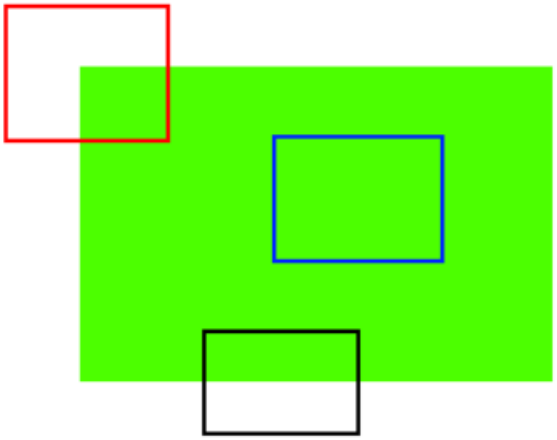
MODUL ESKTRAKSI FITUR DAN FEATURE DETECTION

Apa itu fitur? 'Fitur' merupakan bagian dari citra yang dapat diidentifikasi dengan mudah oleh komputer. Sebagai contoh, pada gambar berikut:



Manakah yang merupakan 'fitur' yang mudah untuk diidentifikasi? Jawabnya adalah ujung persegi panjang yang ditandai oleh kotak berwarna merah. Kedua fitur lain cukup sulit untuk diidentifikasi:

- Kotak warna biru dapat berada pada posisi manapun di dalam persegi panjang warna hijau
- kotak warna hitam dapat berada dimana saja selama berada pada tepian persegi panjang.

Dengan demikian, sebuah citra dapat dilatih untuk mencari titik-titik yang mudah dilacak (Good Features To Track) dengan membuat kategori untuk tiap piksel dengan tetangga piksel masing-masing.

Ekstraksi fitur adalah proses mendapatkan fitur dari citra digital dengan menggunakan metode tertentu untuk menghitung jumlah piksel pada citra (Amirullah, 2018). Ekstrasi fitur terdiri dari beberpa jenis diantaranya ekstrasi fitur berdasarkan bentuk, warna dan tekstur.

a. Ekstraksi Fitur warna

Gambar tersusun dari piksel-piksel yang memiliki ukuran intensitas warna masing-masing. Sebaran warna di tiap-tiap piksel ditunjukkan oleh histogram. Histogram menunjukkan distribusi piksel berdasarkan intensitas graylevel (derajat keabuan) yang dimiliki tiap-tiap piksel. Penggunaan histogram sebagai metode ekstraksi ciri didasarkan pada perbedaan sebaran atau distribusi piksel di masing-masing gambar. Pada proses ektraksi ciri warna diawali dengan merubah aras warna RGB menjadi aras keabuan (grayscale). Nilai warna keabuan dari masing-masing piksel yang menyusun gambar di kelompokkan menjadi 8 kelompok rentang nilai piksel warna (bin). Tiap kelompok jumlah anggota kemudian dinormalisasi dengan cara di bagi dengan hasil perkalian panjang dan lebar gambar (banyak piksel warna penyusun gambar).

b. Ekstrasi Fitur Tekstur

Ciri tekstur merupakan ciri penting dalam sebuah gambar yang merupakan informasi berupa susunan struktur permukaan suatu gambar. Ekstrasi fitur tekstur dapat menggunakan metode **Local Ternary Pattern** (**LTP), local binary pattern (LBP) dan Gray Level oCcurance Matrix (GLCM)** sebagai matrik pengambilan nilai keabuan dari sebuah gambar. Berikut merupakan tahapan yang digunakan dalam pengambilan ciri tekstur dari sebuah gambar menggunakan GLCM.

- Citra warna dirubah menjadi citra grayscale
- Masing-masing nilai dari RGB citra dirubah menjadi abu-abu dengan menggunakan rumus sebagai berikut: keabuan = 0.2989 * R + 0.5870 * G + 0.1140 * B (1)
- Piksel baru = setPixel(255, nilai keabuan, nilai keabuan, nilai keabuan)
- Segmentasi nilai warna ke dalam 16 bin
- Hitung nilai-nilai co-occurance matrix dalam empat arah masing-masing 00 , 450 , 900 , dan 1350
- Hitung informasi ciri tekstur yaitu *contrast, correlation, energy, homogeneity,* dan *entropy*.

Contrast dihitung dengan persamaan berikut:

$$\sum_k k^2 \left[\sum_i \sum_j p(i,j)\right] \quad (2)$$

Correlation dihitung dengan persamaan berikut:

$$\sum_{i,j} \frac{(i-\mu_i)(j-\mu_j)P(i,j)}{\sigma_i \sigma_j} \quad (3)$$

Energy dihitung dengan persamaan berikut:

$$\sum_{i,j} P(i,j)^2 \quad (4)$$

Homogeneity dihitung dengan persamaan berikut:

$$\sum_{i,j} \frac{P(i,j)}{1+|i-j|} \quad (5)$$

Entropy dihitung dengan persamaan berikut:

$$-\sum_{i,j} P(i,j) log P(i,j) \quad (6)$$

c. Ekstrasi Fitur Bentuk

Ciri bentuk merupakan karakter dari suatu objek yang merupakan konfigurasi oleh garis dan kontur. Fitur bentuk dikategorikan bergantung pada teknik yang digunakan. Kategori tersebut adalah berdasarkan batas (boundary-based). dan berdasarkan daerah (region-based). Teknik berdasarkan batas (boundary-based) menggambarkan bentuk daerah dengan menggunakan karakteristik ekstrenal, contohnya adalah piksel sepanjang batas objek. Salah satu metode yang digunakan untuk ekstrasi fitur bentuk adalah **canny edge detection**. Metode **Canny Edge Detection** yang dipergunakan untuk menemukan bagian-bagian tepi dari sebuah objek. Edge detection adalah

menemukan bagian pada citra yang mengalami perubahan intensitas warna secara drastis. Algoritma Canny Edge Detection secara umum beroperasi sebagai berikut:

1) Penghalusan untuk mengurangi dampak noise terhadap pendeteksian tepi
2) Menghitung potensi gradient citra
3) Non-maximal suppression dari gradient citra untuk melokalisasi edge secara presisi
4) Hysteresis thresholding untuk melakukan klasifikasi akhir.

Latihan 1. EKSTRASI FITUR LBP

Load library dan fungsi untuk mendapatkan nilai pinggir untuk LBP pada gambar

```
import cv2 # OpenCV
import numpy as np # NumPy
from matplotlib import pyplot as plt # Matplotlib


def get_pixel(img, center, x, y):
    new_value = 0
    try:
        if img[x][y] >= center:
            new_value = 1
    except:
        pass
    return new_value
```

Fungsi untuk menghitung lbp pada pixel

```
def lbp_calculated_pixel(img, x, y):
    '''
    Format :

     64 | 128 |   1
    ----------------
     32 |   0 |   2
    ----------------
     16 |   8 |   4
    '''
    center = img[x][y]
    val_ar = []
    val_ar.append(get_pixel(img, center, x-1, y+1))     # top_right
    val_ar.append(get_pixel(img, center, x, y+1))       # right
    val_ar.append(get_pixel(img, center, x+1, y+1))     # bottom_right
    val_ar.append(get_pixel(img, center, x+1, y))       # bottom
    val_ar.append(get_pixel(img, center, x+1, y-1))     # bottom_left
    val_ar.append(get_pixel(img, center, x, y-1))       # left
    val_ar.append(get_pixel(img, center, x-1, y-1))     # top_left
```

```
    val_ar.append(get_pixel(img, center, x-1, y))       # top

    power_val = [1, 2, 4, 8, 16, 32, 64, 128]
    val = 0
    for i in range(len(val_ar)):
        val += val_ar[i] * power_val[i]
    return val
```

Fungsi menampilkan output

```
def show_output(output_list):
    output_list_len = len(output_list)
    figure = plt.figure(figsize=(20, 6))
    for i in range(output_list_len):
        current_dict = output_list[i]
        current_img = current_dict["img"]
        current_xlabel = current_dict["xlabel"]
        current_ylabel = current_dict["ylabel"]
        current_xtick = current_dict["xtick"]
        current_ytick = current_dict["ytick"]
        current_title = current_dict["title"]
        current_type = current_dict["type"]
        current_plot = figure.add_subplot(1, output_list_len, i+1)
        if current_type == "gray":
            current_plot.imshow(current_img, cmap = plt.get_cmap('gray'))
            current_plot.set_title(current_title)
            current_plot.set_xticks(current_xtick)
            current_plot.set_yticks(current_ytick)
            current_plot.set_xlabel(current_xlabel)
            current_plot.set_ylabel(current_ylabel)
        elif current_type == "histogram":
            current_plot.plot(current_img, color = "black")
            current_plot.set_xlim([0,260])
            current_plot.set_title(current_title)
            current_plot.set_xlabel(current_xlabel)
            current_plot.set_ylabel(current_ylabel)
            ytick_list = [int(i) for i in current_plot.get_yticks()]
            current_plot.set_yticklabels(ytick_list,rotation = 90)
        elif current_type == "normal":
            current_plot.imshow(current_img)
            current_plot.set_title(current_title)
            current_plot.set_xticks(current_xtick)
            current_plot.set_yticks(current_ytick)
            current_plot.set_xlabel(current_xlabel)
            current_plot.set_ylabel(current_ylabel)
```

```
    plt.show()
```

Fungsi main

```
image_file = 'lenna.jpg'
img_bgr = cv2.imread(image_file)
height, width, channel = img_bgr.shape
img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)

img_lbp = np.zeros((height, width,3), np.uint8)
for i in range(0, height):
    for j in range(0, width):
        img_lbp[i, j] = lbp_calculated_pixel(img_gray, i, j)
hist_lbp = cv2.calcHist([img_lbp], [0], None, [256], [0, 256])
output_list = []
output_list.append({"img": cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB),
                    "xlabel": "",
                    "ylabel": "",
                    "xtick": [],
                    "ytick": [],
                    "title": "Normal Image",
                    "type": "normal"
                   })
output_list.append({"img": img_gray,
                    "xlabel": "",
                    "ylabel": "",
                    "xtick": [],
                    "ytick": [],
                    "title": "Gray Image",
                    "type": "gray"
                   })
output_list.append({"img": img_lbp,
                    "xlabel": "",
                    "ylabel": "",
                    "xtick": [],
                    "ytick": [],
                    "title": "LBP Image",
                    "type": "gray"
                   })
output_list.append({"img": hist_lbp,
                    "xlabel": "Bins",
                    "ylabel": "Number of pixels",
                    "xtick": None,
                    "ytick": None,
```

```
                    "title": "Histogram(LBP)",
                    "type": "histogram"
                   })

show_output(output_list)

cv2.waitKey(0)
cv2.destroyAllWindows()
print("LBP Program selesai ")
```

Mencoba untuk menggunakan birghtness yang berbeda

```
def brighter(nilai, img):
    img_b = np.zeros((height, width,3), np.uint8)
    for y in range(0, height):
        for x in range(0, width):
            red = img[y][x][2] + nilai
            green = img[y][x][1] + nilai
            blue = img[y][x][0] + nilai
            if red > 255:
                red = 255
            if red < 0:
                red = 0
            if green > 255:
                green = 255
            if green < 0:
                green = 0
            if blue > 255:
                blue = 255
            if blue < 0:
                blue = 0
            img_b[y][x] = (red, green, blue)
    return img_b

image_file = 'lenna.jpg'
image_file2 = 'woman.jpg'
img_bgr = cv2.imread(image_file)
height, width, channel = img_bgr.shape
img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
img_gray2 = cv2.cvtColor(brighter(50, img_bgr), cv2.COLOR_BGR2GRAY)
img_bgr2 = cv2.imread(image_file2)
height2, width2, channel2 = img_bgr2.shape
img_gray3 = cv2.cvtColor(img_bgr2, cv2.COLOR_BGR2GRAY)

def lbp(img_gray, height, width, img_bgr):
```

```
    img_lbp = np.zeros((height, width,3), np.uint8)
    for i in range(0, height):
        for j in range(0, width):
            img_lbp[i, j] = lbp_calculated_pixel(img_gray, i, j)
    hist_lbp = cv2.calcHist([img_lbp], [0], None, [256], [0, 256])
    output_list = []
    output_list.append({"img": cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB),
                        "xlabel": "",
                        "ylabel": "",
                        "xtick": [],
                        "ytick": [],
                        "title": "Normal Image",
                        "type": "normal"
                       })
    output_list.append({"img": img_gray,
                        "xlabel": "",
                        "ylabel": "",
                        "xtick": [],
                        "ytick": [],
                        "title": "Gray Image",
                        "type": "gray"
                       })
    output_list.append({"img": img_lbp,
                        "xlabel": "",
                        "ylabel": "",
                        "xtick": [],
                        "ytick": [],
                        "title": "LBP Image",
                        "type": "gray"
                       })
    output_list.append({"img": hist_lbp,
                        "xlabel": "Bins",
                        "ylabel": "Number of pixels",
                        "xtick": None,
                        "ytick": None,
                        "title": "Histogram(LBP)",
                        "type": "histogram"
                       })

    show_output(output_list)

    cv2.waitKey(0)
    cv2.destroyAllWindows()
    print("LBP Program selesai ")
```

```
lbp(img_gray, height, width, img_bgr)
lbp(img_gray2, height, width, img_bgr)
lbp(img_gray3, height2, width2, img_bgr2)
```

LATIHAN 2. EKSTRASI FITUR BERDASARKAN WARNA

Import library

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

Membaca gambar

```
# img = imageio.imread("gambar4.jpg")
img_bgr = cv2.imread("gambar4.jpg")
img = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
height, width, channel = img_bgr.shape
```

Membuat variable untuk menyimpan data gambar

```
hgr = np.zeros((256))
hgg = np.zeros((256))
hgb = np.zeros((256))
hgrgb = np.zeros((768), dtype=np.int32)
```

Mengisi nilai dalam array hg dengan 0

```
def makeItZero():
    for x in range(0, 256):
        hgr[x] = 0
        hgg[x] = 0
        hgb[x] = 0

    for x in range(0, 768):
        hgrgb[x] = 0
```

Menghitung nilai dari gambar

```
makeItZero()

# th = int(256/64)
temp = [0]
for y in range(0, height):
    for x in range(0, width):
        red = int(img[y][x][0])
        green = int(img[y][x][1])
```

```
        blue = int(img[y][x][2])
        red = red + 0
        green = green + 256
        blue = blue + 512
#         temp.append(green)
        hgrgb[red] += 1
        hgrgb[green] += 1
        hgrgb[blue] += 1

binsrgb = np.linspace(0, 768, 100)

binsr = np.linspace(0, 0, 100)
plt.hist(hgr, binsr, color = "red", alpha=0.5)
binsg = np.linspace(0, 256, 100)
plt.hist(hgr, binsg, color = "green", alpha=0.5)
binsb = np.linspace(0, 768, 100)
plt.hist(hgr, binsb, color = "blue", alpha=0.5)
#plt.hist(hgr, binsrgb, alpha=0.5)
plt.plot(hgrgb)
plt.title("Histogram Red Green Blue")
plt.show()
```

Menggunakan OpenCV

```
hist_img = cv2.calcHist([img], [0], None, [768], [0, 768])
plt.plot(hist_img)
plt.title("Histogram Red Green Blue")
plt.show()
```

Menampilkan histogram

```
makeItZero()
for y in range(0, img.shape[0]):
    for x in range(0, img.shape[1]):
        red = img[y][x][0]
        green = img[y][x][1]
        blue = img[y][x][2]
        hgr[red] += 1
        hgg[green] += 1
        hgb[blue] += 1
def plot_result(red, green, blue):
    bins = np.linspace(0, 256, 128)

    fig, (ax1, ax2, ax3) = plt.subplots(1, 3, sharey=True)

    for ax in [ax1, ax2, ax3]:
        ax.spines["top"].set_visible(False)
```

```
        ax.spines["right"].set_visible(False)
        ax.spines["left"].set_visible(False)
        ax.spines["bottom"].set_visible(False)
        ax.grid(color='b', linestyle='--', linewidth=0.5, alpha=0.3)
        ax.tick_params(direction='out', color='b', width='1')

    ax1.set_title('Red')
    ax2.set_title('Green')
    ax3.set_title('Blue')

    ax1.hist(red, bins, color="red", alpha=1)
    ax2.hist(green, bins, color="green", alpha=1)
    ax3.hist(blue, bins, color="blue", alpha=1)

plt.rcParams['figure.figsize'] = [20, 7]
plot_result(hgr, hgg, hgb)
```

LATIHAN 3. EKSTRASI FITUR GLCM

```
import matplotlib.pyplot as plt

from skimage.feature import greycomatrix, greycoprops
from skimage import data
PATCH_SIZE = 21

# load image
image = data.camera()

grass_locations = [(280, 454), (342, 223), (444, 192), (455, 455)]
grass_patches = []
for loc in grass_locations:
    grass_patches.append(image[loc[0]:loc[0] + PATCH_SIZE,
                               loc[1]:loc[1] + PATCH_SIZE])

sky_locations = [(38, 34), (139, 28), (37, 437), (145, 379)]
sky_patches = []
for loc in sky_locations:
    sky_patches.append(image[loc[0]:loc[0] + PATCH_SIZE,
                             loc[1]:loc[1] + PATCH_SIZE])

# menghitung GLCM
xs = []
ys = []
for patch in (grass_patches + sky_patches):
    glcm = greycomatrix(patch, distances=[5], angles=[0], levels=256,
```

```
                        symmetric=True, normed=True)
    xs.append(greycoprops(glcm, 'dissimilarity')[0, 0])
    ys.append(greycoprops(glcm, 'correlation')[0, 0])

fig = plt.figure(figsize=(8, 8))

# tampilkan original image dengan lokasi patches
ax = fig.add_subplot(3, 2, 1)
ax.imshow(image, cmap=plt.cm.gray,
          vmin=0, vmax=255)
for (y, x) in grass_locations:
    ax.plot(x + PATCH_SIZE / 2, y + PATCH_SIZE / 2, 'gs')
for (y, x) in sky_locations:
    ax.plot(x + PATCH_SIZE / 2, y + PATCH_SIZE / 2, 'bs')
ax.set_xlabel('Original Image')
ax.set_xticks([])
ax.set_yticks([])
ax.axis('image')

# plot (dissimilarity, correlation)
ax = fig.add_subplot(3, 2, 2)
ax.plot(xs[:len(grass_patches)], ys[:len(grass_patches)], 'go',
        label='Grass')
ax.plot(xs[len(grass_patches):], ys[len(grass_patches):], 'bo',
        label='Sky')
ax.set_xlabel('GLCM Dissimilarity')
ax.set_ylabel('GLCM Correlation')
ax.legend()

# display
for i, patch in enumerate(grass_patches):
    ax = fig.add_subplot(3, len(grass_patches), len(grass_patches)*1 + i + 1)
    ax.imshow(patch, cmap=plt.cm.gray,
              vmin=0, vmax=255)
    ax.set_xlabel('Grass %d' % (i + 1))

for i, patch in enumerate(sky_patches):
    ax = fig.add_subplot(3, len(sky_patches), len(sky_patches)*2 + i + 1)
    ax.imshow(patch, cmap=plt.cm.gray,
              vmin=0, vmax=255)
    ax.set_xlabel('Sky %d' % (i + 1))


# display
fig.suptitle('Grey level co-occurrence matrix features', fontsize=14, y=1.05)
```

```
plt.tight_layout()
plt.show()
```

FEATURE DETECTION

Pada bagian ini kita akan menggunakan teknik deteksi fitur dan matching untuk melakukan beberapa pengolahan citra dan memperoleh informasi. Ingat bahwa pengolahan citra perlu dilakukan secara bertahap dengan teknik yang boleh jadi berbeda pada kondisi yang berbeda. Untuk itu ingat kembali pelajaran pada minggu-minggu sebelumnya sebelum melanjutkan pada materi di Minggu ini.

Review: Ekstraksi objek pada citra

Pada saat melihat sebuah foto atau gambar digital, manusia dapat dengan mudah membedakan objek yang digambarkan pada foto tersebut. Sebagai contoh, dari gambar berikut:

Manusia dapat dengan mudah membedakan jeruk dan daun, jeruk yang dekat dengan yang jauh, serta menghitung jumlah jeruk yang ada, misalnya. Untuk komputer, hal ini menjadi kompleks karena komputer hanya melihat serangkaian pixel, tanpa kemampuan untuk memisahkan fitur tersebut. Pada awal pertemuan tentang materi PCD telah dibahas mengenai cara 'mengajarkan' komputer agar mampu melihat dan membedakan warna (Hue) sehingga dapat digunakan untuk ekstraksi fitur. Pada minggu lalu juga telah dibahas mengenai metode filtering dan thresholding yang dapat digunakan untuk menonjolkan tepi sebuah objek sehingga dapat dilakukan ekstraksi fitur. Dengan menggunakan nilai piksel pada sebuah citra digital, kita dapat memberikan informasi pada komputer agar dapat mengenali suatu objek. Namun bagaimana dengan kedalaman objek tersebut? Pembahasan ini merupakan pengantar

untuk materi depth reconstruction melalui pengolahan citra digital. Terdapat beberapa algoritma untuk melakukan deteksi pojok ('Corner Detector') dari sebuah citra digital. Beberapa metodr yang cukup populer adalah:

- [Harris Corner Detector](https://docs.opencv.org/3.4/dc/d0d/tutorial_py_features_harris.html)
- [Shi-Tomasi dan Good Features to Track] (https://docs.opencv.org/3.4/d4/d8c/tutorial_py_shi_tomasi.html)

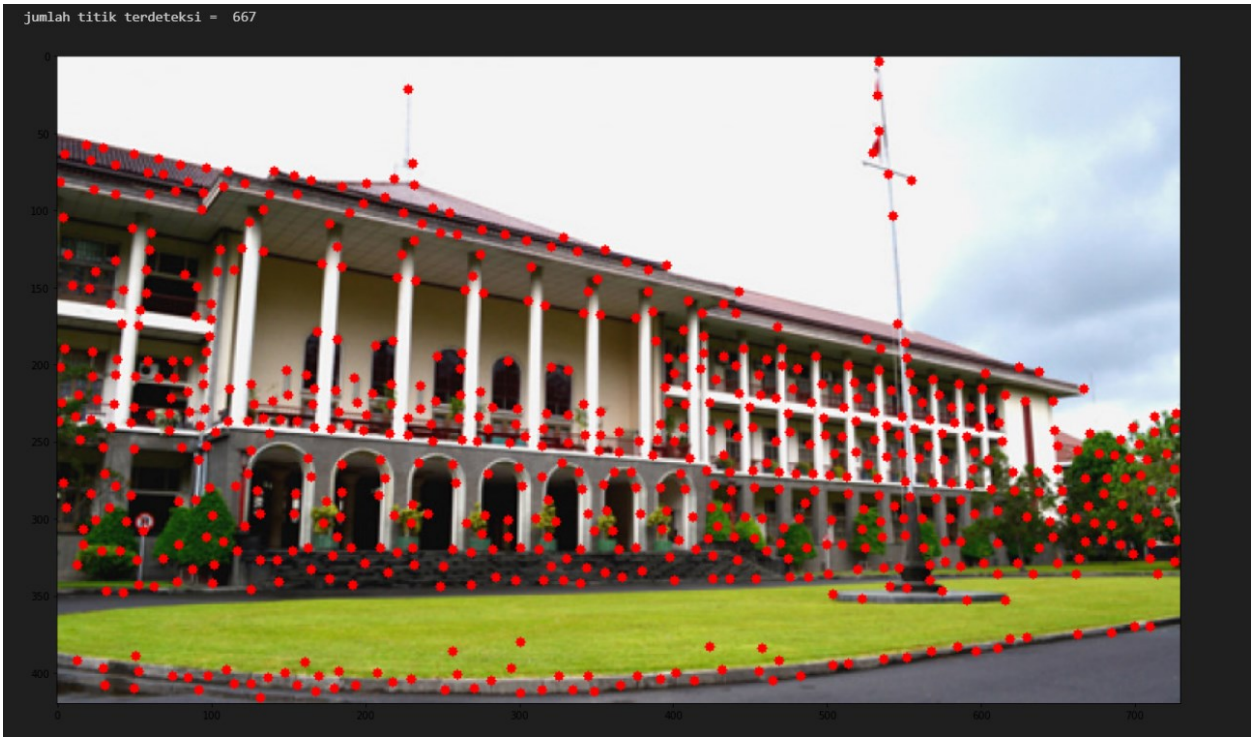
Lakukan latihan berikut untuk memahami mengenai Good Features to Track pada sebuah Citra Digital.

Latihan 4: Menggunakan Corner Detector

Lakukan latihan berikut menggunakan kedua metode yang tersedia pada OpenCV (Harris Corner dan Shi-Tomasi). Gunakan gambar yang berbeda dan bandingkan jumlah corner yang berhasil dideteksi oleh kedua algoritma tersebut.

```
# Menggunakan Shi-Tomasi GFTT untuk deteksi ujung (corner detection)

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

# gunakan gambar
img = cv2.imread('gedungpusat.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)


# deteksi pojok dengan GFTT
corners = cv2.goodFeaturesToTrack(gray,1000,0.01,10)
corners = np.int0(corners)

# menampilkan jumlah titik terdeteksi dengan fungsi numpy (np.ndarray.shape)
print("jumlah titik terdeteksi = ", corners.shape[0])

# untuk ditampilkan di Matplotlib, urutan band dibalik
rgb = cv2.cvtColor(img,cv.COLOR_BGR2RGB)

# perbesar ukuran hasil plotting
plt.rcParams["figure.figsize"] = (20,20)

# untuk tiap pojok yang terdeteksi, munculkan pada gambar
for i in corners:
    x,y = i.ravel()
    cv2.circle(rgb,(x,y),3,255,-1)
plt.imshow(rgb),plt.show()
```

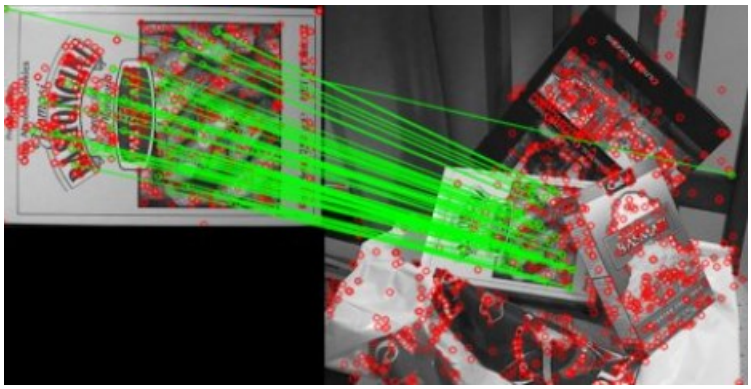
Tampilan ketika kode diatas dijalankan:



a. Feature Matching

Rujukan:https://pysource.com/2018/03/23/feature-matching-brute-force-opencv-3-4-with-python-3-tutorial-26/

Apabila feature detector digunakan untuk mencari titik yang dapat dipasangkan, maka **Feature Matching** digunakan untuk memasangkan masing-masing keypoint yang sudah terdeteksi pada satu citra dengan titik titik yang memiliki kesamaan karakter pada citra lain. Dengan demikian, dapat dikatakan bahwa hasil dari feature matching adalah **Tie-Points** pada sebuah proses fotogrametri.



Pada Gambar di atas, lingkaran berwarna merah merupakan fitur (keypoints) yang dideteksi dengan menggunakan **Feature Detector** pada OpenCV, sedangkan garis-garis berwarna hijau merupakan titik-titik yang berhasil dipasangkan berdasarkan atas kesamaan karakteristik (feature matching).Terdapat beberapa algoritma yang berbeda pula pada metode Feature Matching, antara lain:

- Brute-Force Matching
- FLANN

Penggunaan kedua model feature matching tersebut perlu disesuaikan dengan algoritma keypoint detector yang digunakan.

Latihan 5: Feature Detection and Matching

```
# Contoh Script untuk feature detection and Matching
# Modifikasi script ini untuk mencoba metode yang berbeda

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

# Gunakan gambar yang ada pada laptop masing-masing
img1 = cv2.imread('webarebears.jpg')          # gambar yang dituju
img2 = cv2.imread('ice_bear.jpg')             # gambar yang dicari
gray1= cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
gray2= cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)

# Menggunakan Detector SIFT
sift = cv2.xfeatures2d.SIFT_create()

# Mencari Keypoint dengan SIFT
kp1, des1 = sift.detectAndCompute(gray1,None)
kp2, des2 = sift.detectAndCompute(gray2,None)

# Melakukan Matching dari hasil deteksi keypoints menggunakan
# BruteForce Matcher
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)

# Uji rasio matching sederhana
good = []
for m,n in matches:
    if m.distance < 0.5*n.distance:
        good.append([m])
img3 = None

# menggambar hasil match pada gambar baru (IMG3)
```

```
img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,img3,flags=2)
plt.imshow(cv2.cvtColor(img3, cv2.COLOR_BGR2RGB)),plt.show()
```

Tampilan contoh kode program **latihan 5, silahkan kalian gunakan foto lain dan tampilkan hasil yang muncul**:

Dari contoh di atas dapat dilihat bahwa dengan metode deteksi keypoint menggunakan SIFT kita dapat melakukan matching pada objek yang mengalami rotasi (ingat bahwa SIFT=Scale-Invariant Feature Transform). Aplikasi dari algoritma ini untuk bidang fotogrametri adalah pada deteksi titik tie-points secara otomatis pada dua buah foto yang bertampalan, sedangkan pada bidang remote sensing, metode ini digunakan dalam pembuatan mosaik otomatis untuk mendukung metode registrasi citra, semisal dengan RPC.

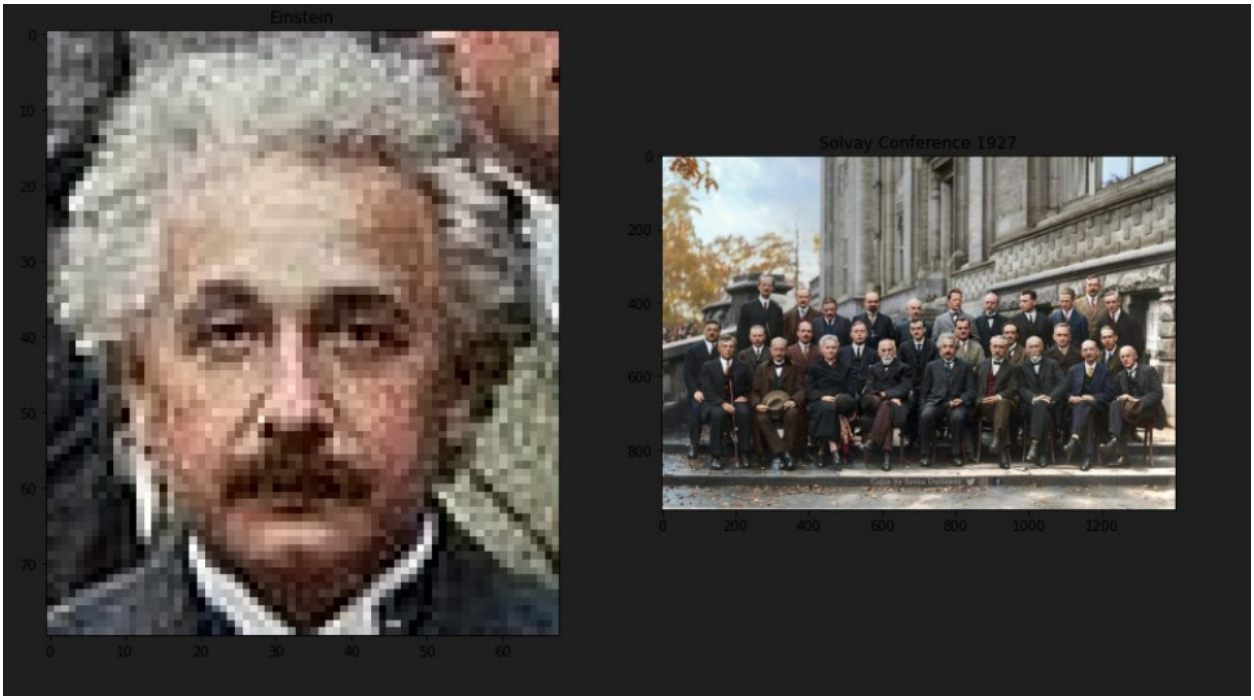
b. Template Matching

Dari pelajaran di atas serta contoh-contoh praktikum beberapa minggu sebelumnya, kita telah melihat bagaimana algoritma level rendah seperti transformasi colorspace, edge detection, feature detection dan matching dapat sangat berguna untuk memperoleh berbagai informasi yang kita inginkan pada sebuah citra. Dengan memahami cara kerja dari masing-masing algoritma tersebut kita akan lebih mudah mengetahui bagaimana perangkat lunak pengolahan citra bekerja di belakang layar serta bagaimana melakukan pengaturan parameter yang dapat digunakan untuk mengoptimalkan hasil pemrosesan citra yang kita inginkan.

Pada bagian ini kita akan menggunakan salah satu fungsi OpenCV yang digunakan untuk meringkas beberapa langkah di atas, yaitu deteksi keypoint, matching dan deteksi tepi untuk mendeteksi posisi wajah dari serangkaian gambar. Algoritma ini disebut sebagai 'Template Matching'. OpenCV menyediakan fungsi `cv2.matchTemplate()` untuk keperluan Template Matching ini dengan berbagai metode deteksi berdasarkan tingkat kemiripan antara `template` atau gambar yang dicari

dengan objek pada gambar yang tersedia , misalnya dengan menghitung rerata korelasi antar nilai piksel pada template tersebut dengan objek yang dicari. Latihan berikut menunjukkan aplikasi dari berbagai metode tersebut untuk mencari gambar yang diberikan.

Latihan 6. template matching mendeteksi einsten pada gambar.

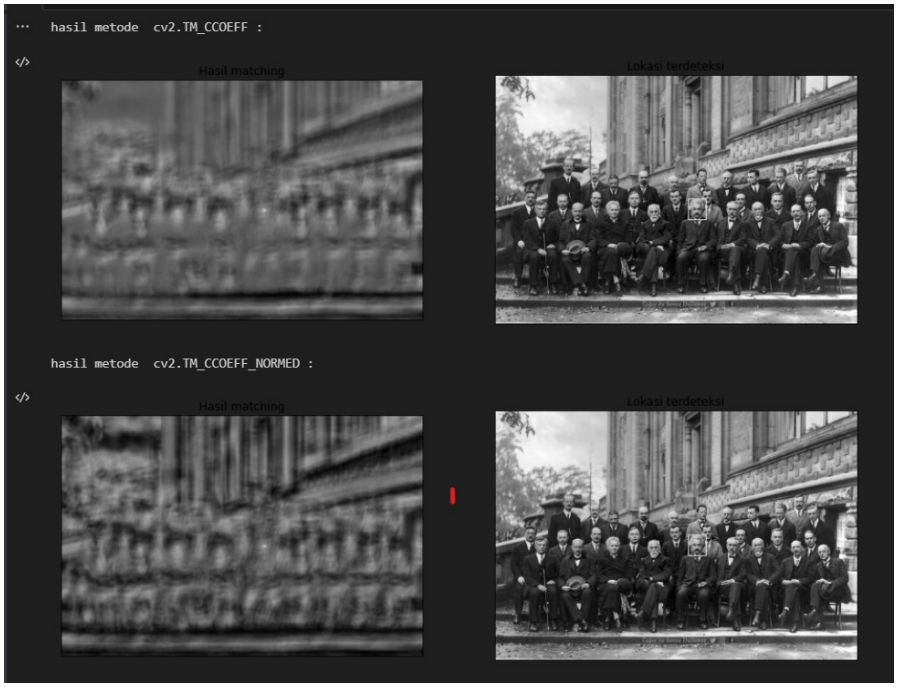


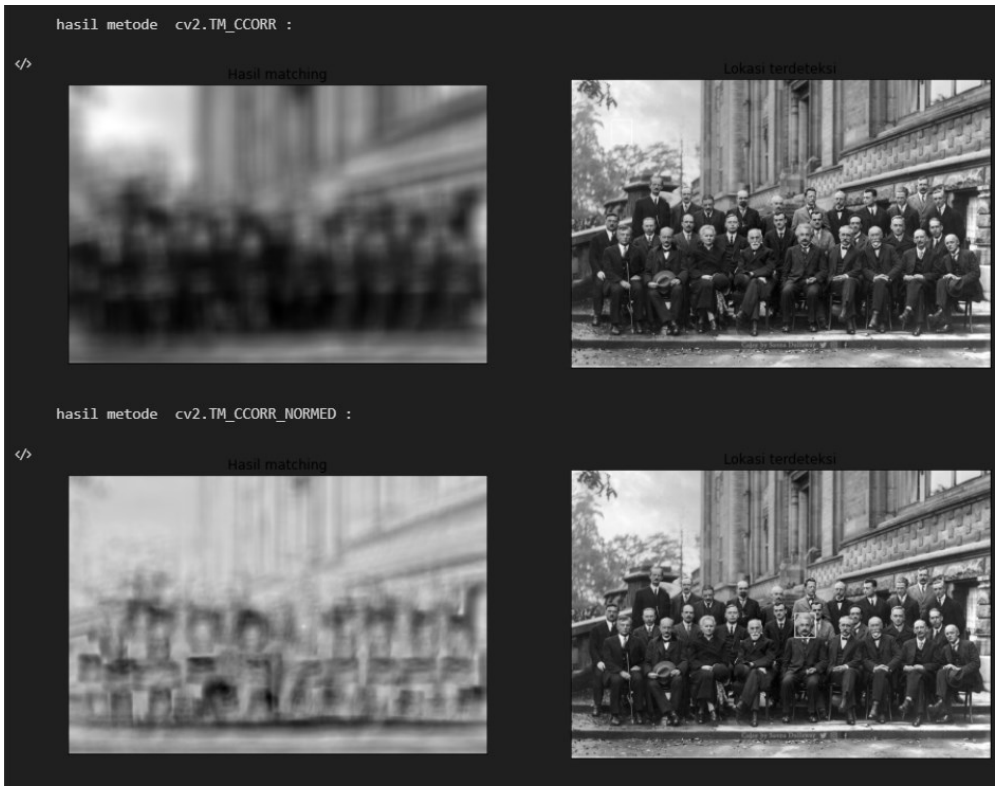

```
# tampilkan kedua gambar
from matplotlib import pyplot as plt

# panggil dan konversi warna agar sesuai dengan Matplotlib
einstein = cv2.imread('einstein.png')
einstein =  cv2.cvtColor(einstein, cv2.COLOR_BGR2RGB) # simpan dengan nama yang
sama = ditumpuk

# panggil dan konversi warna agar sesuai dengan Matplotlib
solvay = cv2.imread('solvayconference.jpg')
solvay =  cv2.cvtColor(solvay, cv2.COLOR_BGR2RGB)

plt.subplot(121),plt.imshow(einstein), plt.title('Einstein')
plt.subplot(122),plt.imshow(solvay), plt.title('Solvay Conference 1927')
plt.show()
```

Selanjutnya, lakukan Template Matching pada gambar Einstein dan Gambar Solvay Conference untuk mencari di mana Einstein duduk:

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

img = cv2.imread('solvayconference.jpg',0)
img2 = img.copy()
template = cv2.imread('einstein.png',0)
w, h = template.shape[::-1]

# All the 6 methods for comparison in a list
methods = ['cv2.TM_CCOEFF', 'cv2.TM_CCOEFF_NORMED', 'cv2.TM_CCORR',
            'cv2.TM_CCORR_NORMED', 'cv2.TM_SQDIFF', 'cv2.TM_SQDIFF_NORMED']

# perbesar ukuran hasil plotting
plt.rcParams["figure.figsize"] = (15,15)

for met in methods:
    img = img2.copy()
    method = eval(met)

    # menggunakan template matching
    res = cv2.matchTemplate(img,template,method)

    # mencari ukuran citra template untuk menggambar kotak
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

    # metode TM_SQDIFF dan TM_SQDIFF_NORMED menggunakan persamaan yang sedikit
berbeda
    # sehingga dibuatkan fungsi khusus untuk mengambil nilai minimum
    if method in [cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED]:
        top_left = min_loc
    else:
        top_left = max_loc
    bottom_right = (top_left[0] + w, top_left[1] + h)



    # buat persegi pada lokasi yang ditemukan
    cv2.rectangle(img, top_left, bottom_right, 255, 2) # 2 adalah ketebalan garis
kotak

    print("hasil metode", met, ": " )
    plt.subplot(121),plt.imshow(res,cmap = 'gray')
    plt.title('Hasil matching'), plt.xticks([]), plt.yticks([])
    plt.subplot(122),plt.imshow(img,cmap = 'gray')
```

```
    plt.title('Lokasi terdeteksi'), plt.xticks([]), plt.yticks([])


    plt.show()
```





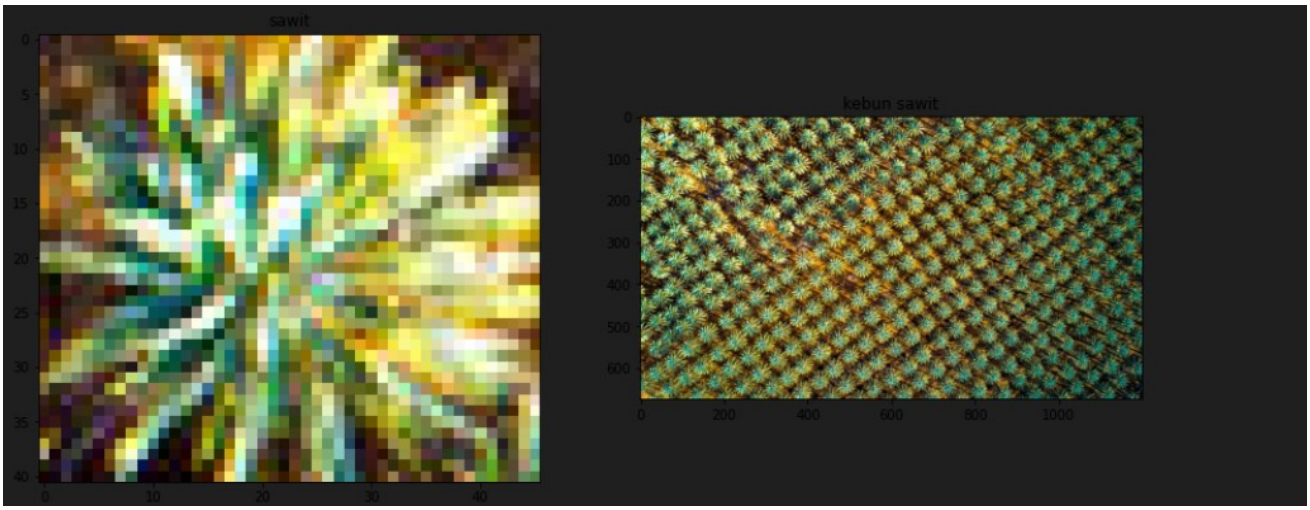
Pada contoh di atas, beberapa metode matching memberikan hasil yang lebih baik dibanding yang lain. Untuk tiap kondisi yang berbeda boleh jadi penggunaan parameter yang berbeda akan memberikan hasil yang lebih baik. Demikian pula, preprocessing (seperti melakukan deteksi tepi) biasanya akan membantu dalam proses matching seperti ini.

Pada bidang remote sensing, aplikasi dari Template Matching ini antara lain adalah untuk menghitung jumlah objek. pada bagian ini kita akan menggunakan teknik template matching berdasarkan deteksi keypoint untuk mendeteksi jumlah objek.

Latihan 7: Menghitung deteksi sawit dengan template matching

Kita akan mencoba melakukan deteksi jumlah pohon sawit dengan algoritma sederhana di atas. Silahkan kalian gunakan gambar masing-masing.

```
# tampilkan kedua gambar
from matplotlib import pyplot as plt

# panggil dan konversi warna agar sesuai dengan Matplotlib
sawit = cv2.imread('sawit.png')
sawit =  cv2.cvtColor(sawit, cv2.COLOR_BGR2RGB)

# panggil dan konversi warna agar sesuai dengan Matplotlib
kebun_sawit = cv2.imread('kelompoksawit.jpg')
kebun_sawit =  cv2.cvtColor(kebun_sawit, cv2.COLOR_BGR2RGB)

plt.subplot(121),plt.imshow(sawit), plt.title('sawit')
plt.subplot(122),plt.imshow(kebun_sawit), plt.title('kebun sawit')
plt.show()
```

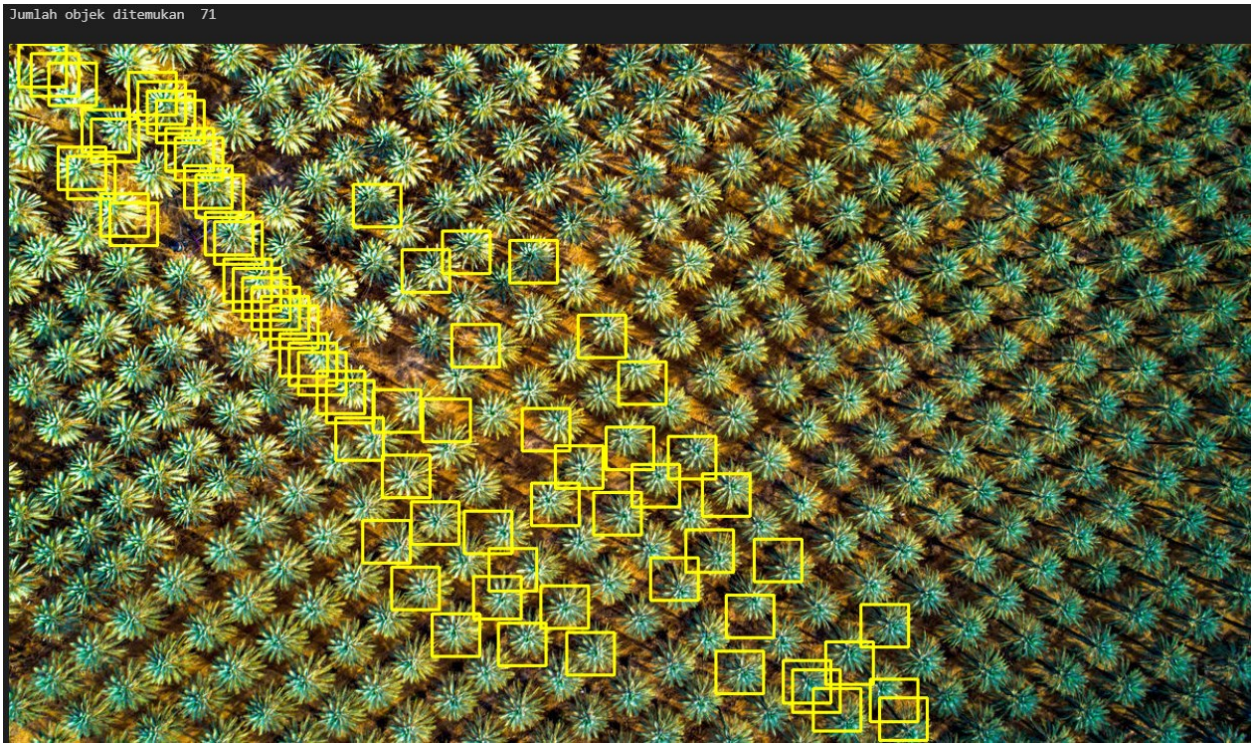


kita akan menggunakan template matching untuk menentukan jumlah pohon sawit berdasarkan template sawit seperti di atas. Parameter threshold di bawah menentukan seberapa mirip keypoint yang ingin kita deteksi berdasarkan template, sehingga kita dapat mengubah nilai tersebut untuk mendapatkan jumlah yang lebih optimal.

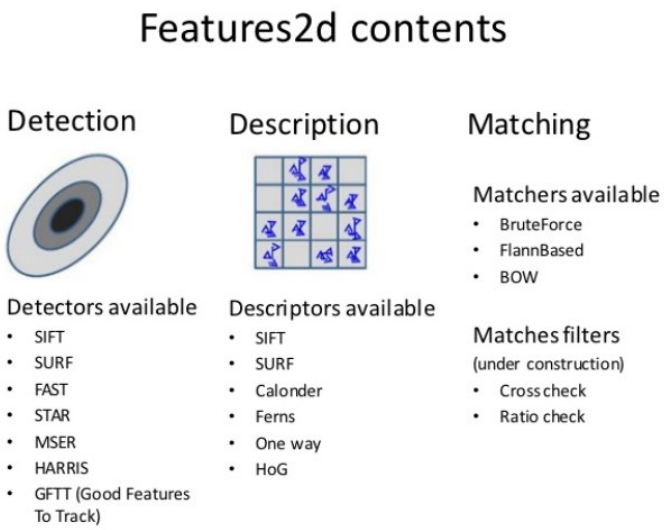
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

## membaca gambar utuh untuk dicari
img_rgb = cv2.imread('kelompoksawit.jpg')
img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)

## membaca template
template = cv2.imread('sawit.jpg',0)

## ukuran template. ukuran ini akan digunakan untuk menggambar kotak
w, h = template.shape[::-1]

# menggunakan metode COEFF-NORMALIZED
res = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED)

# Nilai threshold atau ambang batas deteksi kemiripan titik.
# Lakukan eksperimen dengan merubah nilai ini
threshold = 0.15
loc = np.where(res >= threshold)

## membuat array kosong untuk menyimpan lokasi-lokasi dari hasil deteksi
lspoint=[]
lspoint2=[]
count = 0  # untuk menyimpan jumlah matching yang ditemukan
for pt in zip(*loc[::-1]):
    ## jika sudah ada, skip lokasi tersebut
    if pt[0] not in lspoint and pt[1] not in lspoint2:
        ## gambar persegi warna kuning dengan ketebalan dua poin
        cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,255,255), 2)
        for i in range(((pt[0])-9), ((pt[0])+9),1):
            ## tambahkan koordinat x ke list
            lspoint.append(i)
        for k in range(((pt[1])-9), ((pt[1])+9),1):
            ## tambahkan koordinat y ke list
            lspoint2.append(k)
        count+=1 ### berapa jumlah matching yang ditemukan?
    else:
        continue
print ("Jumlah objek ditemukan ", count)
```

```
## tampilkan dengan imshow
cv2_imshow(img_rgb)
```

Hasil ketika dilakukan running:



Rangkuman algoritma yang ada pada open CV:



Materi ini merupakan dasar dari bagaimana operasi citra pada level rendah digunakan untuk berbagai keperluan lain yang banyak dijumpai pada bidang penginderaan jauh. Untuk lebih memahamkan mengenai materi pada minggu ini, kerjakan latihan-latihan praktek tersebut.

Tugas:

1. Kerjakan latihan diatas dan Gunakan gambar bebas untuk latihan 1-7 dan screenshot hasilnya dan tuliskan analisis perbandingannya.
2. Gunakan gambar bebas (bisa foto/cari di internet) dan lakukan deteksi corner seperti pada latihan 4 menggunakan metode Harris. Berapa jumlah corner yang berhasil dideteksi oleh metode tersebut?
3. Pada latihan 6 dan 7, Gunakan gambar bebas (misalnya wajah masing-masing , bisa dengan foto/selfie) untuk mendemokan fungsi Feature Detection and Template Matching pada OpenCV. Apa yang terjadi jika kita menggunakan foto yang lain? apakah program tetap dapat mendeteksi posisinya?

File yang dikumpulkan:

- Kode program ipynb/py
- File word berisi screenshoot, link github dan analisis perbedaan hasil dari pemahaman yg kalian tangkap dari latihan yang dilakukan.
- Format nama file: nim_nama
- Agar drive LMS tidak penuh disarankan diupload di drive lalu attach link saat pengumpulan tugas. Pastikan link dibagikan dapat diakses oleh semua orang.

Deadline kelas TSEB, TKK, TT : Selasa, 20 Juni 2023, Pukul 23:59 WIB