Task 2-Histogram Of Oriented Gradients

Matriculation Number: 100004430

Date Of submission: 09-07-2025

Introduction:

Image histogram is a type of histogram which reflects the intercity distribution of an image plotting the number of pixels for each intensity values. The number of each intensity value is called a frequency. The histogram shows the number of pixels for every intensity value, ranging from 0 to 255. Each of these 256 values is called a bin in histogram terminology. There are two types of histogram process i.e. 1. AHE (Adaptive Histogram Equalization) – low contrast to high contrast 2. CLAHE (Contrast Limited Adaptive Histogram Equalization) – Use clipping value to increase the impurity.

Let's discuss about the HOG Feature descriptor, after that we can conclude about HOG.
What is feature descriptor?
A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information.
Typically, a feature descriptor converts an image of size width x height x 3 (channels) to a feature vector / array of length n. In the case of the HOG feature descriptor, the input image is of size 64 x 128 x 3 and the output feature vector is of length 3780.

In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and we know that edges and corners pack in a lot more information about object shape than flat regions.

Methodology and steps:

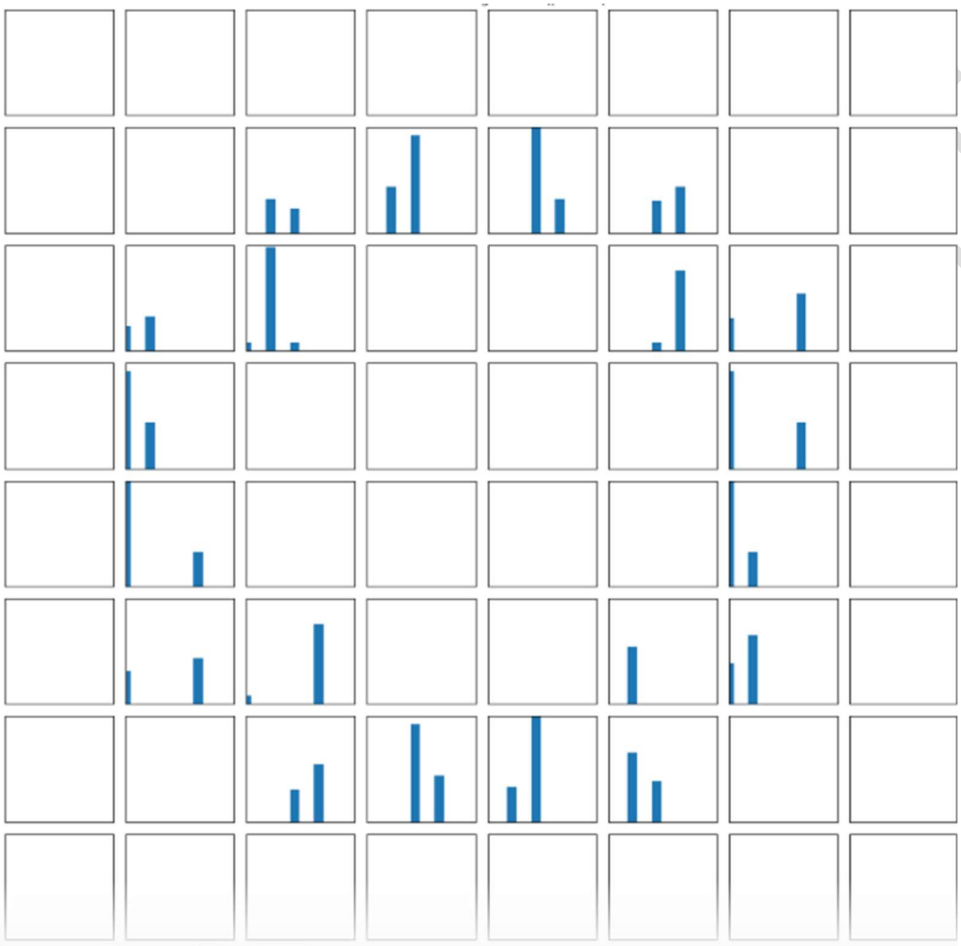
- **Gradient computation**: Calculates the gradient magnitude and direction at each pixel.
- **Orientation binning**: Divides the image into small cells and bins the gradient directions.
- **Block normalization**: Groups cells into blocks and normalizes the histograms to improve invariance to illumination.
- **Feature vector**: Concatenates all normalized histograms into a single feature vector.

Gradient computation:

For a grayscale image I(x,y)$I(x,y)$, the gradient at each pixel is computed in two directions:
Horizontal gradient (Gx):
Gx=I(x+1,y)−I(x−1,y)$Gx = I(x+1,y) - I(x-1,y)$
Vertical gradient (Gy):
Gy=I(x,y+1)−I(x,y−1)$Gy = I(x,y+1) - I(x,y-1)$
These are approximated using **convolution kernels**, like the **Sobel operator**:

Once you have GxGx and GyGy, you compute:
Gradient Magnitude:
M=Gx2+Gy2$M=Gx2+Gy2$
Gradient Orientation (Angle):
θ=tan−1(GyGx)$\theta=\tan-1(GxGy)$

Orientation Binding:



The image was divided into **cells of size 8×8 pixels**.
For each cell, we computed a **histogram of gradient directions** (using 9 bins from 0° to 180°).
Each small plot represents the **distribution of edge directions** within that cell.
This is a key step in HOG feature extraction, where local edge patterns are captured and later normalized across blocks for robustness.\

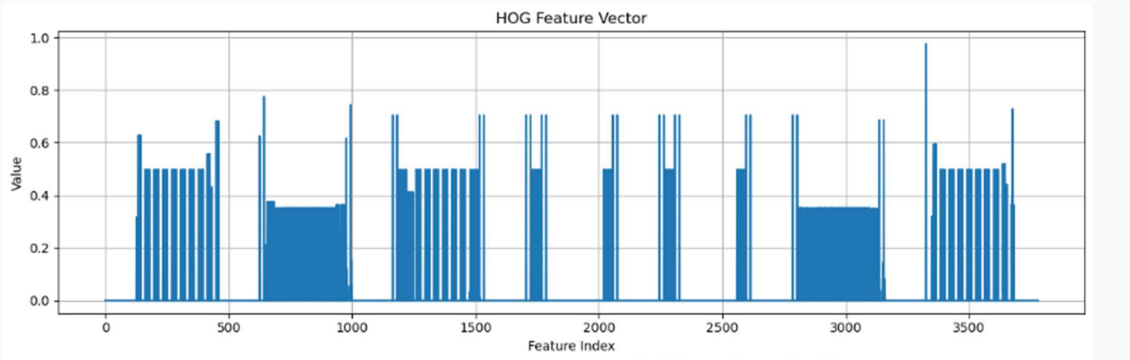
Block Normalization:

A **block** is a group of adjacent cells (typically 2×2).Each cell has a histogram of gradient orientations (9 bins). These histograms are **flattened into a single vector** and then **normalized** using L2 norm-

$$\text{normalized\_vector} = \frac{\text{vector}}{\sqrt{\sum \text{vector}^2 + \epsilon}}$$

This normalization improves **robustness to lighting and contrast changes**.

Feature Vector:



This is the histogram representation for a 3780 feature vector. The feature vector has **3780 elements**, which is typical for a 64×128 image patch using standard HOG settings.Each value represents the strength of a particular gradient orientation in a specific block.This vector is used as input to machine learning models like **SVMs** for tasks such as **object detection** or **image classification**.

Sobel Operator:

The **Sobel operator** is a popular edge detection technique used in image processing and computer vision. It helps identify regions in an image where intensity changes sharply—these are typically edges or boundaries of objects.

What Does the Sobel Operator Do?
It calculates the **gradient** of image intensity at each pixel, which tells us:
How strong the change in brightness is (magnitude).
In which direction the change occurs (orientation).

How It Works
The Sobel operator uses **convolution kernels** to approximate derivatives:
One kernel detects **horizontal edges** (changes along the x-axis).
Another detects **vertical edges** (changes along the y-axis).

Why Use Sobel Instead of Simple Derivatives?

- It **smooths** the image slightly (reduces noise) while computing the gradient.
- It's more robust than basic derivative filters like Prewitt or Roberts.

Applications

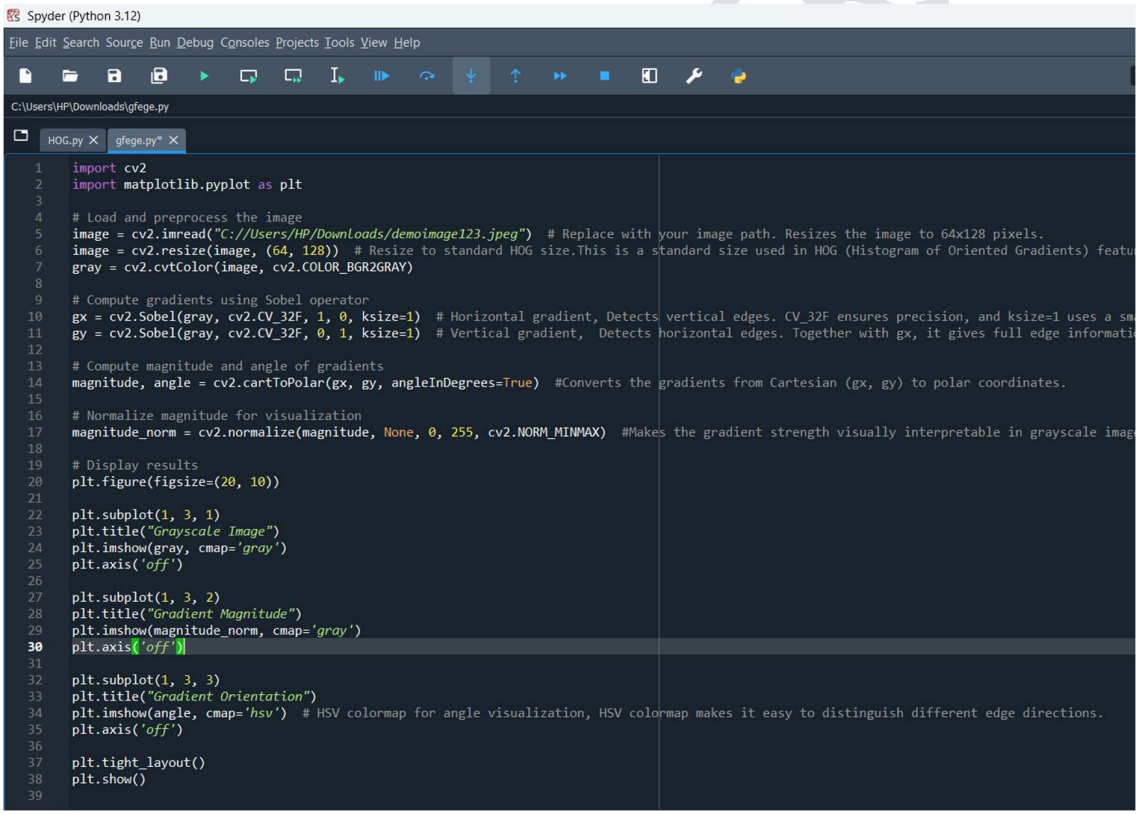
- Edge detection

- Feature extraction (e.g., for object recognition)
- Image segmentation
- Motion detection

Implementation:

- **Platform**: Python
- **IDE**: Spyder
- **Libraries**: Opencv, Matplotlib
- **Input**: Normal jpg image
- **Output**: Image containing grey scale, Gradient magnitude, Gradient orientation image

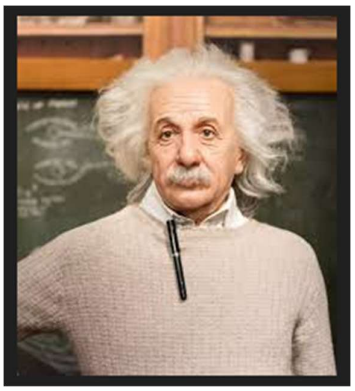
Code:



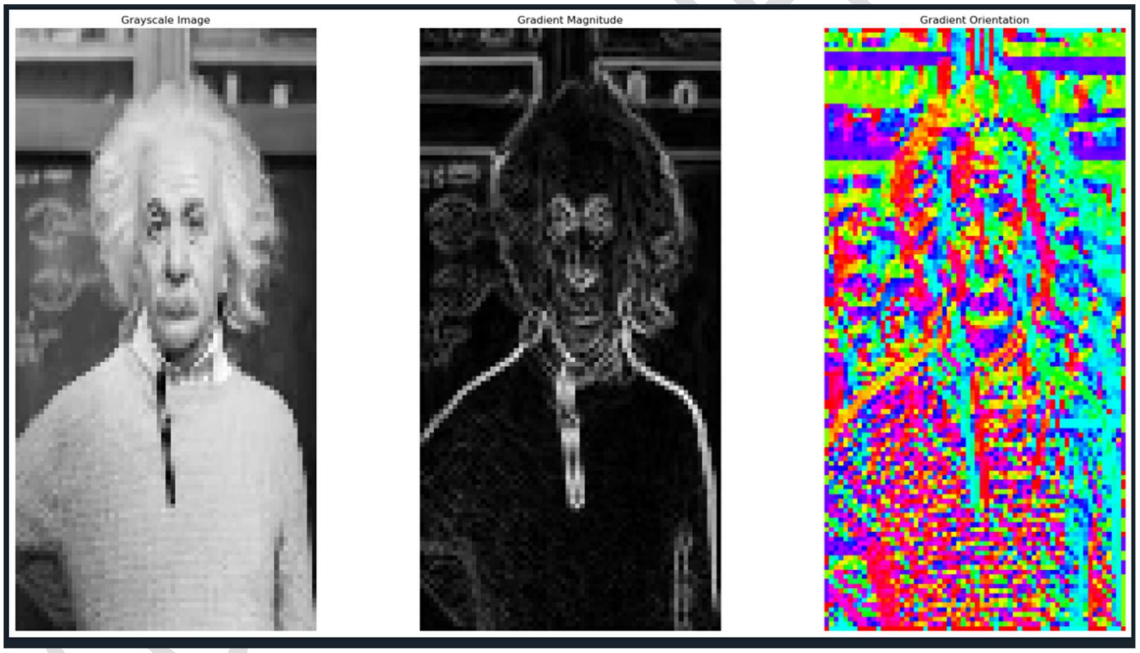
Code File:


gfege.py

Actual Image:



Resultant Image:



HOG Usage and its implementation:

Python script uses OpenCV and Matplotlib to perform **pedestrian detection** in an image using the **Histogram of Oriented Gradients (HOG)** descriptor.
This code is useful for:

- **Surveillance systems**: Detecting people in security footage.
- **Autonomous vehicles**: Identifying pedestrians for safety.
- **Smart cities**: Monitoring pedestrian traffic.
- **Robotics**: Helping robots navigate environments with humans.

Code snippet:

Spyder (Python 3.12)

File Edit Search Source Run Debug Consoles Projects Tools View Help

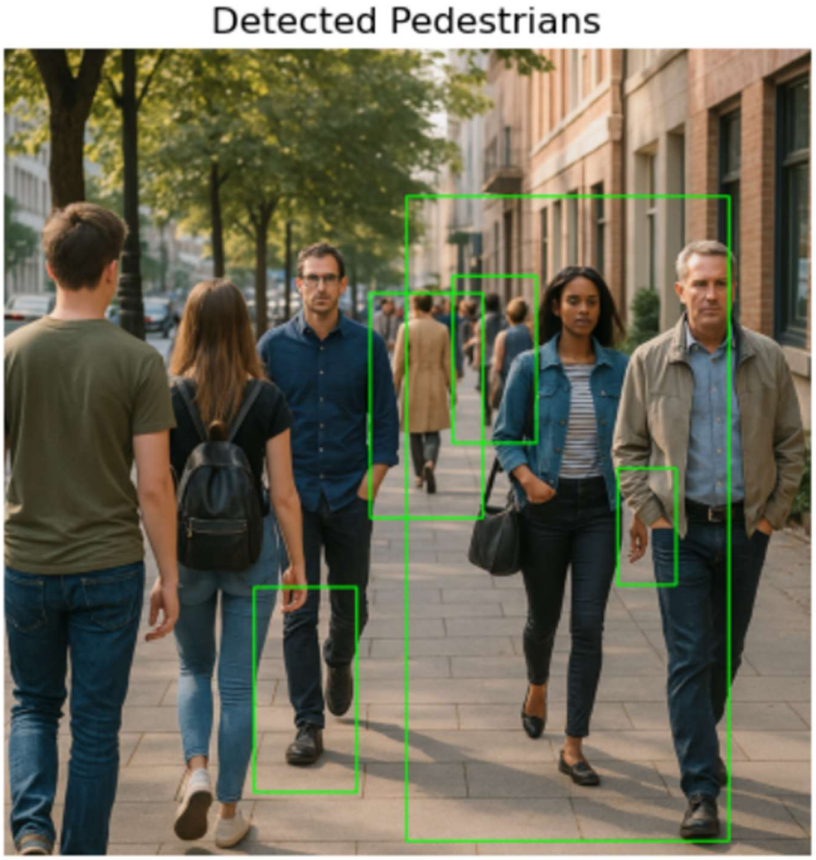
C:\Users\HP\Downloads\HOG.py

HOG.py* gfege.py*

```python
import cv2
import matplotlib.pyplot as plt

# Load image
image = cv2.imread("C://Users/HP/Downloads/bvjegjgerj.jpeg")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Initialize HOG descriptor
hog = cv2.HOGDescriptor()    #Creates a HOG descriptor object. HOG is used to extract features based on edge directions and gradients.

# Compute HOG features
hog_features = hog.compute(gray)    #Computes the HOG feature vector for the grayscale image.
                                    #Prints the shape of the feature vector (useful for understanding the dimensionality of the data).

# Print feature vector shape
print("HOG feature vector shape:", hog_features.shape)

# Optional: Use HOG for pedestrian detection
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())    #Sets a pre-trained SVM detector for pedestrian detection.
                                                                    #detectMultiScale scans the image at multiple scales to detect people.
                                                                    #winStride=(8,8) controls the step size of the sliding window.
boxes, weights = hog.detectMultiScale(gray, winStride=(8,8))

# Draw bounding boxes
for (x, y, w, h) in boxes:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)       #Draws green rectangles around detected pedestrians.

# Show result
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Detected Pedestrians")
plt.axis('off')
plt.show()
```

Actual Image:

After running the code:



Code file:

HOG.py

Conclusion:

Conclusion on HOG (Histogram of Oriented Gradients)
The Histogram of Oriented Gradients (HOG) is a powerful feature descriptor used primarily for object detection, especially pedestrian detection. Here's a concise conclusion:

Key Takeaways:
Gradient-Based: HOG captures edge and gradient structures that are characteristic of local shapes in an image.
Robust to Illumination: By focusing on gradient orientation rather than intensity, HOG is less sensitive to lighting changes.
Effective for Detection: When combined with a classifier like SVM, HOG is highly effective for detecting humans and other objects.
Widely Used: It has been a foundational technique in computer vision, especially before deep learning became dominant.

Reference Used:

AI Platform: Copilot
Lecture notes
Research Paper: https://learnopencv.com/histogram-of-oriented-gradients/
GitHub Repo: https://github.com/spmallick/learnopencv/