Lab 5
Pengolahan Citra
Feature Extraction
Senin, 16 November 2020

A. Brief Introduction

Feature Extraction adalah teknik-teknik yang dapat digunakan untuk mengekstraksi fitur dari sebuah citra. Fitur-fitur ini selanjutnya dapat diproses dan diolah, sebagai contoh untuk melakukan klasifikasi citra. Terdapat beberapa tipe fitur pada citra seperti *statistical feature*, *global feature*, dan *geometric features*.

B. Statistical Feature

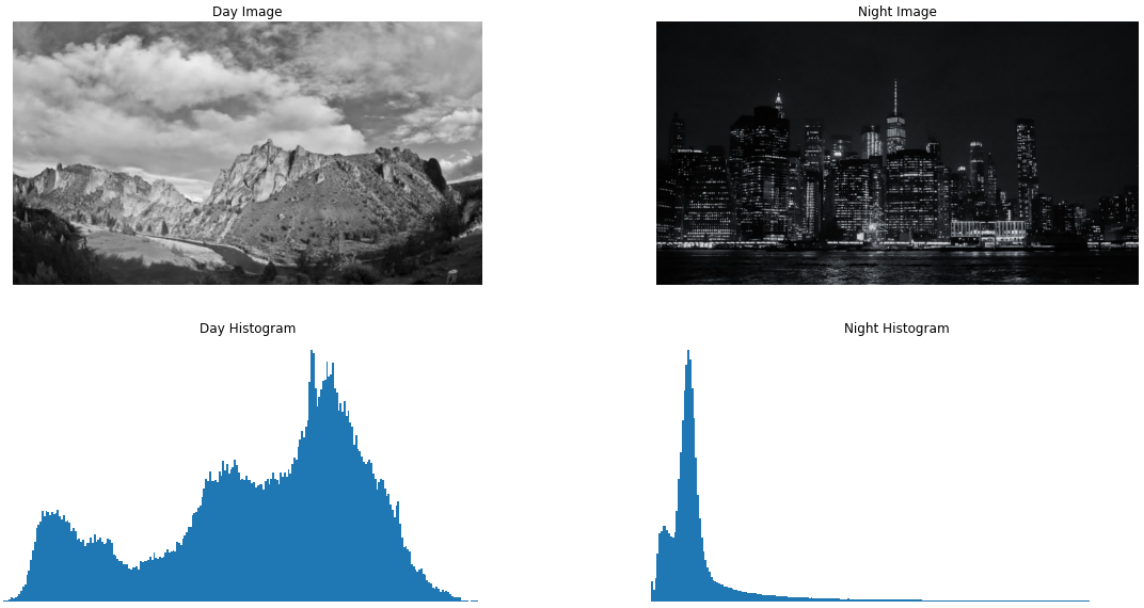
1. Image Histogram

Kita dapat menggunakan piksel-piksel yang terdapat pada citra sebagai fitur. Untuk memudahkan bagaimana cara kita melihat bagaimana piksel-piksel dapat berfungsi sebagai fitur, kita dapat menggunakan histogram untuk meneliti bagaimana cara kita menggunakan fitur tersebut.

Misalkan kita ingin mengklasifikasikan apakah citra *grayscale* yang diberikan diambil pada malam hari atau siang hari. Kita bisa menampilkan terlebih dahulu histogram untuk citra siang dan malam.

```
day = io.imread('day1.jpg')
night = io.imread('night1.jpg')

# Show image
plt.figure(figsize=(20,10))
plt.subplot(2, 2, 1)
plt.title('Day Image')
plt.imshow(day)
plt.axis('off')
plt.subplot(2, 2, 2)
plt.title('Night Image')
plt.imshow(night)
plt.axis('off')
plt.subplot(2, 2, 3)
plt.title('Day Histogram')
plt.hist(day.flatten(), 256, range=(0,255))
plt.axis('off')
plt.subplot(2, 2, 4)
plt.title('Night Histogram')
```

```
plt.hist(night.flatten(), 256, range=(0,255))
plt.axis('off')
plt.show()
```



Dapat kita lihat dari hasil histogram yang ditampilkan di atas bahwa untuk citra pada siang hari memiliki distribusi yang cenderung ke kanan (intensitas lebih tinggi) dibanding citra pada malam hari yang distribusi histogram cendung ke kiri (intenstitas lebih rendah). Hal tersebut masuk akal karena pada umumnya siang hari lebih cerah dan tentu memiliki intensitas lebih tinggi.

Dari fitur ini kita bisa membuat sebuah model (tidak harus berbasis *machine learning*) untuk memprediksi apakah citra yang diberikan merupakan citra siang hari atau malam hari. Berikut adalah contoh cara untuk memprediksi apakah citra merupakan citra siang hari atau malam hari.

```
def predict(file_name):
 img = io.imread(file_name)
 frequency_count = np.bincount(img.flatten())
 cum_sum = np.cumsum(frequency_count)

 # lebih banyak intensitas rendah, prediksi sebagai malam
 if cum_sum[127] > (cum_sum[-1] - cum_sum[127]):
   return 'night'
 else:
   return 'day'
```

```
import os

folders = ['day', 'night']
for folder in folders:
 for filename in os.listdir(folder):
   filename = f'{folder}/{filename}'
   print(f'image {filename} predicted as
{predict(filename)}')
```

Apabila kode tersebut dijalankan akan didapatkan hasil sebagai berikut.

```
image day/2.jpg predicted as day
image day/3.jpg predicted as day
image day/1.jpg predicted as day
image night/3.jpg predicted as night
image night/1.jpg predicted as night
image night/2.jpg predicted as night
```

Dapat dilihat bahwa hasil prediksi yang didapatkan memiliki akurasi 100% hanya dengan menggunakan fitur simpel seperti histogram.

Contoh di atas adalah salah satu contoh yang menggunakan histogram pada citra *grayscale*. Tetapi penggunaan Image Histogram ini tidak hanya berfokus pada histogram *grayscale*. Terdapat banyak variasi lain yang dapat digunakan seperti menggunakan histogram RGB maupun menggunakan histogram HSI.

2. **Image Segmentation**
   Dengan melakukan segmentasi terlebih dahulu pada citra. Maka kita bisa mendapatkan fitur-fitur berupa banyak objek, bentuk objek yang relevan, dan masih banyak lagi. Cara untuk melakukan segmentasi akan dibahas pada lab selanjutnya.

C. Global Feature

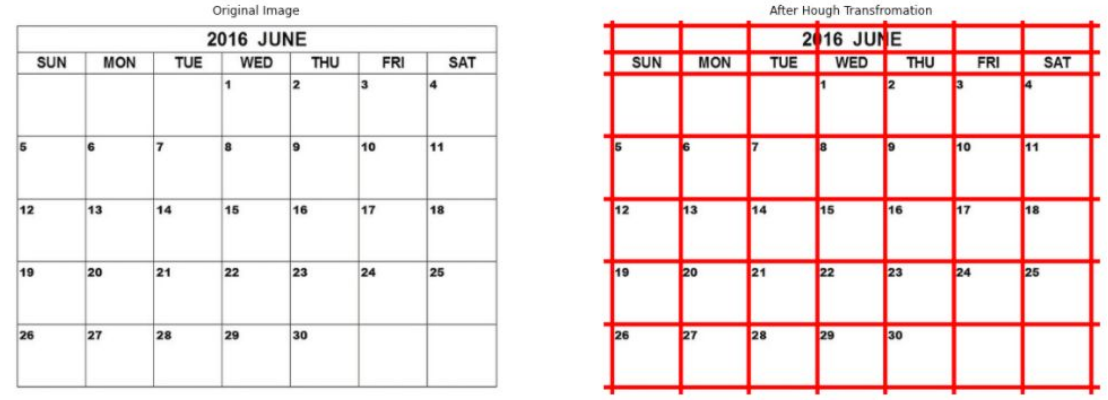
1. **Fourier Transform**
   Seperti yang telah dipelajari pada lab sebelumnya, kita dapat menggunakan frekuensi hasil fourier transform untuk dijadikan sebagai fitur. Teknik ini tidak akan dibahas lebih lanjut pada lab kali ini karena telah dibahas pada lab sebelumnya.

2. **Hough Transformation**
   Hough Transformation adalah algoritma yang dapat mendeteksi bentuk-bentuk pada citra selama bentuk tersebut dapat direpresentasikan dalam garis lurus. Untuk melakukan Hough Transformation, kita harus mencari terlebih dahulu

edge-edge pada citra tersebut. Hal ini bisa dicapai dengan menggunakan algoritma Canny (akan dibahas lebih lanjut pada lab selanjutnya).

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from skimage import io, color

img = io.imread('calendar.jpg')
original_image = np.copy(img)

# convert image to gray
gray = (color.rgb2gray(img) * 255).astype(np.uint8)

# find all of the edges using canny
edges = cv2.Canny(gray, 50, 150, apertureSize=3)
# find all of the lines using hough transformation
lines = cv2.HoughLines(edges, 1, np.pi/180, 200)

# plotting line
for line in lines:
   for rho,theta in line:
       a = np.cos(theta)
       b = np.sin(theta)
       x0 = a*rho
       y0 = b*rho
       x1 = int(x0 + 1000*(-b))
       y1 = int(y0 + 1000*(a))
       x2 = int(x0 - 1000*(-b))
       y2 = int(y0 - 1000*(a))
       cv2.line(img, (x1,y1), (x2,y2), (255,0,0), 2)

# Show image
plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.title('Original Image')
plt.imshow(original_image)
plt.axis('off')
plt.subplot(1,2,2)
plt.title('After Hough Transfromation')
plt.imshow(img,)
plt.axis('off')
```

```
plt.show()
```



Hasil dari hough transform dengan *library* OpenCV di atas adalah daftar garis atau *edge* yang dideteksi dari citra. Dengan informasi ini, kita dapat mengekstrak nilai-nilai fitur sesuai kebutuhan, misal jumlah garis/*edge*, bentuk *edge*, manipulasi aritmetika dari *edge* (misal hitung luas, walau ini lebih mudah dengan segmentasi *region*), dan lainnya. Pemilihan nilai yang diekstrak bergantung pada karakteristik yang dibutuhkan dan *task* yang dikerjakan (misal klasifikasi citra) dan perlu dicoba secara empiris.

D. Geometrical Feature

Mengekstrak bentuk geometris dari sebuah citra.

1. Image Morphology

Mathematical morphology dapat digunakan juga untuk melakukan ekstraksi fitur dari sebuah gambar. Image morphology dapat dilakukan dengan dilatasi, erosi, dan lain-lain seperti yang telah dibahas pada lab 3 sebelumnya.

2. Template Matching

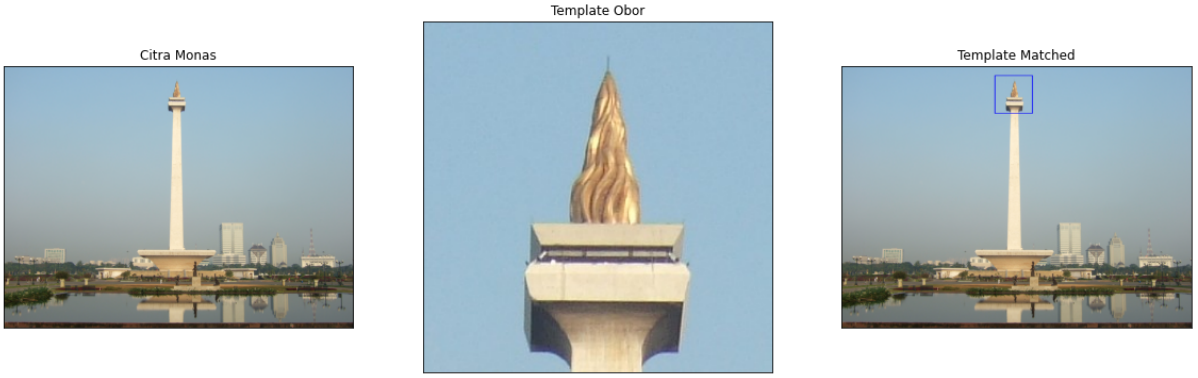
Template matching adalah metode untuk mencari dan menemukan bagian kecil dari sebuah citra sesuai dengan template yang diinginkan. Template tersebut akan berperan sebagai jendela yang akan digeser sepanjang citra (konvolusi) dan mencocokkan template tersebut dengan bagian citra yang tercover oleh template.

Berikut contoh template matching untuk bagian obor dari citra monas.

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

# Import citra yang dibutuhkan
citra_monas = cv.cvtColor(cv.imread('monas.jpg'),
```

```
cv.COLOR_BGR2RGB)
template_obor = cv.cvtColor(cv.imread('obor-monas.jpg'),
cv.COLOR_BGR2RGB)

# Ubah gray
citra_monas_gray = cv.cvtColor(citra_monas,
cv.COLOR_RGB2GRAY)
template_obor_gray = cv.cvtColor(template_obor,
cv.COLOR_RGB2GRAY)

# Menyimpan width dan height template
w, h = template_obor_gray.shape[::-1]

# Apply template matching
# parameter ketiga merupakan method, bisa melihat
dokumentasi openCV untuk penjelasan lebih lanjut dan
pilihan method lainnya
res = cv.matchTemplate(citra_monas_gray,
template_obor_gray, cv.TM_CCOEFF_NORMED)

# Copy citra monas untuk ditandai bounding box
template_matched = citra_monas.copy()

# Tentukan threshold
threshold = 0.9
loc = np.where( res >= threshold)

# Buat bounding box dengan rectangle. Rectangle memiliki 4
parameter yaitu (image, start_point, end_point, color,
thickness)
for point in zip(*loc[::-1]):
  cv.rectangle(template_matched, point, (point[0] + w,
point[1] + h), (0,0,255), 2)

# Menampilkan citra
plt.figure(figsize=(20,10))
plt.subplot(131),plt.imshow(citra_monas)
plt.title('Citra Monas'), plt.xticks([]), plt.yticks([])
plt.subplot(132),plt.imshow(template_obor)
plt.title('Template Obor'), plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(template_matched)
plt.title('Template Matched'), plt.xticks([]),
```

```
plt.yticks([])
plt.show()
```



OpenCV memiliki beberapa pilihan method yang dapat digunakan. Selain OpenCV, dapat juga menggunakan library lain seperti skimage. Selamat mengeksplor :)

credit gambar: Wikimedia