Processamento de Imagens
Melhorias de contraste

Vinicius Gasparini[1]

[1]Departamento de Ciência da Computação
Universidade do Estado de Santa Catarina (UDESC)
Centro de Ciências Tecnológicas – Joinville – SC – Brasil

v.gasparini@edu.udesc.br

1. Questão A

Para a solução desta tarefa foi necessário segmentar o algoritmo em 2 etapas:

1. Extrair da legenda do gráfico a escala para o intervalo de cinzas
2. Percorrer a imagem em tons de cinza aplicando a transformação de cor nos *pixels* de cor superior ao limite definido pelo usuário

Etapa 1

Para a tarefa 1 utilizou-se de uma simples função (Algoritmo 1) que percorre todos os *pixels* da imagem realizando max e min. Ao fim da execução, teremos o pixel mais "preto" e o mais "branco", limite inferior e superior da escala respectivamente.

```
def get_scale(img):
    """
    Funcao que recebe como parametro uma imagem
    em formato de matriz RGB.
    Durante a execucao, convertemos para escala
    de cinza [0,1] e ent o coletamos o maior valor para
    limite superior da escala e o menor para limite inferior
    """
    mini, maxi = 1, 0
    h, w = len(img), len(img[0])
    for i in range(h):
        for j in range(w):
            pixel_gray = rgb_to_gray(img[i][j])
            maxi = max(maxi, pixel_gray)
            mini = min(mini, pixel_gray)
    return mini, maxi


def rgb_to_gray(rgb):
    """
    Transformacao para escala de cinza
    """
    return np.dot(rgb[..., :3], [0.2989, 0.5870, 0.1140])
```

Algoritmo 1. Extração da escala.

Como é possível observar, fez-se necessário converter o *pixel* RGB para escala de cinza através da função *rgb_to_gray*. Essa decisão se deu para facilitar lidar com os 4 canais de (RGBA) posteriormente.

Etapa 2

Com os limites da escala em mãos então aplicaremos o filtro de cor. Para tal, foi adaptado de uma função de limiar para que, caso o pixel analisado no momento esteja fora do limite definido ele seja transformado em vermelho. Este comportamento está descrito no Algoritmo 2.

```
def apply_scale(value, bottom, top):
  """
  Aplica a normalizacao da escala levando em conta
  o limite inf e sup
  """
  return 224 * (value - bottom) / (top - bottom)


def apply_threshold(img, threshold, scale):
  """
  Fun   o de filtro
  Convertemos pixel a pixel em escala de cinza,
  aplicamos a normalizacao e pintamos de vermelho
  caso seja maior que o valor informado e menor que
  224. Isso eh necessario para ignorar branco
  """
  h, w = len(img), len(img[0])
  for i in range(h):
      for j in range(w):
        pixel_gray_scale = rgb_to_gray(img[i][j])
        pixel_in_scale = apply_scale(pixel_gray_scale,
                                     scale[0], scale[1])
        if threshold < pixel_in_scale < 224:
          img[i][j] = [1, 0, 0, 1]
  return img
```

Algoritmo 2. Realiza processo de coloração.

A função *apply_threshold* itera sob todos os *pixels* convertendo-os para tons de cinza. Esse pixel em tons de cinza é um real dentro do intervalo [0,1], ele então passa por uma transformação na função *apply_scale* para ser normalizado dentro do intervalo [0,224]. Caso este valor fique entre o limiar definido pelo usuário e o máximo da nossa escala, ele é pintado de vermelho. Os resultados obtidos podem ser consultados na Figura 1

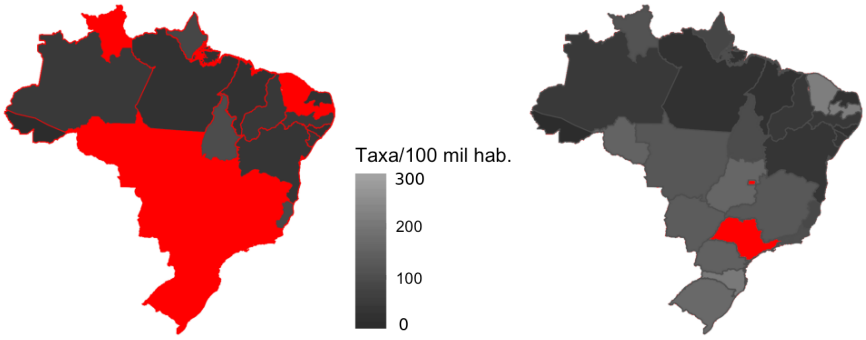

Figura 1. Na esquerda, destacando estado com homicídio acima de 50/100 mil hab. Na direita, acima de 150/100 mil hab. conforme escala.

2. Questão B

Como é possível observar na Figura 2, percebe-se que após a aplicação da fórmula de equalização descrita por [Gonzalez and Woods 2010], os *pixels* mais claros ficaram ainda mais claros, passando de um máximo em torno de 150 para 255. Em contrapartida os *pixels* mais escuros ficaram ainda mais escuros, saindo de um mínimo em torno de 120 para próximo de 0. Estratégia utilizada pode ser consultada no Algoritmo 3.

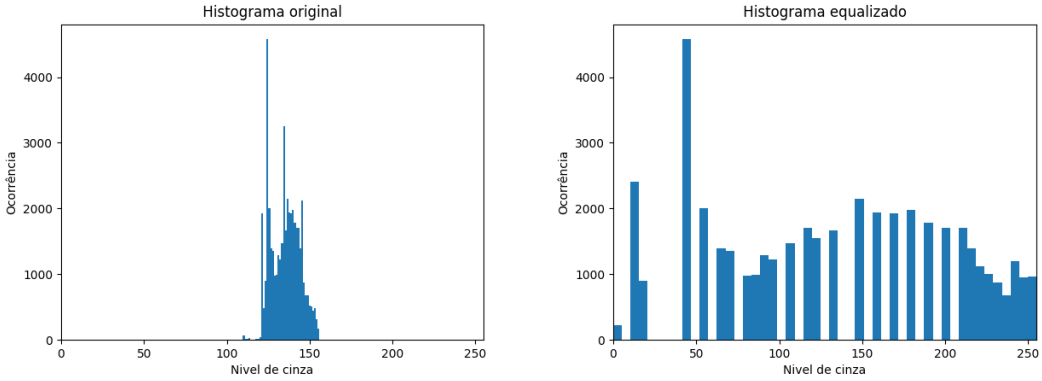

Figura 2. Na esquerda, histograma antes da equalização. A direita, após equalização.

```
qtd_niveis = len(niveis)
mn = img.size[0] * img.size[1]
equalizado = niveis.copy()
nk = 0
for i in range(0, 256):
    # Nk acumulado
    nk += niveis.count(i)
    # Probabilidade de ocorr ncia
    eq = (255 / mn) * nk
    for j in range(qtd_niveis):
        # Buscando tom cinza mais pr ximo
        if equalizado[j] == i:
            equalizado[j] = eq
```

Algoritmo 3. Equalização por Probabilidade.

3. Questão C

Neste exercício foi requisitado equalizar a imagem da biblioteca da disciplina *outono.png* de duas maneiras diferentes:

1. Equalização da imagem sob os canais RGB
2. Equalização da imagem sob o canal Y do padrão YIQ

3.1. Equalização por RGB

Como já discutido em outras oportunidades, a equalização do histograma por RGB pode resultar no aparecimento de cores que antes não estavam presentes na imagem pois os canais RGB são fortemente correlacionados. Aplicando a equalização obtemos os seguintes resultados [Figura 3]. As anomalias de cor podem ser observadas na histograma da Figura 4 resultante da equalização.

Figura 3. Na esquerda, a imagem original. A direita, a imagem equalizada sobre os canais RGB.

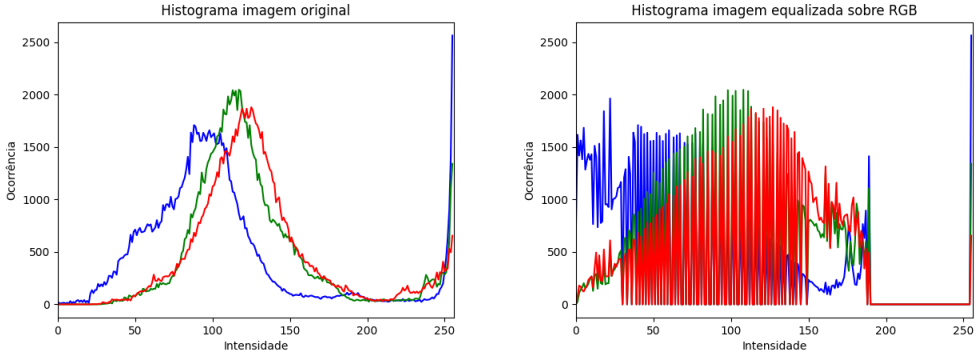

Figura 4. Na esquerda, a histograma da imagem original. A direita, histograma da imagem equalizada sobre os canais RGB.

3.2. Equalização pelo canal Y

Conforme descrito por [Pedrini 2007], o canal Y se refere a característica acrômica da imagem, isto é, indica a intensidade da informação crômica, descrita pelo par [I,Q], no referido pixel.

```
# Convertendo imagem RGB para YIQ
img_yiq = color.rgb2yiq(img[:, :, :3])

# Extraindo o canal de luminancia Y
canal_y = img_yiq[:, :, 0]

# Equalizando o canal Y
canal_y_eq = exposure.equalize_hist(canal_y)

# Retornando a img_yiq o canal Y j   equalizado
img_yiq[:, :, 0] = canal_y_eq

# Convertendo de YIQ para RGB
hist_equalization_result = color.yiq2rgb(img_yiq)
```

Algoritmo 4. Equalização por Probabilidade

Conforme descrito no Algoritmo 4, o procedimento utilizado compreende de converter a imagem RGB para YIQ e então isolar o canal Y para tratamento. Sob posse do canal Y, ele é equalizado e por fim substituído na nossa matriz descritora da imagem YIQ. Os resultados obtidos podem ser consultados nas Figuras 5 e 6

Figura 5. Na esquerda, a imagem original. A direita, a imagem equalizada pelo canal Y.

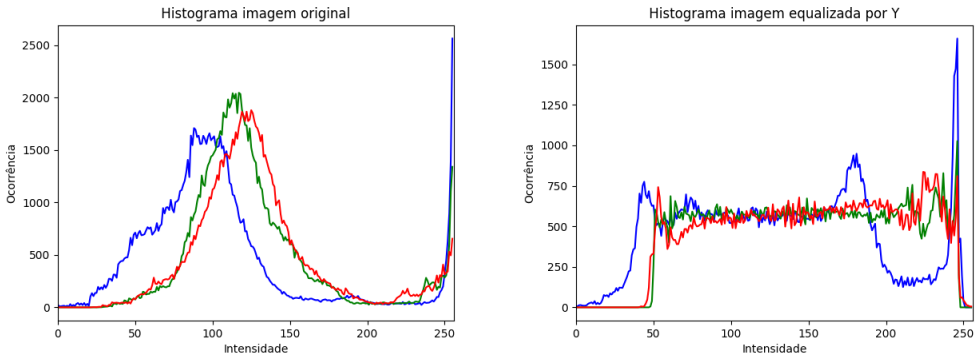

Figura 6. Na esquerda, a histograma da imagem original. A direita, histograma da imagem equalizada sobre o canal Y.

Referências

[Gonzalez and Woods 2010] Gonzalez, R. and Woods, R. (2010). *Processamento digital de Imagens*. Editora Pearson.

[Pedrini 2007] Pedrini, H. (2007). *Análise de Imagens Digitais - Princípios, Algoritmos e Aplicações*. Editora Thomson Learning.