Computer Vision
Degree in Information Technology
2º Semester 2021/2022

Worksheet 7

Goals: Real-time Pedestrian Detection using Python & OpenCV

Pedestrian detection or people detection is a very essential task in some areas such as surveillance systems, traffic control systems, etc. In this machine learning project, we are going to make a very simple pedestrian detection system using OpenCV.

Steps to develop pedestrian detection:

1. Import necessary packages and define the model
2. Reading images/frames and pre-process
3. Detect Pedestrians
4. Draw rectangles around each detection in the frame
5. Post-process output data to filter out the best results
6. Detect real-time from Video

Step 1 - Import necessary packages and define the model

- Install and import opencv
- Install and import imutils: Imutils is basically a helper package that consists of a series of convenience functions to make basic image processing functions such as rotation, resize, etc. Please run below command to install the package: `pip install imutils`
- To extract necessary data from the image we'll use the `cv2.HOGDescriptor()` method and set the descriptor object as HOGCV.
- Using `setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())` method we initialise hog detection object with svm detection for detecting people.

```
import cv2
import numpy as np
import imutils
# Histogram of Oriented Gradients Detector
HOGCV = cv2.HOGDescriptor()
HOGCV.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDet
ector())
```

Step 2 – Reading the image and pre-process

- Download the image pedestrian.jpg
- cv2.imread() function reads images from a given path.
- If image width is greater than defined max_width then apply imutils.resize() function to resize the image to defined max_width.

```
frame = cv2.imread("pedestrian.jpg")
width = frame.shape[1]
max_width = 600
if width > max_width:
  frame = imutils.resize(frame, width=max_width)
```

Step 3 - Detect Pedestrians

```
pedestrians, weights = HOGCV.detectMultiScale(frame,
winStride=(4, 4), padding=(8, 8), scale=1.03)
pedestrians = np.array([[x, y, x + w, y + h] for (x, y, w, h) in
pedestrians])
```

- detectMultiScale() detects the objects from the image and returns their x and y coordinate, height, and width.
- The winStride parameter indicates the step size in both the x and y location of the sliding window.
- The padding parameter indicates the number of pixels in both the x and y direction in which the sliding window ROI is “padded” prior to HOG feature extraction.
- The scale parameter controls the factor in which our image is resized at each layer of the image pyramid.
- Analyse the following image to understand the previous steps

- In the next line, we've processed the data to get the final rectangle coordinates.

Output:

```
[[ 92 36 186 223]
 [508 18 600 220]
 [ 16 25 116 224]
 [266 62 353 235]
 [329 15 409 175]]
```

Here we get 5 nested lists. That means 5 people are detected.

Step 4 – Draw rectangles around each pedestrian detection in the frame

Now we have all the detected pedestrian information. So let's draw the bounding boxes.

- Count parameter tracks how many objects are detected
- Using the cv2.rectangle function we draw a bounding box around detected object
- cv2.putText function draws a text string in the frame

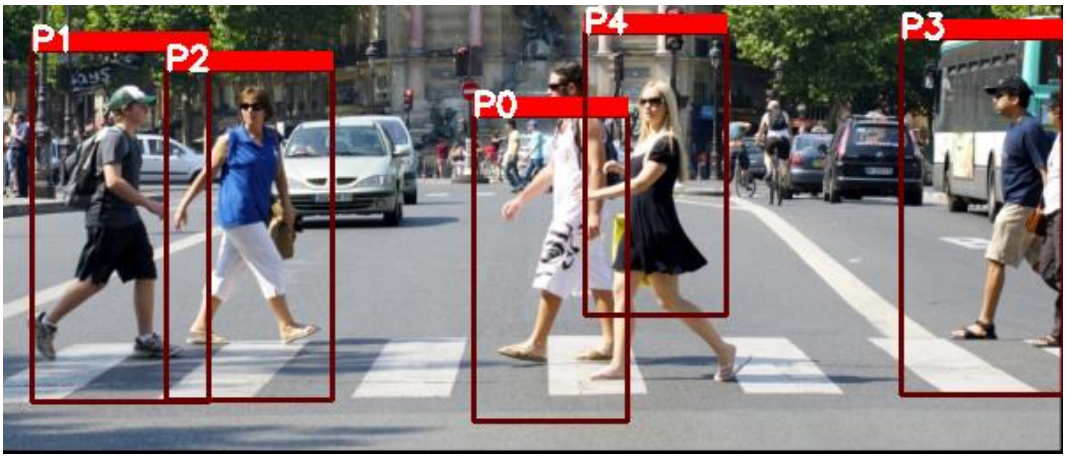
```
count = 0
#  Draw bounding box over detected pedestrians
for x, y, w, h in pedestrians:
    cv2.rectangle(frame, (x, y), (w, h), (0, 0, 100), 2)
    cv2.rectangle(frame, (x, y - 20), (w,y), (0, 0, 255), -1)
    cv2.putText(frame, f'P{count}', (x, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
    count += 1
```

```
cv2.imshow("output", frame)
cv2.waitKey(0)
```

Step 5 - Post-process output data to filter out the best results

In some cases, the bounding boxes may overlap each other. To remove those overlapped boxes then we'll have better output detection. So here we can apply Non-Maxima Suppression (NMS) and suppress bounding boxes that overlap with a significant threshold. The imutils library includes the NMS function.

```
from imutils.object_detection import non_max_suppression
# apply non-maxima suppression to remove overlapped bounding
boxes
pedestrians = non_max_suppression(pedestrians, probs=None,
overlapThresh=0.5)
```



Step 6 – Pedestrian detection in video

Download the video and apply the same steps to detect pedestrians in real-time.

Bom trabalho!