PENGOLAHAN CITRA & VISI KOMPUTER
RTI235007

Minggu 10: Thresholding dan Segmentation

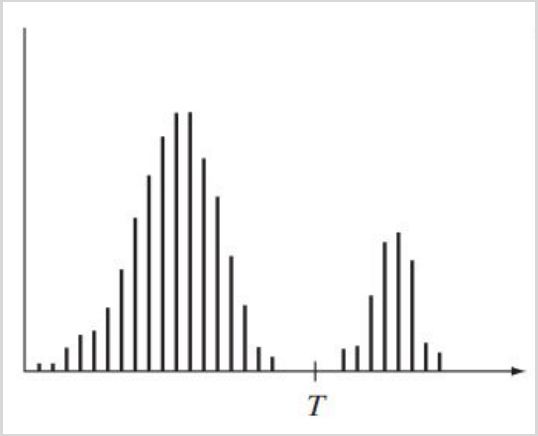
Outline

- Global Thresholding
- Adaptive Thresholding
- Otsu’s Thresholding
- K-Means clustering

Thresholding

- 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.

Global Threshold

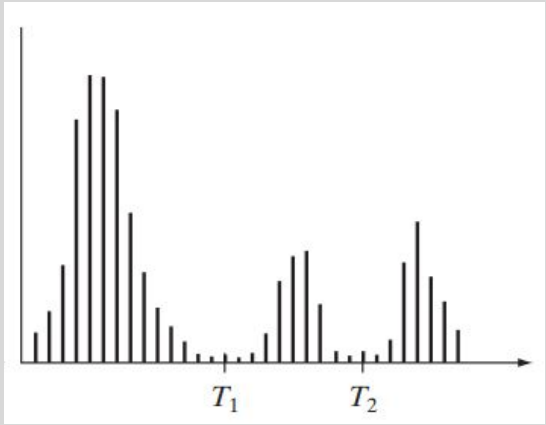


- Dengan asumsi bahwa object dan background memiliki nilai intensitas warna yang dapat dipisahkan dalam 2 grup dominan.
- Cara yang termudah dan paling jelas adalah dengan memilih nilai tertentu (**Threshold, *T***) yang akan memisahkan dua grup tersebut. Tiap titik (x,y) dalam image dimana f(x,y) > T dapat diistilahkan dengan *object point*, sedangkan grup satunya disebut dengan *background point*.
- $g(x,y) = \begin{cases} 1 & jika\ f(x,y) > T \\ 0 & jika\ f(x,y) \leq T \end{cases}$

Global Threshold

- 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.

Global Threshold



- ◦Proses multiple threholding akan mengelompokkan titik (x,y) sebagai background jika f(x,y) <= T_1, ke grup object satu jika T_1 < f(x,y) <= T_2, dan object grup lain jika f(x,y) > T_2. Atau dapat dituliskan sebagai berikut:

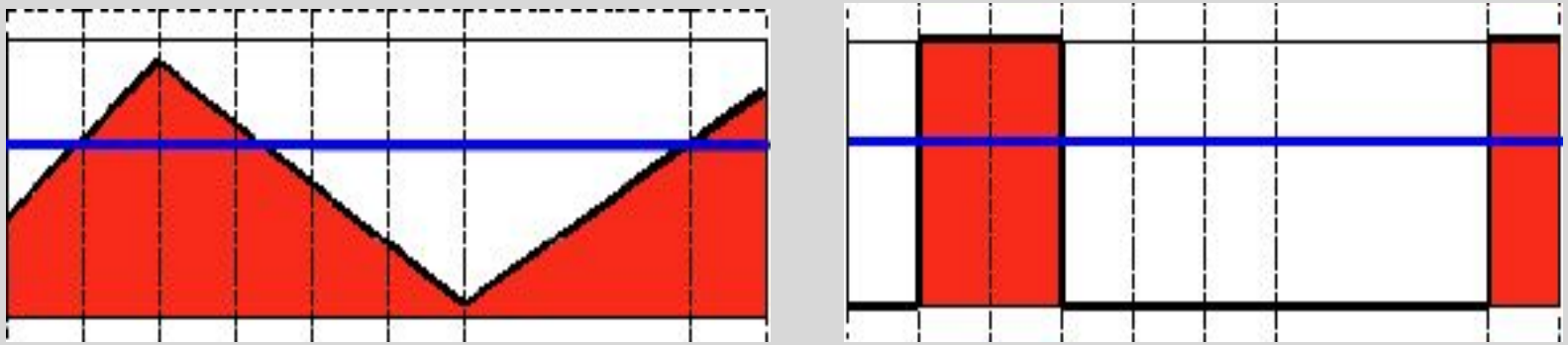
$$\circ g(x,y) = \begin{cases} a & jika\ f(x,y) > T_2 \\ b & jika\ T_1 < f(x,y) \leq T_2 \\ c & jika\ f(x,y) < T \end{cases}$$

Global Threshold

- Beberapa Global Thresholding yang disediakan OpenCV:
  - Binary Threshold
  - Binary-Inverted Threshold
  - Truncate Threshold
  - Threshold To Zero
  - Threshold To Zero-Inverted

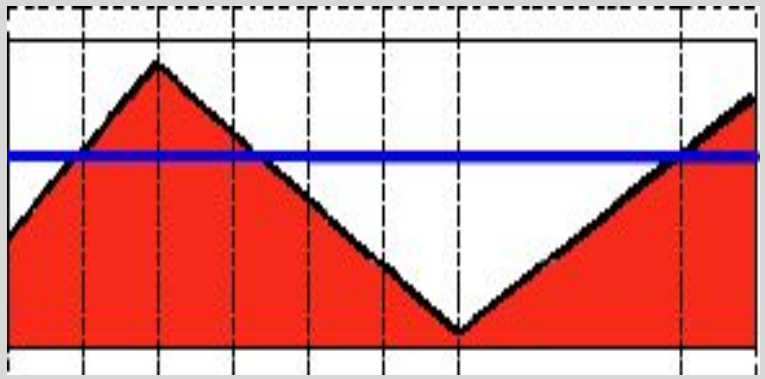
Binary Threshold

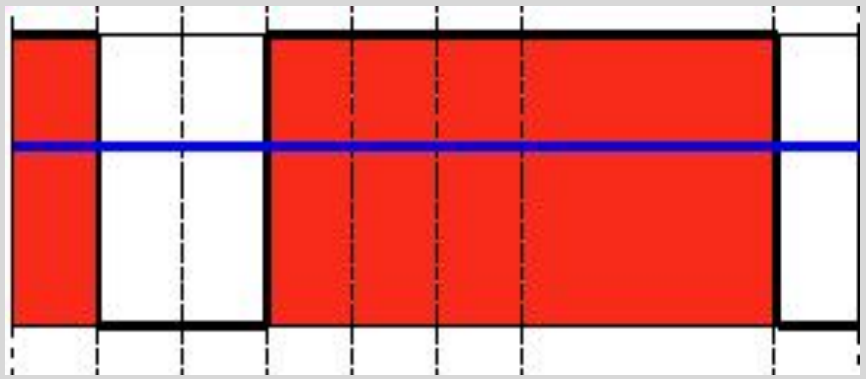
- Masing-masing kelompok warna akan diubah ke nilai gelap (hitam) untuk object yang dianggap background, dan diubah ke nilai terang (putih) untuk object yang dianggap foreground.
- $$dst(x,y) = \begin{cases} maxVal & jika\ src(x,y) > thresh \\ 0 & jika\ lainnya \end{cases}$$



Binary Inverted Threshold

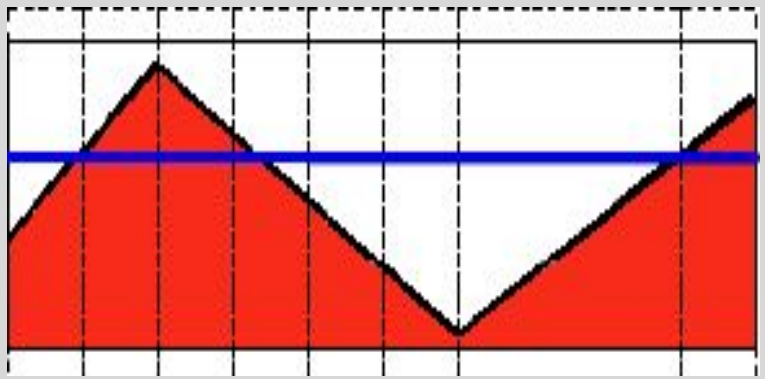
- Jika intensitas warna diatas Threshold, maka nilai akan diubah menjadi 0 dan sebaliknya.
- $$dst(x,y) = \begin{cases} 0 & jika\ src(x,y) > thresh \\ maxVal & jika\ lainnya \end{cases}$$

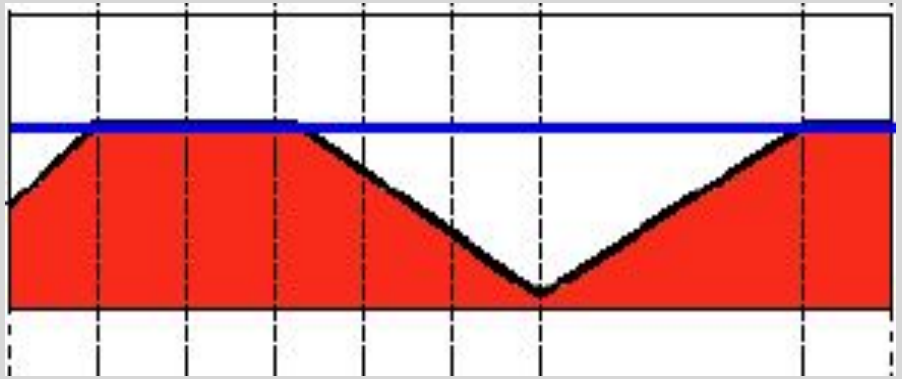




Truncate Threshold

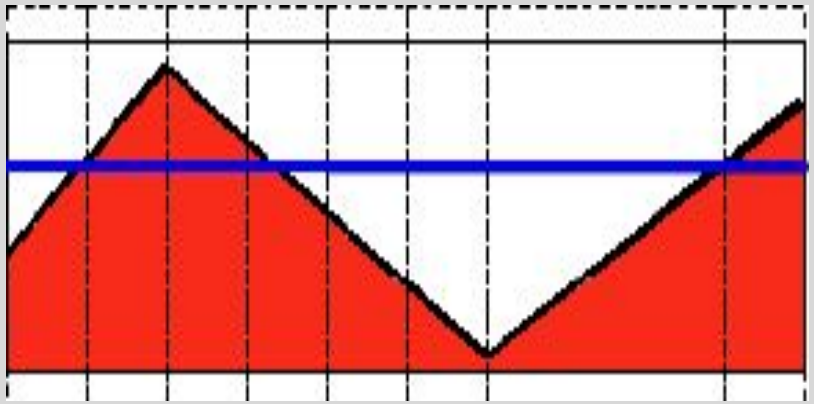
- Jika nilai intensitas warna *src(x,y)* lebih besar dari *thresh*, maka nilainya akan ditruncate.
- $$dst(x,y) = \begin{cases} thresh & jika\ src(x,y) > thresh \\ src(x,y) & jika\ lainnya \end{cases}$$

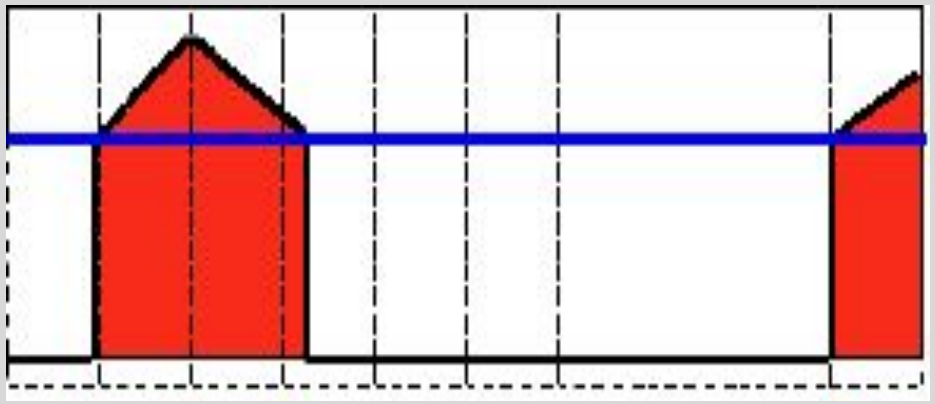




Threshold To Zero

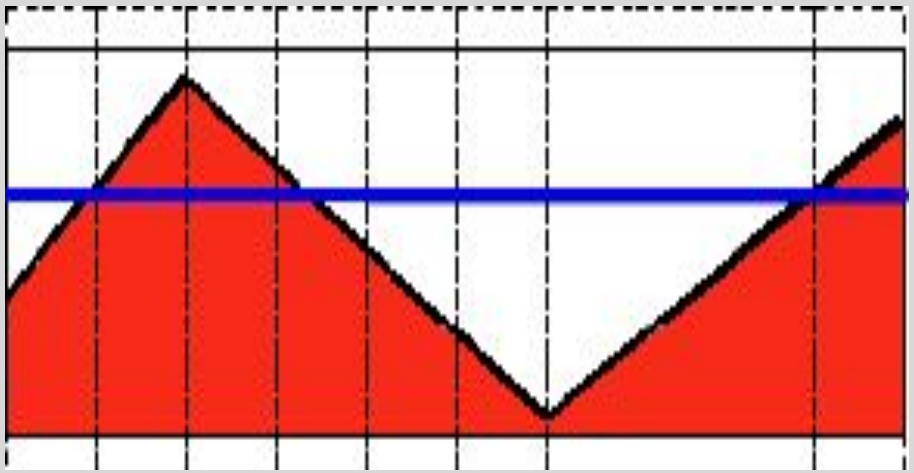
- Jika nilai *src(x,y)* lebih rendah dari nilai *thresh*, maka nilai pixel barunya akan diubah menjadi 0.
- $$dst(x,y) = \begin{cases} src(x,y) & jika\ src(x,y) > thresh \\ 0 & jika\ lainnya \end{cases}$$

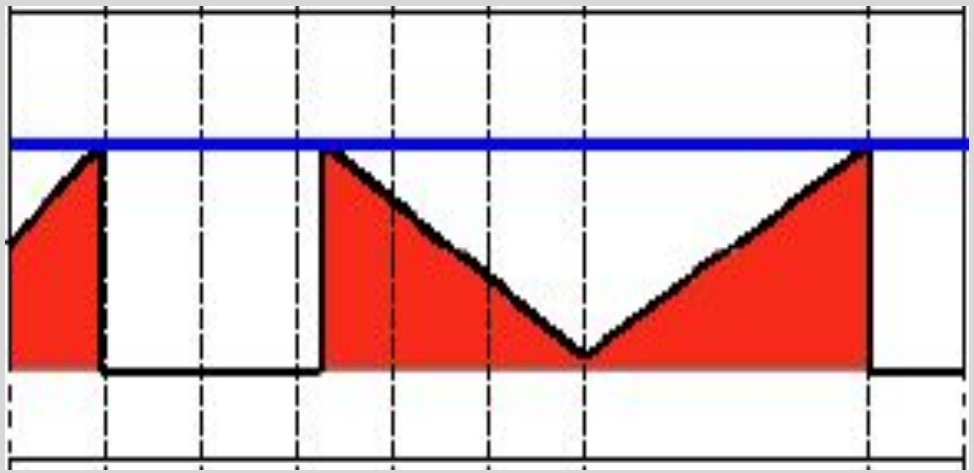




Threshold To Zero - Inverted

- Jika nilai *src(x,y)* lebih besar dari nilai *thresh*, maka nilai pixel barunya akan diubah menjadi 0.
- $$dst(x,y) = \begin{cases} 0 & jika\ src(x,y) > thresh \\ src(x,y) & jika\ lainnya \end{cases}$$

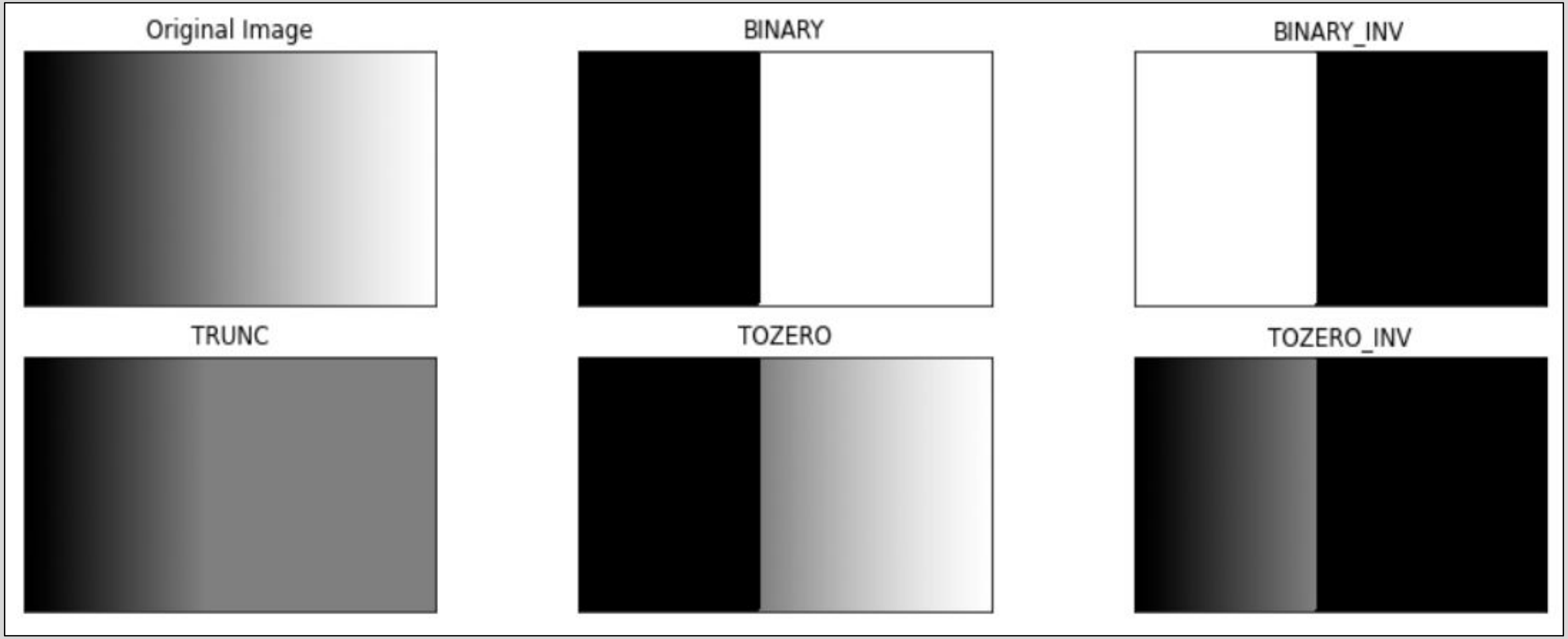




Global Threshold di OpenCV

```
#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)
```

Global Threshold



Adaptive Threshold

- Global Threshold terkadang tidak cukup baik di semua kondisi di mana citra memiliki kondisi pencahayaan berbeda. Pada kasus khusus ini dapat digunakan thresholding adaptif.
- Algoritma ini akan menghitung nilai threshold yang 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.

Adaptive Threshold

- 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.

Adaptive Threshold

```
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)
```

Citra Asli

Global Thresholding (v = 127)

Adaptive Mean Thresholding

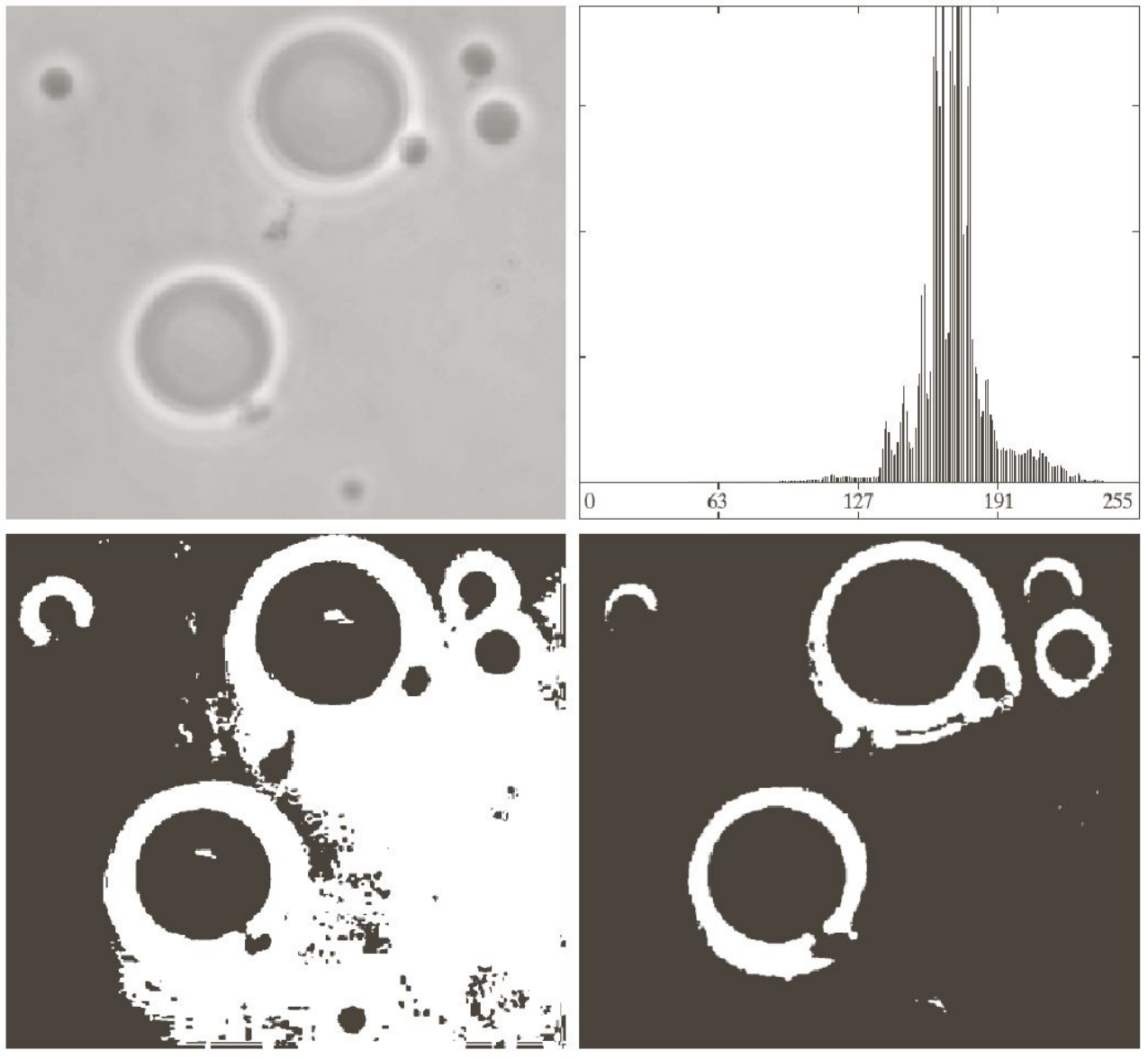
Adaptive Gaussian Thresholding

Otsu's Method

- Algoritma ini secara langsung akan mencari nilai threshold yang meminimasikan variance intra-class, yang dihitung dengan menjumlahkan bobot variance dari dua class:

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

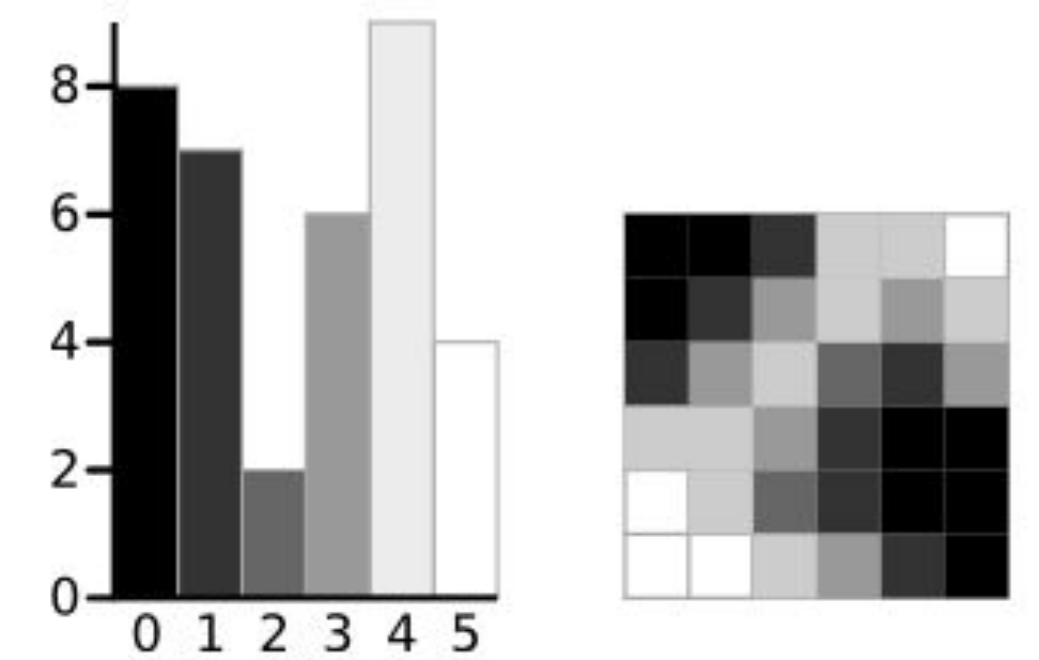
- Bobot ω_0 dan ω_1 adalah probabilitas dari 2 class yang dipisahkan oleh threshold t. σ_0^2 dan σ_1^2 adalah variance dari 2 class.
- Probabilitas class $\omega_{0,1}(t)$ dihitung dari L bins dari histogram:
  - $\omega_0(t) = \sum_{i=0}^{t-1} p(i)$
  - $\omega_1(t) = \sum_{i=t}^{L-1} p(i)$



a b
c d

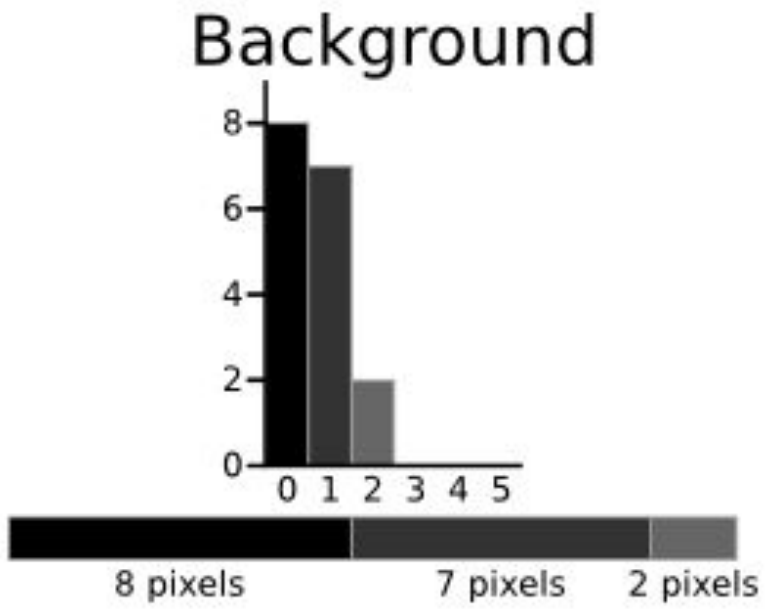
FIGURE 10.39
(a) Original image.
(b) Histogram (high peaks were clipped to highlight details in the lower values).
(c) Segmentation result using the basic global algorithm from Section 10.3.2.
(d) Result obtained using Otsu's method.
(Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)

Otsu's Threshold



Otsu's Threshold

- Lakukan perhitungan Weight, Mean, Variance untuk background dan foreground. Berikut adalah contoh pada nilai threshold 3.



$$\text{Weight} \quad W_b = \frac{8+7+2}{36} = 0.4722$$

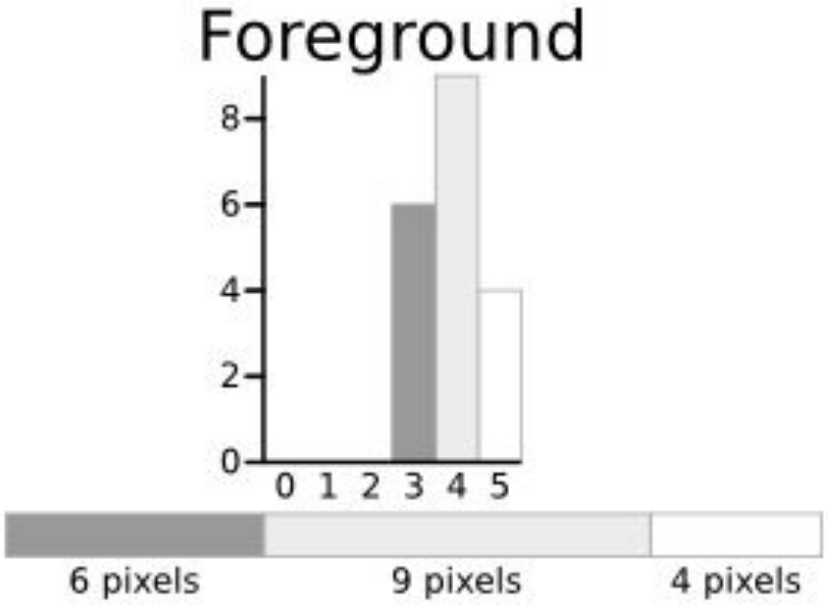
$$\text{Mean} \quad \mu_b = \frac{(0\times 8)+(1\times 7)+(2\times 2)}{17} = 0.6471$$

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

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

$$= 0.4637$$

Otsu's Threshold



$$\text{Weight} \quad W_f = \frac{6+9+4}{36} = 0.5278$$

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

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

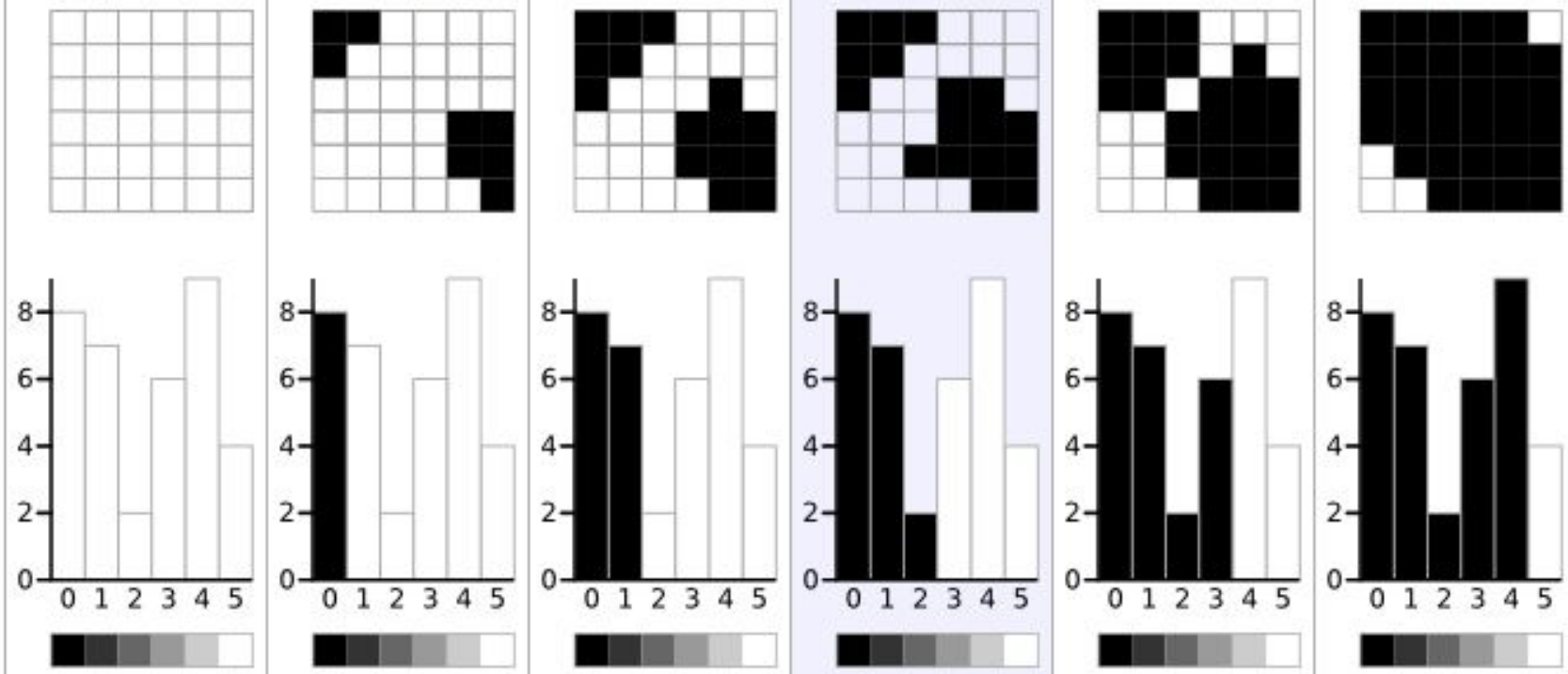
$$= \frac{(4.8033\times 6)+(0.0997\times 9)+(4.8864\times 4)}{19}$$

$$= 0.5152$$

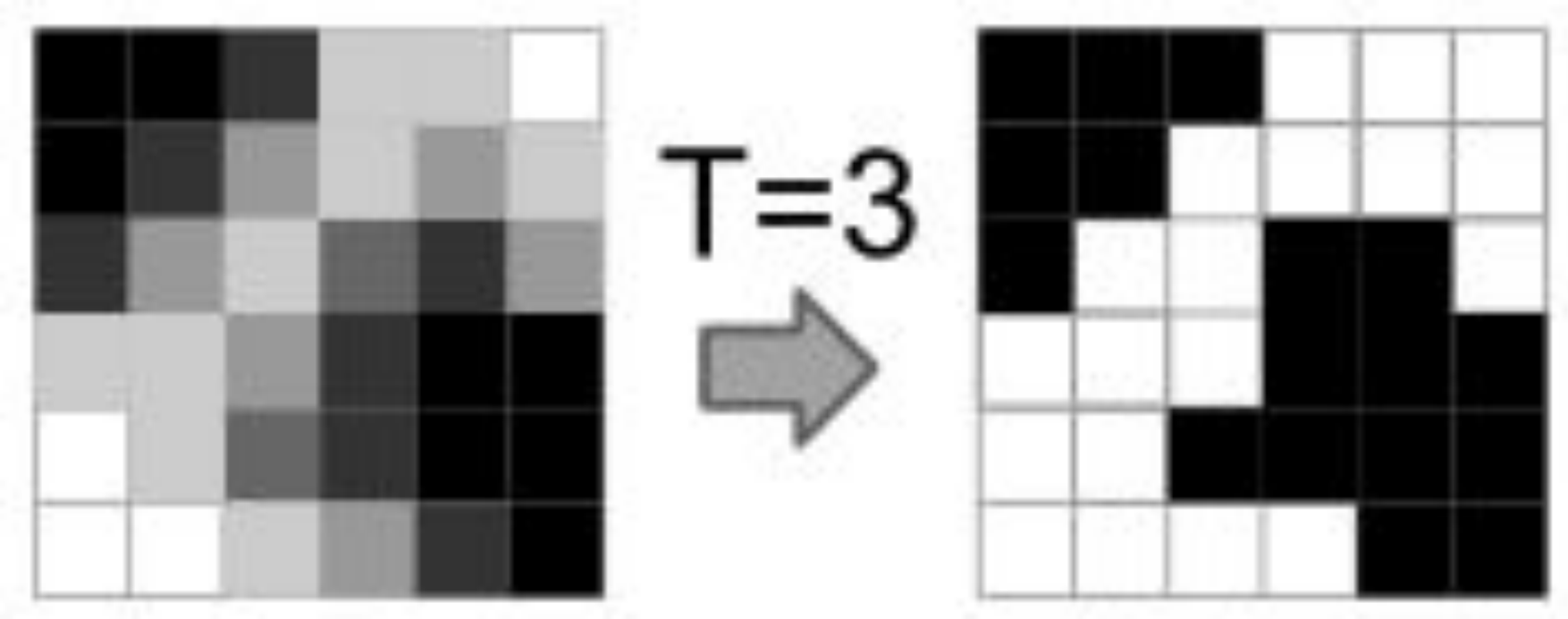
Otsu's Threshold

- Hitung Within Class Variance untuk tiap kemungkinan threshold. Threshold dengan nilai Within Class Variance terkecil adalah threshold yang terpilih

$$\text{Within Class Variance} \quad \sigma_W^2 = W_b\,\sigma_b^2 + W_f\,\sigma_f^2 = 0.4722 * 0.4637 + 0.5278 * 0.5152$$
$$= 0.4909$$

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	$\sigma^2_b = 0$	$\sigma^2_b = 0$	$\sigma^2_b = 0.2489$	$\sigma^2_b = 0.4637$	$\sigma^2_b = 1.4102$	$\sigma^2_b = 2.5303$
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	$\sigma^2_f = 3.1196$	$\sigma^2_f = 1.9639$	$\sigma^2_f = 0.7755$	$\sigma^2_f = 0.5152$	$\sigma^2_f = 0.2130$	$\sigma^2_f = 0$
Within Class Variance	$\sigma^2_W = 3.1196$	$\sigma^2_W = 1.5268$	$\sigma^2_W = 0.5561$	$\mathbf{\sigma^2_W = 0.4909}$	$\sigma^2_W = 0.9779$	$\sigma^2_W = 2.2491$

Otsu's Threshold



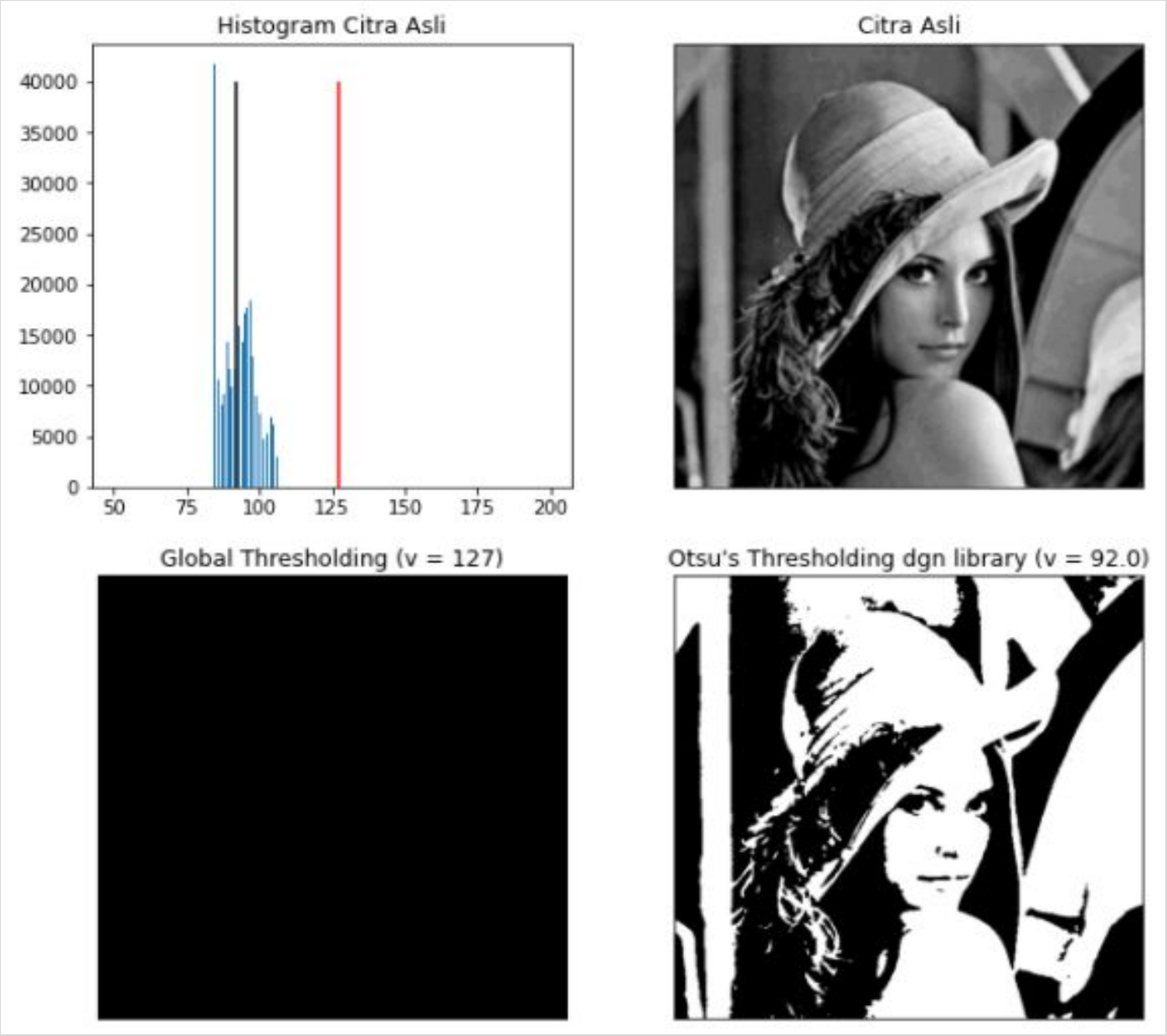
Otsu's Threshold

$$\text{Within Class Variance} \quad \sigma_W^2 = W_b\,\sigma_b^2 + W_f\,\sigma_f^2 \quad \text{(as seen above)}$$

$$\begin{aligned}\text{Between Class Variance} \quad \sigma_B^2 &= \sigma^2 - \sigma_W^2 \\ &= W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2 \quad (\text{where } \mu = W_b\,\mu_b + W_f\,\mu_f) \\ &= W_b\,W_f\,(\mu_b - \mu_f)^2\end{aligned}$$

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

- Nilai threshold dengan Between Class Variance Terbesar adalah yang terbaik



- Garis Vertikal merah adalah nilai threshold global
- Garis Vertikal hitam adalah nilai threshold Otsu's

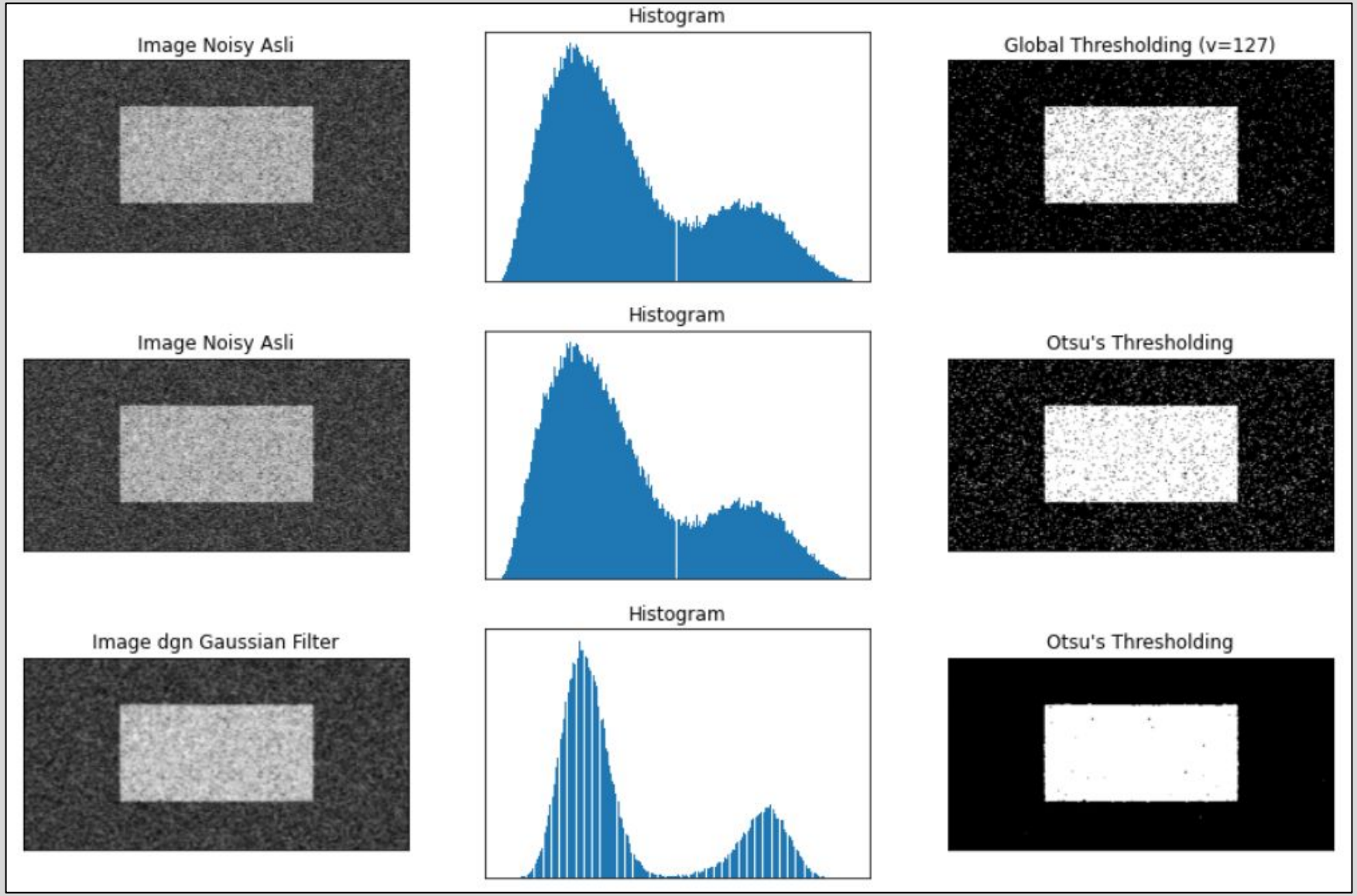
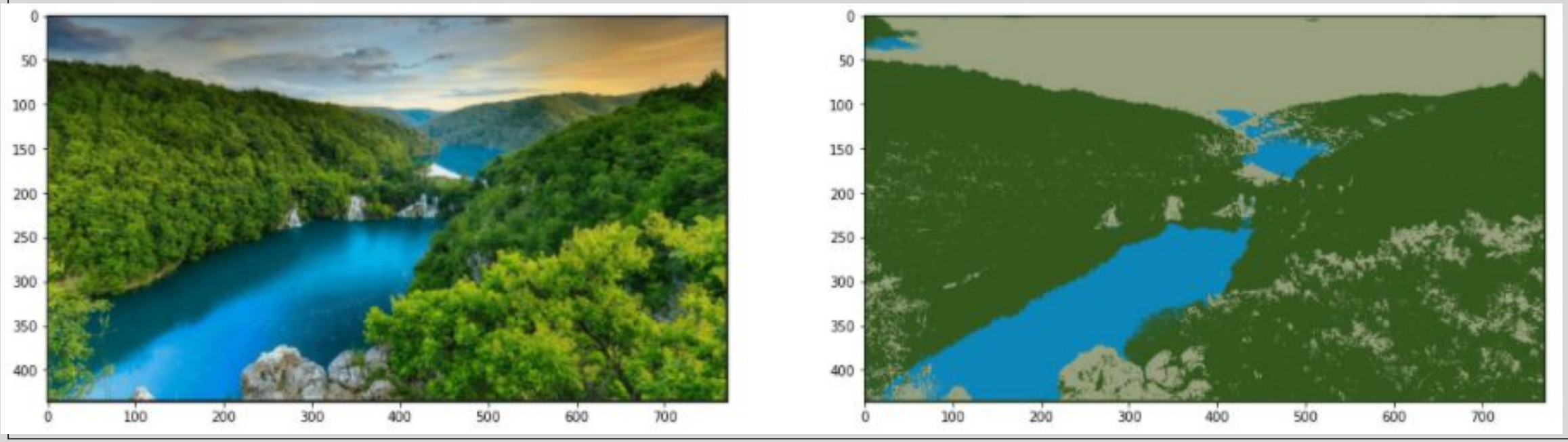


Image Noisy Asli
Histogram
Global Thresholding (v=127)
Image Noisy Asli
Histogram
Otsu's Thresholding
Image dgn Gaussian Filter
Histogram
Otsu's Thresholding

K-Means untuk Segmentasi Citra

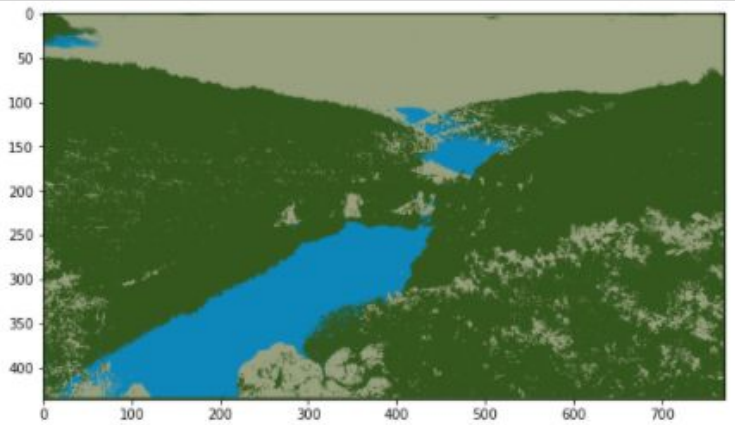
- K-Means merupakan salah satu tool popular yang digunakan untuk melakukan segmentasi citra. Dengan menentukan jumlah segmen sesuai dengan kebutuhan, maka proses segmentasi bisa dilakukan dengan baik. Berikut merupakan hasil dari penggunaan K-Means untuk segmentasi citra.

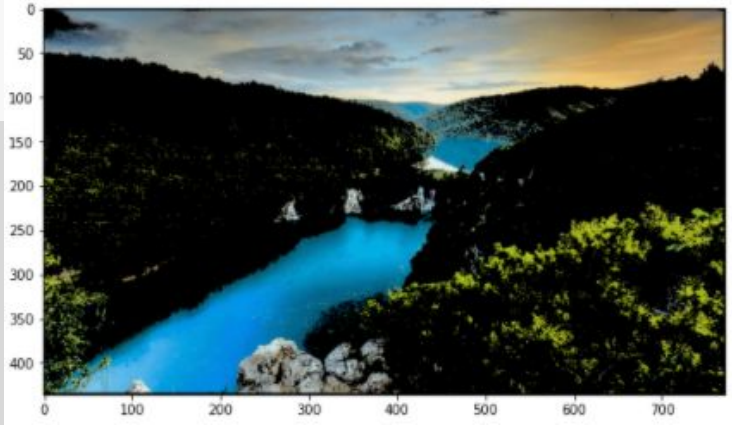


```
filename = ('/content/drive/MyDrive/Polinema/Kuliah/PCVK/Images/jungle.png')

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 iterasi 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.KMEANS_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)
```

Percobaan menghitamkan satu cluster

```
# 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)
```





TERIMA KASIH!