# Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY

(An Autonomous Institute, Affiliated to Visvesvaraya Technological University, Belagavi, Accredited by NAAC, with 'A' Grade)
Near Jnana Bharathi Campus, Mallathahalli, Bangalore – 560056

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**Group Activity Report on**

## "Pedestrian Detection using HOGs & OpenCV"

**Submitted in partial fulfillment of**

## Machine Learning Course
## Course Code: 18CS62
## No. of Credits: 4 = 4: 0: 0 (L: T: P)

**Submitted by**



| Student's Name | USN |
|---|---|
| Puneeth S | 1DA19CS199 |
| Nitin K | 1DA19CS202 |
| Abdur Rahman | 1DA18CS002 |
| Advith Rao | 1DA18CS008 |



**For the Academic Year 2021 – 22**

# Abstract

Pedestrian accidents still represent the second largest source of traffic related injuries and fatalities after accidents involving passenger cars.Pedestrian detection is a key problem in computer vision, with several applications that have the potential to positively impact quality of life. In recent years, many pedestrian classification approaches have been proposed. The pedestrian classification consists of two stages: feature extraction and feature classification. Recently several robust feature extracting methods have been proposed in literature like Scale Invariant Feature Transform (SIFT) , Histogram of Gradients (HOG) , Co-occurrence of Histogram of Gradients (CoHOG) . Also several classifiers exists like Hidden Markov Model (HMM), Support Vector Machines (SVM), and Neural Network. In this paper, we examine the two feature extraction method and we use neural network as classifier instead of SVM. An extensive evaluation and comparison of these methods are presented. The advantages and shortcomings of the underlying design mechanisms in these methods are discussed and analyzed through analytical evaluation and empirical evaluation.

# Table Of Contents

# Chapter 1 Introduction

Pedestrian detection or in simpler terms person detection in streets and footpaths is an essential part of some very prominent tasks. Tasks such as intelligent video surveillance, traffic control systems, and the mighty AI in self-autonomous vehicles that are completely self-driving or just obstacle avoidance and automatic braking systems.

Robots are also capable of using these computer vision techniques for handling the way they interact with the environment.

There are several methods for tackling this problem: using digital cameras, using infrared or heat sensing sensors, TOF or time of flight sensor, etc. All of these work by collecting data about the surroundings and then predict or estimate where a person's location is with respect to the sensor. The way that data collection takes place by different sensors is what differentiates the methods.

There have been notable rise of interest in human detection, object detection in general because of its use in several fields.

Out of all the available methods, we are gonna use images / videos from a camera for our task of pedestrian detection, because these are abundant everywhere and mostly available for our use case. Like there are security cameras on streets, cars are being equipped with parking cameras and cameras all around them nowadays. It is a fairly easy system to build, cost-effective, and with satisfactory speed and accuracy.

In this project, we are going to use a pre-trained machine learning model for detecting people / pedestrian from a video feed, using either a webcam or video file. The pre-trained model that we are going to use for this task is the yolov4-tiny. We are not going to train a model from scratch as obviously we will not be able to match the speed and accuracy of a sota pre-trained model, which is very important in such tasks. Also, the yolo models are available freely for all kinds of tasks including academic, research, and commercial use. So we need not worry about any violation.

## 1.1 Problem Introduction

I will be working on the problem of pedestrian detection, and will use the deep learning model to set a threshold in real time so that if the number of pedestrians increases in an area, the output will be "violation" and if the no. of people will be less than the threshold then it will be "not-violation" . So, according to my problem statement, I have to make a model and run it on a video footage and Image and give the no. of pedestrians detected in each frame. Points to note here is, if there is a footage of pedestrian walking there will only be change in number of pedestrians after a certain number of frames and in those particular frames I need my accuracy to be as precise as possible.

Car Safety is a major concern of most car shoppers, and accidents involving pedestrians are on the rise. Detection is an advanced safety system which is designed to alert a driver to a pedestrian in the path of (or near the path of) a vehicle.

## 1.2 Solution using Machine Learning

Pedestrian detection is an essential component of image and video surveillance system, on account of its provision of information with respect to the semantic aspects of understanding video footage. Its potential applications can be best stated in the field of automotive engineering, as well as improvement of safety systems. A plenitude of Car manufacturers offer this functionality as a part of their Advanced Driver Assistance System. In this work, the dataset, obtained from the MIT People' s data, involves a set of images, that are then bifurcated into the obvious training set and testing set. Prior to the the division, appropriate image processing is performed in order to get the images to the requisite size and format. The training and testing set are then tested against the Machine Learning algorithms and the Histogram of Oriented Gradients (HOG) using Support Vector Machines. The results obtained are then compared with the responses of Pedestrian detection using Non Maximum Suppression Algorithm (NMS) in order to estimate how accurate the adopted pedestrian detection approach is.

# Chapter 2 Software Requirement Specification

## 2.1 Hardware Requirements

- Processor: Intel Core i5
- RAM: 4GB or above
- Hard Disk: 512 GB
- Input device: Keyboard, Mouse

## 2.2 Software Requirements

- Operating system: 64bit Windows Operating system, X64-based processor
- Python 3.4 or above.
- IDE: VS Code.

## 2.3 Functional Requirements

- OpenCV python 3.4.2
- Imutils 0.5.3

## 2.4 Imported Python Libraries

- **Opencv** is used to read frames from our video file or our webcam feed, resize and reshape it according to the model requirement. It also provides the dnn module which we will use to work with our deep neural network model. Also draw the bounding boxes on the image and show it back to the user.
- **Imutils** is another great library for performing different actions on images. It acts as a helper library providing some very useful functions to opencv which already is a vast library.

## 2.5 Performance Requirements

- pre-trained **HOG(Histogram of Oriented Gradients) + Linear SVM model**

# Chapter 3 About Dataset

We tested our detector on two different data sets. The first is the well-established MIT pedestrian database , containing 509 training and 200 test images of pedestrians in city scenes (plus left-right reflections of these). It contains only front or back views with a relatively limited range of poses. Our best detectors give essentially perfect results on this data set, so we produced a new and significantly more challenging data set, 'INRIA' , containing 1805 64×128 images of humans cropped from a varied set of personal photos. Fig. 2 shows some samples. The people are usually standing, but appear in any orientation and against a wide variety of background image including crowds. Many are bystanders taken from the image backgrounds,so there is no particular bias on their pose. The database is available from http://lear.inrialpes.fr/data for research purposes.

Figure 2. Some sample images from our new human detection database. The subjects are always upright, but with some partial occlusions and a wide range of variations in pose, appearance, clothing, illumination and background.

OpenCV has a built-in method to detect pedestrians. It has a pre-trained HOG(Histogram of Oriented Gradients) + Linear SVM model to detect pedestrians in images and video streams.

# Chapter 4 Algorithm

**Case 1 : when input is a image.**

I. Initialize the HOG Person Detector
II. Reading the Image
III. Resizing the image
IV. Detecting all regions in the image that has a pedestrians inside it
V. Drawing the regions in the image
VI. Showing the output Image

**Case 2 : when input is a video.**

I. Initialize the HOG Person Detector
II. Reading the Video Stream
III. Detecting all regions in the image that has a pedestrians inside it
IV. Drawing the regions in the image
V. Showing the output Image

# Chapter 5 Work Flow

This section gives an overview of our feature extraction chain, which is summarized in fig. 1. Implementation details are postponed until 6. The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. Similar features have seen increasing use over the past decade [4,5,12,15]. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. In practice this is implemented by dividing the image window into small spatial regions ( "cells" ), for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries form the representation. For better invariance to illumination, shadowing, etc., it is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram "energy" over somewhat larger spatial regions ( "blocks" ) and using the results to normalize all of the cells in the block. We will refer to the normalized descriptor blocks as Histogram of Oriented Gradient (HOG) descriptors. Tiling the detection window with a dense (in fact, overlapping) grid of HOG descriptors and using the combined feature vector in a conventional SVM based window classifier gives our human detection chain (see fig. 1). The use of orientation histograms has many precursors [13,4,5], but it only reached maturity when combined with local spatial histogramming and normalization in Lowes Scale Invariant Feature Transformation (SIFT) approach to wide baseline image matching [12], in which it provides the underlying image patch descriptor for matching scaleinvariant keypoints. SIFT-style approaches perform remarkably well in this application [12,14]. The Shape Context work [1] studied alternative cell and block shapes, albeit initially using only edge pixel counts without the orientation histogramming that makes the representation so effective. The success of these sparse feature based representations has somewhat overshadowed the power and simplicity of HOG' s as dense image descriptors. We hope that our study will help to rectify this. In particular, our informal experiments suggest that even the best current keypoint based approaches are likely to have false

positive rates at least 1–2 orders of magnitude higher than our dense grid approach for human detection, mainly because none of the keypoint detectors that we are aware of detect human body structures reliably. The HOG/SIFT representation has several advantages. It captures edge or gradient structure that is very characteristic of local shape, and it does so in a local representation with an easily controllable degree of invariance to local geometric and photometric transformations: translations or rotations make little difference if they are much smaller that the local spatial or orientation bin size. For human detection, rather coarse spatial sampling, fine orientation sampling and strong local photometric normalization turns out to be the best strategy, presumably because it permits limbs and body segments to change appearance and move from side to side quite a lot provided that they maintain a roughly upright orientation.

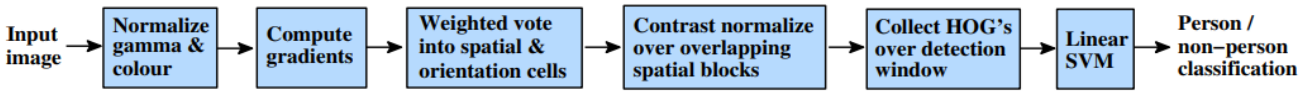



*Figure 1. An overview of our feature extraction and object detection chain. The detector window is tiled with a grid of overlapping blocks in which Histogram of Oriented Gradient feature vectors are extracted. The combined vectors are fed to a linear SVM for object/non-object classification. The detection window is scanned across the image at all positions and scales, and conventional non-maximum suppression is run on the output pyramid to detect object instances, but this paper concentrates on the feature extraction process.*

# Chapter 6 Coding and Implementation

Lets make the program to detect pedestrians in an Image:

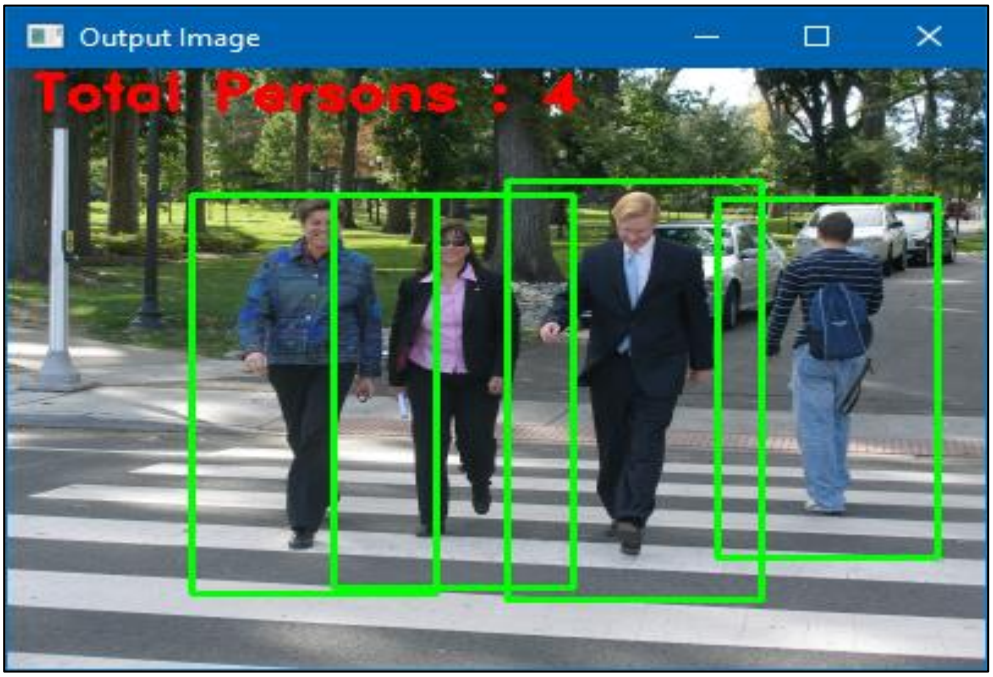
```
import cv2
import imutils

# Initializing the HOG person
# detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# Reading the Image
image = cv2.imread('img.png')

# Resizing the Image
image = imutils.resize(image,
                       width=min(400, image.shape[1]))

# Detecting all the regions in the
# Image that has a pedestrians inside it
(regions, _) = hog.detectMultiScale(image,winStride=(4, 4),padding=(4, 4),
                                    scale=1.05)

# Drawing the regions in the Image
for (x, y, w, h) in regions:
    cv2.rectangle(image, (x, y),
                  (x + w, y + h),
                  (0, 0, 255), 2)

# Showing the output Image
cv2.imshow("Image", image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Lets make the program to detect pedestrians in a video:

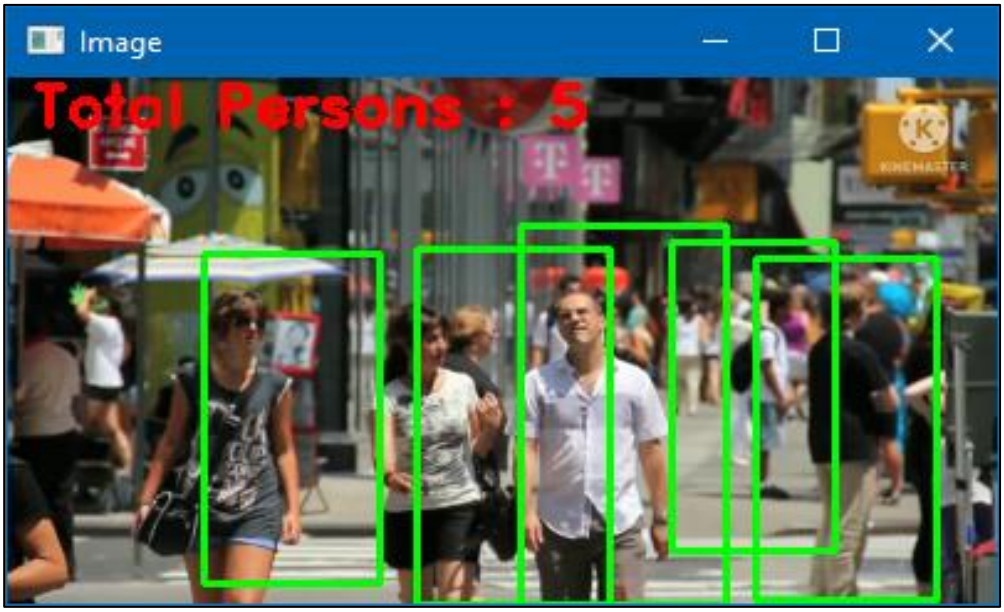
```
import cv2
import imutils

# Initializing the HOG person
# detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

cap = cv2.VideoCapture('vid.mp4')

while cap.isOpened():
      # Reading the video stream
      ret, image = cap.read()
      if ret:
            image = imutils.resize(image,
                                   width=min(400, image.shape[1]))

            # Detecting all the regions
            # in the Image that has a
            # pedestrians inside it
            (regions, _) = hog.detectMultiScale(image,winStride=(4, 4),
                                          padding=(4, 4),scale=1.05)

            # Drawing the regions in the
            # Image
            for (x, y, w, h) in regions:
                  cv2.rectangle(image, (x, y),(x + w, y + h),(0, 0, 255), 2)

            # Showing the output Image
            cv2.imshow("Image", image)
            if cv2.waitKey(25) & 0xFF == ord('q'):
                  break
      else:
            break

cap.release()
cv2.destroyAllWindows()
```

## 6.1 Sample output

### 1.When image is input




### 2.When video is used as input



# Conclusion

This report presented an approach towards pedestrian detection and object detection in general. It has been shown that by using Histograms of Oriented Gradient (HOG) features in combination with a Support Vector Machine, strong pedestrian detectors can be obtained. Through this project, we' ve learned about how a basic object detector works and how to apply it in real-time.

# References


- *www.geeksforgeeks.org*
- *https://data-flair.training*
- *http://lear.inrialpes.fr*
- *Tom M. Mitchell, Machine Learning, India Edition 2013, McGraw Hill Education.*