A PROJECT REPORT ON

# ENHANCED CROWD MONITORING AND RISK ASSESSMENT SYSTEM

*Mini project submitted in partial fulfillment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**
**IN**
**INFORMATION TECHNOLOGY**
**(2020-2023)**
**BY**

| | |
|---|---|
| MALLELA MOUNIKA | 20241A12E7 |
| THOTA GAYATHRI | 20241A12H4 |
| SRIKHAKOLLU VYSHNAVI SRIJA | 20241A12H0 |

*Under the Esteemed guidance*
*of*
**T.N.P MADHURI**
**Assistant Professor**
**Dept of IT.**

**DEPARTMENT OF INFORMATION TECHNOLOGY**
**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY**
**(AUTONOMOUS) HYDERABAD**

## *CERTIFICATE*

This is to certify that it is a bonafide record of Mini Project work entitled **"ENHANCED CROWD MONITORING AND RISK ASSESSMENT SYSTEM" done by MALLELA MOUNIKA(20241A12E7), THOTA GAYATHRI(20241A12H4), SRIKHAOLLU VYSHNAVI SRIJA(20241A12H0),** of **B.Tech (IT)** in the Department of Information Technology, **Gokaraju Rangaraju Institute of Engineering and Technology** during the period 2019-2023 in the partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from GRIET, Hyderabad.

**T.N.P MADHURI**
**Assistant professor**
**(Internal project guide)**

**Dr.N.V.Ganapathi Raju ,**
**Head of the Department,**

**(Project External)**

## ACKNOWLEDGEMENT

We take immense pleasure in expressing gratitude to our Internal guide**, T.N.P MADHURI, Assistant Professor, Dept. of IT**, GRIET. We express our sincere thanks for her encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to **Dr. N. V. Ganapathi Raju,** HOD IT **,** our Project Co-ordinators **P.K Abhilash and B. Kanaka Durga** for their constant support during the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy,** Director, GRIET**,** and **Dr. J. Praveen,** Principal, GRIET**,** for providing us the conductive environment for carrying through our academic schedules and project with ease.

We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad.

Name: Mallela Mounika
Email:mounikareddymallela28@gmail.com
Contact No: 9100951215
Address: Kukatpally, Hyderabad.

Name: Thota Gayathri
Email: thotagayathri29@gmail.com
Contact No: 9533069568
Address: Old Malakpet, Hyderabad

Name: Srikhakollu Vyshnavi Srija
Email: vyshnavisrija2706@gmail.com
Contact No: 9542901998
Address: Srinagar colony, Hyderabad

# DECLARATION

This is to certify that the project entitled "**ENHANCED CROWD MONITORING AND RISK ASSESSMENT SYSTEM**" is a bonafide work done by us in partial fulfillment of the requirements for the award of the degree **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books, and paper publications are mentioned in the Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

| | |
|---|---|
| **MALLELA MOUNIKA** | **20241A12E7** |
| **SRIKHAKOLLU VYSHNAVI SRIJA** | **20241A12H0** |
| **THOTA GAYATHRI** | **20241A12H4** |

# TABLE OF CONTENTS

## LIST OF FIGURES

# ABSTRACT

In today's digital age, many locations still rely on traditional approaches to count crowds, such as manual register maintenance, people counters, and sensor-based counting at entrances. However, these methods prove inadequate in areas where the movement of people is sporadic, highly unpredictable, and constantly changing. These methods are time-consuming and tedious. This Application checks the total count of people and gives the count of people in closer proximity. When the closer proximity count is near to total count it might lead to stampedes or accidents. This System also helps us to monitor the people with more risk based on their distance between each other. In this project we use neural network based predefined model for better accuracy. YOLOv3 algorithm adopts an architecture which detects each person in crowd, spot location with a bounding box, and does the counting. Deploying it on current surveillance systems and drones used by police to monitor large areas can help to prevent stampedes by allowing automated and better tracking of activities happening in the area. It can also be used to alert police in case .It offers real-time insights into the area. of an uncontrollable situation in a particular area.

**Domain(s)**: Machine learning

# 1. INTRODUCTION

## 1.1 Introduction to the project

In this project we aim to design and implement a real-time system for counting people in visual surveillance and this is a challenging and crucial task in the field of crowd management and safety control. Whether it's a bustling city street, a crowded concert venue, or a public gathering, accurately monitoring the number of individuals present is essential for ensuring the well-being and security of everyone involved.

Social gatherings are an integral part of people's lives, bringing together individuals for various events and occasions. However, when people come together in crowds, their behaviour often changes compared to when they are alone or in smaller groups. The dynamics of a crowd can create unique challenges, making it crucial to have experienced crowd management strategies in place to prevent potential disasters. One significant concern in crowded situations is the risk of crowd collapses and crushes. When a large group of people becomes dangerously overcrowded, the pressure on each individual can lead to the collapse of the crowd or create such high density that people can be crushed or asphyxiated. These incidents can have catastrophic consequences, resulting in injuries or even fatalities. However, by implementing effective crowd management strategies, many of these crowd disasters can be prevented.

Crowd monitoring plays a vital role in understanding and managing crowd behaviour. There are two primary approaches to crowd monitoring: macroscopic and microscopic. Macroscopic methods analyze the crowd scene as a whole, focusing on crowd density, crowd counting, and flow estimation. These techniques provide an efficient way to handle high-density crowds and gain a general understanding of crowd behaviour.

On the other hand, microscopic techniques take a more detailed approach by detecting individual subjects within the crowd and analysing their statistics. By tracking and analysing individuals, microscopic techniques can provide accurate information about crowd states, even for sparse groups. Both macroscopic and microscopic approaches have their advantages and trade-offs, and they can be used together to ac have a comprehensive understanding of crowd dynamics. To further enhance crowd monitoring capabilities, advancements in technology have enabled the use of unmanned aerial vehicles (UAVs) to capture video footage of crowds from an elevated perspective. This aerial view allows for a broader view of the scene and facilitates the generation of discrete density maps. These maps classify different areas of the crowd as dense, medium, sparse, or empty, providing valuable insights into crowd states and distribution.

Crowd counting and action recognition techniques are integral to analysing crowd behaviour in the context of social distancing. Density-based approaches are particularly useful, as they can suppress errors and faulty counts. By generating long-term occupancy and crowd density maps, spatial patterns of pedestrians can be captured and analysed. Occupancy maps describe the spatial signature exerted by individuals in the scene, while crowd density maps encode the spatial impression of social distance infractions. Heatmaps can also be generated to identify regions where social distancing violations occur frequently, helping to pinpoint high-risk areas. By utilizing these techniques, short-term and long-term occupancy or crowd density maps become essential tools for identifying high-risk regions within a crowd scene. These maps provide valuable insights into crowd behaviour and allow for a quantification of pedestrians' compliance with social distancing guidelines. By effectively managing crowd control and identifying potential risks, safety can be ensured in various public spaces.

In conclusion, accurate crowd counting and efficient crowd management strategies are essential for maintaining safety and security in crowded environments. The combination of macroscopic and microscopic approaches, along with the analysis of occupancy maps, density-based techniques, and heatmaps, enables a comprehensive understanding of crowd behaviour and assists in identifying high-risk areas. By implementing these methods, authorities can take proactive measures to enforce crowd control, mitigate risks, and ensure the well-being of individuals in public spaces.

**Figure 1.**A Crowded Area

## 1.2 Existing Systems

In the past, manual registers were used to keep track of the number of people in a crowd. However, with advancements in technology, the focus has shifted towards developing systems that not only detect people in a crowd but also manage the associated risks. Currently, the existing systems primarily concentrate on the detection of individuals and ensuring safety measures. However, there is now a growing emphasis on accurately counting the number of people present in a crowd and actively maintaining this count. This involves incorporating live video feeds and real-time detection of crowds into the existing framework.

**Figure 2.**Manual Register

### Disadvantages of Existing System

• The disadvantage of the current existing system is, the detection of people is done only by giving a video feed.

## 1.3 Proposed System

The existing systems focused completely on analyzing the video input fed to them rather than working on live-streaming videos. And the system provides no warnings to the people or users in any form when there is a high-risk situation, only detection of persons is done and the count of persons in the video frame is given as the end output.

To overcome these problems we proposed a system which is built using a machine learning model which is comprised of the following stages:

(1) Read a video frame.

(2) Detect human subjects and localize their positions.

(3) Transform the detected positions to real-world coordinates.

(4) Estimate the subject's interpersonal distances using camera view calibration and Euclidean distance.

(5) With the estimated inter-personal distances. The objects are classified as high risk and safe which are indicated as red and green boxes respectively.

(6) Warnings are triggered when there is a high risk

# 2. REQUIREMENT ENGINEERING

## 2.1 Software Requirements

The product viewpoint and features, operating system and operating environment, graphics requirements, design limitations, and user documentation are all part of the functional requirements or overall description papers.

The application of requirements and implementation restrictions provides an overall picture of the project in terms of what areas of strength and weakness exist and how to address them.

- Operating System: - Windows 10
- Technology: Python
- Software: Visual Studio Code

## 2.2 Hardware Requirements

The minimum hardware requirements vary greatly depending on the product being produced by an Enthought Python / Canopy / VS Code user. Apps that require big arrays/objects in memory will demand more RAM, but applications that require several calculations or activities to be performed rapidly would require a faster CPU.

- Operating system: Windows, Linux
- Processor: minimum intel i3
- Ram: minimum 4 GB
- Hard disk: minimum 250GB

## 2.3 Functional Requirements

- Data Collection
- Data Preprocessing
- Training And Testing
- Modeling
- Predicting

# 3.LITERATURE SURVEY

As a part of the literature survey, we investigated some of the human crowd-monitoring systems that already exist in the market. The aim is to observe how these models work and to see how they can be improved and how are they different. To date, it is identified that the following human crowd-monitoring systems are good and are offering relatively similar services.

In the paper titled "Object tracking and counting in a zone using YOLOv4, DeepSORT, and TensorFlow," authors Shailender Kumar, Vishal, Pranav Sharma, and Nitin Pal present a system that addresses the challenge of multiple object tracking in a given frame. The proposed solution involves three stages: detection, identification, and tracking of objects within a specific zone. To achieve object detection and recognition, the YOLO algorithm is utilized, which is capable of classifying objects into 80 different classes. This allows for accurate identification of objects within the frame. The system also incorporates motion prediction and feature generation, where an estimation model is created to capture the movement of objects. Kalman filters are used to model the states of moving objects, enhancing the tracking process. The DeepSORT algorithm is employed for object tracking, utilizing Kalman filters from the previous frame and associating newly detected objects in the current frame. This integration of YOLO, DeepSORT, and TensorFlow results in an accurate and efficient system for object tracking and counting.

The paper titled "Human Crowd Detection for City Wide Surveillance" by authors Dushyant Kumar Singh, SumitParoothi, Mayank Kumar Rusiac, and Mohd. Aquib Ansari presents a comprehensive autonomous solution for detecting and analyzing human crowd behavior in public spaces. The system aims to reduce the workload of individuals and enhance the effectiveness of law enforcement authorities in patrolling public areas within a city. Real-time detection of unusual behavior, such as violent crowd behavior, and pedestrian presence in restricted areas is achieved through object detection approaches. Human pedestrians are detected using object detection techniques, while crowd behavior analysis is performed using violent flow descriptors and SVM classification. The system also includes a real-time information dissemination component, which enables the prompt resolution of any severe impact caused by crowd violence. Computer vision techniques are employed to detect suspicious activities from real-time video sequences, and an intricate communication system is designed to minimize response time for the police authorities. This integrated system enables the early detection and resolution of potential security threats in public spaces.

In the paper titled "Single-Pixel Thermopile Infrared Sensing for People Counting," authors Ashish Pandharipande, Abhishek Murthy, Erik Hagenaars, and Geert Leu's propose a people counting method that combines a single-pixel thermopile and a passive infrared sensor. The authors develop a thermopile signal model to accurately measure object temperatures in environments with multiple people. The people counting method is based on cumulative sum (CUSUM) change detection in the

object temperature signal. By analyzing the differential mean temperature in detected changes, the system estimates the number of people present. The proposed method is further enhanced through decision fusion with an infrared vacancy sensor, improving the accuracy of the people count. The authors evaluate the method using simulated data generated from the developed signal model and experimental data from a real office/meeting room environment. The results show low counting errors, indicating the effectiveness of the single-pixel thermopile infrared sensing approach for people counting.

In conclusion, these papers present innovative approaches to tackle different challenges in surveillance and crowd management. The first paper focuses on object tracking and counting using YOLOv4, DeepSORT, and TensorFlow, providing a comprehensive solution for accurate and efficient object tracking. The second paper addresses human crowd detection for city-wide surveillance, offering an autonomous system that detects unusual behavior and pedestrian presence in restricted areas, facilitating prompt responses from law enforcement authorities. The third paper introduces a people counting method using a single-pixel thermopile and passive infrared sensing, enabling accurate people counting even in environments with multiple individuals. Overall, these contributions

# 3. TECHNOLOGY

## 4.1 ABOUT PYTHON

Python is a sophisticated and quick programming language that works well with other languages and is user-friendly and simple to learn, and is open source. It is a versatile programming language that is utilized in Dialog flow. Because of its adaptability, simplicity, and long-standing reputation, itis utilized as the foundation for the most popular Abased programming.

Guido van Rossum created Python in the Netherlands' National Research Institute for Mathematics and Computer Science in the late 1980s and early 1990s. Python is based on a variety of languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages. Python has copyright protection. Python source code is now available under the GNU General Public License, much like Perl (GPL). Python is presently maintained by the institute's core development team; however Guido van Rossum continues to play a key role in its direction.

Python is frequently used by software engineers as a support language for build control and administration, testing, and a variety of other tasks. Build control with SCons. For automated continuous compilation and testing, use Buildbot and Apache Gump.

**Figure 3.** Python

## 4.2 APPLICATIONS OF PYTHON

Python is a general-purpose language that can be used in a wide range of tasks, which is a substantial benefit of knowing it. The following are just a few of the most common fields where

Python is used:

- Data science
- Web Development
- Scientific and mathematical computing
- Mapping and geography (GIS software)
- Basic game development
- Computer graphics

## 4.3 PYTHON IS FREQUENTLY USED IN DATA SCIENCE

Python's ecosystem is expanding each time, and it's becoming increasingly capable of statistical analysis.

It's the best balance of size and sophistication (in terms OD data processing).

Python places a premium on efficiency and readability.

Python is a programming language that is used by programmers who want to do data analysis or use statistical approaches (and by devs that turn to data science).

For data visualisation, machine learning, natural language processing, complicated data analysis, and more, Python scientific packages abound. Python is an excellent tool for scientific computing because of all of these features, and it's a good alternative to commercial products like MatLab. The following are the most widely used data science libraries and tools:

### 4.3.1 PANDAS

Pandas gets its name from the econometric phrase "panel data" which refers to multidimensional structured data sets. It's a data manipulation and analysis library. For managing numerical tables and time series, the library contains data structures and functions. "Python Data Analysis Library" is another name for it.

### 4.3.2 NUMPY

NumPy is a Python package for array processing. It includes a high-performance multidimensional array object as well as utilities for manipulating them. This is a foundational Python package that adds support for big, multi-dimensional arrays and matrices, as well as a vast library of high-level mathematical methods to operate on these arrays.

### 4.3.3 MATPLOTLIB

Matplotlib is a Python 2D plotting package that generates high-quality figures in a range of hardcopy and interactive formats across platforms. Plots, histograms, power spectra, bar charts, error charts, scatterplots, and more are all possible with Matplotlib.

### 4.3.4 SCIKIT-LEARN

Scikit-learn is undoubtedly Python's most helpful machine learning library. Classification, regression, clustering, and dimensionality reduction are just a few of the useful capabilities in the sklearn toolkit for machine learning and statistical modelling. It is meant to interact with the Python numerical and scientific libraries NumPy and SciPy, and features support vector machines, random forests, gradient boosting, k-means, and DBSCAN, among other classification, regression, and clustering algorithms.

### 4.3.5 CSV

The CSV (Comma Separated Values) format is the most used spreadsheet and database import and export format. The csv module provides classes for reading and writing CSV tabular data. It enables programmers to say things like "put this data in the format favoured by Excel" or "read data from this file generated by Excel" without having to grasp the specifics of Excel's CSV format. Programmers can also define their own special-purpose CSV format or describe the CSV formats that other apps understand.

## 4.4 Machine Learning

**Figure 4.** Machine Learning

**Need for Machine Learning**

The capacity to analyze, evaluate, and resolve complex problems makes humans the planet's most intelligent and advanced species at this time. AI has not yet completely exceeded human intellect; it is still in its infancy. The next question is: Why is machine learning required? To "make judgements with efficiency and scalability, based on facts" is the greatest basis for doing this. In order to extract crucial information from data and perform a range of practical tasks and handle difficulties, organizations have lately made large investments in more modern technologies like Artificial Intelligence, Machine Learning, and Deep Learning. When the procedure is automated, we may refer to it as machine-driven judgements that are supported by data. When applying programming logic If this is not practicable, data-driven judgments may be used in its stead. Although it is true that human intelligence is necessary for existence, everyone must be able to manage problems in the real world efficiently and comprehensively. This leads to the need for machine learning.

Applications of Machines Learning:-According to experts, machine learning is the technology that is growing the quickest, and we are in the "golden year of AI and ML." It is used to find solutions to a wide range of challenging issues from the actual world that cannot be found using a traditional method.

Here are a few instances of ML's practical applications:

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

**Types of Machine Learning**

• Supervised Learning: Regression and classification models are employed to learn from a training dataset of labeled data. The learning process doesn't end until the target performance level is obtained.

In order to learn more and more about the data itself, unsupervised learning involves analyzing unlabeled data using factor and cluster analysis models to reveal the underlying structure in the data.

Semi-supervised learning is the result of combining unsupervised learning with a little amount of tagged data. Using labeled data significantly increases learning accuracy compared to supervised learning and is also more cost-effective.

• Reinforcement learning: Through trial and error, one learns the best course of action. Therefore, learning habits depend on the present condition and that will determine the next course of action

**Advantages:**

1. Identifies trends and patterns quickly - Machine learning can examine vast amounts of data and identify certain trends and patterns that people would not be able to identify. For an e-commerce site like Amazon, for instance, knowing its customers' browsing patterns and past purchases enables it to offer them the appropriate goods, discounts, and reminders. It makes advantage of the findings to show them relevant adverts.

2. No need for human involvement (automation)

You no longer have to supervise your project at every stage thanks to ML. Giving computers the capacity to learn enables them to make predictions and enhance algorithms on their own. Anti-virus programs are a typical illustration of this; they learn to filter

3. Ongoing Development

ML algorithms keep become more accurate and effective as they gather experience. They can consequently make wiser selections. Take the example of creating a weather forecast model. Your algorithms become faster at making more accurate predictions as your data set expands.

**Disadvantages:**

1. Data gathering

Machine learning demands large, comprehensive, impartial, and high-quality data sets for training. They might occasionally have to wait while new data is generated.

2. Resources and Timing

For machine learning (ML) to be effective, the algorithms must have enough time to mature and learn enough to achieve their goals with a high degree of accuracy and relevance. Additionally, it requires a lot of resources to run. This might result in you needing more processing power from your machine.

3. Results Interpretation

The capacity to correctly comprehend the information produced by the algorithms presents another significant problem. Additionally, you must carefully select the algorithms for your needs.

**Figure 5.** Deep Learning

## 4.5 ALGORITHMS:

### YOLOV3:

The YOLO (You Only Look Once) model, specifically YOLOv3, revolutionized the field of object detection by introducing a highly efficient and accurate approach. Traditional object detection methods involved multi-step processes, including region proposal and classification, which could be computationally intensive. YOLOv3 tackled this challenge by formulating object detection as a single regression problem, significantly mitigating computational complexity.

YOLOv3 is widely regarded as one of the state-of-the-art object detectors in deep learning. It excels in real-time applications, allowing for the identification of specific objects in videos, live feeds, or images with remarkable speed and accuracy. This makes it well-suited for various real-time applications, including surveillance systems, autonomous vehicles, and image analysis.

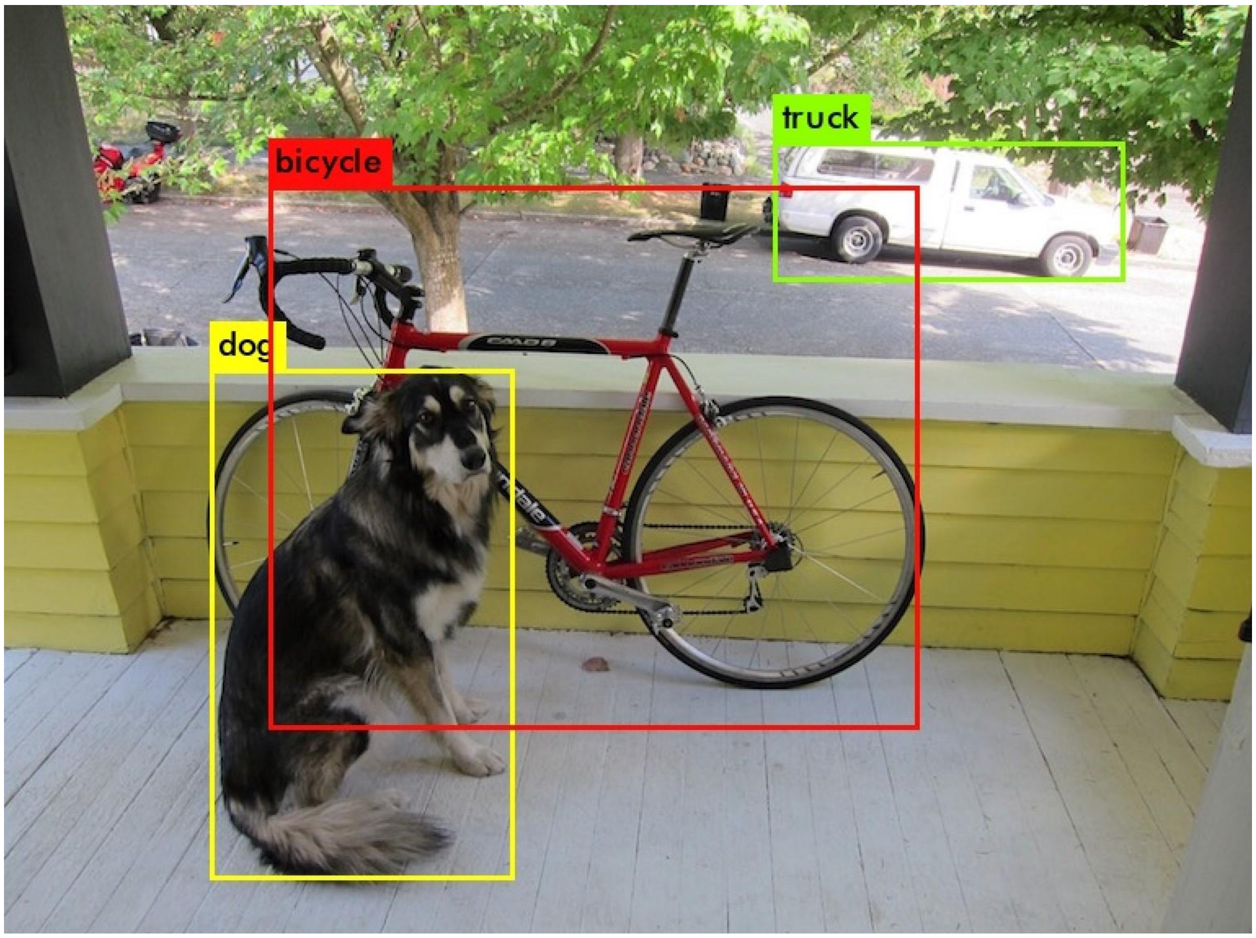


**Figure 6.** YOLOv3 Multiple Object Detection

At the core of the YOLOv3 algorithm is a deep convolutional neural network known as DarkNet-53. This complex network architecture has been specifically designed to learn intricate features that enable accurate object detection. It builds upon the successes of its predecessors, YOLOv1 and YOLOv2, and offers improved performance and accuracy.

YOLOv3 leverages the power of Convolutional Neural Networks (CNNs) to perform real-time object detection. CNNs are highly effective in processing input images as structured arrays of data and recognizing patterns within them. YOLOv3's ability to process images rapidly while maintaining accuracy sets it apart from other object detection approaches.

One key advantage of YOLOv3 is its ability to consider the entire image as a whole during the detection process. Unlike traditional methods that divide an image into regions and process them individually, YOLOv3 takes a holistic approach. By examining the global context of the image, the model's predictions are informed by the relationships and dependencies between different objects within the scene. This global context contributes to the accuracy and robustness of the detection results.

In YOLOv3, the object detection process involves dividing the input image into a grid and predicting bounding boxes and class probabilities for each grid cell. These predictions are weighted by their corresponding probabilities, resulting in accurate and reliable object detection. The model is trained to assign a single predictor to each class, responsible for predicting the bounding box with the highest Intersection over Union (IoU) value compared to the ground truth. This specialization within the predictions enhances the precision and quality of the detected objects.

To optimize the performance of YOLOv3, a loss function based on sum-squared error is employed during the training phase. This loss function accounts for classification, localization, and confidence, allowing the model to continuously improve its accuracy and detection capabilities.

Overall, YOLOv3 is a highly advanced and efficient object detection algorithm that has pushed the boundaries of real-time computer vision applications. Its ability to process images rapidly while maintaining high accuracy makes it a valuable tool in various domains. The combination of speed, accuracy, and versatility positions YOLOv3 as a leading choice for developers and researchers working on object detection tasks.

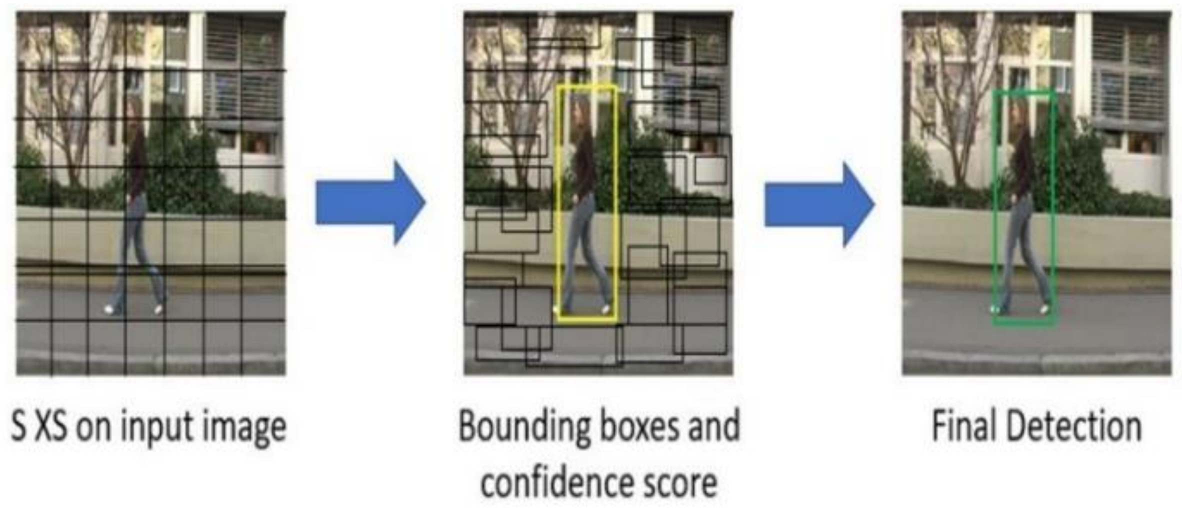


**Figure 7.**Human Detection Using YOLOv3

# 5. DESIGN REQUIREMENT ENGINEERING

**<u>Concept of uml :</u>**

UML is a famous language for four main things which are specifying, visualising, constructing, and documenting software system artefacts. The acronym UML refers to the Unified Modeling Language. UML is distinct from other popular programming languages like Cpp, Java, COBOL, and others. UML is a diagramming language for creating software blueprints. There are several objectives for designing UML, the most important of which is to define a general-purpose modelling language that all modellers can use and that is also simple to comprehend and apply.

**UML DIAGRAMS:**

## <u>5.1 Use case Diagram:</u>

A use case diagram is a form of behavioral diagram specified by and derived from a Use-case analysis in the Unified Modeling Language (UML). Its goal is to provide a graphical picture of a system's functionality in terms of actors, goals (represented as use cases), and any dependencies between those cases. A use case diagram's main aim is to indicate which actor performs which system functions.

The roles of the system's actors can be illustrated.

Figure 8 shows the use case diagram of our system which describes the interaction between actors which are the ones who will interact with the subjects. In our project, there is mainly one actor involved in it namely the user(who uses our model).

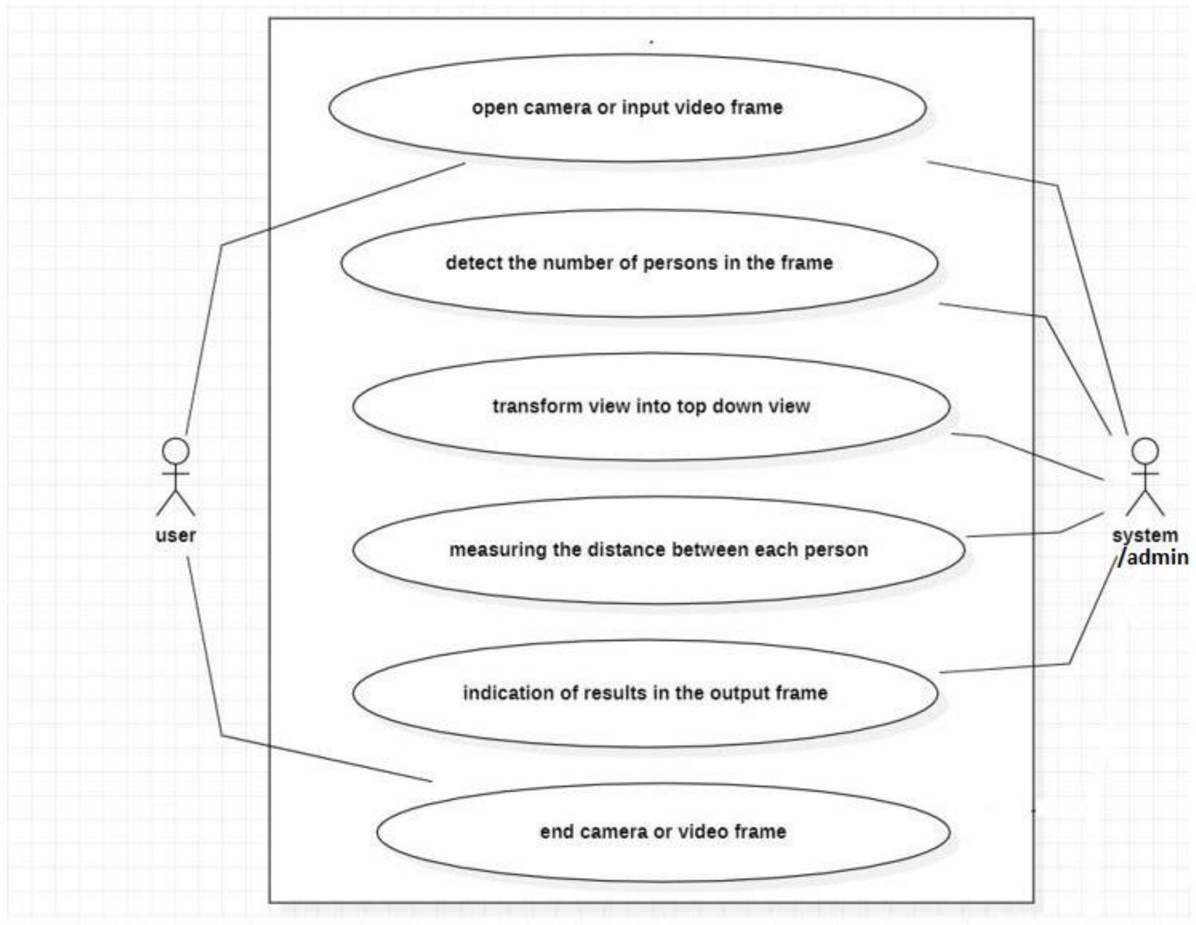

open camera or input video frame
detect the number of persons in the frame
transform view into top down view
measuring the distance between each person
indication of results in the output frame
end camera or video frame
user
system /admin

**Figure 8.** Use Case Diagram

### 5.2 Activity diagram :

Another major behavioral diagram used in uml diagrams to illustrate the dynamic characteristics of the system is the activity diagram. An activity diagram is a more complex version of a flow chart that depicts the flow of information from one activity to the next.

It depicts the system's dynamic behavior. The other four diagrams illustrate the message flow from one object to the next, but the activity diagram shows the message flow from one action to the next.

Figure 9 shows the Activity diagram of our system which helps the model the workflow of a system from one activity to another involving different components or states like initial, final & activity states, etc. It represents the execution of the system.

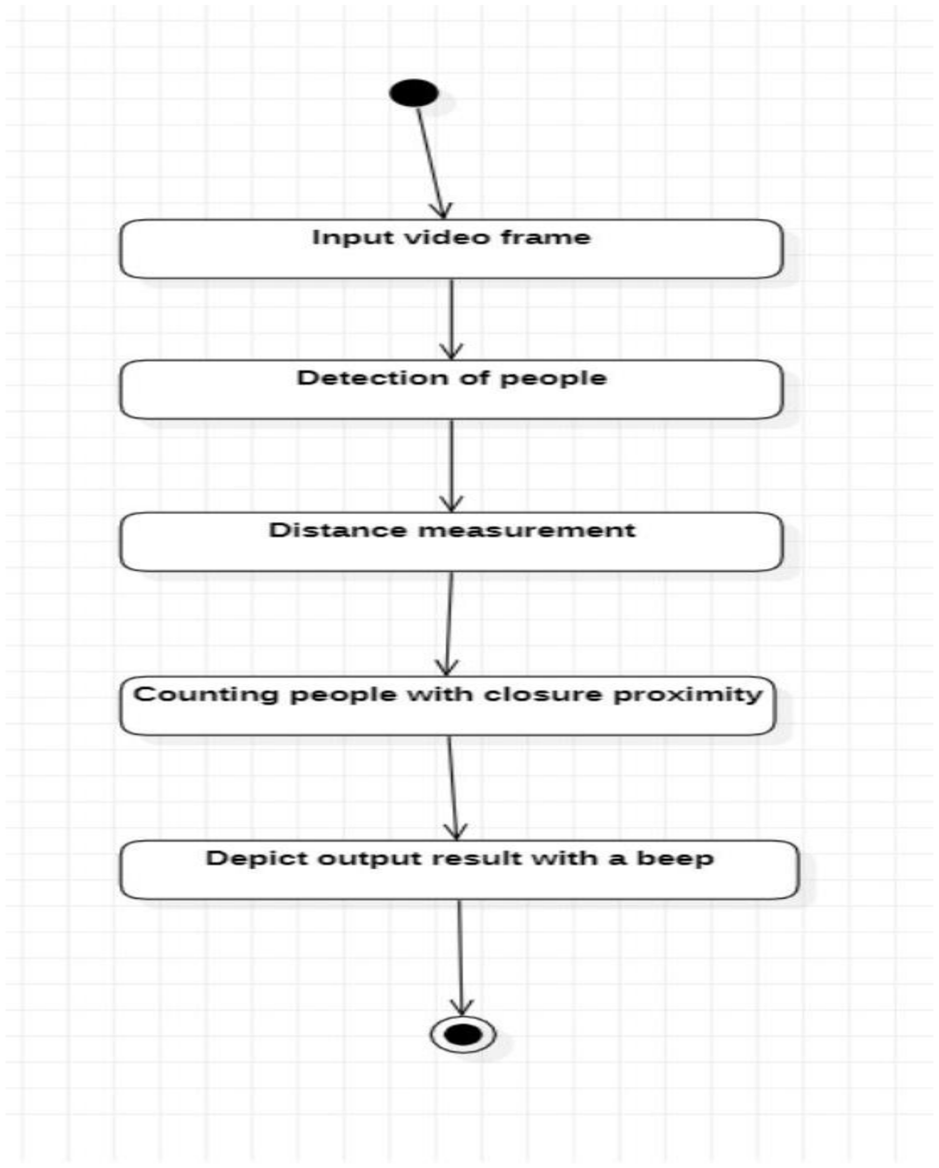

Input video frame
Detection of people
Distance measurement
Counting people with closure proximity
Depict output result with a beep

**Figure 9.** Activity Diagram

## 5.3 Sequence Diagram :

A sequence diagram is a type of UML diagram that shows the interactions and message flow between objects or components in a system. It represents the chronological order of interactions and how objects collaborate over time. Objects are depicted as boxes, and their interactions are shown using arrows or dashed lines called messages. Messages can be synchronous or asynchronous. sequence diagrams also allow for conditions, loops, and parallel execution through combined fragments. They are used to model and analyze system behavior in software development.

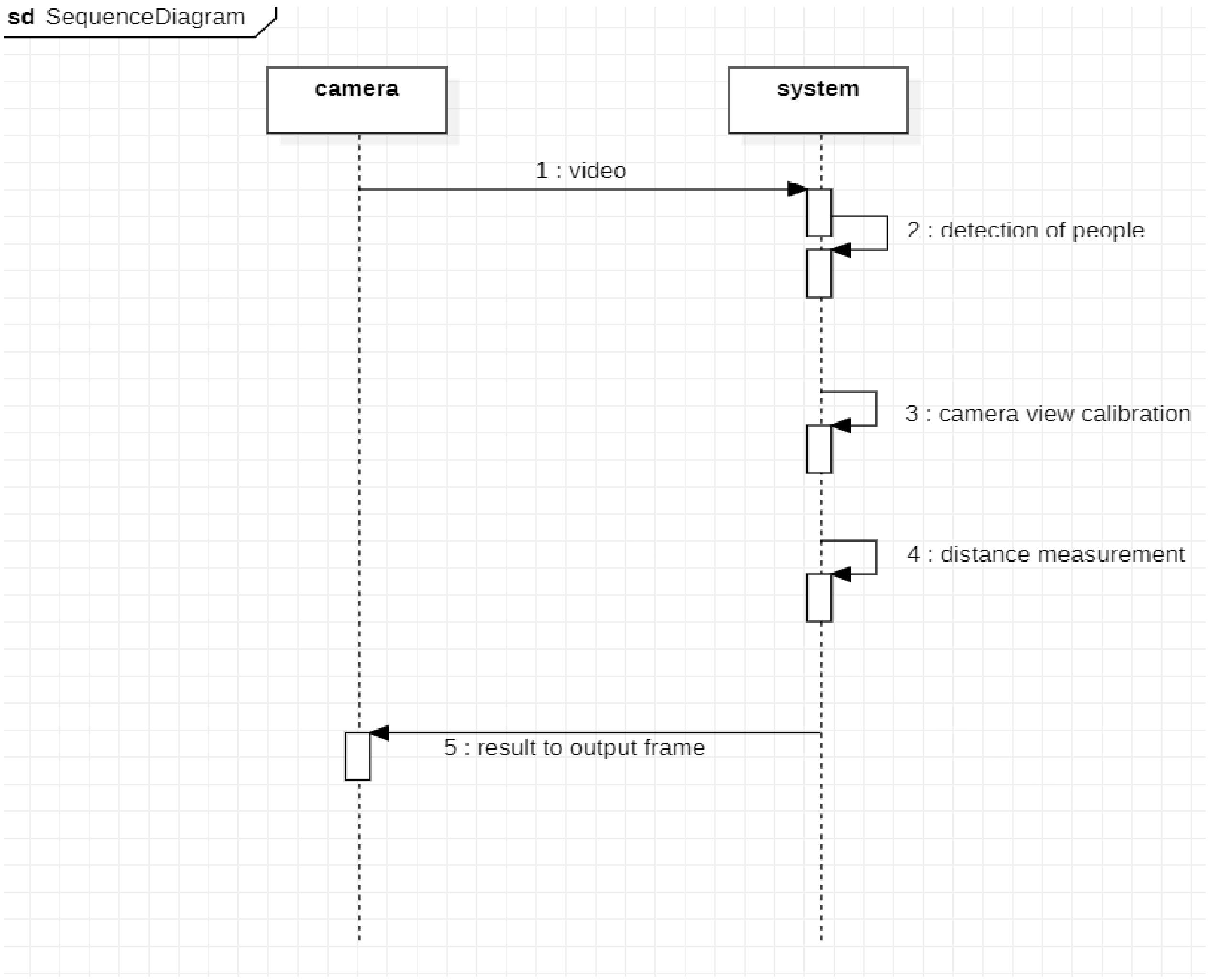


**Figure 10.** Sequence Diagram

## <u>5.4 Data Flow Diagram:</u>

A data flow diagram (DFD) is a visual representation of how data moves within a system. It helps understand processes, inputs, outputs, and data storage. DFDs were introduced by Larry Constantine for system requirements modeling. DFDs use circles or rectangles to represent processes that transform data. Arrows show data flow between processes, or between processes and external entities. External entities are rectangles representing data sources or destinations. Data stores are rectangles with parallel lines, indicating data storage. DFDs have different levels, with Level 0 giving an overview and subsequent levels providing more detail. Decomposing DFDs breaks down complex systems into manageable parts. DFDs visualize and analyze data flow, aid system understanding, and support system development and improvement.

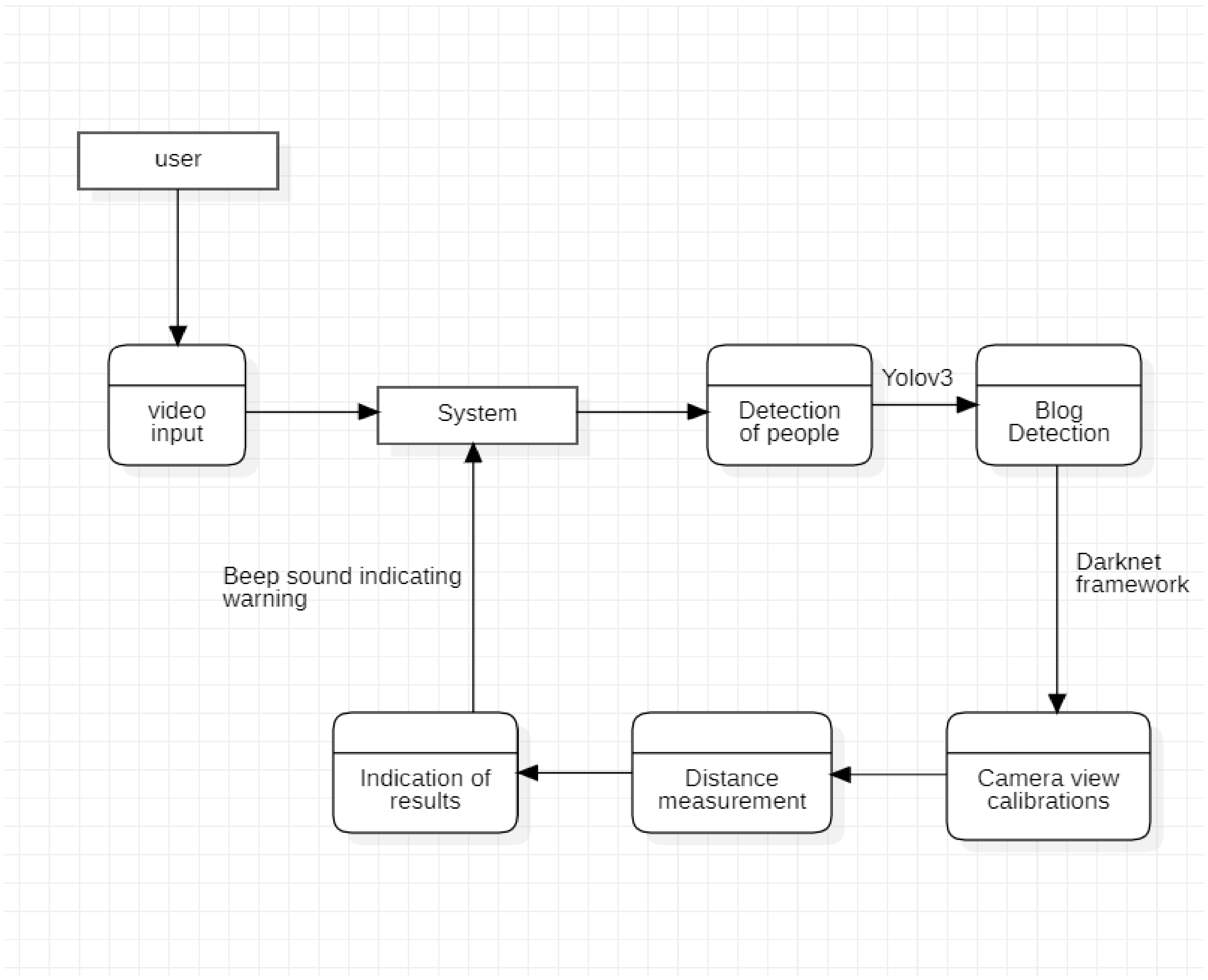


**Figure 11.** Data Flow Diagram

## 5.4 Design Architecture:

System designers and developers can use a basic architecture diagram (UML) to depict the high-level structure of their system or application to verify it satisfies the needs of their users. It can also be used to describe patterns that appear in the design. It depicts a method that individuals use to abstract the broad framework of a software system and construct restrictions, relationships, and boundaries between components. It provides a comprehensive perspective of the software system's physical deployment and evolution plan. This diagram is very useful for developers, designers.

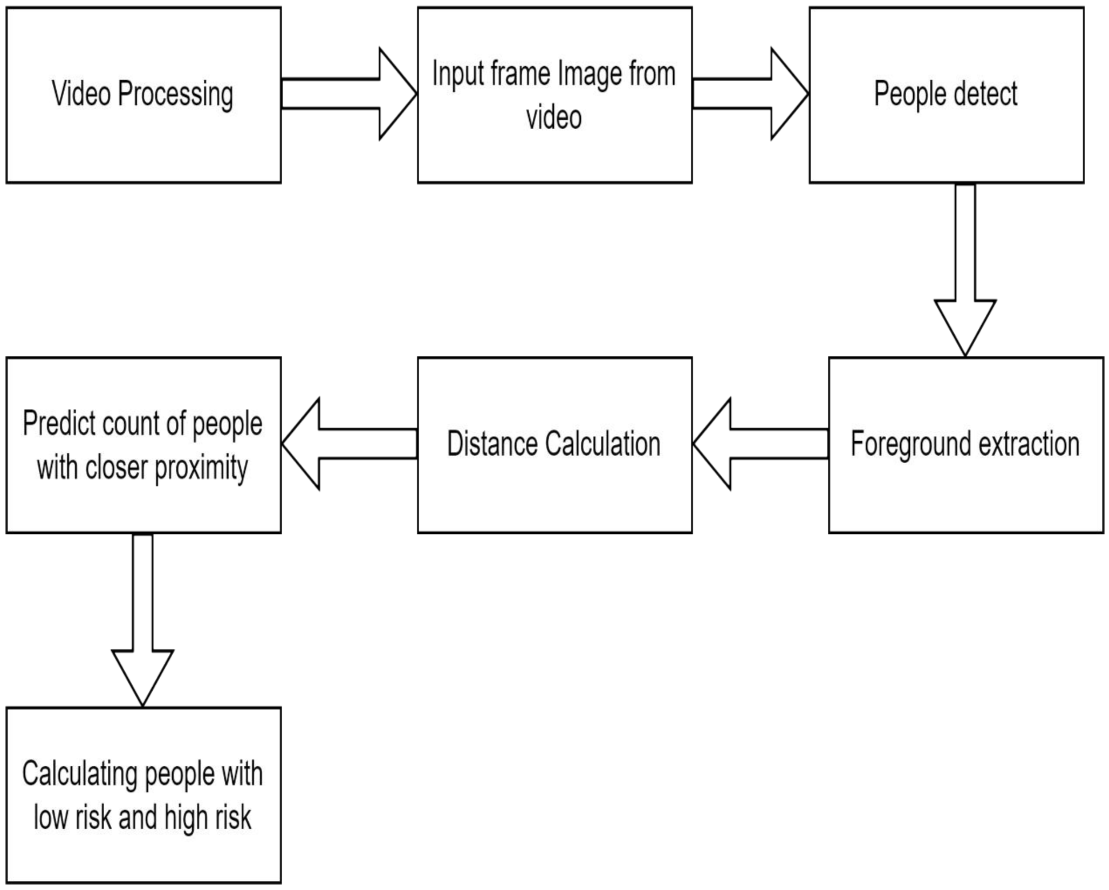

Video Processing
Input frame Image from video
People detect
Foreground extraction
Distance Calculation
Predict count of people with closer proximity
Calculating people with low risk and high risk

**Figure 12.** Design Architecture

# 6. IMPLEMENTATION

## 6.1 Modules

**To implement this project we have designed the following modules**

### 6.1.1 Processing Video Frame :

We utilized the OpenCV library to process our video. OpenCV is a powerful open-source library that offers a wide range of tools for image and video processing tasks. When working with videos, it's important to understand that video streaming is a discrete process rather than a continuous one. This means that a video is essentially a sequence of individual frames. Each frame can be represented as a matrix of pixels, with the size of the matrix determined by the dimensions of the picture. The pixels in each frame can be represented in various ways, such as color intensity, depending on the chosen color model, whether it's grayscale, RGB, or even multispectrum. OpenCV provides us with the necessary functionalities to manipulate and analyze these frames in order to perform various image and video processing operations.

**Capture Video from Camera :**

In order to process our video, we utilized the OpenCV library. OpenCV is an open-source library that offers a comprehensive set of tools for performing various image and video processing tasks. It's crucial to recognize that video streaming operates in a discrete manner, rather than as a continuous process. This means that videos are essentially made up of a sequence of frames. Each frame represents an individual image and can be visualized as an array of pixels with a specific size (m x n), where m and n represent the dimensions of the picture. The pixels within each frame can be represented using different color models, such as grayscale, RGB, or even multispectrum, depending on the specific requirements. OpenCV provides a straightforward interface for capturing live video streams using a camera. By initializing a VideoCapture object, we have the ability to define either the device index or the name of a video file from which we want to capture the desired video. For instance, passing 0 or -1 as the argument selects the first available camera while passing 1 would select the second camera, and so on. Once the video is captured, it can be processed frame-by-frame using the functionalities provided by the OpenCV library

**Giving Video Feed as Input :**

Playing video from a file is the same as capturing it from the camera, the only change is to replace the camera index with a video file name.

**Steps to Capture a Video :**

• Use cv2.VideoCapture() to get a video capture object for the camera.

• Set up an infinite while loop and use the read() method to read the frames using the above created object.

• Use cv2.imshow() method to show the frames in the video.

### 6.1.2 Detecting People:

In our project, we use the YOLOv3 algorithm to detect people in video frames. YOLOv3 uses the complex DarkNet-53 as its architectural model. The YOLO model has been recognized as one of the most effective tools when using deep learning for object detection, with its fast performance making it suitable for applications over time. In our custom work, we use YOLO's unique design for human testing. The YOLO algorithm performs object detection by analyzing the input image and simultaneously estimates the common box (tx, ty, tw, th), confidence factor, and probability of multiple tag lists (P1, P2, ...)..., Pc) The YOLO model was trained on the COCO dataset containing 80 different characters, either human or human. In our project, we focused on extracting from the YOLO search only the relevance of the box, the reliability of the items, and the class labels corresponding to the individuals to identify the right person.

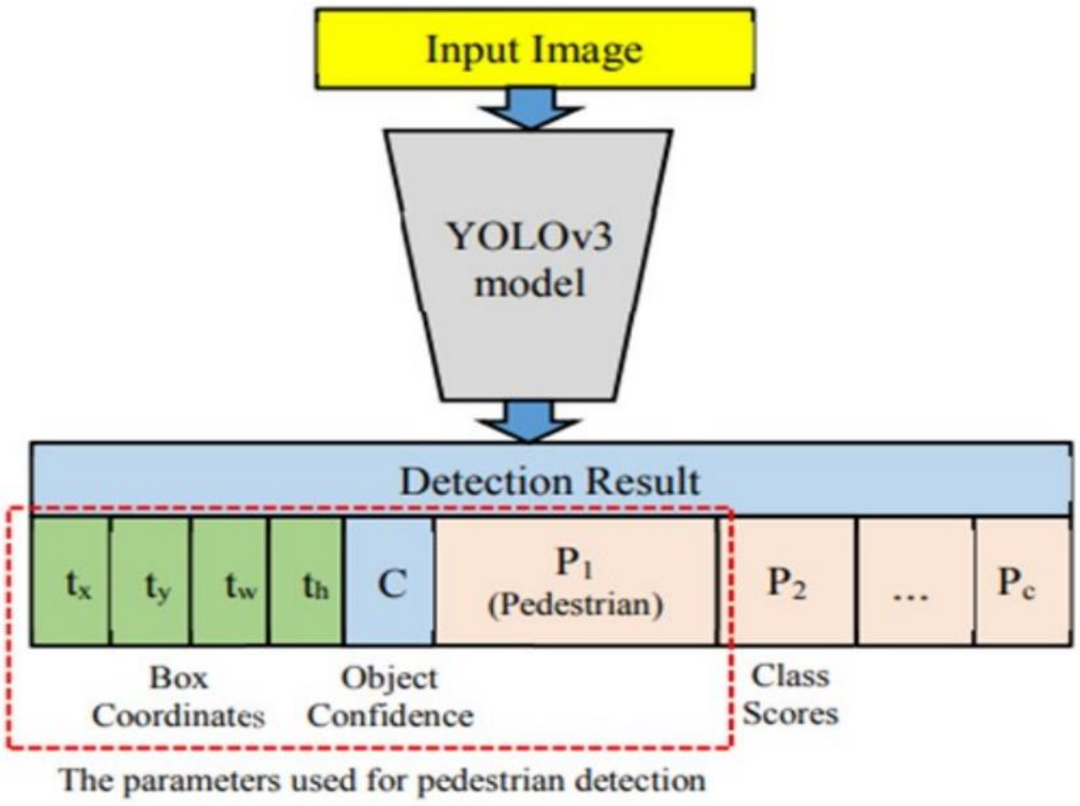
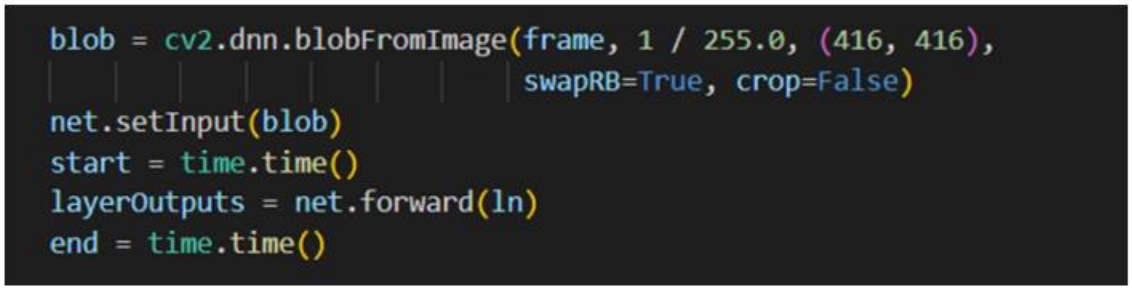


**Figure 13.** YOLO Model Applied for Pedestrian Detection

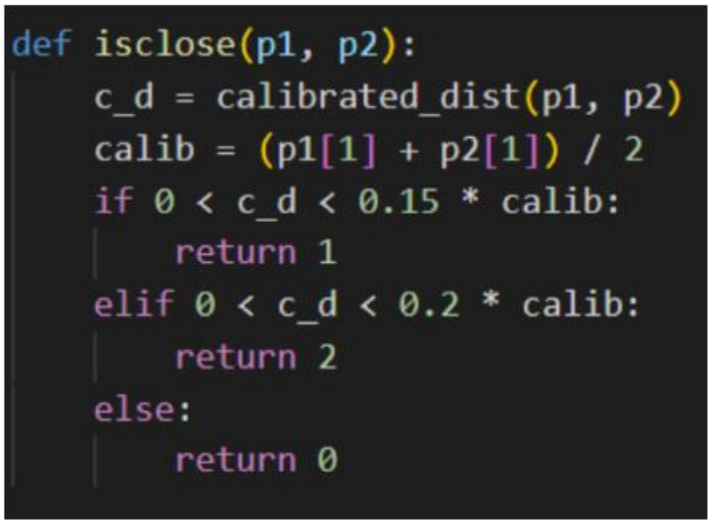


```
blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
                             swapRB=True, crop=False)
net.setInput(blob)
start = time.time()
layerOutputs = net.forward(ln)
end = time.time()
```

**Figure 14.** Blob Detection

**<u>6.1.3 Distance Calculation:</u>**

The distance between individuals is determined using their center of gravity derived from box sensing. These centroids represent the points in the middle of each individual's bounding box. Additionally, we added an angle that helps calculate the distance between individuals at different points. By taking the angles into account, we can calculate the distance between people even if they are not looking directly at each other. This angle will help to adjust the distance according to the direction or angle of the neighboring people.

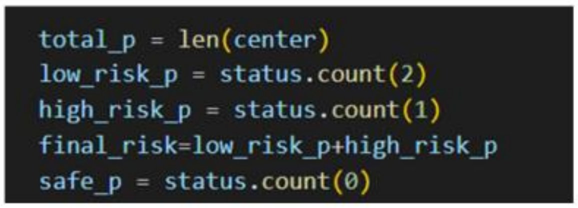


```
def isclose(p1, p2):
    c_d = calibrated_dist(p1, p2)
    calib = (p1[1] + p2[1]) / 2
    if 0 < c_d < 0.15 * calib:
        return 1
    elif 0 < c_d < 0.2 * calib:
        return 2
    else:
        return 0
```

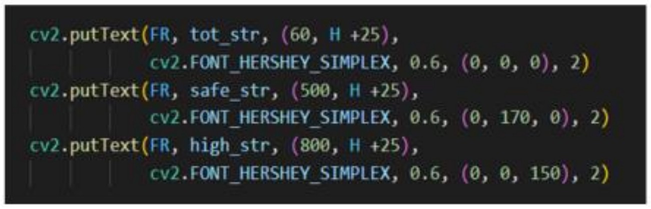
**Figure 15.** Calibrating Distance

**<u>6.1.4 Declaration of Results:</u>**

The system detects people and measures the distance between them. Depending on how close they are, the checkboxes change color to show their close relatives. If people are very close to each other, a red box is shown, otherwise, a green box is shown. In addition to all detected persons, the system also monitors people in high-risk and safe places. If the number of people at risk is more than half of the total, an alarm is given and an audible warning is given.

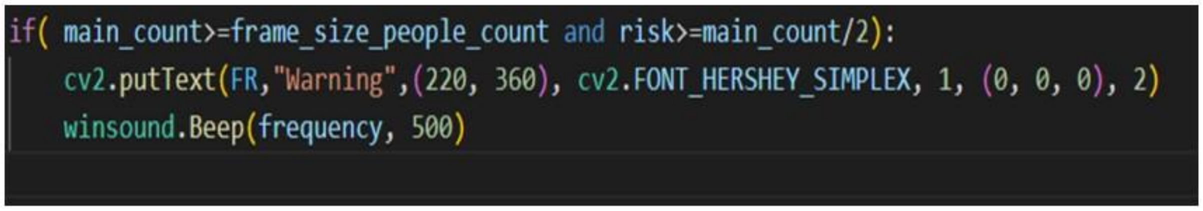


```
total_p = len(center)
low_risk_p = status.count(2)
high_risk_p = status.count(1)
final_risk=low_risk_p+high_risk_p
safe_p = status.count(0)
```



```
cv2.putText(FR, tot_str, (60, H +25),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0), 2)
cv2.putText(FR, safe_str, (500, H +25),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 170, 0), 2)
cv2.putText(FR, high_str, (800, H +25),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 150), 2)
```

**Figure 16.** Displays total, safe, and high-risk individual counts.



```
if( main_count>=frame_size_people_count and risk>=main_count/2):
    cv2.putText(FR,"Warning",(220, 360), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)
    winsound.Beep(frequency, 500)
```

**Figure 17.** Raising Alert

## Modules used:

The code utilizes several Python modules to enable different functionalities:

### Time

This module provides functions for working with time-related operations. It is used in the code to measure the time taken for certain operations.

### Math

The math module offers mathematical functions and operations. In this code, it is used for trigonometric calculations.

### cv2 (OpenCV)

OpenCV is a popular computer vision tool for image and video processing. It provides various functions and algorithms for tasks such as object detection, image processing, and image drawing. This code reads video frames using OpenCV, detects objects using the YOLOv3 model, draws bounding boxes and lines, and parses display frames.

### NumPy (np)

NumPy is a powerful library for numerical operations and array manipulation. It is used in this code to work with arrays and perform calculations related to object detection and proximity analysis.

**Winsound**

The winsound module is specific to Windows systems and allows playing sound alerts. It is utilized in this code to generate a warning beep sound when the risk level exceeds a certain threshold.

By importing and utilizing these modules, the code leverages its functionalities to perform various operations, such as time measurement, mathematical calculations, computer vision tasks, numerical operations, and audio alerts.

## <u>**6.2 DATASET**</u>

https://www.kaggle.com/datasets/valentynsichkar/yolo-coco-data

• The data used in the code is the COCO (Common Objects in Context) dataset.

• The COCO file is commonly used for object detection and  segmentation tasks in computer vision.

• It has many images with 80 different objects, including "People".

• Special files in the COCO dataset used in the script are coco.names, yolov3.Weight and yolov3.cfg.

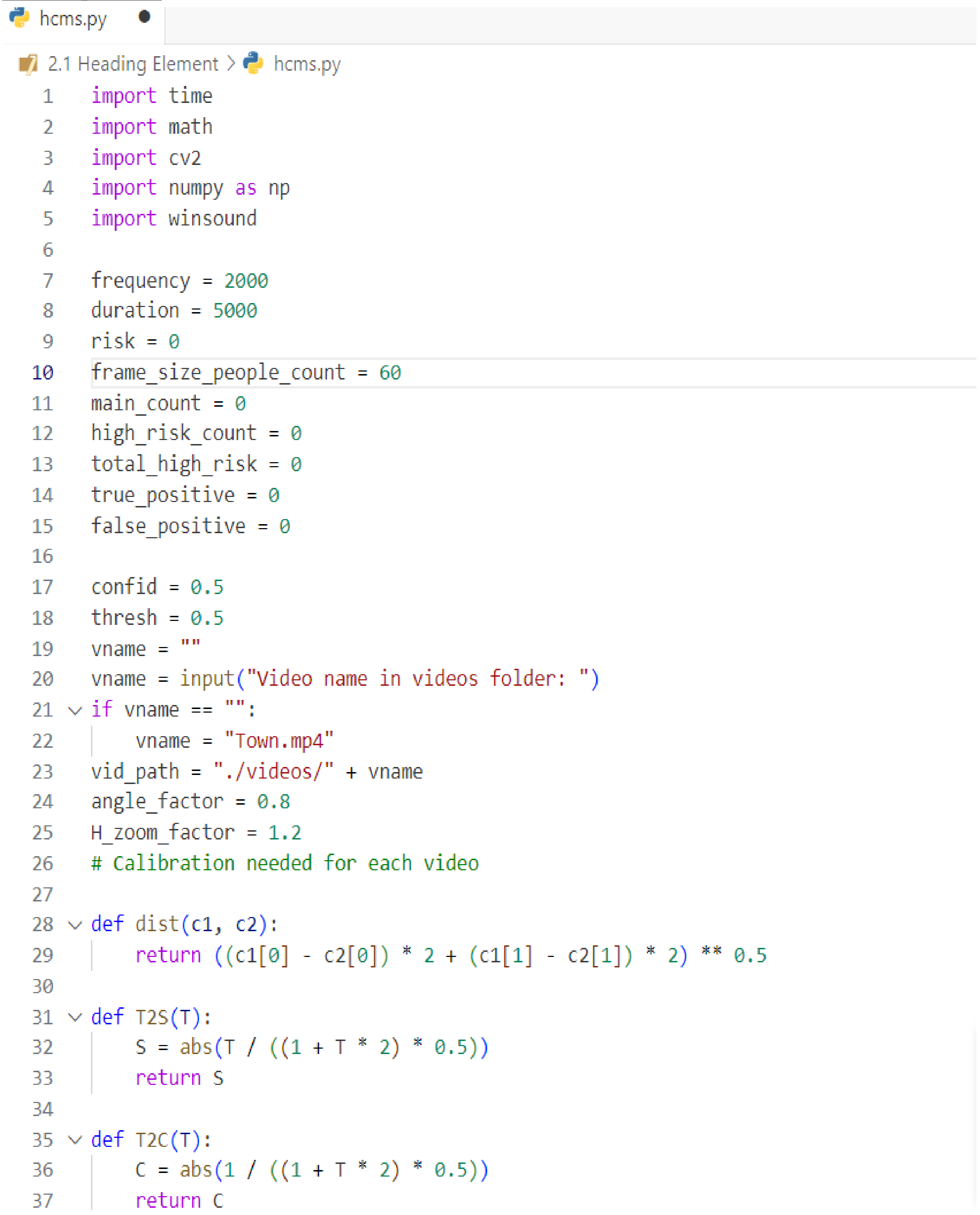
• The coco.names file contains the names of the  classes in the COCO dataset, including "person".

•  yolov3.Weighted data includes previous weights for the YOLOv3 model trained on the COCO dataset.

• The Yolov3.cfg file is the configuration file that defines the architecture and settings of the YOLOv3 model.

• This code uses a pre-trained YOLOv3 model with its weights and settings to detect objects in real time, including human detection.

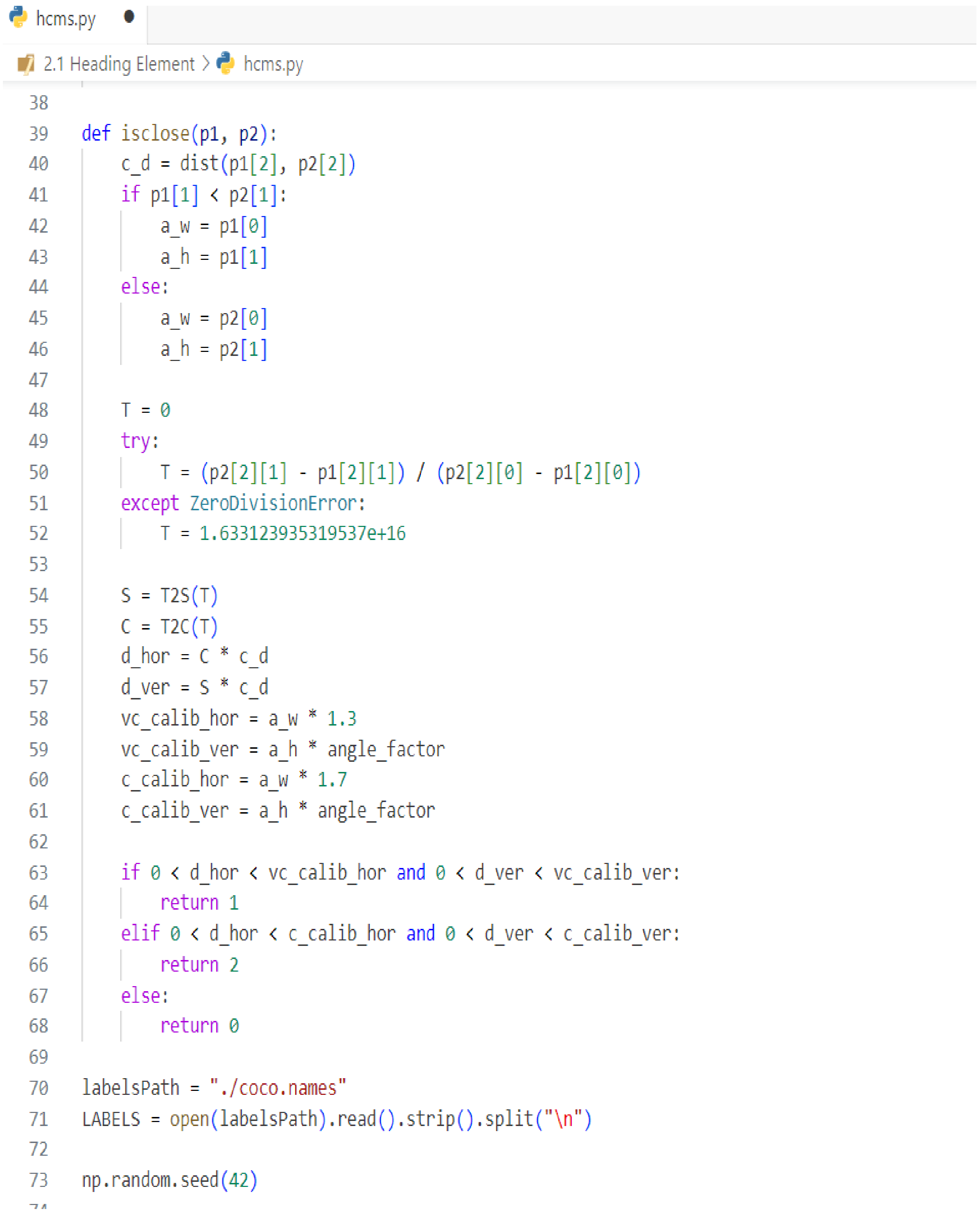
## 6.3 Sample code



hcms.py ●

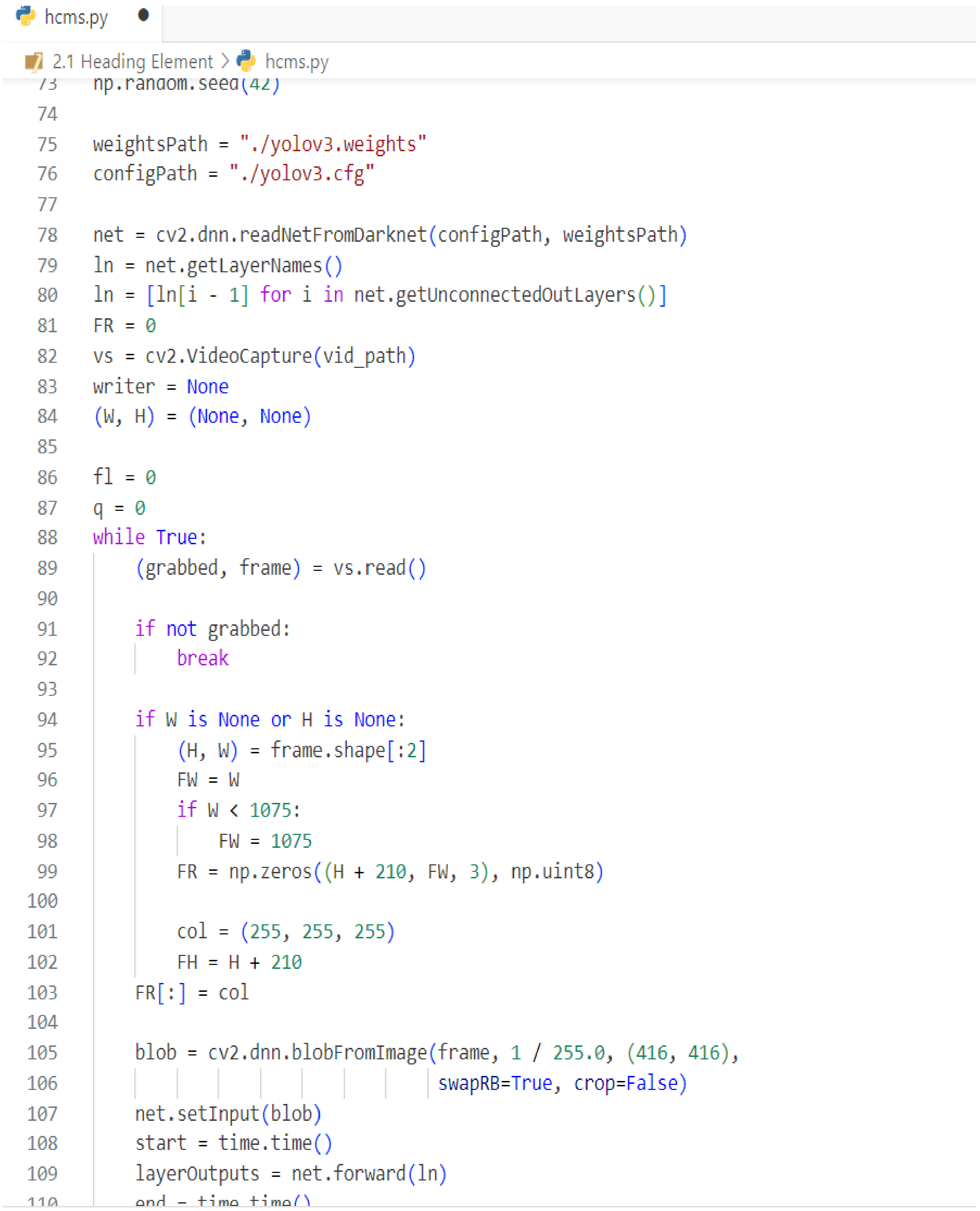
2.1 Heading Element > hcms.py

```python
import time
import math
import cv2
import numpy as np
import winsound

frequency = 2000
duration = 5000
risk = 0
frame_size_people_count = 60
main_count = 0
high_risk_count = 0
total_high_risk = 0
true_positive = 0
false_positive = 0

confid = 0.5
thresh = 0.5
vname = ""
vname = input("Video name in videos folder: ")
if vname == "":
    vname = "Town.mp4"
vid_path = "./videos/" + vname
angle_factor = 0.8
H_zoom_factor = 1.2
# Calibration needed for each video

def dist(c1, c2):
    return ((c1[0] - c2[0]) * 2 + (c1[1] - c2[1]) * 2) ** 0.5

def T2S(T):
    S = abs(T / ((1 + T * 2) * 0.5))
    return S

def T2C(T):
    C = abs(1 / ((1 + T * 2) * 0.5))
    return C
```


hcms.py ●

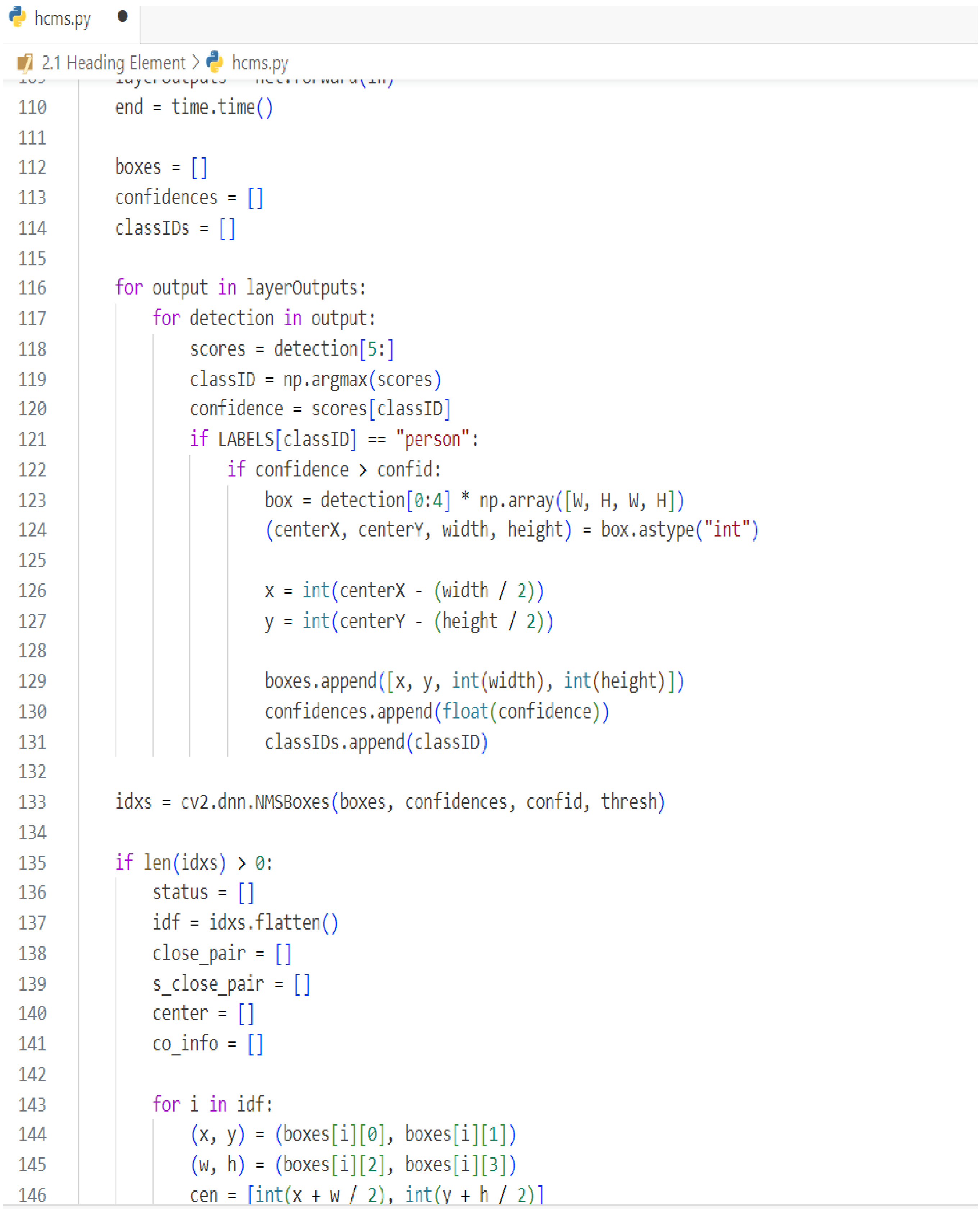
2.1 Heading Element > hcms.py

```python

def isclose(p1, p2):
    c_d = dist(p1[2], p2[2])
    if p1[1] < p2[1]:
        a_w = p1[0]
        a_h = p1[1]
    else:
        a_w = p2[0]
        a_h = p2[1]

    T = 0
    try:
        T = (p2[2][1] - p1[2][1]) / (p2[2][0] - p1[2][0])
    except ZeroDivisionError:
        T = 1.633123935319537e+16

    S = T2S(T)
    C = T2C(T)
    d_hor = C * c_d
    d_ver = S * c_d
    vc_calib_hor = a_w * 1.3
    vc_calib_ver = a_h * angle_factor
    c_calib_hor = a_w * 1.7
    c_calib_ver = a_h * angle_factor

    if 0 < d_hor < vc_calib_hor and 0 < d_ver < vc_calib_ver:
        return 1
    elif 0 < d_hor < c_calib_hor and 0 < d_ver < c_calib_ver:
        return 2
    else:
        return 0

labelsPath = "./coco.names"
LABELS = open(labelsPath).read().strip().split("\n")

np.random.seed(42)
```

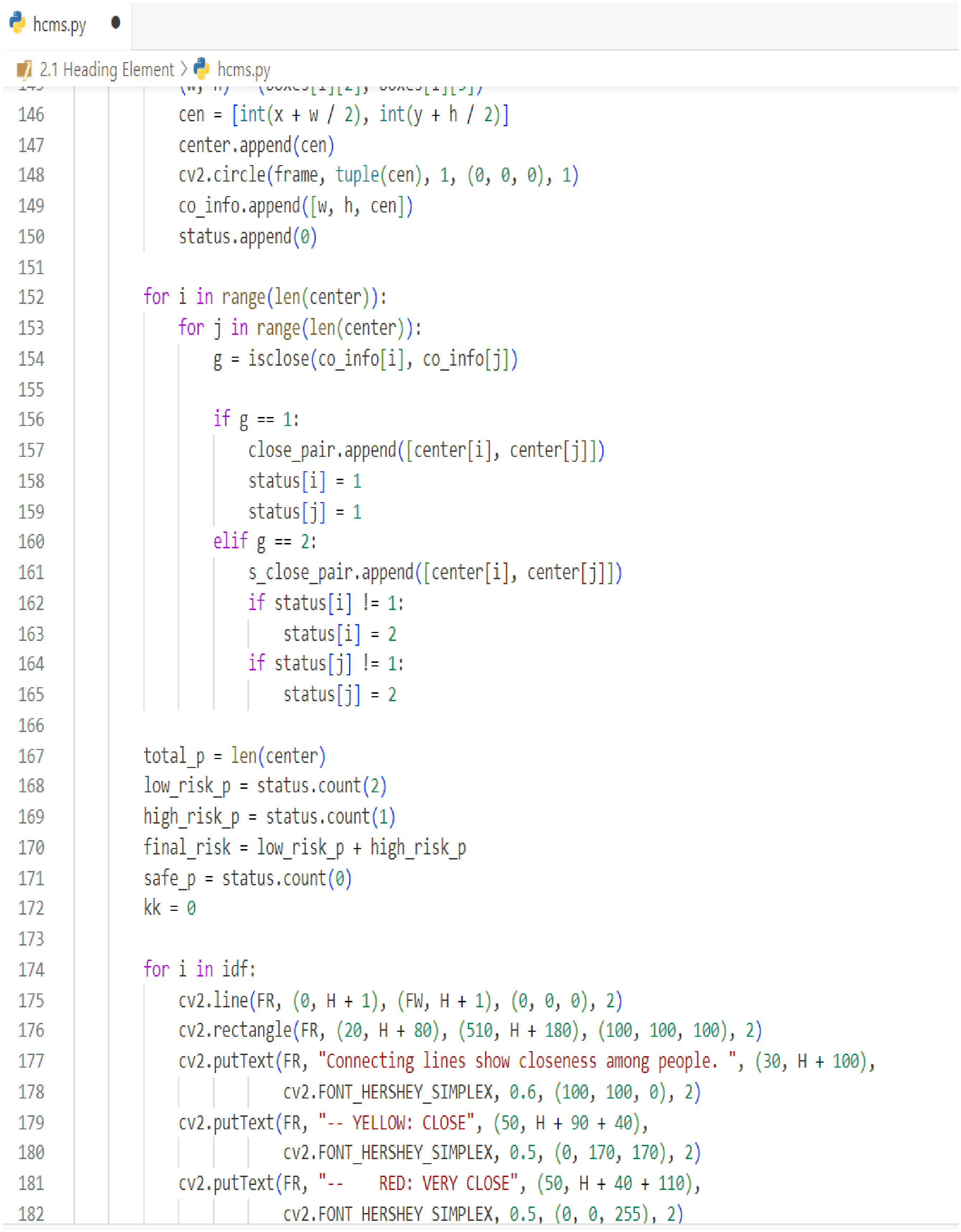

hcms.py ●

2.1 Heading Element > hcms.py

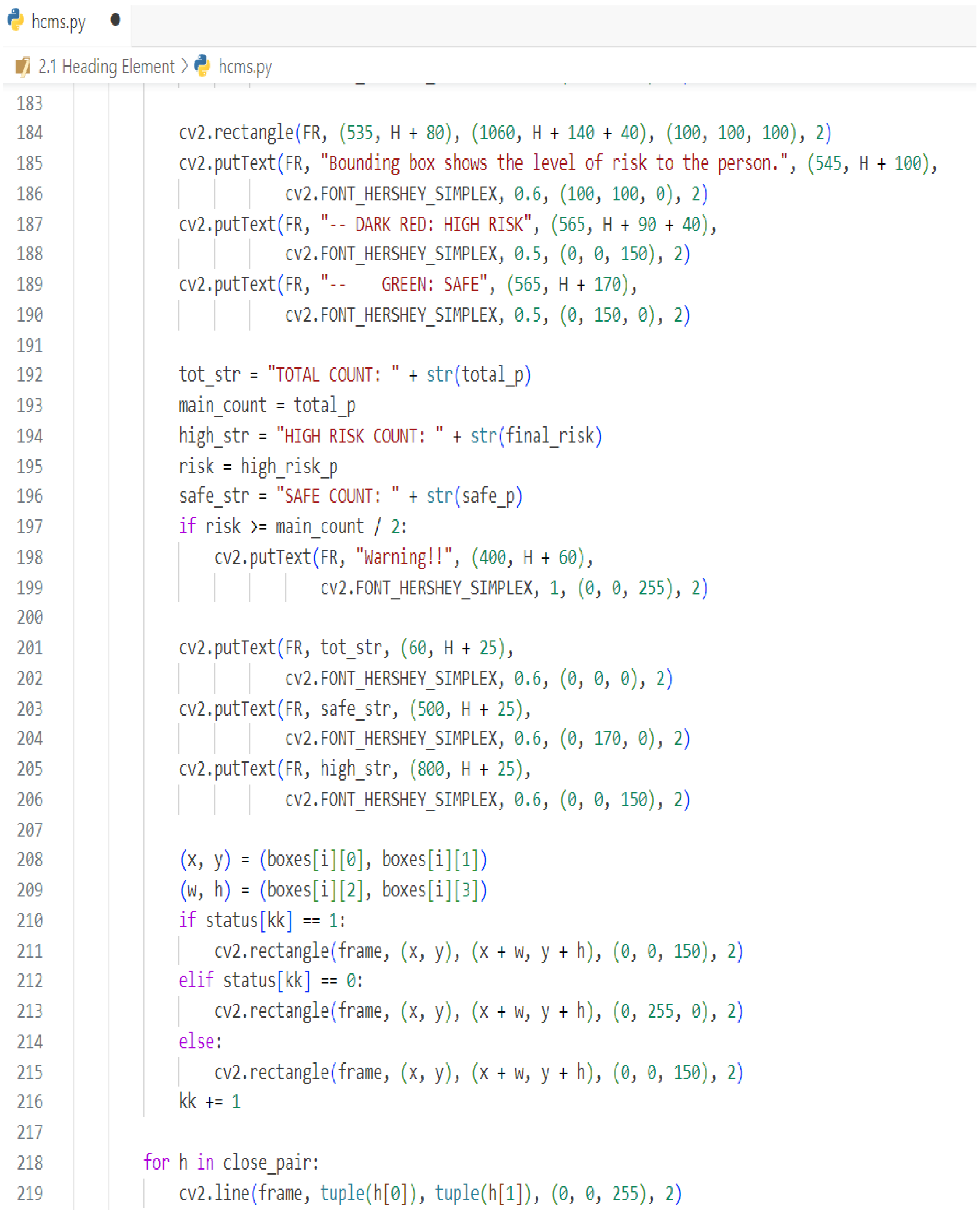
```
np.random.seed(42)

weightsPath = "./yolov3.weights"
configPath = "./yolov3.cfg"

net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
ln = net.getLayerNames()
ln = [ln[i - 1] for i in net.getUnconnectedOutLayers()]
FR = 0
vs = cv2.VideoCapture(vid_path)
writer = None
(W, H) = (None, None)

fl = 0
q = 0
while True:
    (grabbed, frame) = vs.read()

    if not grabbed:
        break

    if W is None or H is None:
        (H, W) = frame.shape[:2]
        FW = W
        if W < 1075:
            FW = 1075
        FR = np.zeros((H + 210, FW, 3), np.uint8)

        col = (255, 255, 255)
        FH = H + 210
    FR[:] = col

    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
                                 swapRB=True, crop=False)
    net.setInput(blob)
    start = time.time()
    layerOutputs = net.forward(ln)
    end = time.time()
```



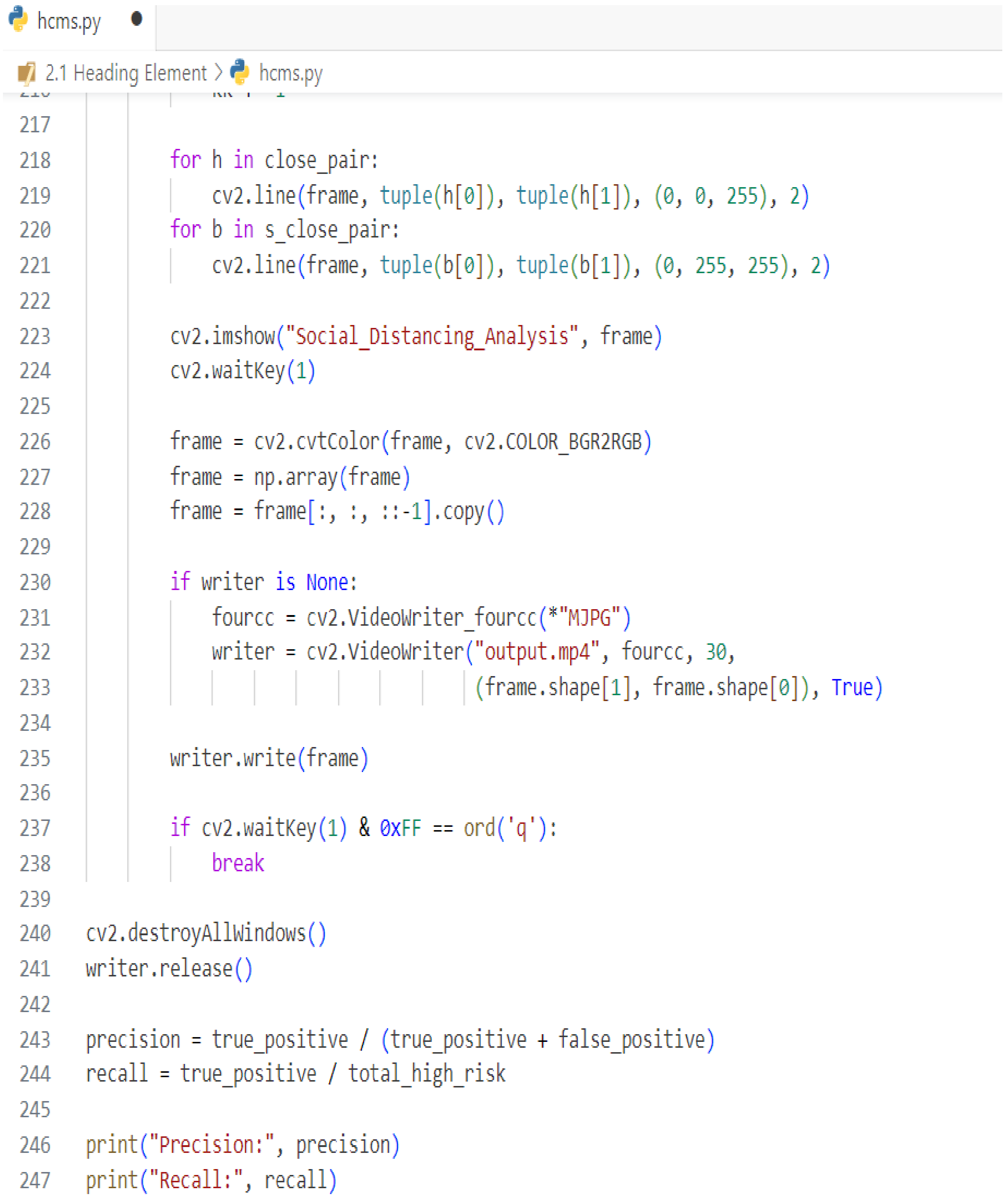
```python
    end = time.time()

    boxes = []
    confidences = []
    classIDs = []

    for output in layerOutputs:
        for detection in output:
            scores = detection[5:]
            classID = np.argmax(scores)
            confidence = scores[classID]
            if LABELS[classID] == "person":
                if confidence > confid:
                    box = detection[0:4] * np.array([W, H, W, H])
                    (centerX, centerY, width, height) = box.astype("int")

                    x = int(centerX - (width / 2))
                    y = int(centerY - (height / 2))

                    boxes.append([x, y, int(width), int(height)])
                    confidences.append(float(confidence))
                    classIDs.append(classID)

    idxs = cv2.dnn.NMSBoxes(boxes, confidences, confid, thresh)

    if len(idxs) > 0:
        status = []
        idf = idxs.flatten()
        close_pair = []
        s_close_pair = []
        center = []
        co_info = []

        for i in idf:
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])
            cen = [int(x + w / 2), int(y + h / 2)]
```



```
            cen = [int(x + w / 2), int(y + h / 2)]
            center.append(cen)
            cv2.circle(frame, tuple(cen), 1, (0, 0, 0), 1)
            co_info.append([w, h, cen])
            status.append(0)

        for i in range(len(center)):
            for j in range(len(center)):
                g = isclose(co_info[i], co_info[j])

                if g == 1:
                    close_pair.append([center[i], center[j]])
                    status[i] = 1
                    status[j] = 1
                elif g == 2:
                    s_close_pair.append([center[i], center[j]])
                    if status[i] != 1:
                        status[i] = 2
                    if status[j] != 1:
                        status[j] = 2

        total_p = len(center)
        low_risk_p = status.count(2)
        high_risk_p = status.count(1)
        final_risk = low_risk_p + high_risk_p
        safe_p = status.count(0)
        kk = 0

        for i in idf:
            cv2.line(FR, (0, H + 1), (FW, H + 1), (0, 0, 0), 2)
            cv2.rectangle(FR, (20, H + 80), (510, H + 180), (100, 100, 100), 2)
            cv2.putText(FR, "Connecting lines show closeness among people. ", (30, H + 100),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 100, 0), 2)
            cv2.putText(FR, "-- YELLOW: CLOSE", (50, H + 90 + 40),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 170, 170), 2)
            cv2.putText(FR, "--    RED: VERY CLOSE", (50, H + 40 + 110),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
```



```
            cv2.rectangle(FR, (535, H + 80), (1060, H + 140 + 40), (100, 100, 100), 2)
            cv2.putText(FR, "Bounding box shows the level of risk to the person.", (545, H + 100),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 100, 0), 2)
            cv2.putText(FR, "-- DARK RED: HIGH RISK", (565, H + 90 + 40),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 150), 2)
            cv2.putText(FR, "--    GREEN: SAFE", (565, H + 170),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 150, 0), 2)

            tot_str = "TOTAL COUNT: " + str(total_p)
            main_count = total_p
            high_str = "HIGH RISK COUNT: " + str(final_risk)
            risk = high_risk_p
            safe_str = "SAFE COUNT: " + str(safe_p)
            if risk >= main_count / 2:
                cv2.putText(FR, "Warning!!", (400, H + 60),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

            cv2.putText(FR, tot_str, (60, H + 25),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0), 2)
            cv2.putText(FR, safe_str, (500, H + 25),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 170, 0), 2)
            cv2.putText(FR, high_str, (800, H + 25),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 150), 2)

            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])
            if status[kk] == 1:
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 150), 2)
            elif status[kk] == 0:
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            else:
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 150), 2)
            kk += 1

        for h in close_pair:
            cv2.line(frame, tuple(h[0]), tuple(h[1]), (0, 0, 255), 2)
```



hcms.py

2.1 Heading Element > hcms.py

```python

        for h in close_pair:
            cv2.line(frame, tuple(h[0]), tuple(h[1]), (0, 0, 255), 2)
        for b in s_close_pair:
            cv2.line(frame, tuple(b[0]), tuple(b[1]), (0, 255, 255), 2)

        cv2.imshow("Social_Distancing_Analysis", frame)
        cv2.waitKey(1)

        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame = np.array(frame)
        frame = frame[:, :, ::-1].copy()

        if writer is None:
            fourcc = cv2.VideoWriter_fourcc(*"MJPG")
            writer = cv2.VideoWriter("output.mp4", fourcc, 30,
                                     (frame.shape[1], frame.shape[0]), True)

        writer.write(frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

cv2.destroyAllWindows()
writer.release()

precision = true_positive / (true_positive + false_positive)
recall = true_positive / total_high_risk

print("Precision:", precision)
print("Recall:", recall)
```

# 7. SOFTWARE TESTING

Software testing involves evaluating the correctness and quality of a software application or system by carefully examining its components and functions.It involves checking for bugs errors, and flaws to ensure that the software performs as intended. By conducting thorough testing. software testers provide an objective and unbiased assessment of the software's performance, reliability and adherence to specified requirements. Testing involves assessing each individual component and verifying that they meet the established criteria. This process also provides valuable insights to the client about the software's overall quality. Testing is essential because deploying untested software can pose risks and potentially lead to failures or undesirable consequences. Therefore, testing is a crucial step that must be pertormed before delivering software to end-user.

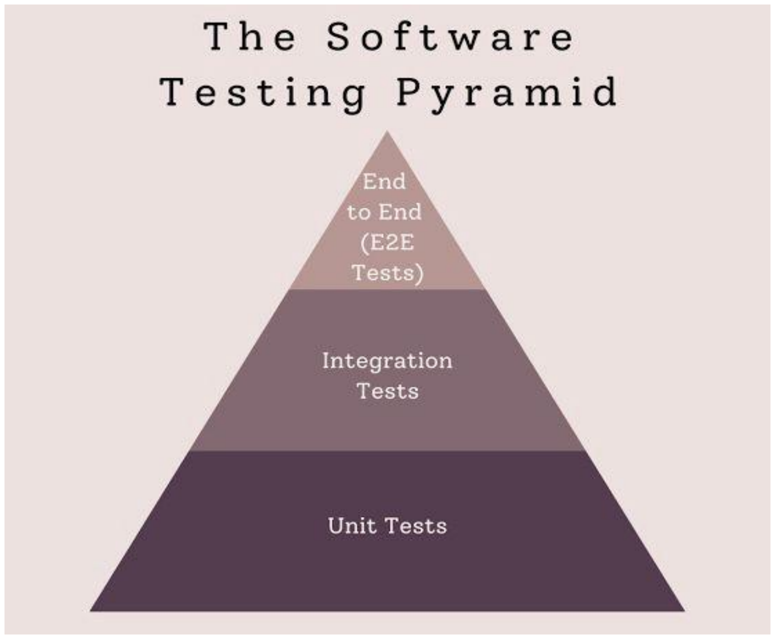


**Figure 18.**The Software Testing Pyramid

Software Testing is further divided into

- Black Box testing
- White Box testing
- Gray Box testing

## <u>7.1 Black Box Testing:-</u>

Black box testing is a software testing method that evaluates the functionality of an application without having access to its internal structure or implementation details. Testers who perform black box testing only have knowledge of the inputs, expected outputs, and observed behavior of the system.

The primary objective of black box testing is to verify whether the software meets the specified requirements and operates correctly from an end-user perspective. It aims to identify any errors, defects, or discrepancies between the expected behavior of the software and its actual behavior.

Black box testing offers several advantages. It is independent of the internal architecture, making it suitable for testing applications developed by different teams or external vendors. It also enables non-technical testers to conduct testing without requiring programming expertise. Additionally, black box testing can uncover issues that may be overlooked by programmers, providing valuable insights into the software's behavior and functionality.

However, black box testing has limitations. It relies on predefined inputs and expected outputs, which may not cover all possible scenarios and edge cases. There is also a risk of missing specific code paths or internal flaws that can only be identified through other testing approaches, such as white box testing or code review. Therefore, a combination of testing techniques is often necessary to achieve comprehensive test coverage and ensure the overall quality of the software.

## 7.2 Unit Testing:

Unit testing is a crucial phase of software testing that involves testing individual modules or units of source code to ensure their proper functioning. Developers perform unit testing in their own development environment to verify the isolated units of code. This process is also referred to as component testing or module testing.

## 7.3 Integration Testing:

Integration testing focuses on testing the connectivity and data transfer between multiple unit-tested modules. It ensures that these modules work together seamlessly when integrated. Integration testing is sometimes called string testing or Integration and Testing (I&T) testing. Different approaches, such as Top-Down, Bottom-Up, and Sandwich (a combination of Top-Down and Bottom-Up), can be used to perform integration testing.

## 7.4 Automation Testing:

Automation testing involves converting manual test cases into automated test scripts using automation tools or programming languages. This approach allows for faster test execution as human intervention is minimized. Automation testing is essential for improving testing efficiency and accuracy.

## 7.5 Non-Functional Testing:

Non-functional testing is a type of software testing that focuses on evaluating the qualities and attributes of a software system beyond its functional requirements. It involves testing non-functional aspects such as performance, reliability, usability, security, scalability, and more. Non-functional testing ensures that the software not only meets functional requirements but also provides a high-quality user experience. It helps identify potential issues and risks related to performance, security, and usability, allowing for necessary improvements before deployment.

## **<u>7.6 White Box Testing:-</u>**

White-box testing is a software testing methodology that delves into the internal structures and workings of an application, going beyond the assessment of its functionality alone. Unlike black-box testing, which focuses on external inputs and outputs, white-box testing takes an internal perspective to design test cases.

In white-box testing, testers have access to the internal components of the software, including the source code, architecture, and implementation details. This allows them to gain insights into how the system is structured, how data flows within it, and how different modules and components interact with each other. Armed with this knowledge, testers can create test cases that specifically target various paths, conditions, and variables within the system.

The primary objective of white-box testing is to ensure that the internal mechanisms of the software are functioning correctly. By scrutinizing the internal logic, control flow, and data flow, white-box testing aims to uncover errors, defects, or vulnerabilities in the code that might affect the overall performance or reliability of the application. It goes beyond surface-level validation to provide a comprehensive assessment of the internal quality of the software.

White-box testing offers several benefits to software development teams. It allows developers to gain a deeper understanding of their codebase, identify potential weaknesses or areas for improvement, and optimize the efficiency and performance of the software. Additionally, by uncovering and addressing issues early in the development lifecycle, white-box testing helps reduce the likelihood of bugs or defects being encountered by end-users.

However, white-box testing also has its limitations. It requires a solid understanding of the system's internal workings and may be time-consuming, especially for complex applications. Moreover, since white-box testing relies on having access to the internal code, it may not be suitable for testing third-party or closed-source software.

In summary, white-box testing plays a crucial role in ensuring the internal integrity of software applications. By examining the inner workings of the system and designing test cases from this perspective, developers and testers can enhance the reliability, efficiency, and overall quality of the software.

# 8. RESULTS

In this specific setup, the video frame is deliberately positioned at a specified angle to capture the street view. The purpose of this arrangement is to improve the accuracy of distance measurements. By transforming the perspective view of the video frame into a top-down view, more precise estimations of distances can be achieved.

When the video sequences are displayed, they are organized in a vertical orientation. Each pedestrian is represented by a designated box, enabling effective crowd monitoring and individual pedestrian tracking within the recorded footage.

To visually represent the proximity between pedestrians, two distinct colors are employed for the boxes. The red boxes indicate pedestrians whose distance from another pedestrian falls below the acceptable threshold. This serves as a visual alert, drawing attention to instances where the recommended safe distance is not being maintained.

Conversely, the green boxes represent pedestrians who are maintaining a safe distance from others. This color differentiation facilitates easy identification and continuous monitoring of individuals who are adhering to social distancing guidelines.

Overall, this setup and visual representation offer a comprehensive approach to analyze crowd dynamics, assess the proximity of individuals, and monitor compliance with safe distance guidelines. It provides valuable insights into the interactions between pedestrians and helps ensure a safer and more controlled environment.

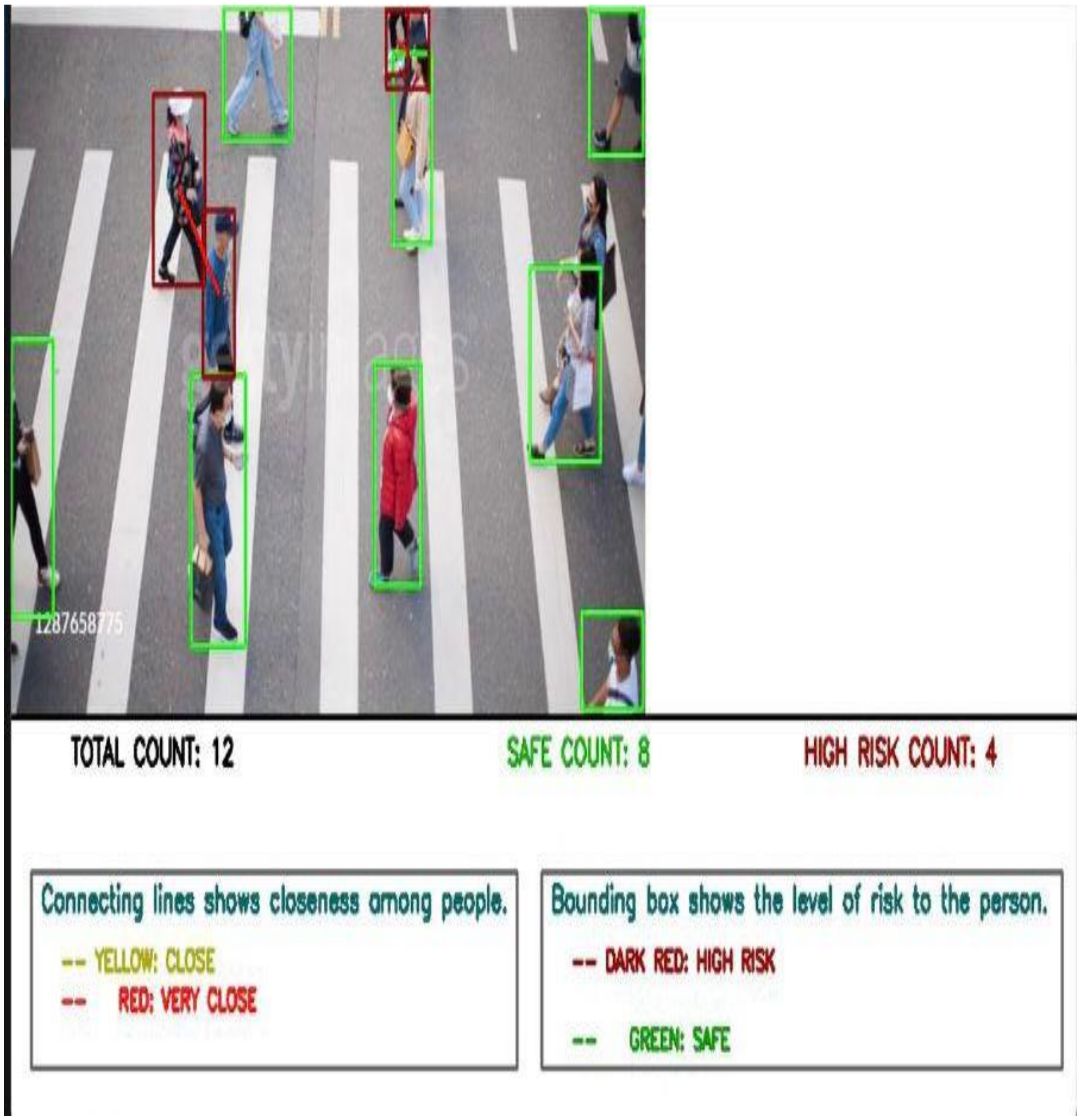
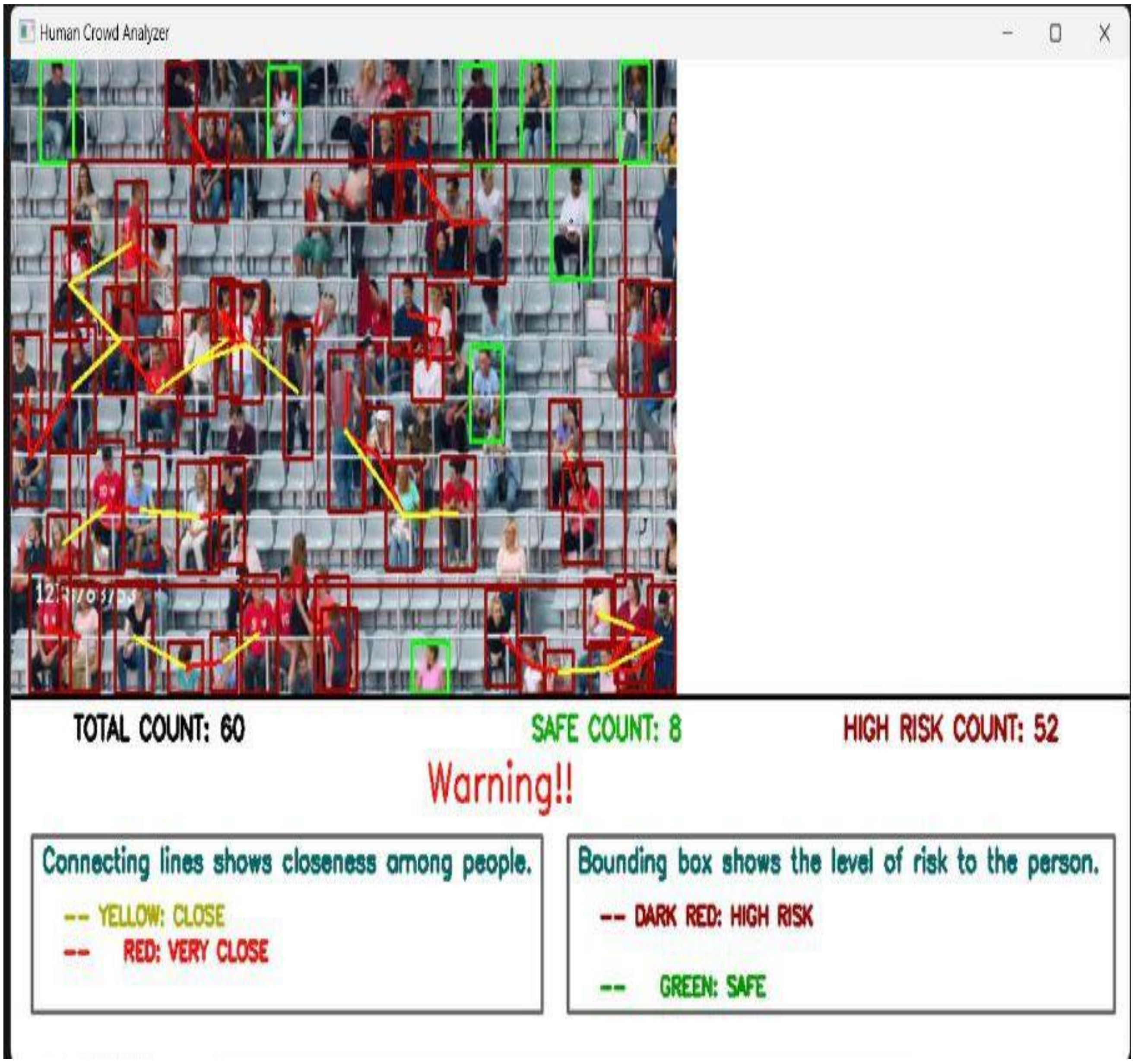


**Figure 19.** Video Feed Detection 1

In the above result as we can observe the high-risk count is not greater the half of total count we don't see any warning displayed on the screen.



TOTAL COUNT: 60

SAFE COUNT: 8

HIGH RISK COUNT: 52

Warning!!

Connecting lines shows closeness among people.

-- YELLOW: CLOSE

-- RED: VERY CLOSE

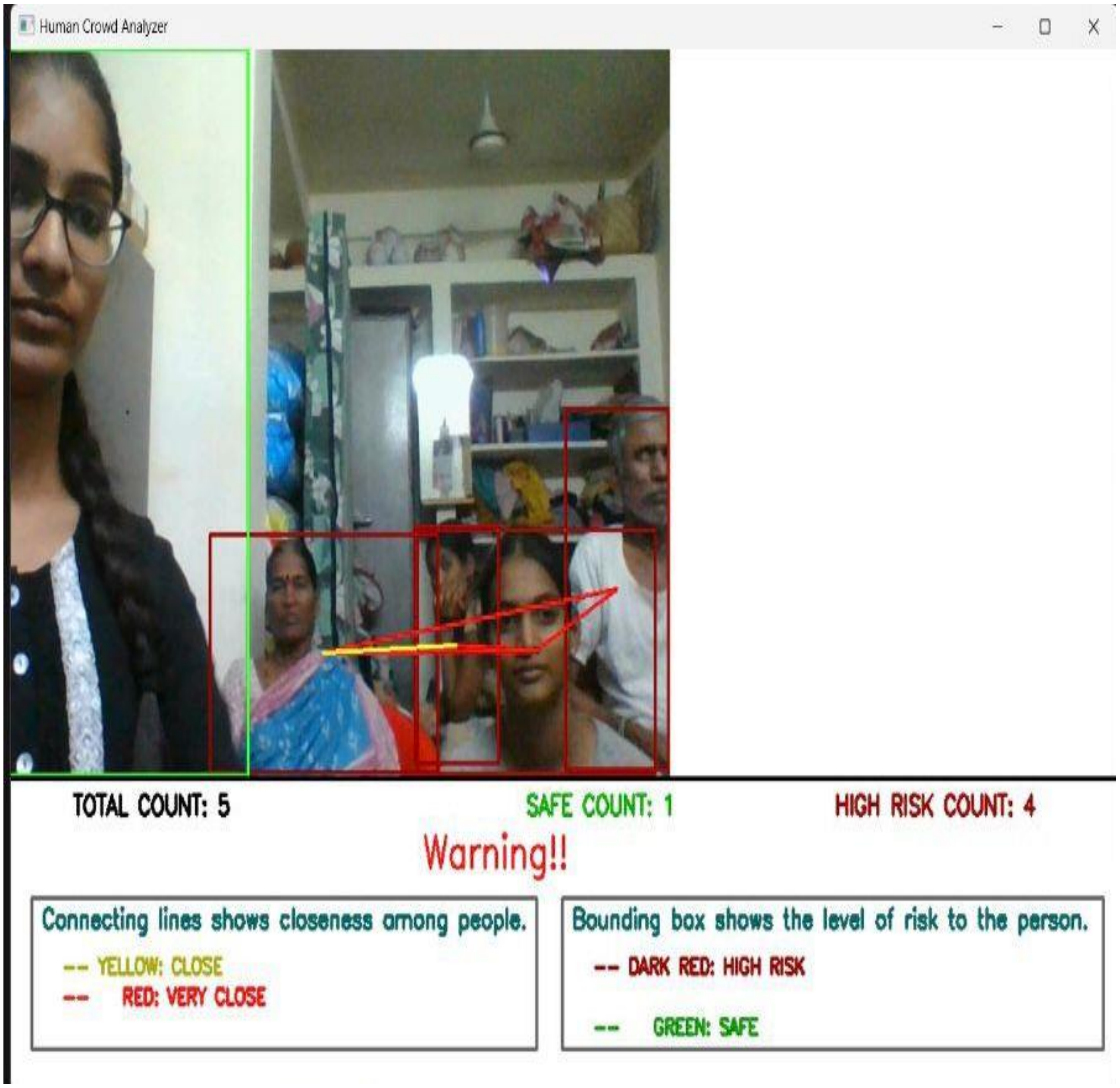
Bounding box shows the level of risk to the person.

-- DARK RED: HIGH RISK

-- GREEN: SAFE

**Figure 20.** Video Feed Detection 2

In the above result as we can observe the high-risk count is  greater the half of total count we can see the warning displayed on the screen.



TOTAL COUNT: 5 SAFE COUNT: 1 HIGH RISK COUNT: 4

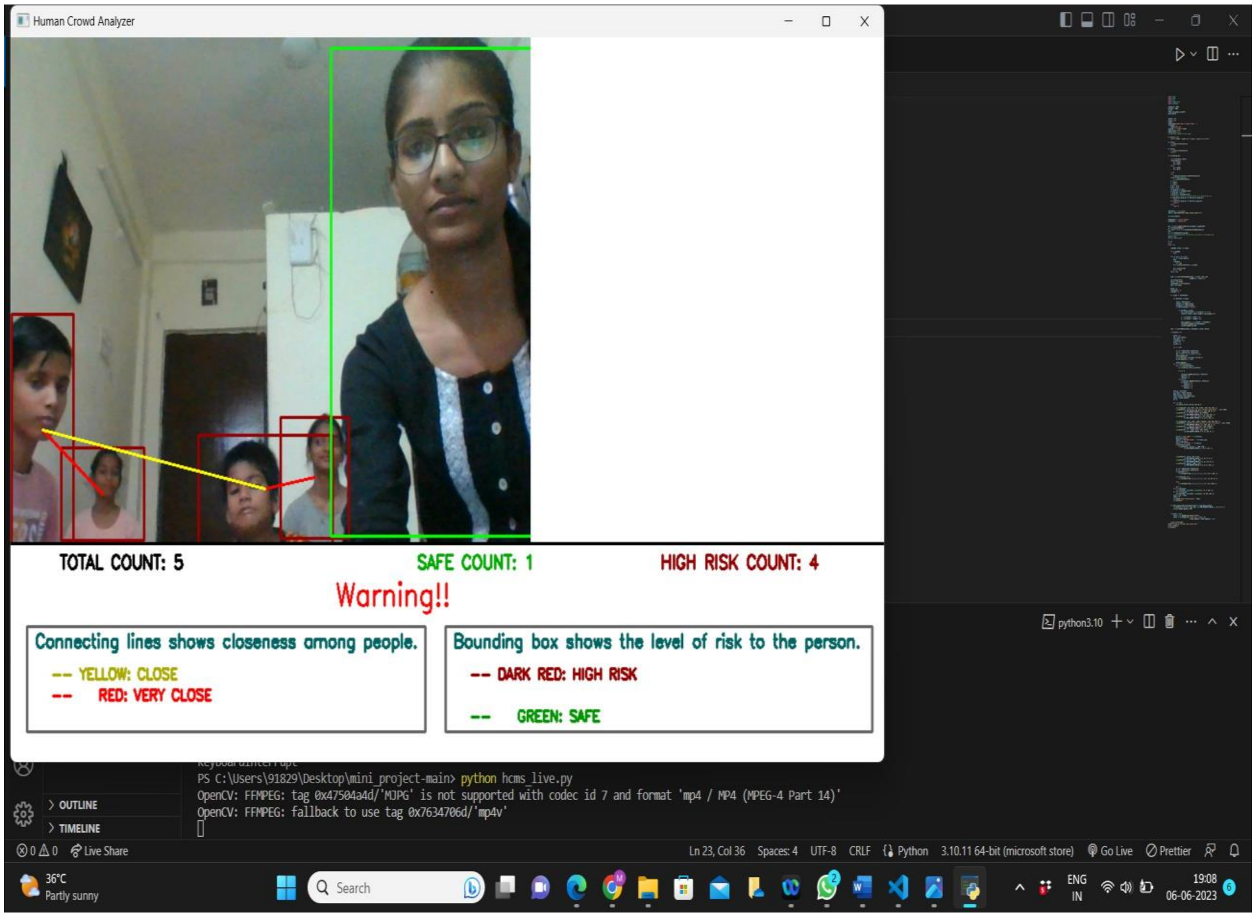
Warning!!

Connecting lines shows closeness among people.
-- YELLOW: CLOSE
-- RED: VERY CLOSE

Bounding box shows the level of risk to the person.
-- DARK RED: HIGH RISK
-- GREEN: SAFE

**Figure 21.** Live Video Detection 1

In the above result as we can observe the high-risk count is greater the half of total count we can see the warning displayed on the screen.



```
PS C:\Users\91829\Desktop\mini_project-main> python hcms_live.py
OpenCV: FFMPEG: tag 0x47504a4d/'MJPG' is not supported with codec id 7 and format 'mp4 / MP4 (MPEG-4 Part 14)'
OpenCV: FFMPEG: fallback to use tag 0x7634706d/'mp4v'
```

**Figure 22.** Live Video Detection 2

In the above result as we can observe the high-risk count is greater the half of total count we can see the warning displayed on the screen.we can also observe a yellow line which indicates they are close.

# 9. CONCLUSION AND FUTURE ENHANCEMENTS

## 9.1 Conclusion

A methodology of human crowd surveillance system using Deep Neural Networks is proposed. By using computer vision the distance between people can be estimated and any non compliant pair of people will be indicated with a red frame and a red line. The proposed method was validated using

a video showing pedestrians walking on a street. The visualization results showed that the proposed method is capable to apply the crowd management strategies to prevent crowd disasters by finding the count of high risk people. This system is capable to generate an alert with a beep sound to the monitoring system when the risk detected.

In conclusion, the provided code demonstrates the implementation of a human crowd analyzer using computer vision techniques. By utilizing the YOLO object detection algorithm and various calculations based on spatial relationships, the code is able to detect and analyze individuals in real-time video streams or webcam input.

The code effectively identifies people in the video feed and determines their proximity and associated risk levels. It accomplishes this by drawing bounding boxes around individuals and connecting lines to visualize their closeness. Additionally, the code counts the total number of people, high-risk individuals, and safe individuals in the frame. The crowd analyzer code offers valuable assistance in monitoring and identifying potentially hazardous situations in crowded environments. It can be utilized in various scenarios, such as monitoring compliance with social distancing guidelines, managing crowds in public spaces, or assessing the risk of overcrowding in specific areas.

Furthermore, the code provides a warning and sound notification if the number of people exceeds a certain threshold and the number of high-risk individuals is above half of the total count. This feature enhances the code's ability to alert users to potentially risky situations in real-time.

In summary, the human crowd analyzer code showcases the application of computer vision techniques to effectively analyze and monitor human crowds. Its potential applications are diverse and can contribute to promoting safety and well-being in crowded environments.

## 9.2 Future enhancements

One possible way to enhance the project even further would be to incorporate a notification system that sends alerts to the mobile phone of the individual responsible for monitoring the crowd.

# 10. BIBLIOGRAPHY

[1] Shailender Kumar, Vishal, Pranav Sharma, Nitin Pal "Object tracking and counting in a zone using YOLOv4, DeepSORT and TensorFlow", 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021.

[2] DushyantKumarSingh, SumitParoothi, Mayank Kumar Rusiac, Mohd. AquibAnsari "Human Crowd Detection for City Wide Surveillance", Third International Conference on Computing and Network Communications (CoCoNet'19), 2020.

[3] Ashish Pandharipande1, Abhishek Murthy2, Erik Hagenaars, Geert Leu's "Single-Pixel Thermopile Infrared Sensing for People Counting", IEEE Sensors Journal, 2020.

[4] Liquan Zhao, Shuaiyang Li "Object Detection Algorithm Based on Improved YOLOv3", IEEE, 2020.

[5] Kang Hao Cheong Sandra Poeschmann "Practical Automated Video Analytics for Crowd Monitoring and Counting" , IEEE, 2019.

[6] S. Salim, O. O. Khalifa, F. A. Rahman and A. Lajis, C¸ rowd Detection And Tracking In Surveillance Video Sequences,"2019 IEEE International Conference on Smart Instrumentation, Measurement and Application (ICSIMA), 2019.

[7] J. Tanaka, M. Shiozaki, F. Aita, T. Seki, and M. Oba, "Thermopile infrared array sensor for human detector application," in Proc. IEEE 27th Int. Conf. Micro Electro Mech. Syst. (MEMS), Jan. 2014, pp. 1213–1216.

[8] M. Kuki, H. Nakajima, N. Tsuchiya, and Y. Hata, "Multi-human locating in real environment by thermal sensor," in Proc. IEEE Int. Conf. Syst., Man, Cybern., Oct. 2013, pp. 4623–4628.

[9] D. Caicedo and A. Pandharipande, "Distributed illumination control with local sensing and actuation in networked lighting systems," IEEE Sensors J., vol. 13, no. 3, pp. 1092–1104, Mar. 2013.

[10] D. Caicedo and A. Pandharipande, "Distributed ultrasonic zoned presence sensing system," IEEE Sensors J., vol. 14, no. 1, pp. 234–243, Jan. 2014.

[11] D. Onoro-Rubio and R. J. López-Sastre. Towards perspective-free object counting with deep learning. In European Conference on Computer Vision, pages 615–629. Springer, 2016.

[12] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. In Proceedings of the IEEE International Conference on Computer Vision, pages 2056–2063, 2013.

[13] W. Ouyang, X. Zeng, and X. Wang. Modeling mutual visibility relationship in pedestrian detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3222–3229, 2013.

[14] V.-Q. Pham, T. Kozakaya, O. Yamaguchi, and R. Okada. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In Proceedings of the IEEE International Conference on Computer Vision, pages 3253–3261, 2015.

[15] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. arXiv preprint, 2017.

[16] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015.

[17] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena, 60(1-4):259–268, 1992.

[18] T. Rahman et al., "Transfer learning with deep Convolutional Neural Network (CNN) for pneumonia detection using chest X-ray," Appl. Sci. (Basel), vol. 10, no. 9, p. 3233, 2020.

[19] H.-W. Luo, C.-S. Zhang, F.-C. Pan, and X.-M. Ju, "ContextualYOLOV3: Implement better small object detection based deep learning," in 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), 2019.

[20] H. Karunasekera, H. Wang, and H. Zhang, "Multiple object tracking with attention to appearance, structure, motion and size," IEEE Access, vol. 7, pp. 104423–104434, 2019.

[21] S. Tang, M. Andriluka, A. Milan, K. Schindler, S. Roth, and B. Schiele, "Learning people detectors for tracking in crowded scenes," in 2013 IEEE International Conference on Computer Vision, 2013.

[22] Sabzmeydani, Payam, and Greg Mori. "Detecting pedestrians by learning shapelet features." In 2007 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-8. IEEE, 2007.

[23] Felzenszwalb, Pedro F., Ross B. Girshick, David McAllester, and Deva Ramanan. "Object detection with discriminatively trained part-based models." IEEE transactions on pattern analysis and machine intelligence 32, no. 9 (2010): 1627-1645.

[24] Lin, Sheng-Fuu, Jaw-Yeh Chen, and Hung-Xin Chao. "Estimation of number of people in crowded scenes using perspective transformation." IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 31, no. 6 (2001): 645-654.

[25] Wu, Bo, and Ram Nevatia. "Detection and tracking of multiple, partially occluded humans

by bayesian combination of edgelet based part detectors." International Journal of Computer Vision 75, no. 2 (2007): 247-266.

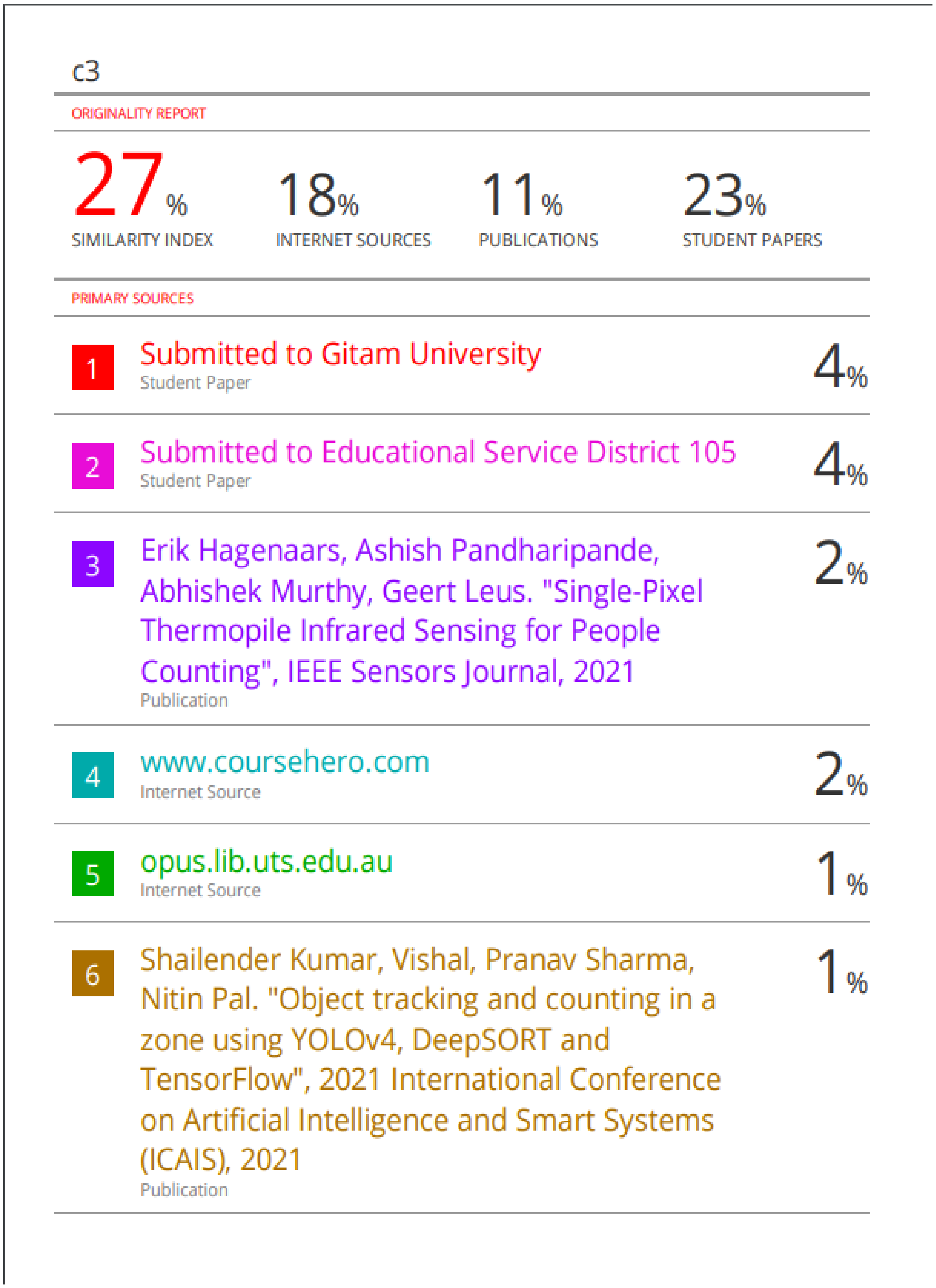
[26] V. Moskalenko, A. Moskalenko, A. Korobov, O. Boiko, S. Martynenko, and O. Borovenskyi, ‘‘Model and training methods of autonomous navigation system for compact drones,’’ in Proc. IEEE 2nd Int. Conf. Data Stream Mining Process. (DSMP), Aug. 2018, pp. 503–508.

[27] A. A. Zhilenkov and I. R. Epifantsev, ‘‘The use of convolution artificial neural networks for drones autonomous trajectory planning,’’ in Proc. IEEE Conf. Russian Young Res. Elect. Electron. Eng. (EIConRus), Jan./Feb. 2018, pp. 1044–1047.

[28] Y. Satılmiş, F. Tufan, M. Şara, M. Karslı, S. Eken, and A. Sayar, ‘‘CNN based traffic sign recognition for mini autonomous vehicles,’’ in Proc. Int. Conf. Inf. Syst. Archit. Technol., J. Świątek, L. Borzemski, and Z. Wilimowska, Eds. Cham, Switzerland: Springer, 2019, pp. 85–94.

[29] M. Hirz and B. Walzel, ‘‘Sensor and object recognition technologies for self-driving cars,’’ Comput.-Aided Des. Appl., vol. 15, no. 4, pp. 501–508, 2018.

[30] A. Shustanov and P. Yakimov, ‘‘CNN design for real-time traffic sign recognition,’’ Procedia Eng., vol. 201, pp. 718–725, Dec. 2017.



c3

ORIGINALITY REPORT

| 27% | 18% | 11% | 23% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | Submitted to Gitam University<br>Student Paper | 4% |
| 2 | Submitted to Educational Service District 105<br>Student Paper | 4% |
| 3 | Erik Hagenaars, Ashish Pandharipande, Abhishek Murthy, Geert Leus. "Single-Pixel Thermopile Infrared Sensing for People Counting", IEEE Sensors Journal, 2021<br>Publication | 2% |
| 4 | www.coursehero.com<br>Internet Source | 2% |
| 5 | opus.lib.uts.edu.au<br>Internet Source | 1% |
| 6 | Shailender Kumar, Vishal, Pranav Sharma, Nitin Pal. "Object tracking and counting in a zone using YOLOv4, DeepSORT and TensorFlow", 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021<br>Publication | 1% |