SUSPICIOUS MILITARY ACTIVITY RECOGNITION AND ALERT SYSTEM USING RASPBERRY PI

A PROJECT REPORT

Submitted by

KP DHIVYESH ANAND	**211419205042**
S MUKUNDHAN	**211419205113**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE,POONAMALLEE

ANNA UNIVERSITY :CHENNAI 600 025

APRIL 2023

BONAFIDE CERTIFICATE

Certified that this project report **SUSPICIOUS MILITARY ACTIVITY RECOGNITION AND ALERT SYSTEM USING RASPBERRY PI** is the bonafide work of **KP DHIVYESH ANAND (21141205042), S MUKUNDHAN (211419205113)** who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Dr. M. HELDA MERCY M.E., Ph.D.,	**Mrs.K. LALITHA M.E., Ph.D.,**
PROFESSOR	**SUPERVISOR**
HEAD OF THE DEPARTMENT	**ASSOCIATE PROFESSOR**
Department of Information Technology	Department of Information Technology
Panimalar Engineering College	Panimalar Engineering College
Poonamallee, Chennai - 600 123	Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on ______________

SIGNATURE	**SIGNATURE**
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

I hereby declare that the project report entitled **SUSPICIOUS MILITARY ACTIVITY RECOGNITION AND ALERT SYSTEM USING RASPBERRY PI** which is being submitted in partial fulfilment of the requirement of the course leading to the award of the 'Bachelor Of Technology in Information Technology ' in **Panimalar Engineering College, Autonomous institution Affiliated to Anna university- Chennai** is the result of the project carried out by me under the guidance **in the Department of Information Technology**. I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

KP DHIVYESH ANAND

Date**:** **S MUKUNDHAN**

Place: Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date: Mrs. K. LALITHA M.E., Ph.D.,

Place: Chennai (ASSOCIATE PROFESSOR/ IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion . We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Honourable Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors , Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR, M.E.,M.B.A.,Ph.D.,** and **Dr.SARANYASREE SAKTHIKUMAR .,B.E., M.B.A., Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY,M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project co-ordinator **Mr. M. DILLI BABU ,M.E., (Ph.D.,) ASSOCIATE PROFESSOR**, Department of Information Technology for his guidance throughout the course of our project. We also express sincere thanks to our supervisor **Mrs.K.LALITHA, M.E., (Ph.D.,) ASSOCIATE PROFESSOR** for providing the support to carry out the project successfully. Last, we our parents and friends for providing their extensive moral support and encouragement during the course of the project.

TABLE OF CONTENTS

ABSTRACT

Computer vision is an important field of computer science that deals with processing digital photos and videos to enable machines to understand and interpret visual information. One of the most significant applications of computer vision is activity recognition, which involves automatically classifying the behavior of an agent, such as a person or object.

In the context of security and surveillance, suspicious activity recognition is a key application of computer vision. This involves analyzing a series of observations to uncover a person's actions and detect potential threats or suspicious behavior. However, this can be a challenging task due to contextual factors, such as lighting conditions, occlusions, and the complexity of human behavior.

To address these challenges, computer vision algorithms use advanced techniques such as deep learning, which can learn complex patterns and features from visual data. These algorithms can be trained on large datasets of labeled examples to improve their accuracy and generalizability.

Once a suspicious activity is detected, the system can send an alert or notification to the user or commando through the internet. This allows for quick and decisive action to be taken, potentially preventing security threats and enhancing public safety.

Overall, computer vision and activity recognition are important fields of study that have numerous applications in security, surveillance, and other domains. As technology continues to evolve, we can expect to see even more advanced and effective systems for detecting and preventing suspicious activity.

LIST OF FIGURES

LIST OF ABBREVIATION

LPMT -	Location Privacy-Preserving mechanism based on Trajectory Obfuscation
HAR -	Human Activity Recognition
FGA -	Fine-Grained Activities
LSTM -	Long Short-Term Memory
ANN -	Artifical Neural Network
ARGB-	Adobe Red Green Blue

CHAPTER-1
INTRODUCTION

1. INTRODUCTION

In recent years, the field of automated human activity identification has seen a significant surge in interest due to the rapid advancements in technology. This has led to the development of intelligent video surveillance systems that can track human movements in real-time, classify them based on regular or unusual behaviour, and generate alerts when necessary. This technology has become increasingly important in ensuring safety both inside and outside the border.

The use of video surveillance systems has become widespread, with many applications for spotting regular and unusual activity, such as gaming, human-computer interaction, exam monitoring, chaos detection, sports analysis, crowd behaviour forecasting, and more. The ability to detect and respond to anomalous or unexpected occurrences is crucial for ensuring public safety and protecting against criminal or violent acts.

The volume of video data that needs to be analysed has become enormous, making manual intervention not only impossible but also error-prone. Automated systems are thus needed to handle the task of monitoring and analysing large volumes of video data. Intelligent video surveillance systems utilise a variety of sensors, including radar, cameras, and cell phones, to detect and track human activity in real-time.

One of the primary uses of video surveillance systems is for security and surveillance. CCTV cameras are commonly used to record video material that can be used in criminal or violent act investigations. However, the passive nature of traditional surveillance systems can be insufficient in preventing such acts from occurring. Intelligent video surveillance systems offer a more proactive approach, by identifying suspicious behaviour and generating alerts that can be acted upon immediately.

Exam monitoring is another application for intelligent video surveillance systems. With the rise of online learning, remote proctoring has become increasingly important. Intelligent systems can monitor students during online exams, detecting irregular behaviour such as looking away from the screen or using other devices, and generating alerts for further investigation.

Human-computer interaction is another area where intelligent video surveillance systems are useful. These systems can track the movements of users and generate commands based on their actions. This can be particularly useful in applications such as gaming, where users can interact with the game using their body movements.

Sports analysis is yet another application of intelligent video surveillance systems. These systems can be used to analyse the movements of players, detecting patterns and generating insights that can be used to improve performance. They can also be used to monitor the behaviour of spectators, detecting crowd behaviour and generating alerts in the event of disturbances.

Crowd behaviour forecasting is another area where intelligent video surveillance systems can be useful. These systems can detect patterns in the movements of crowds and generate forecasts of future behaviour. This can be particularly useful in applications such as public safety, where anticipating crowd behaviour can help prevent disturbances and ensure public safety.

In addition to their numerous applications, intelligent video surveillance systems also offer several advantages over traditional surveillance systems. They are more proactive, detecting and responding to suspicious behaviour in real-time. They are also more accurate, reducing the risk of false alarms and improving the effectiveness of security measures. Finally, they are more efficient, automating the process of monitoring and analysing large volumes of video data.

However, despite their many advantages, there are also several challenges associated with the deployment of intelligent video surveillance systems. One of the biggest challenges is privacy concerns. With the use of cameras and other sensors, there is a risk of invading people's privacy. To address these concerns, it is essential to implement appropriate privacy measures, such as encrypting data and limiting access to sensitive information.

Another challenge is the need for high-quality data. Intelligent video surveillance systems rely on accurate and high-quality data to function effectively. This includes data on human behaviour, as well as data on environmental factors such as lighting and weather conditions. Ensuring the quality of data is essential for the accurate detection and classification of human activity.

1.1 OVERVIEW OF THE PROJECT

The use of Raspberry Pi for detecting suspicious military activity has become increasingly popular in recent years. This is due to the fact that it is a low-cost, easily accessible, and customizable platform for designing and deploying sensor networks that can monitor and detect unusual movements in military installations.

The main objective of this project is to design and implement a system using Raspberry Pi for the detection of suspicious military activity. The system will be designed to detect any unusual movement within the installation premises and notify the security personnel in real-time.

The proposed system will consist of a Raspberry Pi board, a camera module, and a motion sensor. The camera module will capture the video footage of the installation premises, and the motion sensor will detect any movement within the vicinity of the installation.

The captured video footage will be processed using computer vision algorithms such as object detection and tracking, which will help in identifying any suspicious activity in the area. The system will use machine learning algorithms for detecting and classifying the suspicious activities.

The system will also be equipped with an alarm system that will alert the security personnel in real-time when any suspicious activity is detected. The alarm system can be customized to suit the specific needs of the installation.

The project will involve the following stages:

Hardware Design: The first stage of the project will involve designing the hardware components of the system. This will include selecting the appropriate Raspberry Pi board, camera module, and motion sensor.

Software Development: The second stage of the project will involve developing the software components of the system. This will include developing computer vision algorithms for object detection and tracking, as well as machine learning algorithms for detecting and classifying the suspicious activities.

System Integration: The third stage of the project will involve integrating the hardware and software components of the system. This will include connecting the camera module and motion sensor to the Raspberry Pi board and configuring the software components of the system.

Testing and Validation: The final stage of the project will involve testing and validating the system. This will include conducting a series of tests to ensure that the system is functioning as expected and meeting the specific requirements of the installation.

In conclusion, the use of Raspberry Pi for detecting suspicious military activity is a promising area of research. The proposed system will provide a low-cost, easily accessible, and customizable platform for designing and deploying sensor networks that can monitor and detect unusual movements in military installations. The success of this project will pave the way for further research in this area and help in enhancing the security of military installations.

1.2 SCOPE OF THE PROJECT

The scope of this project is to design and implement a system for detecting suspicious military activity using Raspberry Pi. The system will be designed to monitor and detect any unusual movements within the installation premises and alert the security personnel in real-time.

The project will involve the following components:

Hardware Components: The hardware components of the system will include a Raspberry Pi board, a camera module, and a motion sensor. The camera module will be used to capture video footage of the installation premises, and the motion sensor will detect any movement within the vicinity of the installation.

Software Components: The software components of the system will include computer vision algorithms for object detection and tracking and machine learning algorithms for detecting and classifying suspicious activities. The system will also include an alarm system for alerting the security personnel in real-time when any suspicious activity is detected.

The project will have the following objectives:

Design a low-cost, easily accessible, and customizable system for detecting suspicious military activity using Raspberry Pi.

Develop computer vision algorithms for object detection and tracking that can accurately identify any suspicious activity in the installation premises.

Develop machine learning algorithms for detecting and classifying the suspicious activities.

Integrate the hardware and software components of the system to ensure that they are functioning properly.

Test and validate the system to ensure that it is meeting the specific requirements of the installation.

The project will have the following deliverables:

A fully functional system for detecting suspicious military activity using Raspberry Pi.

A detailed report outlining the hardware and software components of the system, including the computer vision and machine learning algorithms used for detecting suspicious activities.

A user manual detailing the installation, configuration, and operation of the system.

The project will have the following timeline:

Hardware Design: This phase of the project will take approximately 2 weeks to complete. It will involve selecting the appropriate Raspberry Pi board, camera module, and motion sensor and designing the hardware components of the system.

Software Development: This phase of the project will take approximately 6 weeks to complete. It will involve developing computer vision algorithms for object detection and tracking, as well as machine learning algorithms for detecting and classifying suspicious activities.

System Integration: This phase of the project will take approximately 2 weeks to complete. It will involve integrating the hardware and software components of the system and configuring the system.

Testing and Validation: This phase of the project will take approximately 4 weeks to complete. It will involve testing and validating the system to ensure that it is meeting the specific requirements of the installation.

The project will have the following resources:

Raspberry Pi boards, camera modules, and motion sensors.

A computer with the necessary software for developing the computer vision and machine learning algorithms.

Access to the installation premises for testing and validation.

Personnel with expertise in computer vision, machine learning, and hardware design.

In conclusion, the scope of this project is to design and implement a system for detecting suspicious military activity using Raspberry Pi. The project will involve designing the hardware components of the system, developing computer vision and machine learning algorithms for detecting suspicious activities, integrating the hardware and software components of the system, and testing and validating the system. The project will require the use of Raspberry Pi boards, camera modules, and motion sensors, as well as personnel with expertise in computer vision, machine learning, and hardware design. The success of this project will pave the way for further research in this area and help in enhancing the security of military installations.

1.3 OBJECTIVE OF THE PROJECT

The goal of this initiative is to develop and put into use a system for monitoring suspicious military activity. The system will be built to keep an eye on and spot any strange motions inside the installation site and immediately notify the security staff. The following elements will be included in the project: Hardware: The system's hardware will consist of a Raspberry Pi board, a video module, and a motion sensor. The motion sensor will be used to identify any movement near the installation while the camera module will be used to record video of the installation site. Software Components: The system's software will consist of computer vision algorithms for tracking and finding objects as well as machine learning algorithms for identifying and categorising them.

The system will also have an alarm system to immediately notify the security staff of any suspicious behaviour. The initiative will aim to accomplish the following: Create a Raspberry Pi-based system that is inexpensive, simple to use, and adaptable for spotting suspicious military behaviour. Create object detection and tracking computer vision algorithms that can precisely pinpoint any suspicious behaviour occurring on the installation's property. Create artificial intelligence (AI) tools to identify and categorize suspicious activity. Integrate the system's hardware and software parts to make sure they are operating correctly. To make sure the system is fulfilling the installation's particular requirements, test and validate it. The following products will be part of the project: a complete system for spotting suspicious military behaviour using Raspberry Pi.

A thorough report outlining the system's hardware and software parts, including the machine learning and computer vision algorithms employed to identify suspicious activity. a user guide that describes how to setup, set up, and use the system. The project's timetable will be as follows: Hardware Design: This stage of the endeavour should be finished in around two weeks. The hardware for the system will need to be designed, along with the proper Raspberry Pi board, camera module, and motion sensor. Software Development: This stage of the endeavour should be finished in around six weeks. It will entail creating machine learning algorithms for identifying and categorising suspicious actions as well as computer vision algorithms for object detection and tracking.

System Integration: This stage of the endeavour should be finished in around two weeks. It will entail setting up the system and merging its hardware and software components. Testing and Validation: This stage of the endeavour should be finished in around 4 weeks. To make sure the system is fulfilling the installation's unique requirements, it will be put through testing and validation. The following tools will be used in the project: boards for Raspberry Pi, video modules, and motion detectors. a system that is equipped with the software required to create the computer vision and machine learning algorithms. For testing and validation, access to the installation's grounds. personnel with knowledge in hardware architecture, machine learning, and computer vision.

In conclusion, the goal of this endeavour is to create and put into use a system that can identify suspicious military activity. The project will entail designing the system's hardware parts, creating computer vision and machine learning algorithms for spotting suspicious activity, fusing the system's hardware and software parts, and testing and verifying the system. Raspberry Pi boards, camera modules, and motion sensors will be used in the project, along with people skilled in computer vision, machine learning, and hardware design. The accomplishment of this endeavour will open the door for additional study in this field and contribute to improving military installation security.

CHAPTER – 2

LITERATURE SURVEY

2.1.1 Human activity analysis: A review

Methods

Human activity recognition is an important area of computer vision research that aims to automatically analyze ongoing activities from videos. This ability enables the construction of several important applications such as automated surveillance systems, real-time monitoring of patients and construction of gesture-based human computer interfaces. Human activities are categorized into gestures, actions, interactions, and group activities based on their complexity. The article provides an overview of state-of-the-art human activity recognition methodologies, which are classified into two categories: single-layered approaches and hierarchical approaches. The former includes space-time and sequential approaches, each with subcategories based on the features they use. The article discusses recent research trends in activity recognition.

Results

On average, the AI was able to identify the different human actions and classify it accordingly on both single and hierarchical approaches.

Conclusions

The article discusses the current state of research in human activity recognition, its applications, and the challenges faced in this field. The authors have provided an overview of the various approaches to human activity recognition and have discussed their advantages and disadvantages. They have also categorized the approaches into nonhierarchical and hierarchical methods and discussed them thoroughly. The article highlights the progress made in the past decade, but also acknowledges that there is still much work to be done before human activity recognition becomes an off-the-shelf technology. The authors have also discussed the challenges posed by the use of moving cameras and the need for robust tracking algorithms. The article concludes by emphasizing the importance of this field and its future directions.

2.1.2 A Survey on Vision Based Activity Recognition, its Applications and Challenges

Methods

Human activity recognition has gained attention in recent years due to its practical applications. It has the potential to improve surveillance systems by identifying unusual activities and alerting authorities. In entertainment environments, it can be used for Human Computer Interaction systems, where the computer responds to the activity of the person. In healthcare systems, activity recognition can aid in patient rehabilitation by automatically monitoring their activities. Other applications include smart homes, sports analysis, and augmented reality. Human activity recognition is a versatile technology that can be used in a wide range of fields to improve efficiency and safety

Objective

The aim of this find the published activity recognition for vision based systems and note them down accordingly

Results

Through this work, various types of activity recognition systems and techniques are reported. It is observed that interaction and action recognition are solved in two different ways. Further there is no single solution that fits all class of problems. There are many algorithms available to solve activity recognition. However, the same sequence of algorithms perform with different efficiency on different data-sets. The activity recognition has widespread applications in different domains. However, vision based activity recognition has its own set of challenges

2.1.3 Advances in Human Action Recognition: A Survey

Methods

Human action recognition is a growing area of computer vision, with applications in video analysis, surveillance, and human-machine interfaces. The process involves matching observed movements to predefined patterns and assigning a label. The complexity of the activity can range from gestures to group activities. Various methodologies have been proposed, relying on different feature representations and learning methods. The accuracy of tracking is a key factor in the recognition process. This paper reviews recent advances in human action recognition, using a similar taxonomy to previous surveys. The paper discusses single-layered and hierarchical approaches, with a focus on recent developments.

Objective

The aim of this find the published activity recognition for vision based systems and note them down accordingly.

Results

This letter provides a survey of recent advances in automated human action recognition, covering 50 specific and influential proposals of the last three years. While much research has been devoted to recognizing human actions directly from videos or images in a single-layered manner, there is increasing interest in hierarchical approaches for high-level activity recognition. The authors hope that more research will be done in the area of high-level action recognition in datasets and real-world scenes, and suggest that a cross-domain framework would be beneficial to the entire community.

2.1.4 A review of state-of-the-art techniques for abnormal human activity recognition

Methods

Abnormal Human Action Recognition (AbHAR) through visual and sensor data has vast potential in various fields, such as AAL, healthcare, video surveillance, HCI, sports, robotics, intrusion detection, multimedia annotation, and indexing. AbHAR is used interchangeably with anomaly detection, referring to non-conforming patterns in a given data. The accuracy of the system depends on the feature representing the actions, leading to better recognition results. Anomaly detection has been reviewed in various articles and books, highlighting various techniques for effective detection under different constraints and applications. This survey focuses on the latest methodologies for a robust AbHAR system based on two-dimensional and three-dimensional inputs, covering a wide range of applications.

Objective

This paper summarizes existing abnormal human activity recognition (AbHAR) approaches using 2D and 3D data, focusing on feature design in various applications, and outlines newly added datasets for validation.

Results

The literature highlights the progress in feature design for real-time and efficient abnormal human activity recognition in various contexts such as surveillance and fall detection, using different input dimensions such as RGB, depth, and skeleton. Depth and skeleton representations are effective for handling view and illumination variations, while RGB images are used for multiple persons. Deep features have shown better performance than hand-crafted features, but require high computational resources.

2.1.5 Interactive activity recognition using pose-based spatio–temporal relation features and four-level Pachinko Allocation Model

Methods

Human activity recognition has numerous applications and has been a subject of research in computer vision and artificial intelligence. Recognizing human activities in the real environment remains a challenge due to variations in appearance, mutual occlusion, and object interactions. Most approaches have focused on low-level features, but recent advancements in human pose estimation have motivated research in human activity recognition. This paper proposes a flexible topic model for human interaction recognition, using spatio-temporal relation features calculated from articulated-pose coordinates. A hierarchical model based on the Pachinko Allocation Model (PAM) is used, along with Support Vector Machine (SVM) for solving the multi-class classification problem. Experimental results are presented and discussed.

Result

The proposed method for human interaction recognition includes articulated-body estimation, spatio-temporal relation feature extraction, codebook construction, topic modeling, and activity classification. The method was evaluated using 10-fold cross-validation on two datasets: BIT-Interaction and UT-Interaction. Experiments were performed on a desktop PC running Windows 7 OS with Matlab 2013a.

Conclusions

We proposed a four-level topic model, developed from Pachinko Allocation Model, for the interactive activity recognition, in which the relationships between the relation features and the interactions are fully described through the interactive poselets. In our approach, the intra and inter-person joint features of distance and angle are calculated in the spatio-temporal dimension from the pose estimation outcome. The poselet layer is composed by two types of codeword, d-word and a-word.

2.1.6 A Vision-Based System for Intelligent Monitoring: Human Behaviour Analysis and Privacy by Context

Objective

The vision@home project aims to increase the personal autonomy of elderly and impaired people by providing vision-based monitoring services for care and safety. The system uses multi-view cameras to recognize people's behavior based on human action recognition, and a privacy-by-context method is proposed to protect privacy when a remote connection occurs. The behavior recognition method shows outstanding performance, and the system supports multi-view scenarios and real-time execution for the proposed services.

Methods

Video cameras are commonly used in public places for surveillance, but they have not been widely used in private homes due to privacy concerns. However, in ambient-assisted living (AAL) applications, vision-based technology can support elderly or impaired individuals to promote independent living. The vision@home project aims to increase personal autonomy by developing an intelligent monitoring system that uses visual and environmental sensors for human behavior analysis. The proposed architecture includes human action recognition, privacy protection, and telecare services. The project also introduces a privacy-by-context technique to preserve the right to privacy of the inhabitants. Experimental results and conclusions are discussed.

Results

The vision@home project presents a feasible solution for vision-based monitoring of elderly and impaired people, enabling independent living at home and supporting different ambient-assisted living services. The project's novel contribution includes human behaviour analysis based on action recognition, which is essential for both supporting the desired AAL services and enabling privacy protection. However, more work needs to be done to recognize more complex activities and understand scene context, as well as validate the proposal in real-world scenarios with healthcare professionals. Despite the challenges, the vision@home project provides a promising solution for intelligent monitoring systems at home and supporting AAL services.

Conclusions

The vision@home project presents a vision-based intelligent monitoring system that supports AAL services by relying mainly on human behaviour analysis for care and safety services. It proposes a privacy-by-context method to protect the inhabitants' privacy while providing telecare services. The method achieves accurate recognition and real-time performance through a variety of publicly available datasets, and outstanding results are obtained using depth sensors to provide human silhouette. Future directions include extending the method to recognise more complex activities and validating the current advances in real scenarios.

2.1.7 A Depth Video Sensor-Based Life-Logging Human Activity Recognition System for Elderly Care in Smart Indoor Environments

Objective

This paper proposes a depth-based life logging system for recognizing daily activities of elderly people using a depth imaging sensor. The system uses Hidden Markov Models for training and achieves satisfactory recognition rates compared to conventional approaches. The proposed system has potential applications in healthcare and indoor activity monitoring for elderly people.

Methods

Depth imaging sensors have improved and become more affordable, leading to increased attention in areas such as human computer interaction and multimedia content analysis. Human activity recognition is a major application of these systems, with potential uses in monitoring daily activities of residents in indoor environments. The aim of this study is to develop an efficient depth-based life-logging system that monitors the activities of residents 24 hours a day, improving their comfort and well-being at home.

Conclusions

The paper proposes an efficient depth video-based life-logging system for human activity recognition (HAR) using skeleton joint features generated by depth video sensors. The proposed system achieves promising results compared to conventional methods and could be useful in consumer applications to improve quality of life. Future work aims to exploit the system's effectiveness in occluded regions and more complex activities.

2.1.8 Continuous human action recognition in real time

Methods

Human activity recognition is an important area of computer vision research that aims to automatically analyze ongoing activities from videos. This ability enables the construction of several important applications such as automated surveillance systems, real-time monitoring of patients and construction of gesture-based human computer interfaces. Human activities are categorized into gestures, actions, interactions, and group activities based on their complexity. The article provides an overview of state-of-the-art human activity recognition methodologies, which are classified into two categories: single-layered approaches and hierarchical approaches. The former includes space-time and sequential approaches, each with subcategories based on the features they use. The article discusses recent research trends in activity recognition.

Results

On average, the AI was able to identify the different human actions and classify it accordingly on both single and hierarchical approaches.

Conclusions

The article discusses the current state of research in human activity recognition, its applications, and the challenges faced in this field. The authors have provided an overview of the various approaches to human activity recognition and have discussed their advantages and disadvantages. They have also categorized the approaches into non-hierarchical and hierarchical methods and discussed them thoroughly. The article highlights the progress made in the past decade, but also acknowledges that there is still much work to be done before human activity recognition becomes an off-the-shelf technology. The authors have also discussed the challenges posed by the use of moving cameras and the need for robust tracking algorithms. The article concludes by emphasizing the importance of this field and its future directions.

2.1.9 Depth video-based human activity recognition system using translation and scaling invariant features for life logging at smart home

Methods

Human activity recognition has numerous applications and has been a subject of research in computer vision and artificial intelligence. Recognizing human activities in the real environment remains a challenge due to variations in appearance, mutual occlusion, and object interactions. Most approaches have focused on low-level features, but recent advancements in human pose estimation have motivated research in human activity recognition. This paper proposes a flexible topic model for human interaction recognition, using spatio-temporal relation features calculated from articulated-pose coordinates. A hierarchical model based on the Pachinko Allocation Model (PAM) is used, along with Support Vector Machine (SVM) for solving the multi-class classification problem. Experimental results are presented and discussed.

Result

The proposed method for human interaction recognition includes articulated-body estimation, spatio-temporal relation feature extraction, codebook construction, topic modeling, and activity classification. The method was evaluated using 10-fold cross-validation on two datasets: BIT-Interaction and UT-Interaction. Experiments were performed on a desktop PC running Windows 7 OS with Matlab 2013a.

Conclusions

We proposed a four-level topic model, developed from Pachinko Allocation Model, for the interactive activity recognition, in which the relationships between the relation features and the interactions are fully described through the interactive poselets. In our approach, the intra and inter-person joint features of distance and angle are calculated in the spatio-temporal dimension from the pose estimation outcome. The poselet layer is composed by two types of codeword, d-word and a-word

2.1.10 Fast human activity recognition based on structure and motion

Methods:

Human action recognition is a growing area of computer vision, with applications in video analysis, surveillance, and human-machine interfaces. The process involves matching observed movements to predefined patterns and assigning a label. The complexity of the activity can range from gestures to group activities. Various methodologies have been proposed, relying on different feature representations and learning methods. The accuracy of tracking is a key factor in the recognition process. This paper reviews recent advances in human action recognition, using a similar taxonomy to previous surveys. The paper discusses single-layered and hierarchical approaches, with a focus on recent developments.

Objective

The aim of this find the published activity recognition for vision based systems and note them down accordingly.

Results

This letter provides a survey of recent advances in automated human action recognition, covering 50 specific and influential proposals of the last three years. While much research has been devoted to recognizing human actions directly from videos or images in a single-layered manner, there is increasing interest in hierarchical approaches for high-level activity recognition. The authors hope that more research will be done in the area of high-level action recognition in datasets and real-world scenes, and suggest that a cross-domain framework would be beneficial to the entire community.

2.2 Feasibility study

The systems objectives outlined during the feasibility study serve as the basic From which the work of system design is initiated. Much of the activities involved at this stage is of technical nature requiring a certain degree of experience in designing systems, sound knowledge of computer related technology and through understanding of computers available in the market and the various facilities provided by the vendors. Nevertheless, a system cannot be designed in isolation without the active involvement of the user. The user has a vital role to play at this stage too. As we know that data collected during feasibility study wills we utilized systematically during the system design. It should, however be kept in mind that detailed study of the existing system is not necessarily over with the completion of the feasibility study. Depending on the plan of feasibility study, the level of detailed study will vary and the system design stage will also vary in the amount of investigation that still needs to be done.

This investigation is generally an urgent activity during the system. Sometimes, But rarely, this investigation may form a separate stage between feasibility study And computer system design. Designing a new system is a creative process, which calls for logical as well as lateral thinking. The logical approach involves systematic moves towards the end product keeping in mind the capabilities of the personnel and the equipment at each decision making step. Lateral thought implies encompassing of ideas beyond the usual functions and equipment. This is to ensure that no efforts are being made to fit previous solutions into new situations. The feasibility study proposes one or more conceptual solutions to the problem set for the project. The objective in assessing feasibility is to determine whether a development project has a reasonable chance of success.

It helps us to determine the input &amp; output of the system. The following are the criteria that are considered to confirm the project feasibility.

The following feasibility study was undertaken for the proposed system

Technical feasibility

At first it’s necessary to check that the proposed system is technically feasible or not &amp; to determine the technology and skill necessary to carry out the project. If they are not available then find out the solution to obtain them. Hardware is already available in the University.

Economic feasibility

While considering economic feasibility, it is checked in points like performance, information and outputs from the system. The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

Social feasibility

Although generally there is always resistance, initially to any change in the system is aimed at reliving the work load of the users to extent the system is going to facilitate user to perform Operations like calculating salary amounts and deductions, generating reports with less possible errors. Thus there is no reason to make system socially unfeasible

CHAPTER – 3
SYSTEM ANALYSIS

The way that is followed while carrying on with the development application is As follows:

3.1 Defining a problem

Defining a problem is one of the important activities of the project. The objective is to define precisely the business problem to be solved &amp; thereby determined the scope of the new system. This phase consist of 2 main tasks. The 1st task within this activity is to review the organization needs that originally initiated the project. The 2nd task is to identify, at an abstract or general level, the expected capabilities of the new system. Thus, it helps us to define the goal to be achieved &amp; the boundary of the system. A clear understanding of the problem will help us in building a better system &amp; reduce the risk of project failure. It also specifies the resources that have to be made available to the project. Three important factors project goal, project bounds &amp; the resource limits are sometimes called the project’s term of reference.

3.2 Proposed Solution

Depending on the system components used, activity recognition can be divided into two categories: sensor-based and vision-based activity recognition. Using a network of sensors, sensor-based activity recognition keeps track of both the actions of a person and outside factors. Important information can be extracted from the data collected by these devices after it has been processed and aggregated. Additionally, the model is frequently trained using methods from machine learning, deep learning, and data analytics.

In contrast, vision-based activity recognition uses a camera-based system to examine and process video data in order to find and classify behaviours within a specific environment. The video, which is basically a collection of images, is often processed using digital image processing techniques to extract pertinent information. For this particular project, a vision-based system was utilized.

3.3 Advantages

Using Python and the OpenCV library, a Raspberry Pi 4 and Raspberry Pi camera have been used to create a house security surveillance system. The system was tried by using the "raspistill -o image.jpg" command to take a picture after configuring the Raspberry Pi 4 with the Raspbian OS and connecting the camera. In order to increase functionality, extra libraries and dependencies were installed after the coding was finished in the terminal. In order to allow email notifications with an attached image when motion is detected, the mail.py file was configured with the user's and system's email addresses. The system constantly watches for movement and emails the user's email address an alert with the image it has just taken. This project was successful in giving users an easy method to spot activity and take the necessary action.

CHAPTER - 4

REQUIREMENT AND SPECIFICATION

4.1Hardware Requirement

Raspberry Pi (2GB RAM or higher)

USB Camera

4.2Software Requirement

Raspberry pi OS

Python IDE

OpenCV library

1. OpenCV library plays a key role in image processing and understanding of

2. The threats are to be categorized as the type of threat they are i.e, Tanks, Soldiers, Grenades, and Guns.

3. If an anomaly is detected the image of that threat will be sent to the user via. Mail.

4. The camera should be connected to a power source.

The captured image of the trespasser is sent to the registered user via mail.

4.3 The camera will be able to identify the threat type

Description: The type of threat will be classified and the message will be sent to the mail id of the user registered.

Implementation: With the help of image processing the camera will be able to identify the threat type.

Criticality: The feature is important because the type of threat is classified and can be marked accordingly.

Risks : The camera might not record if there a multiple anomalies as a result the camera might crash and would require a restart.

Dependency: This depends on the functionality of the raspberry pi i.e the software updates, system configuration etc.

CHAPTER – 5

SYSTEM DEVLOPEMENT

5.1 SYSTEM ARCHITECTURE

Software Side

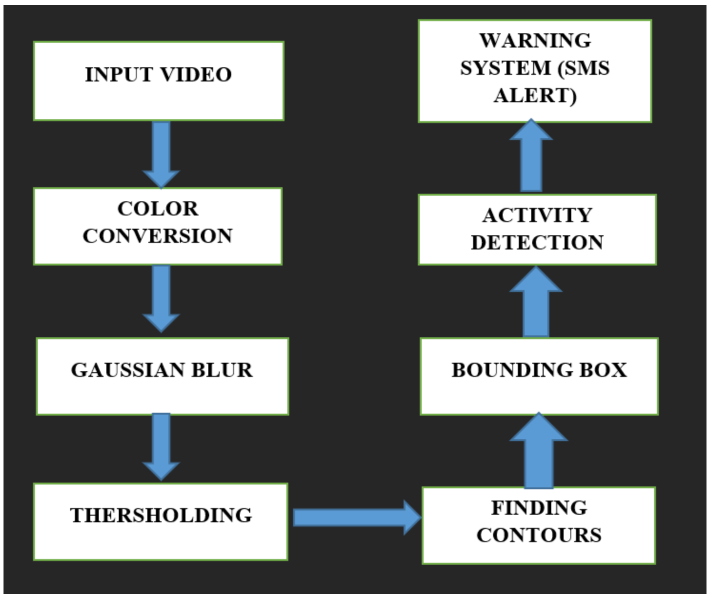

Fig 5.1.1 Software Side Architecture

Hardware Side

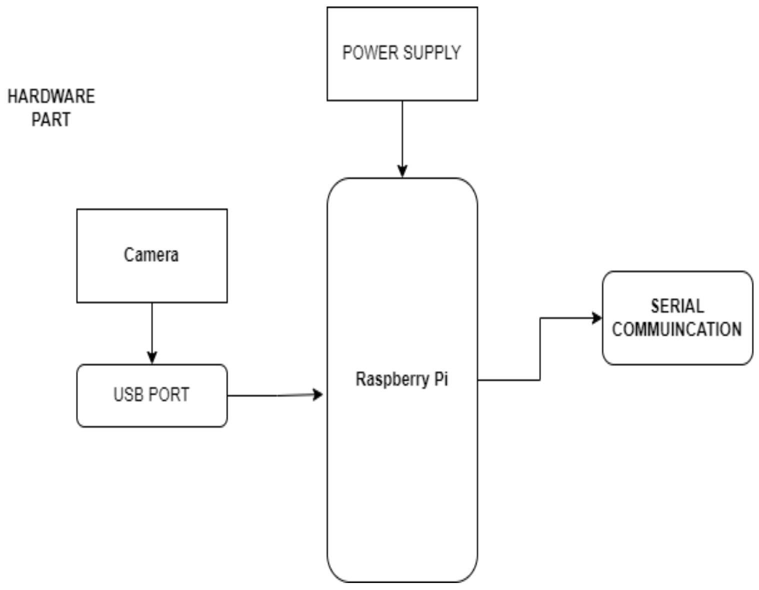

Fig 5.1.2 Hardware Side Architecture

Software Side

Input Video

The captured live footage is run by the systems image process and the image is converted in the image data process system.

Color Conversion

The ARGB color of the image is converted into system color language and is read by the systems software recognition.

Thresholding

Thresholding is an image processing technique that converts a grayscale or color image into a binary image, by selecting a threshold value that separates the foreground (object of interest) from the background. Pixels above the threshold are set to one, while pixels below the threshold are set to zero. The threshold value is set to 7-8 for this project.

Finding Contours

In computer vision, finding contours involves detecting and tracing the boundaries of objects in an image. This is done by applying various edge detection techniques and then using mathematical algorithms to identify the connected edges as contours. Contours can be used for object recognition, segmentation, and shape analysis.

Bounding Box

A bounding box is a rectangular box that is used to enclose or surround a specific object or area of interest within an image or a video. It is often used in computer vision and machine learning applications for object detection, tracking, and segmentation.

Activity Detection

Activity detection refers to the process of identifying and tracking specific movements or actions within a given environment using sensors such as cameras, accelerometers or microphones. This technique is widely used in applications such as security systems, human-computer interaction and sports analysis.

Warning System (Mail Alert)

A warning system or mail alert is a notification system that sends an email or message to users to notify them of an event, issue or potential danger. It can be triggered by different sources, such as sensors, software applications or manual inputs, and can be customized to specific user preferences and conditions.

5.2 UML DIAGRAMS

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by Object Management Group (OMG) and

UML 1.0 specification draft was proposed to the Object Management Group in January 1997.

Object Management Group is continuously putting effort to make a truly industry standard.

- UML stands for Unified Modeling Language
- UML is different from the other common programming languages like C++, Java, COBOL, etc.
- UML is a pictorial language used to make software blue prints.

UML can be described as a general purpose visual modeling language to visualize, specify, constructs and document software system. Although UML is generally used to model software systems but it is not limited within this boundary. It is also used to model non software systems as well like process flow in a manufacturing unit etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization UML is become an Object Management Group standard.

5.3 Usecase diagram

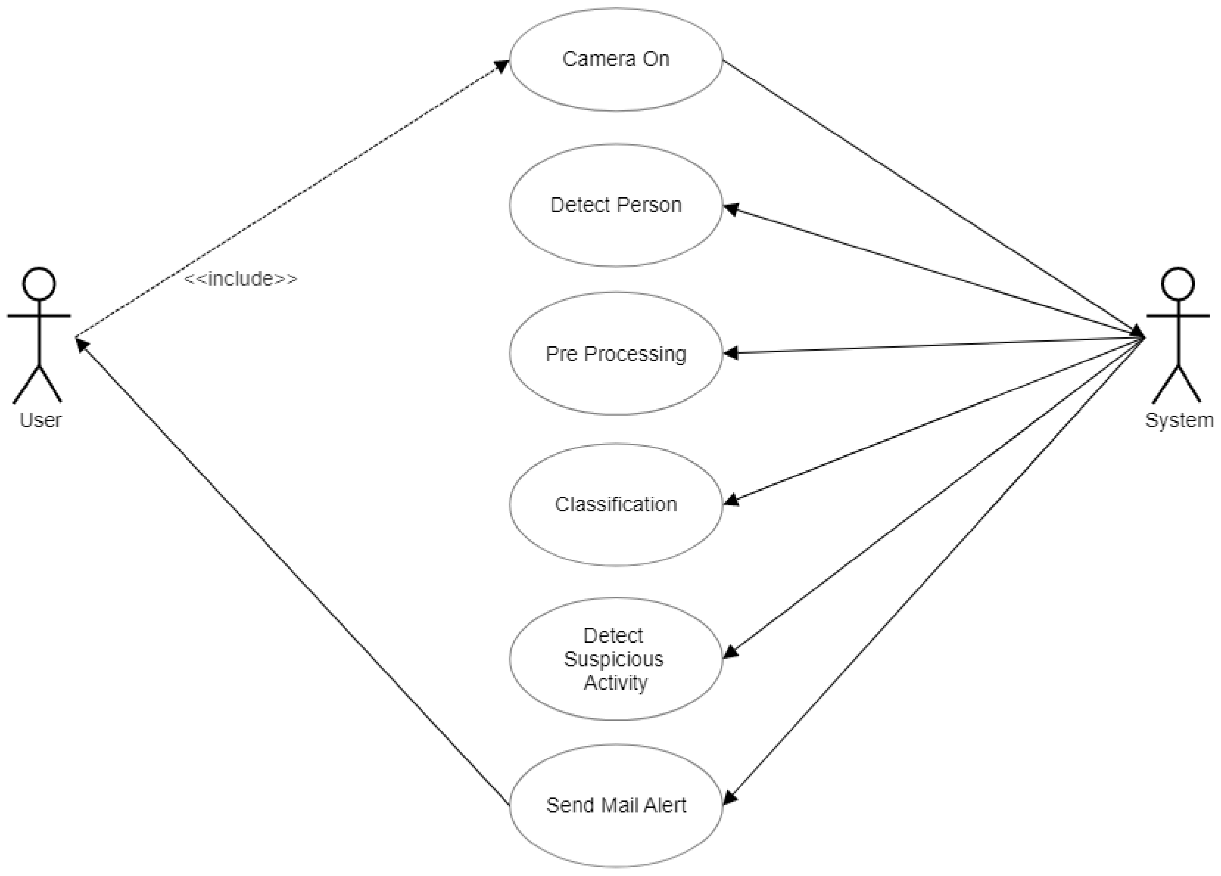

Fig 5.3 Usecase Diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application

5.4 Class diagram

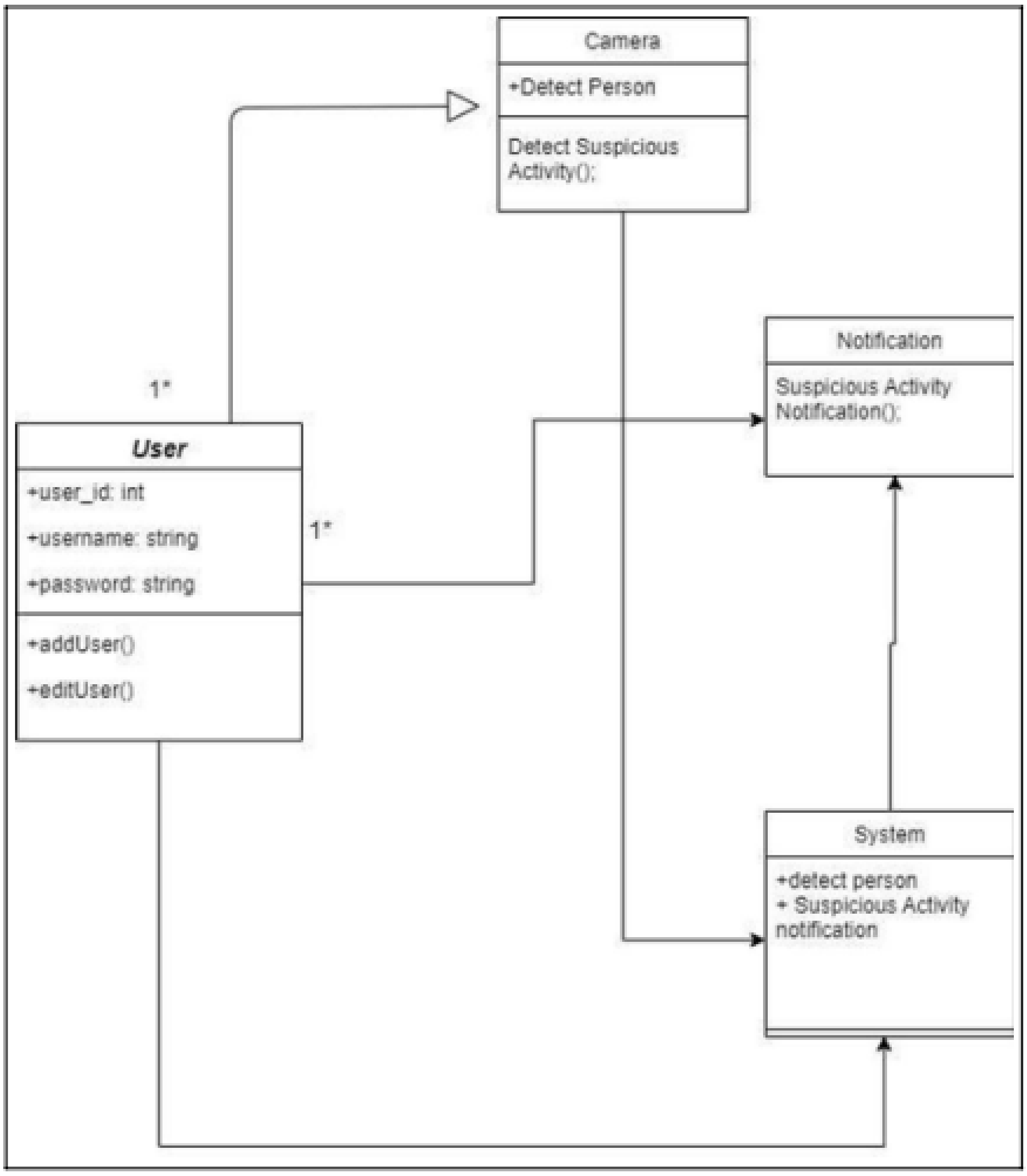

Fig 5.4 Class Diagram

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML).

5.5 E-R Diagram

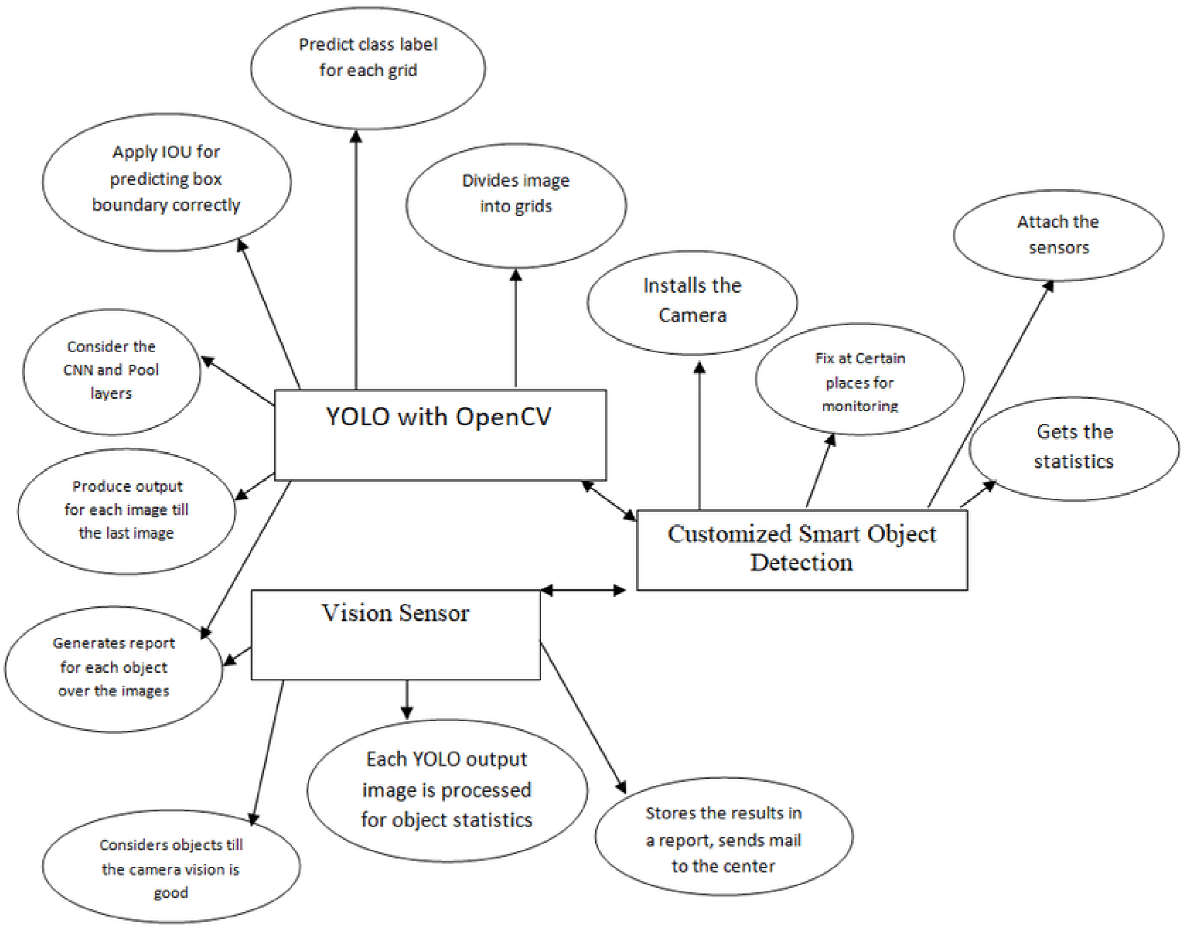
Predict class label for each grid
Apply IOU for predicting box boundary correctly
Divides image into grids
Attach the sensors
Installs the Camera
Consider the CNN and Pool layers
Fix at Certain places for monitoring
YOLO with OpenCV
Gets the statistics
Produce output for each image till the last image
Customized Smart Object Detection
Vision Sensor
Generates report for each object over the images
Each YOLO output image is processed for object statistics
Stores the results in a report, sends mail to the center
Considers objects till the camera vision is good

Fig 5.5 E-R Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system.

5.6 DFD Diagram

Level 0

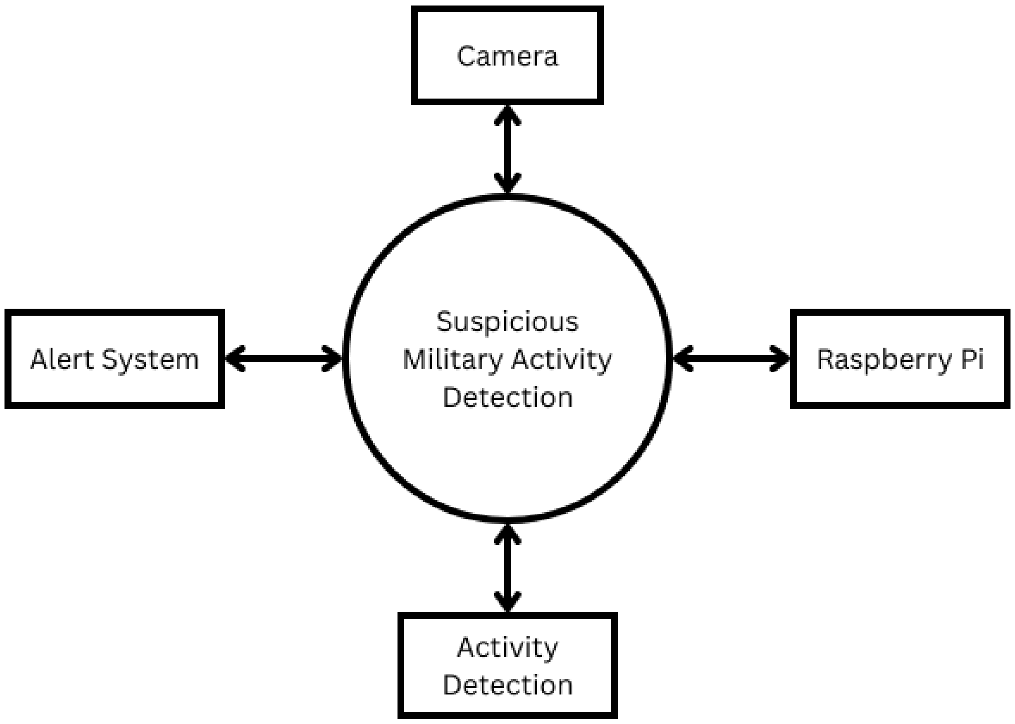

Fig 5.6.1 DFD Diagram Level 0

Level 1

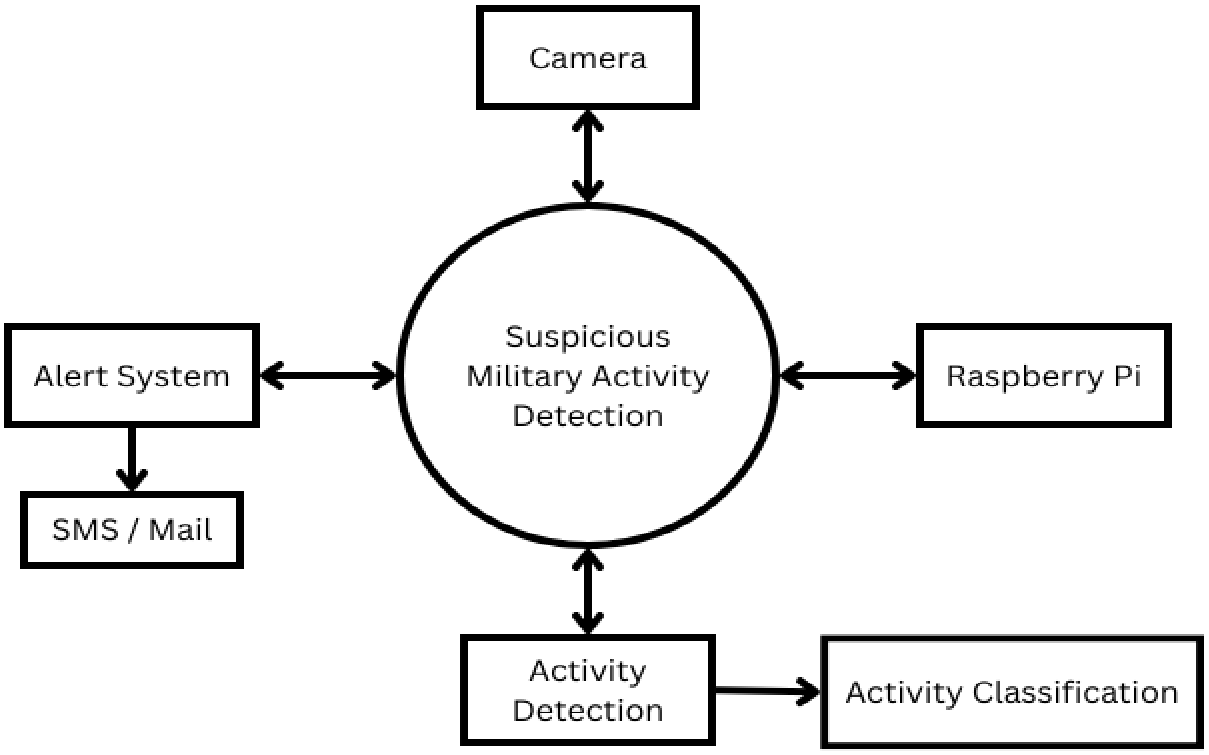

Fig 5.6.2 DFD Diagram Level 1

Level 2

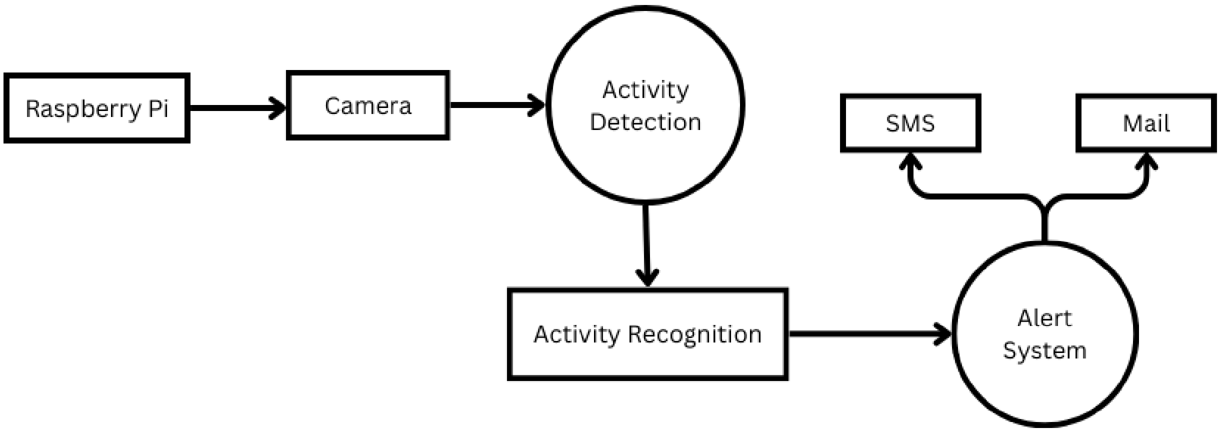

Fig 5.6.3 DFD Diagram Level 2

CHAPTER – 6
MODULES

6.1 HARDWARE MODULE

6.1.1 Pi Board

Raspberry Pi is a small, single-board computer that was developed to promote teaching of basic computer science in schools. However, it has become a popular tool for DIY projects and hobbyists due to its affordability and versatility. The board runs on a Linux-based operating system and has several input/output pins that can be used to connect sensors, motors, and other devices. It can be used for a wide range of projects, such as home automation, media centers, retro gaming consoles, and even robotics. The Raspberry Pi can be programmed in a variety of programming languages, making it accessible for beginners and experienced programmers alike. Overall, the Raspberry Pi board is a powerful and flexible tool that can be used to bring a variety of projects to life.

6.1.2 Dataset Model Training

Dataset model training is the process of teaching a machine learning algorithm to make accurate predictions by feeding it with a large set of labeled data. The process involves splitting the dataset into training and validation sets, selecting an appropriate model architecture, and using an optimization algorithm to adjust the model's parameters to minimize the loss function. The training process iteratively feeds batches of data to the model, which calculates the predicted output and compares it with the actual output to adjust the parameters. This continues until the model achieves a satisfactory level of accuracy on the validation set. The trained model can then be used to make predictions on new, unseen data. Proper dataset preparation, model selection, and hyperparameter tuning are critical in achieving high accuracy in model training.

6.1.3 Image Identification

Image identification refers to the process of recognizing and classifying the content of an image, typically using artificial intelligence and machine learning algorithms. This involves analyzing the visual features of an image, such as color, shape, texture, and patterns, to identify objects, people, animals, and other elements within the image. Image identification has numerous applications, including image search, self-driving cars, security surveillance, medical diagnosis, and augmented reality. The accuracy of image identification algorithms depends on the quality of the training data, the complexity of the image recognition task, and the sophistication of the machine learning model used.

6.1.4 Testing of Dataset

Testing a dataset is an important process in machine learning and data analysis that helps to evaluate the performance of a model or algorithm. The goal of testing is to assess the accuracy, precision, recall, and other relevant metrics of the model's predictions on new and unseen data.

During the testing phase, a subset of the original dataset, often referred to as a test set, is used to evaluate the model's generalization ability. The test set should be representative of the data distribution and must be carefully selected to avoid any bias in the evaluation.

The testing process involves feeding the test set into the model and comparing its predictions to the actual values. This comparison generates a set of evaluation metrics that can be used to assess the model's performance. Based on the results of the testing, adjustments can be made to the model to improve its accuracy and performance. Overall, testing is a crucial step in ensuring the quality and reliability of machine learning models.

6.1.5 Data Transmission

Data transmission refers to the process of sending and receiving data over a communication channel. In this process, data is encoded into a format suitable for transmission and sent through a medium such as a wired or wireless network. The data may be transmitted in different forms such as analog or digital signals depending on the communication technology used. The transmission may be conducted using a variety of protocols that ensure reliable and secure data transfer. This includes error detection and correction, encryption, and other mechanisms to ensure data integrity. At the receiving end, the data is decoded and converted back to its original format to be processed or displayed to the user.

6.2 COMMANDOS MODULE

Data receiving refers to the process of obtaining and collecting information from various sources such as sensors, devices, or networks. The data is typically transmitted through a communication channel and is received by a receiver, which can be a computer, a server, or a storage device. The receiver processes and interprets the data, which can be in various formats such as text, images, audio, or video. The data can be received in real-time or stored for later processing. In modern technology, data receiving plays a crucial role in various applications, including IoT, telecommunications, and big data analytics, enabling the collection of valuable insights and facilitating decision-making processes.

So as a result, the image sent by the camera is viewed by the commandos on the command centre and they will take the necessary measures.

CHAPTER – 7
CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

The application of computer vision technology has enabled the creation of innovative security systems that can detect and respond to motion in real-time. In one example, a model has been developed that is capable of correctly identifying motion and highlighting it with an orange frame. This frame allows the user to quickly and easily identify the area in which motion has been detected.

Additionally, this system utilizes a USB Raspberry Pi camera to capture clear images that make it simple to spot any suspicious activity. These cameras are widely used in various computer vision applications due to their affordability, portability, and high-quality imaging capabilities. With its easy-to-use interface and high-quality image capture, the Raspberry Pi camera is an ideal choice for this type of application.

Furthermore, the system is designed to quickly send an image to the user via Outlook. This feature ensures that the user is immediately notified of any suspicious activity and can take appropriate action as needed. The rapid delivery of information is a key component of any effective security system, as it allows for quick and decisive responses to any potential threats.

Overall, this system demonstrates the potential of computer vision technology in creating highly effective security systems. By utilizing cameras and real-time motion detection algorithms, this model can provide accurate and immediate information to users, enabling them to respond quickly to potential threats. As computer vision technology continues to evolve, it is likely that we will see even more advanced security systems that are capable of providing even greater levels of protection to individuals and organizations. The F1 curve, Confusion Matrix and PR curve are given below,

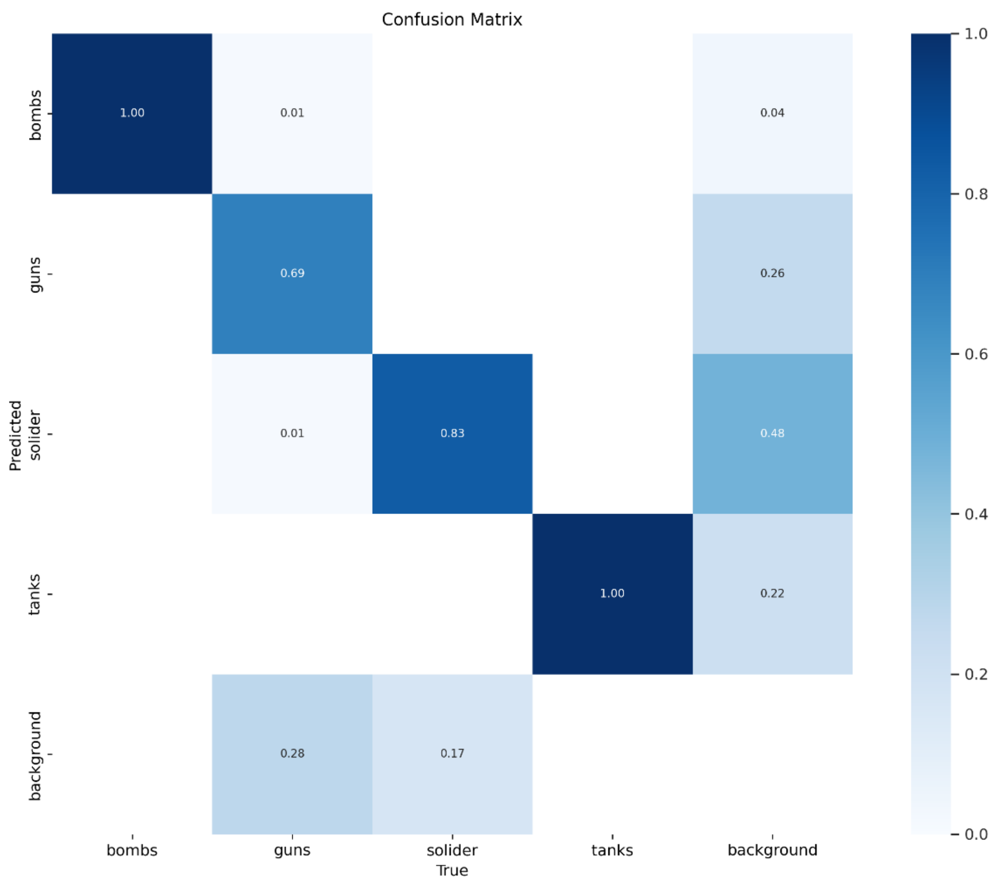

Fig.7.1.1 Confusion Matrix

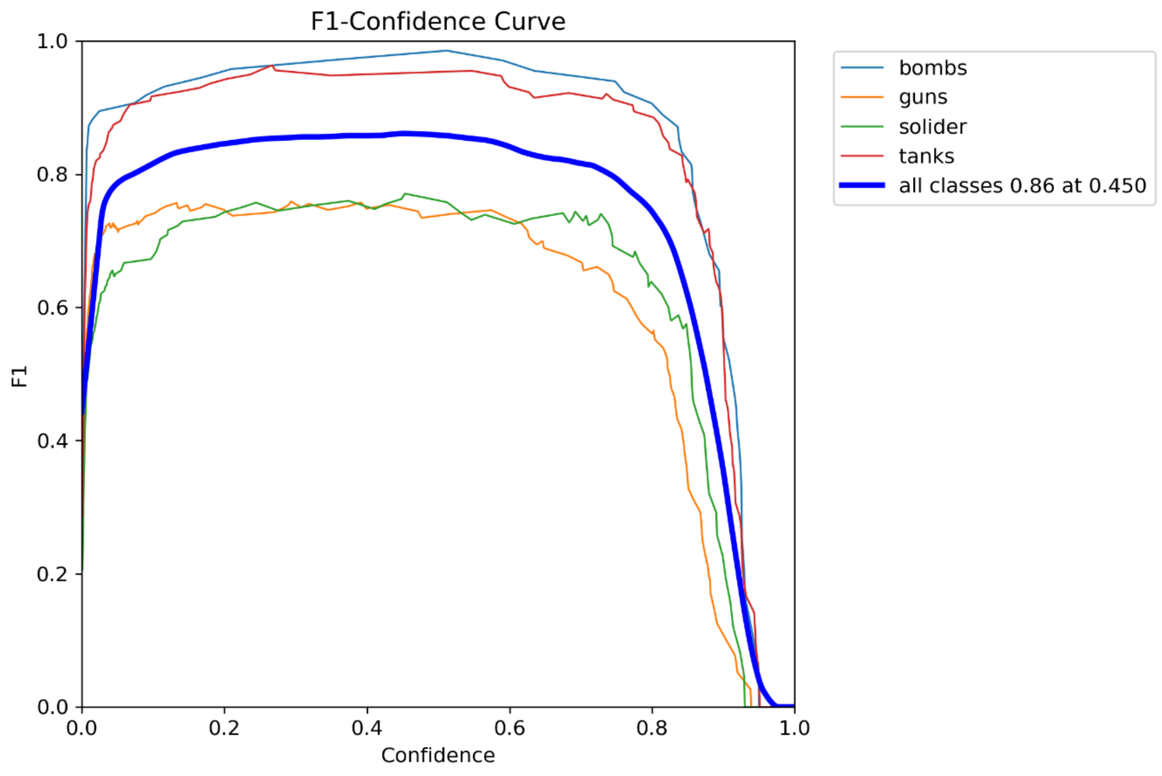

Fig.7.1.2 F1 Curve

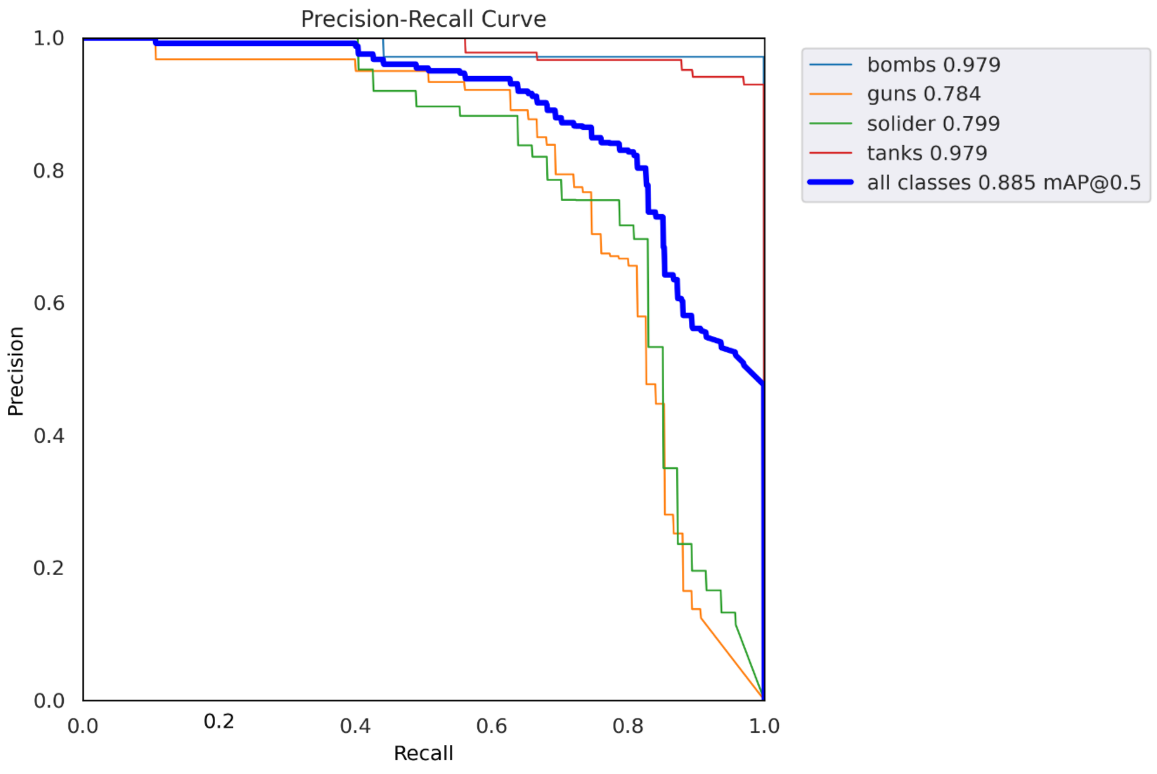

Fig.7.1.3 PR Curve

7.2 FUTURE ENHANCEMENTS

Integration with CCTV cameras: The integration of CCTV cameras, help the local residents or the community to install the cameras and reduce the effort of manpower required i.e watchmen work can be greatly reduced.

Artificial intelligence and machine learning: The use of AI and machine learning could be enhanced to provide even more precise activity recognition and class it accordingly.

CHAPTER – 8

TESTING AND TEST RESULT

TESTING

8.1 Testing types

Performance testing

Performance testing in software engineering generally refers to testing done to ascertain how a system performs in terms of responsiveness and stability under a specific workload. Additionally, it can be used to look into, gauge, confirm, or evaluate other system quality characteristics like scalability, dependability, and resource utilisation.

A subset of performance engineering, which aims to include performance into a system's implementation, design, and architecture, is performance testing. Performance testing is a non-functional software testing technique that determines how the stability, speed, scalability, and responsiveness of an application holds up under a given workload. It’s a key step in ensuring software quality, but unfortunately, is often seen as an afterthought, in isolation, and to begin once functional testing is completed, and in most cases, after the code is ready to release. The goals of performance testing include evaluating application output, processing speed, data transfer velocity, network bandwidth usage, maximum concurrent users, memory utilization, workload efficiency, and command response times.

Reasons for Performance Testing

Organizations run performance testing for at least one of the following reasons:

- To determine whether the application satisfies performance requirements (for instance, the system should handle up to 1,000 concurrent users).
- To locate computing bottlenecks within an application.

- To establish whether the performance levels claimed by a software vendor are indeed true.
- To compare two or more systems and identify the one that performs best.
- To measure stability under peak traffic events.

Load testing

The most basic type of performance testing is load testing. A load test is typically performed to determine how the system will respond to a particular predicted load. This load might be the anticipated number of people using the program concurrently, each doing a set number of transactions in the allotted time. The results of this test will reveal the reaction times of all significant, time-sensitive business transactions. This straightforward test can by itself indicate application software bottlenecks if the database, application server, etc. are also monitored. By the time any software development project nears completion, it likely will have gone through numerous tests, particularly in an Agile testing environment where testing and development happen concurrently. But no matter how many tests you've run, once your application is nearly complete, there's really only one way to know whether or not your software can handle the actual demands your army of end users will soon be placing on it. It's called load testing, and you can use a tool like Load Testing Tool to get the job done. Load testing is the process of putting simulated demand on software, an application or website in a way that tests or demonstrates it's behavior under various conditions.

Stress testing

Stress testing is typically done to determine the system's maximum capacity. This type of test is used to assess the system's robustness under extreme load and aids application administrators in determining if the system will function well if the current load is significantly higher than the anticipated maximum. Server-client Stress Testing: In this stress testing, testing is carried out across all clients from the server.

Product Stress Testing: Product stress testing concentrates on discovering defects related to data locking and blocking, network issues, and performance congestion in a software product.

Transaction Stress Testing: Transaction stress testing is performed on one or more transactions between two or more applications. It is carried out for fine-tuning and optimizing the system.

Systematic Stress Testing: Systematic stress testing is integrated testing that is used to perform tests across multiple systems running on the same server. It is used to discover defects where one application data blocks another application.

Analytical Stress Testing: Analytical stress testing is performed to test the system with abnormal parameters or conditions that are unlikely to happen in a real scenario. It is carried out to find defects in unusual scenarios like a large number of users logged at the same time or a database going offline when it is accessed from a website.

Soak testing

To ascertain whether the system can withstand the continuous predicted load, soak testing, also known as endurance testing, is frequently carried out. Memory use is tracked throughout soak tests to look for any potential leaks. Performance deterioration is very crucial yet sometimes disregarded. That is, to make sure that the throughput and/or reaction times are just as excellent as or better than they were at the start of the test.

In essence, it is placing a heavy strain on a system for an extended period of time. The objective is to learn how the system operates under prolonged use. Soak testing involves testing a system with a typical production load, over a continuous availability period, to validate system behavior under production use.[1]

It may be required to extrapolate the results, if not possible to conduct such an extended test. For example, if the system is required to process 10,000 transactions over 100 hours, it may be possible to complete processing the same

10,000 transactions in a shorter duration (say 50 hours) as representative (and conservative estimate) of the actual production use. A good soak test would also include the ability to simulate peak loads as opposed to just average loads. If manipulating the load over specific periods of time is not possible, alternatively (and conservatively) allow the system to run at peak production loads for the duration of the test.

For example, in software testing, a system may behave exactly as expected when tested for one hour. However, when it is tested for three hours, problems such as memory leaks cause the system to fail or behave unexpectedly.

Soak tests are used primarily to check the reaction of a subject under test under a possible simulated environment for a given duration and for a given threshold. Observations made during the soak test are used to improve the characteristics of the subject under further tests.

In electronics, soak testing may involve testing a system up to or above its maximum ratings for a long period of time. Some companies may soak test a product for a period of many months, while also applying external stresses such as elevated temperatures.

Spike testing

Spike testing involves abruptly boosting the number of users or the load they produce while monitoring the system's response. Finding out whether performance will degrade, the system will malfunction, or it will be able to manage significant changes in load is the aim. Spike Testing is a type of software testing in which a software application is tested with extreme increments and decrements in traffic load. The main purpose of spike testing is to evaluate the behaviour of the software application under sudden increment or decrement in user load and determine recovery time after a spike of user load.Spike Testing is performed to estimate the weaknesses of software applications.

spike Testing Process

Test Environment Setup

Firstly testing environment is set up to perform a successful test. It is set up to get a good quality testing process.

Determine Extreme Load

After setting up the environment, the extreme load is found what a system can resist. Extreme load is the maximum number of users that can use the system or software application at a same time.

Increase Load to Peak

Now the load on the system or software application is increased to the peak point. This process is performed abruptly i.e. suddenly the load is increased not gradually.

Analysis on Peak Point

The behavior of system is observed under the load on the peak point. It is observed that whether the system crashes or survive under this sudden increased load.

Decrease Load to Zero

From the extreme point suddenly the load is decreased to zero or minimum load possible. This process is also performed abruptly i.e. suddenly the load is decreased from peak value to minimum possible value.

Analysis on Minimum Load

The behavior of system is observed under the minimum possible load. It is observed that whether the system crashes or survive under this sudden decreased load.

Performance Graph Analysis

The performance graph of the system formed by abruptly increasing and decreasing the applying load, is analyzed. It is observed that what kind of spike is formed is formed.

Configuration testing

Tests are developed to ascertain the impact of configuration changes to the system's components on the system's performance and behaviour rather than testing for performance from the perspective of load. A frequent illustration would be testing with various load-balancing techniques. Configuration testing is the process of testing a system with each of the supported software and hardware configurations. Configuration Testing is the type of Software Testing which verifies the performance of the system under development against various combinations of software and hardware to find out the best configuration under which the system can work without any flaws or issues while matching its functional requirements. Configuration Testing is the process of testing the system under each configuration of the supported software and hardware. Here, the different configurations of hardware and software means the multiple operating system versions, various browsers, various supported drivers, distinct memory sizes, different hard drive types, various types of CPU etc.

The Execution area supports configuration testing by allowing reuse of the assigned tests. You can create configuration suites with a set of assigned tests, and all execution plans that you add to the configuration suite will also have the set of tests assigned. You can also create configuration suites from existing execution plans and copy and paste or cut and paste execution plans in the Execution tree into a configuration suite. Silk Central enables you to add or remove parameters, keywords, and manual testers to or from the configurations. When you create a configuration suite out of an existing execution plan, all the results of the execution plan are preserved in the configuration suite. If you copy and paste an execution plan into an existing configuration suite, these results are not preserved.

Each execution plan in the configuration suite is displayed in an editable grid. You can view the execution plans or configurations that contain a specific test in the Properties tab of the test. You can also view the execution plans or configurations that are associated with a specific requirement in the Assigned Tests tab of the requirement. To define configurations for automated tests, use the Configurations Suite Configurations page. To define configurations for manual tests use the Configurations page.

Isolation Testing

Isolation testing, which entails repeating a test execution that led to a system issue, is not exclusive to performance testing. used frequently to identify and validate the fault domain.

- It is a time-consuming process and there should be stubs and drivers available to retest each one of them individually
- It is expensive as all the items are to be broken into several pieces to make it atomic in nature.
- It verifies the output of each one of those interfaces/subsystems precisely.

Integration testing

The stage of software testing where separate software modules are merged and tested as a unit is known as integration testing (sometimes known as integration and testing, abbreviated I&T). It takes place between unit testing and validation testing. The goal of integration testing is to produce an integrated system that is ready for system testing by taking as input modules that have undergone unit testing, grouping them into bigger aggregates, applying the tests outlined in an integration test plan to those aggregates. Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules. Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as **integration testing**.

Types of tests to include in system testing

The following examples are different types of testing that should be considered during System testing:

- Graphical user interface testing
- Usability testing
- Software performance testing
- Compatibility testing
- Exception handling
- Load testing
- Volume testing
- Stress testing
- Security testing
- Scalability testing
- Sanity testing
- Smoke testing
- Exploratory testing
- Ad hoc testing
- Regression testing
- Installation testing
- Maintenance testing Recovery testing and failover testing.
- Accessibility testing, including compliance with:
    - Americans with Disabilities Act of 1990

- Section 508 Amendment to the Rehabilitation Act of 1973
- Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C)

This list provides as a broad framework or basis to start with, even though various testing organisations may recommend various tests as part of System testing.

Structure testing

This process involves going through specific execution routes and testing a program's internal logic.Output testing is the process of comparing the outputs of test cases with the predictions made during test case design. Structural testing is a type of software testing which uses the internal design of the software for testing or in other words the software testing which is performed by the team which knows the development phase of the software, is known as structural testing.

Structural testing is basically related to the internal design and implementation of the software i.e. it involves the development team members in the testing team. It basically tests different aspects of the software according to its types. Structural testing is just the opposite of behavioral testing.

• By asking the user what format they prefer, the system under examination evaluates the output it produces or displays.

• In this case, there are two possible output formats: one is a printed format, and the other is an on-screen format.

• The output displayed on the screen is found to be accurate because the format was created during the system design phase taking user demands into account.

• The output is produced as the user's hard copy and meets the requirements.

User Acceptance Testing

• The last stage, before delivery to the customer, is often completed by the customer and involves running the test cases using real data.

• The system under consideration is tested for user acceptance, and changes are made as needed while staying in close contact with the potential system user throughout development.

• To show that the established software system satisfies the requirements provided in the requirement specification, it entails the planning and execution of various types of tests.

User acceptance testing (UAT), also called *application testing* or *end-user testing*, is a phase of software development in which the software is tested in the real world by its intended audience. UAT is often the last phase of the software testing process and is performed before the tested software is released to its intended market. The goal of UAT is to ensure software can handle real-world tasks and perform up to development specifications.

In UAT, users are given the opportunity to interact with the software before its official release to see if any features have been overlooked or if it contains any bugs. UAT can be done in-house with volunteers, by paid test subjects using the software or by making the test version available for download as a free trial. The results from the early testers are forwarded to the developers, who make final changes before releasing the software commercially. UAT is effective for ensuring quality in terms of time and software cost, while also increasing transparency with users. UAT also enables developers to work with real cases and data, and if successful, the process can validate business requirements.

CHAPTER – 9

CODE AND OUTPUT

9.1 CODING

```
    $ python detect.py --weights best.pt --source 0                               # webcam
                                                  img.jpg                    # image
                                                  vid.mp4                     # video
                                                  screen                     # screenshot
                                                  path/                      # directory
                                                  list.txt                   # list of images
                                                  list.streams               # list of streams
                                                  'path/*.jpg'               # glob
                                                  'https://youtu.be/Zgi9g1ksQHc'  # YouTube
                                                  'rtsp://example.com/media.mp4'  # RTSP,
RTMP, HTTP stream

Usage - formats:
    $ python detect.py --weights yolov5s.pt                # PyTorch
                                 yolov5s.torchscript       # TorchScript
                                 yolov5s.onnx              # ONNX Runtime or OpenCV DNN
with --dnn
                                 yolov5s_openvino_model     # OpenVINO
                                 yolov5s.engine            # TensorRT
                                 yolov5s.mlmodel           # CoreML (macOS-only)
                                 yolov5s_saved_model       # TensorFlow SavedModel
                                 yolov5s.pb                # TensorFlow GraphDef
                                 yolov5s.tflite            # TensorFlow Lite
                                 yolov5s_edgetpu.tflite     # TensorFlow Edge TPU
```

```
                    yolov5s_paddle_model       # PaddlePaddle
"""
import requests
from pprint import pprint
from datetime import datetime
import cv2, sys, numpy, os
import urllib
import numpy as np
import time
import os
from subprocess import call
import time
import os
import glob
import smtplib
import base64
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import sys
import smtplib
from email.message import EmailMessage
import imutils
import numpy as np
import subprocess
import sys
import socket
```

```
#https://outlook.live.com/mail/0/options/mail/accounts/popImap
yahoo_user = "dhivyesh12345@outlook.com"
yahoo_pwd = "dhivyesh123"
FROM = 'dhivyesh12345@outlook.com'
TO = ['dhivyesh12345@outlook.com'] #must be a list


import argparse
import os
import platform
import sys
from pathlib import Path

import torch

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS,
LoadImages, LoadScreenshots, LoadStreams
from utils.general import (LOGGER, Profile, check_file, check_img_size,
check_imshow, check_requirements, colorstr, cv2,
                           increment_path, non_max_suppression, print_args,
scale_boxes, strip_optimizer, xyxy2xywh)
from utils.plots import Annotator, colors, save_one_box
from utils.torch_utils import select_device, smart_inference_mode



def mails():
    msg = MIMEMultipart()
    time.sleep(1)
    msg['Subject'] = "SECURITY"

    # BODY with 2 argument

    # body=sys.argv[1]+sys.argv[2]
    body = "SUSPICIOUS ACTIVITY DETECTION SYSTEM"
    msg.attach(MIMEText(body, 'plain'))
    time.sleep(1)

    ###IMAGE
    fp = open("hello.jpg", 'rb')
    time.sleep(1)
    img = MIMEImage(fp.read())
    time.sleep(1)
    fp.close()
    time.sleep(1)
    msg.attach(img)
    time.sleep(1)
    try:
        server = smtplib.SMTP("smtp.office365.com", 587)  # or port 465 doesn't
seem to work!
        print("smtp.outlook")
        server.ehlo()
        print("ehlo")
        server.starttls()
        print("starttls")
        server.login(yahoo_user, yahoo_pwd)
        print("reading mail & password")
        server.sendmail(FROM, TO, msg.as_string())
        print("from")
        server.close()
        print('successfully sent the mail')
    except:
        print("failed to send mail")


@smart_inference_mode()
def run(
        weights=ROOT / 'yolov5s.pt',  # model path or triton URL
        source=ROOT / 'data/images',  # file/dir/URL/glob/screen/0(webcam)
        data=ROOT / 'data/coco128.yaml',  # dataset.yaml path
        imgsz=(640, 640),  # inference size (height, width)
        conf_thres=0.25,  # confidence threshold
        iou_thres=0.45,  # NMS IOU threshold
        max_det=1000,  # maximum detections per image
        device='',  # cuda device, i.e. 0 or 0,1,2,3 or cpu
        view_img=False,  # show results
        save_txt=False,  # save results to *.txt
        save_conf=False,  # save confidences in --save-txt labels
        save_crop=False,  # save cropped prediction boxes
        nosave=False,  # do not save images/videos
        classes=None,  # filter by class: --class 0, or --class 0 2 3
        agnostic_nms=False,  # class-agnostic NMS
        augment=False,  # augmented inference
        visualize=False,  # visualize features
        update=False,  # update all models
        project=ROOT / 'runs/detect',  # save results to project/name
        name='exp',  # save results to project/name
        exist_ok=False,  # existing project/name ok, do not increment
        line_thickness=3,  # bounding box thickness (pixels)
        hide_labels=False,  # hide labels
        hide_conf=False,  # hide confidences
        half=False,  # use FP16 half-precision inference
        dnn=False,  # use OpenCV DNN for ONNX inference
        vid_stride=1,  # video frame-rate stride
):
    source = str(source)
    save_img = not nosave and not source.endswith('.txt')  # save inference
images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://'))
    webcam = source.isnumeric() or source.endswith('.streams') or (is_url and not
is_file)
    screenshot = source.lower().startswith('screen')
    if is_url and is_file:
        source = check_file(source)  # download

    # Directories
    save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)  #
increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True,
exist_ok=True)  # make dir

    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data,
fp16=half)
    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride)  # check image size

    # Dataloader
    bs = 1  # batch_size
    if webcam:
        view_img = check_imshow(warn=True)
        dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt,
vid_stride=vid_stride)
        bs = len(dataset)
    elif screenshot:
        dataset = LoadScreenshots(source, img_size=imgsz, stride=stride, auto=pt)
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt,
vid_stride=vid_stride)
    vid_path, vid_writer = [None] * bs, [None] * bs

    # Run inference
    model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz))  # warmup
    seen, windows, dt = 0, [], (Profile(), Profile(), Profile())
    for path, im, im0s, vid_cap, s in dataset:
        with dt[0]:
            im = torch.from_numpy(im).to(model.device)
            im = im.half() if model.fp16 else im.float()  # uint8 to fp16/32
            im /= 255  # 0 - 255 to 0.0 - 1.0
            if len(im.shape) == 3:
                im = im[None]  # expand for batch dim

        # Inference
        with dt[1]:
            visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if
visualize else False
            pred = model(im, augment=augment, visualize=visualize)

        # NMS
        with dt[2]:
            pred = non_max_suppression(pred, conf_thres, iou_thres, classes,
agnostic_nms, max_det=max_det)
```

```
        # Second-stage classifier (optional)
        # pred = utils.general.apply_classifier(pred, classifier_model, im, im0s)

        # Process predictions
        for i, det in enumerate(pred):  # per image
            seen += 1
            if webcam:  # batch_size >= 1
                p, im0, frame = path[i], im0s[i].copy(), dataset.count
                s += f'{i}: '
                cv2.imwrite('hello.jpg', im0)
                mails()
                time.sleep(5)
            else:
                p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)
                cv2.imwrite('hello.jpg', im0)



            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # im.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image'
else f'_{frame}')  # im.txt
            s += '%gx%g ' % im.shape[2:]  # print string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            imc = im0.copy() if save_crop else im0  # for save_crop
            annotator = Annotator(im0, line_width=line_thickness,
example=str(names))
```

```
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_boxes(im.shape[2:], det[:, :4], im0.shape).round()

                # Print results
                for c in det[:, 5].unique():
                    n = (det[:, 5] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    if save_txt:  # Write to file
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-
1).tolist()  # normalized xywh
                        line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # label
format
                        with open(f'{txt_path}.txt', 'a') as f:
                            f.write(('%g ' * len(line)).rstrip() % line + '\n')

                    if save_img or save_crop or view_img:  # Add bbox to image
                        c = int(cls)  # integer class
                        label = None if hide_labels else (names[c] if hide_conf else
f'{names[c]} {conf:.2f}')
                        annotator.box_label(xyxy, label, color=colors(c, True))
                    if save_crop:
                        save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] /
f'{p.stem}.jpg', BGR=True)
```

```
            # Stream results
            im0 = annotator.result()
            if view_img:
                if platform.system() == 'Linux' and p not in windows:
                    windows.append(p)
                    cv2.namedWindow(str(p), cv2.WINDOW_NORMAL |
cv2.WINDOW_KEEPRATIO)  # allow window resize (Linux)
                    cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
                cv2.imshow(str(p), im0)
                cv2.waitKey(1)  # 1 millisecond

            # Save results (image with detections)
            if save_img:
                if dataset.mode == 'image':
                    cv2.imwrite(save_path, im0)
                else:  # 'video' or 'stream'
                    if vid_path[i] != save_path:  # new video
                        vid_path[i] = save_path
                        if isinstance(vid_writer[i], cv2.VideoWriter):
                            vid_writer[i].release()  # release previous video writer
                        if vid_cap:  # video
                            fps = vid_cap.get(cv2.CAP_PROP_FPS)
                            w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                            h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                        else:  # stream
                            fps, w, h = 30, im0.shape[1], im0.shape[0]
                        save_path = str(Path(save_path).with_suffix('.mp4'))  # force
*.mp4 suffix on results videos
                        vid_writer[i] = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                    vid_writer[i].write(im0)

        # Print time (inference-only)
        LOGGER.info(f"{s}{'' if len(det) else '(no detections), '}{dt[1].dt *
1E3:.1f}ms")

    # Print results
    t = tuple(x.t / seen * 1E3 for x in dt)  # speeds per image
    LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS
per image at shape {(1, 3, *imgsz)}' % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir /
'labels'}" if save_txt else ''
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0])  # update model (to fix
SourceChangeWarning)


def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT /
'yolov5s.pt', help='model path or triton URL')
```

```
    parser.add_argument('--source', type=str, default=ROOT / 'data/images',
help='file/dir/URL/glob/screen/0(webcam)')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml',
help='(optional) dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int,
default=[640], help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25,
help='confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU
threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum
detections per image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3
or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to
*.txt')
    parser.add_argument('--save-conf', action='store_true', help='save
confidences in --save-txt labels')
    parser.add_argument('--save-crop', action='store_true', help='save cropped
prediction boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save
images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --
classes 0, or --classes 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-
agnostic NMS')
```

```
    parser.add_argument('--augment', action='store_true', help='augmented
inference')
    parser.add_argument('--visualize', action='store_true', help='visualize
features')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default=ROOT / 'runs/detect', help='save
results to project/name')
    parser.add_argument('--name', default='exp', help='save results to
project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing
project/name ok, do not increment')
    parser.add_argument('--line-thickness', default=3, type=int, help='bounding
box thickness (pixels)')
    parser.add_argument('--hide-labels', default=False, action='store_true',
help='hide labels')
    parser.add_argument('--hide-conf', default=False, action='store_true',
help='hide confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-
precision inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for
ONNX inference')
    parser.add_argument('--vid-stride', type=int, default=1, help='video frame-
rate stride')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(vars(opt))
    return opt
```

```
def main(opt):
    check_requirements(exclude=('tensorboard', 'thop'))
    run(**vars(opt))


if __name__ == '__main__':
    opt = parse_opt()
    main(opt)
```

9.2 OUTPUT

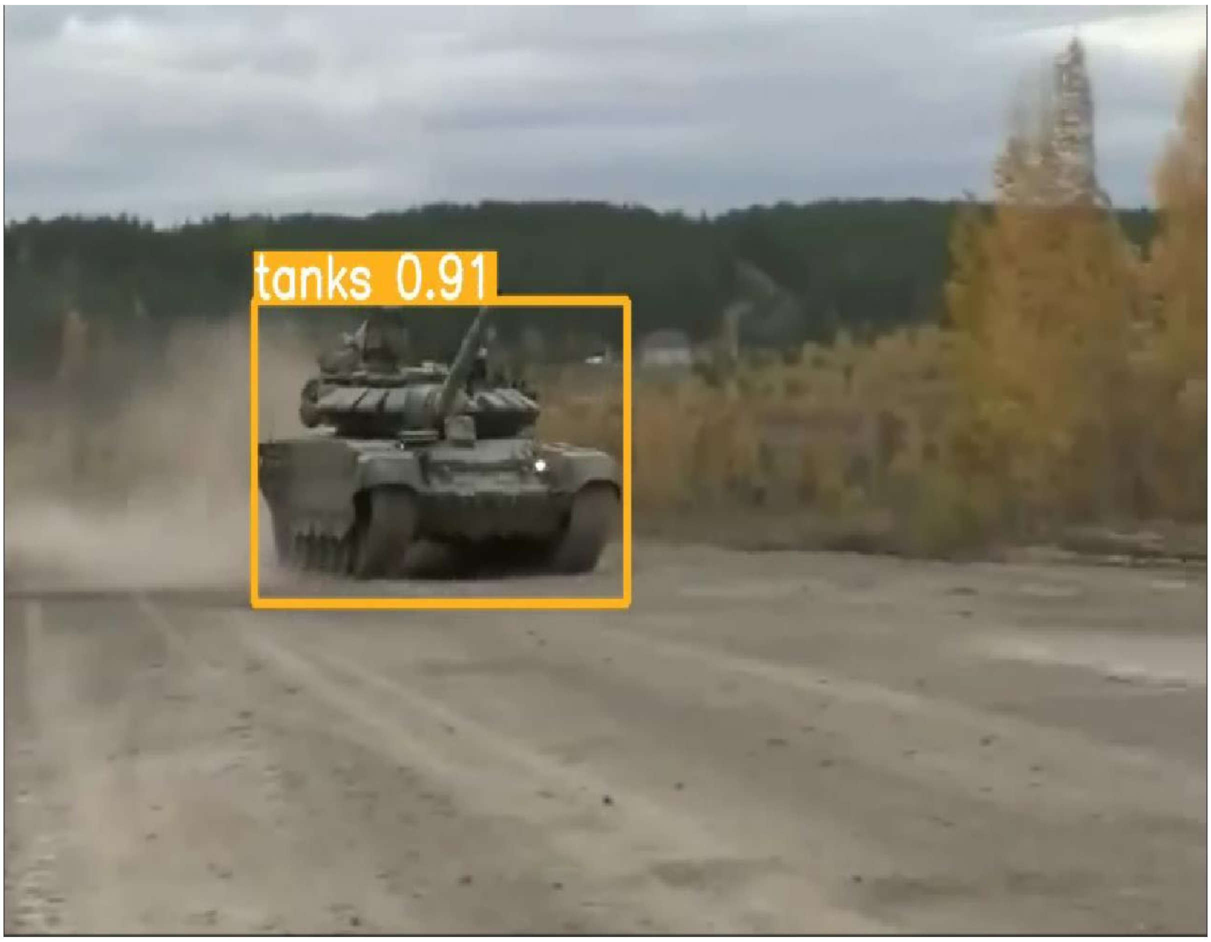

Fig. 9.2.1 Detection of Tanks

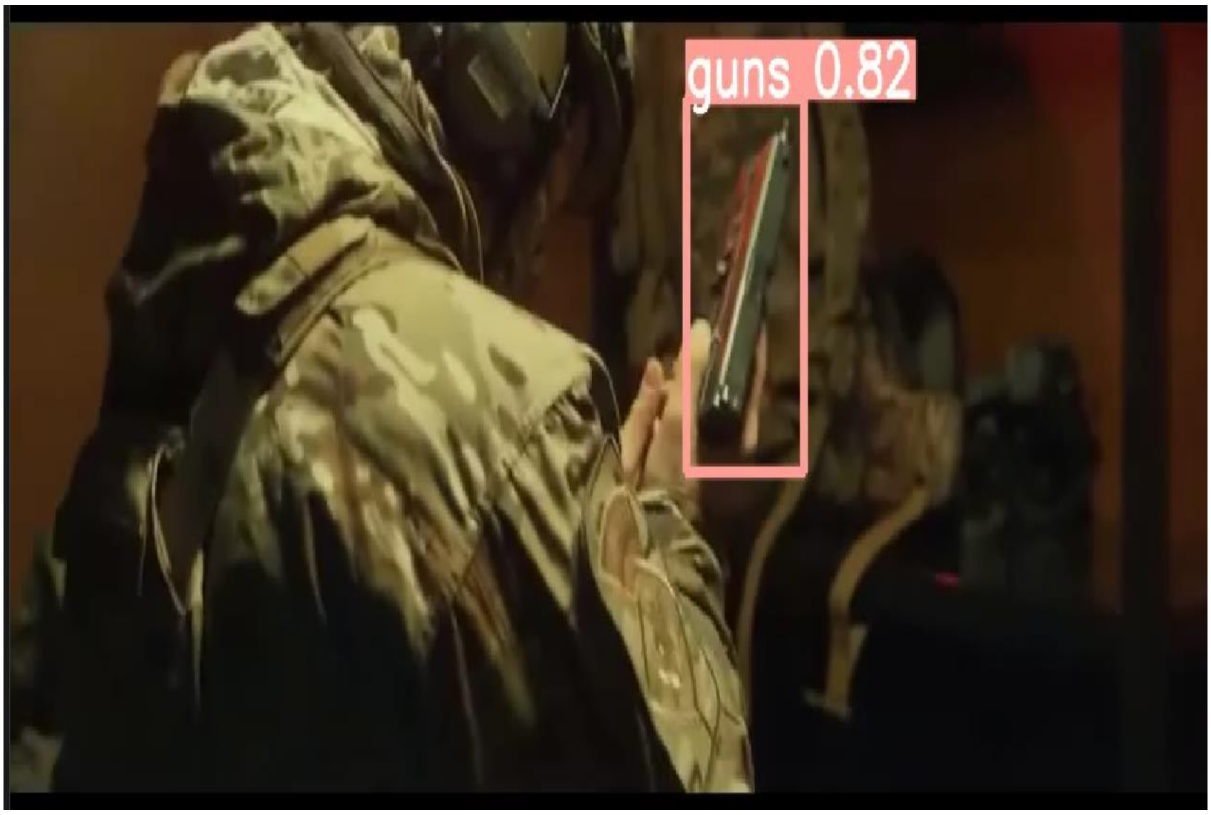

Fig. 9.2.5 Detection of Guns

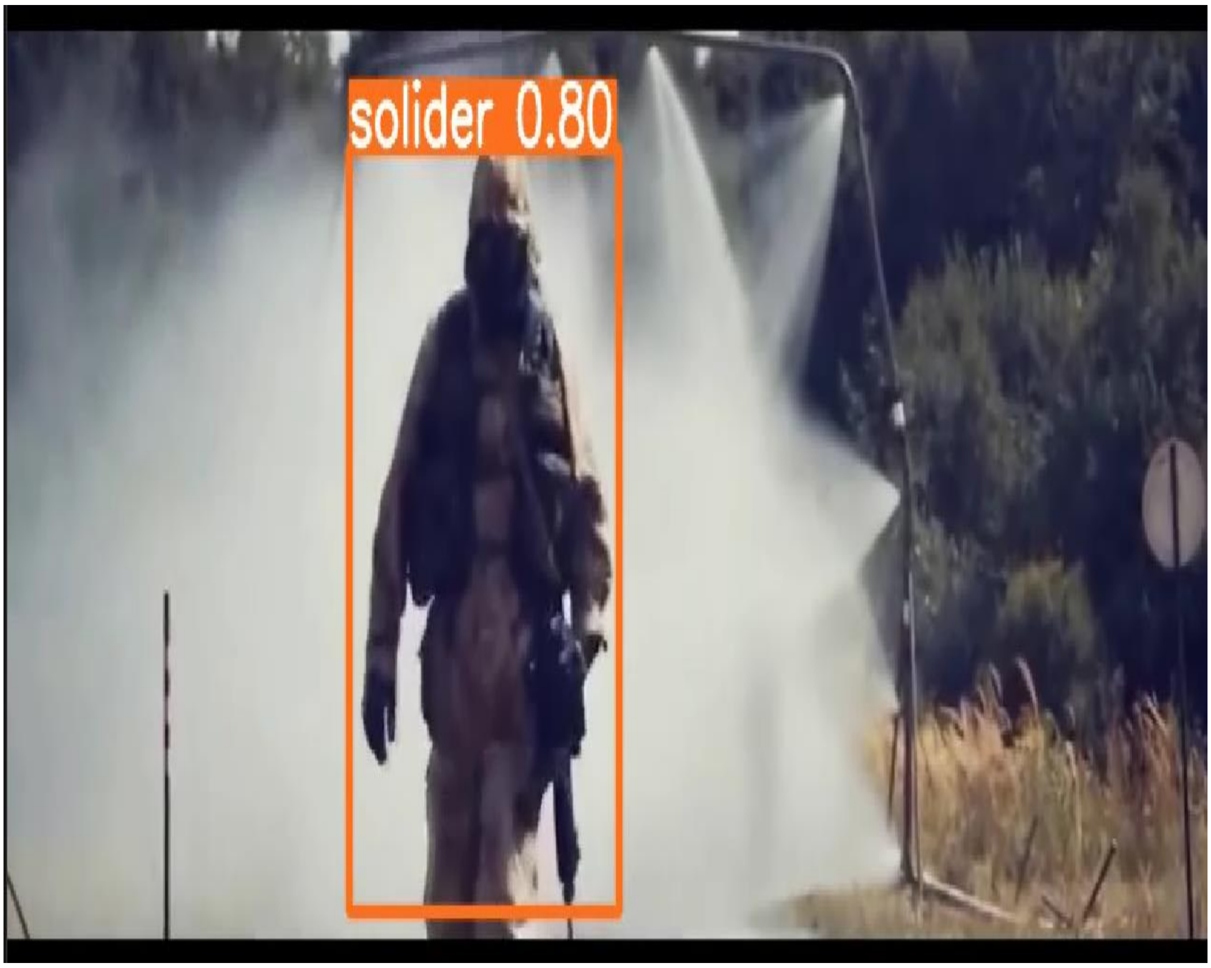

Fig. 9.2.3 Detection of Soldier

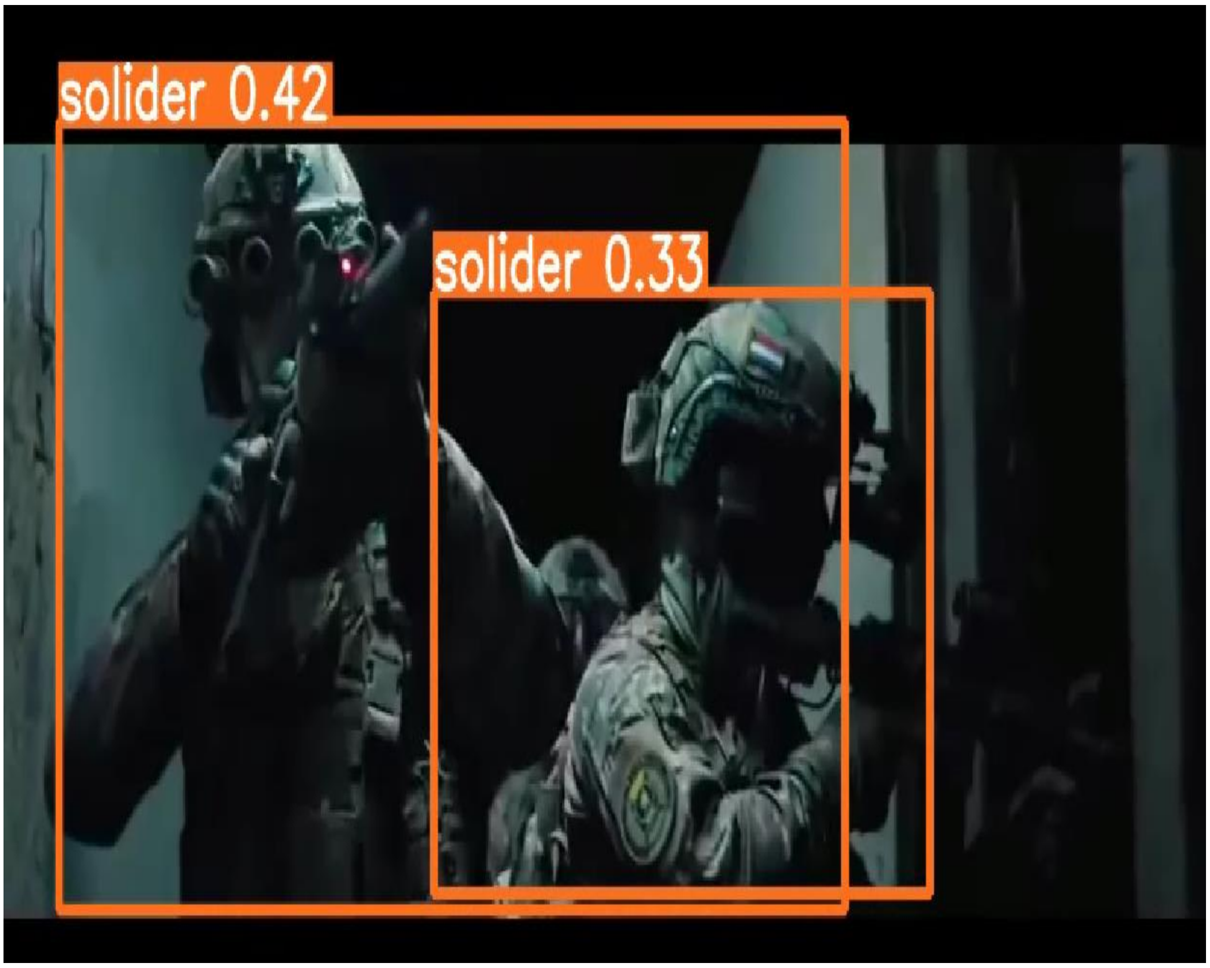

Fig. 9.2.4 Detection of Multiple Objects

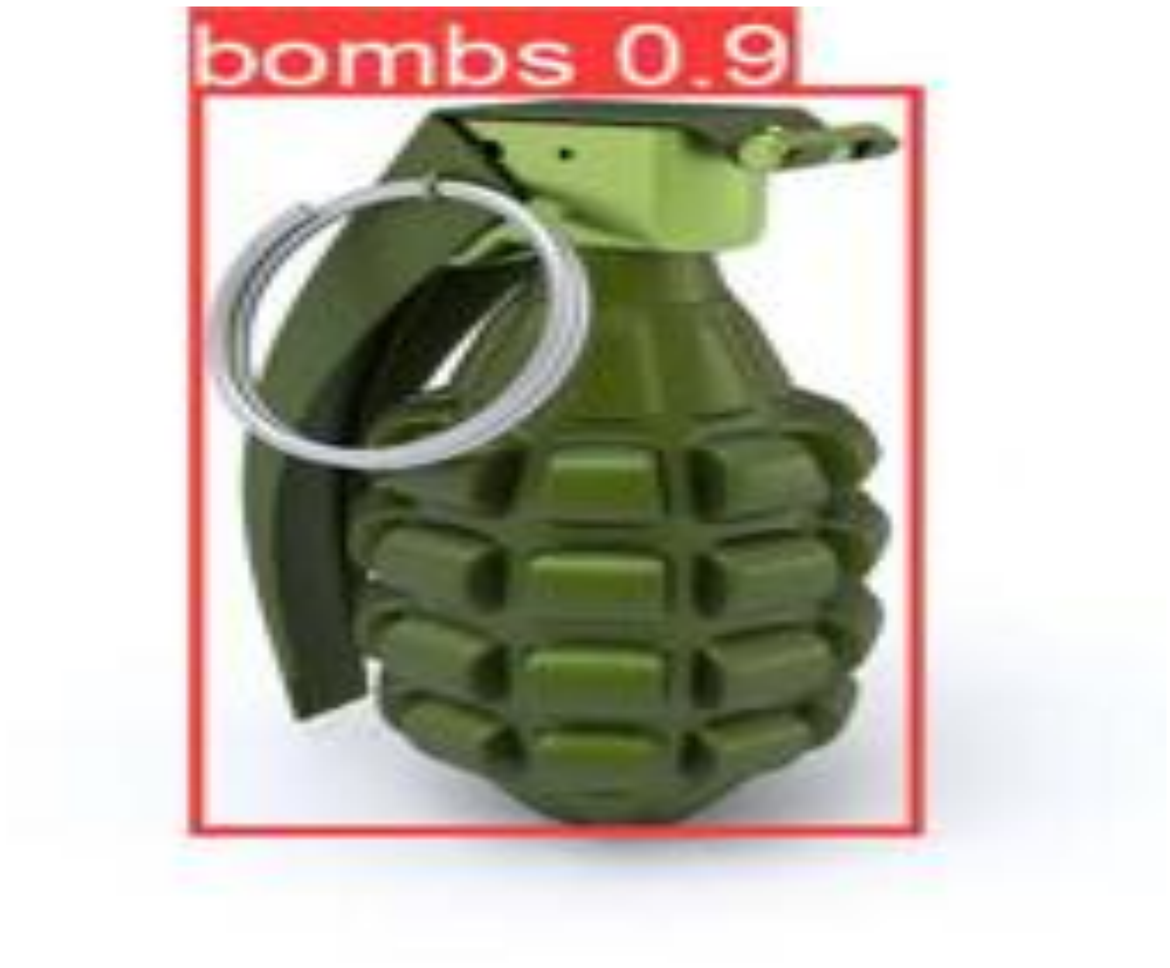

Fig. 9.2.5 Detection of Grenades

CHAPTER – 10

REFERENCE

[1] J. K. Aggarwal and M. S. Ryoo, “Human activity analysis: A review,” ACM Comput. Surv., vol. 43, no. 3, 2011, doi: 10.1145/1922649.1922653.

[2] A. G. D’Sa and B. G. Prasad, “A survey on vision based activity recognition, its applications and challenges,” 2019 2nd Int. Conf. Adv. Comput. Commun. Paradig. ICACCP 2019, pp. 1–8, 2019, doi: 10.1109/ICACCP.2019.8882896.

[3] G. Cheng, Y. Wan, A. N. Saudagar, K. Namuduri, and B. P. Buckles, “Advances in Human Action Recognition: A Survey,” no. February, 2015, [Online]. Available: http://arxiv.org/abs/1501.05964.

[4] C. Dhiman and D. K. Vishwakarma, “A review of state-ofthe-art techniques for abnormal human activity recognition,” Eng. Appl. Artif. Intell., vol. 77, no. August 2018, pp. 21–45, 2019, doi: 10.1016/j.engappai.2018.08.014.

[5] S. A. R. Abu-Bakar, “Advances in human action recognition: An updated survey,” IET Image Process., vol. 13, no. 13, pp. 2381–2394, 2019, doi: 10.1049/ietipr.2019.0350.

[6] T. Huynh-The, B. V. Le, S. Lee, and Y. Yoon, “Interactive activity recognition using pose-based spatio–temporal relation features and four-level Pachinko Allocation Model,” Inf. Sci. (Ny)., vol. 369, pp. 317–333, 2016, doi: 10.1016/j.ins.2016.06.016.

[7] S. Abdelhedi, A. Wali, and A. M. Alimi, “Fuzzy logic based human activity recognition in video surveillance applications,” Adv. Intell. Syst. Comput., vol. 427, pp. 227–235, 2016, doi: 10.1007/978-3-319-29504-6_23.

[8] P. Guo, Z. Miao, Y. Shen, W. Xu, and D. Zhang, “Continuous human action recognition in real time,” Multimed. Tools Appl., vol. 68, no. 3, pp. 827–844, 2014, doi: 10.1007/s11042-012-1084-2.

[9] A. Jalal, M. Uddin, and T. S. Kim, “Depth video-based human activity recognition system using translation and scaling invariant features for life logging

at smart home,” IEEE Trans. Consum. Electron., vol. 58, no. 3, pp. 863– 871, 2012, doi: 10.1109/TCE.2012.6311329.

[10] J. Hu and N. V. Boulgouris, “Fast human activity recognition based on structure and motion,” Pattern Recognit. Lett., vol. 32, no. 14, pp. 1814–1821, 2011, doi: 10.1016/j.patrec.2011.07.013.