

Họ và tên: Nguyễn Trương Xuân NghiêM

MSSV: 2151010246

Bài 1: Detect Pedestrians from image step by step

Source Code:

```
1  from skimage.feature import hog
2  from sklearn.svm import LinearSVC
3  from sklearn.model_selection import train_test_split
4  from matplotlib import pyplot as plt
5  from sklearn import metrics
6  from imutils.object_detection import non_max_suppression
7  from skimage import color
8  from skimage.transform import pyramid_gaussian
9
10 import joblib
11 import glob
12 import os
13 import cv2
14 import numpy as np
15
16
17 X = []
18 Y = []
19
20 pos_im_path = "D:/Computer Vision/OpenCV-Practice/Lab5/positive"
21 neg_im_path = "D:/Computer Vision/OpenCV-Practice/Lab5/negative"
22
23 # Load positive features
24 for filename in glob.glob(os.path.join(pos_im_path, "*.png")):
25     fd = cv2.imread(filename, 0)
26     fd = cv2.resize(fd, (64, 128))
27     fd = hog(
28         fd,
29         orientations=9,
30         pixels_per_cell=(8, 8),
31         visualize=False,
32         cells_per_block=(2, 2),
33     )
34     X.append(fd)
35     Y.append(1)
```

```

36
37 # Load negative features
38 for filename in glob.glob(os.path.join(neg_im_path, "*.png")):
39     fd = cv2.imread(filename, 0)
40     fd = cv2.resize(fd, (64, 128))
41     fd = hog(
42         fd,
43         orientations=9,
44         pixels_per_cell=(8, 8),
45         visualize=False,
46         cells_per_block=(2, 2),
47     )
48     X.append(fd)
49     Y.append(0)
50
51
52 X = np.float32(X)
53 Y = np.array(Y)
54
55 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
56
57 print("Train data: ", len(X_train))
58 print("Train Labels (1, 0) ", len(y_train))
59
60 model = LinearSVC()
61 model.fit(X_train, y_train)
62
63 # predict
64 y_pred = model.predict(X_test)
65
66 # confusion matrix and accuracy
67 print(
68     f"Classification report for classifier {model}:\n"
69     f"{metrics.classification_report(y_test, y_pred)}\n"
70 )
71
72 joblib.dump(model, "models.dat")
73 print("Model saved : {}".format("models.dat"))

```

```

75 # pedestrian detection
76 modelFile = "models.dat"
77 inputFile = "input.png"
78 outputFile = "output.png"
79 image = cv2.imread(inputFile)
80 image = cv2.resize(image, (400, 256))
81 size = (64, 128)
82 step_size = (9, 9)
83 downscale = 1.05
84
85 # List to store the detections
86 detections = []
87 scale = 0
88 model = joblib.load(modelFile)
89
90
91 def sliding_window(image, window_size, step_size):
92     for y in range(0, image.shape[0], step_size[1]):
93         for x in range(0, image.shape[1], step_size[0]):
94             yield (x, y, image[y: y + window_size[1], x: x + window_size[0]])
95
96
97 for im_scaled in pyramid_gaussian(image, downscale=downscale):
98     if im_scaled.shape[0] < size[1] or im_scaled.shape[1] < size[0]:
99         break
100     for x, y, window in sliding_window(im_scaled, size, step_size):
101         if window.shape[0] != size[1] or window.shape[1] != size[0]:
102             continue
103         window = color.rgb2gray(window)
104         fd = hog(
105             window,
106             orientations=9,
107             pixels_per_cell=(8, 8),
108             visualize=False,
109             cells_per_block=(2, 2),
110         )
111
112         fd = fd.reshape(1, -1)
113         pred = model.predict(fd)

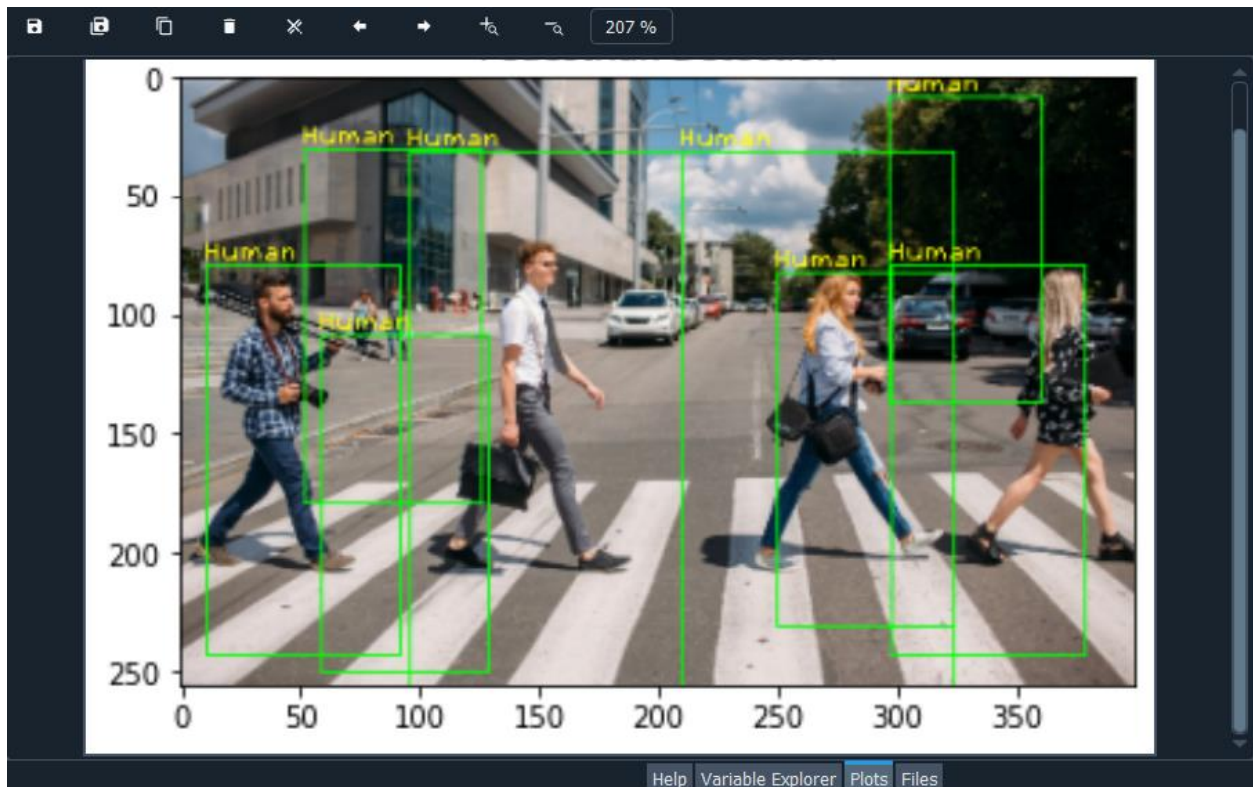
```

```

114         if pred == 1:
115             if model.decision_function(fd) > 0.5:
116                 detections.append(
117                     (
118                         int(x * (downscale**scale)),
119                         int(y * (downscale**scale)),
120                         model.decision_function(fd),
121                         int(size[0] * (downscale**scale)),
122                         int(size[1] * (downscale**scale)),
123                     )
124                 )
125         scale += 1
126
127     clone = image.copy()
128     clone = cv2.cvtColor(clone, cv2.COLOR_BGR2RGB)
129     rects = np.array([[x, y, x + w, y + h] for (x, y, _, w, h) in detections])
130     sc = [score[0] for (x, y, score, w, h) in detections]
131     print("sc: ", sc)
132     sc = np.array(sc)
133     pick = non_max_suppression(rects, probs=sc, overlapThresh=0.5)
134     for x1, y1, x2, y2 in pick:
135         cv2.rectangle(clone, (x1, y1), (x2, y2), (0, 255, 0))
136         cv2.putText(clone, "Human", (x1 - 2, y1 - 2), 1, 0.75, (255, 255, 0), 1)
137
138     cv2.imwrite(outputFile, clone)
139
140     plt.imshow(clone)
141     plt.title("Pedestrian Detection")
142
143     plt.show()
144

```

Result:



Bài 2: Detect Pedestrian in Video

Source Code:

```
Ex_1_Pedestrian_Recognition_StepByStep.py X Ex_2_Pedestrian_Recognition_UsingCV2.py X
<None> <None>
1 import numpy as np
2 import cv2
3
4 hog = cv2.HOGDescriptor()
5 hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
6 cv2.startWindowThread()
7
8 cap = cv2.VideoCapture("video.mp4")
9 while True:
10     ret, frame = cap.read()
11     frame = cv2.resize(frame, (640, 480))
12     gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
13
14     boxes, weights = hog.detectMultiScale(frame, winStride=(8, 8))
15     boxes = np.array([[x, y, x + w, y + h] for (x, y, w, h) in boxes])
16
17     for xA, yA, xB, yB in boxes:
18         # display the detected boxes in the colour picture
19         cv2.rectangle(frame, (xA, yA), (xB, yB), (0, 255, 0), 2)
20
21     cv2.imshow("Pedestrian Detection", frame)
22     if cv2.waitKey(1) & 0xFF == ord("q"):
23         break
24
25 cap.release()
26 cv2.destroyAllWindows()
27
```

Result:

