Experiment No: 8

Name:Pratham Ingawale

Roll No:25

Aim: To Detecting and Recognizing Objects

Objective: Object Detection and recognition techniques HOG descriptor The Scale issues The location issue Non-maximum (or non-maxima) suppression vector machine people detection

Theory:

Object detection and recognition Techniques

This experiment involves using computer vision techniques to identify and categorize objects within images or video frames, making it applicable in various fields, such as surveillance and autonomous systems.

HOG descriptors

Histogram of Oriented Gradients (HOG) is a feature descriptor used for object detection. It quantifies the local gradient information in an image, which can be essential for recognizing objects with different textures and shapes.

The scale issue

Objects may appear at different scales within images, so addressing the scale issue is crucial for accurate detection and recognition. Techniques must be employed to account for size variations.

The Location issue

Identifying the precise location of an object within an image is another challenge. This involves determining the object's position and orientation accurately.

Non-maximum(or Non-maxima)Suppression

After object candidates are identified, it's necessary to eliminate duplicate or overlapping detections. Non-maximum suppression is a technique used to select the most confident and non-overlapping object candidates.

Support vector machines

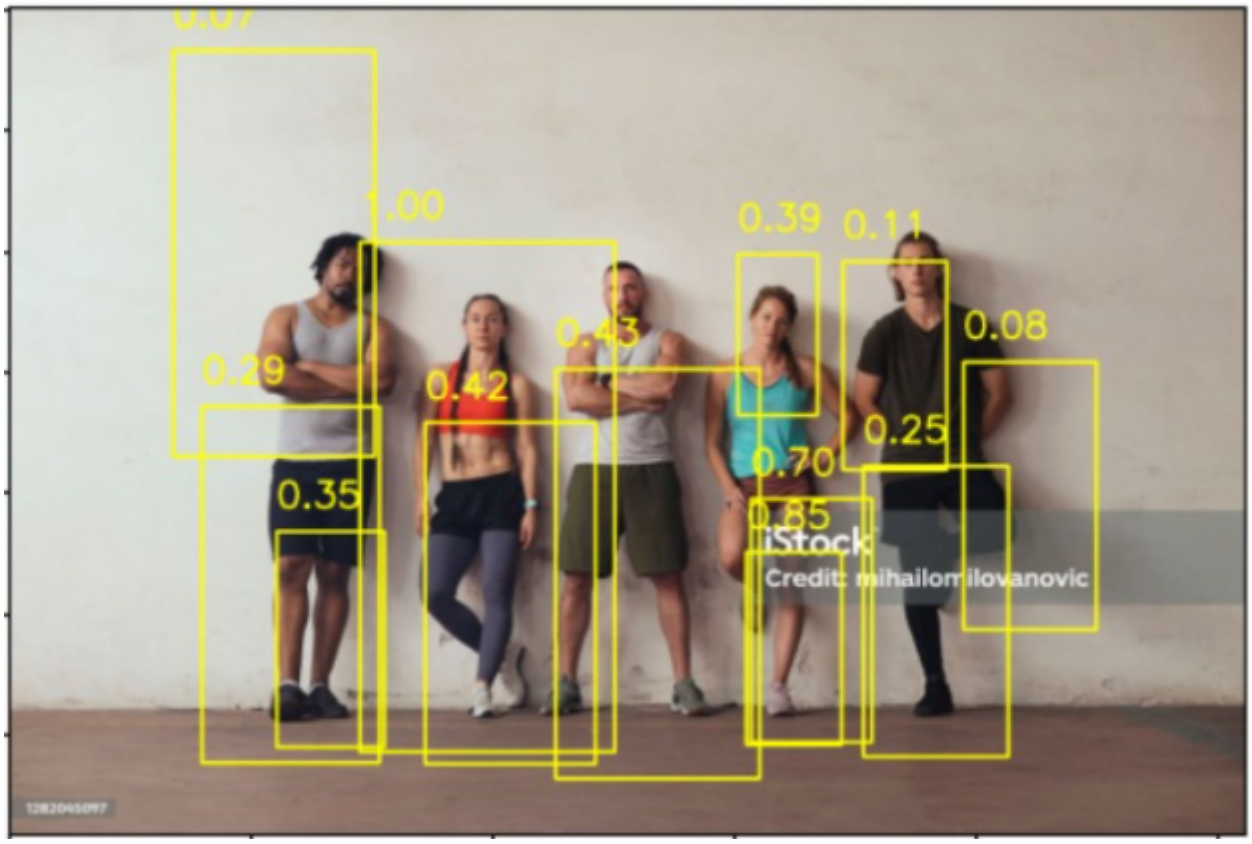
SVMs are machine learning models employed in object detection to classify objects based on extracted features. They are used for various tasks, including people detection, by learning to distinguish between positive (object) and negative (non-object) examples.

Code

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

# Check OpenCV version
OPENCV_MAJOR_VERSION = int(cv2._version_.split('.')[0])
OPENCV_MINOR_VERSION = int(cv2._version_.split('.')[1])

def is_inside(i, o):
    ix, iy, iw, ih = i
    ox, oy, ow, oh = o
    return ix > ox and ix + iw < ox + ow and iy > oy and iy + ih < oy + oh

# Load the image from your local file system in Colab
img = cv2.imread('p2.jpg')

hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

if OPENCV_MAJOR_VERSION >= 5 or (OPENCV_MAJOR_VERSION == 4 and OPENCV_MINOR_VERSION >=
6):
    # OpenCV 4.6 or a later version is being used.
    found_rects, found_weights = hog.detectMultiScale(
        img, winStride=(4, 4), scale=1.02, groupThreshold=1.9)
else:
    # OpenCV 4.5 or an earlier version is being used.
    # The groupThreshold parameter used to be named finalThreshold.
    found_rects, found_weights = hog.detectMultiScale(
```

```
        img, winStride=(4, 4), scale=1.02, finalThreshold=1.9)

found_rects_filtered = []
found_weights_filtered = []

for ri, r in enumerate(found_rects):
    for qi, q in enumerate(found_rects):
        if ri != qi and is_inside(r, q):
            break
    else:
        found_rects_filtered.append(r)
        found_weights_filtered.append(found_weights[ri])

for ri, r in enumerate(found_rects_filtered):
    x, y, w, h = r
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 255), 2)
    text = '%.2f' % found_weights_filtered[ri]
    cv2.putText(img, text, (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)

# Display the image
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
#plt.title('people in desert')
plt.show()

# Save the image
cv2.imwrite('/content/p2.png', img)
```

Output


1.00
0.39
0.11
0.08
0.43
0.29
0.42
0.25
0.70
0.35
0.85
iStock
Credit: mihailomilovanovic