também são conhecidos como distribuição de intensidades ou Função de Densidade de Probabilidade (PDF). Esse último termo deriva do fato de que, estatisticamente, o histograma representa, a probabilidade de se achar um DN de um dado valor dentro de uma imagem [engcar 2018].

Ao se observar o histograma de uma imagem, tem-se uma noção sobre as características da mesma. A forma do histograma fornece informações de grande importância da imagem, tal como intensidade média e espalhamento dos valores de DN; este último, por sua vez, dá a medida do contraste de uma imagem: quanto maior o espalhamento ao longo do eixo dos DNs, maior o contraste da imagem (Figura 1) [engcar 2018].

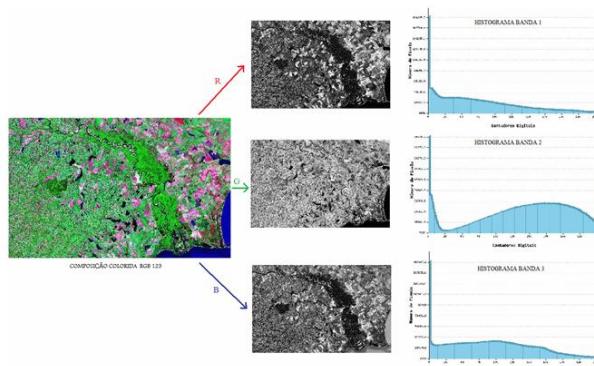


Figure 1: Histograma de uma imagem separados por canais RGB.

Em processamento de imagens, existe uma técnica denominada correspondência ou especificação de histograma que é a transformação de uma imagem para que seu histograma corresponda a um histograma especificado [Wikipedia 2018].

Um detector pode ser usado para normalizar duas imagens, quando as essas foram adquiridas na mesma iluminação local (como sombras) sobre o mesmo local, mas por diferentes sensores, condições atmosféricas ou iluminação global [Wikipedia 2018].

Imagens com valores de pixels de 8 bits, a correspondência de histogramas só pode aproximar o histograma especificado. Todos os pixels de um determinado valor na imagem original devem ser transformados em apenas um valor na imagem de saída [Wikipedia 2018].

Existem diversas técnicas propostas para resolver o problema de correspondência de histograma, uma delas é por meio do Recozimento simulado - *Simulated Annealing*.

O Recozimento simulado é uma meta-heurística para otimização que consiste numa técnica de busca local probabilística, e se fundamenta numa analogia entre o modo como um metal se resfria e congela numa estrutura cristalina de energia mínima e a busca por um mínimo num sistema qualquer [Pratschke e Pelizzoni 1999].

A cada iteração, o algoritmo do Recozimento simulado, procura o próximo candidato a ponto de mínimo na vizinhança do candidato corrente, agindo de acordo com a diferença entre os valores da função-objetivo (função de energia ou potencial). A maior vantagem desse algoritmo é a possibilidade de evitar mínimos locais, para isso, é realizada uma busca aleatória que, por vezes, aceita vizinhos cuja energia seja mais elevada. Ou seja, em algumas iterações, o Recozimento simulado tende a maximizar a função-objetivo em vez de minimizá-la. Entretanto a probabilidade de se aceitar um vizinho de maior energia cai com o tempo [Pratschke e Pelizzoni 1999].

Para realizar a correspondência de histograma de uma imagem por meio da solução do Recozimento simulado, foi utilizado a linguagem Octave. As imagens utilizadas foram tiradas de um mesmo local, porém a que foi modificada seu histograma possui variação na sua intensidade, ou seja, ela está mais avermelhada (Figura 2).



Figure 2: *Esquerda*: Imagem original. *Direita*: Imagem modificada.

Inicialmente, as imagens trabalhadas foram partes da imagem original (Figura 3), pois dessa forma acelerou o mapeamento do Recozimento simulado, e solução obtida pelo algorísmo foi aplicada a imagem modificada de tamanho original (2).



Figure 3: *Esquerda*: Patch da imagem original. *Direita*: Patch da imagem modificada.

O objetivo do código foi mapear: o histograma do patch da imagem modificada de forma que se aproximasse do histograma do patch da imagem original, e o histograma da imagem modificada para se aproximar do histograma da imagem original.

Para atingir o objetivo, os canais RGB das imagens modificadas foram separados e trabalhados individualmente, ou seja, foram mapeado os histogramas dos canais vermelho,

verde e azul, depois esses foram juntados novamente para formar a imagem colorida final modificada. Cada canal retorna uma imagem em escala cinza, com intensidades de 0 a 255.

Interesting Implementation Detail

Inicialmente, o código deve carregar as imagens e separar seus canais RGB. Isso é feito em arquivo chamado Proj2.m, que está descrito abaixo:

```

1 image1 = imread('../data/img1.jpg');
2 image2 = imread('../data/img2.jpg');
3 image1patch = imread('../data/img1_patch.jpg');
4 image2patch = imread('../data/img2_patch.jpg');
5
6 % Canais RGB PATCH1
7 red = image1patch(:,:,1); % Canal vermelho
8 green = image1patch(:,:,2); % Canal verde
9 blue = image1patch(:,:,3); % Canal azul
10
11 % Canais RGB PATCH2
12 red2 = image2patch(:,:,1); % Canal vermelho
13 green2 = image2patch(:,:,2); % Canal verde
14 blue2 = image2patch(:,:,3); % Canal azul
15
16 canalRed = SimulatedAnnealing(red, red2);
17 canalGreen = SimulatedAnnealing(green, green2);
18 canalBlue = SimulatedAnnealing(blue, blue2);
19
20 % Voltar para imagem original
21 img_patch2 = cat(3, canalRed, canalGreen, canalBlue);

```

As imagens são carregadas nas linhas 1 a 4. As linhas 7, 8 e 9 são responsáveis por separar os canais RGB da imagem patch que não será modificada. Já as linhas 12, 13 e 14 separam os canais RGB da imagem patch que será modificado seu histograma. Feito isso, é chamada a função do Recozimento simulado para cada canal do patch (linhas 16, 17 e 18), no qual retorna uma nova imagem. Ao final, todos as novas imagens dos canais devem ser juntadas (linha 21). As separações dos canais também são aplicadas as imagens com tamanhos originais (image1 e image2).

Antes de iniciar o algoritmo do Recozimento simulado, deve ser definido um estado por um conjunto de 4 pontos posicionados sobre uma grade 256×256 que representa um polinômio de grau 3 que passa por cada um dos 4 pontos. Considere o código a seguir:

```

1 %encontra 4 pontos em x e 4 em y
2 intervaloX = 84;
3 intervaloY = 85;

```

```

4   pontos = pontos2 = 1;
5
6   for i = 1: 4
7     pontosX(i) = pontos;
8     pontosY(i) = pontos2;
9     pontos += intervaloX;
10    pontos2 += intervaloY;
11
12 end

```

As ações das linhas 6 a 11 são responsáveis por definir 4 coordenadas, que apresentam uma diferença de 84 para os pontos x e 85 para os y, ou seja, as coordenadas em x serão 1, 85, 169, 153; e em y 1, 86, 171, 256. Essas coordenadas são aplicadas na equação de grau 3 para obter 4 coeficientes, que são os 4 pontos responsáveis por mapear a nova intensidade da imagem. A seguir é descrita a função `resolvSistema` que encontra esses coeficientes:

```

1 function [I] = ResolvSistema(pontosX, pontosY)
2   x1 = pontosX(1);
3   x2 = pontosX(2);
4   x3 = pontosY(3);
5   x4 = pontosY(4);
6
7   y1 = pontosY(1);
8   y2 = pontosY(2);
9   y3 = pontosY(3);
10  y4 = pontosY(4);
11
12 A = [x1^3, x1^2, x1, 1; x2^3, x2^2, x2, 1; x3^3, x3^2, x3, 1; x4
13   ^3, x4^2, x4, 1];
14 B = [y1; y2; y3; y4];
15 [I] = A\B;
16
17 end

```

As linhas 2 a 10 separam cada um das coordenadas em variáveis diferentes. Já as linhas 12, 13 e 14 resolvem a equação de grau 3 (Equação 1) de acordo com a especificação do Octave. O objetivo é substituir os valores de x e y na equação e encontrar os coeficientes a, b, c e d.

$$ax^3 + bx^2 + cx + d = y \quad (1)$$

Além da definição dos 4 pontos, também é criada uma função denominada `DifHist` responsável por calcular a diferença entre dois histogramas:

```

1 function [porcentagemDif] = DifHist(hist1, hist2)
2   sub = hist2 - hist1;

```

```

3 quadrado = sub.^2;
4 porcentagemDif = sum(quadrado);
5
6 end

```

A DifHist retorna um valor que determina a porcentagem de quanto um histograma se diferencia do outro. Isso é calculado por meio da soma dos quadros da diferença dos histogramas (linhas 2, 3 e 4 do código).

Também é criada a função CalculaNovaIntensidade responsável por gerar uma nova imagem por meio dos coeficientes obtidos aplicados à Equação1, ela é descrita a seguir:

```

1 function img2 = CalculaNovaIntensidade(img2, Coord)
2
3 %Calcular as novas intensidades
4 for i=1: size(img2)
5     for j=1: size(img2)
6         novaIntensidade = ( Coord(1)*(img2(i,j)^3)) + (
7             Coord(2)*(img2(i,j)^2)) + (Coord(3)* img2(i,j)
8             ) + (Coord(4) );
9         img2(i, j) = novaIntensidade;
10    end
11 end

```

As linhas 4, 5, 6 e 7 do código percorrem toda a imagem pegando em cada posição o valor de intensidade e o substituindo por um novo valor de acordo com o resultado da equação, que recebe os coeficientes (a, b, c e d) e os valores das intensidades.

O algoritmo do Recozimento simulado é uma busca que começa em um estado inicial escolhido aleatoriamente, como mostra parte do código a seguir:

```

1 (... )
2 temperatura = 100;
3 i = 0;
4 K = 5;
5 contIndiceVizinho = 1;
6
7 while(temperatura > 0.00001 )
8     % Verificar se decrementa temperatura
9     if i == K % Se a mesma temperatura rodou K vezes,
10        decrementar ela
11        temperatura = temperatura - 4;
12        i = 0;
13    endif

```

13
14 (...)

A temperatura inicial é 100 (linha 1), as variáveis *i* e *K* (linhas 2 e 3) são contadores que determinam a quantidade de vezes que a mesma temperatura será utilizada, como mostra da condição das linhas 7, 8 e 9, quando o *i* for igual a 5, a temperatura é decrementada 4 vezes, e *i* volta para 0. Isso é repetido enquanto o valor da temperatura for maior que 0.00001.

A cada iteração, o algoritmo faz uma transição para um estado vizinho aleatório, como demonstra a seguir:

```
1     ( ... )
2     % Pegar os pontos atuais para encontrar quatro novos
3         vizinhos
4         novoX1 = pontosX(contIndiceVizinho);
5         novoX2 = pontosX(contIndiceVizinho);
6         novoY1 = pontosY(contIndiceVizinho);
7         novoY2 = pontosY(contIndiceVizinho);
8
9         % verificar se os polinomios sao crescentes
10         % verificar se os vizinhos sao validos
11         if (pontosX(contIndiceVizinho)+1 >= 1 && pontosX(
12             contIndiceVizinho)+1 <= 256) && (pontosX(
13             contIndiceVizinho)+1 > pontosX(contIndiceVizinho))
14             novoX1 = pontosX(contIndiceVizinho)+1;
15         endif
16         if (pontosX(contIndiceVizinho)-1 >= 1 && pontosX(
17             contIndiceVizinho)-1 <= 256)
18             novoX2 = pontosX(contIndiceVizinho)-1;
19         endif
20         if (pontosY(contIndiceVizinho)+1 >= 1 && pontosY(
21             contIndiceVizinho)+1 <= 256) && (pontosY(
22             contIndiceVizinho)+1 > pontosY(contIndiceVizinho))
23             novoY1 = pontosY(contIndiceVizinho)+1;
24         endif
25         if (pontosY(contIndiceVizinho)-1 >= 1 && pontosY(
26             contIndiceVizinho)-1 <= 256)
27             novoY2 = pontosY(contIndiceVizinho)-1;
28
29         %Escolher aleatoriamente um dos vizinhos para
30             trabalhar
31         r = randi([1, 4]);
32         %calcular a diferenca dos histogramas para o vizinho
33             escolhido
34         testeVizinhoX = pontosX;
```

```

27     testeVizinhoY = pontosY;
28
29     % Substituir o ponto atual pelo ponto do vizinho
      escolhido
30     if r == 1
31         testeVizinhoX(contIndiceVizinho) = novoX1; %(x+1, y
            )
32     elseif r == 2
33         testeVizinhoX(contIndiceVizinho) = novoX2; %(x-1, y
            )
34     elseif r == 3
35         testeVizinhoY(contIndiceVizinho) = novoY1; %(x, y
            +1)
36     elseif r == 4
37         testeVizinhoY(contIndiceVizinho) = novoY2; %(x, y
            -1)
38     endif
39
40 [Coord] = ResolvSistema(pontosX, pontosY);
41 %Mapear a intensidade
42 novaImagem = CalculaNovaIntensidade(img2, Coord);
43 D = DifHist(hist1, imhist(novaImagem));
44
45 %Calcular a diferenca hist para o vizinho escolhido
46 %resolver o sistema
47 [CoordVizinho] = ResolvSistema(testeVizinhoX,
        testeVizinhoY);
48 %Mapear a intensidade
49 novaImagemVizinho = CalculaNovaIntensidade(img2,
        CoordVizinho);
50 DVizinho = DifHist(hist1, imhist(novaImagemVizinho));
51 (...)
```

As linhas 3, 4, 5, e 6 selecionam os quatro possíveis vizinhos de um dos 4 pontos escolhido pelo `contIndiceVizinho` ($x+1$, $x-1$, $y+1$ e $y-1$). Existem algumas condições que devem ser verificadas para determinar se os pontos x e y dos vizinhos são válidos. A primeira é que os pontos não podem ser menor que 1 ou maior que 256, pois irão ultrapassar o limite da matriz. Já a segunda garante que o polinômio seja monotonicamente crescente, de forma que os pontos incrementados em +1 devem ser maiores que os pontos atuais. Essas verificações são realizadas nas condições das linhas 10, 13, 16 e 19.

Os 4 vizinhos devem ser aleatorizados para a escolha de apenas um deles, como mostra na linha 24 que escolhe um valor aleatório, e nas linhas 30 a 37 que define o ponto do vizinho de acordo com esse valor.

Sabendo as coordenadas do vizinho e a atual, aplica-se a função `ResolveSistema` para encontrar os coeficientes, a função `CalculaNovaIntensidade` para gerar uma

nova imagem e a função DifHist para determinar a diferença das duas coordenadas. As linhas 40, 42 e 43 realizam essa ação para os pontos x e y atuais, e as linhas 47, 49 e 50 para os pontos x e y do vizinho escolhido.

Com as duas diferenças, é calculado uma variação entre elas com objetivo de determinar se os pontos do vizinho serão aceitos. Se essa variação for menor que 0 eles são aceitos, caso contrário é calculada uma função dependente da temperatura. Considere a continuação do código do SimulatedAnnealing a seguir:

```

1 (...) 
2 % Calcular a diferenca das duas diferencias dos
   histogramas (atual e vizinho)
3 deltaD = DVizinho - D;
4
5 % Verificar o valor das diferencias para decidir se
   aceita ou nao a nova diferenca
6 if deltaD <= 0 % novoD menor que a diferenca vizinho,
   entao aceita
7   porcentagem = 1;
8 else % calcula a porcentagem de aceitacao
9   % expoencial no octave e = 2.7183
10  resolv = -deltaD/temperatura;
11  porcentagem = 2.7183^(resolv);
12 endif
13
14 %Verificar se a porcentagem sera aceita ou nao
15 p = rand(1); %Sorteia qualquer numero entre 0 e 1
16
17 if (porcentagem == 1) || (porcentagem > p) %aceita
   pontos do vizinho
18   pontosX = testeVizinhoX;
19   pontosY = testeVizinhoY;
20 endif
21 (...)
```

A variação é calculada na linha 2. Já as verificações da variação são feitas nas linhas 5 e 6. Se ela for maior que zero, é calculado a exponencial da variação negativa dividida pela temperatura. Esse cálculo irá retornar um número qualquer entre 0 e 1. Para saber se o resultado desse cálculo será aceito, aleatoriza-se um número entre 0 e 1 (linha 14) e aceita quando ele for maior (linhas 16, 17 e 18).

A última parte da Temperatura simulada é gerar modificar por meio dos últimos pontos encontrados. Isso é feito no código a seguir:

```

1 (...) 
2 % Calcular a img2 final por meio da ultima curva obtida
3 %resolver o sistema
4 [Coord] = ResolvSistema(pontosX, pontosY);
```

```

5 %Mapear a intensidade
6 img2 = CalculaNovaIntensidade(img2, Coord);
7
8 canalRGB = img2;
9 (...)
```

A linha 4 encontra os coeficientes dos pontos, e a linha 6 gera a nova imagem baseado nesses coeficientes. Por fim, a função retorna essa nova imagem gerada (linha 6).

A Result

0.1 Result 1

O primeiro resultado foi a aplicação do algoritmo *Simulated Annealing* para que o histograma da imagem patch 2 ([3, Direita](#)) se aproximasse do histograma da imagem patch 1 ([3, Esquerda](#)).

Os resultados obtidos foram satisfatórios. Quando analisado os histogramas de cada canal RGB das imagens patch, observou-se que os histogramas da imagem patch 2 modificadas se diferenciaram dos histogramas da imagem patch 2. As Figuras [4](#), [5](#) e [6](#), comparam os histogramas de cada canal das imagens patch 2 original e modificada.

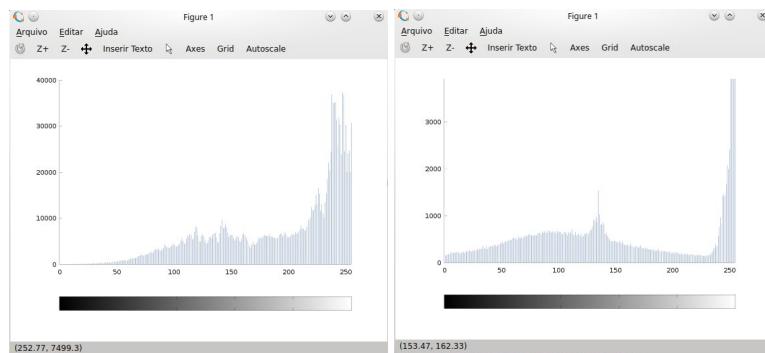


Figure 4: Esquerda: Histograma do canal vermelho da imagem patch 2. *Direita:* Histograma do canal vermelho da imagem patch 2 modificada.

A Figura [7](#) mostra os histogramas dos três canais da imagem patch 1, no qual os histogramas das imagens patch 2 modificadas deveriam se aproximar. Foi possível observar que essa aproximação não foi muito grande, porém, os histogramas do patch 2 modificados se aproximam muito mais do patch 1, do que apenas os patch 2 originais.

Após a junção de cada canal RGB, foi possível analisar a imagem gerada. A Figura [8](#) compara a imagem patch 2 original com a patch 2 modificada.

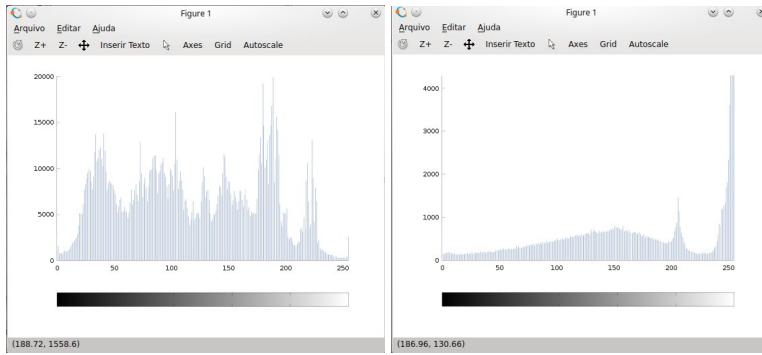


Figure 5: *Esquerda*: Histograma do canal verde da imagem patch 2. *Direita*: Histograma do canal verde da imagem patch 2 modificada.

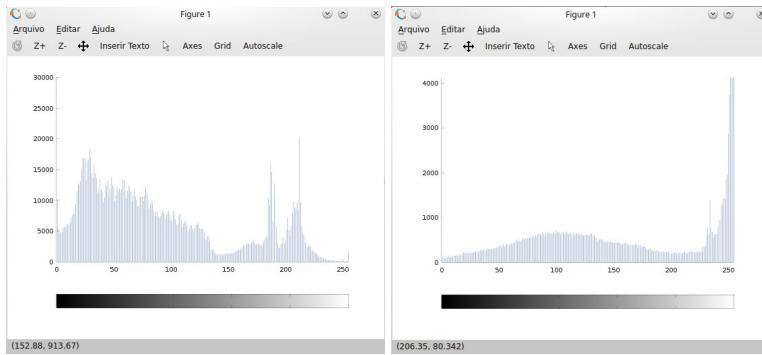


Figure 6: *Esquerda*: Histograma do canal azul da imagem patch 2. *Direita*: Histograma do canal azul da imagem patch 2 modificada.

Assim, foi possível concluir que a aproximação não foi tão grande devido ao valor inicial da temperatura (100). Quanto maior o valor, maior é a aproximação dos histogramas e melhor o resultado, porém, maior é o tempo de execução do algoritmo.

0.2 Result 2

O segundo resultado foi a aplicação da função (pontos x e y) encontrada pelo algoritmo do *Simulated Annealing* para gerar uma nova imagem para cada canal RGB da imagem 2 ([2, Direita](#)).

As Figuras [9](#), [10](#) e [11](#) comparam os histogramas de cada canal das imagens 2 original e modificada. Já a Figura mostra os histogramas dos três canais da imagem 1, no qual os histogramas das imagens 2 modificadas deveriam se aproximar.

Após a junção de cada canal RGB, foi possível analisar a imagem gerada. A Figura [13](#) compara a imagem 2 original com a 2 modificada.

Foi possível concluir que na imagem de tamanho original não houve tanta diferença, pois os pontos X e Y não foram os mesmos utilizados para imagem patch, pois esses

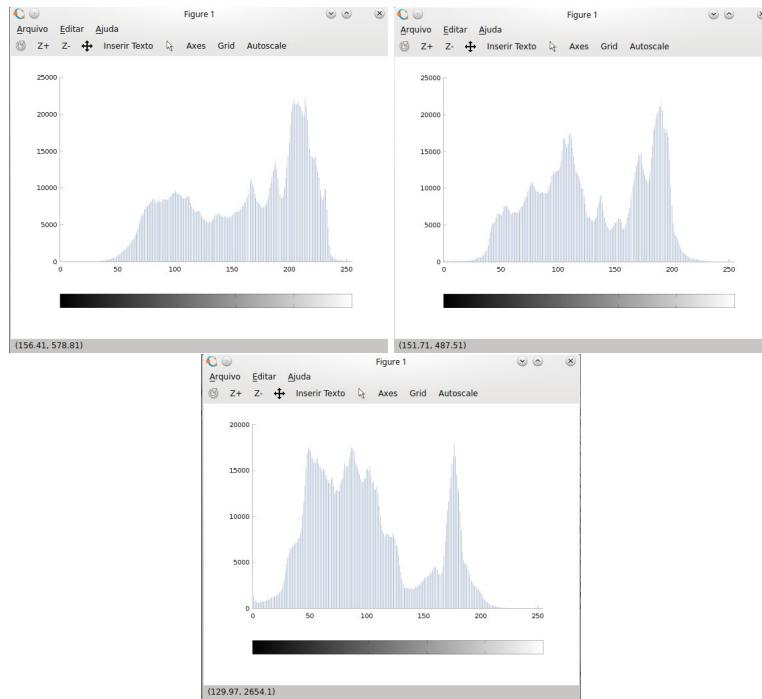


Figure 7: *Esquerda*: Histograma do canal vermelho da imagem patch 1. *Direita*: Histograma do canal verde da imagem patch 1. *Centro*: Histograma do canal azul da imagem patch 1.

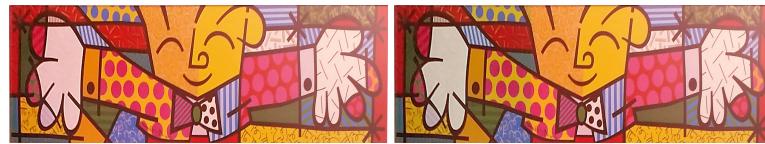


Figure 8: *Esquerda*: Imagem patch 2 original. *Direita*: Imagem patch 2 modificada.

pontos foram perdidos e o algoritmo teve que ser executado novamente para a obtenção de um novo ponto, para poder aplicar na imagem 2.

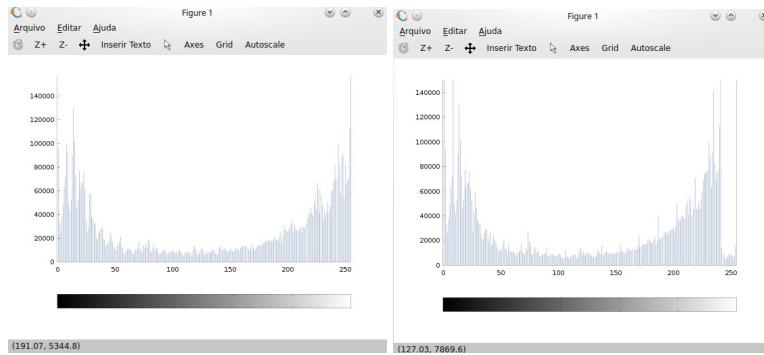


Figure 9: *Esquerda*: Histograma do canal vermelho da imagem 2. *Direita*: Histograma do canal vermelho da imagem 2 modificada.

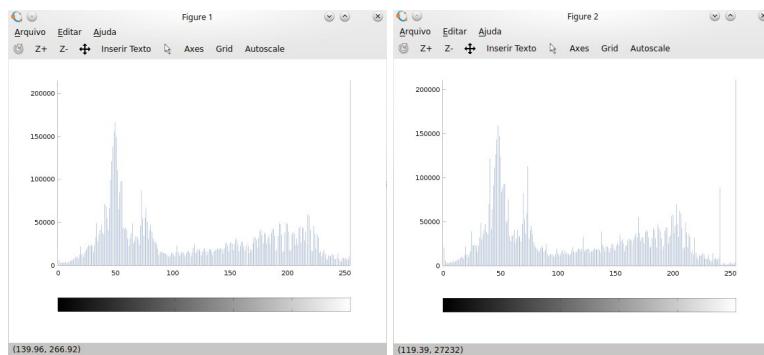


Figure 10: *Esquerda*: Histograma do canal verde da imagem 2. *Direita*: Histograma do canal verde da imagem 2 modificada.

References

- [engcar 2018] ENGCAR. *O Histograma de uma Imagem*. 2018. <http://www.ufrrgs.br/engcart/PDASR/hist.html>. Acessado em 16-05-2018.
- [Pratschke e Pelizzoni 1999] PRATSCHKE, A.; PELIZZONI, J. M. *Simulated Annealing*. 1999. http://conteudo.icmc.usp.br/pessoas/sandra/G9_t2/annealing.htm. Acessado em 16-05-2018.
- [Wikipedia 2018] WIKIPEDIA. *Histogram matching*. 2018. https://en.wikipedia.org/wiki/Histogram_matching. Acessado em 16-05-2018.

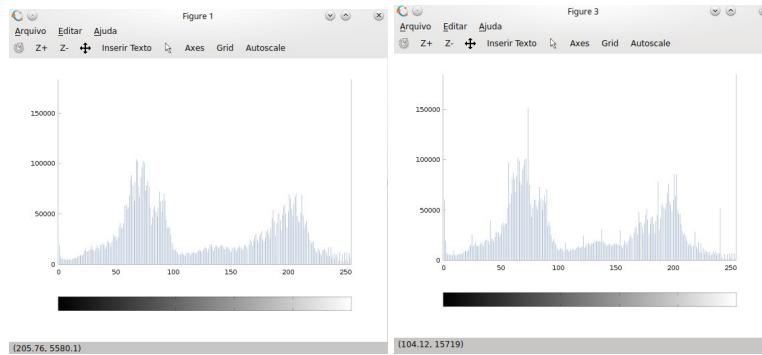


Figure 11: *Esquerda*: Histograma do canal azul da imagem 2. *Direita*: Histograma do canal azul da imagem 2 modificada.

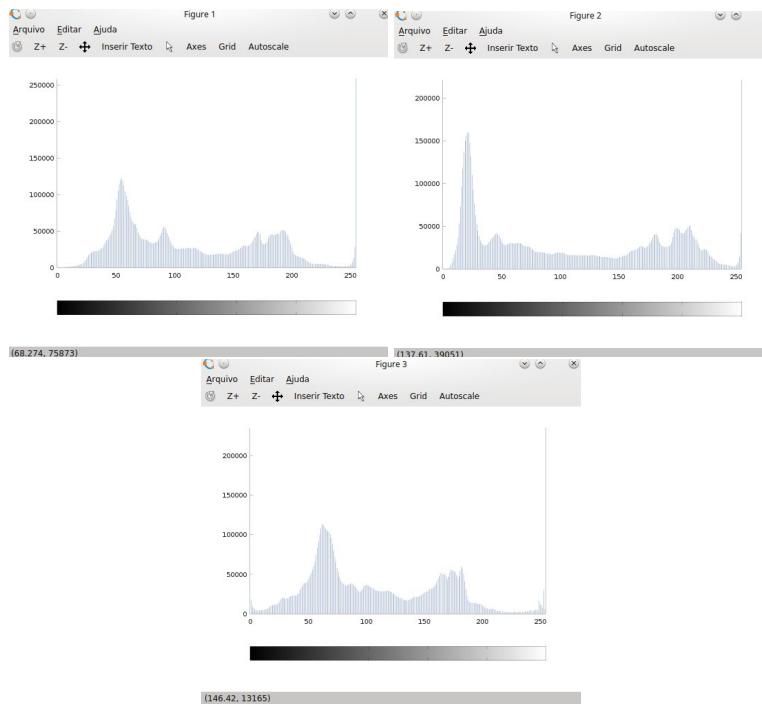


Figure 12: *Esquerda*: Histograma do canal vermelho da imagem 1. *Direita*: Histograma do canal verde da imagem 1. *Centro*: Histograma do canal azul da imagem 1.



Figure 13: *Esquerda*: Imagem 2 original. *Direita*: Imagem 2 modificada.