Lab 2
Pengolahan Citra
Color Image Processing
Senin, 5 Oktober 2020

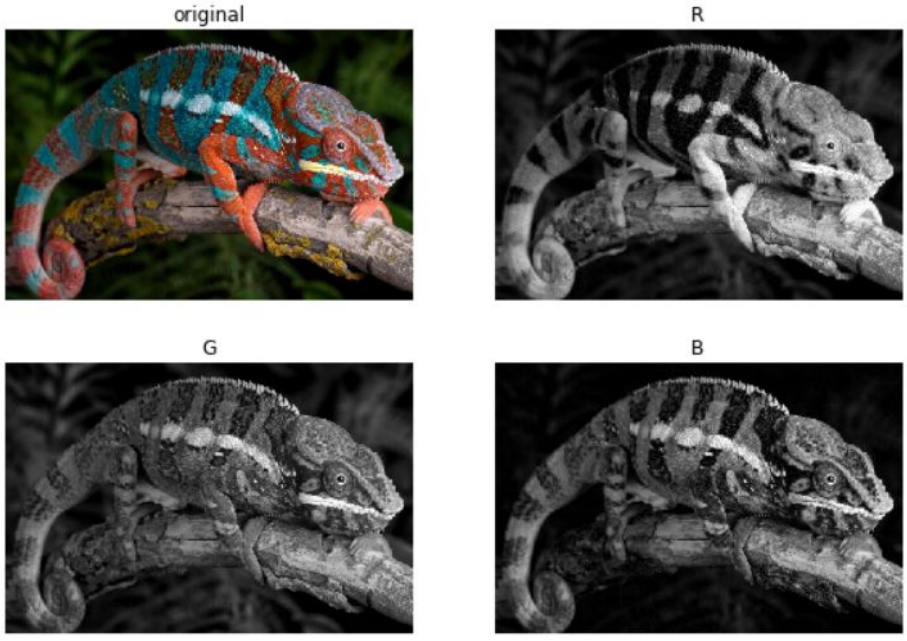
---

1. Color Transformation

Color Transformation merupakan proses pemrosesan citra yang menggunakan transformasi warna.

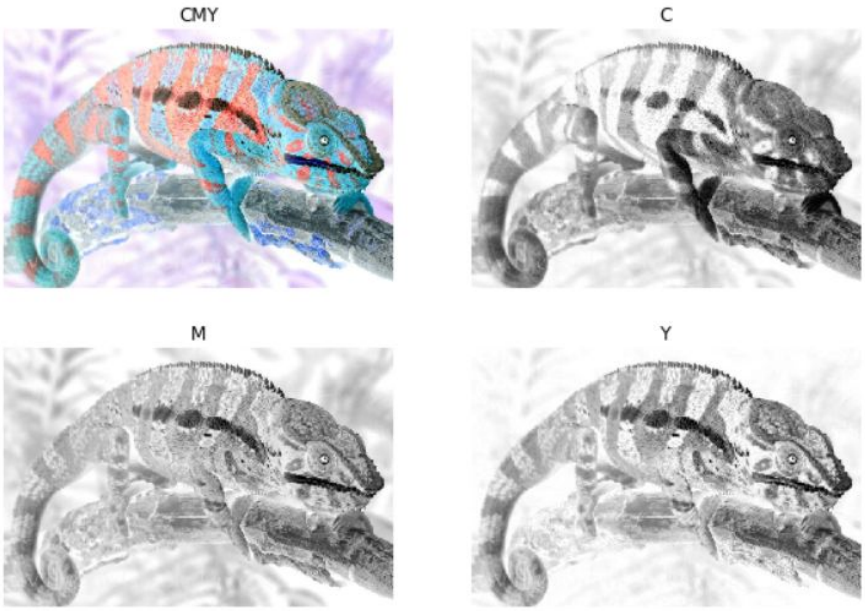
A. RGB

```
image = io.imread('chameleon.jpg')
R = image[:,:,0]
G = image[:,:,1]
B = image[:,:,2]
plt.subplot(2,2,1);plt.imshow(image); plt.title('original');
plt.axis('off')
plt.subplot(2,2,2);plt.imshow(R,cmap='gray',vmin=0,vmax=255);
plt.title('R'); plt.axis('off')
plt.subplot(2,2,3);plt.imshow(G,cmap='gray',vmin=0,vmax=255);
plt.title('G'); plt.axis('off')
plt.subplot(2,2,4);plt.imshow(B,cmap='gray',vmin=0,vmax=255);
plt.title('B'); plt.axis('off')
```



B. CMY

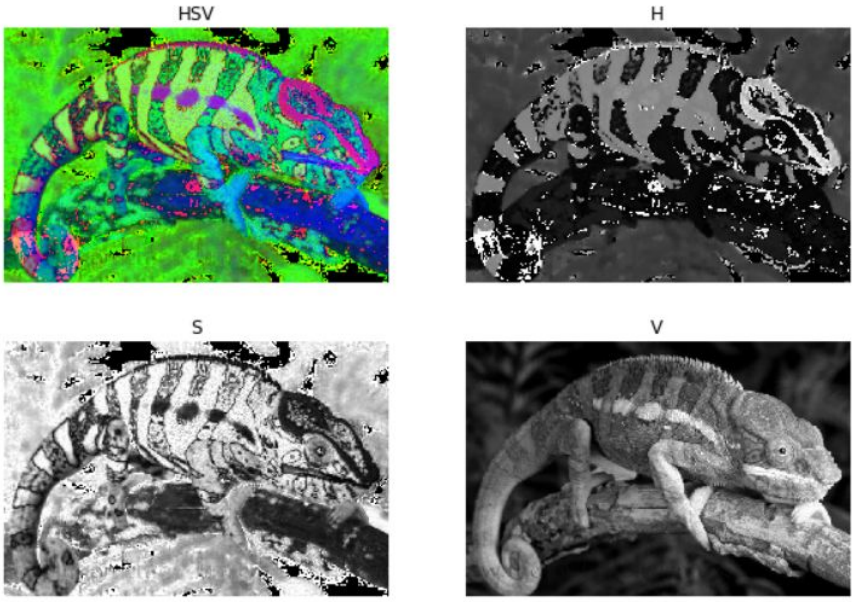
```
from skimage import util
C = 1 - util.img_as_float(R)
M = 1 - util.img_as_float(G)
Y = 1 - util.img_as_float(B)
CMY = np.zeros(image.shape)
CMY[:,:,0] = C
CMY[:,:,1] = M
CMY[:,:,2] = Y
plt.subplot(2,2,1);plt.imshow(CMY); plt.title('CMY');
plt.axis('off')
plt.subplot(2,2,2);plt.imshow(C,cmap='gray');
plt.title('C'); plt.axis('off')
plt.subplot(2,2,3);plt.imshow(M,cmap='gray');
plt.title('M'); plt.axis('off')
plt.subplot(2,2,4);plt.imshow(Y,cmap='gray');
plt.title('Y'); plt.axis('off')
```



C. HSV

```
HSV = color.rgb2hsv(image)
H = HSV[:,:,0]
S = HSV[:,:,1]
V = HSV[:,:,2]
plt.subplot(2,2,1);plt.imshow(HSV); plt.title('HSV');
plt.axis('off')
```

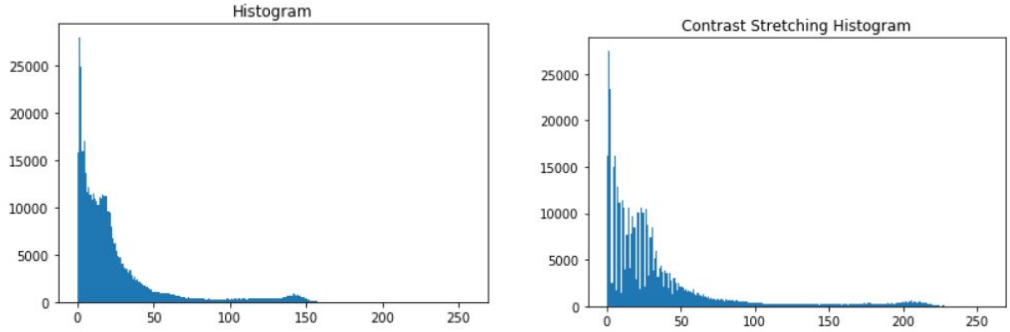
```
plt.subplot(2,2,2);plt.imshow(H,cmap='gray');
plt.title('H'); plt.axis('off')
plt.subplot(2,2,3);plt.imshow(S,cmap='gray');
plt.title('S'); plt.axis('off')
plt.subplot(2,2,4);plt.imshow(V,cmap='gray');
plt.title('V'); plt.axis('off')
```



2. Histogram Processing

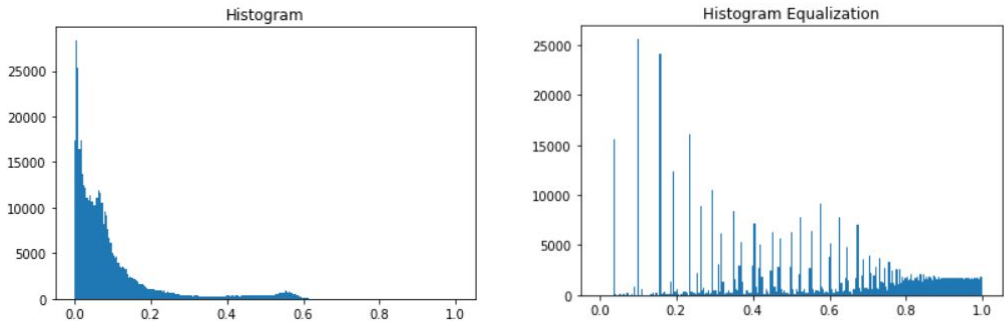
A. Contrast Stretching

Seperti yang sudah disampaikan pada tutorial histogram merupakan suatu teknik yang mampu memberikan deskripsi global pada tampilan citra, pada bagian color contrast stretching dan histogram equalization dapat digunakan untuk masing-masing bagian dari R, G, B dan seterusnya.



B. Histogram Equalization

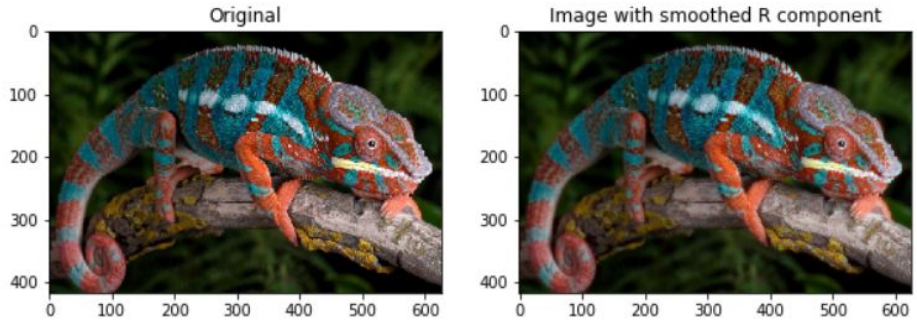
Seperti yang sudah disampaikan pada lab 1, histogram equalization adalah teknik dimana kita dapat membuat pemetaan pixel pada citra lebih menyebar pada kisaran 0-255



3. Smoothing & Sharpening

A. Smoothing

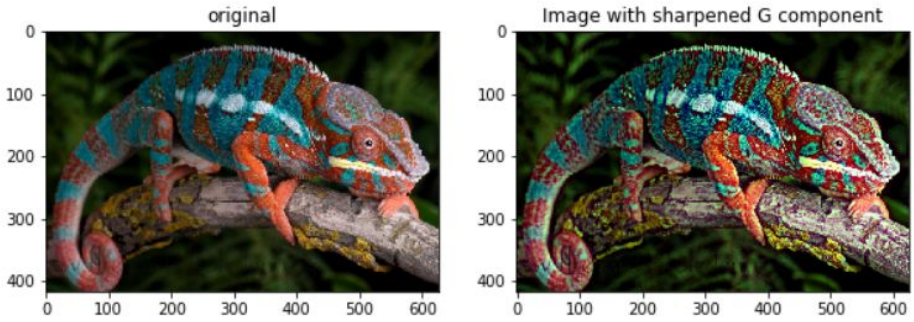
```
from skimage import filters, morphology
R2 = filters.rank.mean(R,selem=morphology.square(9))
RGB = util.img_as_ubyte(np.zeros(image.shape))
RGB[:,:,0] = R2; RGB[:,:,1] = G; RGB[:,:,2] = B
plt.subplot(1,2,1); plt.imshow(image, cmap='gray')
plt.title("Original")
plt.subplot(1,2,2); plt.imshow(RGB, cmap='gray')
plt.title('Image with smoothed R component');
plt.show()
```



B. Sharpening

```
G2 = util.img_as_ubyte(filters.unsharp_mask(G,
radius=5,amount=2))
```

```
RGB2 = util.img_as_ubyte(np.zeros(image.shape))
RGB2[:,:,0] = R; RGB2[:,:,1] = G2; RGB2[:,:,2] = B
plt.figure(figsize=(10,7))
plt.subplot(1,2,1); plt.imshow(image);
plt.title('original')
plt.subplot(1,2,2); plt.imshow(RGB2); plt.title('Image
with sharpened G component');
plt.show()
```



4. Color Segmentation

Color segmentation digunakan untuk mempartisi objek tertentu dari suatu citra menjadi beberapa region berdasarkan warnanya. Tujuan dari color segmentation adalah untuk mendapatkan representasi citra yang lebih bermakna. Metode color segmentation menggunakan distance dan threshold.

A. Distance (K-Means)

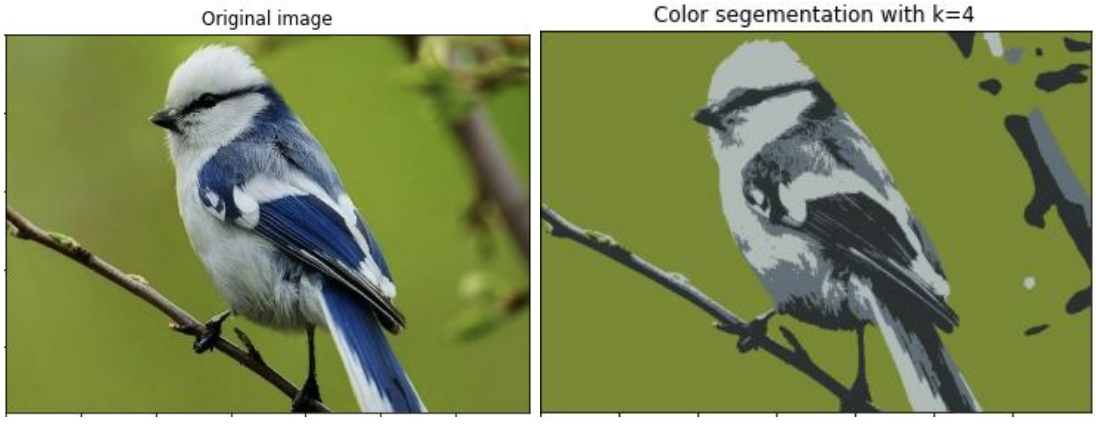
Metode K-Means mempartisi N piksel ke dalam K cluster. Cluster sendiri merupakan dapat dianggap sebagai himpunan yang terdiri dari piksel yang memiliki similaritas (warna) yang mirip.

Algoritma K-Means:

1. Pilih k centroid cluster secara acak.
2. Untuk setiap piksel citra bandingkan distance piksel dengan centroid masing - masing cluster.
3. Assign piksel ke dalam cluster dengan nilai distance piksel-centroid yang paling kecil.
4. Update nilai centroid cluster yang baru berdasarkan nilai piksel dalam masing - masing cluster (umumnya nilai centroid yang baru didapat dengan mencari nilai rata - rata piksel).
5. Bila nilai centroid yang baru maka algoritma telah selesai. Jika nilai centroid tidak sama, ulangi langkah 2 sampai 5.

Pada beberapa kasus kondisi nilai centroid dapat diberi nilai toleransi tertentu untuk menghentikan iterasi.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

bird_image = cv2.imread("bird.jpg")
bird_image = cv2.cvtColor(bird_image, cv2.COLOR_BGR2RGB)

pixel_values = bird_image.reshape((-1, 3))
pixel_values = np.float32(pixel_values)

criteria = (cv2.TERM_CRITERIA_EPS +
cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)

k = 4
_, labels, (centers) = cv2.kmeans(pixel_values, k, None,
criteria, 10, cv2.KMEANS_RANDOM_CENTERS)

centers = np.uint8(centers)
labels = labels.flatten()

segmented_image = centers[labels.flatten()]
segmented_image =segmented_image.reshape(bird_image.shape)
plt.imshow(segmented_image)
plt.title("Color segementation with k=4")
plt.show()
```



B. Threshold

Metode threshold mengambil piksel pada citra yang berada pada range tertentu. Umumnya proses color segmentation lebih mudah dilakukan dalam representasi HSV. Hal ini dikarenakan channel hue memodelkan warna dari citra sehingga mempermudah proses segmentasi.

```
hsv = cv2.cvtColor(bird_image, cv2.COLOR_RGB2HSV)
low_blue = (55, 0, 0)
high_blue = (118, 255, 255)
mask = cv2.inRange(hsv, low_blue, high_blue)

result = cv2.bitwise_and(bird_image, bird_image,
mask=mask)

plt.subplot(1, 2, 1)
plt.title("mask")
plt.imshow(mask, cmap="gray")
plt.subplot(1, 2, 2)
plt.title("result")
plt.imshow(result)
plt.show()
```