LAPORAN PRAKTIKUM PENGOLAHAN CITRA

FORMAT CITRA, LAYER RGB, GRAY SCALE

PRAKTIKUM 3

Disusun oleh :

Nama : Putri Ayu Nisa Az-Zahra

NRP : 3120600018

Kelas : 2 D4 IT A

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

TAHUN 2021/2022

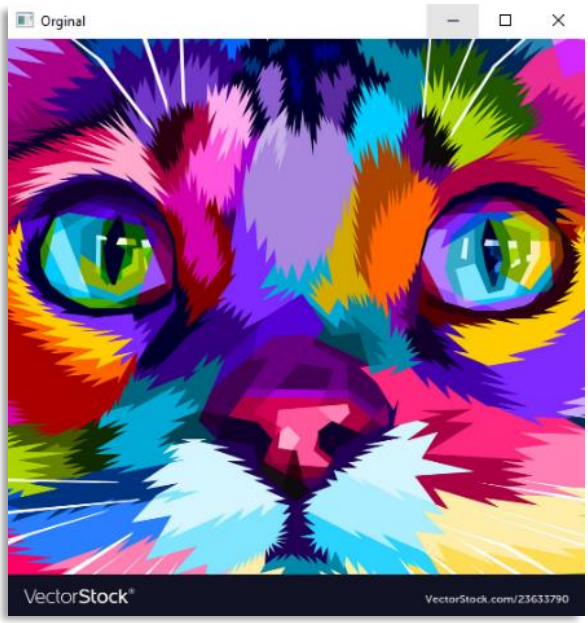
A. Percobaan

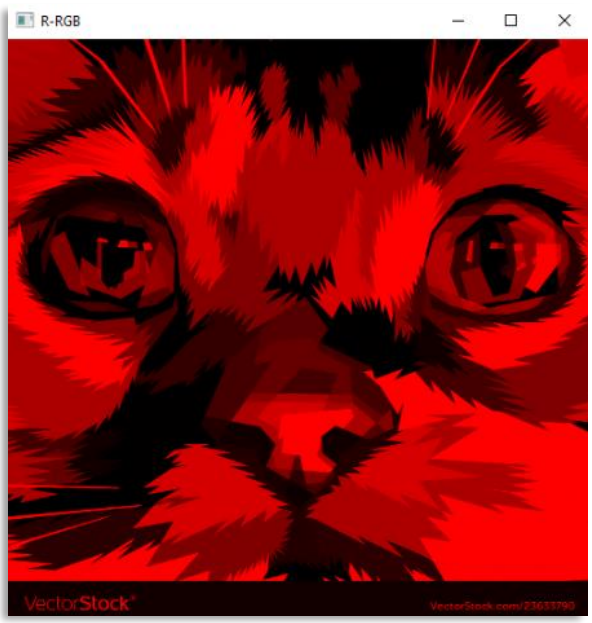
1. Percobaan 1 : Layer B/G/R

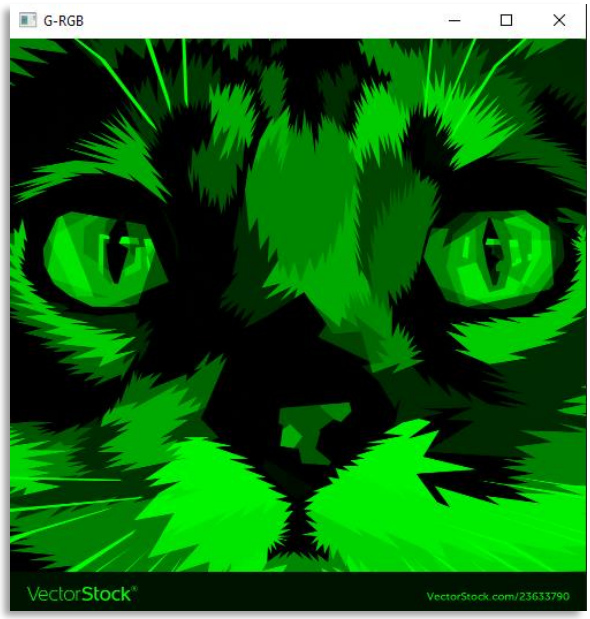
a. Listing

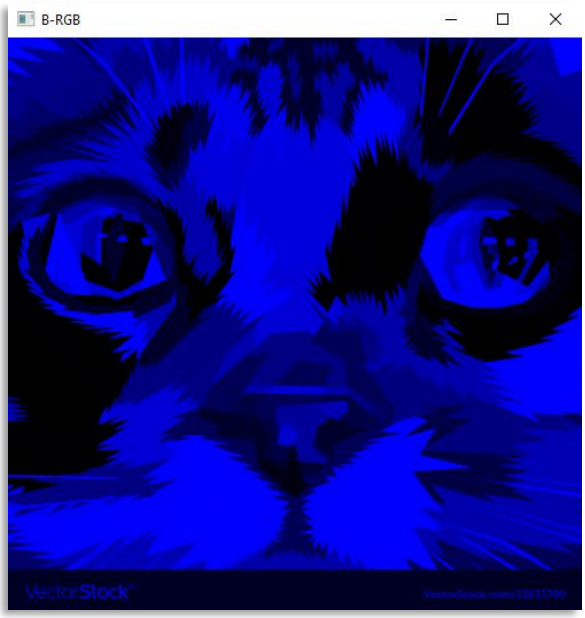
```
import cv2

# membaca data image
img = cv2.imread("cat.jpg")
# resize
img = cv2.resize(img, (500,500))

b = img.copy()
b[:,:,1] = 0
b[:,:,2] = 0

g = img.copy()
g[:,:,0] = 0
g[:,:,2] = 0

r = img.copy()
r[:,:,0] = 0
r[:,:,1] = 0

cv2.imshow("Orginal", img)
cv2.imshow("B-RGB", b)
cv2.imshow("G-RGB", g)
cv2.imshow("R-RGB", r)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

b. Output









c. Analisis

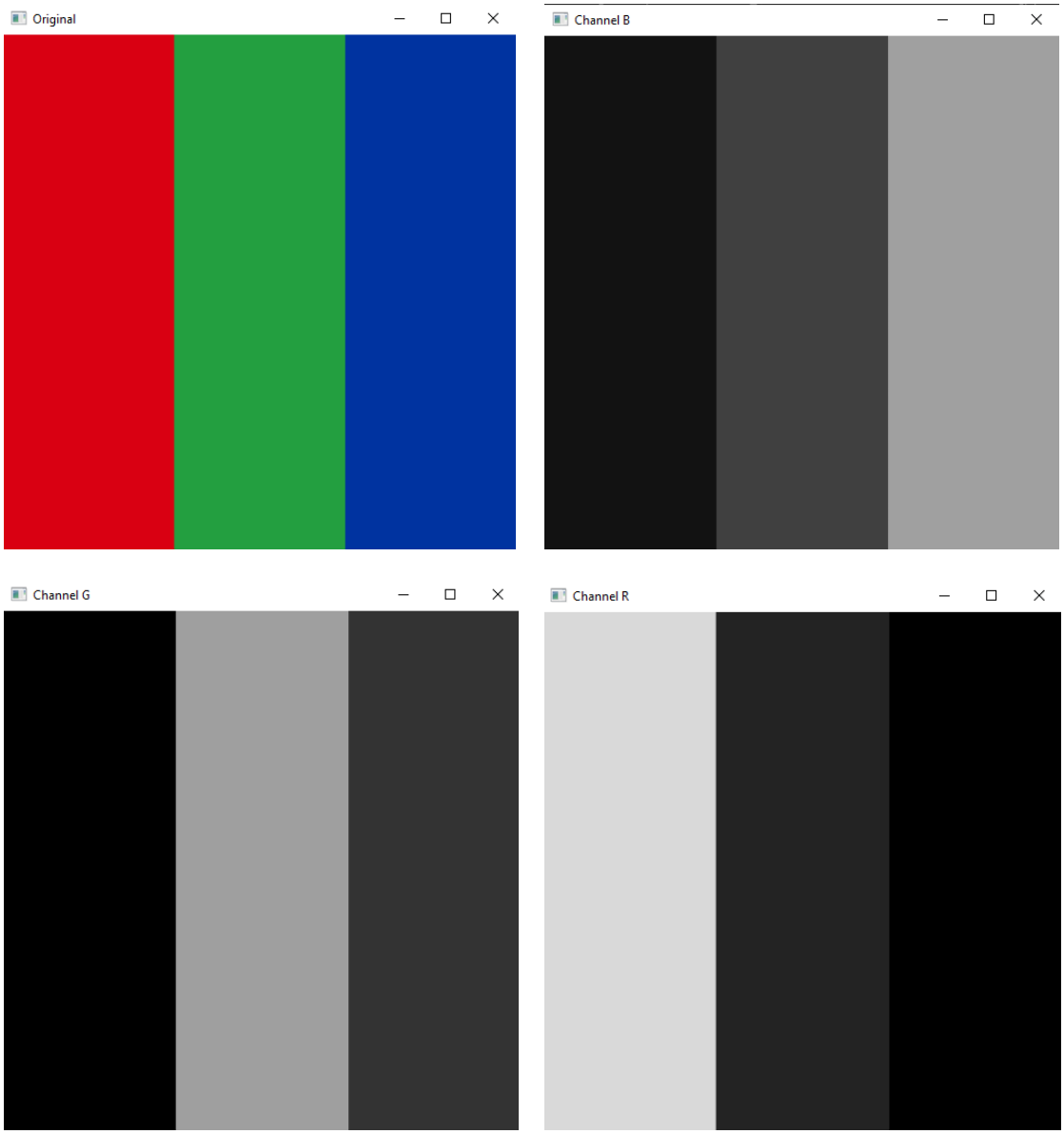
Sebuah gambar BGR terdiri dari 3 layer berwarna yang digabungkan, yaitu layer B (*blue*), G (*green*), dan Red (*red*). Untuk mendapatkan gambar dengan salah satu dari ketiga layer tersebut, caranya adalah dengan menjadikan nilai dari 2 layer warna lainnya menjadi 0 (nol). Contohnya adalah jika ingin menampilkan layer B, maka b[:,:,1] dan b[:,:,2] diinisialisasi dengan nilai 0 yang mana ":" berarti seluruhnya dan "1" mewakili layer G (*green*) lalu "2" mewakili layer R (*red*) sehingga "0" mewakili layer B (*blue*). Pada saat gambar ditampilkan, maka gambar tersebut akan berwarna, biru, hijau, atau merah seperti output di atas.

2. Percobaan 2 : Grayscale dari B/G/R

a. Listing

```
import cv2

# membaca data image
img = cv2.imread("colors.png")
# resize
img = cv2.resize(img, (500,500))

B, G, R = cv2.split(img)
cv2.imshow("Original", img)
cv2.imshow("Channel R", R)
cv2.imshow("Channel G", G)
cv2.imshow("Channel B", B)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

b. Output



c. Analisis

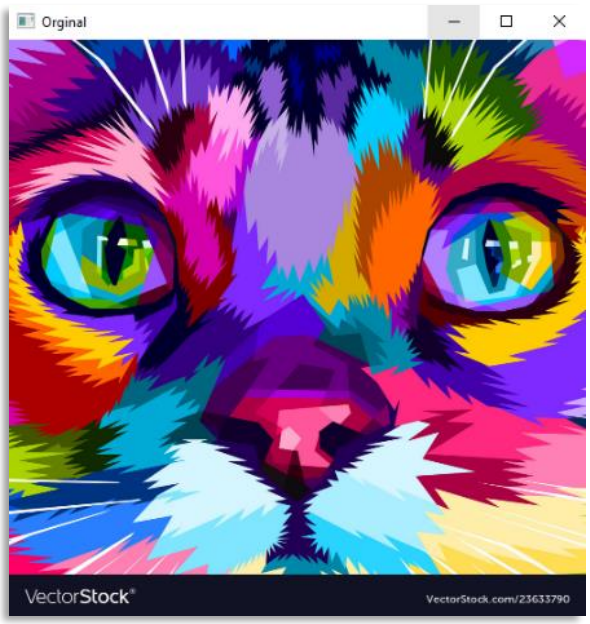
Untuk mendapatkan citra grayscale, nilai B, G, R diisi dengan nilai yang sama misalkan (100,100,100). Pada percobaan kali ini, proses pemisahan nilai untuk membentuk citra grayscale adalah menggunakan fungsi **split()** dari *library* opencv. Maka variabel B akan menyimpan gambar dengan nilai dari layer B (b,b,b), variabel G menyimpan gambar dengan nilai dari layer G (g,g,g), dan variabel R menyimpan gambar dengan nilai dari layer R (r,r,r). Perbedaan dari citra grayscale dengan menggunakan nilai B atau G atau R adalah pada bagian gambar yang berwarna sama dengan layer tersebut akan terlihat lebih terang. Contohnya seperti pada *output* di atas, pada channel B, bagian gambar yang berwarna biru-lah yang terlihat paling terang. Dan begitu juga pada channel yang lain.

3. Percobaan 3 : Grayscale dengan Iluminasi Citra

a. Listing

```python
import cv2
import numpy as np

# membaca data image
img = cv2.imread("cat.jpg")
# resize
img = cv2.resize(img, (500,500))
# split ke abu" masing" channel
B, G, R = cv2.split(img)

img_gray1 = 0.33 * R + 0.33 * G + 0.33 * B
img_gray1 = img_gray1.astype(np.uint8)
img_RG1 = np.minimum(R, G)

img_gray2 = np.minimum(img_RG1, B)
img_RG2 = np.maximum(R, G)

img_gray3 = np.maximum(img_RG2, B)

cv2.imshow("Original", img)
cv2.imshow("Iluminasi Rata-rata", img_gray1)
cv2.imshow("Ilumniasi minimum", img_gray2)
cv2.imshow("Iluminasi maksimum", img_gray3)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

b. Output



c. Analisis

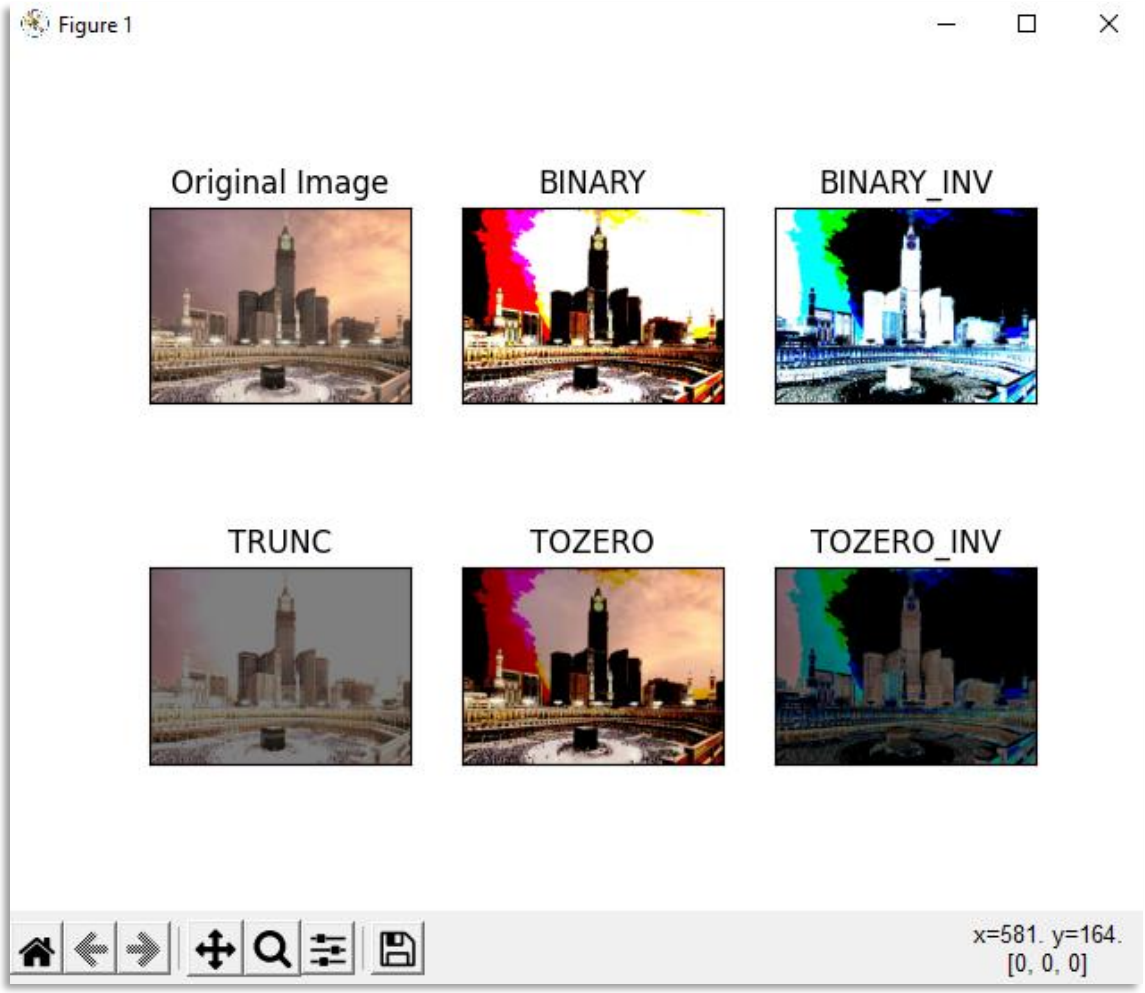
Selain menggunakan nilai dari salah satu channel B, G, R, ada beberapa cara mendapatkan nilai yang dapat digunakan untuk membentuk citra grayscale, yaitu dengan nilai rata-rata dari r, g, b yang akan membentuk warna keabuan yang merata intensitas cahayanya (terangnya merata pada gambar). Kemudian, bisa juga dengan menggunakan nilai minimum dari r, g, b yang akan membentuk citra grayscale dengan intensitas cahaya rendah (gambar tampak gelap). Lalu, dengan menggunakan nilai maksimum dari r, g, b yang akan membentuk citra grayscale dengan intensitas cahaya tinggi (gambar tambak sangat terang).

4. Percobaan 4 : Citra Binary

a. Listing

```
import cv2
import matplotlib.pyplot as plt
from urllib3.connectionpool import xrange

# membaca data image
img = cv2.imread("kabah.jpg")
img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

ret, thresh1 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 127, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO_INV)

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

for i in xrange(6) :
    plt.subplot(2, 3, i+1), plt.imshow(images[i], "gray")
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

b. Output



c. Analisis

Dalam *thresholding*, kita mengonversi gambar berwarna atau grayscale menjadi gambar binar/biner. Untuk setiap pixel, threshold yang sama diaplikasikan. Jika nilai pixel lebih kecil daripada *threshold*, maka nilai pixel tersebut akan diset 0. Jika tidak, maka akan diset dengan nilai maksimum. Ada beberapa metode yang dapat digunakan, yaitu THRESH_BINARY, THRESH_BINARY_INV, THRESH_TRUNC, THRESH_TOZERO, THRESH_TOZERO_INV.

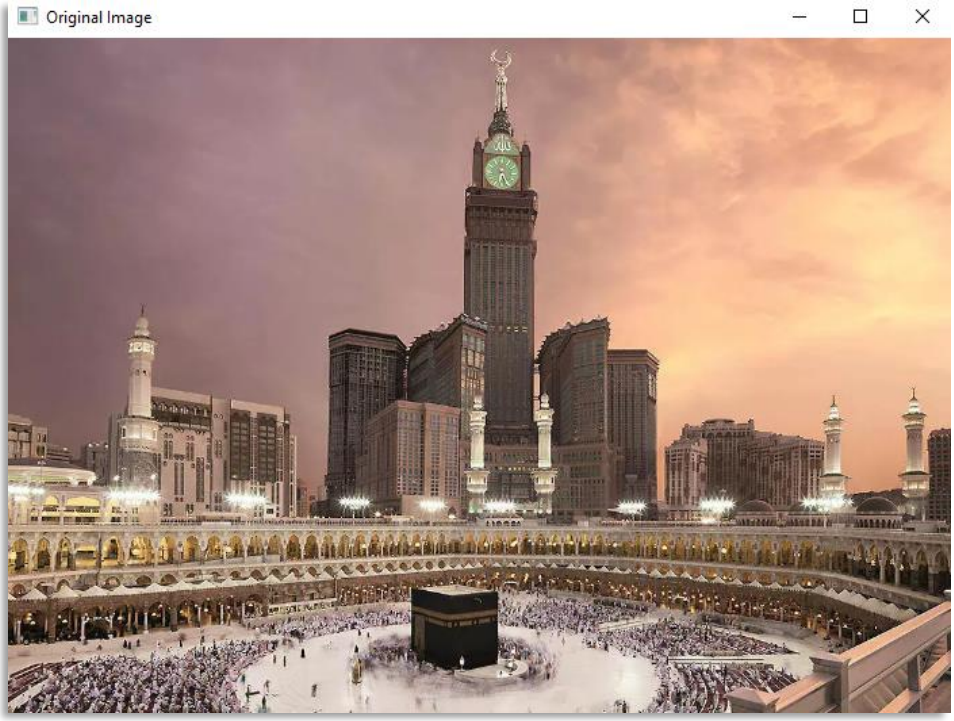
B. Tugas

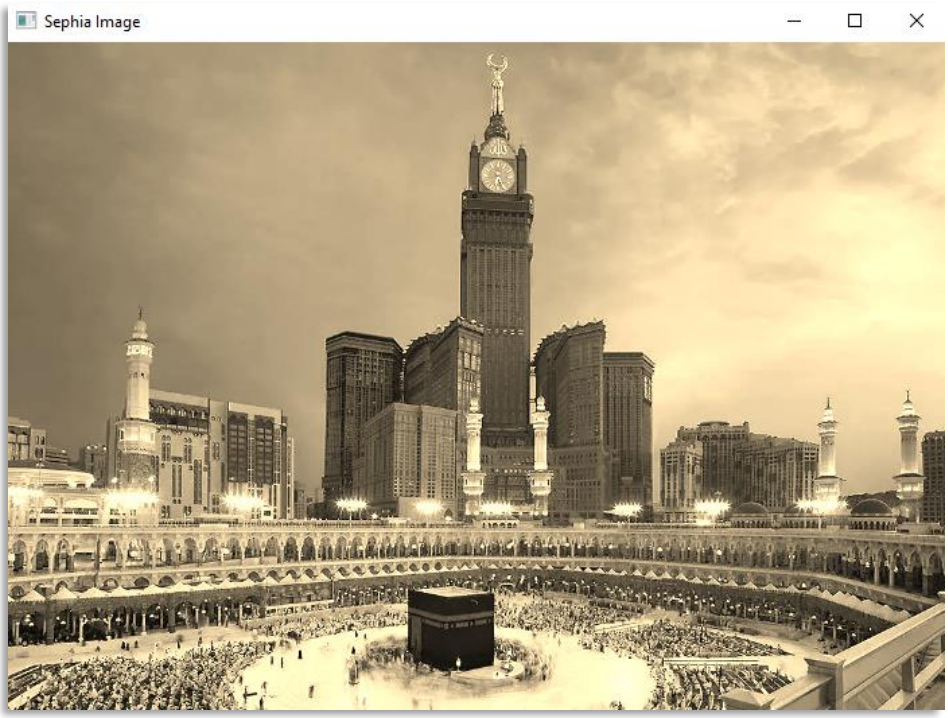
1. Tugas 1 : Sepia Effect

a. Listing

```
import cv2
import numpy as np

# membaca data image
img = cv2.imread("kabah.jpg")
# resize
img = cv2.resize(img, (700, 500))
# copy
sepia_img = img.copy()

sepia_img = np.array(img, dtype=np.float64) # converting to float to prevent
loss
sepia_img = cv2.transform(img, np.matrix([[0.272, 0.534, 0.131],
                                          [0.349, 0.686, 0.168],
                                          [0.393, 0.769, 0.189]])) # multipying
image with special sepia matrix]))

sepia_img[np.where(sepia_img > 255)] = 255 #normalizing values greater than
255 to 255
sepia_img = np.array(sepia_img, np.uint8) # converting back to int

cv2.imshow("Original Image", img)
cv2.imshow("Sephia Image", sepia_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

b. Output





c. Analisis

Efek sepia akan menjadikan gambar tampak berwarna kuning atau kecoklatan. Untuk membuat efek sepia pada gambar, terdapat sebuah **matriks khusus** yang dikalikan dengan setiap elemen pada array gambar. Matriks tersebut adalah $\begin{bmatrix} 0{,}272 & 0{,}534 & 0{,}131 \\ 0{,}349 & 0{,}686 & 0{,}168 \\ 0{,}393 & 0{,}769 & 0{,}189 \end{bmatrix}$. Dengan menggunakan numpy, didapatkan array dengan tipe data *float* yang hasil akhirnya akan diubah ke dalam *integer* agar lebih akurat dalam perhitungan. Kemudian, setiap elemen array gambar dikalikan dengan matriks khusus tersebut dengan menggunakan **cv2.transform()** dengan bantuan **np.matrix()**. Setelah itu, dilakukan normalisasi agar nilai tidak lebih dari 255. Maka, setelah hasil diubah ke dalam *integer*, gambar akan tampak seperti *output* di atas.

2. Tugas 2 : Layer BGR, grayscale, dan binar pada gambar yang ada di soal

a. Listing

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from urllib3.connectionpool import xrange

# membaca data image
img = cv2.imread("penguins.jpg") #source diganti-ganti sesuai gambar
img = cv2.resize(img, (700, 500))
img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

# Layer BGR
b = img.copy()
b[:,:,1] = 0
b[:,:,2] = 0
```

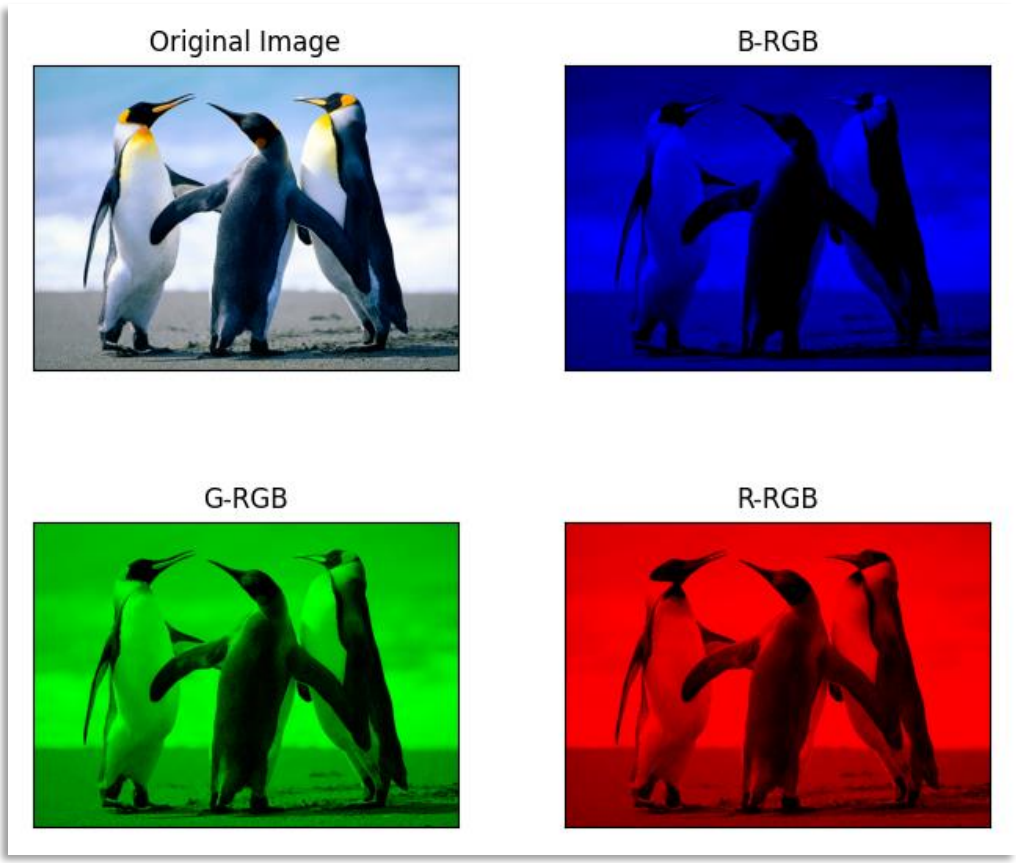
```
b = cv2.cvtColor(b, cv2.COLOR_RGB2BGR)
g = img.copy()
g[:,:,0] = 0
g[:,:,2] = 0
g = cv2.cvtColor(g, cv2.COLOR_RGB2BGR)
r = img.copy()
r[:,:,0] = 0
r[:,:,1] = 0
r = cv2.cvtColor(r, cv2.COLOR_RGB2BGR)

titles = ["Original Image", "B-RGB", "G-RGB", "R-RGB"]
images = [img, b, g, r]

for i in xrange(4) :
    plt.subplot(2, 2, i+1), plt.imshow(images[i], "gray")
    plt.subplots_adjust(hspace=0.5)
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()

# citra grayscale
B, G, R = cv2.split(img)
B = cv2.cvtColor(B, cv2.COLOR_RGB2BGR)
G = cv2.cvtColor(G, cv2.COLOR_RGB2BGR)
R = cv2.cvtColor(R, cv2.COLOR_RGB2BGR)

titles = ["Original Image", "Channel B", "Channel G", "Channel R"]
images = [img, B, G, R]

for i in xrange(4) :
    plt.subplot(2, 2, i+1), plt.imshow(images[i], "gray")
    plt.subplots_adjust(hspace=0.5)
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()

# citra grayscale dengan iluminasi
img_gray1 = 0.33 * R + 0.33 * G + 0.33 * B
img_gray1 = img_gray1.astype(np.uint8)
img_gray1 = cv2.cvtColor(img_gray1, cv2.COLOR_RGB2BGR)
rg = np.minimum(R, G)
img_gray2 = np.minimum(B, rg)
img_gray2 = cv2.cvtColor(img_gray2, cv2.COLOR_RGB2BGR)
rg = np.maximum(R, G)
img_gray3 = np.maximum(rg, B)
img_gray3 = cv2.cvtColor(img_gray3, cv2.COLOR_RGB2BGR)

titles = ["Original Image", "Iluminasi rata-rata", "Iluminasi minimum",
"Iluminasi maksimum"]
images = [img, img_gray1, img_gray2, img_gray3]
```

```
for i in xrange(4) :
    plt.subplot(2, 2, i+1), plt.imshow(images[i], "gray")
    plt.subplots_adjust(hspace=0.5)
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()

# binar
ret, thresh1 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 127, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO_INV)

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

for i in xrange(6) :
    plt.subplot(2, 3, i+1), plt.imshow(images[i], "gray")
    plt.subplots_adjust(hspace=0.5)
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

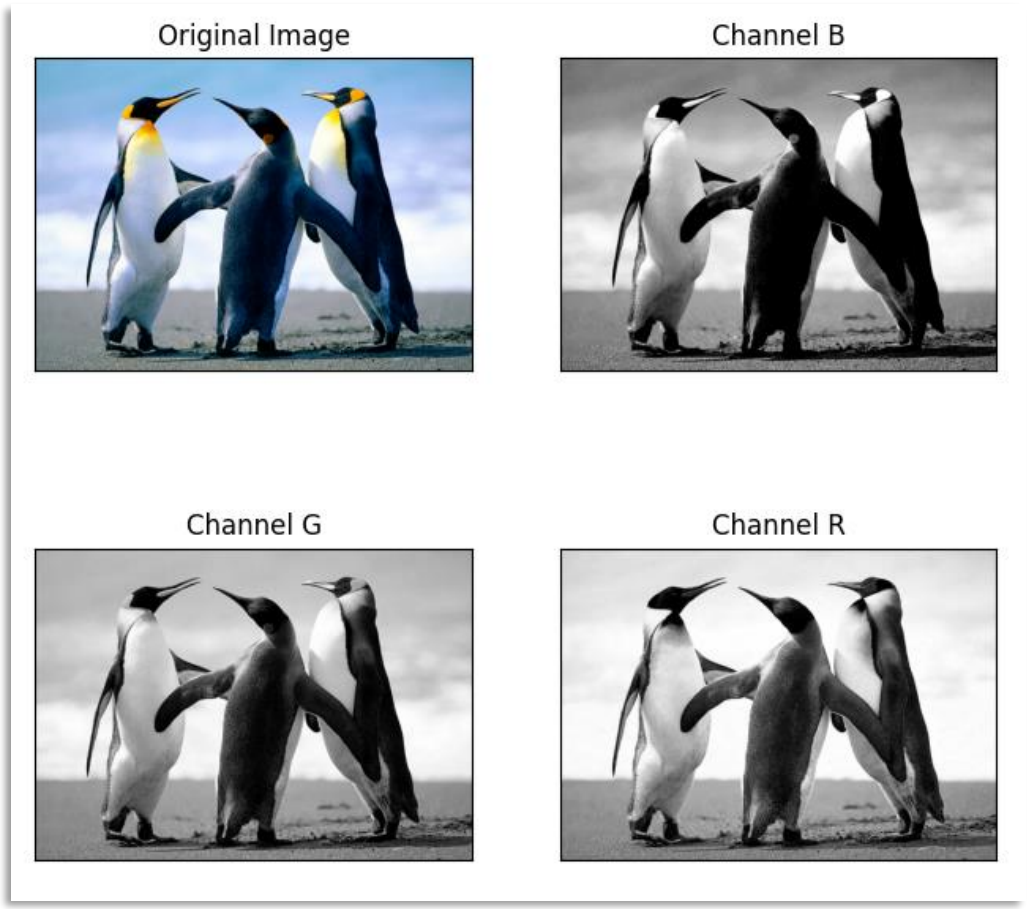
b. Output

Penguins

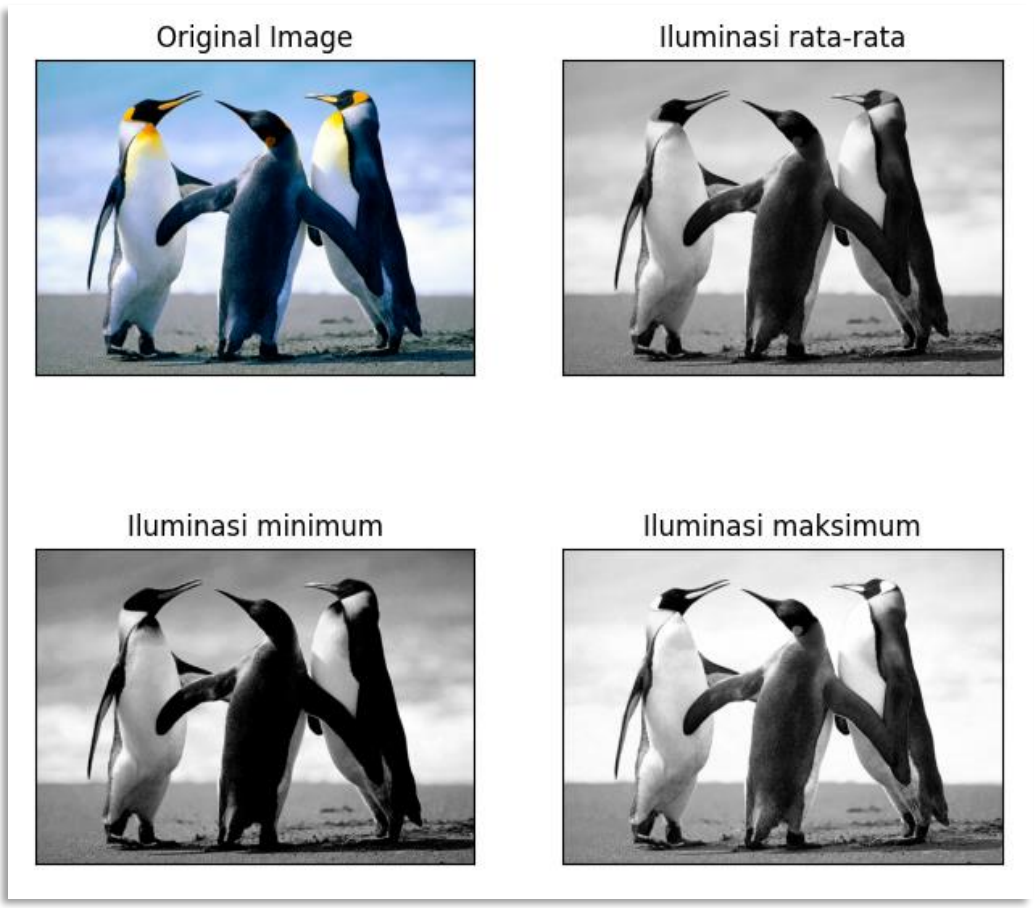
Layer B,G,R



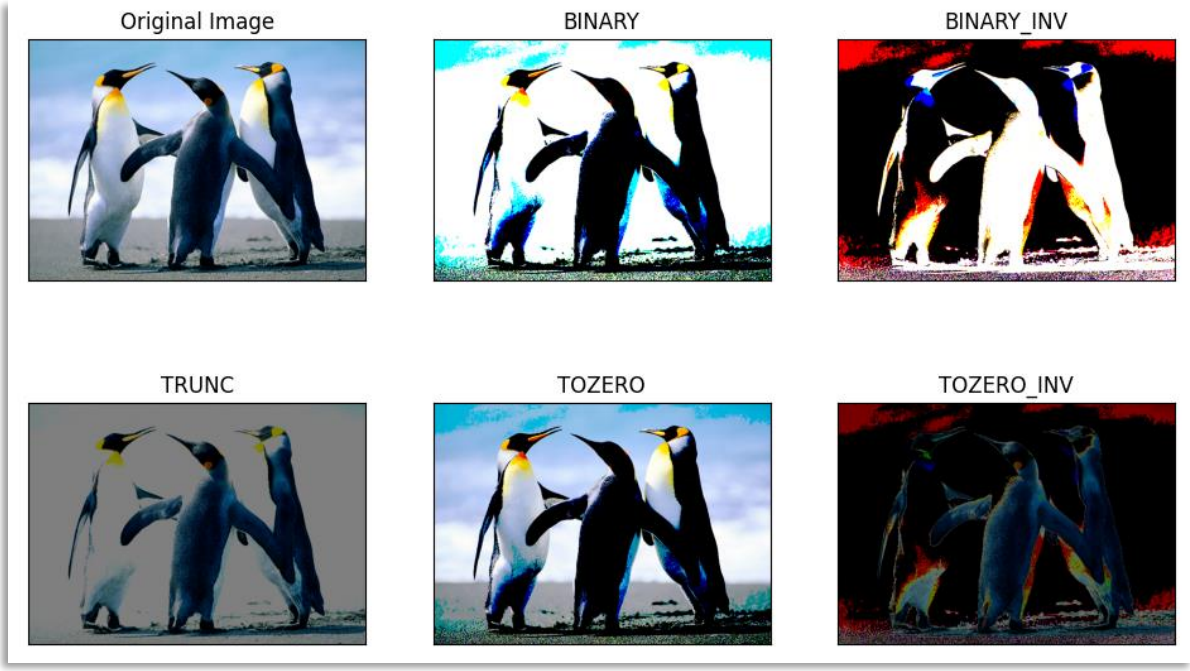
Citra Grayscale



Citra Grayscale Iluminasi

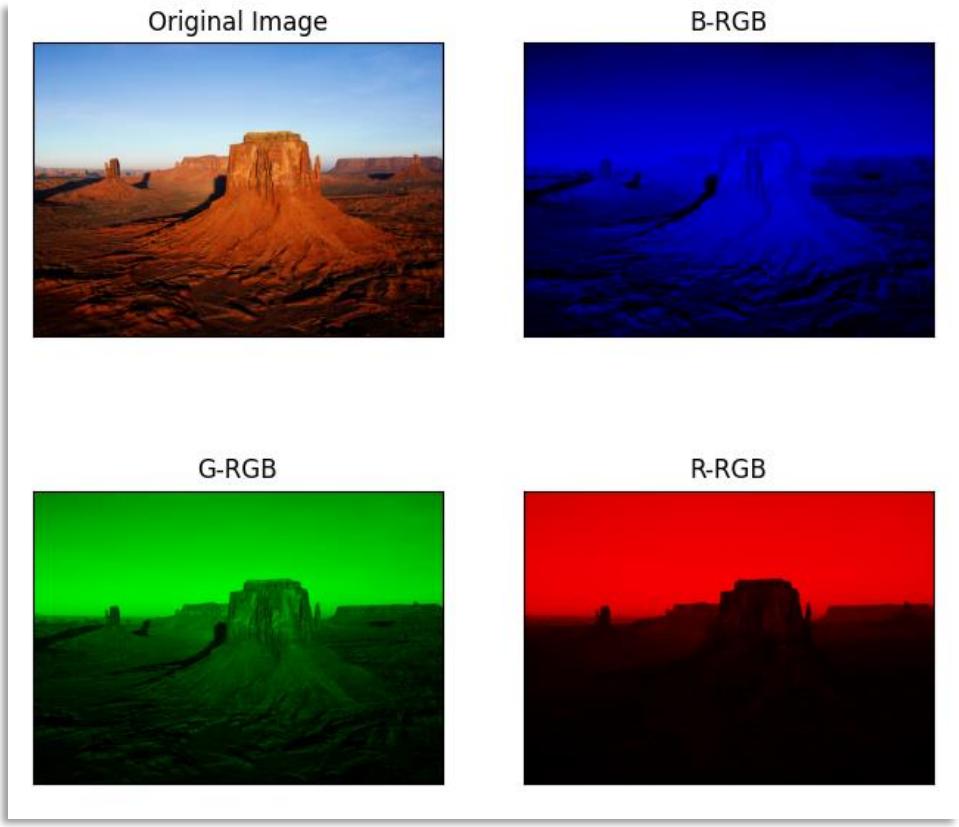


Binar

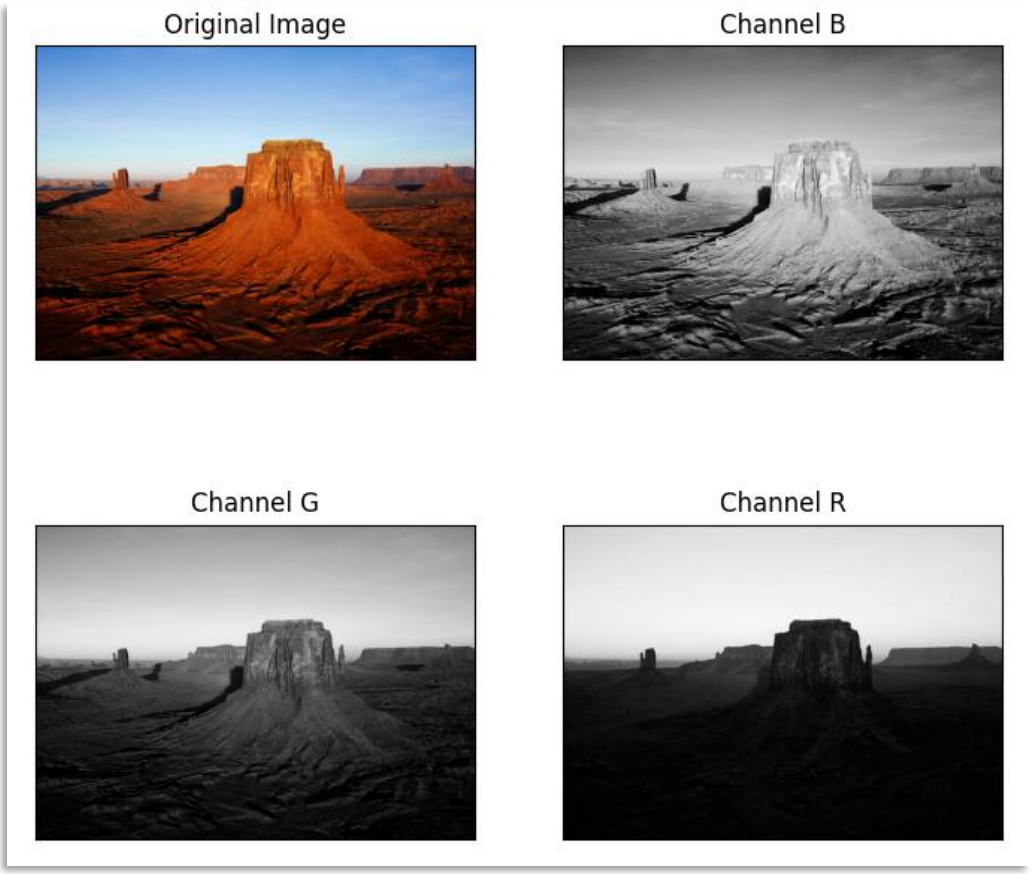


Desert

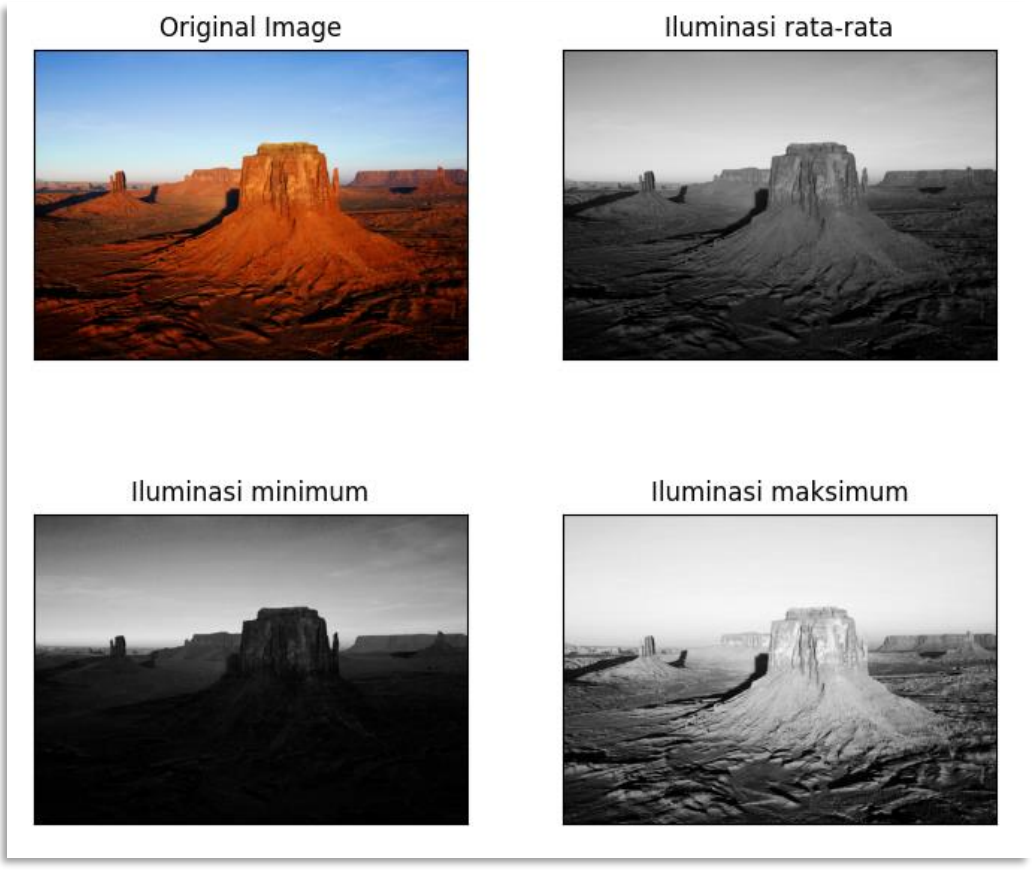
Layer B,G,R



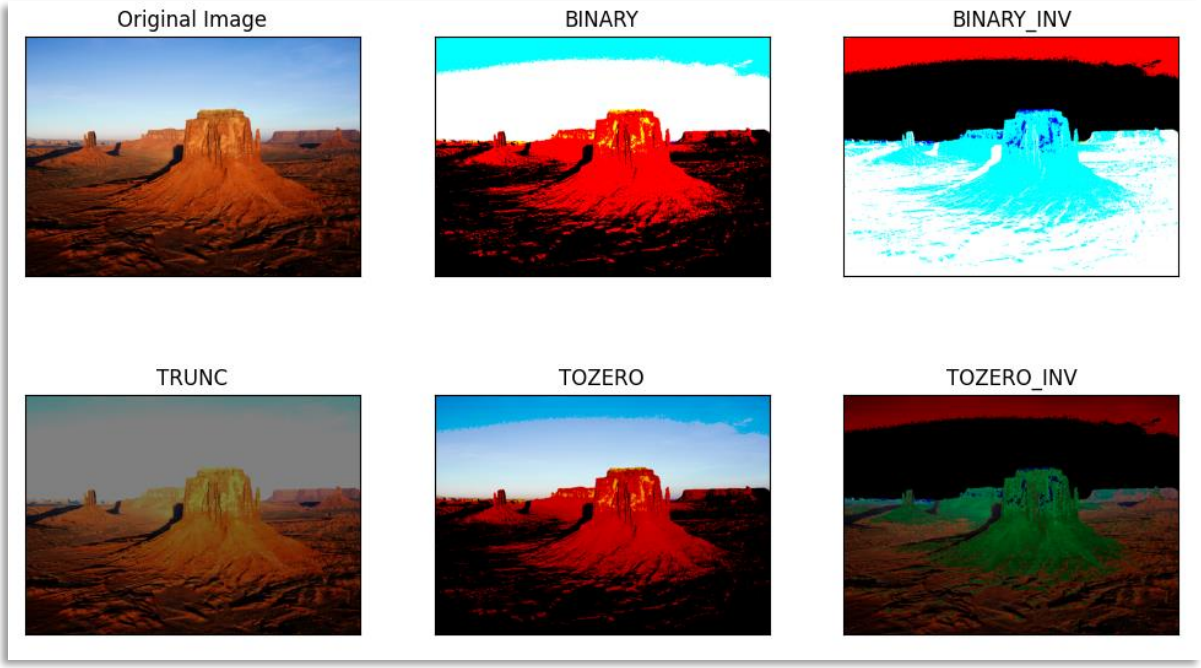
Citra Grayscale



Citra Grayscale Iluminasi

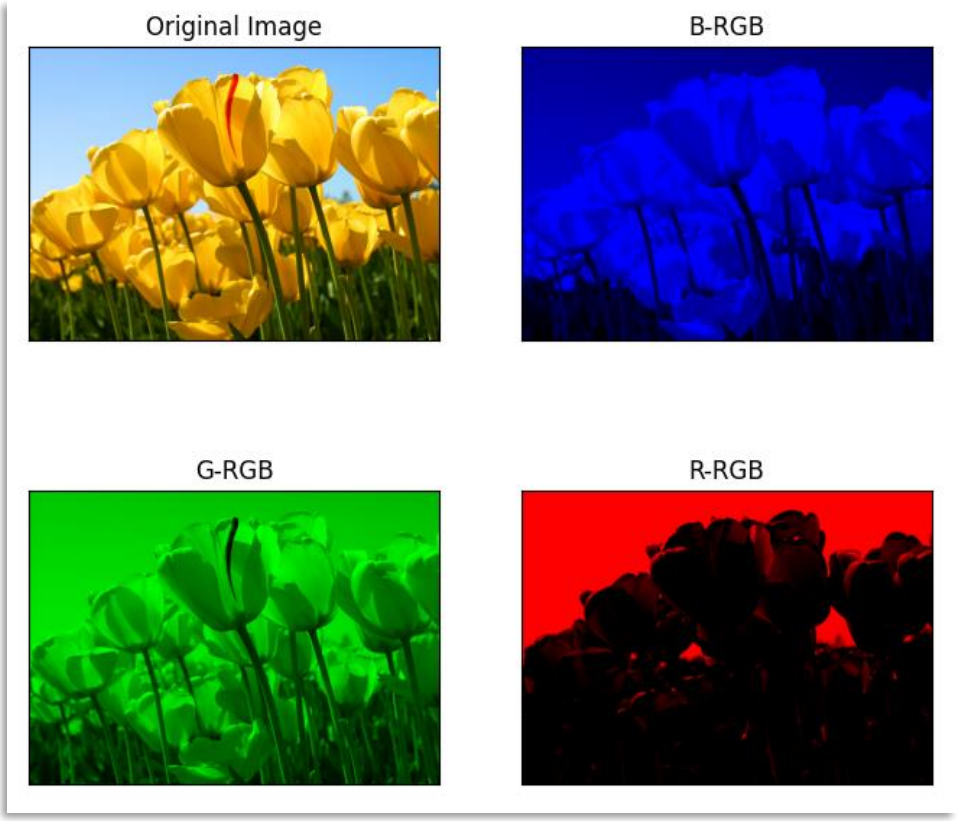


Binar

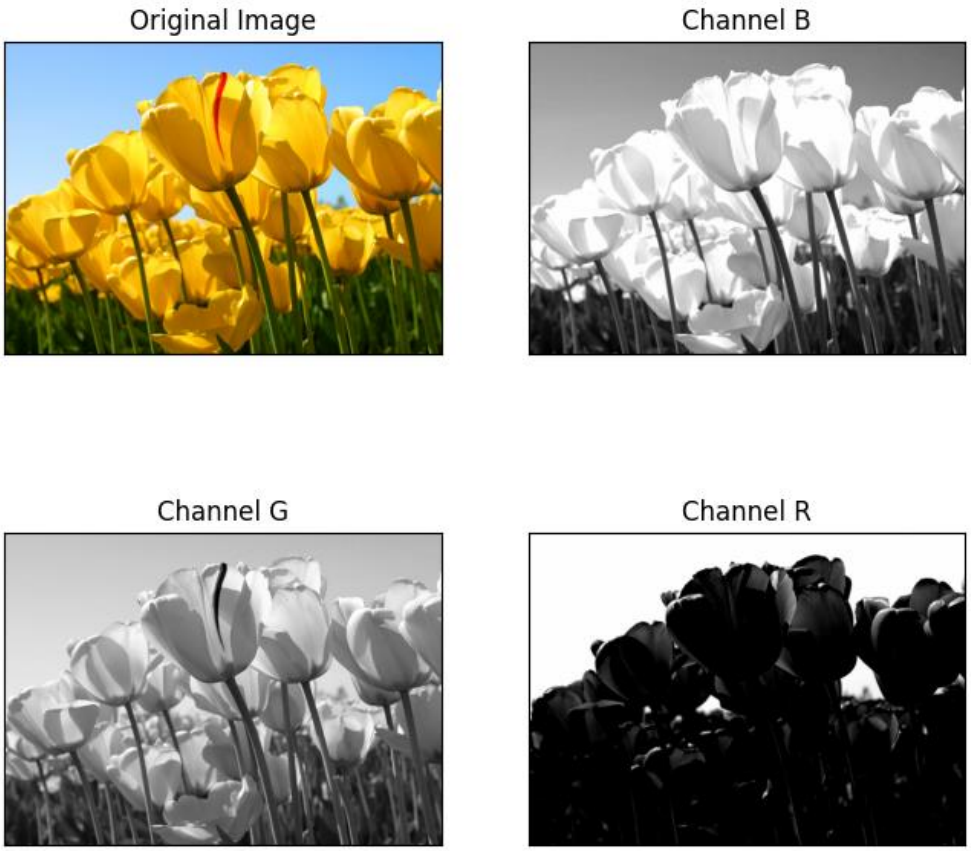


Tulips

Layer B,G,R



Citra Grayscale

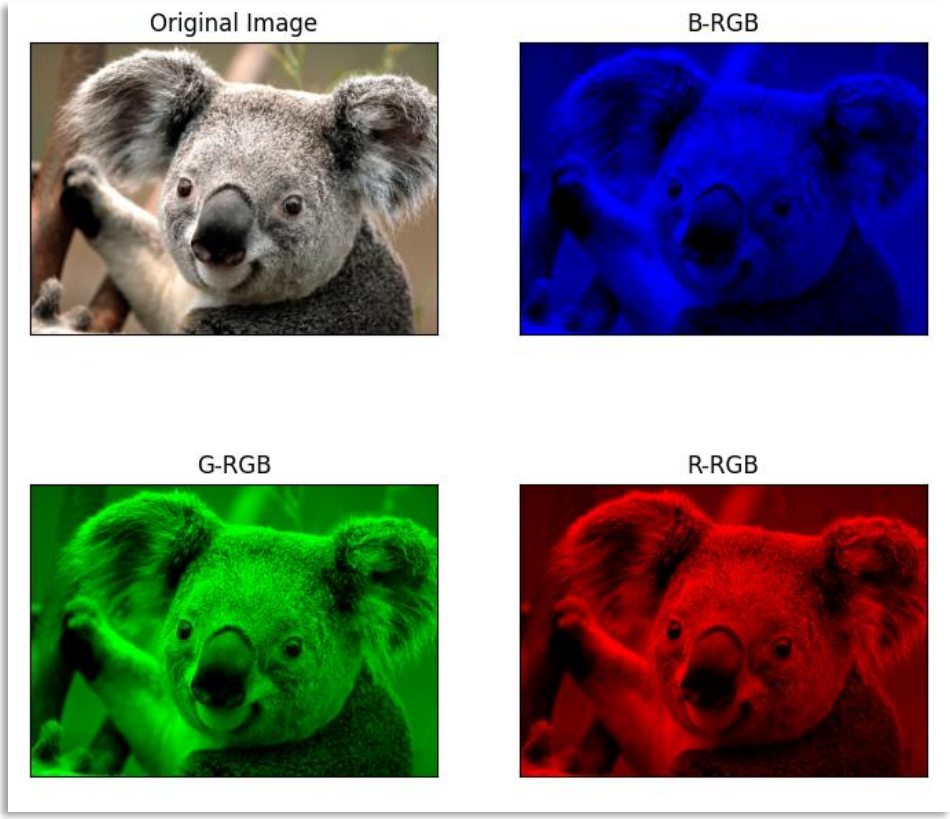


Citra Grayscale Iluminasi

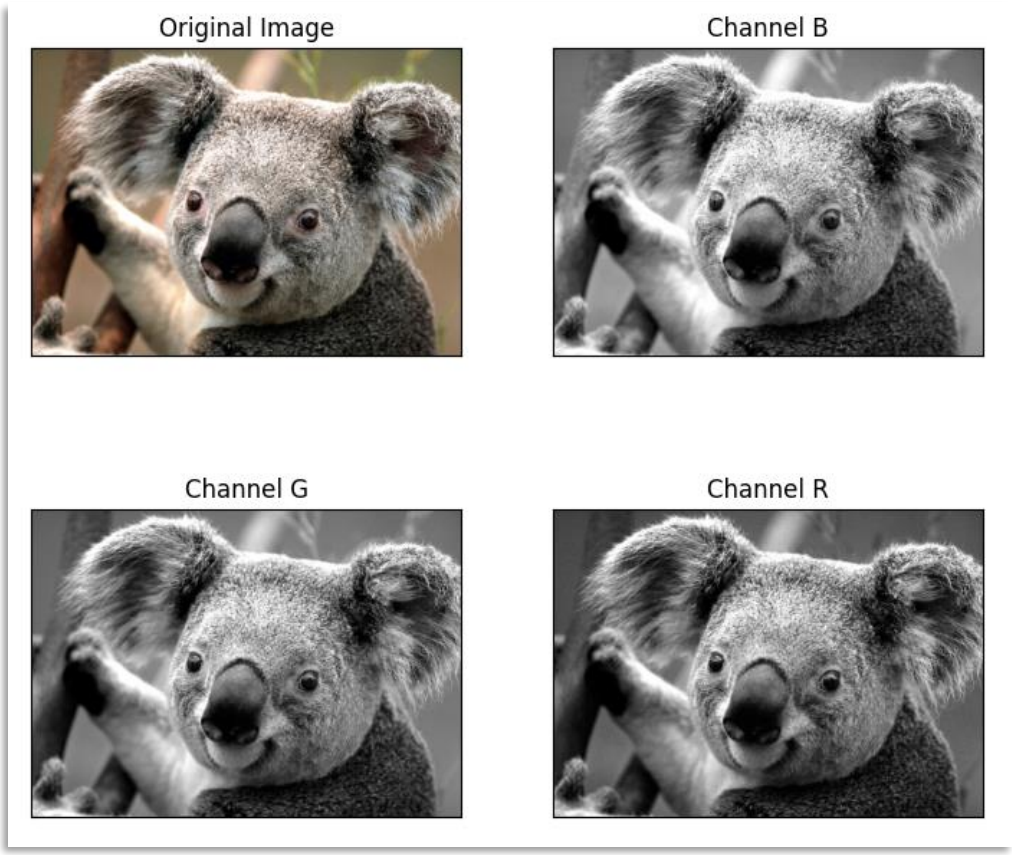
Binar

Koala

Layer B,G,R

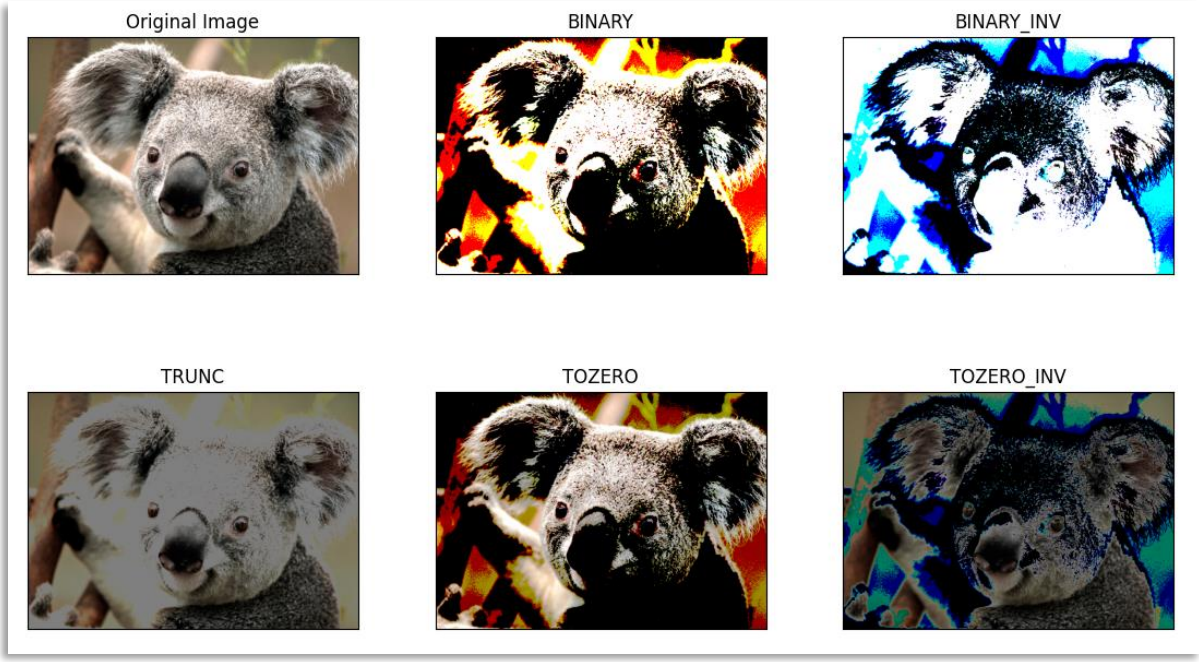


Citra Grayscale



Citra Grayscale Iluminasi

Binar



c. Analisis

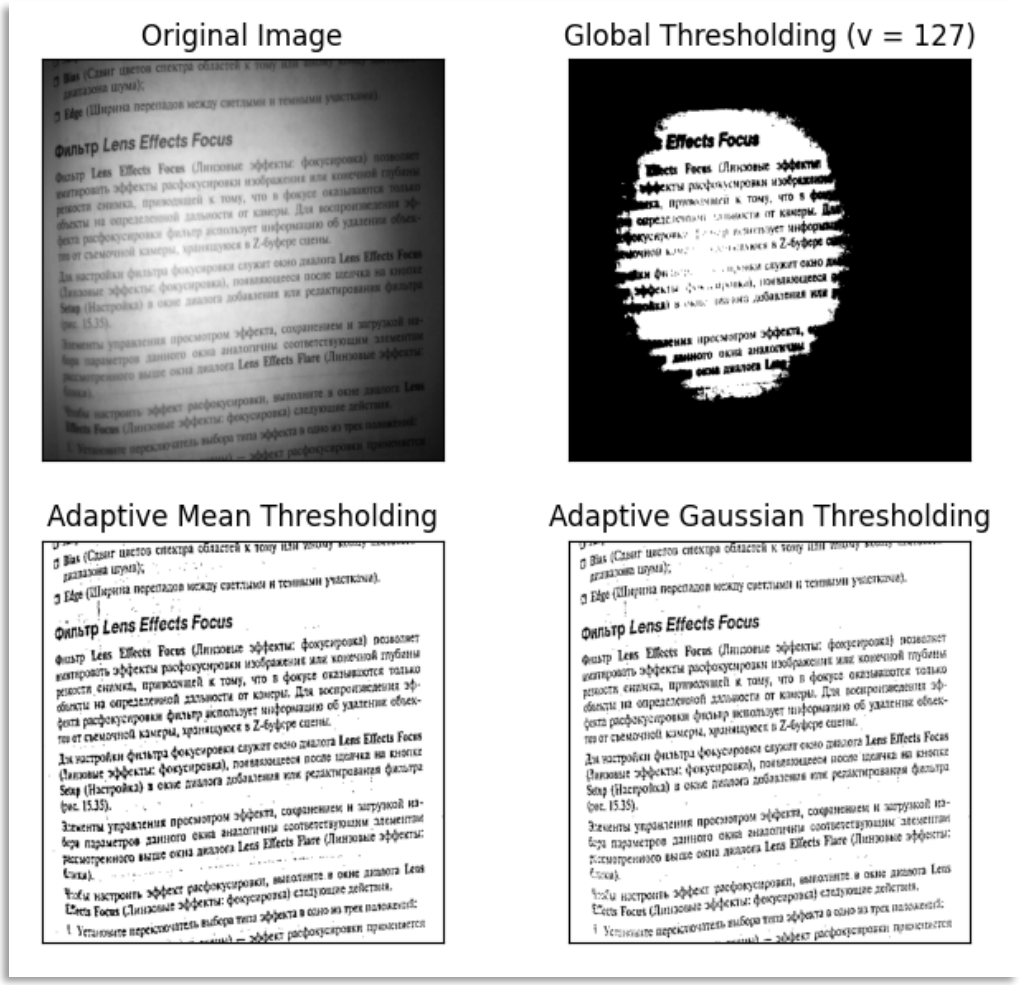
Penjelasan sudah tertera pada percobaan-percobaan sebelumnya, hanya saja kali ini memakai 4 gambar yang mana masing-masing gambar dimanipulasi layer BGR, grayscale, grayscale iluminasi, dan binar.

3. Tugas 3 : Adaptive Threshold

a. Listing

```
import cv2
import matplotlib.pyplot as plt

# membaca data image
img = cv2.imread("dark.jpg")
# resize
img = cv2.resize(img, (500,500))
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

ret, thres0 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
thres1 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY, 9, 5)
thres2 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 9, 5)

titles = ["Original Image", "Global Thresholding (v = 127)", "Adaptive Mean
Thresholding", "Adaptive Gaussian Thresholding"]
images = [img, thres0, thres1, thres2]

for i in range(4) :
    plt.subplot(2, 2, i+1), plt.imshow(images[i], "gray")
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

b. Output



c. Analisis

Menggunakan threshold global tidak bisa menghasilkan *output* yang baik jika gambar memiliki pencahayaan yang kurang atau tidak merata. Dengan demikian, untuk mengatasinya dapat menggunakan **adaptive threshold**. algoritma dari fungsi ini menentukan *threshold* untuk sebuah pixel berdasarkan sebuah daerah kecil di sekitar pixel tersebut. Jadi akan didapatkan *threshold* yang berbeda untuk daerah yang berbeda dari sebuah gambar/citra yang sama yang akan memberikan hasil yang lebih baik dengan iluminasi yang bervariasi. Dalam opencv, dapat menggunakan fungsi **adaptiveThreshold()**. Ada dua metode yang dapat digunakan, yaitu ADAPTIVE_THRESH_MEAN_C dimana nilai *threshold* adalah rata-rata atau mean dari daerah tetangga dikurangi konstanta C dan ADAPTIVE_THRESH_GAUSSIAN_C dimana nilai *threshold* adalah hasil penambahan gaussian dari daerah tetangga dikurangi konstanta C.

C. Pertanyaan Tambahan

1. Jelaskan fungsi dan parameter di CV2 yang berfungsi untuk thresholding binary dan adaptive thresholding.

   Jawab :

   Fungsi yang digunakan pada CV2 untuk melakukan *thresholding* adalah **cv2.threshold()** dan **cv2.adaptiveThreshold()** dimana parameter yang ada adalah :

   a. cv2.threshold(src, thresh, maxval, type[, dst])

      → untuk setiap pixel, threshold yang sama diaplikasikan. Jika nilai pixel lebih kecil daripada *threshold*, maka nilai pixel tersebut akan diset 0.

- **src** : *source image*, harus citra grayscale
- **thresh** : nilai *threshold* yang digunakan untuk mengklasifikasi nilai pixel
- **maxval** : nilai maksimum yang diassign ke nilai pixel melebihi nilai *threshold*
- **type** : tipe *thresholding* yang digunakan

b. cv.AdaptiveThreshold(src, maxValue, adaptive_method, thresholdType, blockSize, C)
→ algoritma dari fungsi ini menentukan *threshold* untuk sebuah pixel berdasarkan sebuah daerah kecil di sekitar pixel tersebut. Jadi akan didapatkan *threshold* yang berbeda untuk daerah yang berbeda dari sebuah gambar/citra yang sama yang akan memberikan hasil yang lebih baik dengan iluminasi yang bervariasi.

- **src** : *source image*, harus citra grayscale
- **maxValue** : nilai maksimum yang diassign ke nilai pixel
- **adaptive_method** : menentukan bagaimana nilai *threshold* dihitung. Ada dua metode, yaitu ADAPTIVE_THRESH_MEAN_C dimana nilai *threshold* adalah rata-rata atau mean dari daerah tetangga dikurangi konstanta C dan ADAPTIVE_THRESH_GAUSSIAN_C dimana nilai *threshold* adalah hasil penambahan gaussian dari daerah tetangga dikurangi konstanta C.
- **thresholdType** : tipe *thresholding* yang digunakan
- **blockSize** : menentukan ukuran dari daerah tetangga
- **C** : sebuah konstanta yang akan dikurangi oleh rata-rata (mean) atau penambahan gaussian dari pixel tetangga.

D. Kesimpulan

Setelah percobaan dan tugas yang dilakukan, dapat disimpulkan bahwa sebuah citra dapat dimanipulasi dengan berbagai efek seperti grayscale, biner, dan sepia. Selain itu, sebuah gambar BGR terdiri dari 3 layer, yaitu layer B atau *blue*, layer G atau *green*, dan layer R atau *red*. Untuk melakukan manipulasi tersebut, Python telah menyediakan *library* yang sangat membantu sehingga prosesnya tidak perlu dilakukan secara manual.