****

**UNIVERSIDADE FEDERAL DE LAVRAS-UFLA**

**DISCIPLINA PGM848- VISÃO COMPUTACIONAL NO MELHORAMENTO DE PLANTAS**

**CAROLINE MARCELA DA SILVA**

**EWERTON LÉLYS RESENDE**

**MARIANA ANDRADE DIAS**

**THIAGO TAVARES BOTELHO**

**PROCESSAMENTO DE IMAGENS EM PYTHON- REO2**

Relatório apresentado como requisito avaliativo da disciplina de Visão Computacional no Melhoramento de Plantas, do Programa de Pós Graduação em Genética e Melhoramento de Plantas.

Prof. Dr. Vinícius Quintão Carneiro

Lavras-MG

Julho/2020

**RESULTADOS- EXERCÍCIO 01:**

**Selecione uma imagem a ser utilizada no trabalho prático e realize os seguintes processos utilizando o pacote OPENCV do Python:**

**a) Apresente a imagem e as informações de número de linhas e colunas; número de canais e número total de pixels;**

A imagem selecionada para nossa análise está representada na figura 1. Para apresentar essa imagem no ambiente Python, criamos uma variável “arquivo” para receber o seu arquivo png/jpg/jpeg. A leitura desta variável foi realizada com a função “read” do pacote “opencv”, usando como argumento a variável “arquivo” e o valor “1”, correspondente ao tipo de imagem utilizada, no caso colorida.

|  |
| --- |
| arquivo = ("trabalho.jpeg") imagem = cv2.imread(arquivo,1) print(imagem) |

A imagem corresponde a uma matriz bidimensional de valores inteiros, e sua leitura, pela função mencionada anteriormente, é feita dessa forma. Cada valor da matriz corresponde a um pixel da imagem, e pode ser convertido em uma intensidade de cor. Os valores de intensidade de cor existentes em cada pixel são chamados de canais, e ajudam, juntamente com outros parâmetros como número de linhas e número de colunas, caracterizar a imagem. Para o cálculo do número de linhas, colunas e canais da matriz, utilizamos a função “np.shape”. O número de pixel da imagem é dado pela multiplicação do número de linhas pelo número de colunas, capturando o número de posições que compõem a matriz.

|  |
| --- |
| nl,nc,canais = np.shape(imagem)  print('Número de linhas: ' + str(nl))  print('Número de colunas: ' + str(nc))  print('Número de canais: ' + str(canais))  numero\_pixels=nl\*nc  print('O número de pixels é:'+ str(numero\_pixels)) |

Dessa forma, a matriz da nossa imagem está apresentada na figura 2, e sua caracterização na figura 3.

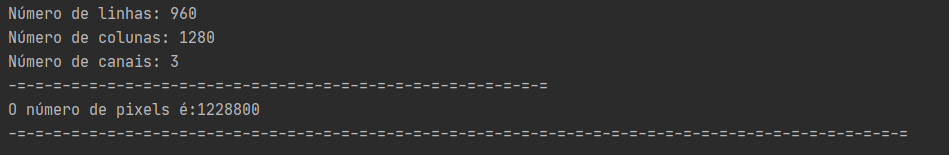
Figura 1: Imagem escolhida.



Figura 2: Matriz da imagem.

|  |
| --- |
| [[[ 94 80 57]  [ 94 80 57]  [ 94 80 57]  ...  [204 204 204]  [204 204 204]  [204 204 204]]  [[ 93 79 56]  [ 92 78 55]  [ 92 78 55]  ...  [203 203 203]  [204 204 204]  [204 204 204]]  [[ 94 78 55]  [ 94 78 55]  [ 93 77 54]  ...  [203 203 203]  [204 204 204]  [204 204 204]]  ...  [[183 183 183]  [184 184 184]  [184 184 184]  ...  [168 167 169]  [168 167 169]  [169 168 170]]  [[183 183 183]  [184 184 184]  [184 184 184]  ...  [168 167 169]  [168 167 169]  [169 168 170]]  [[183 183 183]  [184 184 184]  [184 184 184]  ...  [168 167 169]  [168 167 169]  [168 167 169]]] |

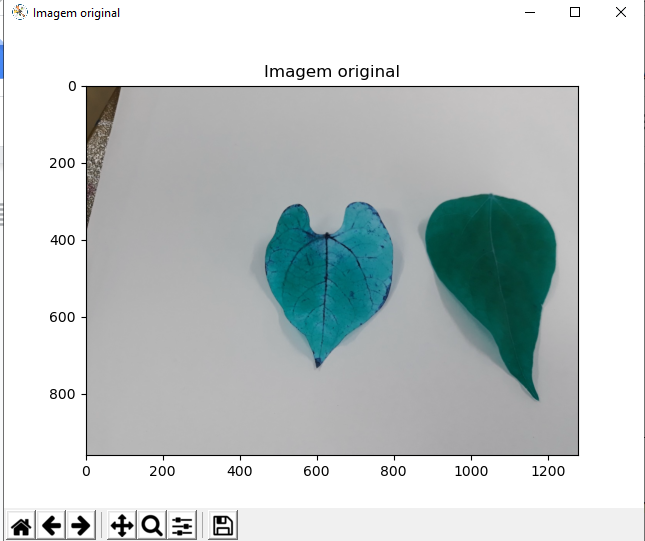
Figura 3: Parâmetros de caracterização da matriz.

****

Além da apresentação da imagem, na forma matricial, nós utilizamos as funções “plt.” para plotar a imagem da figura 1 no console. Com a função “plt. figure” criamos a figura “Imagem original”, “plt.imshow” apresentamos a imagem, demos um título (“plt.title”) e plotamos a figura com o comando “plt.show”.

|  |
| --- |
| plt.figure('Imagem original') plt.imshow(imagem) plt.title("Imagem original")  plt.show() |

Figura 4: Gráfico com a imagem escolhida.



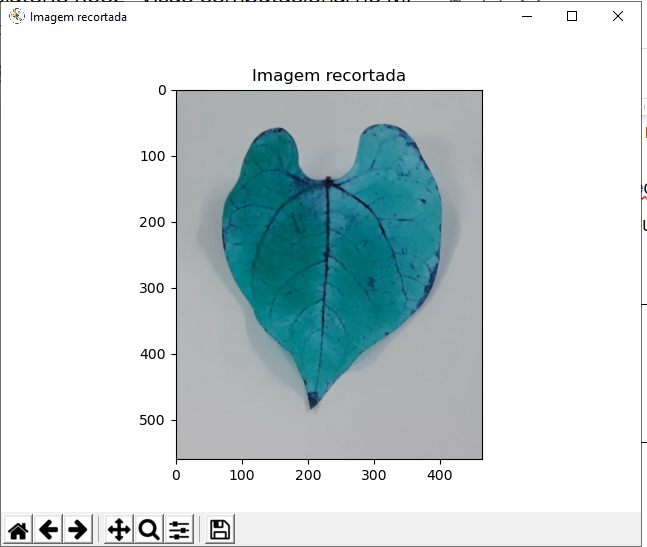
**b) Faça um recorte da imagem para obter somente a área de interesse. Utilize esta imagem para a solução das próximas alternativas;**

Para selecionar a área que desejamos cortar da imagem original (Figura 1), utilizamos o gráfico gerado na questão anterior “Imagem original” (Figura 4), a qual tem as coordenadas de linhas e colunas nos eixos y e x, respectivamente.

Em seguida, criamos uma nova variável “img\_recorte” para receber a nova imagem, que foi cortada das linhas 250 a 810 e das colunas 395 a 860 da imagem original. A nova imagem está representada na figura 5.

|  |
| --- |
| img\_recorte = imagem[250:810,395:860]  plt.figure('Imagem recortada') .  plt.imshow(img\_recorte)  plt.title("Imagem recortada")  plt.show() |

Figura 5: Imagem recortada.



**c) Converta a imagem colorida para uma de escala de cinza (intensidade) e a apresente utilizando os mapas de cores “Escala de Cinza” e “JET”;**

O formato RGB consiste em um espaço de cor, ou seja, uma forma de representar as cores por meio de dispositivos digitais. Este formato abrange três cores principais (canais), que são o vermelho (Red- R), verde (Green- G) e o azul (Blue- B), que quando sobrepostas nos permite formar todas as outras cores da imagem. Na prática, as três cores são representadas por três matrizes, na ordem anteriormente mencionada.

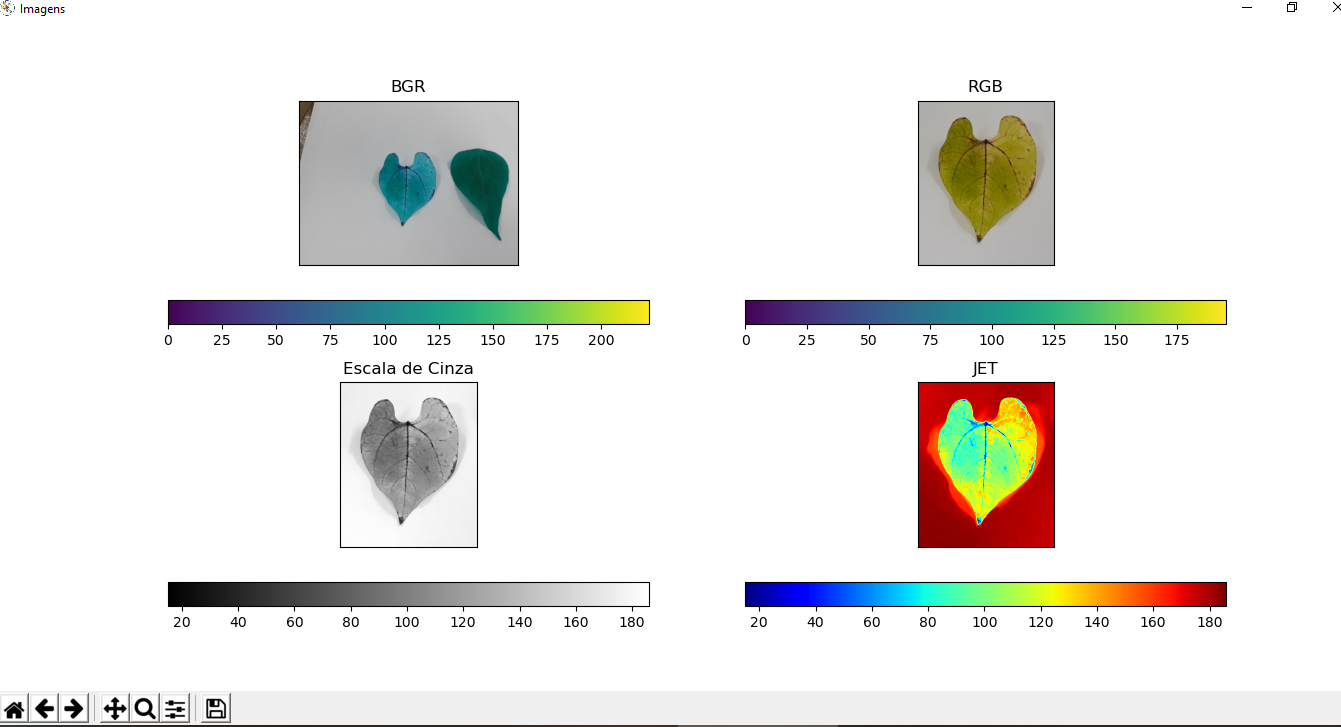
Entretanto, quando carregamos a imagem com o Opencv, há uma inversão na ordem das matrizes, permitindo primeiro a sobreposição das cores azul, verde e posteriormente, vermelho, ou seja, tornando a imagem BGR. Com isso, para melhor visualização durante o trabalho, fizemos a conversão do formato BGR para o RBG a partir da função “cv2.cvtColor” usando a imagem recortada “img\_recorte” e o tipo de conversão, no caso de BGR para (*to* - 2) RGB (“cv2.COLOR\_BRG2RBG”), como argumentos. A conversão também foi feita de BGR para escala de cinza, argumento “GRAY”. Para a escala JET, usamos a imagem convertida para escala de cinza “img\_cinza” e o parâmetro “cmap=jet” aplicados a função “imshow”, quando plotamos o gráfico.

|  |
| --- |
| img\_rgb = cv2.cvtColor(img\_recorte, cv2.COLOR\_BGR2RGB)  img\_cinza = cv2.cvtColor(img\_recorte,cv2.COLOR\_BGR2GRAY) |

A partir de então, criamos uma figura “Imagens” para plotar a imagem original, a recortada RGB, a imagem recortada em escala de cinza e a imagem recortada em escala JET (Figura 6). Para plotar todas as imagens citadas anteriormente em uma só figura, utilizamos a função “plt.subplot” e como argumento, determinamos que seria uma figura 2x2, ou seja, com quatro quadrantes para ocupação. Os comandos para a plotagem a partir da função “plt.” foram semelhantes ao descrito na questão A, com diferença que a cada imagem pedida, indicamos a localização da mesma na figura, a partir do último argumento da função “plt.subplot”(indicado pela cor vermelha na figura 7). Por exemplo, a terceira imagem da figura, é a figura recortada em escala de cinza, para sua plotagem utilizamos os comandos listados a seguir.

|  |
| --- |
| img\_cinza = cv2.cvtColor(img\_recorte,cv2.COLOR\_BGR2GRAY)  plt.subplot(22**3**)  plt.imshow(img\_cinza, cmap = 'gray')  plt.title("Escala de Cinza")  plt.xticks([])  plt.yticks([])  plt.colorbar(orientation='horizontal') |

Figura 6: Subplot Imagens.



Como o olho humano não é capaz de diferenciar todos os tons de cinza, a visualização da imagem com a escala JET permite identificar os diferentes tons presentes. Para esta escala, os valores 0 e 255 são atribuídos para as cores azul e vermelho, respectivamente, enquanto todas as outras cores transitam dentro dessa faixa. Para a escala de cinza o valor 0 é atribuído ao preto e o valor 255 ao branco.

A utilização de diferentes escalas de cores nas imagens consiste em diferentes modos de observação da intensidade dentro da imagem, e de certa forma, isso se torna útil para identificação objetos nas imagens.

**d) Apresente a imagem em escala de cinza e o seu respectivo histograma; Relacione o histograma e a imagem.**

A imagem convertida para escala de cima, foi feita na questão anterior, e armazenada na variável “img\_cinza”.

O histograma foi realizado com a função “cv2.calcHist”, usando como argumentos:

- a imagem em tons de cinza, “img\_cinza”;

- número de canais, que no caso é “0”, por ser uma única matriz que estamos acessando com essa tonalidade;

- presença de máscara, que é uma identificação de uma região específica dentro da imagem, no caso, não foi o nosso objetivo, então usamos “none”;

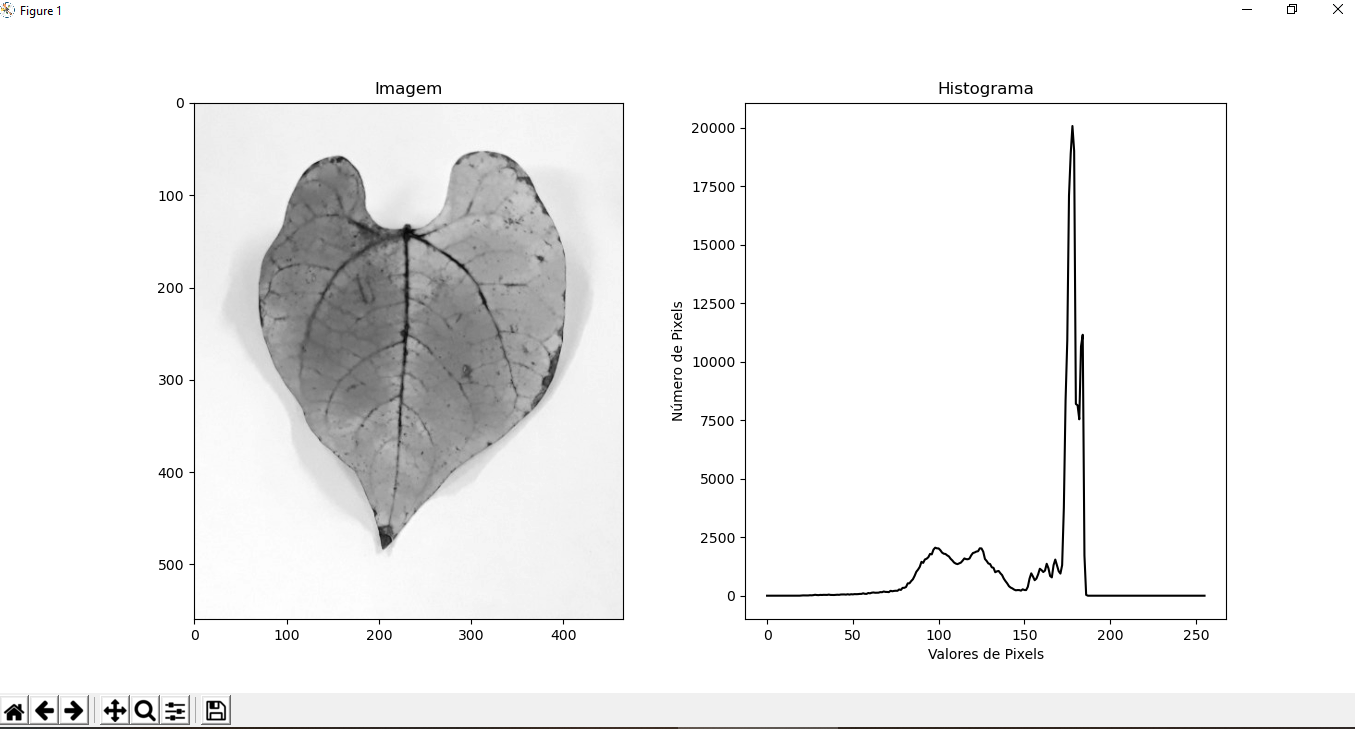
- número de pontos que utilizamos para amostragem, no caso “256”, já que varia de 0 a 255;

- intervalo dos valores dos pixels, que vai de “0 a 256”.

|  |
| --- |
| histograma = cv2.calcHist([img\_cinza],[0],None,[256],[0,256]) print(histograma) |

Com o histograma pronto, construímos uma figura com a função “subplot”, que contém a imagem em escala de cinza e histogramas juntos. O modo como procedemos, já foi citado na questão anterior.

Figura 7: Imagem em escala de cinza e histograma.



A partir da figura 7, podemos observar, no eixo x, o valor dos pixels variando de 0 a 255, faixa essa solicitada como argumento na construção do histograma. E no eixo y, a quantidade de pixels da nossa imagem associada a cada valor. Dessa forma, podemos observar a distribuição dos pixels, e afirmar que cerca de 20.000 deles apresentam valores de intensidade próximos a 170. Também podemos inferir que para a nossa imagem, no geral, os pixels apresentam valores dentro da faixa de intensidade de 80 a 180, não existindo para os demais valores nenhum pixel relacionado.

Ao analisar os valores dos pixels, ao longo da imagem, e de acordo com o histograma, podemos inferir que os dois primeiros picos (menores) são referentes a região da folha. A segunda oscilação, que antecede ao pico maior, refere-se a sombra da folha, e por último, o pico principal que abrange a maioria dos pixels, refere-se ao fundo da imagem.

**e) Utilizando a imagem em escala de cinza (intensidade) realize a segmentação da imagem de modo a remover o fundo da imagem utilizando um limiar manual e o limiar obtido pela técnica de Otsu. Nesta questão apresente o histograma com marcação dos limiares utilizados, a imagem limiarizada (binarizada) e a imagem colorida final obtida da segmentação. Explique os resultados.**

A limiarização consiste na utilização de histogramas para tentar isolar características de interesse em uma imagem. Para isso, usa-se um limiar/threshold que permite a seleção de pixels acima ou abaixo de seu valor. Podemos realizar esse processo de duas maneiras, manualmente ou pela técnica de Otsu.

Limiar Manual

Com a função “cv2.threshold” conseguimos delimitar manualmente um ponto específico para a limiarização. Essa função nos retorna o valor do limiar, nesse caso, o que vamos escolher, e a imagem binária indicando a separação a partir desse valor. Para isso, precisamos especificar alguns pontos na função, como:

- arquivo de entrada, no nosso caso, a imagem em escala de cinza;

- o valor do limiar que se deseja utilizar, optamos por 155;

- o valor máximo de pontos, 255;

- e informar o tipo de threshold, no caso, threshold binário.

O mesmo foi feito, invertendo o threshold binário, alterando o último parâmetro de “cv.thresh\_binary” para “cv.thresh\_binary\_inv”.

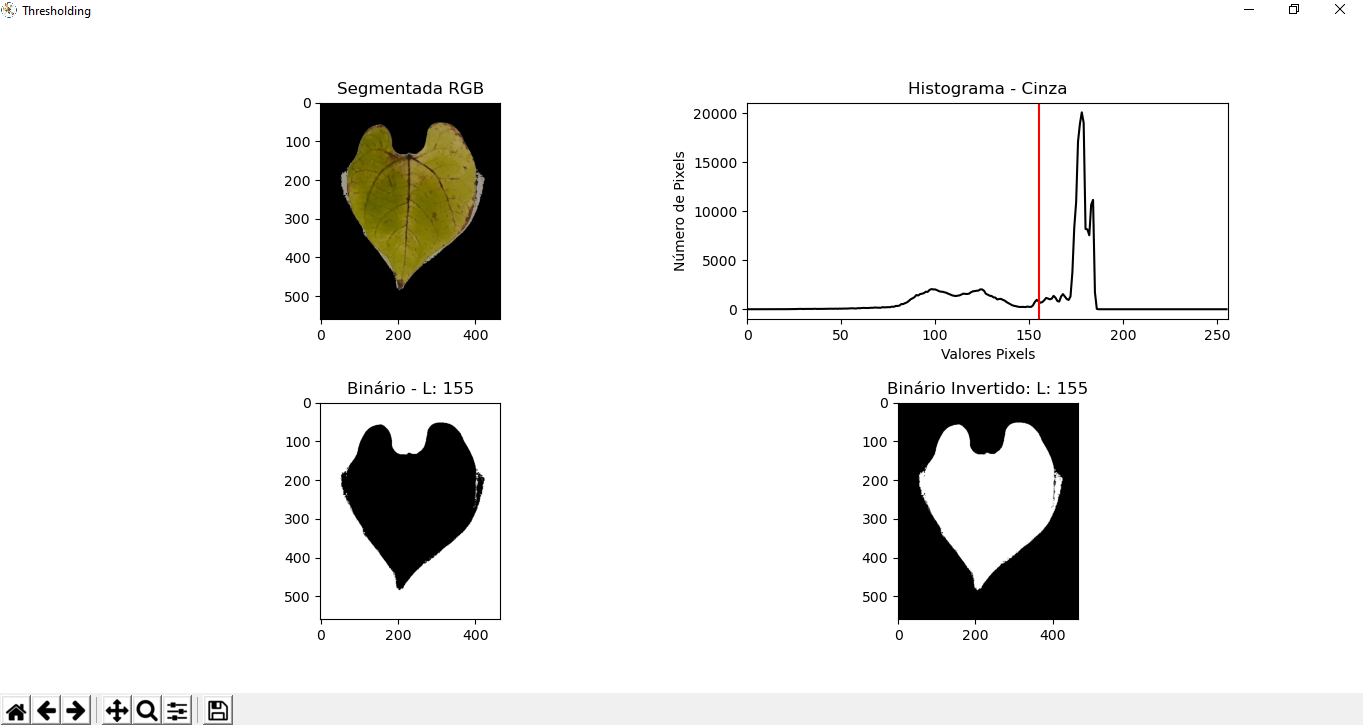
Já a obtenção da imagem colorida segmentada, foi feita utilizando a função “cv2.bitwise\_and” e como argumentos, utilizamos a imagem RGB duas vezes e a imagem segmentada invertida “img\_limiar\_inv”, como máscara.

|  |
| --- |
| limiar\_cinza = 155  (L, img\_limiar) = cv2.threshold(img\_cinza,limiar\_cinza,255,cv2.THRESH\_BINARY)  (L, img\_limiar\_inv) = cv2.threshold(img\_cinza,limiar\_cinza,255,cv2.THRESH\_BINARY\_INV) print('Limiar: ' + str(L))  img\_segmentada = cv2.bitwise\_and(img\_rgb,img\_rgb,mask=img\_limiar\_inv) |

A imagem colorida segmentada, imagem cinza, histograma, as imagens binárias normal e invertida, obtidas a partir do comando mencionado anteriormente, foram plotadas em uma mesma figura “Thresholding” a partir da função “plt.” (Figura 8).

|  |
| --- |
| plt.figure('Thresholding')  plt.subplot(2,2,1)  plt.imshow(img\_segmentada)  plt.title('RGB')  plt.subplot(2,2,2)  plt.plot(histograma,color = 'black')  plt.axvline(x=limiar\_cinza,color = 'r')  plt.title("Histograma - Cinza")  plt.xlim([0,256])  plt.xlabel("Valores Pixels")  plt.ylabel("Número de Pixels")  plt.subplot(2,2,3)  plt.imshow(img\_limiar,cmap='gray')  plt.title('Binário - L: ' + str(limiar\_cinza))  plt.subplot(2,2,4)  plt.imshow(img\_limiar\_inv,cmap='gray')  plt.title('Binário Invertido: L: ' + str(limiar\_cinza))  plt.show() |

Figura 8: Thresholding- limiar manual



Com a figura podemos observar o valor do limiar representado no histograma, e a reação da imagem após a sua aplicação, de forma que, para a imagem binária, as regiões da imagem aonde os pixels apresentaram valores acima do limiar, ficam realçadas de branco, enquanto que, as regiões onde os valores de pixels ficaram abaixo do limiar, ficam demarcadas com preto. Para a imagem binária invertido, o contrário é verdadeiro.

A caracterização da imagem após a limiarização nos permite confirmar a inferência feita na questão anterior sobre a relação da distribuição dos picos com os objetos da imagem, onde propomos que os primeiros picos são referentes a folha, o segundo conjunto de picos à sombra da folha e o terceiro e maior pico, ao fundo, visto que, a região abaixo do limiar, que englobava o primeiro pico e parte do segundo, caracterizada por receber coloração preta, foi exatamente a folha e parte de sua sombra, na imagem binária.

Limiar técnica de Otsu

A partir desta técnica consegue-se determinar automaticamente o melhor limiar para a imagem de interesse. Para isso, utilizamos a mesma função que anteriormente, para a técnica manual de limiarização, “cv2.threshold”. Tal função nos retorna duas variáveis, a primeira “L1” consiste no valor ideal de limiar, e a segunda “img\_otsu”, na imagem linearizada. A diferença dessa função, nesse caso e no anterior, está nos parâmetros utilizados, que são:

- arquivo de entrada, no nosso caso, a imagem em escala de cinza;

- o valor do limiar que se deseja utilizar, no caso, colocamos 0, pois ele será determinado com essa função;

- o valor máximo de pontos que é assumido acima do limiar, 255;

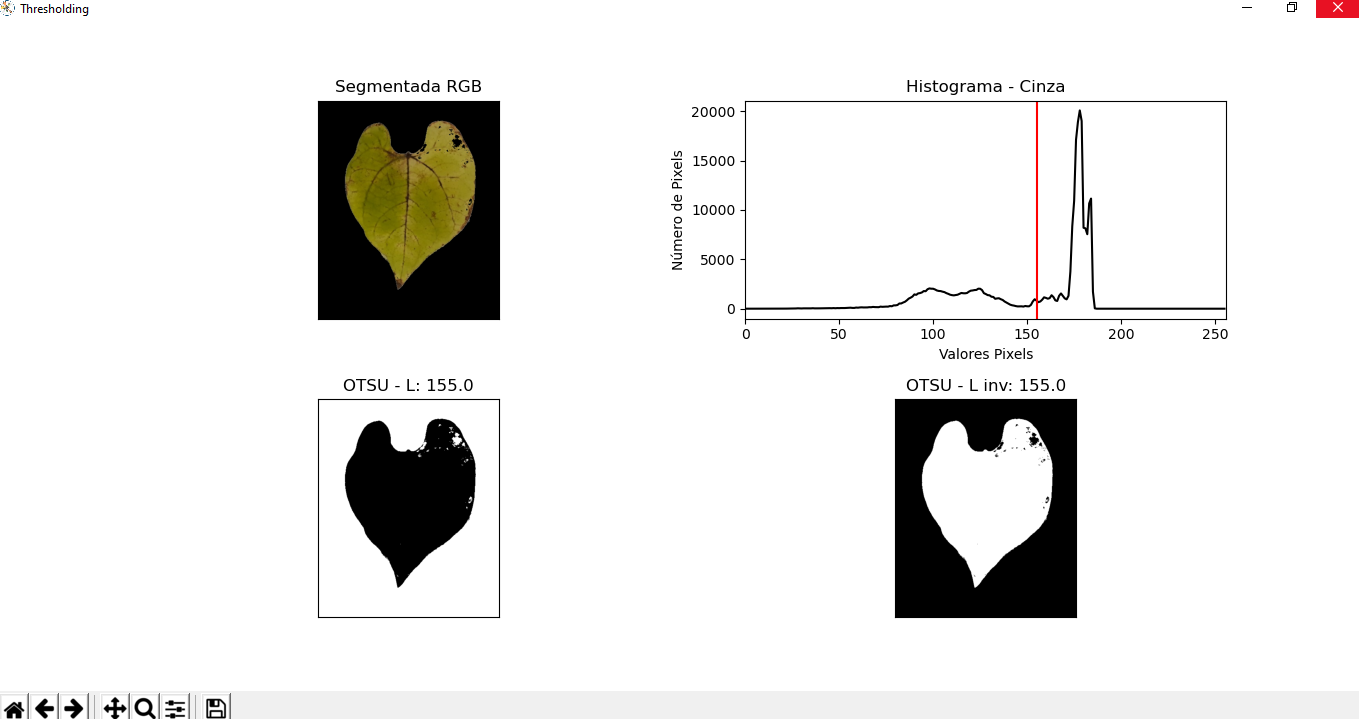
- e informar o tipo de threshold, no caso, “cv2.threshold\_binary”, valores acima do limiar recebe 255 e abaixo 0;

- em adição ao parâmetro do cv do item anterior, adicionamos “cv2.threshold\_otsu”, que é o comando que vai ativar essa análise de Otsu. Fizemos o mesmo para obtenção da imagem binária invertida.

|  |
| --- |
| (L1, img\_otsu) = cv2.threshold(img\_cinza,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)  (Linv, img\_otsu\_inv) = cv2.threshold(img\_cinza,0,255,cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU)  img\_segmentada\_otsu = cv2.bitwise\_and(img\_rgb,img\_rgb,mask=img\_otsu\_inv) |

A plotagem das imagens é semelhante à realizada no item anterior. A figura 9 apresenta as imagens plotadas.

Figura 9: Thresholding- limiar Otsu.



A partir da análise, o limiar ideal para otimizar a separação dos grupos, baseado maximização da variância entre entre eles foi de 155. A divisão contemplou novamente a folha, e detectou diferença em uma região de clorose da folha de feijoeiro, como visto na figura 9.

Quanto aos sintomas da antracnose, visível nas nervuras das folhas, não foram diferenciados por nenhum dos limiares apresentados. Uma hipótese para explicar tal fato, é que a imagem em tons de cinza não foi suficiente para permitir a segmentação e diferenciação dessa característica apresentada na imagem original.

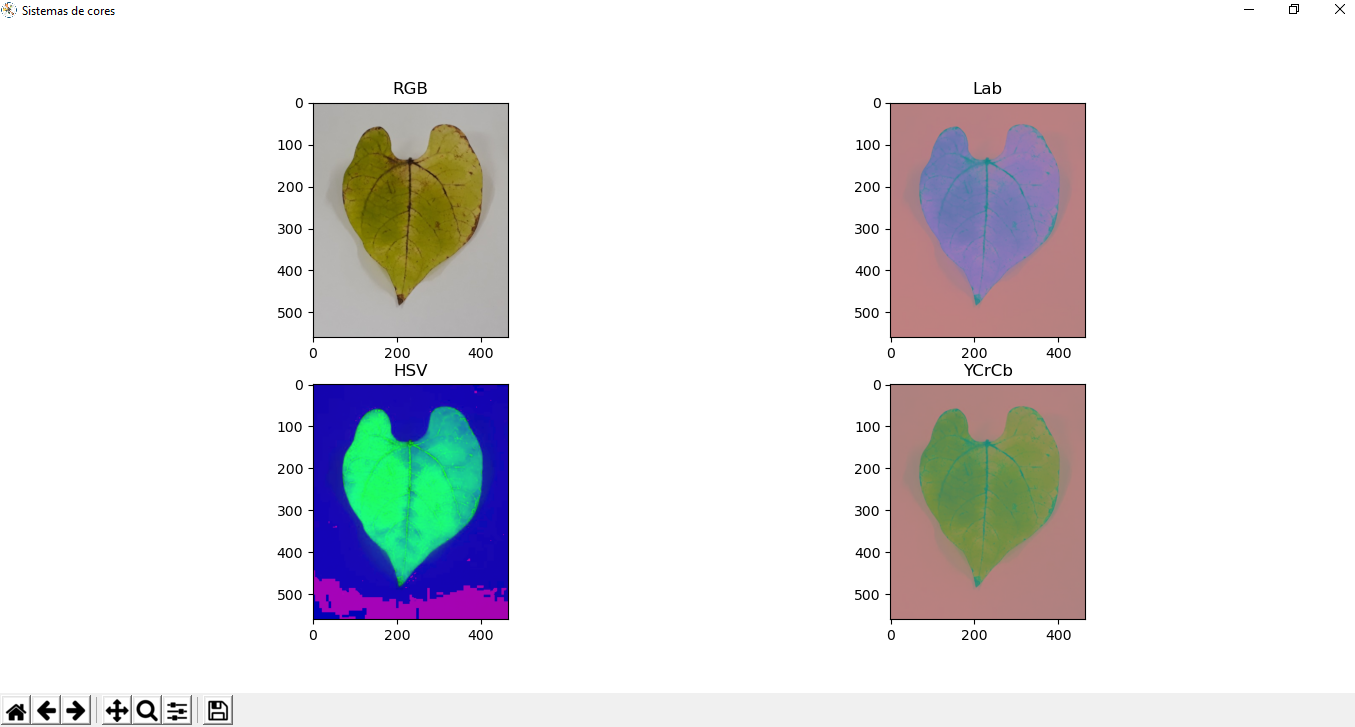
**f) Apresente uma figura contendo a imagem selecionada nos sistemas RGB, Lab, HSV e YCrCb.**

Além do sistema de intensidade de RGB, existem outros espaços de cores que podem ser utilizados para representar imagens coloridas em análises computacionais. Entre esses, podem-se citar o Lab, o HSV e o YCrCb.

A imagem BGR já foi anteriormente convertida para RGB e salva na variável “img\_rgb”. De forma similar, realizamos a conversão para os outros espaços de cores, como representado no quadro abaixo. A partir de então, fizemos um subplot com todas as imagens reunidas. Esse procedimento se encontra demonstrado nas questões anteriores.

|  |
| --- |
| img\_rgb = cv2.cvtColor(img\_recorte,cv2.COLOR\_BGR2RGB)  img\_Lab = cv2.cvtColor(img\_recorte,cv2.COLOR\_BGR2Lab)  img\_HSV = cv2.cvtColor(img\_recorte,cv2.COLOR\_BGR2HSV)  img\_YCrCb = cv2.cvtColor(img\_recorte,cv2.COLOR\_BGR2YCR\_CB) |

Figura 10: Sistemas de cores.



O matplotlib trabalha com o sistema RGB de cores, então quando plotamos a imagem representativa dos outros sistemas, como o Lab, HSV e o YCrCo, há uma mistura de cores resultando no padrão observado na figura 10.

Cada uma desses sistemas apresentam 3 canais que representam cores, espectros de cores, ou parâmetros como luminosidade e saturação. Tais sistemas são melhor explicados a seguir, na próxima questão.

**g) Apresente uma figura para cada um dos sistemas de cores (RGB, HSV, Lab e YCrCb) contendo a imagem de cada um dos canais e seus respectivos histogramas.**

Para acessar os diferentes canais de um sistema de cor, precisamos conhecer suas ramificações e acessar suas matrizes componentes de modo independente. Por exemplo, como já foi citado, o sistema RGB é composto por três matrizes sobrepostas de cores, que corresponde aos seus 3 canais, que são o vermelho, verde e azul.

De modo geral, para a realização da decomposição dos canais na imagem e construção dos seus respectivos histogramas, precisamos primeiramente converter a imagem para os formatos pedidos, criar variáveis para receber os histogramas individuais de cada canal, com a função “cv2.calcHist”, e construir a nova figura reunindo todos os histogramas e suas respectivas imagens, com a função “plt.subplot”.

Os argumentos para a função “cv2.calcHist” são:

- arquivo convertido para o sistema de cores de interesse;

- o canal que se quer acessar, 0, 1 ou 2;

- presença de máscaras, neste caso, não iremos utilizar;

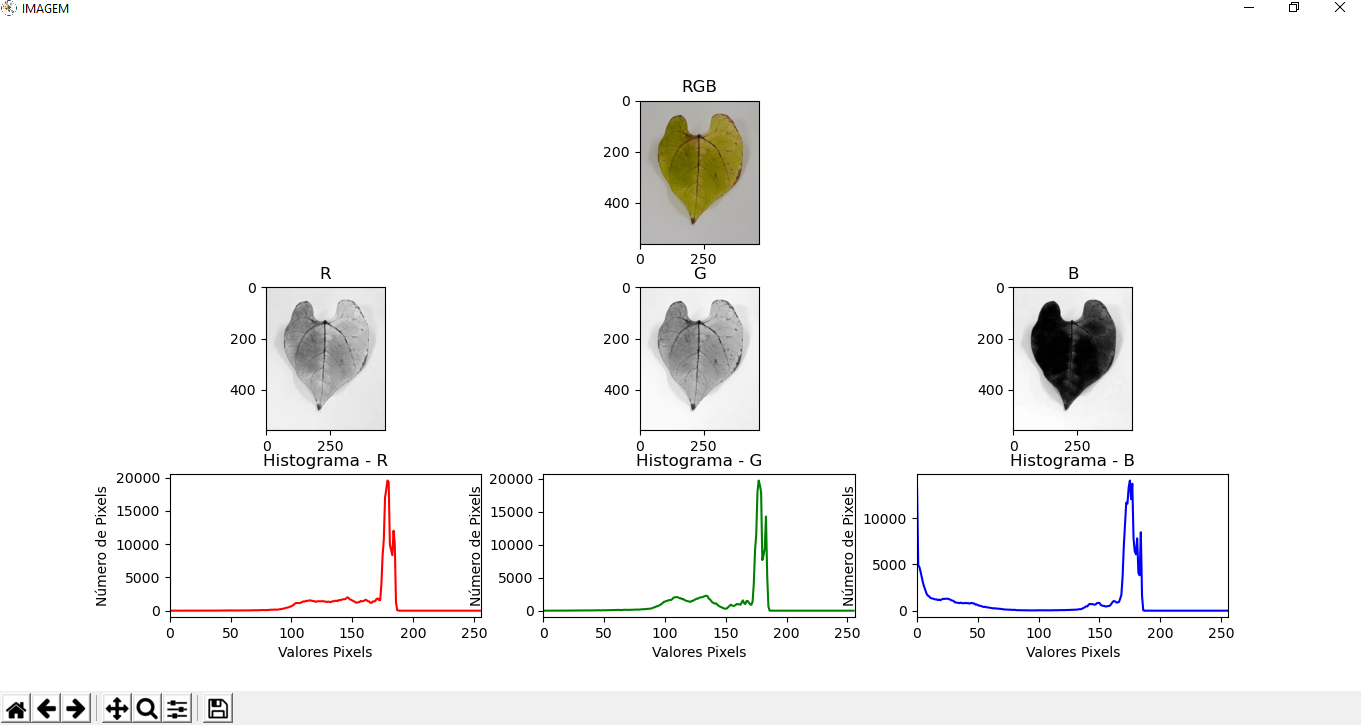
- número de pontos que utilizamos para amostragem, no caso “256”, já que varia de 0 a 255;

- intervalo dos valores dos pixels, que vai de “0 a 256”.

Sistema RGB

|  |
| --- |
| Histograma de imagem em escala de cinza dos três canais:  hist\_r = cv2.calcHist([img\_rgb],[0],None,[256],[0,256])  hist\_g = cv2.calcHist([img\_rgb],[1],None,[256],[0,256])  hist\_b = cv2.calcHist([img\_rgb],[2],None,[256],[0,256])  Apresentação das imagens:  Imagem RGB  plt.figure('IMAGEM')  plt.subplot(3,3,2) plt.imshow(img\_rgb,cmap="gray")  plt.title("RGB")  Imagem- canal vermelho  plt.subplot(3,3,4)  plt.imshow(img\_rgb[:,:,0],cmap='gray') plt.title("R")  Histograma- canal vermelho  plt.subplot(3,3,7)  plt.plot(hist\_r,color = 'r') plt.title("Histograma - R")  plt.xlim([0,256])  plt.xlabel("Valores Pixels")  plt.ylabel("Número de Pixels")  OBSERVAÇÃO: Aqui está representado o código apenas para o canal vermelho da imagem RGB. |

Figura 11: Decomposição dos canais no sistema RGB.



Os diferentes canais, por si só, permitem distinguir a imagem e seus objetos de forma distintas. Assim como na foto RBG, os canais vermelho e verde, permitiram visualizar as nervuras lesionadas pela antracnose na folha, que caracteristicamente apresentam coloração avermelhada/amarronzada, enquanto que o destaque do canal azul foi no sentido de destacar a área verde da mesma.

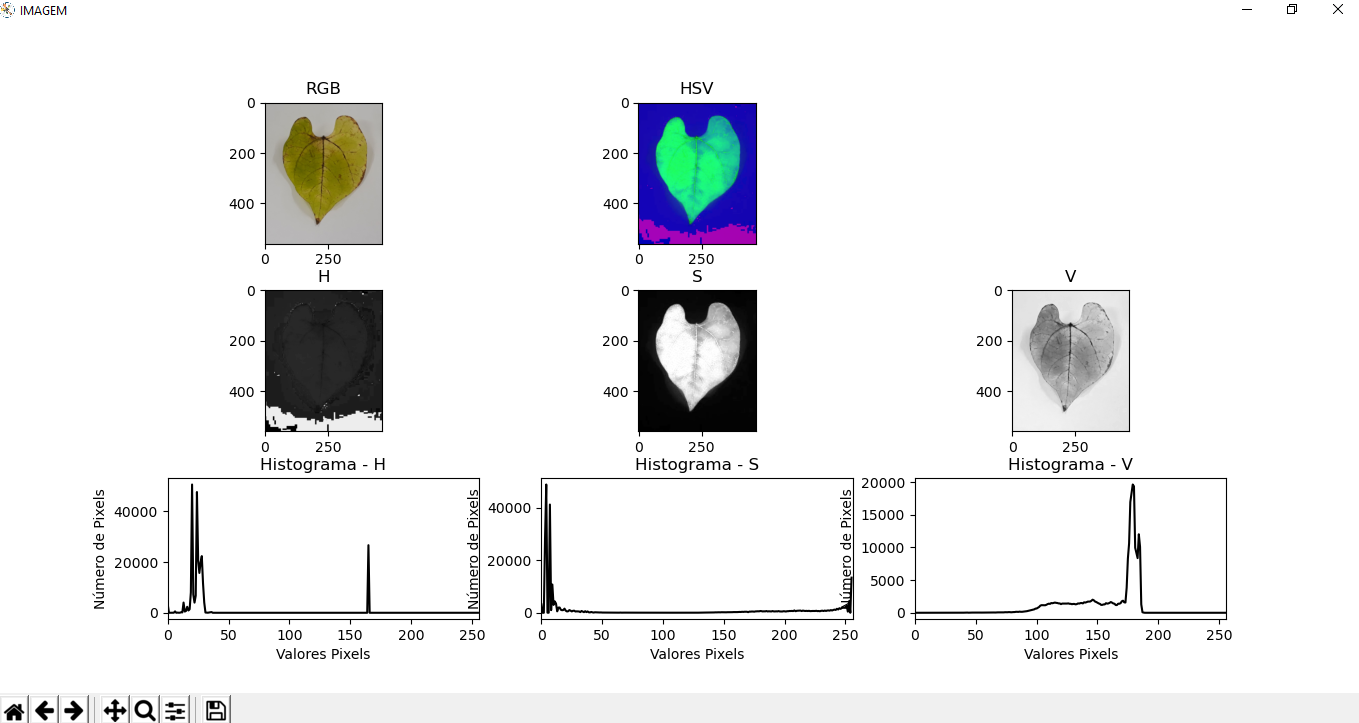
Embora a lesão apresente essa coloração avermelhada, ela não apresenta intensidade suficiente para ser destacada pelo canal vermelho. Se isso acontecesse, ela seria destacada com a cor branca. O pico dos pixels continua sendo relacionado ao fundo da imagem. O raciocínio também se aplica ao canal verde, talvez pela folha ter perdido um pouco de sua pigmentação, devido aos sintomas da doença.

Mesmo que a folha e as lesões não tenham sido destacadas pelos canais verde e vermelho, elas continuam sendo perceptíveis, pois apresentam intensidades relacionadas a essas cores. O mesmo não ocorre quando é aplicado o canal azul, permitindo que a folha fique toda escura, devido a ausência de intensidade azul na imagem.

Sistema HSV

|  |
| --- |
| Histograma de imagem em escala de cinza dos três canais do HSV:  hist\_H = cv2.calcHist([img\_HSV],[0],None,[256],[0,256])  hist\_S = cv2.calcHist([img\_HSV],[1],None,[256],[0,256])  hist\_V = cv2.calcHist([img\_HSV],[2],None,[256],[0,256])  Apresentação das imagens:  Imagem em RGB para comparação:  plt.figure('IMAGEM')  plt.subplot(3,3,1) plt.imshow(img\_rgb,cmap="gray")  plt.title("RGB")  Imagem em HSV:  plt.subplot(3,3,2)  plt.imshow(img\_HSV,cmap="gray")  plt.title("HSV")  Imagem do canal H do sistema HSV:  plt.subplot(3,3,4)  plt.imshow(img\_HSV[:,:,0],cmap='gray')  plt.title("H")  Histograma- canal H do sistema HSV  plt.subplot(3,3,7)  plt.plot(hist\_H,color = 'black')  plt.title("Histograma - H")  plt.xlim([0,256])  OBSERVAÇÃO: Aqui está representado o código apenas para o canal H da imagem HSV. |

Figura 12: Decomposição dos canais no sistema HSV.



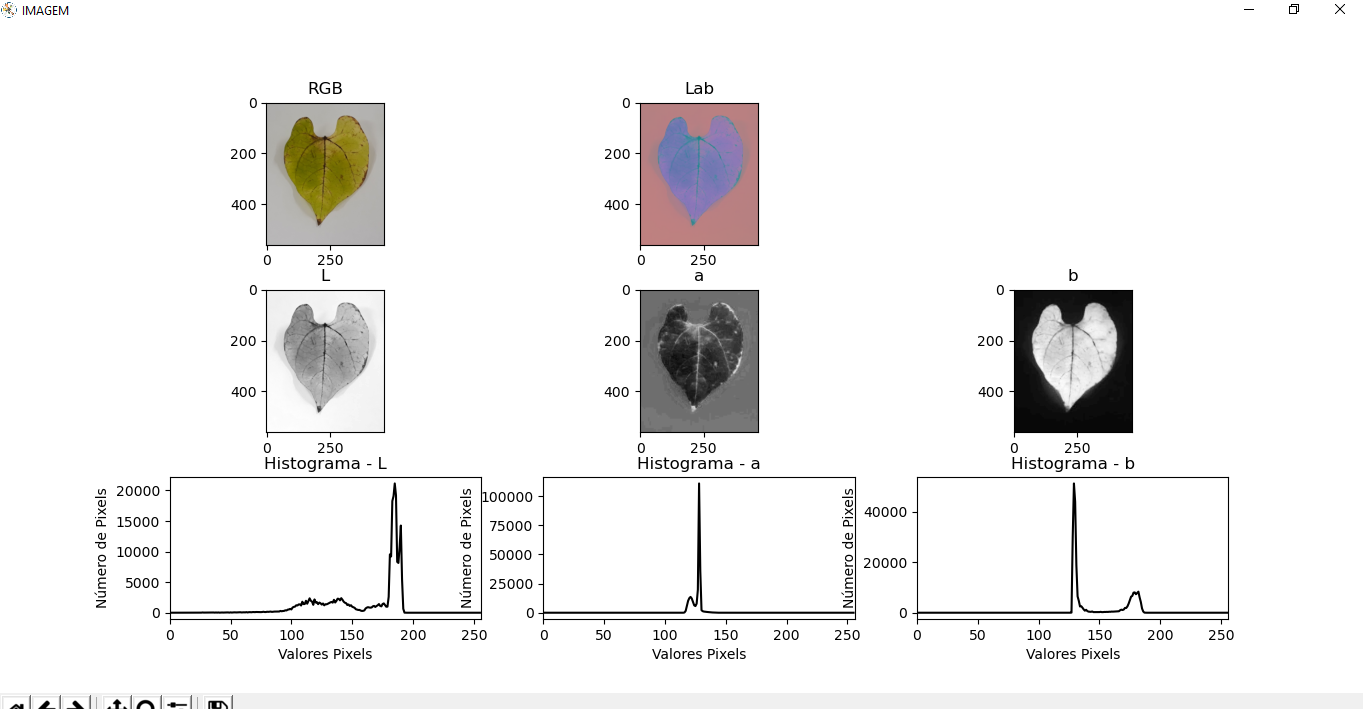
No sistema HSV, o primeiro canal representa a tonalidade das cores da imagem, o segundo a saturação, ou seja, quantidade de cor branca e o terceiro se refere a luminosidade da imagem.

O canal “H” não permite a distinção dos objetos da imagem (folha e lesão), tão pouco, a distinção da folha com o fundo. O canal “S” contempla, por sua vez, essa última distinção, entretanto, também não permite distinguir as lesões nas nervuras da folha. O canal “V” também não apresenta muita diferença, principalmente se observar o histograma, o qual o padrão lembra o da imagem RGB.

Sistema Lab

|  |
| --- |
| Histograma de imagem em escala de cinza dos 3 canais do sistema Lab  hist\_L = cv2.calcHist([img\_Lab],[0],None,[256],[0,256])  hist\_a = cv2.calcHist([img\_Lab],[1],None,[256],[0,256])  hist\_b = cv2.calcHist([img\_Lab],[2],None,[256],[0,256])  Apresentação das imagens:  Imagem RGB:  plt.figure('IMAGEM')  plt.subplot(3,3,1)  plt.imshow(img\_rgb,cmap="gray")  plt.title("RGB")  Imagem Lab:  plt.subplot(3,3,2)  plt.imshow(img\_Lab,cmap="gray")  plt.title("Lab")  Imagem canal L do sistema Lab:  plt.subplot(3,3,4)  plt.imshow(img\_Lab[:,:,0],cmap='gray')  plt.title("L")  Histograma do canal L do sistema Lab:  plt.subplot(3,3,7)  plt.plot(hist\_L,color = 'black')  plt.title("Histograma - L")  plt.xlim([0,256])  plt.xlabel("Valores Pixels")  plt.ylabel("Número de Pixels")  OBSERVAÇÃO: Aqui está representado o código apenas para o canal L da imagem Lab. |

Figura 13: Decomposição dos canais no sistema Lab.



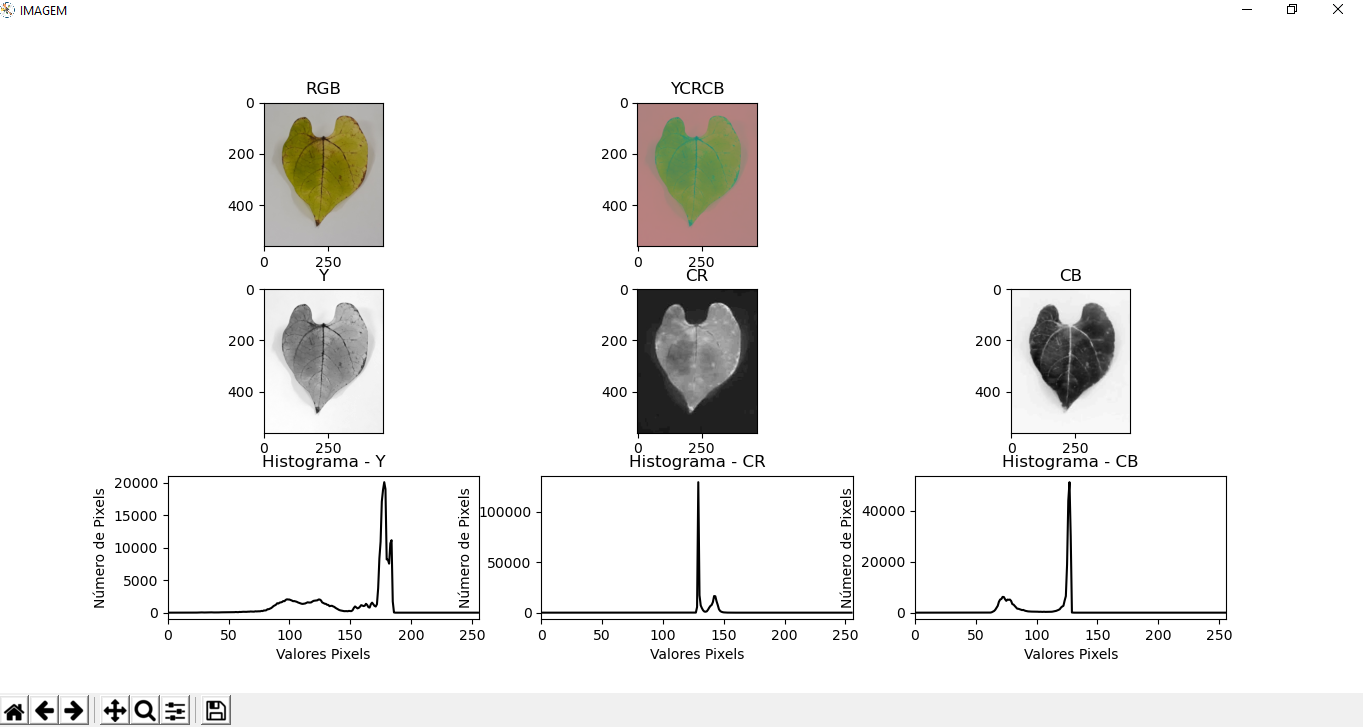
Nesse sistema, os três canais são representados pelo “L”, que é relacionado com a quantidade de luz branca que tem na imagem, o “a” e o “b” que se relacionam com as cores da imagem. O canal “L” vai então do preto ao branco, conforme aumenta a luminosidade de imagem. O canal “a” varia do verde ao vermelho e o “b” varia do azul ao amarelo. Diferentemente do sistema RGB, onde os canais eram representados por uma cor específica, no Lab, o segundo e terceiro canais são representados por um espectro de cor, além de ter um só direcionado a luminosidade, o primeiro. Essa combinação nos oferece todas as outras cores que podemos trabalhar em imagens coloridas.

Em termos de luminosidade, o pico continua relacionado ao fundo claro da imagem, sem muitas alterações na folha. Já para o espectro de verde a vermelho, representado pelo canal “a” houve variação. Neste caso, as lesões da folha foram destacadas com a cor branca, enquanto o restante da folha recebeu coloração mais próximo do preto, ou seja, houve uma discriminação da lesão a partir desse canal. O terceiro canal também não permitiu discriminação dos sintomas, como no segundo.

Sistema YCrCb

|  |
| --- |
| Histograma dos 3 canais da imagem YCrCo em escala de cinza:  hist\_Y = cv2.calcHist([img\_YCRCB],[0],None,[256],[0,256])  hist\_CR = cv2.calcHist([img\_YCRCB],[1],None,[256],[0,256])  hist\_CB = cv2.calcHist([img\_YCRCB],[2],None,[256],[0,256])  Apresentação das imagens:  Imagem RGB:  plt.figure('IMAGEM')  plt.subplot(3,3,1)  plt.imshow(img\_rgb,cmap="gray")  plt.title("RGB")  Imagem YCrCo:  plt.subplot(3,3,2)  plt.imshow(img\_YCRCB,cmap="gray")  plt.title("YCRCB")  Imagem- canal Y do sistema YCrCo:  plt.subplot(3,3,4)  plt.imshow(img\_YCRCB[:,:,0],cmap='gray')  plt.title("Y")  Histograma canal Y do sistema YCrCo:  plt.subplot(3,3,7)  plt.plot(hist\_Y,color = 'black')  plt.title("Histograma - Y")  plt.xlim([0,256])  plt.xlabel("Valores Pixels")  plt.ylabel("Número de Pixels")  OBSERVAÇÃO: Aqui está representado o código apenas para o canal Y da imagem YCrCo. |

Figura 14: Decomposição dos canais no sistema YCrCb.



O sistema YCrCb apresenta os canais Y, Cr e Cb. Com a decomposição dos canais, podemos ver que os canais Y e Cr não permitem uma distinção das lesões foliares quando comparadas com o canal Cb. Neste último, as lesões das nervuras recebem a cor branco enquanto o restante da folha permanece escuro e em contraste com o fundo da imagem. Entretanto, em questão de picos do histograma essa distinção não é tão evidente, uma vez que o pico menor se refere a folha como um todo e o maior, ao fundo da imagem.

**h) Encontre o sistema de cor e o respectivo canal que propicie melhor segmentação da imagem de modo a remover o fundo da imagem utilizando limiar manual e limiar obtido pela técnica de Otsu. Nesta questão apresente o histograma com marcação dos limiares utilizados, a imagem limiarizada (binarizada) e a imagem colorida final obtida da segmentação. Explique os resultados e sua escolha pelo sistema de cor e canal utilizado na segmentação. Nesta questão apresente a imagem limiarizada (binarizada) e a imagem colorida final obtida da segmentação.**

O sistema de cor que permitiram visualmente diferenciar as lesões na folha da imagem foi o Lab e o YCrCB, mais especificamente os canais “a” e “Cb”, respectivamente. Estes sistemas, em geral, permitem identificar bons canais para a segmentação. Entretanto, como mostra os histogramas da questão anterior, não é possível capturar à primeira instância a distinção entre nervuras lesionadas e restante da folha. Essa primeira segmentação inicial permite somente a distinção do fundo da imagem e da folha como um todo. Para capturar a intensidade de cor das lesões poderia ser feito um processo de “multisegmentação”, segmentando primeiro o fundo e posteriormente, utilizando a imagem segmentada, proceder a segmentação dos sintomas, com os filtros mencionados anteriormente, que mostraram bons indícios de distinção dessas regiões.

Para esse relatório, vamos focar na segmentação do fundo da imagem, a partir do canal S do sistema de cores HSV. Posteriormente, a medida que formos expandindo nosso conhecimento neste tipo de análise, tentaremos segmentar os sintomas da doença.

Criamos os histogramas dos três canais para realizar uma pré seleção, de qual permitiu uma boa segmentação. Em seguida, realizamos a partição dos canais utilizando a função “cv2.split”, armazenando cada canal em uma variável, e fizemos o histograma do canal selecionado.

|  |
| --- |
| H,S,V = cv2.split(img\_HSV)  hist\_S = cv2.calcHist([S],[0], None, [256],[0,256]) |

Limiarização- Técnica de OTSU

A partir de então, fizemos a linearização a partir da técnica de OTSU, obtendo o valor “L2” de limiar ideal para utilizarmos em nossa segmentação e o arquivo binário. Para isso, indicamos como parâmetros da função o (1.) canal, (2.) ponto máximo para valores acima do limiar e (3.) a função para a técnica de OTSU.

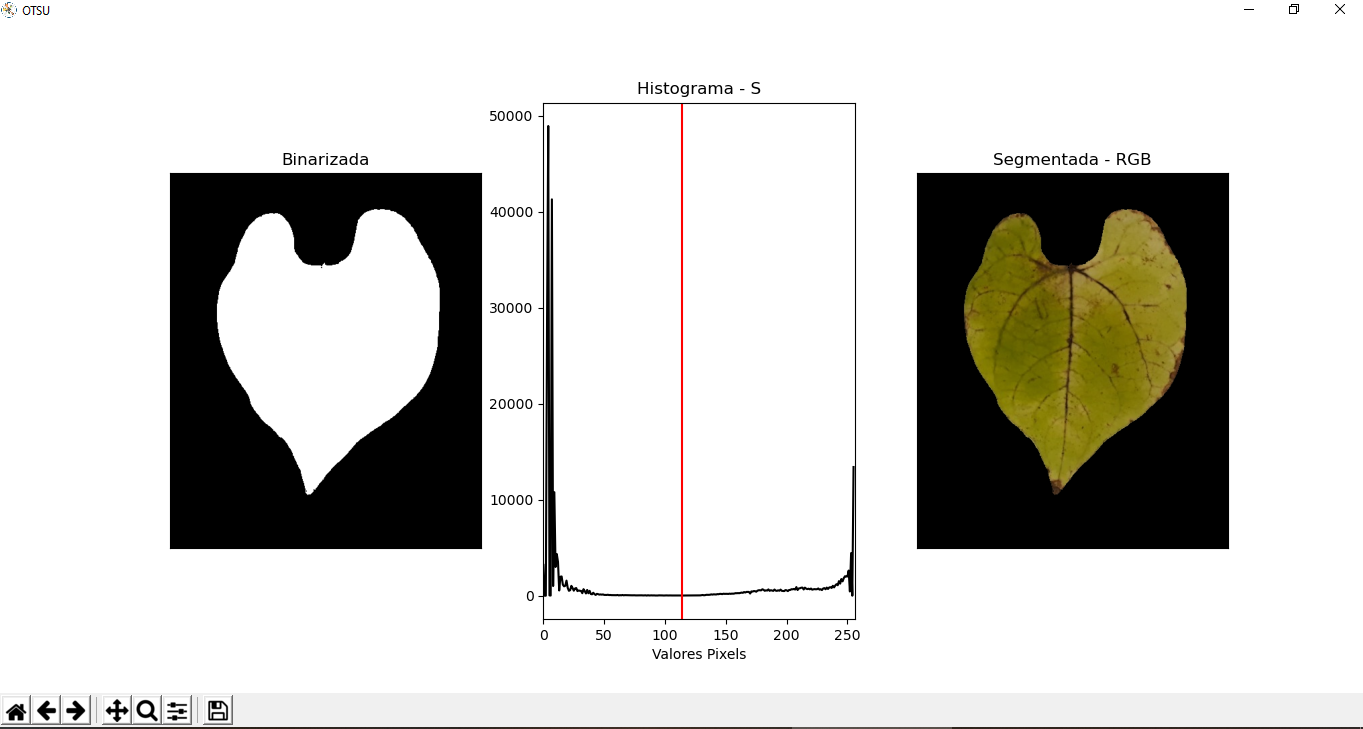
|  |
| --- |
| (L2, img\_limiar) = cv2.threshold(S,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU) |

A imagem limiarizada “img\_limiar” vai permitir a distinção da folha e do fundo da imagem, e será utilizada como máscara para a segmentação, permitindo selecionar o objeto de interesse da nossa imagem. Para isso, utilizamos a função “cv2.bitwise\_and” e como argumentos, utilizamos a imagem RGB duas vezes e a imagem segmentada como máscara.

|  |
| --- |
| img\_segmentada1 = cv2.bitwise\_and(img\_rgb,img\_rgb,mask=img\_limiar) |

Criamos um subplot com o histograma do canal S do sistema HSV, a imagem limiarizada com a técnica de OTSU e a imagem colorida segmentada.

Figura 15: Segmentação a partir da técnica de OTSU.



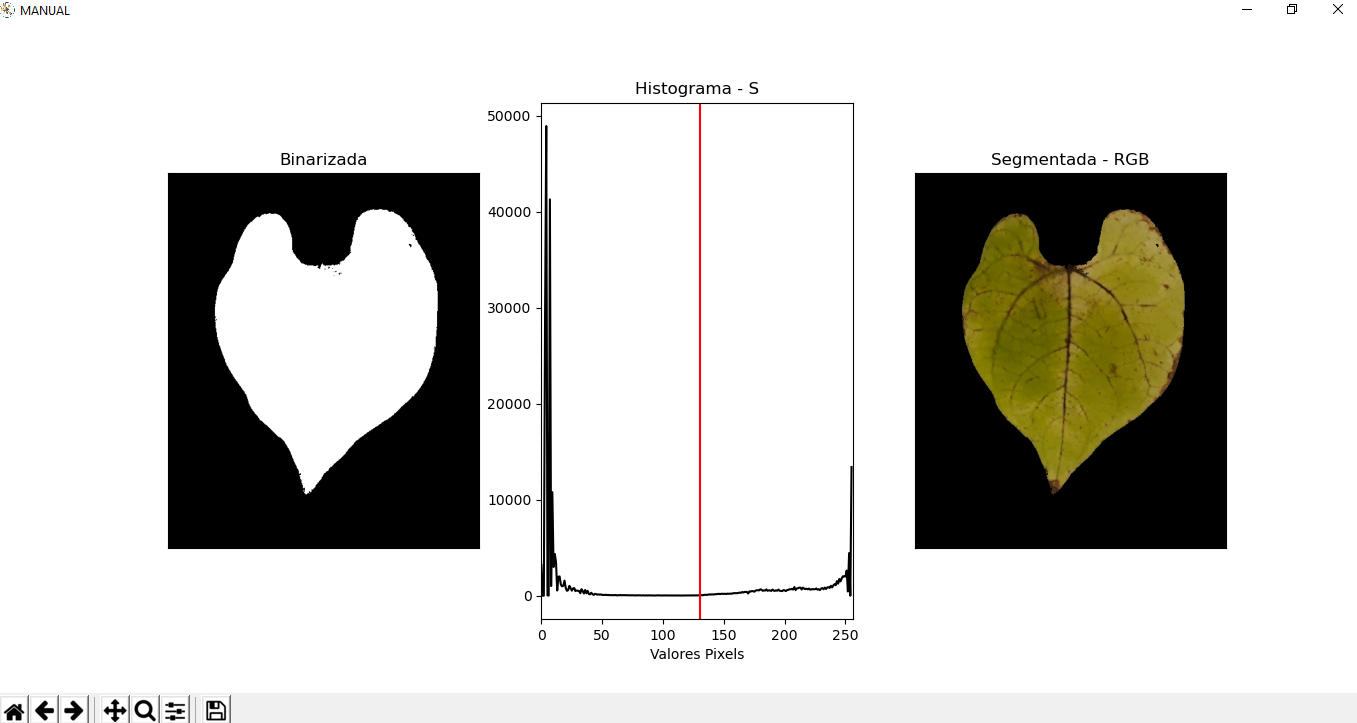
A partir da imagem 15, podemos ver que o limiar proposto pela técnica de OTSU, foi eficiente em separar os picos relacionados ao fundo da imagem e a folha, e que a imagem segmentada diferenciou a folha, deixando as lesões a parte.

Limiarização manual

Com base no histograma obtido anteriormente, para a limiarização manual, selecionamos um limiar de 130. Então, realizamos a limiarização com a função “cv2.threshold” (argumentos: canal, limiar, valor maximo utilizado acima do limiar, e função para formação da imagem binária “cv2.thresh\_binary”) e a segmentação com a função “cv2.bitwise\_and”.

|  |
| --- |
| #Determinação do limiar e limiarização  limiar = 130  (L3, img\_limiar\_a) = cv2.threshold(a,limiar\_S,255,cv2.THRESH\_BINARY)  # Obtendo imagem segmentada  img\_segmentada2 = cv2.bitwise\_and(img\_rgb,img\_rgb,mask=img\_limiar\_S) |

Figura 15: Segmentação a partir da limiarização manual.



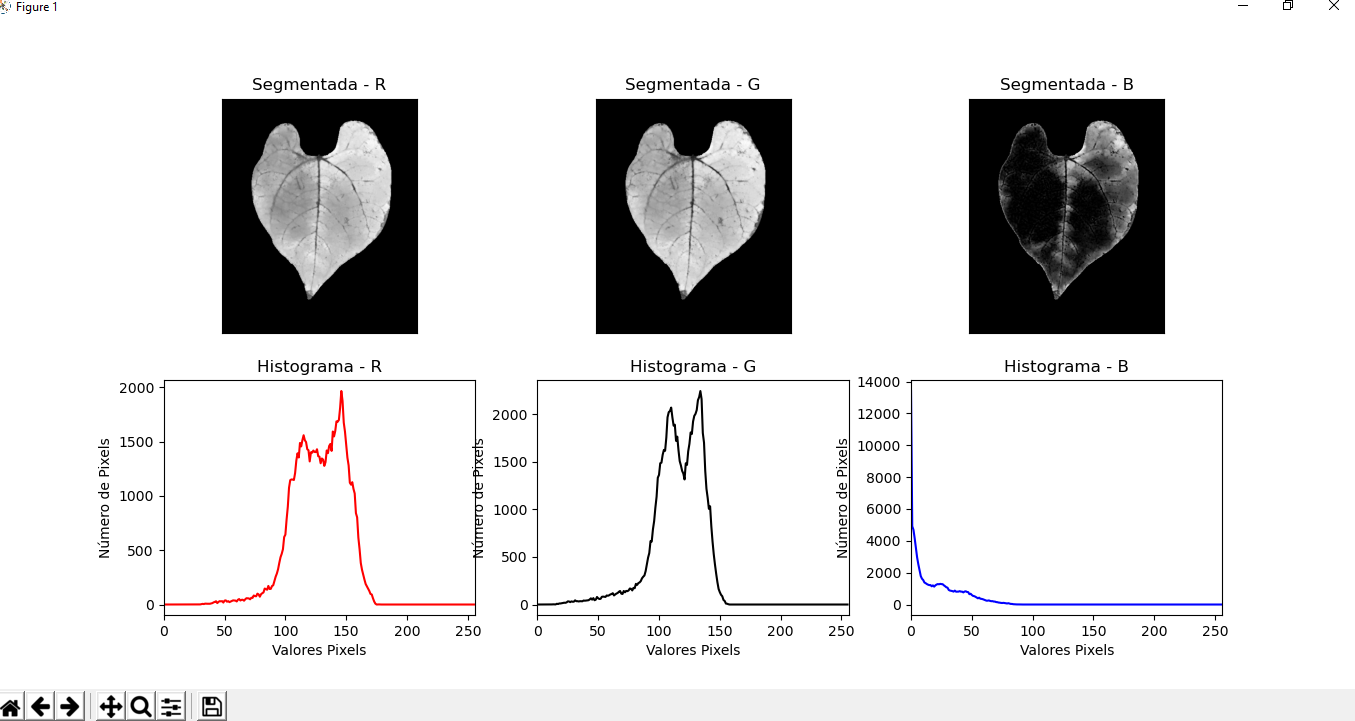
Com o limiar 130 foi possível assim como na técnica anterior fazer a distinção entre folha e fundo, como pode ser observado pela linha do limiar no histograma do canal S do sistema HSV.

**i) Obtenha o histograma de cada um dos canais da imagem em RGB, utilizando como mascara a imagem limiarizada (binarizada) da letra h.**

A partir da imagem RGB segmentada obtida anteriormente com a técnica de OTSU,“img\_segmentada1”, criamos histogramas individuais para cada um dos canais R,G e B, utilizando como máscara a imagem obtida pelo limiar “img\_limiar”. No subplot, colocamos a imagem decomposta em cada canal e seu histograma, como mostra a imagem 16.

|  |
| --- |
| hist\_seg\_r = cv2.calcHist([img\_segmentada1],[0],img\_limiar,[256],[0,256])  hist\_seg\_g = cv2.calcHist([img\_segmentada1],[1],img\_limiar,[256],[0,256])  hist\_seg\_b = cv2.calcHist([img\_segmentada1],[2],img\_limiar,[256],[0,256]) |

Imagem 16: Imagem e histograma dos canais da imagem RGB segmentada.



Com a decomposição dos canais da imagem segmentada RGB e seus respectivos histogramas (Imagem 16), vemos que as cores vermelho e verde continuam permitindo melhor visualização das nervuras lesionadas da antracnose, assim como foi visto na imagem 11. Ainda comparando com a decomposição dos canais RGB da imagem original recortada, observamos mudanças nos picos do histograma, onde antes era perceptível 3 alterações, agora apresentam somente 2, e isso é resultado da segmentação da imagem.

**j) Realize operações aritméticas na imagem em RGB de modo a realçar os aspectos de seu interesse. Exemplo (2\*R-0.5\*G). Explique a sua escolha pelas operações aritméticas.**

As operações envolvendo os diferentes canais de um sistema nos permite obter algumas informações de interesse da imagem, de modo a realçá-las, gerando como resultado uma matriz única nova.

Utilizando a imagem original recortada RGB “img\_rgb”, realizamos três operações com a intenção de realçar os sintomas característicos de antracnose nas nervuras das folha de feijoeiro.

* Primeira operação:

Para a primeira operação, utilizamos os canais azul e vermelho. Multiplicamos o canal azul por 1,5 e em seguida subtraímos dele, o dobro do canal vermelho.

|  |
| --- |
| img\_opr\_01 = 1.5\*img\_rgb[:,:,2] - 2\*img\_rgb[:,:,0] |

* Segunda operação:

Aqui, realizamos novamente uma subtração dos canais azul e vermelho, porém igualamos o valor pelo qual ambos são multiplicados.

|  |
| --- |
| img\_opr\_02 = 1.5\*img\_rgb[:,:,2] - 1.5\*img\_rgb[:,:,0] |

* Terceira operação:

A partir de 1,8 vezes o canal azul, subtraímos 1,5 vezes o canal vermelho, e do resultado, subtraímos o canal verde.

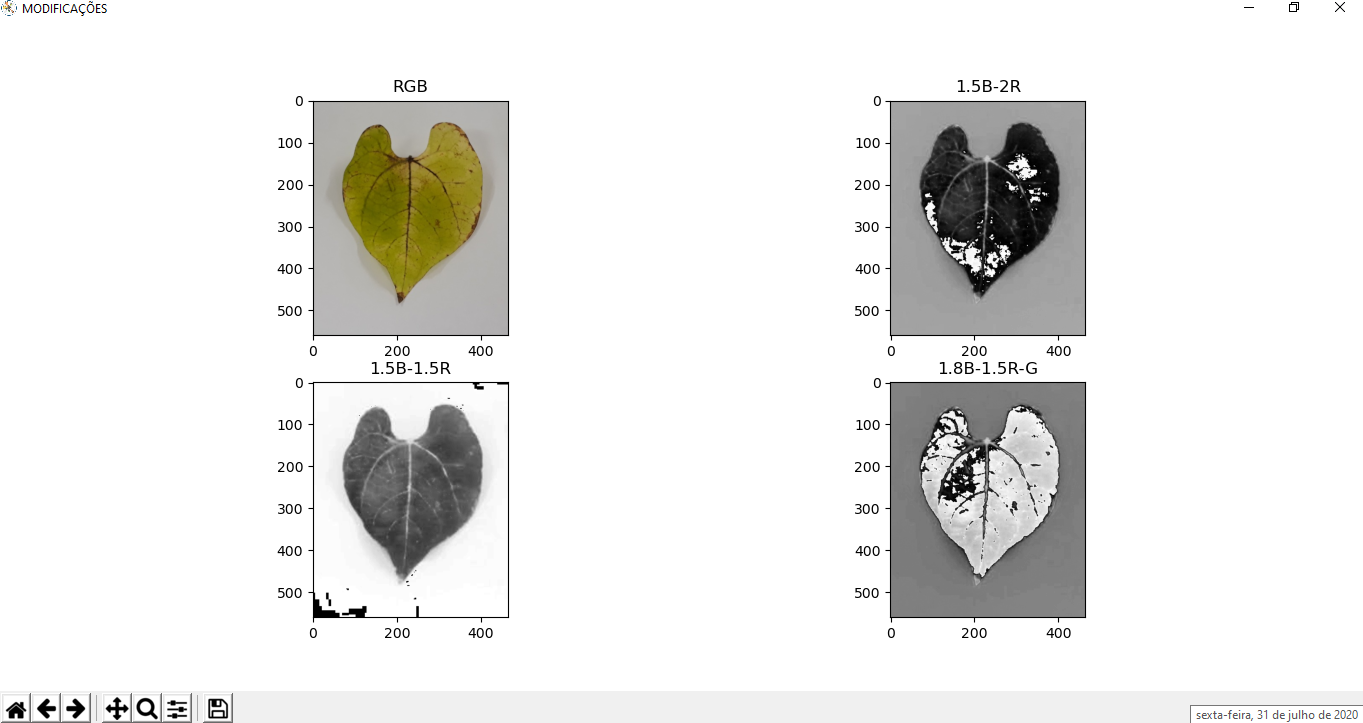
|  |
| --- |
| img\_opr\_03 = 1.8\*img\_rgb[:,:,2] - 1.5\*img\_rgb[:,:,0] - img\_rgb[:,:,1] |

Com as operações, muitas vezes envolvendo multiplicação dos canais, o valor dos pixels podem fugir da faixa 0-255. Para solucionar essa questão, fizemos a conversão para inteiro de 8 bits, a partir da função “.astype” com argumento “np.uint8”.

|  |
| --- |
| img\_opr\_01 = img\_opr\_01.astype(np.uint8)  img\_opr\_02 = img\_opr\_02.astype(np.uint8)  img\_opr\_03 = img\_opr\_03.astype(np.uint8) |

Em seguida, plotamos em uma figura, o resultado das três operações realizadas (Figura 17).

Figura 17: Operações aritméticas com os canais do sistema RBG.



De acordo com o resultado apresentado na figura 17, observamos a imagem resultante da terceira operação aritmética com os canais vermelho, verde e azul, permitiu uma boa diferenciação dos sintomas da doença na folha de feijoeiro.