

**HUMAN DETECTION AND COUNTING**

**A PROJECT REPORT**

***Submitted by***

|  |  |
| --- | --- |
| **GANDHAVALLI CHAITHANYA SRI**  **RAJALAKSHMI R**  **TUMMALA SRIVANI** | **113320104032**  **113320104075**  **113320104099** |

***in partial fulfillment for the award of the degree***

***of***

# BACHELOR OF ENGINEERING

***in***

# COMPUTER SCIENCE AND ENGINEERING

# VELAMMAL INSTITUTE OF TECHNOLOGY,

**CHENNAI 601 204**

**ANNA UNIVERSITY: CHENNAI 600 025**

# MAY 2023

**BONAFIDE CERTIFICATE**

Certified that this project report **“HUMAN DETECTION AND COUNTING”** is the bonafide work of the **GANDHAVALLI CHAITHANYA SRI -113320104032**, **RAJALAKSHMI R** -**113320104075**, **TUMMALA SRIVANI**-**113320104099**, who carried out the project work under my supervision.

# SIGNATURE

**Dr.V.P.GLADIS PUSHPARATHI,**

**HEAD OF THE DEPARTMENT**

Computer Science and Engineering,

Velammal Institute of Technology

Velammal Gardens, Panchetti

Chennai-601 204

# SIGNATURE

**MS.S. SHANTHASHEELA**

**ASSISTANT PROFESSOR**

Computer Science and Engineering,

Velammal Institute of Technology

Velammal Gardens, Panchetti

Chennai-601 204

**HUMAN DETECTION AND COUNTING**

**VIVA-VOCE EXAMINATION**

The viva-voce examination of this project work was done as a part on the Bachelor’s Degree in Computer Science and Engineering held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

|  |  |
| --- | --- |
| **GANDHAVALLI CHAITHANYA SRI**  **RAJALAKSHMI R**  **TUMMALA SRIVANI** | **113320104032**  **113320104075**  **113320104099** |

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We are personally indebted to many who had helped us during the course of this project work. Our deepest gratitude to the God Almighty.

We are grateful and profoundly thankful to our beloved Chairman **Thiru.M.V.Muthuramalingam** for facilitating us with this opportunity.Our sincere thanks to our respected Director **Thiru. M.V.M Sasi Kumar** who took keen interest on us.

We are also thankful to our Advisors **Prof. K. Razak, Shri.M.Vasu** , our Principal

**Dr.N. Balaji,** VicePrincipal **Dr.S.Soundararajan** for never-ending encouragement which accelerates us towards innovation.

We are extremely thankful to our Head of the Department **Dr.V.P.Gladis Pushparathi,** Project Coordinator Mr**.V. Manickavasagan** and **Ms. Pranamita Nanda** Computer Science and Engineering for their valuable teachings and suggestions.

From the bottom of our heart, we would like to thank our guide **Ms.S.Shanthasheela**, who has been the pillar of this project without whom we would not have been able to complete the project successfully.

The Acknowledgment would be incomplete if we would not mention word of thanks to our Parents, Teaching and Non-Teaching Staffs, Administrative Staffs and Friends who had motivated and lend their support throughout the project. Thank you one and all.

**ABSTRACT**

Population growth is rising in modern times. Due to the population's linear increase, many individuals now frequent public spaces. Thus, this technique will give the number of people in a specific area's malls, supermarkets, etc. Some businesses exclusively depend on the timing and schedule of their customers. Therefore, by creating a system for counting people, our work satisfies the problem and offers a solution. Therefore, a mobile single shot detector (SSD) network and centroid tracker are presented. For better feature extraction, this model swaps out the VGG16 base network for a Mobile Net Work, and for classification, it links with six convolutional layers following the base network. In order to calculate the center of a bounding box, the centroid tracking algorithm uses bounding box coordinates from an object detector SSD. Each person will receive an ID when the centroid is calculated, and this operates the dataset with training and testing the data. To address potential challenges, the project incorporates techniques to handle occlusions, scale variations, and complex backgrounds. Additionally, measures are taken to ensure real-time performance by optimizing the model and leveraging hardware acceleration techniques, such as GPU parallelization. The project's success will be evaluated based on various metrics, including accuracy, precision, recall, and processing speed. Extensive testing will be conducted in diverse scenarios to assess the system's robustness and generalization capabilities. Performance comparisons will be made against existing human detection methods to validate the project's contribution.The outcomes of this project have wide-ranging implications, including enhanced public safety, improved crowd management at events, and better monitoring of social distancing compliance.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** | **5** |
|  | **LIST OF FIGURES** | **8** |
|  | **LIST OF ABBREVIATIONS** | **9** |
|  | **LIST OF DIAGRAMS** | **10** |
| **1** | **INTRODUCTION** | **12** |
|  | 1.1. INTRODUCTION | 12 |
|  | 1.2. OBJECTIVE | 13 |
|  | 1.3. MOTIVATION | 13 |
|  | 1.4. OVERVIEW OF THE PROJECT | 14 |
|  | 1.5. CHAPTER WISE SUMMARY | 14 |
| **2** | **ANALYSIS AND DESIGN** | **15** |
|  | 2.1. REQUIREMENTS | 15 |
|  | 2.2 ARCHITECTURE | 16 |
|  | 2.3 USECASE DIAGRAM | 17 |
|  | 2.4 SEQUENCE DIAGRAM | 18 |
| **3** | **IMPLEMENTATION** | **19** |
|  | 3.1 MODULE DESCRIPTION | 19 |
|  | 3.2 IMPLEMENTATION DETAILS | 21 |
|  | 3.3 CODE | 24 |
|  | 3.4 TOOLS USED | 28 |
|  | 3.5 SNAPSHOT | 28 |
| **4** | **TEST RESULTS/EXPERIMENTS/VERIFICATION** | 30 |
|  | 4.1 TESTING | 30 |
|  | 4.2 RESULTS AND VERIFICATION | 31 |
| **5** | **CONCLUSION AND FUTURE SCOPE** | 32 |
|  | **REFERENCE** | 34 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE NAME** | **PAGE NO** |
| 2.1 | ARCHITECTURE | 16 |
| 2.2 | USE CASE DIAGRAM | 17 |
| 2.3 | SEQUENCE DIAGRAM | 18 |

**LIST OF ABBREVIATIONS**

**AI:** Artificial Intelligence

**CV:** Computer Vision

**FPS:** Frames per Second

**HOG:** Histogram of Oriented Gradients

**IoU:** Intersection over Union

**ML:** Machine Learning

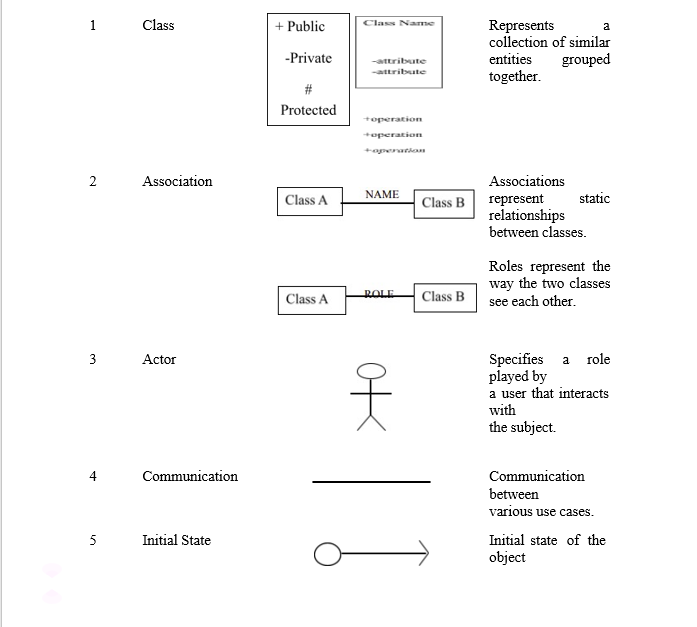
**OpenCV:** Open-Source Computer Vision Library

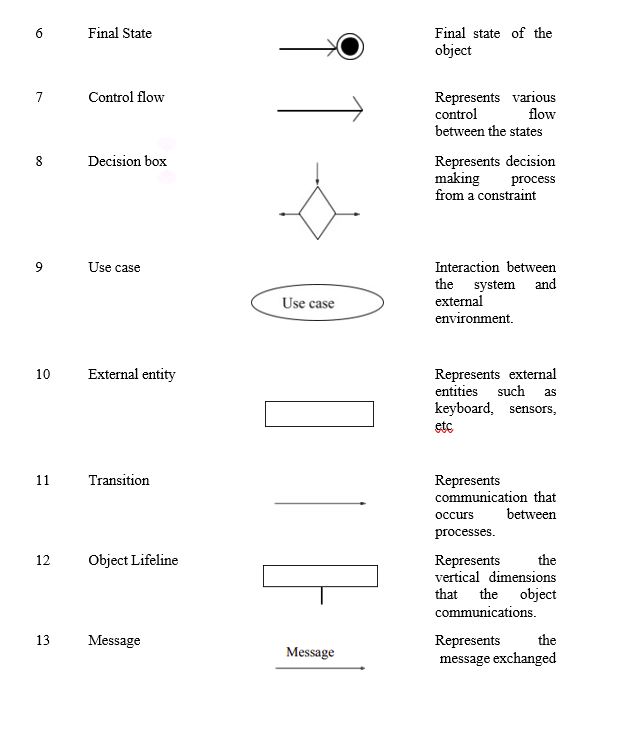
**PIL:** Python Imaging Library

**ROI:** Region of Interest

**SVM:** Support Vector Machine

**LIST OF DIAGRAMS**





**CHAPTER 1**

# INTRODUCTION

* 1. **INTRODUCTION**

This chapter resembles the brief introduction about the most widely used field of study “Computer Vision”. Here talked about the various aspects and uses of computer vision, basic meaning and keywords like detection, enumeration, and discussed the roadmap to the report.

**Computer Vision:**

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. It is most widely used field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs. Different types of computer vision include image segmentation, object detection, facial recognition, edge detection, pattern detection, image classification, and feature matching. Computer Vision itself is a big domain and is divided into various subdomains like scene reconstruction, object detection, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, visual serving, 3D scene modeling, and image restoration.

**Human Detection:**

Human detection is the task of locating all instances of human beings present in an image, and it has been most widely accomplished by searching all locations in the image, at all possible scales, and comparing a small area at each location with known templates or patterns of people. In this we can use various predefined methods and can detect the human in any image, video and can even get various factors like accuracy, each detections counting, etc.

Some common methods are

* Using Haar Cascade Classifier:

Here we make use of .xml file for human detection, and using that we detect the humans in real time videos and images.

* Using HOG (Histogram of Oriented Gradients):

Here we make used of predefined functions and with that we detect, and this case gives somehow better accuracy as compared to Harr Cascade Classifier.

* Using TensorFlow:

TensorFlow is an open-source API from Google, which is widely used for solving machine learning tasks that involve Deep Neural Networks.

And again, this method gives even better accuracy than above two methods.

Here we have implemented the application using the Second method and got almost the better accuracy

* 1. **OBJECTIVES**

The main problem around us we can see is the ‘population problem’. And directly or indirectly this is causing chaos or traffic in any one or the other place. This so-called chaos may sometimes even lead to quarrel or even suffocation sometimes.

In solution to our problem, we will build an AI and Machine Learning based model. This model will help us to easily get the population (no. of persons) or traffic criteria in any real time video cam of any place. Also, we will get the idea of which area is more or less populated or trafficked as compared to normal traffic.

In this model will allow user to upload any real time video and model will show us the traffic condition of the area in video.This model will also give suggestion for future about what should be the normal traffic in that area. Thus, like wise using this model one can take on control of traffic and which will prevent various chaos and accidents. And will also help to regulate traffic flow in that area.

**1.3 MOTIVATION**

The main problem around us we can see is the ‘population problem’. And directly or indirectly this is causing chaos or traffic in any one or the other place. This so-called chaos may sometimes even lead to quarrel or even suffocation sometimes. And directly or indirectly this is causing chaos or traffic in any one or the other place. This so-called chaos may sometimes even lead to quarrel or even suffocation sometimes. And directly or indirectly this is causing chaos or traffic in any one or the other place. This so-called chaos may sometimes even lead to quarrel or even suffocation sometimes. To prevent such chaos, we need to control the traffic and crowds. Using this model one can take on control of traffic and which will prevent various chaos and accidents.

**1.4 OVERVIEW OF THE PROJECT**

Real-time human detection and counting is a vast, challenging and important field, we are going to build the Human Detection and Counting System with python through images provided. This is an intermediate level Deep Learning project on Computer Vision, which will help you to master the concepts and make you an expert in the field of Data Science

**Human Detection with Computer Vision**

The project in Python requires you to have basic knowledge of python programming and the OpenCV library. We will be needing following libraries: OpenCV: A strong library used for Machine Learning, Imutils: To Image Processing, Numpy: Used for Scientific, Computing. Image is stored in a numpy array and Argparse: Used to give input in command line.

* Here in this project, we developed a system that helps to prevent crowds or traffic control and many other uses.
* A real-life application of human object detection model.
* Use of Image Processing, Video Processing, Object Detection and Deep neural networks are used which are rigorous parts of modern Artificial Intelligence.
* The project will focus on human object detection and count of people in an image or in a real time video through camera or some other source.

**1.5 CHAPTER WISE SUMMARY**

In Chapter 1 we are going to see about the introduction of the project summary and the overview of the project.

In Chapter 2 we are going to see the requirements which are required for the project and we also we can see the architecture of the diagram like use case diagram and sequence

diagram.

In Chapter 3 we are going to see about the modules used in projects, brief descriptions of the modules and the code implementation.

In Chapter 4 we are going to see the testing results and the validation results and the verification done regarding the project.

In Chapter 5 we are going to see the Conclusions and further scope of the project.

**CHAPTER 2**

**ANALSIS AND DESIGN**

**2.1 REQUIREMENTS**

**PyCharm**:

This is platform used to develop the projects regarding the python language, Our project is based python language so we used this editor for coding the project

**Python**:

The language used to develop the project, which is most popular language, So this is the main requirement for doing this project. Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990.

**Python 3.0**:

Python 3.0 was released in 2008. Although this version is supposed to be backward incompatibles, later many of its important features have been backported to be compatible with version 2.7.

**Machine Learning**:

Here we use the Machine learning techniques to make prediction for the project and give the suggested results so machine learning is the most important requirement for this project

**Dataset**:

Every machine learning algorithm required the data set to classify the data here we used the label data to make predictions

**Human Detection**:

Human detection is the task of locating all instances of human beings present in an image, and it has been most widely accomplished by searching all locations in the image, at all possible scales, and comparing a small area at each location with known templates or patterns of people.

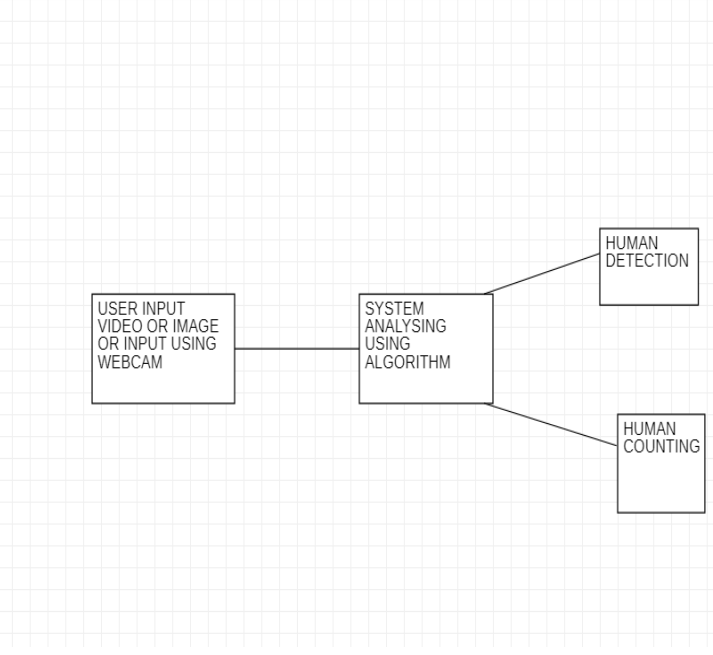
In this we can use various predefined methods and can detect the human in any image, video and can even get various factors like accuracy, each detections counting, etc.

**HOG (Histogram of Oriented Gradients**):

Histogram of Oriented Gradients (HOG) is a feature descriptor used in image processing, mainly for object detection.A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information from it.

**2.2 ARCHITECTURE**

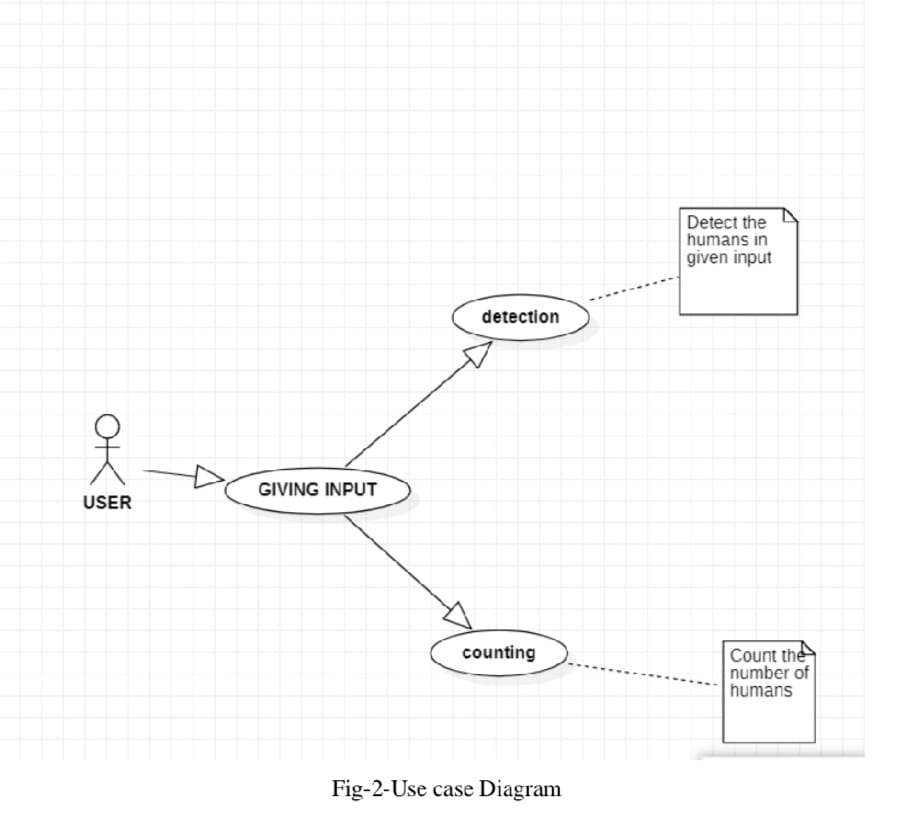
Here we have the basic architecture of the Application where we can find the flow of the system.



**FIG 2.1 ARCHITECTURE**

**2.3 USECASE DIAGRAM**

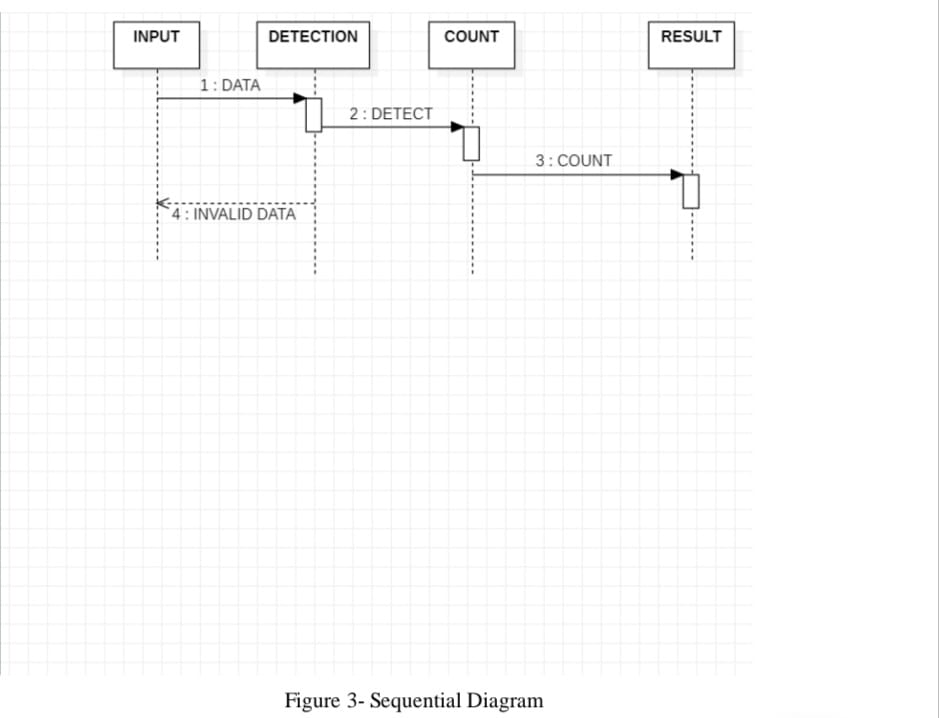
This use case diagram describes the high-level functions and scope of the system. These diagrams also identify the interactions between the system and its actors. The use cases and the actors in use case diagrams describe what the system does and how the actors use it.



**FIG 2.2 USECASE DIAGRAM**

**2.4 SEQUENCE DIAGRAM**

Sequence diagrams, commonly used by developers, model the interactions between objects in a single use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.



**FIG 2.3 SEQUENCE DIAGRAM**

**CHAPTER 3**

**IMPLEMENTATION**

**3.1 MODULE DESCRIPTION**

Here we used many modules for making this project the project these are the modules which are used for implementation.

• python 3

• Numpy

• cv2

• Argparse

• HOG (Histogram of Oriented Gradients)

• PIL

• IMUTILS

• Artificial Intelligence (AI) Algorithm

**Python 3:**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990.

**Python 3.0:** Python 3.0 was released in 2008. Although this version is supposed to be backward incompatibles, later many of its important features have been backported to be compatible with version 2.7.

**NumPy:**

NumPy is s a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

**Cv2:**

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

**Argparse:**

Python argparse () is one of the recommended python modules that take care of multiple scripting needs in an automated fashion by enabling the developer to create reproducible scripts right away from the jupyter notebook code. argparse () enables the user to provide values for the variables during the runtime process.

**HOG (Histogram of Oriented Gradients):**

Histogram of Oriented Gradients (HOG) is a feature descriptor used in image processing, mainly for object detection. A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information from it.

**PIL: Python Imaging Library**

PIL stands for Python Imaging Library. It is a popular open-source library for handling and manipulating images in Python. PIL provides a wide range of functions and methods to perform tasks such as opening, manipulating, and saving various image file formats. . It is a popular open-source library for handling and manipulating images in Python. PIL provides a wide range of functions and methods to perform tasks such as opening, manipulating, and saving various image file formats.With PIL, you can perform operations like resizing, cropping, rotating, filtering, and applying various image effects. It also supports basic image processing tasks such as color manipulation, histogram analysis, and pixel-level operations. PIL provides an easy-to-use and intuitive interface for working with images, making it a convenient choice for many image-related tasks in Python.

**IMUTILS**

Imutils is a Python library that provides convenience functions to make working with OpenCV easier. It is specifically designed to simplify common image processing tasks and streamline the integration of OpenCV with Python.

**Artificial Intelligence (AI) Algorithms**

Artificial Intelligence (AI) algorithms refer to the computational procedures and methods used to enable machines or computer systems to perform tasks that typically require human intelligence. These algorithms form the core of AI systems and enable machines to learn, reason, and make decisions.

LA

**3.2 IMPLEMENTATION DETAILS**

We have implemented the project for three different cases:

1. Image

2. Video

3. Camera

**Detection & Counting through Image:**

This section works with real time images. Here will allow user to select any real time image from the local system and then user can detect the humans in it. And along with that it also gives the count of humans detected.

**Detection & Counting through Video:**

This section works with real time videos. Here will allow user to select any real time video from the local system and then user can detect the humans in it. Now in case of video, since it is running, while the detection process is going on user will be able to see the detected peoples and their count for each frame per second of the video.

**Detection & Counting through Camera:**

This section works somehow similar to case of video. Here user will be asked to first open the webcam, and it will detect humans that will comes in that webcam during the detection process.

The project in Python requires you to have basic knowledge of python programming and the OpenCV library. We will be needing following libraries:

**• OpenCV:** A strong library used for machine learning

**• Imutils**: To Image Processing

**• Numpy:** Used for Scientific Computing. Image is stored in a numpy array.

**• Argparse:** Used to give input in command line.

OpenCV has already been implemented in an efficient way to combine the HOG Descriptor algorithm with Support Vector Machine or SVM.

cv2.HOGDescriptor\_getDefaultPeopleDetector () calls the pre-trained model for Human detection of OpenCV and then we will feed our support vector machine with it.

A video combines a sequence of images to form a moving picture. We call these images as Frame. So, in general we will detect the person in the frame. And show it one after another that it looks like a video.

That is exactly what our Detect () method will do. It will take a frame to detect a person in it. Make a box around a person and show the frame..and return the frame with person bounded by a green box.

Everything will be done by detectMultiScale(). It returns 2-tuple.

There are two ways of getting Video.

1. Web Camera

2. Path of file stored

In this deep learning project, we can take images also.

So, our method will check if a path is given then search for the video or image in the given path and operate.

Otherwise, it will open the webCam.

cv2.VideoCapture(0) passing 0 in this function means we want to record from webcam.

video.read() read frame by frame. It returns a check which is True if this was able to read a frame otherwise False.

Now, for each Frame, we will call detect () method. Then we write the frame in our output file.

The function argparse() simply parses and returns as a dictionary the arguments passed through your terminal to our script.

There will be Three arguments within the Parser:

1. Image: The path to the image file inside your system

2. Video: The path to the Video file inside your system

3. Camera: A variable that if set to ‘true’ will call the cameraDetect() method.

To get the output, Run the following codes for different kinds of inputs in Terminal as an admin.

1. To give video file as input:

python main.py -v ‘Path\_to\_video’

1. To give image file as input:

python main.py -i ‘Path\_to-image’

1. To use the camera:

python main.py -c True

1. To save the output:

Python main.py -c true -o ‘file name’

How this Script works:

• User just need to download the file and run the program on their local system. • User need to open the file location and need to open the terminal of the page.

• Below is the image showing the location and right click to open terminal.

**3.3 SOURCE CODE**

**PYTHON CODE:**

import cv2

import imutils

import numpy as np

import argparse

HOGCV = cv2.HOGDescriptor()

HOGCV.setSVMDetector(cv2.HOGDescriptor\_getDefaultPeopleDetector())

def detect(frame):

bounding\_box\_cordinates, weights = HOGCV.detectMultiScale(frame, winStride=(4, 4), padding=(8, 8), scale=1.03)

person = 1

for x, y, w, h in bounding\_box\_cordinates:

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.putText(frame, f'person {person}', (x, cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255), 1)

person += 1

cv2.putText(frame, 'Status : Detecting ', (40, cv2.FONT\_HERSHEY\_DUPLEX, 0.8, (255, 0, 0), 2)

cv2.putText(frame, f'Total Persons : {person - 1}', (40, 70), cv2.FONT\_HERSHEY\_DUPLEX, 0.8, (255, 0, 0), 2)

cv2.imshow('output', frame)

return frame

def humanDetector(args):

image\_path = args["image"]

video\_path = args['video']

if str(args["camera"]) == 'true' : camera = True

else : camera = False

writer = None if args['output'] is not None

and image\_path is None:

writer = cv2.VideoWriter(args['output'],cv2.VideoWriter\_fourcc(\*'MJPG'), 10, (600,600))

if camera:

print('[INFO] Opening Web Cam.')

detectByCamera(ouput\_path,writer)

elif video\_path is not None:

print('[INFO] Opening Video from path.')

detectByPathVideo(video\_path, writer)

elif image\_path is not None:

print('[INFO] Opening Image from path.')

detectByPathImage(image\_path, args['output'])

def detectByCamera(writer):

video = cv2.VideoCapture(0)

print('Detecting people...')

while True:

check, frame = video.read()

frame = detect(frame)

if writer is not None:

writer.write(frame)

key = cv2.waitKey(1)

if key == ord('q'):

break

video.release()

cv2.destroyAllWindows()

def detectByPathVideo(path, writer):

video = cv2.VideoCapture(path)

check, frame = video.read()

if check == False:

print('Video Not Found. Please Enter a Valid Path (Full path of Video Should be Provided).')

return

print('Detecting people...')

while video.isOpened():

# check is True if reading was successful

check, frame = video.read()

if check:

frame = imutils.resize(frame, width=min(800, frame.shape[1]))

frame = detect(frame)

if writer is not None:

writer.write(frame)

key = cv2.waitKey(1) if key == ord('q'):

break

else:

break

video.release()

cv2.destroyAllWindows()

def detectByPathImage(path, output\_path):

image = cv2.imread(path)

image = imutils.resize(image, width = min(800, image.shape[1]))

result\_image = detect(image)

if output\_path is not None:

cv2.imwrite(output\_path, result\_image)

cv2.waitKey(0)

cv2.destroyAllWindows

def argsParser():

arg\_parse = argparse.ArgumentParser() arg\_parse.add\_argument("-v", "--video", default=None, help="path to Video File ")

arg\_parse.add\_argument("-i", "--image", default=None, help="path to Image File ")

arg\_parse.add\_argument("-c", "--camera", default=False, help="Set true if you want to use the camera.")

arg\_parse.add\_argument("-o", "--output", type=str, help="path to optional output video file") args = vars(arg\_parse.parse\_args())

return args

if \_\_name\_\_ == "\_\_main\_\_": HOGCV = cv2.HOGDescriptor()

HOGCV.setSVMDetector(cv2.HOGDescriptor\_getDefaultPeopleDetector())

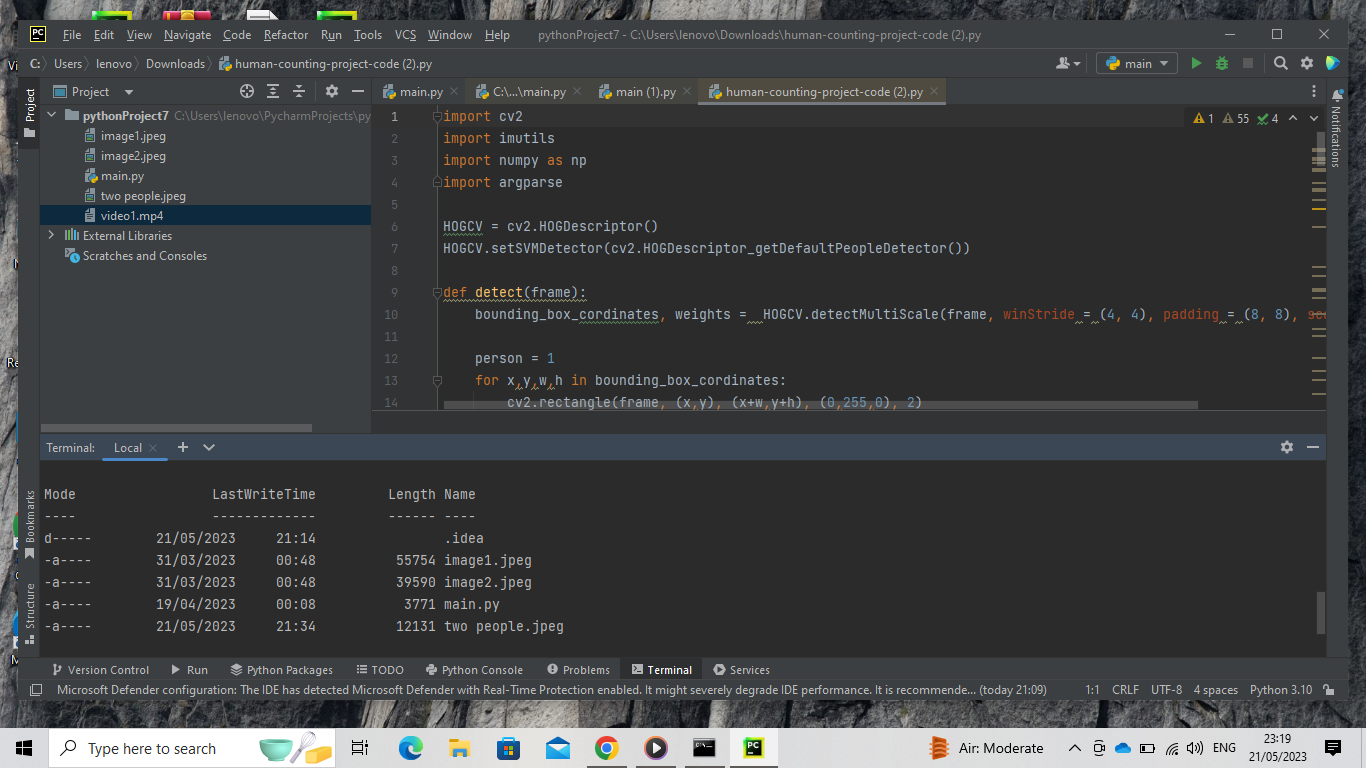
args = argsParser()

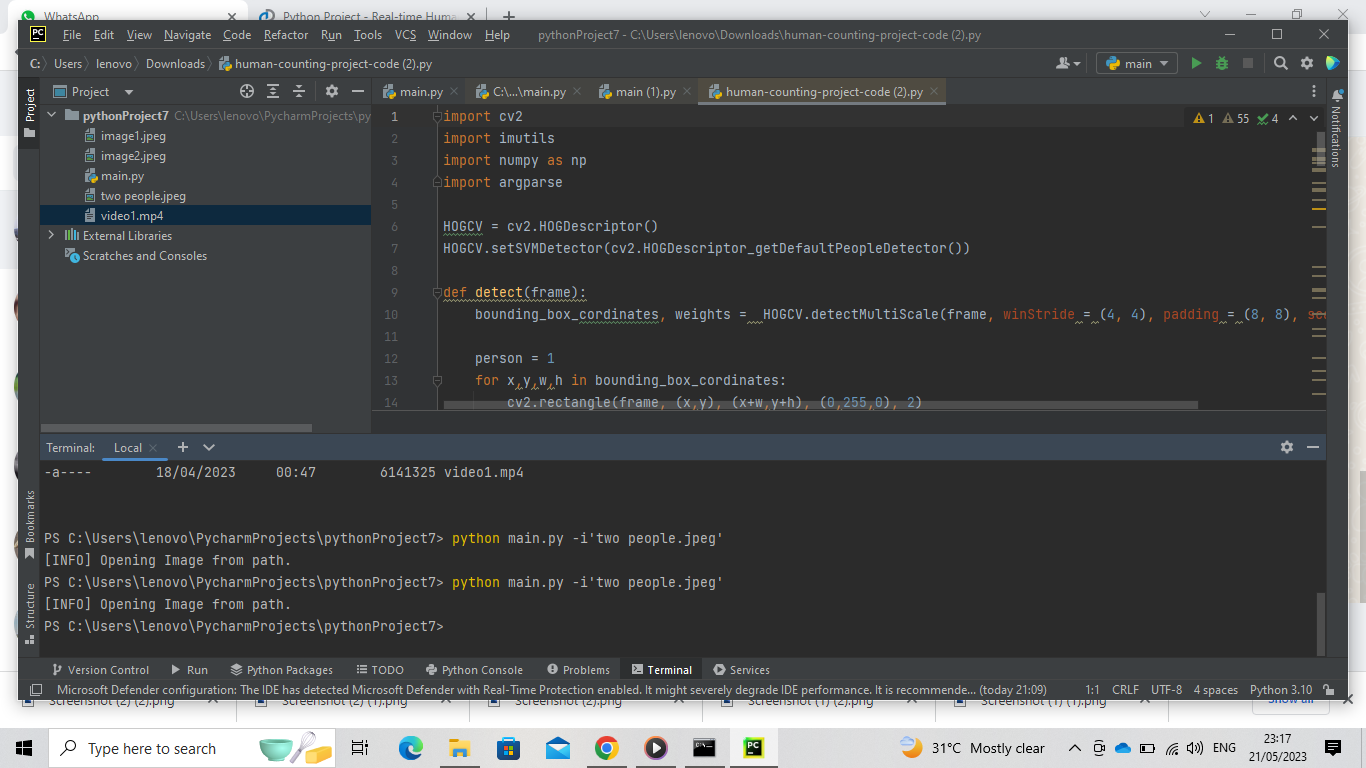
humanDetector(args)

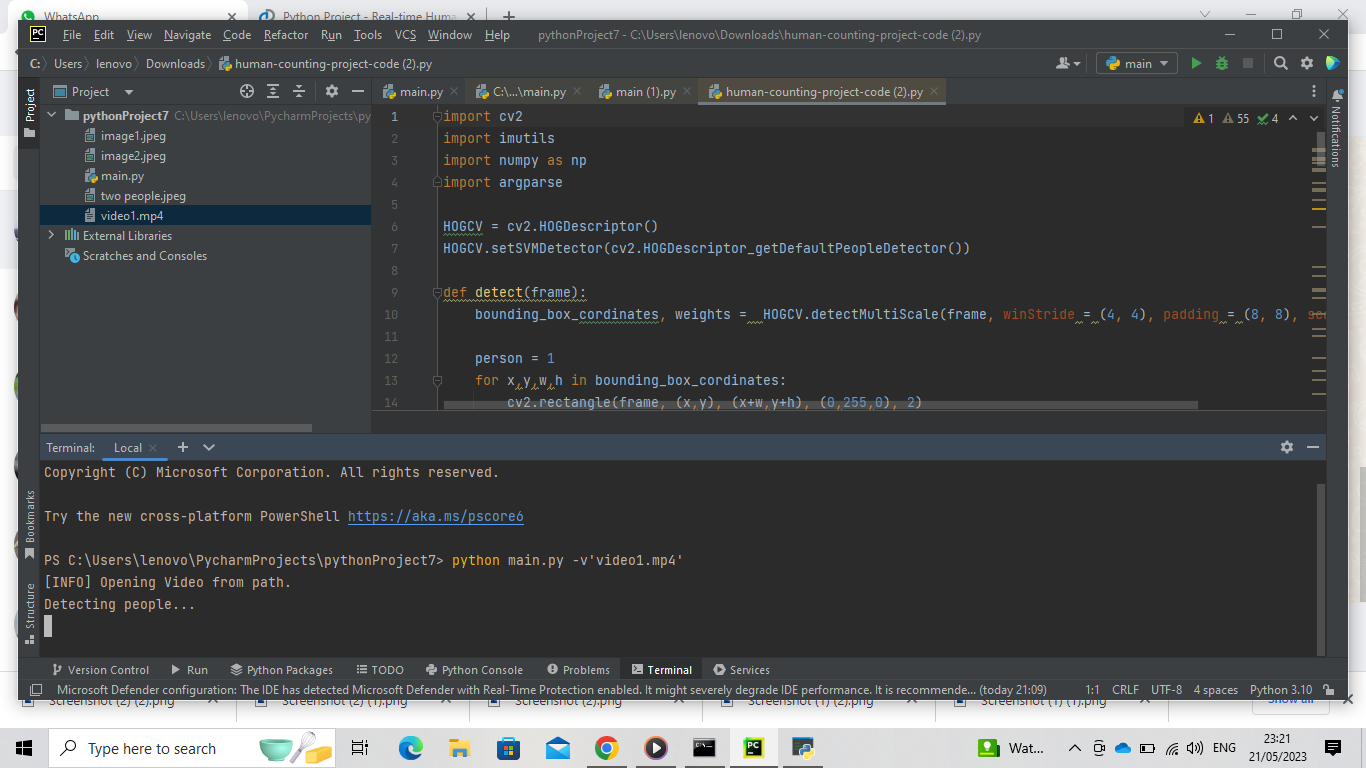
**3.4 TOOLS USED**

Here we have used Pycharm for this project to implement the code pycharm is the easy way to implement the python programs and do the projects.

**3.5 SNAPSHOT**





****

**CHAPTER 4**

**TEST RESULTS/EXPERIMENTS/VERIFICATION**

**4.1 TESTING**

**Test Using video:**

Test video:



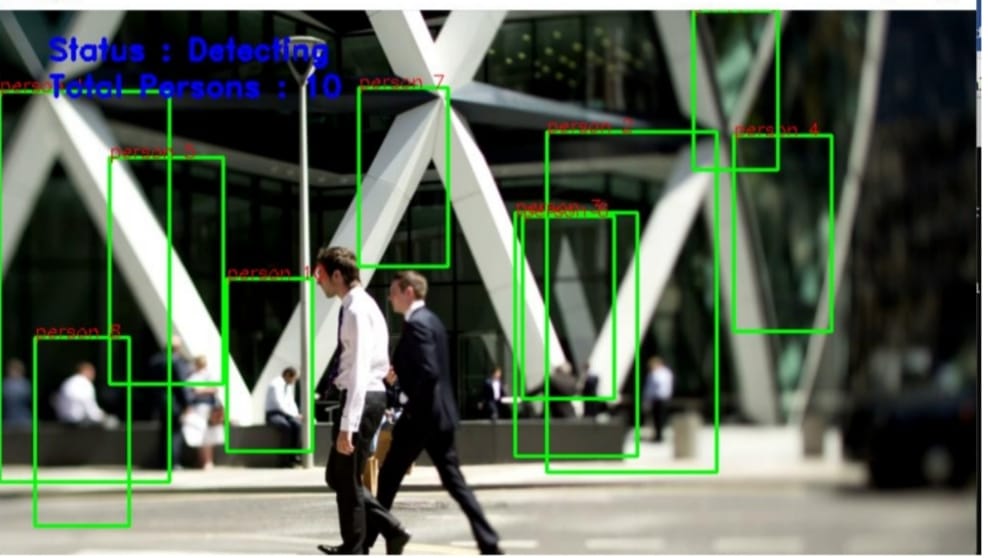
**Test Using Image:**

Test image

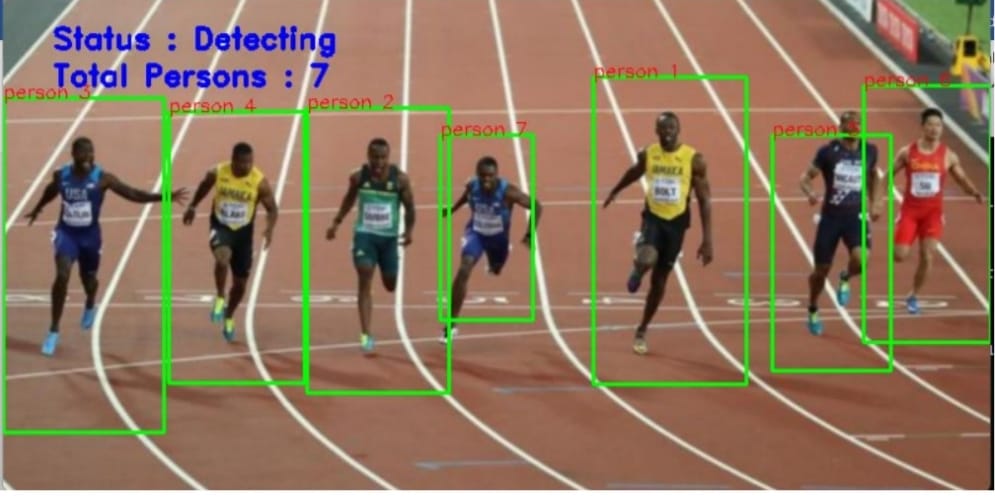
****

* 1. **RESULTS AND VERIFICATION:**

**Result for video test:**



**Result for image test:**

****

**CHAPTER 5**

**CONCLUSION AND FUTURE SCOPE**

**CONCLUSION**

So, with the help of the input, either it is video or with camera we are detecting humans and getting count of them. And with using HOG descriptor we will get a better accuracy than Haar cascade classifier. Now coming to the future scope of this project or application, since in this we are taking any image, video or with camera we are detecting humans and getting count of it, along with accuracy.

Human detection and counting using Python can be accomplished effectively with the help of computer vision techniques and machine learning algorithms. By leveraging libraries such as OpenCV and TensorFlow, it is possible to develop robust solutions for detecting and counting humans in images or videos.

The process typically involves the following steps:

Data acquisition: Collecting images or video footage containing humans.

Preprocessing: Cleaning and preparing the data for analysis. This may involve resizing, normalization, or other image preprocessing techniques.

Feature extraction: Extracting relevant features from the data, such as shape, color, or texture information. This step helps in representing humans in a format that can be understood by machine learning algorithms.

Model training: Utilizing machine learning algorithms, such as support vector machines (SVM), random forests, or deep learning models like convolutional neural networks (CNN), to train a model on the extracted features. The model learns to differentiate between human and non-human objects.

Human detection: Applying the trained model to detect humans in images or videos. This involves running the model on each frame of a video or image and identifying regions.

Counting: Analyzing the detected regions or bounding boxes to count the number of humans present. This can be done by applying counting algorithms or by simply tallying the number of detected instances.

Visualization and output: Presenting the results in a visually informative manner, such as drawing bounding boxes around detected humans or displaying the total count.

Python provides a rich ecosystem of libraries and tools that make it feasible to implement human detection and counting pipelines efficiently. However, it's important to note that the accuracy and performance of the solution heavily rely on the quality of the training data, the chosen algorithms, and the optimization techniques employed.

**FUTURE SCOPE**

The future scope for human detection and counting using Python is promising, as advancements in computer vision and machine learning continue to push the boundaries of accuracy and efficiency. Here are some potential areas of development and improvement:

Real-time performance: Enhancing the speed and efficiency of human detection and counting algorithms to achieve real-time processing. This would enable applications in live video surveillance, crowd management, and interactive systems.

Improved accuracy: Refining the accuracy of human detection models by leveraging more advanced deep learning techniques, such as object detection frameworks like YOLO (You Only Look Once) or EfficientDet. These models can detect humans with high precision and handle complex scenarios with multiple individuals.

Pose estimation: Integrating pose estimation algorithms with human detection to provide additional information about human poses and movements. .

Multi-camera systems: Developing methods to handle human detection and counting in multi-camera systems, where multiple camera feeds are used to track individuals across different views. This could enable applications in large-scale surveillance or tracking in complex environments.

Occlusion handling: Addressing the challenges posed by occlusions, where humans are partially or fully obstructed by objects or other people. Developing algorithms that can accurately detect and count humans even in the presence of occlusions would be valuable for robust tracking and counting.

So, some of the future scopes can be:

1. This can be used in various malls and other areas, to analyze the maximum people count, and then providing some restrictions on number of people to have at a time at that place.
2. This can replace various mental jobs, and this can be done more efficiently with machines.

**REFFERENCE**

1. Programming computer vision with python, 1st Edition,jan Eric Solem,2012, O’Reily
2. Learning OpenCv, Adrian Kachler and Gray Rost Bradski, 2008, O’ Reily
3. Deep Learning with Tensorflow, Giancarlo Zaccone , Md.Rezaul Karim ,Ahmed Menshawy,2017
4. Python GUI Programming with Tkniter,Alan D.Moore,2018
5. Python Standard,Fredrik Lundh,2001, O’Reil
6. AVSS 2005. Real time event detection solutions for enhanced security and safety in public transportation. <http://www-dsp.elet.polimi.it/avss2005/>.
7. H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. Computer Vision and Image Understanding, 89(3):114–141, March 2003.
8. D. Gavrila. Pedestrian detection from a moving vehicle. In Proc. 6th European Conf. Computer Vision, Dublin, Ireland, volume 2, pages 37–49, 2000.
9. J. Giebel, D.M. Gavrila, and C. Schnorr. A bayesian frame- ¨ work for multi-cue 3d object tracking. In Proc. 8th European Conf. Computer Vision, Prague, Czech Republic, pages 241– 252, 2004.
10. Haritaoglu, D. Harwood, and L. S. Davis. HYDRA: Multiple people detection and tracking using silhouettes. In In IEEE International Workshop on Visual Surveillance, pages 6–13, 1999.
11. Oren, M., Papageorgiou, C., Sinha, P., Osuma, E., Poggio, T.: Pedestrian detection victimization wave templates. In: Proceedings of laptop Vision and Pattern Recognition (1997)Google Scholar .
12. Dalal, N., Triggs, B., Schmid, C.: Human detection victimization homeward histograms of flow and look.