

Lab4 : Image Segmentation (Histogram of Oriented Gradients & K-Mean Clustering)

```
1 import cv2
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 from skimage.feature import hog
6 from skimage import measure
7 import glob
8
9 from sklearn.cluster import KMeans
10 from scipy import spatial
11
12 from tqdm import tqdm
13 import cv2
14 import os
15 import random
```


✓ 0.0s Python

Load Image

```
1 ### START CODE HERE ###
2 # image = cv2.imread("image.png")
3 image = cv2.imread("yannix.jpg")
4 # image = cv2.imread("gigachad.jpg")
5 # image = cv2.imread("meme.png")
6 image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
7
8 plt.imshow(image)
9 ### END CODE HERE ###
```

[5] ✓ 0.1s Python

<matplotlib.image.AxesImage at 0x140f13740>



- โหลดรูปตัวอย่างมา แล้วปรับ endian จาก BGR เป็น RGB

Histogram of Oriented Gradients

Blur the image then apply to the `hog()`

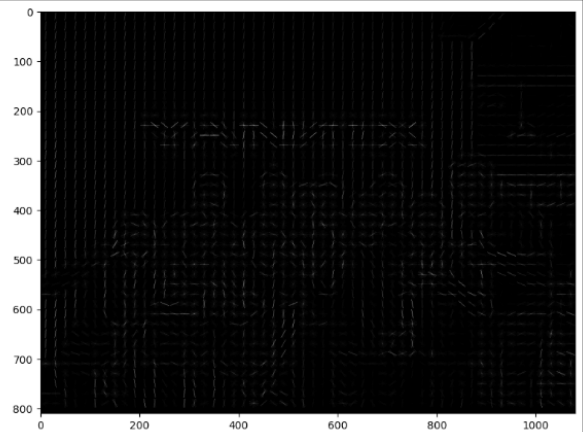
► Expected output

```
1  ### START CODE HERE ###
2  blurry_image = cv2.GaussianBlur(image, (11, 11),0)
3
4  features, hog_image = hog(blurry_image, orientations=9,
5                             pixels_per_cell=(20, 20),
6                             cells_per_block=(2, 2),
7                             visualize=True,
8                             channel_axis=-1)
9  ✨
10
11 # hog_image = exposure.rescale_intensity(hog_image, in_range=(0, 10))
12 fig = plt.figure(figsize=(20,20))
13 axs = fig.subplots(1,2)
14
15 axs[0].imshow(blurry_image)
16 axs[1].imshow(hog_image, cmap="gray")
17
18 # print(fd)
19 # print(len(fd))
20 print(hog_image.shape)
21
22 ### END CODE HERE ###
```

✓ 0.6s

Python

(810, 1080)



- นำรูปมาเบลอโดยใช้ฟังก์ชัน `cv2.GaussianBlur()` โดยเบลอแค่ (11,11) เพราะไม่ต้องการให้เบลอมากเกินไป
- นำรูปที่เบลอมาแล้ว มาเข้าฟังก์ชัน `hog` โดยใช้ `pixel per cell` เป็น 20,20 ก็คือ 20 pixel ในรูปต้นฉบับ จะเท่ากับ 1 cell ในรูป `hog` และเนื่องจากเรา input รูปเป็น RGB เราจึงต้องใส่ `axis=-1` ไปด้วย

```

1  ### START CODE HERE ###
2  class HOGSubimageExtractor:
3      def __init__(self, image, tile_size, stride):
4          self.image = image
5          self.tile_size = tile_size
6          self.stride = stride
7          self.h, self.w, _ = image.shape
8          self.hGrid = range(0, self.h - tile_size[0] + 1, stride)
9          self.wGrid = range(0, self.w - tile_size[1] + 1, stride)
10         self.hog_features = []
11         self.hog_images = []
12         self.extract_hog_features()
13
14     def extract_hog_features(self):
15         for i in self.hGrid:
16             for j in self.wGrid:
17                 tile = self.image[i : i + self.tile_size[0], j:j+self.tile_size[1]]
18                 features, hog_image = hog(tile, orientations=8,
19                                         pixels_per_cell=(25, 25),
20                                         cells_per_block=(1, 1),
21                                         visualize=True,
22                                         channel_axis=-1)
23                 self.hog_features.append(features)
24                 self.hog_images.append(hog_image)
25
26     def plot_hog_images(self):
27         num_tiles = len(self.hog_images)
28         grid_rows = len(self.hGrid)
29         grid_cols = len(self.wGrid)
30
31         fig, ax = plt.subplots(grid_rows, grid_cols, figsize=(grid_cols, grid_rows), facecolor='black')
32         fig.subplots_adjust(hspace=0.2, wspace=0.2)
33
34         for i in range(grid_rows):
35             for j in range(grid_cols):
36                 idx = i * grid_cols + j
37                 ax[i, j].imshow(self.hog_images[idx], cmap='gray')
38                 ax[i, j].axis('off')
39                 ax[i, j].set_xticklabels([])
40                 ax[i, j].set_yticklabels([])
41                 ax[i, j].set_aspect('equal')
42
43         plt.show()
44
45     def get_num_grid(self):
46         return len(self.hGrid), len(self.wGrid)
47
48     ### END CODE HERE ###
49
✓ 0.0s

```

- ฟังก์ชัน `extract_hog_features()` จะทำการลูปตามขวางกว้างกับความสูง โดยลูปตาม `step` ของ `stride` จากนั้นก็เอาแต่ละ `tile` มาเข้าฟังก์ชัน `hog` แล้วก็ `append hog images` เก็บไว้ใส่ใน `list`
- ฟังก์ชัน `plot_hog_images()` เอาไว้สำหรับพล็อต โดยจะใช้ `subplot` และสามารถแบ่งแถวและหลังได้ โดยใช้จำนวนของ `grid` ในแกนตั้งละแกนนอน จากนั้นจะลูปตามจำนวน `grid` ตามแนวนอนและตั้งไปเรื่อยๆ และจะเข้าถึงรูปภาพที่เก็บไว้ผ่าน `index` (โดยที่ `index` มีค่าเป็น `grid_row_iterator * จำนวนคอลัมน์ทั้งหมด + grid_cols_iterator`)

```

1  ### START CODE HERE ###
2  tile_size = (64, 64)
3  stride = 20
4  hog_extractor = HOGSubimageExtractor(blurry_image, tile_size, stride)
5  num_grid = hog_extractor.get_num_grid()
6  print(f'Number of grids: {num_grid}')
7  hog_extractor.plot_hog_images()
8  ### END CODE HERE ###
9

```

✓ 15.8s

- โค้ดแสดงการใช้ instant ข้างต้น โดยมี tile_size เป็น 64 x 64 pixel และให้ขยับไปที่ละ 20 pixel
- ได้ผลลัพธ์ดังรูปด้านล่าง

