Assignment 12 [Face Detection using HOG]

Q) Perform Face Detection using HOG.

```
=> !pip install opencv-contrib-python
!pip install dlib
```

```
Requirement already satisfied: opencv-contrib-python in
/usr/local/lib/python3.10/dist-packages (4.8.0.76)
Requirement already satisfied: numpy>=1.21.2 in
/usr/local/lib/python3.10/dist-packages (from opencv-contrib-python)
(1.25.2)
Requirement already satisfied: dlib in
/usr/local/lib/python3.10/dist-packages (19.24.4)
```

```
def convert_and_trim_bb(image, rect):
  # extract the starting and ending (x, y)-coordinates of the
  # bounding box
  startX = rect.left()
  startY = rect.top()
  endX = rect.right()
  endY = rect.bottom()
  # ensure the bounding box coordinates fall within the spatial
  # dimensions of the image
  startX = max(0, startX)
  startY = max(0, startY)
  endX = min(endX, image.shape[1])
  endY = min(endY, image.shape[0])
  # compute the width and height of the bounding box
  w = endX - startX
  h = endY - startY
  # return our bounding box coordinates
  return (startX, startY, w, h)
```

```
def process_boxes(box):
    xmin = box.left()
    ymin = box.top()
    xmax = box.right()
    ymax = box.bottom()
    return [int(xmin), int(ymin), int(xmax), int(ymax)]
```

```
import dlib
```

```python
import cv2
from google.colab.patches import cv2_imshow

#step1: read the image
image = cv2.imread("/content/drive/MyDrive/CVIP LAB/people.jpg")

#step2: converts to gray image
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#step3: get HOG face detector and faces
hogFaceDetector = dlib.get_frontal_face_detector()
faces = hogFaceDetector(gray, 1)

#step4: loop through each face and draw a rect around it
for (i, rect) in enumerate(faces):
    x = rect.left()
    y = rect.top()
    w = rect.right() - x
    h = rect.bottom() - y
    #draw a rectangle
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

#step5: display the resulted image
cv2_imshow(image)
```

⇒

Theory

Gradient Calculation:

- HOG starts by calculating the gradient of pixel intensities in the image. This is typically done using gradient operators like the Sobel operator.

Orientation Binning:

- The image is divided into small cells, usually 8x8 pixels each. Within each cell, the gradient magnitudes and orientations are computed.
- The gradient orientations are quantized into a fixed number of bins (usually 9 bins covering 0 to 180 degrees).

Histogram Calculation:

- A histogram of gradient orientations is computed for each cell. Each gradient contributes to the histogram according to its orientation, with its magnitude as the weight.
- This histogram represents the distribution of gradient orientations within the cell.

Normalization:

- To make the descriptor robust to changes in lighting and contrast, the histogram values are normalized. This can be done either within each cell or within a block of cells.

Descriptor Formation:

- Finally, the normalized histograms from all cells are concatenated to form the HOG descriptor for the entire image. This descriptor captures the spatial distribution of gradient orientations in the image.

Detection:

- The HOG descriptor is then typically used with a machine learning algorithm, such as a support vector machine (SVM), to classify whether a given region of the image contains the object of interest (e.g., a face).

Conclusion

- **Pros:**
  - HOG features are robust to changes in lighting and contrast since they are based on local gradient information.
  - HOG descriptors provide a compact representation of the spatial distribution of gradient orientations, making them suitable for object detection tasks.
- **Cons:**
  - HOG features may not capture finer details of objects, especially in cluttered or highly variable scenes.
  - They can be computationally expensive to compute, especially when used with large images or a high number of cells.