

MODUL 10 – Thresholding, Segmentasi Sederhana

A. TUJUAN

1. Mahasiswa mampu memahami konsep Thresholding
2. Mahasiswa dapat mengetahui beberapa teknik Thresholding
3. Mahasiswa dapat membuat beberapa teknik Thresholding menggunakan Python pada Google Colab

B. ALAT DAN BAHAN

1. PC/LAPTOP
2. Github
3. *Google Colaborator*

C. ULASAN TEORI

C.1 Pengertian Operasi Thresholding

Thresholding adalah bentuk metode paling sederhana dari segmentasi citra. Biasanya digunakan pada citra grayscale / warna dan ide dasarnya adalah bagaimana memisahkan antara objek foreground dengan objek backgroundnya.

Gambar berikut menunjukkan citra asli sebelum dan sesudah di lakukan thresholding.



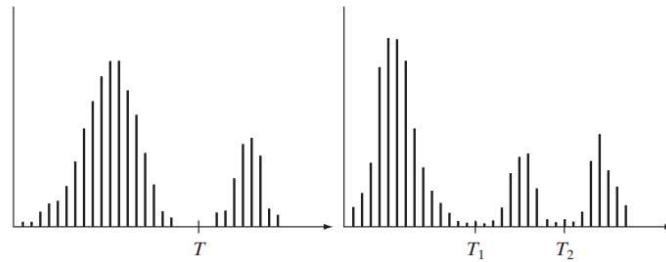
Gambar 1. Citra kiri sebelum di threshold, citra kanan setelah dithreshold (citra binary)

Beberapa metode dari Threshold yang akan dibahas pada modul ini diantaranya:

- a. Global Threshold
- b. Adaptive Threshold
- c. Otsu's Threshold
- d. Pengenalan Segmentasi Citra menggunakan K-Means

C.2 Global Threshold

Perhatikan histogram dari sebuah image yang ditunjukkan pada gambar berikut:



Gambar 2. Intensitas histogram dapat dipisahkan dengan 1 Threshold (a) atau multiple Threshold (b)

Dianggap bahwa gambar diatas adalah histogram dari citra $f(x,y)$, yang terdiri dari object terang pada background gelap. Dengan asumsi bahwa object dan background memiliki nilai intensitas warna yang dapat dipisahkan dalam 2 grup dominan. Cara yang termudah dan paling jelas adalah dengan memilih nilai tertentu (Threshold, T) yang akan memisahkan dua grup tersebut. Tiap titik (x,y) dalam image dimana $f(x,y) > T$ dapat diistilahkan dengan *object point*, sedangkan grup satunya disebut dengan *background point*. Dengan kata lain, citra tersegmentasi $g(x,y)$ dituliskan sebagai berikut:

$$g(x,y) = \begin{cases} 1 & \text{jika } f(x,y) > T \\ 0 & \text{jika } f(x,y) \leq T \end{cases}$$

Ketika T ditentukan secara konstanta pada keseluruhan citra, maka proses inilah yang disebut sebagai *global thresholding*. Ketika nilai T berubah-ubah dalam dalam proses satu citra, maka hal ini disebut dengan *variable/adaptive thresholding*.

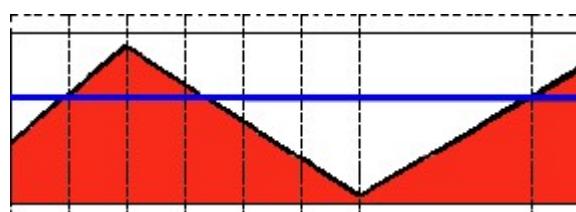
Gambar 2(a) diatas menunjukkan permasalahan thresholding yang melibatkan 3 grup dominan. Diasumsikan bahwa ada 2 grup terang dan 1 grup gelap pada histogram tersebut. Proses multiple thresholding akan mengelompokkan titik (x,y) sebagai background jika $f(x,y) \leq T_1$, ke grup object satu jika $T_1 < f(x,y) \leq T_2$, dan object grup lain jika $f(x,y) > T_2$. Atau dapat dituliskan sebagai berikut:

$$g(x,y) = \begin{cases} a & \text{jika } f(x,y) > T_2 \\ b & \text{jika } T_1 < f(x,y) \leq T_2 \\ c & \text{jika } f(x,y) \leq T_1 \end{cases}$$

Pada OpenCV telah disediakan library untuk Thresholding, baik itu Global, Adaptive, maupun Otsu Threshold. Berikut adalah beberapa Global Thresholding yang disediakan:

- a. Binary Threshold
- b. Binary-Inverted Threshold
- c. Truncate Threshold
- d. Threshold To Zero
- e. Threshold To Zero-Inverted

Untuk mengilustrasikan cara kerja masing-masing threshold, perhatikan histogram dari image asli $src(x,y)$ pada gambar 3 berikut. Garis horizontal biru menunjukkan nilai threshold (tetap).



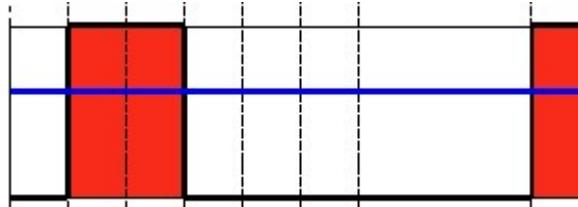
Gambar 3. Histogram dari citra $src(x,y)$

a. Binary Threshold

Binary Threshold seperti yang dijelaskan sebelumnya, merupakan threshold yang akan memisahkan dua grup warna. Untuk threshold ini, masing-masing kelompok warna akan diubah ke nilai gelap (hitam) untuk object yang dianggap background, dan diubah ke nilai terang (putih) untuk object yang dianggap foreground. Berikut adalah persamaannya:

$$dst(x,y) = \begin{cases} maxVal & \text{jika } src(x,y) > thresh \\ 0 & \text{jika lainnya} \end{cases}$$

Jadi jika nilai intensitas piksel warna pada $src(x,y)$ lebih tinggi dari $thresh$, maka nilainya akan diubah menjadi $maxVal$. Selain itu akan diubah menjadi 0.



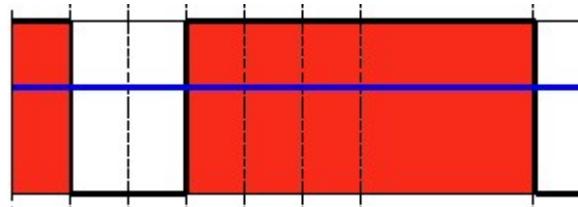
Gambar 4. Histogram hasil dari Binary Threshold

b. Binary-Inverted Threshold

Threshold ini merupakan kebalikan dari Binary Threshold. Jika intensitas warna diatas Threshold, maka nilai akan diubah menjadi 0 dan sebaliknya. Berikut adalah persamaannya:

$$dst(x,y) = \begin{cases} 0 & \text{jika } src(x,y) > thresh \\ maxVal & \text{jika lainnya} \end{cases}$$

Gambar 5 berikut merupakan histogram hasil Binary-Inverted Threshold.

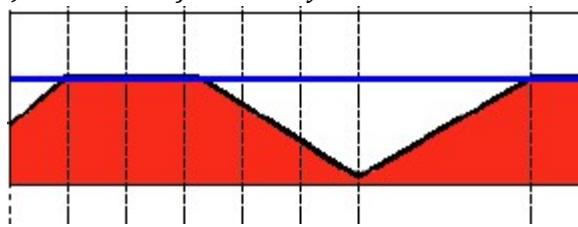


Gambar 5. Histogram hasil dari Binary-Inverted Threshold

c. Truncate Threshold

Jika nilai intensitas warna $src(x,y)$ lebih besar dari $thresh$, maka nilainya akan ditruncate. Berikut adalah persamaan dan Histogram hasilnya.

$$dst(x,y) = \begin{cases} thresh & \text{jika } src(x,y) > thresh \\ src(x,y) & \text{jika lainnya} \end{cases}$$

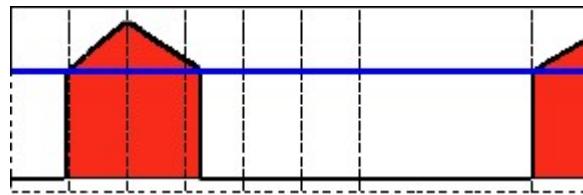


Gambar 6. Histogram hasil dari Truncate Threshold

d. Threshold To Zero

Jika nilai $src(x,y)$ lebih rendah dari nilai $thresh$, maka nilai pixel barunya akan diubah menjadi 0. Berikut adalah persamaan dan gambar Histogram hasil thresholdnya.

$$dst(x,y) = \begin{cases} src(x,y) & \text{jika } src(x,y) > thresh \\ 0 & \text{jika lainnya} \end{cases}$$

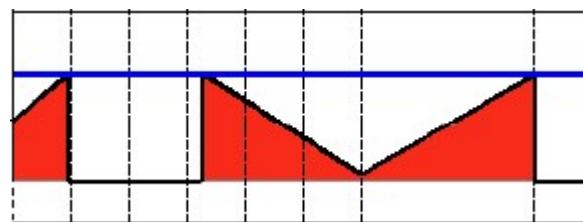


Gambar 7. Histogram hasil dari Threshold To Zero

e. Threshold To Zero – Inverted

Jika nilai $src(x,y)$ lebih besar dari nilai $thresh$, maka nilai pixel barunya akan diubah menjadi 0. Berikut adalah persamaan dan gambar Histogram hasil thresholdnya.

$$dst(x,y) = \begin{cases} 0 & \text{jika } src(x,y) > thresh \\ src(x,y) & \text{jika lainnya} \end{cases}$$



Gambar 8. Histogram hasil dari Threshold To Zero

Global threshold yang telah disebutkan diatas dapat digunakan secara langsung menggunakan library OpenCV cv.Threshold. berikut adalah link dokumentasinya:

https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

Code Berikut menunjukkan penggunaan dari library OpenCV cv.Threshold

```

import cv2 as cv
from google.colab.patches import cv2_imshow
import numpy as np
import matplotlib.pyplot as plt

filename = ('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Images/gradient.jpg')
img = cv.imread(filename)
thresh = 127      #nilai Threshold yang ditentukan

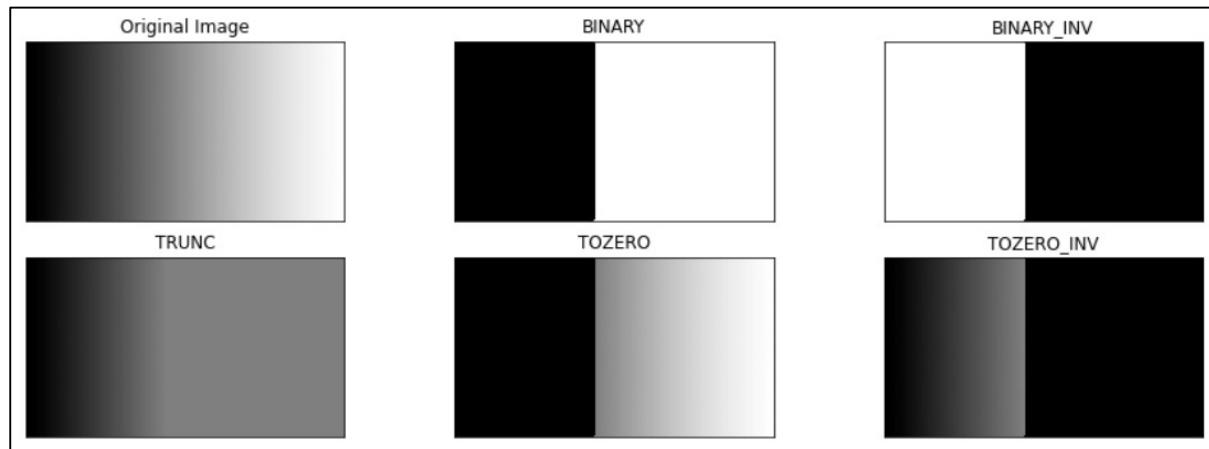
#1. thresh1 jika pixel di img>127, maka thresh1 bernilai 1(putih) selain itu bernilai 0(hitam)
ret,thresh1 = cv.threshold(img,thresh,255,cv.THRESH_BINARY)
#2. thresh2 adalah binary threshold inverse
ret,thresh2 = cv.threshold(img,thresh,255,cv.THRESH_BINARY_INV)
#3. Threshold Truncate
ret,thresh3 = cv.threshold(img,thresh,255,cv.THRESH_TRUNC)
#4. Threshold Tozero
ret,thresh4 = cv.threshold(img,thresh,255,cv.THRESH_TOZERO)
#5. Threshold Tozero Inverse
ret,thresh5 = cv.threshold(img,thresh,255,cv.THRESH_TOZERO_INV)

titles = ['Original Image','BINARY','BINARY_INV','TRUNC', 'TOZERO', 'TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

plt.figure(figsize = (15,5))
for i in range(len(images)):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray', interpolation='nearest')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()

```

Gambar berikut menunjukkan perbedaan masing-masing global threshold yang merupakan keluaran dari code diatas. Seperti yang terlihat pada code tersebut, nilai Global thresholdnya adalah 127 dan nilai tersebut berlaku untuk keseluruhan image.



Gambar 9. Global Threshold dari Library OpenCV

C.3 Adaptive Threshold

Pada bagian sebelumnya, kita menggunakan nilai global sebagai nilai threshold. Hal ini kadang tidak cukup baik di semua kondisi di mana gambar memiliki kondisi pencahayaan berbeda. Pada kasus ini dapat digunakan thresholding adaptif. Algoritma untuk menghitung nilai threshold dikenakan untuk area tertentu dari keseluruhan citra. Sehingga akan didapatkan nilai threshold yang berbeda untuk area yang berbeda dari citra yang sama. Diharapkan dapat memberikan hasil yang lebih baik untuk citra dengan pencahayaan yang berbeda.



Terdapat 2 library yang disediakan yaitu: cv.ADAPTIVE_THRESH_MEAN_C (nilai thresholdnya adalah rata-rata dari area tetangga yang didefinisikan) dan cv.ADAPTIVE_THRESH_GAUSSIAN_C (nilai thresholdnya adalah jumlah bobot dari nilai tetangga dimana bobotnya adalah gaussian window). Area tetangga didefinisikan dengan Block Size, sedangkan C adalah konstanta yang diberikan dimana akan dikurangkan dari nilai rata-rata atau jumlah bobot. Berikut adalah contoh code untuk Adaptive Threshold.

```

filename = ('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Images/sudoku-original.jpg')
citra = cv.medianBlur(cv.imread(filename),5)
gray = cv.cvtColor(citra, cv.COLOR_BGR2GRAY)
#gray = cv.medianBlur(gray,5)

thresh = 127

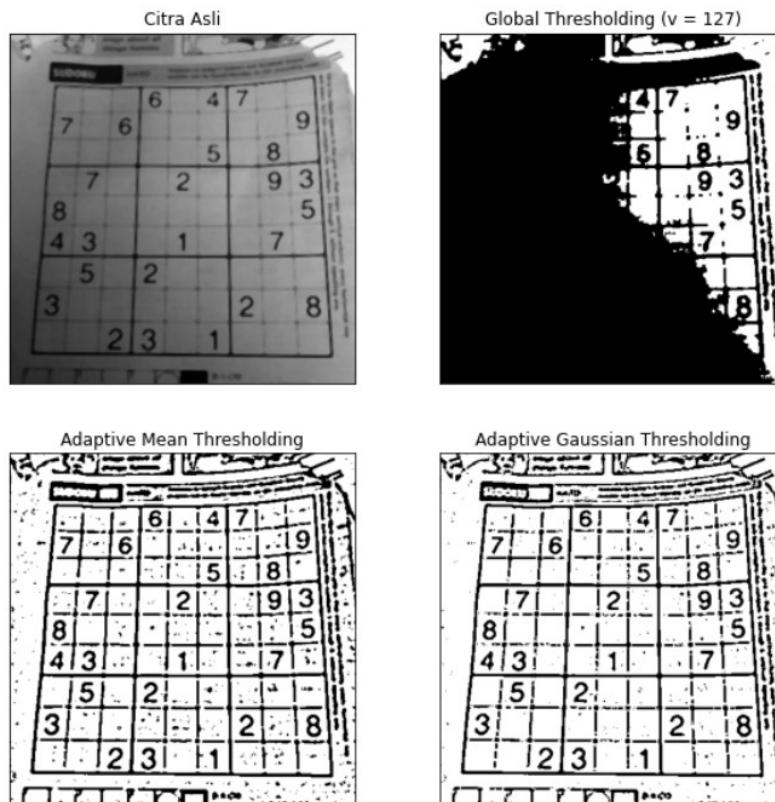
ret,th1 = cv.threshold(gray,thresh,255,cv.THRESH_BINARY)
th2 = cv.adaptiveThreshold(gray,255,cv.ADAPTIVE_THRESH_MEAN_C, cv.THRESH_BINARY,11,2)
th3 = cv.adaptiveThreshold(gray,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.THRESH_BINARY,11,2)

titles = ['Citra Asli', 'Global Thresholding (v = 127)', 'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
citra2 = [gray, th1, th2, th3]

plt.figure(figsize = (10,10))
for i in range(len(citra2)):
    plt.subplot(2,2,i+1),plt.imshow(citra2[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()

```

Berikut adalah gambar hasil dari code diatas.



Gambar 10. Citra hasil Adaptive Threshold Mean dan Gaussian Window

C.4 Otsu's Threshold

Otsu's threshold adalah metode segmentasi otomatis yang sangat populer, bahkan sampai hari ini pun masih sering digunakan sebagai standar metode binary threshold yang handal. Threshold ini dinamakan sesuai dengan nama penemunya yaitu Nobuyuki Otsu. Dalam bentuk yang paling sederhana, algoritma ini akan melakukan segmentasi pixel menjadi dua kelas, yaitu foreground dan background. Threshold ini ditentukan dengan meminimalkan intensitas variance intra-class, atau bisa juga dengan memaksimalkan variance inter-class.

Algoritma Otsu's secara iteratif mencari threshold yang terkecil dari variance intra-class, yang didefinisikan sebagai jumlah bobot dari variance dua class:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Bobot ω_0 dan ω_1 adalah probabilitas dari 2 class yang dipisahkan oleh threshold t . σ_0^2 dan σ_1^2 adalah variance dari masing-masing class.

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i)$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i)$$

Mencari nilai minimal dari variance intra-class ekivalen dengan mencari nilai maksimal dari variance inter-class.

$$\sigma_B^2(t) = \sigma^2 - \sigma_w^2(t) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2$$

$$= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$$

Dimana nilai mean class $\mu_0(t)$, $\mu_1(t)$, dan μ_T adalah:

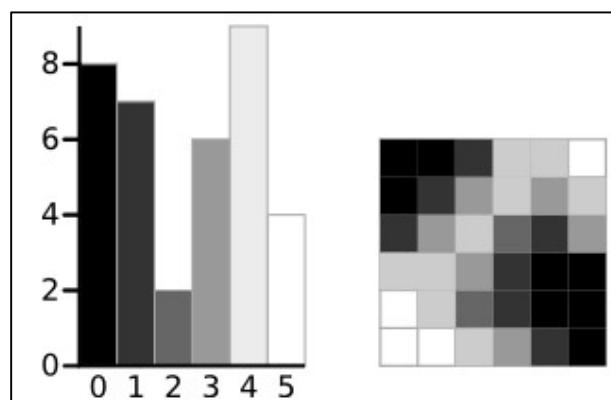
$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)}$$

$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i)$$

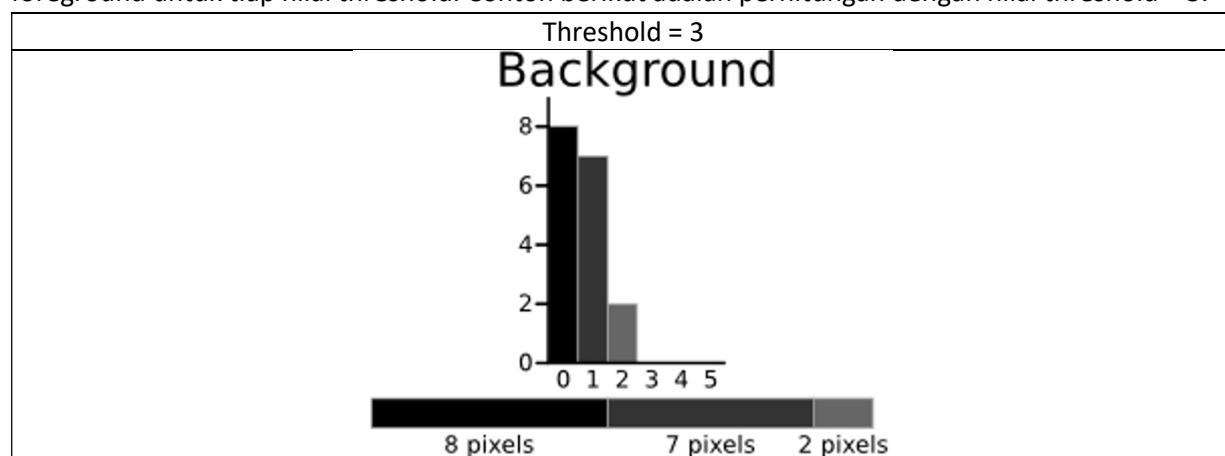
Perhatikan contoh perhitungan algoritma Otsu's berikut:

Gambar berikut menunjukkan sebuah image resolusi 6x6 dan terdiri dari 6 warna, beserta histogramnya.



Gambar 11. Contoh citra resolusi 6x6 beserta histogramnya

Dari gambar diatas, lakukan perhitungan Weight, Mean, Variance untuk tiap background dan foreground untuk tiap nilai threshold. Contoh berikut adalah perhitungan dengan nilai threshold = 3.



Gambar 12. Histogram untuk background object dengan T=3

$$\text{weight: } \omega_b = \frac{8+7+2}{36} = 0.4722$$

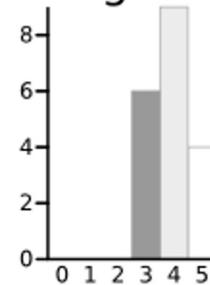
$$\text{mean } \mu_b = \frac{(0 \times 8)+(1 \times 7)+(2 \times 2)}{17} = 0.6471$$

variance

$$\sigma_b^2 = \frac{((0 - 0.6471)^2 \times 8) + ((1 - 0.6471)^2 \times 7) + ((2 - 0.6471)^2 \times 2)}{17}$$

$$= \frac{(0.4187 \times 8) + (0.1246 \times 7) + (1.8304 \times 2)}{17} = \mathbf{0.4637}$$

Foreground



6 pixels 9 pixels 4 pixels

Gambar 13. Histogram untuk Foreground object dengan T=3

$$\text{weight: } \omega_f = \frac{6+9+4}{36} = 0.5278$$

$$\text{mean } \mu_f = \frac{(3 \times 6) + (4 \times 9) + (5 \times 4)}{19} = 3.8947$$

variance

$$\sigma_f^2 = \frac{((3 - 3.8947)^2 \times 6) + ((4 - 3.8947)^2 \times 9) + ((5 - 3.8947)^2 \times 4)}{19}$$

$$= \frac{(4.8033 \times 6) + (0.0997 \times 9) + (4.8864 \times 4)}{19} = 0.5152$$

Selanjutnya nilai within class variance dari 2 variance (background dan foreground) dapat dihitung dengan :

Within class variance:

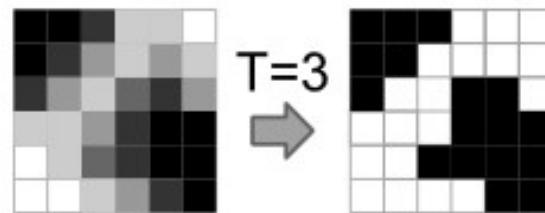
$$\sigma_W^2 = \omega_b \sigma_b^2 + \omega_f \sigma_f^2 = 0.4722 \times 0.4637 + 0.5278 \times 0.5152 = 0.4909$$

Perhitungan ini dilakukan untuk setiap nilai threshold yang mungkin ($0 \leq T \leq 5$). Nilai Within class variance terkecil akan menjadi threshold yang terpilih.

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
Weight, Background	$W_b = 0$	$W_b = 0.222$	$W_b = 0.4167$	$W_b = 0.4722$	$W_b = 0.6389$	$W_b = 0.8889$
Mean, Background	$M_b = 0$	$M_b = 0$	$M_b = 0.4667$	$M_b = 0.6471$	$M_b = 1.2609$	$M_b = 2.0313$
Variance, Background	$\sigma_b^2 = 0$	$\sigma_b^2 = 0$	$\sigma_b^2 = 0.2489$	$\sigma_b^2 = 0.4637$	$\sigma_b^2 = 1.4102$	$\sigma_b^2 = 2.5303$
Weight, Foreground	$W_f = 1$	$W_f = 0.7778$	$W_f = 0.5833$	$W_f = 0.5278$	$W_f = 0.3611$	$W_f = 0.1111$
Mean, Foreground	$M_f = 2.3611$	$M_f = 3.0357$	$M_f = 3.7143$	$M_f = 3.8947$	$M_f = 4.3077$	$M_f = 5.000$
Variance, Foreground	$\sigma_f^2 = 3.1196$	$\sigma_f^2 = 1.9639$	$\sigma_f^2 = 0.7755$	$\sigma_f^2 = 0.5152$	$\sigma_f^2 = 0.2130$	$\sigma_f^2 = 0$
Within Class Variance	$\sigma_W^2 = 3.1196$	$\sigma_W^2 = 1.5268$	$\sigma_W^2 = 0.5561$	$\sigma_W^2 = 0.4909$	$\sigma_W^2 = 0.9779$	$\sigma_W^2 = 2.2491$

Gambar 14. Perhitungan Within Class Variance untuk seluruh nilai T yang mungkin

Gambar berikut menunjukkan hasil citra yang dithreshold dengan Otsu (T=3)



Gambar 15. Citra Threshold hasil Otsu's dengan nilai T=3

Penentuan nilai threshold yang terpilih juga bisa dilakukan menggunakan perhitungan Between class variance, dimana nilai Between class variance terbesar adalah threshold yang terpilih.

Between class variance:

$$\begin{aligned}\sigma_B^2 &= \sigma^2 - \sigma_W^2 \\ &= \omega_b(\mu_b - \mu)^2 + \omega_f(\mu_f - \mu)^2 \quad (\text{dimana } \mu = \omega_b\mu_b + \omega_f\mu_f) \\ &= \omega_b\omega_f(\mu_b - \mu_f)^2\end{aligned}$$

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
Within Class Variance	$\sigma_W^2 = 3.1196$	$\sigma_W^2 = 1.5268$	$\sigma_W^2 = 0.5561$	$\sigma_W^2 = 0.4909$	$\sigma_W^2 = 0.9779$	$\sigma_W^2 = 2.2491$
Between Class Variance	$\sigma_B^2 = 0$	$\sigma_B^2 = 1.5928$	$\sigma_B^2 = 2.5635$	$\sigma_B^2 = 2.6287$	$\sigma_B^2 = 2.1417$	$\sigma_B^2 = 0.8705$

Otsu's telah disediakan pula dalam library OpenCV (cv.THRESH_OTSU) yang biasanya dikombinasikan dengan binary threshold(cv.THRESH_BINARY). Library ini dipanggil dengan object cv.threshold. Berikut adalah code contoh penggunaan Otsu's dengan library OpenCV.

```
# Dengan Library

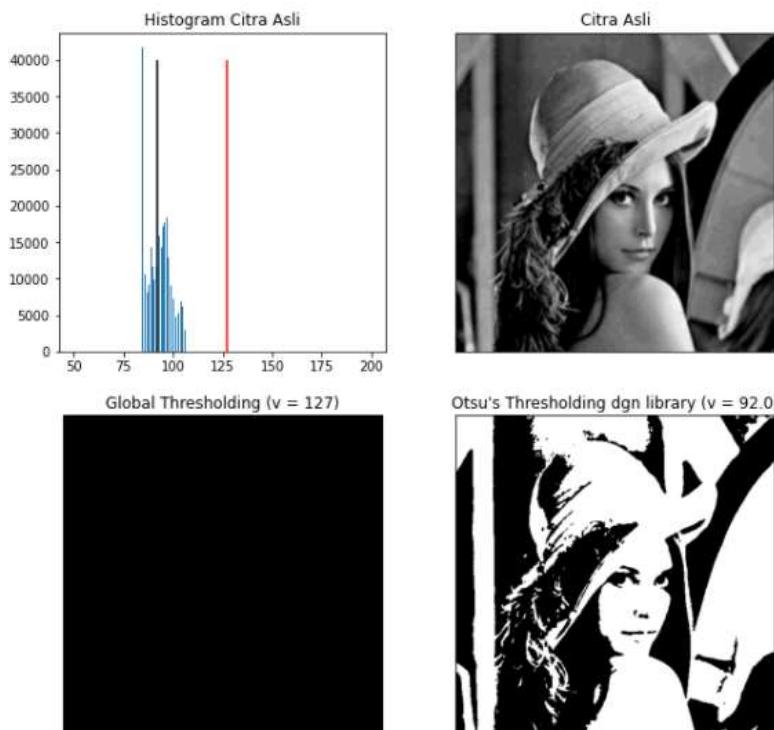
filename = ('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Images/lena_gs_lc2.jpg')
img = cv.imread(filename,0)
blur = cv.GaussianBlur(img,(5,5),0)
thresh = 127

ret,th1 = cv.threshold(blur,thresh,255,cv.THRESH_BINARY)
ret2,th2 = cv.threshold(blur,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)

x = ("Otsu's Thresholding dgn library (v = ")+str(ret2)+"")"
titles = ['Citra Asli', 'Global Thresholding (v = 127)', x]
citra3 = [blur, th1, th2]

plt.figure(figsize = (10,10))
plt.subplot(2,2,1),plt.hist(blur.ravel(),256,[50,200])
plt.vlines(ret,0,40000,colors='red')      #garis vertikal merah menunjukkan threshold global 127
plt.vlines(ret2,0,40000,colors='black')   #garis vertikal hitam menunjukkan threshold 92 hasil otsu's
plt.title('Histogram Citra Asli')
for i in range(len(citra3)):
    plt.subplot(2,2,i+2),plt.imshow(citra3[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

Gambar berikut menunjukkan hasil dari code diatas. Garis vertikal merah pada histogram menunjukkan nilai threshold 127. Dari gambar tersebut terlihat bahwa warna disebelah kiri garis merah akan diubah semua ke warna hitam, sehingga hasil thresholdnya akan menjadi warna hitam semua. Berbeda dengan hasil dari otsu's yang masih dapat memisahkan dua class background dan foreground. Garis vertikal hitam menunjukkan nilai threshold yang dihasilkan oleh otsu's. terlihat bahwa garis tersebut masih bisa membagi histogram menjadi dua bagian yaitu background dan foreground.



Gambar 16. Citra hasil threshold menggunakan binary threshold dan Otsu's

Code berikut akan menampilkan hasil dari threshold Otsu's tanpa Gaussian Filter dan dengan menggunakan Gaussian Filter.

```

filename = ('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Images/noisy2.png')
img = cv.imread(filename,0)

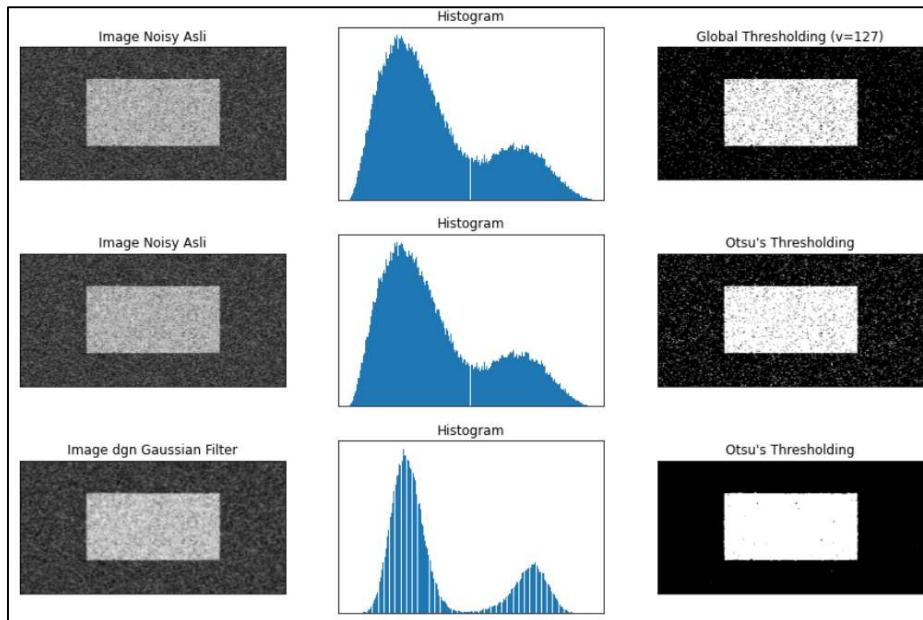
#Global Thresholding
ret1,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
# Otsu's thresholding
ret2,th2 = cv.threshold(img,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
# Otsu's thresholding setelah dilakukan Gaussian filtering
blur = cv.GaussianBlur(img,(5,5),0)
ret3,th3 = cv.threshold(blur,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)

#plotting semua image
images = [img, 0, th1,
          img, 0, th2,
          blur, 0, th3]
titles = ['Image Noisy Asli','Histogram','Global Thresholding (v=127)',
          'Image Noisy Asli','Histogram',"Otsu's Thresholding",
          'Image dgn Gaussian Filter','Histogram',"Otsu's Thresholding"]

plt.figure(figsize = (15,10))
for i in range(3):
    plt.subplot(3,3,i*3+1),plt.imshow(images[i*3],'gray')
    plt.title(titles[i*3]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+2),plt.hist(images[i*3].ravel(),256)
    plt.title(titles[i*3+1]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+3),plt.imshow(images[i*3+2],'gray')
    plt.title(titles[i*3+2]), plt.xticks([]), plt.yticks([])
plt.show()

```

Gambar berikut menunjukkan hasil dari code diatas. Dapat disimpulkan bahwa dengan melakukan filtering terlebih dahulu terhadap citra masukan, hasil threshold Otsu's dapat lebih baik.



Gambar 17. Citra Otsu's tanpa preprocess dan dengan preprocess Gaussian Filter

C.5 Sementasi Citra menggunakan K-Means

K-Means merupakan metode Clustering yang biasa digunakan untuk melakukan segmentasi citra dengan penentuan jumlah cluster sesuai dengan yang kita harapkan. Berikut merupakan link penjelasan K-Means secara umum:

- https://youtu.be/_aWzGGNrcic
- <https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html>

Berikut adalah code untuk penggunaan K-Means pada segmentasi citra.

```
#KMeans Image Segmentation
filename = ('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Images/jungle.png')
)
img = cv.imread(filename)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
'''
kita akan menggunakan fungsi cv.kmeans() yang meminta array 2D sebagai masukan, sedangkan image aslinya adalah array 3D
selanjutnya kita perlu melakukan flattening array image masukan
'''
#reshape array ke bentuk 2D
pixel_values = img.reshape((-1, 3))
# convert to float
pixel_values = np.float32(pixel_values)
```

```

"""
syarat berhenti iterasi dr KMeans adalah jika centroid sudah tidak terlalu
banyak pergeseran posisi antara interasi sekarang
dengan iterasi sebelumnya (konvergen). Karena jumlah data yang besar, maka
kita akan hentikan iterasi saat jumlah iterasi = 100
atau epsilon(selisih antara posisi centroid skrg dgn posisi centroid di ite
rasi sebelumnya) < 0.2
"""

criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 100, 0.2)
"""

jika diperhatikan pada image asli, terdapat 3 warna utama (hijau, biru, dan
putih/orange). untuk percobaan ini kita akan gunakan
3 cluster untuk image ini
"""

k = 3

_, labels, (centers) = cv.kmeans(pixel_values, k, None, criteria, 10, cv.KM
EANS_RANDOM_CENTERS)

#konversi titik centroid kedalam integer
centers = np.uint8(centers)

#flattening label array
labels = labels.flatten()

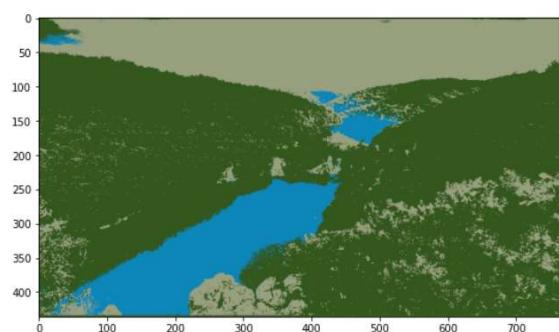
#konversi warna pixel asli kewarna dari tiap centroidnya
segmented_image = centers[labels.flatten()]

# reshape ke bentuk image asli
segmented_image = segmented_image.reshape(img.shape)

plt.figure(figsize = (20,20))
plt.subplot(1,2,1),plt.imshow(img)
plt.subplot(1,2,2),plt.imshow(segmented_image)

```

Berikut adalah hasil dari code diatas



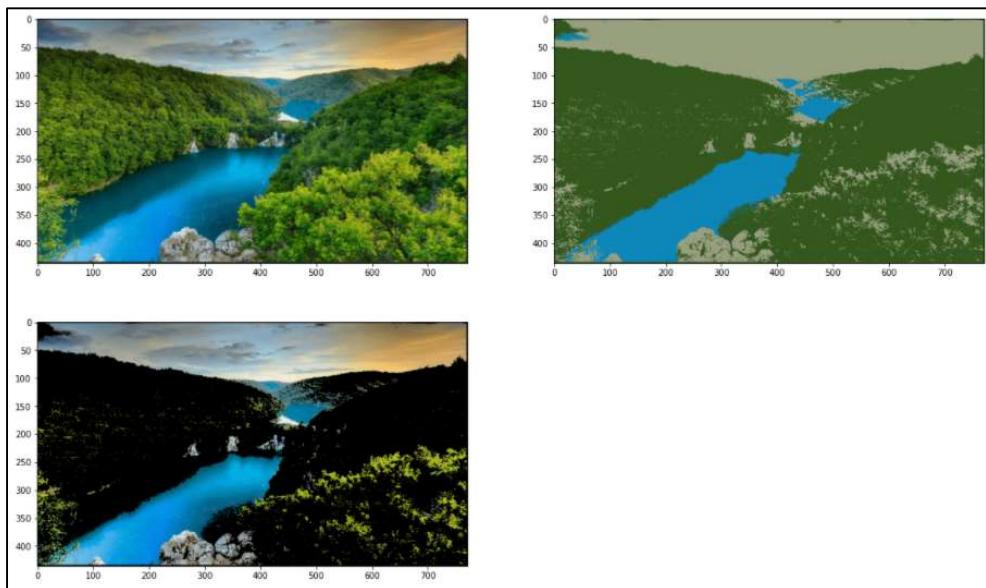
Gambar 18. Hasil segmentasi citra menggunakan K-Means Clustering

Code berikut merupakan lanjutan dari code diatas yang digunakan untuk mengubah warna pada cluster tertentu.

```
# ubah pixel di cluster 2 menjadi hitam
masked_image = np.copy(img)
# konvert ke bentuk vektor
masked_image = masked_image.reshape((-1, 3))
# cluster yang diubah
cluster = 2
masked_image[labels == cluster] = [0, 0, 0]
# konvert ke bentuk asli
masked_image = masked_image.reshape(img.shape)

plt.figure(figsize = (20,12))
plt.subplot(2,2,1),plt.imshow(img)
plt.subplot(2,2,2),plt.imshow(segmented_image)
plt.subplot(2,2,3),plt.imshow(masked_image)
```

Berikut adalah gambar hasil dari code diatas.



Gambar 19. Hasil dari perubahan nilai warna pada cluster tertentu

D. TUGAS PRAKTIKUM

1. Buka <https://colab.research.google.com/>. Setelah dipastikan bahwa google Colab terhubung dengan Github Anda, lanjutkan dengan memilih repository yang telah digunakan pada praktikum minggu lalu, rename file menjadi “Week9.ipynb”.

Kemudian import folder yang ada di Drive Anda dengan cara sebagai berikut.

```
[1]: from google.colab import drive
drive.mount('/content/drive')
```

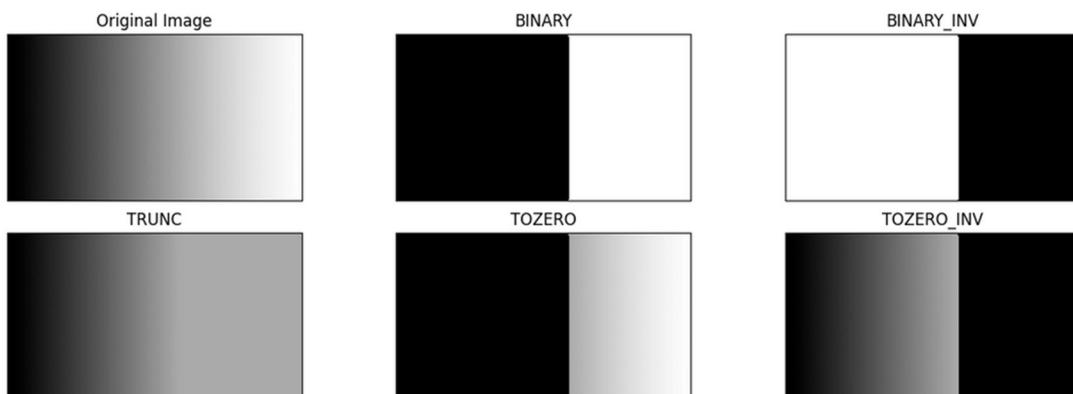
Mounted at /content/drive

- Import beberapa library berikut yang akan digunakan selama uji coba praktikum minggu ke-6 berikut.

Import Library

```
[2]: import cv2
import numpy as np
from matplotlib import pyplot as plt
```

- Buat Global Threshold (BINARY, BINARY_INV, TRUNC, TOZERO, TOZERO_INV), dengan threshold= 170, secara manual sesuai dengan deskripsi dari grafik yang ditunjukkan di atas.



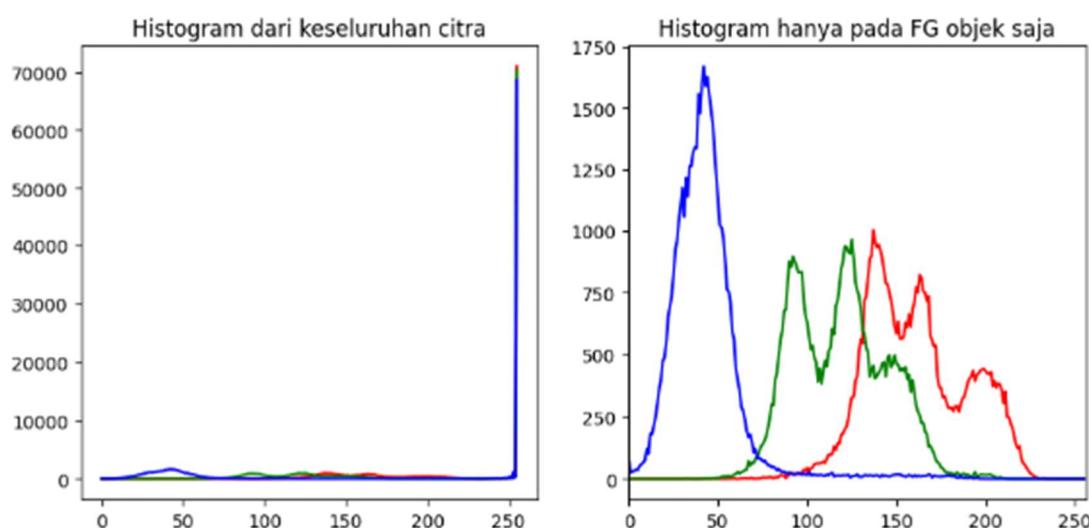
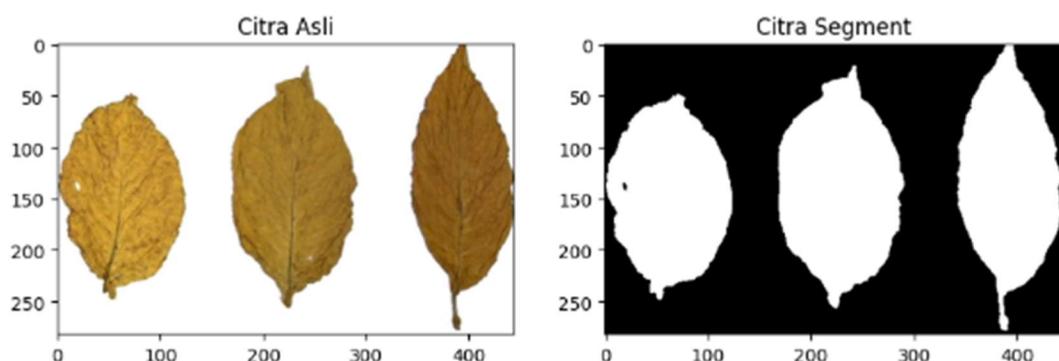
- Buat Otsu Thresholding tanpa menggunakan Library. Tampilkan juga nilai threshold saat anda gunakan Otsu's, seperti terlihat pada gambar hasil berikut. (gunakan image **balloon.jpg** agar terlihat beda antara hasil otsu's dengan global threshold biasa)



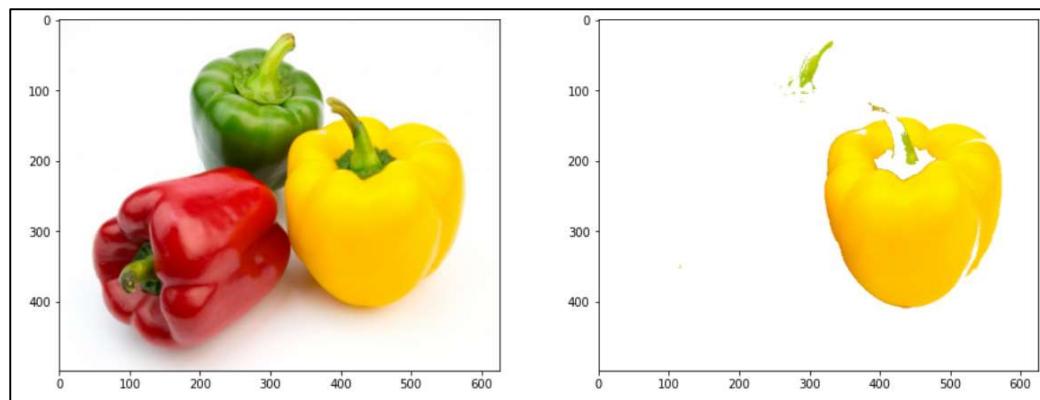
5. Buat histogram dari citra tersegmentasi, Histogram hanya pada foreground image saja. Gunakan image tobacco.jpg yang sudah disediakan di folder images.

Petunjuk:

- anda dapat gunakan cv.calcHist untuk menampilkan histogram.
- Buka link berikut <https://opencv-tutorial.readthedocs.io/en/latest/histogram/histogram.html>
- Dari link tersebut perhatikan bahwa cv.calcHist memiliki salah satu parameter yaitu mask. Jika diset **None**, maka keseluruhan image akan dihitung histogramnya. Jika kita tentukan mask, maka hanya bagian image yang dimasking **warna putih** yang akan dihitung histogramnya (dari contoh dibawah dinamakan dengan **Citra Segment**).



6. Lakukan segmentasi warna pada image "peppers.jpg", munculkan hanya warna yang kuning saja. (Petunjuk: anda dapat gunakan K-Means untuk menampilkan hanya warna tertentu saja)



Saat anda menampilkan warna tertentu, jelaskan kendala yang di hadapi dan mengapa hal itu terjadi.

.....

.....

.....

.....

.....

7. Buka File crossword.jpg. Dengan pengetahuan tentang thresholding yang sudah anda pelajari. Lakukan thresholding biner dengan hasil terbaik menurut anda. Copas code dan hasil image di modul ini.

--- SELAMAT BELAJAR ---